International Journal of Engineering & Technology, 7 (2.3) (2018) 46-49

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research Paper

Identify the effective documentation method for representing the functional software architecture

Iyas Ibriwesh[1*] , Sin-Ban Ho[2], Ian Chai[3]

[1, 2, 3]Faculty of Computing and Informatics, Multimedia University, 63100 Cyberjaya, Selangor, Malaysia
**Corresponding author E-mail: ibrawish2012@gmail.com*

Abstract

Software architecture mainly focuses on the high-level structures of the proposed software, and how to document these structures. A documentation method that represents an incomplete picture is one reason for inadequate requirements. This leads to requirements engineers wasting their time arguing over what to do and how to do it. Four documentation methods are frequently used in order to document stakeholders' statements, particularly for representing the functional perspective, namely, Natural Language (NL), Data Flow Diagram (DFD), Use Case Diagram (UCD), and Activity Diagram (AD). This research was carried out using the electronic market application domain as a test context. A controlled experiment was conducted among 158 participants, comparing among NL, DFD, UCD, and AD methods, which aimed to find out which requirements documentation method is more effective, helpful, and easier to comprehend. The results from this empirical study reveal that the AD method is more effective, understandable, and easier to document the software requirements in the functional perspective. Furthermore, AD had better performance in representing the requirements engineering context, system context, and development context than the other functional documentation methods. These empirical results would help software companies and associated experts enhance the quality of their software products, as well as increase the chance of success of software projects.

Keywords: *Software Architecture, Requirements Documentation, Controlled Experiment, Requirements Engineering Context, Functional Perspective.*

1. Introduction

Software system projects still suffer from shortcomings in the requirements engineering (RE) phase. Approximately 60% of all mistakes that occurred in the software development projects occur in the RE phase [1]. When an error occurred in this phase, the project's budget and schedule will likely overrun. Therefore, RE is considered the hardest and the most important phase in the software development life cycle [2].

Software practitioners depend on documents as the main communication medium. So, effective communication and collaboration between the system's developers and end users is necessary to develop a successful software product with high quality [3]. Poorly documented requirements is considered one of the most important factors causing failure of software projects [4]. If the requirements engineers could not understand what the customer really needs, this leads them to create incorrect diagrams, and subsequently, the software developed would likely fail and be rejected by the customer.

The main goal of this study is to improve the requirements documentation activity, by identifying the best method that should be used for representing the functional perspective in the software to be developed. This could be done by comparing among the different requirements documentation methods (the most common methods) that are used in the functional perspective. The functional documentation methods are the Natural Language (NL), the Data Flow Diagram (DFD), the Use Case Diagram (UCD), and the Activity Diagram (AD) [1]. This comparison aims to determine which method is easier for the participants to read, understand, and use, as well as which method has better ability to represent the requirements engineering context.

This paper starts with requirements documentation methods that are widely used in the functional perspective, which are highlighted in Section 2. The experimental hypotheses and methodology on how the study was conducted are explained in details in Section 3. Section 4 represents the experimental participants' knowledge and characteristics. The statistical analysis, results, and discussions are presented in Section 5. At the end of this paper, the conclusions of this study are presented in Section 6.

2. Documenting software architecture

Functional, data, and behavioural perspectives should be documented separately, using suitable conceptual modelling languages [1]. This empirical study concentrated on the functional perspective, which documents which information of the system context is being manipulated by the proposed system and which data is being transmitted to the system context by the system. DFD, UCD, and AD are commonly used as requirements models to document software requirements in the functional perspective. Furthermore, NL is widely applied to document any kind of requirements, and it is frequently used to model the functional perspective of requirements [1].

3. Experimental design

A controlled experiment has been carried out, in which the E-market application domain, online flea market system (OFMS)

was used as a test context. It is an exercise-based research which is usually applied in empirical software engineering. After the documentation and the exercises were prepared, based on the hypotheses, they were checked for usability and readability before they were ready to be used for data collection. The collected data was then entered and analysed using the proper statistical techniques.

3.1 Hypotheses

The null hypothesis (H0) or expectation that the researchers investigated in this empirical study is stated as follows:
There are no significant differences among NL, DFD, UCD, and AD methods for the participants in the identification of requirement sources exercise; regarding to the subject facet, usage facet, IT facet, sum of system context, development context, requirements engineering context, sum of requirements sources, time spent, and number of difficulties faced. The experiment interpretations are derived from rejecting or accepting this hypothesis for each expectation.

3.2 Exercise

The evaluation approach used in this study is an exercise or examination rather than a survey of opinions. Participants were asked to record the start and end time during answering the exercise, in order to measure the time they needed to finish the comprehension tasks. Each of the participants was assigned randomly to one of the four groups, as shown in Table 1. Each group was asked to finish this task within two hours. After the controlled experiment, the researchers counted the number of the correct answers for the exercise respondent. All the four versions of the system specification with different experimental diagrams were prepared by the researchers, and then they checked the correctness and readability of the exercise. The participants were introduced to the domain by providing them with a small description of the Online Flea Market System (OFMS) domain.

3.2.1 Identification of requirement sources exercise

The philosophy that was used in this experiment is based on the participant's ability to elicit the requirements (identification of requirement sources) using a given method from the chore of DFD, UCD, AD, and NL. This process evaluates the effectiveness and efficiency of each method in representing the requirements engineering context, the system context (the subject facet, the usage facet, and the IT system facet), as well as the development context. Each participant was given a specification of the same system, which was expressed using one of four methods (NL, DFD, UCD, and AD). Each of them was also provided with a small piece of text to define the online flea market system (OFMS) domain in general

3.2.2 Variables of the exercise (identification of requirement sources exercise)

One independent variable as well as nine dependent variables, were used in this experimental design, are explained below.
Independent variable: method used (type): Four methods were used in order to document the software requirements (NL, DFD, UCD, and AD), each with the same purpose.
Dependent variables: the experimental design involved nine dependent variables, which may be affected by the used documentation method. The dependent variables are: subject facet, usage facet, IT system facet, sum system context, development context, requirements engineering context, sum of the whole exercise, number of difficulties faced, and time spent. These dependant variables were used for measuring the comprehension level (the correctness of the answers), and the ease of use (the number of difficulties and the required time to finish the task).

4. Experimental participants

Evaluating an initial hypothesis could be useful when done at a university environment rather than conducted in the industrial setting. Reasons for that were mentioned by [5]. He stated that it is more cost-effective to use students as participants, as well as students are available in a sufficiently large number. In this study, all the participants groups were undergraduates at the Multimedia University, who are undergoing the "software engineering three year specialization program". These participants were taking the software requirements engineering course, and they will be graduated as software engineers soon.
The experimental exercises were executed in four laboratories, each of which was a software engineering laboratory for one group (session). In the laboratory, each participant was located at a computer randomly, the participants were given the exercise forms (the experimental diagrams) as well as they were provided with an online tutorial (documentation), and they were requested to go through it and follow its content, to perform their tasks and fill up the exercise answers. The study involved a total of 158 participants, 122 (77 percent) males and 36 (23 percent) females. The mean age of the participants is 21.8 years. All the study participants (n=158) have done the exercise, 37 participants did the exercise using NL method, 35 did it using the DFD method, 40 did it using the UCD method, and 46 did it using the AD method, see Table 1.

Table 1: Participants Groups and their Used Methods

Identification of Requirement Sources Exercise		
Groups	Number of Participants	Used Method
Group 1 (Session 1)	37 participants	NL
Group 2 (Session 2)	40 participants	UCD
Group 3 (Session 3)	35 participants	DFD
Group 4 (Session 4)	46 participants	AD

Table 2 affirms that the four participating groups are balanced in regards of their cumulative grade point average (CGPA). This was found out by checking the groups' differences using both chi-square test that was used to check for the gender, and ANOVA Test that was used to check for the CGPA. These tests emphasized that there are no significant differences among the groups in their CGPA due to a p-value was higher than 0.050. Thus, this result indicates that the participants' achievements (knowledge and backgrounds) were balanced during conducting this experiment.

Table 2: Characteristics of the Study Participants

Characteristics		NL N=37	DFD N=35	UCD N=40	AD N=46	p-value
Gender	Male N (%)	32 (87)	27 (77)	27 (68)	36 (78)	0.28
	Female N (%)	5 (13)	8 (23)	13 (32)	10 (22)	
CGPA	Min-max	2.03 - 3.97	2.34 - 3.91	2.09- 3.75	2.09 - 3.85	0.08
	Mean (SD)	3.2 (0.51)	2.9 (0.44)	3.0 (0.36)	2.9 (0.38)	

5. Results and discussion

5.1 Statistical analysis

Statistical Package for Social Sciences (SPSS) was used for entering and analyzing the data. For achieving the study aim, the variables were compared using either One-Way ANOVA Test, for comparing the four methods for normally distributed variables, or Kruskal-Wallis Test, for non-normally distributed variables (non-

parametric test). A p-value that is lower than 0.050 is considered statistically significant.

5.2 Experimental results

The compared dependent variables are: the subject facet, usage facet, IT system facet, sum system context, development context, requirements engineering context, sum of the whole exercise, number of difficulties faced, as well as time spent.
The normality for these dependent variables was tested; all these dependent variables except the number of difficulties and the time spent were normally distributed for every participant group. Therefore, we decided to use the One-Way ANOVA Test to compare all normally distributed variables (Table 3), and the Kruskal-Wallis test (non-parametric test) to compare the number of difficulties and time spent (Table I).
Comparison results showed that the AD group had scored statistically significantly higher marks in all of the exercise questions (sum system context, development context, requirement engineering context, and sum of the whole exercise). This indicates that AD can be more effective in representing these dimensions for developing the proposed system, as shown in Table 3 and Fig. 1.

Table 3: Comparison among NL, DFD, UCD, and AD for Normally Distributed Variables using One-Way ANOVA Test

Variables	NL (N=37) Mean (SD)	DFD (N=35) Mean (SD)	UCD (N=40) Mean (SD)	AD (N=44) Mean (SD)	p-value
Subject Facet	1.9 (0.59)	1.7 (0.70)	2.4 (0.74)	**2.6** (0.72)	< 0.001 *****
Usage Facet	1.9 (0.79)	1.2 (1.04)	2.5 (0.91)	**2.7** (0.82)	< 0.001 *
IT-System Facet	1.0 (0.47)	0.9 (0.42)	2.2 (0.84)	**2.4** (0.94)	< 0.001 *
Sum of System Context	4.8 (1.22)	3.9 (1.28)	7.1 (1.43)	**7.7** (1.83)	< 0.001 *
Development Context	1.3 (0.67)	0.9 (0.69)	2.3 (0.79)	**2.7** (0.87)	< 0.001 *
Requirement Engineering Context	1.1 (0.66)	0.7 (0.68)	1.7 (0.56)	**1.8** (0.51)	< 0.001 *
Sum of Whole Exercise	7.2 (1.46)	5.4 (1.80)	11.1 (1.91)	**12.3** (2.57)	< 0.001 *

Note: "Bold-faced fonts to represent the best achievement; * Statistically significant at 0.050 level"

Furthermore, the AD group has faced a statistically significant lower number of difficulties than other groups. This indicates that AD is easier and faster to understand than other methods, as shown in Table 3. The DFD group performed the task faster than the other groups, but this result was not statistically significant as confirmed in Table 4. Fig. 1 shows the mean of scores in each dependent variable.

Table 4: Comparison among NL, DFD, UCD and AD for non-Normally Distributed Variables using Kruskal-Wallis Test

Variables	NL (N=37) Median (SD)	DFD (N=35) Median (SD)	UCD (N=40) Median (SD)	AD (N=44) Median (SD)	p-value
Number of Difficulties	2.0 (2.32)	5.0 (2.94)	2.0 (0.85)	**1.0** (0.72)	< 0.001*****
Time Spent (mm:ss)	28:19 (08:31)	**25:33** (08:56)	28:50 (17:55)	28:18 (09:47)	0.608

The overall study results reveal that the AD method is more effective and helpful than the other functional documentation methods. This finding supports that AD could be clearer to understand, and performs better in representing the functionality of the proposed system, in the context of RE.

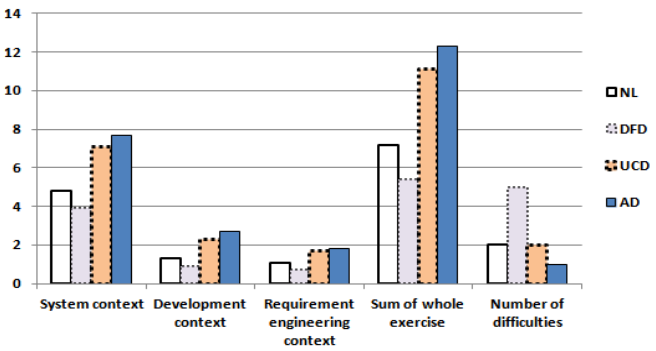

Fig. 1: Comparing the mean of the scores in each exercise section among NL, DFD, UCD, and AD methods.

Comparing NL with DFD, the NL group scored higher than the DFD group, and had fewer difficulties. Ibrahim et al. [6] reported that the natural language is commonly used for documenting the stakeholders' statements in the requirements elicitation activity. In addition, it has also been found that using the language of the customers to describe the software requirements is most effective in gaining the customers understanding and agreement [7].
The DFD appeared to produce worse results than NL, UCD, and AD. Furthermore, DFD has the lowest score in capturing the system context (the subject facet, the usage facet, and the IT facet), the development context, and the requirements engineering context, as shown in Fig. 1. In addition, the DFD group had the highest number of difficulties, as shown in Fig. 1.
This study finding is supported by [8] as they emphasized that AD represents a readable model and allows a hierarchical decomposition by using sub activity states. In addition, Dumas and Hofstede [9] reported that activity diagrams have the ability to capture situations that emerging in practice, in which most of the commercial work flow management systems cannot capture these situations. One of the main advantages of AD is that AD contains the explicit flow of logic of the software system [10]. Furthermore, Hnatkowska and Grzegorczyn [11] conducted a controlled experiment comparing NL, AD, and UCD. The results of their experiment confirmed that AD has a less ambiguous interpretation than the NL and UCD. AD specifications produced more correct interpretations than UCD specifications. This supports our finding that the AD method can be more effective than the other methods in the context of the electronic market application domain.
In addition, the results indicate that the UCD does not perform as well as the AD, as shown in Fig. 1. These results agree with [12] as they confirmed that the UCD has sufficient ability to represent all the development activities, as well as the interaction between the software system and its environment.

6. Conclusion

The experiment result reveals that the AD participants had significantly higher scores and lower number of difficulties than those who used the other functional documentation methods. In addition, AD had the best performance in representing the requirements engineering context, system context, and development context. Therefore, the activity diagram (AD) is the best documentation method for representing the functional software architecture. These results will help requirements engineers to select a better functional documentation method for documenting the stakeholders' requirements.

Acknowledgment

The authors would like to thank the experimental subjects for their cooperation, and the Multimedia University (MMU) for financial and technical support. Additionally, we would particularly like to thank Dr. Maysaa Nemer for her help and support.

References

[1] K. Pohl and C. Rupp, Requirements Engineering Fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level – IREB compliant, 2nd ed., USA: Rocky Nook Inc, 2015.

[2] H. S. Jabbar and T. V. Gopal, "Qualitative analysis model for software requirements driven by interviews," J. Eng. Appl. Sci., vol. 2, no. 1, pp. 1–9, 2007.

[3] C. S. Murugan and S. Prakasam, "Stakeratreet: a new approach to requirement elicitation based on stakeholder recommendation and collaborative filtering," Res. J. Appl. Sci., vol. 10, no. 10, pp. 574–586, 2015.

[4] K. A. Michael and K. A. Boniface, "Inadequate requirements engineering process: a key factor for poor software development in developing nations: a case study," Int. J. Comput. Electr. Autom. Control Inf. Eng., vol. 8, no. 9, pp. 1462–1465, 2014.

[5] S-B. Ho, "Framework documentation with patterns: an empirical study," PhD thesis, Multimedia University, Cyberjaya, Malaysia, Feb. 2008.

[6] N. Ibrahim, W. Kadir, and S. Deris, "Documenting requirements specifications using natural language requirements boilerplates," in Proc. MySEC'14, 2014, p. 19–24.

[7] T. Bures, P. Hnetynka, P. Kroha, and V. Simko, "Requirement specifications using natural languages," Charles Univ., Prague, Czech Republic, Tech. Rep. D3S-TR, 2012.

[8] J. Barros and L. Gomes. (2000) From activity diagrams to class diagrams. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.34&rep=rep1&type=pdf

[9] M. Dumas and A. H. M. Hofstede, Ed., The Unified Modeling Language. Modeling Languages, Concepts, and Tools: UML Activity Diagrams as a Workflow Specification Language, ser. Lecture Notes in Computer Science. Toronto, Canada: Springer, 2001, vol. 2185.

[10] D. Kundu and D. Samanta, "A novel approach to generate test cases from UML activity diagrams," J. Object Technol., vol. 8, no. 3, pp. 65–83, 2009.

[11] B. Hnatkowska and M. Grzegorczyn, "Empirical comparison of comprehensibility of requirement specification techniques based on natural languages and activity diagrams," in Proc. MSVVEIS'12, 2012, p. 27-36.

[12] R. Boudour and M. T. Kimour, "Model transformation for requirements verification in embedded systems," Asian J. Inf. Technol., vol. 4, no. 11, pp. 1012–1019, 2005.