Journal of Software Engineering and Applications, 2023, 16, 31-49
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

An Approach towards Goal-Oriented Requirements Ontology: Consistency and Completeness Based Requirements Analysis

Mohammad Mustafa Taye[1], Said Ghoul[2]

[1]Software Engineering Department, Philadelphia University, Amman, Jordan
[2]Research Laboratory on Bio-Inspired Software Engineering, Philadelphia University, Amman, Jordan
Email: mtaye@philadelphia.edu.jo, sghoul@philadelphia.edu.jo

How to cite this paper: Taye, M.M. and Ghoul, S. (2023) An Approach towards Goal-Oriented Requirements Ontology: Consistency and Completeness Based Requirements Analysis. *Journal of Software Engineering and Applications*, **16**, 31-49.
https://doi.org/10.4236/jsea.2023.162003

Received: January 23, 2023
Accepted: February 25, 2023
Published: February 28, 2023

Abstract

The paper presents a new approach to managing software requirement elicitation techniques with a high level of analyses based on domain ontology techniques, where we established a mapping between user scenario, structured requirement, and domain ontology techniques to improve many attributes such as requirement consistency, completeness and eliminating duplicate requirements to reduce risk of overrun time and budgets. One of the main targets of requirement engineering is to develop a requirement document with high quality. So, we proposed a user interface to collect all vital information about the project directly from the regular user and requirement engineering; After that, the proposal will generate an ontology based on semantic relations and rules. Requirements Engineering tries to keep requirements throughout a project's life cycle consistent necessities clear, and up to date. This prototype allows mapping requirement scenarios into ontology elements for semantically interrupted. The general points of our prototype are to guarantee the identification requirements and improved nature of the Software Requirements Specification (SRS) by solving incomplete and conflicting information in the requirements specification.

Keywords

Requirements Engineering, Requirements Elicitation, Domain Ontology, Ontology

1. Introduction

Requirements Engineering [1] is a document used as a contract between the

customer and developer to identify and specify the requirements. A good document should have attributes such as unambiguous, completeness, consistency, and verifiable [2]. Therefore, lacking or inaccurate requirements cause the entire system's development to be incomplete or erroneous at every stage. On the other hand, it is challenging to create such a document the first time and when making any change is very hard to handle the modification. Therefore, we proposed this prototype.

Several demands, aspirations, and requirements are frequently at odds with one another while developing software systems because of various stakeholder perspectives. Elicitation errors are often essential factors in systems failures with very high costs, either in the total loss or correcting errors [3].

The main process of Requirements Engineering is shown in the following **Figure 1**; the process starts with requirement elicitations which concern how to collect needs and goals from stakeholders. Indeed, the main idea of requirements elicitation techniques is determining the problems, opportunities, and all potential needs of the clients; because of this, a software engineer can develop systems that resolve those issues and cover those opportunities and/or additionally address clients' needs [4]. The next process is to analyze this information to make sure about it, then specify them in many different ways to make more knowledge for the developing team. The last process is to apply the verification and validation techniques to check for any inconsistency or completeness [5].

All projects are dependent on requirement elicitation to achieve their goals. The process of requirement elicitation concentrates on communication among stakeholders and requirements engineers. Moreover, it is used to understand a problem and its application domain to improve the quality of extended requirements [6].

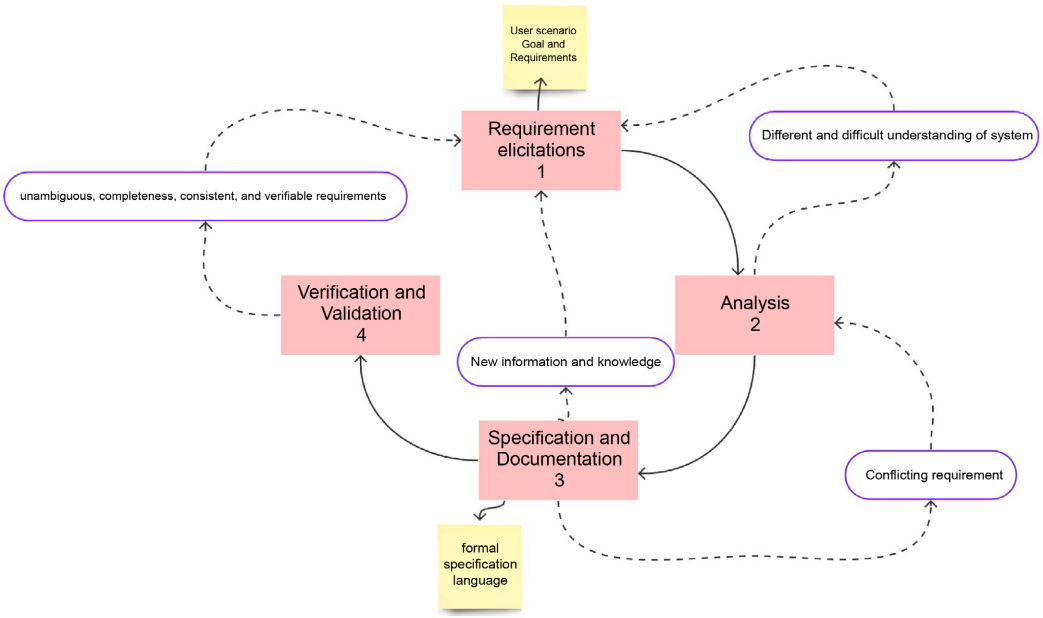

Figure 1. The main process of requirements Engineering.

Most successful or failed software projects are based on Requirements Engineering. Also, different requirements from versus stakeholders lead to incomplete and ambiguous requirements.

Modern IT projects are complex because of the high number and complexity of requirements, as well as because of the different backgrounds and terminologies of stakeholders. Consequently, suitable requirements management tools play a major role in the discourse of these challenges [7].

The software requirements specification is the yield from the requirements elicitation activity, which is written in a client requirements document.

The main activities of requirements engineers using requirement elicitation are:

- Knowledge and understanding of the domain and area where the system is applied.
- Understanding the specific customer problem.
- Knowledge environment and Interaction of system with others.
- Detailed examination of client needs.
- Define the constraints of the system that are applied.

There are essentially two types of Elicitation Techniques [8].

Direct approach: this strategy is used to get requirements from clients who can interact directly with the domain expert. It will be used to improve the understanding of the problems through Interviews, case studies, and Prototypes [9]. Analyses are examples.

Indirect approach: this strategy helps to get information that cannot be easily accessed or obtained from the direct methods. Questioners and Documents analyses are examples of this approach [8].

The main point is not just to collect requirements; it is normally understood that requirements. So, requirement elicitation is considered a complex process involving several activities with various available techniques, approaches, and tools for performing them. In fact, the best idea for using requirements elicitation is to apply a variety of techniques during different stages in the software development life cycle.

In general, incomplete and inconsistent requirements could appear from the gaining and specification of goals and requirements from different stakeholders and sources. Therefore, repairing inconsistent and incomplete requirements is vital to successfully model requirements specifications. In this research, we used first order logic to deal with problems [10]. Moreover, the backbone of this work is the Ontologies that provide conceptual models and the expressivity to capture requirements sufficiently; moreover, checking and reasoning rules are combined to measure the validity and coverage of the evolving requirements model [11].

Based on the previous point, we considered the main challenge for requirements engineering is dealing with inconsistencies and incompleteness in the requirements specification phase.

Obtaining the needs from the relevant parties and additional sources, knowing the application domain, it is important to thoroughly explore and examine the situation or "real world" in which the application will be used before begin-

ning the cycle of requirements elicitation. It is crucial to define the system's scope and thoroughly investigate the demands and preferences of all stakeholders during this activity [11].

Finding the requirements' sources makes it possible for requirements to be dispersed across several sources and to exist in various combinations [12]. Overall, product development opens up several possible hotspots for requirements that may be identified.

To eloquently clarify information regarding the challenges, problems, and customer demands, clients and topic experts are used. The depicted existing systems and processes, particularly when a current or legacy system has to be replaced, are another source for eliciting requirements.

Manuals, organizational structures, and reports regarding the existing system and business processes, as well as the requirements for the new framework and their justification and relevance, may all provide useful information about the association and environment [13].

Analyzing the stakeholders: Stakeholders are everyone who is interested in the system or who will be impacted by its development and deployment. They must thus be questioned as part of the requirements elicitation process. Stakeholders often comprise groups and individuals who may be internal and external to the company [10]. In general, the project sponsor (customer) is the most apparent stakeholder in the system. In some cases, the end users could be the most important. On the other hand, some systems could consider the system operations, customers, and partners, as stakeholders if they are affected [6].

Selecting the techniques, approaches, and tools to use—in general, selecting the elicitation technique depends on what the analyst knows, the analyst's favorite, a specific methodology that is being followed by the system development, and the decision of strategy administered exclusively by the instinct of the examiner to be viable in the current context.

In reality, conceptual domain modeling using ontologies will lessen the consequences of confusing and insufficient requirements procedures. "An explicit statement of a shared idea" describes the ontologies [14] [15]. Ontologies include machine-understandable notions and restrictions explicitly well-defined, typically understood, and well-covered. It might be used to represent, categories, and debate the required papers [15]. Ontology is a formal definition of items and the attributes, connections, limitations, and guidelines that control those connections.

In fact, any problem or inconsistency in requirements will lead to faulty software designs and implementations. Thus, one significant problem requirements engineers have to cope with is to improve Requirements Engineering, which will contribute to building better-quality software; this could lead moreover to reducing the risk of overrun time budgets and eliminating the risk of project failures [16].

We proposed a requirements analysis method by using domain ontology. To

specify the needs, goals, and tasks, this prototype starts with an elicitation page used by different stockholders and developing teams to collect all goals and needs about specific applications. Then the system will create ontology based on their input data, which will be examined manually by engineering and the reasoning system to check any inconsistency between functions.

In our proposal, we collected all information and knowledge from different users, then stored it in spirit ontology, then applied some matching and merging techniques on all these ontologies to create one global ontology about an application from resulted ontology we could Crete some UML diagram (use case) and requirements [1].

Accordingly, the Requirements Ontology empowers the documentation of organized, reusable, unambiguous, traceable, complete, and reliable requirements as requested by the IEEE specification for Software Requirement Specifications (SRS) [17].

The remainder of this paper is organized as follows: In Section 2, an overview of related work is given as literature review. The approach of our proposal is explained in Section 3. Section 4 gives analytical information. In Section 5, the evaluation analysis is explained, and we give the case study. In the last sections, we gave an overview of the work that will be done in the future and provided a conclusion.

2. Literature Review

Recently, many researchers have introduced different approaches for dealings and provided a new requirement elicitation based on ontologies to understand desired functions and the method for expressing stakeholders' and users' problems. The primary objective is to find a way toward looking for, learning, uncovering, procuring, and explaining client necessities to any computer-based system by communicating these needs to the system developers.

Surveys [18] and [19] have shown many studies demonstrating the effectiveness of using the ontology domain in supporting the requirements engineering process.

[20] has proposed a process for developing ontologies as a subprocess of the requirements engineering process.

While [21] used the domain as an infrastructure for specifying software requirements.

[22] proposed an approach to automating the validation process of knowledge about the requirements.

In [23], the ontological methodology is applied to improve the necessities of the designing cycle in the Agile process. The ontology is intended to work with user story templates. Ontology empowers the recognition of interchangeable ideas, hyperonymic and hyponymic relations between the concepts after it empowers the requirements engineering process to describe user stories that must be achieved for user roles of applications that include other roles.

In [24], two kinds of ontologies are recognized, which can be utilized to describe the product area being created: the application domain ontology and the application domain feature model ontology.

In [25], the requirements are considered a specific subset of a lot of information about the area. At the same time, the domain ontology is utilized as a "background source" while extricating requirements for a product item from characteristic language texts.

In [26], a way to deal with the automatic construction of an ontology from many stockholder stories is proposed. For text handling in the regular English language, the spaCy library is utilized, which considers parsing sentences dependent on a reliance tree, looking for named gatherings.

In [27], a way to deal with building up a recommender framework that bolsters the development of the Agile requirements is introduced. It is proposed to utilize the accompanying four ontologies: "Environmental Context Ontology", "Problem Domain Ontology", "Requirements Ontology," and "Agile Requirements Ontology".

Issues of requirements traceability are tended to in the [28] given to the improvement of casing cosmology which empowers to make a predictable model of necessities types for a particular software development project.

The significance of the created way to deal with extraction, computerization, and analysis of the requirements in natural language is dictated by the inconstancy of the necessities and the requirement for a speedy correlation of the requirements texts.

Thus, to summarize the above information about using ontologies in the field of requirements engineering:

1) If you are going to develop ontologies to represent knowledge about the requirements engineering process, you should consider requirements types and attributes of their quality.

2) If you are going to develop ontologies to represent knowledge about the application domain, you should take into account describing the components domain of the system, concepts, relationships, and actions.

3) if you are going to develop requirements ontologies, you should consider identifying conflicts and duplicates between the requirements.

Current requirements management tools ordinarily work with a typical requirements database, which all stakeholders can access to retrieve information on requirements content. Moreover, these kinds of tools could help all stakeholders to keep the overview of large amounts of requirements by supporting the following:

a) Requirements categorize the Requirements and cluster them into user-defined subsets.

b) Analysis and solve the conflict between Requirements (consistency checking).

c) Trace the Requirements and find the dependencies between them.

Requirements management suffers from many limitations, such as: Incompleteness, consistency, and conflict identification and tracking, especially with a huge number of requirements; therefore, the use of semantic technologies looks hopeful for addressing these limitations [29].

Ontologies deliver the means for describing the concepts of a domain and the relationships between these concepts in a way that could allow for automated reasoning to support categorization, conflict, and tracing of requirements.

We propose a prototype to deal with requirements engineering managing and elicitation designing dependent on a combination of the OWL ontology.

3. The Proposal Approach (Method)

In this paper, we presented the prototype of a semantic guidance system that supports normal users and requirements engineers to easily capture requirements. We built our prototype based on the important part information needed for developing modern IT projects. We collect all helpful information to write, analyze and improve requirements using domain ontology.

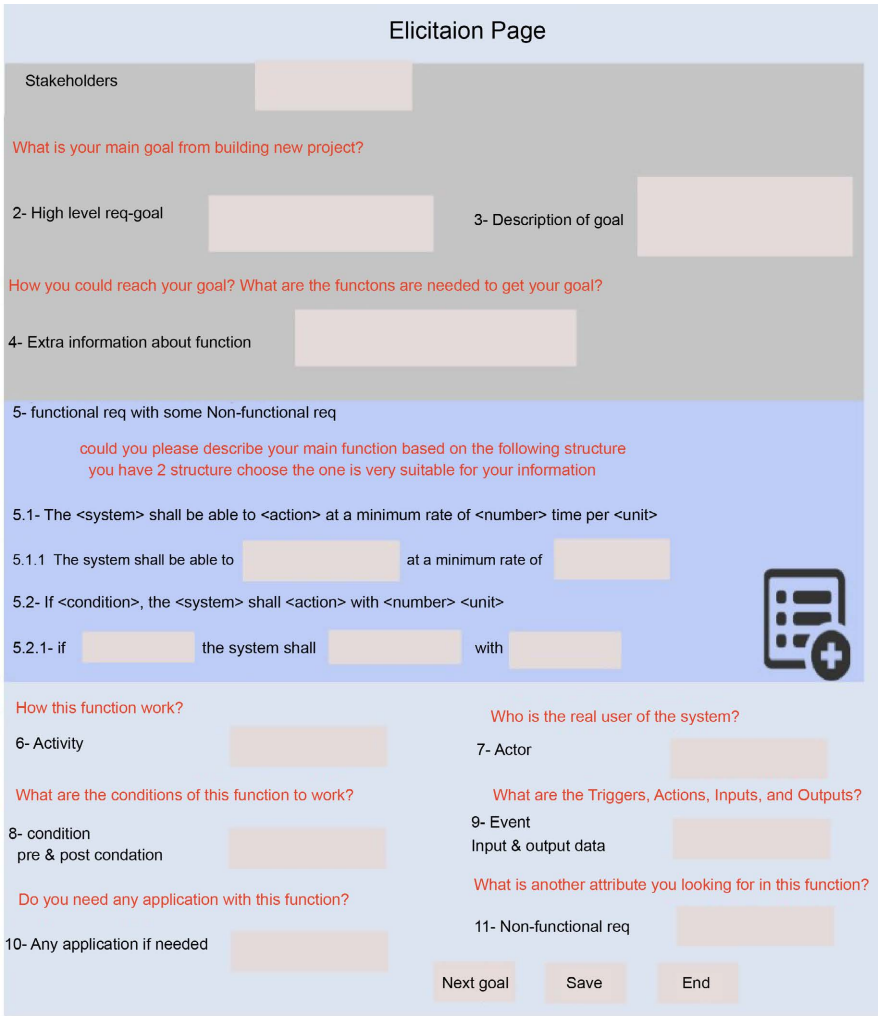

Figure 2. An interface of our prototype.

We built our prototype based on the important part information needed for developing modern IT projects, that allows users to specify the needs, goals, and tasks of an application. The system will make an ontology based on the data they give it. This ontology will be reviewed by engineering and the reasoning system to make sure there are no incompatibilities between features. In our approach, we compiled user data and insights into a single ontology, where they could be matched and fused using various methods.

Our prototype started with the above **Figure 2** user interface to collect the main information about a specific project from building a requirement ontology.

The following **Table 1** shows the main concepts of our proposal.

Natural language descriptions of topics of interest are captured by ontologies. The description section of an ontology includes the concepts that make up the ontology, together with their respective definitions and the connections between them. A "conceptualization" describes this kind of mental representation. In this context, ontology stands in for domain knowledge (domain ontology), and needs may be thought of as a subset of it.

Reasoning component: a logical theory that limits the desired model and includes: 1) integrity rules of the domain model expressing the domain knowledge; 2) derivation rules and constraint rules of the problem model.

While taxonomies have been widely used for modelling, ontologies provide inferential capabilities via reasoning.

Table 1. Main concepts of our proposal.

Concept	Definition
Domain Concept	Domain Concept is a kind of thesaurus that is used as pointers to concepts; we could unify different concepts or terms for the same terms by using synonym relationships among them.
Stakeholder	Groups of stakeholders, they specified what is expected from a system
Goal	Goals are indicative statements to identify and correlate requirements to be achieved by the system under development.
Requirement	(Functional Requirement): is an outcome of behavior that shall be provided by a function of a system, A non-functional requirement (also a quality requirement)
Function Requirements	Define what a product must do. Requirements artifacts comprise all concepts related to requirements knowledge
Non-Functional Requirements	Describe the quality attributes of a system.
Actor	The real users who interact with the system
Activity	High Level Function -> Sub-function -> Process -> Activity
Constraint	Constraints of the functionality.
Scenario	Scenario A textual description of a sequence of user actions that leads to the desired result.

4. Analysis

All of the requirement artifacts, which comprise all concepts related to requirements knowledge (stakeholder, concepts, relations, attributes, data, etc.), must be captured appropriately. We will specify requirements artifacts by using the ontology elements (e.g., classes, properties, instances of classes, and relations between instances). To specify the requirements, we use an Ontology as a metamodel, as Requirements Ontology.

In order to use an ontology, we have borrowed the idea from [30] as the following Figure 3. The potential uses of ontologies in RE embody the illustration of:

Requirements Ontology: The Requirements model imposes and sanctionative a selected paradigmatic manner of structuring needs.

Requirements Specification Document Ontology: Acquisition structures for domain information; In RE, completely different approaches and area units are used as intermediate steps for getting needs. The employment of ontologies for describing the structure of needs specification documents cut back the lean needs' specifications.

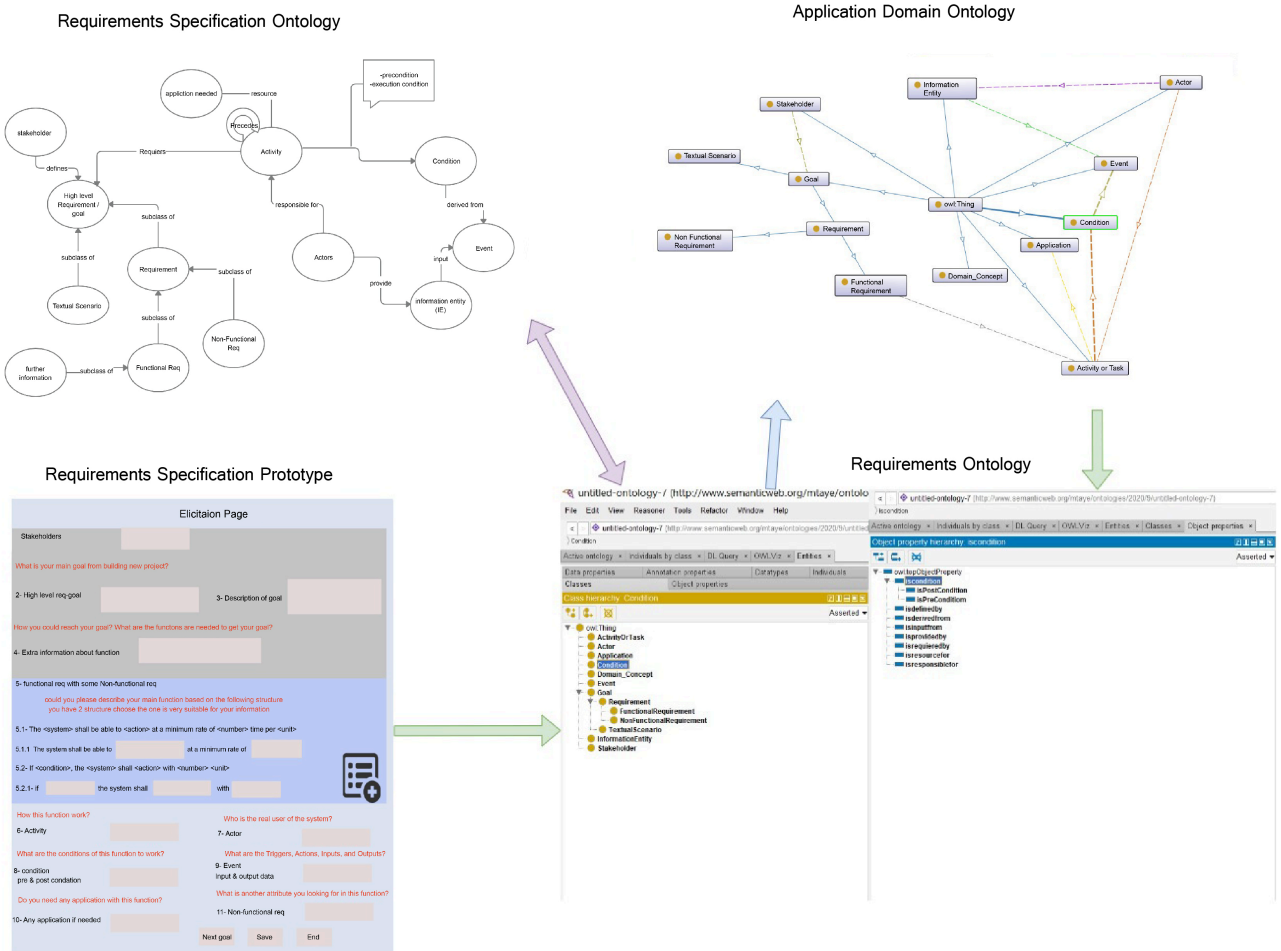

Figure 3. Ontology-based framework.

Application Domain Ontology: The information and fact of the applying domain Application Domain metaphysics. This metaphysics represents the application domain information, object properties and classes characteristic, and business information needed for building code applications in a very specific domain concept [31].

Our approach is semantic guidance which uses <concepts, relations, axioms> of ontologies elements to build on to define requirements [32].

For the design phase of software development, as well as for evaluating and reusing elicited needs, having a well-characterized requirements specification is crucial. Both the format of the document and its contents make up a specification. The way a document is laid out greatly impacts how its contents are understood. To be considered a successful software product, reuse must be a major component. It depends on the way in which needs are articulated, recorded, and organized.

However, a number of obstacles stand in the way of the reuse. Requirements specification papers, the recommendations conclude, may benefit especially from ontologies, especially when the content of such documents expands in a disorganized fashion. One solution to this problem is to structure the knowledge by adding semantics to the documents via metadata enrichment and the discovery of related, valuable material; this way, the semantics are written in a machine-understandable manner as shown in **Figure 4** and **Figure 5**.

In order to define requirements, we used the boilerplate, which states a textual requirement template. In fact, the term boilerplate was first used by Hull, Jackson, and Dick [33]. A boilerplate involves a classification of attributes and fixed syntax elements.

Indeed, many formulas are proposed for dealing with boilerplate, but we have chosen the following two structures, which we think are very suitable for most projects. Also, keeping the number of required boilerplates is relatively low and has high flexibility, as shown in **Table 2**. Moreover, we could use attribute values to state the entities in the ontology domain.

For example

1) The < system > shall be able to < action > at a minimum rate of < number> time per <unit>

2) If < condition> the < system > shall < action> with <number><unit>

Table 2. Boilerplate attributes.

attribute	Description
Action	The behavior of a system to be fulfilled
Number	A quantity ex. 3
Unit	Unit of measurement, ex. second
Condition	An event or condition that happened during system operation
System	The system or any part of it

Requirement artifacts contain all concepts connected to requirements knowledge (e.g., goal, obstacle, stakeholder, use-case, test-case). The object properties reproduce the relations between instances of the ontology classes.

Based on the ontology, the mapping rule is followed to produce a domain model.

- The classes in the ontology are transferred to the classes in the domain model.
- Entities are associated with instances.
- Properties in the ontology are linked to their corresponding counterparts in the domain model.
- Inheritance is mapped to the corresponding synonym for a connection between classes.

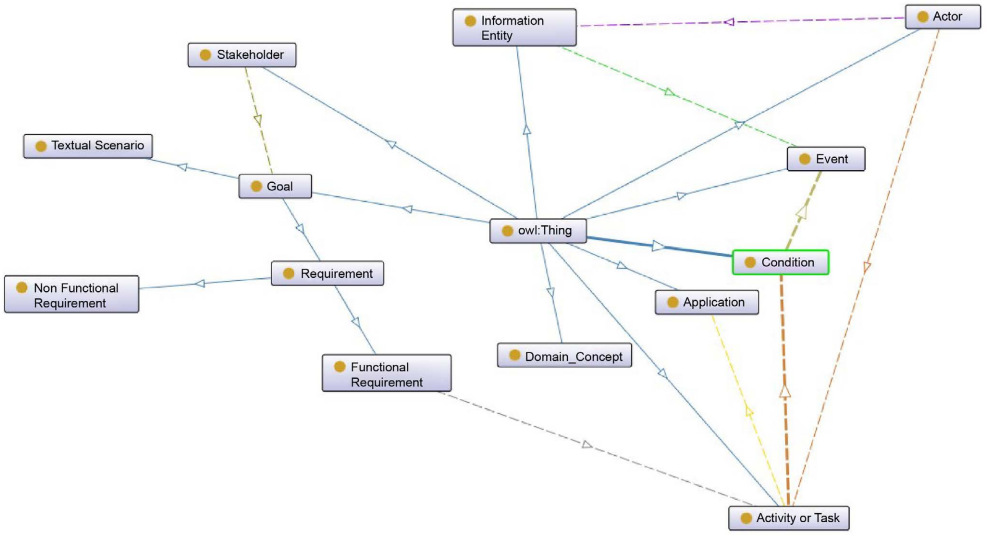

Figure 4. Requirements Taxonomy (artifacts of the requirements metamodel).

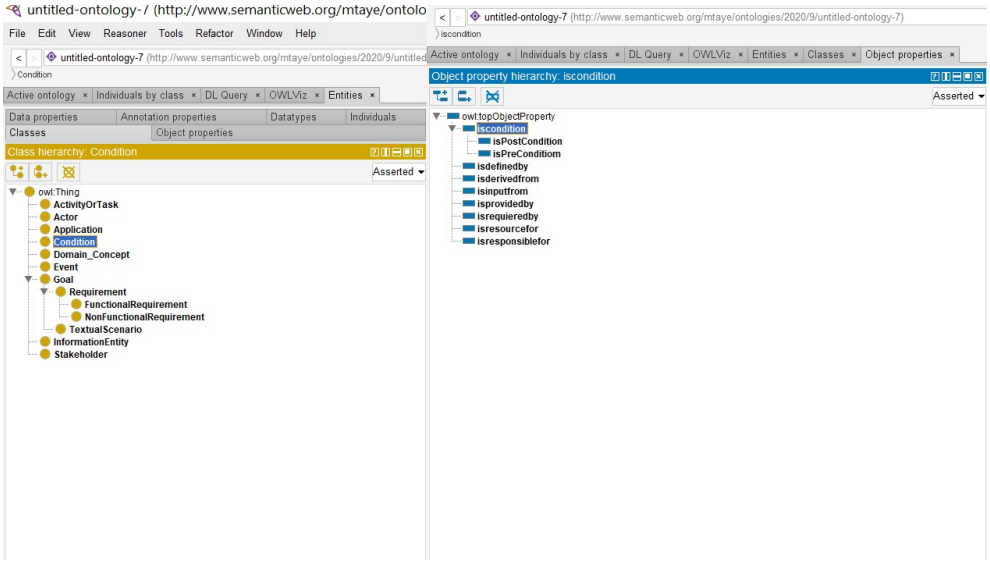

Figure 5. Taxonomy and axioms of the ontology elements in the Protégé Editor.

Domain Concept is a kind of thesaurus that is used as pointers to concepts; we could unify different concepts or terms for the same terms by using synonym relationships among them, as shown in Table 3.

In order to support the process of Requirements Engineering semantically, we established requirements engineering ontology Figure 6 below.

Table 3. The Ontology and their domains and ranges.

Domain	Object property	Range
ActivityOrTask	isCondition	Condition
ActivityOrTask	isPostCondition	Condition
ActivityOrTask	isPreCondition	Condition
Stakeholder	isDefinedby	Goal
Condition	isDerivedFrom	Event
informationEntity	isInputFrom	Event
Actor	isProvidedBy	informationEntity
FunctionalRequirement	isRequieredBy	ActivityOrTask
ActivityOrTask	isResourceFor	Application
Actor	IsResponsibleFor	ActivityOrTask

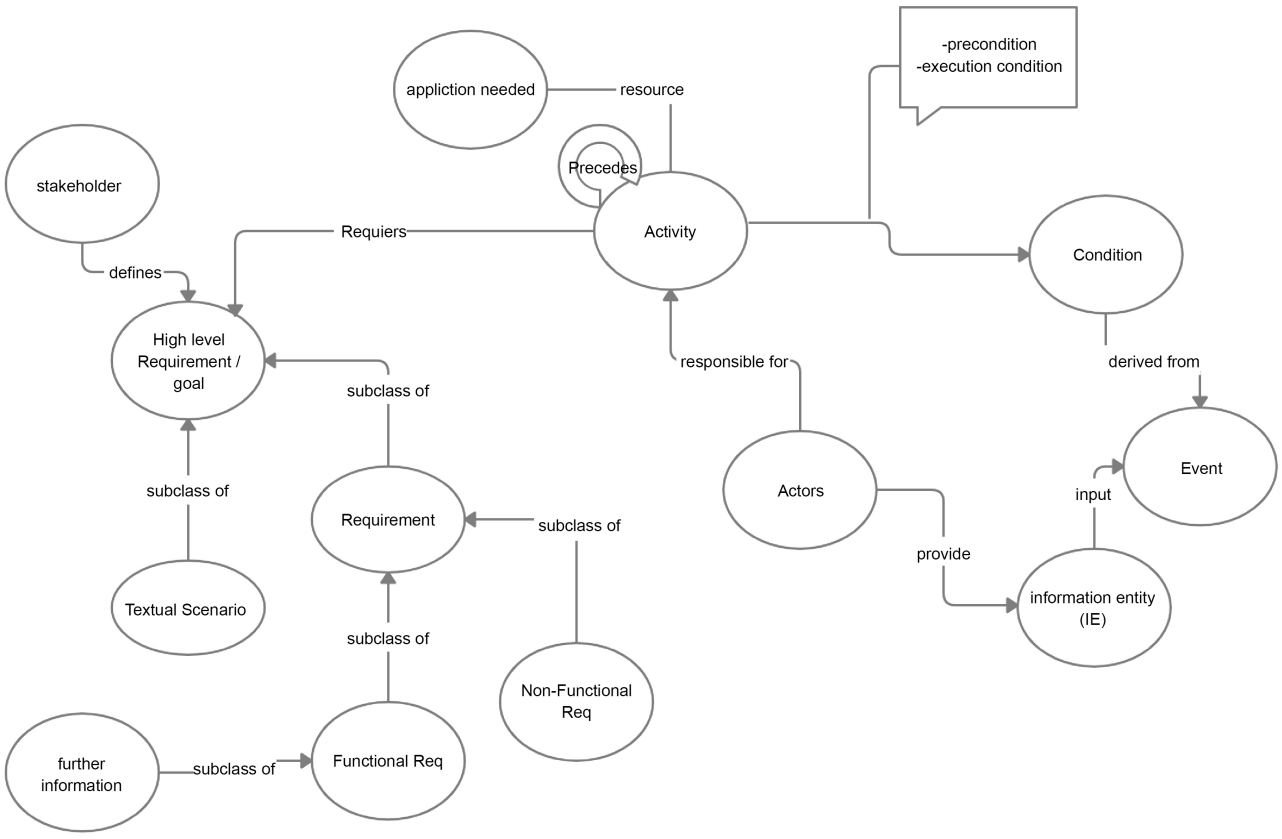

Figure 6. Visualization of Ontology Core.

Sorts of requirements, their descriptions, and the test phrases that correspond to them. Although most of the needs in the analysis fell into a single category, it is important to note that a requirement might fall into many categories and be linked to multiple test expressions.

Class-related (Type of requirement)

- Equivalence: Similarity in function between two classes. X Equivalent To Y
- Subsumption: A (super)class's definition is defined by the relationship it has with its subclasses. The two categories are ineligible for inclusion in this subsumption. X SubClassOf Y.

Property-related

- Property between two concepts: Clarification of a relational quality between ideas P Domain A, P Range B
- Symmetry: a property must have an equal and opposite counterpart, or be symmetric.
- Intersection: Cardinality-based definition of a set of concepts that overlap A SubClassOf P min/max/exactly

Individual related

- Definition of an individual: Instance definition for a certain type s type S

5. Discussion (Evaluation and Case Study)

In order to evaluate our approach, we applied a smart house system which is controlling the house which gives the ability to control the house without making a huge amount of effort.

5.1. System Requirements

- Hardware Requirement:

- lights, motors, smoke sensors, motion sensors, cameras, power resources, wired cables, Bluetooth, logic board, capacitors, and microcontroller.

- Functional Requirement:

- The System allows the owner to control the air system, Doors, and lights system as shown in Figure 7.
- The System will notify the owner when the bill rings.
- The system will give the user some choice if he would like to receive a guest or not.
- The system will allow users to set a specific time to turn on any device according to time.
- The System will send Turn Alerts When Doing Something Strange Like; Fires/Theft problems

- Non-Functional Requirements:

The System should be:

- High Performance when Home Alert (Must Be detected within 1 second).
- The system must work fine with multiple users at home at any time (availability).

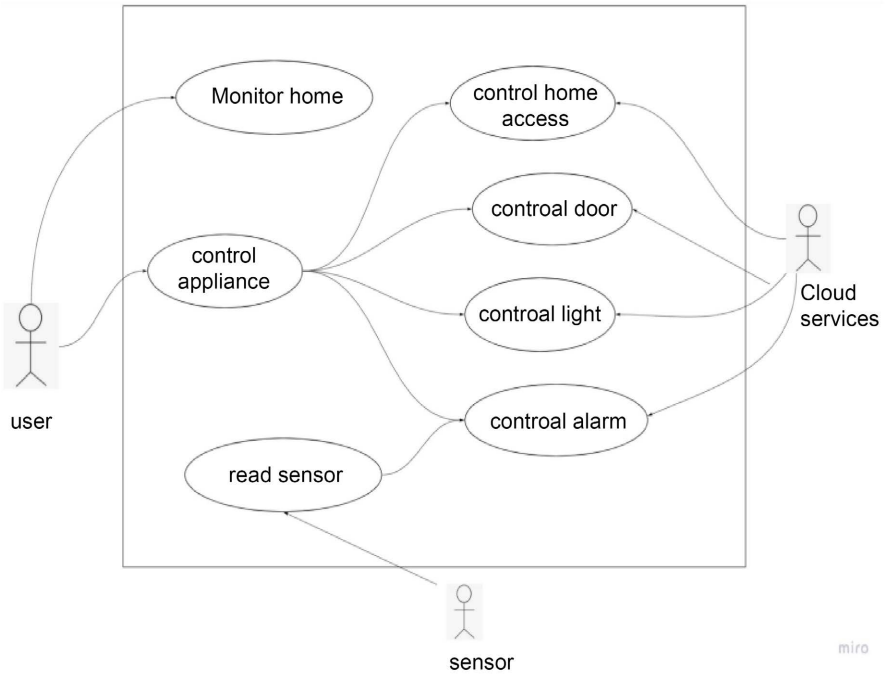

Figure 7. Smart Home use case.

5.2. Requirements Specification Document Ontology

Acquisition structures for domain information, the employment of ontologies for describing the structure of needs specification documents cut back the lean needs' specifications see Figure 8.

5.3. Matching and Merging

In order to get the Requirements Ontology, we need to connect the concepts of different documents to gather. So, there are two options: matching and merging [14].

Ontology Matching is the process of finding semantic equivalence between concepts from different ontologies [34]. The merging step combines two concepts of semantic equivalence from different ontologies and groups them into one ontology [35].

5.4. Approach Steps

Step 1: Goal Identification

The user can control the home remotely

Task 1.1: Identify Goal Task

The user can take control of rooms like; lights turn on or off, opening doors and cameras, and some of the sensors responsible for motion, smoke, and fires to make a secure home.

Task 2.1: Assign Author to Goal

The application gives users three basic functions: "Doors, Lights, and Camera". Also, the home has a motion sensor to detect an illegal access to a home by

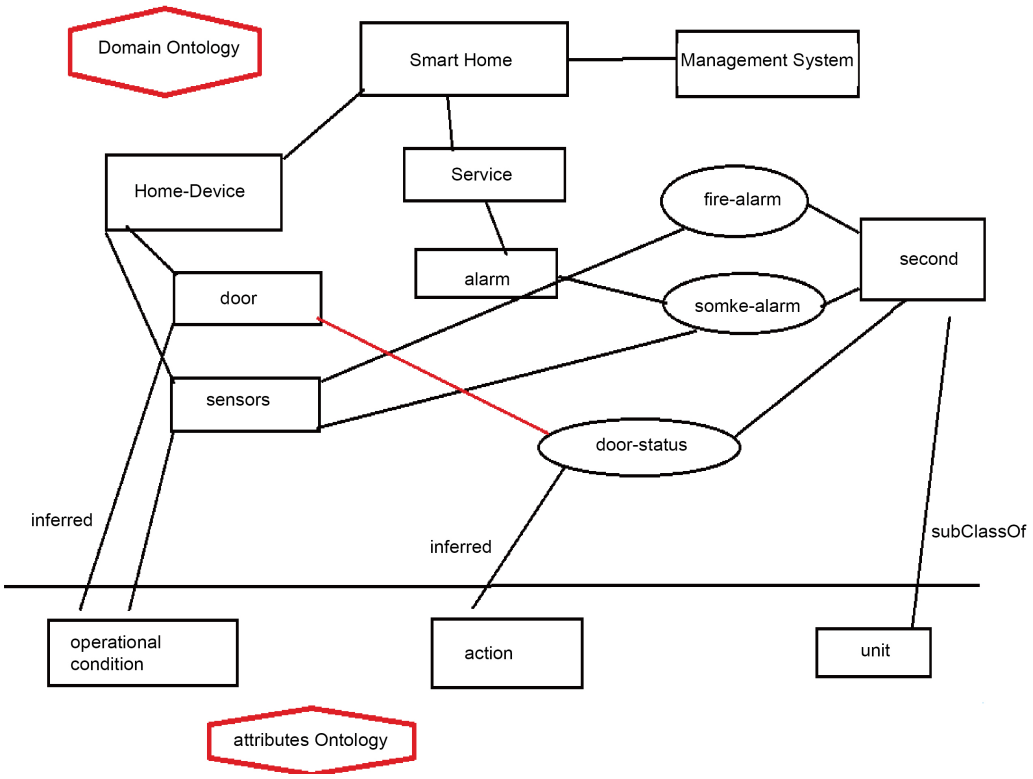

Figure 8. Domain ontology and attributes.

covering a wide area depending on the home area. The same with the fire sensor; it works if it's got smoking in a home and then releases alerts.

Task 1.3: Refine Goal

Check the inputs manually and see if they are right.

Step 2: Requirements Identification

Task 1.2: Identify functional requirements with non -functional requirements

- Open/Close Doors.
- Turn On/Off Lights.
- Turn Alerts When Doing Something Strange Like Fires/Theft problems.

Step 3: Extra information Completion

The system should be smart enough to react to all user input and requests. It should generate other types of security sides like fries and home theft, which notify when doors open and show who is in the door by a simple interface application.

Step 4: Checking

Check both answers, then apply the merging approach in order to get one ontology and SRS (Export the SRS).

This evaluation has shown that the method can deal with a set of requirements from a real-world problem and classify where these requirements are inconsistent or incomplete.

Far more difficult than locating missing data is determining when and where there is inconsistency and offering advice for how to fix it. We need to take into

account several factors for a consistency rule, in contrast to the completeness validation.

In this light, it is crucial that we check for continuity in the setup of the requirements. The requirements engineer selects a subset of needs, and then we construct the requirements configuration by including all of those needs.

As part of this prototype, we include the right features to prompt the requirements engineer to choose the most important criteria and save them as a set of unique objects.

There are three distinct dialects of OWL, each tailored to a different group of developers and end users: OWL Lite, OWL DL, and OWL Full. [36]. However, the Requirements Ontology has been labelled as OWL DL, meaning it guarantees the computational completeness and decidability (all calculations will finish in a limited time) of reasoning systems. Many different reasoners are now available, each with its own set of advantages and disadvantages in areas like reasoning speed, rule support, expressivity, and more [37].

6. Conclusions and Future Works

Today, it is widely accepted that projects will fail if the software requirements specification is absent, contradictory, or conflicting. Therefore, requirements engineering works to maintain consistent, up-to-date requirements across a project's life cycle. To achieve this, we provide a domain ontology-based method for analyzing software requirements.

The Requirements Ontology and Requirements Metamodel, which have been established, serve as the foundation for validation and measurement assistance. It enables requirements analysts to look through a requirements specification according to the application domain's semantics.

This needs ontology considers the conceptualization of requirements knowledge, made possible by ontologies and is suitable for goal-oriented requirements engineering. Requirements Ontology is used as a prototype to demonstrate our technique. By hiding the ontology from the requirements engineer and allowing the validation of the information contained therein, ontology considers the specifics of the requirements definition.

The Requirements Ontology has been exposed to be effective at capturing the knowledge of a software requirements specification's requirements, and it is practical to use ontologies by requirements engineering tools to highlight inconsistencies, incompleteness, and quality flaws during phases of requirement modeling. We utilized a smart home system to evaluate the idea. The focus of future research in this field is on the requirements traceability's direction. Additionally, as future studies should concentrate on the effectiveness and quality of ontology construction, it is necessary to investigate the methodical steps involved.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Joseph, E. (2017) Survey on Requirement Elicitation Techniques: Its Effect on Software Engineering. *International Journal of Innovative Research in Computer and Communication Engineering*, **5**, 9201-9215.

[2] Wiegers, K.E. (2013) Software Requirements. 3rd Edition, Microsoft Press, Redmond.

[3] Van Lamsweerde, A., Darimont, R. and Letier, E. (1998) Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering*, **24**, 908-926. https://doi.org/10.1109/32.730542

[4] Amyot, D. (2003) Introduction to the User Requirements Notation: Learning by Example. *Computer Networks*, **42**, 285-301. https://www.sciencedirect.com/science/article/pii/S1389128603002445

[5] Leffingwell, D. and Widrig, D. (2003) Managing Software Requirements—A User Case Approach. 2nd Edition, Addison-Wesley, Boston.

[6] Loucopoulos, P. and Karakostas, V. (1995) System Requirements Engineering. McGraw Hill, London.

[7] Gavrilova, T. and Andreeva, T. (2012) Knowledge Elicitation Techniques in a Knowledge Management Context. *Journal of Knowledge Management*, **16**, 523-537. https://doi.org/10.1108/13673271211246112

[8] Bourque, P. and Fairley, R.E. (2014) Guide to the Software Engineering Body of Knowledge (SWEBOK (R)). Version 3.0. IEEE Computer Society Press, Washington DC.

[9] Davis, A.M. (1992) Operational Prototyping: A New Development Approach. *IEEE Software*, **9**, 70-78. https://doi.org/10.1109/52.156899

[10] Sajjad, U. and Hanif, M. (2010) Issues and Challenges of Requirement Elicitation in Large Web Projects. School of Computing, Blekinge Institute of Technology, Ronneby.

[11] Mohd Kasirun, Z. (2005) A Survey on the Requirements Elicitation Practices among Courseware Developers. *Malaysian Journal of Computer Science*, **18**, 70-77.

[12] Zhu, X. and Jin, Z. (2005) Inconsistency Measurement of Software Requirements Specifications: An Ontology-Based Approach. 10*th IEEE International Conference on Engineering of Complex Computer Systems* (*ICECCS*'05), Shanghai 16-20 June 2005, 402-410. https://doi.org/10.1109/ICECCS.2005.55

[13] Loucopoulos, P. and Katsouli, E. (1992) Modelling Business Rules in an Office Environment. *SIGOIS Bulletin*, **13**, 28-37. https://doi.org/10.1145/134376.134384

[14] Taye, M.M. (2009) Ontology Alignment Mechanisms for Improving Web-Based Searching. Ph.D. Thesis, De Montfort University, United Kingdom, England.

[15] Taye, M.M. (2010) State-of-the-Art: Ontology Matching Techniques and Ontology Mapping Systems. *The International Journal of ACM Jordan*, **1**, 68.

[16] Aßmann, U., Zschaler, S. and Wagner, G. (2006) Ontologies, Metamodels, and the Model-Driven Paradigm. In: Calero, C., Ruiz, F. and Piattini, M., Eds., *Ontologies for Software Engineering and Software Technology*, Springer, Berlin, 249-273. https://doi.org/10.1007/3-540-34518-3_9

[17] [IEEE-830] Institute of Electrical and Electronics Engineers (1998) IEEE Recommended Practice for Software Requirements Specifications. IEEE Std 830-1998, Institute of Electrical and Electronics Engineers, New York.

[18] Alsanad, A.A., Chikh, A. and Mirza, A. (2019) A Domain Ontology for Software

Requirements Change Management in Global Software Development Environment. *IEEE Access*, **7**, 49352-49361. https://doi.org/10.1109/ACCESS.2019.2909839

[19] Dermeval, D., Vilela, J., Bittencourt, I.I., *et al.* (2016) Applications of Ontologies in Requirements Engineering: A Systematic Review of the Literature. *Requirements Engineering*, **21**, 405-437. https://doi.org/10.1007/s00766-015-0222-6

[20] Breitman, K.K. and Prado Leite, J.C.S. (2003) Ontology as a Requirements Engineering Product. *International Requirements Engineering Conference*, Monterey Bay, CA, 12 September 2003, 309-319.

[21] Zowghi, D. and Coulin, C. (2005) Requirements Elicitation: A Survey of Technique, Approaches and Tools. In: Aurum, A. and Wohlin, C., Eds., *Engineering and Managing Software Requirements*, Springer, Berlin, 19-46.

[22] Siegemund, K. (2014) Contributions to Ontology-Driven Requirements Engineering. Dissertation, Technischen Universitat Dresden, Dresden, 236 p.

[23] Thamrongchote, C. and Vatanawood, W. (2016) Business Process Ontology for Defining User Story. 2016 *IEEE/ACIS* 15*th International Conference on Computer and Information Science* (*ICIS*), Okayama, 26-29 June 2016, 1-4. https://doi.org/10.1109/ICIS.2016.7550829

[24] Bhatia, M.P.S., Kumar, A. and Beniwal, R. (2015) Ontologies for Software Engineering: Past, Present and Future. *Indian Journal of Science and Technology*, **9**, 1-16. https://doi.org/10.17485/ijst/2016/v9i9/71384

[25] Murugesh, S. and Jaya, A. (2015) Construction of Ontology for Software Requirements Elicitation. *Indian Journal of Science and Technology*, **8**, 1-5. https://doi.org/10.17485/ijst/2015/v8i29/86271

[26] Robeer, M., Lucassen, G., *et al.* (2016) Automated Extraction of Conceptual Models from User Stories via NLP. 24*th International Requirements Engineering* (*RE*) *Conference*, Beijing, 12-16 September 2016, 196-205. https://doi.org/10.1109/RE.2016.40

[27] Sitthithanasakul, S. and Choosri, N. (2016) Using Ontology to Enhance Requirement Engineering in Agile Software Process. 2016 10*th International Conference on Software*, *Knowledge*, *Information Management & Applications* (*SKIMA*), Chengdu, 15-17 December 2016, 181-186. https://doi.org/10.1109/SKIMA.2016.7916218

[28] Avdeenko, T.V. and Pustovalova, N.V. (2015) The Ontology-Based Approach to Support the Completeness and Consistency of the Requirements Specification. *International Siberian Conference on Control and Communications* (*SIBCON*2015), Vol. 9, 1-4. https://doi.org/10.1109/SIBCON.2015.7147184

[29] Verhodubs, O. and Grundspenkis, J. (2013) Ontology Merging in the Context of Semantic Web Expert System. 4*th Conference*, *KESW* 2013, St. Petersburg, 7-9 October 2013, 191-201. https://doi.org/10.1007/978-3-642-41360-5_15

[30] Castañeda, V., Ballejos, L., Caliusco, M.L. and Galli, M.R. (2010) The Use of Ontologies in Requirements Engineering. *Global Journal of Researches in Engineering*, **10**, 2-8.

[31] Siegemund, K., Thomas, E.J., Zhao, Y., Pan, J. and Assmann, U. (2011) Towards Ontology-Driven Requirements Engineering. 10*th International Semantic Web Conference* (*ISWC*), Bonn, Bonn, 1-6.

[32] Awal, A., *et al.* (2018) Ontology Development for the Domain of Software Requirement Elicitation Technique. *International Journal of Engineering Research & Technology* (*IJERT*), **7**, 334-338. https://doi.org/10.17577/IJERTV7IS040237

[33] Hull, E., Jackson, K. and Dick, J. (2005) Requirements Engineering. Springer, Ber-

lin.

[34] Wikipedia. Ontology Merging. http://en.wikipedia.org/wiki/Ontology_merging

[35] Chen, X., Yin, B. and Jin, Z. (2011) Ontology-Guided Requirements Modeling Based on Problem Frames Approach. *Journal of Software*, **22**, 177-194. https://doi.org/10.3724/SP.J.1001.2011.03755

[36] Hitzler, P., Krötzsch, M., Parsia, B., *et al.* (2012) OWL 2 Web Ontology Language Primer. Second Edition.

[37] Gruber, T.R. (1993) A Translation Approach to Portable Ontologies. *Knowledge Acquisition*, **5**, 199-220. https://doi.org/10.1006/knac.1993.1008