Benchmarking Static C Bug-Checking Tools

Cristina Cifuentes
Sun Microsystems Laboratories
Brisbane, Australia
cristina.cifuentes@sun.com

1. INTRODUCTION

One of the problems with the large number of static bug-checking tools is that it is hard for users (developers and managers) to determine which tool best fits their organisation; quantifying precision of a tool and its scalability is necessary. *Precision* is the ratio of the number of bugs correctly reported to the total number of bugs reported by a tool. *Scalability* is the ability of a tool to scale proportionally in runtime relative to the size of the input codebase.

Another problem that quality assurance engineers have with these tools is the lack of information on what bugs are missed in the code; quantifying recall of the tool is also needed. *Recall* is the ratio of the number of bugs correctly reported by a tool to the total number of bugs in a codebase. Taking into account both, precision and recall, gives a measure of a tool's accuracy. *Accuracy* is the ability of a bug-checking tool to report correct bugs while at the same time holding back incorrect ones.

2. BENCHMARKING SUITES

Taking into account properties of the established benchmarking work from the compilers and virtual machine communities, and data in the initial benchmarking work from the bug-checking community, we developed 3 ways to benchmark bug-checking tools, leading to different types of benchmark suites; all of which are needed to get a good understanding of a tool's capabilities.

An *accuracy* suite is composed of benchmark programs that have been marked-up with the bug/s that exist in it, hence allowing for measurement of precision and recall. Given that it is hard to mark-up where *all* bugs are in a large codebase, an accuracy suite would normally include small programs, ranging from synthetic programs that are written to showcase a bug (i.e., normally less than 50 lines of code) to subsets of programs that are extracted from real buggy code to showcase a bug "in the wild" (i.e., normally less than 500 lines of code).

A *scalability* suite is composed of various benchmark programs that range in size from small ($>$ 1 KLOC) to large (millions LOC). Given the larger size of these programs compared to the ones in an accuracy suite, it is not practical to mark-up these programs with the location of all the bugs in the code. The range of program sizes allows for the measurement of runtime of a bug-checking tool and the plotting of the runtime against the various benchmarks, to determine how well (or not) the tool scales with program size.

An *end-user* benchmark is a large program in the user's domain area of interest that is used to measure how well the tool behaves. Given the size of the benchmark, only some bugs are marked-up (i.e., some that a user cares about). These marked-up bugs are used to get an idea of precision of the tool, e.g., can the tool find at least some of the bugs that are marked-up?, what other bugs can it find? Its runtime provides an idea of how well the tool works in practice; e.g., did it take 1 hour to run?, 10 hours?, 1 day?, 10 days?.

Our benchmarking infrastructure, BegBunch 0.2, consists of: two accuracy suites [1]—a synthetic suite of 1,690 benchmarks with 88 lines of code on average and an accuracy suite of 66 benchmarks with 481 lines of code on average; one scalability suite of 9 benchmarks that range in size from 18 KLOC to 887 KLOC (non-commented lines of code); and 3 end-user benchmarks, namely, the OpenSolaris™ operating system/networking consolidation, build 93 (6 MLOC), the Linux 2.6.29 operating system kernel (5.8 MLOC) and the OpenBSD operating system kernel (1.2 MLOC). Reusable harnesses exist for the accuracy and scalability suites, while scripts and datasets exist for the real world check benchmarks.

3. THIS TALK

In this talk I will provide an overview of BegBunch, equations used to quantify accuracy, give a demo, and present results of an evaluation of BegBunch's harnesses using open source bug-checking tools: Splint, Clang and UNO.

Benchmarking provides a measure of the maturity of a community. For the bug-checking community, it should allow developers and managers to get a better understanding of the capabilities of available tools and their performance, to find the tool that best suits their needs and budget.

4. REFERENCES

[1] C. Cifuentes, C. Hoermann, N. Keynes, L. Li, S. Long, E. Mealy, M. Mounteney, and B. Scholz. BegBunch: Benchmarking for C bug detection tools. In *Proceedings of the International Workshop on Defects in Large Software Systems*, July 2009.

Software Assurance Forum submission