오픈 소스 C++에서의 유니트 테스팅 프레임워크에 관한 고찰

황로만*, 안성옥*, 박동원*

요약

소프트웨어 개발 을 성공적으로 수행하기 위하여서 유니트 테스팅은 아주 필수 불가결한 요소이다. Python, Java, C# 등의 현대 컴퓨터 개발 언어에서는 유니트 테스팅을 용이롭게 하게 하기 위하여 다양한 기능을 제공하고 있다. 하지만, C++ 언어에 있어서는 워낙 많은 프레임워크를 제공하고 있는 관계로, 유니트 테스팅을 위한 프레임워크 선택이 결코 용이하지 않다. 이 논문에서는 오픈소스 C++ 언어에서의 유니트 테스팅을 위한 프레임워크를 고찰하였다.Mock 오브젝트 프레임워크는 Google Test와 Google Mock를 권장한다. 그들 자신의 테스트 프레임워크를 필요로하는 개발자들은 베이스로서 CppUnitLite를 사용 할 수있다.

The Study of Open Source C++ Unit Testing Frameworks

Roman Hwang*, Syungog An*, Dong-Won Park*

ABSTRACT

Unit testing is proved to be vital for a successful software development process. Modern languages, such as Python, Java and C#, have a great support and tools for unit testing. But when it comes to C++, there are a big number of C++ frameworks available [List], and it becomes hard to make a choice of unit testing framework to use. This paper presents a survey of open source C++ unit testing frameworks. Mock object testing framework are recommended to use Google Test and Google Mock. Developers that need to create their own testing framework can use CppUnitLite as a basis.

Key Words : Unit test, Open Source, CppUnit, Cfix, Boost.Test

* 배재대학교 게임공학과(✉sungohk@pcu.ac.kr)
· 제1저자(First Author) : 황로만 · 교신저자(Correspondent Author) : 안성옥
· 접수일(2012년 5월 16일), 수정일(1차 : 2012년 6월 14일), 게재확정일(2012년 6월 18일)

Ⅰ. Introduction

The size of the software application is constantly increasing. This makes software development process more difficult and expensive. Different software development methodologies arise to solve the problem. These days the most popular are Agile software development methodologies, based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams.

One of the outstanding agile software development methodologies is Extreme Programming (XP), created by Kent Beck in 1999. As a type of agile software development, it advocates frequent "releases" in short development cycles (timeboxing), which is intended to improve productivity and introduce checkpoints where new customer requirements can be adopted.

The core element of Extreme Programming is unit testing of all code. Extreme Programming uses the creation of unit tests for test-driven development (TDD) [5, 8, 9]. The developer writes a unit test that exposes either a software requirement or a defect, before writing the exact code that implements the requirement or fixes the defect.

Not all the methodologies use test-driven development as a core element, as Extreme Programming, but unit testing is widely accepted as a tool, that decreases development cost, and increases overall quality of the code.

Although unit testing gained its popularity at modern programming languages: SmallTalk, Java, C#, Python, etc, it is also can be effectively used for C and C++. With advent of new programming languages, C++ became less popular for the most of the desktop application development. But still it is widely used for development of software products that require high performance execution. The purpose of this paper is to provide description of different framework open source C++ unit testing frameworks and consequentially help C++ software developers to increase their code quality and therefore the overall software development process.

This paper describes the most notable, in our opinion, unit testing frameworks and in the reminder it describes a testing framework specially designed to use mock objects.

Ⅱ. C++ Unit Testing Frameworks

In this paper we divided open source C++ unit testing frameworks into two groups: frameworks with an integrated test runner (Figure 1) and frameworks with a separate test runner (Figure 2).

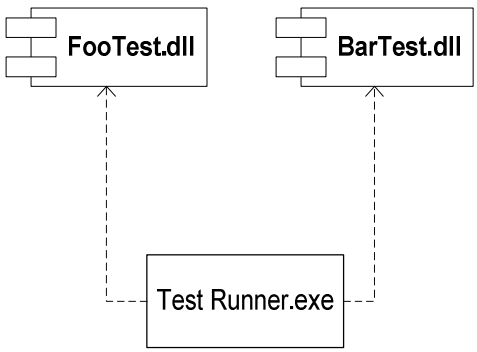


그림 1. 통합된 테스트 러너형의 프레임워크
Fig 1 . Frameworks with an integrated test runner

The former frameworks have unit tests separated from a test runner executable. At unit testing

frameworks examined in this paper (WinUnit [6] and cfix) have all unit tests located in dynamically linked libraries (DLL) and have a separated executable file for actually running the tests. Also this kind of frameworks are called xUnit testing frameworks, such frameworks are based on a design by Kent Beck, originally implemented for Smalltalk as SUnit [1], but are now available for many programming languages and development platforms. The most well-known unit testing framework from this family is JUnit, which is the de facto standard unit testing API for Java development [14].

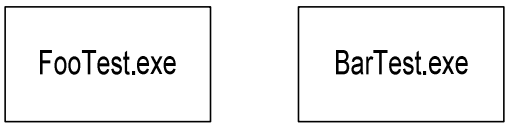


그림 2. 분리된 테스트 러너형의 프레임워크
Fig. 2. Frameworks with a separated test runner

And the latter frameworks have all unit tests combined together with the test runner in a single executable file. We should note that the most of the C++ unit testing frameworks belong to this group.

Later in this paper we first describe frameworks with an integrated test runner, and then frameworks with a separate test runner.

III. Unit Testing Frameworks with Integrated Test Runner

There are many testing frameworks in this group. For this paper we have selected the most notable: CppUnit [13], Boost.Test and Google Test.

3.1 CppUnit

CppUnit is pretty matured, solid and complete unit testing framework. It is probably the most widely used C++ unit testing framework. As the other testing frameworks, CppUnit offers a framework for writing, organizing, and running unit test for your functions, as well as numerous output reporting methods to enable a developer to easily review the rest results [13].

It is widely used by C++ community, but has some downfalls: it requires relatively a lot of work to add even a simple test; it cannot be used on small platforms, because it requires standard template library (STL), run-time type information (RTTI), and exception handling.

To overcome portability limitations and decrease the complexity of the CppUnit, Michael Feathers, the original author of CppUnit, developed an ultra-light unit testing framework CppUnitLite [11]. It is very light, and does not require a lot of work to add unit tests. It lacks a lot of features, but is made to be easily extended by a new functionality, as exception handling, output to file, etc. CppUnitLite can be easily used as a barebones for your own testing framework [11].

3.2 Boost.Test

In C++ community we cannot avoid Boost, a set of extremely useful C++ libraries. Ten Boost libraries are already included in the C++ Standards Committee's Library Technical Report [17] and will be in the new C++0x Standard now being finalized. Boost also has a library for unit testing, called Boost.Test.

Boost.Test has very a rich feature list, such as

checking for infinite loops and different levels of logging. Moreover is has a great documentation and strong supporting community. But it also has some dependencies on other Boost libraries, what makes it heavy-weight.

Boost.Test is a good choice for developers that work on the PC platrom, but it is not recommended and sometimes impossible for using on other platforms, because of its heavy weight.

3.3 Google Test

Google Test is a name of the unit testing framework developed by Google. In addition to many internal projects at Google, Google Test is also used by the following notable projects: The Chromium projects (behind the Chrome browser and Chrome OS) [7], the LLVM (Low-level virtual machine) compiler [12] and Protocol Buffers project [16].

It was initially designed to support a variety of platforms (Linux, Windows, Mac OS X, Cygwin, Windows CE, and Symbian). It is based on xUnit architecture and has many useful features: automatic test discovery, a rich set of assertions, death tests, and various options for running tests, fatal and non-fatal failures, and XML test report generation.

Google Test has a great documentation, and proved its reliability in many projects at Google.

IV. Unit Testing Frameworks with Separate Test Runner

Unit testing frameworks of this type are less popular, because of portability issues. It is hard to create unit testing framework that will work the same way on different platforms, because of difference in binary formats. The frameworks of this type described in this paper (WinUnit [6] and cfix) will work only on Windows platform, since originally they were designed for this platform, but we do not deny the possibility of creating a framework for other platforms.

4.1 WinUnit

There are great unit testing frameworks for Java and C# - JUnit and NUnit, respectively. The absence of such unit testing framework was a motivation for Maria Blees to develop WinUnit [6].

As stated above, all unit tests are located inside dynamically linked libraries (DLLs) and the separate test runner, named WinUnit.exe, is used to run the tests. A single unit test is really a function exported by a dynamically linked library. For example the code on listing A will generate a DLL file with one unit test, called TEST_DummyTest. Then WinUnit.exe will locate, run it and output the result of a testing to console.

표 1. 리스트 A
Table 1. Listing A

```
#include "WinUnit.h"↲
BEGIN_TEST(DummyTest)↲
{↲
  WIN_ASSERT_TRUE(3 > 4, _T("You should see this error message."));↲
}↲
END_TEST↲
```

One of the main benefits of this kind of testing over a framework with integrated test runner is an

ability of testing a functionality of a DLL without exposing the functionality outside the DLL. This can drastically increase the testing coverage and improve the encapsulation.

4.2 Cfix

Although the WinUnit framework provides a great functionality, it has not evolved into a serious project: no new features, no fixes and no community. Cfix is another project with similar to the WinUnit architecture. Cfix has been developing in parallel with WinUnit, but it is continue to evolve and has a strong supporting community.

Moreover from the 1.3 version Cfix is compatible with WinUnit, that means that developers that used WinUnit for their testing purpose, can now easily switch to Cfix without any changes in their unit test code. It is a great advantage for many C++ developers that work on Windows platform, because they can use all the power of the two unit testing frameworks.

Also Johannes Passing, the original author of Cfix, has developed a Visual Assert, C++ unit testing AddIn for Visual Studio, based on Cfix. It helps to easily write, manage, run and debug C/C++ unit tests – without leaving the Visual Studio, which is the main development environment for developers that work on Windows platform.

4.3 Mock Object Testing

Mock object testing is one of the core techniques of unit testing. Mock objects have the same interface as the real objects they mimic, allowing a client object to remain unaware of whether it is using a real object or a mock object. Mock objects are described as a fake object that helps decide if a test failed or passed, by verifying if an interaction on an object occurred or not [15].

Some authors make a distinction between mock and fake objects. In their context a fake object is providing a set of method stubs, piece of code used to stand in for some other programming functionality. A stub method may simulate behavior of existing code or be a temporary substitute for yet-to-be-developed code. This code could be database queries or calculations that take much time and was replaced by a stub method for testing purpose.

Mock objects do a little more their method implementations contain assertions of their own, that are used to checked whether the unit test succeed or not. This means that a mock object, will examine the context of method calls: checking the order in which its methods are called, performing tests on the data passed into the method calls as arguments and other.

A manual writing of mock objects can sometimes require a lot of work. Special testing frameworks were developed for other programming languages to support mock objects, as jMock, EasyMock and Hamcrest. Recently, Google has released Google C++ Mocking Framework (Google Mock for short), that greatly helps to automatically create mock objects for C++ code.

It is recommended to use Google Mock with Google Test, a C++ unit testing framework developed and used internally in many projects by Google, which has a pretty similar to CppUnit feature list. In spite of this Google Mock can be integrated with different C++ unit testing framework with some additional work.

V. Conclusion

Unit testing is an important part of the high quality software product development process. It significantly helps to clarify the structure of the project by forcing a developer to write a module testable. Unit testing greatly decreases the number of bugs in software products. The time spent on debugging now can be used for implementing more features of a product.

In contrast to other programming languages, like C#, Java and Python, C++ has a big number of notable unit testing frameworks. This paper described two families of unit testing frameworks and mock object testing framework, the can support any framework of the both families. There is no best unit testing framework, each of the described above have its pros and cons. CppUnit has proven itself in many projects for a long history. Developers that have PC as a target platform and hardly use Boost libraries will get much benefit from Boost.Test library. Windows programmers may find WinUnit and CFix highly useful for their testing needs, especially if they use Visual Studio as a main development environment. Those who want to support a large variety of platforms and to have ready-to-use mock object testing framework are recommended to use Google Test and Google Mock. Developers that need to create their own testing framework can use CppUnitLite as a basis.

References

[1] Beck, K., Firesmith, D. G., *Simple Smalltalk Testing, Kent Beck's Guide to Better Smalltalk*, chapter 30, Cambridge University Press, December, 1998.

[2] Beck, K., *Embracing Change with Extreme Programming*, Computer 32 (10): 70–77, 1999

[3] Beck, K., *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 1999

[4] Beck, K., Beedle, M., van Bennekum, A., and else, *Manifesto for Agile Software Development*, 2001

[5] Beck, K., *Test Driven Development: By Example*, Addison-Wesley Professional, 2002

[6] Blees, M., *Simplified Unit Testing for Native C++ Applications*, MSDN Magazine, February, 2008

[7] The Chromium projects, http://www.chromium.org/

[8] Jeffries, R., *Test Driven Development: A Practical Guide*, Prentice Hall, 2003.

[9] Koss, R., Lang, J., *Test Driven Development in C, C/C++Users Journal*, October, 2002.

[10] List of unit testing framework for C++ on Wikipedia http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks#C.2B.2B

[11] Llopis, N., *Exploring the C++ Unit Testing Framework Jungle*, http://gamesfromwithin.com/exploring-the-c-unit-testing-framework-jungle

[12] The LLVM compiler infrastructure, http://llvm.org/

[13] Madden, B., *Using CppUnit to Implement Unit Testing*, Game Programming Gems 6, Charles River Media, 2006.

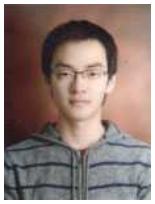
[14] Massol, V., Husted, T., *JUnit in Action*, Manning Publications, 2003.

[15] Osherove, R., *The Art of Unit Testing: With Examples in .Net*, Manning Publications, 2009

[16] Protocol buffers - Google's data interchange format, http://code.google.com/p/protobuf/

[17] Proposed Draft Technical Report on C++ Library Extensions

저자소개



황로만(Roman Hwang)

Kharkov National University of Radio Electronics, 컴퓨터 공학과 (공학사)

2008년~현재 배재대학교 게임공학과 석사과정

안성옥(Sungog An)

She received the Ph.D Degree in Computer Science from Korea University in 1989.
She has been a professor at PAICHAI University since 1991. Her research interests include Computer Graphics, DataBase and Game Development.

박동원(Dong-Won Park)

He received the Ph.D Degree in Computer Science from Texas A&M University in 1993.
He has been a professor at PAICHAI University since 1994.
His research interests include Networked Multimedia and Embedded S/W.