DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OBJECT ORIENTED ANALYSIS AND SOURCE CODE VALIDATION USING NATURAL LANGUAGE PROCESSING

by
Fatma BOZYİĞİT

June, 2019
İZMİR

OBJECT ORIENTED ANALYSIS AND SOURCE CODE VALIDATION USING NATURAL LANGUAGE PROCESSING

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of
Philosophy in Computer Engineering

by
Fatma BOZYİĞİT

June, 2019
İZMİR

Ph.D. THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"OBJECT ORIENTED ANALYSIS AND SOURCE CODE VALIDATION USING NATURAL LANGUAGE PROCESSING"** completed by **FATMA BOZYİĞİT** under supervision of **ASST. PROF. DR. ÖZLEM AKTAŞ** and **ASSOC. PROF. DR. DENİZ KILINÇ** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Doctor of Philosophy.

Asst. Prof. Dr. Özlem AKTAŞ

Supervisor

Assoc. Prof. Dr. Deniz KILINÇ

Supervisor

Thesis Committee Member

Thesis Committee Member

Examining Committee Member

Examining Committee Member

Examining Committee Member

Prof. Dr. Kadriye ERTEKİN

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Assist. Prof. Dr. Özlem AKTAŞ, for her immense knowledge, continuous support, supervision, and useful suggestions throughout this study. She conducted all the time of my research and writing of this thesis. It has been an honor to be her Ph.D. student. I am also grateful to Assoc. Prof. Dr. Deniz KILINÇ, my second supervisor, for his guidance, valuable advices, and encouragement throughout this study. I could not have imagined having a better mentor for my Ph.D. study. Also, I would like to offer my special thanks to Assoc. Prof. Dr. Akın ÖZÇİFT for his valuable advice and encouragement throughout this study.

Last but not least, I would like to thank my brother Alican BOZYİĞİT for supporting me spiritually throughout writing this thesis and my life in general. It would not have been able to complete this thesis without his support and help.

Fatma BOZYİĞİT

OBJECT ORIENTED ANALYSIS AND SOURCE CODE VALIDATION USING NATURAL LANGUAGE PROCESSING

ABSTRACT

Software requirements include description of the features for the target system and express the expectations of users. In analysis phase, requirements are transformed into easy-to-understand conceptual models that facilitate communication between stakeholders. Although creating conceptual model using requirements is mostly implemented manually by an analyst, 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 thesis, we propose an automatic concept identification model which transforms Turkish requirements into Unified Modelling Language (UML) class diagram to ease the work of individuals in the software team, and reduce the cost of software projects.

The proposed thesis is based on the Natural Language Processing (NLP) techniques and a new rule-set containing twenty-six rules is created to find out Object Oriented (OO) design elements from requirements. Since there is no publicly available dataset on the online repositories, we have created a well-defined dataset containing ten software requirements in Turkish and made it publicly available on GitHub to be used by other researchers. We also proposed a novel evaluation model based on Analytical Hierarchy Process (AHP) that considers the experts' views and calculate the performance of the overall system as enough successful. We can state that this result is promising for the future works in this domain.

Keywords: Software requirements, conceptual model, Natural Language Processing, rule-based model, Unified Modelling Language, class diagram, Analytical Hierarchy Process

DOĞAL DİL İŞLEME KULLANIMI İLE NESNE TABANLI ANALİZ VE KAYNAK KOD DEĞERLENDİRMESİ

ÖZ

Yazılım gereksinimleri, hedef sistemin özelliklerini ve kullanıcıların beklentilerini ifade eder. Analiz aşamasında, gereksinimler paydaşlar arasında iletişimi kolaylaştıran anlaşılması kolay kavramsal modellere dönüştürülür. Gereksinimleri kullanarak kavramsal model oluşturma, çoğunlukla bir analist tarafından elle uygulanmasına rağmen, bu süreci otomatikleştiren modellerin sayısı son zamanlarda artmıştır. Modellerin ve araçların çoğu İngilizcedeki gereksinimleri analiz etmek için geliştirilmiştir ve Türkçe ya da Fince gibi sondan eklemeli diller için mevcut bir çalışma bulunmamaktadır. Bu tezde, yazılım takımında yer alan kişilerin çalışmalarını kolaylaştırmak ve yazılım projelerinin maliyetini düşürmek için Türkçe gereksinimleri Birleştirilmiş Modelleme Dili (BMD) sınıf diyagramına dönüştüren bir otomatik kavram tanımlama modeli sunulmuştur.

Önerilen tezde, Doğal Dil İşleme (DDİ) tekniklerinden faydalanılmıştır ve Nesneye Yönelik (NY) tasarım öğelerini gereksinimlerden bulmak için yirmi altı kural içeren yeni bir kural kümesi oluşturulmuştur. Çevrimiçi depolarda diğer araştırmacıların kullanımına açık bir veri kümesi bulunmadığından, Türkçe olarak yirmi yazılım gereksinimi içeren iyi tanımlanmış bir veri kümesi oluşturulmuş ve diğer araştırmacılar tarafından kullanılmak üzere herkese açık bir şekilde GitHub üzerinden kullanıma sunulmuştur. Ayrıca, uzmanların görüşlerinden faydalanarak istatistiksel Analitik Hiyerarşi Süreci'ne (AHP) dayanan yeni bir değerlendirme modeli önerilerek sistem performansı değerlendirilmiştir. Elde edilen sonuçların, bu alanda yapılacak çalışmalar için umut verici olduğunu olduğu gözlemlenmektedir.

Anahtar kelimeler: Yazılım gereksinimleri, kavramsal model, doğal dil işleme, kural tabanlı modelleme, Birleştirilmiş Modelleme Dili, sınıf diyagramı, Analitik Hiyerarşi Prosess

CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER ONE
INTRODUCTION

1.1 General

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 needs of users and involve convenient text-based information about target system (Pohl, 2010). If a requirements document includes vague statements, it may not be understood clearly by software team and causes expensive bugs to fix in next phases (Sagar & Abirami, 2014). These bugs also extend the delivery time of software and increase the total cost of the project. Therefore, it is important to write clear requirements and convert them to conceptual models which 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 (ERM), and Business Models (BM). UML notion was created by Grady Booch, James Rumbaugh, and Ivar Jacobson and has been evolving since the second half of the 1990s (Hunt, 2003). 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.

1.2 Purpose

In Object Oriented Analysis (OOA) phase, UML diagrams are the mostly used models to present a wider view of user's requirements. Although this phase is generally considered as a manual task, literature survey show 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 morphology of English is simple and regular. On the other hand, analyzing of 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 analyses the requirements written in Turkish and automatically generates UML class diagrams is proposed and to the best of our knowledge, it is the first study in literature. Design components of a class (classes, attributes, entities and relationships) are extracted from textual requirements utilizing Natural Language Processing (NLP) methods such tokenization and part of speech (POS) tagging.

1.3 Contributions of this Thesis

The main contributions of this thesis are:

- It is the first study carried out on Turkish. To provide this contribution a novel comprehensive rule-based model involving twenty-six transformation rules is developed for Turkish.
- Considering the literature, there is no common and publicly available dataset for any language in order to be used in the experimental work of other researchers. As the second contribution of study, we have prepared well-defined dataset containing software requirements both in Turkish and English and made it publicly available on GitHub.
- Studies in the literature perform evaluation with commonly used measures such as precision, recall, and F-measure (Bozyiğit, Aktaş, & Kılınç, 2019). These measures assume each evaluation criterion (classes, attributes, methods, relationship types, etc.) has the equal weight that may cause inconsistent evaluation results. This is because the evaluation phase is highly dependent on personal opinions, and so priority/weight of these

criteria varies depending on views of users. In our study, a novel evaluation method based on Multi Criteria Decision Making (MCDM) is proposed.

1.4 Organization of the Thesis

This paper is organized as follows:

In Chapter 2, existing automatic concept identification studies in literature were presented and explained in detail to provide an overview of how OO model generated by using textual software requirements.

In Chapter 3, the principles of Object Oriented Design (OOD) are clarified in a detailed manner. Different types of OOD representations are illustrated with requirements, which are widely used in Software Engineering studies.

In Chapter 4, information about the methods used for implementing the concept identification studies for Turkish and English (NLP analysis models and transformation models) were explained in a detailed manner. Moreover, this chapter gave background information on transformation model including unique transformation rules and Turkish linguistic patterns.

In Chapter 5, the first part which analyze English requirements was experimented on ATM (Rumbaugh et al., 1990) case study. The evaluation results are presented to show the performance of the study in English requirements.

In Chapter 6, the second part which analyze English requirements was experimented on a case study. The experimental results are presented to show the performance of the study in requirements written in Turkish.

In Chapter 7, novel evaluation approach for the systems automatically generating OO conceptual model is conducted. Proposed method calculates weights of the

evaluation criteria by using Analytical Hierarchy Process (AHP), after specific criteria affecting the decision problem are determined.

In Chapter 8, the first part which performs on Turkish requirements was presented. Twenty different case studies were utilized by using a novel evaluation model including experts' view and MCDM methods. Lastly, obtained evaluation results were discussed in this chapter.

Finally, in Chapter 9, some concluding remarks and future directions were presented.

CHAPTER TWO
LITERATURE REVIEW

In this chapter, we systematically summarize the technical aspects and identify limitations of the recent studies, automatically implementing concept identification, using Systematic Literature Review (SLR) (Kitchenham, Dyba, & Jorgensen, 2004) framework makes our research processes more comprehensive. Thirty-nine papers are selected and reviewed with respect to their approaches, outputs, datasets, evaluation methods, and improvable points.

Considering the results of our systematic review, it is clearly seen that there are some gaps must be filled in the current studies. The revealed gaps in the reviewed studies are as follows;

- Majority of the reviewed studies are designed to analyze documents written in English, with a few exceptions (Montes, et al., 2008) and (Liu et al., 2004). Montes et al. perform concept identification on Spanish requirements and Liu et al. analyze on requirements written in German.
- Most of the studies create only UML class diagram as the conceptual model. Generating other types of diagrams beside class diagrams can make the current studies many-sided and user-friendly.
- There are many studies that have some limitations in determining OO design elements especially, relationships between the classes, interfaces, and abstract classes.
- It is realized that there is only one study (Bozyiğit et al., 2019) in the literature which specially created dataset including twenty software requirements in Turkish and English. The other studies in the literature use a few scenarios in English as dataset.
- All of the reviewed studies use standard evaluation metrics such as precision, recall, F-measure during analyze of performance of their approach and ignore the criteria affecting problem solution except (Bozyiğit et al., 2019).

The structure of this chapter is organized as follows: Section 2 gives information about the aim of the study and research procedures used in the SLR. Section 3 presents

evaluation results of the reviewed studies with respect to research questions. Section 4 concludes the paper and includes suggestions for further studies.

2.1 Research Methodology

A review paper is survey of existing literature on a topic in order to explain the current state of the topic (Gülpınar & Güçlü, 2013). Numerous review studies conflicting findings are published in academic platforms each year. These studies can use different research method considering the scope of review. Considering used research methods, review papers are examined under two categories: narrative reviews and SLR conducted by Kitchenham et al (2004). Narrative reviews evaluate the studies related to research topic in a wide spectrum. In the SLR, a detailed and comprehensive survey on particular research questions and subject area is performed (Brereton, Kitchenham, Budgen, Turner, & Khalil, 2007). It also it allows proof-gathering on the literature and provide guidelines for researchers on research trends and gaps.

Since SLR examines on the related studies in a more detailed and systematic way, we developed a research methodology in view of the guidelines Kitchenamn et al. (Kitchenham, 2004), Brereton et al. (Brereton et al., 2007), and Petersen et al. (Petersen et al.,2008).

The general architecture and functional blocks of our proposed SLR model is shown in Figure 2.1. The light blue rectangles in the figure demonstrates the main process in the SLR. The blank box represents the essential factors must be considered in the main processes of the SLR. The grey box illustrates the external tools used in the search process and dark blue rectangle shows the expected outputs obtained in the SLR.

Considering the Figure 2.1, it is seen that first of all, four research questions (RQ_1, RQ_2, RQ_3, and RQ_4 which will be explained in Section 2.1.1) are determined in order to build the framework of SLR. Secondly, we developed a search strategy that includes paper selection procedures (determining search terms based on research questions and creating search queries). Then, results of search process are filtered based on inclusion

and exclusion criteria. Finally, three quality assessment criteria are determined and then studies which do not match these criteria are eliminated.

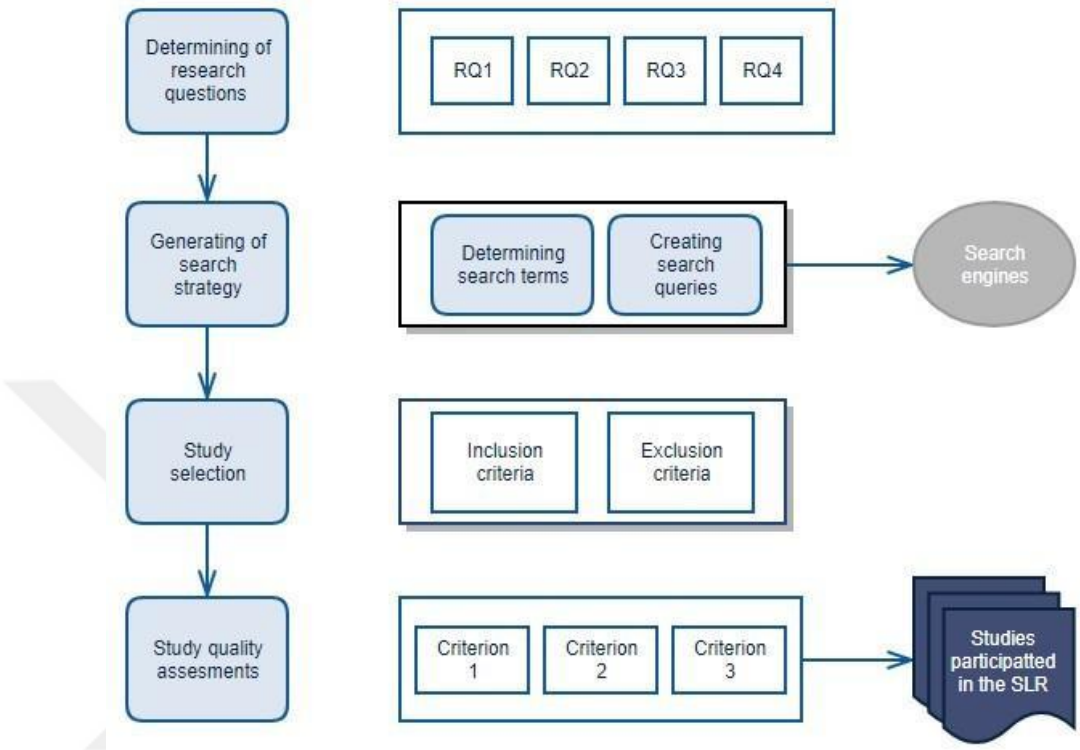

Figure 2.1 Search strategy in the literature

2.1.1 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 (Brereton et al., 2007).

In our study, research questions are determined in order to analyze and evaluate different approaches that used in automatic concept identification studies. These questions are listed as follows:

- Research Question 1 (RQ_1): What are the approaches used for transforming requirements into conceptual model?

- Research Question 2 (RQ_2): What kind of conceptual models are generated by the reviewed systems?
- Research Question 3 (RQ_3): What types of datasets are used for testing the performance of models in the reviewed studies?
- Research Question 4 (RQ_4): Which evaluation methods are used in the reviewed studies?

RQ_1 aims to identify the approaches of the studies in the SLR. According to answers to this question, it is observed which methods are proposed in the reviewed studies. Figure 2.2 gives information about the general architecture of the used approaches in the reviewed studies.

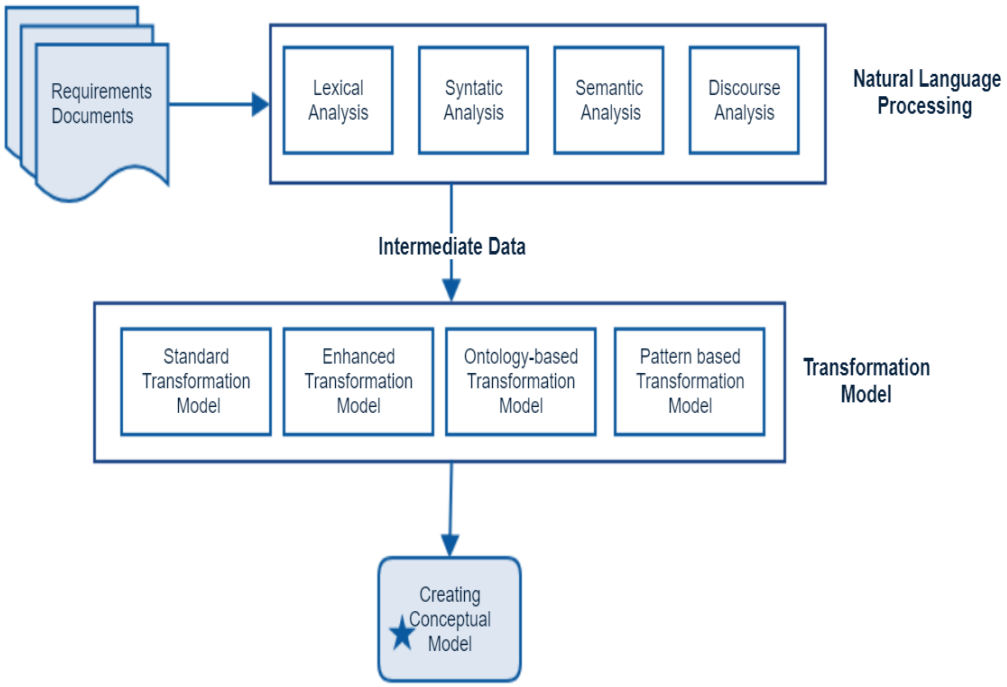

Figure 2.2 General framework of approaches in the reviewed concept identification studies

Then, it is discussed how the used methods affect the performance of the studies. This question is divided into the following sub-questions are as follows:

- Which Natural Language Processing (NLP) methods are implemented to obtain intermediate data from textual requirements? Figure 2.3 gives information about the general analysis models used in the reviewed studies.

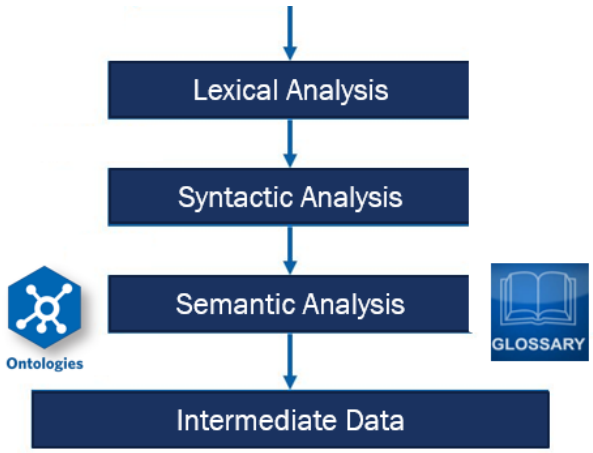

Figure 2.3 NLP analysis methods used in the reviewed concept identification studies

- Which transformation approaches are applied to intermediate data for generating conceptual model?

RQ_2 identifies outputs of the studies generated using transformation processes. That is, types of extracted conceptual models (UML diagrams, program code, etc.) from requirements are determined. This question is divided into the following sub-questions are 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_3 investigates the datasets used in the reviewed studies. This question is divided into the following sub-questions are as follows:

- Do reviewed studies have a comprehensive dataset that includes many requirements documents?
- What is the language of textual requirements in the dataset?
- Is the dataset of the related study publicly available to be used by other researchers?

RQ_4 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.

2.2 Search Strategy

The search strategy is a manual search of specific digital databases. In this thesis, we selected well known digital databases (ACM Digital Library, IEEE Xplore, Springer Verlag, Google Scholar, ScienceDirect, Metapress, and Wiley InterScience) to find high quality papers. The search process is completed on the journal papers and conference proceedings from 1996 to 2019. To find the full text of these studies, determined queries are entered to selected digital databases. Obtained results are first checked whether including similar works (presented by same authors) or not. Then, the inclusion / exclusion and quality assessment criteria are applied to the search results, and the papers to be evaluated under the SLR are filtered finally.

2.2.1 Search Queries

In the search process step, firstly, the keywords related to the research topic area are determined. To create search queries, we specify and categorize the search terms based on research questions using the PICOC (Population, Intervention, Comparison, Outcome, Context) proposed by Brereton et al. (2007).

Population are the criteria that specify the domain of transforming requirements into conceptual model such as “requirements transformation”, “requirements analysis”, and “generating conceptual model”. The definition of criteria specifying

domain are intervention, comparison, outcome, and context. **Intervention** are the keywords that indicate approaches used for transforming requirements into conceptual model such as "NLP methods" 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. Also, we use context criteria as exclusion criteria to eliminate the irrelevant papers (Section 2.3). The search strings are derived from specified search terms including population, intervention and outcome as shown in Table 2.1

Table 2.1 Search terms used in the SLR

Search Terms
(requirements OR "user needs") AND (analyze OR transformation OR generation) AND ("conceptual model" OR "analysis model") (requirements OR "user needs") AND (analyze OR transformation OR generation) AND ("UML diagrams" OR "class diagrams" OR "sequence diagram", "activity diagram" OR "use-case diagrams" OR "object diagram" OR "source code" OR "program code" OR "validation model", "ER model") (generating OR extracting OR creating) AND ("conceptual model" OR "UML diagrams" OR "source code" OR "program code" OR "ER diagram") AND ("NLP methods" OR "rule-based model" OR ontology)

2.3 Study Selection Procedure

By using our search queries, fifty-nine papers are determined as input for the selection process. Selection of appropriate studies are completed by using the specified inclusion and exclusion criteria. In respect to this, first, fifty-one studies providing inclusion criteria are selected. Then, four of the selected studies that include at least one of exclusion criteria are eliminated. Finally, forty-seven studies, which are related to our research scope, are identified.

Inclusion criteria are listed as follows:

- Studies in journal and conferences in the field of computer science, software engineering, information systems, and natural language processing,
- Journal articles and conference proceedings published between 1996 and 2019,
- Paper full versions.

Exclusion criteria are listed as follows;

- Presentations, workshops papers, informal papers, and tools not based on scientific study,
- Duplicate papers of same study,
- Paper that does not mention concept identification model in title and abstract content.

2.4 Quality Assessments

The quality assessment process is used for interpretation of findings and determining the power of detailed investigations (Kitchenham et al., 2004). The quality of each selected study is evaluated according to the criteria shown as follows:

- Quality Assessment Criteria 1: Is the aim of the study explained clearly?
- Quality Assessment Criteria 2: Are the used methods in selected studies explained?
- Quality Assessment Criteria 3: Is the output of study supported by concept identification model?

To sum up, Figure 2.4 shows the steps of filtering process in our SLR methodology. Firstly, fifty-nine papers are extracted regarding the results of search queries on academic search engines. Then, filtering is performed with respect to inclusion and exclusion criteria and so forty-seven papers are selected in this step. At the last step, it

is determined under specified quality assessment criteria whether the selected studies are appropriate to be evaluated in this work. Finally, thirty-nine papers are selected to be evaluated in this study.

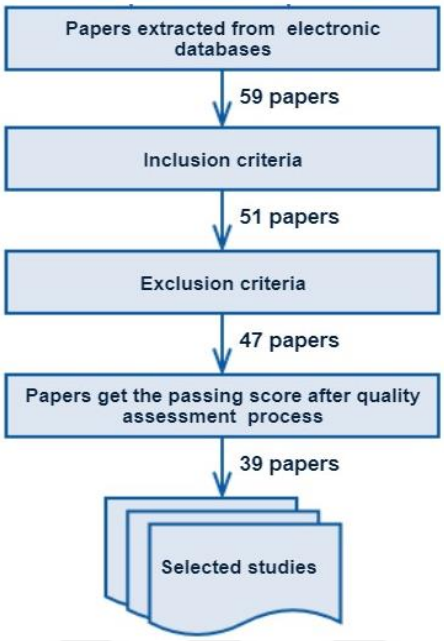

Figure 2.4 Papers screening process from search execution to quality assessment

2.5 Analysis Results

2.5.1 Overview of the Reviewed Studies

This section presents the descriptive results of the SLR. When extracting data from the papers and classifying it, the year of publication is a useful information to understand how the research on the automatic concept identification is active in software engineering area. When the literature review is conducted, it is observed that concept identification studies were initiated in 1996 with a tool called LOLITA (Mich, 1996). Since 1996, almost every year researchers in many different countries have presented papers dealing with the automatic concept identification from software requirements. This indicates that the automatic concept identification issue is gaining popularity in the field of Software Engineering and is worth investigating and developing by researchers. According to our research, it is observed that the countries with the most submitted concept identification studies are Canada and India. It is seen

that Turkey has the least number of publications with only two articles among twenty-three countries.

2.5.2 Discussion

2.5.2.1 RQ 1-Which languages are supported by the reviewed systems?

It is much more difficult to perform knowledge extraction for morphologically rich languages (MRLs) comparing to languages frequently studied within the scope of NLP such as English. Agglutinative languages (Turkish, Finnish, Hungarian, etc.) is exemplary for this since having complex morphology. For example, a sentence consisting of many words in English can be expressed with only one word in Turkish ("Are you one of those people whom we could not make to be Czechoslovakian (English)", "Çekoslavakyalılaştıramadıklarımızdan mısınız? (Turkish)"). Another feature of the agglutinative languages is that the order of grammatical items in the sentence is not determined according to any rules. Therefore, specification of the relationships between the items in a sentence (POS tagging) can be a challenging task for the agglutinative languages such as Turkish comparing to other morphological typologies such as English in which the constituents of a clause have an ordered structure. Moreover, the amount and variety of misspellings can increase especially in the MLRs since having a great number of affixes and derivational morphemes. Considering these difficulties in the MLRs, it is common to think that the current NLP tools used for linguistic analysis do not perform effectively.

According to review of literature, it is seen that most of the tools are developed to analyze requirements in English. This is because English is one of the most spoken languages in the world and morphology of English is not complex comparing especially with agglutinative languages (as mentioned before in the previous paragraph). There is a few study supporting other languages such as (Montes et al., 2008) and (Liu et al., 2004). Montes et al. analyze Spanish requirements and Liu et al. work on requirements written in German.

Considering the language limitations in the current studies, we recommend creating a common rule-set, which supports to analyze requirements written in any language. In our study, we state that some transformation rules can be applied independently of language in concept identification studies. For example; each noun in the document is candidate for classes and attributes. Assume that we apply this rule for a sentence written in both Turkish and English as in Table 2.2.

Table 2.2 Example of common rule any concept identifier can include

Turkish	Her çalışan isim, yaş ve cinsiyet bilgilerine sahiptir. (Each employee has name, age, and gender information.) **Nouns:** çalışan (employee), isim (name), yaş (age), cinsiyet (gender), bilgi (information)
English	Each employee has name, age, and gender information. **Nouns:** employee, name, age, gender, information

As can be seen from the example given in Table 2.2, 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 rule-set for analyzing the textual requirements independently of the language provides motivation for our future work.

2.5.2.2 RQ 2-What are the different approaches used for transforming requirements into conceptual model?

The approaches in the studies transforming requirements into conceptual model 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 model and transformation model. First, requirements are pre-processed with NLP analysis models to obtain an intermediate data. Then, the intermediate data are inputted to the transformation model, and so OO conceptual model is generated automatically.

There are five NLP analysis models available for pre-processing of textual requirements: lexical analysis, syntactic analysis, semantic analysis, discourse analysis

and pragmatic analysis (Tayal, Raghuwanshi, & Malik, 2014). Lexical analysis identifies structure of words and phrases in the 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 (Tayal et al., 2014). POS tags are used for determining design elements of the OO conceptual model. For instance, the verbs in requirement text are candidates for methods in OO design model. Syntactic analysis determines whether the structure of 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 (Yamashita & Matsumoto, 2000).

Transformation model enable identification of classes, attributes, methods, relations, and other elements in OO design by using meaningful statements obtained with NLP analysis models. There are four types of transformation model 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 that are established from linguistic patterns and grammatical structures (Sagar & Abirami, 2014). For example, one of these rules in this model may be “All nouns in the requirements documents are candidates for class and attribute names”. 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” (Sagar & Abirami, 2014). Ontology-based transformation model analyses the textual requirements with respect to the semantics of the application domain (Yue, Briand, & Labiche, 2010). Pattern-based transformation model incorporates specific pattern properties into a proposed model (Kaiya & Saeki, 2005).

Table 2.3 and Table 2.4 gives information about the used methods of the studies (from 1996 to 2019) in SLR. It is observed that most of the studies 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 is seen that there is a little study implements semantic analysis, which is critically task to understand the meaning of the text and extract necessary OO design elements. In fact, semantic analysis is not hard task in English, because there is a lexical database for the English language, WordNet (Miller, 1995), providing the short definitions and synonyms of the words. In WordNet architecture nouns, verbs, and adjectives are classified into synonym sets, each representing one underlying lexical concept. The main relation between statements in WordNet is synonymy. Synonyms is also named as synset. Each of 117 000 synsets defined in WordNet is linked to others by means of a small number of conceptual relations. For each synset, WordNet includes a short description (also known as gloss). Therefore, WordNet is enough beneficial to get semantic relationships between the words. Figure 2.5 illustrates an example of search on WordNet interface.

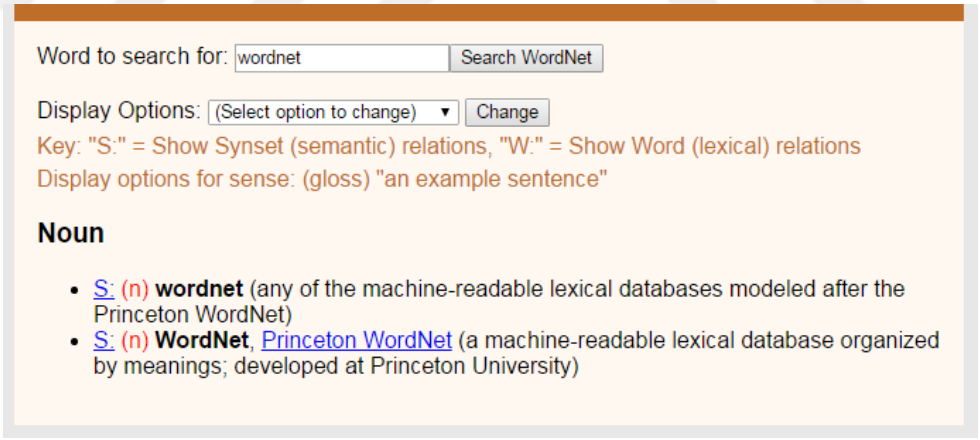

Figure 2.5 An example search in WordNet

Determining semantics of the text may be difficult for the other languages such as Turkish, because there is no comprehensive dictionary for textual analysis applications as WordNet. Thus, we recommend word embedding which is non-language related and widely used to make sense out of the textual data. We believe that performing

semantic analysis using word embedding model will improve the performance in term of accuracy of the generated model and running time. Also, there is only one study benefits from the pragmatic analysis expected to significantly contribute to improving the performance of the work and none of studies implements discourse analysis.

There is a numerous number of source to utilize specialized tasks of software engineering. However, some of the sources may be hard to reach to be used for researches, so designing a knowledge structure regarding the needs of planned system is necessary to increase the performance of the software projects. Considering this, we examine the selected papers within using ontology model and linguistic patterns. Evaluation results show that there is a small number of studies generate a specific set of rules including linguistic patterns and ontology model in their proposed approach. However, we foresee creating specific rule-set including detailed linguistic patterns and ontology will increase the performance of the semantic analysis phase of the studies. Figure 2.6 shows the infrastructure of semantic analysis including WordNet architecture and ontology model.

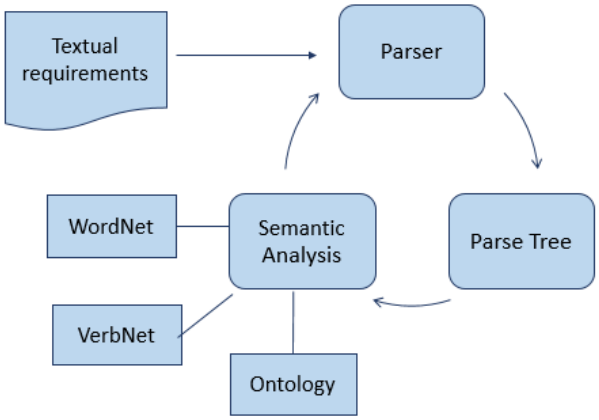

Figure 2.6 General architecture of NLP while analyzing requirements

Table 2.3 Outputs of the studies involved in the SLR (LexA: Lexical Analysis, SynA: Syntatic Analysis, SemA: Semantic Analysis, PrA: Pragmatic Analysis)

Paper Reference	Year	Requirement Representation	Processing method	Transformation Model	Output		
					UML diagram	Source Code	Validation Model
Sagar and Abirami	2014	Textual	LexA	Enhanced transformation	Class	×	×
Montes et al.	2012	Use-Case	SynA, SemA	Standard transformation	Class	×	×
Moreno	1998	Textual	None	Pattern based transformation	Object	×	×
Subramaniam et al.	2004	Textual	LexA	Enhanced transformation	Class, Use-Case	×	×
Mich	1996	Textual	LexA, SynA, SemA	Standard transformation	Object	×	×
Kaiya and Saeki	2005	Textual	PrA	Ontology based transformation	×	×	√
Capuchino et al.	2000	Textual	None	Enhanced transformation	Class	×	×
Overmyer et al.	2001	Textual	LexA, SynA	Standard transformation	Class	×	√
Wahono and Far	2002	Textual	LexA, SynA	Enhanced transformation	Object	×	×
Insfra'n et al.	2002	Textual/UseCase	None	Standard transformation	Class, Sequence	Visual Basic, Java,	×
Perez-Gonzales	2002	Textual	None	Standard transformation	Class, Sequence	×	×
Harmain	2003	Textual	LexA, SynA, SemA	Ontology-based transformation	Class	×	×
Liu	2003	Use case	LA	Enhanced transformation	Class	×	×
Salbrechter et al.	2004	Textual	LexA, SynA, SemA	Enhanced transformation	Activity	×	×
Cysneiros and Leite	2004	Textual	LexA, SemA	Pattern based transformation	Class, Sequence	×	×
Song et al.	2004	Textual	LexA	Enhanced transformation	Class	×	×
Zhou and Zhou	2004	Textual	LexA, SynA, SemA	Pattern based transformation	Class	×	×
Ambriola and Gervasi	2006	Textual	LexA, SynA, SemA	Enhanced transformation	Class, Sequence, Use-Case,	Pseudo code	×
El-Ghalayni et al.	2006	Domain Ontology	LexA, SemA	Ontology based transformation	Class	×	×

Table 2.4 Outputs of the studies involved in the SLR (LexA: Lexical Analysis, SynA: Syntatic Analysis, SemA: Semantic Analysis, PrA: Pragmatic Analysis)

Paper Reference	Year	Requirement Representation	Processing method	Transformation Model	Output		
					UML diagram	Source Code	Validation Model
Cardei et al.	2008	Textual	SemA	Ontology based, Enhanced transformation	×	×	√
Fatwanto and Boughan	2008	Use-Case	SynA	Pattern based transformation	Class	×	×
Giganto and Smith	2008	Use-Case	LexA, SemA	Enhanced transformation	Class	×	×
Seresht and Orndijeva	2008	Textual	None	Enhanced transformation	Domain, Use-Case	×	×
Popescu et al.	2008	Textual	LexA, SynA, SemA	Enhanced transformation	Object		×
Bajwa et al.	2009	Textual	LexA, SynA	Enhanced transformation	Class	Java, VB.	×
Mu et al.	2009	Textual	LexA, SemA	Enhanced transformation	×	×	√
Elbendak et al.	2011	Use-Case	LexA	Enhanced transformation	Class		×
Brambilla	2012	Textual	PrA	Pattern based transformation	Class	Java	×
Shinde et al.	2012	Textual	LexA,SemA	Enhanced transformation	Class, Sequence	Java	×
More and Phalnikar	2012	Textual	LexA, SynA, SemA	Ontology based, Standard transformation	Class	×	×
Deshpande and Joshi	2012	Textual	LexA, SynA, SemA	Pattern based transformation	Class	×	×
Herchi and Abdeselam	2012	Textual	LexA, SynA	Ontology based, Standard transformation	Class	×	×
Tripathy et al.	2014	Textual	LexA, SynA, SemA, PrA	Standard transformation	Class	Java	×
Landha¨ußer et al.	2014	Textual	LexA, SemA	Ontology based, Enhanced transformation	Class, Activity	×	×
Sharma et al.	2014	Textual	LexA, SynA	Pattern based transformation	Sequence, Activity	×	×
Arrelona et al.	2015	Textual	LexA	Ontology based transformation	×	×	√
Ibrahim and Ahmad	2015	Textual	LexA, SemA	Ontology based, Enhanced transformation	Class	×	×
Bozyiğit et al.	2016	Textual	LexA	Enhanced transformation	Class	Java, C#	×
Mu et al.	2009	Textual	LexA, SemA	Enhanced transformation	×	×	√

2.5.2.3 RQ 3- What kind of conceptual models can be generated by the reviewed systems?

Studies responding to this question are collected under three different categories. These categories are listed as follows:

- Studies generating UML diagrams,
- Studies generating source code,
- Studies generating validation model.

In the evaluation process of selected studies, it is easily realized that the conceptual models extracted from the reviewed approaches are generally in the form of UML. According to answers to RQ_2, it is realized that thirty-four out of thirty-nine selected studies aim to transform requirements into UML diagrams and only five of them focus on validation process. Distribution of the generated conceptual models in the reviewed studies can be shown as Figure 2.7.

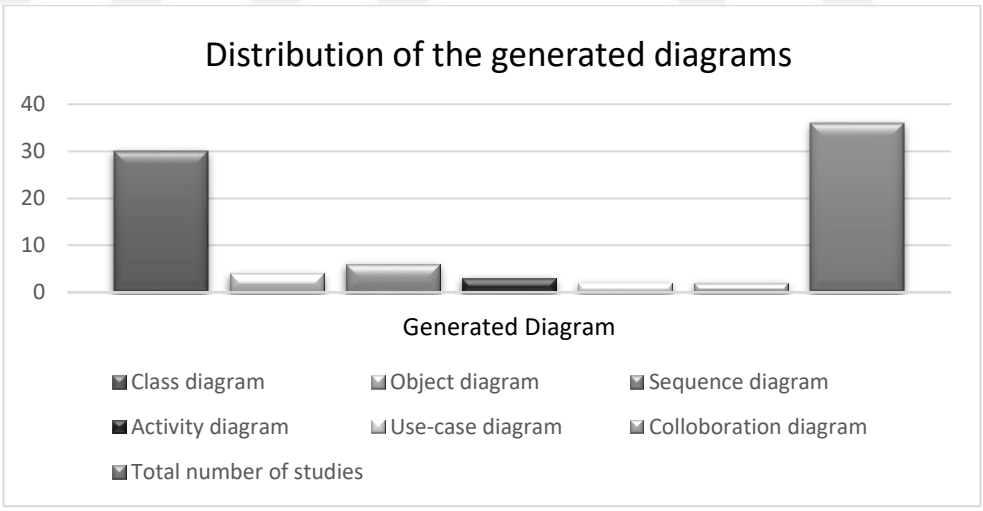

Figure 2.7 Distribution of the outputs the studies in the SLR generate

The studies extracting UML diagrams are examined under six different sub-categories in terms of the diagram types. The categories are class, object, sequence, activity, use-case and collaborative diagrams. Type of diagrams generated in the

reviewed studies (from 2006 to 2019) is shown in Table 2.3 and 2.4. It is realized that most of the studies generate only class diagram and so it can be said that they are not many-sided and user-friendly. However, generation of other diagram types, which have similar architecture with class diagram, by using extracted design elements in the analysis phase is not difficult task. To make it clear, we illustrated class and object diagrams for the same software scenario (ATM model) as seen in Figure x. A class diagram shows what the objects in the system consist of and what they are capable of doing. In contrast, an object diagram shows how objects in your system are interacting with each other, and which states those objects contain when the program runs. Despite having a different target, both of the class and object diagrams uses the same design elements as you can see in the example (Figure 2.8).

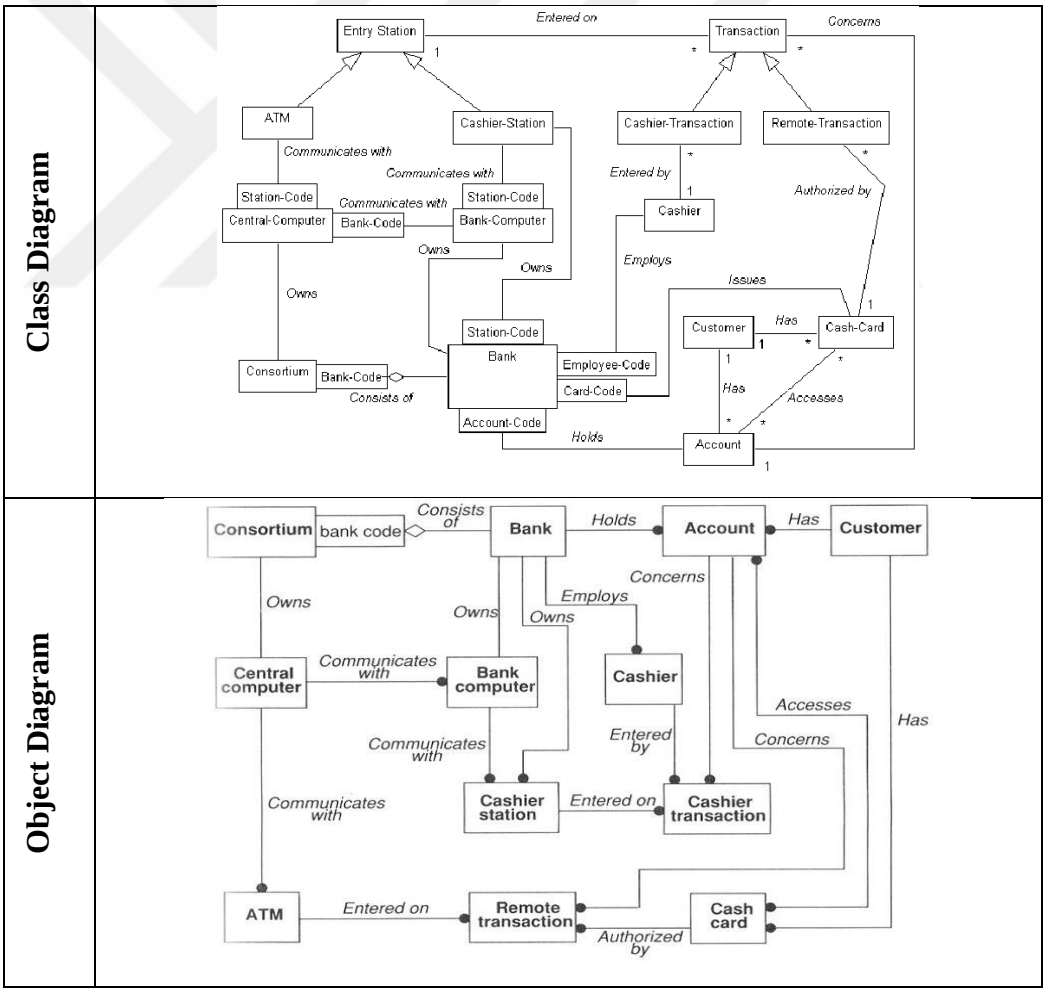

Figure 2.8 Comparison of object and class diagram of ATM problem statement

Additionally, it is observed that eight studies Wahono & Far (2002), Zhou and Zhou (2004), Popescu et al. (2007), Elbendak et al. (2011), Brambilla (2012), Herchi & Abdessalem (2012), and Bozyiğit et al. (2016) performing UML transformation also support code generation.

The main objective of the studies that support generation of UML diagram is to specify classes, attributes, methods, and relationships in the OO paradigm. When evaluating the studies generating UML diagrams, it is seen that majority of the reviewed approaches are successful in determining the classes, their respective attributes and methods. However, one of the important points to be considered in UML diagrams is to determine the relationships between classes. Generalization, aggregation, composition, and association are the examined relationship categories in the scope of SLR. An *association* relationship is established when two classes are connected to each other in any way. *Aggregation* is an association form, which express the one-way relationship between objects. Life cycles of objects are independent of each other. Related objects are not part of each other and there is an ownership relationship between them. *Composition* gives information about the interdependencies of the objects. An object may not be used independently of the related object; in this case, we can say that there is a composition relationship between them. *Generalization* is a relationship in which one model element (the child) is based on another model element (the parent). It is used in class, component, deployment, and use-case diagrams to show that the child inherits all of the attributes, methods, and relationships that are specified in the parent. A *realization* is a relationship between two model elements, in which one model element (the client) realizes the behavior that the other model element (the supplier) defines. When the related studies are examined, it is concluded that most of them have the limitations in specifying types of relationship types. It is realized that only twenty-two of the reviewed works are available to discover relationships between classes. In addition, only nine out of twenty-two studies that detect relationship can perform relationship type determination. Table 2.5 gives information on the studies that can determine relationships.

Table 2.5 Relationship types in the generated models (Abst: Abstraction, Agg: Aggregation, Assoc: Association, Comp: Composition, Gen: Generalization, Spec: Specialization)

Reference	Relationship type						Many to many
	Abst	**Agg**	**Assoc**	**Comp**	**Gen**	**Spec**	
(Sagar & Abirami, 2014)	×	√	√	√	√	×	×
(Moreno et al., 1970)	×	×	√	√	√	×	√
(Kaiya & Saeki, 2010)	×	√	√	×	√	×	×
(Overmyer et al., 2000)	×	×	√	×	√	×	√
(Wahono & Far, 2001)	×	×	×	×	√	×	×
(Insfra'n et al., 2002)	×	√	×	×	√	×	√
(Harmain, 2002)	×	√	√	×	×	×	√
(Liu, 2003)	×	×	√	×	√	×	×
(Song et al., 2004)	×	√	√	×	√	×	√
(Zhou & Zhou, 2007)	×	×	×	×	×	×	√
(Cardei et al., 2007)	×	×	√	×	×	×	×
(Fatwanto & Bough, 2008)	×	×	√	×	×	×	√
(Seresht & Ormandi, 2008)	×	√	√	×	√	×	√
Popescu et al. 2008)	×	√	√	×	×	×	×
(Bajwa et al. 2007)	×	×	√	×	×	×	×
(Elbendak et al., 2009)	×	√	√	√	√	×	√
(Brambilla, 2011)	×	×	√	×	√	×	√
(Shinde et al., 2012)	×	√	√	×	×	×	×
(More & Phalnikar, 2012)	×	√	√	×	√	×	×
(Herchi, 2012)	×	×	√	×	×	×	×
(Tripathy et al., 2012)	×	×	√	√	×	×	×
(Ibrahim & Ahmad, 2015)	×	√	√	√	√	×	×
(Bozyiğit et al., 2016)	×	×	√	×	×	×	×

2.5.2.4 RQ 4- Have any case study implemented to measure the performance of models in the reviewed studies?

The dataset is the most important input for testing the performance of the proposed model in scientific studies. The dataset must be well formed, well-formatted, and available to be used in other scientific studies.

When the studies covered in the research are examined respect to the RQ_3, it is observed that they have some limitations in the dataset used. These limitations are explained as follows:

- The datasets used include a few number of requirement documents.
- The requirements documents are used as input are simple structured and not exactly like real-life scenarios.

Dataset sharing allows researchers working in the same fields to compare their methods more objectively. It is realized that there is no study that uses a common dataset shared on public platform. Since detailed information about the data sets used in the studies cannot be accessed, only case studies scenarios are obtained. These case studies are shown in Table 2.6.

2.5.2.5 RQ 5- Have any case study implemented to measure the performance of models in the reviewed studies?

With recent development in the technology, studies implementing automatic transformation from software requirements into OO-based source code become widespread. The common aim of automatic transformation studies is creating a model to reduce workload of a software development and expedite the preparation phase of a software (Sagar & Abirami, 2014). When these studies are examined, it is seen that commonly used measurements such as precision, recall and F-measure are used for the evaluation process. By using these measurements, it is demonstrated whether a produced model is a reasonable representation of the actual system (Mu, Wang, & Guo, 2009). These measures consider evaluation factors (OO design elements) as if they have the same priority. However, it is necessary to consider that the specified criteria for the evaluation of source codes have different priorities. The priorities of these criteria vary depending on the views of decision makers who evaluate the system. When the studies are examined, it is seen that commonly used measurements such as precision, recall and F-measure are used for the evaluation process. Values of these measurements in the studies are shown in Table 2.6.

Table 2.6 Evaluation results of the reviewed works

Reference	**Case Scenario**	**Precision**	**Recall**	**F-Measure**
(Sagar and Abirami, 2014)	ATM Model	91.67	91.67	×
	EFP	85	94.44	×
	Course Registration	100	81.82	×
(Subramaniam et al., 2004)	Automated Teller Machine	×	×	×
(Wahono and Far, 2002)	Air Traffic Control System	×	×	×
(Yue et al., 2010)	Withdraw Cash	×	×	×
(Song et al., 2007)	Video Rental Store	×	×	×
(Ambriola and Gervasi, 2006)	An industrial case study	×	×	×
(Fatwanto and Bough 2008)	Voter Tracking System	×	×	×
(Seresht and Ormandi, 2008)	Invoicing Order System	×	×	×
(Popescu et al., 2007)	Elevator			
	Monterey Workshop Airport Security	×	88.46	×
(Elbendak et al., 2011)	ATM Model	82	58	×
(Brambilla, 2012)	Social Media data (Comments, chats, likes on social media)	×	×	×
(Tripathy et al., 2014)	ATM Model	100	93	×
(Landha¨ußer et al., 2014)	Modal window	×	×	×
	Musical Store	×	×	×
	Circe	×	×	×
	Monitoring Pressure	×	45.83	×
	ATM Model	×	×	×
	Steam Boiler	×	×	×
	ABC video rental	×	62.17	
	WHOIS protocol	×	×	×
	Cinema	×	×	×
	Timbered House	×	×	×
(Ibrahim and Ahmad, 2010)	Library System	×	×	×
(Bozyiğit et al., 2016)	ATM Model	100	93.5	93

Precision is the 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 obtained by calculating harmonic mean of the precision and recall. By using these measurements, it is demonstrated whether a produced model is

a reasonable representation of the actual system (Bozyiğit, Aktaş, & Kılınç, 2017). 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 inconsistent evaluation results. This is because the evaluation phase is highly dependent on personal opinions, and so priority of these criteria varies depending on views of users.

In this thesis, we suggest a novel evaluation method to evaluate of the studies transforming requirements into OO conceptual model. According to this, OO models must be evaluated by considering expert opinions and weighting of these criteria. The weights of evaluation criteria can be determined by using MCDM tools which are widely used in various decision-making problems. We claim that using MCDM methods including expert opinions much possibly provide more realistic and consistent evaluation results in concept identification studies.

2.6 Discussion about SLR Results

Transforming requirements into OO conceptual model is a vital but challenging task in software development. Although mostly done manually, there are available approaches to automate this step of Software Development Life Cycle. Nevertheless, it seems that there is not a practical, feasible automated solution despite the significant amount of research. In this SLR, the selected thirty-nine primary studies implementing concept identification on requirement documents are reviewed and evaluated by taking into consideration approaches, functionalities, outputs presented, the dataset used, and methods evaluated.

According to the analysis and evaluation of each approach in the studies examined, it is determined that there are some issues to be improved in the literature. Firstly, a clear majority of the reviewed studies deal with English requirement documents to generate conceptual model. The increase in the number of innovative works analyzing the documents in English language provides important contributions to this field. However, working on the other languages enables such systems to reach more users

and become global. Secondly, available approaches generally touch on generating the only class diagram of UML model. Other types of conceptual models and UML diagrams can be generated from requirements by extending current studies. Additionally, relationship identification between classes is not completed properly in existing works and they are weak in specifying relationship types correctly.

The dataset is the most important input to test the performance of the proposed model and it must be publicly available to be used in other studies. It is seen that all studies surveyed have a dataset containing a small number of documents and there is no shared dataset that is publicly available on the online repositories. This is definitely a gap that needs to be filled.

The performance evaluation of concept identification studies is a challenge because there is no definition of an accurate conceptual model. It is possible that two different people differently evaluate the same requirements document, because the priorities of evaluation criteria can be varied from person to person. However, it is seen that the studies reviewed consider that the evaluation criteria have the same priorities, and do not include expert opinions. This approach can lead to inconsistent results in evaluation of the studies. For this reason, we suggest to use MCDM methods that enable determining common weights of evaluation criteria with respect to expert opinions.

To sum up, recommendations for further works and a desirable approach are outlined in this study. As our future work, it is aimed to design a novel system, which extends the previous studies, by the following functionalities:

- Specifying all types of relationships completely,
- Extracting more diagram types beside class diagrams,
- Generating source code for more than one programming language,
- Creating and using a large-scale dataset that includes various requirement examples to test study,
- Using MCDM methods including expert opinion in the performance evaluation

CHAPTER THREE
OBJECT ORIENTED PARADIGM

Software requirements can be expressed in different ways. If a requirements document includes high-level statements, it cannot be understood clearly by the all stakeholders of project, so communication problems occur between customers and software team. Consequently, it is important to set a conceptual model, which is easy to understand, and provides more information compared to textual requirements.

A conceptual model is based on OO paradigm that enables software engineers to think about problems in terms of classes, their instances (objects), and interactions between the objects. Therefore, OO based approaches allow people to think programming elements as real life objects and ease the control of work flow in software projects. A conceptual model can be represented in various forms, such as Unified Modelling Language (UML) diagrams, Entity Relationship Models (ERM), and Business Model (BM). UML diagram is widely used in software engineering domain from initial to the end of the planned software project. In this chapter, different forms of UML models are presented briefly.

3.1 Object Oriented Design

OO-based design is a system which has specific characteristics and behaviors including class structures and relationships between the classes. Attributes of classes hold values of properties. Behaviors in class, also called methods, describe what can be done to an object/class. While attributes and operations give information about semantic of classes, relationships between classes (composition, aggregation, dependency, etc.) give clue about semantic of all systems.

3.2 Unified Modelling Language (UML)

With the recent development in the technology, hardware and software systems are interlaced and the huge network designs occurred. This situation makes the systems

more complex, so programming becomes a challenging task. Therefore, using standard models instead of textual information about a planned design is an essential process of SDLC.

UML notion was created by Grady Booch, James Rumbaugh, and Ivar Jacobson and has been evolving since the second half of the 1990s. It is a language to create common model, which is easy to understand by all stakeholders participating in the software project.

Figure 3.1 The logo of UML

In Object Oriented Analysis (OOA) phase, UML diagrams are the mostly used models to present a wider view of user's requirements. If a requirements of a planned system includes complex statements, it may not be understood definitely by software team and causes serious bugs to be fixed in next phases. These bugs also extend the delivery time of software and increase the total cost of the project. Therefore, it is important to transform textual requirements to UML models, which increase the understanding of the needs of users.

In UML models, big projects are divided into smaller components, which are related to each other. For example, customers, bank personnel, and ATM machine are assumed the different elements of ATM system. Customers use ATM machine to withdraw cash and the bank personnel load money.

UML has four-teen types of diagrams to model software systems and business processes, and all the diagrams are grouped into three categories; structural diagrams, behavioral diagrams, and activity diagrams as illustrated in Figure 3.2.

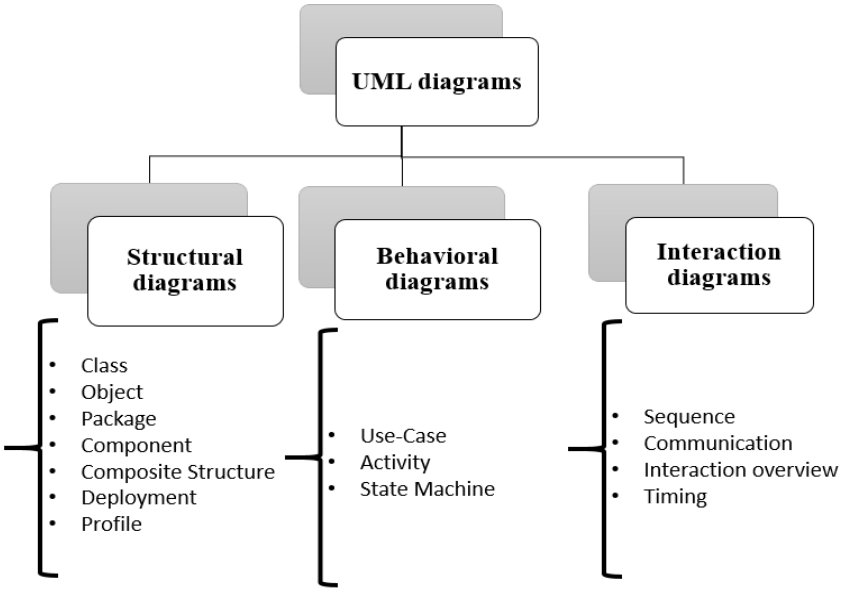

Figure 3.2 UML diagram types

3.2.1 Structural Diagram

3.2.1.1 Class Diagram

Class diagram is the most widely used UML diagram type and aims to clearly represent class structures and relationships between the classes in OO design. One of the areas where class diagrams are used most efficiently is OO programming. A class diagrams has mainly three components such as classes, attributes, methods, and relationships between classes. Attributes of classes hold values of properties. Methods in class, also called behaviors, describe what can be done to an object/class. Relationships shows the dependencies of each classes with the other ones. An example of class diagram can be seen in Figure 3.3. The elements illustrated with rectangles represent the class structure and the arrows placed between the rectangles imply the relationships. The first part in the rectangle includes the class name, the second section shows the attributes, and the third section give information the methods that instance

of the class can perform. As shown in the figure, bank, customer, teller, account, etc. are assumed as the different classes of a bank automation. Customers can withdraw cash from his/her bank account and the bank personnel open or close account for customers.

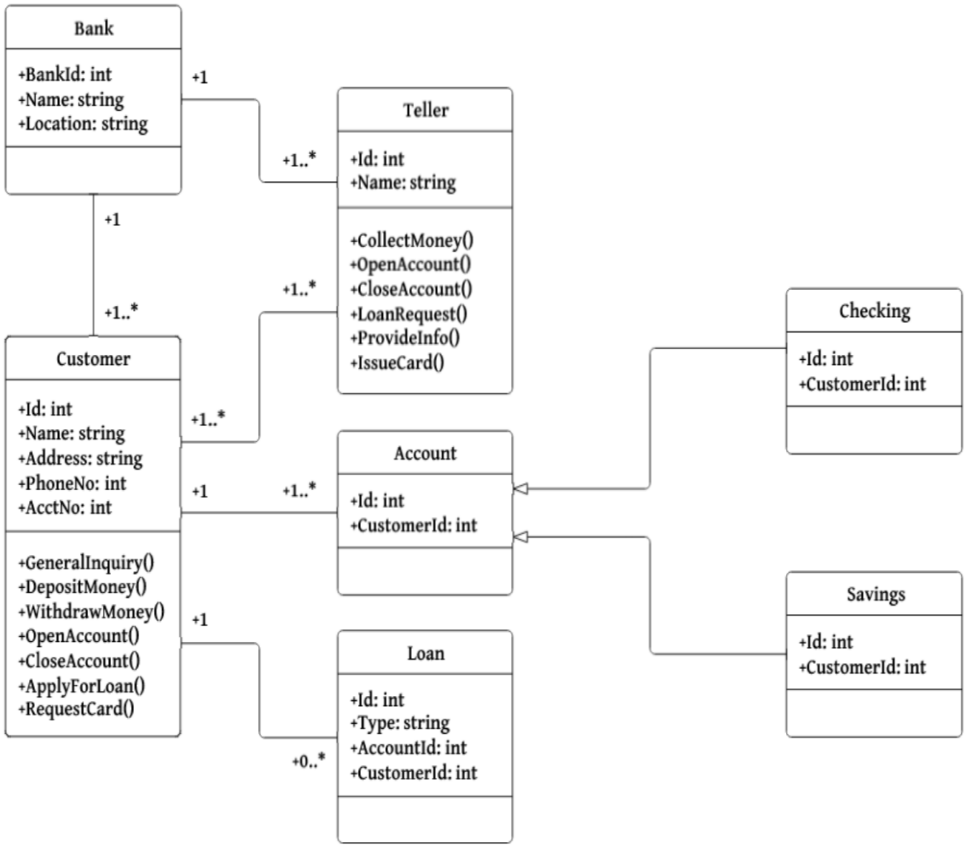

Figure 3.3 An example of class diagram designed for ATM problem

There are various types of relationships that can be used in the class diagrams such as association, composition, aggregation, generalization, and realization.

An *association* relationship is established when two classes are connected to each other in any way. In UML an association relationship is represented by a single arrow. An association relationship can be represented as one-to-one, one-to-many, or many-to-many (also known as cardinality). Assume that we have requirements sentence like that “Users login to the system.” If we consider the words “users” and “system” as

classes in the planned model, we can link them with an association relationship as illustrated with Figure 3.4.

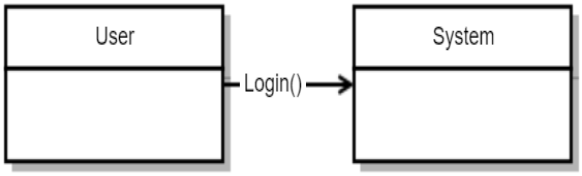

Figure 3.4 An example of association relationship

Aggregation is an association form which express the one-way relationship between objects. Life-cycles of objects are independent of each other. Related objects are not part of each other and there is an ownership relationship between them. It is also called "has a" relationship. Assume that there is such a relationship between user and computer. Both of "user" and "computer" can be considered as exactly separate parts of the system and existences of classes can continue even when the relationship between them is removed. Aggregation relationship has the following characteristics:

- It is a semantically week relationship,
- It is a "has a" relationship,
- If a part is deleted, it's relative can continue its existence.

Aggregation is expressed as a binary association and illustrated with a filled blank diamond as seen in Figure 3.5.

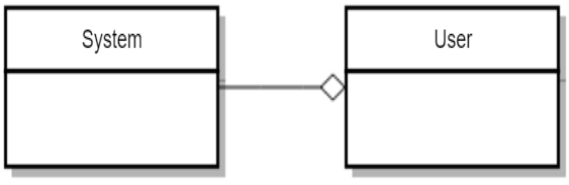

Figure 3.5 An example of aggregation relationship

Composition gives information about the interdependencies of the objects. An object may not be used independently of the related object; in this case we can say that there is a composition relationship between them. This type of relationship can be also

names as “is-a-part-of”. Assume that we have requirements sentence like that “Users can login to the system by using password.” As we can understand from this sentence, the users can’t enter the systems without password and each password belongs to exactly one user. Accordingly, if the user is deleted, so is his password. That is to say there is a composite aggregation (composition) which is a “strong” form of aggregation between user and password classes. Composition relationship has the following characteristics:

- it is binary association,
- it is a whole/part relationship,
- a part could be included in at most one composite (whole) at a time,
- if a whole is deleted, all of its parts are deleted with it.

Composite aggregation is depicted as a binary association and illustrated with a filled black diamond as seen in Figure 3.6.

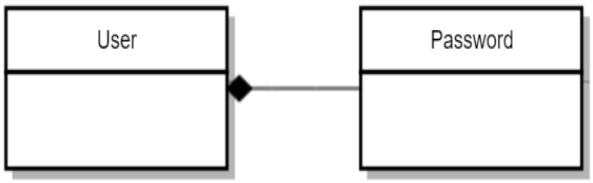

Figure 3.6 An example of composition relationship

Generalization is a relationship in which one model element (the child) is based on another model element (the parent). It is used in class, component, deployment, and use-case diagrams to show that the child inherits all of the attributes, methods, and relationships that are specified in the parent. The parent can have one or more children, and any child model element can have one or more parents. Generalization relationships can be used for taking attributes, behaviors, and relationships in a parent and then reuse them in one or more child. Because the child model elements in generalizations inherit the attributes, operations, and relationships of the parent, elements that are distinct from the parent must be also defined. For example, assume that there are to two authentications to login a website; administrator and customer.

Designer can bring these authentications together under same class such as user since they may have common attributes (id, name, surname, etc.,) as seen in Figure 3.7.

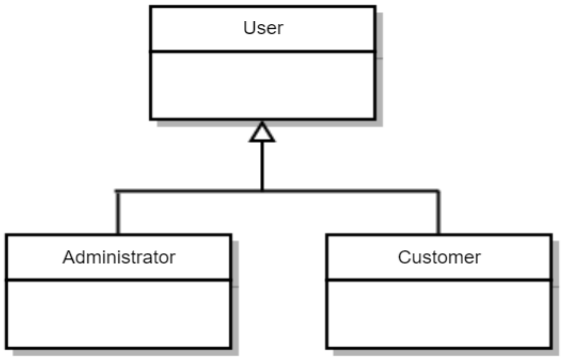

Figure 3.7 An example of generalization relationship

A *realization* is a relationship between two model elements, in which one model element (the client) realizes the behavior that the other model element (the supplier) defines. Numerous clients can realize the behavior of a single supplier. Realization relationships can be used in class diagrams and component diagrams. As in Figure 3.8 illustrates, a realization is displayed in the diagram editor as a dashed line with an unfilled arrowhead that points from the client (realizes the behavior) to the supplier (specifies the behavior).

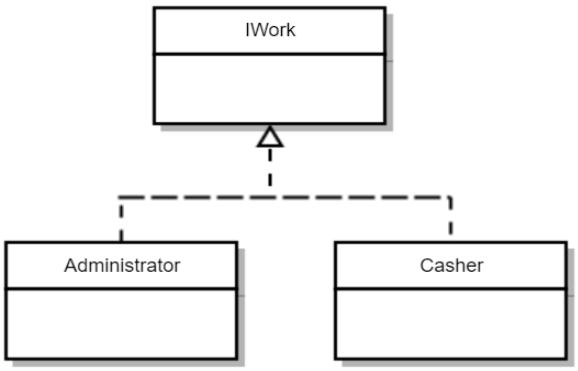

Figure 3.8 An example of realization relationship

3.1.1.2 Object Diagram

An *object diagram* is derived from class diagrams so it is built considering design elements in the relevant class diagram. The basic notions are similar for class diagrams and object diagrams. Object diagrams also represent the static view of a system at a particular moment. Object diagrams are used to provide a set of objects (instance of classes) and interactions between them. The difference is that a class diagram describes an abstract model including classes and their relationships. However, an object diagram represents an instance at a particular time. It means the object diagram is coherent with the functions of the actual system. The purpose is to obtain the static aspect of a system at a particular time.

3.1.1.3 Component Diagram

Component refers to a module of classes that serve as subsystems with the ability to interact with the rest of the system. The aim of a *component diagram* is to give information about the relationships between different components in a system. Component diagrams provide the planner to classify the different components so all parts of the system utilize the tasks as it is expected. More commonly, in an OO programming approach, the component diagram allows software developers to group classes together based on a common model so that the stakeholders can look at a software development project at a high level.

3.2.2 Behavioral Diagrams

3.2.2.1 Use-case diagram

A use case diagram is the initial form of software requirements for a new project being developed. Use cases illustrate the expected interaction between user and system. Concept of use case modeling helps the members of the software team to design a system from the end user's perspective. It is a useful way to show communication between user and system since specifying all externally visible system behavior.

The general characteristics of use case diagram is as following:

- It summarizes relationships between use cases, actors, and systems.
- It gives information about the order of steps a user performs to achieve the goal in the system.

Figure 3.9 shows a use case diagram example for a car selling system. As seen in the figure, there are three main actors (manager, sales person, and customer) and their behaviors in the system.

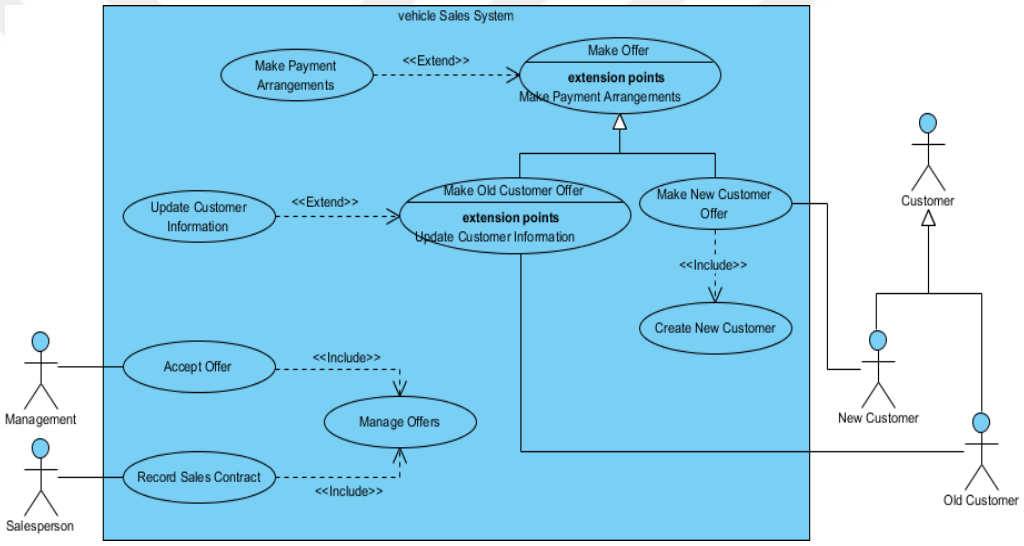

Figure 3.9 An example of use case diagram

3.2.2.2 State Machine Diagram

A state is an abstraction of the attribute values and relatives of an object. State machine diagram is used to illustrate state-dependent behavior for an object. An object reacts differently to the same issue in consideration of its state. State machine diagrams are applied to objects that have distinct behavior to other entities such as actors, use cases, methods, subsystems, and etc. There are several characteristics of states in state machine diagram as following:

- A state performs in an interval of time.
- A state is often related to an abstraction of attribute values of an object meeting some conditions.
- An object can change its state regarding both current input and the history of its input values.

Each state diagram typically starts with a dark circle that shows the initial and end states with a bordered circle that indicates the final state. They illustrate specific kinds of behavior that changes from one state to another.

State diagrams describe states and transitions. States are illustrated with rectangles with rounded corners. Transitions are indicated with arrows that pass from one state to another and give information about situations of states.

3.2.3 Structural Diagram

3.2.3.1 Sequence Diagrams

Sequence diagrams show how objects communicate with each other objects and utilize operations. They detail the interaction between objects in the context of a collaboration. These diagrams are based on time intervals of the defined operations. They demonstrate the order of the interaction between the object and the system.

In the diagram, messages (written with horizontal arrows) indicates the interaction between the objects. Thick arrowheads denote synchronized calls, blank arrowheads denote asynchronous messages, and dashed lines denote answers of the messages in the system. If an object sends a synchronous dialog, it must wait until the task is performed. However, when an object sends an asynchronous dialog, it doesn't have to wait for a response since processes are not interrupted. Method-calls are represented with opaque rectangles indicating that processes are being implemented in case response to the dialog.

3.2.3.2 Communication Diagrams

A communication (collaboration) diagram shows the interactions in the proposed model. It is mainly based on relationships between the instances of classes (objects). In these diagrams, objects are shown by using association connectors between the objects. Dialogs are joined to the associations and illustrated with short arrows heading in the direction of the communication.

CHAPTER FOUR
PROPOSED METHODOLOGY FOR TURKISH LANGUAGE

This chapter gives the background information about knowledge extraction from software requirements written in Turkish and automatic concept identification studies. It explains analysis of textual requirements with its methods; NLP techniques and rule-based model including domain ontologies and linguistic patterns.

4.1 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 (Prakash, Lucila & Wendy, 2011). The word structure in agglutinative languages is based on the addition of derivational or inflectional morphemes to the roots as suffixes. Since morpheme changes 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) (Aşlıyan, Günel & Filiz, 2006). Turkish as 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 to diagram transformation tool is a challenging task for Turkish.

4.2 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 (Rehman et al., 2013). NLP can be used in many disciplines, such as computer science, software engineering, computational linguistics, and so on. It enables to fill the gap between human expressions and artificial intelligence. The developments in information technologies have given a momentum to the studies dealing with natural languages in the literature. The basic NLP steps are tokenization, stemming, POS tagging etc.

4.2.1 Tokenization

One of the preliminary steps of text processing is tokenization, which is the process of separating sentence structure into word groups (Webster & Kit, 1992). To do this, the punctuation marks and spaces are considered as separators, and the sentences are separated into their components. In order to simplify the process of analysis extracting information from requirements documents, the tokenization is applied firstly and word sequences are obtained.

4.2.2 Stemming

The 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 in the text document are held and the inflectional suffixes are removed (Can et al., 2008). Derivational suffixes are used to derive words. The inflectional suffixes are added to stem of the name and verbs to specify the state, possession, plurality, time. The stemming process varies according to the language. It is possible to achieve a stemming system by looking only in the dictionary of suffixes in English. However, stemming is a difficult task in agglutinative languages such as Turkish, Finnish and Korean. Because, the sequence of inflectional suffixes can be added to the stem of a word.

After the text has been tokenized into words, it is cleaned from the inflectional morphemes through the stemming process and the next step, POS tagging, is applied in our proposed model. During the study, the stem of the word is accessed by clearing the word from the inflectional by removing them.

4.2.3 POS Tagging

The process of categorizing word groups considering their function in a sentence is called POS tagging (Straková, Straka, & Hajič, 2014). As the result of this labelling, each word is separated into categories such as name, verb, conjunction, etc. Table 4.1

shows an example of POS tagging. In this example, the sentence structure is first tokenized into words, and each word was marked according to their task in the sentence.

Table 4.1 An example for illustrating basic NLP methods

Part	**Description**
Sentence	Bütün cümleler öğelerine ayrılır (All sentences are parsed into tokens).
Tokens	bütün (all), cümleler (sentences), öğelerine (tokens), ayrılır (parsed)
Stems	bütün (all), cümle (sentence), öğe (token), ayrılmak (parse)
Pos tags	bütün (adjective), cümle (noun), öğe (noun), ayrılmak (verb)

4.3 Rule-based Model

Rule construction is an effective method to extract information from natural language texts. Rules are relied on human knowledge and expertise to find out candidate design elements in generated conceptual model. 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 under five different topics as seen in Table 4.2. The relationship rules have also three sub-categories, taking different types of relationships into account.

Table 4.2 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)	Aggregation	2
	Composition	4
	Generalization	4

To perform the information extraction task, the rules in the different categories are sequentially applied for each input sentence. First, each sentence is get 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, 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. Figure 4.1 shows the pseudo code algorithm of our rule-based model.

Input: Rs: Requirements**, GRS:** General rule-set**, CRS:** Class rule-set**, ARS:** Attribute rule-set**, RPRS:** Relationship pattern and rule-set

Output: OODE: OOD elements

Algorithm:

For each sentence *S* in Rs **Do**
  For each *GR* in GRS **Do**
    Execute GR_j on the S_i **Then**
    Find the deficient keywords which cannot be determined by NLP methods
  End For
  For each *CR* in CRS **Do**
    Execute CR_j on the S_i **Then**
    Find the names of classes in the diagram
  End For
  For each *AR* in ARS **Do**
    Execute AR_j on the S_i **Then**
    Find the names of corresponding attributes of each classes
  End For
  For each *MR* in ARS **Do**
    Execute MR_j on the S_i **Then**
    Find the names of corresponding methods of each classes
  End For
  For each *RPR* in RRS **Do**
    Match S_i with *RPRP* **Then**
    Define the relationship type between related classes
  Exit
  End For

Figure 4.1 Pseudo code algorithm of proposed rule-based model

4.3.1 General Rules

The aim of the rules in "General" category is to perform a general analysis and a pre-filtering process for the requirements documents. Considering the language limitations in the current studies, we recommend creating a common rule-set which supports to analyze requirements written in any language. In our study, we state that some transformation rules can be applied independently of language in concept identification studies. For example,

- Each noun in the document is candidate for classes and attributes.

Assume that we apply this rule for a sentence written in both Turkish and English as in Table 4.3. As can be seen from the example given in Table 4.3, 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 rule-set for analyzing the textual requirements independently of the language provides motivation for our future work.

Table 4.3 Example of general rule any concept identifier can include

English	Each employee has name, age, and gender information.
Turkish	Her çalışan isim, yaş ve cinsiyet bilgilerine sahiptir.
Nouns	employee (çalışan), name (isim), age (yaş), gender (cinsiyet), information (bilgi).

In this thesis, five rules are defined in general-rule category and some of them are explained with the use of examples.

General Rule #1: Nouns in sentences are candidates for class and attribute names.

General Rule #2: Proper nouns are removed from the candidate pool of classes and properties.

General Rule #3: Succession of the nouns in the sentences are aggregated and they are formed into single name, if the first noun has no affix.

Example: *ders kataloğu (course catalogue)* → *ders_katalog*

General Rule #4: Verbs in the sentences are included by the pool of methods' names.

General Rule #5: Succession of the verbs in the sentences are aggregated and they are formed into a single verb. This is a specific rule for Turkish.

Example: *işe almak (employ)* → *işe_almak*

To implement GR_5, 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 (Bozyiğit et al., 2019) which does not include auxiliary verbs such as "etkileşim sağlamak (interact)", "iletişim kurmak (communicate)", "veri yüklemek (load)", and so on.

4.3.2 Class Rules

In 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. Failure to include an entity or making an incorrect classification in the design can lead to major problems in the later phases of the software development process. Therefore, class extraction is one of the most critical step when modelling a target software product. The classes in the conceptual model must be in a correct and complete form. In this study, three rules are defined in addition to general rules for the identification of the classes' names. These rules, their explanations and examples are shown following.

Class Rule #1: The frequency of names above a certain threshold are labelled as classes.

Class Rule #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, bölüm(department)* → *class*

Class Rule #3: If the verbs such as "sahip olmak (have)", "içermek (include)", and "bulundurmak (contain)" exist in a sentence, the first name is labelled as a class.

Figure 4.2 presents an example in text format and it is processed using both general (GR_1, GR_3) and class (CR_3) rules.

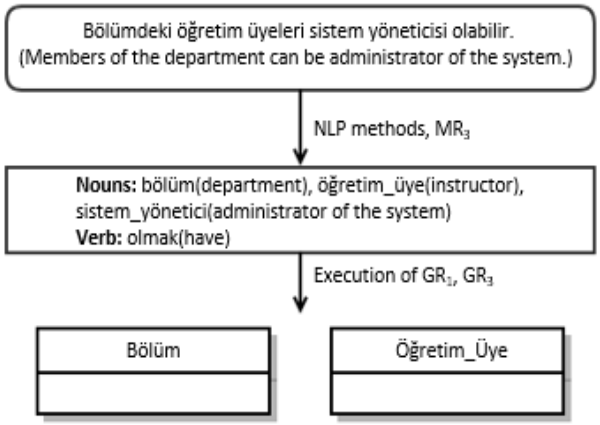

Figure 4.2 An example of processing general and class rules

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 below, there are five rules in the "Attribute" category.

Attribute Rule #1: Adjectives can provide information about the properties of a class.

Example: *Yeşil(Green) kart (card)*

renk(colour) is attribute of kart class.

To implement AR_1, system uses the list of adjectives which is provided by Türk Dil Kurumu (TDK) to be used in academic works. We created a new adjective list by adding definitions for some basic adjectives in the list of TDK (http://sozluk.gov.tr/, 2019). Thus, meanings of the basic adjectives such as color, number, shape, direction etc. are easily retrieved and used for specification of the attributes.

Attribute Rule #2: If there is a possessive construction in a sentence and the first name in the construction takes possessive or place suffixes, second name is pointed as attribute of the first name. This is a special rule for Turkish.

For example, there are many inflectional suffixes can be added to "okul (school)" and "öğrenci (student)" words 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 English language.

Consequently, all of the possessive constructions above give information about two design elements: "okul" is determined as class and "öğrenci" is specified as attribute of related class considering AR_2.

Attribute Rule #3: Object of the class derived from a noun can also be attributes of other classes extracted from the same sentence.

Example: *Mağaza asistanı, galerideki arabaların plaka, model, kiralama ücretini sisteme (system) kaydeder. (Store assistant record the information of cars in the gallery such as plate, model and renting price.).*

Assume that the given example is a sentence in a requirements including needs of a rented car gallery system. "Gallery" and "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 the "car" as attribute of "gallery" class as a result of AR_2. That is, "car" has both class and attribute labels in the system.

galeri → class, araba →class and attribute of "galery" class

Attribute Rule #4: If the 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.

Attribute Rule #5: Time, location, and percentage attributes of a class is retrieved by using according to Named Entity Recognition (NET) supported by the ITU NLP tool (Şeker & Eryiğit, 2017).

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. "Flight" is determined as classes, because frequency of it in the requirements text exceed a certain threshold value as stated in CR_1. Moreover, NET process retrieve the TIME entity from the sentence.

uçuş(flight) → class, zaman(time) → attribute of "uçuş" class

Figure 4.3 presents an example sentence processed using both general (GR_1, GR_5), class (CR_3), and attribute (AR_4) rules.

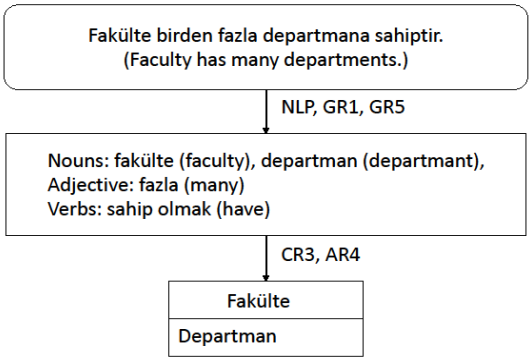

Figure 4.3 An example of executing general, class, and attribute rules

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 as following.

Method Rule #1: Each verb in documents is a candidate method, except the 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 (incude)”, and similar verbs listed in our repository (Bozyiğit et al., 2019).

Method Rule #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 Course Enrollment System. “öğrenci (student)” and “öğretim_üye (instructor)” are determined as classes, because their frequency in the requirements text exceed 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 both “öğrenci” and “öğretim_üye” classes.

öğrenci → $Class_1$, instructor → $Class_2$, anket_yapmak() → method of $Class_1$ and $Class_2$

Method Rule #3: The verb in the sentence having the class information is the method belonging to that class.

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 Turkish language by the authors of the study. Each sentence in the requirements is processed regarding to relationship rules and patterns. If input data is matched with a rule or pattern, relationship between two classes and its type are revealed. The defined rules and patterns in this study are split into three sub-categories to get Aggregation, Composition, and Generalization relationship types. These three sub-categories are described in Figure 4.4. Aggregation is a kind of association between two classes describing a part of relationship. Related classes in this type of relation are not affected if a container class is deleted. On the other hand, Composition relationship indicates a strict aggregation relation between the classes. If a container class is deleted all its classes are also needed to be deleted (Kim, Lu, & Lee, 2017). Generalization relationship is used to generate a derived class which inherits all elements in the parent class. The rules in the Generalization and Composition sub-categories are defined using pattern-based modelling. For relationships indicating generalization and composition, a list of patterns covering relevant cases is defined by the authors of study.

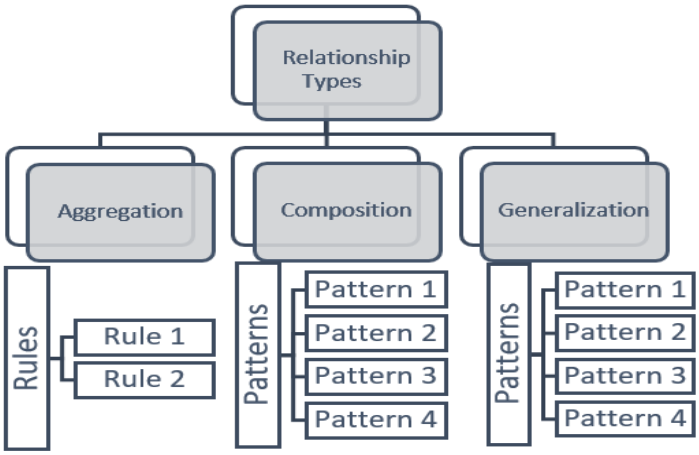

Figure 4.4 Sub categories in the relationship rule-set

4.3.5.1 Rules in the aggregation rule-set

Aggregation Rule #1: If all nouns are labelled as the class in a noun phrase, there is a certain relationship between them. This rule support both English and Turkish languages.

Example: *Banka'nın müşterileri (Customers of the bank)*.

There is an aggregation relation between Banka ($Class_1$) and Müşteri ($Class_2$) classes.

Aggregation Rule #2: If an attribute in a sentence is also labelled as a class, there is a relationship. This rule can be executed on requirements written in both English and Turkish.

Example: *The courses a student takes might open up a career opportunity as well as affecting his GPA.*

Assume that "course" and "student" are specified as $Class_1$, $Class_2$ respectively by executing GR_1. If $Class_1$ is also determined as attribute of $Class_2$, we can say that there is an aggregation relationship between "course" and "student" as shown with Figure 4.5.

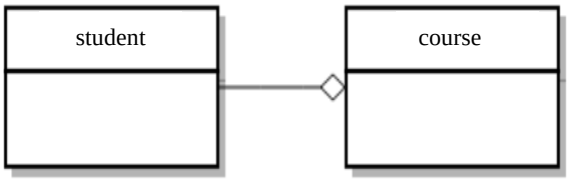

Figure 4.5 Aggregation relationship between student and course classes

4.3.5.2 Patterns in the composition pattern-set

- **Composition Pattern #1:** Bir ($Class_1$) birden fazla ($Class_2$) oluşmaktadır/içermektedir/bulundurmaktadır.
- **Composition Pattern #2**: ($Class_1$) bir tür ($Class_2$).

- **Composition Pattern #3:** ($Class_1$) ($Class_2$) parçasıdır/kısımıdır/elemanıdır/ oluşmaktadır/içermektedir.
- **Composition Pattern #4:** ($Class_1$) ($Class_2$) ait bir parçadır/kısımdır/bölümdür.

4.3.5.3 Patterns in the generalization pattern-set

- **Generalization Pattern #1:** Bir ($Class_1$) ($Class_2$) kategori yer almaktadır/dahildir/bulunmaktadır.
- **Generalization Pattern #2:** ($Class_1$) ($Class_2$)'dır/dir.
- **Generalization Pattern #3:** ($Class_1$) ($Class_2$) ait bir kategoridir.
- **Generalization Pattern #4:** ($Class_1$) ($Class_2$) bir altdalıdır/kategorisidir/alanıdır.

Figure 4.6 illustrates an example sentence processed by the pattern GP_1.

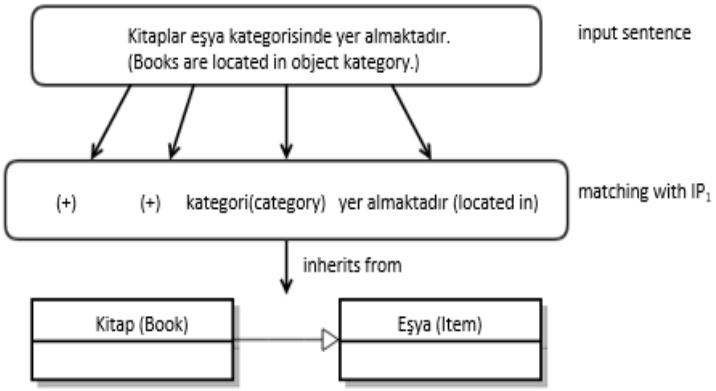

Figure 4.6 An illustration of matching a sentence with GP_1

CHAPTER FIVE
OBJECT ORIENTED ANALYSIS OF ENGLISH REQUIREMENTS

In this chapter, automatic concept identification is conducted for the linguistic aspects of the English language. The model proposed for analyzing English requirements consists of two parts; basic NLP methods (morphologic analysis) and rule-based model including ten rules. In this chapter, English software requirements are processed to build class diagram and generate C# program code with respect to OO paradigm. Considering the experimental results, it can be confidently stated that the performance results of proposed approach (designed for English) is more accurate than reviewed studies in terms of finding classes in OO design.

5.1 Proposed Model for Analyze English Requirements

5.1.1 NLP Methods Used in English Language

Tokenization is the identification of each “atomic” unit / word in a sentence (Habert et al., 1998). This process enables consisting of tokens arranged in a syntactically valid combination. So, in order to analyze a text document, tokenization must firstly be performed and groups of word must be obtained. Thus, it gets easy to make sense out of the textual documents.

After tokenization morphologic analysis is performed. First, the process of assignment grammatical tags to the word in a sentence is utilized. It is a semantic analysis approach that expresses grammatical rules and how a word is used in a sentence. So, intermediate data is obtained by using grammatical construct of the sentence. The relationships between tokens can also be identified by using POS tagging. POS tags include nouns, verbs, adverbs, adjectives, pronouns, conjunction and their sub-categories. An example of POS tagging is shown in Figure 5.1.

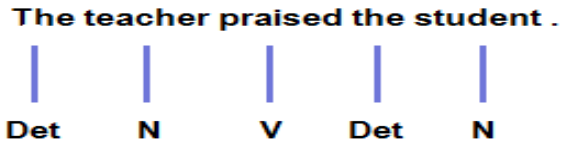

Figure 5.1 An example of POS tagging

A stemming algorithm is a computational procedure which reduces derived words to their base forms (Lovins, 1968). Derivational affixes and inflectional suffixes are removed and words are transformed into stem morph. For example, as seen in Figure 5.2, words like “computing” and “computed” are converted to “compute” after the stemming process. After POS tagging, Lancaster stemmer is used to stem tagged words (Paice & Hooper, 2005). Then, duplicate words are removed from database.

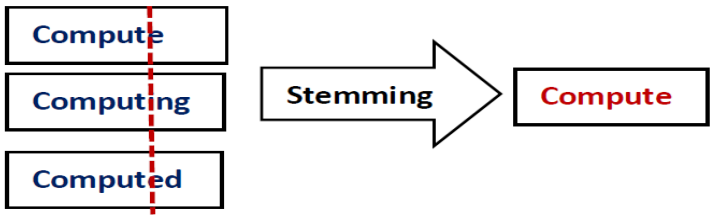

Figure 5.2 An example about stemming

In this study, we used Stanford Parser to obtain the POS tags of all words in documents (Manning et al., 2014). Words are classified into nouns, verbs, adjectives or adverbs and stored according to their assigned POS. Further, nouns are classified into proper and common nouns.

5.1.2 Proposed Rule-based Model for English Language

The extraction of design elements that belong to OO models is the base of concept identification model. In this study, class diagrams and source codes is generated by using elements (classes, attributes, methods and relations) in OO model. The aim of using a rule-based model is to specify the valid components of the conceptual model.

A set of rules is formed to create the rule-based model. While some rules are taken from reference's paper, some are created specifically by authors of this study. The rules are explained as following;

- Nouns in the sentences are candidate for class' and attribute's names.
- Proper nouns are cleaned from the textual data because they do not indicate any class or attribute.
- The nouns having the maximum frequency are favorite candidates for class names.
- Verbs in sentences are included by pool of methods' names.
- Linking verbs between two nouns indicate a relationship.
- Verbs included by phrasal verb inform about relationships.
  Example: "The family bring up a child."
  Family is $Class_1$, *bring up* is association relationship, *child* is $Class_2$.
- If a verb is included by the following list {include, involve, contain, consist of, etc.}, there may be aggregation and composition relationship type.
- Succession of the nouns in the sentences (if there are no elements between them except space) are aggregated and they are formed into single name.
  Example: "concept identification model" is formed as single noun "concept_identification_model".
- Subordinating conjunction such as "of" before a noun helps to find attributes of a class.
  Example: "Gender of student"
  Gender is attribute of *Student* class.
- Possessive endings such as 's after a noun helps to find attributes of a class.
  Example: "Student's ID"
  ID is attribute of *Student* class.

5.2 Experimental Study on English Requirements

The evaluation of the concept identification studies is a challenging task, because extracted design elements from requirements may be differ from the real results. In order to compare and obtain more accurate results, a particular use case study, which

is widely known in Software Engineering domain, is used. The case study is an ATM problem statement (Rumbaugh et al., 1991) which is used in the experimental works of Sagar and Abirami (Sagar & Abirami, 2014). Sagar and Abirami indicate that they generate the different versions of the ATM case study by rephrasing original text. Figure 5.3 shows the original version of ATM problem statement (Rumbaugh et al., 1991).

> The system must support a computerized banking network that includes both human cashiers and ATMs. The computerized banking network will be shared by a consortium of banks. Each bank provides a computer that maintains the bank's accounts and processes transactions against the accounts. Cashier stations are owned by individual banks and communicate directly with the bank's computers. Human cashiers enter the account data and transaction data. An ATM communicates with a central computer. The central computer clears transactions with the banks. An ATM accepts a cash card and interacts with the user. An ATM communicates with the central computer to carry out transactions. An ATM dispenses cash, and prints receipts. The system requires appropriate record-keeping and security provisions. The system must handle concurrent access to the same account correctly. The banks will provide the bank's own software for the bank's own computers.

Figure 5.3 ATM problem statement (Rumbaugh et al., 1991)

In the experiments, we compare the performance results of this part of the thesis (which is developed for analyze English requirements) with the results of Sagar and Abirami's work. Therefore, we also create three examples of requirement which are categorized as simple, average, and complex by modifying original ATM problem statement (Rumbaugh et al., 1991) to obtain more consistent comparison results Table 5.1 shows accuracy rates of all documents in class, attribute, method and relationship categories.

Table 5.1 Evaluation results of AutoClass

Document	Class	Attribute	Method	Relationship
Simple	100%	100%	80%	95%
Average	100%	100%	74%	89%
Complex	100%	100%	70%	78%

Number of correct identified design elements (classes, attributes, methods, and relationships) which are extracted in the thesis and Sagar's study (Sagar & Abirami, 2014) are compared in as illustrated in Figure 5.4. Considering the results of experimental study, it can be easily seen that the module developed for the thesis gives more accurate results than Sagar's work (Sagar & Abirami, 2014) in terms of determining classes and attributes.

Figure 5.4 Experimental results on ATM problem statement

CHAPTER SIX
PROPOSED EVALUATION MODEL

With recent development in the technology, studies implementing automatic transformation from software requirements into OO-based conceptual model become widespread (Lamsweerde, 2013). The common aim of automatic transformation studies is creating a model to reduce workload of a software development and expedite the preparation phase of a software (Budinsky, Finny, Vlissides, & Yu, 1996). When these studies are examined, it is seen that commonly used measurements such as precision, recall and F-measure are used for the evaluation process. By using these measurements, it is demonstrated whether a produced model is a reasonable representation of the actual system (Zelkowitz & Wallace, 1997). These measures consider evaluation factors (OO design elements) as if they have the same priority. However, it is necessary to consider that the specified criteria for the evaluation of source codes have different priorities. The priorities of these criteria vary depending on the views of decision makers who evaluate the system.

In this thesis, we propose Analytical Hierarchy Process (AHP) (Saaty, 1994) based validation of the studies, which utilize automatic OO-based source code transformation from textual requirements. According to this, OO-based source code documents are evaluated by considering weights of the criteria (classes, attributes, methods, relationship type, etc.) which can be varied depending on decision makers. The weights of evaluation criteria are determined by using AHP method, which is a well-known Multi Criteria Decision Making (MCDM) tool and widely used in various decision-making problems. The proposed approach is evaluated on a particular study, which is presented by Bozyiğit, Aktaş, and Kılınç (2016). As a result, it has been observed that the evaluation by using AHP gives more realistic results than the accuracy rate calculated with precision, recall and F-measure in the study of Bozyiğit et al. (2016).

The validation of studies implementing OO-based conceptual model generation checks whether the obtained code contains the keywords in the software requirement

document. These keywords are classes, attributes, methods, relationships, relationship type, etc. that should be included in OO design.

It is observed that similar approaches are utilized in the studies, which transform textual requirements into OO-based source codes. Figure 6.1 shows the general framework of these studies. First, the textual requirements are formatted by pre-processing step within certain constraints. Then, the source code generation is completed by compositing the model elements revealed by the transformation process. Finally, evaluation of proposed model is performed.

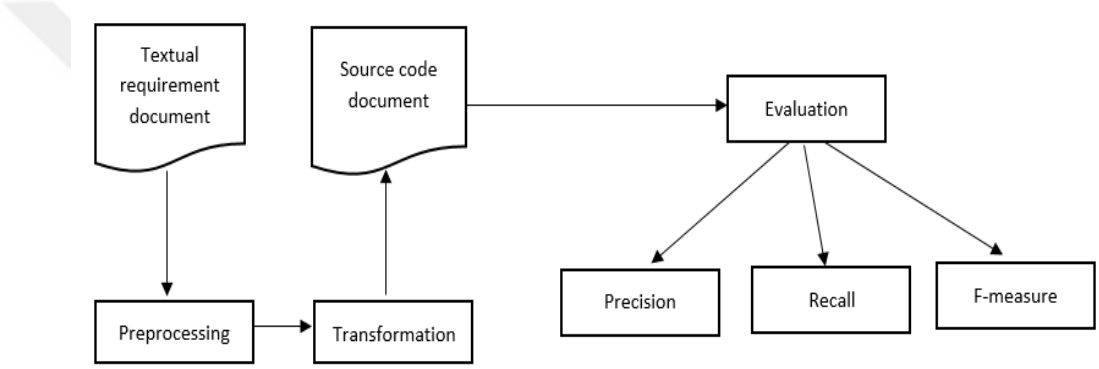

Figure 6.1 General architecture of reviewed approaches

6.1 Standard Evaluation Methods

It is observed that all of the studies discussed in the paper benefit from the precision, recall and F-measure parameters, which are the scoring concept of statistical science in the evaluation phase (Makhoul, 1999). The number of the classes, attributes, methods and relationships in the generated source code are compared with a standard model and missing model elements are determined. Accuracy of proposed model is calculated by taking into consideration the elements that are required, missing or excessive.

The precision is obtained by the ratio of the correct data to the total data, and it is calculated according to the equation given in Formula 1. TP (true positive) denotes the

number of objects that are correctly extracted by system. FP (false positive) refers to the number of objects that the system confirms as true when indeed it is not.

$$Precision\ (Pr) = \frac{TP}{TP + FP} \tag{6.1}$$

The recall metric is calculated by the ratio of the correct data to the expected correct data, it is given in Formula 6.2. FN (false negatives) in the equation refers to the number of correct data, which could not be found.

$$Recall\ (Re) = \frac{TP}{TP + FN} \tag{6.2}$$

In order to calculate the accuracy of the proposed model, the harmonic mean of the precision and recall values are obtained and the F-measure is calculated according to the equation given in Formula 6.3.

$$F_{measure} = \frac{2(Pr \times Re)}{Pr + Re} \tag{6.3}$$

Table 6.1 shows the evaluation results of a case study (one of the reviewed studies) conducted by Bozyiğit et al. (2016). The precision, recall and F-measure values, which are calculated based on the expected and obtained data as a result of the study, are shown in Table 6.2.

Table 6.1 Evaluation results of Bozyiğit et al. (2016)

	Class	**Relationship**	**Attribute**	**Method**
AutoClass	10	11	3	8
Expected values	10	13	3	9

Table 6.2 F-Measures of the evaluation criterıa in Bozyiğit et al. (2016)

	Class	**Relationship**	**Attribute**	**Method**
Precision	1	0.92	1	0.8
Recall	1	0.85	1	0.89
F-measure	1	0.88	1	0.84

The accuracy rate Bozyiğit's work (Bozyiğit et al., 2016) is calculated as 93% regarding the F-measure value for each criterion in Table 6.2. All criteria are considered equally weighted and arithmetic mean of all F-measure values is calculated as shown in Equation 6.4.

$$F_{measure} = \frac{Class + Relationship + Attribute + Method}{4}$$
$$F_{measure} = \frac{1 + 0.88 + 1 + 0.84}{4} \quad (6.4)$$
$$F_{measure} = 0.93$$

6.2 Analytical Hierarchy Process (AHP)

Analytical Hierarchy Process (AHP) is a mathematical notation of problem- solving that is widely used to make a decision in complex queries (Saaty, 1994). AHP method has been formed after getting the structure of a query and the challenges decision makers have to overcome.

The AHP method examines decision-making problems in three steps. The first step is the definition of the problem. The second step is the determine solutions alternatives to solve the problem. The most critical step is the specification of criteria used to evaluate the alternative solutions.

Although there are several criteria, the importance of each criterion may not be equal in multi-criteria decision-making problems. For example, if somebody has to make a decision to buy between two shoes, conform and price are two factors which can be comparative according to decision makers' perception. The comfort may be more essential than the price in buying behavior of anybody. Accordingly, he/she can score comfort criteria as two for the comfort and one for the price.

In the MCDM problems, the weights of each criterion must be calculated to ensure reaching the correct solution. Although this process is considered as easy task, decision-makers can be confused during the determination of alternatives' priority order which enable problem-solving.

6.3 AHP-based Evaluation Method

It is a challenging task to validate the studies, which utilize conceptual model transformation from requirement documents, because the evaluation phase is highly dependent on personal opinions and it is subjective in nature. Therefore, firstly, decision-making problem should be defined, and then factors affecting decision points and importance values of these factors should be determined (Triantaphyllou, 2010). In this case, AHP method that is created under MCDM would be so befitting. Thus, a novel evaluation approach is presented in this study, and it is demonstrated in Figure 6.2. The evaluation part, which includes AHP, has a novelty, because none of the studies in the literature generating source code from requirements uses criteria weighting process in the evaluation phase.

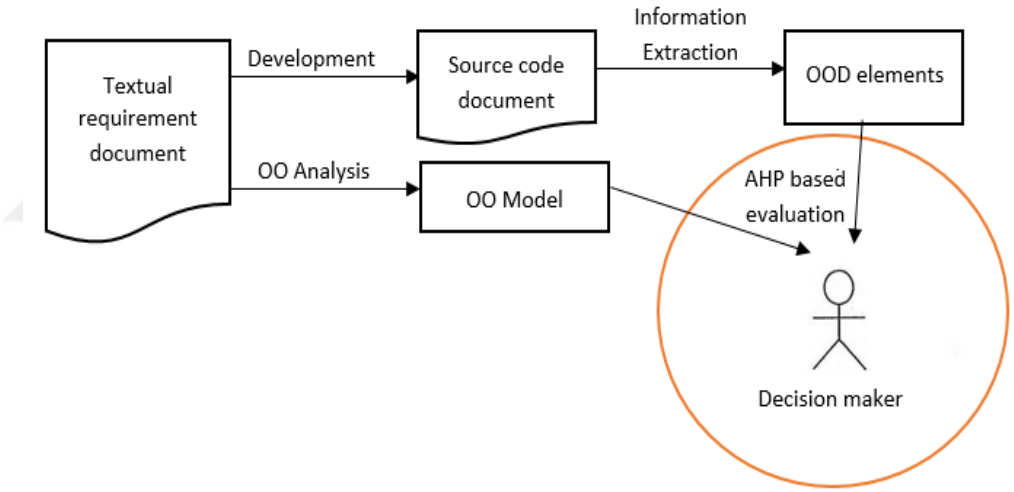

Figure 6.2 Proposed approach

In the literature, AHP is widely used tool for dealing with various decision-making problems. It enables the decision makers to define criteria and calculate weight for each criterion to make the best decision. AHP starts with the construction of a functional hierarchy to decompose complex systems into their basic parts according to their required relationships (Saaty, 2000). To determine weights of criteria by using AHP, there are four main steps to be done.

Step 1: The evaluation criteria are taken into consideration and alternatives are chosen to make the best decision.

- There are multiple criteria affecting the accuracy of the OO-based source code and the importance of the criteria is relative. The criteria specified by three decision makers participating in this study are as follows.
- Classes: Finding all classes in the OO-based model completely,
- Attributes: Finding set of attributes and placing them in the related classes,
- Methods: Finding set of methods and placing them in the related classes,
- Relationships: Finding all relationships between classes and organize them correctly,
- Relationship type (composition, aggregation, inheritance, association vb.): Defining relationships type inter classes correctly.
- Many to many relationship: Showing whether there are many to many relationships between the two classes.

Step 2: Comparison matrix of factors which is N×N square matrix A, is created (Nasiboğlu, Bozyiğit, & Diker, 2015).

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

In this step, decision makers are asked to compare the criteria at a given level on a pairwise basis in order to estimate their relative importance. A nine-point scale is commonly used to demonstrate the experts' preference between options as equal importance, moderate importance, strong importance, very strong importance, or extreme importance preferred. Table 6.3 shows the pairwise comparison of each criteria belongs to three decision makers participating in the study. It is assumed that the Criterion 1 is equally or more important than the Criterion 2.

Table 6.3 Importance of each criterion with respect to AHP priorities

#	Criterion 1	Criterion 2	Equal (1)	The importance value of Criterion 1 over Criterion 2							
				2	3	4	5	6	7	8	9
1	Class	Relationship				X					
2	Class	Attribute					X				
3	Class	Method					X				
4	Class	Relationship type							X		
5	Class	Many to many									X
6	Relationship	Attribute		X							
7	Relationship	Method		X							
8	Relationship	Relationship type				X					
9	Relationship	Many to many association						X			
10	Attributes	Method	X								
11	Attributes	Relationship type		X							
12	Attributes	Many to many				X					
13	Methods	Relationship type		X							
4	Methods	Many to Many				X					
15	Relationship type	Many to Many		X							

Considering the values in Table 6.3, $N \times N$ comparison matrix A is created. The diagonal elements of the comparison matrix are set to 1, because the relevant factor is compared with itself in this case. As seen in Table 3, the first criterion in the first row is four times important than second one according to the decision maker. Thus, the element in the first row and second column is set to 4 and the element in the second row and first column is set to $1/4$ in the matrix A according to Equation 6.5. Accordingly, the comparison matrix is showed in Table 6.4.

$$a_{ij} = \frac{1}{a_{ij}} \tag{6.5}$$

Table 6.4 Comparison Matrix A

	1	**2**	**3**	**4**	**5**	**6**
1	1	4.00	5.00	5.00	7.00	9.00
2	0.25	1	2.00	2.00	4.00	6.00
3	0.20	0.50	1	1.00	2.00	4.00
4	0.20	0.50	1.00	1	2.00	4.00
5	0.14	0.25	0.50	0.50	1	2.00
6	0.11	0.17	0.25	0.25	0.50	1

Step 3: AHP assigns a score to each option according to the decision makers' pairwise comparison with respect to the considered criterion.

The column vectors are determined before calculating the weights of all of these criteria; a column vector is denoted as B_i where *i* is from 1 to n (the number of criteria). They are represented as Equation 6.6.

$$B_i = [b_{11}\ b_{21}\ \dots\ b_{n1}]^T \tag{6.6}$$

Equation 6.7 is used for calculating values of the elements in the column vector B_i.

$$a_{ij} = \frac{a_{ij}}{\sum_{i=1}^{n} a_{ij}} \tag{6.7}$$

All column vectors B_i are joined in order to form *C* matrix as shown in Equation 6.8.

$$C = \begin{bmatrix} 0.52 & 0.62 & 0.51 & 0.51 & 0.42 & 0.35 \\ 0.13 & 0.16 & 0.21 & 0.21 & 0.24 & 0.23 \\ 0.11 & 0.08 & 0.10 & 0.10 & 0.12 & 0.15 \\ 0.11 & 0.08 & 0.10 & 0.10 & 0.12 & 0.15 \\ 0.07 & 0.04 & 0.05 & 0.05 & 0.06 & 0.08 \\ 0.06 & 0.03 & 0.03 & 0.03 & 0.03 & 0.04 \end{bmatrix} \tag{6.8}$$

Percentage importance distribution of each criterion is obtained by using *C* matrix. Therefore, arithmetic average of row components, which is comprised *C* matrix, is

taken and then column vector *W* that is called priority vector is calculated. Using *C* matrix, percentage importance distribution of criteria is calculated and *W* is illustrated in Formula 6.9.

$$W = [\, w_1\, w_1 \ldots\, w_n]^T,\ a_{ij} = {\sum_{j=1}^{n} c_{ij}}/{n} \tag{6.9}$$

Table 6.5 shows the elements of the priority vector obtained from the comparison matrix constructed by averaging the pairwise comparison values of the three decision-makers.

Table 6.5 Weights of the criteria

Category		**Priority**	**Rank**
C1	Class	50.0%	1
C2	Relationship	19.3%	2
C3	Attribute	10.8%	3
C4	Method	10.8%	3
C5	Relationship type	5.8%	5
C6	Many to many association	3.3%	6

The F-measure value obtained for each criterion in the Bozyiğit's study (2016) (in Table 5.2) is multiplied by the criteria weights determined according to the AHP result as in Equation 6.10, and a new accuracy ratio is obtained. The F-measure value of a criterion that is weighted by AHP but not evaluated in Auto Class is set to 0. As a result, new accuracy rate is determined as 86%. Remark that the accuracy rate is determined as 93% in AutoClass. by calculating precision, recall, and F-measure.

$$\begin{aligned} Accuracy &= (w_1 \times C_1) + (w_2 \times C_2) + \cdots + (w_6 \times C_6) \\ F_{measure} &= (0.5 \times 1) + (0.19 \times 0.88) + \cdots + (0.03 \times 0) \\ F_{measure} &= 0.86 \end{aligned} \tag{6.10}$$

Transformation of requirements into source code is a vital step in software development. Although it is mostly done manually, there are available approaches to automate this step of SDLC. Performance evaluation and validation process in the

studies which automatically transform requirement documents into source codes are evaluated by commonly used measure methods such that recall, precision and F-measure. The evaluation criteria are considered to be equal in these methods. However, the priority of these criteria varies depending on views of users. Thus, we proposed a novel approach in which accuracy rate calculated by using weights of criteria. The weights of criteria are obtained by using opinion of the three experts, and then AHP method. Our novel approach evaluated on the particular study, which is presented by Bozyiğit et al. (2016). It is observed that our novel approach gives more realistic results in the evaluation of studies generating OO-based source codes from requirements. This is because; we base our approach on expert opinions while other studies do not include them.

CHAPTER SEVEN
EXPERIMENTAL STUDY

7.1 Dataset

Datasets have a 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 dataset used. These limitations are explained as follows:

- The datasets used include a few 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 the researchers, who work in the same field, to compare efficiency of their methods. It is realized that there is no study that uses a common dataset shared on public platform.

The dataset (Bozyiğit et al.,2019) was constructed with the use of "SENG 2115 - Object Oriented Programming (OOP)" course questions taught in Software Engineering Department 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 Software Engineering Department between 2015 and 2018. The dataset was made publicly available to be used in the works of other researchers. Table 7.1 shows detailed information about the dataset used in this study. As is also seen from the information in Table 7.1, the requirements in our dataset includes neither short nor so long texts. Thus, we can say that the text in our dataset is mid-range and all contributed studies in this domain test their approach easily using it.

Table 7.2 represents a sample requirements named "Restoran (Restaurant)" that is written both in English and Turkish languages.

Table 7.1 Some properties of 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 7.2 A sample requirements in the dataset

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
English	Yılmaz Restaurant has more than one personnel and dining tables. Personnel may be employed or discharged at certain times. Each personnel have name, age and gender information. The restaurant has more than one sections. 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.

7.2 A Case Study

Each requirement in the dataset is experimented to validate our proposed approach. We selected the requirements named “Restoran” shown in Table 7.3 as a case study for this section. First, the preprocessing phase is applied to the text parts of requirements using basic NLP steps as mentioned in Chapter 3. 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 7.3

Table 7.3 Intermediate data obtained through NLP methods on Restaurant requirements

POS tags & keywords	**Elements**
Nouns	restoran (restaurant), çalışan (personnel), servis_masa (dining table), isim (name), yaş (age), cinsiyet (gender), bilgi (information), bölüm (section), mutfak (kitchen), servis (service), kasa (cashe), aşçı (cook), garson (waiter), kasiyer (cashier), sipariş (order), sistem (system), sipariş_detay (order detail), masa_numara (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)
Relationship keywords	sahip_olmak, olmak, bulunmak

Next, the list of intermediate data in Table 7.3 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 methods) are determined to generate the related class diagram. The results of this process for the case study is presented in Table 7.4

Moreover, the associations between the specified classes are determined performing the relationship rules and patterns 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 7.5. As a result, our proposed approach of indexing comes to the fore as enough effective because the findings are coherent with the real results. Hence, it can be confidently stated that matching proper design elements of class diagram in our architecture is successfully utilized.

Table 7.4 Design elements in the “Restoran” requirements

	Classes	**Attributes**	**Methods**	**Keywords**	**Used rules**
1	restoran	çalışan, servis_masa	-	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_detay, masa_numara	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_numara	göstermek()	-	GR_1, GR_4, GR_5, CR_1
11	kasiyer	-	hesap_kesmek()	-	GR_4, CR_1, MR_1

Table 7.5 Relationships extracted from “Restoran” requirements

Sentence	**$Class_1$**	**$Class_2$**	**Relationship**	**Used rules**
1	restoran	çalışan	Composition	$CompP_1$
2	restoran	bölüm	Composition	$CompP_2$
3	çalışan	çalışan, aşçı, garson, kasiyer	Generalization	$GenP_1$
4	bölüm	mutfak, servis, kasa	Generalization	$GenP_1$
5	aşçı	mutfak	Aggregation	$AggR_2$
6	garson	servis	Aggregation	$AggR_2$
7	kasiyer	kasa	Aggregation	$AggR_2$

For the “Restoran” requirements, nine classes, nine attributes, three methods, and eleven relationships are obtained performing proposed rule-based model. The generated class diagram is illustrated in Figure 7.1. Participated experts confirm that all the design elements are found correctly in the generated class diagram. The results yielded by experimental study provide convincing evidence that our proposed study effectively performs the transformation of requirements texts into the class diagrams.

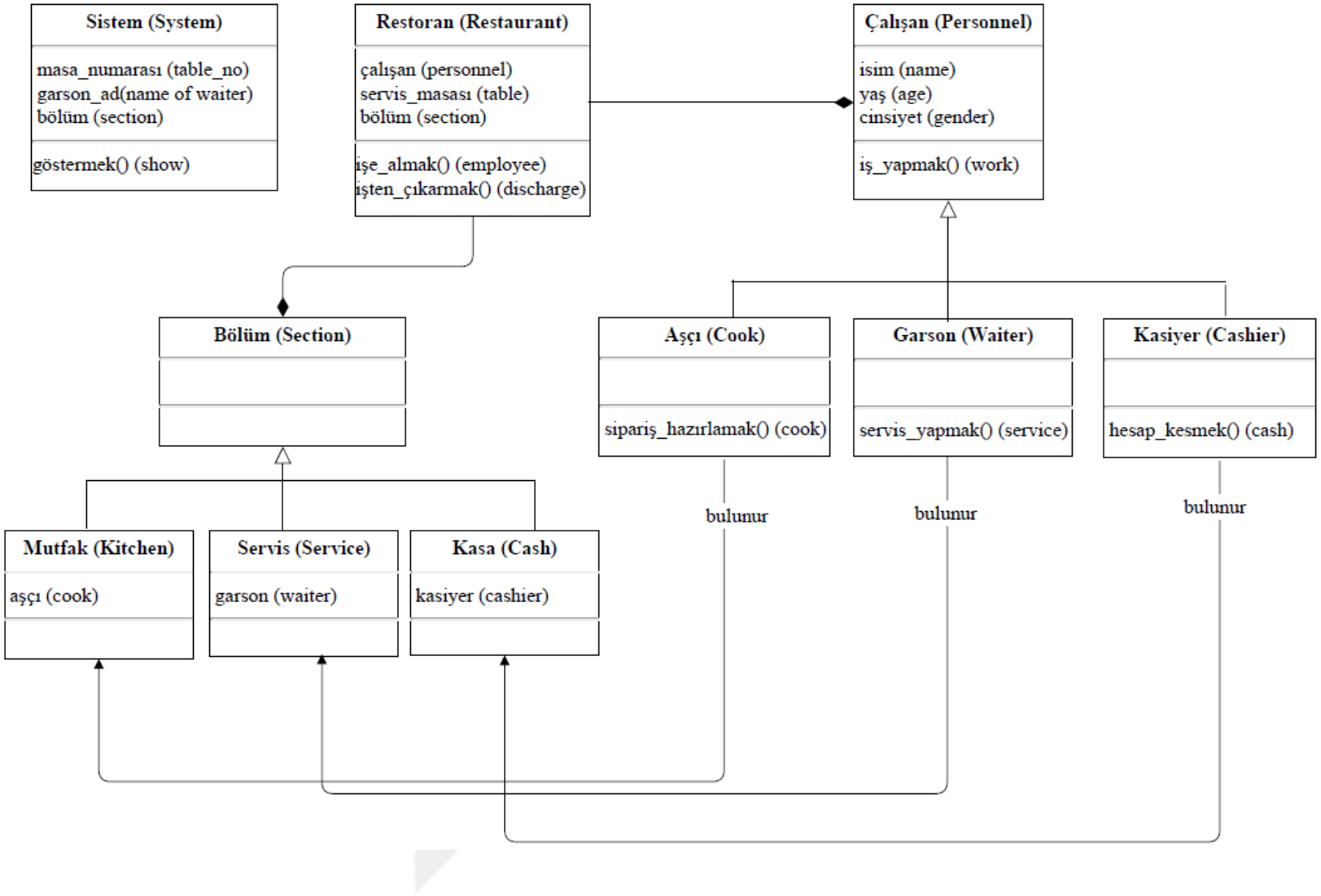

Figure 7.1 Generated class diagram for Restaurant case study

CHAPTER EIGHT
EVALUATION RESULTS

8.1 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. Further, 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 provide decision makers to prioritize criteria in order to deal with complex decision making problems. The flow chart of AHP is showed in Figure 8.1.

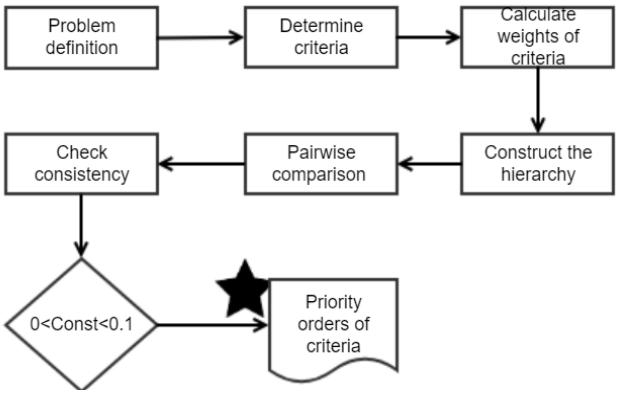

Figure 8.1 Flow chart of AHP

- The decision-making problem is defined and the criteria affecting decision points are determined.
- Decision makers perform pairwise comparison between the specified criteria by using ranking scale proposed by Saaty (2008).
- Matrix calculations are performed with the following Saaty's proposed methodology 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 8.1.

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

Table 8.1 Definition of evaluation criteria

Criterion	**Definition**
Criterion 1	Finding the classes completely
Criterion 2	Finding the relationships between the classes completely
Criterion 3	Finding the attributes on the classes completely.
Criterion 4	Finding the methods on the classes completely
Criterion 5	Specifying the relationship types correctly

Step 3 (Calculating weights of the criteria): The weights of criteria were calculated by applying the matrix calculation following Saaty's proposed study (Saaty, 2008) explained in Chapter 5. The results are shown in Table 8.2.

Table 8.2 Weight of each criterion calculated by AHP

Criterion	**Definition**
Criterion 1	53.7
Criterion 2	21.1
Criterion 3	10.0
Criterion 4	10.0
Criterion 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 view of the experts in our study.

8.2 Evaluation of System Performance

The evaluation process with respect to the specified criteria (stated in Section 8.1) is performed by comparing the outputs of the system with the class diagrams generated by the experts participated in this study. Assume that the set of design elements specified in the experts' model are denoted by **E** and the set of elements revealed by the system is denoted by **S**. The set of S and E are illustrated in Figure 8.2.

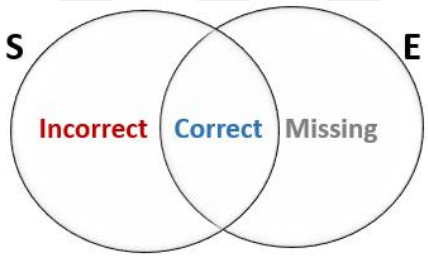

Figure 8.2 The illustration of S and E

- The cardinality of intersection of **S** and **E** gives the number of elements correctly identified by the system (it is donated as $N_{correct}$).
- 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 donated as $N_{incorrect}$).
- The cardinality of difference of **E** and **S** gives the number of missing elements that could not be extracted by the system (it is donated as $N_{missing}$).

In this part, the proposed system is experimented and evaluated for each criterion C_i (stated before in Section 8.1) using all the requirements documents in the dataset. The detailed experimental results for each requirement are presented in Table 8.3.

Table 8.3 Detailed experimental results regarding each criterion (CiC: Number of correct elements providing Ci, CiM: Number of missing elements providing Ci, CiI: Number of incorrect elements providing Ci)

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	0	1	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 galery)	5	0	0	3	0	1	12	2	0	8	1	0	2	1	1
R_{13} (Enrollment)	6	1	0	8	1	1	8	1	2	9	2	2	10	2	3
R_{14} (ATM)	8	1	0	8	0	2	3	1	0	0	3	0	9	1	2
R_{15} (Video rental)	4	0	1	4	1	0	8	2	1	8	2	0	4	1	1
R_{16} (Cinema)	4	0	0	4	0	1	4	1	0	6	1	1	4	0	2
R_{17} (Timbered house)	9	0	0	7	1	0	3	0	0	1	1	0	3	3	1
R_{18} (Musical store)	6	0	0	9	1	1	4	0	1	8	2	1	8	2	3
R_{19} (Pressure)	4	1	1	4	1	2	5	1	1	3	1	0	3	0	3
R_{20} (Airport)	7	1	0	4	1	2	7	0	0	5	0	2	4	0	1

We calculated performance measure (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 how much of the output extracted by the system is correct. It is obtained by finding ratio of the correctly identified data to the total extracted data, in the generated model. Its formula is given in Equation 8.1.

$$Precision\ (Pr) = \frac{Ncorrect}{Ncorrect + Nincorrect} \tag{8.1}$$

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 experts' model. The formula of recall is given in Equation 8.2.

$$Recall\ (Re) = \frac{Ncorrect}{Ncorrect + missing} \tag{8.2}$$

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 Equation 8.3.

$$Fmeasure\ (Fm) = \frac{2 \times Pr \times Re}{Pr + Re} \tag{8.3}$$

Pr, **Re**, and **Fm** values for all requirements in the dataset regarding each evaluation criterion are presented in Table 8.4.

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 criteria varies depending on views of users. Thus, we proposed a novel evaluation model including AHP to assign a weight to each criteria in the direction of the experts' opinions (as stated in Section

6.2). The F-measure value of each evaluation criterion (in Table 8.4) is multiplied by the weights of criteria (Table 8.2), and a new accuracy ratio is calculated for generated class diagrams. The formula for calculating weighted F-measure value is given in Equation 8.4.

$$Accuracy = (w_1 \times C_1) + (w_2 \times C_2) + (w_3 \times C_3) + (w_4 \times C_4) + (w_5 \times C_5) \quad (8.4)$$

Table 8.4 Precision, Recall, and F-measure values (Reqs: Requirements)

Reqs	C_1			C_2			C_3			C_4			C_5		
	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

Table 8.5 shows the comparison of performances which are calculated by conventional and AHP based evaluation method in terms of precision, recall, F-measure on twenty requirements in the dataset.

Table 8.5 Conventional and AHP based evaluation results

Requirements	**Conventional evaluation**			**Evaluation using AHP**		
	Pr	**Re**	**F_m**	**Pr**	**Re**	**F_m**
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	0.86	0.92	0.85	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 8.5, it is seen that nearly all classes and relationships in the generated Restoran (R_1) class diagram are correctly determined. This states 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, C_5 criterion is not met properly in the generated Restoran model. As seen in the Table 8.5, F-measure of 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 which

don't meet the C_5 considerably reduce the value of F-measure. Experts participated in our study state 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 claim that evaluation using AHP gives more realistic results than conventional method. For this reason, we can state that using MCDM methods including expert opinions much possibly provide more realistic and consistent evaluation results in the 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 comparing to the others. It is because that both of two requirements are not well-written in Turkish and the structure of sentences is complex.

CHAPTER NINE
CONCLUSION AND FUTURE WORK

9.1 Conclusion

Transforming requirements into OO conceptual model 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 the reviewed approaches deal with English 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 the 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 on the 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 and English. 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 be varied 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 AHP based evaluation model and decision makers' feedbacks. As the result of the evaluation, average accuracy of the proposed model is measured as 89%.

We cannot compare our results with the 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 is motivating enough for the future works, although the evaluation is performed against experts' model including their assumptions and implicit information.

9.2 Future Work

Considering studies in the literature, it is observed that most of the studies benefit from lexical analysis which includes common techniques in the NLP frame such as tokenization, stemming, and POS tagging. It is seen that there is a little study implements semantic analysis which is critically task to understand the meaning of the text and extract necessary OO design elements. In fact, semantic analysis is not hard task in English, because there is a lexical database for the English language, WordNet, providing the short definitions and synonyms of the words. However, determining semantics of the text may be difficult, because there is no comprehensive dictionary for textual analysis applications as WordNet for the other languages such as Turkish. Thus, we would like to use word embedding which is non language related and widely used to make sense out of the textual data. We believe that performing semantic analysis using word embedding model will improve the performance in term of accuracy of the generated model and running time.

As the future work, it is also aimed to design a novel system which extends our study by the following functionalities:

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

REFERENCES

Abrial, J. R., Börger, E., & Langmaack, H. (1996). The steam boiler case study: Competition of formal program specification and development methods. In *Formal Methods for Industrial Applications*, 1-12.

Ambriola, V., & Gervasi, V. (2006). On the systematic analysis of natural language requirements with circe. *Automated Software Engineering*, *13*(1), 107-167.

An example: The lift (elevator) problem. (n.d.). Retrieved May 26, 2019, from http://www-users.cs.umn.edu/heimdahl/formalmodels/elevator.htm.

Arellano, A., Carney, E., & Austin, M. A. (2015, April). Natural language processing of textual requirements. In *The Tenth International Conference on Systems (ICONS 2015)*, Barcelona, Spain, 93-97.

Aşlıyan, R., Günel, K., & Filiz, A. (February, 2006). Türkçe otomatik heceleme sistemi ve hece istatistikleri. In *Akademik Bilişim*, Denizli, Turkey.

Bajwa, I. S., Samad, A., & Mumtaz, S. (2009). Object oriented software modeling using NLP based knowledge extraction. *European Journal of Scientific Research*, *35*(1), 22-33.

Ball, C. G. & Kim, R. L. (1991). *An object-oriented analysis of air traffic control*. McLean: The MITRE Corporation.

Berry, D. M. (2000). *From contract drafting to software specification: Linguistic sources of ambiguity-a handbook version 1.0*. Retrieved May 01, 2016, from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.9.7928.

Bjork, R. (2004). *An example of object-oriented Design: An ATM simulation*. Retrieved May 26, 2019, from http://www.mathcs.gordon.edu/local/courses/cs211/ATMExample.

Bozyiğit, F., Aktaş, Ö., & Kılınç, D. (2016). AutoClass: Automatic text to OOP concept identification model. *International Journal of Computer Applications*, 150(10), 29-34.

Bozyiğit F., Aktaş Ö., & Kılınç D. (December, 2017). A novel evaluation approach for the systems transforming software requirements to object oriented source code. *In International Conference on Engineering Technologies*, Konya, Turkey, 129-134.

Bozyiğit F., Aktaş Ö., & Kılınç D. (2018). *Adjective list*. Retrieved May 26, 2019, from http://github.com/ftmBozyiğit/TurkishAutoConceptIdentifier/blob/master/AdjectiveList.txt.

Bozyiğit F., Aktaş Ö., & Kılınç D. (2018). *Compound verbs exceptions*. Retrieved May 26, 2019, from https://github.com/ftmBozyiğit/TurkishAutoConceptIdentifier/blob/master/CompoundVerbsExceptions.txt

Bozyiğit F., Aktaş Ö., & Kılınç D. (2018). *Verbs which do not indicating method*. Retrieved May 26, 2019, from https://github.com/ftmBozyiğit/TurkishAutoConceptIdentifier/blob/master/Verbs(notMethods).txt

Bozyiğit, F., Aktaş, Ö., & Kılınç, D. (2019). Automatic concept identification of software requirements in Turkish. *Turkish Journal of Electrical Engineering & Computer Sciences*, *27*(1), 453-470.

Brambilla, M. (2012). From requirements to implementation of ad-hoc social Web applications: an empirical pattern-based approach. *IET Software*, *6*(2), 114-126.

Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, *80*(4), 571-583.

Can, F., Kocberber, S., Balcik, E., Kaynak, C., Ocalan, H. C., & Vursavas, O. M. (2008). Information retrieval on Turkish texts. *Journal of the American Society for Information Science and Technology*, *59*(3), 407-421.

Capuchino, A. M., Juristo, N., & Van de Riet, R. P. (2000). Formal justification in object-oriented modelling: A linguistic approach. *Data & Knowledge Engineering*, *33*(1), 25-47.

Chow, C., & Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, *14*(3), 462-467.

Christiansen, H., Have, C. T., & Tveitane, K. (2007). From use cases to UML class diagrams using logic grammars and constraints. In *RANLP*, 128-132.

Cysneiros, L. M., & do Prado Leite, J. C. S. (2004). Nonfunctional requirements: From elicitation to conceptual models. *IEEE transactions on Software engineering*, *30*(5), 328-350.

Daigle, L. (2004). *WHOIS protocol specification*. Retrieved May 26, 2019, from http://www.ietf.org/rfc/rfc3912.txt.

Derr, K. W. (1997). *Applying OMT: A Practical step-by-step guide to using the Object Modelling Technique*. UK: Cambridge University Press.

Elbendak, M., Vickers, P., & Rossiter, N. (2011). Parsed use case descriptions as a basis for object-oriented class model generation. *Journal of Systems and Software*, *84*(7), 1209-1223.

El-Ghalayini, H., Odeh, M., & McClatchey, R. (2007). Engineering conceptual data models from domain ontologies: a critical evaluation. *International Journal of Information Technology and Web Engineering (IJITWE)*, *2*(1), 57-70.

Eryiğit, G. (2014). ITU Turkish NLP web service. In *14th Conference of the European Chapter of the Association for Computational Linguistics*, Gothenburg, Sweden 1-4.

Fatwanto, A., & Boughton, C. (2008, December). Analysis, specification and modeling of non-functional requirements for translative model-driven development. In *2008 International Conference on Computational Intelligence and Security*, 405-410.

Gelhausen, T., & Tichy, W. F. (2007, September). Thematic role based generation of UML models from real world requirements. In *International Conference on Semantic Computing (ICSC 2007)*, 282-289.

Giganto, R., & Smith, T. (2008, April). Derivation of Classes from Use Cases Automatically Generated by a Three-Level Sentence Processing Algorithm. In *Third International Conference on Systems*, 75-80.

Gülpınar, Ö., & Güçlü, A. G. (2013). How to write a review article? *Turkish Journal of Urology*, *39*(1), 44.

Harmain, H. M., & Gaizauskas, R. (2003). Cm-builder: A natural language-based case tool for object-oriented analysis. *Automated Software Engineering*, *10*(2), 157-181.

Herchi H., & Ben Abdessalem W. (July, 2012). From user requirements to UML class diagram. In *International Conference on Computer Related Knowledge (ICCRK'2012)*, Sousse, Tunisia.

Ibrahim, M., & Ahmad, R. (2010, May). Class diagram extraction from textual requirements using Natural language processing (NLP) techniques. In *2010 Second International Conference on Computer Research and Development*, 200-204.

Insfrán, E., Pastor, O., & Wieringa, R. (2002). Requirements engineering-based conceptual modelling. *Requirements Engineering*, *7*(2), 61-72.

Gervasi, V. (2000). *Environment support for requirements writing and analysis*, Phd Thesis, University of Pisa, Pisa.

Habert, B. et al. 1998. Towards Tokenization Evaluation. In *Proceedings of LREC. 98*, 427-431.

Hunt, J. (2006). *Guide to the Unified Process featuring UML, Java and Design Patterns* (2nd ed.). London, UK: Springer-Verlag.

Kaiya, H., & Saeki, M. (2005, September). Ontology based requirements analysis: lightweight semantic processing approach. In *Fifth International Conference on Quality Software (QSIC'05)*, 223-230.

Kılınç, D., Özçift, A., Bozyiğit, F., Yıldırım, P., Yücalar, F., & Borandag, E. (2017). TTC-3600: A new benchmark dataset for Turkish text categorization. *Journal of Information Science*, *43*(2), 174-185.

Kim, D. K., Lu, L., & Lee, B. (2017). Design pattern-based model transformation supported by QVT. *Journal of Systems and Software*, *125*, 289-308.

Kitchenham, B. A., Dyba, T., & Jorgensen, M. (2004, May). Evidence-based software engineering. In *Proceedings of the 26th international conference on software engineering*, 273-281.

Kiyavitskaya, N., Zeni, N., Mich, L., & Berry, D. M. (2008). Requirements for tools for ambiguity identification and measurement in natural language requirements specifications. *Requirements Engineering*, *13*(3), 207-239.

Koehler, J., & Schuster, K. (2000, April). Elevator Control as a Planning Problem. In *AIPS*, 331-338.

Kordon, F. (2008). *Case study: Air traveling requirements*. Berlin: Springer.

Kumar, D. D., & Sanyal, R. (2008, December). Static UML model generator from analysis of requirements (SUGAR). In *2008 Advanced Software Engineering and Its Applications*, 77-84.

Landhäußer, M., Körner, S. J., & Tichy, W. F. (2014). From requirements to UML models and back: how automatic processing of text can support requirements engineering. *Software Quality Journal*, *22*(1), 121-149.

Liu, D., Subramaniam, K., Eberlein, A., & Far, B. H. (2004, May). Natural language requirements analysis and class model generation using UCDA. In *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, 295-304.

Lovins, J. B. (1968). Development of a stemming algorithm. *Mechanical Translation. and Computational Linguistics*, *11*(1-2), 22-31.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd*

annual meeting of the association for computational linguistics: system demonstrations, 55-60.

Maynard, D., Peters, W., & Li, Y. (2006, May). Metrics for evaluation of ontology-based information extraction. In *International world wide web conference*, 1-8.

Mich, L. (1996). NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA. *Natural Language Engineering*, *2*(2), 161-187.

Miller, G. A. (1995). WordNet: a lexical database for English. *Communications of the ACM*, *38*(11), 39-41.

Montes, A., Pacheco, H., Estrada, H., & Pastor, O. (2008, June). Conceptual model generation from requirements model: A natural language processing approach. In *International Conference on Application of Natural Language to Information Systems*, 325-326.

More, P., & Phalnikar, R. (2012). Generating UML diagrams from natural language specifications. *International Journal of Applied Information Systems*, *1*(8), 19-23.

Moreno, A. M. (1997, June). Object-oriented analysis from textual specifications. In *Ninth International Conference on Software Engineering and Knowledge Engineering*, Madrid, Spain.

Mu, Y., Wang, Y., & Guo, J. (2009, December). Extracting software functional requirements from free text documents. In *2009 International Conference on Information and Multimedia Technology*, 194-198.

Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, *18*(5), 544-551

Nasiboglu, E., Bozyiğit, A., & Diker, Y. (2015). Analysis and evaluation methodology for route planning applications in public transportation. In *Application of Information and Communication Technologies (AICT)*, Moscow, Russia, 477-481.

Overmyer, S. P., Lavoie, B., & Rambow, O. (2001, July). Conceptual modelling through linguistic analysis using LIDA. In *Proceedings of the 23rd international conference on Software engineering*, 401-410. IEEE Computer Society.

Paice, C. D. (1990). "Another stemmer". In *ACM SIGIR Forum*, 56-61.

Perez-Gonzalez, H. G., & Kalita, J. K. (2002, November). GOOAL: a graphic object-oriented analysis laboratory. In *ACM SIGPLAN*, 38-39.

Pohl, K. (2010). *Requirements engineering: fundamentals, principles, and techniques* (1st ed.). Berlin, Germany: Springer-Verlag.

Popescu, D., Rugaber, S., Medvidovic, N., & Berry, D. M. (2007, September). Reducing ambiguities in requirements specifications via automatically created object-oriented models. In *Monterey Workshop*, 103-124. Springer, Berlin, Heidelberg.

Pressman, R. S. (2005). *Software engineering: a practitioner's approach* (8th ed.). Palgrave: Macmillan.

Ramdhani, A., Ramdhani, M. A., & Amin, A. S. (2014). Writing a literature review research paper: A step-by-step approach. *International Journal of Basic and Applied Science*, *3*(1), 47-56.

Rehman, Z., Anwar, W., Bajwa, U. I., Xuan, W., & Chaoying, Z. (2013). Morpheme matching based text tokenization for a scarce resourced language. *PloS one*, *8*(8), 1-8.

Rumbaugh, J. R., Blaha, M. R., Lorensen, W., Eddy, F. & Premerlani, W. (1990). *Object-oriented modelling and design* (1st ed.). Englewood Cliffs, NJ: Prentice-Hall.

Saaty, T. L. (2008). Decision making with the Analytic Hierarchy Process. *International Journal of Services Sciences, 1,* 83-98.

Sagar, V. B. R. V., & Abirami, S. (2014). Conceptual modeling of natural language functional requirements. *Journal of Systems and Software, 88,* 25-41.

Salbrechter, A., Mayr, H. C., & Kop, C. (2004). Mapping pre-designed business process models to UML. In *Software Engineering and Applications: Proceedings of the Eighth IASTED International Conference*.

Seresht, S. M., Ormandjieva, O., & Sabra, S. (2008, August). 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*, 133-142. IEEE.

Sharma, R., Gulia, S., & Biswas, K. K. (2014, April). Automated generation of activity and sequence diagrams from natural language requirements. In *2014 9th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*,.1-9. IEEE.

Shinde, S. K., Bhojane, V., & Mahajan, P. (2012). NLP based object oriented analysis and design from requirement specification. *International Journal of Computer Applications*, *47*(21).

Sighireanu, M. (1998). *Requirement capture, formal description and verification of an invoicing system*. PhD thesis, INRIA, Bordeaux.

Song, I. Y., Yano, K., Trujillo, J., & Luján-Mora, S. (2005). A taxonomic class modeling methodology for object-oriented analysis. In *Information Modeling Methods and Methodologies: Advanced Topics in Database Research*, 216-240. IGI Global.

Straková, J., Straka, M., & Hajič, J. (2014). Open-source tools for morphology, lemmatization, POS tagging and named entity recognition. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, 13-18.

Şeker, G. A., & Eryiğit, G. (2017). Extending a CRF-based named entity recognition model for Turkish well-formed text and user generated content 1. *Semantic Web*, *8*(5), 625-642.

Tayal, M. A., Raghuwanshi, M. M., & Malik, L. (2014, January). Syntax Parsing: Implementation Using Grammar-Rules for English Language. In *2014 International Conference on Electronic Systems, Signal Processing and Computing Technologies*, 376-381. IEEE.

The old new thing. MSDN blogs (2019). Retrieved May 26, 2019, from http://blogs.msdn.com/b/oldnewthing.

Tripathy, A., Agrawal, A., & Rath, S. K. (2014, December). Requirement analysis using natural language processing. In *Fifth International Conference on Advances in Computer Engineering*, Kochi, India, 26-27.

Türk Dil Kurumu (2019). *Büyük Türkçe sözlük*. Retrieved May 26, 2019, from http://tdkterim.gov.tr/bts.

Wahono, R. S., & Far, B. H. (2002). A framework for object identification and refinement process in object-oriented analysis and design. In *Proceedings First IEEE International Conference on Cognitive Informatics*, 351-360. IEEE.

Yamashita, T., & Matsumoto, Y. (2000, April). Language independent morphological analysis. In *Proceedings of the sixth conference on Applied natural language processing*, 232-238.

Yue, T., Briand, L. C., & Labiche, Y. (2010, June). An automated approach to transform use cases into activity diagrams. In *European Conference on Modelling Foundations and Applications*, 337-353. Springer, Berlin, Heidelberg.

Zelkowitz, M. V., & Wallace, D. (1997). Experimental validation in software engineering. *Information and Software Technology*, *39*(11), 735-743.

Zhou, X., & Zhou, N. (2004). Auto-generation of class diagram from free-text functional specifications and domain ontology. *Artificial Intelligence*, *26*.