HARVEY MUDD COLLEGE

2004–2005 Google Computer Science Clinic

Differential Test Coverage Analysis in the Context of the Wine Project

Google™

Background

Google's mission is to organize the world's data and make it easier to access. While primarily known for its web search engine, Google provides desktop applications as well, including Picasa (image management), Hello! (blog image management), Keyhole (satellite map search) and Google Desktop Search (PC search engine).

While these programs currently run in Windows, Google would like to make them available on as many platforms as possible. For this project, Google is particularly interested in Linux. They are exploring an open source Windows translation layer for Linux called Wine. This approach avoids both the cost of porting their applications to Linux and the performance overhead of using a Windows emulator under Linux.

What is Wine?

Wine (www.winehq.com) is an open-source project that runs under Linux. It provides a translation layer that sits between an application and the operating system. This layer intercepts Windows function calls and seamlessly provides similar functionality using Linux-based alternatives (Fig. 1).

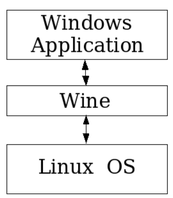

Fig. 1: Windows Applications send instructions which are intercepted by Wine. Wine passes these instructions to Linux, and returns the results to the application.

What is Wrong With Wine?

There are a variety of factors that make Wine's objective difficult to achieve.

Windows is a moving target. There have been a number of releases of Windows since Wine first began, including new APIs such as DirectX and .NET. Wine attempts to maintain compatibility with all versions of Windows starting with DOS/Windows 3.1 through today's Windows XP.

Wine is an open source project that relies on volunteer developers. Features are implemented at the developers' prerogatives so functionality that is either difficult or uninteresting is often ignored. Testing of the code base is often inadequate.

Project Goals

Since Wine is a large and complicated open source project, it is difficult to debug. Our project aims to develop methods for identifying untested portions of Wine used by an application. Specifically, our task was to build a *differential* code coverage tool, use it to diagnose untested areas of Wine used by Google apps, and write tests for those areas.

Wine Test Suite

Wine has a built-in test infrastructure that serves two distinct purposes. First, the tests are used to certify that Wine and Windows behave consistently. Second, the tests are used to make sure that new functionality does not introduce bugs into existing code, a process called regression testing.

Code Coverage

In order to identify untested portions of code, we developed tools to analyze the portions of Wine's source code that are used during a program's execution. We incorporated existing open source code coverage utilities into the Wine test infrastructure. These utilities compare the current coverage of the test suite with the code executed by specific application with specific user input.

Using this tool we expected to be able to identify untested portions of Wine code that are most likely to contain bugs.

Differential Code Coverage

Differential code coverage analysis is a term used to describe the comparison of the code coverage of an application to the code coverage of the Wine test suite. Our tools are designed to quickly and easily highlight these differences in coverage (Fig. 2).

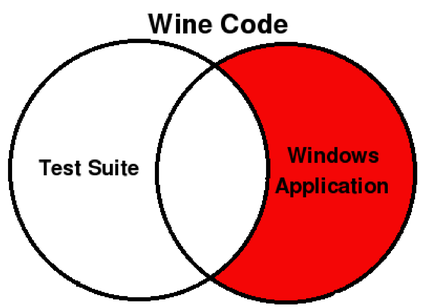

Fig 2. The red section represents the portion of Wine that is used by the application but not tested by the test suite, which is easily identified using our new differential code coverage tool.

GCOV/LCOV

GCOV is an open source code coverage utility that works in conjunction with the GNU gcc compiler to track the lines of program code executed at run-time. LCOV is a postprocessor to GCOV, designed to make coverage output more intuitive and manageable for large, complicated projects (Fig. 3).

We have added GCOV support to Wine, and we have enhanced LCOV to support differential code coverage analysis (Fig. 4).

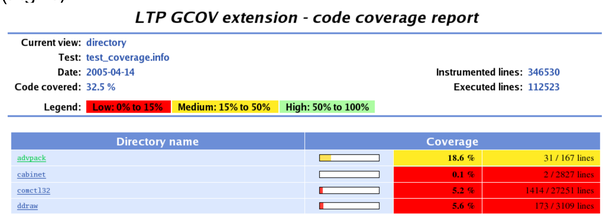

Fig. 3. Sample code coverage from LCOV of Wine's test suite. The directory level of output makes large projects manageable.

Fig. 4a. Code coverage from LCOV of Google's Picasa.

Fig. 4b. Code coverage from LCOV of Wine's test suite.

Fig. 4c. Differential code coverage from our enhanced LCOV comparing the Wine source code profiled under Google's Picasa against Wine's test suite.

CXtest

CXtest (www.cxtest.org) is an open source package for scripting fully automated GUI tests for Windows apps running under Wine. We created a test script using CXtest that installs Picasa. This test is now part of the standard CXtest package, which is used by the Wine developers for automated regression testing.

Automated Coverage

We also created a script that automates the generation of code coverage for Wine as well as differential code coverage comparing Picasa and the Wine test suite.

Coverage-Driven Tests

With our additions to Wine and LCOV, developers can discover untested portions of Wine's code that are used by an application. These regions may be more likely to contain errors. The developer can then write tests specifically targeted at these untested regions. This has the potential to speed Wine development.

The flow chart below (Fig. 5) illustrates this process of coverage-directed test writing. Using our own process, we have identified two libraries, shell32 and ntdll, that are used by Picasa but are insufficiently tested.

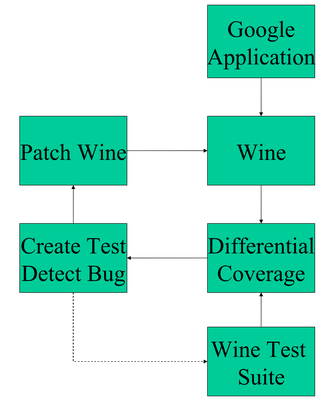

Fig. 5 The Wine testing process after our additions.

Deliverables

- Submitted code coverage support to Wine
- Submitted differential code coverage support to LCOV
- Submitted sample CXtest package for Picasa
- Submitted Wine unit tests for LZExpand and AdvApi32
- Authored automated Wine coverage script
- Final Report

All submissions were accepted by their respective open source projects.

Acknowledgments

Liaisons: Dan Kegel

Faculty Advisor: Prof. Elizabeth 'Z' Sweedyk

Team Members: Cal Pierog (Project Manager), Aaron Arvey, Edward Kim, Evan Parry