Computing and Informatics, Vol. 40, 2021, 1025–1055, doi: 10.31577/cai_2021_5_1025

CONTROLLED EXPERIMENT FOR ASSESSING THE CONTRIBUTION OF ONTOLOGY BASED SOFTWARE REDOCUMENTATION APPROACH TO SUPPORT PROGRAM UNDERSTANDING

Sugumaran Nallusamy, Meei Hao Hoo

University Tunku Abdul Rahman
Sungai Long Campus Jalan Sungai Long
Cheras 43000, Kajang
Selangor, Malaysia
e-mail: {sugumaran, hoomh}@utar.edu.my

Farizuwana Akma Zulkifle

Universiti Teknologi MARA
Kuala Pilah Campus
72000 Kuala Pilah
Negeri Sembilan, Malaysia
e-mail: farizuwana@uitm.edu.my

Abstract. Redocumentation is an approach that is used to recover knowledge from raw software artifacts by using alternative presentations. Several legacy systems have been developed based on event-driven programming which require redocumentation. However, these existing repository and query techniques emphasize only on lexical and syntactical based queries which come with limitations in providing the semantic relationship for program understanding. We are using ontology based approach that uses both ontology reasoning and querying techniques to generate software documentation from the knowledge repository. We present a controlled experiment for the empirical evaluation on the proposed ontology based approach and implemented in a tool called Ontology Based Software Redocumentation (OBSR). In this experiment, two existing tools namely Universal Report (UR) and Microsoft Visual Studio specifically for Visual Basic (VB) programming environment have been selected to be compared with the OBSR tool. The goal is to provide experimental evidence of the viability of our approach in the context of program understanding using HTML based semantic software documentation. The experiment shows that the software maintainers are able to understand and provide significant improvement in program understanding to accomplish the maintenance task easily. We describe in detail the experiment performed, discuss its results and reflect the lesson learned from the experiment.

Keywords: Software documentation, ontology, reverse engineering, software maintenance, event-driven programming, program understanding

1 INTRODUCTION

Software redocumentation is one of the approaches used as an aid for program understanding to support the maintenance and evolution. According to Chikofsky and Cross, "Redocumentation is a creation or revision of a semantically equivalent representation within the same relative abstraction level" [1]. In other words, redocumenting a code is transformation of one existing (and other documents and stakeholder knowledge) into a new or updated documented code. It therefore becomes an aid for the recovery and recording in software comprehension.

Even though legacy systems are sometimes viewed as outdated software system, other reviews claim that legacy systems are worth keeping and supporting since they are essential to a business. Previous studies [2, 3, 4, 5] have highlighted the challenges in modernization of legacy systems in terms of technical and business aspects. Thus, software redocumentation is one of the important methods to assist the modernization of legacy system. Redocumentation consists of four main components, which are Software Work Product (SWP), or artifact, parser, repository and documentation. These four components create a process to rescue knowledge from a software system. Redocumentation helps to extract knowledge from the SWP via reverse engineering techniques, and presents it in the form of documentation. A repository component in redocumentation is an important component to process data and generate documentation [6]. It provides the query functionality to build the content in the documentation, as well as, to enable browsing and searching for relevant content in the hypertext documentation [7]. These repositories require query capabilities to find and build various views or documents requested by software maintainers [6, 8]. The existing query techniques used in the current redocumentation approaches and tools to query the source code are lexical, relational, graph, algebra and network structure queries.

To evaluate the advantages and disadvantages of the tools in acquiring knowledge of any such technique we required to conduct empirical evaluation. There are few typical empirical evaluation used namely, case studies, observations, surveys, or controlled experiment [9]. However, according to Sjoeberg et al. [10], a controlled experiment is suitable to identify the capability of tools or approaches in acquiring knowledge for specific people in a specific environment. In addition, controlled experiments used in a variety of current approaches to software documentation [11, 12, 13, 14].

In the previous study, we have formulated an approach called Ontology Based Software Redocumentation (OBSR) [15, 16]. The OBSR approach generates software documentation from the Source Code Ontology (SCO) which allows the software maintainers to semantically search and browse for the transitive relations and concept hierarchy within the source code. The SCO describes and presents the knowledge from the programming source code semantically in the repository. The documentation generated from the SCO is able to enhance the search, navigation and inference capabilities to assist in finding the relevant data from a source code. Thus, a meaningful information can be provided to understand the source code to solve software maintenance tasks. Furthermore, a tool called Ontology Based Software Redocumentation Tool (OBSRT) was developed to proof the concept of the OBSR approach. A detailed overview of the established OBSRT tool will be discussed in the following section.

However, the main goal of this paper is to evaluate the OBSR approach using the developed OBSR tool for its support in software maintenance. The OBSR tool was developed to redocument the source code from the VB 6.0. Source code from VB 6.0 was chosen because the support for VB has been discontinued by Microsoft in 2008 [17, 18], and many programmers switched to other programming languages, such as PHP, and C#. The changes to the VB 6.0 framework caused issues and difficulty in understanding the legacy system for purpose of upgrading or rewriting [19]. The OBSR tool solves this problem by automatically generating HTML software documentation and representing the mental model for the software maintainers to understand the program residing in the software systems semantically.

Therefore, we conducted a controlled experiment to verify whether the OBSR approach is able to improve the efficiency and effectiveness to program understanding tasks during software maintenance. The controlled experiment was initiated by formulating the null hypothesis from the research question of this study. UR and VB are used as baselines for the proposed OBSR tool. These two baselines are also referred to as independent variables in this controlled experiment. The tasks designed consist of two sections, namely, subjects' work backgrounds and questions on program understanding to handle the software maintenance tasks during the experiment. The subjects that participated in the controlled experiment consist of 33 participants from various positions; nine participants were programmers, five participants were systems analysts and three participants were software consultants.

The rest of the paper is structured as follows. Section 2 outlines the author's formulated OBSR method. Section 3 explains all tasks carried out in a controlled experiment. Section 4 addresses the outcome. Section 5 outlines the findings of the controlled experiment. Section 6 explains how the risks to authenticity have been minimised. Finally, Section 7 ends the remarks and the potential work.

2 ONTOLOGY BASED SOFTWARE REDOCUMENTATION (OBSR) APPROACH

The main problem in the existing redocumentation approaches and tools is the knowledge repository and querying functionalities emphasize on lexical, syntactical and graph based query techniques, which are not able to use reasoning capabilities to identify, justify and verify the relation in the source code and provide relevant information as needed by software maintainers in the specific time. This affects the efficiency and effectiveness of the exploration functionalities in software documentation such as browsing, searching and visualizing the source code for program understanding. Therefore, OBSR approach was proposed as shown in Figure 1, and it consists of three levels of input, namely, Knowledge Extraction (Level 1), Knowledge Transformation (Level 2) and Knowledge Representation (Level 3). The approach was formulated based on the importance of the navigation functionalities, which required two main components, namely, repository and reasoning tools. Therefore, the proposed OBSR approach provides an ontology based repository and reasoning capabilities in Level 2 (Knowledge Transformation) by extracting the source code component from Level 1 (Knowledge Extraction).

The approach was formulated based on the importance of the navigation functionalities, which required two main components, namely, repository and reasoning tools. Therefore, the proposed OBSR approach provides an ontology based repository and reasoning capabilities in Level 2 (Knowledge Transformation) by extracting the source code component from Level 1 (Knowledge Extraction). The tool called OBSRT was developed based on the formulated OBSR approach. The source code is transformed to ontology and the user is able to use OBSR tool to generate the documentation automatically. In addition, to visualize the search result in graphical form, OBSR tool provides the component viewer to have better understanding on the source code.

3 EXPERIMENT DESIGN

The proposed OBSR approach was evaluated using a controlled experiment to quantitatively evaluate the effectiveness and efficiency of the OBSR approach based on the correctness and time. The efficiency has been measured based on how quickly and timely the software maintainers understand the software system during the software maintenance tasks. On the other hand, the effectiveness of the OBSR approach has been measured based on the correct answer provided for each question considered. The controlled experiment has been designed and conducted based on factors such as choosing baseline, choosing subjects, providing training for subjects, selecting tasks and distributing tasks, choosing real world systems and preparing guidelines to repeat the experiment, as specified by Wettel et al. [20]. Detail study and analysis have also been done regarding procedure and steps required to be followed before and during the experiment [21, 22].

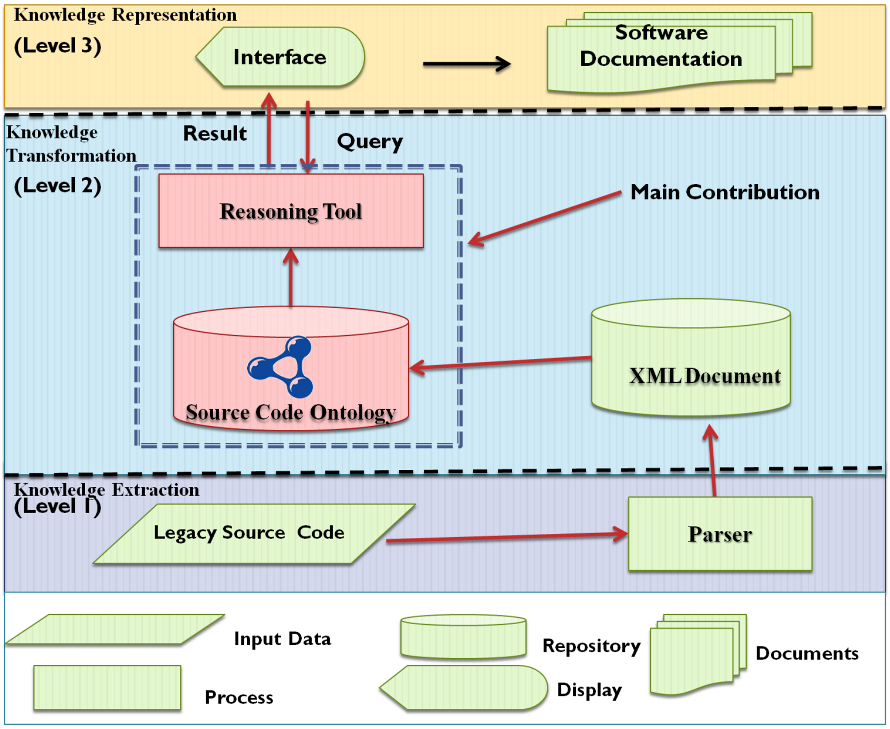


Figure 1. Proposed OBSR approach

3.1 Research Question and Hypothesis

The main research question for this controlled experiment is defined as *"How to validate the significance of the proposed OBSR approach in improving program understanding during the software maintenance tasks?"* Thus, two different sets of hypotheses have been formulated to measure efficiency and effectiveness individually.

a) Hypothesis to Measure Efficiency

A null hypothesis formulated to measure efficiency based on the main research question is as follows:

> $H1_0$: There is no significant difference in time required to complete program understanding task among the UR, VB, and OBSR tools during software maintenance.

An alternative hypothesis used to measure efficiency in this controlled experiment is as follows:

> H1: The ontology based software documentation approach significantly improves the time required to complete the program understanding task during software maintenance.

b) Hypothesis to Measure Effectiveness

A null hypothesis formulated to measure effectiveness based on the main research question is as follows:

> $H2_0$: There is no significant difference in correctness of the solutions to program understanding tasks between the UR, VB, and OBSR tools during software maintenance.

An alternative hypothesis used to measure effectiveness in this controlled experiment is as follows:

> H2: The ontology based software documentation approach significantly improves the correctness of the solutions to program understanding tasks during software maintenance.

After defining the hypothesis of this controlled experiment, two existing redocumentation tools have been selected as a baseline for this study.

3.2 Choose Baseline for Controlled Experiment

UR [23] and VB [24] are the two tools that have been used as baselines for the proposed OBSR tool. These two baselines are also referred to as independent variables in this controlled experiment. The UR and VB tools have been selected as the baseline for this controlled experiment for two main reasons, namely, they support VB programming and they support search and exploration functionality, as specified below:

Support VB Programming Language. As specified earlier in Section 2, the OBSR tool generates software documentation for VB source code. Therefore, the baseline tool was selected from existing tools which can support VB source code. In our controlled experiment, we used UR and VB. UR supports the redocumentation process for VB source code and provides many functionalities, such as source code metrics, object level interaction and method level interaction. On the other hand, the VB editing environment is one of the common graphical user interface application programming languages in the year 2000. Many large scale applications have been developed using VB, and are currently maintained as legacy systems. In general, the UR tool has better functionality compared to other existing VB redocumentation tools, such as VBDocman, which only provides a list of objects and methods, and related source code.

Presence of Search and Exploration Functionalities. Both UR and VB tools have the ability to present technical documentation with exploring and query functionality. The UR tool provides these functionalities in the form of HTML based documentation. On the other hand, even though VB 6.0 is programming environment but VB provides basic functionalities required to understand the program to handle the software maintenance tasks, such as searching and browsing functions which software maintainer use this functionalities in VB to support the program understanding and maintenance task. Therefore, in this experiment, VB becomes a part of tool to compare the searching and browsing functionalities with UR and OBSR tool.

3.3 Select Dependent Variables

The dependent variables were chosen based on the goal of this experiment, which is to find the efficiency and effectiveness of the proposed OBSR tool. The efficiency was measured based on the time taken to answer each question given to the subjects. On the other hand, the effectiveness was measured based on the score of correct answers provided by the subjects for the questions asked. Therefore, score (S) and time (T) have both been chosen as the dependent variables for this study. These dependent variables, time (T) and score (S), have been captured from the answers provided by the subjects during the experiment. The time taken consists of two dependent variables, namely, T_1 for time taken regardless of correct answers, and T_2 for time taken with the consideration of the correct answers. Next, statistical analysis was conducted to test the significance of the OBSR tools used by the subjects in this controlled experiment.

3.4 Statistical Analysis for Dependent Variables

The statistical analysis starts with the Box's M-test to verify the assumption on the homogeneity of equal covariance matrices across the cells formed by the between-subjects effects. T_2 is not required to be tested with the Box's M test because the data size is not equal and varies, as specified by Johnson and Wichern [25]. However, Multivariate Analysis of Variance (MANOVA) and Analysis of Variance (ANOVA) tests still need to be conducted when the Box's M-test result leads to rejection of the null hypothesis.

After the Box's M-test, MANOVA has been used to test the significant mean difference in time taken (T_1) and score (S) for all six questions. Meanwhile, ANOVA has been used to test the significant differences between means based on time taken (T_1 and T_0) and score by questions. After ANOVA, a detailed Post-Hoc test has been utilized to find the significant mean pair from the ANOVA results. The Post-Hoc test was conducted to find the significant pair difference and accept the alternative hypothesis if the OBSR tool shows significant improvement from any of the six program understanding questions compared to the UR and VB tools.

There are two types of Post-Hoc tests used, namely, Tukey High Significance Difference (HSD) and Hocbergs's GT2. Tukey HSD was used to test the results from T_1 and S because the sample used were equal in size, which were 11 subjects for each tool. Hocbergs's GT2 was used to test T_2 because the sample size varied based on each question.

The statistical analysis analyses the significance value from the MANOVA test for variable S and T_1 based on the probability value; $\alpha = 0.1$ can be used if the sample size is small; this applies in this controlled experiment because the sample size is small, and consists of only 33 subjects. If the significant value (p-value) is less than $\alpha = 0.1$, the null hypothesis for $H1_0$ and $H2_0$ can be rejected. On the other hand, the alternative hypotheses are accepted if the OR group shows a significant improvement from any of the six program understanding questions compared to the UR and VB tools. The steps on the statistical analysis for each dependent variable are shown in Figure 2. These steps have been used as a guideline during the statistical analysis in this controlled experiment.

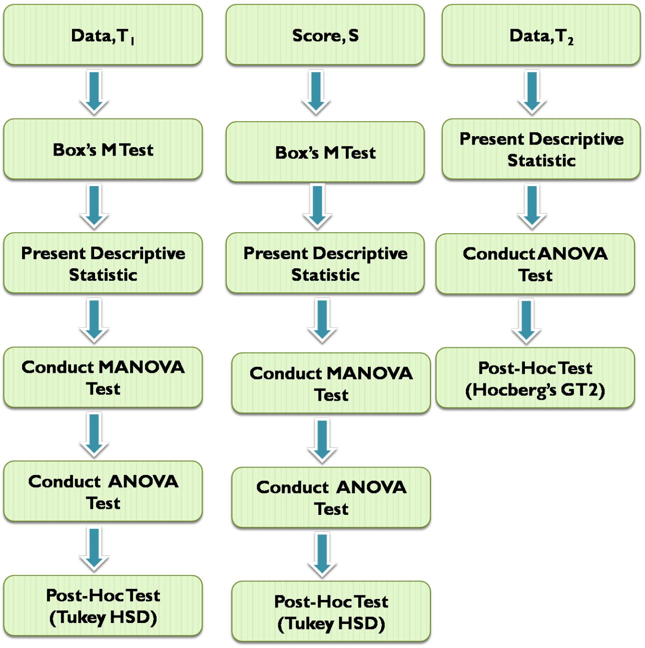

Data, T1
Score, S
Data, T2
Box's M Test
Box's M Test
Present Descriptive Statistic
Present Descriptive Statistic
Present Descriptive Statistic
Conduct ANOVA Test
Conduct MANOVA Test
Conduct MANOVA Test
Post-Hoc Test (Hocberg's GT2)
Conduct ANOVA Test
Conduct ANOVA Test
Post-Hoc Test (Tukey HSD)
Post-Hoc Test (Tukey HSD)

Figure 2. Statistical analysis steps for the dependent variables

3.5 Subject Systems for Controlled Experiment

The subject system used in the controlled experiment is a College Management System (CMS) that was developed 11 years ago (in 2003) for a private college in Malaysia, termed as College A (name obfuscated for non-disclosure reasons). The CMS was developed using VB 6.0, and consisted of about 100 000 lines of code.

The CMS basically handles the management process in the college environment, including student admissions, personnel data management, course registration, examination scheduling, student attendance, examination reports, facility management and maintenance scheduling. The CMS is a medium sized system which consists of 179 forms, 7 modules and uses 102 tables in a database. This legacy source code was used as a subject system in the controlled experiment to evaluate the proposed approach.

3.6 Subject for Controlled Experiment

In the controlled experiment, 33 participants were involved in several sessions of the experiment. As stated earlier, the group of experiments was divided into three groups based on the used used by each group: the OR group (Ontology Based Software Redocumentation tool), the UR group (Universal Report tool), and the VB group (VB tool). Each group consists of 11 subjects. Table 1 shows the frequency distribution of the different types of positions in the software development industry and PhD academic work. The mixed group of students and industry employees chosen for this control experiment also match the arguments of adequately represent the intended user population for the study [26]. The students involved in this experiment have some working experience before continuing their postgraduate studies. The subjects have participated in this experiment on a voluntary basis, and it is assumed that the subjects have an honest engagement towards this experiment.

Position	**Frequency**	**Percent**	**Valid Percent**	**Percent**
Ph.D. Students	9	27.30 %	27.30 %	27.30 %
Lecturers – Software Consultants/Developers	7	21.20 %	21.20 %	48.50 %
Software Programmers	9	27.30 %	27.30 %	75.8 %
Software Consultants	3	9.00 %	9.00 %	84.8 %
System Analysts	5	15.20 %	15.20 %	100 %
Total	**33**	**100 %**	**100 %**	

Table 1. Position of subjects

Each participant is required to have at least basic knowledge with the VB programming language. This is considered a challenging task in this controlled experiment, because it is difficult to find people with knowledge in VB, as this software has been obsolete since 2008. Tables 2 and 3 show cross tabulation based on the position and experience in developing and maintaining the program using VB for each group respectively. In Table 4, the groups of the subjects were distributed fairly based on their programming skills with VB. The VB programming skill levels was determined based on their development expertise, and was correlated to the five stage typology defined by Artherton [27]. The number of subjects involved in each skill level is the same except, for Proficient and Expert, which are four and three, respectively. This is due to the difficulties to obtain the skilled person in VB programming.

Position \ Tool	OR	UR	VB	Total
PhD Students	3	4	2	9
Lecturers – Software Consultants/Developers	2	3	2	7
Software Programmers	3	2	4	9
Software Consultants	1	1	1	3
System Analysts	2	1	2	5
Total	**11**	**11**	**11**	**33**

Table 2. Cross tabulation of position of subjects versus tool

Experience \ Tool	OR	UR	VB	Total
Less 1 year	1	0	1	2
1–3 years	2	3	2	7
4–6 years	2	2	0	4
More than 6 years	6	6	8	20
Total	**11**	**11**	**11**	**33**

Table 3. Cross tabulation of experience versus tool

Experience \ Tool	OR	UR	VB	Total
Novice	3	4	2	9
Advance Beginner	3	3	2	9
Competent	3	2	4	9
Proficients	1	1	2	4
Expert	1	1	1	3
Total	**11**	**11**	**11**	**33**

Table 4. Cross tabulation of subjects based on VB programming skill

3.7 Task Design for Controlled Experiment

In this research, the questions used to measure the significance of time (T) and score (S) in program understanding were designed to represent three different levels of abstractions. The first abstraction level represents the interaction among the objects, the second level represents the interaction among methods, and the third level represents the interaction among methods and data.

Moreover, these questions were devised based on program understanding tasks defined by Pacione et al. [28] and Sillito et al. [29] for the software maintenance task.

Silito's questions covered various aspects in the software maintenance tasks, namely, domain, packages, classes, modules, methods, data, and variables. These questions are categorized into four categories, namely, initial focus points, building the points, understanding the relationship and relationship between the components. Silito's questions focused on the most common questions asked by the programmers, or software maintainers, which need to be answered during the maintenance tasks. On the contrary, the questions proposed by Pacione support both dynamic and static information. However, in this study, only questions on static information have been considered for static analysis.

The task design consists of the questions are related to understand the program to handle the maintenance tasks. The questions are required to be related to program understanding from the software documentation for the maintenance tasks. Therefore, in this study, the program understanding framework used by Pacione et al. [28] was considered to define the questions. There are nine tasks defined, covering the aspects of reverse engineering tasks such as the redocumentation process and other related components of reverse engineering to extract dynamic and static information. The list of the nine program understanding tasks are applied by Cornelissen et al. [30] to evaluate program understanding to complete the software maintenance tasks. In additional, Storey [31] has specified and summarized the study by Erdos and Sneed [32] with the most common questions asked about the tool designed to support the maintenance tasks. Lange et al. [33] and Ko et al. [34] have listed some of the questions to extract the software components for program understanding. The program understanding questions in this study are tailored with the existing studies specified above, as shown in Table 5. The questions basically cover the abstraction on low and high levels, and map the questions to the specific tasks based on abstraction level specified by Pressman [35].

3.8 Pilot Study

The pilot study needs to be conducted to make sure that the task designed for the control group is achievable in the allocated time. The pilot study was conducted with the Software Engineering Master's degree program students studying at University Teknologi of Malaysia (UTM). From the pilot study the questions, especially in Section B, were required to go through several refinement processes to ensure the subjects understood the questions asked and to avoid biases that had occurred among the grouping.

3.9 Conduct the Controlled Experiment

During the controlled experiment, the subjects were divided into three groups. The subjects were distributed into groups based on academic and industrial working experience in the software industry (past or present) and the expertise level in VB. Each group was provided with the tools (OBSR, UR, and VB), and the groups were labeled based on the tools they used, such as OBSR, UR, and VB. Training

No.	Questions	Abstraction Level	Tasks
QB1	What are the forms called by the form *frmStudentDetail*?	High (L3)	A1, A4, A6, A8, A9
QB2	Which event_procedures and forms call the form *frmStudentDetai*l?	High (L3)	A3, A4, A6, A8
QB3	List the methods called by function *isregistered* from form *frmSubjReg*. What are the methods name and the relevant object type?	Low (L4)	A1, A3, A4, A6, A8
QB4	Identify the function that retrieves the subject status which is called by the function *isregistered* from form *frmSubjReg*. What is the return value by the particular function?	Low (L4)	A1, A2, A3
QB5	How many methods call the database table *tblpayment*?	Low (Data Interaction) (L5)	A1, A3, A6
QB6	Which function and object type creates the database connection? Identify the variable return type for this function.	Low (Data Interaction) (L5)	A1, A69

Table 5. Proposed question in section B

session for an hour was conducted before the experiment itself with an explanation and demonstration on the tools provided. In addition, the technical manual of each tool was provided to the respective group. A brief introduction on the subject system (CMS) was also given to the subjects by conducting a demo of the CMS and providing a short description hand out about the modules available in the CMS. As specified earlier, with a brief description on the CMS and the familiarity of the subjects in the procedures of college and university environments, the subjects should be able to obtain an overview about the subject system.

A challenging task to organize the VB programmers for this controlled experiment. Additionally, those that know VB programming come from diverse workplaces and universities. Conducting the experiment in a single session was not possible. Therefore, the experiment was done in sessions for both academic and industrial subjects. Academic subjects were tested in the university, whereas the industrial tests were conducted after work. No distractions, subjects were not in a contact with others, and they were not given alternate methods to assist with the experiments. Because of this, equivalent steps could not be used for all the experiments. Nonetheless, recommendations had been developed beforehand.

To carry out the experiment, the subjects were provided with a laptop and the test tool installed. They were asked to follow the following procedures to answer the questionnaire given to them:

1. Read the description given on the first two pages on the purpose and the term definition of this experiment.

2. Next, continue with answering the survey in Section A, which was related to the previous or current working experience.
3. Give a description on the maintenance tasks related questionnaire in Section B to the subjects by using the related tools.

The subjects were also provided with a digital watch to record the start time and end time for answering the questions. However, there was no time limit for them to answer the questions. They were allowed to clarify some questions regarding the tools used. All the sessions were supervised closely to prevent the subjects from consulting others or using other tools.

4 RESULT AND DISCUSSION

In this section, the analysis of the results is discussed after the controlled experiment is conducted with the 33 participants from three different groups, namely, OR group (Ontology Based Software Redocumentation tool), UR group (Universal Report tool), and VB group (VB tool), as specified in Subsection 3.6. The results are analysed based on dependent variables time (T_1 and T_2) and score (S) to answer the six questions in Table 6. The analysis of the results for each dependent variable is discussed separately in the following subsections.

4.1 Controlled Experiment Result Analysis Based on Time (T_1)

The first step in the experiment analysis is to record the time it takes to answer the six questions in section B. In Figure 3, the OR group answered all questions in the shortest time, except for Questions 2 and 6. The VB and UR groups took the longest time to answer Question 2 (224 s) and Question 3 (253 s), respectively.

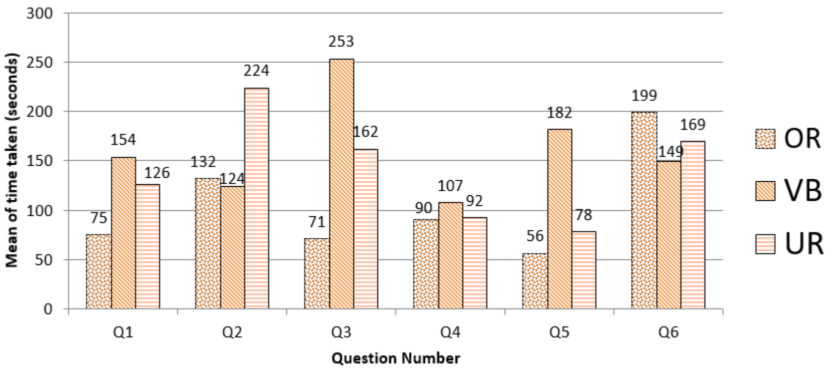


Figure 3. Mean value for T_1 for each redocumentation tool (regardless of correctness)

The summary value of descriptive statistics (mean, median, and standard deviation) for each tool for 11 participants is shown in Table 6. There is no major difference in timing based on T_1 identified in the UR group. However, the VB group shows a median of 16.49 % compared to the mean value. There are six subjects that had taken over 300 seconds to answer questions in the VB group, which affects the mean value. However, overall, based on the mean data, the OR group obtained a 26.80 % decrease in completion time compared to the UR group, and a 35.80 % decrease in completion time compared to the VB group.

Question	OR			UR			VB		
	Mean	Median	Standard Deviation	Mean	Median	Standard Deviation	Mean	Median	Standard Deviation
Q1	74.727	80.000	20.967	125.727	120.000	37.721	153.818	135.000	70.188
Q2	131.909	115.000	62.168	223.636	237.000	61.296	124.000	126.000	36.984
Q3	70.546	65.000	24.345	161.818	165.000	25.945	253.364	237.00	103.160
Q4	90.000	88.000	18.319	92.091	81.000	30.622	107.455	90.000	50.049
Q5	56.364	51.000	14.868	77.909	74.000	18.223	181.909	165.000	82.725
Q6	199.091	163.000	115.915	169.455	162.000	53.573	143.394	114.000	81.543
Overall	**103.773**	**84.500**	**72.937**	**141.773**	**129.500**	**63.409**	**161.652**	**135.00**	**85.630**
Median Compare to Mean		**−18.57 %**			**−8.66 %**			**−16.49 %**	
Time, T_1 reduce based on mean				**26.80 %**			**35.80 %**		

Table 6. Descriptive statistics for time taken T_1

Next, the Box's M test was conducted. The results demonstrate that the p-value was less than 0.1, and null hypothesis $H1_0$ is rejected for all six questions. On the other hand, Johnson and Wichern [25] say that this result can be ignored if the sample size for each tool is less than 20, which is applicable in our experiment. After that, the MANOVA might proceed to look for the significant value. To summarize, the MANOVA test for the six questions shows that time taken T_1 has a significant difference among the groups, with a p-value of 0.001, as shown in Table 7. The null hypothesis $H1_0$ is rejected based on the MANOVA results. ANOVA test found a statistically significant difference in the time required to answer questions Q1, Q2, Q3, and Q5, as seen in Table 8, which is below the 0.1 probability threshold.

	Box's M	**Wilks' Lambda**	**F statistic**	**Sig.**
Six Questions	132.839	0.106	8.627	< .001

Table 7. Overall test using MANOVA

More analysis is required to identify the group that has a substantial mean pair to answer the program understanding questions. Post-hoc, the Tukey HSD test was performed. Table 9 shows the summaries of the findings of the comparisons, except for Q4 and Q6. The negative sign in the mean difference values suggests that the first group completed the question in a shorter amount of time as compared to the second group. The OR group was able to exhibit a significant improvement for Q1 and Q3, as shown in Table 9.

Question	**F(2, 32)**	**p-value**
Q1	7.814	0.02*
Q2	11.260	< 0.000*
Q3	23.155715	< 0.000*
Q4	0.634	0.538
Q5	20.108	< 0.000*
Q6	0.900	0.417

Table 8. ANOVA test results for time taken (T_1) regardless of correctness

Questions	**Significance Pair**	**Mean Difference**	**p-value**
Q1	OR-UR	-51.000	0.045
	OR-VB	−79.091	0.001
Q2	OR-UR	−91.727	0.001
	VB-UR	−99.636	0.001
Q3	OR-UR	−91.273	0.005
	OR-VB	−182.81	0.000
	UR-VB	−91.546	0.005
Q5	OR-VB	−125.546	0.000
	UR-VB	−104.000	0.000

Table 9. List of pairs that have a significant difference based on the mean of completion time (T_1) for each question using the Tukey HSD test ($\alpha = 0.1$)

The Post-Hoc analysis shows that the OR group's completion time was able to show a significant mean difference in three questions compared to UR and VB, namely, Q1, Q2, and Q3. However, the UR group was only able to show a significant mean difference compared to VB for two questions, namely, Q3 and Q5. Overall, the UR and VB group were not able to show a significant improvement based on T_1 compared to the OR group. However, the OR group was able to show a significant improvement compared to both the UR and VB groups. Based on the Tukey HSD test results, the alternative hypothesis (H2) can be accepted for Q1, Q2, Q3, and Q5.

4.2 Controlled Experiment Result Analysis Based on Score (S)

The following analysis is based on the score (S). A higher average for the score was observed for all questions answered by the tools in the experiment. Figure 4 shows the highest scores for Q1 (10), Q3 (10), Q4 (10), Q5 (11) and Q6 (10), by the OR group. For Q4 (10), the UR group shares the same value with the VB group. The lowest score was obtained by the UR group and the VB group, which also share the same value for Q3 (2).

Table 10 shows the descriptive statistics for each question, and the questions overall: Descriptive data reveal that for each question, the OR group's mean difference is bigger compared to other tools. The Box's M test was conducted. The p-value is less than 0.1, hence the null hypothesis $H2_0$ can be rejected for the Box's

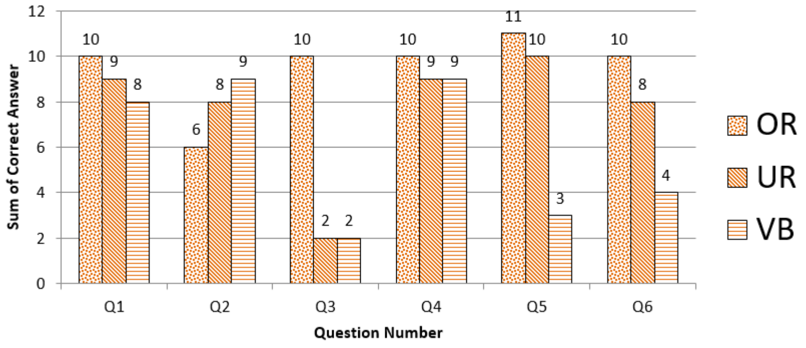


Figure 4. Sum of score for correct answers by group

M test. Refer to Subsection 3.4 for clarification on the Box's M test, which does not equate covariance matrices. This finding can be ignored in this experiment if the sample size for each tool is less than 20, which is feasible with 11 subjects per tool. Next, the MANOVA test was continued, and the results showed that the score (S) obtained had a significant difference among the groups, with a p-value of $< .001$, as shown in Table 11. Therefore, $H2_0$ can be rejected.

Question	**OR**			**UR**			**VB**		
	Mean	**Median**	**Standard Deviation**	**Mean**	**Median**	**Standard Deviation**	**Mean**	**Median**	**Standard Deviation**
Q1	0.909	1.000	0.302	0.818	1.000	0.405	0.727	1.000	0.467
Q2	0.545	1.000	0.522	0.727	1.000	0.467	0.818	1.000	0.405
Q3	0.909	1.000	0.302	0.182	0.000	0.405	0.182	0.000	0.405
Q4	0.909	1.000	0.302	0.818	1.000	0.405	0.818	1.000	0.405
Q5	1.000	1.000	< 0.000	0.909	1.000	0.302	0.273	0.000	0.467
Q6	0.909	1.000	0.302	0.727	1.000	0.467	0.364	0.000	0.505
Overall	**0.864**	**1.000**	**0.346**	**0.697**	**1.000**	**0.463**	**0.530**	**1.000**	**0.503**
Median Compare to Mean				**−23.90 %**			**−62.85 %**		

Table 10. Descriptive statistics for Score (S))

The ANOVA test was conducted and test results show that the significant difference in the score was obtained by the groups for Q3, Q5, and Q6, as shown in Table 12 (less than an alpha value of 0.1). The significant pair difference of the score among the groups for the Q3, Q5, and Q6 was identified using Post-Hoc ANOVA Tukey HSD, as shown in Table 13. In particular, the OR group shows a significant improvement in Q3 compared to the UR group (0.727) and VB group (0.727); Q5 and Q6 for VB (Q5: < 0.001) (Q6: 0.016) group only. The OR group was able to show 50 % effectiveness compared to the UR group, which only obtained 17 % effectiveness in answering the program understanding questions. Based on the Tukey HSD test results shown in Table 13, the alternative hypothesis H2 can be accepted for Q3, Q5, and Q6.

	Box's M	**Wilks' Lambda**	**F statistic**	**Sig.**
Six Questions	46.416	0.222	4.680	$< .001$

Table 11. Overall test using MANOVA

Question	**F(2, 32)**	**p-value**
Q1	0.576923	0.568
Q2	0.972222	< 0.390
Q3	13.913043	< 0.001*
Q4	0.217391	0.806
Q5	16.764706	< 0.001*
Q6	4.516129	0.019*

Table 12. ANOVA test results for correctness (S)

4.3 Controlled Experiment Result Analysis Based on Time (T_2)

The following analysis process proceeds to attempt to capture the time for the correct answer only T_2. The mean of the time taken T_2 by the OR group is the shortest for all of the questions except for Question 2 and 6. The UR group takes the longest time to answer Question 2 and 4 correctly. The VB group takes the shortest time to answer Question 2 and 6 correctly, and the longest time taken for answering Question 1, 3 and 5, as shown in Figure 5.

Mean, median, and standard deviation T_2 for each question are displayed in Table 14. The OR group had a 29.96 % drop in completion time when compared to the UR group, and a 68.47 % drop in completion time when compared to the VB group. The next test carried out was the MANOVA test. The MANOVA test could not be done due to the fact that the sample size was varied for each group. Alternatively, the ANOVA test was conducted. For every question except question Q4, as shown in Table 15, the results demonstrate a statistically significant difference. Additional study is necessary to discover which of the groups that shown a significant improvement to the program understanding questions.

Hence, a post-hoc analysis test was performed using Hochberg's GT2, as shown in Table 16, for all questions except Q4. The negative sign in the mean difference values suggests that the first group completed the question in a shorter amount of

Questions	**Significance Pair**	**Mean Difference**	**p-value**
Q3	OR-UR	0.727	< 0.001
	OR-VB	0.727	< 0.001
Q5	OR-VB	0.727	< 0.001
	UR-VB	0.636	< 0.001
Q6	OR-VB	0.545	0.016

Table 13. List of pairs that have a significant difference based on the mean of correctness for each question using the Tukey HSD test ($\alpha = 0.1$)

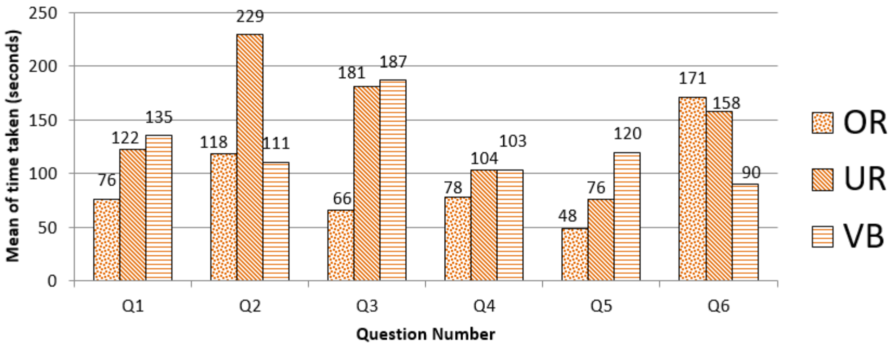


Figure 5. Mean of time taken T_2 to answer each question correctly for each tool

Question	OR			UR			VB		
	Mean	Median	Standard Deviation	Mean	Median	Standard Deviation	Mean	Median	Standard Deviation
Q1	76.200	82.000	21.493	122.444	112.000	40.544	135.125	127.000	39.139
Q2	118.333	114.500	17.374	229.375	228.000	65.459	110.667	116.000	24.510
Q3	65.800	63.000	19.578	181.000	181.000	43.841	187.00	187.000	70.7111
Q4	88.000	86.500	18.000	92.333	73.000	33.864	103.222	86.000	54.408
Q5	56.364	51.000	14.868	76.000	73.000	18.012	119.667	103.000	34.196
Q6	171.000	162.000	72.696	157.875	163.000	42.466	90.000	101.500	31.885
Overall	**93.684**	**83.000**	**52.048**	**133.761**	**119.500**	**66.999**	**117.114**	**103.000**	**44.402**
Median compare to Mean		**−11.40 %**			**−10.67 %**			**−12.05 %**	
Time, T_2 reduced based on mean				**−29.96 %**				**−68.47 %**	

Table 14. Descriptive statistics for Time T_2

time, as compared to the second group. Table 16 demonstrates that the OR group had an improved time taken T_2 to answer Q1, Q3, and Q5 (when compared to both the UR group (Q1: −46.244, Q3: −115.200, Q5: −19.636) and the VB group (Q1: −46.244, Q3: −121.200, Q5: −63.303)) and Q2 with UR group (Q2: −111.043). The VB groups has improved compared to the OR group for Q6. The OR group was able to properly answer 83 % of the questions, compared to the UR group, with an efficiency of only 33 %. Based on the Hochberg's GT2 test, the null hypothesis $H1_0$ can be rejected, and the alternative hypothesis H1 should be accepted for all questions, except for Q4, based on time T_2.

The results presented in Table 16 show that the OR group shows a significant improvement in the time taken T_2 to correctly answer Q1, Q3, and Q5, compared to both the UR group (Q1: −46.244, Q3: −115.200, Q5: −19.636), and the VB group (Q1: −46.244, Q3: −121.200, Q5: −63.303), and Q2 with UR group (Q2: −111.043). For Q6, the VB groups shows an improvement compared to the OR group. The OR group was able to show 83 % efficiency to correctly answer the

Question	**Between Groups**	**Within Groups**	**F**	**p-value**
Q1	2	24	7.640	0.003*
Q2	2	20	19.274	0.001*
Q3	2	11	21.187	0.001*
Q4	2	25	0.401212	0.673742
Q5	2	21	13.580795	0.001*
Q6	2	19	2.893	0.080*

Table 15. ANOVA test result for Time T_2

questions compared to the UR group, with an efficiency of only 33 %. The detailed findings of how the tools and subjects involvement contributes to the efficiency to correctly answer the questions are discussed in Section 5. Based on the Hochberg's GT2 test, the null hypothesis $H1_0$ can be rejected, and the alternative hypothesis H1 should be accepted for all questions, except for Q4, based on time T_2.

Questions	**Significance Pair**	**Mean Difference**	**p-value**
Q1	OR-UR	−46.244	0.021
	OR-VB	−58.925	0.004
Q2	OR-UR	−111.043	< 0.001
	UR-VB	118.708	< 0.001
Q3	OR-UR	−115.200	0.002
	OR-VB	−121.200	0.001
Q5	OR-UR	−19.636	0.076
	OR-VB	−63.303	< 0.001
	UR-VB	−43.667	0.006
Q6	OR-VB	81.000	0.080

Table 16. List of pairs that have a significant difference based on the mean of time T_2 for each question using Hochber's GT2 test ($\alpha = 0.1$)

5 FINDINGS FROM THE CONTROLLED EXPERIMENT

Based on the analysis in the previous sections, the results show that the software maintainers are able to understand and provide significant improvement in program understanding by utilizing the OBSR tool to accomplish the maintenance task. In this section, each question is discussed to find the impact of the tools to understand the program and support the maintenance task.

Q1: What are the forms called by the form frmStudentDetail?

The main purpose of Question 1 is to identify and understand the overall view of the software components in the subject system (CMS). The specific question has the aim to understand the interaction between frmStudentDetail and other forms during the maintenance task.

The OR group showed significant improvement in T_1 and T_2 by a 68 % and a 61 % mean difference, respectively, when compared to the UR group, and 91 % of the subjects obtained the correct response. This OBSR tool was able to show the forms called by the frmStudentDetail by searching the object name in the Search Form section. For subjects, the form must be named frmStudentDetail and clicking the Generate Graph button, as shown in Figure 6 a), and generates the dependency graph, as shown in Figure 6 b). The subjects gained replies by querying the semantic links between objects in the OBSR tool. For the UR tool, subjects must find the location of the form fromStudentDetail from the tree view, which is taking 60.5 % longer. Conversely, the VB group used the search capability built into the tool to search the forms called by frmStudentDetail with the ".show" word. Therefore, by using the local search, the subject is able to get the forms called by frmStudentDetail. This happened, but only to a few participants in the VB group who had unintentionally clicked on global search. This caused inaccurate results. Several of the individuals began from the beginning, which influenced the VB group's results on T_1 and T_2 by a 51 % and 44 % mean difference, respectively, with the OR group.

In the UR tool, users can click on the Form Diagram section, and the dependencies between forms will be shown in tree view. The subjects have to find the location of form frmStudentDetail from the tree view, which consumes 60.5 % more mean difference in time to get the correct answer in searching the called form, compared with the OR group. The efficiency of finding the form will be worse if the software system becomes more complex.

Q2: Which event procedures and forms call the form frmStudentDetail?

Question 2 Lists the form(s) that invoke frmStudentDetail in order to identify the internal structure and relationships within the subject system (CMS). Using VB 6.0 in the tool, the VB group discovered the solutions for T_1 and T_2 with a 6.5 % and 6 % percent mean difference, respectively, as compared to the OR group. The subjects' experience in Q1 improves the use of the search function in Q2.

The average time taken to answer Q2 correctly by the UR group is 106 % longer than by the VB group. The UR group was required to trace the frmStudent-Detail.show command in every single method, and some of the participants overlooked of the content.

On the other hand, surprisingly, the OR group had taken an average time of 6.3 % longer to correctly answer the question compared to the VB group. The OR group provides the semantic navigation (isCallBy), where the users are only required to click on the frmStudentDetail shown in Figure 6 b). This will list the method name, which is called the frmStudentDetail, as shown in Figure 7. Some of the subjects in the OR groups were confused with the semantic link provided, and manually searched in the method diagram section.

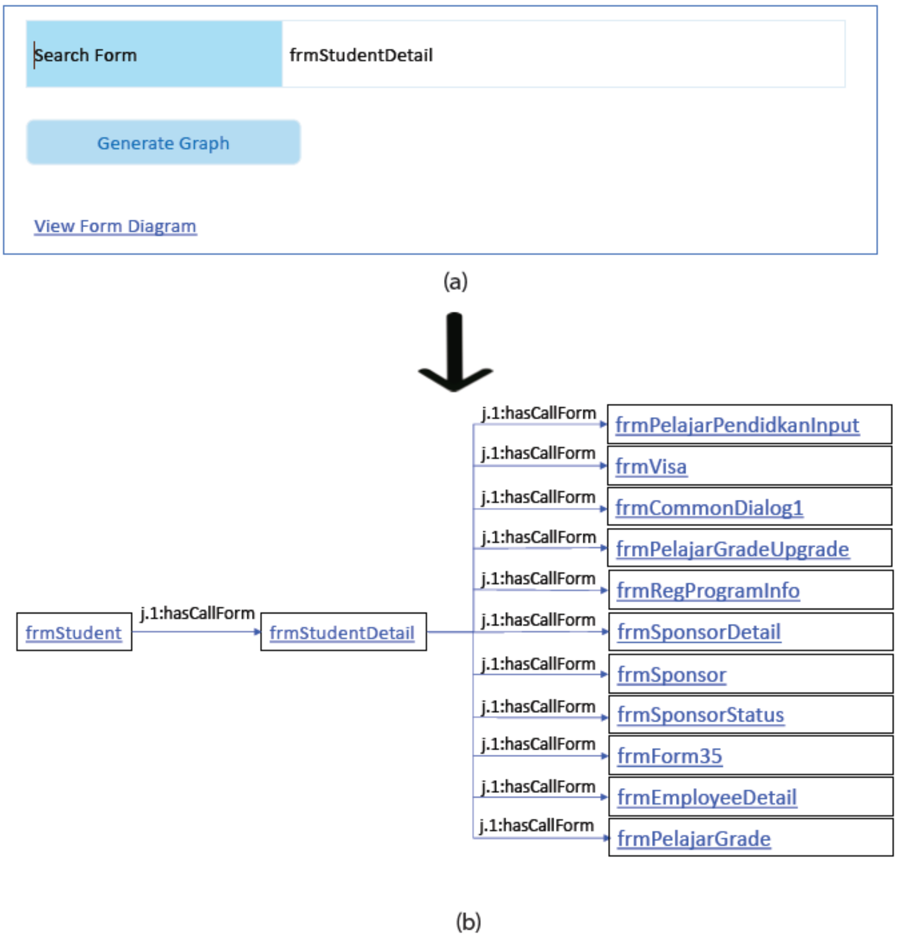
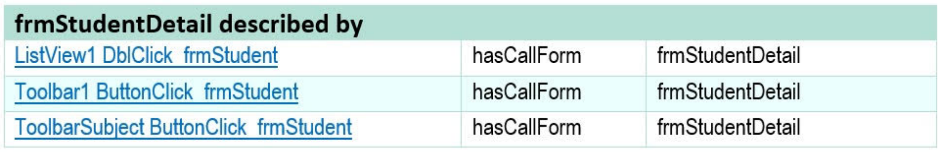


Figure 6. a) Search forms function. b) List of the forms called by form frmStudentDetail generated in the dependency graph view via OBSR tool.



frmStudentDetail described by		
ListView1 DblClick frmStudent	hasCallForm	frmStudentDetail
Toolbar1 ButtonClick frmStudent	hasCallForm	frmStudentDetail
ToolbarSubject ButtonClick frmStudent	hasCallForm	frmStudentDetail

Figure 7. List of the methods called by the form frmStudentDetail using OBSR tool

Q3: List the methods called by function isregistered from form frmSubj-Reg. What are the methods names and the relevant object types?

Question 3 is useful for identifying functional dependencies within other methods. The OR group responded in the quickest time for both T1 and T2, and their overall efficiency was 56 % and 64 % greater, respectively, compared to the UR group, and 91 % of the individuals obtained the correct answer. The OR group was able to get the solution to this question by examining the term isregistered_frmSubjReg in Figure 8 a). Figure 8 b) depicts the dependence graph. The hasCallFunction semantically links the functions, illustrating the dependency relationship. Instead, the subject might utilize the textual search feature to build the list.

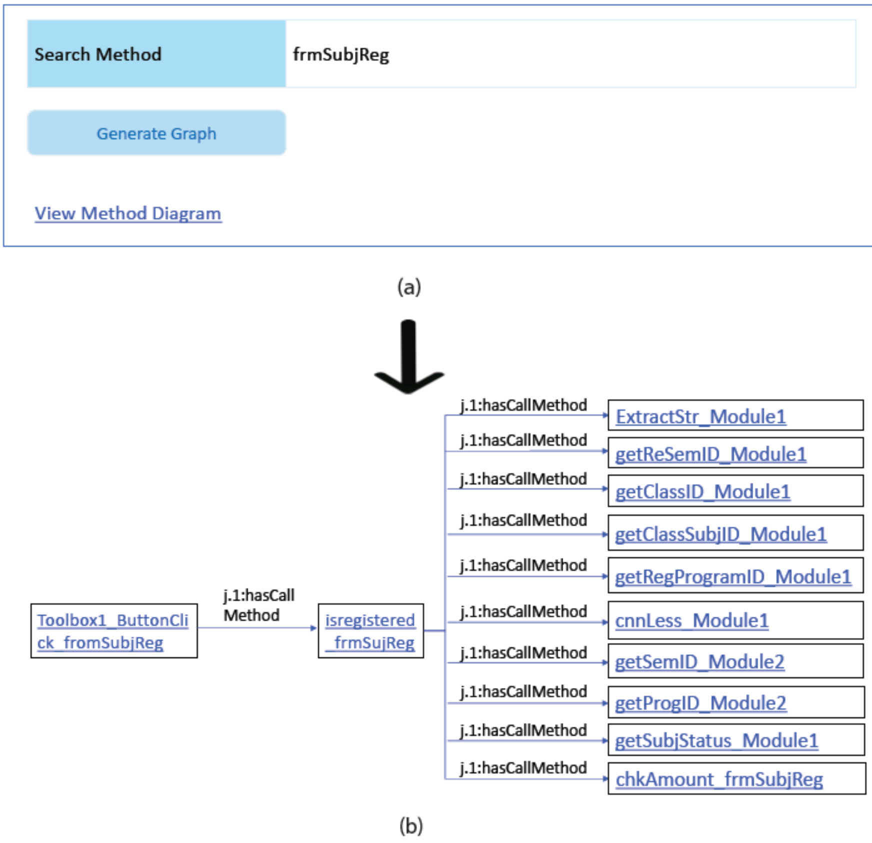


Figure 8. a) Search method. b) List of the methods called by the function isregistered_frmSubjReg method generated in the dependency graph view via OBSR tool.

The UR group took 128 % and 174 % more time than the OR group, respectively, for both T_1 and T_2 mostly due to the lack of a search option in the UR group. Only 18 % of the students were able to correctly answer the test questions. The subjects opted for an alternative approach in choosing the methods: the UR offers hyperlink and blue text color to distinguish the techniques. This feature benefits the subjects, although all the approaches are not indicated in the UR tools. The subjects were instructed to look very closely to see if anyone employed an alternative strategy. The subjects were also unsure – it is a method or a variable? And so several of them chose the incorrect answer.

The VB group had the greatest discrepancy in time T_1 and T_2, with a mean difference of 256 %. This shows that the subjects were unable to locate the methods using only the source code. The VB participants had the same challenge as the UR group, and they had to look for each term individually. Click "definition" to find whether the word is a method or a variable. Some of the individuals were able to identify procedures based on their parameters or values. This illustrates that the OBSR tool looks for and finds related components based on semantic relationships, and outperforms the other two tools.

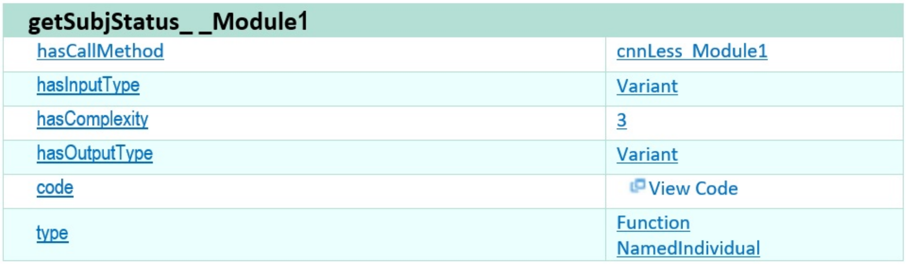
Q4: Identify the function that retrieves the subject status which is called by function isregistered from form frmSubjReg? What is the return value by the particular function?

Question 4 basically requires the groups to find the function used in the function isregistered under the form frmSubjReg. The required function retrieves the subject status from the list of functions. The subjects were required to find the function name from the list identified in Q3. In addition, the groups were required to identify the return value of that function.

The OR group was the fastest group for T_1 by 2 % compared to the VB group. For T_2, it had a significant improvement of 25 % compared to the UR group. The OR group was required to go through the 10 methods generated from Q3 and find the function which retrieves the subject status. The OR group was able to answer this question more efficiency and effectiveness compared to the other two groups, as the answer can be retrieved from the function name itself, which represents the task of its function in the dependency diagram. The OR group navigated the functional dependency diagram using a link provided by the function getSubjStatus to find the return value, as shown in Figure 9.

As in Q3, the UR group identified the function name from the link provided for each method in the source code view. However, the subjects had to manually search through source code view and find the function name getSubjStatus. The subjects had taken 2.2 % more time compared to the OR group to go through the source code and find the getSubjStatus function. Next, the subjects clicked on the function getSubjStatus to view the code and find the return value.

The VB group took a 19 % longer time compared to the OR group in T_1. Although, in T_2, they were 0.1 % better than the UR group. Additionally, 82 % of



getSubjStatus_ _Module1	
hasCallMethod	cnnLess_Module1
hasInputType	Variant
hasComplexity	3
hasOutputType	Variant
code	View Code
type	Function NamedIndividual

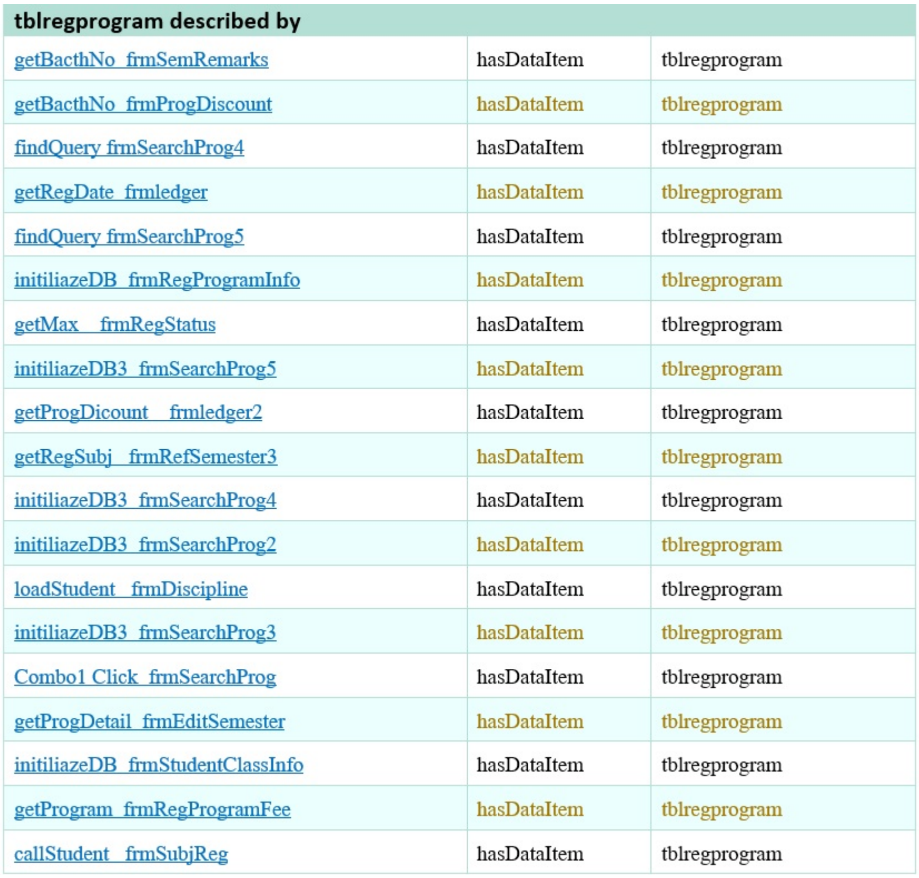
Figure 9. Return value (hasOutputType) and other related descriptions for function getSubjStatus in OBSR tool

the participants scored correctly, which is little lower than the OR group with 91 % of the respondents achieving the correct answer. The VB group was further forced to manually look through the source code line by line to find the procedure. Some of the subjects simply submitted the answer after first finding the solution using the same methodology, which is erroneous. In conclusion, the answer status was "subjstatus", but in reality it was a variation variable. It is a positive result because the activity was completed more efficiently, with useful links and navigation provided.

Q5: How many methods call the database table tblRegProgram?

Question 5 required the subject to understand the data level interaction with the methods in the program. This aids the program maintainers through the adaptive and perfective maintenance tasks. This question asked the subjects to list the functions that interact with or use the table tblRegProgram. During the controlled experiment, the OR group was able to improve the program understanding by a 28 % and 37 % mean difference when compared to the UR group. The OR group obtained a 100 % correct answer for Q5. In the OBSR tool, the subjects were required to locate the table name from the general search option. These results reveal all the tables associated by the semantic relationship of "hasCalledData" and which methods use the table name (Figure 10). Subjects who opted to search and traverse the table tblRegProgram in the main part on "Data Item" found the table tblRegProgram. Once the user clicks on it, it lists the object properties and value. Using the object attributes, the OR group found the list of the table. When a function's use of a table is significant, a detailed description of the table is essential for the function.

Although, the UR group's performance at time T_1 and T_2 were 39 % and 58 % lower, respectively, than the OR group, 10 % UR group offers the search feature, which is slower and requires more time to find the table tblRegProgram with keyword search in the complete program. This is usually done by using the Glossary Function in the UR Tool. The glossary offers the list



tblregprogram described by		
getBacthNo frmSemRemarks	hasDataItem	tblregprogram
getBacthNo frmProgDiscount	hasDataItem	tblregprogram
findQuery frmSearchProg4	hasDataItem	tblregprogram
getRegDate frmledger	hasDataItem	tblregprogram
findQuery frmSearchProg5	hasDataItem	tblregprogram
initiliazeDB frmRegProgramInfo	hasDataItem	tblregprogram
getMax frmRegStatus	hasDataItem	tblregprogram
initiliazeDB3 frmSearchProg5	hasDataItem	tblregprogram
getProgDicount frmledger2	hasDataItem	tblregprogram
getRegSubj frmRefSemester3	hasDataItem	tblregprogram
initiliazeDB3 frmSearchProg4	hasDataItem	tblregprogram
initiliazeDB3 frmSearchProg2	hasDataItem	tblregprogram
loadStudent frmDiscipline	hasDataItem	tblregprogram
initiliazeDB3 frmSearchProg3	hasDataItem	tblregprogram
Combo1 Click frmSearchProg	hasDataItem	tblregprogram
getProgDetail frmEditSemester	hasDataItem	tblregprogram
initiliazeDB frmStudentClassInfo	hasDataItem	tblregprogram
getProgram frmRegProgramFee	hasDataItem	tblregprogram
callStudent frmSubjReg	hasDataItem	tblregprogram

Figure 10. List of functions using table tblRegProgram in OBSR tool

of code phrases in alphabetical order. Nevertheless, the glossary gives you the source code for each procedure in the table tblRegProgram. Long results lead to the number of methods that use the table tblRegProgram being difficult to find. Overall, the UR group's efficiency differs from the other two groups.

Compared to the OR group, the VB group had the longest duration by a 225 % and 150 % mean difference for T_1 and T_2. 29 methods utilised the tblRegProgram, and the users had to manually find and count each. Many subjects had to check and tally procedures to make sure that the table tblRegProgram is not misused. Additionally, it does not provide a single viewpoint to skim at all the methods in the list to comprehend their behavior. Thus, for quick response during maintenance jobs, relevant searching output and input to the future maintenance actions are required.

Q6: Which function and object type creates the database connection? Identify the variable return type for this function.

The last question is Question 6, requires the subjects to find return type of the method, and the database connection string supplied by the function. This is a portion of the previous question. When we have methods using table tblReg-Program, we can expect to see that table function in action. Subject should be able to locate the function in the list of methods on Question 3. The OR group took the longest time for T1 and T2 by a 34 % and 90 % mean difference, respectively, but had the greatest total, with 91 % of subjects answering correctly. Only one function from Question 3 was required, and the function that found the database connection was shown. More than half of the OR group members found the cnnLess function. The subjects were also required to check into the real code to verify the answer. The subjects were simply required to explore the semantic link through the object characteristics hasCallMethod and hasOutput-Type. It takes the longest since subjects have to comprehend the proposition, devise a strategy, then apply it in the OBSR tool to get the proper response. The navigation delivers relevant links that assists the subjects in finding the proper solution.

In contrast, the UR group took longer time for both T_1 and T_2 by a 17 % and 76 % mean difference, respectively, compared to the VB group. UR group had to choose one of the methods offered in Q5 and find the connection string. UR group was tasked with going through the source code line by line to discover the correct connection string function. The List of Routine was required in order to verify the chosen approach. This included fewer steps, and hence, the UR group had taken a longer time.

The VB group took the shortest time for T_1 and T_2 by a 15 % and 43 % mean difference, respectively, compared to the UR group. However, only 37 % of the VB group subjects were able to score correctly. The VB group had to find from the source code related to Question 5, which have highlighted the term tblpayment.

The OR group performed better on all of the questions except for Question 2, although the substantial improvement was seen for Questions 3, 5, and 6. Nevertheless, only in Question 3 is the OR group's mean different from the UR group and the VB group. Based on the correctness, the OBSR technique has a significant statistical advantage in program understanding. Hence, $H2_0$ can be rejected as the null hypothesis, and H2 is accepted for the alternative.

The OR group exhibits a substantial mean difference for both variables T1 and T2 for Questions 1 and 3. Only in Question 5 do the OR group and the VB group have substantial differences. But, VB outperforms OR for variable T_2's in Question 6. However, in comparison to the UR group, the OR group experiences greater improvements for both T_1 and T_2 in Question 2, and in Question 5 as well. It was found that OBSR considerably improved the program knowledge to perform the maintenance task for Questions 1, 2, 3, 5, and 6. Therefore, $H1_0$ can be rejected, and the alternative H1 can be accepted for all questions except for Question 4.

6 ANALYSING VALIDITY THREATS

This section provides a detailed discussion on the possible threats to this controlled experiment design. The validity threats discussed in this study are based on four categories; internal, external, construct and conclusion validity [36]. Each validity test consist of several factors that are important to make sure proper planning is in place to avoid problems that can mislead the results of the controlled experiment.

6.1 Internal Validity

The internal validity basically includes the factors that affect the treatment of the dependent variable. Some of the factors are as follows:

Task: The task design may be biased to the advantage of the OBSR tool. This threat has been diminished by using the task design proposed by the existing evaluation using the controlled experiment in program understanding [28, 31, 33].

Different session: The fact that the subjects were required to be involved in this experiment in separate sessions may affect the results. To mitigate this threat, the pilot experiment was conducted with the Master's degree program students to obtain a stable and reliable experimental setup by refining the question structure or the experimental procedure. In addition, for each session, the subjects were provided with clearly defined procedures and guidelines to understand the overall flow of the experiment.

6.2 External Validity

The external validity basically includes the factor which gives generalizability of experimental results. In this controlled experiment, subjects are categorized incorrectly and may not be skilled enough. Before the trial, subjects were examined to determine how long they have been active in different types of projects and utilizing VB in software development. To make sure that the subjects understood the term, they were also provided an explanation relating the skill level (novice, advanced beginners, competent, proficient and expert). The second issue is if expertise is not dispersed properly according to the numerous independent factors, which may impact the findings. This is due to the shortage of software practitioners with basic and expert skills.

7 CONCLUSION AND FUTURE WORK

We presented a controlled experiment aimed at evaluating the OBSR approach to understand the extensive of the OBSR tool in improving program understanding during the software maintenance task. The experiment was measured based on time (T_1 and T_2) and correctness (S) of the answers provided by the subjects. The analysis of the experiment was conducted by using MANOVA and ANNOVA tests. In addition, the Post-Hoc test was also conducted to identify the significant mean difference from the ANOVA results. The results revealed that the OBSR prototype tool showed a significant difference based on the time required to complete the questions. Overall, for T_1, the alternative hypotheses for both $H1_0$ and $H2_0$ were accepted based on the significant improvement in the Post-Hoc analysis by the OR group compared to the UR and VB groups. However, further evaluation is required to strengthen the capability of the OBSR approach in terms of usability and features provided by the OBSR approach compared to the existing approach. The usability study is important to identify whether the OBSR tool is able to provide features and functions that can assist software maintainers as equivalent as or better than to the existing tools. On the other hand, feature analysis also required to compare the capabilities of the OBSR approach and existing redocumentation approaches in querying the repository and presenting the exploration functionalities.

REFERENCES

[1] Chikofsky, E. J.—Cross, J. H.: Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software, Vol. 7, 1990, No. 1, pp. 13–17, doi: 10.1109/52.43044.

[2] Geist, V.—Moser, M.—Pichler, J.—Beyer, S.—Pinzger, M.: Leveraging Machine Learning for Software Redocumentation. 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2020, pp. 622–626, doi: 10.1109/SANER48275.2020.9054838.

[3] Le Borgne, A.: ARIANE: Automated Re-Documentation to Improve Software Architecture Understanding and Evolution. Doctoral dissertation, École nationale supérieure des Mines d'Alès (IMT Mines Alès), 2020.

[4] Sneed, H—Verhoef, C.: Re-Implementing a Legacy System. Journal of Systems and Software, Vol. 155, 2019, pp. 162–184, doi: 10.1016/J.JSS.2019.05.012.

[5] Purbasari, A.: Software Redocumentation to Support the Maintenance of an Integrated Information System at University of Pasundan Bandung. National Conference of Information System (KNSI), Bandung Institute of Technology. Available at: `http://jurnal.atmaluhur.ac.id/index.php/knsi2018/article/view/494/419`, 2018. (Software Redocumentation untuk Mendukung Pemeliharaan Sistem Informasi Terpadu Universitas Pasundan (SITU)).

[6] Canfora, G.—Di Penta, M.—Cerulo, L.: Achievements and Challenges in Software Reverse Engineering. Communications of the ACM, Vol. 54, 2011, No. 4, pp. 142–151, doi: 10.1145/1924421.1924451.

[7] van Deursen, A.—Moonen, L.: Documenting Software Systems Using Types. Science of Computer Programming, Vol. 60, 2006, No. 2, pp. 205–220, doi: 10.1016/j.scico.2005.10.006.

[8] Tilley, S. R.: A Reverse-Engineering Environment Framework. Technical Report CMU/SEI-98-TR-005, Software Engineering Institute, Carnegie Mellon University, 1998. Available at: `http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=13047`, doi: 10.21236/ada343688.

[9] Quante, J.: Do Dynamic Object Process Graphs Support Program Understanding? – A Controlled Experiment. 2008 16th IEEE International Conference on Program Comprehension, 2008, pp. 73–82, doi: 10.1109/ICPC.2008.15.

[10] Sjoeberg, D. I. K.—Hannay, J. E.—Hansen, O.—Kampenes, V. B.—Karahasanovic, A.—Liborg, N.-K.—Rekdal, A. C.: A Survey of Controlled Experiments in Software Engineering. IEEE Transactions on Software Engineering, Vol. 31, 2005, No. 9, pp. 733–753, doi: 10.1109/TSE.2005.97.

[11] Easterbrook, S.—Singer, J.—Storey, M.-A.—Damian, D.: Selecting Empirical Methods for Software Engineering Research. In: Shull, F., Singer, J., Sjøberg, D. I. K. (Eds.): Guide to Advanced Empirical Software Engineering. Springer, London, 2008, pp. 285–311, doi: 10.1007/978-1-84800-044-5_11.

[12] Kitchenham, B.—Linkman, S.—Law, D.: DESMET: A Methodology for Evaluating Software Engineering Methods and Tools. Computing and Control Engineering Journal, Vol. 8, 1997, No. 3, pp. 120–126, doi: 10.1049/cce:19970304.

[13] Pfleeger, S. L.: Experimental Design and Analysis in Software Engineering. Annals of Software Engineering, Vol. 1, 1995, No. 1, pp. 219–253, doi: 10.1007/BF02249052.

[14] D'Avila, L. F.—Farias, K.—Barbosa, J. L. V.: Effects of Contextual Information on Maintenance Effort: A Controlled Experiment. Journal of Systems and Software, Vol. 159, 2020, Art. No. 1110443, doi: 10.1016/j.jss.2019.110443.

[15] Nallusamy, S.—Ibrahim, S.—Mahrin, M. N.: A Proposed Framework for Software Redocumentation Using Ontology Based Approach and Integration with Standard Software Documentation. The Third International Conference on Computer Engineering and Technology (ICCET 2011), 2011, Art. No. 79, doi: 10.1115/1.859735.paper79.

[16] Nallusamy, S.—Ibrahim, S.—Mahrin, M. N.: A Software Redocumentation Process Using Ontology Based Approach in Software Maintenance. International Journal of Information and Electronics Engineering, Vol. 1, 2011, No. 2, pp. 133–139, doi: 10.7763/ijiee.2011.v1.21.

[17] Harris, J.: CS1: Where is Visual Basic? Journal of Computing Sciences in Colleges, Vol. 16, 2001, No. 2, pp. 223–228.

[18] Logan, W.: Life After Visual Basic 6.0 – Where to Go from Here. 2008 IEEE International Automatic Testing Conference (AUTOTESTCON), 2008, pp. 518–521, doi: 10.1109/AUTEST.2008.4662673.

[19] Lakshmi, D. S—Perumal, R. S.—Mouli, P. V. S. S. R.C.: Challenges and Issues in Code Migration from VB 6.0 to VB.NET. International Journal of Computer Information Systems, Vol. 2, 2011, No. 3, pp. 56–59.

[20] Wettel, R.—Lanza M.—Robbes R.: Software Systems as Cities: A Controlled Experiment. Proceedings of the 33rd International Conference on Software Engineering (ICSE '11), 2011, pp. 551–560, doi: 10.1145/1985793.1985868.

[21] Jedlitschka, A.—Pfahl, D.: Reporting Guidelines for Controlled Experiments in Software Engineering. 2005 International Symposium on Empirical Software Engineering, 2005, pp. 95–104, doi: 10.1109/ISESE.2005.1541818.

[22] Juristo, N.—Vegas, S.: Analyzing Software Engineering Experiments: Everything You Always Wanted to Know But Were Afraid to Ask. Proceedings of the 38th International Conference on Software Engineering Companion, 2016, pp. 900–901, doi: 10.1145/2889160.2891054.

[23] Tadonki, C.: Universal Report: A Generic Reverse Engineering Tool. 12th IEEE International Workshop on Program Comprehension, 2004, pp. 266–267, doi: 10.1109/WPC.2004.1311073.

[24] Meijer, E.: Visual Basic. Companion to the 22nd ACM SIGPLAN Conference on Object-Oriented Programming Systems and Applications Companion (OOPSLA '07), 2007, pp. 860–861, doi: 10.1145/1297846.1297926.

[25] Johnson, R. A.—Wichern, D. W.: Comparisons of Several Multivariate Means. Applied Multivariate Statistical Analysis. Pearson, 2007, pp. 273–359.

[26] Di Penta, M.—Stirewalt, R. E. K.—Kraemer, E.: Designing Your Next Empirical Study on Program Comprehension. Proceedings of the 15th IEEE International Conference on Program Comprehension (IOPC '07), 2007, pp. 281–285, doi: 10.1109/ICPC.2007.17.

[27] Atherton, J.: Competence, Proficiency and Beyond. 2013 [cited 2013 27 May 2014]; Available at: `http://www.doceo.co.uk/background/expertise.htm`.

[28] Pacione, M. J.—Roper, M.—Wood, M.: A Novel Software Visualisation Model to Support Software Comprehension. 11th Working Conference on Reverse Engineering, 2004, pp. 70–79, doi: 10.1109/WCRE.2004.7.

[29] Sillito, J.—Murphy, G. C.—De Volder, K.: Questions Programmers Ask During Software Evolution Tasks. Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering (SIGSOFT '06/FSE-14), 2006, pp. 23–24, doi: 10.1145/1181775.1181779.

[30] Cornelissen, B.—Zaidman, A.—van Deursen, A.: A Controlled Experiment for Program Comprehension Through Trace Visualization. IEEE Transactions on Software Engineering, Vol. 37, 2011, pp. 341–355, doi: 10.1109/TSE.2010.47.

[31] Storey, M.-A.: Theories, Tools and Research Methods in Program Comprehension: Past, Present and Future. Software Quality Journal, Vol. 14, No. 3, pp. 187–208, doi: 10.1007/s11219-006-9216-4.

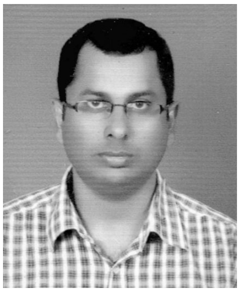
[32] Erdos, K.—Sneed, H. M.: Partial Comprehension of Complex Programs (Enough to Perform Maintenance). 6th International Workshop on Program Comprehension Proceedings (IWPC '98), 1998, pp. 98–105, doi: 10.1109/WPC.1998.693322.

[33] Lange, C.—Sneed, H. M.—Winter, A.: Comparing Graph-Based Program Comprehension Tools to Relational Database-Based Tools. Proceedings 9th International Workshop on Program Comprehension (IWPC 2001), 2001, pp. 209–218, doi: 10.1109/WPC.2001.921732.

[34] Ko, A. J.—DeLine, R.—Venolia, G.: Information Needs in Collocated Software Development Teams. 29th International Conference on Software Engineering (ICSE '07), 2007, pp. 344–353, doi: 10.1109/ICSE.2007.45.

[35] Pressman, R.: Software Engineering: A Practitioner's Approach. 7th Edition. McGraw Hill, 2010.

[36] Karoulis, A.—Stamelos, I. G.—Angelis, L.—Pombortsis, A. S.: Formally Assessing an Instructional Tool: A Controlled Experiment in Software Engineering. IEEE Transactions on Education, Vol. 48, 2005, No. 1, pp. 133–139, doi: 10.1109/TE.2004.837047.



Sugumaran Nallusamy received his Ph.D. in computer science (2015) from the University of Technology Malaysia. He is Assistant Professor attached to the Department of Internet Engineering and Computer Science, University Tunku Abdul Rahman, Sungai Long. He has been awarded with the Best Session Presentation at the International Conference on Advanced Computer Science and Information System, Indonesia in 2010 and also Certificate of Excellent Paper at the International Conference on Software and Information Engineering (ICSIE 2011). His field of interest is reverse engineering, big data, data analysis and redocumentation for software maintenance.

Meei Hao Hoo is Assistant Professor and Researcher at the Department of Internet Engineering and Computer Science, Universiti Tunku Abdul Rahman. She received her Ph.D. degree in information science from the National University of Malaysia. Her research interests include usability engineering, persuasive design and data mining. She has more than 10-year experience in teaching related to system development and design, and 5-year experience in the private sector in areas related to information system audit and system administration.

Farizuwana Akma Zulkifle is Senior Lecturer and Researcher at the Faculty of Computer Science and Mathematics at the University Teknologi Mara (UiTM). She holds Ph.D. degree in computer science from the University of Technology Malaysia. She teaches and researches in areas of data analysis, big-data, image processing and web application development. For 7 years she worked in the private sector in areas related to information systems planning and implementation projects.