CHANGE IMPACT ANALYSIS APPROACH IN A CLASS HIERARCHY

Khine Khine Oo
University of Computer Studies,Yangon, Myanmar

Keywords: Impact analysis, Class Hierarchy, Program Slicing, Maintaining, Debugging, Testing

Abstract: Change impact analysis is a technique for determining the potential effects of changes on a software system. As software system evolves, changes made to those systems can have unintended impacts elsewhere. Although, object-oriented features such as encapsulation, inheritance, polymorphism, and dynamic binding contribute to the reusability and extensibility of systems. However, we have to face the more difficult to identify the effected components due to changes because there exits complex dependencies between classes and attributes. In this paper, we propose change impact analysis approach for a class hierarchy. Our approach is based on the program slicing techniques to extract the impact program fragment with respect to the slicing criterion of change information but aim to minimize unexpected side effects of change. We believe that our impact analysis approach provides the software developer in their maintaining process as well as debugging and testing processes.

1 INTRODUCTION

Most of the programmers are now facing with the problem of debugging and maintaining a large program. Software maintenance has been recognized as the most costly and difficult part of software development. The two most expensive activities in software maintenance are understanding the system and evaluating the affects of any proposed change (Barros,1995) Given a reasonable understanding of the system, a key objective in maintenance is to understand how a proposed change in the implementation will affect the system. As a result, software developers need mechanisms to understand how a software change will impact the rest of the system.

Software change impact analysis is a technique for predicting the potential effects of changes before they are made, or measuring the potential effects of changes after they are made, with the goal of reducing the costs and risks. Applied before modifications, impact analysis can help maintainers estimate the costs of proposed changes and select among alternatives. Applied after modifications, impact analysis can alert engineers to potentially affected program components requiring retesting, thus reducing the risks associated with releasing changed software.

Program slicing is useful in software re-engineering, such as program understanding, impact analysis and business rule extraction. Object-oriented (OO) paradigm introduces concepts such as encapsulation, inheritance, polymorphism and dynamic binding. Such features tend to encode much of the complexity in the relationships among classes and attributes. This makes the more difficult to identify the components to be affected by the changes. Moreover, the complex relationships among classes make it difficult to anticipate and identify the ripple effects of changes (Kung,1994). So the ripple effects of object oriented systems far more difficult to control than in procedural system (Weiser,1984). These results tend to address the problem of change impact in a class hierarchy based on program slicing techniques.

We propose analyzing change impact approach for a class hierarchy. Our approach is based on program slicing techniques. We also propose a new hierarchical program slicing techniques, namely statement slicing, method slicing, class slicing and package slicing. Our approach aims for the software developer and maintainer to identify potential impacts before making a change. Our approach also permits to extract the program fragments which are affected by the changes.

The rest of our paper is organized as follows; Section 2 reviews the related work of program slicing and impact analysis. Section 3 defines an overview of change impact model, Section 4 explains change impact analysis in a class hierarchy and Section 5

Khine Oo K. (2005).
CHANGE IMPACT ANALYSIS APPROACH IN A CLASS HIERARCHY.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 465-468
DOI: 10.5220/0002555604650468

gives our propose hierarchical program slicing techniques. Our conclusion is given in Section 6.

2 RELATED WORK

Research in program slicing can be divided into three broad categories: applications, criteria and algorithms. Ottenstein et al (Ottenstein,1984) used graph reachability algorithm on a Program Dependence Graph to compute a slice. Horwitz (Horwitz,1990) developed the System Dependence Graph and a two-phase graph reachability algorithm on the System Dependence Graph to compute interprocedural slice. Some researchers have extended the System Dependence Graph to represent OO programs, such as Larsen and Harrold's (Harrold,1996) used Class Dependence Graph for representing single classes, derived classes and interacting classes. Class Dependence Graphs represent classes in the system and it can be reused in constructing other class dependence graphs or system dependence graphs. Chen (Chen,1997) proposed two types of program slices, namely state and behavior slices, by taking the dependence of OO features into consideration. Fan (Fan,2001) defined the hierarchical slicing model based on the stepwise slicing algorithm and graph reachability algorithm to slice Java programs.

There are many impact analysis approaches applied in object oriented software. Some approaches are determined by a set of affecting changes for software maintainers. Other approaches are intended for software developers able to know small changes can ripple throughout the system which become unintended impact elsewhere. Pfleeger (Pfleeger,1990) recognized impact analysis as a primary activity in software maintenance and present a framework for software metrics that could be used as a basic for measuring stability of the whole system. Kung et al (Kung,1994) described an algorithm to identify the impacted parts of the system by comparing the original system against the new modified version.

3 CHANGE IMPACT MODEL

One way to accessing the changeability of an OO system is by performing a change impact analysis. When a change to a system is considered, it is necessary to identify the components of the system which will be impacted as a result of that change. This is to ensure that the system will still run correctly after the change is implemented.

The impacts of changes depend on two main factors. One factor is the type of change and the other factor is the nature of the links involved (Chaumun,1999). Impact analysis can be partitioned into two parts: traceability based analysis and dependencies based analysis. Our model focuses only on the dependencies based analyses. There are a number of different dependencies between the components. If one of the given component is subject to change, we are interested in knowing which other parts in the rest of the system will be affected by this change. In this paper, impact on changes to classes, member functions and data members in a class hierarchy are to be analyzed.

A class contains class members and a class member composed of member functions and data members. The relationships are shown in Figure 1. When a class member changes, it could impact other components in a class hierarchy. For example, a change to a variable type has an impact on all classes referencing that variable. Similarly, the scope of the method is changed from public to protected, the classes that invoke the method will be impacted, with the exception of the derived classes.

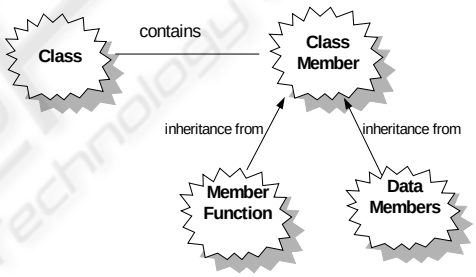

Figure 1: Class component graph

4 CHANGE IMPACT ANALYSIS IN A CLASS HIERARCHY

We call impact of a change the set of classes that require correction as a result of that change. Our approach implies that one single change is considered at a time.

Let $\tau = \{\tau_1, \tau_2, \tau_3, \ldots\ldots\ldots, \tau_m\}$ be the set of class hierarchies and $\tau_i = \{c_1, c_2, c_3, \ldots\ldots\ldots, c_n\}$ be the set of classes in a class hierarchy τ_i ,where $1 < i \leq m$, $\tau_i \in \tau$.

Definition 1

Let $D = \{d_1, d_2, d_3\}$ be the set of all dependencies of τ_i such that d_1 stands for association, d_2 stands for aggregation and d_3 stands for inheritance respectively.

And d_i is the dependency between two classes c_g and c_h, where $c_g, c_h \in \tau_i$ and $d_i \subset D$. If c_g is changed then c_h is need to considered to be changed. So c_h is said to be an impact related class of c_g.

Definition 2

Let c_k be a class in a class hierarchy τ_i and let $M = \{m_1, m_2, m_3, ...m_n\}$ be the methods or member functions in c_k where $1 \le k \le n$ and $c_k \in \tau_i$. If we consider a change in a method m_j then the other methods which invokes m_j will be impacted and it needs to be consider changed.

Definition 3

Any datum (ie., a global variable, a local variable or a class data member) can be changed by its type as well as can be changed by addition or deletion its variable components. Let $V = \{v_1, v_2, v_3, ..., v_j\}$ be the set of data members in a class c_k, where $v_h \in c_k$, $v_i \in c_k$ and $v_h \neq v_i$. If we consider a change in a data member/variable in a class c_k then the other classes referencing this data member/variable becomes impacted.

Proofs:

The major task of class hierarchy impact analysis is to find all affected classes due to the changes of inner class relationships. Taking into consideration the difference relationships between classes are inheritance, aggregation and association relationships. In figure 2, there exits two relationships between classes, namely inheritance and aggregation relationships. If we delete the Student class, it is obvious that there will become impacts on Part time Student class and Full time student class because the relationship between the classes is aggregation. It is obviously notice that, change impact analysis depends on the changes between classes in definition 1.

As for the definition 2, consider a change in the scope of show_marks() method in Student class is changed from public to protected, the Professor class which invokes this method will be impacted. Thus, we can say that impact of change depends on a change in the scope of the member function.It is simply see that, if we delete variable marks in Student class, the methods named show_marks() and calculate_marks() will be impacted because theses methods refer the variable marks. Thus, we can say that a change in a variable has an impact in all the classes referencing that variable as defined in definition 3.

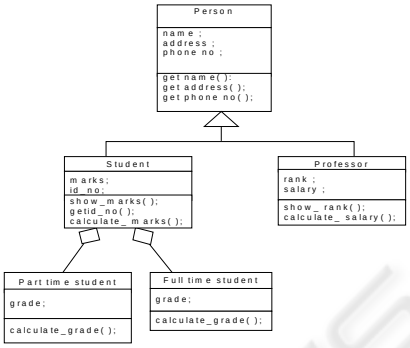

Figure 2: Class dependencies relationship

Assume that we want to extract out the impacted components in a class hierarchy, we have to identify the change criterion, whether a change is in a class or function member and data member. First, the impacted class set is initialized which includes the set of changes classes. Then the impacted function member set and impacted data member set of each class is initialized.

5 PROPOSE PROGRAM SLICING TECHNIQUE

The main purpose of program slicing is to take out the interested parts from a large system. In this section, we propose a new hierarchical program slicing techniques with respect to the slicing criterion. The proposed slicing techniques are statement slicing, method slicing, class slicing and package slicing. After analyzing an impact part in a class hierarchy, then we use the appropriate slicing criterion to extract the impacted parts from a program.

Given a java program J, we denote P_J, C_J, M_J, S_J be the set of packages, classes , methods and statements defined in program J, respectively. $\forall s \in S_J$, we get the set of statements affecting or affected by a variable V_S, where V_S is the variables set in statement S. Similarly, we get a set of methods, classes and packages by mapping $M(s)$, $C(s)$, $P(s)$, respectively.

Every program slice can have two directions. backward and forward directions. A backward slice is a slice that contains a set of statements affecting on a variable v in a slicing criterion. A forward slice is a slice that contains a set of statements affected by a variable v in a slicing criterion. We denote backward slice as $\mathcal{BS}(s)$ and forward slice as a $\mathcal{FS}(s)$. Let $H(s) = \mathcal{BS}(s) \cup \mathcal{FS}(s)$.

Now, we say that a method $m \in M_J$ that can affect or affecting a variable set V is $\exists s \in S_J, v \in V.M(s) = m \wedge H(s)$ and a class $c \in C_J$ that can affect or affecting a variable set is $\exists s \in S_J, v \in V.C(s) = c \wedge H(s)$ and a package $p \in P_J$ that can affect or affecting by a variable V is $\exists s \in S_J, v \in V.P(s) = p \wedge H(s)$.

Thus , all the methods that can affecting on or affected by a variable set V is given by $\sum_{M,V} = \{m \mid \exists s \in S_J, v \in V.M(s) = m \wedge H(s)\}$. All the classes that can affecting on or affected by a variable set V is given by $\sum_{C,V} = \{c \mid \exists s \in S_J, v \in V.C(s) = c \wedge H(s)\}$ and all packages that can affecting on or affected by a variable set V is given by $\sum_{P,V} = \{p \mid \exists s \in S_J, v \in V.P(s) = p \wedge H(s)\}$.

Given a program J , The slicing criterion for a program J with respect to a variable V in a statement s is a turple $\zeta = (s_\xi, v_\xi)$,where $s_\xi \in S_J$ and $v_\xi \in V_J$.

5.1 Statement Slicing

The statement level concerns about program statements, control predicates and different kinds of variables. Given a program J and a slicing criterion ζ , we get the statement slicing for J with respect to the slicing criterion $\zeta = (s_\xi, v_\xi)$, where $V \in v_\xi$.

5.2 Method Slicing

The method level contains the methods or variables defined in a class. Given a program J and a slicing criterion ζ , we get the method slicing for J with respect to the slicing criterion is $\zeta = < M(s), V >$, where $V \in v_\xi$.

5.3 Class Slicing

The class level composed of a set of top level classes or interfaces. Given a program J and a slicing criterion ζ , we get the class slicing for J with respect to the slicing criterion is $\zeta = < C(s), V >$, where $V \in v_\xi$.

5.4 Package Slicing

Java programs are composed of a set of packages, whose naming structure is hierarchical. Given a program J and a slicing criterion ζ , we get the package slicing for J with respect to the slicing criterion is $\zeta = < P(s), V >$, where $V \in v_\xi$.

6 CONCLUSION

Many software maintainers are used to extract the affected portion of the program that he wants to debug or test this program segment. It was sound for the procedural base language but it will be difficult for the object based programming language such as java. In our system, we proposed new program slicing techniques for object oriented java programs based on their hierarchical structure. Statement slicing enables the user to inspect the effects of the particular statement on the slicing criteria. Method slicing identifies the methods that are directly effects on the given slicing criteria. Class slicing can make the data members and statements in methods of the class that might affect the slicing criteria. Package slicing enables user to extract the sub packages, import packages that also affect the slicing criteria. Base on the resulted slice, we can easily examine the affected portions of the program and can also be use in the software metric, testing and maintenance processes.

REFERENCES

Barros, S., Bodhun, T., Escudie, A., Quille, J., and Voidrot, J., 1995. Supporting impact analysis: A semi-automated technique and associated tool.

Chaumun, A., Kabaili, H., Keller, R., and Lustman, F.,1999.A change impact model for changeability assessment in object-oriented systems.

Chen, J., Wang, F., and Chen, Y.,1997. Slicing Object Oriented Programs. *Proceeding of IEEE Conference*.

Fan, J., 2001. JATO: Slicing Java program hierarchically, TUCS Technical Reports , Turku, Finland.

Horwitz, S., Reps, T., and Brinkly, D., 1990. Interprocedural slicing using dependence graph. *ACM Transaction on Programming Languages and System*.

Karstu, S., 1994. An examination of the behavior of Slice-Based Cohesion Measures,Michigan Technological University.

Kung, D., Gao, J., Hsia, P., Wen, F., Toyoshima, Y., and Chen, C., 1994.Change impact identification in object-oriented software maintenance. *In Conference on Software Maintenance*, Piscataway,NJ, IEEE.

Larsen, L., and Harrold, M., 1996. Slicing Object oriented Software. In *the 18th International conference on Software Engineering.*

Ottenstein, K., and Ottenstein, L., 1984. The program dependence graph in a software development environment ,*ACM Software Engineering Notes.*

Peeger, S., and Bohner, S.,1990. A framework for software maintenance metrics. *IEEE Transactions.*

Weiser, M., 1984. Program Slicing", IEEE Transaction on Software Engineering.