Run Time Evaluation by using Object Oriented Debugging Tool

Ashvini A. Patil, Swapnil V. Suryawanshi

Abstract: ***In the process of Software Development and evolution, Developer has to answer multiple questions about how the code or software behaves at runtime and already many options available for debugging.***

Debugging is an essential part of programming language and what sets great programmers apart from average ones. Beginners are often pleased if a bug/virus that was seen earlier inexplicably disappears. Inexperienced programmers have a tendency to shy away from error messages or be frightened by observable errors, whereas skilled programmers rely heavily on error messages and he is aware about fixing of bugs by using different debugging tool. And programmer can easily detect and remove it at run time. The traditional or classical debugger while debugging gives developer bunch of breakpoints in the source code.

Object based debugging offer, interruption when a given or a particular object is accessed or modified. Programmers, who try to find violations in such source code, need new tool that allows them to explore objects in the system effectively. The implementation of the proposed debugging described here offers programmers an effective tool which will allows searching of objects even for programs that have huge number of objects. Therefore Successful debugging tool involve efficient exploratory ability and a proper understanding of troubleshooting in programming code.

I. INTRODUCTION

As stated from traditional tools that, the complication of object oriented system increases, as the number of different objects in programs increases debugging becomes relatively difficult. Developer needs a dedicated user interface for object oriented programming.

Object based debugging tool able to detect and analyse the relationship in between the objects during the runtime. So the key behind this is to focus on a particular object instead of the execution stack. Traditional debuggers are focused on the execution stack which may create chance of bug availability as well as time consuming process because programmer has to spot the different object parts in code as per their views and interest. We have to fix multiple breakpoints accordingly. The software then runs until a breakpoint is reached, and the developer can then inspect and interact with the code and entities in the scope of the breakpoint. Unfortunately developer may not be fix breakpoints properly at run time. As a result, identifying the right place to set breakpoints in the source code requires a deep understanding of what happens during the execution. Second, debugging operations are focused on the execution stack, rather than on the objects. There exists therefore a considerable conceptual gap between the interface offered by the debugger and the questions of interest by the developer [20] [21].

Manuscript Received November, 2013.

Ashvini A. Patil\, ME computer Engg. VBCOE, Ahmednagar**,** India.
Swapnil V. Suryawanshi, BE IT. VBCOE, Ahmednagar, India.

Object based debugging offer, interruption when a given or a particular object is accessed or modified. Programmers, who try to find violations in such source code, need new tool that allows them to explore objects in the system effectively. The implementation of the proposed debugging actually offers programmers an effective tool which will allows searching of objects even for programs that have huge number of objects.

Object based debugging tool looks forward to analyse the relationship in between the objects during the runtime. This allows functioning operations directly on objects rather than on the execution stack. Our tool can provide an interface to programmer which will make easy different operations, which going to perform on a particular object. Object based operations directly act on objects by intercepting access to runtime state; thus monitoring how objects interact and Support interactions.

There exists therefore conceptual gap between the interface offered by the debugger and the need of the developer, hence to overcome or fill the gap; there is a need for object based debugging tool which is helpful to have previous object states and object reference flow information at hand during debugging.

Our debugging tool is able to capture object state at runtime. It also monitors object specific interactions and it support live interaction that is at run time. For this we keep track the relevant data that is it store object history information together with the regular objects in the application memory.

II. RELATED WORKS [21]

For developing proposed work, we have gone through different existing system to become better approach for object oriented debugging tool. Following literature is study about existing systems working and critically evaluated on some evaluation method to find shortcomings from them.

In Query Based debugging approach user defines a query in a higher-level language that is then applied to the data Queries can test complex object interrelationships and sequences of related events.

Trace oriented Debugger: it is collected of a well-organized instrumentation for incident making, a specific database for scalable storage space, and support for partial traces to reduce trace volume [2].

While this method has the advantage that nowhere data is lost, its drawback is that it requires large hardware power, which is not available for many developers today [6].

The why line debugging interface approach.

Why line tool which facilitate developer to ask, “Why did” and “Why did not” questions regarding their program’s output Why line tries to facilitate developer by applying static as well as dynamic analyses and after that answer Some of the developer questions [7].

In Back-in-time debuggers approach; these are

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

extremely useful tools for identifying the causes of bugs. Compare to the "omniscient" approaches that try to remember all previous states are impractical because they consume too much space or they are too slow. So many approaches to limit these penalties, but they ultimately end up giving out too much relevant information. In this paper a practical approach that attempts to keep track of only the relevant data. In contrast to other approaches, it keeps object history information together with the regular objects in the application memory. This method has the effect that data not reachable from current application objects that's why not useful further.

This approach, present idea which explains that memory utilization stays in practical limits. Furthermore, the performance penalty is significantly less than with other approaches [1].

Back-in-Time Debugging:

Back-in-Time Debuggers are useful tool for identifying the cause of errors, not the omniscient debugger which always remembers all previous states.

To overcome this drawback of omniscient debugger back in time debugger is developed. Omniscient Debugging: also known as back-in-time debugging or reversible debugging.

These debuggers store the total history and execution trace of a debugged program. Developers can explore the history by simulating step-by-step execution both forward and backward [1] [6].

In Auto Flow an automatic debugging approach; Aspect-oriented programming (AOP) is gaining popularity with adoption of languages such as AspectJ.

During AspectJ software evolution, when tests fail, it may be lengthy or difficult for programmers to find out the failure minimising changes by manually inspecting all code editing.

To beat the costly attempt spent on debugging developed AutoFlow, an automatic debugging approach for AspectJ system. AutoFlow meets the potential of delta debugging algorithm with the benefit of change impact analysis to slow down the search for imperfect changes. It primary uses change collision analysis to identify a subset of responsible changes for a failed test, after this ranks these changes according to proposed heuristic (indicating the likelihood that they may have contributed to the failure), finally this improved delta debugging algorithm to determine a minimal set of faulty changes.

The important advantage of AutoFlow is that it can automatically reduce a big portion of irrelevant change in an early stage, eventually then locate not fixed changes effectively [8].

NUDA a Non-Uniform Debugging approach.

This paper is proposed a novel non-uniform debugging architecture (NUDA). This makes hardware-assisted debugging both feasible and scalable for many-core processing scenarios. Here, theme is to distribute the debugging support structures across a set of hierarchical clusters while avoiding address overlap. It allows the address space to be monitored using non-uniform protocols and propose approach to lockset-based race detection supported by the NUDA. Here, page-based monitoring cache in every NUDA node to keep track of footprints. The union of all the caches know how to take in account as a race detection probe without violating execution ordering. [10].

How helpful are automated debugging tools :

The Area of automated debugging, which is with the automation of identifying and correcting a failure's root cause, made tremendous advancements in the past years. However, some of the reported progress may be due to unrealistic assumptions that with the evaluation of automated debugging tools.

These unrealistic assumptions concern the work process of developers and their ability to detect wrong code without explanatory context, or the size and arrangement of fixes. Instead of trying to locate the fault, this proposes to help the developer understand it, thus enabling her to decide which fix they deems most appropriate.

This came to know the need to employ a completely different evaluation scheme that bases on feedback from actual users of the tools in realistic usage scenarios [9].

"A Review of reverse debugging"

Reverse debugging is defined as of a debugger to stop after a failure in a program has been observed and go back into the history of the execution to find reason for the failure.

Reverse execution has become a practical technique available in a number of free and commercial tools. This article review the history and techniques of reverse debugging, as researched, implemented, and used until today [11].

There is a need to find or steer in area where programmers actually face problems during debugging scenario [12].

This strategy works well, trying to understand the general performance for objects. When addressing polymorphism or delegation the performance of objects of same class changes on their composition. In these scenarios need an object-specified analysis and simple breakpoint strategy is not the best option. In application development when programmers require interrupting the execution of the application when a particular code is evaluated, requires breakpoint strategy. The programmer wants to locate the particular object he is concerned. The programmer specifies a suitable condition to recognize the particular object previously found, without interacting with it. This approach may be practicable, if exist few objects to analyze in given code [13].

2.1 Related work shortcoming

Studding and analysing different literature survey following are the outcomes.

- Back in time debugging debugger have to remember history of all previous states.
- There is pretty need of a useful and dedicated user interface for debugging scenario.
- Developer comfortable with using object oriented dedicated user interface for debug situations.
- Trace oriented debugger requires more hardware power, which is practically not possible. Omniscient debugger depend on more memory because, to store history of last stages. Reverse debugging is to stop after a failure in a program has been observed and go back into the history of the execution to uncover the reason for the failure.
- AutoFlow can automatically reduce a large portion of irrelevant change in an early phase, eventually then locate faulty changes effectively.
- After going through literature survey came to know that developer faced some kind of problems while doing debugging. Major problem is that

developer cannot answers about objects. And after taking view on problems faced by developer they do not get answer to their question regarding object.

When complex object oriented system taken in account then traditional debuggers fails to act on object related operations and relationship between different objects. To eliminate these problems new tool should be developed on object based approach and useful dedicated user interface for it [20].

III. MOTIVATION SCENARIOS:

The motivation for doing this project was primarily an interest in undertaking a challenging project in an interesting area of debugging. This gives opportunity to learn about new area of software engineering. This area is possibly an area that I might study at postgraduate level. As the debugging area taken into account developer came across different problems, which are faced by developer. The traditional debugging technique used by programmer is concentrated on stack orientation so developer face problems regarding objects in the code given.

The debuggers not designed to answer many of the questions that developer typically uses to ask after analysing different papers related to approaches of debugging, found that one can develop a debugging tool which is based on objects, and possesses following some points to understand runtime behaviour of the system. It will be helpful to continue interacting with the runtime, applying operations directly to objects without working with static representation of the system. This is useful in to monitor communications with entity objects without taking stepwise breakpoints [20] [21].

So it is required to develop object based debugging tool that facilitated with user interface which fulfil needs of developer such as, different interruption related to objects or keep watch on object interactions and do operations related to objects using user interface telling suggestions.

3.1 System Description

Looking on problems faced by user or developer they do not get answer to their question regarding object. When complex object oriented system taken in account then traditional debuggers fails to act on object related operations and relationship between different objects. To overcome this object based debugging tool is very helpful in this scenario. In this tool Brifost reflection framework is being used. The tool of object based debugging is built on top of the Bifrost reflection framework. Bifrost offers fine grained unanticipated dynamic structural and reflection through meta-objects. Instead of providing different reflective capabilities as an external mechanism integrate all deeply into the environment. Explicit meta objects providing a range of features, thereby evolving both application models and the host language. Meta-objects provide a sound basis for different coexisting meta-level architectures by giving traditional object-oriented techniques to the meta-level. Our proposed system answers to different users requirements like;

- If user wants to find out when method is called during the execution of code.
- If user wants to find out where the instances of this class created at runtime they can easily track it.
- In code, user defines different variables and wants to trace these variable flows in program and wants to know at different break point where these variables are accessed.
- User can trace at different stages or at different break point what is the values of the argument at runtime.
- User can easily trace out how data is passed to the different object at different break point.
- User can easily trace out the relationship between objects.

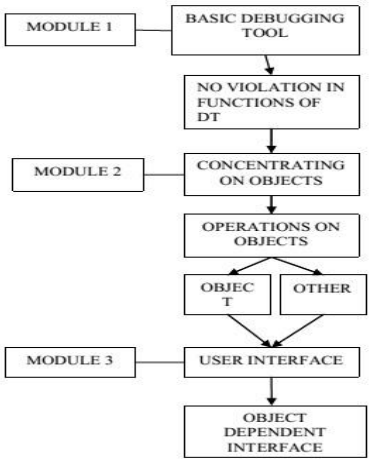

Figure: 1 system description of automatic object based debugging

3.1.1 System Overview

The source code when debug using object based debugging tool, particular object required by developer is searched and made available to developer. Developer further acting on object do the specified operation by using user interface concentrated on objects. The code file taken into proposed tool, then code parsing done for all particular objects. After going through execution and isolates the points needed by developer needs.

The parser extracted all objects from provided code file then supplied or given to execution module. This parser also converted it into intermediate forms which give response to object related errors or bugs. In code generating module there is code which gives object related error findings [20].

Finally execution step it operates on the code parsed taking objects in consideration using a dedicated useful interface for it. The stepwise execution is stated in system workflow.

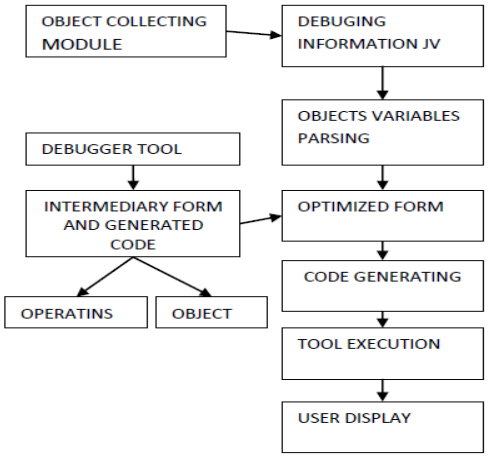

Figure: 2 overview for object based debugging system

3.2 System Workflow:

System workflow of object based debugging have following steps in the system workflow.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Step1: Input source code into object based debugging tool.
Step2: It finds out appropriate required object from given input code.
Step3: It finds relationship like dependency, inheritance between different objects.
Step4: Developer now acts on object.
Step5: Using user interface user do different operation on object.
Step6: Trace out how data is passed to the different object at different break point.
Step7: Trace at different break point what is the values of the variables and different argument at runtime
Step8: Apply this procedure repetitively on whole Source Code document for desired objects.
Step9: Object related operations performed.
Step10: Make changes in objects.
Step11: Prevent problems and so improve performance [21].

IV. EXPECTED RESULT:

Understanding and debugging software systems is difficult. Most used debuggers offer only a limited low-level view of the program state. For the exploration of large data structures, provided a system that allows programmers to ask the program state, helping to check object relationships in large object-oriented programs. This debugger combines several original features.

A new approach to debugging is instead of exploring a single object at a time, an object based debugger allows the programmer to quickly get a set of interesting objects from a potentially very large number of objects, or to check a certain property cause for errors from a large number of objects.

A flexible tool conceptually, evaluates expression for all members of the complex objects. This is simple to understand and to learn, yet it allows a large range of complexity of objects to be formulated concisely. Debugging easier for programmers and facilitating the development of more robust object-oriented systems [20].

- It performs object based debugging and it check for the errors in code also.
- It finds relation between objects.
- It interacts with objects.
- It performs different operations related to objects.

V. CONCLUSION

In this paper we have presented new better approach towards debugging, which is based particularly on objects. Traditional debuggers focused on instances of class and general code file. Developer face problems during interrogating with object oriented arising questions. In this paper Object based debugging tool have dedicated user interface which having object specific dependent operations, this are helpful in dealing with object related errors. In this paper modified traditional debugging tool have stack oriented state but there previous function are not violated, and dedicated user interface is very helpful interacting with the objects. Stack based debugging tool work on entire code by pointed line by line, while object based debugging tool works on desired objects doing operations directly on them. When source code having huge number of objects in case of problems related to objects this approach is useful. This approach is helpful improving the performance of object oriented software's.

VI. ACKNOWLEDGMENTS

I am very thankful to the people those who have provided me continuous encouragement and support to all the stages and ideas visualize. I am very much thankful to the complete VBCOE, Ahmednagar for open handed me all facilities and work environment which enable me to complete my task. I express my sincere thanks to HOD and PG Coordinator, VBCOE, Ahmednagar who gave me their valuable and rich guidance and help in presentation of this research paper.

REFERENCES

[1] Adrian lienhard, tudor Girba and Oscar Nierstrasz "Practical Object Oriented Back-In-Time Debugging"LNCS 5142, pp 592-615.
[2] Raimondas Lencevicius, Urs Holzle And Ambuj K. Singh, "Query-based Debugging of Object-Oriented Programs" OOPSLA 97 Atlanta, USA.
[3] Mark Minas "Cyclic Debugging For pSather, a Parallel Object-Oriented Programming Language" Jan 31 2002
[4] Tanja Mayerhofer,"Testing and Debugging UML Models Based On Fuml" ICSE 2012.
[5] G. Pothier, E. Tanter, and J. Piquer, "Scalable omniscient Debugging, "Proceedings of the 22nd Annual SCM SIGPLAN Conference on Object-Oriented Programming Systems, Languages And Applications (OOPSLA'07), vol. 42, no. 10, pp.535–552, 2007.
[6] C. Hofer,M. Denker, and S. Ducasse, "Design and implementation of a backward-in-time debugger," in Proceedings of NODE'06, ser. Lecture Notes in Informatics, vol. P-88 (GI), Sep 2006, pp. 17-32.
[7] J. KO and B. A. Myers, "Designing the whyline: a debugging interface for asking questions about program behaviour," in Proceedings of the 2004 conference on Human factors in computing systems. ACM Press, 2004, pp. 151–158.
[8] Sai Zhang; Zhongxian Gu; Yu Lin; Jianjun Zhao "AutoFlow: An automatic debugging tool for AspectJ software" ICSM 2008. IEEE International Conference on 2008, pp. 470 – 471.
[9] Rossler, J. "How helpful are automated debugging tools?" User Evaluation for Software Engineering Researchers (USER), 2012 IEEE Conference Publications, pp. 13 – 16.
[10] Chi-Neng Wen;shu-hsuan Chou;chih Chen ;tien-fu chen. "NUDA: A Non-Uniform Debugging Architecture and Nonintrusive Race Detection For Many core system" IEEE transaction, vol.61, 2012, pages.199-212.
[11] Engblom, J. "A Review of Reverse debugging" System, Software, SC and Silicon Debug Conference (S4D), 2012, pp. 1 – 6.
[12] Chris parnin and alessandro orso, "Are automated debugging techniques actually helping programmers" ISSTA' July 2011
[13] Jorge ressia, Alexandre Bergel and Oscar Nierstrasz "object centric debugging" ICSE 2012
[14] Renee McCauley, Sue Fitzgerald, Gary Lewandowski, Laurie Murphy, Beth Simon, Lynda Thomas and Carol Zander "Debugging: a review of the literature from an educational perspective" June 2008
[15] K. Glerum, K. Kinshumann, S. Greenberg, G. Aul, V. Orgovan, G. Nichols, D. Grant, G. Loihle, and G. Hunt, "Debugging in the large: ten years of implementation and experience," Proc. SOSP, 2009, pp. 103-116.
[17] Noor Fazlida Mohd Sani, Noor Afiza Mohd Arifin and Rodziah Atan "Design of object-oriented debugger model using unified modeling language" JCSSP 2013, pp 15-18.
[18] Potanin, A., Noble, J., Biddle, R.: Snapshot query-based debugging. In: Proceedings of the 2004 Australian Software Engineering Conference (ASWEC'04), Washington, DC, USA, IEEE Computer Society (2004) 251
[19] P. Iyenghar, C. Westerkamp, J. Wuebbelmann, E. Pulvermueller, A Model Based Approach for Debugging Embedded Systems in Real-time, in 10th
[20] Jorge Ressia, Alexandre Bergel, Oscar Nierstrasz "Object-Centric Debugging" ICSE 2012, IEEE, Zurich, Switzerland.
[21] D.M.Thakore, Tanveer S Beg "An Automatic Debugging Tool Extension for Object Oriented Software" IJSCE.