Class hierarchy method to find Change-Proneness

Prof.Malan V.Gaikwad
Department of Computer Engineering SCOE, pune.
M-tech IT, bharati Vidyapeeth University COE, pune.
Pune, India

Prof.Aparna S.Nakil

Department of Computer Engineering
Singhgad College Of Engg. pune.India

Prof.Akhil Khare
Department of Information Technology
Bharati Vidyapeeth University COE, pune.
Pune, India

***Abstract*—Finding Proneness of software is necessary to identify fault prone and change prone classes at earlier stages of development, so that those classes can be given special attention. Also to improves the quality and reliability of the software. For corrective and adaptive maintenance we require to make changes during the software evolution. As such changes cluster around number of key components in software, it is important to analyze the frequency of changes in individual classes and also to identify and show related changes in multiple classes. Early detection of fault prone and change prone classes can enables the developers and experts to spend their valuable time and resources on these areas of software. Prediction of change-prone and fault prone classes of a software is an active topic in the area of software engineering. Such prediction can be used to predict changes to different classes of a system from one release of software to the next release. Identifying the change-prone and fault prone classes in advance can helps to focus attention on these classes.**

In this paper we are focusing on finding dependency of software that can be achieved by estimating the proneness of Object Oriented Software. Two main types of proneness are associated with OO software. Fault Proneness and Change Proneness.

Keywords-change proneness, adjacency matrix, adjancency list, dependency between classes

I. INTRODUCTION

As we know in couple of research work, where used the probabilistic approach to predict the changes in any object oriented paradigm might produce nearly accurate results for change proneness, but the process is seems to be bit lengthen. thus in our work we represent a class hierarchy method in which prediction becomes more easier and correctable as compared to the relevant methods.

In this paper we build a model for predicting the change prone classes by using class hierarchy method. in previous work of probabilistic approach they used the UML diagrams to find the dependencies between the classes and then classes are identified which are change prone in nature; but in this work we are finding the dependencies between the classes by using class hierarchy method in which we are traversing all classes, sub classes, and inherited classes and then constructing the binary trees for each class dependency. Then these BT's are merged together by using merging techniques.

In our historical approach we need to read all the class names and their contents and need to keep both in double dimensional array of two columns and n number of rows,where each class name is bind with its contents in the respective matrix cell.

If a maintenance process can identify what parts of the software are change-prone then specific remedial actions can be taken. Thus, knowing where most changes are made over time can identify key change-prone classes, key change-prone interactions, and the evolution process can focus attention on them.

II. BACKGROUND

Several researchers have proposed the use of historical data related to a software system to assist developers gain a better understanding of their software system and its evolution .Let's see some of these.

Arnold and Bohner [1] give an overview of several formal models of change propagation. The models propose several tools and techniques that are based on code dependencies and algorithms such as slicing and transitive closure to assist in code propagation. The methodology represents a system as a set of data dependency graphs, which is an effective approach for object oriented designs **Graves et al.** [2] have a slightly different goal of predicting faults, which are a subset of changes, in aged software systems. They find the change-history of the system to be a better predictor than code metrics. In that model, Graves et al. assign weights to the perceived changes, with the most recent receiving the most. These weighted values provide a trend that is used to predict the number of faults in an upcoming period.

Mockus and Weiss [4] attempt to predict faults in a software system based on information extracted from changes to the system (e.g., lines of code modified, the changed components, etc.). This approach differs from many of other the related work in the respect that it uses changes to the state of the system, rather than the state itself.

Girba et al.[3] recentaly proposes an approach to summarize the changes in the history of a system that can offer a solid basis for starting a reverse engineering effort. The methodology consists of identifying the classes that were changed the most in the recent history and at the same time checking whether the same classes are among the most changed ones in the successive versions. However, only the addition or removal of methods is considered as changes. **Arisholm et al**.[6] investigate the use of dynamic coupling measures as indicators of change proneness. Their approach is based on correlating the number of changes to each class with dynamic coupling measures and other class-level size and static coupling measures.

III. PROPOSED MODEL

In this approach we are concentrating not only the classes but also the class components.

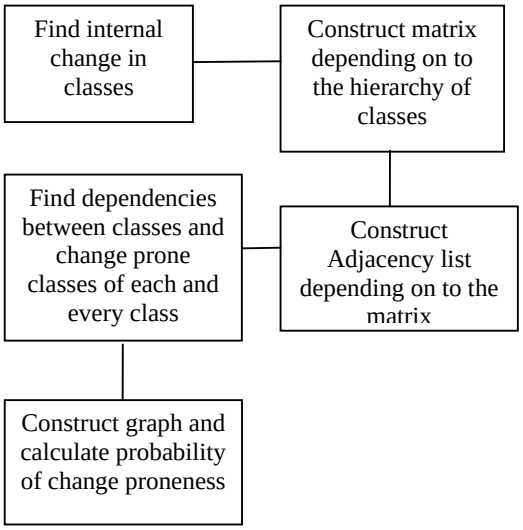

Figure.1 Proposed model of change-Proneness

Proposed method involves following Analysis methods-

1. Find the internal changes in class
2. Find the dependencies between classes
3. Find the adjacency matrix
4. Find the adjacency list depending onto the adjacency matrix
5. Find the graph of dependency list
6. Find the change proneness of a class(using probability)
7. Find the change prone classes

A. Internal Change

A class may change by a programmer while implementing or may change by designer while designing or may change by a person involved in that project.once a class or contents of class changed then that becomes a internal chage;that means change is local to that class only it will not affects on any other class which is depending or inheriting from that class. So it is important to find that whether the change made by person is internal change of that class or external chage. When the change made in class is affecting to other class then it is external change to that class.

The first step in this project is to find whether the internal change is occurring in class or not. If the change is internal then there is no any change prone classes in project .But if affecting a class due to external change then we need to find the change prone classes.

B. Adjacency Matrix

The model uses class hierarchy method to find the change prone classes. it needs to find the class dependencies between number of classes to find the dependency between classes, matrix method is used in previous work number of different technologies and algorithms are used which are bit complex. This class hierarchy method is easier and faster. The model uses the matrix method to find the class hierarchy and dependency.

Adjacency Matrix Algorithm
Start
Take array for matrix creation
Read classes
Find class name as a substring in all classes
If substring found then
Place 1 in array as matrix component
Otherwise place 0 in matrix
Display matrix
End

Figure.2 Adjacency Matrix algorithm

We can use number of ways to represent the graphs like adjacency matrix and adjacency list.In this approach I am going to consider no of different classes s nodes of graphs and then depending on to these nodes I am constructing graphs.
Suppose there are three classes A,B,C and class B is dependeing on to the class B, class C is depending on to the class A. then we can show the dependency graph just like this.

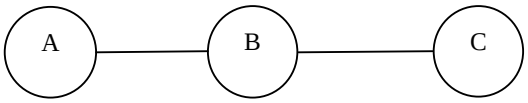

Figure.3 Dependency graph for classes A,B,C

A two dimensional array which stores the values 0 and 1 for graph is called as Adjacency matrix when the matrix a such that each cell a[i][j]=1if there is an edge i ➔ j and a[i][j] = 0 if there is not edge in between I and j. so in above example if we want to show the dependency using matrix then we can say that A➔B=1 and B➔C=1 as there is edge between them. so the graph becomes

	A	B	C
A	0	1	0
B	1	0	1
C	0	1	0

Figure.4 Adjacency matrix for fig.3

Suppose there are six classes in a project and we want to find dependency between these classes we can use matrix method here.find adjacency graph first then find matrix that shows dependency. Let's see a dependency graph for six classes shown in figure no 3. Depending onto this graph the matrix is shown in figure no.4

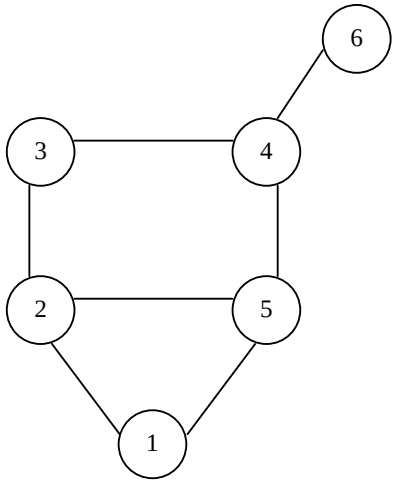

Figure.5 graph containing classes

	1	2	3	4	5	6
1	0	1	0	0	1	0
2	1	0	1	0	1	0
3	0	1	0	1	0	0
4	0	0	1	0	1	1
5	1	1	0	1	0	0
6	0	0	0	1	0	0

Figure. 6 Adjacency matrix for fig.5

An advantage of using adjacency matrix is very easy and convenient method to find the classes.

Along with these advantages there are number of disadvantages of using adjacency matrix let's see some of these.

- Adjacency matrix requires large amount of memory for storing big graphs. As we know that all graphs can be divided into two categories, *sparse* and *dense* graphs. In sparse graphs less number of edges are available than the vertices and we can show the relation of edges and vertices in sparse graph as ($|E| << |V|^2$).
- On the other hand, dense graphs contain large number of edges as compared to sparse one. we show the relation between edges and vertices in dense graph as edges are equal to square of number of vertices. Using Adjacency matrix is optimal solution for dense graphs, but for sparse graphs it is superfluous.
- the matrix requires huge efforts for adding or removing a vertex in graph. If a graph is used only for analysis only then its easy one, but to construct fully dynamic structure make it quite slow for big graphs.

Adjacency matrix is a good solution for dense graphs, which implies having constant number of vertices.

C. Adjacency List

Using adjacency list is the graph representation which is used as one of the alternatives to adjacency matrix. It requires less amount of memory and, in particular situations even can do better than adjacency matrix. For every vertex in the graph the adjacency list stores a list of vertices, which are adjacent to current vertices.

```
Adjacency List Algorithm
    Start
    Read classes from matrix
    Find substring in matrix
    If substring found then
    Place class no array as list component
    Display list
    End
```

Figure.7 Adjacency List algorithm

Let us see an example.

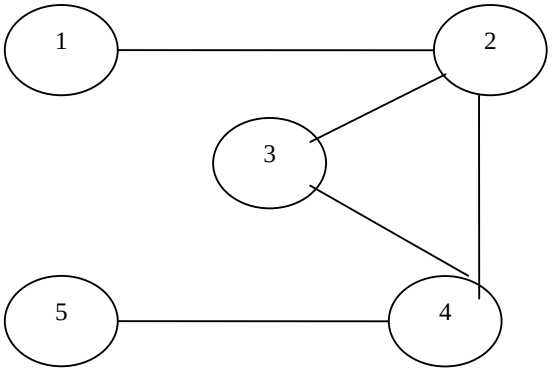

Figure.8 Dependency graph for five classes

	1	2	3	4	5
1	0	1	0	0	0
2	1	0	1	1	0
3	0	1	0	1	0
4	0	1	1	0	1
5	0	0	0	1	0

Figure.9 Adjacency matrix for fig.8

Adjacent list stores graph in more compact form, than adjacency matrix. using adjacent list we can get the list of adjacent vertices in O(1) time. Using adjacency list is a good solution for sparse graphs. Instead of using adjacency matrix we can chang number of vertices more efficiently. Let's take a following figure as an example which shows a adjacency list for above adjacency matrix. Vertices adjacent to 1 are vertices 2,for a node 2 the adjacent vertices are 1,3 and 4.similarly for 3 the vertices are 2 and 4,for 4 the vertices are 2,3 and 5 and for node 5 the adjacent vertices are 4.

1	2
2	1 3 4
3	2 4
4	2 3 5
5	4

Figure. 10 Adjacency List for Matrix in Fig.9

Although there are disadvantages of both matrix and list, but here in my proposed model I used both the adjacency matrix and the adjacency list to get the desired output.

Finally the dependency and classes which are change prone to the particular class are identified. The probability of change proneness is calculated by using probability method.

CONCLUSION

In this paper we studied two methods, matrix and list method, used for finding the proneness and dependency of classes. In previous work all data which required is taken manually and from UML diagrams. The methods used by various peoples are difficult to understand and quite complex to implement.

The proposed method of using class hierarchy method is too simple to understand and implementation. And one important thing is that I have not taken any data manually, it is calculated by this model only.

REFERENCES

[1] R. Arnold and S. Bohner, "Impact Analysis – Toward a Framework for Comparison," in Proceedings of the IEEE International Conference on Software Maintenance (ICSM), 1993, pp. 292–301.

[2] T. L. Graves, A. F. Karr, J. S. Marron, and H. Siy, "Predicting fault incidence using software change history," IEEE Trans. on Soft. Eng., vol. 26, no. 7, pp. 653–661, 2000

[3]]T. Girba, S. Ducasse, and M. Lanza, "Yesterdays Weather: Guiding Early Reverse Engineering Efforts by Summarizing the Evolution of Changes," in Proceedings of the IEEE International Conference on Software Maintenance (ICSM), 2004, pp. 284–293

[4] A. Mockus and D. M. Weiss, "Predicting risk of software changes," Bell Labs Technical Journal, vol. 5, no. 2, pp. 169 – 180, April 2000.

[5] Ali R. Sharafat and Ladan Tahvildari "Change Prediction in Object-Oriented Software Systems: A Probabilistic Approach"Journal of software vol.3,no.5 May 2008.

[6] E. Arisholm, L. Briand, and A. Foyen, "Dynamic Coupling Measurement for Object-Oriented Software," IEEE Trans.on Soft. Eng., vol. 30, no. 8, pp. 491–506, 2004.