8/21/2015

MITK Testing with CppUnit

Thomas Kilgus

dkfz.
DEUTSCHES KREBSFORSCHUNGSZENTRUM
IN DER HELMHOLTZ-GEMEINSCHAFT

What's important for tests?

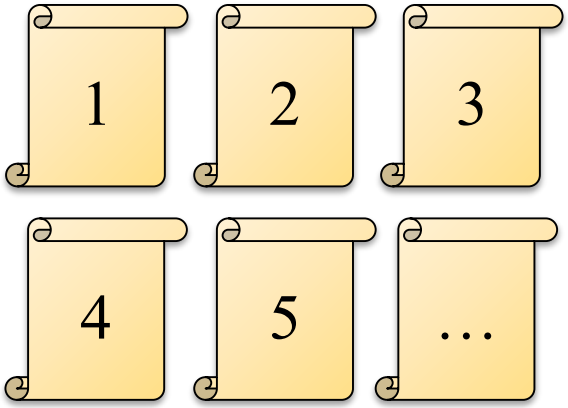
- Maintainability
- Readability
- Trustworthiness

- The obvious: check actual logic of code
- To have documentation!
- To identify quickly which line of code is broken

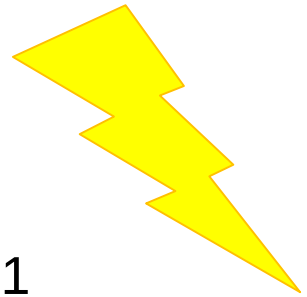
- You want users to be able to read tests like a book
- You don't want to debug/disable a test
- You don't want to spend much time if a test fails

What is a Unit test?

- Small piece of code which runs automatically in memory
- Unit test = **Isolation**
- One Unit test for exactly **one** thing
- (No database- or internet-connection, system calls, hard disc access nor threads are used)



Nomenclature issue



- CppUnit output for a test suite:
  - OK (6 tests)
- CTest output:
  - 100% tests passed, 0 tests failed out of 1

- **Test suite**
  - test class = set of tests = 1 entry on CDash
  - Commonly referred to as „test“ which technically wrong!

- **Test**
  - test method = test case = (assertion) = test in CppUnit

- We have **test suites**/classes and **test methods**!

Test suite and test method examples

```
class mitkImageEqualTestSuite : public mitk::TestFixture
{
CPPUNIT_TEST_SUITE(mitkImageEqualTestSuite);
MITK_TEST(Equal_CloneAndOriginal_ReturnsTrue);
MITK_TEST(Equal_DifferentPixelTypes_ReturnsFalse);
…
CPPUNIT_TEST_SUITE_END();
…
```

- Create a suite
- Register your test methods

setUp()

```
mitk::Image::Pointer m_Image;
mitk::Image::Pointer m_AnotherImage;

void setUp()
{
//generate a gradient test image
m_Image = itk::ImageGenerator::GenerateGradientImage
        <unsigned char>(3u, 3u, 1u);
m_AnotherImage = m_Image->Clone();
}
```

- Use setUp() to freshly initialize each test
- Test suites may have members

tearDown()

```
void tearDown()
{
m_Image = nullptr;
m_AnotherImage = nullptr;
}
```

- Use tearDown() to clean up memory for each test

2 simple tests

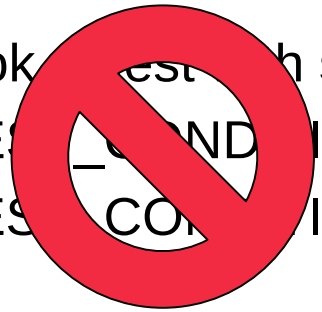
```
void Equal_CloneAndOriginal_ReturnsTrue()
{
MITK_ASSERT_EQUAL( m_Image, m_Image->Clone(),
"A clone should be equal to its original.");
}

void Equal_DifferentPixelTypes_ReturnsFalse()
{
m_AnotherImage = //generate float image
MITK_ASSERT_NOT_EQUAL(m_Image, m_AnotherImage
, "One pixel type is float, the other unsigned
char. Result should be false.");
}
```

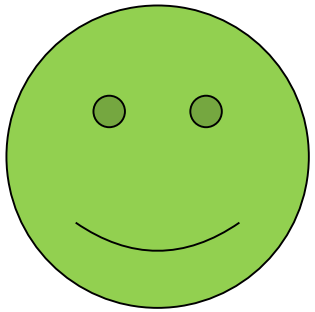
Naming conventions

- Test suite is usually named
  - <CLASS_TO_TEST>TestSuite, but not always!
  - <FUNCTIONALITY_TO_TEST<TestSuite
  - E.g. ImageTestSuite, ImageEqualTestSuite
- Test methods are usually named horribly e.g.:
- → TestImage1() = no (new) information for the reader
- How about more information?
  - Equal_ValidImageAndClone_ReturnsTrue()
- <METHOD_TO_TEST>_<INPUT>_<EXPECTED_RESULT>

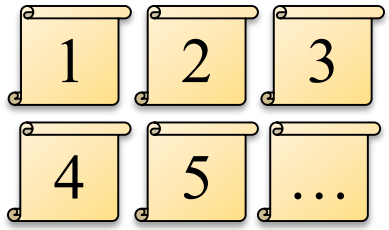
Don’t use deprecated methods and old code



- Never look [illegible] std::cout! It’s usually a trap!
- MITK_TE[illegible]ND[illegible]ION_REQUIRED
- MITK_TE[illegible]CO[illegible]ION



- CPPUNIT_ASSERT
- MITK_ASSERT_EQUAL
- Small tests with nice names!



- Questions?

For more information…

- Please refer to:
  - General documentation
  - http://docs.mitk.org/nightly/GeneralTests.html
  - My last seminar about the CppUnit framework
  - http://www.mitk.org/images/5/5d/BugSquashingSeminars%24CppUnitFrameworkSeminar.pdf