Constructing Activity Diagrams from Arabic User Requirements using Natural Language Processing Tool

CONFERENCE PAPER · APRIL 2015

DOI: 10.1109/IACS.2015.7103200

2 AUTHORS:

Ibrahim Nassar

Palestine Polytechnic University

5 PUBLICATIONS **0** CITATIONS

SEE PROFILE

Faisal Khamayseh

Palestine Polytechnic University

13 PUBLICATIONS **6** CITATIONS

SEE PROFILE

Constructing Activity Diagrams from Arabic User Requirements using Natural Language Processing Tool

Ibrahim N. Nassar
Department of graduate studies
Palestine Polytechnic University
Hebron, Palestine
inassar@ppu.edu

*Faisal T. Khamayseh
Department of graduate studies
Palestine Polytechnic University
Hebron, Palestine
faisal@ppu.edu

**Corresponding Author*

Abstract—**Automated software engineering is one of the most important current research problems especially when it comes to requirements analysis and modelling. The main advantage of this automation process is to improve the quality and productivity of software development. Automated software engineering requires detailed analysis of user requirements using Natural Language Processing tools. This paper presents a semi-automated approach for constructing the activity diagrams from Arabic user requirements using MADA+TOKAN parser. This research aims to help software engineers - in the analysis phase - reducing cost and time required in performing manual processes and activities.**

Keywords- Automated Software Engineering, Arabic User Requirements, Activity Diagram, MADA+TOKAN.

I. INTRODUCTION

Automated Software Engineering (ASE) is one of the most important research problems of software user requirements. ASE focuses on how to automate the software systems to improve the quality and productivity of software development. In addition, ASE lowers the cost and time of analysis of user requirements [1]. ASE tools and methods are useful in a wide area of applications. The first phase in building the automated software engineering applications is writing the user requirements. User requirements are Natural Language sentences that describe the high-level objectives of a software system [2]. User requirements describe a particular problem in an effort towards understanding the problem which is is vital to develop effective software with high accuracy.

Unified Modelling Language (UML) models are one of the most important parts of software engineering to use the automated software tools and user requirements [3]. Drawing UML diagrams requires analysis of user requirements carefully to get the desired diagram. However, for large and complex requirements, generating UML diagrams manually is error prone and behavior conflict. Therefore, there is a need for developing algorithms and tools to enhance object-oriented software systems.

The main motivation is to come up with a cross activity from Arabic user requirements to activity diagram avoiding the traditional methods of creating sets of UML models hence saving time and cost. In addition, there is a lack of the research of transforming Arabic user requirements to UML design models.

Activity Diagram (AD) is the most important UML model that is needed to be generated from user requirements [4]. AD represents the dynamics of the system and model business processes. AD describes how the activities are coordinated to provide and show how the system achieves the operations.

The research goal of this paper is to generate high-level contextual view on the software system's elements of the activity diagram automatically and thus objectively from Arabic user requirements. We need an Arabic Natural Language Processing tool to extract the notations of the activity diagram from Arabic user requirements such as the MADA+TOKAN. It used in this paper to split and

tokenize the Arabic user requirements. This paper organized as follows: Section 2 and 3 highlight the related work and background respectively. Section 4 presents the construction of activity diagram. Section 5 proposes the algorithm of generating the activity diagram. Finally, conclusion is presented in section 6.

II. Related Work

Recently, automated software engineering methods have been applied in several areas in software engineering field. We present some set of related research work in the area of the paper.

Yue et al. [5] proposed an automated method to construct activity diagrams from use cases. This approach implemented in aToucan tool. Requirements engineers defined use cases manually using Restricted Use Case Modeling approach. The result is a textual use case model that expressed in a restricted natural language. aToucan reads textual use case specifications to identify part-of-speech and grammatical relation dependencies of sentences. Then, it records the information into an instance of the UCMeta that is the intermediate model in aToucan and used to bridge the gap between a textual UCMod and a UML analysis model. Finally, it transforms the instance of UCMeta into activity diagram based on many transformation rules.

Khan and El-Attar [6] proposed an automated approach to generating activity diagram from Use Case Maps. They implemented a model transformation from UCM notation to activity diagram notation.

Rodriguez et al. [7] proposed an automated method to auto-generate activity diagram of each use case written in a specific format. The generation is performed by a model transformation, taking use case textual scenario as an input and producing the corresponding activity sub-diagram. The transformation is defined using QVT (Query/ View/ Transformation) relational language and implemented in Java as a prototype tool.

Kaewchinporn and Limpiyakorn [8] proposed an automation approach to reviewing activity diagrams based on action description language (ADL). They transform the requirements to XML file. Then, the ADL script is created using the information extracted from the XMI file. Then, ADL scripts will be parsed to ADL semantic model.

Some commercial auto generator tools generate activity diagrams from use cases with some constraints. There exist three closed tools: Visual Paradigm, CaseComplete and Ravenflow [5] to perform generating process. None of them use free written requirements as an input.

Arman and Jabbarin [9] proposed a semi-automated approach to generating use case models from Arabic user requirements using natural language processing. This paper presents a set of steps as their approach for constructing a use case model. Set of heuristics are presented to obtain use cases. These heuristics use the tokens produced by a natural language processing tool, namely Stanford parser.

III. Background

This section gives a theoretic background related to the context of generating activity diagrams from requirements.

A. Arabic User Requirements

User requirements are Natural Language sentences that describe the high-level objectives of software systems. Having the Arabic user requirements effective and correct is essential skill. We present the general guidelines for writing Arabic user requirements [10]. The following general guidelines are needed for writing Arabic user requirements which mean that requirements should be written in simple and correct format:

1. User requirements should be grammatically correct. Write statements in a form of: subject-verb-object or verb-subject-object only. Arabic language has a free word order. There are four different forms of formal statements: SVO, OVS, VSO, and VOS.
2. Arabic language of the user requirements should written in standard Arabic.
3. User requirements should be consistent and complete sentences rather than bulleted buzz phrases.
4. User requirements should be free of misspellings and typos.

5. User requirements should be an intelligible rationale. It should be simple, unambiguous, and short statements.
6. Each sentence must be ended with full stop.

B. Activity Diagram

Activity diagram (AD) provides a complete view of how the system is working by describing the sequence of the activities. AD is suitable for building and illustrating business processes and modeling the procedural flow of activities [4]. The paper uses the standard notations of the activity diagram:

Initial Node: Shows the first action state of a workflow on an activity diagram.

Final Node: Shows the last node of a workflow on an activity diagram.

Activity node: The main component of an activity diagram. It represents the process, task of the operation for the system.

Control Flow Transition: An arrow connects the elements and represents the workflow between two or more elements of the activity diagram. Flow transition shows how the elements are ordered.

Decision Node: Is a control node shown as a diamond shape, which is a single control flow of incoming edge and multiple outgoing edges leaving it. Transitions that leave a decision node have guard conditions with conditions enclosed in square brackets.

C. MADA+TOKAN Tagger

Although there are many tools for Arabic morphological analyzers, each one has different characteristics that are based on the way these tools are developed and the use of database. We searched most of the used tools and found that two of them are freely available and very suitable for tokenization Arabic text. These tools are Stanford and MADA+TOKA taggers. The reason for choosing MADA+TOKAN is the diacritization process as part of a wider Arabic NLP toolkit. It also gives a unique analysis of each recognized word. The accuracy to take different tags for each word is very high which surpasses 96.1% and with errors about 2.4%.

MADA+TOKAN [11] uses a set of tags to describe the Arabic words in the statements. Each word has a tag, and every tag has a special meaning. Table 1 shows all tags of MADA+TOKAN tagger.

Table 1: Tags of MADA+TOKAN

Part of Speech Definition	Tags
Verbs	VBN, VBP, VBD
Nouns	NN, NNS, NNP, NNPS, DTNN, DTNNP, DTNNS
Adjectives	JJ, DTJJ
Adverbs	RB, WP, WRB
Particles	IN, DT, RP
Coordinating Conjunction	CC
Cardinal Number	CD
Pronoun	PRP, PRP$
Interjection	UH
Punctuation	PUNC
Foreign Word	FW

IV. ACTIVITY DIAGRAMS CONSTRUCTION

This section describes how the main elements of Activity Diagrams extracted from Arabic user requirements. We describe how to use MADA+TOKAN parser to splitting and tokenizing the Arabic text.

1. Initial Node: The beginning of the activity diagram; it must be only one initial node and connected with the next activity element. It does not need to be extracted from user requirements.

2. Final Node: The last element of an activity diagram and connected to the previous element. It does not need to be extracted from user requirements since it is used to show the end of an activity diagram.

3. Activity Node: It is the main notation of the activity diagram; which is used to represent the operations of user requirements. Each activity node has one operation. Activity nodes represent the verbal sentences. In Arabic language, there are two types of verbal sentences; simple and complex. Simple sentence can be constructed with Verb-Subject-Object while the complex sentence consists of more than one subject, verb and object.

We present the basic grammar rules for the simple verbal sentences [12-14] in the following:

1. Verbal Sentence → Verb + subject + object.
2. Verb → Past | Present | Imperative mood.
3. Subject → Noun | Pronoun.

4. Noun → Proper noun | demonstrative | Relative.
5. Object → Noun | Quasi Sentence.
6. Quasi Sentence → Preposition phrase | Adverb phrase.
7. Preposition Phrase → Preposition + Noun.
8. Adverb Phrase → Adverb + Noun.

To represent the activity node with reliance on simple verbal Arabic sentences, we propose the following heuristic points:

1. **Verb + Subject** → Verb + (Noun | Pronoun). This type of sentences has corresponding tags pattern as: VP (VBP|VBD|VBN) + NP | PRP (DTNN|NN|NNS|DTTNS|NNP|NNPS|DTN NP| RPRP| PRP$) as indicated in table 1.
2. **Verb + Object** → Verb + (Noun | Preposition + Noun | Adverb + Noun). This sentence has tags pattern as: VP (VBP| VBD| VBN) + NP (DTNN| NN| NNS| DTTNS| NNP| NNPS| DTNNP) | PP+ NP (IN+DTNN) | NP+NP (NN|NNP + DTNN|NN).
3. **Verb + Subject + Object** → Verb + (Noun | Pronoun) + (Noun | Preposition + Noun | Adverb + Noun) in which its corresponding tag pattern is: VP (VBP| VBD|VBN) + NP | PRP (DTNN|NN| NNS| DTTNS| NNP|NNPS|DTNNP| RPRP| PRP$) + NP (DTNN| NN|NNS| DTTNS|NNP| NNPS| DTNNP) | PP+NP (IN+DTNN) | NP+NP (NN|NNP + DTNN|NN).
4. **Subject + Verb** → (Noun | Pronoun) + Verb**.** This sentence has tags pattern like: NP | PRP (DTNN| NN|NNS| DTTNS| NNP| NNPS| DTNNP| RPRP| PRP$) + VP (VBP|VBD|VBN).
5. **Subject + Verb + Object** → (Noun | Pronoun) + Verb + (Noun | Preposition + Noun | Adverb + Noun) has a corresponding tags pattern as: NP | PRP (DTNN| NN|NNS|DTTNS|NNP|NNPS|DTNNP|RPR P|PRP$) + VP (VBP|VBD|VBN) + NP (DTNN|NN|NNS|DTTNS|NNP|NNPS|DTN NP) | (IN+DTNN) | NP+NP (NN|NNP + DTNN|NN)

To identify the activity nodes from complex sentences of Arabic user requirements, we propose a set of heuristics for the verbal sentences. These heuristics presented as follows:

1) **Verb + Subject + Object (1) | Object (2) | Object (3)** → Verb + (Noun | Pronoun) + (Noun | Preposition + Noun | Adverb + Noun) + (Noun | Preposition + Noun | Adverb + Noun | Adjective) + (Noun | Preposition + Noun | Adverb + Noun | Adjective).

2) **Subject + Verb + Object (1) | Object (2) | Object (3)** → (Noun | Pronoun) + Verb + (Noun | Preposition + Noun | Adverb + Noun) + (Noun | Preposition + Noun | Adverb + Noun | Adjective) + (Noun | Preposition + Noun | Adverb + Noun | Adjective) **.**

4. Decision node: This is a control node that has one incoming edge and two leaving outgoing edges leaving. The structure of conditional sentences includes four parts [15]. These parts include:
Conditional particle + conditional sentence + answer particle + conditional answer.

4.1 Conditional Particle: There are two common condition particles in Arabic Language (إذا ، لو). The corresponding tags are RB and NNP respectively. Conditional particle represents the decision node.

4.2 Conditional sentence: Conditional sentence is a verbal sentence. There are two types of the conditional sentences: the proof-sentence and negation-sentence. Theses sentences represented into the decision node.

4.3 Answer Particle: Answer particle is an adverb for the condition answer. Answer particles in Arabic are: (فان، سوف، فسوف، حرف اللام). The tags for them are NNP: فان, CC+RP: فسوف, RP: سوف, IN + NN: حرف اللام + الاسم المجرور.
Answer particle represents the control flow of the decision node. It relies on the type of the conditional sentence; if it is a proof sentence then the answer particle is an arrow going to the true branch of the activity node. But if it is a negation sentence then the answer particle is an arrow going to the false branch of the activity node.

4.4 Conditional answer: Conditional answer is a verbal sentence. It represents an activity node. We propose four heuristics for the two common conditional sentences:

1. Conditional Particle + Verbal sentence$_1$ + Answer Particle + Verbal sentence$_2$: This sentence has tags pattern like the following:
(RB | NNP) + Verbal sentence$_1$ + (NNP| CC+RP | RP| IN+NN) + Verbal sentence$_2$
2. Conditional Particle + Verbal sentence$_1$ + Verbal sentence$_2$: In this case, the corresponding tags pattern is:
(RB | NNP) + Verbal sentence$_1$ + Verbal sentence$_2$.

The following heuristics illustrate the tag structure of negation statements:
1. Conditional Particle + Negation Particle + Verbal sentence$_1$ + Answer Particle + Verbal sentence$_2$: The corresponding tags pattern is:
(RB | NNP) + PRT + Verbal sentence$_1$ + (NNP| CC+RP | RP| IN+NN) + Verbal sentence$_2$.
2. Conditional Particle + Negation Particle + Verbal sentence$_1$+ Verbal sentence$_2$.
This sentence has tags patterns like:
(RB | NNP) + PRT + Verbal Sentence$_1$+ Verbal Sentence$_2$.

5. Control Flow Transition: Control flow transition is an arrow used to connect the notations of the activity diagram from one node to other, and shows how the paths of the system are ordered. To identify the control flow transition from Arabic user requirements, we look to the connectors in the sentence and full stops that are shown at the ends of sentences. We present our method to analysis the connectors that exist inside the requirements. There are four common connectors in Arabic Language that are (و، أو، ثم، ف). If the connector is (ثم), then the tag is RB, it hence separates between two verbal sentences or verbal sentence and a noun. If the connector is any of (أو /و /ف), then the corresponding tag is CC. In this case, parts are separated by two verbal sentences or verbal sentence and a noun. CC tag has different meaning in Arabic Language; it may be a connector or a decision connector. If the word that precedes CC is noun or adjective, or both of them, and the next word is a noun, adjective, or both of them then, we don't separate them. But, if a verb occurs after CC, we check the next statement. If it begins with conditional particles (لو، اذا), then CC is a decision connector. Otherwise, CC separates between two verbal statements.
Full stop in the user requirement controls the flow transition in the activity diagram. We can identify the control flow transition between the notations by identify the full stop at the end of each sentence. Full stop has a tag (**.** / PUNC) separating the source and the target notations using an intermediate arrow.

V. ALGORITHM FOR EXTRACTION THE ACTIVITY DIAGRAMS

Following algorithmic steps represents the generation of Activity Diagrams from Arabic User Requirements.

Input: Arabic User Requirements.
Output: Activity Diagram.
Step 1: Parse Arabic User Requirements using MADA+TOKAN.
Step 2: For each word in the list of requirements, there exist a tag using MADA+TOKAN.
Step 3: Construct one initial node to show the start of an activity diagram.
Step 4: If the requirements are simple verbal or complex verbal requirements in forms of (VS, VO, SV, VSO, SVO, $VSO_1O_2O_3$, $SVO_1O_2O_3$), then generate a corresponding activity node.
Step 5: If there is a connector such as (CC) or (RB) between the verbal requirements in Step 4, then:

1. If the connector is (ثم/RB), then an arrow separates between two verbal sentences or verbal sentence and a noun.
2. If the connector is any of ((أو /و /ف)/CC), then:

a. If the word that precedes CC is a noun or adjective, or both of them, and the next word is a noun, adjective, or both of them, then, we leave them without separation.
b. If a verb comes after CC, we check the next statement. If it begins with a conditional particle (لو، اذا) / (NNP, RB), then CC is a decision connector. Otherwise, CC separates between two verbal statements using arrow.

Step 6: The full stop (**.** / PUNC) after the end of each sentence, is converted to an arrow separating notations.

Step 7: If the sentence is a conditional sentence in a form of the following, then we generate the decision node. These forms are:

1. Conditional Particle (RB | NNP) + Verbal sentence$_1$ + Answer Particle (NNP| CC+RP | RP| IN+NN) + Verbal sentence$_2$.
2. Conditional Particle (RB | NNP) + Verbal sentence$_1$ + Verbal sentence$_2$.
3. Conditional Particle (RB | NNP) + Negation particle (PRT) + Verbal sentence$_1$ + Answer Particle (NNP| CC+RP | RP| IN+NN) + Verbal sentence$_2$.
4. Conditional Particle (RB | NNP) + Negation particle (PRT)+ Verbal Sentence$_1$+ Verbal Sentence$_2$.

Step 8: Construct one final node to show the end of the activity diagram.

VI. CONCLUSIONS

The paper studies the various techniques used in generating some UML models. In this work, an algorithm of extracting the activity diagram and high-level algorithm to be generated from Arabic user requirements using natural language processing tool is presented. The main tool used is the MADA+TOKAN tagger. Set of new heuristics are presented to obtain the notations of activity diagram. This work aims to help software engineers during the analysis phase of the object-oriented software development in the process of generating UML models. The main contribution of this paper is the new semi-automated generation of activity diagram from requirements written in Arabic language in which reduces cost and time. On the practical side, the algorithm will be implemented with some real cases written in Arabic language. Some analysis and measurements will be conducted.

ACKNOWLEDGMENT

We would like to thank Prof. Nabil Arman, Dr. Diya Abu Zeina, and Khalid Daghameen from Palestine Polytechnic University (PPU) for their useful notes and feedbacks.

REFERENCES

[1] GEGENTANA (2011). A Systematic Review of Automated Software Engineering. Göteborg, Sweden: University of Gothenburg.

[2] Chakraborty A., Baowaly M., Arefin A., Bahar A. (2012). The Role of Requirement Engineering in Software Development Life Cycle. Proceeding of Journal of Emerging Trends in Computing and Information Sciences, VOL. 3, NO. 5.

[3] Terry Quatrani (2001). Introduction to the Unified Modeling Language. IBM.

[4] Alhir, Sinan Si (2003). Activity Diagrams. Learning UML (pp. 156 - 164). Sebastopol, CA.: O'Reilly.

[5] Yue, T, Briand L. C. and Labiche Y. (2010). An Automated Approach to Transform Use Cases into Activity Diagrams. Proceeding of ECMFA'10 Proceedings of the 6th European conference on Modeling Foundations and Applications, pp. 337-353.

[6] Khan, Y. A. and El-Attar M. (2012). Automated Transformation of Use Case Maps to UML Activity Diagrams. ICSOFT 2012, pp. 184-189.

[7] Rodriguez J. G., Nebut C., Cuaresma M. E., Risoto M. M. and Roman I. R. (2008). Visualization of Use Cases Through Automatically Generated Activity Diagrams. MoDELS'08: 11th International Conference on Model Driven Engineering Languages and Systems, Toulouse, France. ACM/IEEE, Model Driven Engineering Languages and Systems (5301), pp.83-96.

[8] Kaewchinporn and Limpiyakorn (2013). Enhancement of Action Description Language for UML Activity Diagram Review. Proceeding of International Journal of Software Engineering and Its Applications Vol. 7, No. 2.

[9] Arman N. and Jabbarin S. (2014). Generating Use Case Models from Arabic User Requirements in a Semiautomated Approach Using a Natural Language Processing Tool. Journal of Intelligent Systems.

[10] Hanan Elazhary (2013). An Interactive System for Arabic Software Requirements Elicitation. Proceeding of International Journal of Scientific & Engineering Research, Volume 4, Issue 12.

[11] Nizar Habash, Owen Rambow and Ryan Roth (2010). MADA+TOKAN: A Toolkit for Arabic Tokenization, Diacritization, Morphological Disambiguation, POS Tagging, Stemming and Lemmatization. Center for Computational Learning Systems, Columbia University, USA.

[12] Kevin Daimi (2001). Identifying Syntactic Ambiguities in Single-Parse Arabic Sentence. Computers and the Humanities 35: 333–349.

[13] Karin C. Ryding, (2005). A Reference Grammar of Modern Standard Arabic, (Reference Grammars) Paperback, Cambridge University Pres.

[14] أبو المكارم، ع. (2007). الجملة الفعلية. القاهرة - مؤسسة المختار للنشر والتوزيع

[15] أبو المكارم، ع. (2007). التراكيب الاسنادية (الجمل: الظرفية - الوصفية - الشرطية) القاهرة - مؤسسة المختار للنشر والتوزيع