ISSN:2249-5789
Mohd.Ishrat et al , International Journal of Computer Science & Communication Networks,Vol 2(5), 615-617

Comparison of Static and Dynamic Analysis for Runtime Monitoring

Mohd. Ishrat1, Manish Saxena2 and Dr. Mohd. Alamgir3

1 Research Scholar, Singhania University,
Pacheri Bari, Jhujhunu, Rajasthan, India. Pin - 333515
ishratgzp@gmail.com

2 Asst. Professor, MCA Department, FGIET,
Raebareli, UP, India. Pin - 229001
manish.mohan.saxena@gmail.com, URL : www.manishsaxena.in

3 Asst. Professor, Singhania University, CSE Department,
Pacheri Bari, Jhujhunu, Rajasthan, India. Pin - 333515
geeralam@yahoo.com

Abstract

The run-time verification of security properties (Integrity, Availability and Confidentiality) received increased attention from researchers. In particular a security property that relate to information that is made available by end users is achievable only to a limited degree using static and dynamic verification techniques. The more sensitive the information, such as banking data, government intelligence or military information being processed by software, the more important it is to ensure the confidentiality of this information.

This paper aims to compare the static and dynamic methodology for run time monitoring. This paper will help to carried out the actual performance of these two different methodologies and their performance in different conditions. The objective of this paper is to find the better solution for run time monitoring using static or dynamic analysis.

Keywords- Static Analysis, Dynamic Analysis, Dynamic Vs. Static Analysis.

1. Introduction

Software systems play an important role in our economy, government, and military. This requires a constant need to understand a program's behavior by exploring its correctness during the runtime. Traditional methods, testing and verification are not enough to guarantee that the current execution of a running system is correct. There are possibilities to introduce errors into an implementation of the design that has been verified. Testing may scale well and check implementation directly, but it is mostly informal and it does not guarantee completeness. Verification is formal and may guarantee completeness, but it does not scale well and deals mostly with design instead of implementation. An approach of continuously monitoring and checking a running system with respect to particular specifications is required to use for filling the gap between these two approaches.

Runtime software monitoring has been used for profiling, performance analysis, software optimization as well as software fault-detection, diagnosis, and recovery. Software fault detection provides evidence whether a program behavior complies with specified properties during program execution.

There are two approaches that can be taken into account for the program understanding:

1.1 Static Analysis

It examines program code and reasons over all possible behaviors that might arise at runtime. Compiler optimizations are standard static analysis. Typically, static analysis is sound and conservative. Soundness guarantees that the analysis result is an accurate description of the program's behavior, no matter on what inputs or in which environment the program is run. Conservatism means reporting weaker properties than may actually be true. The weak properties are guaranteed to be true, preserving

soundness, however may not be strong enough to be useful. Static analysis usually uses an abstracted model of a program state. On one hand, such an abstraction may cause losing some information. On the other hand, it provides a model which is compact and straightforward for manipulation.

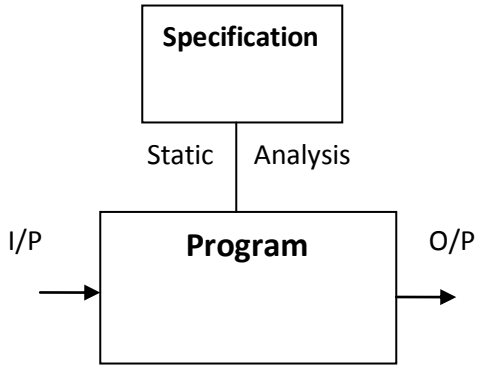

Fig 1 Static Analysis

1.2 Dynamic Analysis

It operates by executing a program and observing the executions. Standard dynamic analysis includes testing and profiling. Dynamic analysis is precise because no approximation or abstraction needs to be done.

In other words, the analysis can examine the actual and exact runtime behavior of a program. Dynamic analysis can be as fast as program execution. Some dynamic analyses run quite fast, but in general, obtaining accurate results entails a great deal of computation, especially when large programs are analyzed. The disadvantage of dynamic analysis is that its results may not be applicable for future executions.

There is no guarantee that the test suite, over which the program is run, can cover all possible program executions.

2. Static vs. Dynamic Analysis

An important difference in software analysis is whether this is done before or during runtime. Static analysis will be used to refer to all the techniques (including theorem provers and model checkers) used to verify a program for all possible execution paths before the actual execution. On the other hand, the term dynamic analysis is usually used to refer to all the techniques used to verify the system's properties for one particular execution trace obtained by executing the program. Being able to verify properties for all possible executions paths makes static analysis a very desirable objective.

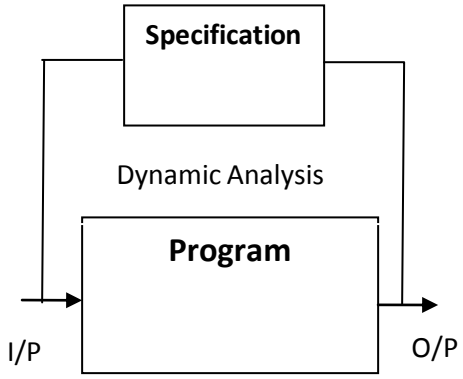

Fig 2 Dynamic Analysis

However, for the algorithm to be tractable when applied to systems of a practical magnitude, static analysis depends on having a decidable domain. Various abstraction and reduction techniques have been proposed to scale up static analysis, but full verification of large-scale software systems is still largely unattainable. In contrast, using dynamic analysis, one checks that a given system property holds along a particular execution path. This is particularly useful to ensure that at no time during the execution of the system, are any of the system properties violated.

Conversely, it can also identify execution paths along which the properties to be verified are not satisfied. Essentially, dynamic analysis links the abstract specification to the actual concrete implementation. Thus, dynamic analysis can be used as a protection from potential faults at runtime, by implementing monitors to react to any property violations encountered. Another

Motivation for using dynamic analysis is that certain information is only available at runtime. Furthermore, behaviors of the system may possibly depend on the environment where the system is running.

Although, there is this fundamental difference between static analysis and dynamic analysis, ways have been proposed in which these two approaches can complement each other by exploiting the benefits of one to aid the other. The difference between static analysis and dynamic analysis is over-emphasized. Complementarily can be achieved by applying both approaches and taking advantage of the "soundness" of static analysis while also benefiting of the "efficiency and precision" of dynamic analysis. Soundness refers to the fact that any statically verified property holds for any possible path.

Dynamic analysis is efficient because it needs to verify only a single path and precise because the

properties are verified on the actual implementation and not on an abstracted version of the system. The purpose of this integration is to help the developers in identifying errors in the system by showing concrete executions where the errors arise.

Another way in which static analysis and dynamic analysis can complement each other is for test-case generation. Static analysis can be used to "intelligently" generate test cases for a dynamic analysis tool to find errors during testing.

This is possible because static analysis provides the structure of the program and thus it is easier to produce test cases which cover most of the program paths.

3. Combining Static and Dynamic Analysis

As both the analysis, static as well as dynamic analysis is necessary for run time monitoring for that we have to implement both in well organized manner to use their advantages efficiently. Both the approaches have their own advantages and efficiency. We have to implement them in a combine manner so that the results may be good and efficient.

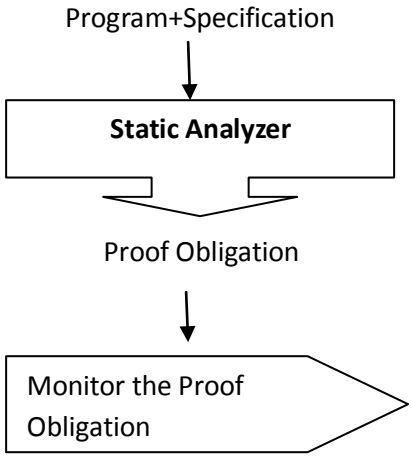

Fig 3 Static and Dynamic Analysis

4. Conclusion

As a result it can be said that both static and dynamic analysis is necessary for runtime monitoring. As both of the approaches have their own importance. One approach works before execution of program on the other hand another approach works during the runtime of program. Dynamic Analysis works on the different execution path traces by the Static Analysis before execution. In static analysis we trace different execution path of program but in case of dynamic analysis we monitor the program during runtime in single path which makes it more efficient in compare to static analysis.

References:

[1] J. S. Bradbury, J. R. Cordy, and J. Dingel. An empirical framework for comparing effectiveness of testing and property-based formal analysis. In Proceedings of the ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering, pages 2-5, 2005.

[2] A. G. M. Cilio and H. Corporaal. Global program optimization: register allocation of static scalar objects. In Proceedings of 5th Annual Conference of the Advanced School for Computing and Imaging, pages 52-57, 1999.

[3] R. Morgan. Building an Optimizing Compiler. Butterworth-Heinemann, Boston, Massachusetts, 1998.

[4] M. D. Ernst. Static and dynamic analysis: synergy and duality. In Proceedings of the ICSE Workshop on Dynamic Analysis, pages 24-27, 2003.

[5] A. M. Memon. Employing user profiles to test a new version of a GUI component in its context of use. Software Quality Journal, 14(4):359-377, 2006.

[6] N. Ubayashi and T. Tamai. Aspect-oriented programming with model checking.In AOSD '02: Proceedings of the 1st international conference on Aspect-oriented software development, pages 148–154, New York, NY, USA, 2002.ACM Press.

[7] K. Zee, V. Kuncak, M. Taylor, and M. Rinard. Runtime checking for program verification. In RV'07, 2007.

[8] A. Goldberg and K. Havelund. Automated runtime verification with eagle.In MSVVEIS, 2005.

[9] B. Finkbeiner and H. Sipma. Checking finite traces using alternating automata. Form. Methods Syst. Des., 24(2):101–127, 2004.

[10] I. Lee, H. Ben-Abdallah, S. Kannan, M. Kim, O. Sokolsky, and M. Viswanathan. A monitoring and checking framework for run-time correctness assurance. In Korea-U.S. Technical Conference on Strategic Technologies, 1998.

[11] J. Sifakis, S. Tripakis, and S. Yovine. Building models of real-time systems from application software. Proceedings of the IEEE, 91:100–111, 2003.

[12] M. D. Ernst. Static and dynamic analysis: Synergy and duality. In WODA 2003: ICSE Workshop on Dynamic Analysis, pages 24–27, Portland, OR, May 9 2003.