Enhancing Exception Handling and Security in Object-Oriented by using Aspect-Oriented Programming

Sarita Singh[1] Prof. A. K. Sisodia[2]
[1,2]Department of Computer Science Engineering
[1,2]Galgotias University Greater Noida, Uttar Pradesh

Abstract— Exception handling is a mechanism that provides an efficient way to handle errors in software applications. In object-oriented programming handling of exceptions constitute problems of code tangling and repetitive code. A spoofed parent class in place of parent class may be used to manage the exception. If we implement security concern using Object-Oriented Programming, we face not only the problem of code scattering and code tangling but also there is a problem of unreliable implementation of security concern. Traditional techniques have lack of modularity to implement concerns like exception handling and security that creates many difficulties in the development and maintenance for the concerns. In order to centralize security and exception handling that increase its modularity and reusability and provide intensified verification and security in accessing of resources, we put forward existing methodology for Aspect-Oriented security and exception handling into an Object-Oriented System.

Key words: object-oriented, aspect-oriented, exception handling, security

I. INTRODUCTION

For better software quality reusability is an essential feature of any programming language. In object-oriented programming by using inheritance the common features and behaviour of parent class can be obtained into child class so three is no need to redefine in the sub class. Through its attribute of polymorphism child class legalized to rewrite the operations of their parent class according to their requirements with less coding. By using these features of object-oriented programming expanded software reusability and resulting decrease the required amount of coding [1].

Implementation of exception handling and security concerns in object-oriented programming must be done for each object resulting repetition of code and security cannot be organized centrally. Additionally misplacement of code because of implementation code for both normal behavior and exception handling are intertwined mutually. Aspect-oriented programming renders a key to solve these problems by uniting associated functionality into concerns that crosses the boundary of module. Ongoing object-oriented programming is not able to provide secure exception handling; there are many possibilities for hackers to attack. One kind of security problem is spoofed parent class. When a raised exception is not handled in the child class then it must be propagated to the parent class where hackers may be used spoofed parent class to catch unhandled exception and they are able to access information of the system. Another security issue is information leakage and hackers may be used this leaked information to plan an attack. A requisite error handling technique can be used to prevent this information leakage which can disclose internal information [2].Exceptions are run time environment errors and static analysis cannot be used to detect exceptions. A large software system expends tremendous development resource to handle exceptions. System analyst is responsible to take decisions and to determine exceptional situations. Occurrence of Exception in the system designates the operations will not be accomplished successfully. Through exception handling techniques these exceptions can be either recovered or ignored [3].

In this paper we present a method to improve exception handling using aspect-oriented programming. In order to reduce code space which support software usability certain aspects are centralized by using this methodology that render uniform security and exception handling.

II. RELATED WORK

The concept of exception handling was introduced three decades ago as a systematic way to handle programming faults [6], [7].To avoid problems of code tangling and code scattering current programming are imposed to manage normal code and exception code separately. Object-oriented paradigm is accepted as the most widely used technique for programming [4]. Object-oriented approach includes binding of data and methods into classes that hides implementation details. System scalability is critically affected by this approach. A logging operation in this approach requires to call related methods scattered in different objects within the program resulting duplication and disruption of code that increase cost of maintenance and difficult to understand. As the size of system increases it is very difficult to enlarge security ability. Specifically in this approach scalability is a very important concern to implement security [8]. Programming language performed a large amount of work to achieve the goal of providing reinforce for exception detection and handling. Hoare introduced the concept of Contracts consists of precondition, post condition and invariants that are used to determine how a computational entity is used and what result is expected from the computational entity [10].

A large software system uses large amount of development resource to handle exceptions. A process of identifying exceptions that occurred when a system is in functioning and to develop a handler to handle exceptions is mostly a manual process. When required exceptions are missed to handle appropriately lead to catastrophic results [8]. Programming languages provide direct construct to handle exceptions but by using these constructs the code for normal functionality and exception functionality are entwined resulting code tangling. In the hierarchy of classes developers cannot reuse parent class exception handler into child class without redefining. Reusability of these handlers is complicated and addition of these handler's function inside functional code resulting loss of abstraction [3]. In case of wrong data type in user interface that occurs an exception and give additional information to hackers [11], [1]. When an exception is generated by a sub class which is not managed in that class and transmitted to the super class where this exception may be handled by hackers using spoofed parent class and able to access execution of program. To avoid the problem of spoofed parent class all exception handling is centralized in security aspect.

Shah recommends the aspect-oriented security framework where security concerns are separated from application to implement security solutions constantly and globally [4].

Lippert insists separation of normal and exceptional code. By this separation intercepts the problem of reusability, documentation, reduction of abstraction and excessive of exceptional code can be reduced [3].

The main motivation behind aspect-oriented programming is to separate cross-cutting concerns by modularizing these concerns. Cross-cutting concerns are generic functionality includes logging, error handling and security. Concerns that crosses modular boundaries can be comprehensively enclosed into modules by using aspect-oriented programming. Join points are fix points inside different modules defined by programmer on which advice may be implemented [11]. Separation of concerns by using aspect-oriented programming facilitate reusability of exception handler and faster implementation [1].

III. APROACH

In existing approach a methodology is devised for exception handling that construct more durable software for reusability. For this they deal with some security concerns built-in exception handling within object-oriented programming by using aspect-oriented programming.

This approach is implemented on an existing JWAM framework which is developed in Java for interactive business applications at Hamburg University. Its component includes client/server computing and distribution to cover all requirements for business applications. Advantage of this approach regarding to LOC and exceptional behaviour unplug ability.

JWAM contains 600 classes and implemented by using Contracts methodology that confirms methods would not be misused by callers. If any contracts is broken then generates a run time error [3].In his work they handle five exceptions by using AspectJ that extremely reduce exception handling code.

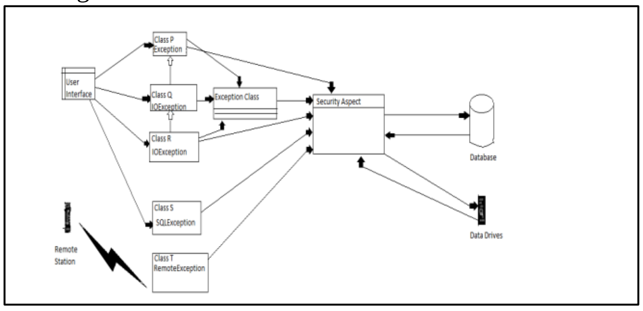

FIG. 1:

Figure 1 be composed of small object-oriented system that has been improved by the aspect-oriented programming to handle all exceptions inside security aspect nevertheless of where they arise. All exceptions are redirected to security aspect for handling and verification. This system include a user interface that communicate with other objects. Class P, class Q and class R implement inheritance and an exception that arise in child class will not be handled by that class and propagated to the parent class. An Exception class which is user defined is used to handle redirection of exceptions. Class T is called remotely. All exceptions include IOException, SQLException and remote connection error were redirected to security aspect for handling and verification. By providing a centralised control in security aspect for all exception handler the required code for exception handling was extremely decreased. It provide a clean separation of normal behaviour from exception handling code.

REFERENCES

[1] Richard, Evans "Aspect-oriented security and exception handling within an object-oriented system" 2011 35th IEEE.

[2] Faisal, Nazir, Mustafa "Safety and Security Framework for Exception Handling in Concurrent Programming", 2013 IEEE.

[3] Martin Lippert, C. Lopes "A Study on Exception Detection and Handling Using Aspect-Oriented Programming", ICSE 2000.

[4] Shah, V., F. Hill "An Aspect-Oriented Security Framework", DISCEX, 2003.

[5] Liu, H., Zhang, J., Wang, L. "The Research and Application of Web-Based System with Aspect-Oriented Features ", 2nd International Conference on Computer Engineering and Technology, 2010.

[6] J. B. Goodenough, "Exception handling: issues and a proposed notation," ACM, Dec. 1975.

[7] F. Cristian, "Exception handling and software fault tolerance," IEEE, Jun. 1982.

[8] R. Millham and E. Dogbe, "Aspect-oriented security and exception handling within an object oriented system," in Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual, July 2011.

[9] J. B. Goodenough: "Exception Handling: Issues and Proposed Notation", in Communications of the ACM, 1975.

[10] C. A. R. Hoare: "An Axiomatic Basis for Computer Programming, in Communications of the ACM", October 1969.

[11] Sinn, R. "Software Security Technologies: A Programmatic Approach", 2008.

[12] AspectJ Web Site, <http://www.aspectj.org/>.

[13] C. Lopes, G. Kiczales: Recent Developments in AspectJ, in ECOOP '98 Workshop Reader, Springer-Verlag, 1998.

[14] JWAM framework Web Site, <http://www.jwam.de/>.

[15] B. Stroustrup, "The C++ Programming Language, Third Edition, 3rd ed.1997.

[16] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M.Loingtier, and J. Irwin, "Aspect-oriented programming," in ECOOP, 1997.

[17] Mariano Ceccato," Migrating Object Oriented code to Aspect Oriented Programming".

[18] D. Binkley, M. Ceccato, M. Harman, F. Ricca, and P. Tonella."Automated refactoring of object oriented code into aspects", (ICSM 2005). IEEE Computer Society, September 2005.

[19] D. Binkley, M. Ceccato, M. Harman, F. Ricca, and P. Tonella.”Tool-supported refactoring of existing object-oriented code into aspects”. IEEE 2006.
[20] M. Ceccato. “Migrating Object Oriented code to Aspect Oriented Programming”. PhD Thesis, University of Trento, Italy, December 2006.