



Bernard Vander Beken
Clear2Pay Services
bvb@iname.com

Unit Testing: an Introduction



Introduction

- Ideal World

Programmers testing all the code ...
... whenever they change the system.



Introduction

- Real-World Problems
 - Time is money
 - Remember that *single* last-minute change causing all the problems?
 - Manually repeated tests
 - *Time-intensive, error-prone and ... boring*
 - Requirements *do* change - must be agile
 - Keep programmers happy

The “Unit” in Unit Testing

- `comp.software.testing` FAQ:
 - “Unit. The smallest compilable component. A unit typically is the work of one programmer (At least in principle). As defined, it does not include any called sub-components (for procedural languages) or communicating components in general. ”
- Just remember
 - Smallest testable software component
 - Even *perfection* at this level would not guarantee nor replace higher-level testing.



Unit Testing

- Helpful features for effective Unit Testing
 - Repeatable
 - Programmable
 - Test execution
 - Command-line and GUI
 - Interactive and batch
 - IDE integration
 - Nevertheless: keep it simple
 - Least amount of overhead and redundancy



Unit Testing

- What to test
 - “Everything” that could possibly break
- The good news
 - “The unit test makes the programmer satisfied that the software does what the programmer thinks it does.”
- The other news
 - This may or may not be what the customer wants in the end.



Unit Testing Problems

- Traditional approach
 - Code first, tests follow
- Problems
 - Not all code is easily testable
 - Might indicate bad design
 - No full code coverage
 - Could use tool



Another Unit Testing Approach

- From Extreme Programming (XP)
 - Program incrementally & test-first
- Advantages
 - Testable code
 - Tested tests
 - Repeatable tests
 - Tests document the code
 - Minimal design



Test/Code Cycle in XP

- Steps in one cycle
 - Write one test
 - Compile the test → probably fails
 - Implement just enough to compile
 - Run the test and see it fail
 - Implement just enough to make the test pass
 - Run the test and see it pass
 - Refactor for clarity and to remove duplication
 - Repeat from top
- Typical Cycle duration
 - 1-5 minutes

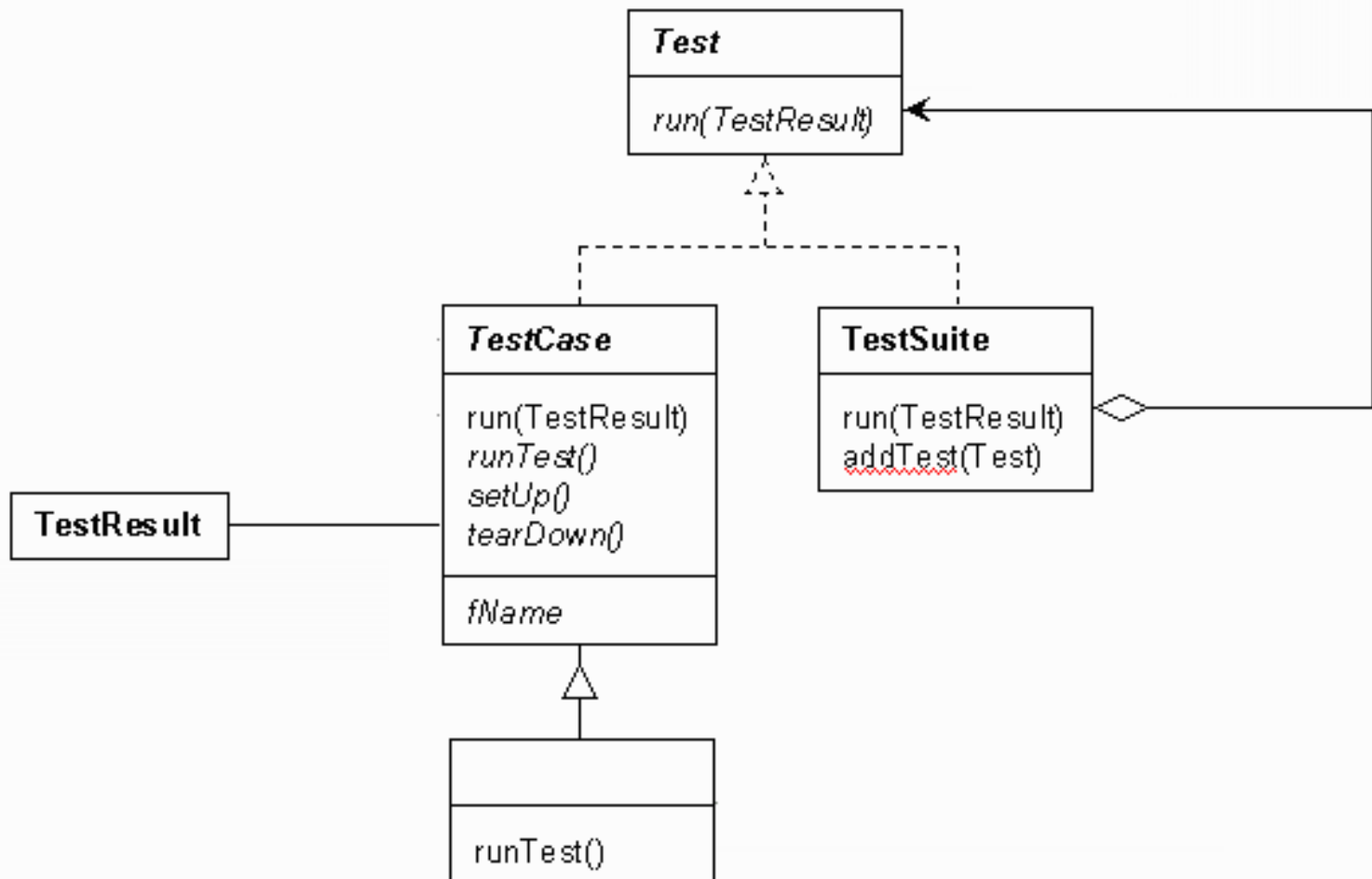


*Unit Testing Framework

- “A way” that is an answer to the needs
- Creators with OO background
 - Erich “Design Patterns” Gamma
 - Kent “*Also Refactoring*” Beck
- Original implementations
 - Smalltalk and Java
- Some of the other implementations
 - .NET, C++, Visual Basic 6

*Unit Testing Framework

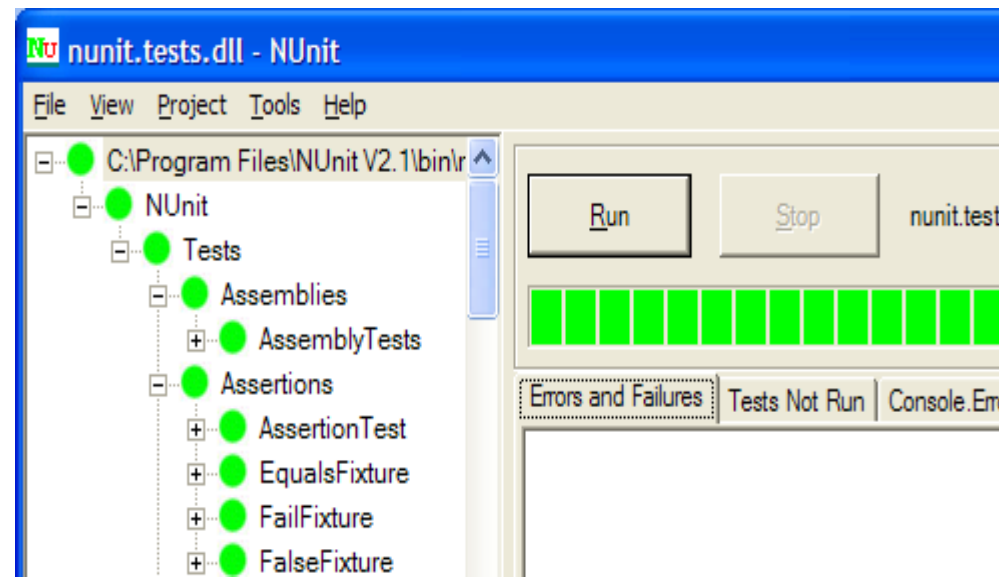
- *Unit = Framework + ≥ 1 Test





Sample Frameworks: NUnit

- Open Source
- Command-line or GUI-based TestRunner
- Possible integration with Visual Studio.NET using NUnitAddin or VS NUnit





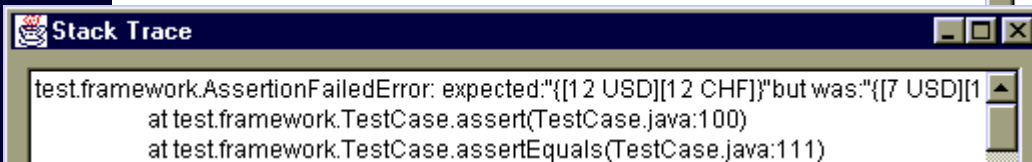
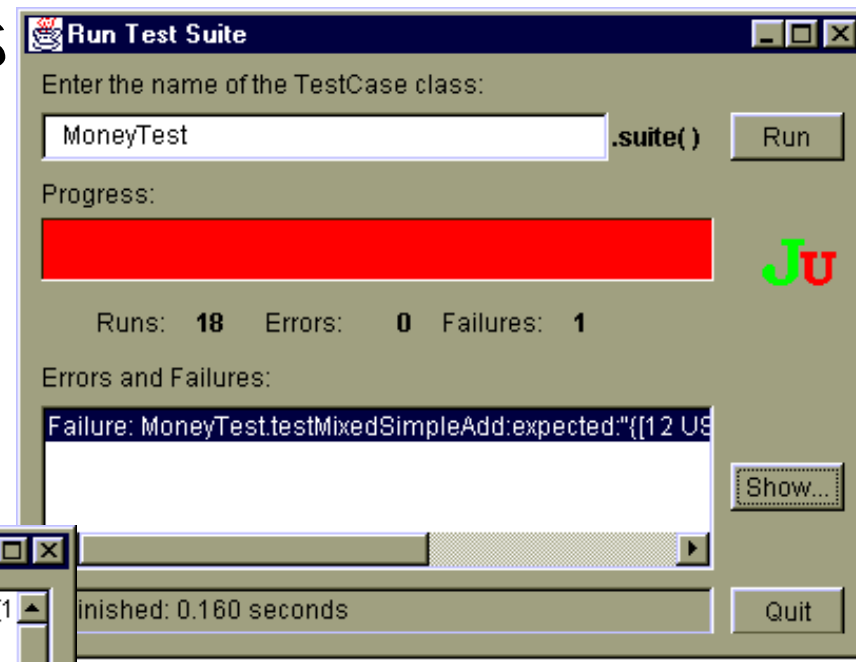
NUnit and beyond

- XMLUnit – NUnit testing for XML
 - <http://xmlunit.sourceforge.net/>
- NUnitASP – ASP.NET unit testing
 - <http://nunitasp.sourceforge.net/>



Sample Frameworks: JUnit

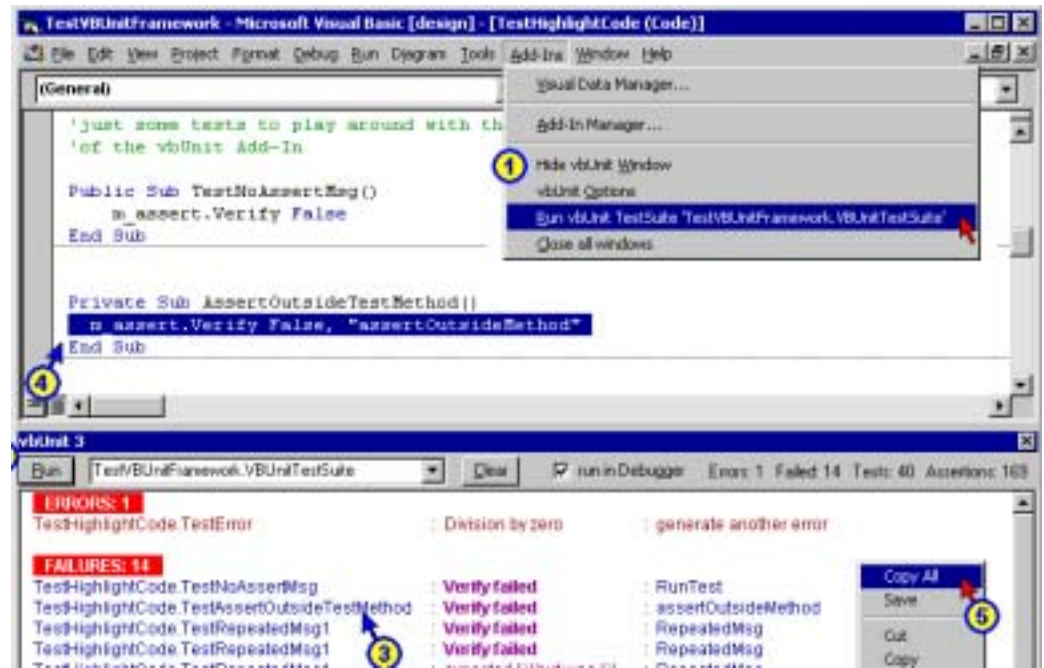
- Released using IBM Public License
- Command-line or GUI-based TestRunner
- Possible integration with Visual Age for Java and NetBeans IDEs





Sample Frameworks: VbUnit

- Version 3.0
- Framework: free to use , with source code
- Free (inferior) or commercial TestRunner





Resources

- Books
 - “The XP Series”
- Online
 - Many mailing lists, search:
groups.yahoo.com
 - XP Intro
 - <http://www.extremeprogramming.org>
 - *Unit Testing Frameworks
 - <http://www.xprogramming.com/software.htm>
 - Unit Testing Database Code
 - <http://www.dallaway.com/acad/dbunit.html>



Thanks!

- Q&A