Run Time Evaluation by Using Object Oriented Debugging Tool
Piyush Bhatia¹ Rahul Sehrawat²
¹,²Maharishi Dayanand University, Rohtak

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.

Key words: Debugging, source code, programming, evolution, errors

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 analyze 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 object.

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 helpful to have previous object states and object reference flow information at hand during debugging.

II. RELATED WORKS

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. 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. In Back-in-time debuggers approach; these are extremely useful tools for identifying the causes of bugs. 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.

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.

A. 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 deem 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.

B. Related work shortcoming

Studying and analyzing 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.

III. 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 Bifrost 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.

A. 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.

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.

B. System Workflow:

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

IV. 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.

REFERENCES