Interactive scalable condensation of reverse engineered UML class diagrams for software comprehension

Osman, M.H.B.

Part III

Validation and Conclusion

Chapter 10

Validation

This chapter describes a study to validate the Software Architecture Abstraction (SAAbs) framework. This framework is purposely invented to simplify reverse engineered class diagrams by selecting the important classes in a system. Using the condensed class diagrams generated by the SAAbs tool, we validate the SAAbs framework based on the user view of these class diagrams. This study focuses on the usefulness of the condensed class diagrams for program comprehension.

10.1 Introduction

In this chapter, we present the validation of the SAAbs framework to enhance software comprehension of the RE-CD. Based on the previous chapters, we have found that Random Forests is the most suitable classification algorithm for this approach. Based on this, we have developed a tool (called Software Architecture Abstractor (SAAbs) tool - described in Chapter 9) to apply our approach to the RE-CD. As a result, the tool produces a ranking of importance classes for all classes in a class diagram. A higher score indicates that the class is important. With this ranking, condensations of a class diagram based on the importance of classes can be constructed. By using the condensations of the class diagrams, we carried out a survey to validate our approach. This validation aims at i) discovering the understandability of condensed class diagrams, ii) finding whether the condensed class diagram generated by this approach is helpful in understanding the software design and, iii) validating the SAAbs tool in assisting software developers to understand the software.

This chapter is part of a publication entitled **"Interactive Scalable Abstraction of Reverse Engineered UML Class Diagrams"**, In Proceedings of the 21st Asia-Pacific Software Engineering Conference (APSEC 2014)

The chapter is structured as follows. Section 10.2 describes the research questions and is followed by Section 10.3 that explains the experiment design. Section 10.4 demonstrates the results and we discuss our findings in Section 10.5. This is followed by the conclusions and our suggestions for future work in Section 10.6.

10.2 Research Question

In this chapter, we aim at answering the following question:

> **MQ:** *Can the SAAbs framework help the software developers for system comprehension?*

In order to answer the main question, four research questions need to be explored. The research questions are the following:

- **RQ1:** *What is the level of understandability of the condensed class diagrams created by SAAbs framework?*
- **RQ2:** *What level of abstraction is preferred to produce an overview of a software design (in particular, based on the respondents' background)?*
- **RQ3:** *Does the condensed class diagram represent the important information out of the reverse engineered class diagram?*
- **RQ4:** *What is the relative usefulness of SAAbs tool for different respondents' background?*

10.3 Experiment Design

In this section, we describe the study design. We explain the design of the questionnaire and how we conduct the study.

10.3.1 Questionnaire Design

The questionnaire is divided into four parts, which are: Personal Background, Choices of Class Diagram, Software Architecture Abstraction (SAAbs) framework and The SAAbs tool. The explanations of these parts are the following.

Part A: Personal Background

In this part, we collect information about the respondent background. The respondents were asked about their skill, experience, frequency of using UML class diagrams and also their experience in reverse engineering source code into UML Class Diagrams. This information is used to discover the relation between the respondents' background and their answers.

Table 10.1: *Type of Class Diagram used in this study*

Class Diagram	Type of Class Diagram	Description
CD.FD	Forward Design	Displays the forward design
CD.25	25% condensation of the RE-CD	Displays only 25% of classes in the class diagram and excludes the other classes.
CD.50	50% condensation of the RE-CD	Displays only 50% of classes in the class diagram and excludes the other classes.
CD.75	75% condensation of the RE-CD	Displays only 75% classes of classes in the class diagram and excludes the other classes.
CD.RE	RE-CD	Shows 100% of all reverse engineered classes in the class diagram

Part B: Choices of Class Diagram

This part aims at discovering the level of understanding of class diagrams generated by the SAAbs framework. We used an ATM simulation system [28] as a sample case. We describe the ATM system requirement at the beginning of this question. Then, we provide five class diagrams in which the respondents are obliged to rate all presented class diagrams in term of design comprehension. Detailed information about the class diagrams is shown in Table 10.1.

The ATM simulation project in [28] provides a complete UML diagrams. We use the forward design of this system (CD.FD) and we reverse engineer the ATM simulation system source code into a class diagram to produce the CD.RE. In order to produce the condensation of the RE-CDs (CD.25, CD.50 and CD.75), the following information is used:

- the RE-CD (in XMI) as the diagram to be condensed.
- the analysis phase class diagram as the input for the important classes candidate.

We load all information into the SAAbs tool to generate the ranking of the important classes in the class diagram and also reconstruct CD.25, CD.50 and CD.75.

In this study, the respondents are required to give opinions on their understandability (based on a 6-point scale) of all provided class diagrams (see Table 10.1). They also need to choose the preferred class diagram for program comprehension and give their suggestion(s) to improve the presented class diagrams. This part is formulated to answer RQ1 and RQ2.

Part C: Software Architecture Abstraction (SAAbs) Framework

This part aims at verifying the condensation result from the SAAbs framework. We want to investigate whether the condensed class diagram generated by the SAAbs framework represents the important information out of the RE-CD; and investigate whether this information (important classes) is useful to understand the system.

A Pacman game [44] was used as a case study of this part. For this question, we provide: a) the explanation of the game at the beginning of this question, b) A Pacman game RE-CD, and c) A 50% condensation of the Pacman game RE-CD. The Pacman's game RE-CD was constructed by using the latest version of the source code (version 4). Then, a 50% condensation of the RE-CD was created by using the SAAbs tool. The 50% of condensation is chosen to represent the condensation of the RE-CD because the number of classes that appear is not too small and not too large. We condense this diagram by using the first version (release) of forward design of this project as candidates of important classes.

The respondents are asked to study the class diagrams and give their judgment in terms of the usefulness of condensed RE-CDs in understanding the system. This part is formulated to answer RQ3.

Part D: The SAAbs Tool

This part aims at validating the SAAbs tool. We asked the respondents to give their opinion on the tool in assisting them for software comprehension. Respondents were also asked to indicate the best feature of the tool and suggestions for improvement and enhancement of the tool. This part will answer RQ4.

10.3.2 Experiment Description

In this subsection, we clarify how the experiment is conducted. The flow of this experiment is shown in Figure 10.1. The respondents were selected from different types of background, i.e. students, academic researchers and IT professionals. Depending on the respondent's background, we asked the respondents to answer different questions. For students, they are obliged to answer Part A, B and C, and graduate students (academic researcher, IT professional) need to answer all questions. The respondents are divided into these groups because we are focused on the experienced respondents in the software implementation in evaluating the tool in Part D.

The answers were written on the prepared response sheets. The respondents may answer the questions in Part A, B and C directly. However, before answering Part D, the respondents are given a "live" tool demonstration (individual or in a small group). The respondents are offered to use the tool or to load their own input to the tool after the presentation. We also conducted an informal question and answer session to make sure that the respondents have a detailed understanding of the tool. Then, the respondents are allowed to answer Part D.

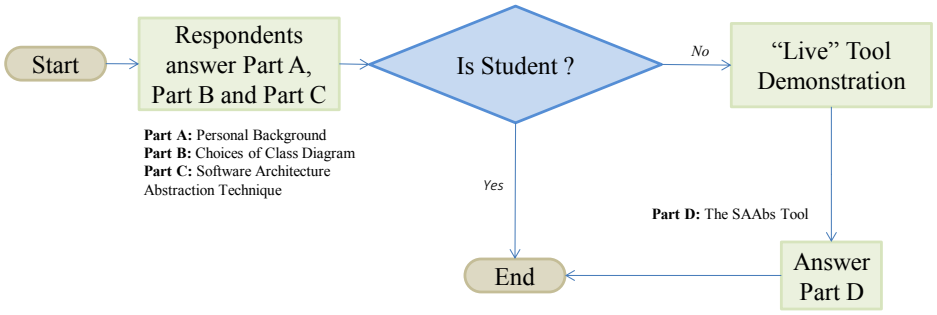
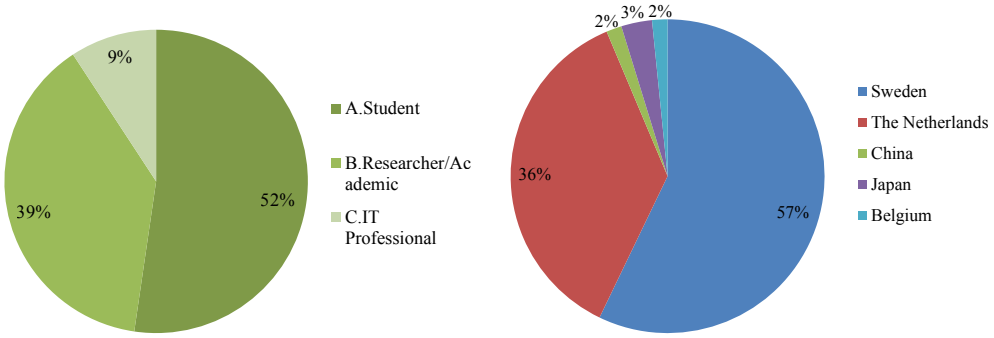


Figure 10.1: *Flow of the Experiment*



(a) Respondents' Role (b) Respondents' Location

Figure 10.2: *Distribution of the Respondents*

10.4 Results

In total, we received 65 responses to this survey. However, due to incomplete of answers, we only count 63 respondents. The answered questionnaires can be found at [121]. The respondents are coming from three different roles which are i) Students (52%), ii) Academic Researcher (39%) and iii) IT professional (9%). The majority of the respondents are from Sweden and the Netherlands, and other locations are China, Japan and Belgium. The distribution of the respondents' role is illustrated in Figure 10.2. In terms of the respondents' experience in class diagrams, 27% of the respondents have < 1 year of experience while another 36% of the respondents have in between 1 to 3 years (see Figure 10.3). 35% of the respondents have more than 3 years of experience in class diagrams. In Figure 10.4, we show the skill of the respondents in understanding the class diagram. This figure demonstrates that 76% of the respondents have an *average* and above skill in understanding class diagrams while in total 22% of the respondents have *low* or *poor* skill in understanding class diagrams.

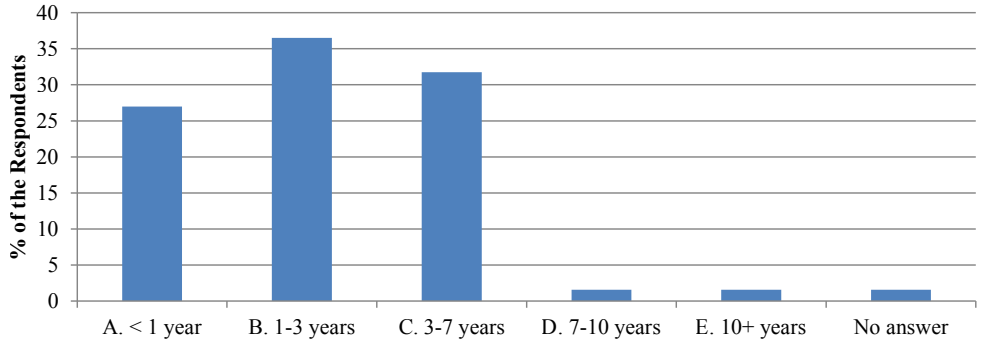


Figure 10.3: *Respondents' Experience with Class Diagram*

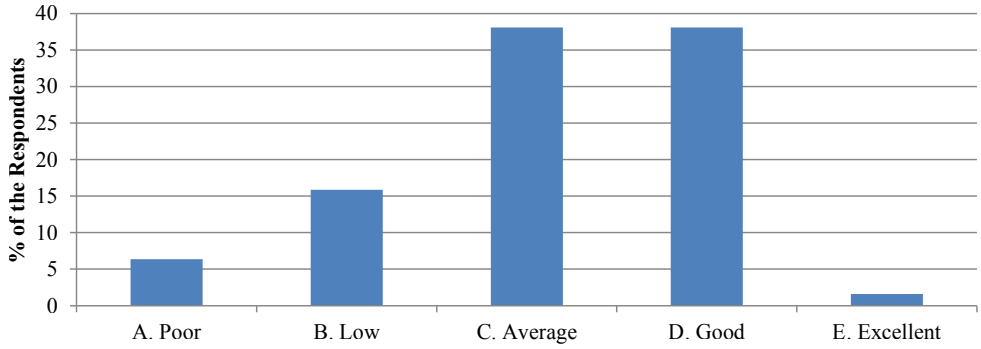


Figure 10.4: *Skills of Understanding Class Diagram*

In terms of the usage of class diagrams, 14% of the respondents *do not use* class diagrams at all for software comprehension. 33% of the respondents use class diagrams *if required*, and 43% of the respondents sometimes use class diagrams. Only 10% of the respondents frequently use class diagrams for software comprehension.

In terms of RE-CD, the majority of the respondents (70%) has *low* or *poor* experience in RE-CDs, while 30% indicated that they have an *average* and *good* experience in reverse engineering source code into class diagrams. The remaining results are presented based on the research questions mentioned in Section 10.2.

10.4.1 RQ1: The Understandability of Condensed Class Diagrams

Figure 10.5 shows the overall results of the understandability of condensed class diagram. The detail information on CD.FD, CD.25, CD.50, CD.75 and CD.RE is explained in Table 10.1. We found that one of the condensed RE-CDs generated using our approach is more understandable than the FD. The CD.25 rated on average 4.58 points (out of 6) compared to CD.FD that is rated on average 4.08 points. CD.RE has been

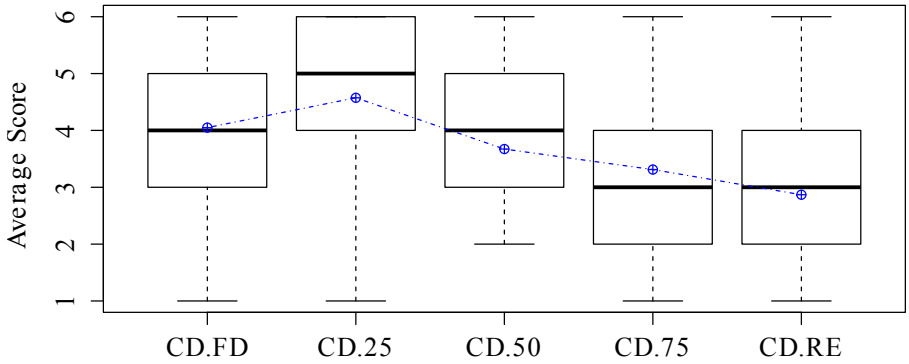


Figure 10.5: *The Understandability of Condensed Class Diagram*

rated 2.87 points on average that makes this diagram has the lowest rating from the respondents. This result shows that condensed RE-CDs scored better ratings than the RE-CD. On the other hand, the line chart (in Figure 10.5) indicates the understandability rating decreases when the number of classes increases. Therefore, we believe that the number of classes influences the respondents' judgment. In the next subsections, we further investigate the rating results by relating the respondents' background and their average rating.

Respondents' Role vs. Understandability Score

Figure 10.6 illustrates the average rating for each class diagram based on the respondents' role. The overall result shows that the rating of CD.25 is higher than the CD.FD. However, referring to Figure 10.6, the IT professional prefer the forward design (CD.FD) over the CD.25. It is quite clear that the rating of the class diagram is decreasing according to the number of classes for the academic researcher group. However, the average rating for students for CD.50 and CD.75 is almost equal.

Respondents' Experience vs. Understandability Score

From the perspective of the respondents' experience, the results show that respondents that have 3 - 7 years experience in class diagrams prefer forward design (CD.FD) compared to condensed class diagrams (CD.25, CD.50, CD.75) and the CD.RE. The results (see Figure 10.7) also show that the average rating for CD.50 and CD.75 is almost equal for the respondents that have > 1 year experience in class diagrams.

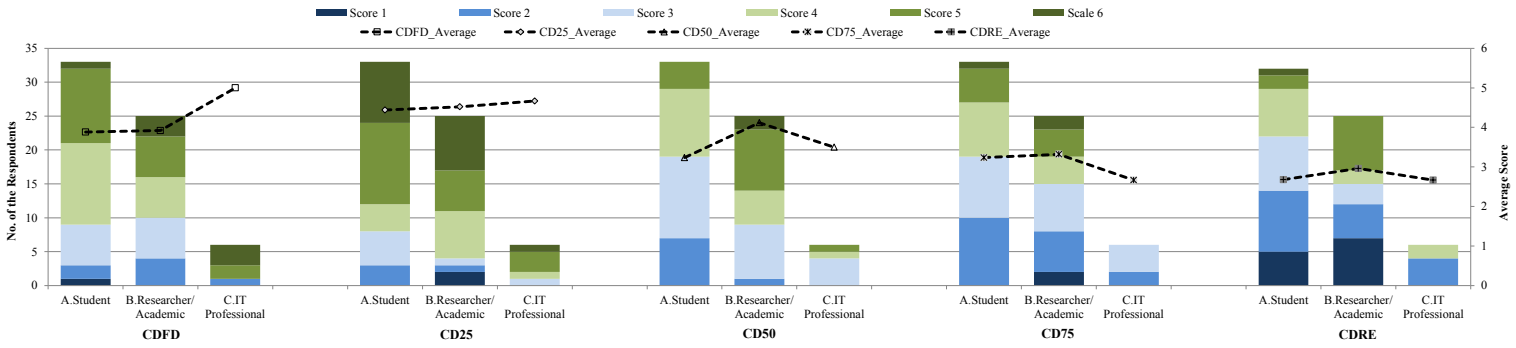


Figure 10.6: *The Respondents' Role vs. Understandability Score*

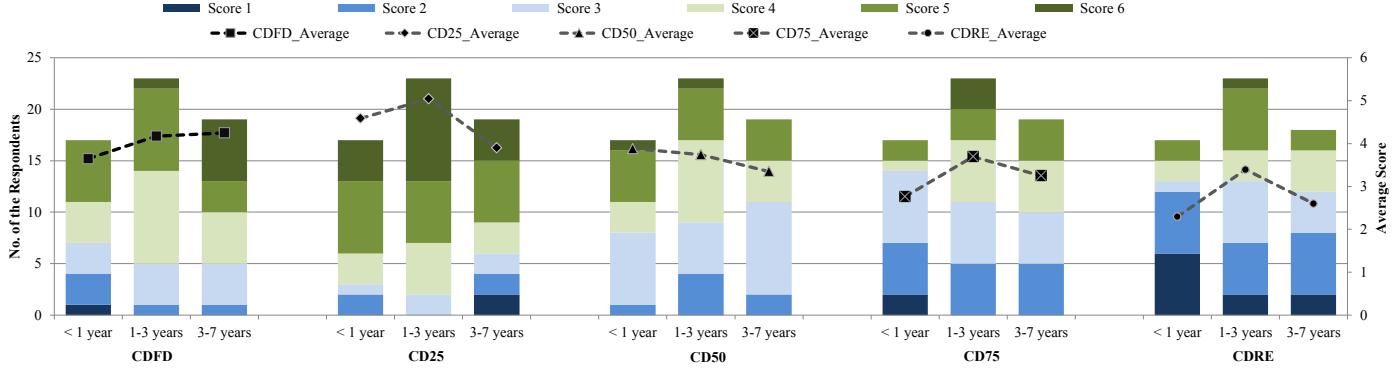


Figure 10.7: *The Respondents' Experience vs. Understandability Score*

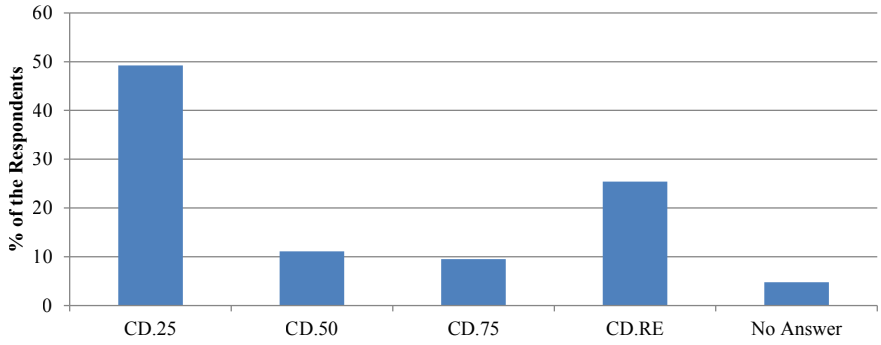


Figure 10.8: *Choices of Class Diagrams*

10.4.2 RQ2: Choices of Class Diagram

In the previous question, we assess the understandability of the condensed class diagrams. In contrast, this question is intended to compare the respondents' preference between the condensed RE-CDs (CD.25, CD.50, CD.75) and the RE-CD (CD.RE). We want to know which class diagram is preferred to be used for understanding the system. The forward design is not offered as an option because this diagram is most likely will be chosen by the respondents as the preferred class diagram. The respondents were asked to choose the class diagram that they preferred for software comprehension. Therefore, the respondents may only choose those between 4 diagrams and they also need to explain their motivation behind their choice. The overall results of the respondent's choices are demonstrated in Figure 10.8.

Based on the responses in RQ1, we expected that the number of classes presented in the question will play a significant role in choosing the class diagram. However, it is interesting to see that the number of classes does not seem to influence the decision in choosing a class diagram. The results demonstrated that CD.25 (the smallest amount of classes) and CD.RE (the highest amount of classes) are two main class diagrams chosen by the respondents. Figure 10.8 shows that almost half (49%) of the respondents chose CD.25 as their preferred class diagram for viewing the system. In total, 31 of the respondents prefer the class diagram in CD.25 for system understanding. 42% of these respondents prefer this class diagram because it is small, simple and easy to look into. 32% of these respondents mentioned that this class diagram showed an appropriate level of condensation of the class diagram that show important classes in the class diagram.

For the CD.RE, 25% of the respondents prefer this diagram for system understanding. Completeness is the main reason for choosing this class diagram. 63% of the respondents that prefer this diagram mentioned that this class diagram is complete and shows detailed information. 19% of these respondents stated that this class diagram is compact, concise and comprehensive. Even though only 6 respondents (11%) chose the

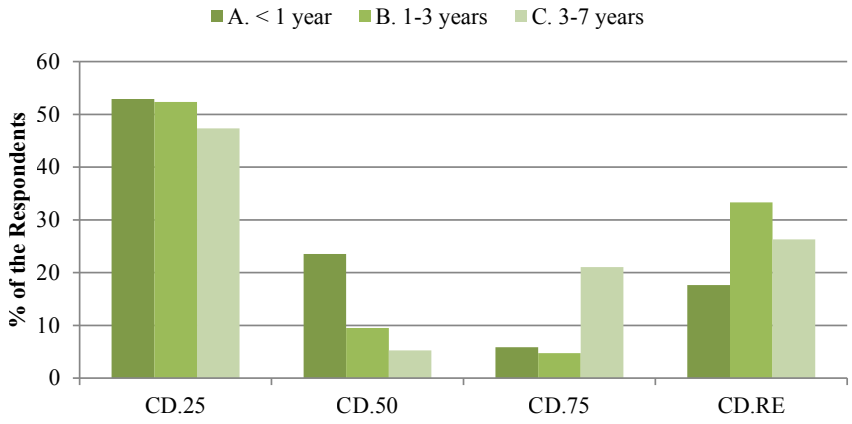


Figure 10.9: *The Respondents' Experience vs. Choice of Class Diagram*

CD.50, 85% of these respondents indicate that this class diagram has the appropriate level of condensation that helps the overview of the system.

Respondents' Experience vs. Choices of Class Diagram

From the perspective of respondents' experience, the results in Figure 10.9 show that from 25% of condensation to 50% of condensation the percentage of the respondents that chosen these diagrams is decreasing and start increasing from CD.75 to CD.RE. Figure 10.9 also suggests that the more experienced respondents chose class diagram with a higher amount of classes.

10.4.3 RQ3: Software Architecture Abstractor Framework

In part C, we asked the respondents to compare the RE-CD and the condensed RE-CD to know how useful this diagram (generated by the SAAbs tool) in representing the important information in the RE-CD. Out of 63 respondents, 5 respondents did not answer this question. Figure 10.10 shows the results of this part. On average, the respondents have given a rate of 4.72 on a scale of 1 to 6. This indicates the 50% condensation of RE-CD constructed by using this framework is useful for software comprehension. 67% of the respondents rated 5 and above while 9% of the respondents rated below 4.

10.4.4 RQ4: Usefulness of the SAAbs Tool

In total, 29 respondents answered this question (only academic researcher and IT professional). As can be seen in Figure 10.11, 93% of the respondents have given the

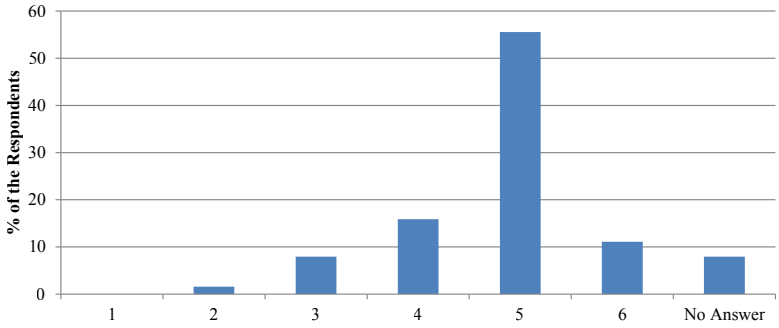


Figure 10.10: *Level of Abstraction of RE-CD for Software Comprehension*

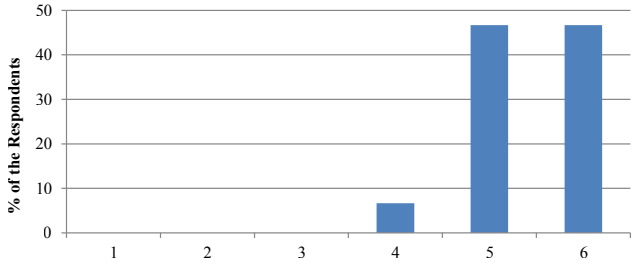


Figure 10.11: *The Rating of the Usefulness of SAAbs Tool*

rate of 5 and above while 7% respondents have given the rate below 5. On average, the respondents give 5.40 out of 6 points. This suggests that the SAAbs tool may be useful for understanding a system.

The respondents were also asked to answer the reason for the score given to the tool. As a result, 50% of the respondents mentioned that the tool is able to show the multiple levels of condensation. 40% of the respondents mentioned the ability of the tool to show the important classes (e.g. highlighting with color) and 40% of the respondents mentioned that this tool is helpful for showing an overview of systems. The remainder of this section describes the results in detail.

SAAbs Features vs Respondents' Background

We further analyze the relationship between the choices of the best features and the respondent's background information. The respondent's background covers the respondent's role, skill and experience in using class diagrams. The respondents' experience only includes group < 1 year, 1-3 years and 3-7 years of experience because the majority of the respondents (97%) is coming from those groups. The reasons mentioned by the respondents are analyzed by capturing the keywords of their answers. Then, we group these keywords to the related features. Overall, four main features which are mostly mentioned by the respondents: (1) Multilevel Class Diagram Abstraction, (2)

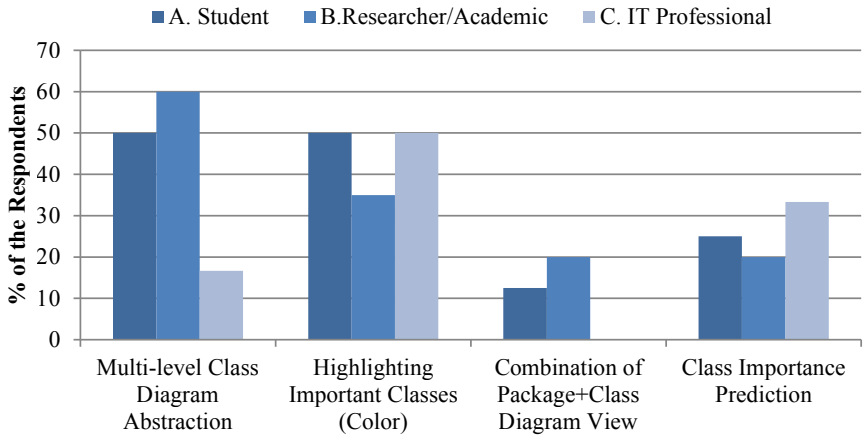


Figure 10.12: *The Respondent's Role vs the Tool's Features*

Highlighting Important Classes, (3) Combination of Package and Class Diagram View and (4) Class Importance Prediction. Hence, we only analyze those four mostly chosen features.

Role vs. Features
As can be seen in Figure 10.12, the academic researcher group prefers the Multi-level Class Diagram condensation over Highlighting (Coloring) important classes. In contrast, the IT Professionals like the Highlighting of Important Classes more than the Multi-level Class Diagram Abstraction.

Experience vs. Features
Figure 10.13 shows the relationship between the respondent's role and their choices of most valued features. It is interesting to see that the preference of Multi-level Class Diagram Abstraction decreases when the years of experience in using the class diagrams is increased. This may indicate that the more experienced respondents are, the less likely they want to simplify the diagram. On the other hand, the preference for Highlighting Important classes increases with the years of experience in using class diagrams. This result suggests that highlighting important classes is preferred when a software developer becomes more experienced. However, both features are of equal interest to (60%) respondents that have 1-3 years of experience. Figure 10.13 demonstrates that the respondents are slightly more interested in the Combination of Package+Class Diagram view as the years of experience are higher.

Skill vs. Features
Figure 10.14 provides the overall result of the relationship between skill in understanding class diagram and the tool's features. There is not much different for the

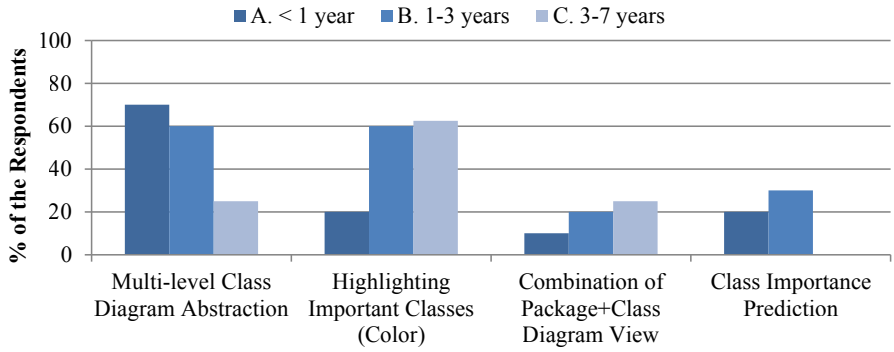


Figure 10.13: *The Tool's Features vs the Respondent's Experience*

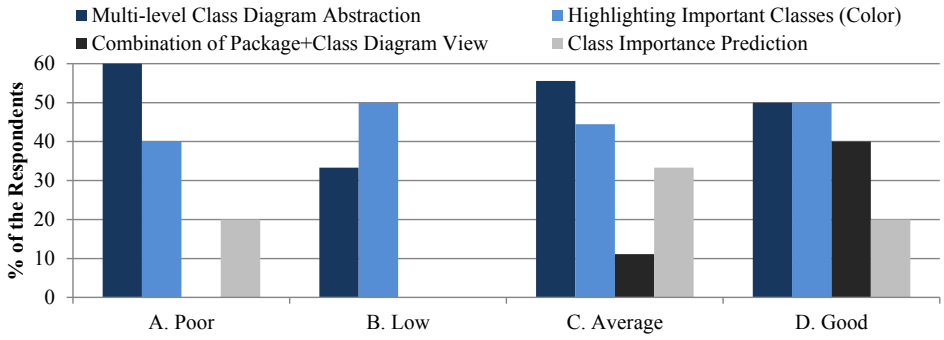


Figure 10.14: *The Respondent's Skill vs the Tool's Features*

Multi-level Class Diagram Abstraction and Highlighting Class Diagram features for all levels of skill. However, there is a small percentage that shows that the Combination of Package+Class Diagram View is more attractive for the respondents that have the skill of *average* and *good* in understanding class diagrams.

Limitations and Improvements

To improve this tool, we asked the respondents about: (1) the weakness of the tool and (2) the improvement needed for this tool. In general, both results shows that the enhancements or improvements needed for this tool mainly referred to: (1) Important Classes Prediction, (2) Layout and (3) Additional Tool's Features. Figure 10.15 shows the results on the limitations of the tool.

For Important class prediction, the respondents show their concern about the prediction result. They are concerned about the suitability of the machine learning techniques in classifying the important classes of "big systems" and also the validation of the result (human validation of the important classes prediction of the tool). In terms of layout, the respondents suggested that the tool should give more options to the

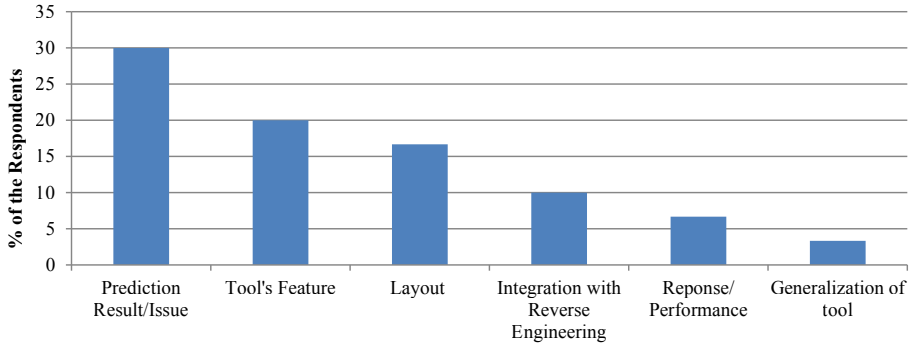


Figure 10.15: *The Limitations of the SAAbs Framework and Tool*

tool's user to modify the layout. The modification includes: editing class diagrams, change of color, various types of class diagram layout (e.g. centering, hierarchy) and highlighting of private and public attributes.

10.5 Discussion

In this section, we discuss the scoring of class diagrams, the limitation of SAAbs tool and the threats to validity.

10.5.1 Choosing a Class Diagram

At the beginning of our analysis, we expected that the respondents preferred class diagrams that have a low number of classes. However, our expectation applied for CD.25 but the other preferred class diagram is CD.RE. This result shows that the number of classes is not the main reason for choosing the class diagram. Also, the representation of the class diagram also influences the decision of choosing the best class diagram for system understanding.

According to the reasons of choosing a class diagram, the respondents indicate that the chosen class diagram is easy to understand. Even though CD.RE shows the highest number of classes, the respondents mentioned that this class diagram is understandable. One of the reasons is that the class diagram presents the separation of features where the two major classes are presented in the middle of the class diagram. Those classes represent two independent groups of functionalities that are ATM related system and the simulation/graphical user interface (GUI) related purposes (see questionnaire in [121] for more detail). This suggests that class diagrams with a high number of classes are practical for software comprehension provided that the layout shows suitable clustering or grouping of functionality (as suggested by the software developers in Chapter 5).

10.5.2 Limitation of SAAbs

As described in the previous sections, we proposed the SAAbs framework and tool to assist the software developer for the software comprehension assignment. However, several limitations have been identified before and after the validation experiment was conducted.

For important class prediction, object-oriented design and text metrics are used as predictors. As mentioned in [127], combining those metrics results in a better prediction than using only one particular set of metrics for important class prediction. However, the text metrics are limited to systems that are developed using the English language (for the class element name); Thus, if a system is developed based on another language, these predictors need to be adjusted. This problem due to the stemming algorithm that is used to formulate the text metrics.

Although the object-oriented design metrics have reasonable predictive power, we are considering other features to enhance the prediction results. For instance, the use of network metrics as described by Thung et al. [164].

Another concern about the framework component is the usage of "Show Suggestion" to suggest the candidates for important classes. This suggestion may not be correct because it is based on the amount of coupling and number of operations. It may include several lookup/descriptor classes (like a lookup table) that have a lot of relationships, but these classes are not frequently considered as important classes. In the SAAbs tool, the selected important class candidates are shown in the class list. Users can select the lookup/descriptor classes and other insignificant classes to be excluded from the important class candidates.

10.5.3 Threats to Validity

In this subsection, we discuss the internal and external threats to validity of the validation experiments.

Internal Validity: The selection of case studies may influence the result of this experiment. The respondents need to comprehend a system in a limited time. We believe that we have minimized this threat as we used a small to medium size of class diagrams, well-known projects domain, and the respondents were given a briefing and comprehensive description of the projects.

In this experiment, we tried to discover the suitable features of the tool in regard to the respondents' background. We asked the respondents to indicate the best feature of the tool. We realized that the term "best feature" in the question might not precisely indicates suitable feature. Nonetheless, we believe that the question yields the interest of the respondent on features that are considered suitable features for a particular respondent.

External Validity: The distribution of the respondent's role could be a threat to external validity. The role of respondents is not equally distributed where the majority of the respondents are students. However, we have shown that the distribution of the students and graduate students (academic researcher and IT professional) is almost equal. Although we do not deny that the experience in industry and academia are different, in terms of class diagram understanding, we believe that the difference is small.

10.6 Conclusion and Future Work

This experiment aims at validating the SAAbs framework and tool in assisting the software developer in understanding software systems. We showed that the condensed RE-CDs produced by the SAAbs tool (particularly 25%, 50% and 75% of condensation) are understandable. Furthermore, the respondents considered that the tool that realized the SAAbs framework is useful in aiding the software developer to understand software.

From the results, it is interesting to see that the main class diagram (that is chosen for system overview) is the 25% condensation of RE-CD and the RE-CD (full RE-CD). This result shows that the layout influences the selection of a class diagram for viewing the system rather than the number of classes in class diagrams. The respondents indicate that the condensed RE-CD is also useful for understanding the system. On average, the respondents give 4.72 out of 6 points that indicate the condensed RE-CD is useful for software comprehension.

The SAAbs tool was developed to realize the SAAbs framework in providing a platform for assisting software comprehension. In this experiment, we assess the respondent's judgment on the usefulness of this tool. On average, the respondents gave 5.4 out of 6 points that indicate this tool is helpful during the software comprehension. The result also demonstrated that the respondents that have more experience intend to view all classes in the class diagram and highlight the important classes. In contrast, the less experienced respondents prefer to limit the classes in the class diagram by reducing the class diagram for software comprehension task.

This validation experiment shows an interesting and promising result. There are several ways to enhance and strengthen this validation. We are looking forward to improving the validation by:

- A user study on the usage of the SAAbs tool in the real software comprehension task (task-oriented validation), and
- Online user validation (i.e. publish the tool online and assess the user experience about the tool).