

Quickly Testing Legacy Code

Clare Macrae
5 February 2019



@ClareMacraeUK





Contents

- Introduction
- Legacy Code
- Golden Master
- Approval Tests
- Example
- Resources
- Summary



Contents

- **Introduction** ←
- Legacy Code
- Golden Master
- Approval Tests
- Example
- Resources
- Summary



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



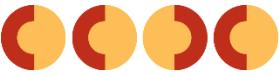
Clare Macrae

@ClareMacraeUK

Replies to [@LlewellynFalco](#)

Would be interested in hearing more.

2:46 PM - 26 Nov 2017



A year of remote pairing later...



Goal:
**Share techniques for easier testing
in challenging scenarios**



Contents

- Introduction
- **Legacy Code** ←
- Golden Master
- Approval Tests
- Example
- Resources
- Summary



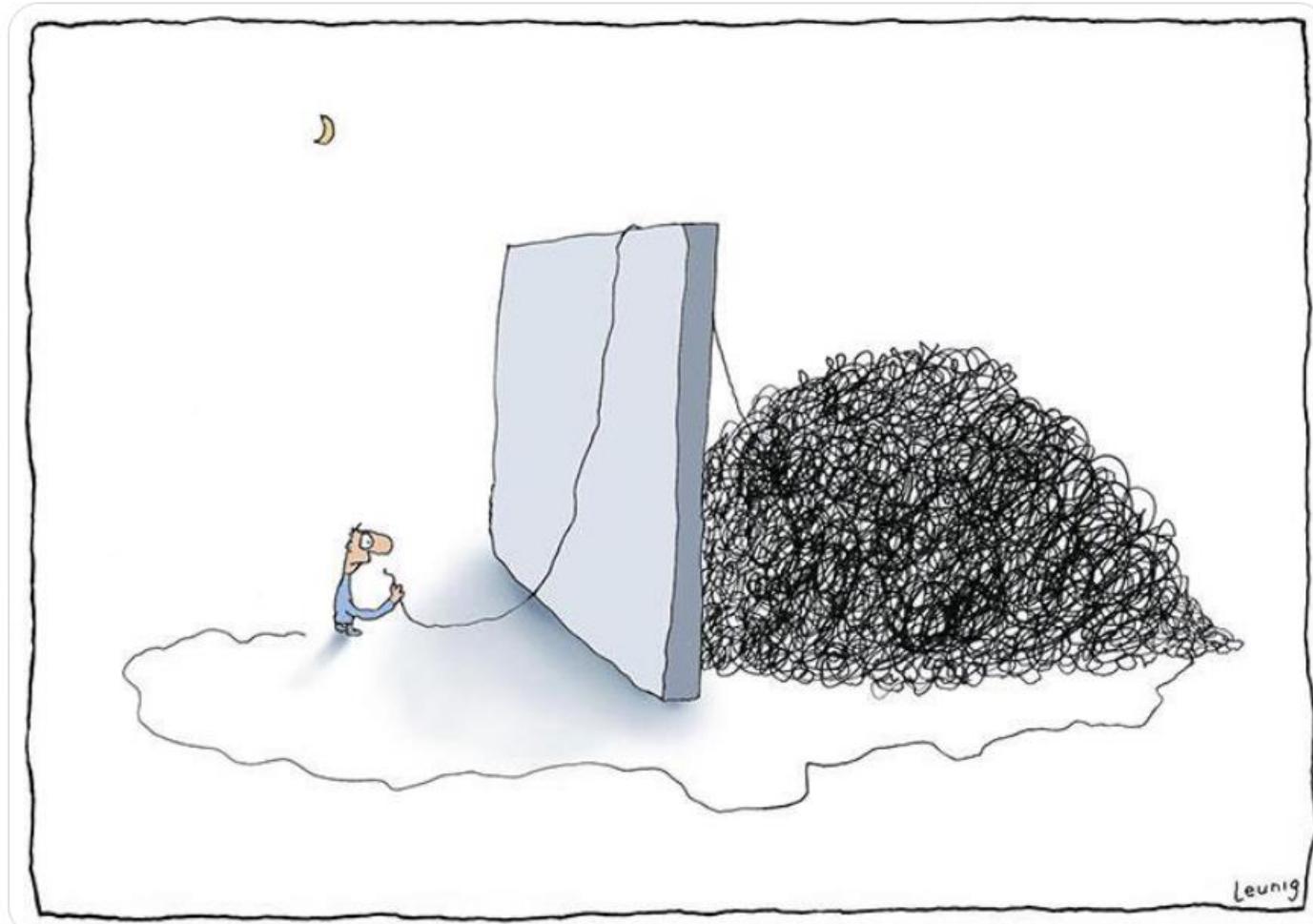
Quickly Testing Legacy Code

kenbot

@KenScambler



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





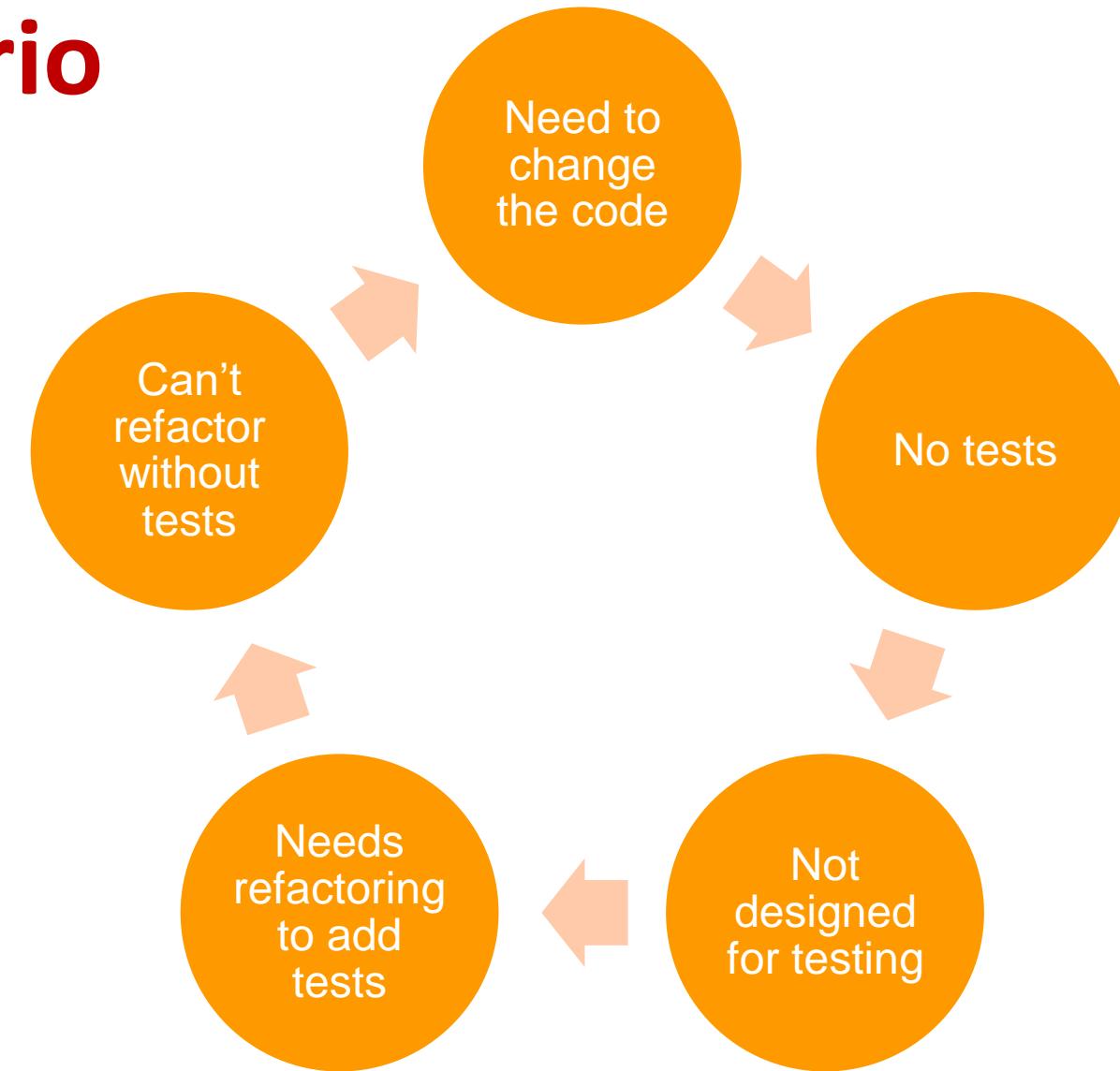
What does Legacy Code really mean?

- 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

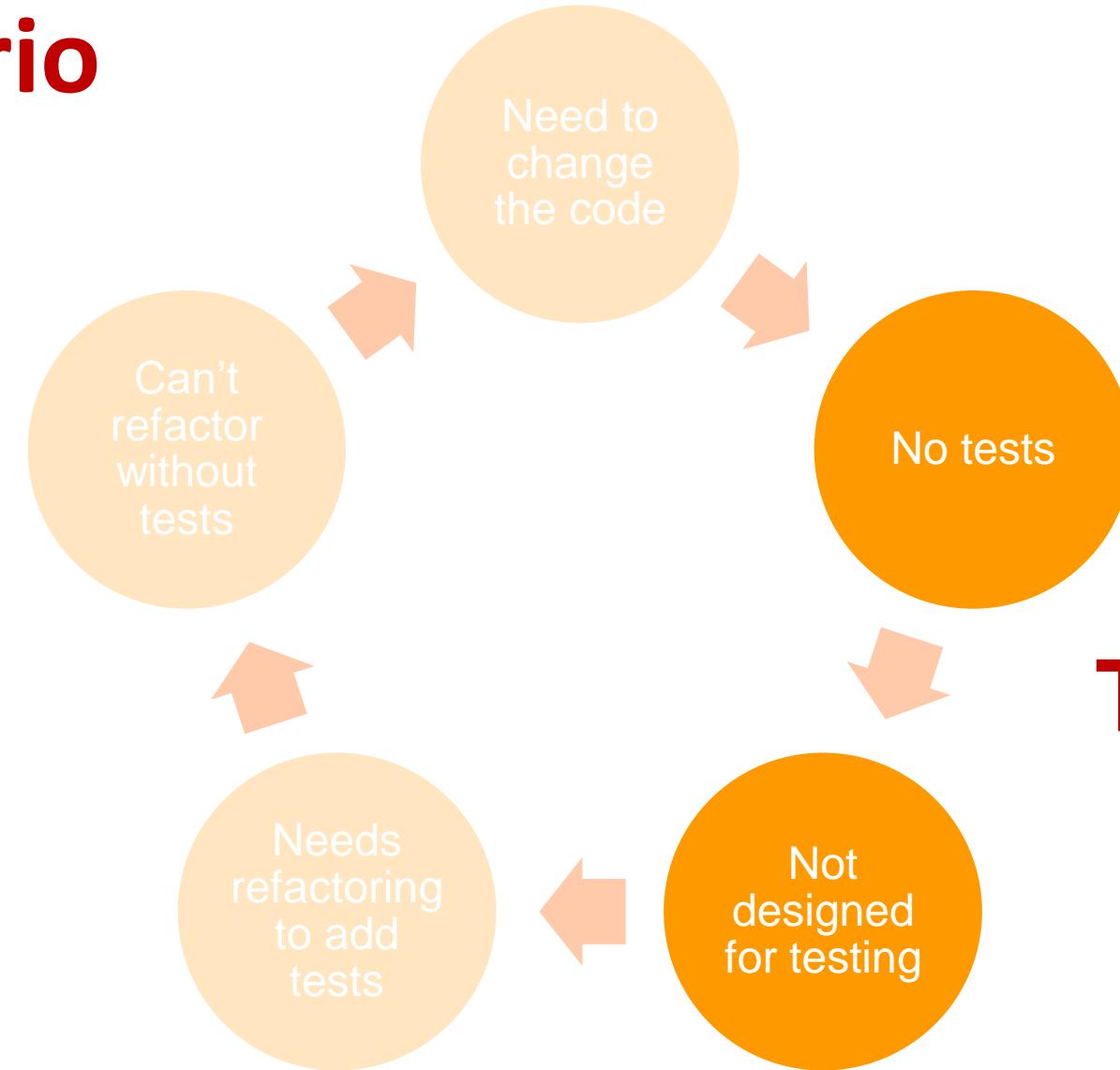
- I've inherited some legacy code
- It's valuable
- I need to add feature
- Or fix bug
- How can I ever break out of this loop?





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 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>





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

File Edit View Project Build Debug Team Tools Test ReSharper Analyze Window Help

Debug Win32 Local Windows Debugger

GildedRoseApprovals.cpp GildedRoseTest.cpp GildedRose.cc

gilded_rose_refactoring_kata GildedRose updateQuality()

```
1 #include "GildedRose.h"
2
3 GildedRose::GildedRose(vector<Item> & items) : items(items)
4 {
5 }
6
7 void GildedRose::updateQuality()
8 {
9     for (int i = 0; i < items.size(); i++)
10    {
11        if (items[i].name != "Aged Brie" && items[i].name != "Backstage passes to a TAFKAL80ETC concert")
12        {
13            if (items[i].quality > 0)
14            {
15                if (items[i].name != "Sulfuras, Hand of Ragnaros")
16                {
17                    items[i].quality = items[i].quality - 1;
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)
```

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

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\Debug\gilded_rose_refactoring_kata.exe	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\gildedroseapprova	100%	46	0	46
d\users\clare\documents\programming\github\cpponsea2019\gilded_rose_refactoring_kata\gildedrosetest.cpp	100%	8	0	8

Coverage Unit Test Explorer Output Error List

Ready Ln 26 Col 38 Ch 38 INS ↑ 7 ⌂ 2 cpronsea2019 master ⌂

100% Test Coverage

Or is it?

OpenCppCoverage does not show branch/condition coverage.



test.csv - 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     {
10        if (
11            items[i].name != "Aged Brie" &&
12            items[i].name != "Backstage passes to a TAFKAL80ETC concert")
13        {
14            if (items[i].quality > 0)
15            {
16                if (items[i].name != "Sulfuras, Hand of Ragnaros")
17                {
18                    items[i].quality = items[i].quality - 1;
19                }
20            }
21            else
22            {
23                if (items[i].quality < 50)
24                {
25                    items[i].quality = items[i].quality + 1;
26                }
27                if (items[i].name == "Backstage passes to a TAFKAL80ETC concert")
28                {
29                    if (items[i].sellIn < 11)
30                    {
31                        if (items[i].quality < 50)
32                        {
33                            items[i].quality = items[i].quality + 2;
34                        }
35                    }
36                }
37            }
38        }
39    }
40}
```

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



If tests need improving...
And unit tests not viable...



Contents

- Introduction
- Legacy Code
- **Golden Master** ←
- Approval Tests
- Example
- Resources
- Summary



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

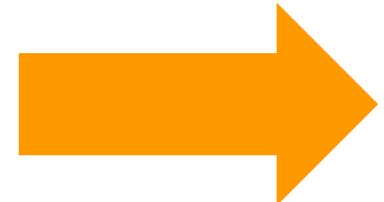


Golden Master Test Setup

**Input
Data**



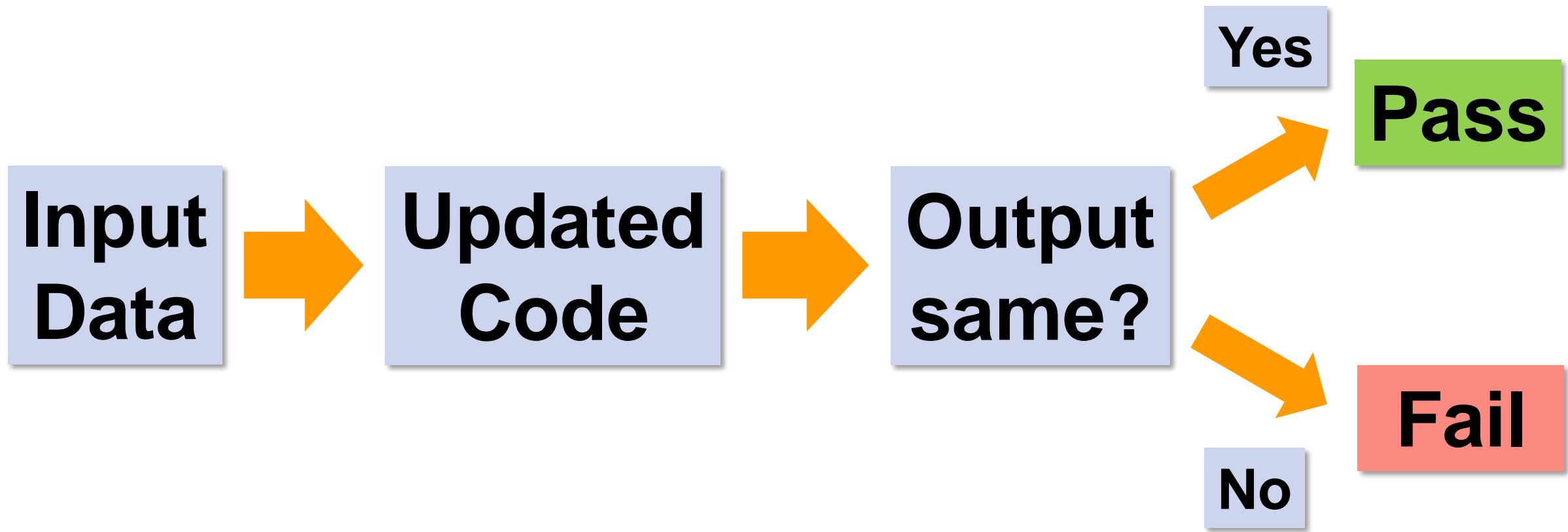
**Existing
Code**



**Save
“Golden
Master”**



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 Master by mistake!
- Doesn't matter that it's not a Unit Test



Contents

- Introduction
- Legacy Code
- Golden Master
- **Approval Tests** ←
- Example
- Resources
- Summary



Quickly Testing Legacy Code



One approach: “ApprovalTests”

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

GO

Java

Lua

.NET

.Net.ASP

NodeJS

Objective-C

PHP

Perl

Python

Swift

TCL

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



StarterProject with Catch2

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

The screenshot shows a GitHub repository page for 'approvals / ApprovalTests.cpp.StarterProject'. The repository has 34 commits, 2 branches, and 0 releases. The file tree includes '.idea', 'code', 'lib', 'tests', and 'visual-studio-2017'. A 'Clone or download' button is highlighted in green, with options for 'Clone with SSH' and 'Use HTTPS'. There are also links to 'Open in Desktop' and 'Open in Visual Studio', and a prominent 'Download ZIP' button.



How to use it: Catch2 Boilerplate

- Your 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 );
}
```



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");
}
```

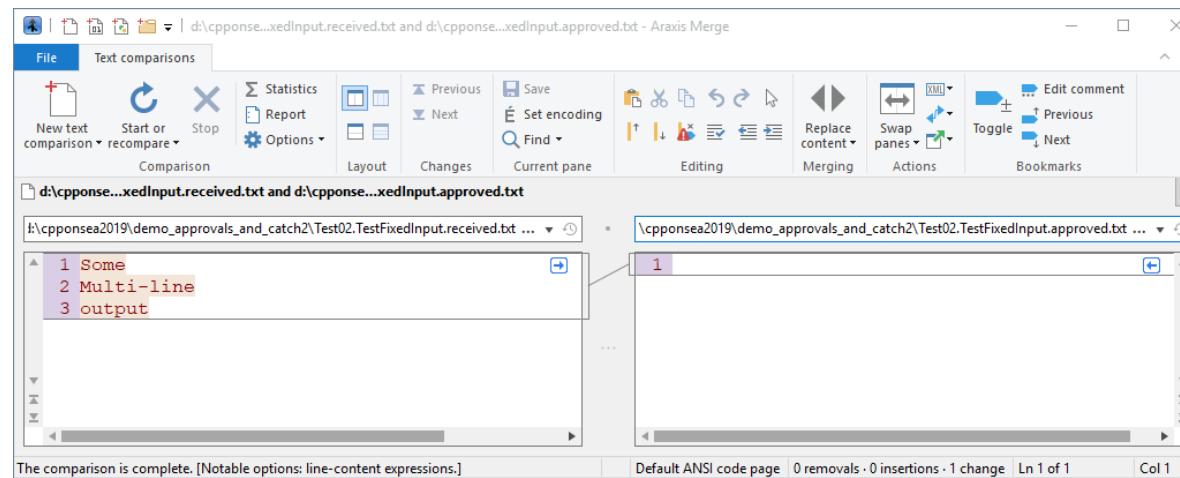


First run

- 1. **TestFixedInput failed**

```
d:\cpponsea2019\demo_approvals_andCatch2\test02.cpp(5): exception: Failed Approval:  
Approval File Not Found  
File: "d:\cpponsea2019\demo_approvals_andCatch2\Test02.TestFixedInput.approved.txt"
```

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





Actual/Received

Expected/Approved

The comparison is complete. [Notable options: line-content expressions.]

Default ANSI code page 0 removals · 0 insertions · 1 change Ln 1 of 1 Col 1



Actual/Received

Expected/Approved

d:\cpponse...xedInput.received.txt and d:\cpponse...xedInput.approved.txt

I:\cpponsea2019\demo_approvals_and_catch2\Test02.TestFixedInput.received.txt ...

\cpponsea2019\demo_approvals_and_catch2\Test02.TestFixedInput.approved.t ...

1 Some
2 Multi-line
3 output

1 Some
2 Multi-line
3 output

The comparison is complete. [Notable options: line-content expressions.]

Default ANSI code page

The files are identical | L 3 of 3

Col 7



Actual/Received

The screenshot shows the Araxis Merge application interface comparing two files: `d:\cpponse...xedInput.received.txt` and `d:\cpponse...xedInput.approved.txt`. The 'Actions' menu is open, and a green arrow points to the 'Edit comment' option under the 'Bookmarks' section.

Comparison Details:

- Left pane (Actual/Received):
 - File path: `I:\cpponsea2019\demo_approvals_and_catch2\Test02.TestFixedInput.received.txt`
 - Content:
 - 1 Some
 - 2 Multi-line
 - 3 output
- Right pane (Expected/Approved):
 - File path: `\cpponsea2019\demo_approvals_and_catch2\Test02.TestFixedInput.approved.txt`
 - Content:
 - 1 Some
 - 2 Multi-line
 - 3 output

Status Bar:

The status bar at the bottom indicates: "The comparison is complete. [Notable options: line-content expressions.] Default ANSI code page The files are identical Ln 3 of 3 Col 7".

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	02/02/2019 17:42	CPP File	4 KB
	Test02.TestFixedInput.approved.txt	02/02/2019 17:42	Text Document	1 KB
	Test03CustomReporters.cpp	02/02/2019 17:42	CPP File	1 KB
	Test03CustomReporters.UseCustomReporter.approved.txt	02/02/2019 17:42	Text Document	1 KB
	Test03CustomReporters.UseQuietReporter.approved.txt	02/02/2019 17:42	Text Document	1 KB
	Test03CustomReporters.UseSpecificReporter.approved.txt	02/02/2019 17:42	Text Document	1 KB
	Test04ConsoleReporter.cpp	02/02/2019 17:42	CPP File	1 KB
	Test04ConsoleReporter.UseConsoleReporter.approved.txt	02/02/2019 17:42	Text Document	1 KB
	Test05VerifyAll.cpp	02/02/2019 17:42	CPP File	3 KB
	Test05VerifyAll.verifyAllWithHeaderAndVector.approved.txt	02/02/2019 17:42	Text Document	1 KB
	Test05VerifyAll.verifyAllWithHeaderBeginEndAndLambda.approved.txt	02/02/2019 17:42	Text Document	1 KB
	Test05VerifyAll.verifyAllWithHeaderContainerAndLambda.approved.t	02/02/2019 17:42	Text Document	1 KB
	Test05VerifyAll.verifyAllWithVector.approved.txt	02/02/2019 17:42	Text Document	1 KB



How to use it: GoogleTest Boilerplate

- Your main.cpp

```
#define APPROVALS_GOOLETST  
#include "ApprovalTests.hpp"
```



Pure Google Test

- A test file

```
#include <gtest/gtest.h>

// Google-only test
TEST( Test01, SumsAreCalculated )
{
    EXPECT_EQ( 1 + 1, 2 );
    EXPECT_EQ( 1 + 2, 3 );
}
```



Approvals Google Test

- A test file

```
#include "ApprovalTests.hpp"
#include <gtest/gtest.h>

// googletest - test static value, for demo purposes
TEST( Test02, TestFixedInput )
{
    Approvals::verify("Some\nMulti-line\noutput");
}
```



Reference: Add to existing GoogleTest

- Your main.cpp

```
#define APPROVALS_GOOGLETST_EXISTING_MAIN
#include "ApprovalTests.hpp"

int main(int argc, char** argv)
{
    ::testing::InitGoogleTest(&argc, argv);
    initializeApprovalTestsForGoogleTests();
    return RUN_ALL_TESTS();
}
```



Reference: Supporting a new test framework

- Any test framework that supplies the following:
 - Current test's name
 - Current test's source file (with correct case of filename)
 - Report unexpected exceptions as a test failure
- That's all that's needed



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 OSs

- Line-endings



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>(
            // 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



Challenge: Golden Master is a log file

- Dates and times?
 - Object addresses?

2019-01-29 15:27



Options for unstable output

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



Tip: Rewrite output file

File Path	File Name	File Type	File Size
Root Path	Index.html	HTML	123456789
Second Path	Index.html	HTML	123456789
Third Path	Index.html	HTML	123456789
Fourth Path	Index.html	HTML	123456789
Fifth Path	Index.html	HTML	123456789
Sixth Path	Index.html	HTML	123456789
Seventh Path	Index.html	HTML	123456789
Eighth Path	Index.html	HTML	123456789
Ninth Path	Index.html	HTML	123456789
Tenth Path	Index.html	HTML	123456789
Eleventh Path	Index.html	HTML	123456789
Twelfth Path	Index.html	HTML	123456789
Thirteenth Path	Index.html	HTML	123456789
Fourteenth Path	Index.html	HTML	123456789
Fifteenth Path	Index.html	HTML	123456789
Sixteenth Path	Index.html	HTML	123456789
Seventeenth Path	Index.html	HTML	123456789
Eighteenth Path	Index.html	HTML	123456789
Nineteenth Path	Index.html	HTML	123456789
Twentieth Path	Index.html	HTML	123456789

2019-01-29 15:27



File Path	File Name	File Type	File Size
Root Path	[date-time-stamp]	HTML	123456789
Second Path	[date-time-stamp]	HTML	123456789
Third Path	[date-time-stamp]	HTML	123456789
Fourth Path	[date-time-stamp]	HTML	123456789
Fifth Path	[date-time-stamp]	HTML	123456789
Sixth Path	[date-time-stamp]	HTML	123456789
Seventh Path	[date-time-stamp]	HTML	123456789
Eighth Path	[date-time-stamp]	HTML	123456789
Ninth Path	[date-time-stamp]	HTML	123456789
Tenth Path	[date-time-stamp]	HTML	123456789
Eleventh Path	[date-time-stamp]	HTML	123456789
Twelfth Path	[date-time-stamp]	HTML	123456789
Thirteenth Path	[date-time-stamp]	HTML	123456789
Fourteenth Path	[date-time-stamp]	HTML	123456789
Fifteenth Path	[date-time-stamp]	HTML	123456789
Sixteenth Path	[date-time-stamp]	HTML	123456789
Seventeenth Path	[date-time-stamp]	HTML	123456789
Eighteenth Path	[date-time-stamp]	HTML	123456789
Nineteenth Path	[date-time-stamp]	HTML	123456789
Twentieth Path	[date-time-stamp]	HTML	123456789

[date-time-stamp]



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;
};
```



Feature: Run on build servers

```
TEST_CASE("UseQuietReporter")
{
    // QuietReporter does nothing if a failure occurs.
    // Failing tests will still fail, but nothing is launched.
    Approvals::verify(
        "Some\nMulti-line\noutput",
        QuietReporter()));
}
```



Customisability: Reporters

- 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



Tip: currentReporter()

- Easy to switch reporter behaviour



Feature: Convention over Configuration

- Fewer decisions for developers
- Users only specify unusual behaviours

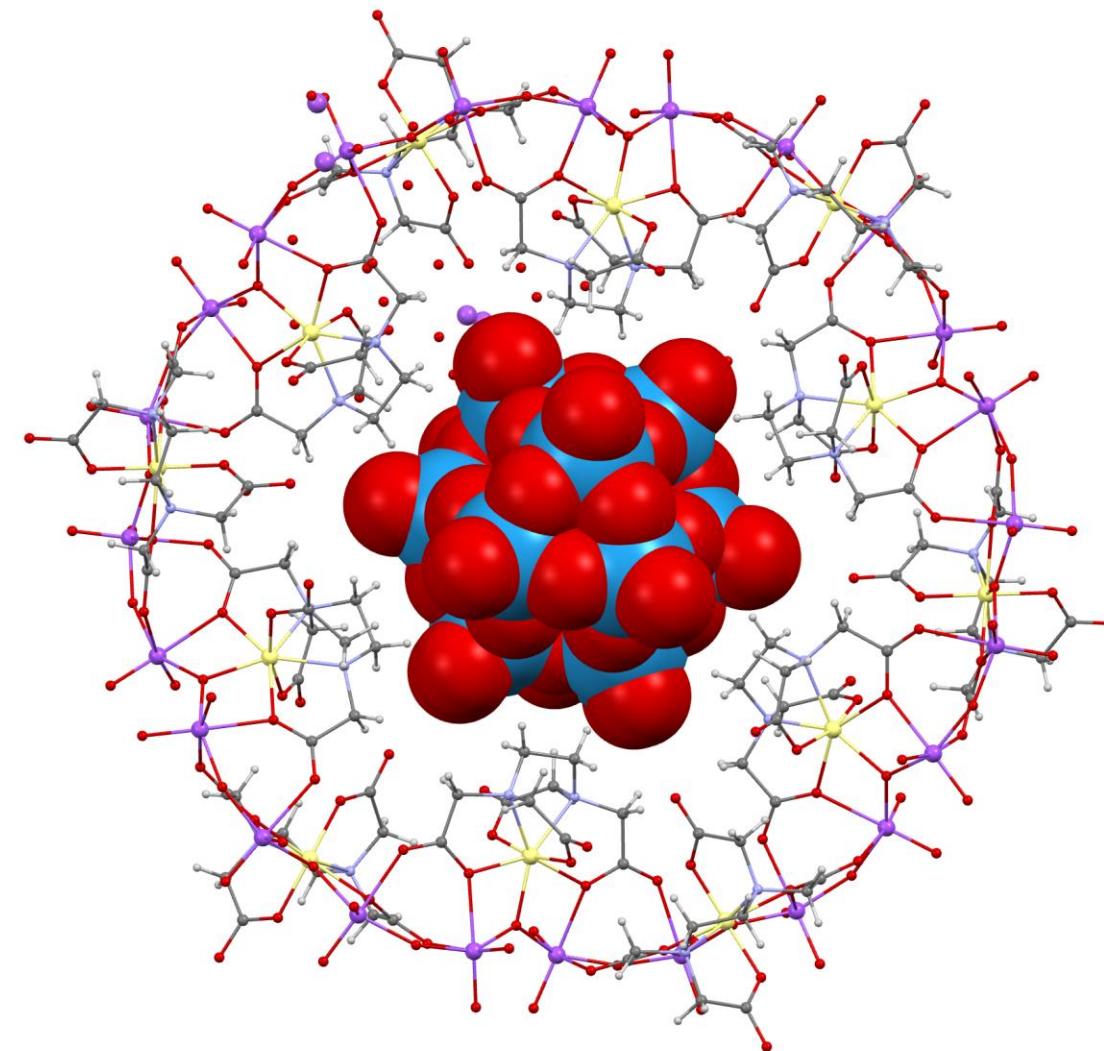


Contents

- Introduction
- Legacy Code
- Golden Master
- Approval Tests
- **Example** ←
- Resources
- Summary

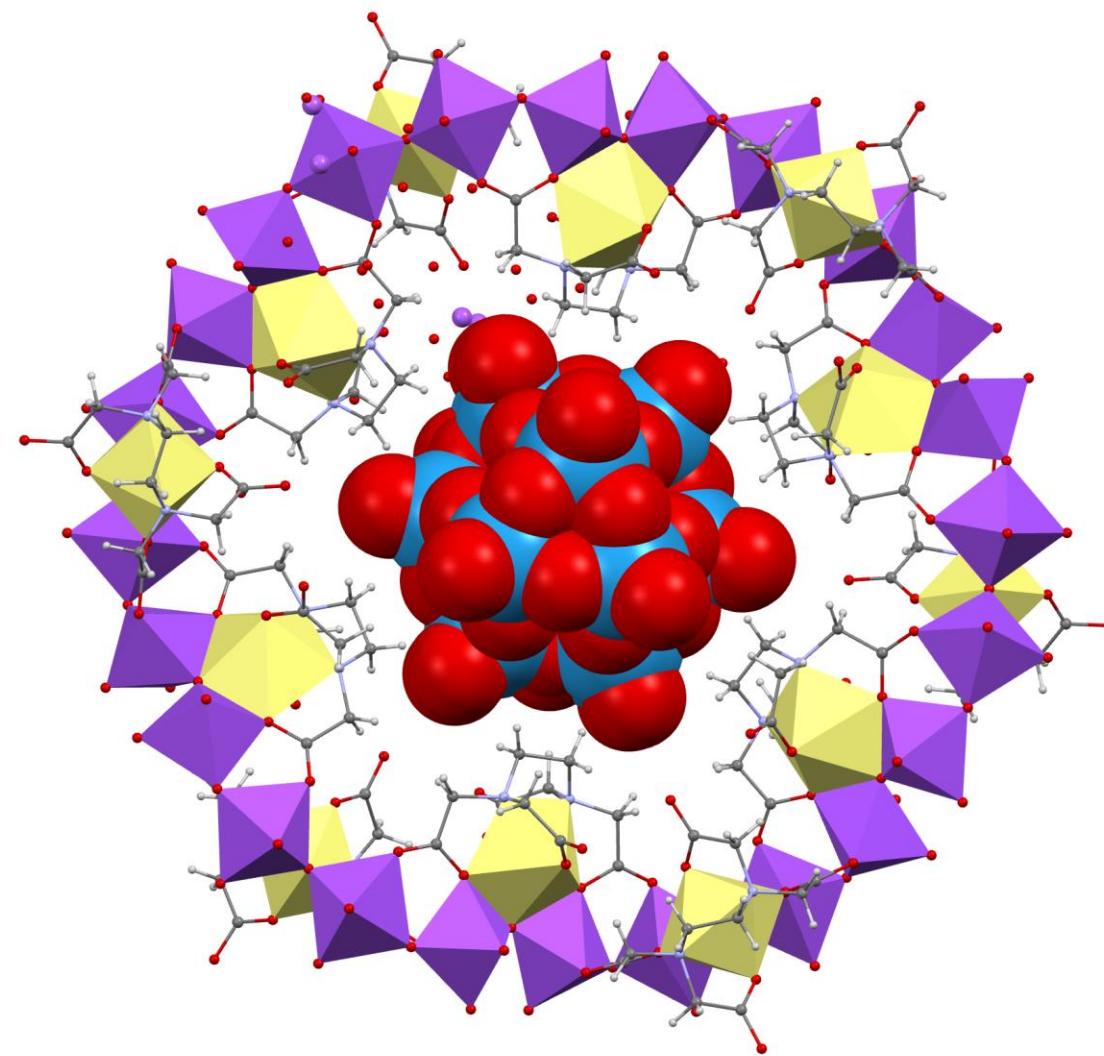


What I can do:





What I need to do:



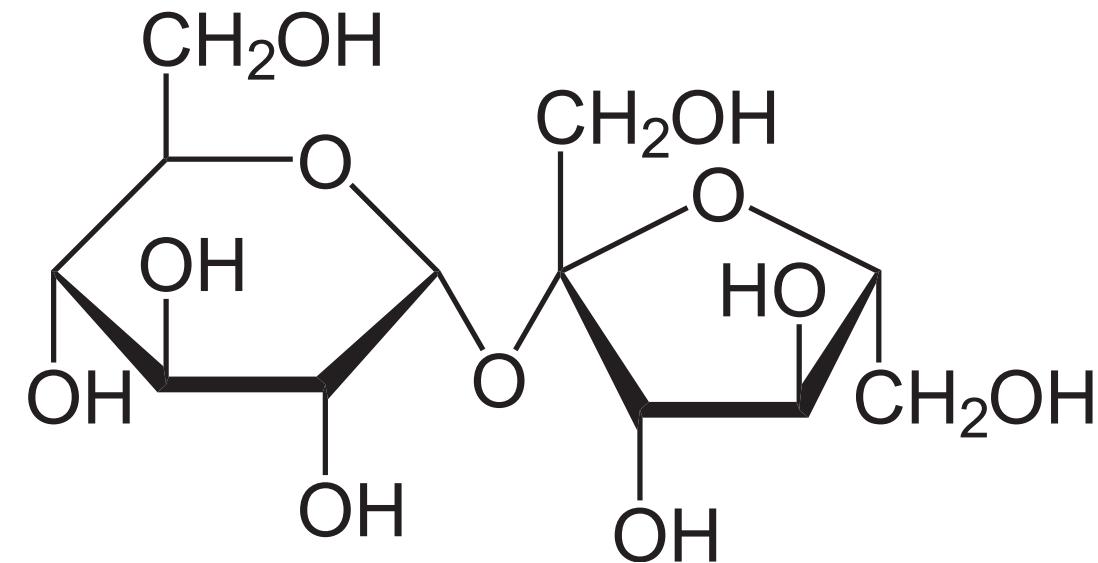


Sugar - sucrose



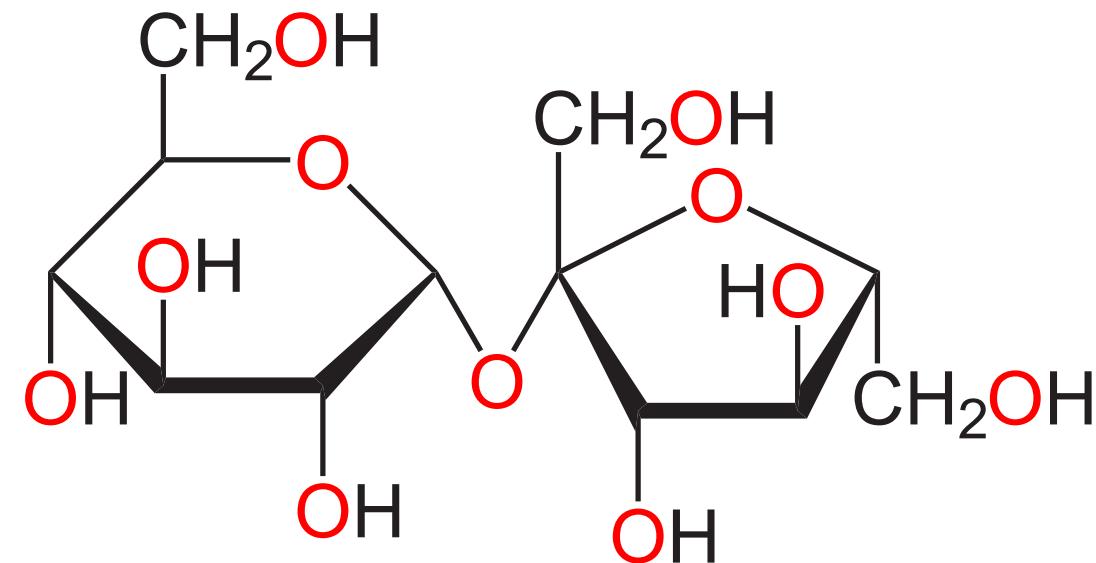


Sugar - sucrose



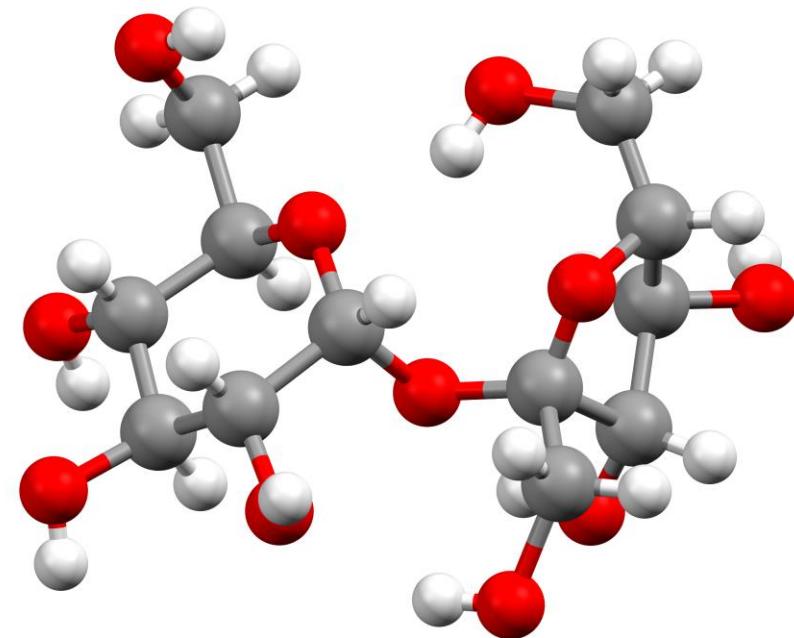


Sugar - sucrose

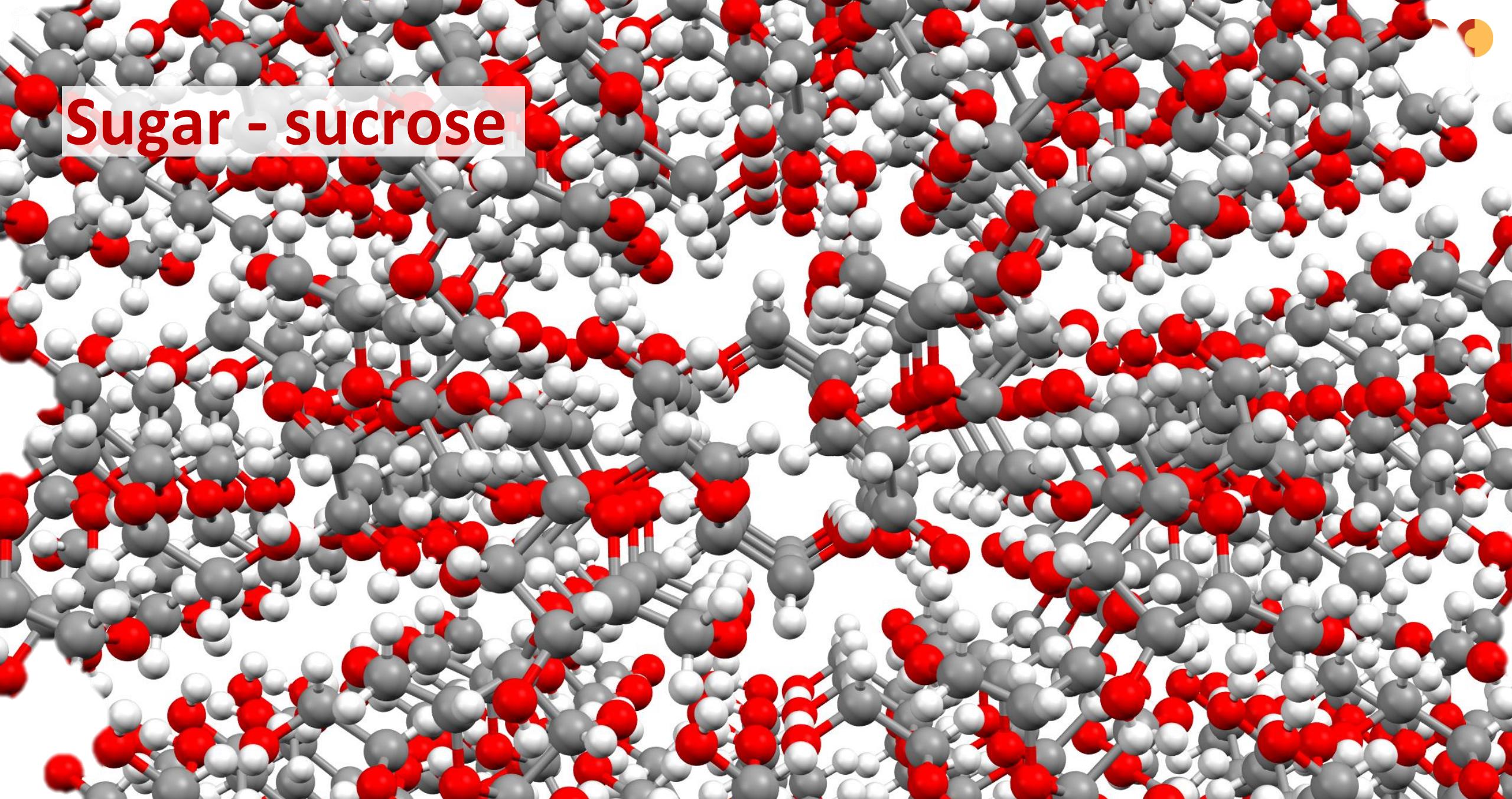




Sugar - sucrose



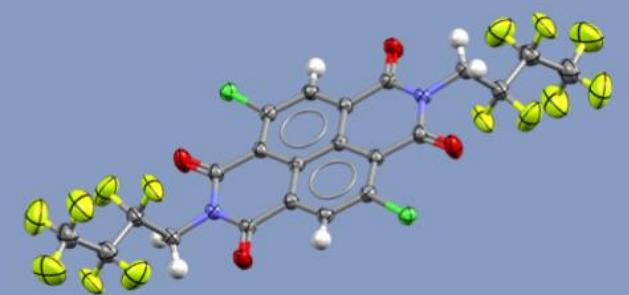
Sugar - sucrose





The Cambridge Structural Database (CSD)

The world's comprehensive and up-to-date database of crystal structures with over 950,000 curated entries





Approving Images



What I'm aiming for

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



Foundation of Approval tests

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



Idiomatic verification of Qt image objects

```
void verifyQImage(  
    QImage image, const Reporter& reporter = DiffReporter())  
{  
    ApprovalTestNamer namer;  
    QImageWriter writer(image);  
    FileApprover::verify(namer, writer, reporter);  
}
```



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;
};
```



QImageWriter declaration

```
class QImageWriter : public ApprovalWriter
{
public:
    explicit QImageWriter(QImage image, std::string fileExtensionWithDot = ".png");
    std::string getFileExtensionWithDot() override;
    void write(std::string path) override;
    void cleanUpReceived(std::string receivedPath) override;

private:
    QImage image_;
    std::string ext;
};
```



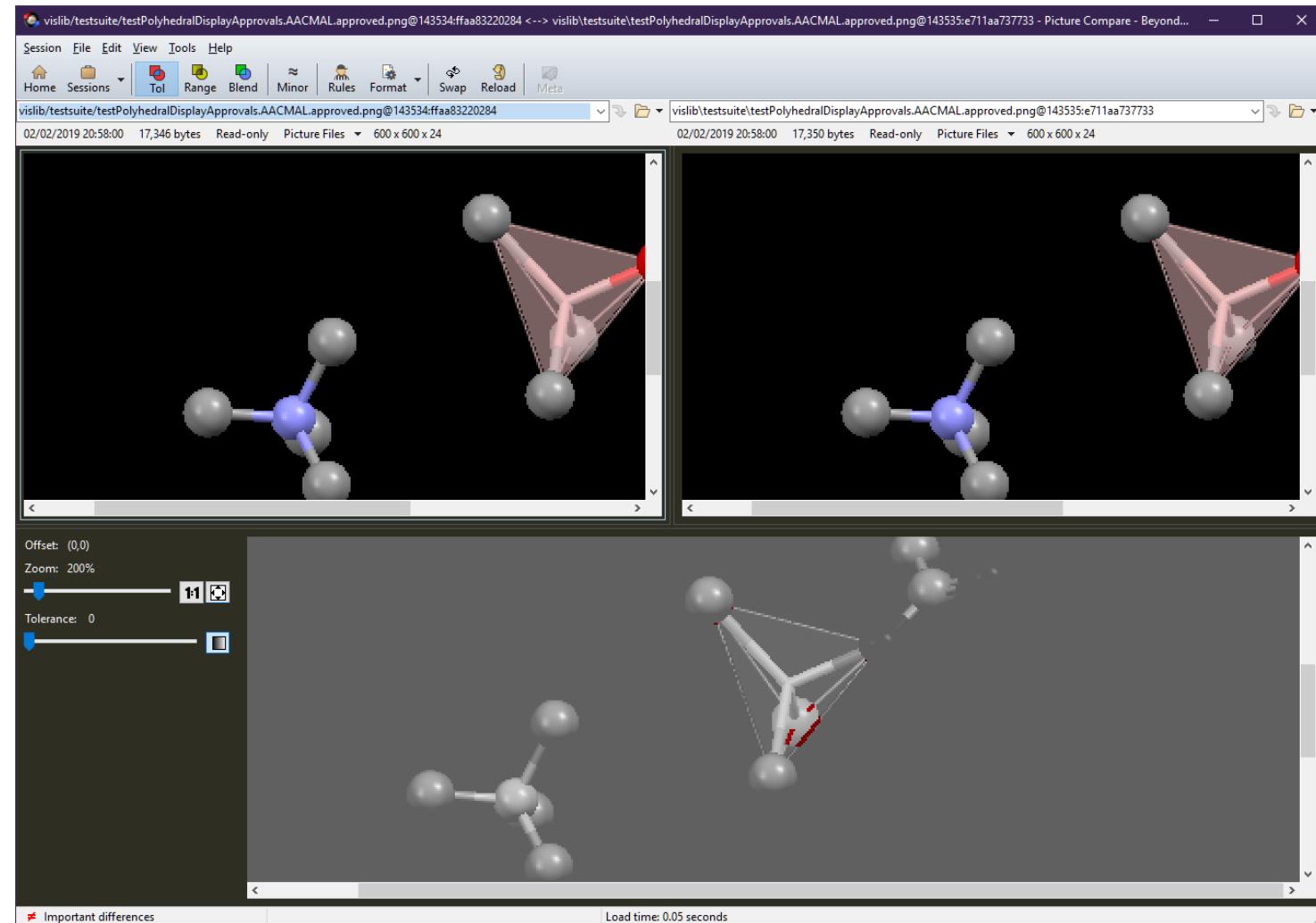
QImageWriter::write()

```
void QImageWriter::write(std::string path)
{
    // Have to convert std::string to QString
    image_.save( QString::fromStdString(path) );
}
```



Useful feedback

BeyondCompare 4





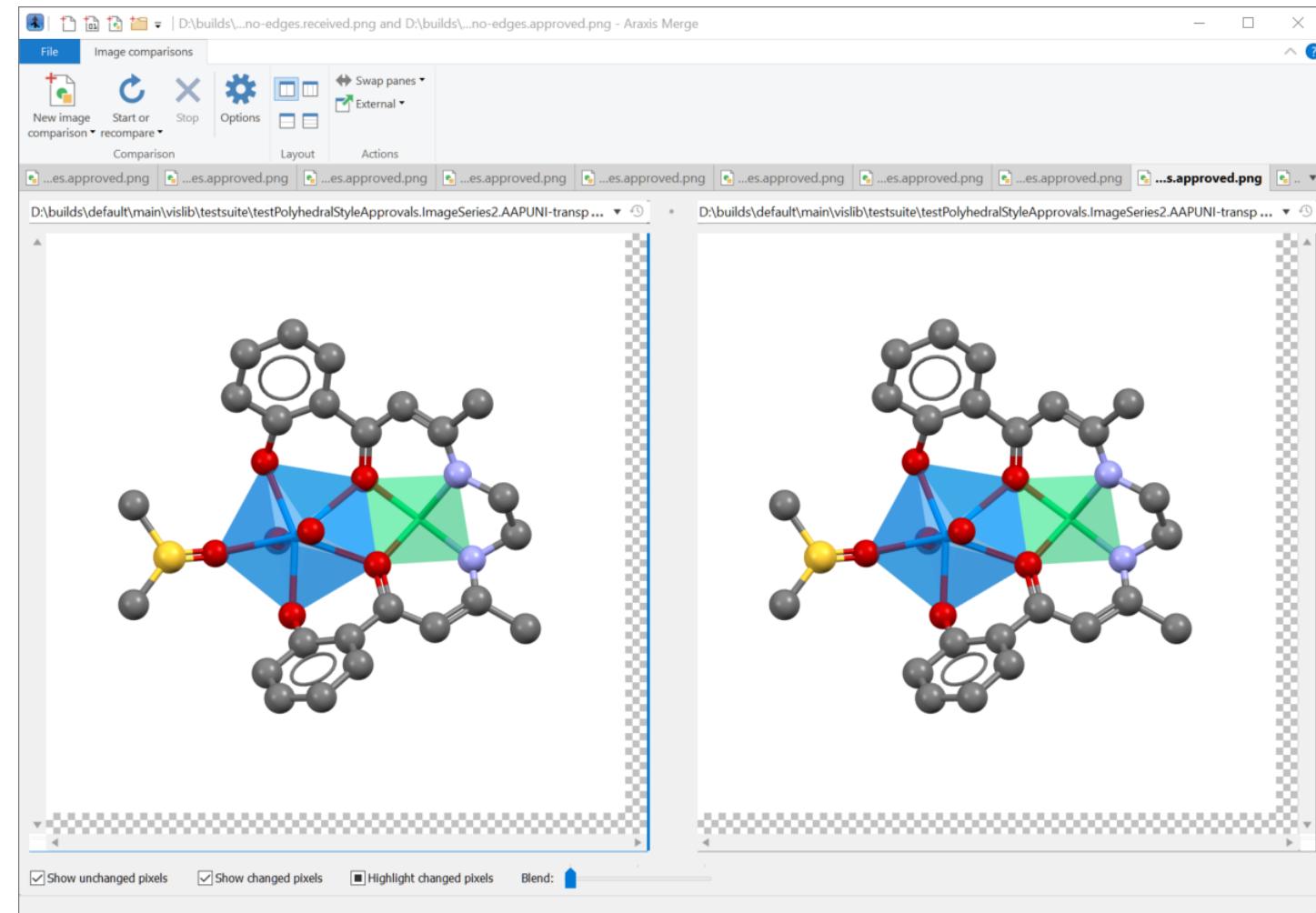
Back In work...

- Previously, working from home via Remote Desktop
- The tests started failing when I ran them in work



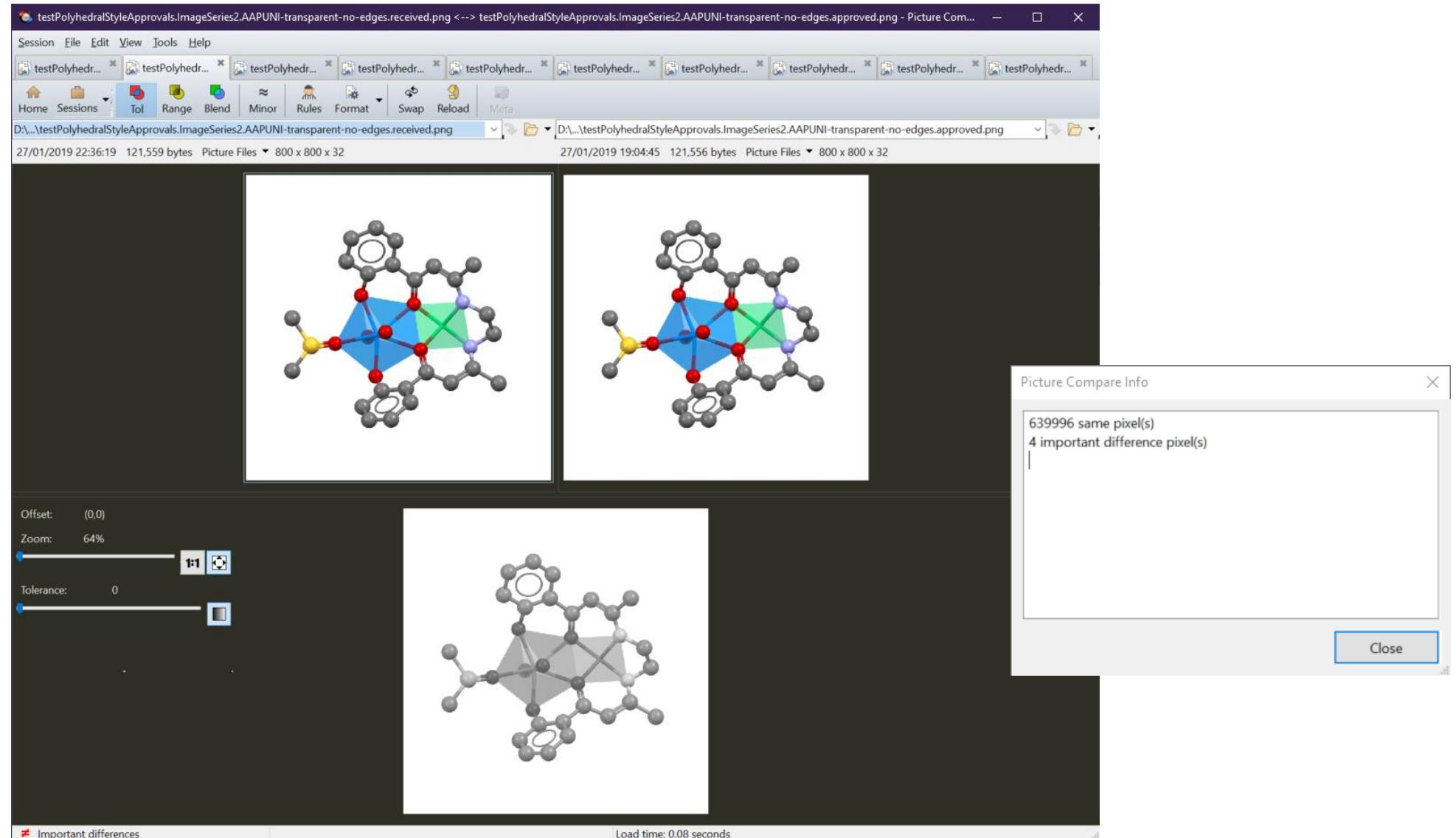
What? Re-running now gives random failures

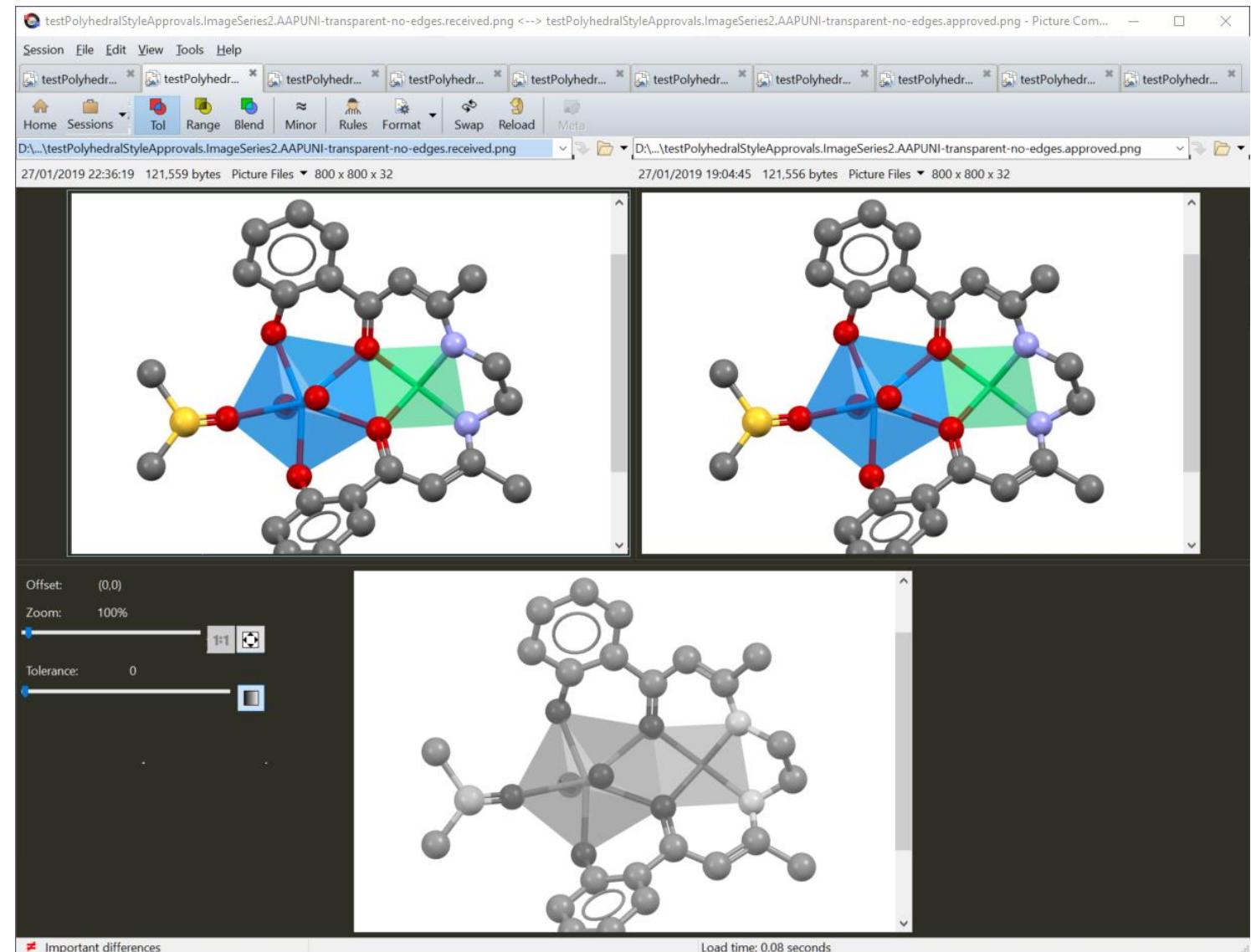
Araxis Merge

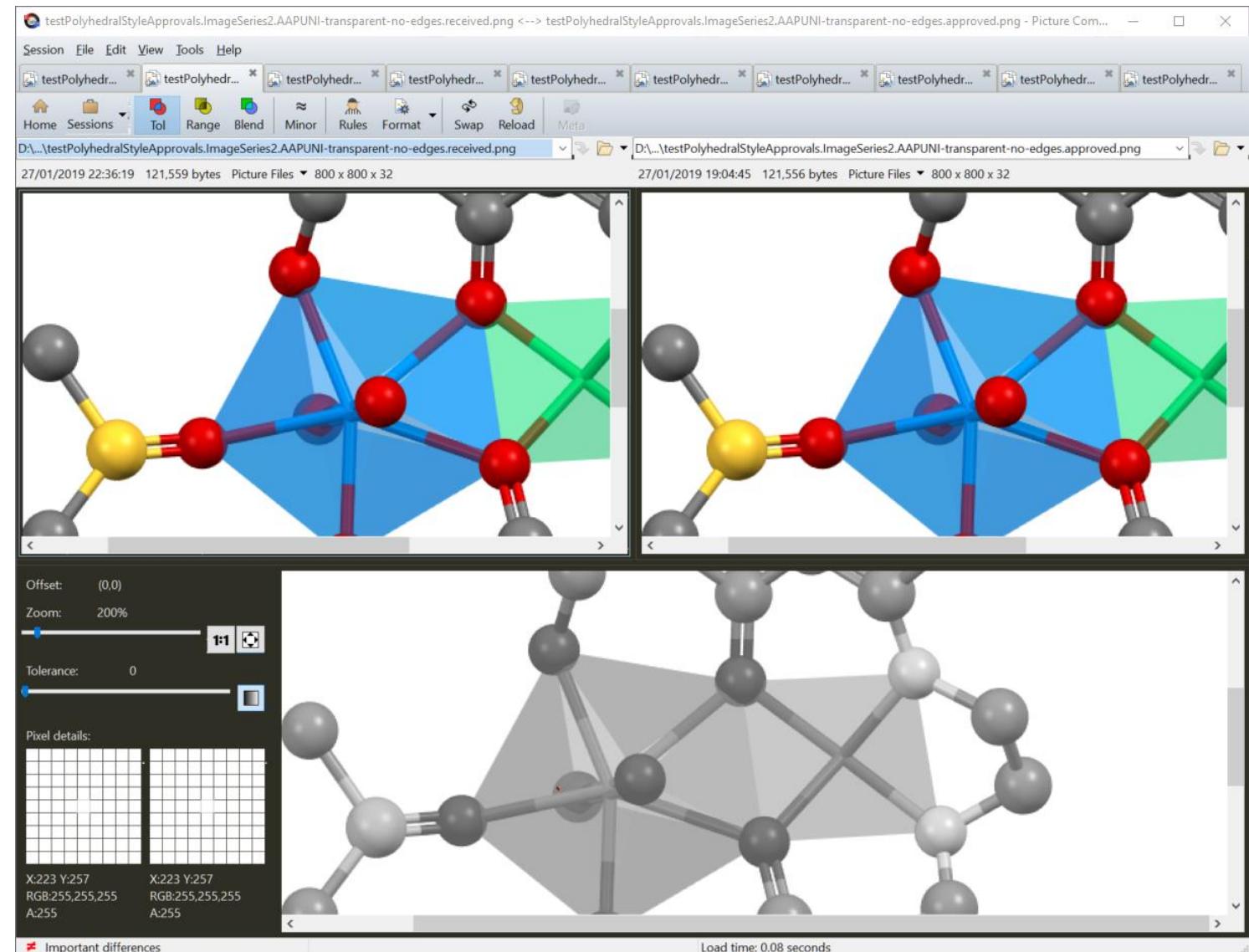


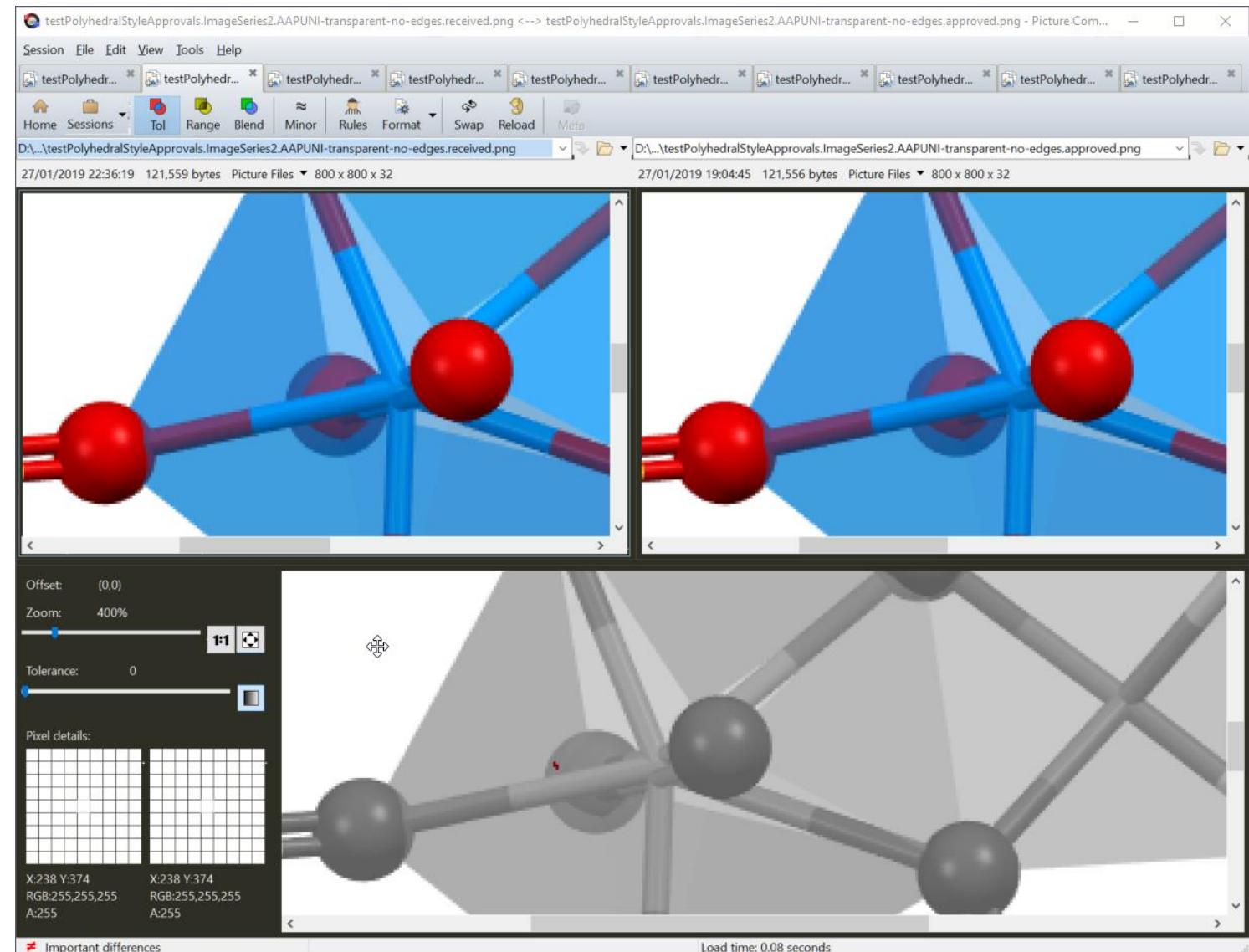


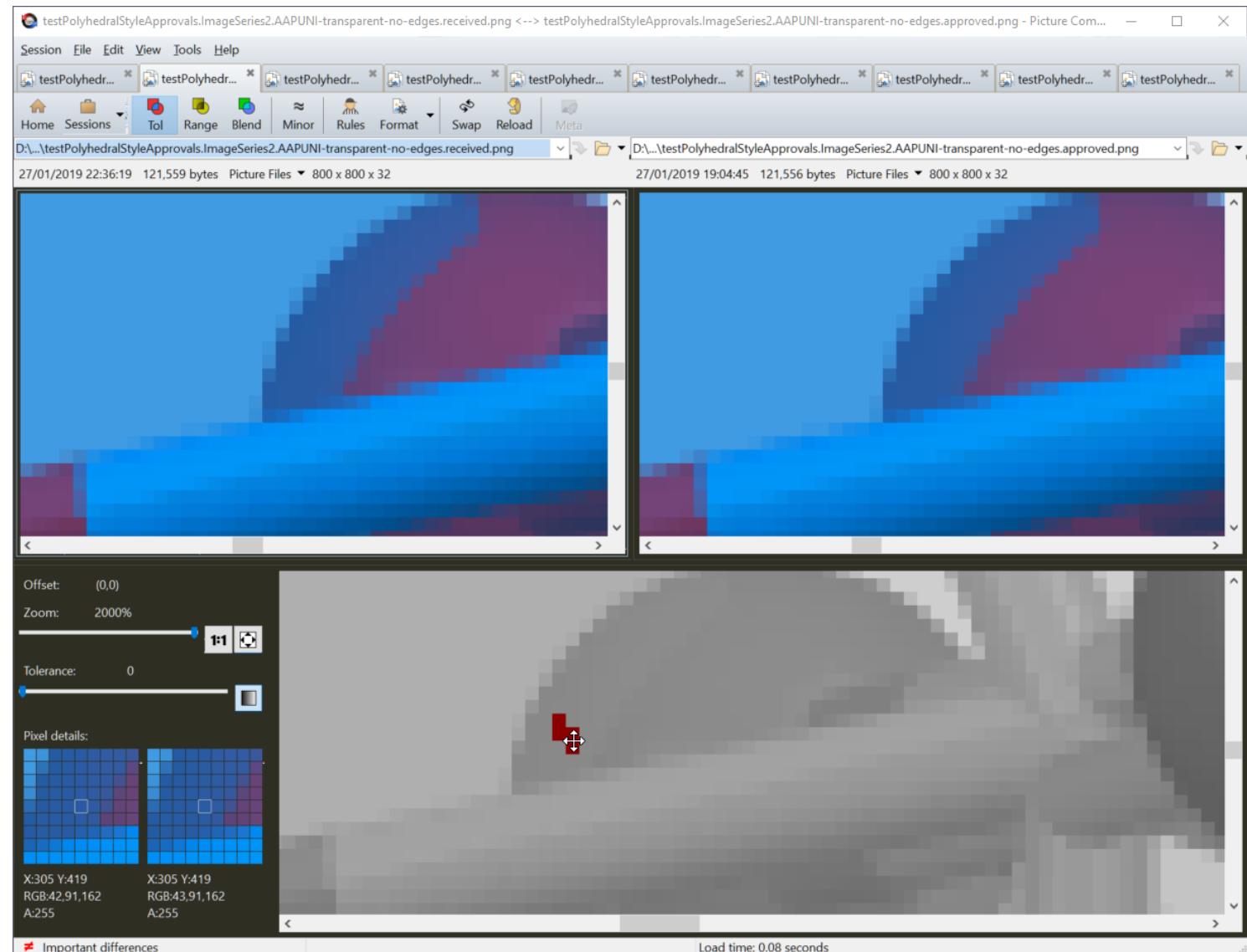
BeyondCompare 4

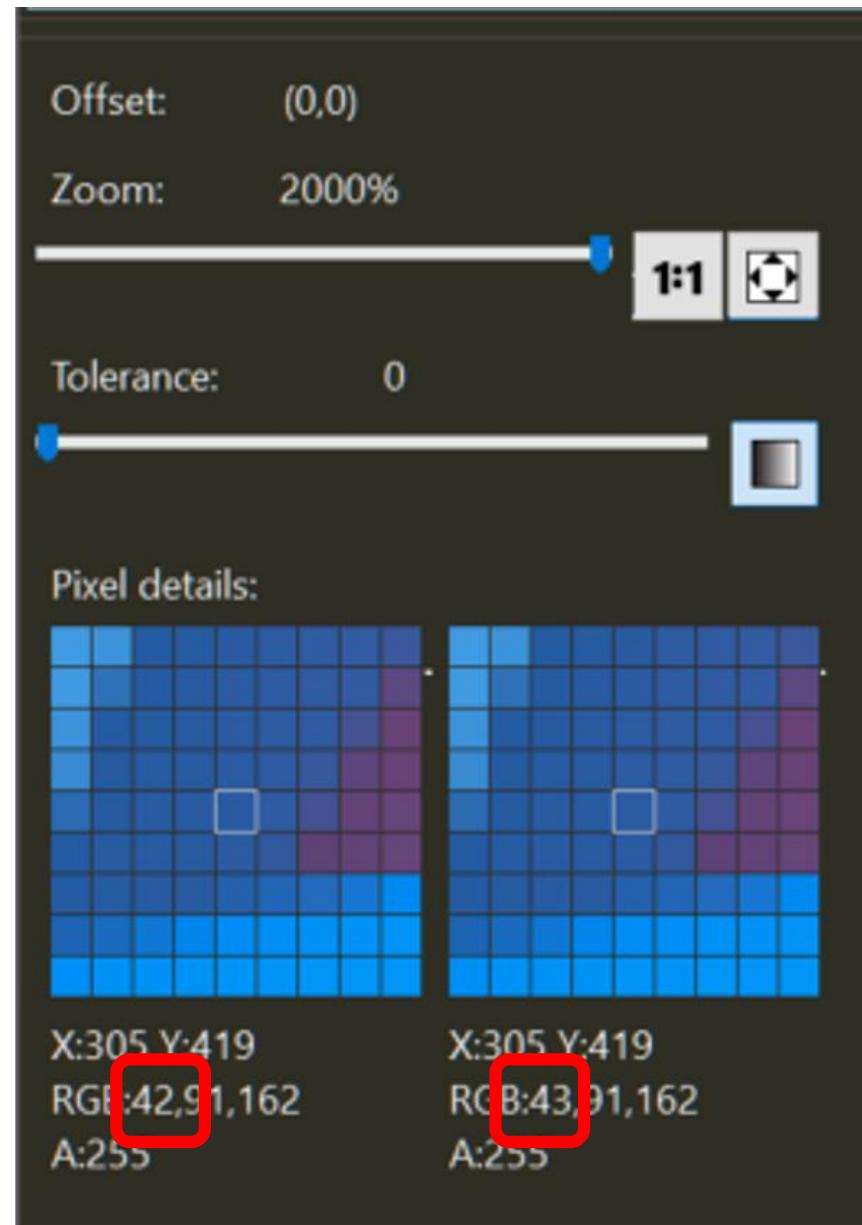














Customisability: ApprovalComparator

```
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



Contents

- Introduction
- Legacy Code
- Golden Master
- Approval Tests
- Example
- **Resources** ←
- Summary



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>

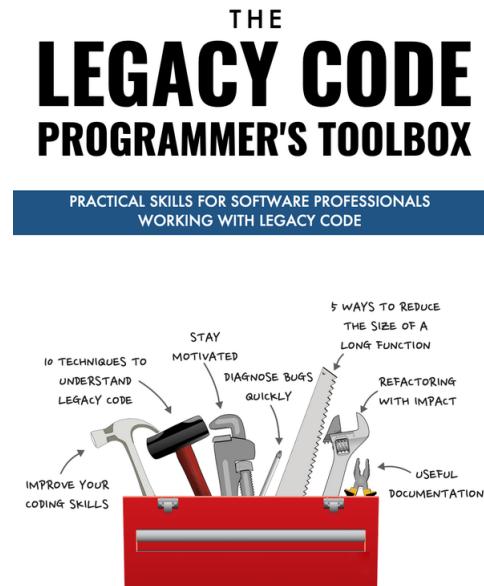
The screenshot shows the course landing page for 'Beautiful C++: Updating Legacy Code'. The title is prominently displayed at the top. Below it, the author's name, 'by Kate Gregory', is visible. A descriptive blurb explains the course's purpose: maintaining old codebases and updating them to modern standards. To the right of the blurb, a code editor window displays a snippet of C++ code, likely demonstrating a refactoring or modernization technique.

If you maintain an old large codebase, there's a good chance you don't understand parts of it. This course will show you how to update code to a more readable, understandable, and maintainable state by taking full advantage of modern C++ techniques.



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



JONATHAN BOCCARA
FOREWORD BY KEVLIN HENNEY



Videos

The thumbnail features a blue header with the title 'Practical Refactoring' and subtitle 'with @WoodyZuill & @Llewellyn Falco'. Below the header is a large yellow circle. In the bottom left corner, there's a small video preview showing two men in an office setting, one standing and one sitting at a desk with a laptop. The video has a duration of 0:00 and 58,016 views. The main video title '2 Minutes to Better Code' is displayed in purple text on the right side of the thumbnail.

Practical Refactoring
with
@WoodyZuill &
@Llewellyn Falco

SPUN LABORATORIES INCORPORATED

2 Minutes
to
Better Code

By Llewellyn Falco & Woody Zuill

Practical Refactoring - How to clean code in many small steps

58,016 views

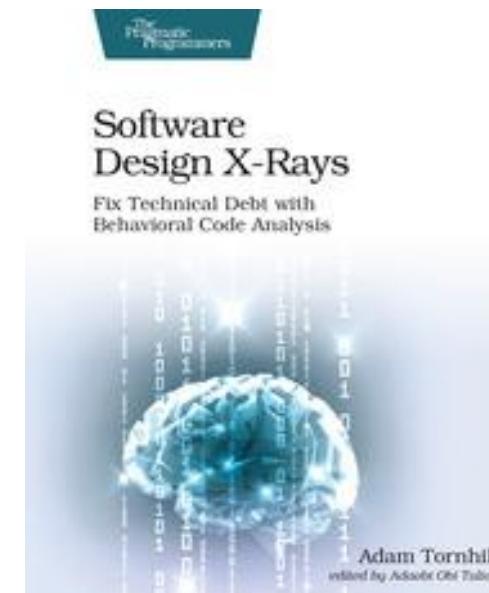
486 likes 23 dislikes SHARE SAVE ...

https://youtu.be/aWiwDdx_rdo



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

- Introduction
- Legacy Code
- Golden Master
- Approval Tests
- Example
- Resources
- **Summary** 



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



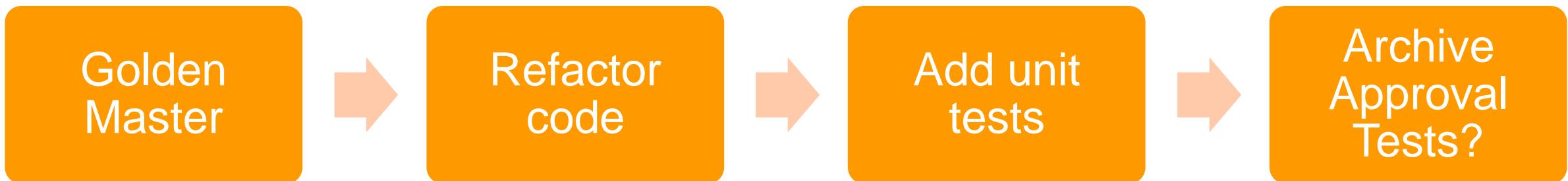
ApprovalTests

- 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/cpponsea2019>
- Slides: <https://www.slideshare.net/ClareMacrae>
 - (later this week)
- ApprovalTests.cpp
 - <https://github.com/approvals/ApprovalTests.cpp>
 - <https://github.com/approvals/ApprovalTests.cpp.StarterProject>