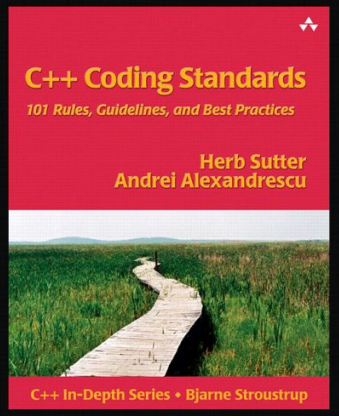
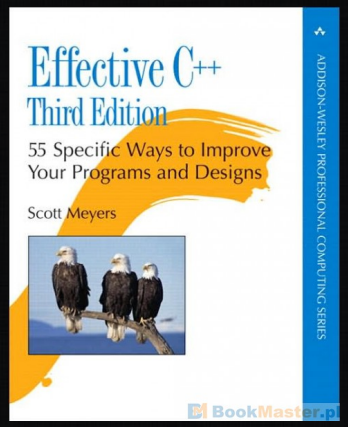
Boost.Test

www.github.com/richelbilderbeek/CppPresentations

About Boost



...one of the most highly regarded and expertly designed C++ library projects in the world.



Item 55:
Familiarize yourself with Boost

The obvious solution for most programmers is to use a library that provides an elegant and efficient platform independent to needed services. Examples are Boost… [1]

[1] Bjarne Stroustrup, Abstraction, libraries, and efficiency in C++

Boost.Test

- What: a testing framework
- Why:
  - measure that your code is correct
  - improve your software architecture
  - use an expertly designed testing framework
- Mastery:
  - write tests with high code coverage run by a continuous integration service
  - use other testing frameworks when those fit better, e.g. Qtest when using Qt

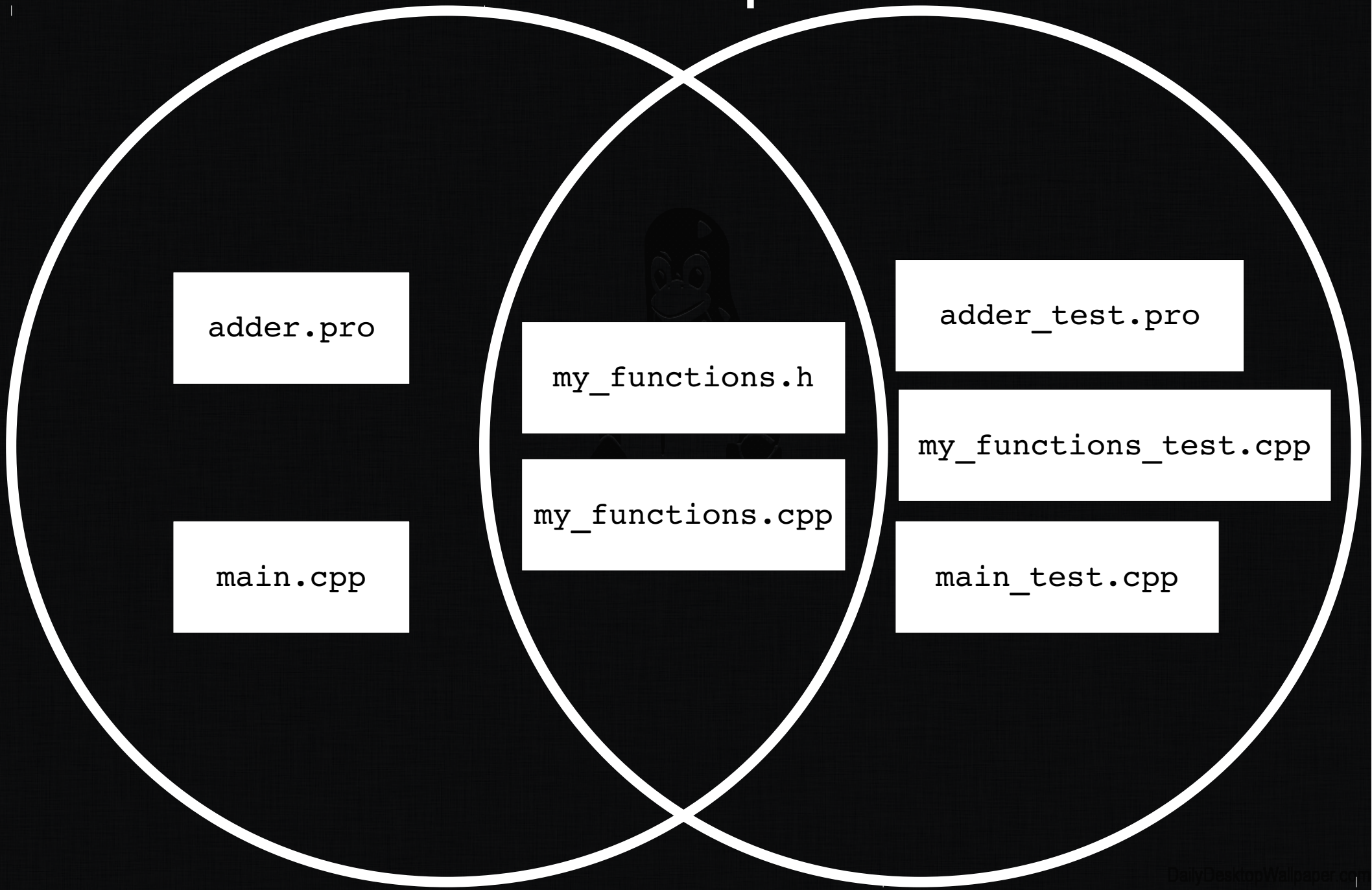
This presentation

- First example: ‘add’
- Testing for exceptions: ‘divide’
- Add Travis CI

First example

- Function ‘add’ that adds two integers
- ‘add’ is used in a main
- ‘add’ is tested by Boost.Test

Setup



main.cpp

```
#include "my_functions.h"
#include <iostream>


int main()
{
  std::cout << add(40, 2) << '\n';
}
```

my_functions.h

```
#ifndef MY_FUNCTIONS_H
#define MY_FUNCTIONS_H

int add(const int i, const int j);

#endif // MY_FUNCTIONS_H
```

my_functions.cpp

```
#include "my_functions.h"

int add(const int i, const int j)
{
  return i + j;
}
```

main_test.cpp

```
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE my_functions_test_module
#include <boost/test/unit_test.hpp>

//No main needed, BOOST_TEST_DYN_LINK creates it
```

my_functions_test.cpp

```
#include <boost/test/unit_test.hpp>
#include "my_functions.h"

BOOST_AUTO_TEST_CASE(add_works)
{
  BOOST_CHECK(add(1, 1) == 2);
  BOOST_CHECK(add(1, 2) == 3);
  BOOST_CHECK(add(1, 3) == 4);
  BOOST_CHECK(add(1, 4) == 5);
}
```

Output

```
Running 1 test case...


*** No errors detected
```

Testing for exceptions

- Because exceptions are part of your interface
- Code coverage correlates with code quality [2]

Develop an error-handling strategy early in a design [1]

[1] C++ Core Guidelines, E.1

[2] Del Frate, Fabio, et al. "On the correlation between code coverage and software reliability." Software Reliability Engineering, 1995. Proceedings., Sixth International Symposium on. IEEE, 1995.

Testing for exceptions

```
#include <stdexcept>


double divide(const double n, const double d)
{
  if (d == 0.0)
  {
    throw std::invalid_argument(
      "divide: denominator cannot be zero"
    );
  }
  return n / d;
}
```

Testing for exceptions

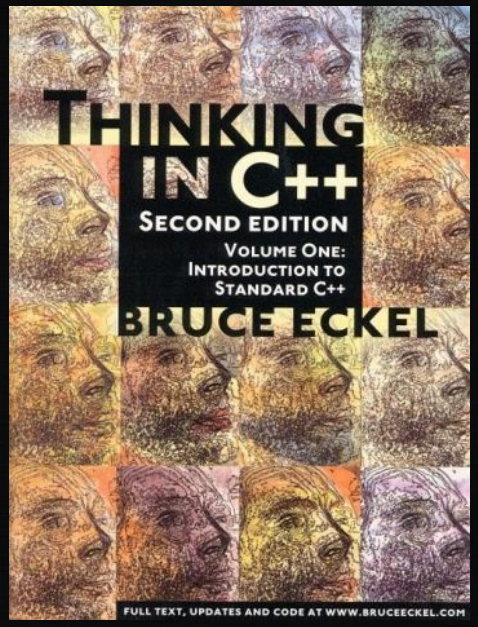
```
BOOST_AUTO_TEST_CASE(divide_use)
{
  {
    const double measured = divide(1.0, 1.0);
    const double expected = 1.0;
    const double error = std::abs(measured - expected);
    const double tolerance = 0.01;
    BOOST_CHECK(error < tolerance);
  }
  {
    const double measured = divide(1.0, 2.0);
    const double expected = 0.5;
    const double error = std::abs(measured - expected);
    const double tolerance = 0.01;
    BOOST_CHECK(error < tolerance);
  }
}
```

Testing for exceptions

```
BOOST_AUTO_TEST_CASE(divide_abuse)
{
  BOOST_CHECK_THROW(
    divide(0.0, 0.0),
    std::invalid_argument
  );
}
```

Add Travis CI

- Will do whatever you want after a ‘git push’
- Plays well with GitHub
- Free for FOSS



Automate the running
of your tests through
a makefile or similar tool

.travis.yml

```
sudo: true
language: cpp
compiler: gcc
addons:
  apt:
    packages: libboost-all-dev

script:
 - ./build.sh
 - ./build_test.sh
```

build.sh

```
#!/bin/bash
qmake adder.pro
make
./adder
```

build_test.sh

```
#!/bin/bash
qmake adder_test.pro
make
./adder_test
```

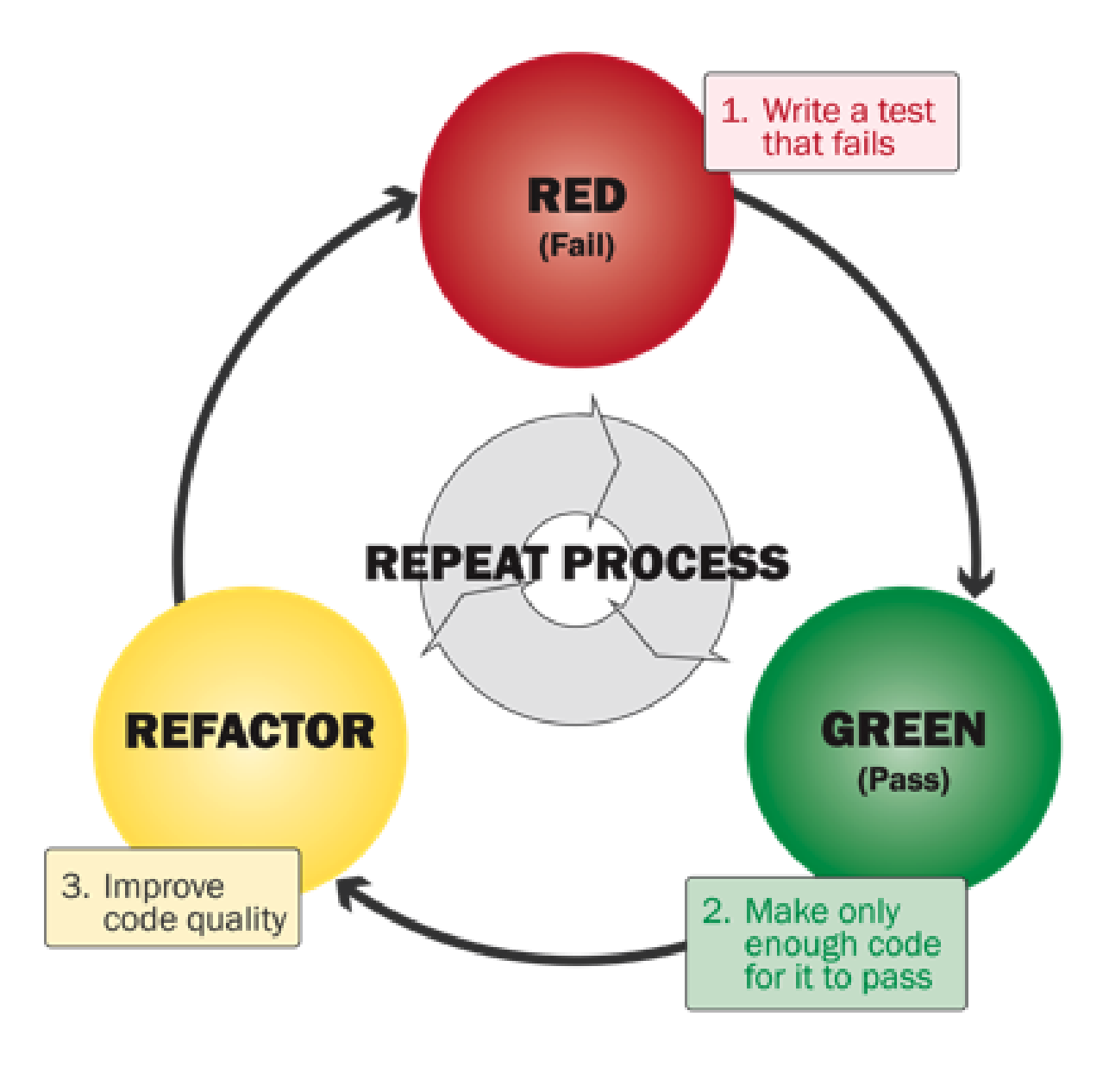
Remove log | Raw log

```
Worker information                                                    worker_info
Build system information                                              system_info

$ export DEBIAN_FRONTEND=noninteractive                               fix.CVE-2015-7547
$ git clone --depth=50 --branch=master                                git.checkout   0.86s
Installing APT Packages (BETA)                                        apt
$ export CXX=g++
$ export CC=gcc
$ gcc --version
gcc (Ubuntu/Linaro 4.6.3-1ubuntu5) 4.6.3
Copyright (C) 2011 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

$ ./build.sh                                                          0.57s
make -f Makefile.Release
make[1]: Entering directory `/home/travis/build/richelbilderbeek/travis_qmake_gcc_cpp98_boost_test'
g++ -c -m64 -pipe -O2 -Wall -W -D_REENTRANT -DQT_WEBKIT -DNDEBUG -DQT_NO_DEBUG -I/usr/share/qt4/mkspecs/linux-g++-64 -I. -I/usr/include/qt4 -Irelease -o release/main.o main.cpp
g++ -c -m64 -pipe -O2 -Wall -W -D_REENTRANT -DQT_WEBKIT -DNDEBUG -DQT_NO_DEBUG -I/usr/share/qt4/mkspecs/linux-g++-64 -I. -I/usr/include/qt4 -Irelease -o release/my_functions.o my_functions.cpp
g++ -m64 -Wl,-O1 -o travis_qmake_gcc_cpp98_boost_test release/main.o release/my_functions.o    -L/usr/lib/x86_64-linux-gnu -lpthread
make[1]: Leaving directory `/home/travis/build/richelbilderbeek/travis_qmake_gcc_cpp98_boost_test'
42


The command "./build.sh" exited with 0.
$ ./build_test.sh                                                     1.13s
make -f Makefile.Release
make[1]: Entering directory `/home/travis/build/richelbilderbeek/travis_qmake_gcc_cpp98_boost_test'
g++ -c -m64 -pipe -O2 -Wall -W -D_REENTRANT -DQT_WEBKIT -DNDEBUG -DQT_NO_DEBUG -I/usr/share/qt4/mkspecs/linux-g++-64 -I. -I/usr/include/qt4 -Irelease -o release/main_test.o main_test.cpp
g++ -c -m64 -pipe -O2 -Wall -W -D_REENTRANT -DQT_WEBKIT -DNDEBUG -DQT_NO_DEBUG -I/usr/share/qt4/mkspecs/linux-g++-64 -I. -I/usr/include/qt4 -Irelease -o release/my_functions_test.o my_functions_test.cpp
g++ -m64 -Wl,-O1 -o travis_qmake_gcc_cpp98_boost_test_test release/my_functions.o release/main_test.o release/my_functions_test.o -L/usr/lib/x86_64-linux-gnu -lboost_unit_test_framework -lpthread
make[1]: Leaving directory `/home/travis/build/richelbilderbeek/travis_qmake_gcc_cpp98_boost_test'
Running 1 test case...

*** No errors detected


The command "./build_test.sh" exited with 0.

Done. Your build exited with 0.
```

Top

Advice from TDD

- Only add tests that do break the build



Advice from me

- Be friendly to yourself:
  - NO: `BOOST_CHECK(my_sqrt(0.0) == 0.0)`
  - YES: `BOOST_CHECK(my_sqrt(0.0) < 0.01)`
- When working within framework X (e.g. Qt), that includes a testing framework (e.g. Qtest), prefer to use it over Boost.Test

Conclusion

- Boost.Test is a testing framework, that
  - Makes it easy to add new tests
  - Makes it hard to forget adding tests
  - Has good exception testing
  - Can easily be used by Travis CI