Indonesian Journal of Electrical Engineering and Computer Science
Vol. 40, No. 2, November 2025, pp. 1059∼1073
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v40.i2.pp1059-1073

Development and evaluation of a generalized ontology framework for software requirement specification

Sourav Kundu[1], Soumay Kanti Das[1], Abu Rafe Md Jamil[1], Md Kamrul Islam[1], SK. Shalauddin Kabir[1], Mostafijur Rahman Akhond[1,2]
[1]Department of Computer Science and Engineering, Jashore University of Science and Technology, Khulna, Bangladesh
[2]Department of Electrical Engineering and Computer Science, York University, Ontario, Canada

Article Info

Article history:

Received Nov 21, 2024
Revised Aug 3, 2025
Accepted Oct 15, 2025

Keywords:

Knowledge base
Ontology
Protege
Software requirement specification
SPARQL

ABSTRACT

This paper presents an ontology developed to address challenges such as communication gaps, risks of errors, and inconsistencies during the manual process of creating software requirement specifications (SRS). The proposed ontology offers a systematic and formal depiction of the requirements, enhancing consistency and communication among stakeholders. The ontology has been developed from the software requirements documents to facilitate the development process. This paper discusses the process of creating the ontology and demonstrates using Pellet Reasoner for inference and Protégé for ontology construction to save and reuse information. The ontology seems to be efficient in managing complex software projects, enabling accurate requirement retrieval through SPARQL queries. This study emphasizes how incorporating ontologies into requirement engineering can significantly enhance the quality and reliability of SRS.

Corresponding Author:

Mostafijur Rahman Akhond
Department of Computer Science and Engineering, Jashore University of Science and Technology
Jashore, Khulna, Bangladesh
Email: mr.akhond@just.edu.bd

1. INTRODUCTION

In recent years, semantic technologies have advanced rapidly. The adoption of semantic data models like ontologies and knowledge graphs has increased substantially. Ontologies are considered as the foundation of the semantic web [1]. An ontology is a formal and explicit description of concepts within a specific area of discourse (classes/concepts), attributes of each concept that characterize distinct qualities (slots/roles/properties), and constraints on these attributes (facets/role limitations) [2]. A knowledge base consists of an ontology and a collection of distinct instances of classes. As the proverb goes, a knowledge base begins where the ontology ends [2]. Requirement engineering (RE) describes the process of gathering, analyzing, and validating the features and constraints of a software system. The final output of the RE is a written document known as the software requirement specification (SRS) [3]. The software development team, including software engineers, will follow the SRS document when developing the intended software. One of the challenges in software engineering is developing and managing the SRS report. The reports must be clear to reflect the specifications [4]. However, manually developing requirements specifications can lead to inconsistencies, ambiguity, and errors [5]. Lack of domain knowledge among non-technical customers complicates accurately communicating the requirements with experienced analysts. Failing to identify and address the ambiguities on time can have severe consequences [6]. To overcome the knowledge gap and the ambiguity of the SRS document, we have proposed this work representing a generalized knowledge database framework. One way to describe it is as a machine-processable "Web of Data" [7]. We have used the core concept of knowledge databases known as ontology' [8]. Several studies use ontologies in many aspects of requirements engineering issues, such as the elicitation process, handling inconsistencies, dealing with incomplete requirements, reducing ambiguities, and reusing requirements [9] but most of the ontologies are 'domain-specific' ontologies. Several ontologies have been developed for this because there is no accurate methodology for constructing an ontology. Developing an ontology for the SRS report can facilitate software developers, testers, and other stakeholders. This paper proposed a generalized framework that enables the sharing and reusing of knowledge from the existing SRS reports and is also usable for different types of domains. In this thesis, we construct an ontology that enables the sharing and storing the knowledge related to the SRS for all types of software. We follow the IEEE 830-1998 format of SRS and the software engineering body of knowledge (SWEBOK) provided by IEEE. The proposed framework can guide the users to follow the standard IEEE format of the SRS while developing their SRS report, and it can help in sharing the data of the SRS with the community to reuse in the future and reduce the communication gap. Table 1 summarizes the comparison of significant related works.

Table 1. Comparison of previous ontologies and the proposed one

Reference	Domain-specific	Reduces-ambiguity	Generalized	Evaluation method
Haridy *et al.* [10]	Yes	Yes	No	Manual and application
Bencharqui *et al.* [5]	Yes	Partial	No	Manual
Ahmed *et al.* [6]	Yes	Yes	No	Application
Ahmed and Ahmed [11]	Yes	Yes	No	Expert and application
Tan *et al.* [12]	Yes	Partial	No	Expert and application
Proposed ontology	No	Yes	Yes	Manual, expert, and application

Creating a document that can be methodically reviewed, assessed, and approved is what is usually meant by "software requirements specification" in software engineering. The significant contributions of the work are listed as:
- Designing an ontology that can standardly represent the SRS document using the IEEE 830-1998 format.
- Developing the ontology to represent the knowledge of the SRS in a generalized way and reducing the gap of knowledge sharing among the stakeholders.
- Developing the framework that can change the unstructured requirements to a structured class-based model so that the ambiguities of the specifications can be identified clearly.

Our methods include the following:
- An ontology that we have constructed enables formal definitions of concepts and connections within the system.
- We have followed the standard format of the SRS document according to IEEE 830-1998 format [13].
- We have then populated the SRS document for two real-time systems.
- Then, we have tested the results using Reasoner and SPARQL queries.

The rest of the paper is structured as follows: section 2 provides the earlier work in order to obtain a summary of previously proposed work and to determine how to come up with new solutions. Section 3 describes our model's design and the recommended methods we have used thus far. Section 4 outlines the requirements and procedures for evaluation and section 5 provides an overview of the potential for future research as well as the conclusion section, which provides an outline of our thesis.

2. LITERATURE REVIEW

Software engineering is the application of fundamental engineering principles to develop cost effective and reliable software that functions efficiently on actual devices [14]. Sommerville added that software engineering is an engineering work involved with all aspects of software development, from the early stages of system specification to system maintenance after the release for use [15]. Requirement specification is a crucial phase of RE where functional requirements and non-functional requirements are detected, which affects the development of software. Generally, Software requirements depend on the stakeholders; in software development, the requirements are divided into two main parts: i) functional requirement and ii) non-functional requirement [16]. Requirement engineering is crucial to all software development life cycles (SDLC), but because development processes differ, it is frequently neglected or undervalued [11]. A frequently used reference guide for the SRS report writing is IEEE Std. 830-1998 [13]. In recent years ontologies are being used successfully in the RE phase. Ontologies may reduce the ambiguity, inconsistency, and incompleteness of those requirements. An ontology refers to a structured and precise representation of concepts, slots and facets within a specific do main of discussion [2].

An ontology framework is presented in the study by Wongthongtham *et al.* [15] to resolve communication issues in multi-site software development. They emphasize how important it is for managers, stakeholders, and developers to communicate effectively to reduce errors in software products, especially when the requirement engineering phase is underway. Through the use of diagrams, the suggested framework facilitates the sharing of domain and instance knowledge by utilizing ontology as a communication tool. The writers hope to reduce language barriers in the sector by offering an editable flowchart diagram and a case-building framework.

To improve software projects and address common issues like ambiguous and insufficient requirements Yue [17] suggest a two-step procedure for consistency checking in ontology-based requirements elicitation methodologies. This approach combines rule-driven completeness tests with ontology consistency checks. In 2021, Khair and Meziane [18] published a review study on the application of ontologies to the elicitation of software requirements. The study is considered one of the best in this field. According to the analysis, 83.6% of the research supported specifications. According to the survey, 52.24% of respondents support functional requirements, 2.99% support non-functional requirements, and 44.78% support both when it comes to the usage of ontologies to express requirements.

For writing software requirements specifications that use requirements engineering data elements suggested by the SWEBOK, Elliott and Allen [19] developed an approach with automated support. It consists of seven use cases, an ontological framework, and three empirical retrospective case studies that demonstrate the utilization of the methodology. The case studies also demonstrated the ease with which the ontology may be useable for other application domains. They conclude that ontological support can enhance procedures that lead to the specification of software requirements and the subsequent creation of ontologies.

An approach for evaluating how well an existing ontology meets the needs of knowledge stakeholders in requirement elicitation was created by Ermolayev [20]. The comprehensiveness and precision of the interpretation are guaranteed by this methodology, and the evaluation of these requirements for ontology is crucial for ontology engineering. The basis of the strategy employed in the published research is the ontology refinement methodology and the OntoElect ontology process, which consists of three stages: feature extraction, requirements conceptualization, and ontology evaluation.

In a research study on the automatic identification and updating of software requirement specifications using ontology-driven software development, Bhatia *et al.* [21] described how the ontology will automatically find the requirements that have been added, removed, or changed to the current software requirements specification on a particular case study. For evaluating the ontology, they used the Result management system of a specific university.

Jones *et al.* [22] describe that there are four important methodologies for ontological engineering named: TOVE (Toronto virtual enterprise), enterprise model approach, METHONTOLOGY, KBSI IDEF5. Jones *et al.* [22] introduced some other non-comprehensive methodologies, named: Ontolingua, CommonKADS and KACTUS, PLINIUS, ONIONS, Mikrokosmos, MENELAS, PHYSSYS, and SENSUS.

Raad and Cruz [23] carried out a survey on ontology assessment techniques. This paper presents the current ontology evaluation techniques and discusses their benefits and limitations in order to address the problem of finding an effective ontology evaluation method. The methods for evaluating ontologies that are offered can be divided into four groups: approaches based on criteria, tasks, corpuses, and gold standards.

Bhatia *et al.* [7] conducted a study on ontologies for software engineering: past, present, and future. The study talks about different kinds of ontologies which are used in a software life-cycle. The study categorized the ontologies against each step of the software life-cycle. Moreover, it shows the application scope of the ontologies as well as it shows the application scope of ontologies in the software engineering area. The study shows in which software development phase which ontology have been used as well as in future in which phases will be able to use ontology in software engineering.

Bencharqui *et al.* [5] created an ontology-based procedure for requirement specification, in which system domain knowledge is described using a requirement description ontology, common domain knowledge is gathered using ontoReq, and requirements are controlled and improved using ReqDL, which they developed in earlier research. In this work they used Methontology and their work is domain specific and they used competency questions for ontology evaluation and for implementation, they used Portege as a tool. Overall the research enhances the requirement's specification process.

Tan et al. [12] described the development and evaluation process of a software requirement ontology. In this thesis, they designed an ontology for avionics software development that is strongly domain-specific. For evaluation of ontology, they talked about two methods: evaluation by user and application-based evaluation and they prefer evaluation by user method as it's cost-friendly and user-friendly.

These works, however, can be considered independent ontologies because they work with problem-specific to a software project or product. As a result, we require a method that can handle a maximum number of software projects while reducing ambiguities in the RE process and, ultimately, saving the cost of future software development and maintenance.

3. METHOD

This chapter covers the proposed ontology's design process, required tools, and the development process. For ontology-based RE, we generally need four major actors. They are: requirement engineers, stakeholders, an ontology system, and a reasoner system. Requirements are obtained through elicitation, analysis, specification, verification, validation, and creation of a software requirements document [24].

3.1. Research design

The design process of the suggested ontology, which includes class and subclass identification, object property identification, and data property identification, will be covered in this section. Figure 1 depicts the proposed methodology. The following subsections provide a detail explanation of each steps. Though, there is no accurate process or model for developing an ontology, we will follow the steps of Figure 1 for our proposed ontology.

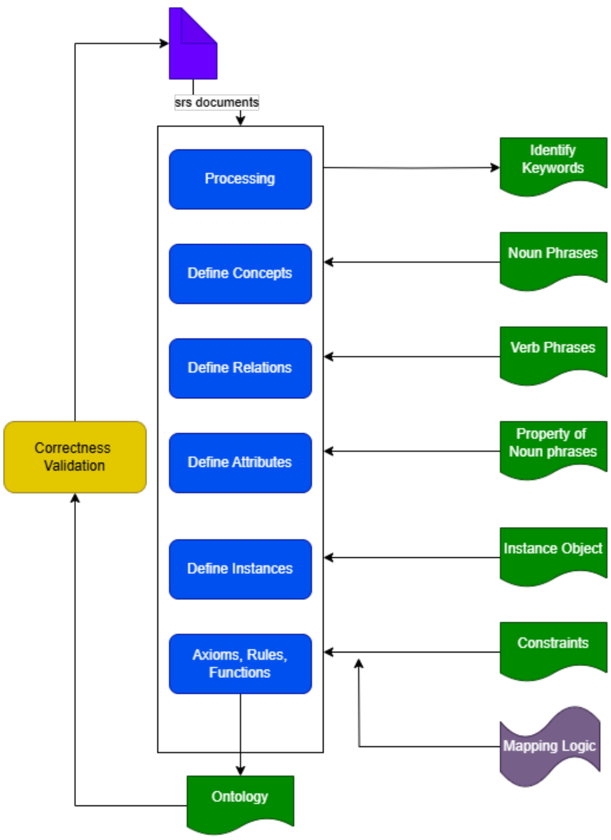

Figure 1. Proposed methodology

3.1.1. Identification of classes and subclasses/concepts

We have followed the top-down development process to build the ontology, which starts with the most general concepts (classes) and subsequent specializations of the concepts (subclasses). The list of required classes and subclasses is shown in Table 2. We have gathered the classes and subclasses list for building SRS ontology from IEEE std. 830-1998 format [13], SWEBOK [16] and the book entitled "Software engineering: a practitionar's approach" [25] by Pressma and Maxim.

– SRS document: will be the main class (concept) for our ontology. All sections will be sub_classes of this class.
– Requirement: individual requirements and their descriptions are captured in a requirement.
– Requirement_Artefact: extra requirements information that may be connected to a requirement is captured by a requirement_Artefact.
– Requirement_Elicitation_Technique: is a process of eliciting a requirement.
– Stakeholder: is an individual, group, or entity that has an interest or concern in a particular process, project, organization, or system. We have got the subclasses from the book of Pressma and Maxim [25].
– External interfaces: capture different kinds of interfaces that are required for the software system. System requirement is the equivalent class of it.
– System other non-functional requirement: captures all the non-functional requirements that are needed for the software system.
– System feature: refers to a specific capability or functionality of a system or software components working together to achieve a particular goal.
– Document's section: is an important class that holds the main section's of a SRS document.
– Appendix: is a supplementary section at the end of a document or book that provides additional information, details, or supporting documentation.

Table 2. Classes and subclasses list

Classes	Sub-classes
SRS_Document	Appendix, Document_s_Section, External_Interface, Requirement, Requirement_Artefact, Requirement_Elicitation_Technique, Stakeholder, System_Feature, System_Requirement, Systems_Other_Non-Functional_Requirement
Appendix	-
External_Interface	Communication_Interface, Hardware_Interface, Software_Interface, User_Interface
Requirement	Functional_Requirement, Non-Functional_Requirement, Other_Requirements
Requirement_Artefact	Goal, Limitation, Obstacle, Source, Story, TestCase
Requirement_Elicitation_Technique	-
Stakeholder	Business_Operation_Managers, Consultants, Customers, End_Users, Marketing_People, Other_Stakeholders, Product_Engineers, Product_Managers, Software_Engineers, Support_and_Maintenance_Engineers
System_Feature	-
System_Requirement	-
System's_Other_Non-Functional_Requirement	BusinessRule, PerformanceRequirement, SafetyRequirement, SecurityRequirement, SoftwareQualityAttribute
Story	Scenario, UseCase
Customers	ExternalCustomers, InternalCustomers

3.1.2. Identification of object properties

When the subject and object are entities (i.e., individuals), a binary predicate known as an object property is employed to express facts in the subject-predicate-object form. The classes in the requirements ontology are interrelated using the object properties listed in Table 3. The ability to relate requirements knowledge to the object properties makes it easier to derive suggestions for solutions, such as alternative requirements. The domains and ranges of the object properties in the requirements ontology are listed in Table 3.

3.1.3. Identification of data properties

Data properties generally refer to the characteristics or attributes of data that describe its various aspects. These properties help to define and understand the data, making it possible to organize, analyze, and utilize it effectively. The specifications as indicated in Table 4, ontology currently offers two data properties to specify in addition to the requirements' validation status.

Table 3. Object properties list

Domain	Object property	Range
Requirement	belongsToFeature	System_Feature
Appendix	belongsToSection	Document_s_Section
System's_Other_Non-Functional_Requirement	belongsToSection	Document_s_Section
Story	describesRequirement	Requirement
SRS_Document	hasExternalInterface	System_Requirement
SRS_Document	hasFeature	System_Feature
Requirement	hasGoal	Goal
Requirement	hasObstacle	Limitation
Requirement	hasObstacle	Obstacle
SRS_Document	hasRequirement	Requirement
Requirement	hasScenario	Scenario
Requirement	hasSource	Source
Requirement	hasUseCase	UseCase
Requirement	isAlternativeOf	Requirement
System_Feature	isConflictWith	System_Feature
Requirement	isElicitedByTechniqueOf	Requirement_Elicitation_Technique
Requirement	isPositiveContributionToGoal	Goal
System_Feature	refinesTo	Requirement
External_Interface	belongsToSection	Document_s_Section
System_Feature	belongsToSection	Document_s_Section
Other_Requirements	belongsToSection	Document_s_Section
SRS_Document	hasAppendix	Appendix
SRS_Document	hasExternalInterface	External_Interface
System_Feature	hasFunctionalRequirement	Functional_Requirement
System_Feature	hasNonFunctionalRequirement	Non-Functional_Requirement
Requirement	hasObstacle	Limitation
System_Feature	hasRequirement	Requirement
Story	hasScenario	Scenario
SRS_Document	hasSection	Document_s_Section
Requirement	hasTestCase	TestCase
Requirement	isAuthoredBy	Stakeholder
Requirement	isConflictWith	Requirement
Requirement	isNegativeContributionToGoal	Goal
Requirement	refinesTo	System_Feature

3.2. Implementation

This section on implementation will cover the environment that ontologies run in, the software and tools needed to build ontologies, and the process of actually creating the suggested ontology. For testing, we will use two SRS documents from real-world systems that were founded by university students. They are titled: i) "Undergraduate Final Admission System of JUST" and ii) "Online Job Application of JUST."

3.2.1. Necessary tools and softwares

RDF/RDFS: a language for knowledge representation of resources on the Internet is called the resource description framework. The majority of data stored on the internet is contained in ontologies, and a large portion of that data is referenced using RDF [24]. An extension of RDF is called the RDF vocabulary description language schema, or RDF(S). RDF(S) gives language users greater freedom to define every term that will be used when defining web resources. Users can define resources as instances in multiple classes, and it offers the ability to create hierarchical class relationships [24].

Web ontology language (OWL): created by the W3C, the OWL allows applications to do more than just display the data found in documents; it also allows them to process the content of that data. OWL has a larger vocabulary than XML, RDF, and RDF(S), which enhances the degree to which machines can process and interpret that data. In support of the Semantic Web, this language is the most recent language specification approved by the W3C [24].

SPARQL: the common protocol and query language for RDF and linked open data databases is SPARQL. Because it is made to query a wide range of data, it can effectively retrieve information that is hidden in data that is not uniform and is stored in different formats and sources [26]. To put it simply, SPARQL is to knowledge graphs and the Semantic Web what SQL is to relational databases [27].

Protégé: for creating and managing ontologies, Protégé has emerged as the most popular tool. To facilitate the creation and administration of OWL ontologies, a desktop system called Protégé 5 offers a wide range of sophisticated functionalities [28]. Protégé 5.5.0 will be used to construct the software requirement ontology.

Pellet, 'an intelligent reasoner': Pellet is an open-source OWL-DL reasoner built on Java that can be used for ontology classification, consistency checks, and ontology validation. Additionally, the reasoner establishes if it is possible to confirm classes and relationships within ontologies as consistent or satisfiable. Pellet can also be used to access class hierarchy information, which serves as a visual aid for describing the ontology [24].

Table 4. Data properties list

Domain	Data property	Range
Requirement	content	xsd:string
System_Requirement	content	xsd:string
External_Interface	content	xsd:string
System_Feature	content	xsd:string
Stakeholder	content	xsd:string
External_Interface	descriptionOfCharacteristics	xsd:string
Requirement	elicitedDate	xsd:dateTime
Requirement	hasDescription	xsd:string
Requirement	hasInput	xsd:string
Requirement	hasPriorityLevel	xsd:string
System_Feature	hasValidationState	xsd:string
Requirement	isValidated	xsd:boolean
Requirement_Artefact	label	xsd:string
Requirement	label	xsd:string
Document_Section	label	xsd:string
System_Requirement	label	xsd:string
Stakeholder	lastName	xsd:string
SRS_Document	operatingEnvironment	xsd:string
SRS_Document	purposeOfSystem	xsd:string
Requirement_Elicitation_Technique	content	xsd:string
Document_Section	content	xsd:string
Requirement_Artefact	content	xsd:string
Appendix	content	xsd:string
Document_Section	descriptionOfCharacteristics	xsd:string
Stakeholder	designation	xsd:string
Stakeholder	firstName	xsd:string
System_Feature	hasDescription	xsd:string
Requirement	hasOutput	xsd:string
SRS_Document	hasScope	xsd:string
Requirement	hasValidationState	xsd:string
Appendix	label	xsd:string
Stakeholder	label	xsd:string
System_Feature	label	xsd:string
Requirement_Elicitation_Technique	label	xsd:string
External_Interface	label	xsd:string
Stakeholder	middleName	xsd:string
SRS_Document	productPerspective	xsd:string
Requirement	validationDate	xsd:dateTime

3.2.2. Ontology development

The ontology must be constructed in real-world applications once the class hierarchy, object properties, and data properties are located in Tables 2 to 4. We will use Protégé [28] 5.5.0 version to construct the software requirement ontology.

- Classes and subclasses: we should run the Protégé application and give name of the ontology as SRS ontology. Then we have to incorporate the class hierarchy founded on Table 2. SRS_Document, the subclass of the thing class on Figure 2, is our primary class. All the other classes will be the sub-class of it. All of the classes and subclasses listed in Table 2 have been added correspondingly.

– Object properties: the next step is to add object properties, which are listed in Table 3 and will be a subclass of topObjectProperty. The graphical form in Figure 3 can be identified. The domain and ranges for each object property must be defined after creating as Figure 3 by adhering to Table 3, which will later create relationships between various classes in the ontology.
– Data properties: the next step is to include the data properties, which are listed in Table 4 and will be a subclass of topDataProperty. The data for any individual will be represented by Table 4, so after creating as Figure 4, we must define the domain and ranges for each object property. The term "data type" refers to the ranges for data properties.

Onto graph allows us to create taxonomies for classes after the ontology is created on Protege. Requirement, requirement artefact, and stakeholder taxonomy are depicted in Figures 5 to 7, respectively. The requirement class is divided into three other sub_classes as shown in Figure 5. The system has some requirements. Requirement_Artefact class has also some subclasses. From them, the story class has again two subclasses, and the limitation class is equal to the obstacle class, as shown in Figure 6. The stakeholder may be an individual or a company. The class stakeholder has subclasses. Again, the customers class has two subclasses as shown in Figure 7.

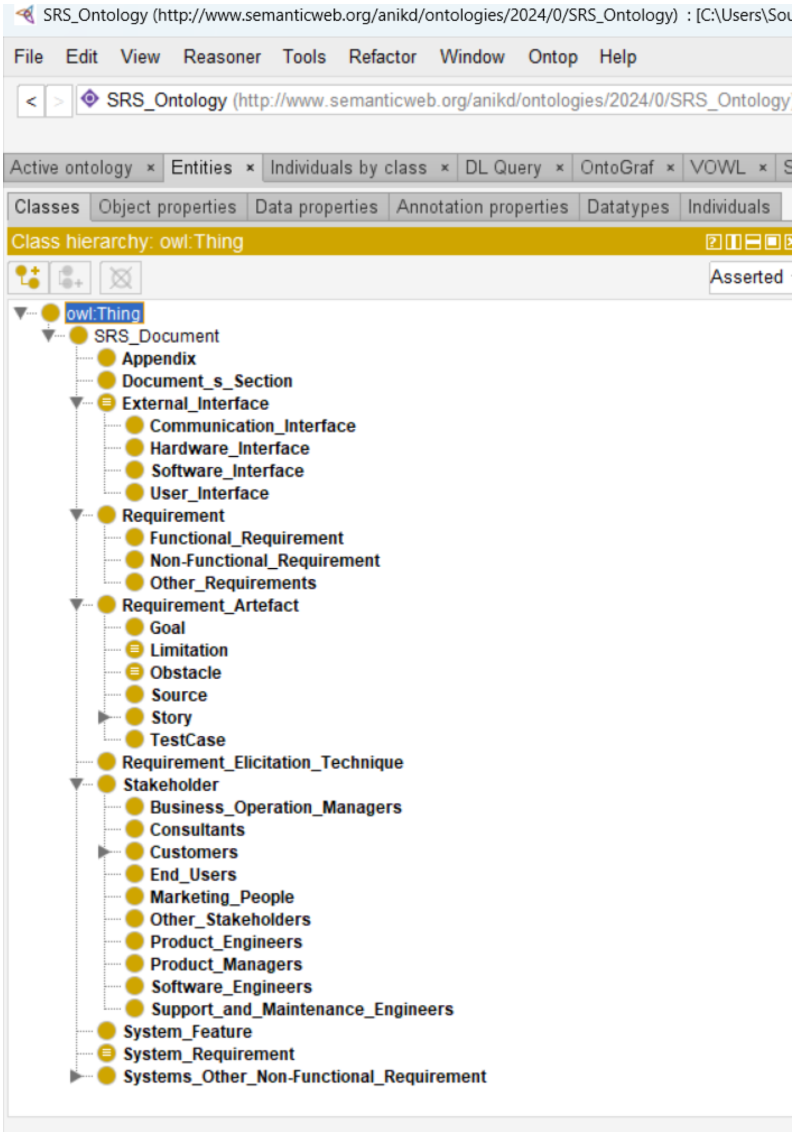

Figure 2. Class heirachy in Protege

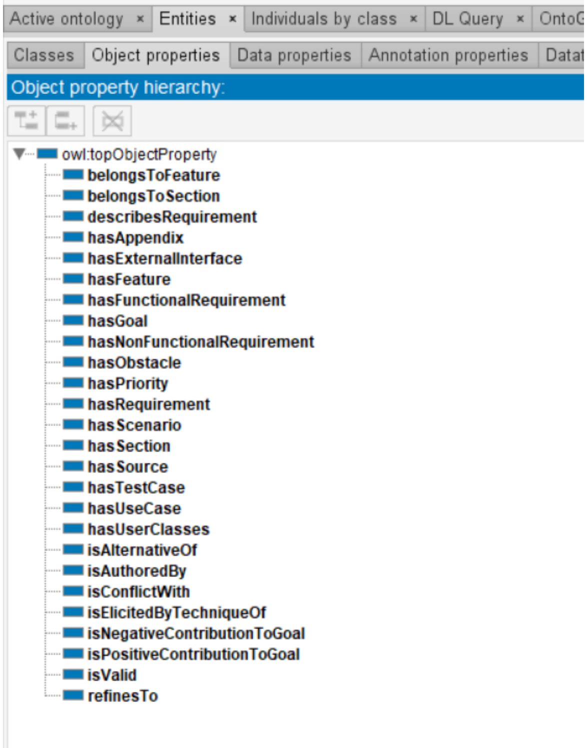

Figure 3. Object properties in Protégé

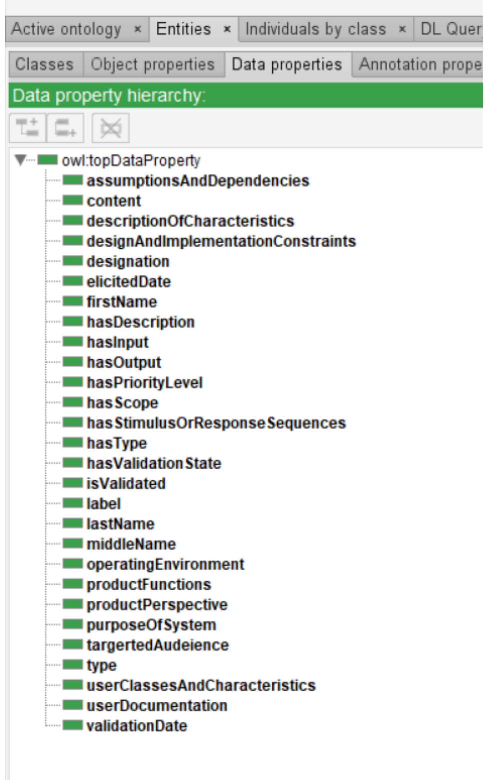

Figure 4. Data properties in Protégé

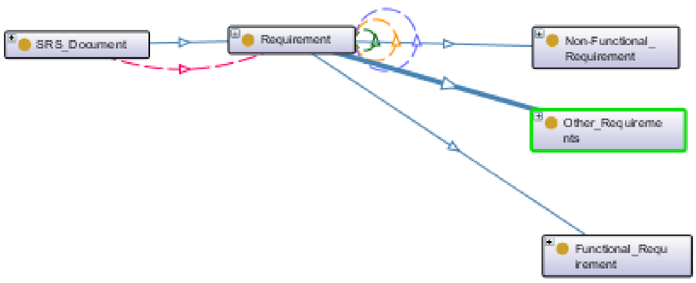

Figure 5. Requirement taxonomy

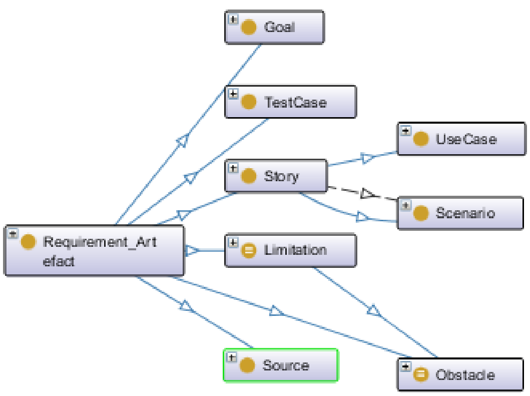

Figure 6. Requirement artefact taxonomy

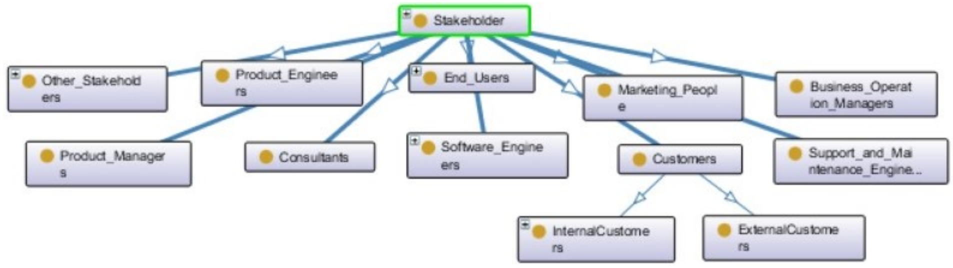

Figure 7. Stakeholder taxonomy

4. EVALUATION AND DISCUSSION

We will discuss the evaluation procedures, the proposed ontology's evaluation process, and how to ensure that our research goals are met in this section.

4.1. Evaluation

A set of criteria is examined using measurements and techniques in ontology evaluation. The main differences between the ontology evaluation approaches are the number of criteria they target and the rationale behind their assessment of the taxonomy [23]. Numerous methods of evaluation exist; however, the four most feasible and effective methods are listed as: gold-standard-based, corpus-based, task-based, and criteria-based [23], [29]. Competency questions (CQs) are a common process used in ontology validation [5]. We can declare ontology to be validated if the approaches described above satisfy the CQs. CQs can be easily completed with software-dependent or task-based approaches using SPARQL queries or reasoning [30]. A few competency questions for our suggested ontology are displayed in Table 5. Our ontology will be consistent if we receive answers to those questions [8].

Table 5. Competency questions list

Q. No.	Competency questions (CQs)
CQ 1	Given a feature, what are the functional requirements related to it?
CQ 2	Is the SRS unambiguous?
CQ 3	Can any requirement artefact be determined with a requirement?
CQ 4	Can it represent the knowledge correctly?

4.1.1. Reasoner: Pellet

Pellet works with the Protege framework and can accomplish all of the aforementioned tasks on OWL ontologies [24]. It is compatible with Proteg´e. To initiate the reasoner, click "Reasoner," choose "Pellet," and then click "Start Reasoner." Next, switch from asserted mode to inferred mode, choose any individual, and if a new connection is formed by ontology, it will turn yellow. For a clearer understanding, see Figure 8. Whereas the ontology infers five new relations in Figure 8. In addition, a reasoner can identify ontology inconsistencies. It provides recommendations for the ontology to address issues if it is inconsistent or the class hierarchy is incorrect. The reasoner runs our ontology error-free, indicating consistency in our ontology.

4.1.2. SPARQL queries

The common protocol and query language for RDF and linked open data databases is SPARQL [26]. It is utilized in the semantic web to retrieve information from ontologies in relation to a legitimate relationship between the ontology's instances. Here, we present some results from SPARQL queries on both systems. The successful retrieval of system features related to the university admission system is demonstrated by the SPARQL query in Figure 9. The query shown in Figure 10 retrieves every system feature related to the online job application system. Functional requirements are displayed in relation to system features in Figure 11.

4.1.3. Metric based evaluation

An additional evaluation that is predicated on the computation of ontology quality measures is feasible. OntoMetrics [10], a free online tool for metric definition and computation, is provided by the author of

[31]. Five distinct metrics are calculated in the suggested ontology using OntoMetrics and are divided into two groups: knowledge base and schema. The equations presented in Table 6 were originally formulated by Haridy *et al.*[10]. Specifically, (1), (2), and (3) are employed to evaluate the accuracy of the ontology, while (4) and (5) are used to assess its conciseness.

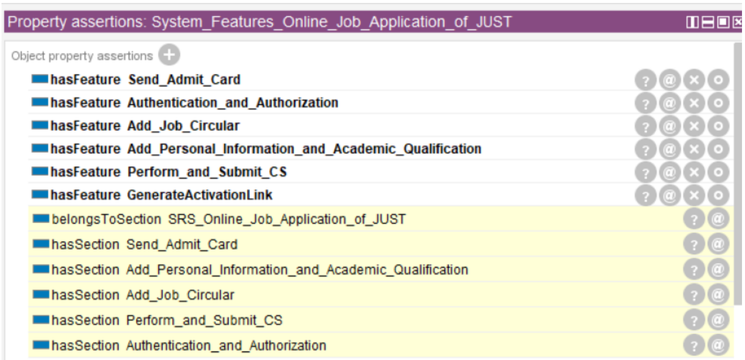

Figure 8. A system feature of online job application

Figure 9. System features for university admission system

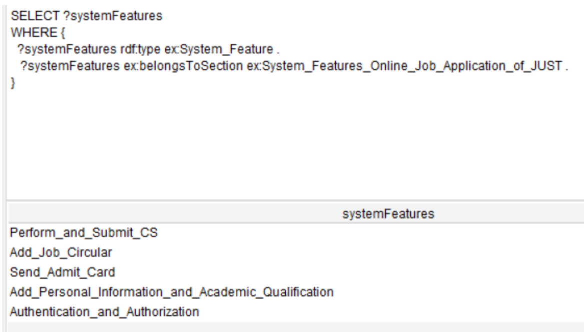

Figure 10. System features online job application system

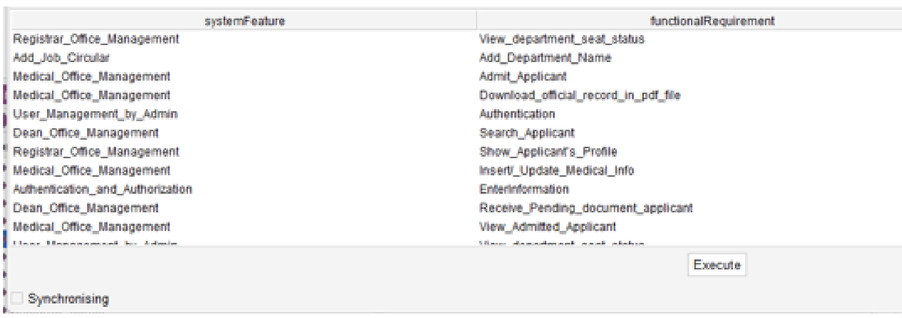

Figure 11. Requirements with features

Table 6. OntoMetrics equations

Metric	Equation		Description
Attribute richness (AR)	$AR = \|att\|/\|C\|$	(1)	|att| = total number of attributes; $\|C\|$ = total number of classes
Inheritance richness (IR)	$IR = \|H\|/\|C\|$	(2)	$\|H\|$ = number of subclass relations; $\|C\|$ = total number of classes
Relationship richness (RR)	$RR = \|P\|/(\|H\| + \|P\|)$	(3)	$\|P\|$ = number of non-inheritance relations; $\|H\|$ = number of inheritance relations
Average population (AP)	$AP = \|I\|/\|C\|$	(4)	$\|I\|$ = total number of individuals (instances); $\|C\|$ = total number of classes
Class richness (CR)	$CR = \|C'\|/\|C\|$	(5)	$\|C'\|$ = number of classes with at least one instance; $\|C\|$ = total number of classes

Figure 12 represents the ontology metrics for the proposed one, where axiom, class count, object property, and data property count are shown. The OntoMetrics results of the proposed ontology are displayed in Table 7. Furthermore, a comparison is made with three more domain-specific ontologies (Hontology, Travel, and EGYTour) [10].

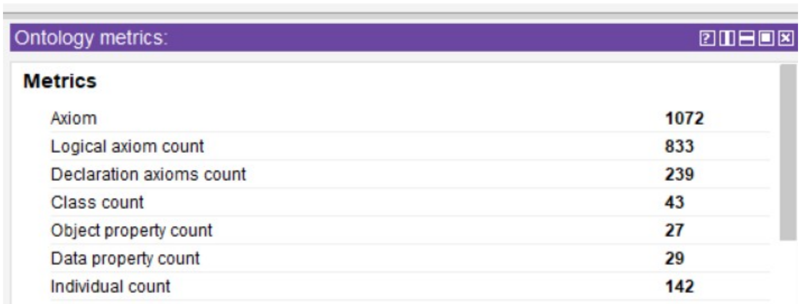

Figure 12. Proposed ontology metrics

Table 7. Comparison of OntoMetrics

Ontology	AR	IR	RR	AP	CR
Hontology	0.1092	0.9613	0.3209	0.0000	0.0000
Travel	0.1143	0.8571	0.4340	0.4000	0.2286
EGYTour	1.0789	1.6930	0.3216	7.0263	0.4605
Proposed	**0.6744**	**0.9767**	**0.4247**	**3.3023**	**0.4419**

4.2. Discussion

Criteria-based methods are the most effective for assessing the clarity of an ontology [23]. The role of CQs in ontology validation is significant. When the test result matches the correct response to the CQ, the ontology is considered validated. Requirements are obtained in Figure 11 based on its features that satisfy our first CQ (CQ 1). We checked the ontology for ambiguity and inconsistency using Pellet Reasoner, and the results (Figure 8) were positive, meeting our CQ (CQ 2). IEEE std. 830-1998 [13] served as the gold standard for this ontology, and it adheres to the standard correctly. Likewise, the ontology mapping for both SRS document datasets was correct, indicating that the SRS ontology fulfilled CQ (CQ 4) by successfully supplying users with knowledge. Finally, Figure 9 and Figure 10 retrieve features for a requirement, thereby fulfilling another CQ (CQ 3). We can declare this to be clear and consistent because this ontology successfully satisfies all of our CQs.

Table 7 shows that the proposed ontology and previous models differ significantly in ontology metrics. The suggested ontology has an AR score of 0.6744, which puts it in second place after EGYTour. With a score of 0.9767 in IR, it comes in second place, again behind EGYTour. The suggested model's RR of 0.4247 is extremely close to Travel's highest value of 0.434, suggesting a balanced use of both inheritance and non-inheritance relationships. The suggested ontology ranks second, only behind EGYTour, with scores of 3.3023 for AP and 0.4419 for CR. The suggested ontology consistently performs well across all dimensions, despite not being at the top in any one.

The proposed ontology is unique in that it is independent of domains. In contrast to earlier ontologies like EGYTour and Travel, which are domain-specific, the suggested model is made to be cross-domain applicable, which means it may be used for a variety of software requirement specifications, including e-governance, education, recruitment, and e-commerce. Together with its continuously high metric performance, this generality makes the suggested ontology a more scalable and reusable solution for real-world applications in a variety of domains. The ontology is publicly available on the internet via GitHub [32].

5. CONCLUSION

A generalized ontology framework for SRS is presented in this study to solve the common problems of ambiguity, inconsistency, and lack of standardization in traditional requirement engineering procedures. By combining formal ontology principles, reasoning tools (Pellet), and semantic queries (SPARQL), we showed how our method improves software requirements' traceability, reusability, and clarity. This work's importance arises from its cross-domain generalization, which makes it appropriate for a variety of applications from recruitment platforms to educational systems, without requiring significant customization. In alignment with IEEE 830-1998, this ontology provides the research community with a reusable knowledge model that serves as a basis for further research into intelligent requirement engineering and model-driven development. The framework will be enhanced with visual editors and user-friendly tools in the future, NLP pipelines will be integrated for automatic ontology population, and domain-specific modules will be added to the ontology for deployment that can be customized. In order to get closer to the objective of completely intelligent and collaborative software engineering environments, these improvements are aimed at increasing acceptance in both academic and industry environments.

FUNDING INFORMATION

Authors state no funding involved.

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] R. Rawat, R. K. Chakrawarti, A. S. A. Raj, G. Mani, K. Chidambarathanu, and R. Bhardwaj, "Association rule learning for threat analysis using traffic analysis and packet filtering approach," *International Journal of Information Technology*, vol. 15, no. 6, pp. 3245–3255, Aug. 2023, doi: 10.1007/s41870-023-01353-0.

[2] N. F. Noy and D. L. Mcguinness, "Ontology development 101: a guide to creating your first ontology." Stanford Knowledge Systems Laboratory Technical Report KSL-01-05, 2001.

[3] K. Siegemund, Y. Zhao, J. Z. Pan, and U. Aßmann, "Measure software requirement specifications by ontology reasoning," in *8th International Workshop on Semantic Web Enabled Software Engineering (SWESE'2012)*, 2012, pp. 1–15.

[4] E. M. Zamzami and E. K. Budiardjo, "Documenting software requirements specification using R2UC ontology." Universitas Sumatera Utara, 2012.

[5] H. Bencharqui, S. Haidrar, and A. Anwar, "Ontology-based requirements specification process," *E3S Web of Conferences*, vol. 351, p. 01045, May 2022, doi: 10.1051/e3sconf/202235101045.

[6] U. Ahmed, A. Farooq, and T. Farhat, "ReqSpecOnto: investigating explicit software requirements specification," *Innovative Computing Review*, vol. 1, no. 2, pp. 44–70, Dec. 2021, doi: 10.32350/icr.0102.03.

[7] M. P. S. Bhatia, A. Kumar, and R. Beniwal, "Ontologies for software engineering: past, present and future," *Indian Journal of Science and Technology*, vol. 9, no. 9, pp. 1–16, Mar. 2016, doi: 10.17485/ijst/2016/v9i9/71384.

[8] V. Castañeda, L. Ballejos, M. L. Caliusco, and M. R. Galli, "The use of ontologies in requirements engineering," *Global Journal of Researches in Engineering*, vol. 10, no. 6, pp. 2–8, 2010.

[9] D. Dermeval *et al.*, "Applications of ontologies in requirements engineering: a systematic review of the literature," *Requirements Engineering*, vol. 21, no. 4, pp. 405–437, 2016, doi: 10.1007/s00766-015-0222-6.

[10] S. Haridy, R. M. Ismail, N. Badr, and M. Hashem, "An ontology development methodology based on ontology-driven conceptual modeling and natural language processing: tourism case study," *Big Data and Cognitive Computing*, vol. 7, no. 2, 2023, doi: 10.3390/bdcc7020101.
[11] S. Ahmed and B. Ahmad, "Transforming requirements to ontologies," Jönköping University, School of Engineering, 2020.
[12] H. Tan, M. Ismail, V. Tarasov, A. Adlemo, and M. Johansson, "Development and evaluation of a software requirements ontology," in *SKY 2016 - 7th International Workshop on Software Knowledge, Proceedings - In conjuction with IC3K 2016*, 2016, pp. 11–18, doi: 10.5220/0006079300110018.
[13] "IEEE recommended practice for software requirements specifications." IEEE, Piscataway, NJ, USA, Jun. 25, 1998, doi: 10.1109/IEEESTD.1998.88286.
[14] "Software engineering: report on a conference sponsored by the Nato Science Committee." Brussels: Scientific Affairs Division, NATO, Garmisch, Germany, 1968.
[15] 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, 2009, doi: 10.1109/TKDE.2008.209.
[16] P. Bourque and R. E. Fairley, *Guide to the software engineering body of knowledge - SWEBOK V3.0*. IEEE and IEEE Computer Society, 2014.
[17] K. Yue, "What does it mean to say that a specification is complete?," in *Proceeding of IWSSD-4, Fourth International Workshop on Software Specification and Design*, 1987, pp. 42–49.
[18] A. M. M. Khair and F. Meziane, "The use of ontologies in software elicitation," *Humanitarian and Natural Sciences Journal*, vol. 2, no. 9, pp. 427–441, Sep. 2021, doi: 10.53796/HNSJ2923.
[19] R. A. Elliot and E. B. Allen, "Creating a software requirements specification document using an ontology based methodology," *International Journal of Advanced Research in Science, Engineering and Technology*, vol. 3, no. 9, pp. 2616–2630, 2016.
[20] V. Ermolayev, "OntoElecting requirements for domain ontologies the case of time domain," *Enterprise Modelling and Information Systems Architectures. International Journal of Conceptual Modeling*, vol. 13, no. Sp. Issue, pp. 86–109, 2018, doi: 10.18417/emisa.si.hcm.9.
[21] M. P. S. Bhatia, A. Kumar, R. Beniwal, and T. Malik, "Ontology driven software development for automatic detection and updation of software requirement specifications," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 23, no. 1, pp. 197–208, 2020, doi: 10.1080/09720529.2020.1721884.
[22] D. Jones, T. Bench-Capon, and P. Visser, "Methodologies for ontology development," In *Proceeding of ITKNOWS Conference of the 15th IFIP World Computer Congress*, 1998.
[23] J. Raad and C. Cruz, "A survey on ontology evaluation methods," in *Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 2015, vol. 2, pp. 179–186, doi: 10.5220/0005591001790186.
[24] R. A. Elliott, "Software requirements elicitation, verification, and documentation: an ontology based approach," Mississippi State University, 2012.
[25] R. S. Pressma and B. R. Maxim, *Software engineering: a practitioner's approach*, 8th ed. McGraw-Hill Education, 2014.
[26] "What Is SPARQL?," *Ontotext A Graphwise Company*, 2016. [Online]. Available: https://www.ontotext.com/knowledgehub/fundamentals/what-is-sparql.
[27] M. DeBellis, "A practical guide to building OWL ontologies using Protégé 5.5 and Plugins," *ResearchGate Preprint*, 2021.
[28] M. A. Musen, "The Protégé project: a look back and a look forward.," *AI matters*, vol. 1, no. 4, pp. 4–12, 2015, doi: 10.1145/2757001.2757003.
[29] J. Brank, M. Grobelnik, and D. Mladenic, "A survey of ontology evaluation techniques," in *Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005)*, 2005, pp. 166–170.
[30] E. Blomqvist, A. Seil Sepour, and V. Presutti, "Ontology testing - methodology and tool," in *Knowledge Engineering and Knowledge Management. EKAW 2012*, Springer Berlin Heidelberg, 2012, pp. 216–226.
[31] B. Lantow, "OntoMetrics: putting metrics into use for ontology evaluation," in *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, 2016, pp. 186–191, doi: 10.5220/0006084601860191.
[32] S. Kundu, "SRS_Ontology," *Github*, 2024. [Online]. Available: https://github.com/SouravKunduSK/SRS_Ontology.

BIOGRAPHIES OF AUTHORS

Sourav Kundu is a lecturer at Bangladesh University of Business and Technology (BUBT), in the Department of Computer Science and Engineering. Structured programming language, object-oriented programming language, discrete mathematics, and data communication are some of the courses he has taught. He graduated from Jashore University of Science and Technology (JUST), Bangladesh, with a Bachelor of Science degree in computer science and engineering. Software engineering, machine learning, and knowledge-based technologies are the main areas of his research interest. He can be contacted at email: souravkunduss1@gmail.com.

Soumay Kanti Das is currently working as junior software developer at CSM, Bangladesh. He completed B.Sc. in computer science and engineering from Jashore University of Science and Technology. His is interested in research in the field of software engineering, and knowledgebase technologies. He can be contacted at email: 180148.cse@student.just.edu.bd.

Abu Rafe Md Jamil is a teacher at Bangladesh's Jashore University of Science and Technology (JUST) in the area of computer science and engineering. He has been working in this field for more than two years, and his areas of expertise are research and programming, specifically in the area of reinforcement learning for transportation systems. His objectives are to assist students reach their greatest potential, stimulate critical thinking, and arouse curiosity. He can be contacted at email: arm.jamil@just.edu.bd.

Md Kamrul Islam received his bachelor's degree in computer science and engineering in 2009 from Khulna University of Engineering and Technology, Bangladesh. He started his career in the software industry, especially in software development jobs. At the end of 2010, he finally joined as a faculty member in the Department of Computer Science and Engineering, Jashore University of Science and Technology (JUST), Bangladesh. At JUST, he is fully involved in academic and research-oriented jobs. He has completed his Master's degree in Computing from Universiti Malaysia Pahang, Malaysia in 2019 and Ph.D. in computer science from University of Lorraine, France in 2023. His research interest includes big data processing, especially data mining, graph mining, graph representation learning, knowledge graphs, drug repurposing and social network analysis. He can be contacted at email: mk.islam@just.edu.bd.

SK. Shalauddin Kabir is a lecturer at Jashore University of Science and Technology (JUST), Bangladesh, in the Computer Science and Engineering (CSE) Department. He was employed as a lecturer in the Computer Science and Engineering (CSE) Department at Northern University of Business and Technology Khulna (NUBTK), Khulna, Bangladesh, prior to joining JUST. Sk. Shalauddin Kabir graduated from Jashore University of Science and Technology (JUST), Bangladesh, with a B.Sc. (Engg.) and an M.Sc. (Engg.) in Computer Science and Engineering (CSE). At JUST's fourth convocation in 2023, he received the Dean's Award for demonstrating exceptional academic achievement in his bachelor's degree. AI, SDN, IoT, image processing, and machine learning are some of his areas of interest. He can be contacted at email: sks.kabir@just.edu.bd.

Mostafijur Rahman Akhond is employed at Jashore University of Science and Technology as an assistant professor at the moment. Under the guidance of Prof. Young Koo Lee at the Data and Knowledge Engineering (DKE) Lab, he earned his master's degree in computer science and engineering from Kyung Hee University in South Korea. The South Korean President Scholarship allowed Mr. Akhond to complete his master's degree. By the time of the master's program, he was also employed as a research assistant. Mr. Akhond majored in software engineering and earned his bachelor's degree from the University of Dhaka's Institute of Information Technology. He is now working toward a Ph.D. at York University's Department of Electrical Engineering and Computer Science, located at 4700 Keele St., North York, Canada. He can be contacted at email: mr.akhond@just.edu.bd.