Constructing Use Case Models from Arabic User Requirements in a Semi-Automated Approach

Sari Jabbarin and Nabil Arman*
Department of Computer Science and Engineering
Palestine Polytechnic University
Hebron, Palestine
*Corresponding author email: narman@ppu.edu

Abstract— **Automated software engineering has attracted a large amount of research efforts. The use of object-oriented methodologies for software systems development has made it necessary to develop approaches that automate the construction of different UML models in a semi-automated approach from textual user requirements. UML use case models represent an essential artifact that provide a perspective of the system under analysis or development The development of such use case models is very crucial in an object-oriented development methodology. The main principles used in obtaining these models are described. A natural language processing tool is used to parse different statements of the user requirements written in Arabic to obtain lists of nouns, noun phrases, verbs, verb phrases, etc. that aid in finding potential actors and use cases. A set of steps that represent our approach for constructing a use case model is presented.**

Keywords- User Requirements, Use Case Model, National Language Processing (NLP) tool.

I. INTRODUCTION

Object-oriented methodologies are used for software systems development for the many benefits they provide like software reuse, reducing software development costs, to name just a few. Therefore, there is a need for development of automated tools that can help in constructing different components of an object-oriented software system.

A use case diagram shows a set of use cases and actors and their relationships. Use case diagrams address the static use case view of a system. These diagrams are especially important in organizing and modeling the behaviors of a system. This paper addresses the problem of generating a use case model from user requirements, written in Arabic, in a semi-automated approach. An Arabic natural language processing tool/software, namely Stanford Tagger/Parser, is used to parse different statements of the user requirements, written in Arabic, to obtain lists of nouns, noun phrases, verbs, verb phrases, etc. that aid in finding potential actors and use cases. A set of steps that represent our approach for constructing a use case model is presented.

II. RELATED WORKS

Recently there is a great interest in automating software engineering activities. Many tools were developed to automate different activities of software systems development like normalizing relational database schemas, reverse engineering of relational database and generating the corresponding entity-relationship data model, …etc. [1, 2]. In addition, many CASE tools were developed to aid in drawing different diagrams of UML. For example, Rational Rose is an object-oriented Unified Modeling Language (UML) software design tool intended for visual modeling and component construction of enterprise-level software applications [3].

Rational Unified Process (RUP) is an object-oriented Web-enabled program development methodology. According to Rational (developers of Rational Rose and the Unified Modeling Language), RUP is like an online mentor that provides guidelines, templates, and examples for all aspects and stages of program development. RUP and similar products, such as Object-Oriented Software Process (OOSP), and the OPEN Process are comprehensive software engineering tools that combine the procedural aspects of development (such as defined stages, techniques, and practices) with other components of development (such as documents, models, manuals, code, and so on) within a unifying framework [4].

More advanced tools were developed to automate software engineering activities that are more complicated than just aiding in drawing a UML diagram or checking its overall structure. Arman and Daghameen proposed a systematic approach that generates class diagrams from textual software requirements. They presented some steps to build a matrix that was used to obtain classes and their associations to generate class diagrams [5].

Kothari proposed an approach that can extract the basic elements for generating a class diagram from user requirements written in a clear way. The Natural Language Processing for Class (NLPC) can extract classes, data members and member functions from the given user requirements [6]. This approach was implemented as a software tool to generate the class diagrams.

Seresht and Ormandjieva proposed an approach to generate use case diagrams from software requirements, but this approach depends on other models to obtain the use case by combining two technologies: Recurcive Object Model(ROM) and Expert Comparable Contextual (ECC) Models. The ROM provides a formal graphical model of the text and the knowledge it carries and the ECC can extract the stakeholder role. This approach can generate the Context Use

Case Model (CUCM) by applying the knowledge included in the ECC model to identify the actors and devising rules for extracting CUCM elements [7].

Cayaba et al. proposed an approach called computer automated use case diagram generator (CAUse), that can generate the use case diagrams from a text described using a special language called ADD [8]. However, this approach depends on the ADD language to generate the use case diagrams.

Mala and Uma proposed an approach to extract the object-oriented elements of system requirements. This approach started by assigning the parts of speech tags to each word in the given requirements. Thus, this approach resolved the ambiguity posed by the pronouns, the pronoun resolutions by normalizing the text at the beginning. After that, the elements of the object-oriented system such as names of the classes, the attributes, methods and relationships between the classes, sequence of actions, the use-cases and actors are identified by mapping the ‘parts of speech- tagged’ words onto the Object Oriented Modeling Language elements using mapping rules [9].

III. CONSTRUCTING USE CASE MODELS

This section describes how the actors and use cases are extracted from user requirements written in Arabic. There is a need for an Arabic Natural Language Processing tool such as the Stanford Tagger, which is used in this research to help in splitting and tokenizing the Arabic user requirements text. Once this is performed, a set of heuristics are used to construct the use case model as presented in subsections A though E.

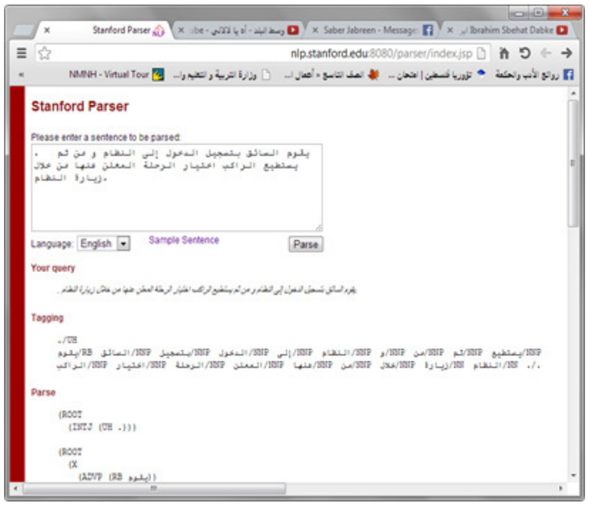
A. *Stanford Tagger*

Stanford Tagger is a piece of software that reads text in some language, in our case Arabic, and assigns parts of text to each word, such as noun, verb, adjective, etc., This software is implemented in Java programming language. The tagger was originally written by Kristina Toutanova. Since that time, Dan Klein, Christopher Manning, William Morgan, Anna Rafferty, Michel Galley, and John Bauer have improved its speed, performance, usability, and support for other languages [10].

All user requirements are processed using the Stanford tagger by writing the requirements in the text area provided for that purpose as shown in Fig. 1.

A set of user requirements for a system implementing ridesharing is used. The requirements were written in Arabic and some of these requirements are used in our examples. The ridesharing system includes many requirements. Two examples are presented below:

- يقوم السائق بتسجيل الدخول الى النظام ومن ثم يستطيع الاعلان عن الرحله التي سيقوم بها ويقوم في هذه المرحله بتحديد ومتطلباتها وتشمل: وقت الرحله (الانطلاق) و المسار الذي سيسلكه اضافة الى عدد المقاعد الفارغه. كما ويستطيع حذف رحلة بعد انتهاءها او الغائها.


Stanford Parser

nlp.stanford.edu:8080/parser/index.jsp

Please enter a sentence to be parsed:

يقوم السائق بتسجيل الدخول إلى النظام و من ثم يستطيع الراكب اختيار الرحلة المعلن عنها من خلال زيارة النظام.

Language: English

Your query

يقوم السائق بتسجيل الدخول إلى النظام و من ثم يستطيع الراكب اختيار الرحلة المعلن عنها من خلال زيارة النظام .

Tagging

```
./UH
يقوم/RB السائق/NNP بتسجيل/NNP الدخول/NNP إلى/NNP النظام/NNP و/NNP من/NNP ثم/NNP يستطيع/NNP
الراكب/NNP اختيار/NNP الرحلة/NNP المعلن/NNP عنها/NNP من/NNP خلال/NNP زيارة/NN النظام/NN ./.
```

Parse

```
(ROOT
  (INTJ (UH .)))

(ROOT
  (X
    (ADVP (RB يقوم))
```

Figure 1. Stanford Tagger Screen Snapshot

- يقوم السائق بقبول الركاب او رفضهم , يستطيع ايضا تتبع المسافرين باستخدام ال GPS ان توفرت هذه الخاصيه عند الركاب في النهايه يقوم بتسجيل الخروج.

In addition, Stanford Tagger uses a set of tags to describe different components of a statement.

Stanford Tagger tokenizes the statements and uses a large number of tags. Table 1 shows the ones used in our approach.

Table 1. Tags’ Descriptions

Tag	Description
CC	conjunction, coordinating
IN	preposition or conjunction, subordinating
JJ	adjective or numeral, ordinal
NN	noun, common, singular or mass
NNP	noun, proper, singular
VB	verb, base form
VBD	verb, past tense
VBN	verb, past participle
VBP	verb, present tense, not 3rd person singular
VBZ	verb, present tense, 3rd person singular

B. *Actors Identification*

To identify the actors from the user requirements written in Arabic, a set of heuristics are presented. These heuristics are used to extract the actors from the tagging of the user requirements generated from the Stanford Tagger. These heuristics are presented as follows:

- If the statement is simple (i.e. it contains only a verb, a subject and an object) then the actor is the main subject in the statement.

  e.g. يقوم السائق بتسجيل الدخول

  Here the main subject is السائق and it's the actor.

  Generalization: If the statement is in the form of <VBZ> < NNP> < NNP> <NNP> when using the Stanford Tagger, then the first NNP is the actor. To simplify referencing, the form can be write as <VBZ> < NNP$_{(1)}$> < NNP$_{(2)}$> < NNP$_{(3)}$>, where the subscripts determine the order of the NNPs.
- When there are two statements combined with a connection then, there are three cases:

  a) The subject is the actor.
  e.g. يقوم السائق بتسجيل الدخول إلى النظام و من ثم يستطيع الإعلان عن الرحلة

  The actor is السائق.

  b) If the subject is redundant in the second statement then the actor doesn't change.
  e.g. يقوم السائق بتسجيل الدخول إلى النظام و من ثم يستطيع السائق الإعلان عن الرحلة

  The actor is السائق.

  c) If the subject changes in the second statement then this is another actor.
  e.g. يقوم السائق بتسجيل الدخول إلى النظام و من ثم يستطيع الراكب اختيار الرحلة المعلن عنها من خلال زيارة النظام.

  The actors are السائق and الراكب .

  Generalization: If the statement is in the form of <VBP> <DTNN> <NN> <DTNN> <IN> <DTNN> < DTJJ> <CC> <VBP> <DTNN> <IN> <DTNN> <WP> <VBD> <NNP> <CC> <NNP> <IN> <DT> <DTNN> <NNP> <NNP> <NNP> <PUNC> <NN> <JJ> <DTNN> <DTJJ> <CC> <DTNN> <WP> <VBP> <NN> <IN> <NN> <DTNN> <DTJJ> <PUNC> <CC> <VBD> <NN> <NN> <NN> < NN> <CC> <NN> when using the Stanford Tagger, then the first NN is the actor, or if there is an CC found in the statement, then we check the first NN after the CC, if this NNP is redundant after the CC then the actor doesn't change, but if the NNP has changed after the CC then this is another actor.
- As mentioned previously, to simplify referencing, subscripts are used. Thus the form is <VBP$_{(1)}$> <DTNN$_{(1)}$> < NN$_{(1)}$> <DTNN$_{(2)}$> <IN$_{(1)}$> <DTNN$_{(3)}$> <DTJJ$_{(1)}$> <CC$_{(1)}$> <VBP$_{(2)}$> <DTNN$_{(4)}$> <IN$_{(2)}$> <DTNN$_{(5)}$> <WP$_{(1)}$> < VBD$_{(1)}$> <NNP$_{(1)}$> <CC$_{(2)}$> <NNP$_{(2)}$> <IN$_{(3)}$> <DT$_{(1)}$> <DTNN$_{(6)}$> <NNP$_{(3)}$> <NNP$_{(4)}$> <NNP$_{(5)}$> <PUNC> <NN$_{(2)}$> <JJ$_{(1)}$> <DTNN$_{(7)}$> <DTJJ$_{(1)}$> <CC$_{(3)}$> <DTNN$_{(8)}$> <WP$_{(2)}$> <VBP$_{(3)}$> <N$_{(3)}$> <IN$_{(4)}$> <NN$_{(4)}$> <DTNN$_{(9)}$> <DTJJ$_{(2)}$> <PUNC> <CC$_{(4)}$> <VBD$_{(2)}$> <NN$_{(5)}$> <NN$_{(6)}$> <NN$_{(7)}$> <NN$_{(8)}$> <CC$_{(5)}$> <NN$_{(9)}$>

C. *Use Cases Identification*

To identify the use cases from the user requirements, more heuristics that can be used to extract the use cases from the user requirements are presented.

- If the statement is simple (i.e. it contains only a verb, a subject and an object) then the use case is the main object in the statement.

  e.g. يقوم السائق بتسجيل الدخول

  The main subject is تسجيل الدخول and it's the use case.

  Generalization: If the statement is in the form of <VBZ> < NNP> <NNP> < NNP> or in the form <VBZ$_{(1)}$> <NNP$_{(1)}$> <NNP$_{(2)}$> <NNP$_{(3)}$> after using subscripts when using the Stanford Tagger, then the first VB is the use case.
- If the statement contains the connector (و) without any verb or actor in the second statement then the second statement is the use case.

  e.g. يستطيع الراكب الانضمام إلى الرحلة و الحجز فيها

  In the above example, the statement contains a connector (و) so there are two use cases 1- يستطيع الانضمام 2- يستطيع الحجز

  Generalization: If the statement is in the form of <VBZ> <NNP> <NNP> <IN> <NNP> <CC><NNP><NNP> when using the Stanford Tagger, then

  a) The use case is the first VB with the second NNP.

  b) The use case is the first VB with the first NNP after the CC.

  Again the form can be re-written using subscripts to tags as, <VBZ$_{(1)}$> <NNP$_{(1)}$> <NNP$_{(2)}$> <IN$_{(1)}$> <NNP$_{(3)}$> <CC$_{(1)}$> <NNP$_{(4)}$> <NNP$_{(5)}$>
- If the statements that contain the connector(أو) without any verb or actor in the statement then the first verb in the statement with the first noun after each connector is a use case.

  e.g. يستطيع الراكب الانضمام إلى الرحلة و الحجز فيها أو الانسحاب منها.

In the above example, the statement contains a connector (e.g. أو) so there are three use cases 1- يستطيع الانضمام 2- يستطيع الحجز 3-يستطيع الانسحاب.

Generalization: If the statement is in the form of <VBZ> <NNP> <NNP> <IN> <NNP> <CC> <NNP> <NNP> <CC> <NNP> <NNP> when using the Stanford Tagger, then

a) The use case is the first VB with the second NNP.

b) The use case is the first VB with the first NNP after the CC. The form after adding the subscript to each tag is <VBZ$_{(1)}$> <NNP$_{(1)}$> <NNP$_{(2)}$> <IN$_{(1)}$> <NNP$_{(3)}$> <CC$_{(1)}$> <NNP$_{(4)}$> <NNP$_{(5)}$> <CC$_{(2)}$> <NNP$_{(6)}$> <NNP$_{(7)}$>

D. *Use Case Model Generation*

To complete the generation of the use case model, a structure that depicts the relationships among the different tokens is needed. A matrix consisting of columns with headings, which contain the potential use cases, and rows with labels, which contain the potential actors, is used. These are obtained from the heuristics explained previously. The matrix is filled by arrow symbols. An arrow means that an actor is associated with one or more particular use cases as shown in Table 2.

Once the matrix is constructed, the use case model is obtained by taking an actor with all its associated use cases to generate a use case diagram. The set of all use case diagrams represent the use case model. According to the above description, this approach can be implemented easily to generate a use case model.

Table 2. Matrix of Potential Actors and their Use Cases

Potential Actors / Potential Use Case	الراكب	السائق	المدير
تسجيل الدخول	←	←	←
يستطيع الاعلان		←	←
تحديد متطلبات		←	
قبول الركاب		←	
نتبع المسافرين		←	
تسجيل الخروج	←	←	←
يستطيع الانضمام	←		
يستطيع الحجز	←		
يستطيع الانسحاب	←		
يوفر تغذيه	←		
يستطيع اضافة			←
يستطيع حذف			←
تصنيف مستخدمين			←
يستعرض الرحلات	←		←

E. *Further Refinement to Use Case Model*

In UML there are three associations between use cases, namely "include", "extend", and "generalization". From the matrix, we can obtain the "include" relationship, by asking if a certain use case depends on another use case based on the meaning or semantics. So we can ask if the use case 1 is required by use case 2, then if the answer is "Yes" we can suggest there is an "include" relationship and if the answer is "No" then there is no "include" relationship between these use cases. The "extend" and "generalization" relationships can be handled similarly. These steps need human intervention, thus our approach can't be fully automated but semi-automated.

IV. CONCLUSIONS

The proposed approach of developing use case models is very essential in the practice of object-oriented software engineering. This approach can be implemented and incorporated in any Integrated CASE (Computer Aided Software engineering) Tool to aid in the process of obtaining the use case models from user requirements written in Arabic. The approach has the main advantage of dealing with Arabic language. In addition, a set of heuristics are presented to obtain the use cases. These heuristics use the tokens produced by a natural language processing tool, namely Stanford Tagger.

ACKNOWLEDGMENT

The authors would like to than the Software Engineering Research Group (SERG) at Palestine Polytechnic University (PPU) for their feedback to the original idea and Dr. Diya Abu Zeina for his help with Stanford Tagger.

REFERENCES

[1] N. Arman, "Normalizer: A Case Tool to Normalize Relational Database Schemas, Information Technology Journal, pp. 329-331, Vol. 5, No. 2, ISSN: 1812-5638, 2006.

[2] N. Arman, "Towards E-CASE Tools for Software Engineering," International Journal of Advanced Corporate Learning, pp. 16-19, Vol. 6, No. 1, 2013.

[3] http://searchciomidmarket.techtarget.com/home/0,289692,sid183,00.html, accessed: October 15, 2013.

[4] "Rational Unified Process (RUP)": ch1, Prentice Hall 1990, ISBN 0-13-629841-9.

[5] N. Arman. and K. Daghameen, "A Systematic Approach for Constructing Static Class Diagrams from Software Requirements," International Arab Conference on Information Technology (ACIT2007), November 26-28 2007, Amman, Jordan.

[6] P. Kothari, "Processing Natural Language Requirement to Extract Basic Elements of a Class," International Journal of Applied Information Systems (IJAIS) , ISSN : 2249-0868.

[7] S. Seresht and O. Ormandjieva, "Automated Assistance for Use Cases Elicitation from User Requirements Text," 11th. Workshop on Requirement Engineering, 2009.

[8] C. Cayaba, J. Rodil and N. Lim, "CAUse: Computer Automated Use Case Diagram Generator", 2006.

[9] G. Mala and G. Uma, "Automatic Construction of Object Oriented Design Models [UML Diagrams] from Natural Language Requirements Specification", 2006.

[10] http://nlp.stanford.edu:8080/parser/index.jsp, accessed: October 1, 2013.