ATLANTIS PRESS
Advances in Computer Science Research, volume 71
4th International Conference on Machinery, Materials and Information Technology Applications (ICMMITA 2016)

Alarm Correlation Research and Implementation Based on Similar Data Sources

Yi Tang[1, a] and Dahai Jin[2, b]

[1] Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

[2] Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China

[a]tangyi_bupt@163.com, [b]jindh@bupt.edu.cn

Keywords: Alarm correlation, software test system, similar data sources.

Abstract. In the modern society, static defect detection technology become an effective way to build reliable software. But the phenomena of misinformation and failing to report in the result of static defect detection makes testers needed to confirm the false alarms one by one manually. It is necessary to improve efficiency of alarm confirmation. In this paper, an alarm correlation method for similar data sources is proposed. The method uses data sources of the alarm to generate symbol information – the define point information. With the symbolic information, we can generate a feature function which is consist of all alarm variable's DEF point symbols. According to feature function, we can generate an equivalent signature code. Finally, by comparing the similarity of the signatures to determine whether there is an association between the two alarm. Testers only need to pick up one or a few alarms in a class for confirmation after apply this technology which greatly improve the system's efficiency.

1. Introduction

Static defect detection technology is a kind of software test technology based on the program fragment analysis. It has become an effective way to build reliable software for its no program execution and effective in small probability defects test. But the phenomena of misinformation and failing to report in the result of static defect detection makes testers needed to confirm the false alarms one by one manually[1]. A lot of artificial confirmation work waste time and energy, reduce the defect efficiency, and even lead to software developers and managements refuse to use static detection tools in the process of software development. To improve efficiency of alarm confirmation, the alarm correlation technology developed[2]. This technology gets together for a class of associated alarms. Testers only need to pick up one or a few alarms in a class for confirmation, which greatly shorten the confirm time.

Nowadays, There are two main techniques in alarm correlation field. One is a method based on data mining and machine learning to associate defect automatically[3], while the other one is based on abstract interpretation to calculate error state into precise semantics, then resect, disseminate or backtrack with this precise semantics information to associate the defect[4]. The advantage of these methods is that more defect can be grouped together. However, the existing method based on machine learning is difficult to keep balance between association accuracy and correlation efficiency, on the other side, the alarm correlation technology based on symbol execution and constraint solving is limited within the process, and it needs super control flow graph to associate grouping technology which limits the scale of the program and affects the association efficiency.

In this paper, an alarm correlation method for similar data sources is proposed. The method uses data sources of the alarm to generate symbol information – the value information, and with the structure of the alarm to generate into feature function, then with the feature function to generate a

signature code. Finally, by comparing the similarity of the signatures to determine whether there is an correlation between the two alarm. The higher the similarity, the higher the association degree.

2. Related Work

The root cause of an alarm is its assignment operation of related variables, which is directly associated with the assigned object, the method to assign the object, and the composition way of the assigned data. Here are some basic concepts about assignment and the classification of alarms.

A. Basic Concepts

Two kinds of variables appear in the program: DEF and USE.

- DEF Point: To Assign the variable (write to variable memory)
- USE Point: Take the value of the variable (read from the variable memory)

DEF point is the data source of a variable, which would affect the current variable values.

B. Types of Alarm Correlation Based on Similar Data Sources

The origin of alarms are DEF points. Alarm point and DEF points may appear in the same function, and also in different functions. According to the above rules, DEF points could be divided into inside a function (in the same function with alarm point) and across functions(not in the same function with alarm point). Across functions points include function parameters, global variables, return values and side effects. As shown in the figure below:

- Inside a Function

```
1  int jhb_npd_29_f3(void)
2  {
3     FILE *stream;
4     fpos_t filepos;
5     stream = fopen("MYFILE.TXT", "w+"); //fopen may return null pointer
6     fseek(stream, 0L, SEEK_END); //NULL POINTER DEFECT
7     fgetpos(stream, &filepos); // NULL POINTER DEFECT
8     ftell(stream); // NULL POINTER DEFECT
9     return 0;
10 }
```

In this code snippet, null pointer alarms are reported in 6, 7, 8 lines. The fopen function may returns null pointer may in line 5. If the pointer is assigned to variable stream, fseek、fgetpos、ftell functions would throw null pointer exceptions when dereference variable stream. So that the alarm in lines 6,7,8 are all caused by the fopen function's return pointer may be null. Above all, these alarms have the same data source, so they can get together to one class.

- Across Functions

```
1  char * str_to_nstr (str_t * str)
2  {
3     char *nstr = xmalloc (str->len + 1); //xmalloc() may return null pointer
4     int pos = 0;
5     for (; pos < str->len; pos++)
6     {  …
7        nstr[pos] = str->str[pos]; // NPD
8     }
9     nstr[pos + 1] = 0;
10    return nstr;
11 }
12 str_t * str_make (str_t * str)
13 {
14    if (!str)
15    {
16       str = xmalloc (sizeof *str);//xmalloc()may return null pointer
```

```
17        str->str = xmalloc (str->mem = CHUNK); // NPD
18      }
        …
19      return str;
20  }
```

In this code snippet, null pointer alarms are reported in 7, 9, 17 lines. The variable nstr is defined and receive the return value of xmalloc() in function str_to_nstr(), which is dereference in 7,9 lines; function parameter str receive the return value of xmalloc() in function str_make(). As the return value of xmalloc() may be null, nstr and str, which is dereference in line7,9 and 17 may also be null. These alarms could get together for a class because all of them were caused by the null pointer of xmalloc() returns.

3. Architecture of Our Alarm Correlation System

DTS(Defect Testing System) is a static alarm detection tool based on defect mode, which is developed by Beijing University of Posts and Telecommunications in many national 863 projects.

At present, it supports three languages of C, C++ and Java. The system is installed on the operating system in the form of client software, and the potential defects in the source code are found by using the static analysis program.

All the alarms used in this paper is detected by DTS.

In this paper, we proposed a method of alarm corrlation based on symbolic DEF point. The method design is shown in the figure below:

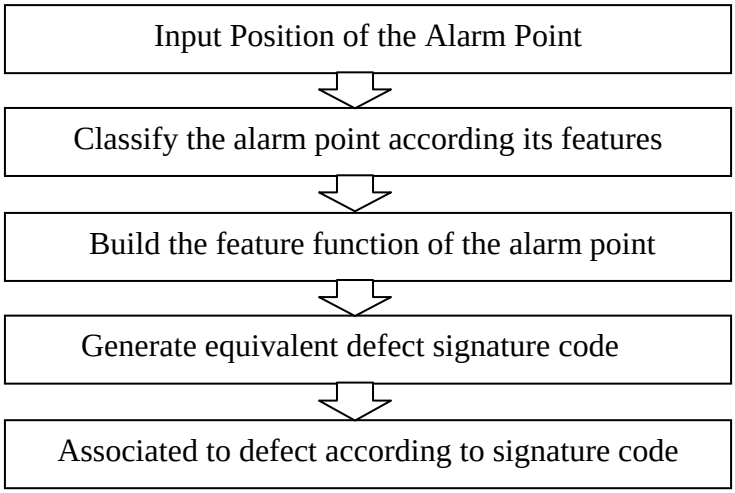

Fig. 1 The System Design Figure

A. Input Position of the Alarm Point

Input location information of the alarm point, which including file path, abstract syntax tree and control flow graph nodes.

B. Classify the alarm point according its features

Classify the alarm point according to the following three features:

- According to the number of variables in alarm point, the alarm point could be divided into single-valued and multiple-valued;
- According to the define level, the alarm point is divided into single layer and multilayer;
- According to the number of alarm data sources, the alarm can be divided into single-source and multi-sources.

C. Build the feature function of the alarm point

Alarm is rooted in the variable's DEF points. Now we symbolic all the DEF points into structure S（Structure）, C（Constant）, V（Variable）, P（Position）. For Example, In the first code snippet line 5:

Code : stream = fopen("MYFILE.TXT", "w+");

Symbol:[structure:[Function:fopen] constancts:["MYFILE.TXT" "w+"] variables:[null] position:[method: jhb_npd_29_f3 line:5]

According to the symbolic information of DEF points and feature of alarm points, we can generate a DEF function which is a consist of all alarm variable's DEF point symbols. For single layer alarm, the DEF function is only one layer; for multilayer alarm, each layer has a DEF function. The DEF point symbol information is got from control flow graph for inside function alarm. While for across functions situation, the information is taken from function abstract messages.

D. Generate equivalent defect signature code

According to the feature function information, generate equivalent signature code. Every layer has a signature code, which is the binary hash code of the feature function.

E. Associated to defect according to signature code

As the signature code is equal with feature function, which is the root cause of an alarm, we could compute equivalence partitioning for alarm points. In each partition, only one or few alarm need to be confirmed manually.

4. Implementation

In this section, we make some experiments to detect the effect of this design. All the alarms is detected by DTS(Defect Test System).

A. Experimental Environment

Table 1 Experimental Environment Table

Hardware environment	Software environment:
Processor: Intel(R) Xeon(R) CPU E5620 @2.4GHz. Memory: 4.00GB.	Operating system: Microsoft Windows 7 JDK: 1.7.0_02. DTS: 8.0

B. Result and analysis

The experimental results show in table below, including the files number of the project, the alarms number, the associated alarms number and correlation.

Table 2 Alarm Correlation Result

Project name	File number	Alarm number	Associated alarms			correlation
			Inside a function	Across functions	Total	
sphinxbase-0.3	120	285	70	12	82	28.77%
Spell-1.0	6	40	9	2	11	27.50%

From the above experimental results, it can be seen that the alarm correlation method based on similar data sources can be effective to establish a link between equivalent alarms, for the correction is nearly 30 percent. Clearly, application of this technology would significant shorten the time of confirmation.

5. Conclusion and future work

In this paper, the alarm correlation method is based on similarity of symbolic expression and calculation of DEF-USE chain, which have higher demand to the precision of the abstract explanation of. Besides, since it depends on the information generated in alarm detection process instead of special correlation calculation, the correlation effect is not significant compared with some other methods.

In the future work, we will try our best to further reduce the energy needed by the human review on

the premise of not reduce correlation reliability. For example, combining with Heckman and William's actionable alert identification techniques(AAIT)[5], we can just check the dominant "active" defects, namely according to check whether the leading defects is in a state of "activity" or not. In a word, the method that this paper has put forward need much more practice and studying to enrich and perfect, and it will be more practicably.

References

[1]. Dahai Jin，Yunzhan Gong，Yawen Wang. Software testing technology. Information and Communication Technologies, 2015(3).

[2]. Dalin zhang. Research on static defect detection to optimize some key technologies. Beijing University of Posts and Telecomunications，2014(5).

[3] Kremenek T, Ashcraft K, Yang J, et al. Correlation exploitation in error ranking. ACM SIGSOFT Software Engineering Notes, ACM, 2004, 29(6): 83-93.

[4] Le W, Soffa M L. Path-based fault correlations[C]//Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering. ACM, 2010: 307-316.

[5] Heckman S, Williams L. A model building process for identifying actionable static analysis alerts. In: Proceedings of IEEE ICST'09. Denver: IEEE Press, 2009. 161-170.