The Impact of Native Language on Use Case Modeling: A Controlled Experiment

Sajjad Mahmood
Information and Computer Science Department
King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia
Email: smahmood@kfupm.edu.sa

Samuel A. Ajila
Department of Systems and Computer Engineering,
Carleton University, Ottawa, KIS 5B6, On, Canada
Email: ajila@sce.carleton.ca

Abstract—Software development is a collaborative activity where the quality of the end product depends to a great extent on the quality of the requirements engineering process. Hence, the requirements engineering phases are crucial during the development of a software system. If done incorrectly, it may result in incomplete and inadequate system-to-be functionalities. The requirements engineering processes use natural language for communication with system stakeholders. Requirements ambiguity has been a key area of concern in overall software development process. An attempt to bridge the gap between a pure natural language approach to specifying software requirements and a formal language is the use case modeling approach. To the best of our knowledge, there is no reported evaluation of the impact of stakeholder requirements written in ones native language on the use case modeling approach. In this paper, we present a controlled experiment to investigate the usefulness of using native language (Arabic in this paper) system description on the use case modeling. The results show that using a native language for system description improves the functional correctness and reduces ambiguity related errors in a use case model. However, the time required to perform use case modeling is not affected by using either native or English (as a second language).

Index Terms—Use Case Model, Software Quality, Control Experiment

I. INTRODUCTION

It has been often mentioned in requirements engineering literature (for example, [1]) that a large number of defects are direct result of deficient requirements analysis process and is the most costly ones to fix. Specifying stakeholder requirements in a complete and unambiguous format is fundamental to the success of a software project. Software application stakeholders usually communicate the system requirements in a natural language [2] format and relies less on formal description of the problem. The use of natural language to express requirements introduces the challenge of ambiguous requirements which has a profound impact on the other phases of a software development life cycle. Due to the importance of natural language to the requirements engineering process, a number of natural language processing techniques, ranging from lexical analysis [3] to semantic analysis [4], have been applied to analyze textual requirements. The objective of these techniques has been to overcome the inherent ambiguous nature [3] of natural language, which originates from different possible interpretations of natural language sentences.

Lately, the software industry has also transformed itself from a traditional one location in-house development to a global software development format where teams with different cultural backgrounds collaborate across different continents of the world. These collaborative activities have further enhanced the increasingly important role of requirements engineering in the development of software. The requirements analysis and specification process in itself is a challenging task when done using a single natural language (in most cases, English) environment. It is even more difficult when the stakeholders speak different languages, come from different cultural backgrounds, and when the functional specification documentation is written in a second (or even third) language compared to the native language of the requirements engineers.

Use case modeling is a de-facto industry standard for eliciting and documenting functional requirements [6]–[8]. Use case modeling approach helps in eliciting functional requirements and facilities in reducing overall ambiguity [9], [10] in software requirements documents. The current trends toward globalization and the increasing diversity in software industries require an investigation of the impact of native language on the use case approach. We believe that it is important to investigate whether stakeholder requirements written in a native language for use case modeling can make a positive impact at the requirements specification phase of a software development life cycle.

This paper examines the impact of using stakeholder requirements document written in native language on the use case models. The study is done through a controlled experiment performed at King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. The experiment involved students whose native language is Arabic and have a university level education in English. Furthermore, the participants of the experiment have a
solid understanding of requirements engineering concepts and experience in use case modeling. The motivation of our work is to better understand the impact of a native language (i.e. Arabic) on the 'functional correctness' and 'ambiguity' quality attributes of a use case model. Further, we also investigate the required effort in terms of time for the use case modeling using both native (i.e. Arabic) and second languages (i.e. English).

The contribution of our work is the empirical study to show the potential benefits of using native language for the use case modeling of a system. The results indicate that using native language help in specifying functionally correct use case models. The use of native language to specify stakeholder requirements also helps in reducing the ambiguity related errors in use case models. Furthermore, the use of native language documentation does not require extra effort in terms of time for the use case modeling phase of the requirements process.

The rest of the paper is organized as follows. Section II reviews related literature. In Section III, we discuss the experiment definition, context and hypothesis. Section IV and V presents the experiment design and measurements, respectively. Section VI describes the result analysis, while Section VII deals with threats to the validity of the experiment. Section VIII discusses the observations made during the study. We conclude the paper and discuss future work in Section IX.

II. RELATED WORK

A. Natural Language Processing and Requirements Analysis

Significant research has been done in the application of natural language processing to the requirements phase. The problem of assessing the textural requirements has been approached using linguistic techniques. For example, Fernandes and Cowie [11] used natural language techniques for bridging the gap between informal and formal requirements specification. Kitapci and Boehm [12] used the experienced gained during the development of 'Win Win negotiation model' and 'Easy Win Win requirements negotiation model' to develop an integrated set of gap-bribing methods to formalize informal stakeholder requirements inputs.

Fliedl et al. [4] presented an approach to systematically map natural language requirements specifications to a conceptual model by first linguistically analyzing textual specifications and generating a conceptual schema. Casamayor et al. [13] presented a semi-automated text characterization approach for the identification and classification of non-functional requirements. The empirical study showed that the semi supervised approach resulted in accuracy rates above 70% which are significantly higher than the results obtained with supervised approach. In a later article, Casamayor et al. [14] proposed an approach for mining and classifying functionality from textual descriptions of requirements using text mining techniques. The experimental evaluation case studies show that technique assists software designers in this complex and time consuming task. Lately, Genova et al. [11] presented a framework to identify quality indicators in textual requirements such as requirements size, number of ambiguous terms and overlapping requirements.

Gervasi and Zowghi [15] integrate natural language parsing techniques to automatically discovering inconsistencies in the requirements originating from multiple stakeholders. The method addresses the challenges associated with the direct use of non-monotonic logic for expressing stakeholder requirements. Gnesi et al. [16] proposed an automated tool, namely QuARS, for natural language requirements analysis and validation. QuARS only handles syntax related issues of natural language requirements documentation. ReqSimile [17] is a tool that supports specifying relationship between customer wishes and product requirements. ReqSimile suggests potential linkage by using a standard query retrieval technique. Yang et al. [18] presented an automated approach for detecting nocuous ambiguities using a machine learning algorithm. The algorithm to detect nocuous ambiguities uses a set of heuristics that are based on human judgments.

B. Natural Language Processing and Use Case Models

Natural language processing has also been applied to verify Unified Modeling Language (UML) diagrams including use case models. For example, Meziane et al. [19] proposed a system, called GeNLangUML, to generate natural language specifications form UML class diagram.

Fantechi et al. [3] have applied the natural language processing concepts to identify defects related to ambiguity within requirements documents. They have used the linguistic techniques to semantically analyze the use case models. A use case is analyzed as a set of sentences and metrics such as the average number of words per sentence are collected to perform a quantitative evaluation of a requirements document. Som’e [20] presented an approach to support use case based requirements elicitation, clarification, composition and simulation. The paper presents a restricted form of natural language for use cases that helps in automatic derivation of specification and preserves the readability and understandability of use cases from the stakeholders point of view. Sinha et al. [5] presented a linguistic engine using configurable linguistic components for understanding natural language use case specifications and conducted an empirical study to report on the effectiveness of applying linguistic analysis to use cases.

C. Use Case Models and Quality Attributes

Anda et al. [7] presented an empirical study of the changes made to the use case models in a software project. The results show that most mistakes were related to identifying the actors and use cases. Furthermore, a number of mistakes were made with reference to confusing actors with stakeholders, use cases and domain classes. Rago et al. [21] presented a tool, namely QAMiner, that helps
system analysts to extract quality attributes from use case models. QAMiner uses semantic aspect extractor tool to generate a list of early aspects with crosscutting relations to the use cases and uses predefined quality attribute ontology to derive a list of candidate quality attributes.

Use case map notation has also been used to assist system analysts in bridging the gap between requirements and system design. For example, Mussbacher et al. [22] showed that scenario based aspects can be modeled at requirements level with the help of use case map notation that allows visualization of early aspect with use case map models.

To date, significant research work has been done in the area of requirements ambiguity which has resulted in development of automated tool support for textual specification analysis. Furthermore, research work on use case models has focused on applying natural language concepts to identify ambiguous requirements and improve overall quality of UML models. However, to the best of our knowledge, none of the existing work investigates the impact of using stakeholder requirements written in native language for the use case models.

III. EXPERIMENT PLANNING

This section discusses the experiment definition, context, hypotheses, subject selection, design and result analysis, following the guidelines developed by Wohlin et al. [23].

A. Experiment Definition

The objective of our study is to investigate the impact of native languages on the requirements analysis phase of a software development life cycle. The aim is to assess the impact of the native language (i.e. Arabic) and second language (i.e. English) on the time, functional correctness and ambiguity quality attributes of the use case models.

B. Experiment Context

To analyze the above research objectives, a controlled experiment was conducted with students enrolled in a Bachelor of Software Engineering at King Fahd University of Petroleum and Minerals, Saudi Arabia. The experiment was carried out during a half day workshop. The session was organized as two modules. The first module utilized the first half an hour of the session to recap use case modeling concepts. The second module involved two exercises (two hours each) that constituted the experiment. There was a fifteen minutes tea/coffee break between the two exercises. Furthermore, the students did not know the hypothesis under investigation.

C. Experiment Hypotheses Formulation

The experiment has one independent variable (a system-to-be description document) and two treatments (Arabic as a native language and English as a second language). The experiment has three dependent variables: time to perform the use case modeling (T), functional correctness of the use case model (C) and ambiguity quality of the use case model (A). It is important to note that we did not include ‘language ability’ as part of the control variable because the subjects have the same opportunity from their birth - i.e. they all have the same native language, went to schools with Arabic as a medium of instruction up until entry university; and they all studied English as a second language together in the same language preparatory school inside the university. Hence, we believe that the subjects are equally likely to have the same level of language ability. Figure 1 shows the research model for the controlled experiment.

The hypotheses for analyzing the impact of a multi-language environment on the dependent variables are as follows:

H1-a: A software requirement written in Arabic a native language will require less time to perform the use case modeling compared to a requirements written in English a second language.

H1-b: A software requirements written in Arabic a native language will require more time to perform the use case modeling compared to a requirement written in English a second language.

H1-c: A software requirements written in Arabic a native language will take the same time to perform the use case modeling as a requirement written in English a second language.

H2-a: A software requirement written in Arabic a native language will have positive impact on the correctness of the use case modeling compared to a requirement written in English a second language.

H2-b: A software requirements written in Arabic a native language will have negative impact on the correctness of the use case modeling compared to a requirement written in English a second language.

H2-c: A software requirements written in Arabic a native language will have no impact on the correctness of the use case modeling compared to a requirement written in English a second language.

H3-a: A software requirement written in Arabic a native language will require less ambiguity compared to a requirements written in English a second language.

H3-b: A software requirements written in Arabic a native language will require more ambiguity compared to a requirement written in English a second language.

H3-c: A software requirements written in Arabic a native language will require the same ambiguity compared to a requirement written in English a second language.

Figure 1. Research Model
**H2-b:** A software requirement written in Arabic a native language will have **negative** impact on the correctness of the use case modeling compared to a requirement written in English a second language.

**H2-c:** A software requirement written in Arabic a native language will have **same** impact on the correctness of the use case modeling compared to a requirement written in English a second language.

**H3-a:** A software requirements written in Arabic a native language will have **positive** effect on reducing ambiguity of the use case modeling compared to a requirements written in English a second language.

**H3-b:** A software requirements written in Arabic a native language will have **negative** effect on reducing ambiguity of the use case modeling compared to a requirements written in English a second language.

**H3-c:** A software requirements written in Arabic a native language will have **no** effect on reducing ambiguity of the use case modeling compared to a requirements written in English a second language.

### D. Subject Selection

The experiment involved third and fourth year software engineering students. The third year students had taken a minimum of two software engineering courses with use case related material and had completed two semester projects, which used the use case approach for the requirements analysis process. Furthermore, they were enrolled in a third UML course at the time of the experiment. The fourth year students had taken at least four software engineering courses and had completed four semester projects, which include use case modeling for the requirements analysis phase of a software development life cycle. Furthermore, they had applied the use case approach during the requirements specification phase of their respective software engineering degree capstone projects. All third and fourth year software engineering students were invited to participate in the experiment. In total, 32 students voluntarily agreed to participate in the experiment. All 32 students were included in the experiment because all of them had passed the required courses and hence, we believe that they had the required training to perform the tasks of the experiment.

### IV. EXPERIMENTAL DESIGN

The experiment used two distinct systems: (1) an Automated Teller Machine (ATM) system and (2) a gas station dispensing system. The ATM system had nine use cases and three actors. The ATM system had eight association relationships, one ‘include’ relationship and three ‘extend’ relationships in the ATM system. The gas station dispensing system contains twelve use cases and six actors. The gas station dispensing system has ten association relationships, five ‘include’ relationships and three ‘extend’ relationships. Table I shows details of the systems. Both systems had an English and an Arabic version of the requirements documentation. The Arabic translations of the system requirements were prepared by professional translators to remove any possible biases in the experiment. In each module of the experiment, subjects were given the system description document (either in English or Arabic) of the respective systems and were required to develop a use case model.

<table>
<thead>
<tr>
<th>Dependent variable</th>
<th>ATM System</th>
<th>Gas Station Dispensing System</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of Use Cases</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>No. of Actors</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>No. of Association relationships</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>No. of &lt;&lt;include&gt;&gt; relationships</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>No. of &lt;&lt;extend&gt;&gt; relationships</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

In our experiment, we used the concept of blocking to mitigate the effect of individual and group abilities. The blocking was based on the number the software engineering courses taken by the participants. As a result, two blocks were considered according to whether a student had taken three or more software engineering courses. There were 15 students in the first block (three software engineering courses) and 17 students in the second block (four or more software engineering courses), respectively. The students in each block were randomly divided into two groups (Group A, Group B) of 16 subjects respectively. In the first exercise, Group A started with the ATM system’s English version document and Group B worked on the Arabic version document. In the second exercise, Group A worked on the gas station dispensing system’s Arabic version document and Group B worked on the English version document. This ensured that neither of the two systems would benefit from learning effects. The experiment design is summarized in Table II.

<table>
<thead>
<tr>
<th>Workshop Modules</th>
<th>Group A</th>
<th>Group B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>Use Case Modeling Recap</td>
<td>Use Case Modeling Recap</td>
</tr>
<tr>
<td>Exercise 1</td>
<td>ATM System (English)</td>
<td>ATM System (Arabic)</td>
</tr>
<tr>
<td>Exercise 2</td>
<td>Gas Station Dispensing System (Arabic)</td>
<td>Gas Station Dispensing System (English)</td>
</tr>
</tbody>
</table>

---

1At King Fahd University of Petroleum and Minerals, all software engineering capstone projects have industrial sponsors.
V. MEASUREMENT

For each exercise, subjects were given the relevant system description document and were asked to draw the use case model. In our experiment, a list of potential mistakes based on the existing use case literature [7], [10] were used to validate all the use case models. Table III shows the use case attributes and their respective common mistakes. All the solutions were manually inspected to assess the correctness and ambiguity quality attributes of the use case model. Correctness is measured as a ratio of the number of correctness common mistakes over the total number of potential correctness mistakes for a given use case model. Similarly, ambiguity quality is measured as a ratio of the number of ambiguity quality common mistakes over the total number of potential ambiguity quality mistakes for a given use case model. Furthermore, after completion of each exercise, the subjects fill in a questionnaire to collect data about the estimated time spent on the use case model; and their respective views on the strengths and weakness of using the given language (i.e. Arabic or English) to describe the system-to-be.

TABLE III.
USE CASE ATTRIBUTES

<table>
<thead>
<tr>
<th>Use Case Attribute</th>
<th>Description</th>
<th>Common Mistakes</th>
</tr>
</thead>
</table>
| Correctness        | Customer requirements are correctly represented in the use case model. | (1) Missing use case  
(2) Missing actor  
(3) Missing relationship between use cases  
(4) Missing relationship between use case and an actor. |
| Ambiguity          | There is no ambiguity in the use case description and use of terminologies. | (1) Actors names do not reflect their role  
(2) Use case names do not reflect corresponding requirements goals  
(3) Ambiguous description of pre and post conditions  
(4) Ambiguous main and alternative flows of use case |

VI. RESULT ANALYSIS

In this section, we analyze the data collected during the experiment to validate each of hypotheses defined in section III-C. As part of our analysis, we assume that the hypothesis testing significance level is 0.05 ($\alpha = 0.05$). A box-plot graph is used to illustrate the analysis results for both Arabic and English documents. Furthermore, Table IV shows the descriptive statistics of the experiment.

A. Time to Perform the Use Case Modeling

We analyze the time required to perform the use case modeling using native (i.e. Arabic) language and second (i.e. English) language system description documents. The use case modeling time for the Arabic system document ranged from 36 to 53 minutes. Similarly, the time required for use case modeling using the English language document ranged from 37 to 52 minutes. The mean times for the Arabic and English language documents were 42.8 and 43.6, respectively. Figure 2 shows that the subjects using the second language (i.e English) document spent slightly more time on use case modeling than those using the native language (i.e. Arabic) document.

![Figure 2. Time to Perform Use Case Modeling using Arabic and English documents](image)

The mean times for the ATM system with native and second language documentation were 39.8 and 40.6, respectively. The mean times for the gas station system with native and second language documentation were 45.75 and 46.6, respectively. The t-test and Mann-Whitney U-test indicate that the difference is not statistically significant ($p$-value = 0.457). The results indicate that there was no significant difference between the times of different groups with native and second language documents. Thus, based on the above analysis, we accept the hypothesis (H1-c) and conclude that the use of native and second language documents does not have any significant effect on the time required to develop use case models for a software system.

B. Use Case Model - Functional Correctness

In this subsection, we analyze the impact of native and second language system-to-be description documentation on the correctness of a use case model. For the ATM system, the mean number of correctness mistakes when
using Arabic and English description documents is 3.75 and 5.2, respectively. Similarly, for the gas station system, the mean number of correctness mistakes when using Arabic and English language description documents is 4.6 and 6.4, respectively. Figure 3 shows that analyzing the requirements of a system-to-be in a native language (i.e. Arabic) helps in reducing the number of functional correctness mistakes compared to using a second language (i.e. English) to describe the system-to-be requirements. The mean number of correctness mistake count for native and second language based system-to-be description documents is 4.2 and 5.8, respectively.

C. Use Case Model - Ambiguity

We assess the impact of native (i.e. Arabic) language and second (i.e. English) language system description documents on the ambiguity quality of a use case model. For the ATM system, the mean number of ambiguity quality related mistakes for using Arabic and English description documents was 2.12 and 2.75, respectively. Similarly, for the gas station system, the mean number of ambiguity quality related mistakes when using Arabic and English language description documents was 3.18 and 3.94, respectively. Furthermore, the mean number of ambiguity quality related mistakes for both the ATM and gas station systems was 2.66 and 3.4, respectively. Figure 4 shows that subjects using native language document made a smaller number of ambiguity quality related mistakes during use case modeling of both the ATM and gas station systems.

![Figure 3. Correctness - Arabic Vs English](image)

![Figure 4. Ambiguity - Arabic Vs English](image)
second language documents. Thus, based on the above analysis, we accept the hypothesis (H3-a) and conclude that a system description written in a native language (i.e. Arabic) does have a positive impact in decreasing the ambiguity quality related mistakes in the use case models compared to a system description written in a second language (i.e. English).

D. Qualitative Analysis

This subsection summarizes the strengths and limitations as indicated by participants of the experiment. Information on the participants’ experience was collected from four open-ended descriptive questions, namely, strengths of using a native language (i.e. Arabic) for requirements analysis, weaknesses of using a native language (i.e. Arabic) for requirements analysis, strengths of using a second language (i.e. English) for requirements analysis and weaknesses of using a second language (i.e. English) during the use case modeling phase of a software development life cycle.

The feedback from the subjects indicated three main strengths of using a native language for use case modeling: (1) it is easy to understand a system’s description in one’s native language due to having a strong vocabulary; (2) ambiguity is reduced due to better comprehension skills in a native language; and (3) it is possible to gain a better understanding of a system-to-be due to spending more time on actually understanding the system’s features. The subjects also indicated that it is relatively easy to identify actors and develop use case models using native language system descriptions.

The subjects who participated in the experiment indicated that the main advantage of English lies in it being the de-facto standard for technical terminologies. English also has the added advantage of being the global language and a significant amount of technical literature is available on the use case model approach. It is also important to have a solid background in English grammar and vocabulary to understand system descriptions. The participants indicated that they prefer to translate new and difficult words into their native language to better understand the text. We believe that these concerns reinforce the importance of our experiment to assess the impact of multi-language environments on the requirements analysis phase of a software development life cycle.

VII. Threats to Validity

We believe the empirical study does not suffer from the construct validity threat because the controlled experiment was internally pre-tested with two graduate students to ensure that all tasks were meaningful. An overview of use case modeling was presented before the experiment. Furthermore, we believe that the experiment does not suffer from an insufficient time threat as all subjects completed their tasks within the allocated time. We believe that time is not a confounding factor in our study.

The study has no serious internal validity threat because the experiment was designed through the blocking and counter-blocking process. Another threat to internal validity is possible ambiguity in the questions. To avoid these misunderstandings, the first author was available during the study to answer any questions regarding the tasks. Furthermore, all subjects participated on a voluntary basis which helps in avoiding any morality threats. In this study, we used standard statistical techniques to either accept or reject the null hypotheses. We used both t-test and non-parametric Mann-Whitney test techniques to validate our results. Furthermore, Table IV presents the descriptive statistics of the experiment.

The inherent limitations of control experiments lie in their external validity because they are conducted with students in an academic environment. Hence, it is not easy to generalize these results for all domains. In our experiment, the participating students are well trained in use case modeling and have a sound knowledge of the software engineering development life cycle. We believe that the results of the experiment can be generalized for small sized applications where requirements engineers are involved in analyzing a requirements document for a software system.

VIII. Discussion

From the experiment results, we make a few useful observations as follows:

- Time to perform use case modeling: It is important to correctly understand the stakeholder’s needs for delivering a software system on time and within the allocated budget. The result analysis shows that using a native language slightly decreases the time required for use case modeling. It is important to note that the decrease is not statistically significant. Further, the results show that a multi-language environment does not negatively impact the overall effort required in the use case modeling phase of a software development life cycle.

- Functional correctness of a use case model: The functional correctness of a use case model is crucial for delivering a quality software application. As shown in our study, on average, using a native language for system description improves the functional correctness of a use case model. This is due to the fact that system analysts are better at understanding system requirements when there is no need to translate and introduce potential ambiguities during the requirements analysis process.

- Ambiguity Quality of a use case model: Our study shows that the ambiguity related mistakes in a use case model are decreased with using native language and hence, the hypothesis (H3-a) is accepted. Fundamental to overall quality of a use case model is the description of the system without any references to the design details. Hence, the quality of a use case model depends on the ability of system analysts to understand the requirements written in natural language and the native language can play an important role.
role in decreasing the ambiguity related issues of use case models.

- **Usefulness of results:** The results provide empirical evidence to highlight the impact of natural languages on the software requirements process and reinforce the general perceptions about the importance of natural languages in requirements elicitation and analysis. The results provide an insight to the role of multi-language environments on the use case modeling phase. In future, we plan to use these results to develop a requirements analysis readiness framework to assist software developers in improving their requirements analysis readiness prior to starting projects in multi-language environments.

- **Impact of Experience on the Use Case Model:** The study only evaluates the impact of native and second languages on the use case model. There is a need to study the correlation between experience and multi-language environments on the requirements elicitation and analysis of a software system. It is also important to investigate whether native languages facilitate experienced system analysts to overcome a potential lack of coherence in the testing of a system description document. We believe that it is important to further evaluate whether domain knowledge helps in producing high quality use case models and helps system analysts to overcome potential ambiguities in natural language descriptions.

**IX. Conclusions and Future Work**

This paper presents the results of a controlled experiment with Bachelor of Software engineering students. The goal was to assess the impact of native and second languages in the use case modeling process of a software development life cycle. We focus on whether requirements analysis in a native language help system analysts to make functionally correct and less ambiguous use case models. We also analyze the time required to carry out the use case modeling using both native and second languages. The results indicate that using native languages helps in improving the functional correctness of a use case model. Similarly, the ambiguity related mistakes in the use case model does decrease by using the native language in the use case based requirements analysis and specification process. However, there is no significant savings in the time between using native and second languages documents.

The feedback from the subjects indicates that native language documents facilitate a system analyst in assessing stakeholder needs and helps them better understand the functions of the system-to-be. The subjects participating in the controlled experiment also indicate that second language descriptions are prone to cause a misunderstanding of the requirements of a system-to-be due to one’s limited vocabulary and grammar skills. Furthermore, results of this initial experiment indicate the need for a theoretical framework to identify the factors that can contribute to success in multi-language global software development.

For future work, there is a need to conduct industrial surveys to better understand the correlation between system description languages and the requirements elicitation and analysis process. There is also a need to better understand the impact of experience on the requirements analysis phase and its impact in multi language environments. This will provide insight into how native language skills can be used to improve the requirements elicitation and analysis phase of a software development life cycle.

**Acknowledgement**

The authors would like to thank King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia for its continuous support in research. We thank Fahmi H. Qurada’a for preparing Arabic documentation. The author, Samuel A. Ajila, will also like to thank the information and computer science department for the support received during his sabbatical year (2008 - 2009).

**References**


Samuel A. Ajila is an Associate Professor of Engineering at Carleton University, Ottawa, Canada. He obtained his PhD in Software Engineering and Knowledge-based Systems from LORIA, Université Henri Poincaré Nancy I, Nancy, France. He is a professional engineer (P.Eng) in the province of Ontario, Canada. He was at the King Fahd University of Petroleum and Minerals (KFUPM), Dhahran, Saudi Arabia in 2008/09 as part of his sabbatical leave. He is a Member of IEEE, ACM, International Association for Management of Technology (IAMOT), and PICMET. His research interests are in fields of software requirement engineering, software metrics, reengineering, product and process evolution, product lines engineering and cloud computing (SLAs and VMs management).

Sajjad Mahmood received his PhD in computer science, from La Trobe University, Melbourne, Australia. He is an assistant professor in the Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia. His research interests are software reuse, component-based software engineering and software product lines.