Journal of Computer Science, 9 (1): 16-29, 2013
ISSN 1549-3636

doi:10.3844/jcssp.2013.16.29 Published Online 9 (1) 2013 (http://www.thescipub.com/jcs.toc)

Design of Object-Oriented Debugger Model by Using Unified Modeling Language

Nor Fazlida Mohd Sani, Noor Afiza Mohd Ariffin and Rodziah Atan

Department of Computer Science, Faculty of Computer Science and Information Technology, University Putra Malaysia, Selangor, Malaysia

Received 2012-08-17, Revised 2012-08-28; Accepted 2013-01-22

ABSTRACT

Debugging on computer program is a complex cognitive activity. Although it is complex, it's still one of the popular issues in computer programming task. It is a difficult task, which is to understand what the error is and how to solve such error? In computer programming the difficulty is to understand the Object-Oriented programming concept together with the programming logic. If the programming logic is incorrect, the program codes will have such error named as logic error and can caused highly maintenance cost. Logic error is a bug in a program that causes it to operate incorrectly, without terminating or crashing the program. It will produce unintended output or other behavior than what we are expecting. Method that use to develop a propose model Object Oriented Debugger is Unified Modeling Language (UML). It is the best choice model and suitable to design the Object Oriented Debugger which will be developed in an object oriented programming environment. The model will provide an ability to capture the characteristics of a system by using notations in the process of designing and implementing the system. The model of Object Oriented Debugger has been successfully implemented. This model has been developed using Unified Approach methodology, which consists of two methods such as Object-Oriented Analysis (OOA) and Object-Oriented Design (OOD). The model developed is to capture the structure and behavior of the Object Oriented Debugger by using the UML diagram. The model also can ease the readability of the documentation for the maintenance purposes. The design of the Object Oriented Debugger Model has been developed using the UML notation. It's consisting of two parts that are object-oriented analysis and object-oriented design. All the developing and designing are based on the model in UML.

Keywords: Object-Oriented Analysis and Design, Unified Modeling Language, Debugging Model, Logic Error

1. INTRODUCTION

Debugger is a computer program that is used to reduce the errors in programming code. In development activity, the debugging process is very important. Its offer more sophisticated function such as single stepping, breakpoint and fix the errors depending on the programmer understanding or expertise. For novice programmers, it is a big challenge to understand the meaning of each error in program code especially for logic errors. The same problem also confront by the experienced. The process of finding and fixing the logic errors is more difficult rather than finding and fixing syntax errors. To helps and solve the problem that occurs among the novice programmers this research try to making the debugger that can be more understandable to the novice programmers. So, we present an automated debugger named Object Oriented Debugger, a system for analyzing code written in Java language which can handle the problem of understanding on object oriented programming and debugging program among the novice programmers. It also to determine the difficulty of finding logic error among the beginning programming student by the analyze source code to localize, find logic

Corresponding Author: Nor Fazlida Mohd Sani, Department of Computer Science, Faculty of Computer Science and Information Technology, University Putra Malaysia, Selangor, Malaysia

error and provide more user-friendly error messages when an uncaught exception occurs.

The purpose of this research is to develop the Object Oriented Debugger that has been designed to understand a program written in a structured language, such as Java. This research has been carried out from the CONCEIVER++ (Sani *et al*., 2009) and Adil (Zin *et al*., 2000) system as well as extending the plan formalism to include the logical errors and designed as a new automated debugger which can debug on different style of written programming code. The objective of this study is to capture the structure and behavior of an Object Oriented Debugger based on the diagram UML, also to make documentation more readable.

In this study, we will describe and explain the design of an Object Oriented Debugger Model by using the Unified Modeling Language (UML). The UML has become the standard notation for object-oriented modeling system (Ali, 1999) The UML use the notations to express the design of a software system. The UML process model starts by seeking the requirement and ideas of the system via Use Case diagram, the identification the steps involve in each requirement in Activity diagram and define the external interfaces that need in the system.

1.1. Related Works

1.1.1. Debugging

In computer area debugging is the process of locating and fixing errors in computer program code. When to debug a program, it is start with a problem, isolate the source of the problem and then fix it. Normally, the method that used in debugging process is comparing the program behaviour with the correct behaviour in details. Debugging process is a necessary process in almost any software development process such as commercial product or an enterprise or personal application program. For some complex product, the debugging is done as the result of the unit test for the smallest unit of a system when the product will be out in a real world situation. Debugging process is very important to make sure the system that will be deliver is well function without any errors. It is because most computer programs contain thousands of line of code, almost any new system is likely to contain a few errors.

1.2. Debugging in Programming Education

In the past few years, the object-oriented programming has becoming widely used in the worldwide. This current trend influences the change in programming education. A growing numbers of college and universities have change to object-oriented languages for teaching novice programmers. Impacts from this, many novice programmers are learning an object-oriented language as their first programming language. According to the Wiedenbeck *et al*. (1999) all of these changes make an understanding of novice learning of object-oriented languages is very important. However, in the teaching programming, the teacher needs more emphasis on the comprehending the novice programmers not only in writing programs because many novices considered a programming as a difficult subject.

It is not an easy ways especially to provide a good teaching practice during teaching the programming courses to make the novices to fully understand the programming concepts. An introductory programming course is a basic concept of programming in learning of programming. Most computing students were interested to participant in Computer Science curriculum such as game programming and multimedia application (Pears *et al*., 2007). It is interesting to note that programming support tools such as interactive incremental code execution, visualization, editing and syntax support are some of the solutions, which can use to help the novices to improve skills in programming. Generally, programming tools are developed to meet expert programmer’s needs. The concepts and features that are provided in programming tool has become problematic for novices, also their error and warning message may be hard for novices to understand.

For this reason, there have been efforts to develop programming tools that are especially designed for the novices needs. Besides to help novices, it also can help tutor to reduce or simplify tutoring workload such as automatic assessments and course management. According to Pears *et al*. (2007), research project can act as tools to solve a local problem either in a specific institution or a specific course. There are several excellent examples of tools that have been widely effort on teaching practices such as BlueJ programming environment by Kolling *et al*. (2001) and Course Marker automatic assessment tool and its predecessors by Higgins *et al*. (2005).

Despite the best effort of tutor teaching the programming subject, many students are still challenged by programming. Reports from teachers of programming and results from some empirical studies now suggest that the teaching of programming has created significant difficulties for high school and university students and has failed to catalyze the development of higher order thinking skills. According to the ITiCSE 2001 Working Group had established that many students do not know to program at the end of their introductory courses. The explanation of this incapacity is that the students lack the ability to problem solving skills. They lack the ability to take a problem description, decompose it into sub-problems, implement them and assemble the pieces into a complete solution. An overall of the explanation shows that many students have a fragile grasp of both the basic programming principles and the ability to carry out routine programming tasks such as tracing through the code. So the ITiCSE 2004 Working Group produce a report based on the explanation that given by the ITiCSE 2001. The ITiCSE 2004 tests the students from seven countries in two ways. First, test the ability of the students to predict the outcome of executing a short piece of code. Second, the ability of students is tested when given the desired function of short piece of near-complete code to select the correct completion of the code from a small set of possibilities. The ITiCSE 2001 explored the twelve Multiple Choice Questions (MCQs) by asking students to demonstrate their comprehension of existing code. If a student can consistently demonstrate an understanding of existing code but struggles to write similar programs that is reasonable to conclude that the student lack the skills of problem solving. If a student cannot consistently demonstrate understanding of existing code that can be conclude as a lack of knowledge and skills that are a pre-requisite for non trivial and problem solving. Regarding on the pre-requisite matter, many students are weak at this task. For easy question, approximately 68% of students answer correctly and 38% of students answer correctly for most difficult questions. The conclusion from this, students has a fragile grasp of skills that are a pre-requisite for problem solving. So, to make the novices easy to debug the error in their program code, there are many debugging method that proposed in object-oriented programming, which discuss next.

1.3. Debugging Method in Object-Oriented Programming

The debugging technique is needed in teaching programming in order that novice programmers can easily understand the flow of code execution. Much research has been carried out the major issue in Computer Science educational research in process by which novices learn to program. It is difficult to find an effective method of teaching that is suitable for all novices. According to Ahmadzadeh *et al*. (2005), they try to refine a teaching method by a careful examination of novice’s mistake. This research investigates the pattern of compiler error and pattern of logical errors among the novices debugging activities. The research also discovers that many novices with a good understanding of programming do not acquire the skills to debug program effectively. From this research, it shows that the debugging process is very important and the skill at debugging can increase a level of programmers confident.

In the development phases, debugging is an important program that helps in locating and correcting programming errors. There are many ways in the area of making debugging that can be understandable to the novice programmer, but we are not currently aware of any ways in making logic error easier to interpret by novice programmers. Algorithmic debugging from Cheda and Silve (2009) is a semi automatic debugging technique, which is based on the answers of an Oracle of questions generated automatically by the algorithmic debugger. The algorithmic debugger has a front-end which produces a data structure representing a program execution also called execution tree and a back-end which uses the execution tree to ask the questions and process the oracle answer to locate the bugs. This debugger only captures the declarative aspects of the execution and ignores the operational details.

Debugging on multi agent tools, which are consist of complex components and concurrent, are difficult. Agent Oriented Programming and Design class (Poutakidis *et al*., 2003) is developed to describe and categorize a range of bugs found in multi agent tool which is developed by students. It is a mechanism for taking protocol diagrams developed in Agent UML (AUML), converting them to Petri nets and then using them to monitor execution and detect problem. Instead of presenting messages to the programmer and relying on programmers to detect problems, the debugging agent proposed in monitor conversation and detect problem by identifying messages that are not permissible in the conversation defined by the protocols. This debugging agent uses a Petri nets to monitor the interaction for errors and to know how the error may have occurred. Despite many novice programmers apply techniques as ineffectively and inconsistently in detecting and fixing a bug but they have some strategies use to found and fixing bugs such as tracing, commenting out code, diagnostic print statements and methodical testing. According to Katz and Anderson (1987), found that three general bug location strategies were used by students such as mapping program behavior to a bug, hand tracing of the code and causal reasoning.

Another debugger perform debugging process on object-oriented programs with introduce a new kind of abstraction such as the behavior views that can be used to specify the expected actions occur for a program task in various scenarios. It also can be used to monitor the actions for the tasks have been performing correctly. According to Liang and Xu (2005), bug detection and localization activities should also be organized based on scenarios. To facilitate debugging activities, it propose scenario-driven debugging approach to allows the software developers to effectively use their knowledge of scenarios built during the requirements analysis and design to detect and pinpoint problems in the implementation and this approach can improve the effectiveness of debugging. So, to improve the novice programmers debugging skills, the novice programmers need receive a formal debugging training at an early stage to become better programmers in tool. During the training, the novice programmers would gain debugging experience and this experience could assist them. It found some of debugging technique use a formal debugging training to helps novice programmers develop skills in debugging tasks with design multiple activities which include debugging exercise, debugging logs, development logs, reflective memos and collaborative assignments (Chmiel and Loui, 2004).

Some of the researcher tries to apply the idea of declarative debugging to the object-oriented language as an alternative to traditional trace debuggers. The declarative debugger introduced by Caballero *et al.* (2007) is used when a wrong computation occur with build a suitable computation tree containing information about method invocations. The tree is then navigated with ask the user questions in order to compare the intended semantic of each method with its actual behavior until a wrong method is found out. This declarative debugger is used for debugging on object-oriented language, Java. Hybrid Debugging Technology, HDT (Kouh and Yoo, 2003) is a debugging technique to debug a Java program with combines an algorithmic debugging method with a traditional stepwise debugging method. This technique can improve the drawback of two methods and also tries to do towards an automated debugging that users can conveniently to debug a Java program. In traditional debugging, there are two ways for debugging logical errors in a Java programs. First, the novice programmer directly analyzes the source code or directly inserts the screen output instruction in the suspected location. This ways is a simple and effective in most language but these ways not easy to locate the logical errors because it is difficult for them to correctly anticipate the error location. Second, the novice programmers also can debug a Java program using the instruction such as step-over, step-into, go and break-point. As a step-over and step-into instruction can execute a statement per one instruction. The instructions go execute a statement with break-point. But this technique is needed much time from novice programmers because the novice programmers should execute all statements in the worst case.

Convergence debugging (Nikolik, 2005) is a new automatic debugging method. It isolates a set of text cases that converge on the internal root cause of a failure. This method use a new measure of code level distance to evaluate the debugging effectiveness of a set of test cases between a set of debug test cases and the test cases that caused the failure. From the previous work that the researchers introduce the HDT for debugging logical errors in Java program has some disadvantages. It can reduce the number of programmers debugging in Java program but it cannot reduce the number of debugging because the size of the recent programs still increase than the past programs and the number of methods also is increasing. So the researchers from Kouh and Yoo (2003) propose the HDTS using a Program Slicing Technique (PST) at the HDT. It is combining of Hybrid Debugging Technique (HDT) and Program Slicing Technique (PST) where the PST can reduce the number of programmer debugging. The PST function can remove the correct nodes at an execution tree and the correct statements in the erroneous method when a programmer debugs programs using HDTS.

Much effort is spent on the development of tools to help programmers in constructing, debugging and verifying programs such as PHENARETE tool that introduce by Wertz (1982), which understands and improves incompletely defined LISP programs those written by students beginning to program in LISP. As an input in this tool, it takes a program without any additional information. To understand the program, the tool Meta-evaluates it using a library of pragmatic rules, describing the construction and correction of general program constructs and a set of specialists, describing the syntax and semantics of the standard LISP functions. The tool can detect errors, eliminate them and justify its proposed modifications by analyzing the text of a program and detects informalities or inconsistencies and proposed possible corrections or improvements. There are some researches that implement by Lee and Wu (1999), reported about improving the programming skills of novice programmers that focus on the program debugging practices. This research working on expanding the scope of DebugIt to cover other programming constructs introduced in the CS1 and CS2 courses. This research presents a model of debugging practices called DebugIt to uncover and to correct any misconceptions of the programmers and to improve the debugging abilities of the programmers. DebugIt was developed specifically for debugging on program with loop related errors in introductory Pascal courses. This proposed model called for supervised debugging practices on short programs involving frequently committed programming errors.

1.4. Debugging Tool for Programming Code

There are several available debugger tools exist. The capabilities of some of the available debuggers will be explained and discussed. There are such as jBixbe, DBG | PHP, Jswat, Backstop tool, WPOL, CMeRun, CnC and OOCD.

Bixbe (2006) apply debugging in Java applications on the conceptual level of the Unified Modeling Language (UML) at which they are designed and makes it possible to find not only simple bugs but also weaknesses and insufficiencies in application design. It can show the details of application so that can realize classes, objects, their relationships and interaction. jBixbe provides a new quality of debugging complex Java applications by showing their structure and functioning on the conceptual level of the UML. The advantages of jBixbe debugger are perfect debugging of multi-threaded applications, user-friendly GUI and good representation of data structures. This debugger also teaches object-oriented concepts (training, teaching) and provides source code debugging and breakpoints. But there are some disadvantages of these debuggers. Some researches felt that it is very complicated when debug a large application because jBixbe is created for high level object-oriented Java debugger. So, it is very difficult for novices' learner to study and understand the error because the jBixbe do not locate the error. This debugger also do not have pop-up window to tell a user what should do if errors occurs.

The second example is a DBG | PHP–Debugger DBG (NuSphere, 2009) it is an open source debugger and profiler for PHP programming language. PHP Debugger is the best tool for helping the bugs fast and eliminates them from the PHP programs. It supports a GUI interface as well as a command-line interface. DBG is a full-featured PHP debugger, an interactive tool that helps debugging PHP scripts. It works on a production or development web server and allows debug your scripts locally or remotely, from an IDE or console. PHP Debugger provides a powerful and easy way to simplify PHP debugging because it gives complete visibility and control over the execution of PHP scripts. It also doesn't require that you make any changes to your PHP code. PHP Debugger can be debugging PHP applications on eighteen different platforms either locally or remotely. Support for the debugging of nested calls (PHP Scripts calling PHP Script), multiple parallel debug sessions and debugging of PHP CLI scripts set PhpED apart from other PHP IDE's. The advantages of DBG|PHP-Debugger are lets user step by step through the execution of a PHP scripts, line-by-line and user friendly GUI. This debugger also has good representation of data structures, have the call stack window displays the function call that brought user to the current script location and allows multiple debugger processes running simultaneously. Even though with advantage that DBG | PHP-Debugger can give, but there are some researches implement need Object-oriented Java Debugger which cannot support by this debugger. Another problem is PHP is an old script and many of organization have change to JSP in web development.

The third example is a Jswat debugger (Swat, 2009). It is a standalone and a graphical Java debugger, written to use the Java Platform Debugger Architecture. The Jswat have several features such as breakpoints with conditionals and monitors, colorized source code display, graphical display panels showing threads, stack frames, visible variables and loaded classes, command interface for more advanced features and Java-like expression evaluation including method invocation. The advantages of Jswat debugger are simple and user friendly GUI. This debugger is suitable for analyzing applications (maintenance) and display object relationships (structure diagrams). Even with advantage that Jswat debugger can give, it also have some disadvantages such as do not have pop-up window to tell a user what should do in step by step through the execution of a PHP scripts and do not show as a line-by-line but it just only pin-pointed on the specific error.

Backstop tool (Murphy *et al*., 2004), identify the common runtime error in Java applications and do not identify the logic error. This tool designed for programmers studying Java at the entry level and it provides more user-friendly error messages when an uncaught runtime error (exception) occurs. It also provides the debugging support by allowing users to watch the execution of the program and the changes to the values of variables. Similarly with the educational tool Expresso proposed by Hristova *et al*. (2003) identify the common error in Java programming and generate error messages that provide suggestions on how to fix the code. Some existing tools have been designed to identify logic errors quickly but not give any suggestions to solve them.

The propose Static Object-Oriented Debugging Model (SOODM) is most similar with the CMeRun (Etheredge, 2004). It is a tool in UNIX environment to debug a logic error and allows the programmers to see what is happening inside a program while it is executing. Many novices apply the debugging techniques ineffectively or inconsistently. There also has been investigation of debugging techniques among the novice programmers. Although, the tool that presented in this research can be used to make debugging techniques among novice stable and not fragile.

CAP (Schorsch, 1995), developed to aid novice programmers in a user-friendly fashion by reporting common syntax, logic and style errors that they make in Pascal program. It also give feedback information to the novices about the error, the reason error has been occur and give the solution to fix the problem.

According to Ebrahimi and Schweikert (2006), novices may not detect negative interactions between section or block of code that are locally correct but globally incorrect. For example, the code to perform the output may be correct but in the wrong place in the program. So WPOL (Ebrahimi and Schweikert, 2006), is designed to facing the problem. WPOL is being designed to incorporate the Plan-Object-Paradigm, Web and assessment with focus on plan integration. WPOL is plan object-oriented and teaches novices programming by plan management as to how they are integrated and bridges the gap between object and functions. A plan that used in WPOL is an abstraction of a concept, requirement, object and programming code. The plan is used for structured knowledge representation in natural language processing.

Another available debugger is Check 'n' Crash (CnC) introduced by Csallner and Smaragdakis (2005) is an automatic error detection approach which combines the static checking and automatic test generation to get the best of both function in order to detecting errors. The CnC tool combines the advantages of ESC/Java static checker and JCrasher tool automating testing tool. On this tool, it consist of taking the abstract error conditions using theorem proving techniques by a ESC/Java static checker and deriving the specific error conditions using a constraint solver then produce concrete test cases that are executed to determine whether an error truly exists by JCrasher tool. Visual tool is an alternative ways, which can help the novices more understandable when learning the programming language. It can show for novices what happens when the code is executed. Visual Debugger for Java programs (JVD) introduced by Rafieymehr and McKeever (2007) is developed using the graphical animation and runtime state retention to display program state during execution. These functions to detect runtime errors by determine which classes have main methods and ask user to choose one and display the code with highlight showing current line. The code also displayed in balloon boxes. Interpreting compiler error messages is challenging for novice programmers (Hartmam *et al*., 2010). HelpMeOut by Hartmam *et al*. (2010) is a social recommender tool that aids novices with the debugging of compiler error messages by suggesting successful solution to the errors that other programmers have found. Its comprises IDE instrumentation to collect example of code changes that fix by compiler errors then store the fix reports from many users to the central database and queries the database and present the suggestion solution on relevant fixes to the novices. Many technique have been designed in debugging area to find bugs in software but some of available technique are difficult to use and not effective in finding real bugs. In order to study in deep of programming processes there have two kind of important thing. Firstly, the researchers must control the knowledge structures that programmers possess if they wish to measure the effects of factors that influence programmer performance. Secondly, the researchers should understand the knowledge structures that novice programmers possess (Vessey, 1985).

This research has focused the problems that occur in debugging process to object-oriented programming among the novice programmers. Next will discuss the ways that can be used to improve the learning of object-oriented programming.

1.5. Learning of Object-Oriented Programming

Among proposed solutions tools visualization is discussed about the visualization should be a means to make the abstract concepts illustrative and concrete. However, quite often the empirical evidence for the effectiveness is missing as well as didactic knowledge about why and how tools visualization enables teach (Valentine, 2004). One experiment has been done in order to evaluate the effect of a program visualization tool for helping students to better understand the dynamic of object-oriented programs. The experiment evaluate the effect of a visual debugger to help novices learning the object interaction by using BlueJ's debugger and object inspector as a control group experiment in an introductory programming course. This experiment is focus especially on the object-oriented paradigm. The result of the experiment show that the novices who used BlueJ's debugger and object inspector statistically performed significantly better than novices manually tracing the program execution on the same exercise given in the experiment.

There are some researches that presents case studies which is illustrates a Problem-Based Learning (PBL) environment with appropriate use of resources for a first year course in Java. In PBL environment, students work in group in real life problems and have the opportunity to determine for themselves the requirement that needed to learn in the relevant subject area. The features of problem-based learning are to provide students with a range of resources that assist them to solve the problems.

Learning to program is a time-consuming and frustrating process for most novice programmers. According to Johnson (1990), one reason for this is that they have to expend so much effort in debugging their programs. The process of analyzing programs for syntactic errors is well understood for the techniques exist which do a fairly good job of identifying syntactic errors and of correcting it.

Research on the difficulties novice programmers meet when they attempt to program a computer has been quite active in Computer Science Education area. Web Integrated Programming Environment (WIPE) is designed specifically to teach novices the fundamentals of programming. The environment is designed for use as a first programming course in order to help students become familiar with the main programming concepts. WIPE is educational software developed to introduce novices to programming. It teaches programming based on the accumulated experience and practice gained from numerous related research efforts in the broader area of the teaching of programming. WIPE designed based upon and was influenced by some fundamental didactic principles and the experience obtained by former research regarding the teaching in introductory concepts on programming in order to make its more effective. WIPE is software that targeting the teacher rather than the student that can assist teachers with pinpoints the specific area where the students have difficulties.

Many researchers can create more effective learning environment if they understand the process of learning a first programming language (Garner *et al*., 2005). They found some research to analysis the programming student's problem in an Introductory Java programming class at the University of Otago. These researches discuss the tool and methods that use to present the list of problem definitions, which is used to classify student's problems during the laboratory work for Introductory Java programming class. The discussion in the context of the novice programming literature is involving a data collected during 2003. The result from this discussion is consistent with trends noted in the literature and highlights the significance of both fundamental design issues and the procedural aspects of programming. One of the purposes of the research is to get any comments and suggestions for improvements of novice programming.

1.6. Logic Error

Debugger has very close related with the term called 'logic errors'. There are several kinds of errors, such as syntax error, runtime error and logic error. Each of these errors can be detected in different ways. The logic error has occurred when the program run and gets stuck and crashes or the code compiles and runs without stuck and crashing. At the same time, the codes not produce the intended result. Syntax error has occurred when the code typed is not correctly or not formatted. For the runtime error has occurred during the time being executed. The runtime error maycausebythe computer viruses, bugs in the program or an incompatibility between different computer programs. The most difficult errors to detect are logic error. Logic errors are difficult in terms of finding the location of the logic error occur and do not have error messages which is more understandable way which do not provided by the existing debugger. Many novices unaware the existing of logic error when they compile their code. Thus, this research try to propose a Static Object-Oriented Debugging Model which can help novices or even experiences to find logic error and provide more user-friendly error messages with provide location that logic error occurs and suggestions to solve the errors.

In computer programming, logic error is a bug in a program that occur when the code compiles and runs without crashing but does not do what intended to. Typically, it will be discovered from the incorrect output. These kinds of errors are harder to fix, because we don't necessarily know what causes the error. There are two kinds of logic errors. The first one is when the program is run and the program gets stuck and crashes. The second is where the program doesn't crash, but gives the wrong results. There are some example of logic error that we collected from Hristova *et al.* (2003) and Liang (2011) such as below.

1.7. Improper Casting

This problem occurs when a variable is declared rather than cast (i.e., the casting parentheses are missing). One possible result may involve truncation of important data. An error of this type can also occur as a result of integer division. Introductory novices tend to believe that numbers are just numbers and fail to comprehend the differences and data type necessities between int and float.

1.8. Invoking a Non-Void Method in a Statement that Requires a Return Value

A method that is supposed to return a variable of some type (i.e., it must be made equal to a variable of the return type) is instead called as a void method or as a statement. If this mistake is made, the value returned by the method is lost since it is not stored anywhere.

1.9. Flow Reaches End of Non-Void Method

A non-void method is supposed to return a value of some type, but the return statement is missing due to misunderstanding about the role or type of the method or just forgetfulness.

1.10. Methods with Parameters

1.10.1. Confusion between Declaring Parameters of a Method and Passing Parameters in a Method Invocation

When a method is defined, the parameter types need to be declared. However, in a method invocation the types of the variables passed are not given, only the variable names. There exists confusion between passing parameters, declaring them and identifying them in the method's definition.

1.11. Incompatibility between the Declared Return Type of a Method and in its Invocation

A non-void method that is supposed to return a value of a particular data type but the variable that will receive the return value is of an incompatible type.

1.12. Class Declared Abstract Because of Missing Function

A class that implements some interfaces but is missing one of the major methods that the interface must define and support.

1.13. Invalid Additional Assignment Operator

This type of errors occurs when to assign a value to the variable with using wrong additional assignment operator. The example of errors as:

```
public class ShowLogicError {
    public static void main (String[]args) {
        int number1 = 3;
        int number2 = 3;
number2+= number1 + number2;
Tool.out.println("number2 is " + number2);
}
}
```

1.14. Forgetting Necessary Braces

This error is common among novice programmers. They are forgetting the braces when they are needed for grouping multiple statements. The code below is wrong, it should be written with braces to group multiple statements.

```
if (radius >= 0)
    area = radius * radius * PI;
    Tool.out.println("The area" + "is" + area);
```

1.15. Wrong Semicolon at the if line

Adding a semicolon at the if line is a common mistake that have done by novices. This error often occurs when the novices use the next-line block style. The example of this type of error is shown above.

```
if (radius >= 0) ;
 {
area = radius * radius * PI;
Tool.out.println("The area" + "is" + area);
 }
```

1.16. The Problem is that the Variable y is only Declared Within the Init Method, not Within the Class Itself

That means it cannot be accessed outside the Init method. This type of error can commonly arise if do not careful because the programmers will often wish to assign the initial value to a variable in the Init method.

```
Class test {
    staticint x = 30;
    staticint y;
    /*method*/ ststic void Init() {
    y = 20;
    int x = 10;
    }/*endInit*/
public static void main(String[]args) {
Init();
for(inti=1; i<=10; i++)
Tool.out.println(I + " " + x + " " + y);
}/*end for loop*/
}/*end main*/
}/*end test*/
```

1.17. A Common Error in Writing Mutator Methods is using the Instance Variable name for the Parameter name

When a method parameter has the same name as an instance variable, the parameter "hides" the instance variable. In other words, the parameter has name precedence, so any reference to that name refers to the parameter not to the instance variable:

```
public void setmodel (String model){
model = model;
}
```

Because the parameter (model) has the same identifier as the instance variable (model), the result of this method is to assign the value to the parameter. This is called No-Op ("NO operation") because the statement has no effect.

Some of the research has propose the ways to detect defect at an early stage to reduce their impact, but Chang *et al*. (2008) has proposed a new approach to prevent defects from occurring and has been applied to improve software quality and productivity in many organizations. To prevent defects from occurring in advance, it uses a causal analysis approach to discover the cause of defects and take corrective actions on it. Defect Prediction approach is based on Association Rules, which applies association mining technique. To analyze among large amounts of reported defects is time consuming and requires significant effort. So this approach can solve this problem where the reported defects and performed actions are utilized to discover the patterns of actions, which are likely to cause high defects.

Rather than detecting the errors, the novices also inable to interpret and resolve the compiler messages. Coull (2003), tries to help novices to interpret compiler error messages by develop an application that attempts to provide novices with solutions to compiler error message. The application utilizes a program that parses compiler error messages from a Java Integrated Development Environment to a text file and a database that contains common compiler error message and their solutions. Most of researcher tries to investigate errors made by novices. There are many ways to investigate the errors made by novices on the first time their study a programming language. When studying a programming language for the first time, the majority of student's errors fall into broad (and well-documented) categories (Barr *et al*., 1999). In this research, they aims to investigate errors made by novices in Blue which is a new object-oriented language that are specifically designed at the University of Sydney in purpose of teaching novices. The investigation is done by a survey that is delivering over the World Wide Web. The survey consists of multiple choice and free-form short answer questions. Blue is a programming language and environment developed specifically for teaching object-oriented programming to first year Computer Science students. It has been designed to make teaching programming concepts to become easy by removing complexity from the language at the expense of performance. As a result from this survey, it shown that a student who learns with Blue is no more likely to make errors that is commonly made by novices. The Blue is not necessary better equipped to design and write code in an object-oriented paradigm. It is need further research to be identified in this area. This problem leads to our research in order to suggest a new model of tool, such like the object-oriented debugger model. The importance to come out with a good design of this tool will aid and help novices to pinpoint selected logic error, which at the same improve their programming skill. The identified logic error will be as the input data of the model that will be design.

2. MATERIALS AND MEHTODS

2.1. Methods

The UML is a modeling language for designing the software system. All the requirement of the system will be describe and model in UML structure. The UML notation is very important in modeling. Use the appropriate notations to make the model more understandable. The standard UML notation that used to describe the Object Oriented Debugger is Class notation, Collaboration notation, Use Case notation and Interaction notation. Object oriented programming is a new approach to programming which address these systems issues, moving the focus from programs to software (Anderson, 1988). With using the object oriented, it easier to build and understand the systems. So, in this research we present an automated debugger for object-oriented programming.

Steps involve for this research work on come out the model are as follows (a) creating use case; (b) capturing process flow and (c) finding object interaction. The details output will be explained next.

3. RESULTS AND DISCUSSION

3.1. Creating Use Case

Use case is a type of behavioral of the tool that illustrates using the use case diagram. Use case is useful for analysts to more understand the flow of the tool and it does also can help the analysts to partitioning the functionality of a tool. It's present a functionality of a tool that shows each function that will be performed by an actor. In this diagram, it is shows the relationship between the use case and actors in the tool. There is one actor as a user that involve in the Static Object-Oriented Debugging Model. In this use case diagram, it is identified one use case, which can access by user that is static debugger. This use case begins when user enter to the tool and select the program. The tool will parse each of line codes in the cod program. The codes that already parse will be stored into database. After that, the tool will check the code refer to the plan base in database. Finally, the tool produce the output that is description about the error appears in the program code. **Figure 1** shows the use case diagram for Static Object-Oriented Debugging Model.

3.2. Capturing Process Flow

Activity diagram can be used to describe the stepwise activities of each component in the tool. There are seven components that involve in the Static Object-Oriented Debugging Model, it's begin with the user enter to the tool and select the program. The tool will parse each of line codes in the program code. The codes that already parse will be stored into database. After that, the tool will check the code refer to the plan base in database. Finally, the tool produce the output that is description about the error occurs in the program code. The Activity Diagram for Static Object-Oriented Debugging Model is shows **Fig. 2**.

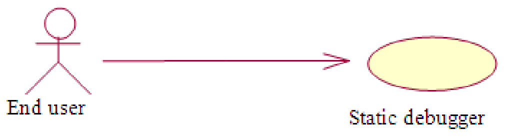

Fig. 1. Use Case Diagram for Static Object-Oriented Debugging Model

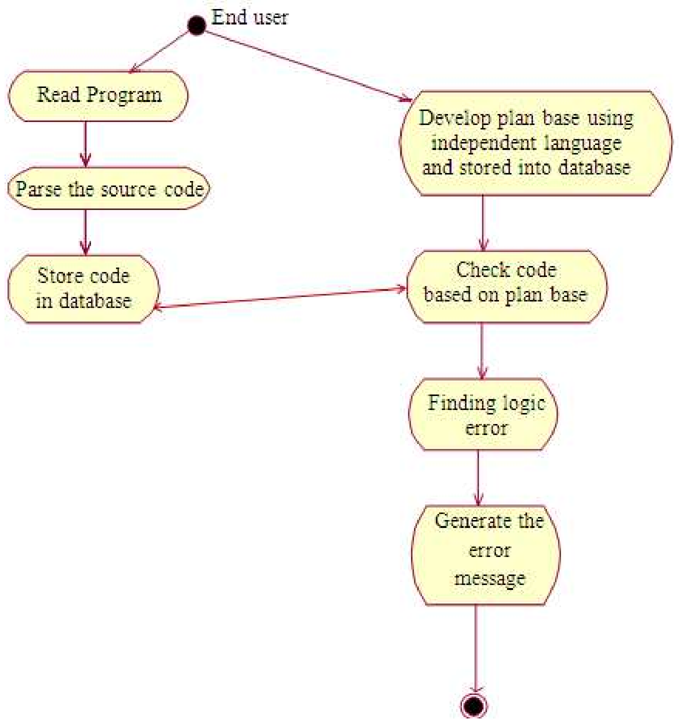
End user
Read Program
Parse the source code
Store code in database
Develop plan base using independent language and stored into database
Check code based on plan base
Finding logic error
Generate the error message

Fig. 2. Activity diagram for static object-oriented debugging model

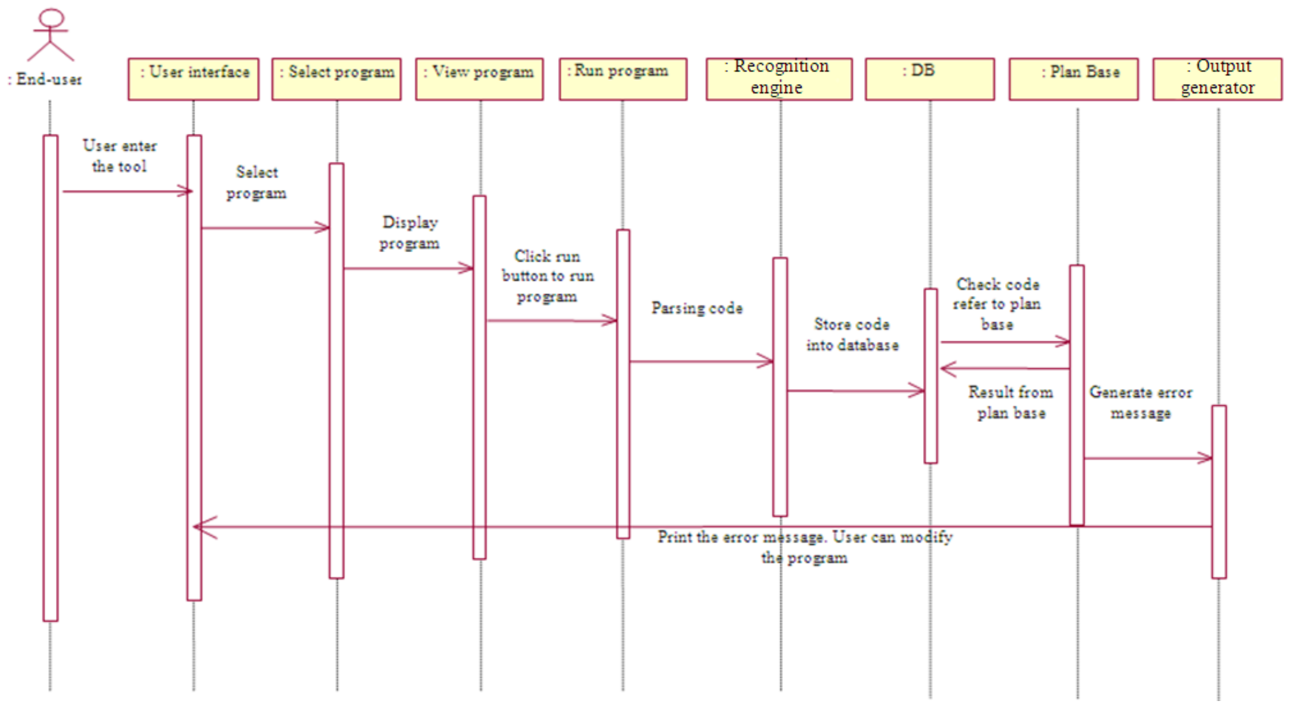

Fig. 3. Sequence diagram for static object-oriented debugging model

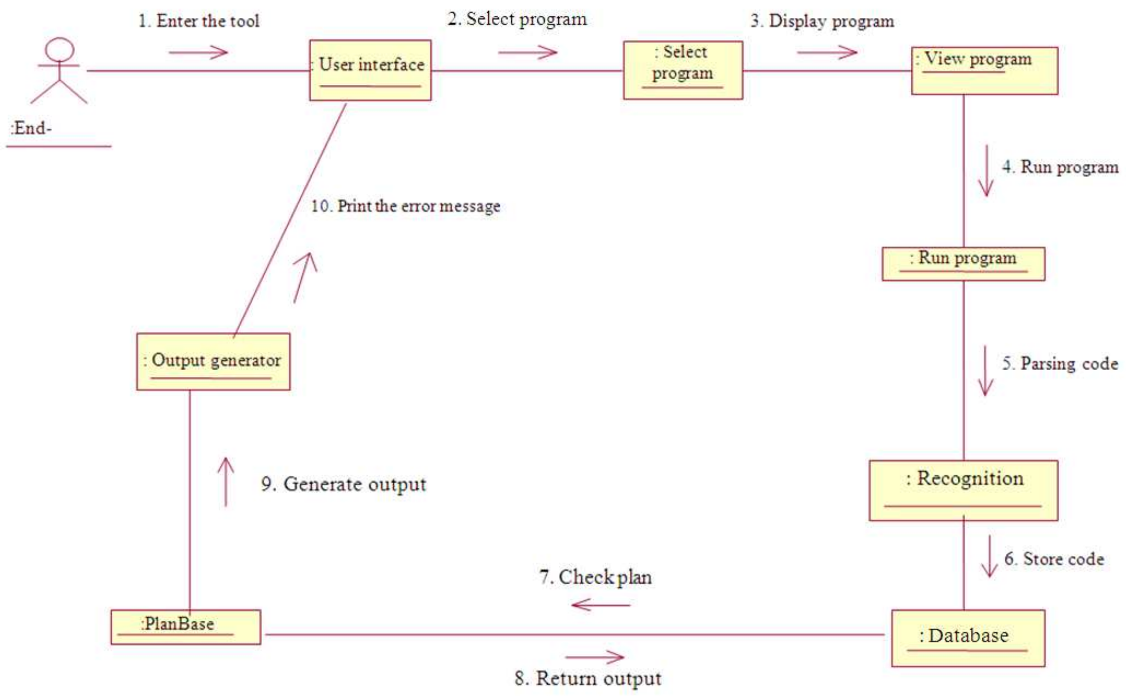

Fig. 4. Collaboration diagram for static object-oriented debugging mode

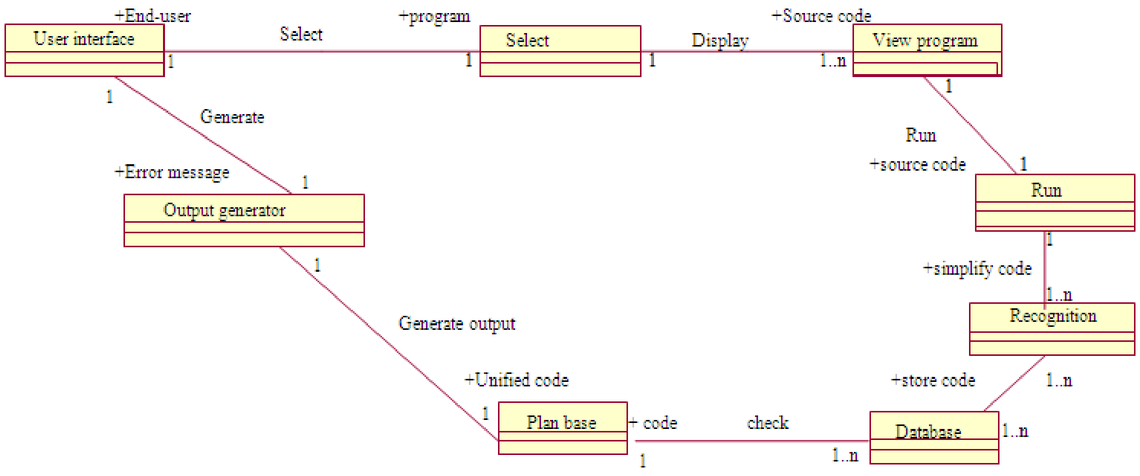

Fig. 5. Class diagram for static object-oriented debugging model

3.3. Finding Object Interaction

There are two types of object interaction diagram that are sequence diagram and collaboration diagram. We use the sequence diagram to represent the flow of messages, events and actions between the objects of a tool in a time sequence. It's also used to describe the sequence of actions that need to complete a scenario of the tool. The collaboration or interaction diagram shows the relationship and interactions among the objects in the tool. Class diagram is one type of diagram or model in the Unified Modeling Language. It's used to describe the structure of a tool by showing the tool classes, attributes and relationship between the classes. In Static Object-Oriented Debugging Model, it's classified seven classes that are User Interface, Select Program, View Program, Run Program, Code Parser, Database and Plan Base. The object in this tool consists of End-User. This tool begins with User Interface class which user can enters to the tool and the Select Program class allowed the user to select the program. The View Program class allow user to view the program that selected by the user. When Run Program is function, the program will be processed, the tool will parse each of line codes in the program code is done by Recognition Engine class. The codes that already parse will be stored into Database. After that, the tool will check the code refer to the Plan Base in database. Finally, the tool produce the output that is description about the error occurs in the program code. The several main classes in Static Object-Oriented Debugging Model are shown in **Fig. 3** below. The sequence diagram and collaboration diagram are shown in **Fig. 4 and 5**.

4. CONCLUSION

The design of the Object Oriented Debugger by using the Unified Modeling Language Model has been discussed detail in this study. It's consisting of two parts that are object-oriented analysis and object oriented design. All the developing and designing are based on the model in UML. This study has explained on the concepts of Object Oriented Debugger and has divided into three sections, which include creating use case, capturing process flow and finding objects interactions. Appropriate UML diagrams for Object Oriented Debugger system also has presented.

5. ACKNOWLEDGEMENT

Special thanks from author to financial support (Fundamental Research Grant Scheme, FRGS) received from the Ministry of Higher Education (MoHE), Malaysia via Universiti Putra Malaysia. The principle investigator of the research project is Dr. Nor Fazlida MohdSani.

6. REFERENCES

Ahmadzadeh, M., D. Elliman and C. Higgins, 2005. An analysis of patterns of debugging among novice computer science students. Proceedings of the 10th Annual SIGCSE Conference on INNOVATION and Technology in Computer Science Education, (ITiCSE' 05), ACM Press, New York, pp: 84-88. DOI: 10.1145/1151954.1067472

Ali, B., 1999. Object Oriented Systems Development: Using the Unified Modeling Language. 1st Edn., McGraw-Hill, Boston, ISBN-10: 0072349662.

Anderson, J.D., 1988. Education of Blacks in the South, 1860-1935. 1st Edn., University of North Carolina Press, Chapel Hill, ISBN-10: 0807842214, pp: 366.

Barr, M., S. Holden, D. Phillips and T. Greening, 1999. An Exploration of novice programming errors in an object-oriented environment. Proceedings of the Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education, Jun. 27-30, ACM Press, USA., pp: 42-46. DOI: 10.1145/349316.349392

Bixbe, J., 2006. Another way to debug.

Caballero, R., C. Hermans and H. Kuchen, 2007. Algorithmic debugging of java programs. Elect. Notes Theoretical Comput. Sci., 177: 75-89. DOI: 10.1016/j.entcs.2007.01.005

Chang, C.P., C.P. Chu and Y.F. Yeh, 2008. Integrating in-process software defect prediction with association mining to discover defect pattern. Inform. Software Technol., 51: 375-384. DOI: 10.1016/j.infsof.2008.04.008

Cheda, D. and J. Silva, 2009. State of the practice in algorithmic debugging. J. Elect. Notes Theoretical Comput. Sci., 246: 55-70. DOI: 10.1016/j.entcs.2009.07.015

Chmiel, R., M.C. Loui, 2004. Debugging: From novice to expert. Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, Mar. 03-07, ACM Press, USA., pp: 17-21. DOI: 10.1145/971300.971310

Coull, N., 2003. Helping novice programmers interpret compiler error messages. Proceedings of the 4th Annual LTSN-ICS Conference, (ALIC' 03), pp: 26-28.

Csallner, C. and Y. Smaragdakis, 2005. Check 'n' crash: Combining static checking and testing. Proceedings of the 27th International Conference on Software Engineering, May 15-21, ACM Press, St. Louis, MO, USA., pp: 422-431. DOI: 10.1145/1062455.1062533

Ebrahimi, A. and C. Schweikert, 2006. Empirical study of novice programming with plans and objects. SIGCSE Bull., 38: 52-54. DOI: 10.1145/1189215.1189169

Etheredge, J., 2004. CMeRun: Program logic debugging courseware for CS1/CS2 students. Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, Mar. 03-07, Norfolk, USA., pp: 22-25. DOI: 10.1145/1028174.971311

Garner, S., P. Haden and A. Robins, 2005. My program is correct but it doesn't run: A preliminary investigation of novice programmers' problems. Proceedings of the 7th Australasian Conference on Computing Education, (ACE' 05), Australian Computer Society, Inc. Darlinghurst, Australia, Australia, pp: 173-180.

Hartmam, B., D. MacDougall, J. Brandt and S.R. Klemmer, 2010. What would other programmers do: Suggesting solutions to error messages. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, (CHI' 10), ACM Press, New York, USA., pp: 1019-1028. DOI: 10.1145/1753326.1753478

Higgins, C.A., G. Gray, P. Symeonidis and A. Tsintsifas, 2005. Automated assessment and experiences of teaching programming. J. Educ. Resources Comput. DOI: 10.1145/1163405.1163410

Hristova, M., A. Misra, M. Rutter and R. Mercuri, 2003. Identifying and correcting java programming errors for introductory computer science students. Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education, (CSE' 03), ACM Press, New York, USA., pp: 153-156. DOI: 10.1145/611892.611956

Johnson, W.L., 1990. Understanding and debugging novice programs. Artif. Intell., 42: 51-97. DOI: 10.1016/0004-3702(90)90094-G

Katz, I.R. and J.R. Anderson, 1987. Debugging: An analysis of bug-location strategies. Hum. Comput. Interact., 3: 351-399.

Kolling, M., B. Quig, A. Patterson and J. Rosenberg, 2001. The BlueJ system and its pedagogy. Pennsylvania State University.

Kouh, H.J. and W.H. Yoo, 2003. The efficient debugging system for locating logical errors in java programs. Proceedings of the International Conference on Computational Science and its Applications, May 18-21, Springer Berlin Heidelberg, Canada, pp: 684-693. DOI: 10.1007/3-540-44839-X_72

Lee, G.C. and J.C. Wu, 1999. Debug it: A debugging practicing system. Comput. Educ., 32 165-179. DOI: 10.1016/S0360-1315(98)00063-3

Liang, D. and K. Xu, 2005. Debugging object-oriented programs with behavior views. Proceedings of the 6th International Symposium on Automated Analysis-Driven Debugging, (ADD' 05), ACM Press, New York, USA., pp: 133-142. DOI: 10.1145/1085130.1085148

Liang, Y.D., 2011. Introduction to Java Programming: Comprehensive Version. 8th Edn., Prentice Hall, Boston, ISBN-10: 0132130807, pp: 1342.

Murphy, C., E. Kim, G. Kaiser and A. Cannon, 2004. Backstop: A tool for debugging runtime errors. Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education, (CSE' 04), ACM Press, New York, USA., pp: 173-177. DOI: 10.1145/1352135.1352193

Nikolik, B., 2005. Convergence debugging. Proceedings of the 6th International Symposium on Automated Analysis-Driven Debugging, (ADD' 05), ACM Press, New York, USA., pp: 89-98. DOI: 10.1145/1085130.1085142

NuSphere, 2009. Php Debugger in NuSphere PhpED. Sphere Corp.

Pears, A., S. Seidman, L. Malmi, L. Mannila and E. Adams *et al*., 2007. A survey of literature on the teaching of introductory programming. Proceedings of the Working Group Reports on ITiCSE on Innovation and Technology in Computer Science Education, (CSE' 10), ACM Press, New York, USA., pp: 204-223. DOI: 10.1145/1345443.1345441

Poutakidis, D., L. Padgham and M. Winikoff, 2003. An exploration of bugs and debugging in multi-agent systems. Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems, (AMS' 03), ACM Press, New York, USA., pp: 1100-1101. DOI: 10.1145/860575.860815

Rafieymehr, A. and R. McKeever, 2007. Java Visual Debugger. SIGCSE Bull. DOI: 10.1145/1272848.1272889

Sani, N.F.M., A.M. Zin and S. Idris, 2009. Implementation of conceiver++: An object-oriented program understanding system. J. Comput. Sci., 5: 1009-1019. DOI: 10.3844/jcssp.2009.1009.1019

Schorsch, T., 1995. Cap: An automated self-assessment tool to check pascal programs for syntax, logic and style errors. Proceedings of the 26th SIGCSE Technical Symposium On Computer Science Education, Mar. 2-4, Nashville, Tennessee, USA., pp: 168-172. DOI: 10.1145/199688.199769

Swat, J., 2009. JSwat java debugger.

Valentine, D.W., 2004. CS educational research: A meta-analysis of SIGCSE technical symposium proceedings. Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, (CSE' 04), ACM Press, New York, USA., pp: 255-259. DOI: 10.1145/971300.971391

Vessey, I., 1985. Expertise in debugging computer programs: A process analysis. Int. J. Man-Mach. Stud., 23: 459-494. DOI: 10.1016/S0020-7373(85)80054-7

Wertz, H., 1982. Stereotyped program debugging: An aid for novice programmers. Int. J. Man-Mach. Stud., l: 379-392. DOI: 10.1016/S0020-7373(82)80047-3

Wiedenbeck, S., V. Ramalingam, S. Sarasamma and C. Corritore, 1999. A comparison of the comprehension of object-oriented and procedural programs by novice programmers. Int. Comput., 11: 255-282. DOI: 10.1016/S0953-5438(98)00029-0

Zin, A.M., S.A. Aljunid, Z. Shukur and M.J. Nordin, 2000. A Knowledge-based automated debugger in learning system. Proceedings of the 4th International Workshop on Automated Debugging, (WAD' 00), ACM Press, Munich.