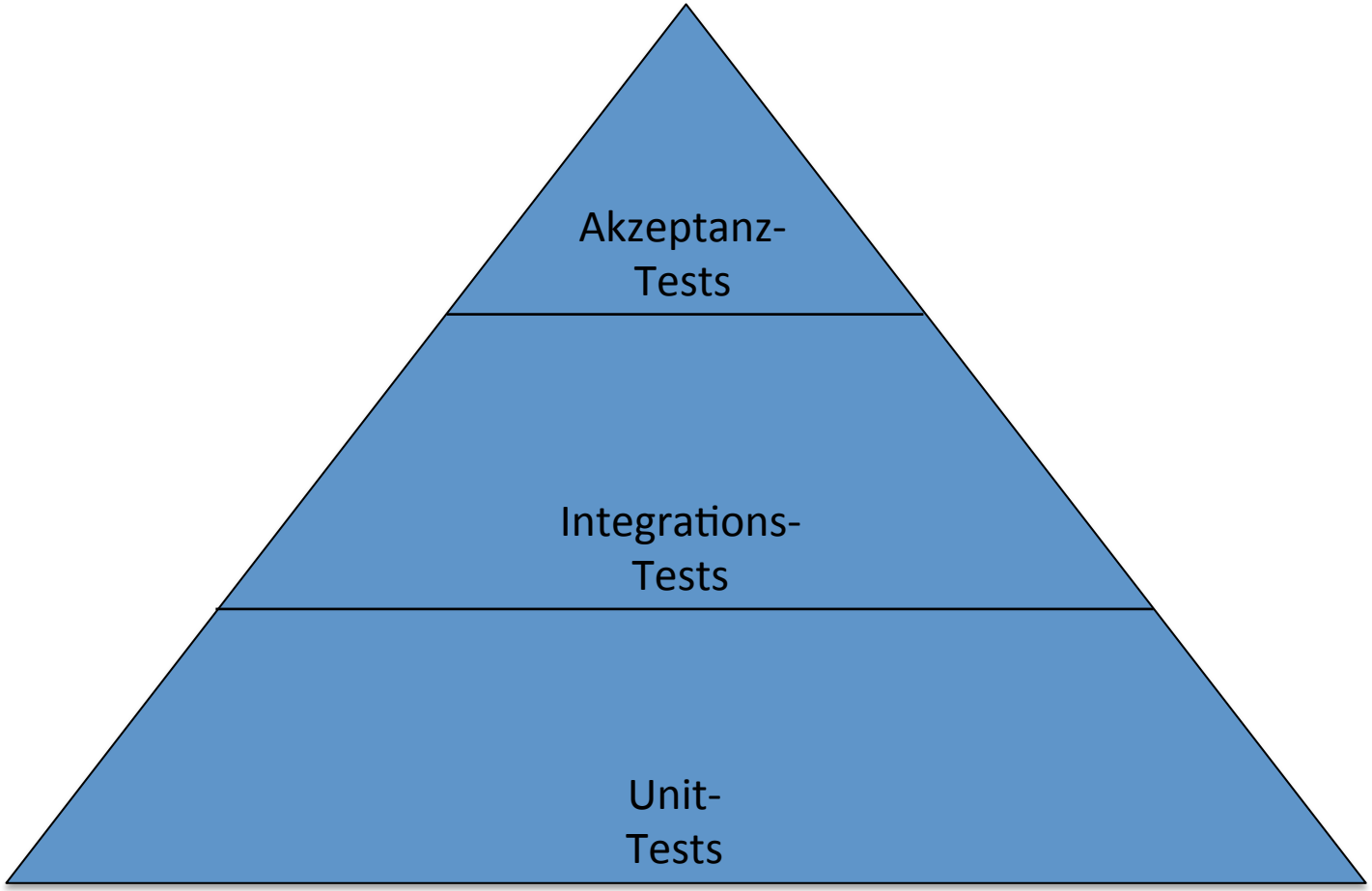
Effektives Testen in C++

C++ UG KA; Mai 2017; Lightning-Talk

Dr. David Faragó; QPR Technologies; farago@gmx.de

Test-Struktur

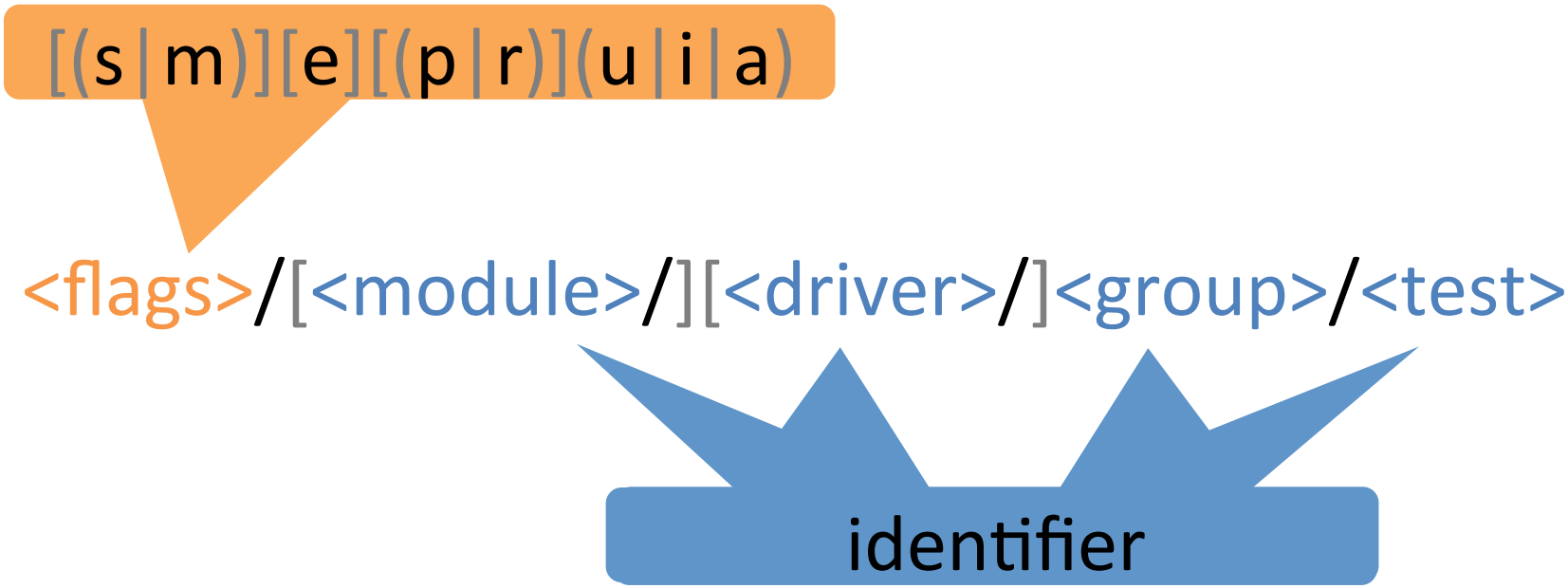
Mike Cohn:
„An effective test automation strategy calls for automating tests at three different levels.“



Test-Struktur

	Unit-Tests	Integrations-Tests	Akzeptanz-Tests
System Under Test	Eine Unit	Kommunikation zwischen mehreren Units	ganzes Programm
Umgebung	**Stubs/Fakes/Mocks** und Produktivcode	Stubs/Fakes/Mocks und **Produktivcode**	Ende-zu-Ende-Test des ganzen Programms
Qualität	intern (developer facing)	intern & extern	extern (customer facing)
Speed	fast	fast or medium	fast or medium or slow

Test-Benennung



Test-Ausführung (z.B. ctest)

Wann	Welche	Wie
Compile	Fast non-experimental functional Unit-Tests des Moduls	-R ^u.* <module>
Commit	Fast non-experimental functional Unit-Tests	-R ^u
Push	Non-slow non-experimental Tests	-E ^"[s|e|me]"
Sprint	Non-experimental Tests	-E ^.?e

gtest-Integration mit cmake & ctest

Mit cmake:

- `include(CMakeLists.Gtest)`, siehe https://github.com/google/googletest/tree/master/googletest
- `add_executable(FooTarget FooTest.cpp)`
- `target_link_libraries(FooTarget gtest)`

Mit ctest via Macro (Discoverability jedes einzelnen Tests):

```
add_executable(${TESTNAME}Test ${TESTNAME}Test.cpp)
target_link_libraries(${TESTNAME}Test ${TEST_MODULE_NAME} ${TEST_TARGET_LINK_LIBRARIES})
set_property(TARGET ${TESTNAME}Test PROPERTY FOLDER "Modules/${TEST_MODULE_NAME}/UnitTests")
file(STRINGS "${TESTNAME}Test.cpp" TEST_SOURCES)
foreach(LINE ${TEST_SOURCES})
  string(REGEX MATCH "^[ ]*TEST[ ]*[(][ ]*([^,]*)[ ]*,[ ]*([^,]*)[ ]*[)][ ]*$" MATCH ${LINE})
  if (MATCH) add_test(NAME u/${TEST_MODULE_NAME}/${TESTNAME}/${CMAKE_MATCH_1}/${CMAKE_MATCH_2} COMMAND
          $<TARGET_FILE:${TESTNAME}Test> "--gtest_filter=${CMAKE_MATCH_1}.${CMAKE_MATCH_2}") endif (MATCH)
endforeach(LINE ${TEST_SOURCES})
```

value-parameterized gtests

```
class ParamTest : public ::testing::TestWithParam
                  <::testing::tuple<int, std::string>> {
protected:
    virtual void SetUp() {
        m_intParam = ::testing::get<0>(GetParam());
        m_stringParam = ::testing::get<1>(GetParam());
    }
    virtual void TearDown() {}
    int m_intParam;
    std::string m_stringParam;
};

TEST_P(ParamTest, testName) { ... }

INSTANTIATE_TEST_CASE_P(evenParams, ParamTest, ::testing::Combine(
    ::testing::Values(2,4,10,16,32,100,256,1000,1024),
    ::testing::Values("foo", "bar")));
```

interface gtests

```
typedef std::function<std::unique_ptr<MyI>(void)> IFactory;

class ITest : public ::testing::TestWithParam<IFactory> {
protected:
    virtual void SetUp() {
        m_implementation = (GetParam())();
    }
    virtual void TearDown() {}

    std::unique_ptr<MyI> m_implementation;
};

TEST_P(ITest, ITestName) { ... }

INSTANTIATE_TEST_CASE_P(nullSimpleAndFullImpl, ITest, ::testing::
      Values(createNullImpl, createSimpleImpl, createFullImpl));
```

Diskussion

Literatur

M. Casperson. *Just Say Yes To More End-to-End Tests*. Article, 2016 (https://dzone.com/articles/just-say-yes-to-more-end-to-end-tests).

M. Cohn. *Succeeding with Agile: Software Development Using Scrum.* Addison-Wesley, 2009.

M. Fowler. *TestPyramid*. Blogpost, 2012 (https://martinfowler.com/bliki/TestPyramid.html).

S. Freeman, N. Pryce. *Growing Object-Oriented Software, Guided by Tests*. Addison Wesley, 2009.

A. Hunt, D. Thomas. *Pragmatic Unit Testing in Java with Junit.* Pragmatic Bookshelf, 2003.

K. Kapelonis. *How to Split JUnit Tests in a Continuous Integration Environment*. Blogpost, 2016 (https://semaphoreci.com/community/tutorials/how-to-split-junit-tests-in-a-continuous-integration-environment)

D. Lindner. *Look at the automated tests to diagnose the project ailments*. Blogpost, 2017 (https://schneide.wordpress.com/tag/egg/)

M. Wacker. *Just Say No to More End-to-End Tests*. Blogpost, 2015 (https://testing. googleblog.com/2015/04/just-say-no-to-more-end-to-end-tests.html).