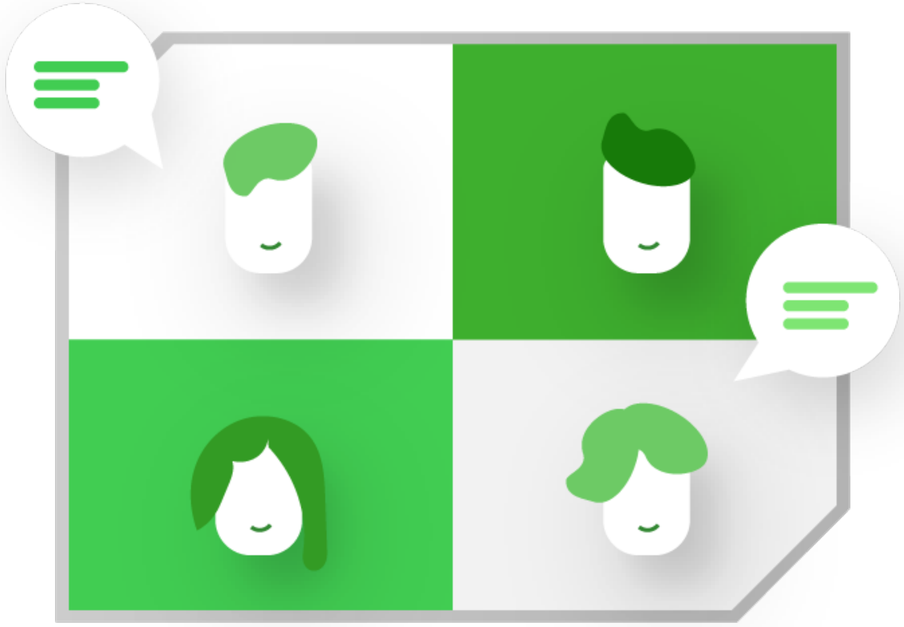
Qt DESKTOP|DAYS
September 7th - 11th



The "Test Smarter" Approach to Improving Product Quality Using Automated Code Coverage Analysis

froglogic

Nick Medeiros, nicholas@froglogic.com

September 10th, 2020

$(whoami)

Nick Medeiros

froglogic

- Marketing @ froglogic since 2 years
- B.S./M.S. in Chem Eng/Computational Mech
- Boston, MA, USA
- Francophile, poetry reader, Subaru driver, (very) amateur runner

What is Coco?

froglogic

Cross-platform **Co**de **Co**verage analysis toolchain

- Identifies: **untested code — redundant tests — dead code**

- 3-in-1 tool**: coverage analysis** + **code complexity** + **function profiling**

  + special features to optimize *your entire team's* testing workflow

Coverage Levels

An introduction to code coverage

froglogic

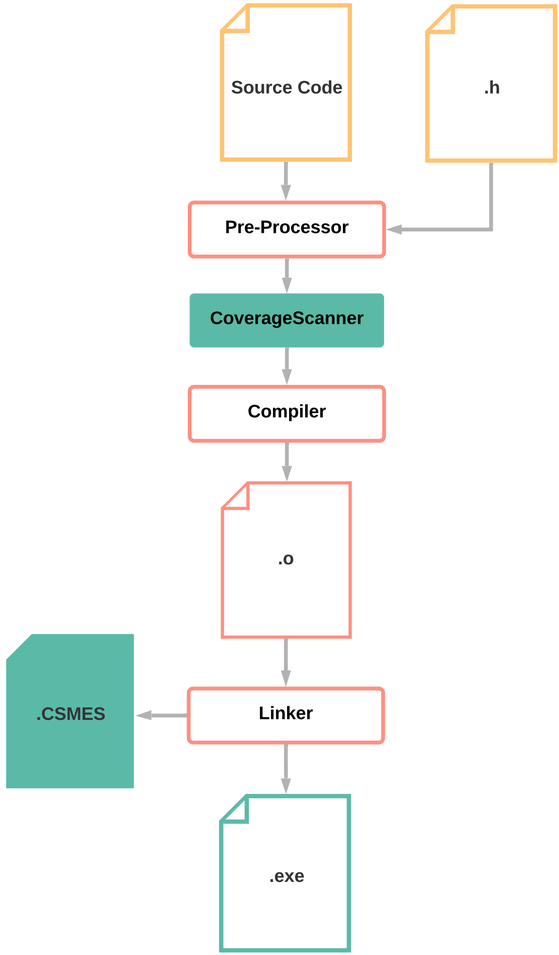
- Function
- Line
- Statement
- Decision (Branch)
- Condition
- MC/DC (Modified Condition/Decision Coverage)
- MCC (Multiple Condition Coverage)

Instrumentation

What is the *CoverageScanner?*

froglogic

- CoverageScanner is Coco's backend instrumentation program
- We must replace the compiler with the CoverageScanner's wrappers
- Instrumentation instructions get inserted into the PP code, and then this modified code is compiled
- The *user* does not modify his/her/their source code

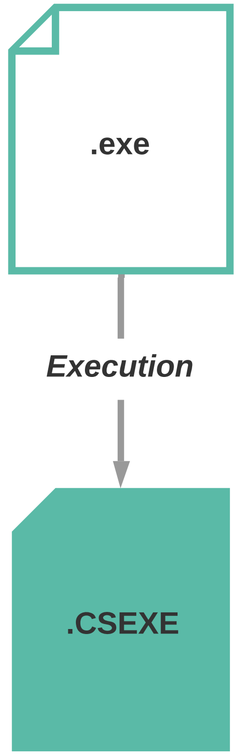


Execution

Running our instrumented binary

froglogic

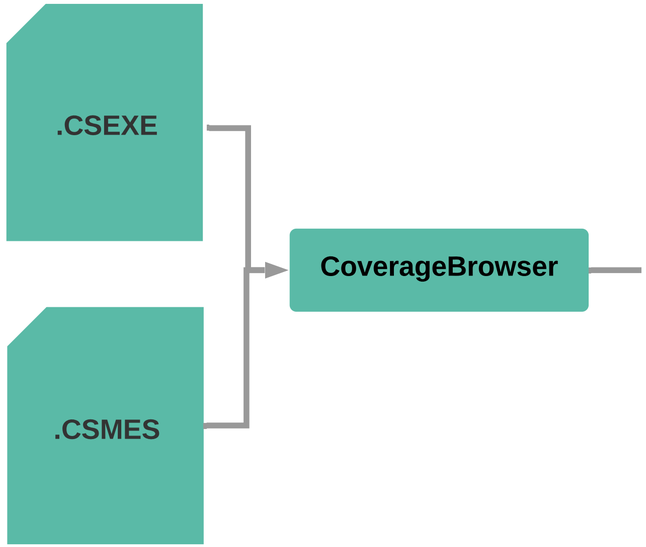
- Any use of the instrumented binary + termination will create a second file: *.csexe.*
- This is the execution report.
- “Any use” could be running a suite of unit tests, or even just opening and closing the app.



Reporting

What is the *CoverageBrowser*?

- Coco's GUI tool, for viewing, analyzing & managing execution reports
- Used to browse the code coverage results interactively
- & Create reports, spreadsheets, ...



froglogic

Qt TextEdit App

Instrumenting a Qt App

Looking at the project file

froglogic

```
01 . . .
13 CodeCoverage {
14
17  COVERAGE_OPTIONS  = --cs-output=textedit
18  COVERAGE_OPTIONS += --cs-exclude-file-abs-wildcard=*/qrc_*
19  COVERAGE_OPTIONS += --cs-exclude-file-abs-wildcard=*.h
20
21  QMAKE_CFLAGS   += $$COVERAGE_OPTIONS
22  QMAKE_CXXFLAGS += $$COVERAGE_OPTIONS
23  QMAKE_LFLAGS   += $$COVERAGE_OPTIONS
24
25  QMAKE_CC=csg++
26  QMAKE_CXX=csg++
27  QMAKE_LINK=csg++
28  QMAKE_LINK_SHLIB=cs$$QMAKE_LINK_SHLIB
29  QMAKE_AR=cs$$QMAKE_AR
30  QMAKE_LIB=cs$$QMAKE_LIB
31 }
32 . . .
```

1. - Define a coverage *scope*

Instrumenting a Qt App

Looking at the project file

```
01 . . .
13 CodeCoverage {
14
17  COVERAGE_OPTIONS  = --cs-output=textedit
18  COVERAGE_OPTIONS += --cs-exclude-file-abs-wildcard=*/qrc_*
19  COVERAGE_OPTIONS += --cs-exclude-file-abs-wildcard=*.h
20
21  QMAKE_CFLAGS   += $$COVERAGE_OPTIONS
22  QMAKE_CXXFLAGS += $$COVERAGE_OPTIONS
23  QMAKE_LFLAGS   += $$COVERAGE_OPTIONS
24
25  QMAKE_CC=csg++
26  QMAKE_CXX=csg++
27  QMAKE_LINK=csg++
28  QMAKE_LINK_SHLIB=cs$$QMAKE_LINK_SHLIB
29  QMAKE_AR=cs$$QMAKE_AR
30  QMAKE_LIB=cs$$QMAKE_LIB
31 }
32 . . .
```

1. - Define a coverage *scope*

2. - Customize the coverage & exclude files from instrumentation

Instrumenting a Qt App

Looking at the project file

froglogic

```
01 . . .
13 CodeCoverage {
14
17  COVERAGE_OPTIONS  = --cs-output=textedit
18  COVERAGE_OPTIONS += --cs-exclude-file-abs-wildcard=*/qrc_*
19  COVERAGE_OPTIONS += --cs-exclude-file-abs-wildcard=*.h
20
21  QMAKE_CFLAGS   += $$COVERAGE_OPTIONS
22  QMAKE_CXXFLAGS += $$COVERAGE_OPTIONS
23  QMAKE_LFLAGS   += $$COVERAGE_OPTIONS
24
25  QMAKE_CC=csg++
26  QMAKE_CXX=csg++
27  QMAKE_LINK=csg++
28  QMAKE_LINK_SHLIB=cs$$QMAKE_LINK_SHLIB
29  QMAKE_AR=cs$$QMAKE_AR
30  QMAKE_LIB=cs$$QMAKE_LIB
31 }
32 . . .
```

1. - Define a coverage *scope*

2. - Customize the coverage & exclude files from instrumentation

3. - Set compiler & linker flags

Instrumenting a Qt App

Looking at the project file

```
01 . . .
13 CodeCoverage {
14
17  COVERAGE_OPTIONS  = --cs-output=textedit
18  COVERAGE_OPTIONS += --cs-exclude-file-abs-wildcard=*/qrc_*
19  COVERAGE_OPTIONS += --cs-exclude-file-abs-wildcard=*.h
20
21  QMAKE_CFLAGS   += $$COVERAGE_OPTIONS
22  QMAKE_CXXFLAGS += $$COVERAGE_OPTIONS
23  QMAKE_LFLAGS   += $$COVERAGE_OPTIONS
24
25  QMAKE_CC=csg++
26  QMAKE_CXX=cs$$QMAKE_CXX
27  QMAKE_LINK=cs$$QMAKE_LINK
28  QMAKE_LINK_SHLIB=cs$$QMAKE_LINK_SHLIB
29  QMAKE_AR=cs$$QMAKE_AR
30  QMAKE_LIB=cs$$QMAKE_LIB
31 }
32 . . .
```

1. - Define a coverage *scope*

2. - Customize the coverage & exclude files from instrumentation

3. - Set compiler & linker flags

4. - Instruct *qmake* to use the CoverageScanner's compiler wrappers

Instrumenting a Qt App

Looking at the project file


```
01 . . .
13 CodeCoverage {
14
17  COVERAGE_OPTIONS  = --cs-output=textedit
18  COVERAGE_OPTIONS += --cs-exclude-file-abs-wildcard=*/qrc_*
19  COVERAGE_OPTIONS += --cs-exclude-file-abs-wildcard=*.h
20
21  QMAKE_CFLAGS   += $$COVERAGE_OPTIONS
22  QMAKE_CXXFLAGS += $$COVERAGE_OPTIONS
23  QMAKE_LFLAGS   += $$COVERAGE_OPTIONS
24
25  QMAKE_CC=csg++
26  QMAKE_CXX=cs$$QMAKE_CXX
27  QMAKE_LINK=cs$$QMAKE_LINK
28  QMAKE_LINK_SHLIB=cs$$QMAKE_LINK_SHLIB
29  QMAKE_AR=cs$$QMAKE_AR
30  QMAKE_LIB=cs$$QMAKE_LIB
31 }
32 . . .
```

1. - Define a coverage *scope*

2. - Customize the coverage & exclude files from instrumentation

3. - Set compiler & linker flags

4. - Instruct *qmake* to use the CoverageScanner's compiler wrappers

Creating the app:

```
$ qmake CONFIG+=CodeCoverage
$ make
```

Testing Strategies

Combining dev & QA

froglogic

- Coco supports a range of testing strategies.
- Today, we'll look at:
  - Unit tests
  - Interactive tests
  - Automated GUI tests

froglogic

Our first test

The Unit Test

froglogic

- Both the application and the unit test must be instrumented in the same way.

```
1  #include "tst_textedit.h"
2
3  void TestTextEdit::tst_saveFile()
4  {
5    TextEdit textEdit;
6    textEdit.fileName="/";
7    QVERIFY( ! textEdit.fileSave() );
8  }
9
10 QTEST_MAIN(TestTextEdit);
11
```

```
1  #include "testcoverageobject.h"
2  #include <QTest>
3  #include <QMetaObject>
4  #include <QString>
5
6  void TestCoverageObject::init()
7  {
8  #ifdef __COVERAGESCANNER__
9    __coveragescanner_clear();
10 #endif
11   initTest();
12 }
13
14 void TestCoverageObject::cleanup()
15 {
16   cleanupTest();
17 #ifdef __COVERAGESCANNER__
18   static const char *default_name = "unnamed";
19   QString test_name="unittest/";
20   test_name+=metaObject()->className();
21   test_name+="/";
22   test_name+=QTest::currentTestFunction();
23   __coveragescanner_testname(test_name.toLatin1());
24   if (QTest::currentTestFailed())
25     __coveragescanner_teststate("FAILED");
26   else
27     __coveragescanner_teststate("PASSED") ;
28   __coveragescanner_save();
29   __coveragescanner_testname(default_name);
30 #endif
31 }
```

Automated GUI Tests

froglogic

- We’ll use the Squish GUI Tester as our framework

froglogic

A few Coco features

Continuous Integration

froglogic

Coco in the DevOps pipeline

- **Problem**:
  - Increasing demand on quality with ever-shortening release cycles
  - Must integrate code coverage in build/test infrastructure for fast feedback
- **Solution**: CI support in Coco: Jenkins, Bamboo, SonarQube, ...
  - Set coverage thresholds on build pass/fail
  - Supports report generation
  - View coverage over time

Test Case Prioritization

froglogic

Optimized Execution Order

- **Problem**: Must eliminate redundancy
- **Solution**: Schedule tests to maximize coverage efficiency

Test Impact Analysis

froglogic

Patch Analysis

- **Problem**: Last-minute fix must be evaluated, but no time to run the full suite
- **Solution**: Determine which tests exercise the changed code, and run only those

Blackbox Testing

For Distributed Teams

froglogic

- **Problem**: Distributed or outsourced QA team
- **Solution**:
  - Generate blackbox database & ship instrumented builds to QA
    - Blackbox database shows only coverage levels, commented executions, etc. No source code.
  - Developer with master access to the src collates QA's reports into one master report

Learn More

froglogic

Coco homepage: froglogic.com/coco

Free, fully-supported & fully-featured evaluation: froglogic.com/coco/free-trial/

Online docs: doc.froglogic.com/squish-coco

Stay In Touch With Us

froglogic

Contact support via email:

squish@froglogic.com

Follow us:

For developer tips, feature previews, product news & more....

Twitter: @froglogic
Facebook: /froglogic
LinkedIn: /company/froglogic
YouTube: /froglogic

froglogic

Q&A