International Business Management 10 (17): 3812-3820, 2016
ISSN: 1993-5250

An Empirical Investigation of Methods for Maximizing Knowledge and Performance in Framework Documentation Using Use Case Maps (UCM)

[1]Hee Wei Loon, [1]Ian Chai, [2]Tan Chuie Hong and [1]Ho Sin Ban
[1]Faculty of Computing and Informatics, Multimedia University,
Persiaran Multimedia, 63100 Cyberjaya, Selangor, Malaysia
[2]Faculty of Management, Multimedia University, Persiaran Multimedia,
63100 Cyberjaya, Selangor, Malaysia

Abstract: In today's world of software engineering, timeframe and design consistency of a project have become a major, if not, crucial concern among developers. When developing an application, developers cannot afford to exceed the timeframe that has been allocated but at the same time, need to maintain the consistency of design throughout the whole development process. One way to support this is to have good framework documentation. Framework documentation is a process whereby a framework is documented in a proper manner so that it could be easily reused in any future application development. Two main documentation styles have been included in this study-patterns and minimalist styles. It is not easy to document a framework well. Therefore, well-established theories or rules to follow when documenting are helpful. The aim of this study is to discover a new method that can improve the learning curve and to make mass learning possible among users in framework documentation. It focuses on minimalist documentation and pattern documentation but with a new add-on approach-augmented with Use Case Maps (UCM). Patterns have the idea of providing the solution to a problem in its context. It has contextual information which is not minimal, unlike minimalist. Minimalist documentation promotes minimal reading and random access which m ans that the learner can proceed in a self-directed manner as well as using small pages that can be read as per the learner's desires on the reading order. The use of UCM in these documentation styles has been examined in detail using a quantitative research method. We found UCM to be more effective in allowing learners to perform efficiently and understand the framework better when used in conjunction with Minimalist documentation.

Key words: Framework documentation, patterns, minimalist, UCM, Malaysia

INTRODUCTION

A framework is a collection of reusable parts that can be reused for a particular entity (Zamir, 1998). This framework can be reused or manipulated into new framework designs when there is a need to do so. Eventually, this framework allows tasks at hand to be executed effectively, efficiently and consistently without any major flaw.

This framework concept works effectively in Object-Oriented Programming (OOP). The framework is expressed in code. In order to reuse a framework to build new applications, one has to document them in a proper manner so that they can be easily understood when needed later for application development. In the world of software engineering today, timeframe and consistency of a project have become a major, if not, crucial concern among developers. When developing an application, developers cannot afford to exceed the timeframe that has been allocated and at the same time, should maintain the consistency of design throughout the whole development process. If one of the above criteria is not fulfilled, the whole project development might be in jeopardy and finally fail. So, one question is how to make the application development process a success with minimal flaws in terms of timeframe usage and consistency of design? In order to achieve that, framework documentation plays an important role in today's OOP development to ensure the application development process meets its timeframe and achieves consistency in its design as well (Fayad and Schmidt, 1997). This is the most effective and efficient way to develop an application.

Last but not least, there is a lot of research being carried out throughout the world today in order to

Corresponding Author: Hee Wei Loon, Faculty of Computing and Informatics, Multimedia University, Persiaran Multimedia, 63100 Cyberjaya, Selangor, Malaysia

improve or further enhance current framework documentation theories such as John Carroll's Minimalist theory, etc.

The focus of this study is to maximize knowledge and performance in framework documentation using Use Case Maps (UCM). The use of UCM in these documentation styles has been examined in detail using a quantitative research method. We found UCM to be more effective in allowing learners to perform efficiently and understand the framework better when used in conjunction with Minimalist documentation.

Literature review: It is not easy to create good framework documentation (Aguiar and David, 2000). A framework is basically useless if the learners are not able to understand and utilize it to meet their objectives. Thus, well-established theories or guidelines to follow are needed when documenting a framework. By having good framework documentation, the learners will be able to understand and know exactly how to apply it in their work, hence, making the learning process a success. This research uses several pedagogical framework documentations methods. The two main documentation styles being used are patterns and minimalist. We add Use Case Maps (UCM) to these styles to study if it enhances their pedagogical utility. The benefits and reasons of having UCM will be discussed later in this chapter. Therefore, in this research, we will have four documentation styles-patterns, patterns with UCM, minimalist and minimalist with UCM.

In today's information technology world, each piece of software has its own dedicated documentation. Most documentation tells people how to perform the task, instead of telling them the purpose of the task itself and why they need to perform the task (Dix *et al*., 2004). This indirectly decreases the understanding of the readers and eventually prevents them from having the ability to reuse the same experience in other computer software. This is one of the main drawbacks of traditional documentation. To address this concern, patterns were introduced. One of the objectives of Patterns is to allow readers to understand the rationale behind a solution-why a particular task is required, so that they can have the ability to make their own judgement on when to apply the pattern. Patterns are proven successful solutions for recurring problem in a context (Schumacher, 2003). According to Meszaros and Doble (1996), patterns should have these attributes:

- Pattern name: for tracking purposes so that reader can refer to the pattern easily
- Problem: a precise and detailed description of the problem it solves
- Solution: a detailed description of the solution to the problem and a prescription on how the solution works to solve the problem
- Context: the circumstances of the problem that provide constraints on the solution
- Forces: common contradictory considerations that must be taken into account when choosing a solution to a problem

Minimalist documentation was pioneered by Dr. John Carroll based on two concepts, firstly, information which is irrelevant to the task at hand is not needed, and secondly, step-by-step instructions are hard to follow. Carroll (1998) created this Minimalist documentation to give the reader a minimal amount of information to read to accomplish the task. At the same time, it uses short pages or index cards of information and allows readers to read them in any order they prefer. Carroll gives some guidelines for minimalist documentation as follows:

- Training on real tasks: readers are more determined and motivated to complete a task if it is something they want to achieve
- Getting started fast: readers will lose interest easily if there is too much to read before performing a task and this will cause them to miss things easily
- Reading in any order: topics are brief and short which allow readers to choose the order they prefer and suit them the best
- Coordinating system and training: allow readers to interact with the system instead of providing detailed steps to guide them
- Supporting error recognition and recovery: instead of assuming readers follow step-by-step instructions flawlessly, expect them to fail and provide resources for them to understand how to recover
- Exploiting prior knowledge: exploiting prior knowledge by using analogies of readers' prior experience to help them understand, instead of using jargon
- Using the situation: take advantage of the readers' expectations that are brought to the situation

In today's world of programming, requirement gathering and precise software engineering are very crucial to ensure software runs correctly and more importantly, meet the objective of creating the program. According to Amyot and Mussbacher (2001), several common issues have been identified for requirements gathering:

- Early focus on low-level abstractions
- Specific requirements and high-level decisions buried in the details
- Change of functionalities hard to manage and
- Introduction of new services are delayed
- They have also identified common issues faced in software engineering
- Difficulties to come out with requirements or analysis models required to support new dynamic systems
- The need to change from requirements or analysis models to design models seamlessly

Use Case Maps (UCM) tackles these issues. UCM is a notation to illustrate a scenario path relative to optional components that may exist in a scenario. It describes causal relationships between responsibilities of use cases in a map-like diagram.

MATERIALS AND METHODS

The experiments use learning Java ME as a test bed task. These experiments test to see if having UCM embedded into a documentation style will actually help the learner to perform and understand better. All of the documentation styles were converted and presented online. Thus, all of the subjects participated and completed the work task by following one of the sets of online documentation. There are four sets online documentations: Patterns, Patterns with UCM, Minimalist and Minimalist with UCM (Table 1).

There are two types of variables-independent (factors) and dependent. In statistical analysis, it is a common practice (Ho, 2008; Ho *et al*., 2007) to denote independent variables as x values such as x_1, x_2, x_3 and so forth until x_n and dependent variables as Y values from Y_1, Y_2, Y_3 until Y_n. Using formula expansion, we can show the explanation above as follows:

$$(Y_j) = c_0 + c_1x_1 + c_2x_2 + c_3x_3 + .. + c_nx_n$$

where, c_i represent any constant to fulfill the equality between Y_j and x_i. Table 2 shows the independent and dependent variables derived from the experiments.

The listed variables above form the questions in the tutorial exercise. Dependent variables which have no significant values in homogeneity (Ho *et al*., 2009) during the statistical tests will be omitted.

Table 3 shows examples from the experiments with all the mapping between dependent variables and their exercise questions that serve as a base to establish test items for the framework.

Patterns documentation style includes the background of the problem or issue as well as the solution

Table 1: Describes the variables used in the experiments

Type	Name	Value	Description
Independent variables	DOCTYPE	{PAT, PAT_UCM, MIN, MIN_UCM}	"Documentation type": all four sets of documentation have the same objective, to complete the given work task but are based on different documentation styles
Dependent variables	ENDFINAL	Time (Hr:Min:Sec)	"Completion time": the time taken to complete the entire experiment. At the end of the work tasks, the subject was to record this completion time
	END2NDQ	Time (Hr:Min:Sec)	"2nd-quarter completion time": the time taken for the subject to complete the third compilation
	ENDSEMI	Time (Hr:Min:Sec)	"Semi completion time": the time taken for the subject to complete the second compilation
	END1STQ	Time (Hr:Min:Sec)	"Quarter completion time": the time taken for the subject to complete the first compilation
	COMPR	Ordinal	"Comprehension": the subject has to identify the method, procedure, line of code and constants that are required to perform the given task. There are a number of questions to test their understanding of the code
	WORK	Ordinal	"Workings": this is to test how well the subject can follow the instructions for creating the default methods or constructors with their modifiers and arguments
	DIFF	Integer	"Number of difficulties": total accumulated problems subject encountered. Instead of giving all of the detailed the steps, some parts of the documentation allow the learner to interact with the system

Table 2: Independent and dependent variables

Variable type	Variables
Independent (x_i)	x_1: Documentation type, X_2: Gender, Y_1: Completion time, Y_2: 2nd quarter completion time,
Dependent (Y_j)	Completion time, Y_3: Semi Y_4: 1st quarter completion time, Y_5: Number of difficulties, Y_6: Comprehension, Y_7: Workings

Table 3: Mapping between dependent variables and their exercise questions

Dependent variable	Examples from experiment
Comprehension	Q1: Which package(s) will be called does <the event>?
	Q2: Which abstract class to be extended when <the event>?
	Q3: Which interfaces to be implement when <the event>?
	Q4: Which line of code will be executed for <the event>?
	Q5: Which method in the program that does <the event>?
Workings	Q1: Write the necessary Java ME code when importing classes
	Q2: Write the necessary Java ME code when creating main class
	Q3: Write the necessary Java ME code when defining variables
	Q4: Write the necessary Java ME code when creating classes and procedures

<the event> refers to functionality achieved from code execution

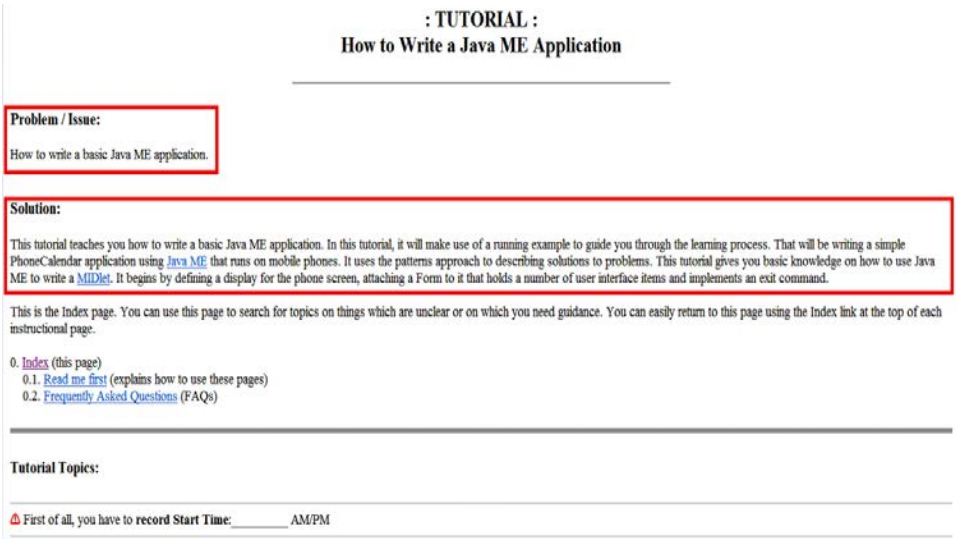


Fig. 1: Patterns

needed to solve the problem or issue as highlighted in Fig. 1. Patterns documentation with UCM is the same as Patterns but with the addition of UCM (Fig. 2). UCM consists of a path that shows the causal relationships between responsibilities of one or more use cases in a map-like diagram.

Minimalist documentation contains as little information as possible for the reader to read in line with the idea of having people read the minimal amount of information to get the task done. Minimalist documentation does not state the background of a problem and the solution needed to solve the problem, unlike patterns. This is shown in the screenshot in Fig. 3.

Minimalist documentation with UCM is the same as the minimalist but with the addition of UCM (as shown in the second screenshot in Fig. 4.

Apart from setting up the sets of documentation in the various styles, a survey collected data from subjects. According to Trochim (2006), surveys can be divided into two main categories which are the questionnaire and the interview. The questionnaire method usually involves paper-and-pencil instruments by having a set of questionnaires prepared for the subjects to complete. While interviews are conducted by the interviewer recording what the subjects say. The questionnaire method can be divided into three types-mail survey questionnaire, group-administered survey questionnaire and household drop-off survey questionnaire. Interviews are divided into two types personal interview and telephone interview.

For this research, the group-administered survey questionnaire method is adopted and performed. By having this group administered method, the subjects involved in the survey are brought together to respond to a structured sequence of questions in a more controlled environment (i.e., classes) and are also more likely to have the same background or experience. The rest of the survey types are not suitable for this type of pedagogical research. The survey questionnaire form used to collect data from the subjects for this research is shown in Fig. 2.

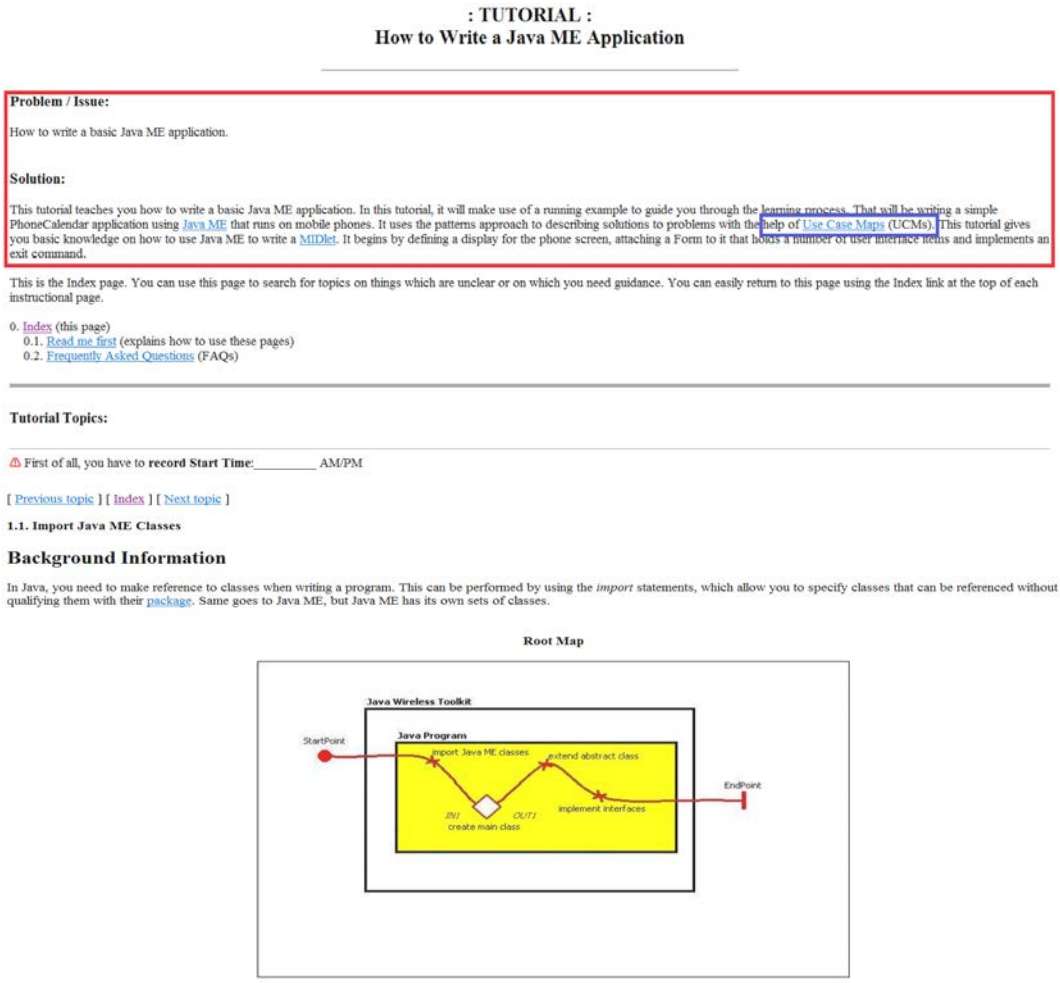
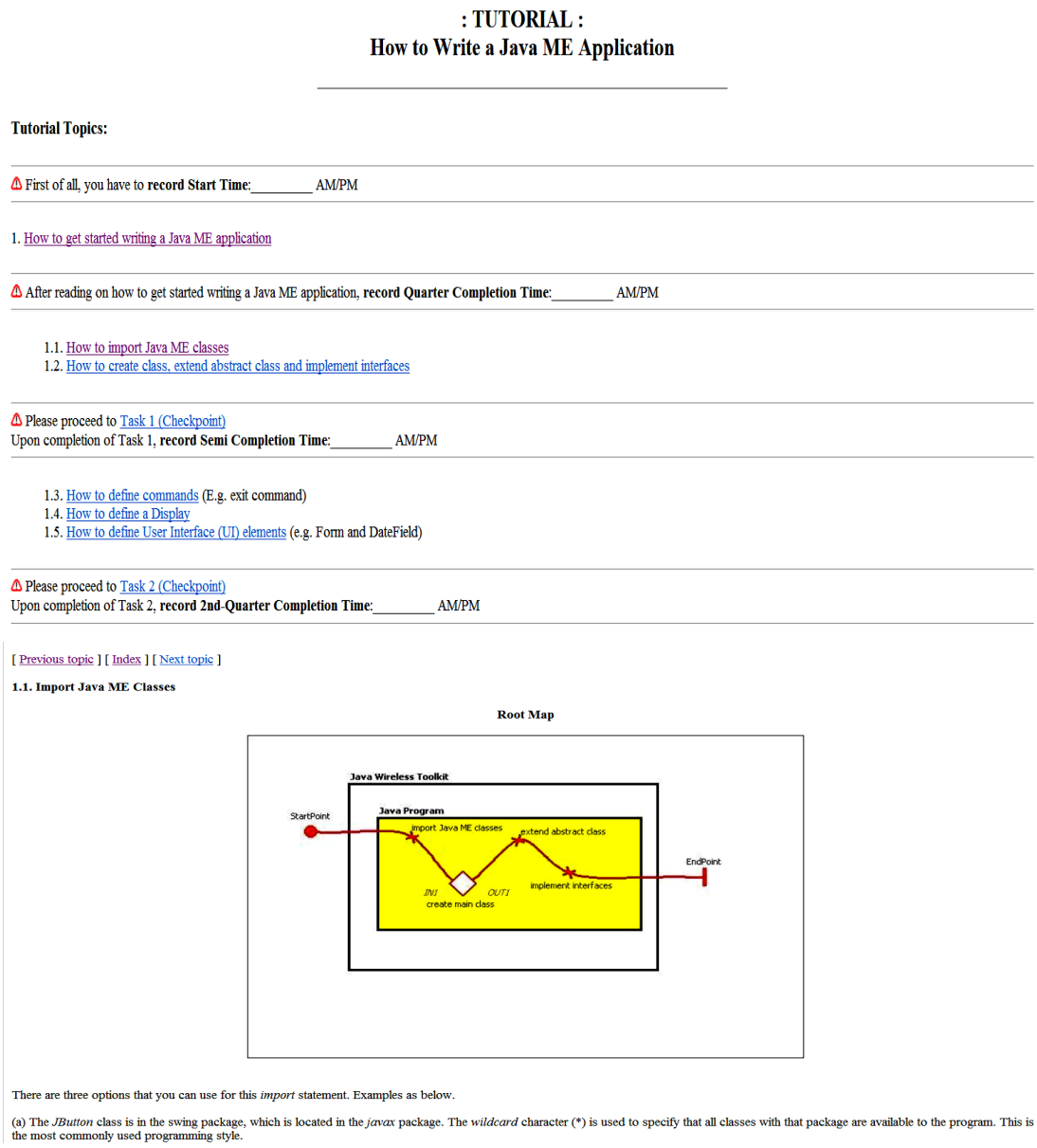


Fig. 2: Patterns with UCM

Fig. 3: Minimalist



: TUTORIAL :
How to Write a Java ME Application

Tutorial Topics:

⚠ First of all, you have to **record Start Time**:_____ AM/PM

1. How to get started writing a Java ME application

⚠ After reading on how to get started writing a Java ME application, **record Quarter Completion Time**:_____ AM/PM

1.1. How to import Java ME classes
1.2. How to create class, extend abstract class and implement interfaces

⚠ Please proceed to Task 1 (Checkpoint)
Upon completion of Task 1, **record Semi Completion Time**:_____ AM/PM

1.3. How to define commands (E.g. exit command)
1.4. How to define a Display
1.5. How to define User Interface (UI) elements (e.g. Form and DateField)

⚠ Please proceed to Task 2 (Checkpoint)
Upon completion of Task 2, **record 2nd-Quarter Completion Time**:_____ AM/PM

[Previous topic] [Index] [Next topic]

1.1. Import Java ME Classes

Root Map

There are three options that you can use for this *import* statement. Examples as below.

(a) The *JButton* class is in the swing package, which is located in the *javax* package. The *wildcard* character (*) is used to specify that all classes with that package are available to the program. This is the most commonly used programming style.

Fig. 5: Minimalist with UCM

RESULTS AND DISCUSSION

The data analysis will be carried out to identify which documentation style allows the subjects to complete parts of the documentation (END1STQ, ENDSEMI, END2NDQ) and complete the overall documentation the fastest (ENDFINAL) with the number of difficulties recorded by the subjects at the intervals (DIFF). The documentation style which allows the subjects to understand the most are also analyzed and recorded (COMPR) as well as their knowledge in the inner workings of Java ME (WORK).

The meanings of the variables are:

- END1STQ-end of 1st quarter
- ENDSEMI-end of semi completion
- END2NDQ: end of 2nd quarter
- ENDFINAL: end of completion
- DIFF: number of difficulties faced
- COMPR: comprehension
- WORK: workings

In order to identify whether the seven dependent variables above are normally distributed or not, the

normality of the variables was tested using the One-Sample Kolmogorov-Smirnov test (K-S test). The normality test show that four (out of seven) dependent variables are normally distributed as shown in Table 4.

One-sample Kolmogorov-Smirnov test (K-S test): K-S test is used to determine the normality of the dependent variables (D'Agostino *et al*., 1990). The calculation result is focused on the significance of the p-value. In particular, p-values that are <0.05 are considered " statistically significant" and interpreted as being small enough to justify rejection of the null hypothesis (Lane, 2015). Based on the K-S test being carried out for all of the dependent variables, we use the 5% level (i.e., p<0.05) as the indicators to determine whether it is normally distributed or non-normally distributed. If the p<0.05, it is statistically significant and thus, it is non-normally distributed. For non-normally distributed variables, we use the Kruskal-Wallis test (Bluman, 2004). If the p>0.05, it is not statistically significant which means that the data is normally distributed. This means that we can use Multivariate Analysis of Variance (MANOVA) for the analysis. The Kruskal-Wallis and MANOVA tests will be discussed in detail later in their respective sections. Table 5 shows the results of the normality test. The median is used as the expected values due to the non-parametric outcome of the dependent variables of END1STQ, COMPR and DIFF.

Table 4: Dependent variables vs. normally distributed results

Category (dependent variables)	Normally distributed
END1STQ	No
ENDSEMI	Yes
END2NDQ	Yes
ENDFINAL	Yes
COMPR.	No
WORK.	Yes
DIFF.	No

Means and standard deviations: Secondly, we present the means and standard deviations of the seven dependent variables for each of the four documentation styles. The purpose of this analysis is to observe which documentation style performs the best. The fastest END1STQ, ENDSEMI, END2NDQ and ENDFINAL are the lowest achievement time whereas the best COMPR and WORK are the highest scores. The lowest DIFF indicates the least number of difficulties the participants face in learning Java ME.

Table 6 shows that Minimalist with UCM has the best ENDFINAL. The fastest ENDFINAL in Minimalist with UCM can be due to the minimal reading materials within the documentation style. Apart from this, by having UCM (Amyot and Mussbacher, 2001) integrated with the traditional Minimalist style, the documentation is enhanced, resulting in a faster completion time (ENDFINAL) than traditional Minimalist documentation.

However, Minimalist documentation has the best WORK score, perhaps due to the fact that it has the least amount of material to read among all of the documentations; thus, it is slightly easier to follow. On the other hand, patterns documentation has the highest achievement for the COMPR score. However, by adding UCM, COMPR increases when we compare Minimalist to Minimalist with UCM. Patterns with UCM is the worst in terms of performance and understanding of the learners.

Table 5: Results for one-sample Kolmogorov-Smirnov test

Category (dependent variables)	Kolmogorov-Smirnov: Z-value	N	Asymp. Sig. (2-tailed)	Statistically significant (p<0.05)
END1STQ	1.646	120	0.009	Yes
ENDSEMI	1.080	120	0.194	No
END2NDQ	1.283	120	0.074	No
ENDFINAL	0.877	120	0.425	No
COMPR	1.767	120	0.004	Yes
WORK	1.207	120	0.109	No
DIFF	2.979	120	0.000	Yes

Table 6: Means and standard deviations of all categories

	Mean (standard deviation)			
Category (dependent variable)	Pattern (μPat)	Pattern with UCM (μPat_U)	Minimalist (μMin)	Minimalist with UCM (μMin_U)
END1STQ (hh.mm.ss)	0:10:38 (0:05:05)	0:14:57 (0:08:30)	0:15:36 (0:14:37)	0:09:20 (0:05:49)
ENDSEMI (hh.mm.ss)	0:20:22 (0:08:29)	0:28:24 (0:14:46)	0:20:03 (0:14:10)	0:17:36 (0:07:31)
END2NDQ (hh.mm.ss)	0:27:49 (0:09:44)	0:36:07 (0:16:47)	0:25:05 (0:14:30)	0:25:07 (0:11:56)
ENDFINAL (hh.mm.ss)	0:49:04 (0:16:13)	0:59:45 (0:20:21)	0:47:56 (0:18:11)	0:43:20 (0:14:56)
COMPR Scale: 0-5 point (1 point for each correct answer)	11.43 (2.445)	10.30 (3.250)	10.97 (2.414)	11.03 (2.671)
WORK Scale: 0-13 point (1 point for each correct answer)	18.57 (5.482)	16.67 (8.096)	22.77 (6.765)	21.47 (8.199)
DIFF	0.77 (1.073)	3.07 (4.425)	1.50 (1.815)	1.13 (1.383)

Table 7: MANOVA results for the normally distributed variable

Category	F-values	Significance
ENDSEMI	4.822	0.003*
END2NDQ	4.477	0.005*
ENDFINAL	4.692	0.004*
WORK	4.392	0.006*

*Statistically significant at 0.05 level

Table 8: Mean ranks of END1STQ, COMPR and DIFF

Documentation Style	Pattern	Pattern with UCM	Minimalist	Minimalist with UCM
Sample size (n)	30.00	30.00	30.00	30.00
Mean rank of END1STQ	57.92	73.65	63.02	47.42
Mean rank of COMPR	66.78	55.12	59.18	60.92
Mean rank of DIFF	50.63	71.18	62.72	57.47

Table 9: Kruskal-Wallis test on END1STQ, COMPR and DIFF

Category	Chi-square	df	Asymptotic significance
END1STQ	8.855	3	0.031*
COMPR	1.780	3	0.619
DIFF	6.320	3	0.097

*Statistically significant at 0.05 level

Multivariate Analysis of Variance (MANOVA): The dependent variables identified as normally distributed during the K-S test are further analyzed using the MANOVA test. This is to identify whether the differences between the four documentation styles are significant or not.

Table 7 shows that all of the categories, i.e., ENDSEMI, END2NDQ, ENDFINAL and WORK are significant at the 0.05 level. This means that Minimalist with UCM can be beneficial in terms of performance when guiding the learners to learn to develop the Java ME phone calendar application.

Kruskal-Wallis test: Lastly, for the remaining three dependent variables, i.e., END1STQ, COMPR and DIFF, we carried out the Kruskal-Wallis test (Bluman, 2004). This is because of their non-parametric results from the previous K-S test. The mean ranks shown in Table 8 are consistent with Table 6 with END1STQ being the best for Minimalist with UCM while COMPR and DIFF are the best for Patterns.

Table 9 shows that END1STQ is statistically significant at the 0.05 level while COMPR and DIFF have no significant differences. The non-significance of COMPR and DIFF may due to not much of comprehension or understanding being required to write a simple phone calendar application in Java ME. However, the significant difference in END1STQ shows that Minimalist with UCM style is effective in guiding learners to follow through the documentation section of the phone calendar application. This supports the hypothesis that using UCM increases performance (i.e., efficiency).

CONCLUSION

This empirical study evaluated the effectiveness of Use Case Maps (UCM) in helping mass learning when used with two different framework documentation styles, i.e., Patterns and Minimalist. Hence, four Pedagogical Framework Documentations (PFD) have been tested, Patterns vs. Patterns with UCM and Minimalist vs. Minimalist with UCM.

This study focuses on teaching readers how to use frameworks with the help of Use Case Maps (UCM) within pedagogical documentation. The motivation of the research is to close the gap between the requirements gathering and design stage by introducing UCM. However, UCM is not necessary applicable to any documentation style. Thus, this study helps to identify which documentation style suits UCM better-Patterns or Minimalist.

The analysis results show that adding UCM to pedagogical documentation increases knowledge (understanding) and performance of the learners in developing Java Micro Edition (JME) applications. However, the amount to read in pedagogical documentation affects or impacts the effectiveness of UCM. This is due to the fact that more reading (i.e., Pattern with UCM) causes more confusion to the learners. This will indirectly cause the understanding of the learners to decrease and therefore, impacts their performance in completing the framework. Based on the above, we can conclude that UCM works better in Minimalist documentation (less to read) than Patterns (more to read).

This study solves several questions but at the same time raised a few more. First, there is a need to run these experiments using other programming language than Java ME, to see if the type of programming language used in the framework documentation has an impact on the overall effectiveness and efficiency of the documentation. Second, how did the subjects know how to answer the comprehension questions correctly if they skipped the section in the documentation style describing the internal workings? Third, if we apply this programming framework documentation to other different frameworks such as project management frameworks etc., will the observations still be valid? Another possible future work is to investigate other alternatives than UCM-how would other types of diagrams such as entity relationship diagrams (Jain *et al*., 2004; Schach, 2005) or flowcharts affect the outcome of the experiments?

REFERENCES

Aguiar, A. and G. David, 2000. A minimalist approach to framework documentation. Proceedings of the ACM Conference on Object-Oriented Programming, Systems, Languages and Applications, October 15-19, 2000, Minneapolis, MN., USA., pp: 143-144.

Amyot, D. and G. Mussbacher, 2001. Introduction to use case maps. International Telecommunication Union (ITU), Geneva, Switzerland.

Bluman, A.G., 2004. Elementary Statistics: A Step by Step Approach. 5th Edn., McGraw-Hill, New York, USA., ISBN-13: 9780072549072, pp: 640-648.

Carroll, J.M., 1998. Minimalism Beyond the Nurnberg Funnel. MIT Press, Cambridge, MA., USA., ISBN-13: 9780262032490, Pages: 416.

D'Agostino, R.B., A. Belanger and R.B. D'Agostino Jr., 1990. A suggestion for using powerful and informative tests of normality. Am. Statistician, 44: 316-321.

Dix, A., J. Finlay, G.D. Abowd and R. Beale, 2004. Human-Computer Interaction. 3rd Edn., Pearson Prentice Hall, Essex, UK., ISBN-13: 978-0130461094, pp: 532-533.

Fayad, M. and D.C. Schmidt, 1997. Object-oriented application frameworks. Commun. ACM, 40: 32-38.

Ho, S.B., 2008. Framework documentation with patterns: An empirical study. Ph.D. Thesis, Multimedia University, Cyberjaya, Selangor, Malaysia.

Ho, S.B., I. Chai and C.H. Tan, 2007. An empirical investigation of methods, for teaching design patterns within, object-oriented frameworks. Int. J. Inform. Technol. Decis. Making, 6: 701-722.

Ho, S.B., I. Chai and C.H. Tan, 2009. Comparison of different documentation styles for frameworks of object-oriented code. Behav. Inform. Technol., 28: 201-210.

Jain, S.K., M.M. Gore and G. Singh, 2004. An extension to ER model for top-down semantic modeling of databases of applications. Proceedings of the 7th International Conference on Intelligent Information Technology, December 20-23, 2004, Hyderabad, India, pp: 253-262.

Lane, D.M., 2015. Logic of hypothesis testing. http://onlinestatbook.com/2/logic_of_hypothesis_testing/logic_hypothesis.pdf.

Meszaros, G. and J. Doble, 1996. Metapatterns: A pattern language for pattern writing. Proceedings of the 3rd Pattern Languages of Programming Conference, September 4-6, 1996, Monticello, IL., USA.

Schach, S.R., 2005. Object-Oriented and Classical Software Engineering. 6th Edn., McGraw Hill, New York, USA., ISBN-13: 9780072865516, pp: 333-334, 369-374.

Schumacher, M., 2003. Security Engineering with Patterns: Origins, Theoretical Model and new Applications. Springer-Verlag, New York, USA., ISBN-13: 9783540407317, Pages: 208.

Trochim, W.M.K., 2006. Type of surveys. Research Methods Knowledge Base. http://www.socialresearchmethods.net/kb/survtype.php.

Zamir, S., 1998. Handbook of Object Technology. CRC Press, USA., ISBN-13: 9781420049114, Pages: 1168.