DTS-A Software Defects Testing System

Yang Zhao Hong[1][2] Gong Yun Zhan[2] Xiao Qing[1][2] Wang Ya Wen[2]
(*1. Department of Information Engineering of the Academy of Armored Force Engineering, Beijing 100072, China;*
2. State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)
yang_zhaohong2003@yahoo.com.cn

Abstract

This demo presents DTS (Software Defects Testing System), a tool to catch defects in source code using static testing techniques. In DTS, various defect patterns are defined using defect patterns state machine and tested by a unified testing framework. Since DTS externalizes all the defect patterns it checks, defect patterns can be added, subtracted, or altered without having to modify the tool itself. Moreover, typical interval computation is expanded and applied in DTS to reduce the false positive and compute the state of defect state machine. In order to validate its usefulness, we perform some experiments on a suite of open source software whose results are briefly presented in the last part of the demo.

1. DTS's architecture

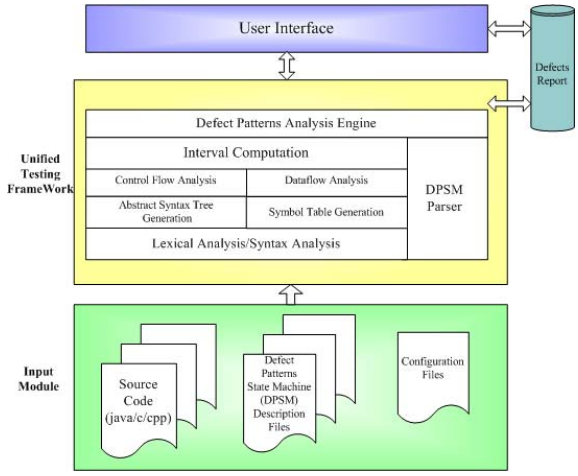

Figure1. DTS's architecture

DTS is a tool to catch defects in source code using static testing techniques. Its core features include defect patterns driven, high efficiency and less false positive. Figure1 shows DTS's Architecture.

This work was supported by the 863 project of China. （2006AA01Z184）（2007AA010302）

1.1 Defects patterns

Defects patterns define the patterns in code that often indicate defects. DTS is defect patterns driven. The tool can never report a defect outsize its defect patterns set. Defect patterns tell the defect patterns analysis engine how to model the environment and the effects of library and system call. The tool's defects patterns set can be divided into four groups:

(1) Fault patterns. Fault patterns define the defects that may cause a program error, such as MLF(Memory Leak Fault), RLF(Resource Leak Fault), NPDF(Null Pointer Dereference Fault), OBAF(Out of Bounds Array Access Fault), ILCF(Illegal Computing Fault), and so on.

(2) Vulnerability patterns. Vulnerability patterns define the defects that may cause a security problem, such as Buffer Overflow, Tainted data, API abuse, and so on.

(3) Style patterns. Style patterns enforce rules related to whitespace, naming, deprecated functions, commenting, program structure and the like. They often affect the readability and the maintainability of the code but do not indicate that a particular error will occur when the program runs.

(4) Poor performance patterns. Poor performance patterns define the defects that may degrade performance, such as using poor performance functions, redundant code, and so on.

1.2 underlying technology

(1) Techniques borrowed from the compiler world.

The first thing DTS needs to do is transform the code to be analyzed into a program model, a set of data structures that represent the code. AST(Abstract Syntax Tree) is a standardized version of the program suitable for later analysis. DTS's AST Construction is based on JavaCC, which is the most popular parser

978-0-7695-3353-7/08 $25.00 © 2008 IEEE
DOI 10.1109/SCAM.2008.12

generator for java. As the ASR is being built, the tool builds a symbol table alongside it. For each identifier in the program, the symbol table associates the identifier with its type and a pointer to its declaration or definition. To explore the different execution paths a control flow graph is built on top of AST. To examine the way data move through a program, Dataflow analysis is performed by traversing control flow graph and noting where data values are generated and where they are used.

(2) Interval computation

DTS expands the typical interval computation and proposes the interval set computation. Then it proposes the interval computation of real variable, boolean variable, pointer variable and array variable. Finally, it proposes the interval computation of conditional statement and the interval computation in control flow traverse. The purpose of interval computation is to compute the values range of variables, which can be used to reduce the false positive and compute the state of defect state machine.

(3) DPSM(Defect patterns state machine)

Various defect patterns are defined using defect patterns state machine. Each DPSM includes an initial state the defect patterns takes on, and has any number of transitions leading to other states that the analysis algorithm will follow whenever it encounters some conditions associated with the transition, such as some code construct. If a DPSM is transited to an error state then a defect is reported.

DTS externalizes all the defect patterns it checks, defect patterns can be added, subtracted, or altered without having to modify the tool itself. Each DPSM is defined in an xml file.

2. Some experiments

Taking NPDF and RLF as example, DTS has been used to find a number of defects in some famous open source software such as Tomcat and Spring. The results are shown in Table1.

Table1. Experiments results

Software	Version	Source files	Lines	NPDF	RLF	IP
Tomcat[1]	4.1.24	1127	349,542	38	3	78
Spring[2]	1.2.9	1763	279,783	5		32
Azureus[3]	3.0.5.0	2770	563,415	31	2	93
Zk[4]	3.0.3	962	133,200	9	2	32
Freemind[5]	0_8_1	509	102,112	25	2	36

Since static analysis tools often produce many false positives, so the IP (Inspection point) reported by DTS is reviewed by our testing team and report about 108 NPDF and 9 RLF. It takes only several minutes for DTS to test above open source software. Figure2 shows the GUI of DTS when testing Tomcat.

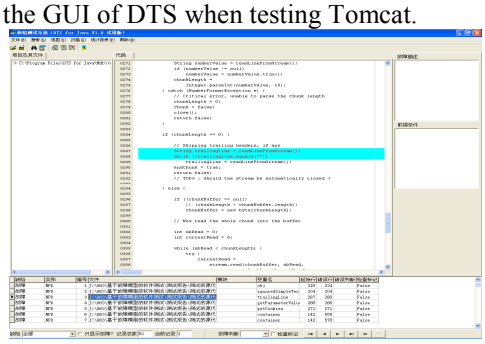

Figure2. GUI of DTS

3. Existing tools and techniques

There are many kinds of static analysis tools, each with different goals, such as type checking, style checking, program understanding, program verification, property checking, bug finding, and security review. DTS covers style checking, bug finding and security review. The bug find tools are most similar to DTS. Findbugs[6] does an excellent job of identifying bugs in java code. Coverity[7] makes a bug finder for C and C++. Microsoft's Visual studio 2005 includes the \analyze option that checks for common coding errors in C and C++. Klocwork[8] offers a combination program understanding and bug finding static analysis tool that enables graphical exploration of large programs. Fortify[9] is a security-focused static analysis tool.

4. Conclusion

DTS is a tool to catch defects in source code using static testing techniques. It covers fault patterns, Vulnerability patterns, Style patterns and Poor performance patterns. Its core features include defect patterns driven, high efficiency and less false positive.

5. References

[1]http://tomcat.apache.org/
[2]http://sourceforge.net/projects/springframework
[3]http://sourceforge.net/projects/azureus
[4]http://sourceforge.net/projects/zk1
[5]http://sourceforge.net/projects/freemind
[6]http://findbugs.sourceforge.net/
[7]http://www.coverity.com
[8]http://www.klocwork.com
[9]http://www.fortify.com