Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 453 – 470
© TÜBİTAK
doi:10.3906/elk-1803-172

Automatic concept identification of software requirements in Turkish

Fatma BOZYİĞİT[1*], **Özlem AKTAŞ**[2], **Deniz KILINÇ**[3]
[1]Department of Computer Engineering, Graduate School of Natural and Applied Sciences, Dokuz Eylül University, İzmir, Turkey
[2]Department of Computer Engineering, Faculty of Engineering, Dokuz Eylül University, İzmir, Turkey
[3]Department of Software Engineering, Faculty of Technology, Manisa Celal Bayar University, Manisa, Turkey

Received: 25.03.2018 • **Accepted/Published Online:** 04.10.2018 • **Final Version:** 22.01.2019

Abstract: Software requirements include description of the features for the target system and express the expectations of users. In the analysis phase, requirements are transformed into easy-to-understand conceptual models that facilitate communication between stakeholders. Although creating conceptual models using requirements is mostly implemented manually by analysts, the number of models that automate this process has increased recently. Most of the models and tools are developed to analyze requirements in English, and there is no study for agglutinative languages such as Turkish or Finnish. In this study, we propose an automatic concept identification model that transforms Turkish requirements into Unified Modeling Language class diagrams to ease the work of individuals on the software team and reduce the cost of software projects. The proposed work is based on natural language processing techniques and a new rule-set containing twenty-six rules is created to find object-oriented design elements from requirements. Since there is no publicly available dataset on the online repositories, we have created a well-defined dataset containing twenty software requirements in Turkish and have made it publicly available on GitHub to be used by other researchers. We also propose a novel evaluation model based on an analytical hierarchy process that considers the experts' views and calculate the performance of the overall system as 89%. We can state that this result is promising for future works in this domain.

Key words: Software requirements, conceptual model, natural language processing, rule-based model, Unified Modeling Language, class diagram, analytical hierarchy process-based evaluation

1. Introduction

The software development process has many activities starting from requirements analysis to deployment. Requirements analysis is considered as the most important phase in the software development life cycle (SDLC). Software requirements determine the needs of users and involve convenient text-based information about the target system [1]. If a requirements document includes vague statements, it may not be understood clearly by the software team and may cause expensive bugs to be fixed in the next phases [2]. These bugs also extend the delivery time of the software and increase the total cost of the project. Therefore, it is important to write clear requirements and convert them to conceptual models that increase the understanding of the users' needs. The aim of drawing a conceptual model is to map domain information from user's side to software components on the developer's side.

A conceptual model can be represented in different forms, such as Unified Modeling Language (UML) diagrams, entity relationship models (ERMs), and business models (BMs). The UML notion was created by

*Correspondence: ftmbozyigit@gmail.com

Grady Booch, James Rumbaugh, and Ivar Jacobson and has been evolving since the second half of the 1990s [3]. UML has fourteen types of diagrams to model software systems and business processes, and all the diagrams are grouped into two categories: structural diagrams and behavioral diagrams.

In the object-oriented analysis phase, UML diagrams are the most used models to present a wider view of user requirements. Although this phase is generally considered as a manual task, a literature survey shows that automatic generation of UML models from text-based requirements has become an area of interest for researchers. Considering the literature, it is seen that the majority of studies achieve automatic generation of requirements documents written in English. This is because English is one of the most spoken languages in the world and the morphology of English is simple and regular. On the other hand, analyzing textual requirements is a challenging task for morphologically complex languages such as Turkish and Finnish when their agglutinative structure is considered.

In this study, a rule-based method that analyzes requirements written in Turkish and automatically generates UML class diagrams is proposed, and to the best of our knowledge, it is the first such study in the literature. Design components of a class diagram (classes, attributes, methods, and relationships) are extracted from textual requirements utilizing natural language processing (NLP) methods such as tokenization and part of speech (POS) tagging. The main contributions of this study are as follows:

1. It is the first such study carried out on Turkish. To provide this contribution a novel comprehensive rule-based model involving twenty-six transformation rules is developed for Turkish.
2. Considering the literature, there is no common and publicly available dataset for any language to be used in the experimental works of other researchers. As the second contribution of the study, we have prepared a well-defined dataset containing software requirements both in Turkish and English and made it publicly available on GitHub.[1]
3. Studies in the literature perform evaluations with commonly used measures such as precision, recall, and F-measure [4]. These measures assume that each evaluation criterion (classes, attributes, methods, relationship type, etc.) has equal weight, which may cause inconsistent evaluation results. This is because the evaluation phase is highly dependent on personal opinions, and so the priority/weight of these criteria varies depending on views of users. In our study, a novel evaluation method based on multicriteria decision making (MCDM) is proposed.

This paper is organized as follows. In Section 2, related works are presented. Section 3 gives information about the methods used for implementing the automatic model generation of the requirements in Turkish. Section 4 gives information about the proposed methodology of our study and experimental study. Section 5 includes the evaluation results. Section 6 concludes the paper and gives information about our future work.

2. Related works

There are many studies implementing automatic transformation of software requirements into UML diagrams in the literature. Most of the studies are usually carried out in the English language. The first work in this domain is a tool named LOLITA (Large-scale Object-based Linguistic Interactor, Translator, and Analyzer) that was presented by Mich in 1996 [5]. In this study, basic NLP steps (lexical, syntactic, and semantic analysis) and a straightforward rule-based model are used to analyze software requirements. The object diagrams are produced with a transformation process of natural language requirements documents in English.

[1]"Repository of Turkish-AutoConceptIdentifier," accessed 29 July 2018, https://github.com/ftmbozyigit/Turkish-AutoConceptIdentifier.

It has been observed that the reviewed studies commonly generate the class diagram as a conceptual model by analyzing textual requirements. For instance, Sagar and Abirami [6] aimed to create a rule-based model including specific rules to identify classes and relationships in class diagrams by using functional specifications written in English. In their study, they created thirty-eight rules that are gathered under three categories of class, attribute, and relationship. The test of the study was done using a modified ATM problem statement from Rumbaugh's ATM model [7]. Additionally, Ibrahim and Ahmad proposed a model called RACE for transforming software requirements into class diagrams by using NLP methods [8]. They used a rule-based system built at the start of study, and then the NLP techniques were applied to the textual data with the specified rules. Finally, domain ontology was used to refine candidate classes in the diagram to obtain higher accuracy rates. Another example is the study done by Zhou and Zhou [9]. They used domain knowledge with NLP spider model techniques and extracted object-oriented (OO) design elements from written requirements to build class diagrams. It is emphasized that the use of domain ontology improves the performance of the transformation process.

Furthermore, there are some studies in which both class diagrams and a code generation process are proposed. In the study of Bajwa et al., a methodology called UMLG including basic NLP techniques, semantic analysis, and a rule-based model was used to analyze a text and generate source code in Java [10]. They used a dataset including five different requirements documents consisting of simple English sentences to test their study. They indicated that average recall for English requirements specifications was calculated as 80.73% while average precision was calculated as 85.27%. Additionally, Tripathy et al. used NLP techniques to create class diagrams and generate source code from incomplete and ambiguous requirements documents [11]. They only tested the problem statement of the Bank ATM[2] in their study, and an additional dataset was not used. It was pointed out in their study that the accuracy rate was calculated as 96%.

From the studies discussed above, it is clearly seen that there is no study implementing automatic transformation of software requirements for Turkish, which has a complex morphology. Also, studies do not have comprehensive and well-formed datasets that are publicly available. Finally, the studies do not take into account expert opinions in the evaluation of the systems. Table 1 briefly introduces the differences of the studies reviewed.

Table 1. General information about reviewed studies (lang.: language, avg.: average, reqs.: requirements, Pr: precision, Re: recall).

Paper	Supported lang.	Method	Diagram	Source code	# of reqs.	Avg. # of words	Evaluation
[5]	English	Rule-based	Object	-	1	94	-
[6]	English	Rule-based	Class	-	3	87	Pr, Re
[8]	English	Ontology-based	Class	-	1	97	-
[9]	English	Pattern-based	Class	-	1	78	-
[10]	English	Rule-based	Class	Java	5	102	Pr, Re
[11]	English	Rule-based	Class	Java, VB	1	85	Pr, Re

3. Materials and methods

3.1. An overview of the Turkish language

Turkish is a member of the Altaic language family and has distinctive characteristics such as vowel harmony and extensive agglutination [12]. The word structure in agglutinative languages is based on the addition of

[2]Bjork RC. "An example of object-oriented design: an ATM simulation," accessed 12 November 2018, http://www.math-cs.gordon.edu/local/courses/cs211/ATMExample/.

derivational or inflectional morphemes to the roots as suffixes. Since morphemes change the meaning of the stems or roots that they are added to, many different words may be derived from one word by adding morphemes. An example for this situation is the word "Osmanlılaştıramadıklarımızdanmışsınızcasına" ("as if you are among the ones that we could not Ottomanize") [13]. Turkish, being an agglutinative language, has difficulties in NLP, since it has more complex morphology when compared with other languages like English. Therefore, development of an automated text for the diagram transformation tool is a challenging task for Turkish.

3.2. Natural language processing (NLP)

NLP is a science and engineering field, which designs and applies computer systems to be used in processing and understanding natural languages [14]. The developments in information technologies have given momentum to the studies dealing with natural languages in the literature. The basic NLP steps are tokenization, stemming, POS tagging, etc.

3.2.1. Tokenization

One of the preliminary steps of text processing is tokenization, which is the process of separating sentence structure into word groups [15]. To implement this process, the punctuation marks and spaces are considered as separators, and the sentences are separated into their components. In order to simplify information retrieval from requirements, the tokenization is applied first and word sequences are obtained.

3.2.2. Stemming and part of speech (POS) tagging

'Stem' is the name given to the words derived from the roots of nouns and verbs through derivational morphemes. Stemming means that the derivational suffixes added to the words are held and the inflectional suffixes are removed [16]. Note that the derivational suffixes are used to derive new words from roots of words, whereas the inflectional suffixes are added to stem of the name and verbs to specify such information as the state, plurality, time, etc.

After tokenized words are cleaned of inflectional morphemes through the stemming process, POS tagging, which is the process of categorizing word groups considering their function in a sentence, is applied in our proposed model [17]. As a result, each word is separated into categories such as noun, verb, conjunction, etc.

4. Proposed methodology

4.1. Dataset

Datasets have remarkable importance in scientific studies, so they must be well formed, well formatted, and available to be used in other scientific studies. When the studies covered in the literature are examined, it is observed that they have some limitations in the datasets used. These limitations are explained as follows:

- The datasets used include a small number of requirements documents.
- The requirements in the datasets are simple structured and not exactly like real-life scenarios.
- Publicly sharing the datasets commonly allows researchers who work in the same field to compare the efficiency of their methods. It is realized that there is no study that uses a common dataset shared on a public platform.

The dataset was constructed with the use of the "SENG 2115 - Object Oriented Programming (OOP)" course questions taught in the Department of Software Engineering of Manisa Celal Bayar University (MCBU).

Additionally, we collected requirements used in case studies of the related works and translated them into Turkish to enhance our dataset. The dataset contains twenty different requirements in Turkish with their English translations. Twelve of twenty requirements were prepared by the instructors of the Department of Software Engineering between 2015 and 2018. The dataset was made publicly available to be used in the works of other researchers. Table 2 shows detailed information about the dataset used in this study.

Table 2. Some properties of the created dataset.

Property	Value
Number of requirements documents	20
Supported languages	Turkish and English
Supported diagrams	Class
Average number of sentences in requirements	11
Average number of words in requirements	108
Average number of classes in requirements	8
Average number of attributes in requirements	4
Average number of methods in requirements	4

Table 3 represents sample requirements named "Restoran (Restaurant)", written both in English and Turkish languages.

Table 3. Sample requirements in the dataset.

English	Yılmaz Restaurant has more than one personnel and dining tables. Personnel may be employed or discharged at certain times. Each personnel has name, age, and gender information. The restaurant has more than one section. These sections are kitchen, service, and cash. Personnel works according to their sections. Personnel can be cook, waiter, and cashier. The cook stays in the kitchen and prepares the orders. When the cook prepares the order, the system shows the order details and table number. The waiter in the service section serves the costumer. When the waiter serves, the system shows the name of the waiter and the table number. The cashier prepares the check according to the order details in the system.
Turkish	Yılmaz Restoran birden fazla çalışan ve servis masasına sahiptir. Restorana belirli zamanlarda çalışan işe alınır veya işten çıkarılır. Her çalışan isim, yaş ve cinsiyet bilgilerine sahiptir. Restoranda birden fazla bölüm bulunmaktadır. Bu bölümler mutfak, servis ve kasa olmaktadır. Çalışanlar bulundukları bölüme göre iş yapmaktadırlar. Çalışanlar aşçı, garson ve kasiyer olabilmektedir. Aşçı mutfakta bulunur ve sipariş hazırlar. Aşçı siparişi hazırladığında sistem sipariş detayı ve masa numarasını gösterir. Serviste bulunan garson müşteriye servis yapar. Garson servis yaptığında sistem garsonun adı ve masa numarasını gösterir. Kasiyer kasada bulunur ve sistemdeki sipariş detayına göre adisyon hazırlayıp hesap keser.

4.2. General architecture of proposed system

Figure 1 shows the general architecture of the proposed system. First, a tokenization process is implemented. Then the obtained tokens are stemmed and each token's position in the sentence is labeled with POS tagging. After these preprocessing steps, intermediate data to be used as input for the rule-based model are obtained. Then twenty-six transformation rules are applied and OO design elements (class, attribute, method, and relation) are determined. Finally, a class diagram is generated with use of extracted OO design elements.

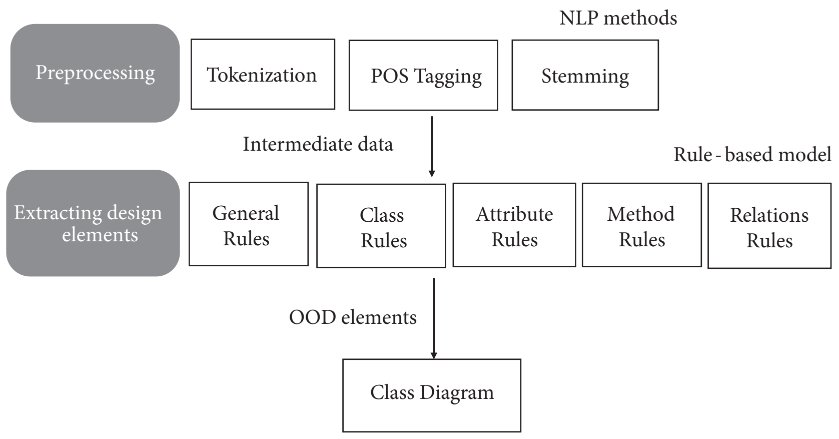

Figure 1. General architecture of proposed model.

4.3. Proposed rule-based model

Rule construction is an effective method to extract information from natural language texts. Rules are based on human knowledge and expertise to find out candidate design elements in generated conceptual models. The major contribution of our study is transforming intermediate text-based data in Turkish to OO design elements using a rule-based model. In this study, a rule-set containing twenty-six rules is created to find out OO design elements from requirements. The rules are categorized as five different topics as seen in Table 4. The relationship rules have also three subcategories, taking different types of relationships into account.

Table 4. Rule-set categories.

Rule category	Number of rules in the category
General rule (GR)	5
Class rule (CR)	3
Attribute rule (AR)	5
Method rule (MR)	3
Relationship rule (RR)	2 (Aggregation)
	4 (Composition)
	4 (Generalization)

To perform the information extraction task, the rules in the different categories are sequentially applied for each input sentence. First, each sentence is gotten as input for the general rules category to determine basic keywords. Then class, attribute, and method rules are applied to identify the name of classes and their corresponding elements. After the extraction of classes and their elements, the relationships rules are performed. This task starts with applying aggregation rules and then the sentences are matched with composition patterns and generalization patterns. If a sentence does not match with any of the defined patterns, it means that it does not contain any relationship to be used in the generated class diagram.

4.3.1. General rules

The aim of the rules in the "General" category is to perform a general analysis and a prefiltering process for the requirements documents. As shown in Table 5, five rules are defined in this category and some of them are explained with the use of examples.

Table 5. Rules in the general rule-set.

GR	Definition
GR_1	Nouns in sentences are candidates for class and attribute names.
GR_2	Proper nouns are removed from the candidate pool of classes and attributes.
GR_3	Succession of the nouns in the sentences is aggregated and they are formed into a single name if the first noun has no affix.
	Example: ders kataloğu (course_catalogue) → ders_katalog
GR_4	Verbs in the sentences are included by the pool of methods' names.
GR_5	Succession of the verbs in the sentences is aggregated and they are formed into a single verb. This is a specific rule for Turkish.
	Example işe almak (employ)→ işe_almak
	To implement this rule, we aggregated auxiliary verbs (olmak, etmek, yapmak, vermek, buyurmak, olunabilmek, geçmek, getirmek, ettirilmek) in the sentences with the words in front of them. On the other hand, we also created and shared a new compound verb exceptions list,[*]which does not include auxiliary verbs such as "etkileşim sağlamak (interact)", "iletişim kurmak (communicate)", "veri yüklemek (load)", and so on.

[*] "Compound verb exceptions", accessed 29 July 2018, https://github.com/ftmbozyigit/Turkish-AutoConceptIdentifier/blob/master/CompoundVerbsExceptions.txt.

4.3.2. Class rules

In the OOP paradigm, a class is an abstract way of describing a real-world entity that includes the properties and behaviors of an object to be created. In this study, three rules are defined in addition to general rules for the identification of the class names. These rules, their explanations, and examples are shown in Table 6.

Table 6. Rules in the class rule-set.

CR	Definition
CR_1	The frequencies of names above a certain threshold are labeled as classes.
CR_2	The second name in the definite noun phrase declares the class strictly if it is stated in the document more than once.
	Example: fakültenin bölümleri (departments of faculty) Fakülte(faculty)→ $Class_1$, bölüm(department)→ $Class_2$
CR_3	If verbs such as "sahip olmak (have)", "içermek (include)", and "bulundurmak (contain)" exist in a sentence, the first name is labeled as a class.

4.3.3. Attribute rules

After the completion of general and class rules, the next step is to extract the attributes of classes from requirements documents. Attributes of classes identify the states of objects. As shown in Table 7, there are five rules in the "Attribute" category.

Figure 2 presents an example sentence processed using general (GR_1, GR_5), class (CR_3), and attribute (AR_4) rules.

4.3.4. Method rules

Behaviors of an object in the class diagram are methods that change the state of the system. In this study, the rule-set involving three different rules is defined to determine the methods of classes. The rules and their explanations are shown in Table 8.

Table 7. Rules in the attribute rule-set.

AR	Definition
AR_1	Adjectives can provide information about the properties of a class.
	Example: Yeşil (Green) kart (card). Renk (Color) is attribute of Kart class.
	To implement this rule, the system uses the list of adjectives provided by Türk Dil Kurumu (TDK) [18] to be used in academic works. We created a new adjective list* by adding definitions for some basic adjectives in the list of TDK. Thus, meanings of the basic adjectives such as color, number, shape, direction, etc. are easily retrieved and used for specification of the attributes.
AR_2	If there is a possessive construction in a sentence and the first name in the construction takes possessive or place suffixes, the second name is pointed out as an attribute of the first name. This is a special rule for Turkish.
	For example, there are many inflectional suffixes can be added to the words "okul (school)" and "öğrenci (student)" and change the situation of these words as following: "okulun öğrencisi", "okulun öğrencisinde", "okuldaki öğrencileri", "okuldaki öğrencilerde", and so on. All of these noun phrases are represented with "student(s) of school", "school's student(s)", and "student(s) in school" in the English language.
	Consequently, all of the possessive constructions above give information about two design elements: "okul" is determined as a class and "öğrenci" is specified as an attribute of the related class considering AR_2.
AR_3	Object of the class derived from a noun can also be attribute of other classes extracted from the same sentence.
	Example: Mağaza asistanı, galerideki arabaların plaka, model, kiralama ücretini sisteme kaydeder. (Store assistant records the information of cars in the gallery such as plate, model, and renting price.)
	Assume that the given example is a sentence in requirements including needs of a rented car gallery system. "galeri (gallery)" and "araba (car)" are determined as classes, because frequencies of them exceed a certain threshold value as stated in CR_1. On the other hand, we can specify "araba" as an attribute of the "galeri" class as a result of AR_2. That is, "araba" has both class and attribute labels in the system.
	galeri $\rightarrow$ $Class_1$, araba$\rightarrow$ $Class_2$ and attribute of $Class_1$
AR_4	If verbs such as "sahip olmak (have)", "içermek (include)", and "bulundurmak (contain)" exist in a sentence, all the names except the name of the class are the attributes of that class.
AR_5	Time, location, and percentage attributes of a class are retrieved according to named entity recognition (NET) supported by the ITU NLP tool [19].
	Example: FB073 nolu uçuşun saat 08:45'te kalkışı yapılmıştır. (FB073 flight departed at 08:45.)
	Assume that the given example is a sentence in the requirements document including needs of an airport system. "uçuş(flight)" is determined as class, because the frequency of it in the requirements text exceeds a certain threshold value as stated in CR_1. Moreover, the NET process retrieves the TIME entity from the sentence.
	uçuş $\rightarrow$ Class, zaman(time) $\rightarrow$ attribute of "uçuş" class

* "Adjective list," accessed 29 July 2018, http://github.com/ftmbozyigit/Turkish-AutoConceptIdentifier/blob/master/AdjectiveList.txt.

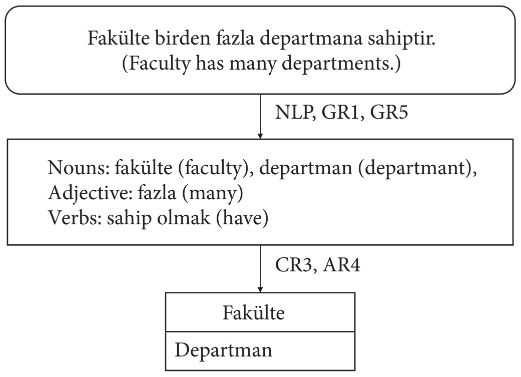
Fakülte birden fazla departmana sahiptir.
(Faculty has many departments.)
NLP, GR1, GR5
Nouns: fakülte (faculty), departman (departmant),
Adjective: fazla (many)
Verbs: sahip olmak (have)
CR3, AR4
Fakülte
Departman

Figure 2. An example of executing general and class rules.

Table 8. Rules in the method rule-set.

MR	Definition
MR_1	Each verb in documents is a candidate method, except verbs such as “sahip olmak (have)”, “içermek (include)”, “bulundurmak (contain)”, “kapsamak (involve)”, “bulundurmak (provide)”, “oluşmak (comprise)”, “oluşturmak (compose)”, “dahil olmak (participate)”, “varolmak (exist)”, “meydana gelmek (consist)”, “kapsamına almak (include)”, and similar verbs listed in our repository.*
MR_2	A verb identified as a method can belong to more than one noun identified as classes in the same sentence.
	Example: Öğrenciler ve öğretim üyeleri, sistem değerlendirme anketlerini yapabilirler. (Students and instructors can conduct system evaluation surveys.)
	Assume that the example sentence is in a requirements document including the needs of a course enrollment system. “öğrenci (student)” and “öğretim_üye (instructor)” are determined as classes, because their frequency in the requirements text exceeds a certain threshold value as stated in CR_1. Our system labels “anket_yapmak() (conduct survey)” verb as methods for both “öğrenci” and “öğretim_üye” classes. Thus, “anket_yapmak()” method belongs to both “öğrenci” and “öğretim_üye” classes.
	öğrenci $\rightarrow$ $Class_1$, instructor $\rightarrow$ $Class_2$, anket_yapmak() $\rightarrow$ method of $Class_1$ and $Class_2$
MR_3	The verb in the sentence having the class information is the method belonging to that class.

* “Verbs not indicating method,” accessed 29 July 2018, https://github.com/ftmbozyigit/Turkish-AutoConceptIdentifier/blob/master/Verbs(notMethods).txt.

4.3.5. Relationship rules

Relationships identify the ways of communication between the classes in the conceptual models. In our study, we define two rules and eight linguistic patterns to find out relationships in the generated class diagrams. Linguistic patterns are specifically formed regarding the grammatical structure of the Turkish language by the authors of the study. Each sentence in the requirements is processed regarding relationship rules and patterns. If input data are matched with a rule or pattern, the relationship between two classes and its type are revealed. The defined rules and patterns in this study are split into three subcategories to get aggregation, composition, and generalization relationship types. These three subcategories are described in Tables 9, 10, and 11, respectively. Aggregation is a kind of association between two classes describing a part of a relationship. Related classes in this type of relation are not affected if a container class is deleted. On the other hand, a composition relationship indicates a strict aggregation relation between the classes. If a container class is deleted all its classes also need to be deleted [20]. The generalization relationship is used to generate a derived class that inherits all elements in the parent class. The rules in the generalization and composition subcategories are defined using pattern-based modeling. For relationships indicating generalization and composition, a list of patterns covering relevant cases is defined by the authors of study.

Table 9. Rules in the aggregation rule-set.

AggR	Rule definition
$AggR_1$	If all nouns are labeled as the class in a noun phrase, there is a certain relationship between them.
	Example: Banka'nın müşterileri (Customers of the bank)
	There is a relation between Banka and Müşteri classes.
$AggR_2$	If an attribute in a sentence is also labeled as a class, there is a relationship.

Table 10. Patterns (Turkish) in the composition set.

CompP	Pattern definition
$CompP_1$	Bir ($Class_1$) birden fazla ($Class_2$) oluşmaktadır/içermektedir/bulundurmaktadır.
$CompP_2$	($Class_1$) bir tür ($Class_2$).
$CompP_3$	($Class_1$) ($Class_2$) parçasıdır/kısımıdır/elemanıdır oluşmaktadır/içermektedir.
$CompP_4$	($Class_1$) ($Class_2$) ait bir parçadır/kısımdır/bölümdür.

Table 11. Patterns (Turkish) in the generalization set.

GenP	Pattern definition
$GenP_1$	Bir ($Class_1$) ($Class_2$) kategori yer almaktadır/dahildir/bulunmaktadır.
$GenP_2$	($Class_1$) ($Class_2$)'dır/dir.
$GenP_3$	($Class_1$) ($Class_2$) ait bir kategoridir.
$GenP_4$	($Class_1$) ($Class_2$)'nın bir alt dalıdır/kategorisidir/alanıdır.

4.4. Analytical hierarchy process (AHP)-based evaluation

Current research appears to validate that there is no related study in the literature employing MCDM methods to evaluate the performance of the system. The evaluation of a study transforming requirements into conceptual models is a comprehensive process, because there are various criteria affecting the performance of the produced model. Furthermore, the importance of the criteria may vary depending on the view and preferences of the decision makers. Since the proposed method requires handling various evaluation criteria, we developed a new evaluation model by applying AHP that allows decision makers to prioritize criteria in order to deal with complex decision making problems [4]. The structure of the AHP is as follows:

1. The decision-making problem is defined and the criteria affecting decision points are determined.
2. Decision makers perform pairwise comparison between the specified criteria by using the ranking scale proposed by Saaty [21].

3. Matrix calculations are performed with the following proposed methodology of Saaty [21] and then weights/priority orders for each criterion are specified.

Our AHP-based evaluation model consists of three basic steps, which are explained in the following:

Step 1 (Defining problem and determining criteria): The determined criteria for the evaluation of our study are presented in Table 12.

Table 12. Definitions of evaluation criteria.

Number of criterion	Acronym	Definition
Criterion 1	C_1	Finding the classes completely.
Criterion 2	C_2	Finding the relationships between the classes completely.
Criterion 3	C_3	Finding the attributes of the classes completely.
Criterion 4	C_4	Finding the methods of the classes completely.
Criterion 5	C_5	Specifying the relationship types correctly.

Step 2 (Pairwise comparison): After specification of the problem statement and the criteria, we asked three academicians (from MCBU and Dokuz Eylül University) and the head of the software department at Commensis Software Company, who are experts in the OO programming domain, to be participants in the evaluation of our study. They compared each of the determined criteria using a ranking scale from one to nine.

Step 3 (Calculating weights of the criteria): The weights of criteria were calculated by applying the matrix calculation following Saaty's proposed study [21]. The results are shown in Table 13.

Table 13. Weight of each criterion calculated by AHP.

Criterion	Weight
C_1	53.7%
C_2	21.1%
C_3	10.0%
C_4	10.0%
C_5	5.2%

Results of AHP regarding feedbacks of the experts indicate that the criteria used for the evaluation of conceptual models may have different weights. In the related studies, weights of each evaluation criterion are considered as equal. This assumption may not always yield accurate results. For instance, it is admitted that finding all the specified classes correctly in a conceptual model is the most important factor according to AHP results including the views of the experts in our study.

4.5. Experimental study

Each requirement in the dataset is tested to validate our proposed approach. We selected the requirements named "Restoran" shown in Table 3 as a case study for this section. First, the preprocessing phase is applied to the text parts of the requirements using basic NLP steps as mentioned in Section 3.2. The output of this phase is a list of intermediate data including POS tags and keywords, which specify the relationships. The list of data is shown in Table 14.

Next, the list of intermediate data in Table 14 is processed by applying the first four categories of the rule-based model (GR, CR, AR, and MR). Thus, classes and their corresponding elements (attributes and

Table 14. Intermediate data obtained through NLP methods on Restaurant requirements.

Nouns	restoran (restaurant), çalışan (personnel), servis_masa (dining table), isim (name), yaş (age), cinsiyet (gender), bilgi (information), bölüm (section), mutfak (kitchen), servis (service), kasa (cash register), aşçı (cook), garson (waiter), kasiyer (cashier), sipariş (order), sistem (system), sipariş_detayı (order detail), masa_numarası (table number), müşteri (customer), garson_ad (waitress's name), adisyon (check)
Proper nouns	Yılmaz
Adjectives	-
Verbs	sahip_olmak (have), işe_almak (employee), işten_çıkarmak (discharge), bulunmak, olmak (be), iş_yapmak (work), sipariş_hazırlamak (prepare an order), göstermek (show), servis_yapmak (service), adisyon_hazırlamak (prepare cash), hesap_kesmek (cash)
Adverbs	-
Relationship keywords	sahip_olmak, olmak, bulunmak

methods) are determined to generate the related class diagram. The results of this process for the case study are presented in Table 15.

Table 15. Design elements in the "Restoran" requirements.

Sentence no.	Classes	Attributes	Methods	Specific keywords	Used rules
1	restoran	çalışan, servis_masası	-	sahip olmak	GR_1, CR_1, CR_4, AR_4
2	restoran	çalışan	işe_almak, işten_çıkarmak	-	GR_1, GR_4, GR_5, CR_1
3	çalışan	isim, yaş, cinsiyet	-	sahip_olmak	GR_1, CR_4, AR_4
4	restoran	bölüm	-	bulunmak	GR_1, GR_4, CR_4, MR_1
5	bölüm	mutfak, servis, kasa	-	olmak	GR_1, CR_1, AR_4, MR_3
6	çalışan	-	-	olmak	GR_1, CR_1, AR_4, MR_3
7	aşçı	mutfak	-	bulunmak	GR_1, GR_4, CR_1, MR_3
8	sistem	sipariş_detayı, masa_numarası	göstermek	-	GR_1, GR_3, GR_4
9	garson	servis	servis_yapmak	bulunmak	GR_4, GR_5, CR_1, MR_3
10	sistem	garson_ad, masa_ numarası	göstermek	-	GR_1, GR_4, GR_5, CR_1
11	kasiyer	-	hesap_kesmek	-	GR_4, CR_1, MR_1

Moreover, the associations between the specified classes are determined performing the relationship rules and patterns (RPR) to the "Restoran" requirements. Finally, all design elements are extracted and the transformation process is accomplished. Resulting relationships and used rules/patterns are shown in Table 16.

For the "Restoran" requirements, nine classes, nine attributes, three methods, and eleven relationships are obtained performing the proposed rule-based model. The generated class diagram is illustrated in Figure 3. Participating experts confirmed that all the design elements are located correctly in the generated class diagram. The results yielded by the experimental study provide convincing evidence that our proposed study effectively performs the transformation of requirements texts into class diagrams.

Table 16. Relationships extracted from "Restoran" requirements.

Sentence no.	Class_1	Class_2	Relationship type	Used rules
1	restoran	çalışan	Composition	CompP_1
2	restoran	bölüm	Composition	CompP_2
3	çalışan	çalışan aşçı, garson, kasiyer	Generalization	GenP_1
4	restoran	bölüm mutfak, servis, kasa	Generalization	GenP_2
5	aşçı	mutfak	Aggregation	AggR_2
6	garson	servis	Aggregation	AggR_2
7	kasiyer	kasa	Aggregation	AggR_2

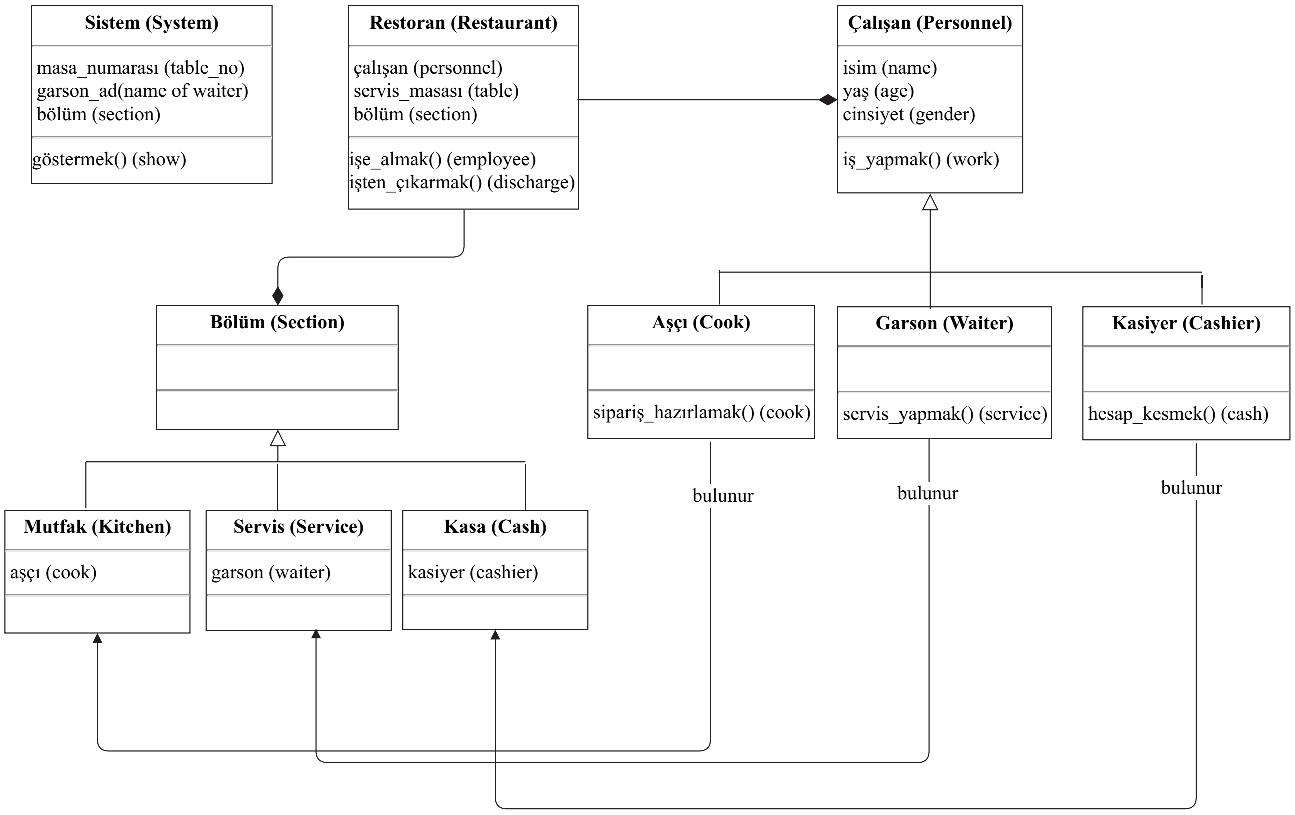

Figure 3. Class diagram of "Restoran" model.

5. Evaluation of system performance

The evaluation process with respect to the specified criteria (stated in Section 4.4) is performed by comparing the outputs of the system with the class diagrams generated by the experts who participated in this study. Assume that the set of design elements specified in the experts' model is denoted by E and the set of elements revealed by the system is denoted by S. Numbers of correct, incorrect, and missing elements determined by the comparison between S and E are as follows.

1. The cardinality of intersection of S and E gives the number of elements correctly identified by the system (it is denoted as $N_{correct}$).

2. The cardinality of difference of S and E gives the number of incorrect determined elements in the generated class diagram by the system (it is denoted as $N_{incorrect}$).
3. The cardinality of difference of E and S gives the number of missing elements that could not be extracted by the system (it is denoted as $N_{missing}$).

In this part, the proposed system is tested and evaluated for each criterion C_i (stated before in Section 4.4) using all the requirements documents in the dataset. The detailed experimental results for each requirement are presented in Table 17.

Table 17. Detailed experimental results regarding each criterion (C_iC: number of correct elements regarding C_i, C_iI: number of incorrect elements regarding C_i, C_iM: number of missing elements regarding C_i).

Requirements	C_1C	C_1I	C_1M	C_2C	C_2I	C_2M	C_3C	C_3I	C_3M	C_4C	C_4I	C_4M	C_5C	C_5I	C_5M
R_1(Restaurant)	11	0	0	11	0	0	9	2	1	6	1	0	4	2	3
R_2(Company)	8	0	1	6	1	3	7	0	0	5	1	2	6	1	3
R_3(Library)	9	2	0	5	2	0	3	0	0	5	2	1	5	2	0
R_4(Game)	5	0	2	3	1	2	4	0	1	4	2	0	4	0	2
R_5(Music band)	7	0	1	6	1	0	4	3	1	3	1	0	3	0	0
R_6(Timetable)	7	2	0	5	0	1	10	2	2	6	1	0	3	1	2
R_7(Super market)	6	1	2	3	1	2	7	2	3	7	0	3	4	0	1
R_8(Hotel reservation)	9	0	2	5	2	0	12	0	4	5	1	1	5	1	1
R_9(Fitness center)	8	1	0	4	1	1	9	3	2	3	3	2	3	1	0
R_{10}(File manager)	7	0	0	6	0	0	4	1	1	3	0	2	5	1	2
R_{11}(Football team)	10	0	0	6	2	1	5	2	0	7	1	1	7	2	2
R_{12}(Car gallery)	5	0	0	3	0	1	12	2	0	8	1	0	2	1	1
R_{13}(Enrollment [6])	6	1	0	8	1	1	8	1	2	9	2	2	10	2	3
R_{14}(ATM [7])	8	1	0	8	0	2	3	1	0	3	0	1	9	1	2
R_{15}(Video rental [22])	4	0	1	4	1	0	8	2	1	8	2	0	4	1	1
R_{16}(Cinema [23])	4	0	0	4	0	1	4	1	0	6	1	1	4	0	2
R_{17}(Timbered house [23])	9	0	0	7	1	0	3	0	0	1	1	0	3	3	1
R_{18}(Musical store [24])	6	0	0	9	1	1	4	0	1	8	2	1	8	2	3
R_{19}(Pressure [25])	4	1	1	4	1	2	5	1	1	3	1	0	3	0	3
R_{20}(Airport [26])	7	1	0	4	1	2	7	0	0	5	0	2	4	0	1

We calculated performance measures (precision, recall, and F-measure) for each evaluation criterion to evaluate the system. Precision (Pr) refers to the accuracy of the proposed system and gives information on how much of the output extracted by the system is correct [27]. It is obtained by finding the ratio of the correctly identified data to the total extracted data in the generated model. Its formula is given in Eq. (1).

$$Pr = \frac{N_{correct}}{N_{correct} + N_{incorrect}} \tag{1}$$

Recall (Re) indicates the ability of the system to generate all design elements correctly. It is the ratio of the correct design elements extracted by the system to the number of true elements in the experts' model. The formula of recall is given in Eq. (2).

$$Re = \frac{N_{correct}}{N_{correct} + N_{missing}} \tag{2}$$

The F-measure (Fm) of the proposed system is obtained by calculating the weighted harmonic mean of its precision and recall. The formula of Fm is given in Eq. (3).

$$F_{measure} = \frac{2 \times Pr \times Re}{Pr + Re} \tag{3}$$

Pr, Re, and Fm values for all requirements in the dataset regarding each evaluation criterion are presented in Table 18.

Table 18. Precision, recall, and F-measure values.

	C_1			C_2			C_3			C_4			C_5		
Requirements	Pr	Re	Fm	Pr	Re	Fm	Pr	Re	Fm	Pr	Re	Fm	Pr	Re	Fm
R_1	1.00	1.00	1.00	1.00	1.00	1.00	0.80	0.90	0.86	0.75	1.00	0.86	0.67	0.57	0.62
R_2	1.00	0.88	0.94	0.86	0.67	0.75	1.00	1.00	1.00	0.86	1.00	0.92	0.87	0.67	0.76
R_3	0.82	1.00	0.90	0.71	1.00	0.83	1.00	1.00	1.00	0.71	0.83	0.77	0.71	1.00	0.83
R_4	1.00	0.71	0.83	0.75	0.60	0.67	1.00	0.80	0.89	0.67	1.00	0.80	0.67	1.00	0.80
R_5	1.00	0.88	0.93	0.86	1.00	0.92	0.57	0.80	0.67	0.75	1.00	0.86	1.00	1.00	1.00
R_6	0.78	1.00	0.88	1.00	0.83	0.91	0.83	0.83	0.83	0.86	1.00	0.92	0.75	0.60	0.67
R_7	0.87	0.91	0.89	1.00	0.89	0.94	0.84	0.93	0.88	0.82	0.96	0.88	0.93	1.00	0.96
R_8	1.00	0.82	0.90	0.71	1.00	0.83	1.00	0.75	0.86	0.83	0.83	0.83	0.83	0.83	0.83
R_9	0.89	1.00	0.95	0.80	0.80	0.80	0.75	0.82	0.78	0.50	0.60	0.55	0.75	1.00	0.86
R_{10}	1.00	1.00	1.00	1.00	1.00	1.00	0.80	0.80	0.80	1.00	0.60	0.75	0.83	0.71	0.77
R_{11}	1.00	1.00	1.00	0.75	0.86	0.80	0.71	1.00	0.83	0.88	0.88	0.88	0.78	0.78	0.78
R_{12}	1.00	1.00	1.00	1.00	0.75	0.86	0.86	1.00	0.92	0.89	1.00	0.94	0.67	0.67	0.67
R_{13}	0.86	1.00	0.92	0.89	0.89	0.89	0.89	0.80	0.84	0.82	0.82	0.82	0.83	0.77	0.80
R_{14}	0.89	1.00	0.94	1.00	0.80	0.89	0.75	1.00	0.86	1.00	0.75	0.86	0.90	0.82	0.86
R_{15}	1.00	0.80	0.89	0.80	1.00	0.89	0.80	0.89	0.84	0.80	1.00	0.89	0.80	0.80	0.80
R_{16}	1.00	1.00	1.00	1.00	0.80	0.89	0.80	1.00	0.89	0.86	0.86	0.86	1.00	0.67	0.80
R_{17}	1.00	1.00	1.00	0.88	1.00	0.93	1.00	1.00	1.00	0.50	1.00	0.67	0.50	0.75	0.60
R_{18}	1.00	1.00	1.00	0.90	0.90	0.90	1.00	0.80	0.89	0.80	0.89	0.84	0.80	0.73	0.76
R_{19}	0.80	0.80	0.80	0.80	0.67	0.73	0.83	0.83	0.83	0.75	1.00	0.86	1.00	0.50	0.67
R_{20}	0.88	1.00	0.93	0.80	0.67	0.73	1.00	1.00	1.00	1.00	0.71	0.83	1.00	0.80	0.89

The studies in the literature calculate the values of precision, recall, and F-measure metrics assuming that all evaluation criteria have the same weights (standard calculation). However, considering these criteria as equally weighted may cause misleading evaluation results, since the priority of each criterion varies depending on views of users. Thus, we propose a novel evaluation model including AHP to assign a weight to each criterion in the direction of the experts' opinions (as stated in Section 4.4). The F-measure value of each evaluation criterion (in Table 18) is multiplied by the weights of criteria (Table 13), and a new accuracy ratio is calculated for generated class diagrams. The formula for calculating the weighted F-measure value is given in Eq. (4).

$$Accuracy = (w_1 \times C_1) + (w_2 \times C_2) + (w_3 \times C_3) + (w_4 \times C_4) + (w_5 \times C_5) \tag{4}$$

Table 19 shows the comparison of performances calculated by conventional and AHP-based evaluation methods in terms of precision, recall, and F-measure for ten requirements in the dataset.

Table 19. Conventional and AHP-based evaluation results.

	Conventional evaluation			Evaluation using AHP		
Requirements	Pr	Re	Fm	Pr	Re	Fm
R_1	0.87	0.88	0.88	0.95	0.92	0.94
R_2	0.93	0.85	0.89	0.95	0.82	0.88
R_3	0.79	0.97	0.87	0.80	0.98	0.88
R_4	0.88	0.75	0.83	0.91	0.73	0.81
R_5	0.84	0.93	0.87	0.90	0.91	0.90
R_6	0.84	0.85	0.85	0.83	0.93	0.88
R_7	0.88	0.76	0.79	0.87	0.90	0.87
R_8	0.88	0.85	0.86	0.92	0.85	0.87
R_9	0.74	0.84	0.79	0.81	0.90	0.85
R_{10}	0.93	0.82	0.87	0.97	0.92	0.95
R_{11}	0.82	0.90	0.86	0.90	0.95	0.92
R_{12}	0.88	0.88	0.88	0.96	0.93	0.94
R_{13}	0.85	0.85	0.85	0.86	0.92	0.89
R_{14}	0.91	0.87	0.89	0.91	0.92	0.92
R_{15}	0.84	0.90	0.87	0.91	0.87	0.89
R_{16}	0.93	0.86	0.90	0.95	0.93	0.94
R_{17}	0.78	0.95	0.85	0.90	0.99	0.94
R_{18}	0.90	0.86	0.88	0.95	0.93	0.94
R_{19}	0.84	0.76	0.80	0.81	0.78	0.79
R_{20}	0.94	0.84	0.88	0.89	0.89	0.89

When we review the results in Table 19, it is seen that nearly all classes and relationships in the generated Restoran (R_1) class diagram are correctly determined. This shows that C_1 and C_2 criteria are successfully met by the system for R_1. However, it is seen that there are two incorrect and three missing relationship types determined. That is, the C_5 criterion is not met properly in the generated Restoran model. As seen in Table 19, the F-measure of the Restoran model calculated with the AHP-based evaluation is 94%; however, it is measured as 88% by performing conventional evaluation. Since evaluation criteria are assumed to be equal in conventional evaluation, elements that do not meet the C_5 considerably reduce the value of the F-measure. Experts participating in our study stated that the incorrect and missing elements for C_5 (relationship type) do not affect the system performance dramatically, because it has lower priority order than the other evaluation criteria. Thus, they claimed that evaluation using AHP gives more realistic results than the conventional method. For this reason, we can state that using MCDM methods including expert opinions possibly provide more realistic and consistent evaluation results in concept identification studies.

Additionally, as can be understood from the evaluation results in the table, our study achieved a success rate of over 85% on a large majority of twenty requirements in the dataset. However, the performance results on the R_4 (81%) and R_{19} (79%) requirements are significantly lower compared to the others. This is because both of the two requirements are not well written in Turkish and the structure of the sentences is complex.

6. Conclusion

Transforming requirements into OO conceptual models is a vital but challenging task in software development. Although mostly done manually, there are available approaches to automate this step of SDLC. A clear majority of these approaches deal with English requirements and there is no study generating conceptual models for agglutinative languages such as Korean, Finnish, and Turkish. Thus, the main contribution of our study, automatically generating class diagrams from Turkish requirements, is accomplished by using NLP techniques and a novel rule-based model including twenty-six transformation rules. It is seen that all studies in this domain have a dataset containing a small number of documents and there is no shared dataset that is publicly available from online repositories. This is definitely a gap that needs to be filled. Hence, we have prepared an enhanced dataset that contains twenty software requirements in Turkish. Additionally, it is publicly available on GitHub to be used by other researchers in this domain. The performance evaluation of concept identification studies is vague because there is no definition for an accurate conceptual model. It is possible that two different people differently evaluate the same requirements document, because the priorities of evaluation criteria can vary from person to person. However, it is seen that the reviewed studies consider that the evaluation criteria have the same priorities and do not include expert opinions for performance measurement of the systems. This approach can lead to inconsistent results in evaluation of the studies. For this reason, the third contribution is achieved by using an AHP-based evaluation model and decision makers' feedback. As a result of the evaluation, average accuracy of the proposed model is measured as 89%. We cannot compare our results with other studies, because our work is the primary study carried out on Turkish requirements in the literature. It is clearly seen that the results of our study are motivating enough for future works, although the evaluation is performed against an experts' model including their assumptions and implicit information.

As our future work, it is aimed to design a novel system that extends our study with the following functionalities:

- Specifying all types of relationships between the classes completely,
- Extracting more diagram types beside class diagrams,
- Generating source code.

References

[1] Pohl K. Requirements Engineering: Fundamentals, Principles, and Techniques. 1st ed. Berlin, Germany: Springer-Verlag, 2010.

[2] Sagar V, Abirami S. Conceptual modeling of natural language functional requirements. J Syst Software 2014; 88: 25-41.

[3] Hunt J. Guide to the Unified Process Featuring UML, Java and Design Patterns. 2nd ed. London, UK: Springer-Verlag, 2003.

[4] Bozyiğit F, Aktaş Ö, Kılınç D. A novel evaluation approach for the systems transforming software requirements to object oriented source code. In: International Conference on Engineering Technologies; 7–9 December 2017; Konya, Turkey. pp. 129-134.

[5] Mich L. NL-OOPS: From natural language to object oriented requirements using the natural language processing system LOLITA. Lect Notes Artif Int 1996; 2: 161-187.

[6] Sagar VBRV, Abirami S. Conceptual modeling of natural language functional requirements. J Syst Software 2014; 88: 25-41.

[7] Rumbaugh J, Blaha M, Premerlan W, Eddy F, Lorensen W. Object-Oriented Modeling and Design. 2nd ed. New York, NY, USA: Pearson Education, 2007.

[8] Ibrahim M, Ahmad R. Class diagram extraction from textual requirements using natural language processing techniques. In: 2010 Second International Conference on Computer Research and Development; 7–10 May 2010; Kuala Lumpur, Malaysia. New York, NY, USA: IEEE. pp. 200-204.

[9] Zhou X, Zhou N. Auto-generation of class diagram from free-text functional specifications and domain ontology. In: Artificial Intelligence; 2004.

[10] Bajwa IS, Samad A, Mumtaz S. Object oriented software modelling using NLP based knowledge extraction. European Journal of Scientific Research 2009; 35: 22-33.

[11] Tripathy A, Agrawal A, Rath, SK. Requirement analysis using natural language processing. In: Fifth International Conference on Advances in Computer Engineering; 26–27 December 2014; Kochi, India. pp. 463-472.

[12] Kılınç D, Özçift A, Bozyiğit F, Yıldırım P, Yücalar F, Borandağ E. TTC-3600: A new benchmark dataset for Turkish text categorization. J Inf Sci 2017; 43: 174-185.

[13] Aşlıyan R, Günel K, Filiz A. Türkçe Otomatik Heceleme Sistemi ve Hece İstatistikleri. In: Akademik Bilişim ’06; 9–11 February 2006; Denizli, Turkey (in Turkish).

[14] Prakash M, Lucila O, Wendy W. Natural language processing: an introduction. J Am Med Inform Assn 2011; 18: 544-551.

[15] Rehman Z, Anwar W, Bajwa UI, Xuan W, Chaoying Z. Morpheme matching based text tokenization for a scarce resourced language. PLoS One 2013; 8: e68178.

[16] Can F, Kocberber S, Balcik E, Kaynak C, Ocalan HC, Vursavas OM. Information retrieval on Turkish texts. J Assoc Inf Syst 2008; 59: 407-421.

[17] Eryiğit G. ITU Turkish NLP web service. In: Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics; 2014; Gothenburg, Sweden. pp. 1–4.

[18] Türk Dil Kurumu. Büyük Türkçe Sözlük. Ankara, Turkey: TDK, 2018 (in Turkish).

[19] Şeker GA, Eryiğit G. Extending a CRF-based named entity recognition model for Turkish well formed text and user generated content of Turkish. Semant Web 2017; 8: 625-642.

[20] Kim DK, Lu L, Lee B. Design pattern-based model transformation supported by QVT. J Syst Software 2017; 125: 289-308.

[21] Saaty TL. Decision making with the analytic hierarchy process. International Journal of Services Sciences 2008; 1: 83-98.

[22] Kiyavitskaya N, ZeniMich L, Berry DM. Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. Requir Eng 2008; 13: 207–239.

[23] Landhäußer M, Körner SJ, Tichy WF. From requirements to UML models and back: how automatic processing of text can support requirements engineering. Software Qual J 2014; 22: 121-149.

[24] Kumar DD, Sanyal R. Static UML model generator from analysis of requirements (SUGAR). In: 2008 Advanced Software Engineering and Its Applications; 2008; Hainan Island, China. pp. 77–84.

[25] Berry DM. Ambiguity in natural language requirements documents. In: Monterey Workshop; 2007; Monterey, CA, USA. pp. 1-7.

[26] Ball CG, Kim RL. An Object-Oriented Analysis of Air Traffic Control. McLean, VA, USA: The MITRE Corporation, 1991.

[27] Harmain HM, Gaizauskas R. Cm-builder: A natural language-based case tool for object-oriented analysis. Automat Softw Eng 2003; 10: 157-181.