System Development: What, Why, When and How CASE Tools Should Support Novice Software Engineers

Shahida Sulaiman, Ahamad Tajudin Khader, Zurinahni Zainol
School of Computer Sciences
Universiti Sains Malaysia
11800 USM, Penang
{shahida,tajudin,zuri}@cs.usm.my

Sarina Sulaiman
Faculty of Computer Science & Information Systems
Universiti Teknologi Malaysia
81310 Skudai
Johor
sarina@utm.my

ABSTRACT

Novice software engineers particularly computer science students need to be trained with theoretical knowledge and practical skills in system developments. The knowledge and skills may encompass the activities involve in all phases of system development including analysis, design, coding, testing and maintenance. Computer Aided Software Engineering (CASE) tools can be introduced to novices as a supporting element to understand software engineering aspects and principles, which are supposed to enhance the activities. Thus, this paper will discuss problems faced by novice software engineers mainly computer science students during the process of gaining the theoretical knowledge and practicing it while developing software systems for their projects. The analysis of the study will be based on four types of elements, which are characteristics, behavior, belief and attitude. The findings will highlight what, why, when and how CASE tools should support novice software engineers from analysis to implementation phases of system development compared to that of expert software engineers.

Keywords

System development, computer aided software engineering (CASE) tools, novice software engineers.

1. INTRODUCTION

In order to produce competitive computer science graduates particularly those specialized in software engineering, higher learning institutions should have a curriculum that has the balance between theories delivered via lectures and practical aspects trained during lab hours. In addition, computer science students will gain more exposure of practical aspects when they undergo industrial training for three to five months. For hands-on training, our principals generally attempt to avoid teaching students who are novice software engineers to know specific technology but to equip them with the knowledge of problem solving. For instance in a database class, we do not want to train them how to model a database using the computer aided software engineering (CASE) tool such as Rational Rose [3] but how to solve the problems of modeling a database system using the current modeling notation such as Unified Modeling Language (UML).

A number of CASE tools are available that can support the activities in system developments. They are mostly useful for large software projects but they might be quite complex to be used by novices such as computer science students. There is a challenge to educate and ensure novice software engineers equipped with both theoretical and practical knowledge in implementing best practices in system development by using proper CASE tools.

Nevertheless we believe the introduction of an appropriate CASE tools to novice software engineers could support their understanding in theoretical aspects of system development life cycle (SDLC) that includes the common phases: planning, analysis, design, coding, testing and maintenance.

The work of van Vliet [7] classifies CASE products according to the parts of the lifecycle they support: CASE tools – support one task (e.g. programming using Visual Basic), CASE workbench – limited set of activities (e.g. analysis and design using Rational Rose [3], CASE environment – entire software process (e.g. Rational Suite [3]).

Section 2 will discuss some related work in CASE tools that motivates this study. The survey and its questionnaire are described in Section 3. The analysis and findings of the survey are illustrated and explained in Section 4 and 5 respectively. Finally we conclude the work and outline some possible future work in Section 6.

2. RELATED WORK

There are a lot of studies regarding CASE tools from various perspectives such as in the work of Sulaiman et al. [4] and Finnigan et al. [1]. The study conducted by Sulaiman et al. focuses on the use of CASE tools to support system documentation. In contrast, the research by Finnigan et al. [1] investigated factors affecting the acceptance of CASE tools within New Zealand. The questionnaire results suggested that there was a poor uptake of CASE within New Zealand. The two main reasons were high cost and lack of education regarding CASE tools.

Some studies explicitly focus on the approaches to support novice software engineers. For instance in the early work of Honda et al. [8] the novices are supported by a training program that assist them in building client-server systems in three phases: understanding, programming and debugging. The work shows that

3rd Malaysian Software Engineering Conference '07, Dec 3-4, 2007, Selangor, Malaysia.
 ISBN 978-967-5026-23-2

by having a proper training mechanism, novice software engineers can be trained theoretically and practically to develop required systems efficiently. This study highlights the other perspective in supporting novices besides CASE tools. On the other hand Werth [9] highlights the importance of teaching and applying software process concepts in the beginning of an undergraduate project. The work concludes that the empowerment, shared learning and more stable environment promote better understanding and learning of software process improvement among novices. More recent work of Reza and Grant [10] proposed a method that supports novice software engineers in selecting the most suitable software architecture for the system being developed. The method is based on the universally accepted design principles and tactics to satisfy non-functional requirements of the concerned system. In this work no specific CASE tools or workbenches were proposed to support the method but the researchers had planned to work on the tool to recommend architectural styles based on non-functional requirements.

As we discussed earlier in the introduction, CASE products of type environment provide a comprehensive system development environment for software engineers or computer science students. Such tools aid them to draw diagrams during analysis and design stage and then transform the design into corresponding source codes. Rational Suite [3] is a ubiquitous commercial tool that can assist both forward and reverse engineering so-called roundtrip engineering. The suite can support the whole phases of SDLC. However the tool is very expensive causing the academic institutions or small software departments opt to employ open source tools available or do not even use such tools at all. In this case novice software engineers will use drawing tool to design their software and then transform manually into source codes using any available Integrated Development Environments (IDEs) for the software languages they use. For instance they might draw a class diagram using Microsoft Word and then implement the coding using Borland JBuilder to develop a Java-based software system.

Besides, commercial tools like Rational Rose [3] are quite complicated to be used by novice software engineers particularly students. Such tools are more appropriate to be used by experienced or expert software engineers if they are fully equipped with best practices in software engineering in particular software design and coding. Users of the tools will be able to generate diagrams such as use case diagrams for analysis and then create the respective sequence diagrams during the design stage. The tool allows generation of source codes' skeleton from the sequence diagram. However the tool does not check whether the diagrams in both analysis and design stage are correct or conform to software engineering discipline or best practices. They do not provide direct guidance or tutoring tool to suggest to novice software engineers or computer science students how to analyze, design and write source codes conform to the best practices in order to produce high quality software as what they learn theoretically.

Thus, the gaps exist between CASE tools for expert and novice software engineers have motivated us to conduct this survey to gauge the expectations of novice software engineers towards CASE tools in the aspects of characteristic, behavior, attitude and belief.

3. THE SURVEY

The survey was conducted among 171 Bachelor of Computer Science (Hons.) final year students at the School of Computer Sciences, USM, Penang, Malaysia. They had studied the courses needed in system development such as Programming, Data Structure, Database Design, System Analysis and Design, and Software Engineering. They also had undergone industrial trainings for five months. Hence, such criteria could fulfill the objectives of the survey that are:

(i) To study what are the level of understanding and skills in software engineering among the respondents.
(ii) To investigate why the required knowledge and skills required are difficult to be captured by novice software engineers.
(iii) To determine when is the best time to introduce Computer-Aided Software Engineering (CASE) tools to novice software engineers.
(iv) To indicate how CASE tools will be able to support novice software engineers.

The questionnaire as shown in Table1 was developed using four elements that are characteristics, behavior, belief and attitudes [2]. The columns show the value of each code (CD) given to the data, its description, and question number (Q#).

Table 1. List of questions

CD	Description	Q#
Characteristic		
Cr1	Highest academic level.	A1
Cr2	Level of software engineering expertise.	A2
Cr3	Types of software projects ever involved.	A3
Cr4	Frequency of projects that follow software engineering disciplines.	A4
Cr5	Frequency of CASE tools used in projects conducted.	A5
Cr6	Types of CASE tools ever used.	A6
Behavior		
Bv1	Level of understanding in software engineering.	B1
Bv2	Level of skills in software engineering.	B2
Bv3	Difficulties in applying theories in project conducted.	B3
Belief		
Bf1	How CASE tools will help to understand and practice software engineering better	C1
Bf2	When novices think CASE tools should be introduced	C2
Attitude		
At1	Features of CASE tools should be provided to novices	C3

4. THE ANALYSIS

The analysis will be based on the four categories of data: characteristic, behavior, beliefs and attitude as in Table 1. The quantitative data is analyzed using Statistical Package Software System (SPSS) 9.0. The questions without any response were considered missing.

4.1 Characteristic

All the respondents were final year students who had started their final year projects after undergoing five months industrial attachments. Two of the students had their Diploma in Information Technology before joining the BSc (Hons.) programme (refer Cr1 in Table 1). For Cr2: level of software engineering expertise, 108 (63.5%) of the students described themselves as 'novice/beginners' while 62 (36.5%) students stated 'intermediate'. One student did not respond to the question. None were 'expert'.

Regarding the types of projects they ever involved during the study and industrial attachments are revealed in Table 2. Besides undergraduate projects for final year students, quite a number of them have or are involved in industrial project for instance software to support manufacturing process (35), research project (32), organizational project for example information system to support businesses or services (31), commercial (8) and others (2).

Table 2. Types of projects (Cr3)

Types of Project	Sum
Undergraduate	158
Research	32
Commercial	8
Industrial	35
Organizational	31
Others	2

Table 3 shows that most of the respondents (126 or 73.7%) "sometimes" follow the software engineering disciplines they have learned. Thirty of them (17.5%) "always" follow while fourteen (8.2%) "never" follow the disciplines learned. There was one missing value for this question.

Table 3: Frequency of projects following software engineering disciplines (Cr4)

	Frequency	Percent	Valid Percent	Cumulative Percent
Never	14	8.2	8.2	8.2
Sometimes	126	73.7	74.1	82.4
Always	30	17.5	17.6	100.0
Total	170	99.4	100.0	

Table 4 shows the frequency of projects that use CASE tools (Cr5). Most of them (124 or 72.5%) "sometimes" used CASE tools. In addition 35 (20.5%) "always" and 12 (7%) "never" used CASE tools.

Table 4: Frequency of CASE tools used (Cr5)

	Frequency	Percent	Valid Percent	Cumulative Percent
Never	12	7.0	7.0	7.0
Sometimes	124	72.5	72.5	79.5
Always	35	20.5	20.5	100.0
Total	171	100.0	100.0	

The most common type of CASE tool used is IDE (86), followed by visual modeling (43), and round-trip engineering tool such as Rational Rose (7). Surprisingly 42 students were not sure whether they ever used CASE tools and 11 students stated as never used any (Refer Table 5).

Table 5: Types of CASE tools used (Cr6)

Types of CASE tools	Sum
Visual modeling	43
Integrated Development Environment (IDE)	86
Round-trip engineering	7
None	11
Not sure	42
Others	6

4.2 Behavior

The behavior element (Bv1, Bv2, Bv3) of the survey (see Table 1) attempted to identify the knowledge and skills for software development among the students. Both theoretical knowledge and practical skills were rated using Likert scales: 1:Very low, 2:Low, 3:Normal, 4:High, 5:Very High.

The null hypothesis that should be rejected is stated as "There is no significance difference of means for each pair of theoretical knowledge (T) and practical skills (P)" that is H_o: $\mu_T = \mu_P$. We believe some pairs would reject the null hypothesis because the students might have good theoretical knowledge but not practical skills.

The results in Table 6 reflect that there are significant mean differences ($\mu_T \neq \mu_P$) between the seven pairs of software engineering body of knowledge for both theoretical and practical skills (less than 0.025): software design (0.006), software construction/coding (0), software testing (0.006), software configuration management (0.009), software engineering management (0.005), software engineering tools and methods (0.018), and software quality (0.003).

Table 6: Paired-sample test for Bv1 and Bv2

Theoretical vs. Practical Skills	95% Confidence Interval of the Difference		t	Sig. (2-tailed)
	Lower	Upper		
Software requirements/analysis	-1.3745E-02	8.392E-02	1.418	.158
Software design	2.415E-02	.1396	2.800	**.006**
Software construction/coding	9.682E-02	.2307	4.830	**.000**
Software testing	2.415E-02	.1396	2.800	**.006**
Software maintenance	-6.5096E-02	.1002	.419	.676
Software configuration management	2.255E-02	.1529	2.657	**.009**
Software engineering management	3.062E-02	.1682	2.853	**.005**
Software engineering process	-8.0214E-02	8.021E-02	.000	1.000
Software engineering tools and methods	1.633E-02	.1708	2.391	**.018**
Software quality	4.489E-02	.2241	2.963	**.003**

Table 7 shows the descriptive statistics of mean values of the Likert scales (1:Strongly disagree, 2:Disgaree, 3:Normal, 4:Agree, 5:Strongly agree) and the respective standard deviations for Bv3 that is the difficulties in applying theories in the practical aspects during software development or maintenance. All the difficulties except for 'others' range between the scales of 2 (disagree) to 4 (agree), which are mostly almost to "normal" scale.

Table 7: Mean values for Bv3

Difficulties in applying theories in practical aspects	Mean	Std. Deviation
Not interested to follow	2.7251	2.5019
Cannot see the importance	2.6316	.9387
Too many diagrams	**3.1520**	.8333
Cannot see the link of diagrams	2.7368	.9241
Theories learned are not enough	**3.0643**	.9831
Theories learned are difficult	**3.0585**	.9121
CASE tools are not available	2.6959	.9707
CASE tools are complex to use	**3.0117**	.9330
No guidance from CASE tools	**3.0117**	1.0232
No guidance from instructors	**3.0936**	1.0304
Books have lack of guidance	2.9766	.9201
Books are difficult to understand	2.9123	1.0563
Others	.1287	.6286

4.3 Attitude

The analysis is illustrated as in Table 8. Likert scales (1:Strongly disagree, 2:Disgaree, 3:Normal, 4:Agree, 5:Strongly agree) were used to evaluate the expected features. Most mean values are between the scales 3 (Normal) to 4 (Agree).

Table 8: Expected features of CASE tools for novices (At1)

Features of CASE Tools for Novices	Mean	Std. Deviation
Make software engineering (SE) interesting	3.7041	.7988
Highlight the importance of SE	3.7160	.7574
Explain the use of diagrams	**3.8155**	.6441
Explain the link of diagrams	**3.7988**	.7036
Relate the theories learned	**3.7857**	.7353
Make theories easy to understand	**3.8571**	.7525
Available easily	**3.7784**	.8243
Easy to use	3.7278	.8573
Provide an on-going guidance	**3.7560**	.7777
Guide as instructors	3.6607	.7879
Complement guide from books	3.6450	.7665
Easier to understand compared books	3.6627	.8721
Others	3.4615	.5818

4.4 Belief

Regarding the students' belief towards how CASE tools will aid to understand and practice software engineering better (Bf1), majority of them (132 or 77.2%) stated, "Yes". On the other hand (32 or 18.7%) stated "Not sure". Five students (2.9%) said that it would not help.

In addition, most of them (105 or 61.4%) believed that CASE tools should be introduced while learning the theories followed by 45 (26.3%) of them believed that it should be introduced while doing the practical or software development/maintenance. Besides, eleven students (6.4%) suggested CASE tools to be used after learning the theories. There were eight students (4.7%) who were not sure.

5. THE FINDINGS

From the analysis, we managed to gauge the level of understanding and skills in software engineering among the respondents. They mostly perceived themselves as beginners despite of undergoing industrial attachments for five months and had started their final year projects. This would indirectly answer why most graduates are not ready for industry.

On the other hand, the types of projects they involved during industrial attachments were quite diverse. Despite of the theories learned regarding software engineering disciplines, the students mostly did not always follow the disciplines learned such as using proper models, techniques and tools throughout software development or maintenance phases. The distribution of frequencies for this question is almost the same for the issue regarding the use of CASE tools. The type of CASE tool mostly used was IDE. This is probably because they can write source codes faster using the selected IDEs both for their programming assignments or projects.

The analysis on theoretical versus practical skills shows that there was a significant gap particularly for software construction/coding that yields zero value. This implies that even though the novice software engineers have very high level of theory understanding in how to code programs, they have very low practical skills. Another skill that had significant difference was software quality. This result was probably due to the aspects of quality that was highlighted or taught in the related undergraduate courses but the students might be not very sure how to check the quality of the software written. Despite of taking the course System Analysis and Design, the students seemed to be balance in both theories and practical of analysis aspect but not in design. The same trend was observed in software testing and software engineering management that were not balance in both theoretical and practical skills, which were incorporated in the same course of System Analysis and Design.

The study also managed to investigate why the required knowledge and skills were difficult to be captured by novice software engineers like our students. One of the main reasons included that too many diagrams used in modeling their software projects. Among other reasons include theories learned are not enough and difficult, CASE tools are complex to use and no guidance from instructors. However the scale given were around 3.0 to 3.2, which is between normal to agree. This indirectly reveals that the students were not very sure about the reasons.

The study also managed to derive the expected features of CASE tools. Among the highest scale chosen were between 3.756 to 3.857 that is almost towards the scale of agree (4). The famous features include CASE tools should: explain the use and the link of diagrams, relate the theories learned, make theories easy to

understand, available easily and the tools that provide an on-going guidance. These features reflect how CASE tools will support novices.

It is also notified that CASE tools should be introduced to novices while learning the theories, thus students can directly relate the theories and how the tools will support the activities in their software projects. This could be possible since the students mostly believed that CASE tools would help them to understand and practice software engineering better.

5.1 What, Why, When and How Aspects

From the findings we can summarize what, why, when and how CASE tools "should" support novice software engineers.

(i) **What tools should support:** Both theoretical and practical aspects of software engineering should be supported. By supporting both elements, we anticipate the gap among novices in theoretical knowledge and practical skills will be reduced.

(ii) **Why tools should support:** The difficulties faced by novices in applying theories in system development derived in this study reflect the need of appropriate tools that should support them. Besides such tools will make them appreciating the importance of applying the theories learned in the practical aspects of system development.

(iii) **When tools should support:** From the findings in the study, CASE tools should be introduced while learning the theories. By introducing earlier, students will be able to grasp the idea of CASE tools much better and acknowledge how such tools are useful to support the practical aspects of software engineering.

(iv) **How tools should support:** Provide the features required by novices that can balance between theories and practical aspects of software engineering principles in such tools. Appropriate features should be incorporated in the tools to promote a fast learning curve among novices.

We anticipate the aspects above will provide an insight of the most suitable and effective CASE tools that support an encouraging environment to learn and practice software engineering principles among beginners or novices.

6. CONCLUSION AND FUTURE WORK

The results of the survey imply that proper-designed CASE tools that suit novice software engineers' needs, will probably be able to support the novices to balance both theoretical knowledge and practical skills in software engineering disciplines. The aspects of what, why, when and how tools should support novice software engineers are also discussed.

Our future work will be to realize the aspects of CASE tools derived in this study in order to produce a CASE tool that can meet the requirements of novice software engineers.

7. ACKNOWLEDGMENTS

This research is supported by MOSTI e-Science Fund, grant number: 01-01-05-SF0075 (305/PKOMP/613119).

8. REFERENCES

[1] Finnigan, D., Kemp, E. A. and Mehandjiska, D. Towards an Ideal CASE Tools. In *Proceedings of International Conference on Software Methods and Tools (SMT 2000).* IEEE Computer Society Press, USA, 2000, 189-197.

[2] Kendall, K. E. and Kendall, J. E. *System Analysis and Design – 4th Edition.* Prentice Hall, USA, 1998.

[3] Rational, *IBM Rational Software*, http://www-306.ibm.com/software/rational/, 2007.

[4] Rigi, *Rigi Group Home Page,* http://www.rigi.csc.uvic.ca/, 2007.

[5] Sulaiman, S., Idris, N. B. and Sahibuddin, S. Production and Maintenance of System Documentation: What, Why, When and How Tools Should Support the Practice. In *Proceedings of 9th Asia Pacific Software Engineering Conference (APSEC 2002).* IEEE Computer Society Press, USA, 2002, 558-567.

[6] Sulaiman, S., Idris, N. B., Sahibuddin, S. and Sulaiman, S. Re-documenting, Visualizing and Understanding Software Systems Using DocLike Viewer. In *Proceedings of 10th Asia Pacific Software Engineering Conference.* IEEE Computer Society Press, USA, 2003, 154-163.

[7] van Vliet, H. *Software Engineering Principles and Practice.* John Wiley, England, 2000.

[8] Honda, S., Hiramatsu, A., Yamamatsu, H., Morihisa, H., Ikkai, Y. A. Training Program for Building Client Server Systems and Its Evaluation Using WWW Browser. In *Proceedings of Systems, Man, and Cybernetics.* Vol. 3, 1998, 2866-2871.

[9] Werth, L. H. Software Process Improvement for Student Projects. In *Proceedings of Frontiers in Education Conference.* Vol. 1, 1995, 2b1.1-2b1.4.

[10] Reza, H., Grant, E. Quality-Oriented Software Architecture. In *Proceedings of International Conference on Information Technology: Coding and Computing (ITTC2005).* Vol. 1, 2005, 140-145.