Quickly Testing Legacy Code

Clare Macrae (She/Her)

CPPP Conference, Paris: 15 June 2019

@ClareMacraeUK

Contents

Llewellyn Falco
@LlewellynFalco

As part of expanding my c++ I was thinking of improving #ApprovalTests and making it work with GoogleTest. Anyone interested in pairing on that?

12:52 PM - 26 Nov 2017

Llewellyn Falco
@LlewellynFalco

As part of expanding my c++ I was thinking of improving #ApprovalTests and making it work with GoogleTest. Anyone interested in pairing on that?

12:52 PM - 26 Nov 2017

Clare Macrae
@ClareMacraeUK

Replying to @LlewellynFalco

Would be interested in hearing more.

2:46 PM - 26 Nov 2017

18 months of remote pairing later…

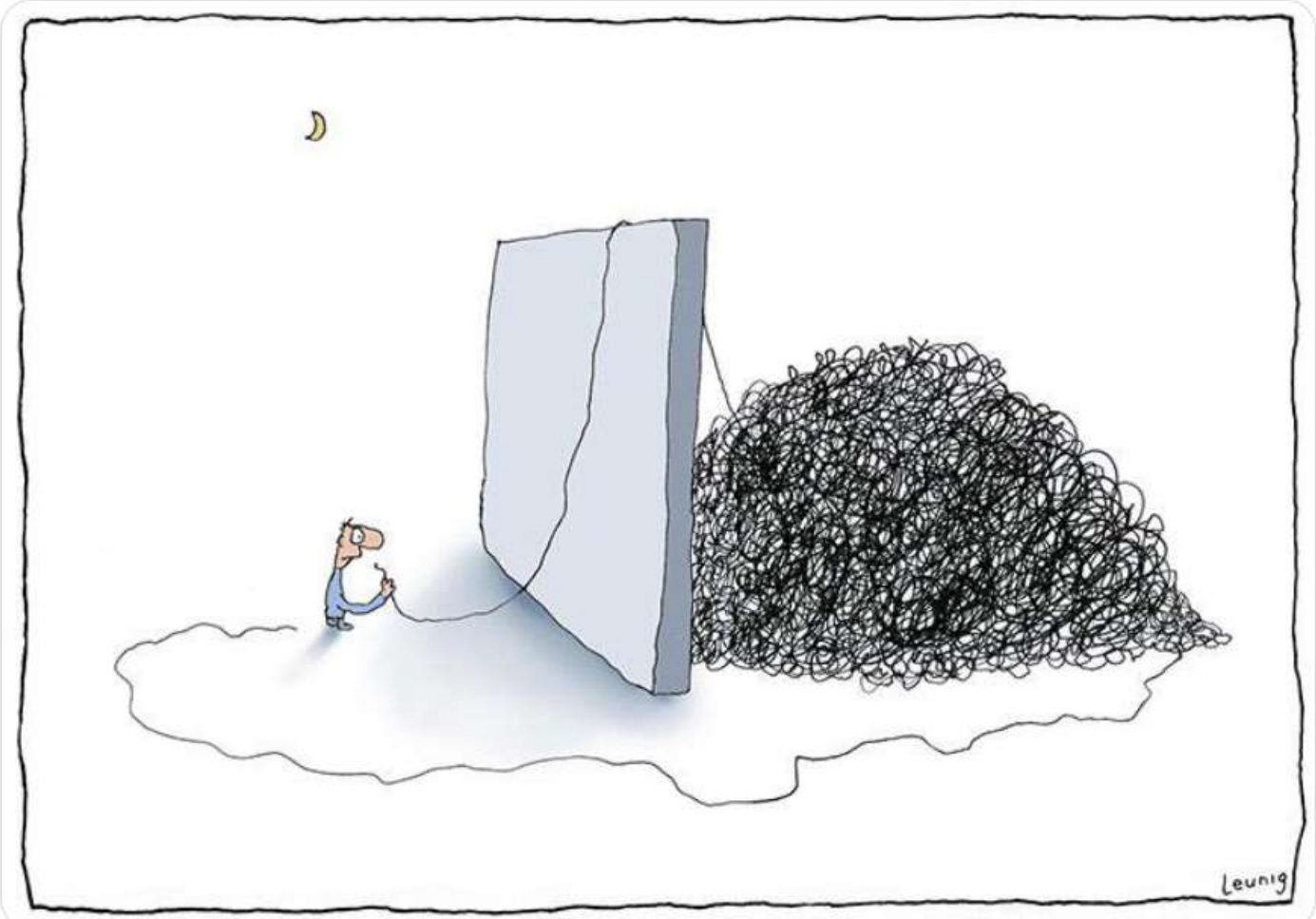
Goal:
Share techniques for easier testing in challenging scenarios

Contents

Quickly Testing **Legacy Code**

kenbot
@KenScambler

Legacy. Looks easy; should be done in half an hour I reckon.

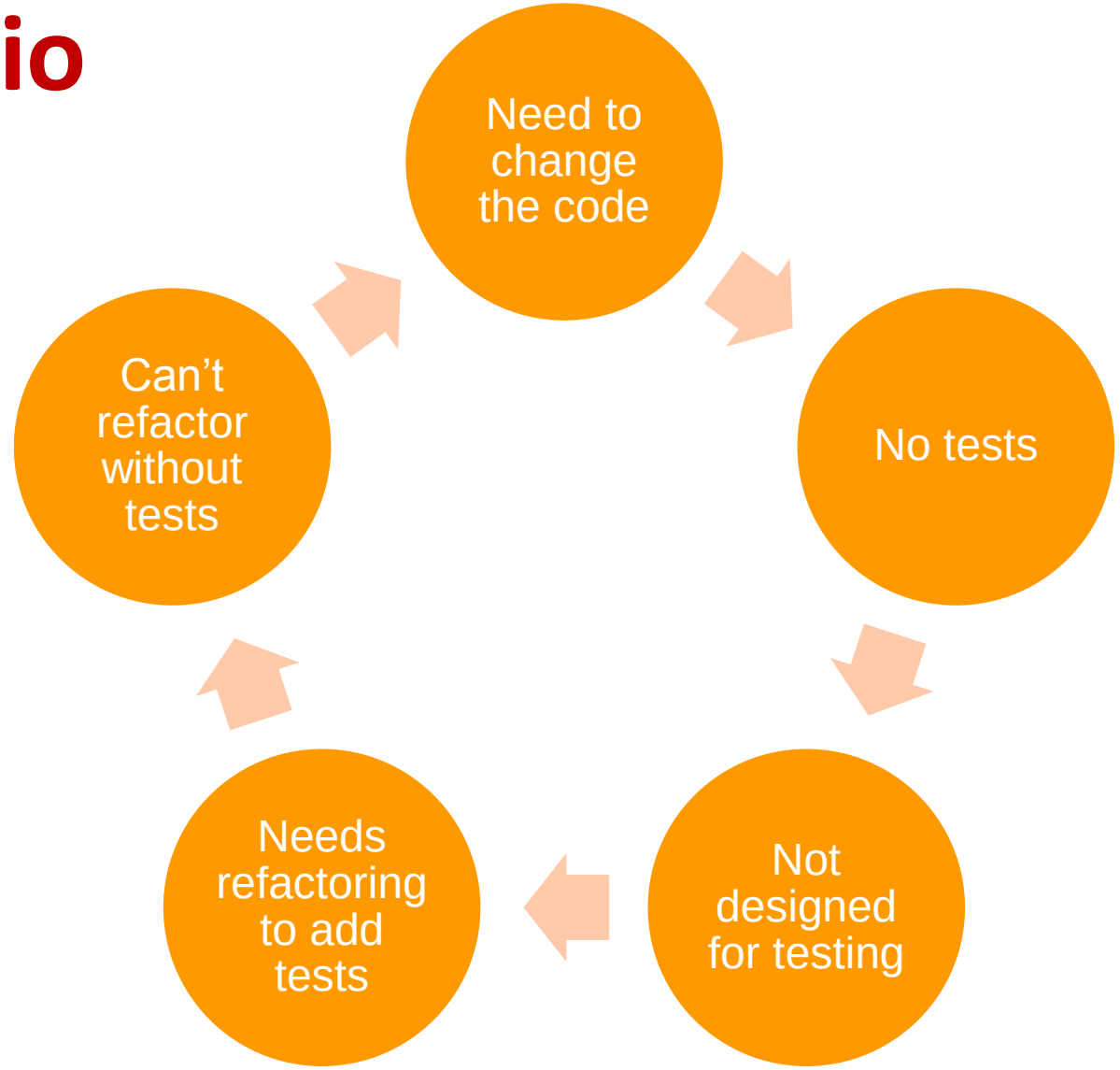


What does Legacy Code really mean?

- **Michael Feathers**
  - "code **without unit tests**"
- **J.B. Rainsberger**
  - "**profitable code** that we feel afraid to change."
- **Kate Gregory**
  - "any code that you **want to change, but are afraid to**"

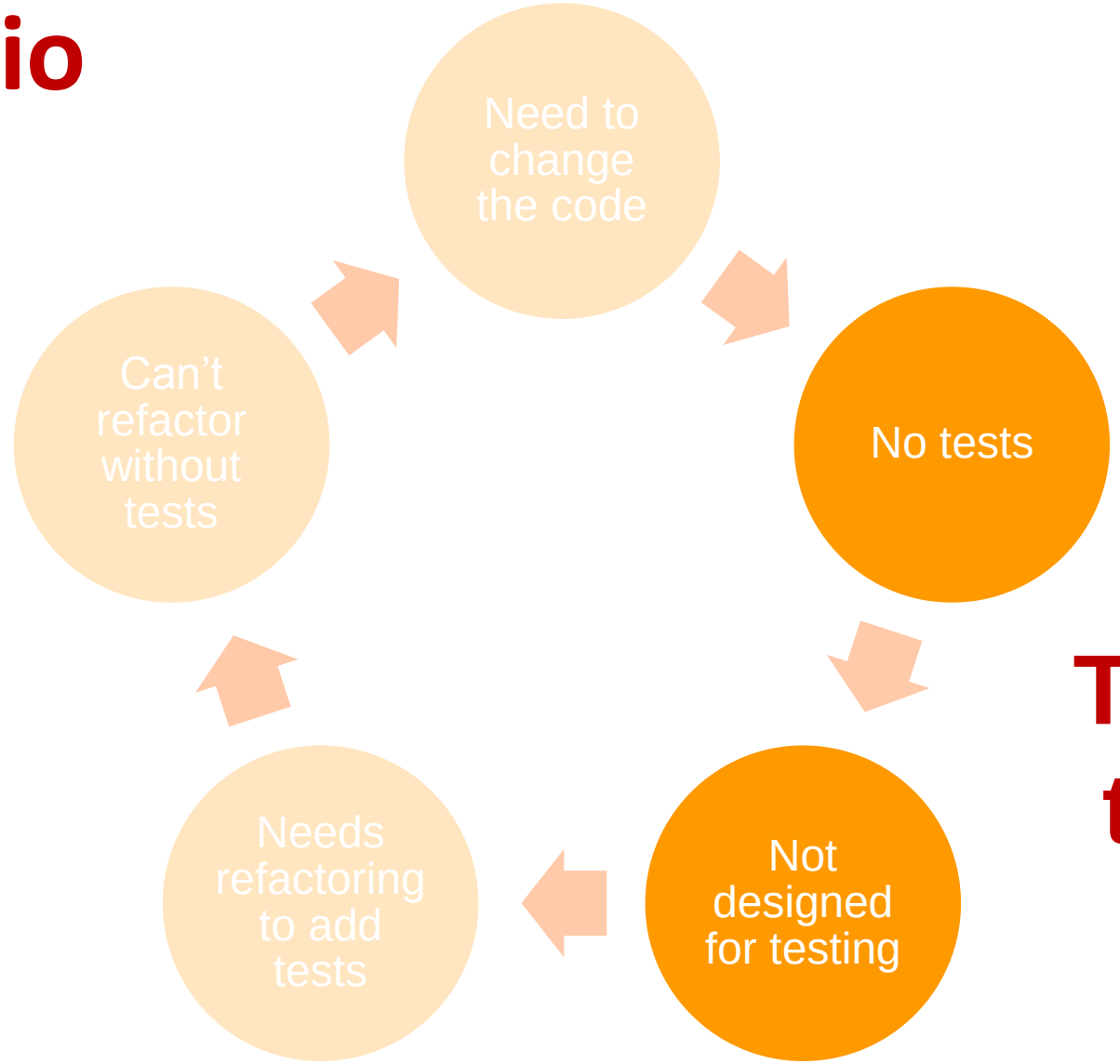
Typical Scenario

- I've inherited some legacy code
- It's valuable
- I need to add feature
- Or fix bug
- How can I ever break out of this loop?



Typical Scenario

- I've inherited some legacy code
- It's valuable
- I need to add feature
- Or fix bug
- How can I ever break out of this loop?



Topics of this talk

Assumptions

- Value of automated testing
- No worries about types of tests
  - (unit, integration, regression)

Any existing tests?

What, no tests?

- If absolutely no tests…
- Stop now!
- Set one up!
- Existing framework

Popular Test Frameworks

Google Test

- Google's C++ test framework
- https://github.com/google/googletest

Catch

- Phil Nash’s test framework
- https://github.com/catchorg/Catch2

The Pragmatic Programmers

Modern C++ Programming with Test-Driven Development

Code Better,
Sleep Better

Jeff Langr

Foreword by Robert C. Martin
(Uncle Bob)

Edited by Michael Swaine

How good are your existing tests?

First evaluate your tests

- If you do have tests….
- **Test the tests**!
- In area you are changing
- Reliable?

Code Coverage

- Caution!
- Unexecuted code

- **Techniques**
- Debugger
- Code coverage tool
  - What to measure?

cpponsea2019 - Microsoft Visual Studio

GildedRoseApprovals.cpp | GildedRoseTest.cpp | GildedRose.cc

gilded_rose_refactoring_kata → GildedRose → updateQuality()

```
#include "GildedRose.h"

GildedRose::GildedRose(vector<Item> & items) : items(items)
{}

void GildedRose::updateQuality()
{
    for (int i = 0; i < items.size(); i++)
    {
        if (items[i].name != "Aged Brie" && items[i].name != "Backstage passes to a TAFKAL80ETC concert")
        {
            if (items[i].quality > 0)
            {
                if (items[i].name != "Sulfuras, Hand of Ragnaros")
                {
                    items[i].quality = items[i].quality - 1;
                }
            }
        }
        else
        {
            if (items[i].quality < 50)
            {
                items[i].quality = items[i].quality + 1;

                if (items[i].name == "Backstage passes to a TAFKAL80ETC concert")
                {
                    if (items[i].sellIn < 11)
                    {
                        if (items[i].quality < 50)
```

Coverage

Filter: gild Display coverage

	Coverage	Covered line	Uncovered line	Total line
gilded_rose_refactoring_kata.exe	36%	3568	6282	9850
D:\Users\Clare\Documents\Programming\github\cpponsea2019\cmake-build-vs\gilded_rose_refactoring_kata\Debu	36%	3568	6282	9850
d:\users\clare\documents\programming\github\cpponsea2019\gilded_rose_refactoring_kata\gildedrose.cc	100%	33	0	33
d:\users\clare\documents\programming\github\cpponsea2019\gilded_rose_refactoring_kata\gildedrose.h	100%	2	0	2
d:\users\clare\documents\programming\github\cpponsea2019\gilded_rose_refactoring_kata\gildedroseapprove	100%	46	0	46
d:\users\clare\documents\programming\github\cpponsea2019\gilded_rose_refactoring_kata\gildedrosetest.cpp	100%	8	0	8

100 100% Clare Macrae, 31 minutes ago | 1 author, 1 change

Coverage | Unit Test Explorer | Output | Error List

Ready Ln 26 Col 38 Ch 38 INS cpponsea2019 master

100% Test Coverage

Or is it?

OpenCppCoverage does not show branch/condition coverage.

test.cov - BullseyeCoverage Browser

File Edit View Go Region Tools Help

Classes Files

d:/builds/github/cpponsea2019/gilded_rose_refactoring_kata/GildedRose.cc

```
1  #include "GildedRose.h"
2
3  GildedRose::GildedRose(vector<Item> & items) : items(items)
4  {}
5
6  void GildedRose::updateQuality()
7  {
8      for (int i = 0; i < items.size(); i++)
9      {
10a        if (
10b            items[i].name != "Aged Brie" &&
10c                                         items[i].name != "Backstage passes to a TAFKAL80ETC cc
11         {
12             if (items[i].quality > 0)
13             {
14                 if (items[i].name != "Sulfuras, Hand of Ragnaros")
15                 {
16                     items[i].quality = items[i].quality - 1;
17                 }
18             }
19         }
20         else
21         {
22             if (items[i].quality < 50)
23             {
24                 items[i].quality = items[i].quality + 1;
25
26                 if (items[i].name == "Backstage passes to a TAFKAL80ETC concert")
27                 {
28                     if (items[i].sellIn < 11)
29                     {
30                         if (items[i].quality < 50)
31                         {
```

updateQuality() | Function coverage 100% | Uncovered functions 0 | Condition/decision coverage 73% | Uncovered conditions/decisions 10

Coverage build enabled

BullseyeCoverage:

Unrolls If conditions (line 10)

73% of conditions covered

Mutation testing: Sabotage the code!

- Test the tests
- Small changes
- Re-run tests
- Fail: ☺
- Pass: ☹

Mutation testing approaches

- By hand,
  - e.g. + to -
- By tool
  - e.g. Mutate++
    - https://github.com/nlohmann/mutate_cpp
  - e.g. Mull – see CppCast episode 198, Alex Denisov
    - https://cppcast.com/2019/05/alex-denisov/
    - https://github.com/mull-project/mull
  - Awesome Mutation Testing:
    - https://github.com/theofidry/awesome-mutation-testing

If tests need improving…
And unit tests not viable…

Contents

Quickly **Testing** Legacy Code

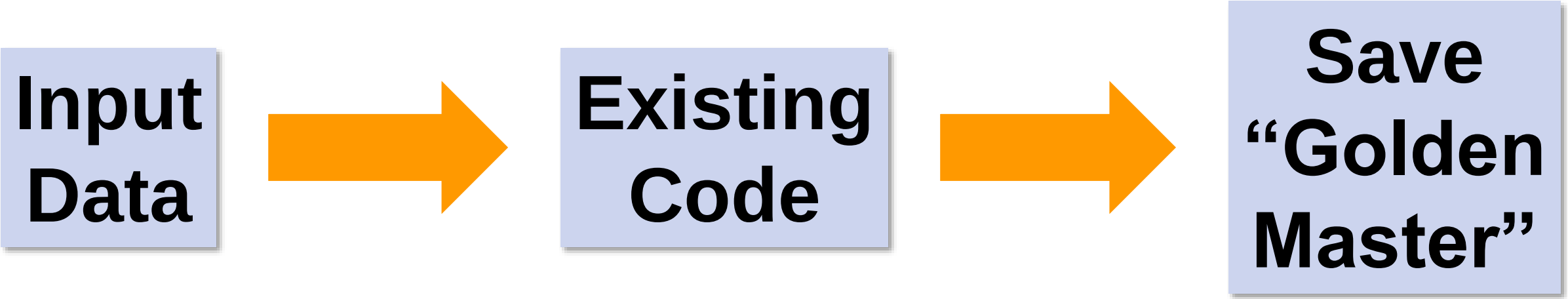
Key Phrase :
“Locking Down Current Behaviour”

Writing Unit Tests for Legacy Code

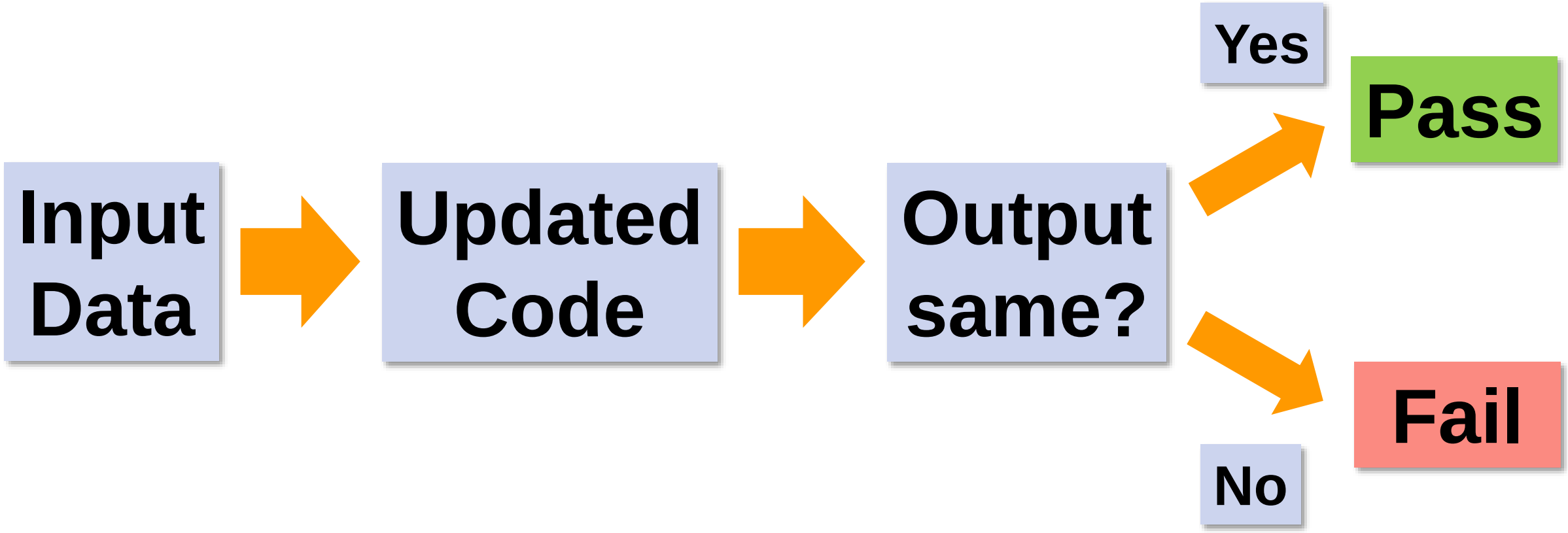
- Time-consuming
- What’s intended behaviour?
- Is there another way?

Alternative: Golden Master Testing

Golden Master Test Setup



Golden Master Tests In Use



Thoughts on Golden Master Tests

- Good to start testing legacy systems
- Poor Person’s Integration Tests
- Depends on ease of
  - Capturing output
  - Getting stable output
  - Reviewing any differences
  - Avoiding overwriting Golden Master by mistake!
- Doesn’t matter that it’s not a Unit Test

Contents

Quickly Testing Legacy Code

One approach: “ApprovalTests”

- Llewellyn Falco’s convenient, powerful, flexible Golden Master implementation
- Supports many language

GO	.Net.ASP	Perl
Java	NodeJS	Python
Lua	Objective-C	Swift
.NET	PHP	

And now C++!

ApprovalTests.cpp

- New C++ library for applying Llewellyn Falco's "Approval Tests" approach
- For testing cross-platform C++ code (Windows, Linux, Mac)
- For legacy and green-field systems
- It's on github

ApprovalTests.cpp

- Header-only
- Open source - Apache 2.0 licence

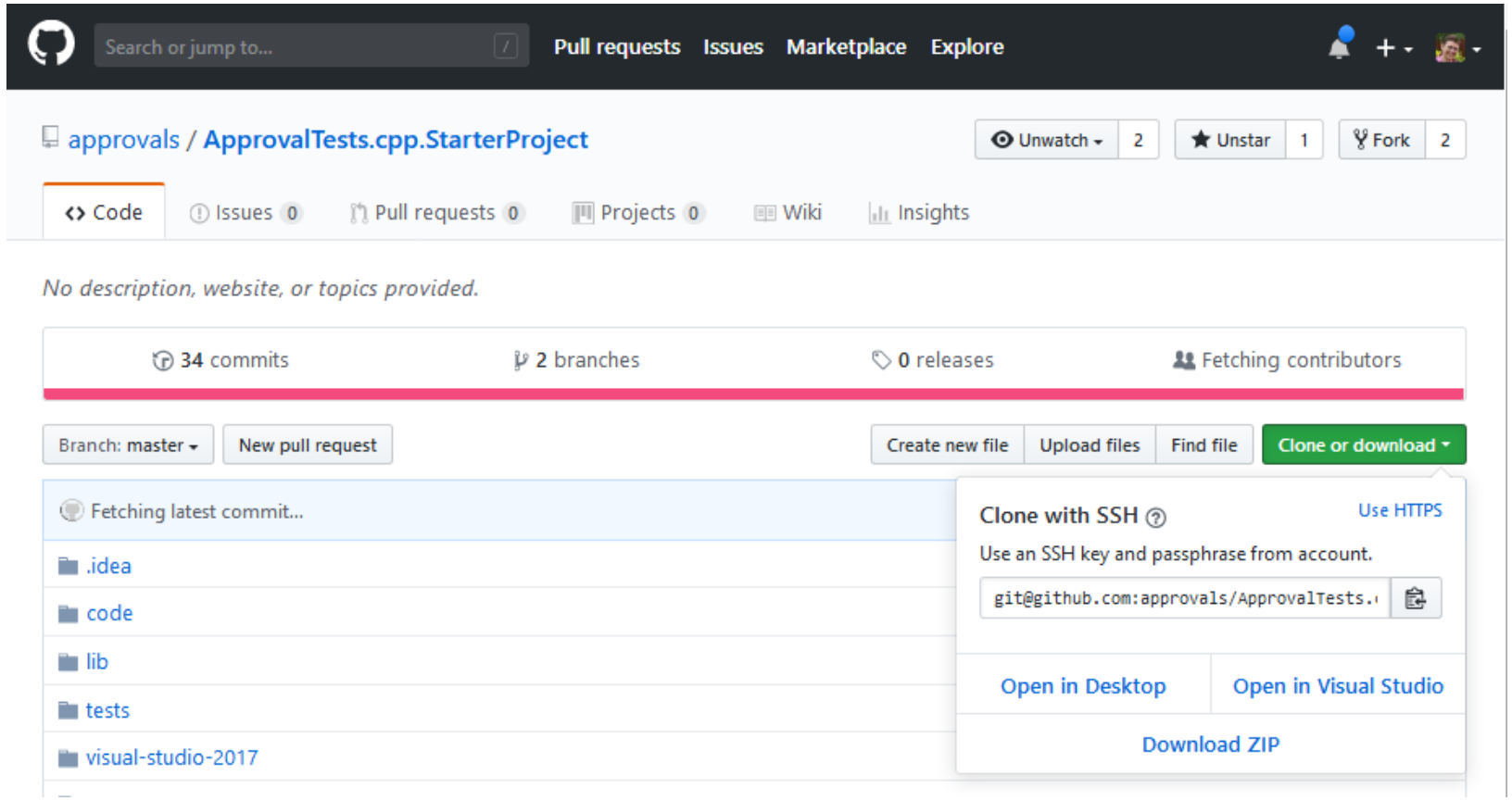
ApprovalTests.cpp

- Works with a range of testing frameworks
- Currently supports Catch1, Catch2, Google Test and Okra

Getting Started with Approvals in C++

StarterProject with Catch2

- https://github.com/approvals/ApprovalTests.cpp.StarterProject
- Download ZIP



Download it Yourself…

GitHub - approvals/ApprovalTest

GitHub, Inc. [US] https://github.com/approvals/ApprovalTests.cpp

Why GitHub? Enterprise Explore Marketplace Pricing Search / Sign in Sign up

approvals / ApprovalTests.cpp

Watch 12 Star 34 Fork 2

Code Issues 3 Pull requests 0 Security Insights

Native ApprovalTests for C++ on Linux, Mac and Windows

c-plus-plus cpp11 testing header-only single-file approval-test approval-testing regression-testing golden-master snapshot-testing

546 commits 3 branches 13 releases 4 contributors Apache-2.0

Branch: master New pull request Find File Clone or download

claremacrae t Make a google test more readable Latest commit d4f42c5 4 days ago

ApprovalTests	B fix compilation error	11 days ago
ApprovalTests_Catch1_Tests	e Switch back to user #includes, to see compiler warnings	2 months ago
ApprovalTests_Catch2_Tests	B Fix Windows CI build	11 days ago

Releases · approvals/ApprovalTes

GitHub, Inc. [US] | https://github.com/approvals/ApprovalTests.cpp/releases

Why GitHub? Enterprise Explore Marketplace Pricing Search / Sign in Sign up

approvals / ApprovalTests.cpp

Watch 12 Star 34 Fork 2

Code Issues 3 Pull requests 0 Security Insights

Releases Tags

Latest release

v.3.4.0

a3a0bb9

Single Hpp File

claremacrae released this 11 days ago · 2 commits to master since this release

- Breaking changes
  - None
- New features
  - Added AutoApproveIfMissingReporter

- Added a mechanism to shorten the names of output files, by omitting redundant Test Case Names, when using Google Test.
- Started adding hooks for TCR in Catch: (test && commit || revert)

- **Bug fixes**
  - Added virtual destructors for base classes with virtual methods
  - Suppress most unused-parameter warnings
  - Included `<stdexcept>` for `std::runtime_error` to fix build on Clang 5
- **Other changes**
  - In `ApprovalTestNamer`:
    - Added `getSourceFileName()`
    - Added `getOutputFileBaseName()`
    - Deprecated `getFileName()`

▼ Assets 3

ApprovalTests.v.3.4.0.hpp	73.1 KB
Source code (zip)	
Source code (tar.gz)	

Naming Options

- About that version number in the download…

- Could just call it by your preferred convention:

```
ApprovalTests.h
ApprovalTests.hpp
```

- Or use a wrapper and keep the version number, e.g.:

```
// In ApprovalTests.hpp
#include "ApprovalTests.v.3.4.0.hpp"
```

Getting Started with Approvals in C++

How to use it: Catch2 Boilerplate

- Your main.cpp

```
// main.cpp:
#define APPROVALS_CATCH
#include "ApprovalTests.hpp"
```

How to use it: Catch2 Boilerplate

- Your main.cpp

```
// main.cpp:
#define APPROVALS_CATCH
#include "ApprovalTests.hpp"
```

How to use it: Catch2 Boilerplate

- Your main.cpp

```
// main.cpp:
#define APPROVALS_CATCH
#include "ApprovalTests.hpp"
```

Pure Catch2 Test

- A test file

```
#include "Catch.hpp"

// Catch-only test
TEST_CASE( "Sums are calculated" )
{
    REQUIRE( 1 + 1 == 2 );
    REQUIRE( 1 + 2 == 3 );
}
```

Pure Catch2 Test

- A test file

```
#include "Catch.hpp"

// Catch-only test
TEST_CASE( "Sums are calculated" )
{
    REQUIRE( 1 + 1 == 2 );
    REQUIRE( 1 + 2 == 3 );
}
```

Pure Catch2 Test

- A test file

```
#include "Catch.hpp"

// Catch-only test
TEST_CASE( "Sums are calculated" )
{
    REQUIRE( 1 + 1 == 2 );
    REQUIRE( 1 + 2 == 3 );
}
```

Approvals Catch2 Test

- A test file (Test02.cpp)

```
#include "ApprovalTests.hpp"
#include "Catch.hpp"

// Approvals test - test static value, for demo purposes
TEST_CASE( "TestFixedInput" )
{
    Approvals::verify("Some\nMulti-line\noutput");
}
```

Approvals Catch2 Test

- A test file (Test02.cpp)

```
#include "ApprovalTests.hpp"
#include "Catch.hpp"

// Approvals test - test static value, for demo purposes
TEST_CASE( "TestFixedInput" )
{
    Approvals::verify("Some\nMulti-line\noutput");
}
```

Approvals Catch2 Test

- A test file (Test02.cpp)

```
#include "ApprovalTests.hpp"
#include "Catch.hpp"

// Approvals test - test static value, for demo purposes
TEST_CASE( "TestFixedInput" )
{
    Approvals::verify("Some\nMulti-line\noutput");
}
```

Approvals Catch2 Test

- A test file (Test02.cpp)

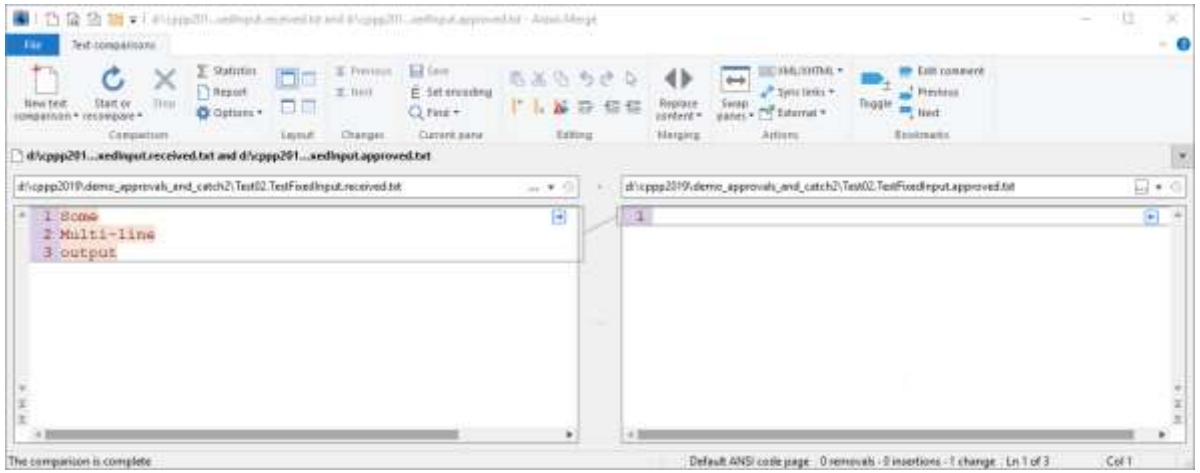
```
#include "ApprovalTests.hpp"
#include "Catch.hpp"

// Approvals test - test static value, for demo purposes
TEST_CASE( "TestFixedInput" )
{
    Approvals::verify("Some\nMulti-line\noutput");
}
```

First run

- 1. TestFixedInput failed

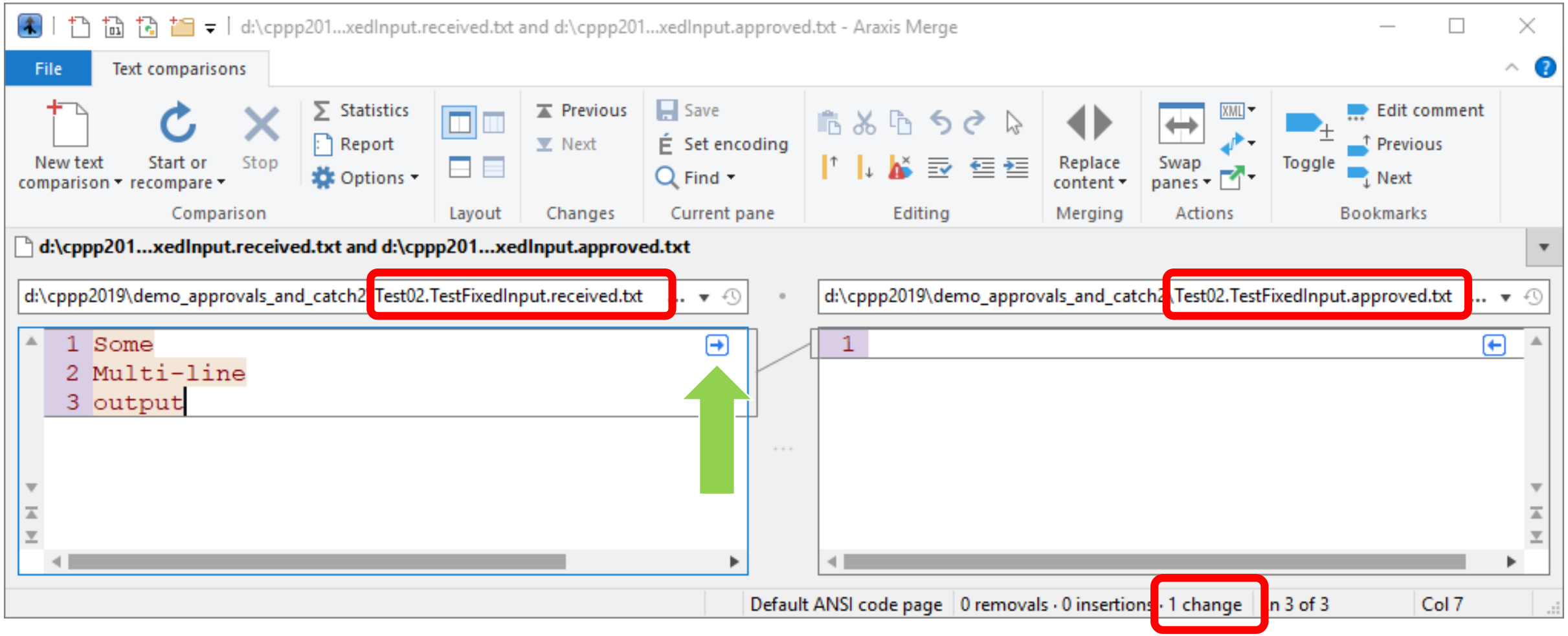
```
d:\cppp2019\demo_approvals_and_catch2\test02.cpp(5): exception: Failed Approval:
Approval File Not Found
File: "d:\cppp2019\demo_approvals_and_catch2\Test02.TestFixedInput.approved.txt"
```

- 2. Differencing tool pops up (Araxis Merge, in this case)



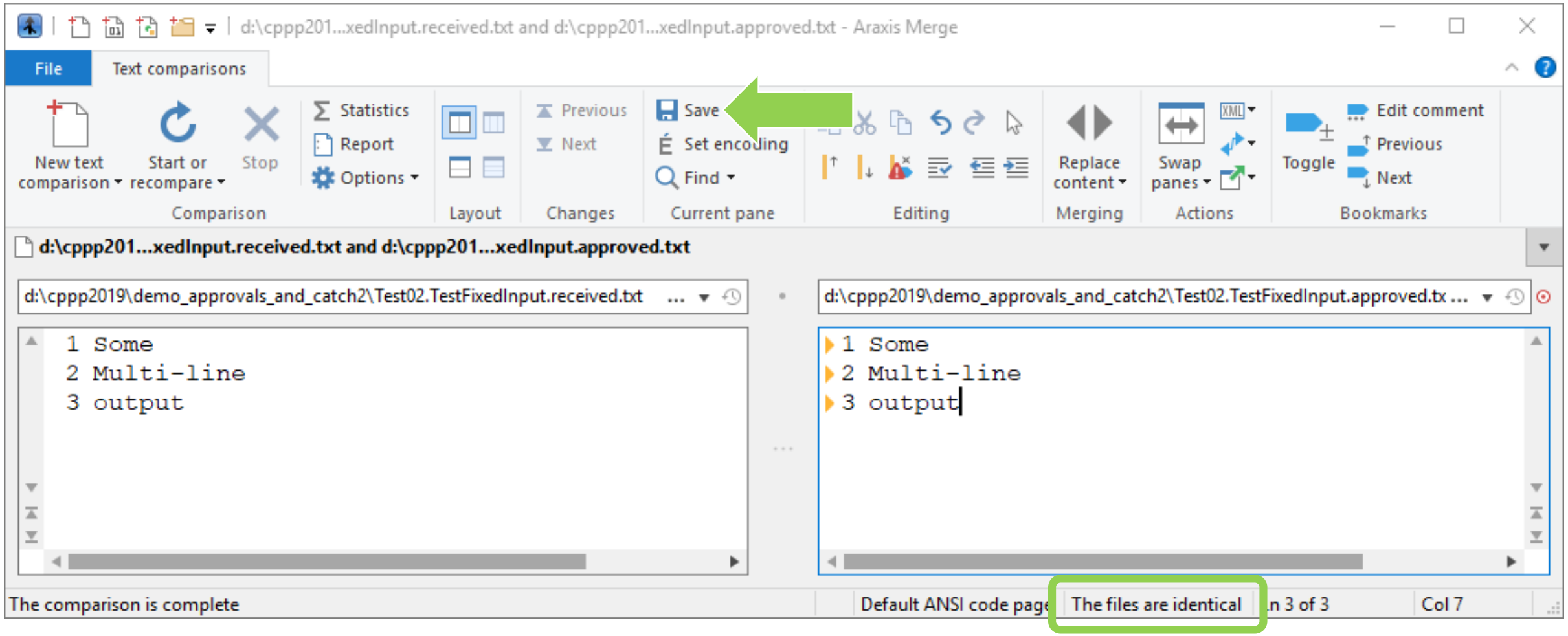
Actual/Received

Expected/Approved



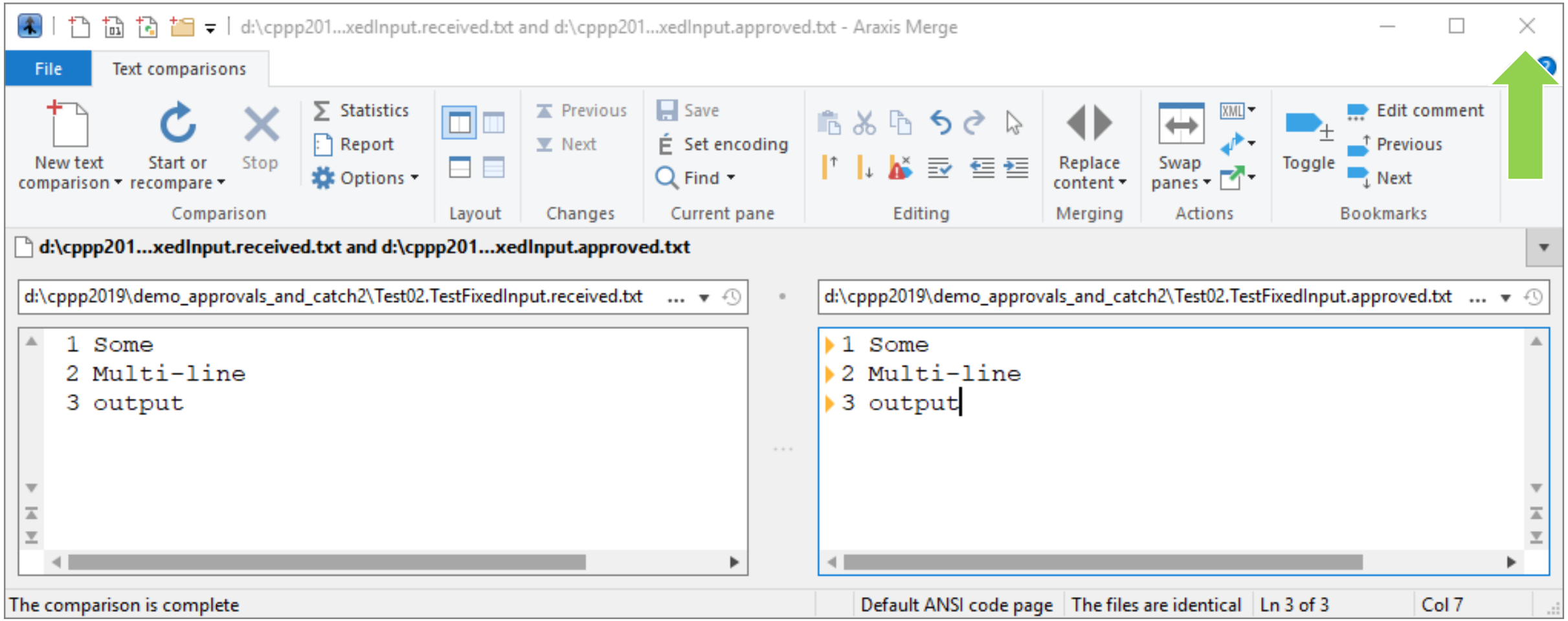
Actual/Received

Expected/Approved



Actual/Received

Expected/Approved



Second run

- I approved the output

```
TestFixedInput passed
```

- Then we commit the test, and the approved file(s) to version control
- The diffing tool only shows up if:
  - there is not (yet) an Approved file or
  - the Received differs from the Approved

What’s going on here?

- It’s a very convenient form of Golden Master testing
- Objects being tested are stringified – that’s a requirement
- Captures snapshot of current behaviour
- Useful for locking down behaviour of existing code
  - Not intended to replace Unit Tests

Example test directory

```
Test02.cpp
Test02.TestFixedInput.approved.txt
Test03CustomReporters.cpp
Test03CustomReporters.UseCustomReporter.approved.txt
Test03CustomReporters.UseQuietReporter.approved.txt
Test03CustomReporters.UseSpecificReporter.approved.txt
Test04ConsoleReporter.cpp
Test04ConsoleReporter.UseConsoleReporter.approved.txt
Test04ConsoleReporter.UseConsoleReporter.received.txt
```

How to use it: Catch2 Boilerplate

- Your main.cpp

```
#define APPROVALS_CATCH
#include "ApprovalTests.hpp“

// Put all approved and received files in "approval_tests/"
auto dir =
   Approvals::useApprovalsSubdirectory("approval_tests");
```

Example test directory

```
Test02.cpp
Test03CustomReporters.cpp
Test04ConsoleReporter.cpp

approval_tests/
    Test02.TestFixedInput.approved.txt
    Test03CustomReporters.UseCustomReporter.approved.txt
    Test03CustomReporters.UseQuietReporter.approved.txt
    Test03CustomReporters.UseSpecificReporter.approved.txt
    Test04ConsoleReporter.UseConsoleReporter.approved.txt
    Test04ConsoleReporter.UseConsoleReporter.received.txt
```

Delving Deeper

Reference: Evolving User Guide

- https://github.com/approvals/ApprovalTests.cpp/tree/master/doc#top

ApprovalTests.cpp User Guide

Contents

Reference: Evolving User Guide

Setup

- Obtaining ApprovalTests.cpp
  - Get a test framework
  - Downloading the latest release
    - Adding a wrapper header
    - Renaming the header file to remove version number
  - ApprovalTests.cpp.StarterProject
  - Cloning repo and using CMake
- Getting Started - setting up `main()`
- Using Approval Tests With Catch
- Using Approval Tests With Google Tests

Reference: Evolving User Guide

Writing Tests

- Tutorial
- Testing single objects
- Testing containers
- Testing exceptions
- Testing combinations - containers of containers (of containers...)
- To String
- Worked example of getting to 'make the thing; verify the thing' - 'do; verify'
- Features

Reference: Evolving User Guide

Extras

- Troubleshooting
- 4 benefits of testing
  - Spec
  - Feedback
  - Regression
  - Granularity

Advanced Topics

- Supporting a new test framework
- Contributing to ApprovalTests.cpp

A note on Tools

- I'll be using a variety of tools in the talk
- I'll mention them as I go
- Slides of references at the end

How does this help with legacy code?

Applying this to legacy code

```
TEST_CASE("New test of legacy feature")
{
    // Standard pattern:
    // Wrap your legacy feature to test in a function call
    // that returns some state that can be written to a text file
    // for verification:
    const LegacyThing result = doLegacyOperation();
    Approvals::verify(result);
}
```

Implementation

```
class LegacyThing;

std::ostream &operator<<(std::ostream &os, const LegacyThing &result) {
    // write interesting info from result here:
    os << result…;
    return os;
}
LegacyThing doLegacyOperation() {
    // your implementation here..
    return LegacyThing();
}
```

Feature: Consistency over machines

- Naming of output files
- Approved files version controlled

Feature: Consistency over Operating Systems

- Line-endings

Feature: Consistency over Languages

- Consistent concepts and nomenclature in all the implementations

Feature: Quick to write tests

```
TEST_CASE("verifyAllWithHeaderBeginEndAndLambda")
{
  std::list<int> numbers{ 0, 1, 2, 3};
  // Multiple convenience overloads of Approvals::verifyAll()
  Approvals::verifyAll(
    "Test Squares", numbers.begin(), numbers.end(),
    [](int v, std::ostream& s) { s << v << " => " << v*v << '\n' ; });
}
```

Feature: Quick to get good coverage

```
TEST_CASE("verifyAllCombinationsWithLambda")
{
    std::vector<std::string> strings{"hello", "world"};
    std::vector<int> numbers{1, 2, 3};
    CombinationApprovals::verifyAllCombinations<
        std::vector<std::string>,       // The type of element in our first input container
        std::vector<int>,               // The type of element in our second input container
        std::string>(                   // The return type from testing one combination of inputs
            // Lambda that acts on one combination of inputs, and returns the result to be approved:
            [](std::string s, int i) { return s + " " + std::to_string(i); },
            strings,                    // The first input container
            numbers);                   // The second input container
}
```

All values in single output file:

```
(hello, 1) => hello 1
(hello, 2) => hello 2
(hello, 3) => hello 3
(world, 1) => world 1
(world, 2) => world 2
(world, 3) => world 3
```

Customisability: Reporters

- Reporters act on test failure
- Very simple but powerful abstraction
- Gives complete user control over how to inspect and act on a failure
- If you adopt Approvals, do review the supplied reporters for ideas

Feature: Change the Reporter in one test

```
TEST_CASE("Demo custom reporter")
{
    std::vector<std::string> v{"hello", "world"};
    Approvals::verifyAll(v, MyCustomReporter());
}
```

Feature: Change the default Reporter

```
// main.cpp - or in individual tests:
#include <memory>

auto disposer = Approvals::useAsDefaultReporter(
    std::make_shared<Windows::AraxisMergeReporter>() );

auto disposer = Approvals::useAsDefaultReporter(
    std::make_shared<GenericDiffReporter>(
        "E:/Program Files/Araxis/Araxis Merge/Compare.exe") );
```

Feature: Change the default Reporter

```
// main.cpp - or in individual tests:
#include <memory>

auto disposer = Approvals::useAsDefaultReporter(
    std::make_shared<Windows::AraxisMergeReporter>() );

auto disposer = Approvals::useAsDefaultReporter(
    std::make_shared<GenericDiffReporter>(
        "E:/Program Files/Araxis/Araxis Merge/Compare.exe") );
```

Feature: Change the default Reporter

```
// main.cpp - or in individual tests:
#include <memory>

auto disposer = Approvals::useAsDefaultReporter(
    std::make_shared<Windows::AraxisMergeReporter>() );

auto disposer = Approvals::useAsDefaultReporter(
    std::make_shared<GenericDiffReporter>(
        "E:/Program Files/Araxis/Araxis Merge/Compare.exe") );
```

Feature: Don’t run GUIs on build servers

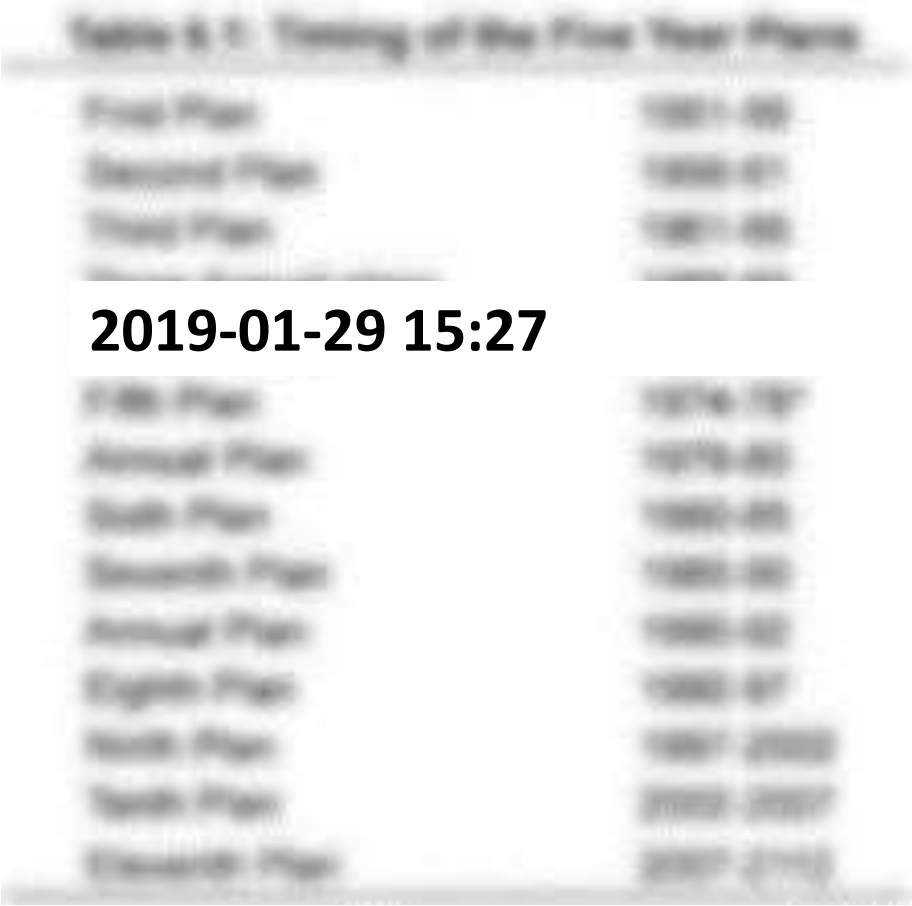
```
// main.cpp
auto disposer = Approvals::useAsFrontLoadedReporter(
    BlockingReporter::onMachineNamed("MyCIMachineName") );
```

Feature: Convention over Configuration

- Fewer decisions for developers
- Users only specify unusual behaviours

Challenge: Golden Master is a log file

- Dates and times?
- Object addresses?



Options for unstable output

- Introduce date-time abstraction?
- Customised comparison function?
- Or: strip dates from the log file

Tip: Rewrite output file

Customisability: ApprovalWriter interface

```
class ApprovalWriter
{
public:
    virtual std::string getFileExtensionWithDot() = 0;
    virtual void write(std::string path) = 0;
    virtual void cleanUpReceived(std::string receivedPath) = 0;
};
```

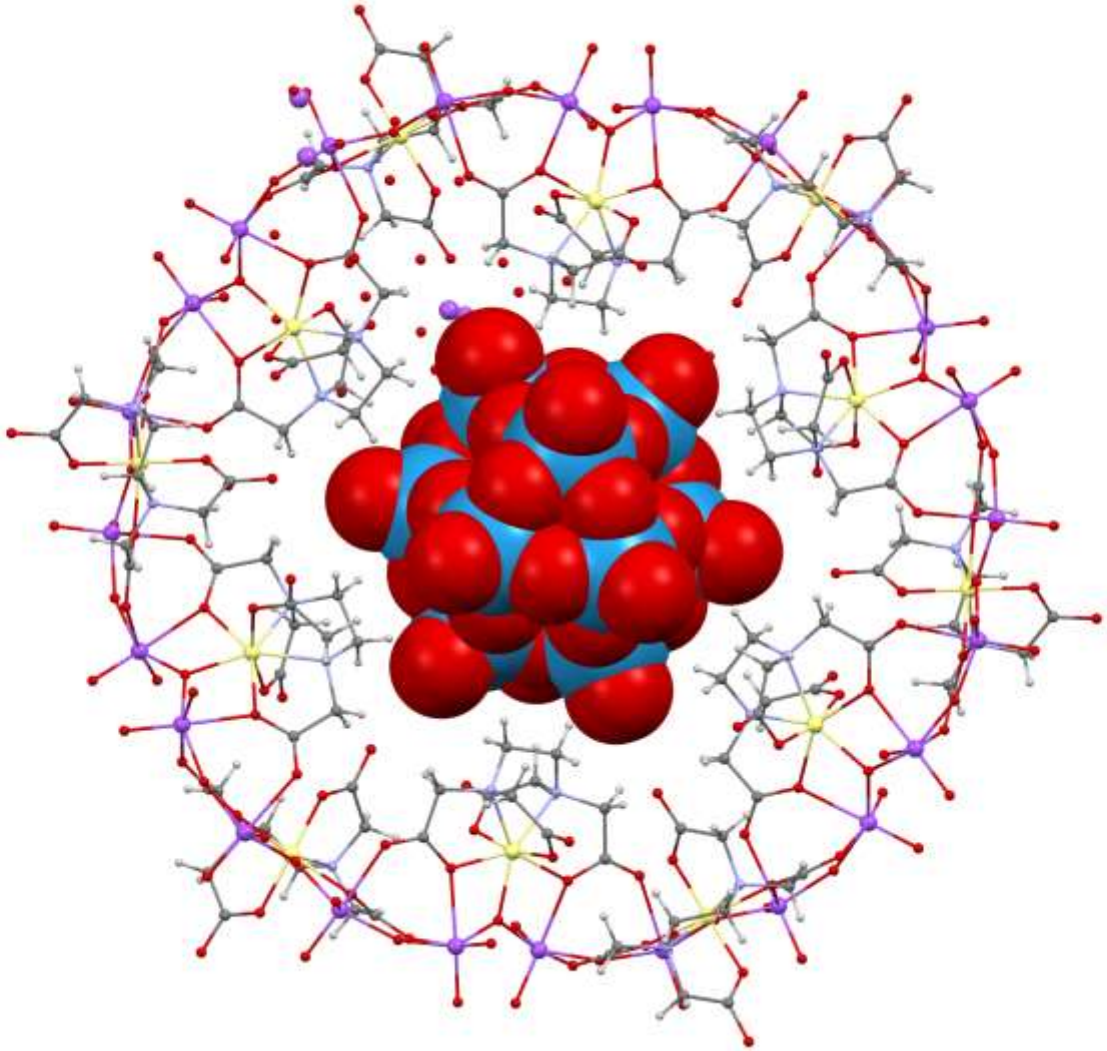
Flexibility gives non-obvious power

- Approval Tests approach is deceptively simple
- Is a lot more than Golden Master
- Reporter could:
  - Convert numbers to picture for comparison
  - Or Excel file
- ApprovalWriter could:
  - Write out a hash value of contents of very large file

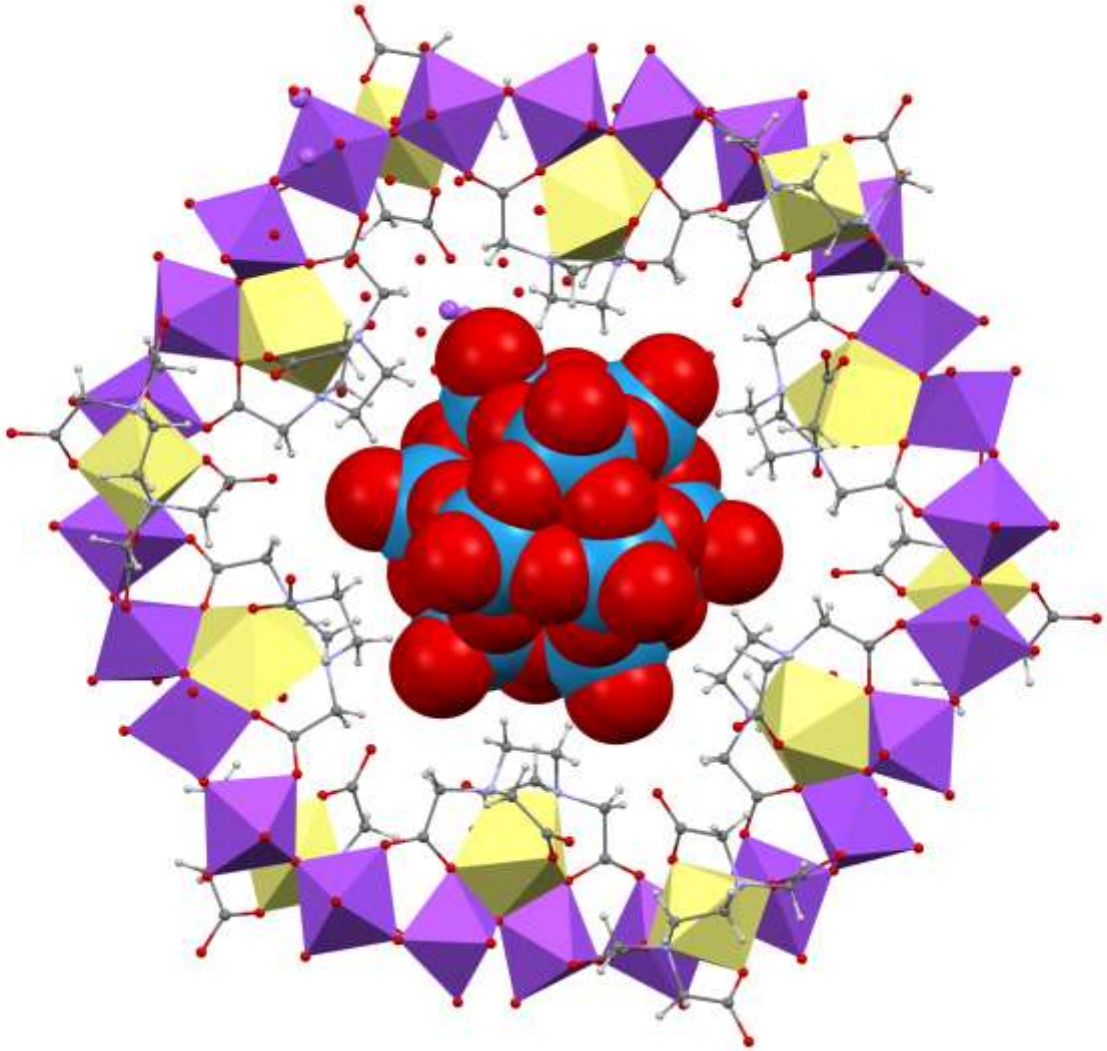
Contents

Requirement

What I could do:

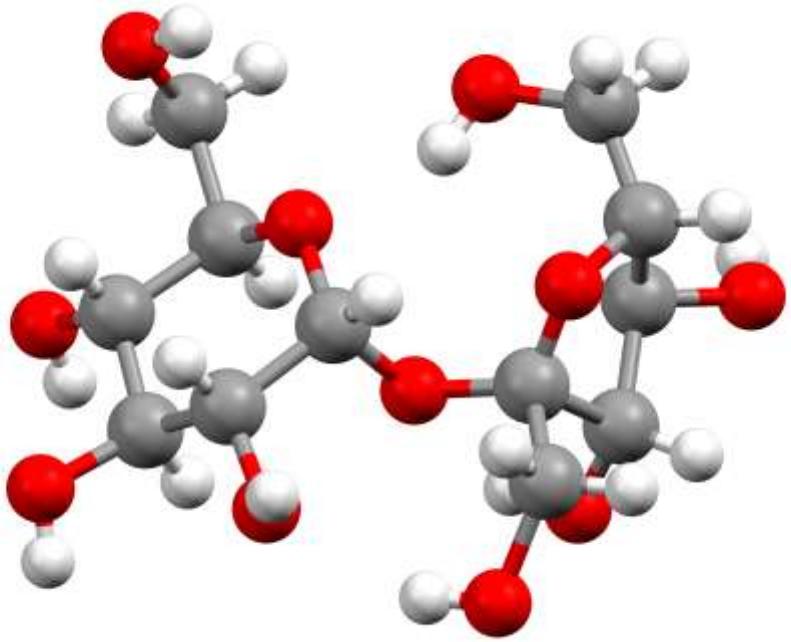


What I needed to do:

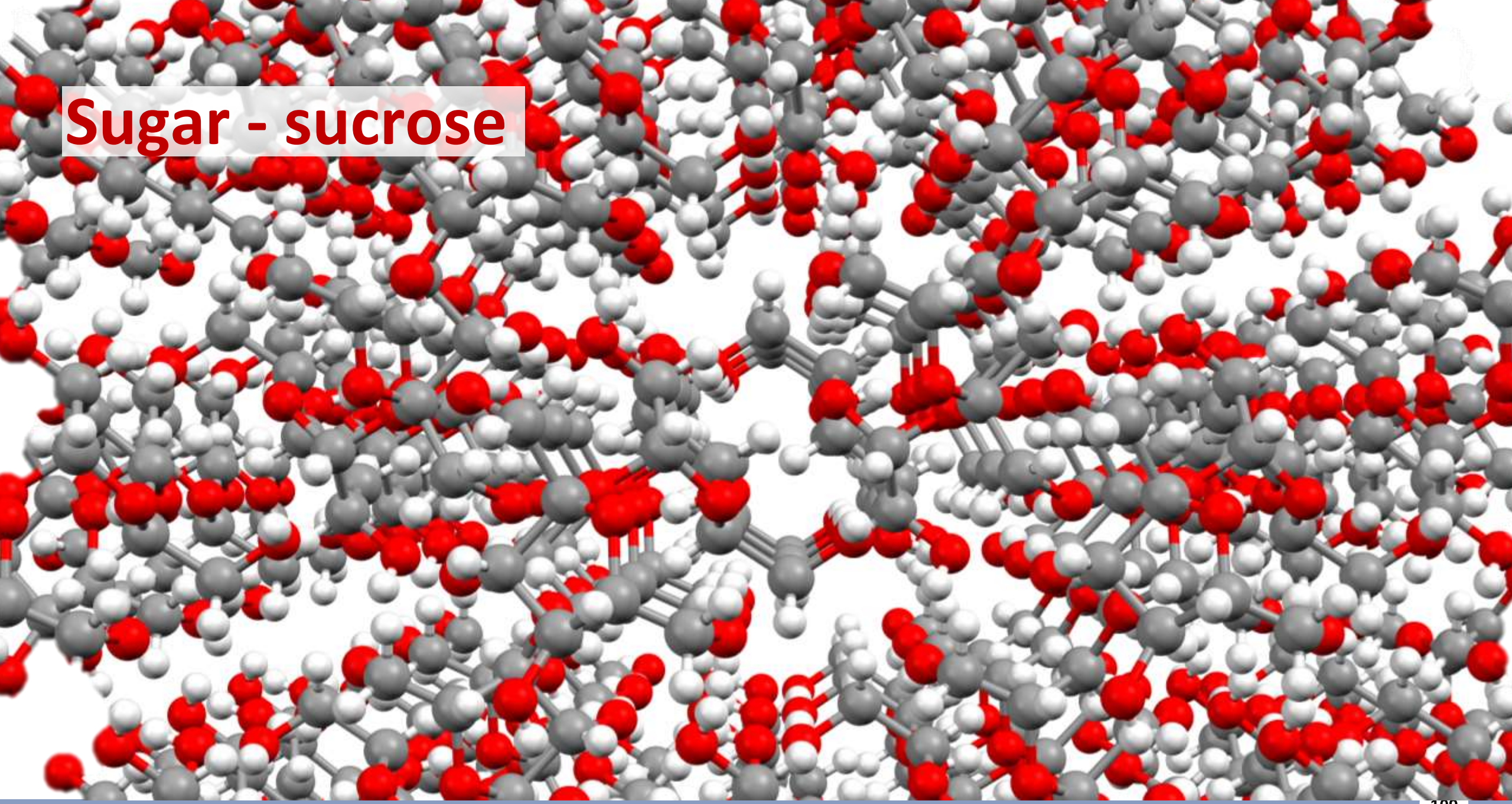


Sugar - sucrose

Sugar - sucrose



Sugar - sucrose



Approving Images

What I'm aiming for

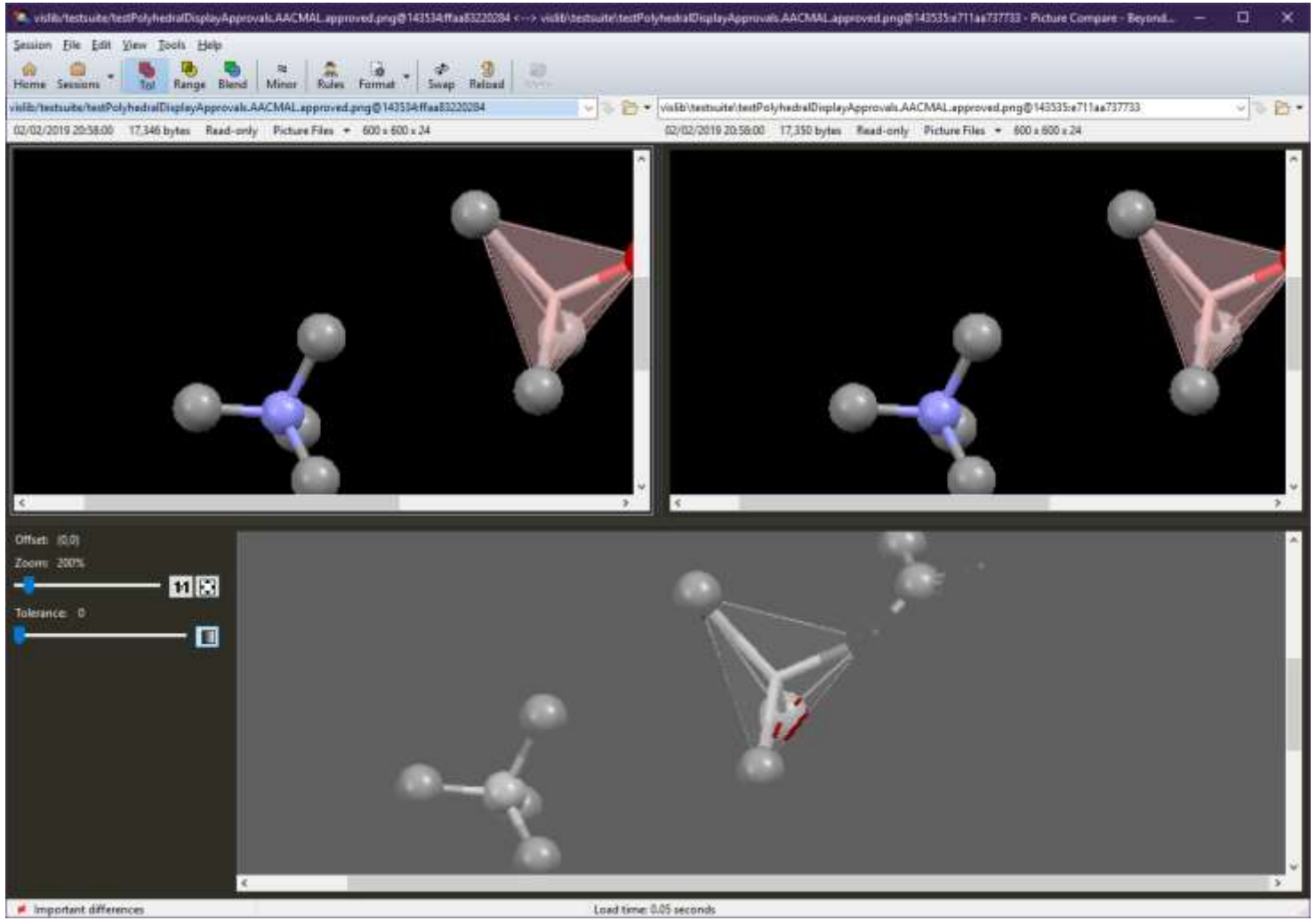
```
TEST(PolyhedralGraphicsStyleApprovals, CUVXAT)
{
    // CUVXAT identifies a crystal structure in the database
    // For code, please download the slides
    const QImage crystal_picture = loadEntry("CUVXAT");
    verifyQImage(crystal_picture, *currentReporter());
}
```

Foundation of Approval tests

```
void FileApprover::verify(
    ApprovalNamer& namer,
    ApprovalWriter& writer,
    const Reporter& reporter);
```

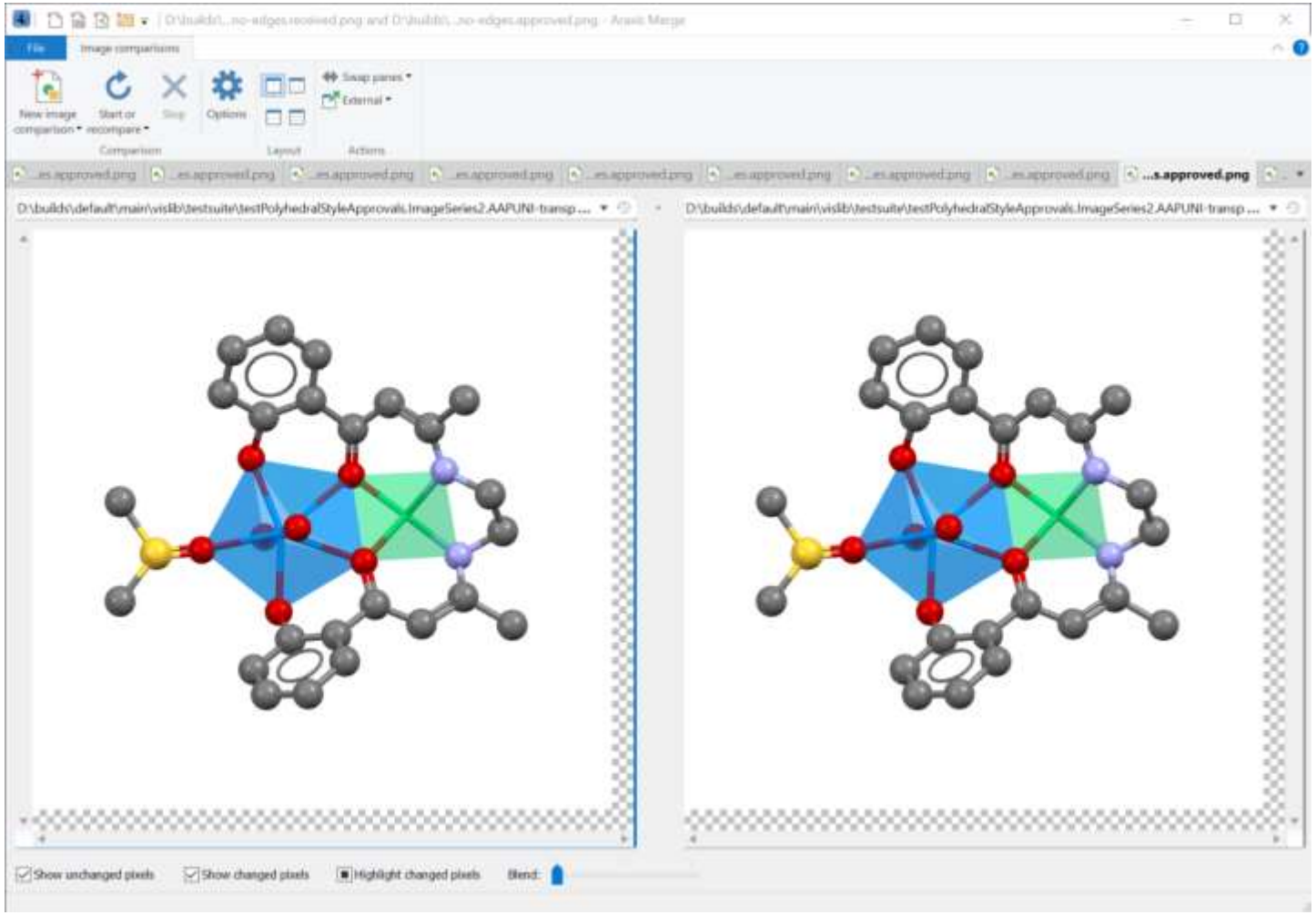
Useful feedback

BeyondCompare 4

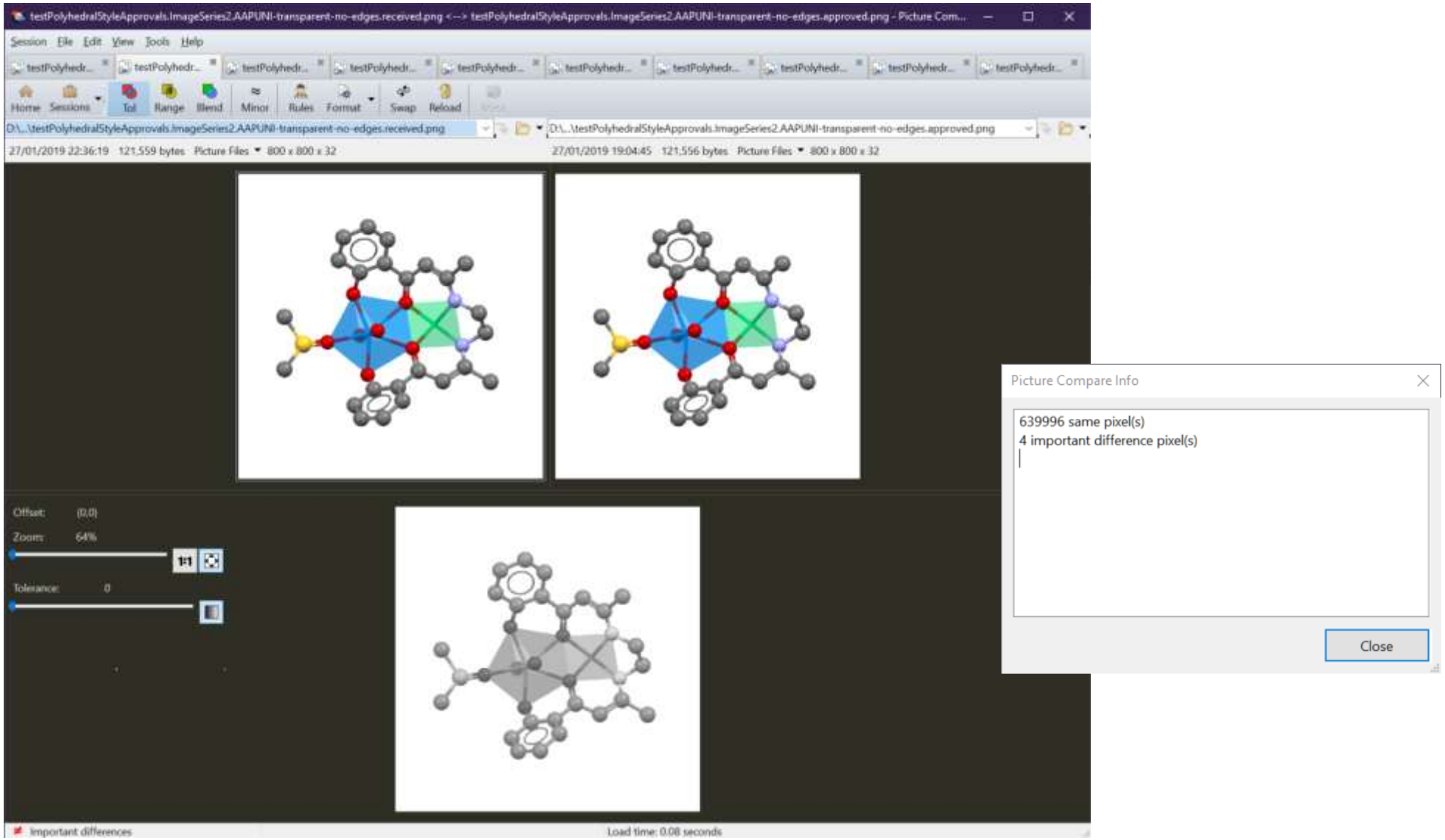


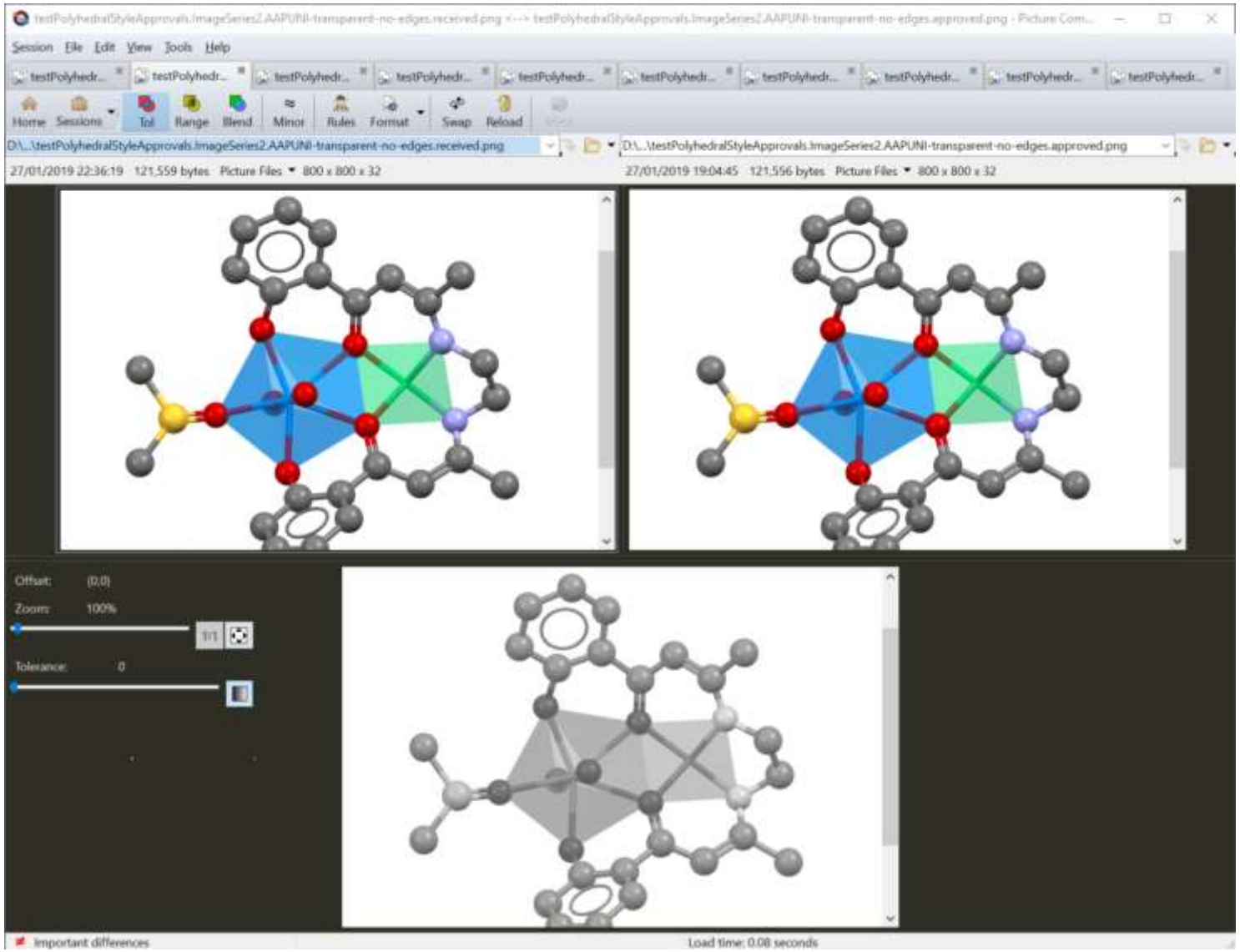
What? Re-running now gives random failures

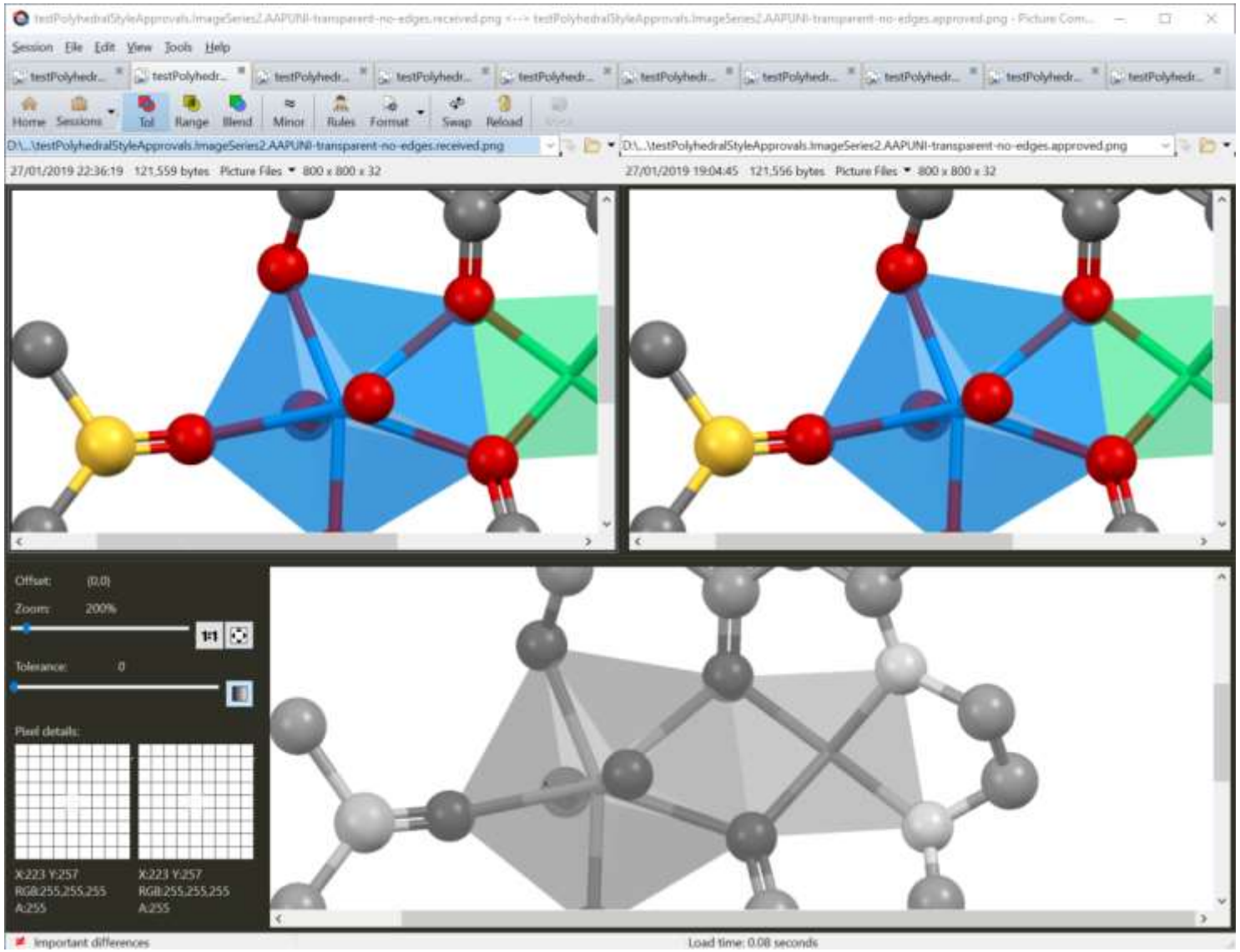
Araxis Merge

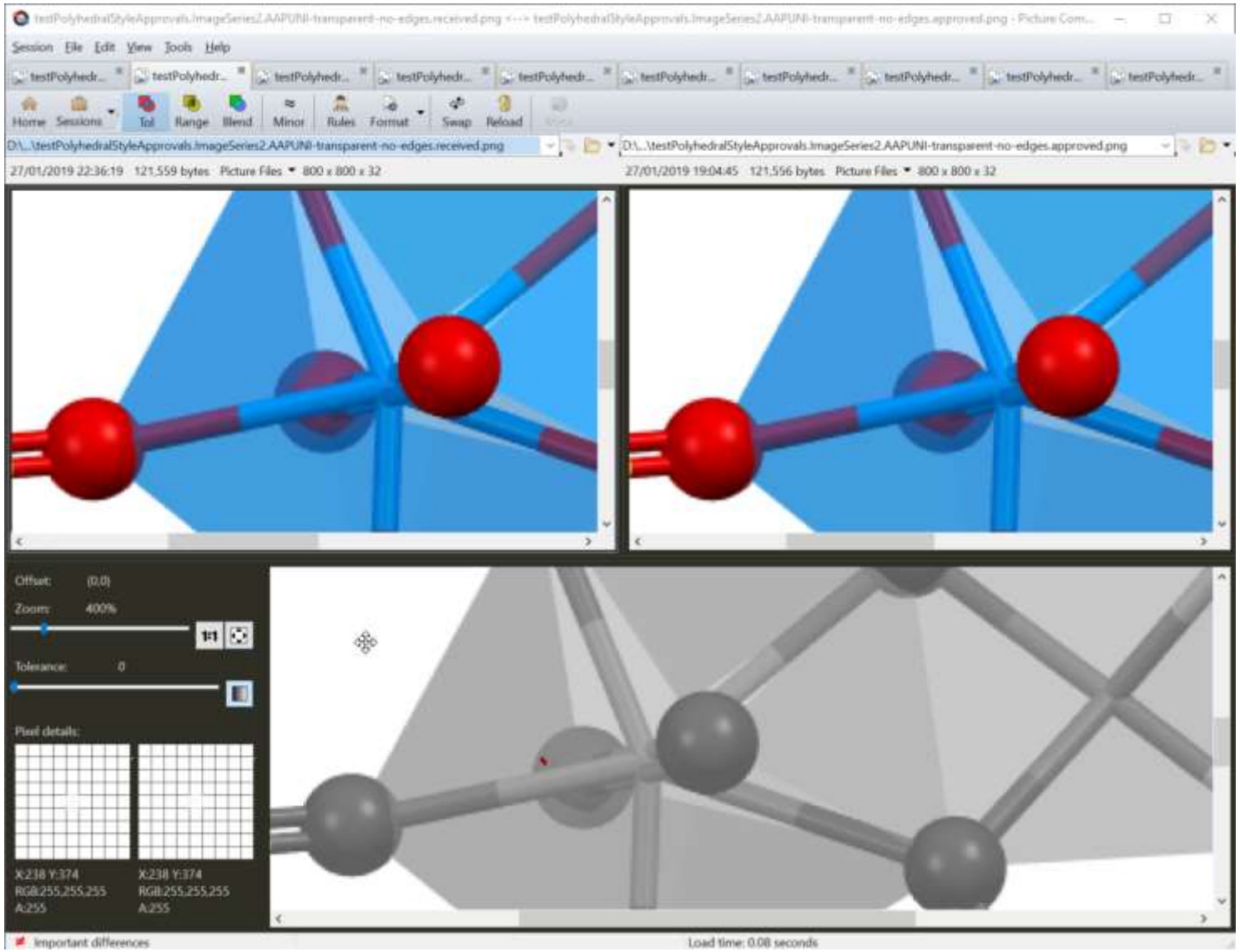


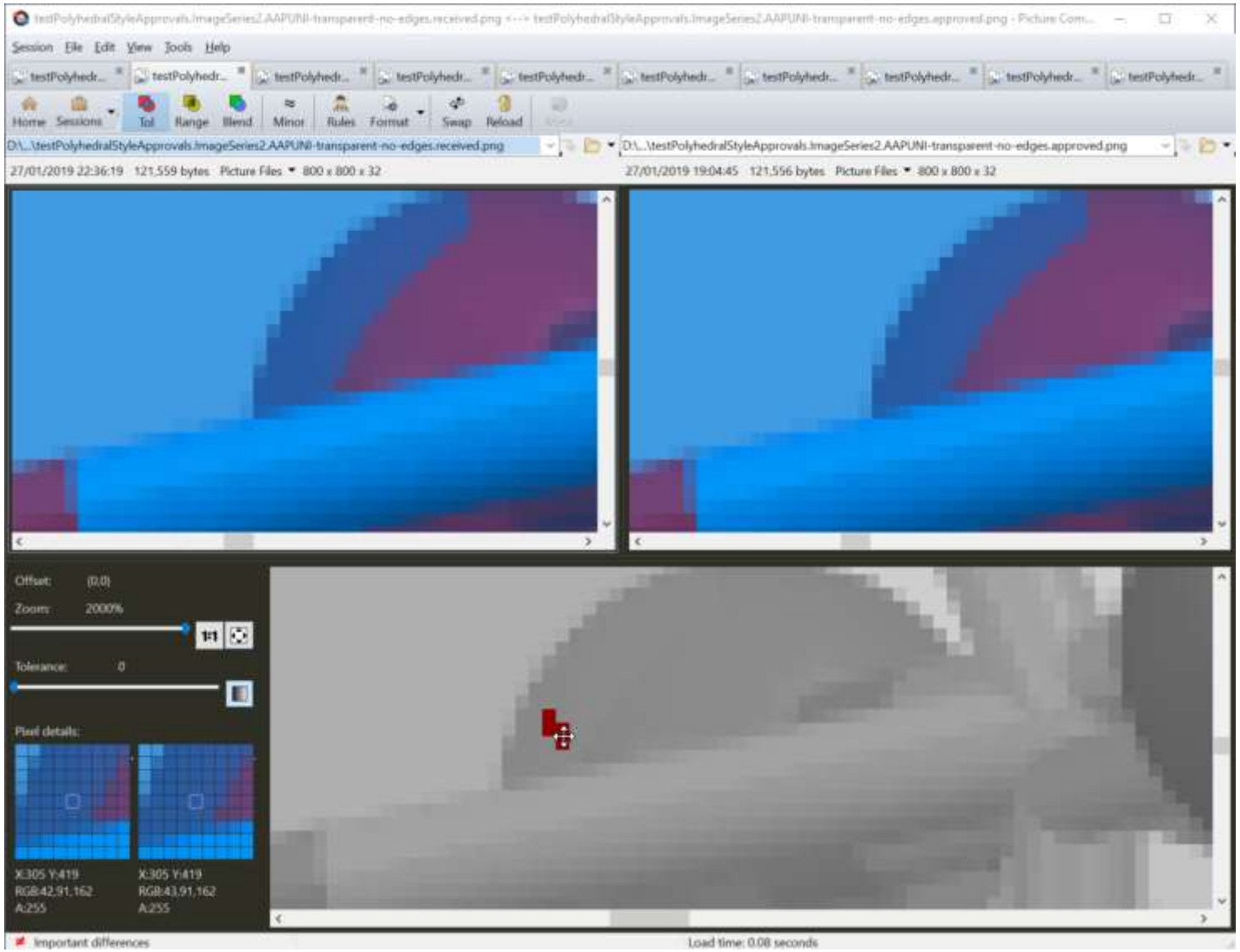
BeyondCompare 4

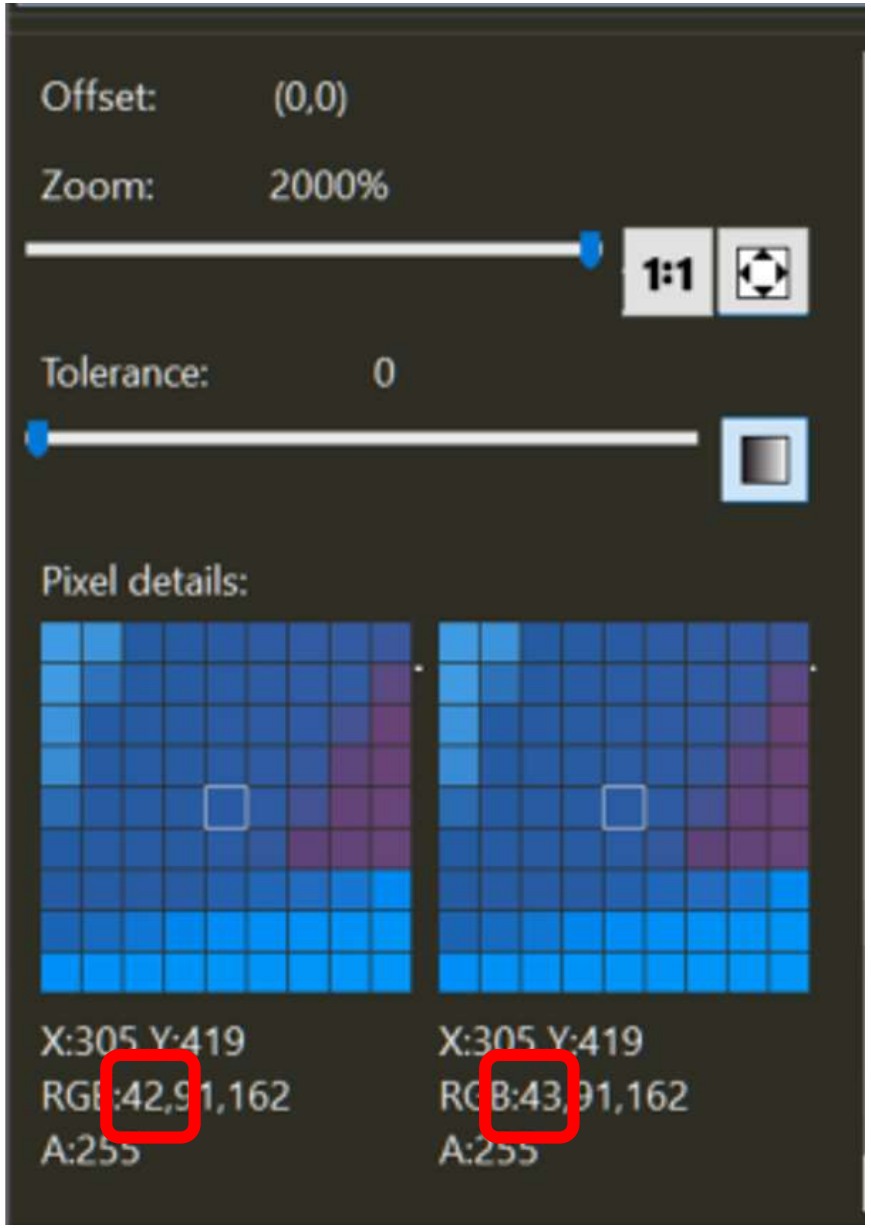



Session File Edit View Tools Help
Home Sessions Tol Range Blend Minor Rules Format Swap Reload
D:\...\testPolyhedralStyleApprovals.ImageSeries2.AAPUNI-transparent-no-edges.received.png
D:\...\testPolyhedralStyleApprovals.ImageSeries2.AAPUNI-transparent-no-edges.approved.png
27/01/2019 22:36:19 121,559 bytes Picture Files 800 x 800 x 32
27/01/2019 19:04:45 121,556 bytes Picture Files 800 x 800 x 32
Offset: (0,0)
Zoom: 100%
1:1
Tolerance: 0
Important differences
Load time: 0.08 seconds


Session File Edit View Tools Help
Home Sessions Tol Range Blend Minor Rules Format Swap Reload
D:\...\testPolyhedralStyleApprovals.ImageSeries2.AAPUNI-transparent-no-edges.received.png
D:\...\testPolyhedralStyleApprovals.ImageSeries2.AAPUNI-transparent-no-edges.approved.png
27/01/2019 22:36:19 121,559 bytes Picture Files 800 x 800 x 32
27/01/2019 19:04:45 121,556 bytes Picture Files 800 x 800 x 32
Offset: (0,0)
Zoom: 200%
Tolerance: 0
Pixel details:
X:223 Y:257
RGB:255,255,255
A:255
X:223 Y:257
RGB:255,255,255
A:255
Important differences
Load time: 0.08 seconds


Session File Edit View Tools Help
Home Sessions Tol Range Blend Minor Rules Format Swap Reload
D:\...\testPolyhedralStyleApprovals.ImageSeries2.AAPUNI-transparent-no-edges.received.png
D:\...\testPolyhedralStyleApprovals.ImageSeries2.AAPUNI-transparent-no-edges.approved.png
27/01/2019 22:36:19 121,559 bytes Picture Files 800 x 800 x 32
27/01/2019 19:04:45 121,556 bytes Picture Files 800 x 800 x 32
Offset: (0,0)
Zoom: 400%
Tolerance: 0
Pixel details:
X:238 Y:374
RGB:255,255,255
A:255
X:238 Y:374
RGB:255,255,255
A:255
Important differences
Load time: 0.08 seconds


Session File Edit View Tools Help
Home Sessions Tol Range Blend Minor Rules Format Swap Reload
D:\...\testPolyhedralStyleApprovals.ImageSeries2.AAPUNI-transparent-no-edges.received.png
D:\...\testPolyhedralStyleApprovals.ImageSeries2.AAPUNI-transparent-no-edges.approved.png
27/01/2019 22:36:19 121,559 bytes Picture Files 800 x 800 x 32
27/01/2019 19:04:45 121,556 bytes Picture Files 800 x 800 x 32
Offset: (0,0)
Zoom: 2000%
Tolerance: 0
Pixel details:
X:305 Y:419
RGB:42,91,162
A:255
X:305 Y:419
RGB:43,91,162
A:255
Important differences
Load time: 0.08 seconds


Offset: (0,0)
Zoom: 2000%
1:1
Tolerance: 0
Pixel details:
X:305 Y:419
RGB:42,91,162
A:255
X:305 Y:419
RGB:43,91,162
A:255

Customisability: ApprovalComparator

```
class ApprovalComparator
{
public:
    virtual ~ApprovalComparator() = default;

    virtual bool contentsAreEquivalent(std::string receivedPath,
                                       std::string approvedPath) const = 0;
};
```

Create custom QImage comparison class

```
// Allow differences in up to 1/255 of RGB values at each pixel, as saved in 32-bit images
// sometimes have a few slightly different pixels, which are not visible to the human eye.
class QImageApprovalComparator : public ApprovalComparator
{
public:
    bool contentsAreEquivalent(std::string receivedPath, std::string approvedPath) const override
    {
        const QImage receivedImage(QString::fromStdString(receivedPath));
        const QImage approvedImage(QString::fromStdString(approvedPath));

        return compareQImageIgnoringTinyDiffs(receivedImage, approvedImage);
    }
};
```

Register the custom comparison class

```
// Somewhere in main…
FileApprover::registerComparator(
    ".png",
    std::make_shared<QImageApprovalComparator>());
```

Does the difference matter?

- Legacy code is often brittle
- Testing makes changes visible
- Then decide if change matters
- Fast feedback cycle for efficient development

Looking back

- User perspective
- Learned about own code
- Enabled unit tests for graphics problems
- Approvals useful even for temporary tests

ApprovalTests Customisation Points

- `Reporter`
- `ApprovalWriter`
- `ApprovalComparator`
- `ApprovalNamer`

- More documentation of these coming soon

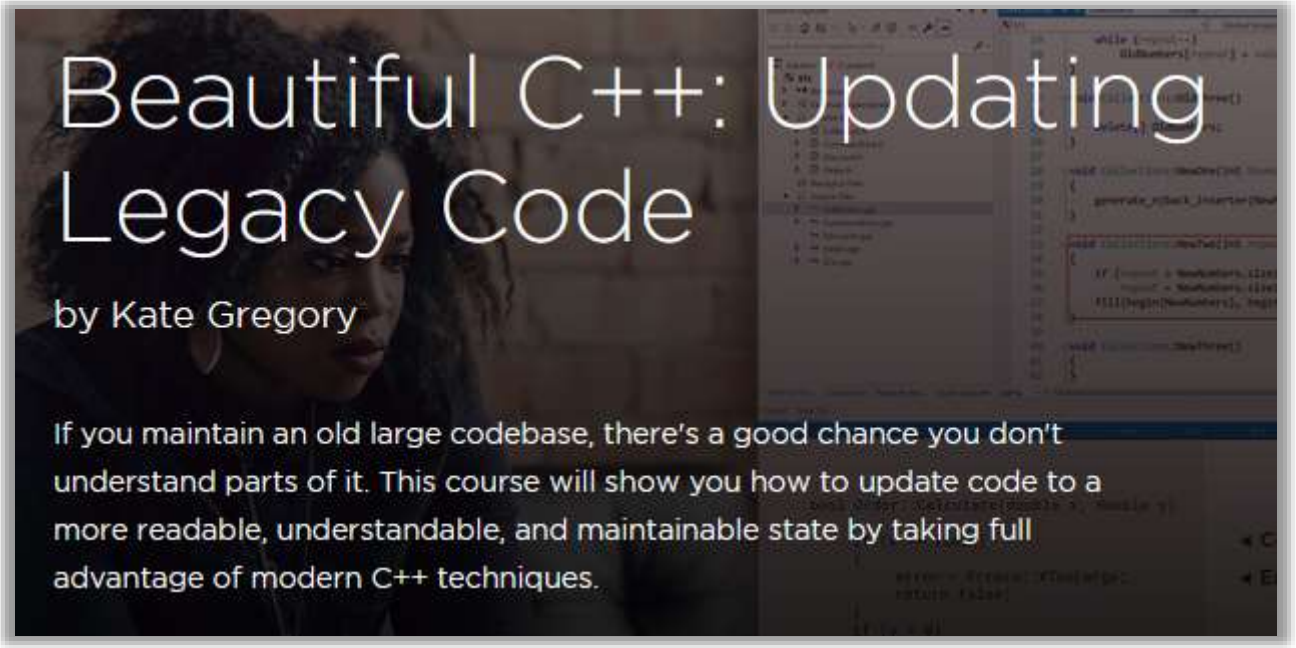
Contents

References: Tools Used

- Diffing
  - Araxis Merge: https://www.araxis.com/merge/
  - Beyond Compare: https://www.scootersoftware.com/
- Code Coverage
  - OpenCppCoverage and Visual Studio Plugin: https://github.com/OpenCppCoverage
  - BullseyeCoverage: https://www.bullseye.com/

Beautiful C++: Updating Legacy Code

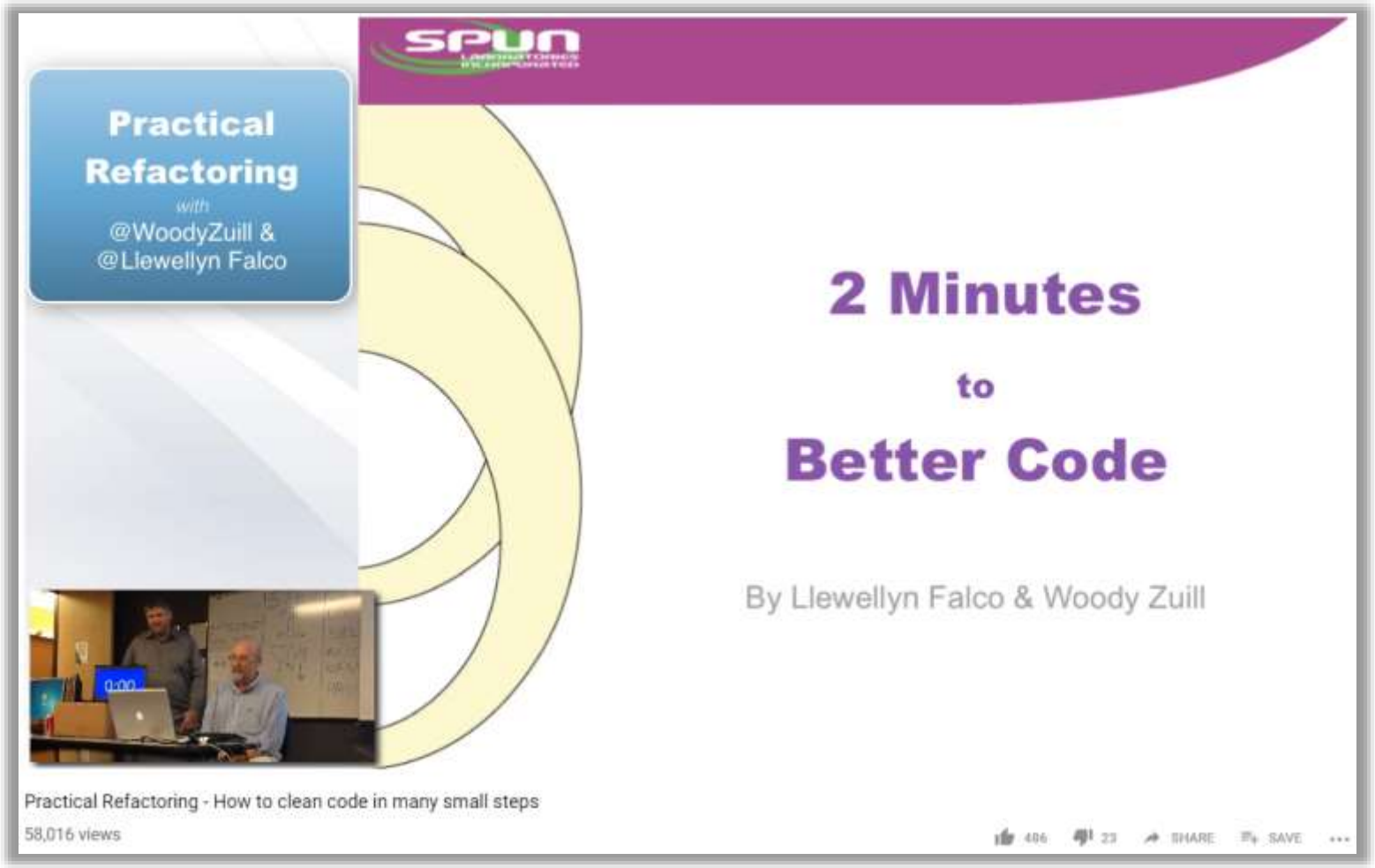
- https://app.pluralsight.com/library/courses/cpp-updating-legacy-code/table-of-contents



The Legacy Code Programmer’s Toolbox

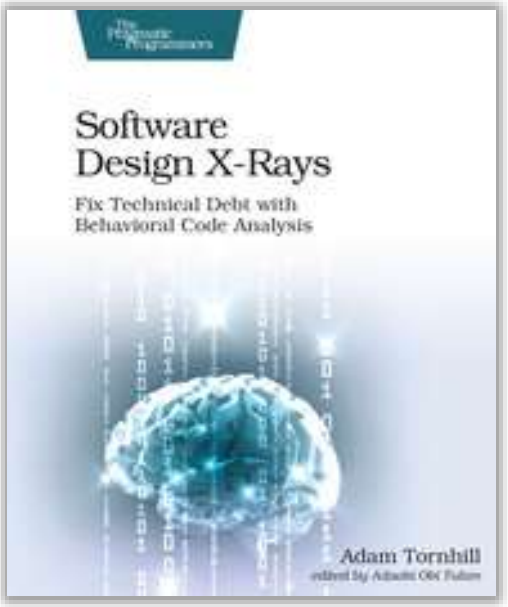
- https://leanpub.com/legacycode
- The Legacy Code Programmer's Toolbox: Practical skills for software professionals working with legacy code, by Jonathan Boccara

Videos


Practical Refactoring
with
@WoodyZuill &
@Llewellyn Falco
2 Minutes
to
Better Code
By Llewellyn Falco & Woody Zuill
Practical Refactoring - How to clean code in many small steps
58,016 views
SHARE
SAVE

Adam Tornhill's Books

- Mining actionable information from historical code bases (by Adam Tornhill)
  - Your Code as a Crime Scene
  - Software Design X-Rays: Fix Technical Debt with Behavioural Code Analysis



Contents

Adopting legacy code

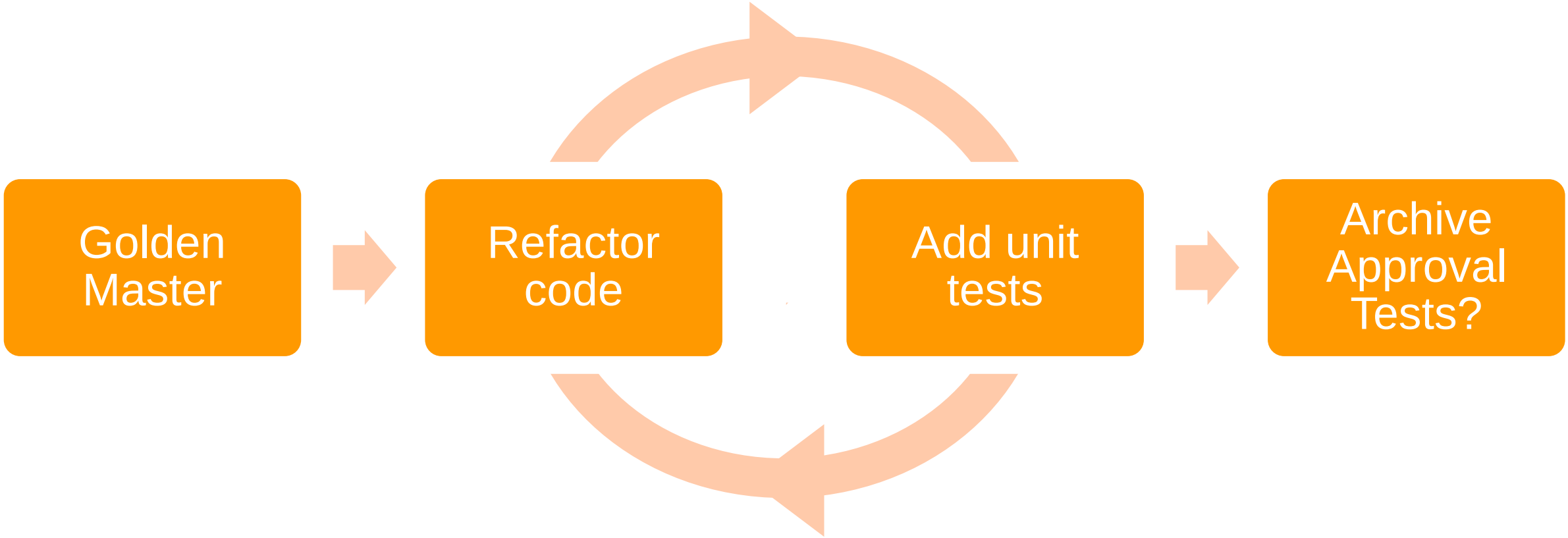
- You can do it!
- Evaluate current tests
- Quickly improve coverage with Golden Master
- ApprovalTests.cpp makes that really easy
- Even for non-text types

ApprovalTests

- Powerful Golden Master tool
- verify(), verifyAll(), verifyAllCombinations()
- Adjust output files, after writing, to simplify comparison

ApprovalTests.cpp

- Not (always) a replacement for Unit Tests!



Thank You: Any Questions?

- Example Code: https://github.com/claremacrae/cppp2019/

- Slides: https://www.slideshare.net/ClareMacrae
  - (early next week)

- ApprovalTests.cpp
  - https://github.com/approvals/ApprovalTests.cpp
  - https://github.com/approvals/ApprovalTests.cpp.StarterProject