HOSTED BY

ELSEVIER

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: www.elsevier.com/locate/jestch

Review

Linking software requirements and conceptual models: A systematic literature review

Fatma Bozyiğit [a,*], Özlem Aktaş [b], Deniz Kılınç [a]

[a] İzmir Bakırçay University, Department of Computer Engineering, Menemen, İzmir, Turkey
[b] Dokuz Eylül University, Department of Computer Engineering, Tınaztepe, Buca, Turkey

A R T I C L E I N F O

Article history:
Received 20 January 2020
Revised 12 September 2020
Accepted 17 November 2020
Available online 6 January 2021

Keywords:
Software requirements
Concept identification
Conceptual model
Systematic literature review

A B S T R A C T

Identification of stakeholder needs and documentation of software requirements are the critical steps to launch a software project. Natural language requirements serve as an agreement among the project stakeholders and they must be transformed into easy-to-understand conceptual models to avoid communication problems. Although conceptual models are mostly created manually with human involvement from the software team, it is seen in recent times that there is a significant increase in studies that automatically generate conceptual models from software requirements. In this study, a Systematic Literature Review (SLR) based on the search of forty-four primary studies (published between 1996 and 2020), which automatically transform software requirements into conceptual models, is reported. These studies are evaluated regarding their approaches, functionalities, dataset used, evaluation methods, generated model types, and languages supported. Finally, several improvable points in the current approaches are highlighted and suggestions are provided as further works.

Contents

* Corresponding author.
E-mail addresses: fatma.bozyigit@bakircay.edu.tr (F. Bozyiğit), ozlem@cs.deu.edu.tr (Ö. Aktaş), deniz.kilinc@bakircay.edu.tr (D. Kılınç).

Peer review under responsibility of Karabuk University.

https://doi.org/10.1016/j.jestch.2020.11.006
2215-0986/© 2020 Karabuk University. Publishing services by Elsevier B.V.

1. Problem definition

Software requirements are the descriptions of the necessary functions to be designed and developed in a given software system [1]. If a requirement includes vague declarations, it cannot be clearly comprehended by the stakeholders of a project, so communication problems may arise between customers and software team [2]. These problems may also cause delays in the next phases of Software Development Life Cycle (SDLC) and increase the cost of a software project. Therefore, it is critical that these requirements are needed to be transformed into easy to understand conceptual models [3].

Generally, systems analysts manually design the conceptual models of software systems after defining all user needs. They make effort to extract entities with properties and relationships. In recent years, existence of a significant increase in the number of studies that automatically transform software requirements into the conceptual models (e.g., [4–6] and so on). In this study, we systematically analyzed the technical aspects and limitations of the automatic concept identification studies, following Systematic Literature Review (SLR) guidelines [7]. We evaluated the forty-four selected studies considering the used methods, outputs, experimented datasets, and evaluation methods. We specified the constraints and concerns in this domain regarding detailed evaluation results. We also concluded the critical issues for designing complete and consistent conceptual models.

1.1. Research questions

Specification of the research questions is the most important phase in the SLR methodology, since research questions identify the scope and the objective of a review study [8].

In our study, five research questions were determined to analyze and evaluate different approaches used in automatic concept identification studies as follows:

- Research Question 1 (RQ_1): What languages are analyzed to transform software requirements into conceptual models?
- Research Question 2 (RQ_2): What methods are used for transforming requirements into a conceptual model?
- Research Question 3 (RQ_3): What kind of conceptual models are generated by the reviewed studies?
- Research Question 4 (RQ_4): What are the details of the datasets used in the reviewed studies?
- Research Question 5 (RQ_5): What are the evaluation methods used for accuracy assessment in the reviewed studies?

RQ_1 investigates what languages (requirements written in) are supported by the reviewed studies. RQ_2 aims to identify the approaches of the studies in the SLR. According to answers to this question, proposed methods in the reviewed studies are observed. RQ_3 identifies the outputs of the studies. That is, types of extracted conceptual models (UML diagrams, program code, etc.) by analyzing requirements are determined. This question is divided into the sub-questions as follows; What kinds of UML diagrams are created in the studies, if they generate UML model? Is there any missing design element in the generated conceptual model? For example, is there any relationship between two classes uncovered? RQ_4 investigates the datasets used in the reviewed studies. This question is divided into the sub-questions as follows; Do reviewed studies have a comprehensive dataset that includes many requirements documents? How many requirements are included in employed dataset? Are there any publicly available datasets presented by the researchers of the reviewed studies? RQ_5 gives information about evaluation methods in the studies. It investigates whether MCDM techniques are performed and view of experts are included in the evaluation phase.

1.2. Scope

This paper reports a review of approaches transforming software requirements into conceptual models. In this section, we refined the scope of the SLR by defining fundamental concepts of the related domain such as requirement document and conceptual model. A requirement document is defined in [9] as "a statement that identifies a necessary attribute, capability, characteristic, or quality of a system". There are many ways to document requirements. One common way is to use textual descriptions only. Other ways to document requirements include use cases, formal specifications, user stories, and customized document templates. In our study, we limited our scope to requirements documented using textual descriptions and use cases.

A conceptual model is a description of what a system is required to do functionally and aims to be less ambiguous than textual requirements. This model can be represented in various forms, such as UML diagrams, Entity Relationship Models (ERM), and Business Models (BM). In a typical object-oriented software development process, the analysis model is generally represented as a UML model containing various diagrams. In our study, we limited our scope to conceptual model in the form of UML diagrams.

1.3. Research motivation

Considering the researches involved, it is seen that most of the existing concept identification approaches are designed to analyze documents written in English (e.g., [10–12], and so on). Due to upsurge of World Wide Web usage and intercultural communication through technology, multilingualism is revealed as a raising issue in digital platforms [8]. Therefore, concept identification studies must support to analyze software requirements in different languages beside English. Moreover, existing studies do not provide a scalable solution for generating conceptual models. For example, transformation process is commonly utilized to create UML class or use-case diagrams as the conceptual model. However, there are fourteen different types of UML diagrams for modeling software designs. The other considerable concern is that existing approaches are unable to extract some design elements like relationships between the classes. In addition to these, nearly all datasets in the current studies (except [5]), are either not accessible or do not contain necessary number of requirements. Although there are substantial amounts of studies conducted on concept identification process, there is a lack of a large-scale benchmark dataset to be used in similar researches.

In this study, we highlighted required aspects which are not provided by the reviewed studies. We believe that this SLR will help in developing more efficient and user-friendly models transforming software requirements (written in multiple languages) into different kind of conceptual models.

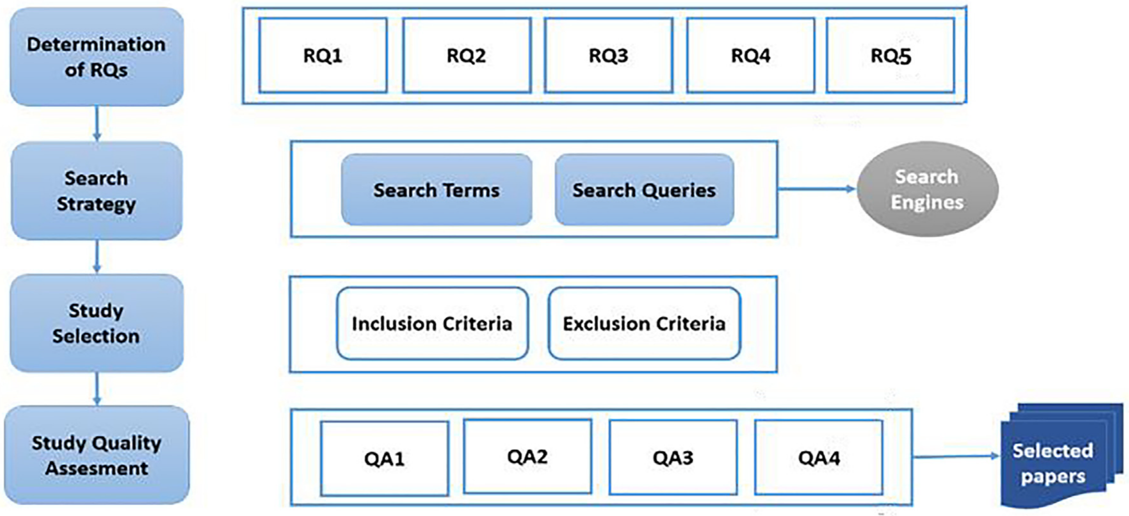

Fig. 1. Search strategy in the literature.

1.4. Structure

The structure of this paper is organized as follows: Section 2 gives information about the research methodology and it explains the study selection process in a detailed manner. Section 3 presents evaluation results of the reviewed studies regarding research questions. Section 4 briefly describes threats to validity of research findings. Section 5 concludes the paper and includes suggestions for further studies.

2. Research methodology

SLR is a review that uses systematic approaches to identify, determine, and evaluate relevant researches. It presents the analysis of collected data from the current studies by addressing well-defined research questions. Since SLR examines related studies in a more detailed and systematic way, we developed a research methodology in view of the SLR guidelines presented by Kitchenham [7].

The general architecture and functional blocks of the systematic review are shown in Fig. 1. First five research questions explained in Section 1.1 are determined to build the framework of SLR. Secondly, a search strategy that includes paper selection procedures (determining search terms based on research questions and creating search queries) is developed. In the study selection phase, we used well-known digital databases (Science Direct, IEEE Explore, ACM, and Springer) and Google Scholar academic search engine to find high quality papers. The search process was completed on the journal papers and the conference proceedings from 1996 to 2020. Then, the results of the search process were filtered based on inclusion and exclusion criteria and the papers which do not match these criteria are eliminated. Finally, four quality assessment (QA) criteria were determined and then the researches to be evaluated under the SLR were obtained.

2.1. Keywords and query terms

In the search process step, first the keywords related to the research topic are determined. To create search queries, the search terms are specified and categorized based on research questions using the PICOC (Population, Intervention, Comparison, Outcome, Context) proposed by Brereton et al. [8].

Population are the criteria that specify the domain of transforming requirements into conceptual model such as “requirements transformation”, “requirements analysis”, and “generating conceptual model”. Intervention are the keywords that indicate approaches used for transforming requirements into conceptual model such as “Natural Language Processing” (or “NLP”) and “rule-based model”. Comparison are the keywords about the position of the studies analyzing requirements without using automatic concept identification methods. Outcome are the keywords about generated conceptual models from the software requirements such as “UML diagrams”, “ontology model”, and “source code”. Context are the keywords about context in concept identification studies.

We did not use comparison criteria to formulate search strings. We used context criteria as exclusion criteria to eliminate the irrelevant papers (explained in Section 2.2.). The search strings were derived from specified search terms including population, intervention and outcome as shown in Table 1. The search strings were adjusted to advanced methods of source searching in each digital library selected. For example while we created a search string as (TITLE-ABS-KEY((“software requirements” or “requirements” or “user needs”) AND (analyze OR transformation or generation) AND (“conceptual model” or “analysis model”) AND LIMIT-TO (PUBYEAR,1996))) for ScienceDirect, we enter the string as

Table 1
Created search queries using search terms.

Search queries
(“software requirements” OR requirements OR “user needs”) AND (analyze OR transformation OR generation) AND (“conceptual model” OR “analysis model”)
(“software requirements” OR requirements OR “user needs”) AND (analyze OR transformation OR generation) AND (“UML diagrams” OR “class diagrams” OR “sequence diagram” OR “activity diagram” OR “use-case diagrams” OR “object diagram” OR “source code” OR “program code” OR “validation model” OR “ER model“)
(generating OR extracting OR creating) AND (“conceptual model” OR “UML diagrams” OR “source code” OR “program code” OR “ER diagram”) AND (“Natural Language Processing” OR NLP OR “rule-based model” OR ontology)

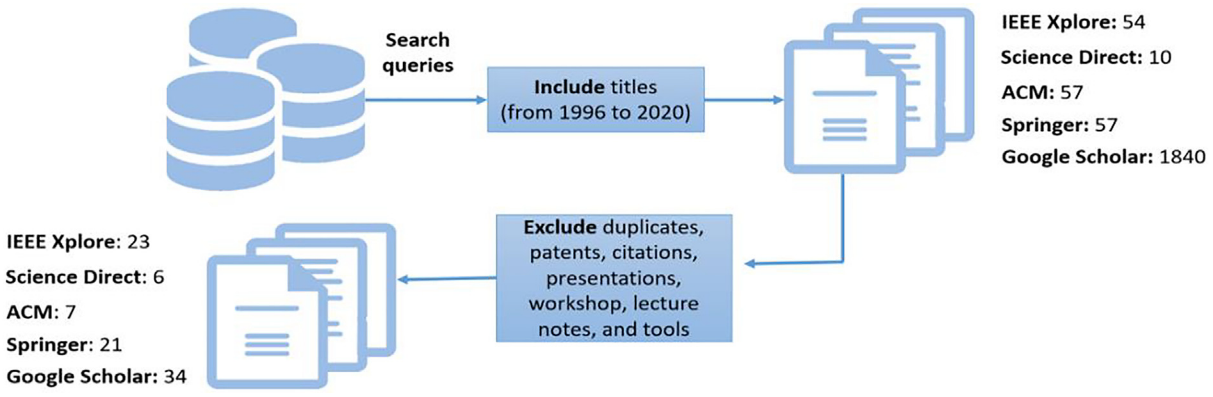

Fig. 2. Number of included articles during the study selection process.

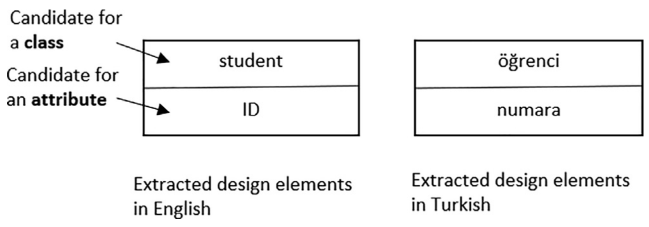
Candidate for a class
Candidate for an attribute
student
ID
öğrenci
numara
Extracted design elements in English
Extracted design elements in Turkish

Fig. 3. Illustration of the data in Table 3.

Table 2
Number of articles after implementation of quality assessment criteria.

Platform	Number of papers Conference Paper	Article
IEEE Xplore	9	3
ScienceDirect	2	8
Springer	5	3
Google Scholar	4	10
Total	20	24

(“software requirements” or “requirements” or “user needs”) AND (analyze OR transformation or generation) AND (“conceptual model” or “analysis model”) AND (year>=1996 AND year<=2020) for same query.

2.2. Identification of selection criteria

Fig. 2 gives information about the number of included articles during the study selection processes. By using our search queries, a total of 2018 papers were determined as input for the selection process. Selection of appropriate studies was completed by using the specified inclusion and exclusion criteria. Accordingly, first, 2008 studies providing inclusion criteria were selected. Then, 1917 of the selected studies that include at least one of the exclusion criteria were eliminated. Finally, 91 studies which are related to our research scope were identified.

Inclusion criteria are listed as; studies in journals and conferences in the field of computer science, software engineering, information systems, and natural language processing were selected. Journal articles and conference proceedings were filtered with respect to publication dates between 1996 and 2020.

Exclusion criteria are listed as; patents, presentations, workshops papers, technical notes, informal papers, and tools not based on scientific study were eliminated. Duplicate papers of same study are neglected. Paper that does not mention conceptual model in title and abstract content were not included in SLR content.

2.3. Performing and conducting the search

The quality assessment process is used for interpretation of findings and determining the power of detailed investigations [7]. After study selection process illustrated with Fig. 3, the quality of each selected study was evaluated according to the four questions such as: Are the context of the study clearly presented? Are the objectives of the study reasonably described? Are the applied methods detailed explained? Is the output of study visualized by a concept identification model?

All researchers in this SLR labeled the primary studies as “include”, “exclude”, or “uncertain” tags for each criterion. It was decided on whether to include a study into the review considering the number of tags. If the number of “include” tags was greater than others, the study was determined to evaluate in the SLR scope.

Initially, 2018 papers were extracted regarding the results of search queries on digital databases and academic search engine. Then, filtering was performed regarding inclusion and exclusion criteria and so 91 papers are selected in this step. At the last step, it was determined under specified quality assessment criteria whether the selected studies are appropriate to be evaluated in this work. Finally, forty-four papers were selected to be evaluated in this study (see Table 2).

3. Analysis and results

3.1. RQ$_1$ What languages are analyzed to transform software requirements into conceptual models?

As the internet resources become more easily accessible, it is necessary to design efficient approaches for information systems across languages [13]. Providing a generic approach to multilingualism in requirements analysis enables broad coverage and accessibility.

According to review of literature, it is seen that most of the tools analyze requirements in English since it is the one of the most widely used languages in the world. Furthermore, simple morphol-

Table 3
Example of common transformation rule.

Language	Sentence
Turkish	Her öğrenci kendine mahsus numaraya sahiptir. (Each student has a unique ID.) Nouns: öğrenci (student), numara (ID)
English	Each student has a unique ID. Nouns: student, ID.

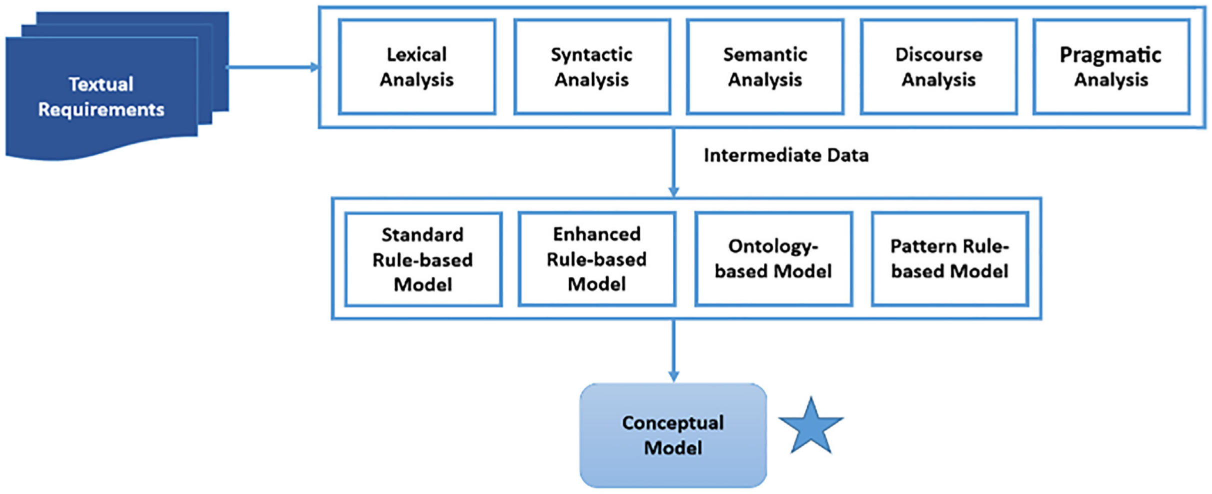

Fig. 4. General framework of approaches in the reviewed concept identification studies.

ogy of English compared to other languages makes it easier to obtain more accurate analysis results. There are a few studies supporting other languages such as [5], [14], [15], and [16]. [5] performs on Turkish requirements, [14] analyses Spanish requirements and [15] works on requirements written in German. [16] examines in software requirements written in Thai.

Considering the language limitations in the current studies, we recommend creating a common ruleset which supports the analysis of requirements written in any language to promote multilingual practices. In our study, it is stated that some transformation rules can be applied independently of language in concept identification studies. For example,

- Each noun in the document is a candidate for class and attribute design elements in an object-oriented based conceptual model.

Assume that we applied this rule for a sentence written in both Turkish and English as in Table 3.

As it can be seen from the example given in Table 3 and Fig. 3, the same design elements can be extracted when a general rule is applied to different translations of a clause. Consequently, the idea of creating a ruleset for analyzing the textual requirements independently of the language provides motivation for our future works.

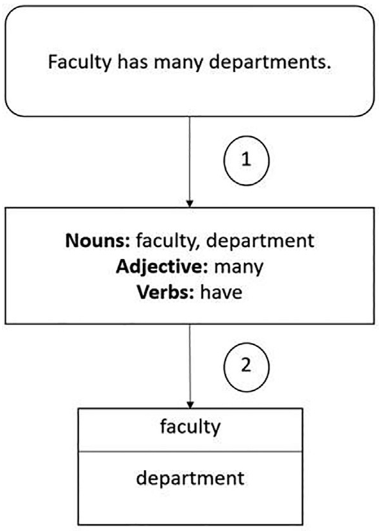

Fig. 5. An example of executing transformation rules.

3.2. RQ$_2$ what methods are used for transforming requirements into conceptual model?

The approaches in the studies transforming requirements into conceptual models are examined with this question. It is observed that generally similar approaches are implemented in the concept identification studies. These approaches consist of NLP analysis models and rule-based model as illustrated in Fig. 4. Initially, requirements are pre-processed with NLP analysis models to obtain an intermediate data. Then, the intermediate data are given as input to the rule-based transformation models, and finally conceptual model is generated automatically. There are five NLP analysis models available to preprocess textual requirements: lexical analysis, syntactic analysis, semantic analysis, discourse analysis, and pragmatic analysis [17].

Lexical analysis identifies structure of words and phrases in sentences. It includes many steps, such as tokenization, stemming/lemmatization, part of speech (POS) tagging and so on. Tokenization facilitates extraction of information from text documents by separating words, abbreviations, punctuations, and number groups in the sentence. Stemming or lemmatization derives a base form of a word by reducing all inflectional forms. POS tagging enables identification of words in a sentence according to the linguistic properties such as noun, verb, and adjective [18]. POS tags are used for determining design elements of the conceptual model. For instance, the verbs in requirement text are candidates for methods in design model. Syntactic analysis determines whether the structure of a sentence is correct according to the grammar. Semantic analysis figures out the meaning of linguistic input. Discourse analysis is the process of specifying contextual information in textual data. Pragmatic analysis facilitates normalization of textual data with detecting inconsistencies.

Rule-based models enable identification of classes, attributes, methods, relationships, and other elements in design using structured data obtained with NLP analysis models. There are four types of rule-based models used in the studies: standard transformation model, enhanced transformation model, ontology-based transformation model and pattern-based transformation model. Standard transformation model has many rules established from linguistic patterns and grammatical structures [2]. For example, Fig. 5 illustrates a sentence which is processed using standard transformation

Table 4
Used methods and outputs of the reviewed studies (LexA: Lexical Analysis, SynA: Syntactic Analysis, SemA: Semantic Analysis, PrA: Pragmatic Analysis, DiscourseA: Discourse Analysis, Enh: Enhanced, Ont: Ontology, Stan.: Standard, Pat.: Pattern).

Paper	Requirements	NLP Model	Transformation Model	Output
[2]	Text (English)	LexA, SynA, SemA	Enh. rule-based	Class
[4]	Text (English)	LexA	Pat. rule-based	Class
[5]	Text (Turkish)	LexA, SynA.	Ont., Pat., Enh. rule-based	Class
[6]	Text (English)	LexA, SynA, SemA	Ont rule based	Class
[10]	Text (English)	LexA, SynA	Stan. rule-based	Use-case
[11]	Text (English)	LexA, SynA	Pat. rule-based	Domain model
[12]	Text (English)	None	Pat., Enh. rule-based	Activity
[14]	Use cases (Spanish)	SynA, SemA	Stan. rule-based	Class
[15]	Text (German)	LexA	Enh. rule-based	Class, Use-case
[16]	Text (Thai)	LexA	Stan. rule-based	Class
[20]	Text (English)	LexA, SynA, SemA	Stan. rule-based	Object
[21]	Text (English)	LexA, SynA	Stan. rule-based	Class
[22]	Text (English)	None	Pat. rule-based	Class
[23]	Text (English)	None	Pat. rule-based	Object
[24]	Use cases (English)	None	Pat. rule-based	Class, Sequence, Java, C++, VB
[25]	Text (English)	SemA, DiscourseA	Pat. rule-based	Class, Sequence
[26]	Text (English)	LexA, SynA, SemA	Ont. rule-based	Class
[27]	Use cases (English)	LexA	Enh. rule-based	Class
[28]	Text (English)	LexA, SynA, SemA	Enh. rule-based	Activity
[29]	Text (English)	LexA, SemA	Pat. rule-based	Class, Sequence, Collaboration
[30]	Text (English)	LexA	Enh. rule-based	Class
[31]	Text (English)	LexA, SynA, SemA	Ont., Pat. rule-based	Class
[32]	Text (English)	LexA, SynA, SemA	Enh. rule-based	Class, Sequence, pseudo code
[33]	Domain Ont. (English)	LexA, SemA	Ont. rule-based	Class
[34]	Use cases (English)	SynA, SemA, DiscourseA	Enh. rule-based	Class
[35]	Text (English)	SemA	Enh. rule-based	Class
[36]	Use cases (English)	SynA	Pat. rule-based	Class
[37]	Use cases (English)	LexA, SemA	Ont., Enh. rule-based	Class
[38]	Text (English)	None	Enh. rule-based	Domain model, Use-case
[39]	Text (English)	LexA, SynA, SemA	Enh. rule-based	Object
[40]	Text (English)	LexA, SynA	Enh. rule-based	Class, Java, VB.
[41]	Use cases (English)	LexA	Enh. rule-based	Class
[42]	Text (English)	PrA	Pat. rule-based	Class, Java
[43]	Text (English)	LexA, SemA	Enh. rule-based	Class, Sequence, Java
[44]	Text (English)	LexA, SynA, SemA	Ont. rule-based	Class
[45]	Text (English)	LexA, SynA, SemA	Pat. rule-based	Class
[46]	Text (English)	LexA, SynA	Ont. rule-based	Class
[47]	Text (English)	LexA, SynA, SemA, PrA	Stand. rule-based	Class, Java
[48]	Text (English)	LexA, SemA	Ont. rule-based	Class, Activity
[49]	Text (English)	LexA, SynA	Pat. rule-based	Sequence, Activity
[50]	Text (English)	LexA, SemA	Ont., Enh. rule-based	Class
[51]	Text (English)	LexA	Enh. rule-based	Class, Java, C#
[52]	Text (English)	LexA, SynA	Pat. rule-based	Class
[53]	Text (English)	LexA, SynA	Pat. rule-based	Use-case

rules. First the lexical analysis is applied to the sentence in the rounded rectangle to obtain a list of intermediate data including POS tags (nouns, adjectives, and verbs). Next, the list of intermediate data is processed by applying the transformation rules presented in Bozyiğit's research [5]:

- Nouns in sentences are candidates for class and attribute names.
- Verbs in the sentences are included by the pool of methods' names.
- If verbs such as "have", "include", and "contain" exist in a sentence, the first name is labeled as a class.
- If verbs such as "have", "include", and "contain" exist in a sentence, all the names except the name of the class are the attributes of that class.

As it can be seen, the rules in the first and second bullet points above implies that "faculty" and "department" may be included class and attribute rules. The third and last points make certain of class (faculty) and attribute (department) in the generated diagram.

Enhanced transformation model has a rule set that includes specific rules which are not used in previous studies. For instance, one of these specific rules is "An adjective that qualifies a noun, where the adjective cannot be classified combines with the noun subject to generate compound words" [2]. This rule states whether an adjective characterizes the underlying noun as an attribute or not. Accordingly, an adjective qualifying a noun as quantity, shape, size, color, etc. states an attribute. In case that it adds a meaning or categorize the noun, it states a class. For example, the adjective in the following statement, "exit door", is tagged as class, since it becomes a domain conceptual class. On the other hand, the adjective in the following statement, "grey door", is tagged as attribute (color). Ontology-based transformation model analyses the textual requirements regarding the semantics of the application domain. Pattern-based transformation model incorporates specific pattern properties into a proposed model [19]. Although using this transformation to extract specific design elements from the software requirements can improve the accuracy of the proposed model, building a set of linguistic patterns can be time consuming and challenging task.

Table 4 gives information about the used methods of the studies in the SLR. It is observed that out of total number of forty-four studies, thirty-one benefit from lexical analysis which includes common techniques in the NLP frame such as tokenization, stemming, and POS tagging. Considering the answers of this question, it

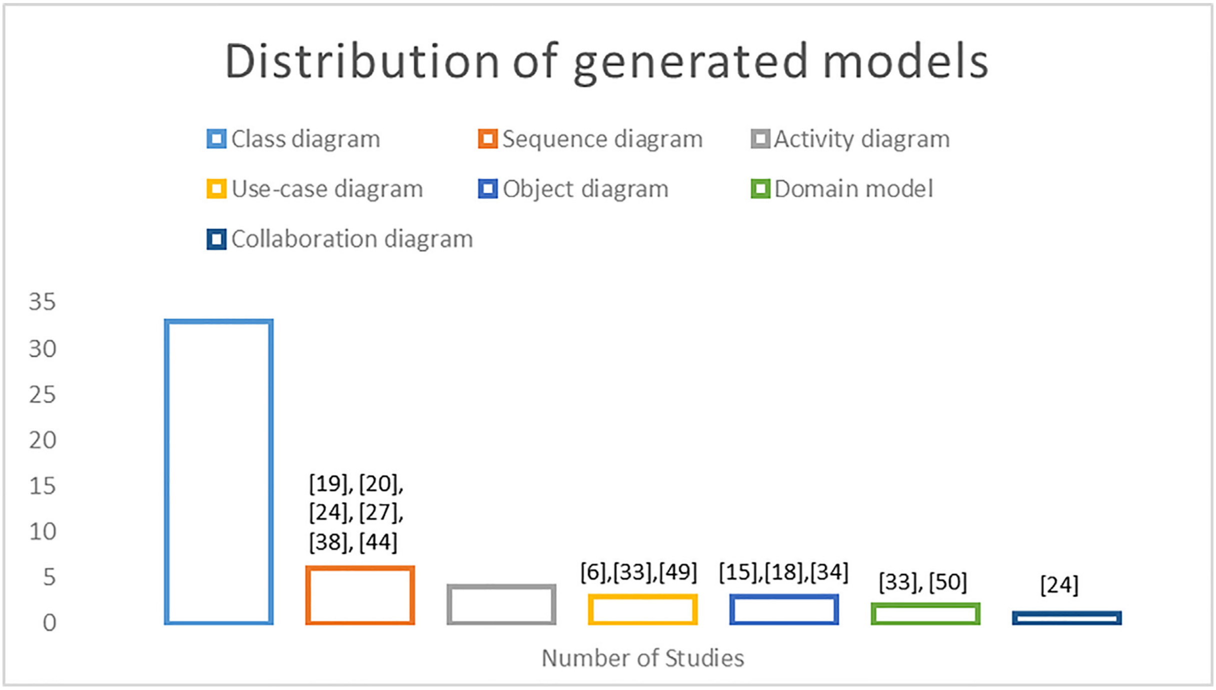

Fig. 6. Distribution of the generated conceptual model.

is seen that only twenty-one of total researches implement semantic analysis. Since semantic analysis is one of the most critical NLP tasks to extract meaning from the statements, researchers who do not utilize this analysis might have skipped necessary design elements in generated models. In fact, semantic analysis of English text is not difficult, since there is the lexical database, WordNet [54], provides the short definitions for the words. However, determining semantics of the text may be difficult for the other languages such as agglutinative ones (Turkish, Finnish, Hungarian, Korean, etc.), because there is no comprehensive dictionary (like WordNet) available for those languages to be used in NLP studies. Thus, we recommend language independent approaches such as word embedding which is widely used to get semantics from textual data. We foresee that performing semantic analysis using word embedding model will increase accuracy of generated models. Also, there are two research [25,34] that benefit from the discourse analysis.

Another factor that appears in the evaluation results is that eighteen out of forty-four studies create a specific set of rules and only ten studies design ontology model in their proposed approach. However, designing a knowledge structure regarding the needs of planned system is sometimes necessary to increase the performance. Considering this, we examined the selected papers within using ontology model and linguistic patterns. We anticipate that creating specific ruleset including detailed linguistic patterns will improve the semantic analysis phase in concept identification studies.

3.3. RQ$_3$ what kind of conceptual models can be generated by the reviewed systems?

A conceptual model is an alternative to describe software design elements. It provides guidelines for the proposed model under operation. This model may be in various forms, such as UML diagrams, Entity Relationship Models (ERM), and Business Model (BM). UML including fourteen diagram types is the most used conceptual model for specifying the structure and the behavior of software model. Therefore, we limit our scope to conceptual model in the form of UML diagrams. Based on a detailed analysis of reviewed studies, we observe that the conceptual models generated by the reviewed studies are generally in the form of a UML class diagram. It is seen that twenty-eight out of forty-four studies generate only class diagram (see Fig. 6) and seven studies [15,24,25,29,32,38,43,48] generate other types of UML diagrams beside class. Additionally, it is realized that of the studies generating UML diagrams, only seven of them [24,32,40,42,43,47,51] also support code generation.

A complete analysis model should describe both the structural and behavioral aspects of the planned software model. Structural diagrams (e.g., class, object, component, deployment) represent the static design elements such as class/objects, attributes, methods, and relationships. Behavioral diagrams (e.g., sequence, state machine, activity) synthesize the design elements that conduct the system's dynamic concepts. As a result, partitioning a full design view into separate diagrams makes the overall design easier to understand and possess the integrity of the modules. According to our view, future studies can extend their study to generate more UML diagrams by using extracted design elements in the analysis phase. To make it clear, we illustrated class and object diagrams for the same software requirements (ATM model [55]) as seen in Fig. 7. As it can be seen in the figure, nearly all design elements in both class and object diagrams are the same.

The main objective of the studies that generate UML diagram is to specify classes, attributes, methods, and relationships in object-oriented design. When evaluating the studies generating UML diagrams, it is seen that all of them are successful in determining the classes and their respective elements (attributes and methods). However, the major point to be considered in UML diagrams is to determine the relationships between classes (generalization, aggregation, composition, association, etc.). When the related studies are examined, it is concluded that most of them have limitations in the specification of relationship types. It is realized that only twenty of the reviewed studies discover relationships between the classes. In addition, only nine out of these twenty

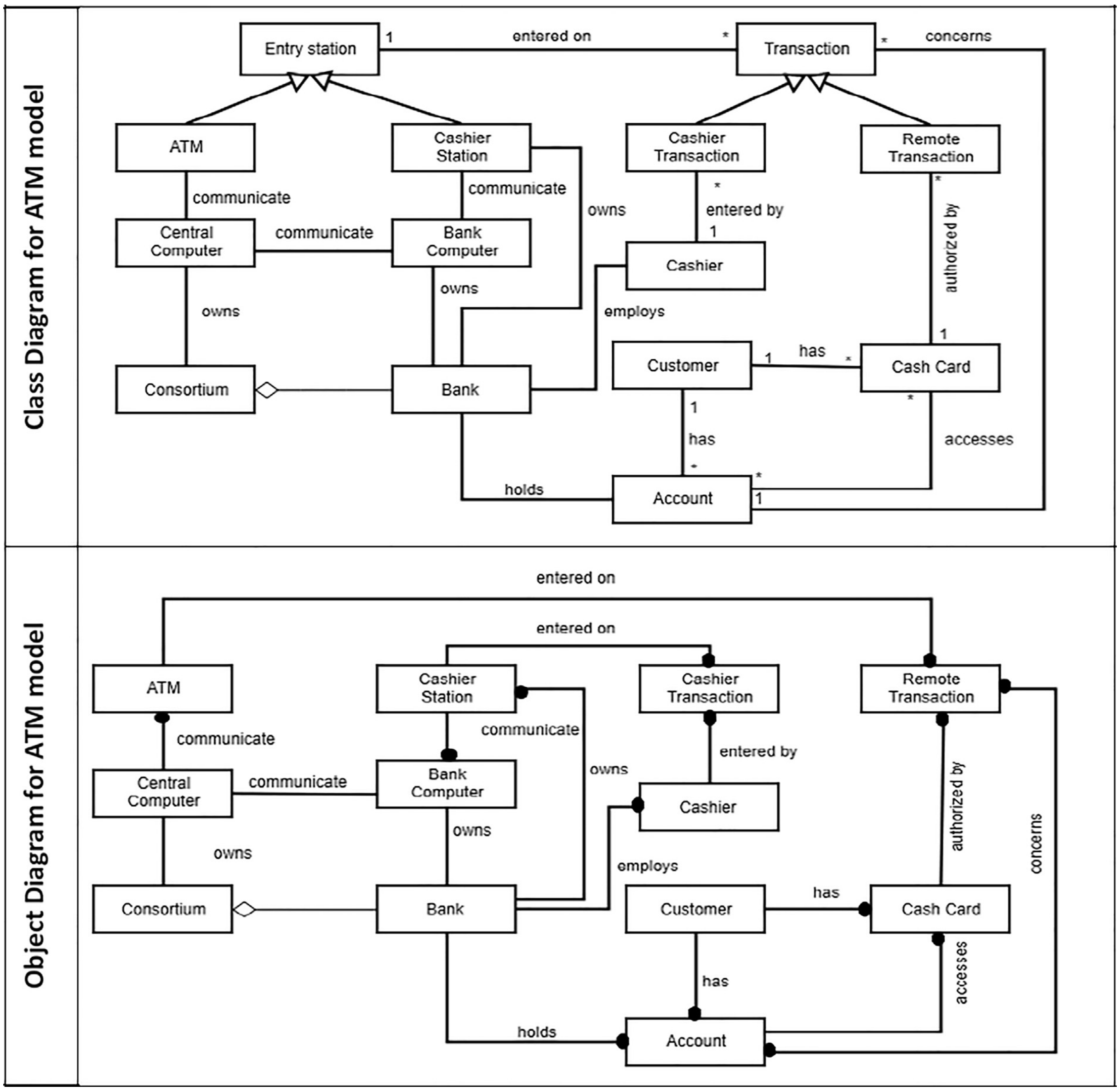

Fig. 7. Class and Object diagrams for ATM problem statement [55].

studies [5,6,23,25,27,30,38,43,44] detect many to many relationship determinations. Table 5 gives information about the studies that can determine relationships.

3.4. RQ$_4$ What are the details of the datasets used in the reviewed studies?

Dataset is the most important input to evaluate proposed approaches in scientific works. Answers given to RQ$_4$ demonstrate that dataset of current studies have some limitations which are explained as follows: 1) The experimental datasets in the current studies include a few numbers of software requirements. 2) The requirements in the datasets include simple and short sentences.

From the evaluation results, it is seen that the study having the largest dataset is [11] and it includes fifty software requirements. The other study having comprehensive dataset is [5]. The researchers in [5] present a dataset including twenty software requirements (both in Turkish and English). According to our point of view, the experimental results will be more consistent and realistic when performed on more data. Thus, the provided dataset should be extended with more software scenarios and gold annotations so that it could be useful for further studies.

The other issue addressed in this SLR is that datasets in the reviewed studies are not publicly shared. Across the forty-four primary studies, it is seen that two out of total ([5] and [11]) publicly share used dataset. Public sharing of datasets provides researchers working on the same domain opportunity to evaluate their proposed approaches in a more objective manner. Consequently, we hope that the results of our SLR contribute to the dataset sharing initiatives.

3.5. RQ$_5$ What are the evaluation methods used for accuracy assessment in the reviewed studies?

The results of the RQ$_5$ give an overview of the evaluation methods used in the papers selected. We categorized the evaluation methods utilized in the reviewed papers as case study and experimental study. Twenty-one out of the total studies do not report any evaluation results to illustrate the accuracy of their approaches. Sixteen of the total studies validate their approach by performing case studies.

Table 5
Supported relationship types in generated models.

Paper	Relationship type						Many to Many
	Abstraction	Aggregation	Association	Composition	Generalization	Specialization	
[2]	no	yes	yes	yes	yes	no	no
[4]	no	yes	no	no	yes	yes	no
[5]	no	yes	yes	yes	yes	yes	yes
[6]	no	no	yes	no	no	no	yes
[10]	no	no	yes	no	no	no	no
[12]	no	no	yes	no	yes	yes	no
[14]	no	yes	yes	yes	no	no	no
[21]	no	yes	yes	no	yes	no	no
[23]	no	no	yes	no	yes	no	yes
[25]	no	no	yes	no	yes	no	yes
[26]	no	no	no	no	no	yes	no
[27]	no	yes	no	no	yes	no	yes
[30]	no	yes	yes	no	no	no	yes
[31]	no	no	yes	no	yes	no	no
[39]	no	no	yes	no	no	no	yes
[40]	no	yes	yes	no	no	no	no
[41]	no	no	yes	no	no	no	no
[42]	no	yes	yes	yes	yes	no	yes
[43]	no	no	yes	no	yes	no	yes
[44]	no	yes	yes	no	no	no	no
[46]	no	yes	yes	no	yes	no	no
[47]	no	no	yes	no	no	no	no
[50]	no	no	yes	yes	no	no	no
[51]	no	yes	yes	yes	yes	no	no
[52]	no	yes	yes	yes	yes	no	yes

It is also worth noting that there are only five studies [5,16,47,48,52] that conducted experimental studies to measure the effectiveness of concept identification approach.

When the studies in the SLR scope were examined, it is seen that commonly used measurements such as precision, recall and F-measure were used for the evaluation process. Precision is the

Table 6
Details of experimental studies in the reviewed works.

Paper	Case Scenario	Evaluation Results		
		Precision	Recall	F-measure
[2]	ATM problem statement [55]	91.67	91.67	91.67
[4]	Immediate payment service [57]	no	no	no
[5]	Restaurant [5]	92.00	91.00	91.00
[6]	No detailed information provided	87.20 (entities), 62.51 (relationship)	84.46 (entities), 64.03 (relationship)	no
[10]	ATM problem statement [55]	98.00 (actors), 87.00 (use case), 87.00 (relationship)	98.00 (actors), 85.00 (use case), 85.00 (relationship)	no
[11]	User manual of customer relationship management (CRM) software system of Roche Diagnostics GmbH company	no	no	92.00
[12]	Login use case of Amazon web service	no	no	no
[15]	ATM problem statement [55]	no	no	92.25
[16]	Thirteen different case studies in a published object-oriented analysis and design book	0.85 (for classes) 0.77 (for attributes)	0.90 (for classes) 0.78 (for attributes)	no no
[23]	Air traffic control system	no	no	no
[27]	Withdraw cash	no	no	no
[30]	Video rental store	no	no	no
[32]	An industrial case study	no	no	no
[36]	Voter tracking system	no	no	no
[38]	Invoicing order system	no	no	no
[39]	Elevator	no	no	no
[41]	Air traffic control system	82.00	58.00	no
[47]	ATM problem statement [55]	100.00	93.00	no
	Modal window	no	no	no
	Musical store	no	no	no
	Circe	no	no	no
[48]	Monitoring pressure	no	45.83	no
	Steam boiler	no	no	no
	ABC video rental	no	62.17	no
	WHOIS protocol	no	no	no
	Timbered house	no	no	no
[50]	Library system	no	no	no
[51]	ATM problem Statement [55]	100.00	93.50	93.00
[53]	E-store project [53]	90.00	88.00	no
	Inventory	50.00	71.00	no
	Philips	73.00	70.00	no
	MHCPMS	67.00	50.00	no

ratio of the number of the correctly selected design elements to the number of all design elements in the conceptual model. Recall is calculated by the ratio of number of the correctly selected design elements to the number of expected correct elements. F-measure is measured through harmonic mean of the precision and recall. Using these measurements demonstrated whether a produced model is a reasonable representation of the actual system or not. However, the evaluation criteria (classes, attributes, methods, relationship type, etc.) are assumed to be equal in calculation of these metrics and this assumption may cause errors in experiments because priorities of these criteria vary depending on views of users. Therefore, using objective evaluation criteria including human judgements to measure completeness of the generated model is an important issue that must be considered. In responses to RQ_5, [5] is the single study that uses Analytical Hierarchy Process (AHP) [56] to involve evaluation criteria and experts' opinions into evaluation phase. Table 6 shows details of the evaluation methods used in the reviewed studies.

As mentioned above, there are two necessary steps to evaluate conceptual models in a more objective way; 1) determining criteria which affects the system performance, 2) weighting of these criteria. The weights/priorities of evaluation criteria can be determined by using MCDM methods techniques (e.g., TOPSIS, PROMETHEE, ELECTRE.) which are widely used in various decision-making problems. We foresee that MCDM methods based on expert opinions provide more objective and consistent evaluation results in concept identification studies.

4. Threats to validity

The validity assessment is an essential part of systematic review studies. This section addresses the threats to validity that might have affected the SLR scope using the classification of Wohlin et al. [58].

4.1. Construct validity

Construct validity refers to operational measures for the concepts being researched [58]. The threats in this category are related to components of search process such as digital databases and search queries. As the first step in this direction, we selected four digital libraries (IEEEXplore, ACM, Science Direct, and SpringerLink) and Google Scholar search engine which contain an extensive set of publications in the Software Engineering discipline. The search string is crucial to finding relevant primary studies, and if it is not correctly formed, the review is likely to miss important studies. To eliminate this threat, the search strings were created using different combinations of terms extracted from papers concerning Software Engineering, Requirement Engineering, and design modeling techniques. We improve the search strings several times to reach the maximum number of papers related to the SLR. We also consider stems of search terms in the queries. The first difficulty we encountered is that it is not possible to use the same search strings (given in Section 2.2) for all digital databases. For example, while ScienceDirect provides logical combination of search terms and allows the search string to be used exactly as determined, Google Scholar presents few sorting options to create a query. Thus, we practiced patterns for search terms and modify the search string to each digital library selected. This makes the reproducibility of the automatic search for results possible. We assigned the tasks to ensure that each research was checked by three researchers independently to decrease the potential impact of any bias. The conflicts on study selection processes were solved in team meetings during the review process.

4.2. Internal validity

Internal validity refers to operational measures for data extraction and synthesis. It provides to obtain valuable outcomes and evidence to substantiate the claim.

In data extraction phase, one of the most critical threats is that non-English papers might have prevented the necessary research findings. Since we typed the search terms in English, the papers written in other languages might have been skipped. The other threat is publication bias. It is the problem that positive results are more likely to be published than negative consequences [59]. We attempted to eliminate this problem by observing relevant conference proceedings and well-regarded journals. We did not include patents, technical reports, workshop papers, and thesis into SLR scope. We tried to collect all the search results that are representative of the research questions. Each researcher in this SLR put effort to restrain the threats that cause inconsistencies in data extraction by analyzing the papers.

4.3. External validity

External validity refers to how well the findings of a study can be expected to apply to other researches. Three researches of our study double checked all evaluation results (in Table 4, Table 5, and Table 6) before concluding and determining of research findings.

5. Conclusion

Transforming requirements into conceptual model is a significant but challenging task in SDLC. Although there exist many approaches to automate this process in the literature, it seems that there is not a practical and feasible automated solution yet. In this systematic review, we analyze forty-four previous studies obtained after a carefully designed procedures for selecting papers in digital databases. These studies were evaluated with regard to their approaches, functionalities, dataset used, evaluation methods, generated model types, and languages supported.

The SLR evaluation results guide us to propose some concerns that should be addressed by a concept identification study which links software requirements and conceptual model. First a clear majority of the reviewed studies deal with English requirements to generate conceptual model. The increase in the number of innovative works analyzing the documents in English language provides important contributions to Software Engineering domain. However, working on the other languages beside English enables such systems to reach more users. Accordingly, we conclude that providing a generic approach to multilingualism in requirements analysis ensures broad coverage and accessibility for concept identification researches.

The other considerable limitation is that current studies generally focus on generating only UML class diagrams. However, there are fourteen UML diagram types that represent the different aspects and characteristics of a planned software. Accordingly, describing a full design using separate diagrams makes the overall design easier to understand and maintains the integrity of modular components. Additionally, relationship identification between components of generated models is not completed properly, although relationship identification is the most critical task to build complete and consistent conceptual models. Being unable to determine relationships may limit the traceability between design and implementation phases. To overcome these deficiencies, we recommend to establish a well-designed algorithm including more specific transformation rules.

The dataset is the most important input to test the performance of a proposed model and it must be publicly available to be used in other studies on the same domain. It is seen that nearly all reviewed studies have a dataset that includes a few numbers of requirements and there is no publicly shared dataset on the online repositories except [4] and [50]. This is definitely a research gap which must be considered in the future works.

The performance analysis of a concept identification study is a difficult task, as there is no exact definition for an accurate conceptual model. It is possible that people interpret the same requirements differently because the priorities of evaluation criteria may differ according to users' perspective. Current researches consider that the evaluation criteria have the same priorities and this assumption may lead to inconsistent performance evaluation. This approach can lead to inconsistent results in the evaluation. For this reason, we suggest using statistical MCDM methods that enable the determination of common weights of evaluation criteria according to expert opinions.

As a further study, we propose to design a novel system, which extends the previous studies, by the following functionalities:

- Determining all types of relationships completely,
- Generating more diagram types beside class diagrams,
- Generating source code for more than one programming language,
- Creating a large-scale dataset which includes various software problems to test study,
- Experimenting with different MCDM methods that include expert opinions in the performance evaluation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Pressman, R. S.; Maxim, B. Software Engineering: A Practitioner's Approach. McGraw-Hill Education, 2014.
[2] V.B. Sagar, S. Abirami, Conceptual modeling of natural language functional requirements, J. Syst. Softw. 88 (2014) 25–41.
[3] J. Johnson, A. Henderson, Conceptual models: begin by designing what to design, Interactions 9 (1) (2002) 25–32.
[4] O. Dawood, A. Sahraoui, Toward requirements and design traceability using Natural Language Processing, Eur. J. Eng. Res. Sci. 3 (7) (2018) 42–49.
[5] F. Bozyiğit, Ö. Aktaş, D. Kılınç, Automatic concept identification of software requirements in Turkish, Turk. J. Electr. Eng. Comput. Sci. 27 (1) (2019) 453–470.
[6] M. Omar, G. Baryannis, Semi-automated development of conceptual models from natural language text, Data Knowl. Eng. (2020).
[7] Kitchenham, B. Guidelines for performing Systematic Literature Reviews in software engineering. Version 2.3, EBSE Technical Report EBSE-2007-01, 2007.
[8] P. Brereton, B.A. Kitchenham, D. Budgen, M. Turner, M. Khalil, Lessons from applying the systematic literature review process within the software engineering domain, J. Syst. Softw. 80 (4) (2007) 571–583.
[9] G. Koelsch, Requirements writing for system engineering, Apress, Berkeley, 2016.
[10] M. Elallaoui, K. Nafil, R. Touahni, Automatic transformation of user stories into UML use case diagrams using NLP techniques, Proc. Comput. Sci. 130 (2018) 42–49.
[11] T. Quirchmayr, B. Paech, R. Kohl, H. Karey, G. Kasdepke, Semi-automatic rule-based domain terminology and software feature-relevant information extraction from natural language user manuals, Empirical Software Eng. 23 (6) (2018) 3630–3683.
[12] C.R.M. Reddy, D.E. Geetha, R.R. Rao, T.S. Kumar, Transformation of user interface to activity models and assessing performance of WA/WS, J. Software Eng. Appl. 12 (05) (2019) 101–127.
[13] Sadat, F.; Yoshikawa, M.; Uemura, S. Cross-language information retrieval using multiple resources and combinations for query expansion. In: Advances in Information Systems, 2002, pp. 114-122.
[14] Montes, A.; Pacheco, H.; Estrada, H.; Pastor, O. Conceptual model generation from requirements model: A Natural language processing approach. In Natural Language and Information Systems, 2008, pp. 325–326.
[15] Liu, D.; Subramaniam, K.; Eberlein, A.; Far, B. H. Natural language requirements analysis and class model generation using UCDA. In: Innovations in Applied Artificial Intelligence, 2004, pp. 295–304.
[16] M. Jaiwai, U. Sammapun, Extracting UML class diagrams from software requirements in Thai using NLP, in: 14th International Joint Conference on Computer Science and Software Engineering (JCSSE), 2017, pp. 1–5.
[17] Heine, B., & Narrog, H. (Eds.). The Oxford handbook of linguistic analysis. Oxford Handbooks in Linguistic, 2015.
[18] Tayal, M. A.; Raghuwanshi, M. M.; Malik L. Syntax parsing: Implementation using grammar-rules for English language. In: 2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies, 2014, pp. 376–381.
[19] D.K. Kim, L. Lu, B. Lee, Design pattern-based model transformation supported by QVT, J. Syst. Softw. 125 (Supplement C) (2017) 289–308.
[20] L. Mich, NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA, Natural Language Eng. 2 (2) (1996) 161–187.
[21] S.P. Overmyer, L. Benoit, R. Owen, Conceptual modeling through linguistic analysis using LIDA, in: Proceedings of the 23rd International Conference on Software Engineering, 2001, pp. 401–410.
[22] A.M. Capuchino, N. Juristo, R.P. Van de Riet, Formal justification in object-oriented modelling: a linguistic approach, Data Knowl. Eng. 33 (1) (2000) 25–47.
[23] R.S. Wahono, B.H. Far, A framework for object identification and refinement process in object-oriented analysis and design, Proc. First IEEE Int. Conf. Cogn. Inform. (2002) 351–360.
[24] E. Insfrán, O. Pastor, R. Wieringa, Requirements engineering-based conceptual modelling, Requirements Eng. 7 (2) (2002) 61–72.
[25] Perez-Gonzalez, H. G.; Kalita, J. K. GOOAL: A graphic object-oriented analysis laboratory. In Companion of the 17th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, 2002, pp. 38–39.
[26] H.M. Harmain, R. Gaizauskas, CM-Builder: a natural language-based case tool for object-oriented analysis, Automated Software Eng. 10 (2) (2003) 157–181.
[27] T. Yue, L.C. Briand, Y. Labiche, An automated approach to transform use cases into activity diagrams, Modelling Foundations Appl. (2010) 337–353.
[28] A. Salbrechter, H. Mayr, C. Kop, Mapping pre-designed business process models to UML, in: Proceedings of the Eight IASTED International Conference on Software Engineering and Applications, 2004, pp. 400–405.
[29] L.M. Cysneiros, J.C.S. Leite, Nonfunctional requirements: from elicitation to conceptual models, IEEE Trans. Software Eng. 30 (5) (2004) 328–350.
[30] Song, I. Y.; Yano, K.; Trujillo, J.; Luján-Mora, S. A taxonomic class modeling methodology for object-oriented analysis. Information Modeling Methods and Methodologies, IGI GLOBAL. 2007, pp. 216-240.
[31] X. Zhou, N. Zhou, Auto-generation of class diagram from free-text functional specifications and domain ontology, Artif. Intell. (2004).
[32] V. Ambriola, V. Gervasi, On the systematic analysis of natural language requirements with CIRCE, J. Automated Software Eng. 3 (2006) 107–167.
[33] H. El-Ghalayini, M. Odeh, R. McClatchey, Engineering conceptual data models from domain ontologies: a critical evaluation, Int. J. Inform. Technol. Web Eng. (IJITWE) 2 (1) (2007) 57–70.
[34] H. Christiansen, C.T. Have, K. Tveitane, From use cases to UML class diagrams using logic grammars and constraints, Proc. Recent Adv. Natural Lang. Process. (2007) 128–132.
[35] Gelhausen, T.; Tichy, W. F. Thematic role based generation of UML models from real world requirements. In International Conference on Semantic Computing (ICSC 2007), 2007, pp. 282–289.
[36] A. Fatwanto, C. Boughton, Analysis, specification and modeling of non-functional requirements for translative model-driven development, in: 2008 International Conference on Computational Intelligence and Security, 2008, pp. 405–410.
[37] Giganto, R.; Smith, T. Derivation of classes from use cases automatically generated by a three-level sentence Processing Algorithm. In Third International Conference on Systems (icons 2008), 2008, pp. 75–80.
[38] Seresht, S. M.; Ormandjieva, O.; Sabra, S. Automatic conceptual analysis of user requirements with the requirements engineering assistance diagnostic (READ) tool. In 2008 Sixth International Conference on Software Engineering Research, Management and Applications, 2008, pp. 133–142.
[39] Popescu, D.; Rugaber, S.; Medvidovic, N.; Berry D. M. Reducing ambiguities in requirements specifications via automatically created object-oriented models. In Innovations for Requirement Analysis, Monterey Workshop; 2007, pp. 103–124.
[40] I.S. Bajwa, A. Samad, S. Mumtaz, Object oriented software modelling using NLP based knowledge extraction, Eur. J. Scientific Res. 35 (2009) 22–33.
[41] M. Elbendak, P. Vickers, N. Rossiter, Parsed use case descriptions as a basis for object-oriented class model generation, J. Syst. Softw. 84 (7) (2011) 1209–1223.
[42] M. Brambilla, From requirements to implementation of ad-hoc social web applications: an empirical pattern-based approach, IET Software 6 (2) (2012) 114–126.
[43] S.K. Shinde, V. Bhojane, P. Mahajan, NLP based object oriented analysis and design from requirement specification, Int. J. Computer Appl.. 47 (2012) 30–34.

[44] P. More, R. Phalnikar, Generating UML diagrams from natural language specifications, Int. J. Appl. Inform. Syst. (2012).

[45] S.D. Joshi, D. Deshpande, Textual requirement analysis for UML diagram extraction by using NLP, Int. J. Computer Appl. 50 (8) (2012).

[46] Herchi H, Ben Abdessalem W. From user requirements to UML class diagram. In International Conference on Computer Related Knowledge (ICCRK 2012), 2012.

[47] Tripathy A, Agrawal A, Rath S. Requirement analysis using Natural Language Processing. In Fifth International Conference on Advances in Computer Engineering (ACE 2014), 2014.

[48] M. Landhäußer, S.J. Körner, W.F. Tichy, From requirements to UML models and back: how automatic processing of text can support requirements engineering, Software Qual J. 22 (1) (2014) 121–149.

[49] R. Sharma, S. Gulia, K.K. Biswas, Automated generation of activity and sequence diagrams from natural language requirements, in: 2014 9th International Conference on Evaluation of Novel Approaches to Software Engineering, 2014, pp. 1–9.

[50] M. Ibrahim, R. Ahmad, Class diagram extraction from textual requirements using Natural Language Processing (NLP) Techniques, in: 2010 Second International Conference on Computer Research and Development, 2010, pp. 200–204.

[51] F. Bozyiğit, Ö. Aktaş, D. Kılınç, AutoClass: automatic text to OOP concept identification model, Int. J. Comp. Appl. 150 (10) (2016) 29–34.

[52] Ben Abdessalem Karaa, W., Ben Azzouz, Z., Singh, A., Dey, N., S. Ashour, A., and Ben Ghazala, H. Automatic builder of class diagram (ABCD): an application of UML generation from functional requirements. Software: Practice and Experience, 2016, 46.11: 1443-1458.

[53] Hamza, Z.A. and Hammad, M. Generating UML use case models from software requirements using natural language processing. In 2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO), Manama, Bahrain, 2019, pp. 1-6.

[54] G.A. Miller, WordNet: A lexical database for English, Commun. ACM 38 (11) (1995) 39–41.

[55] J.R. Rumbaugh, M.R. Blaha, W. Lorensen, F. Eddy, W. Premerlani, Object-oriented modelling and design, 1st ed., Prentice-Hall, Englewood Cliffs, NJ, USA, 1990.

[56] T.L. Saaty, Decision making with the Analytic Hierarchy Process, Int. J. Serv. Sci. 1 (2008) 83–98.

[57] G. Lucassen, F. Dalpiaz, J.M.E. Werf, S. Brinkkemper, in: The use and effectiveness of user stories in practice, Springer, Cham, 2016, pp. 205–222.

[58] C. Wohlin, M. Host, P. Runeson, M. Ohlsson, B. Regnell, A. Wesslen, Experimentation in Software Engineering: An Introduction, Kluwer Academic Publishers, 2000.

[59] A. Mlinarić, M. Horvat, V. Šupak Smolčić, Dealing with the positive publication bias: why you should really publish your negative results, Biochemia medica 27 (3) (2017) 447–452.