ugtest

Tobias Trautmann

GCSC

May 14, 2020

Outline

Introduction to testing

Goals of testing

- meets its requirements

Goals of testing

- meets its requirements
- performs its functions within an acceptable time

Goals of testing

- meets its requirements
- performs its functions within an acceptable time
- is sufficiently usable

Goals of testing

- meets its requirements
- performs its functions within an acceptable time
- is sufficiently usable
- can be run in its intended environments

Goals of testing

- meets its requirements
- performs its functions within an acceptable time
- is sufficiently usable
- can be run in its intended environments
- increase trust in results

Goals of testing

- meets its requirements
- performs its functions within an acceptable time
- is sufficiently usable
- can be run in its intended environments
- increase trust in results
- make code maintainable & refactorable

Goals of testing

- meets its requirements
- performs its functions within an acceptable time
- is sufficiently usable
- can be run in its intended environments
- increase trust in results
- make code maintainable & refactorable

$\Rightarrow$ Testing software is necessary

Definitions

Defects

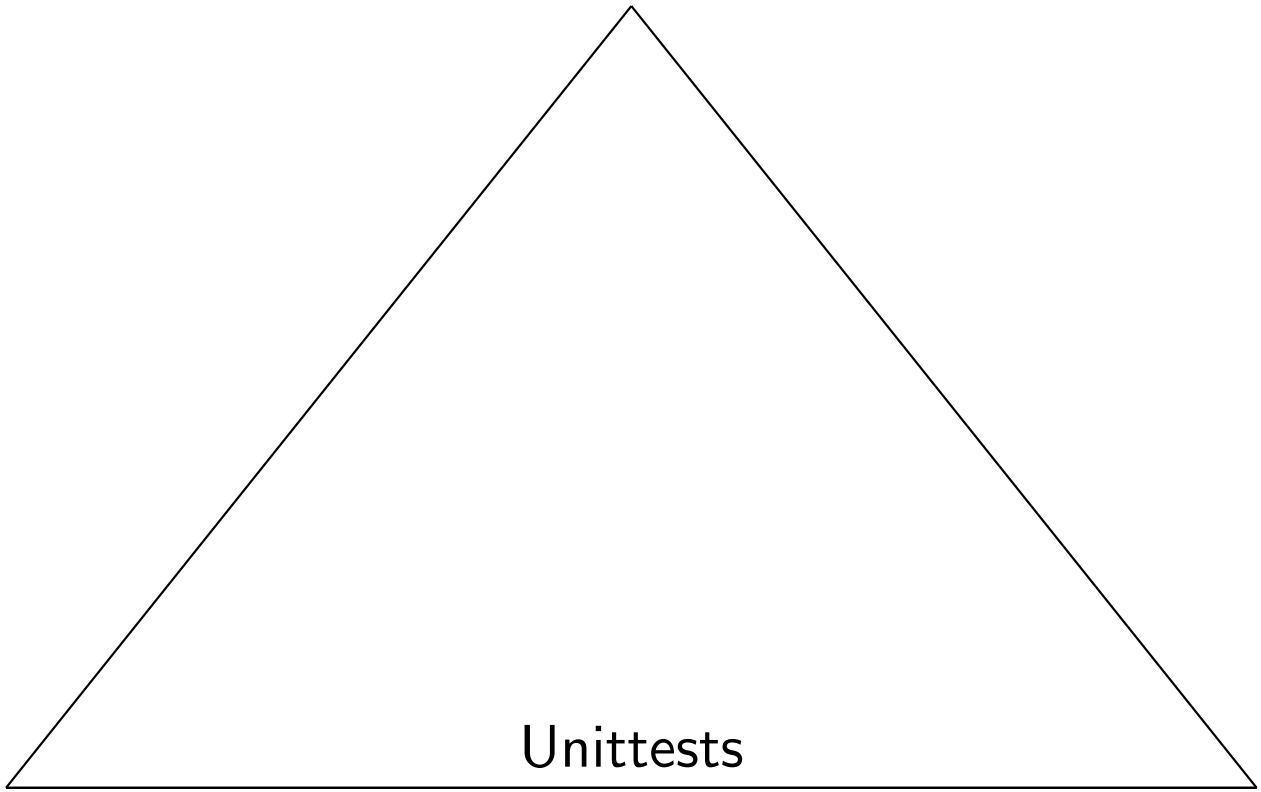
mehr buch
Where do defects come from?
Prioritize defects
Are you responsible for it?

mitigation	bug in code	integration	error in design

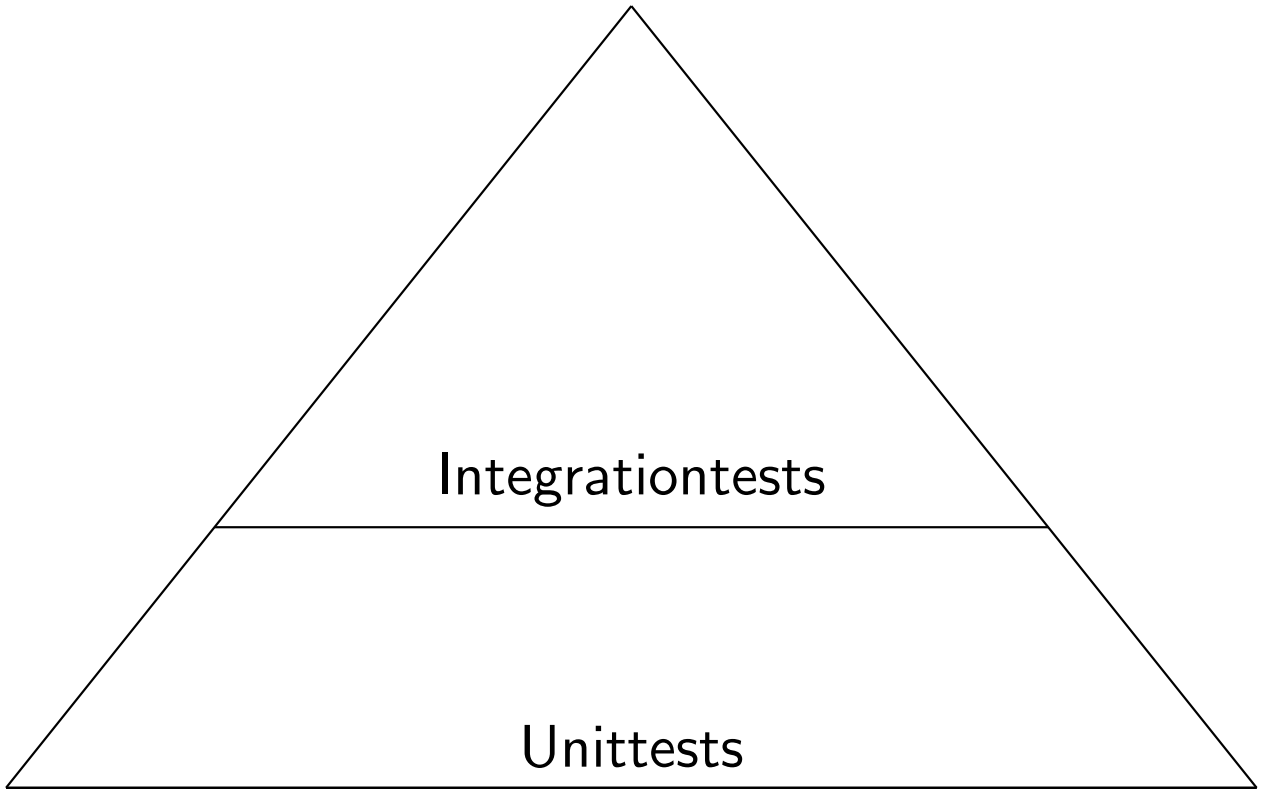
Definition of done

- Code
  -
- Tests
  - Coverage
- Documentation
  - User
  - Maintainer

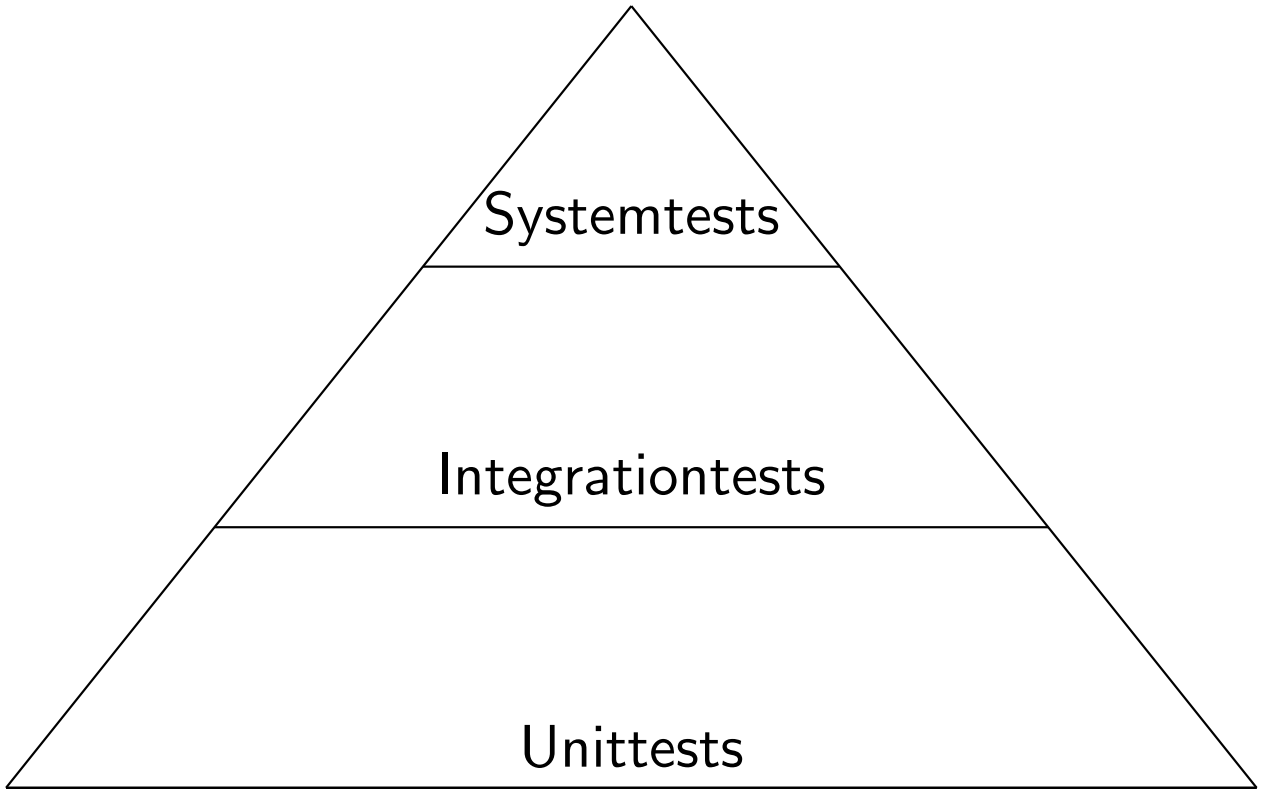
Economics / Efficency



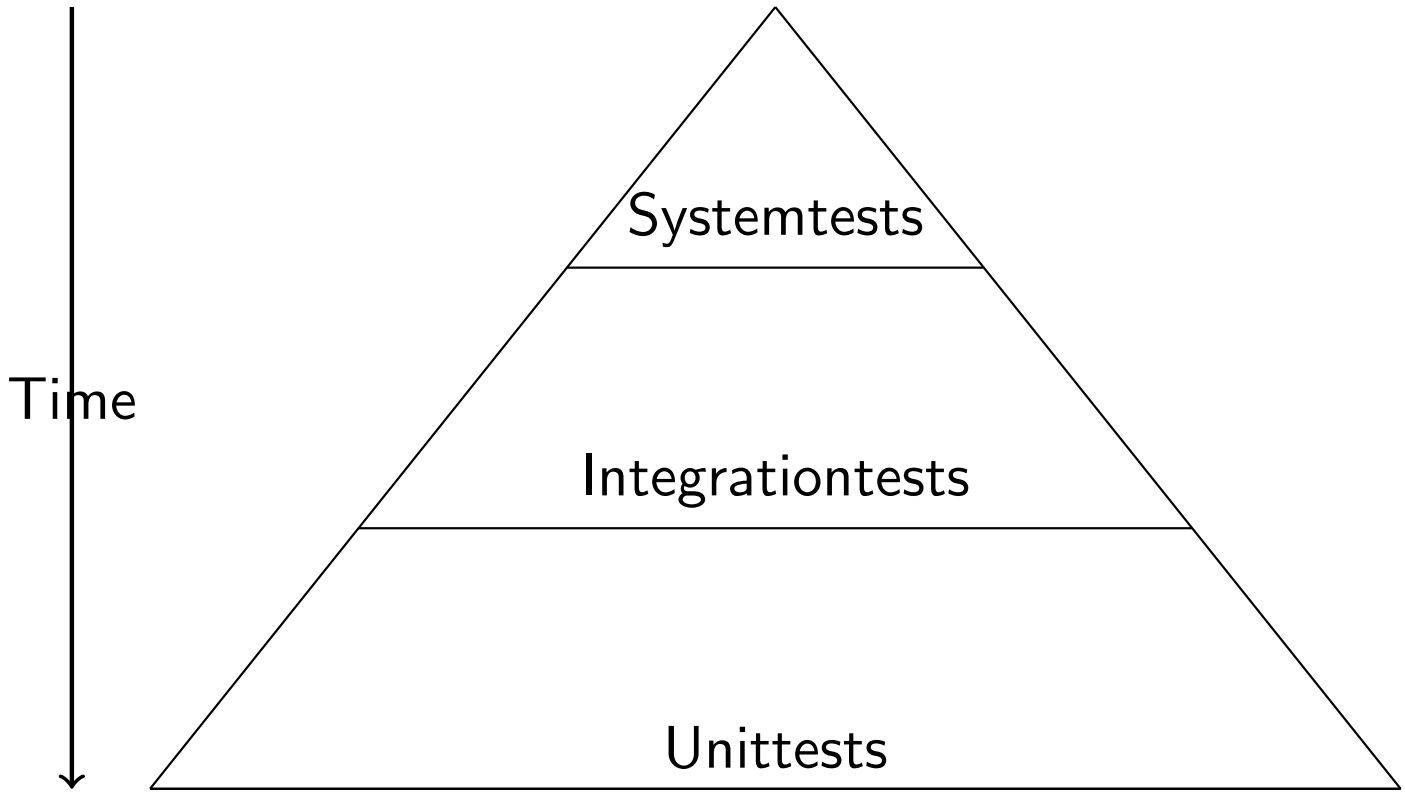
Economics / Efficency



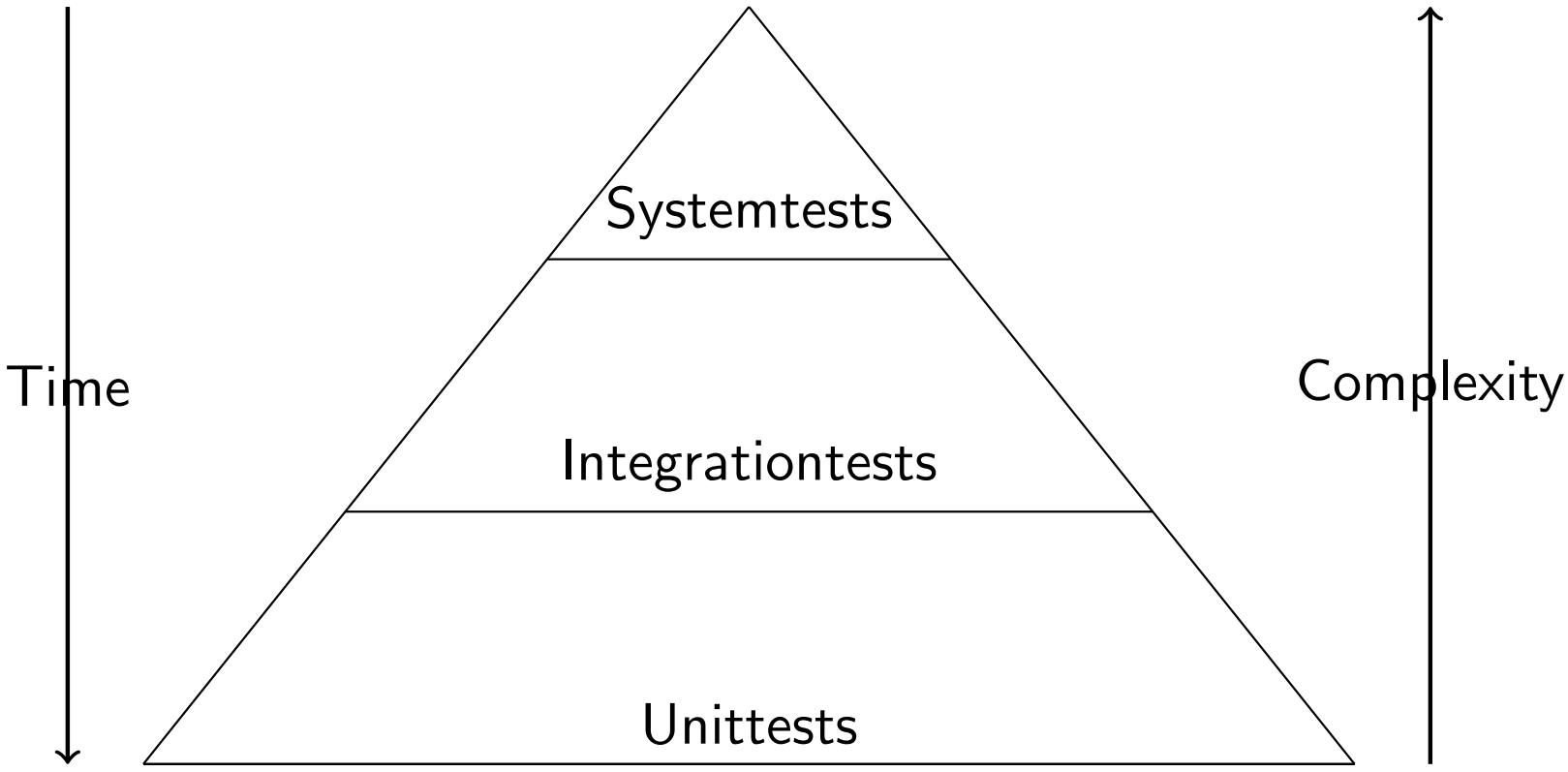
Economics / Efficency



Economics / Efficency

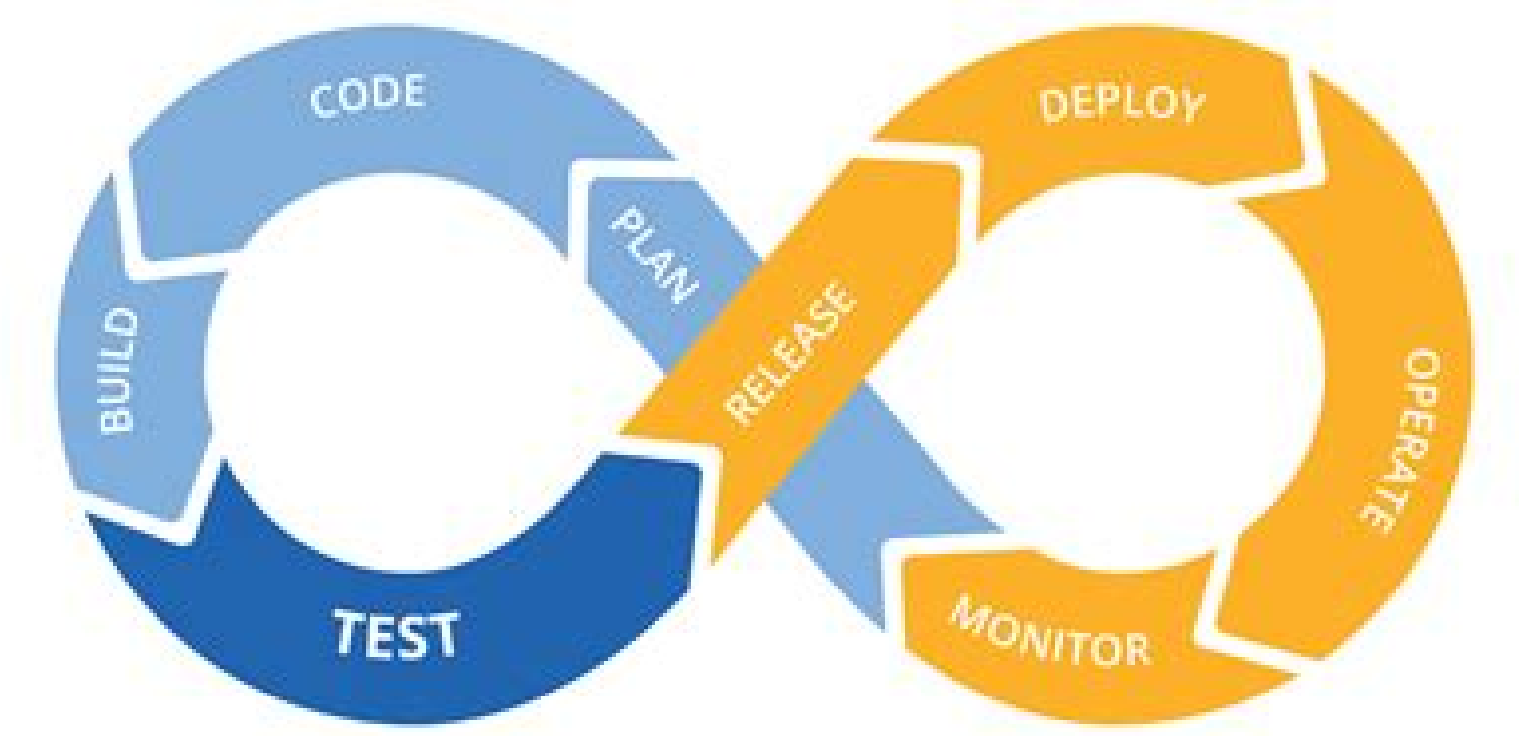


Economics / Efficency

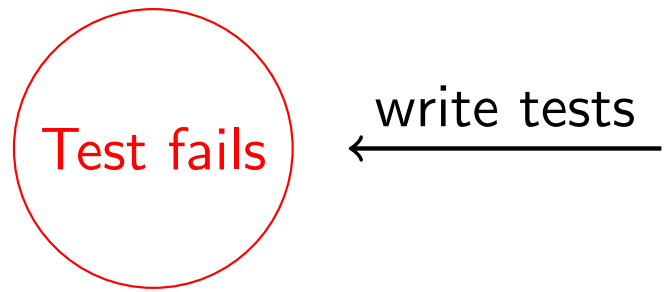


Approaches

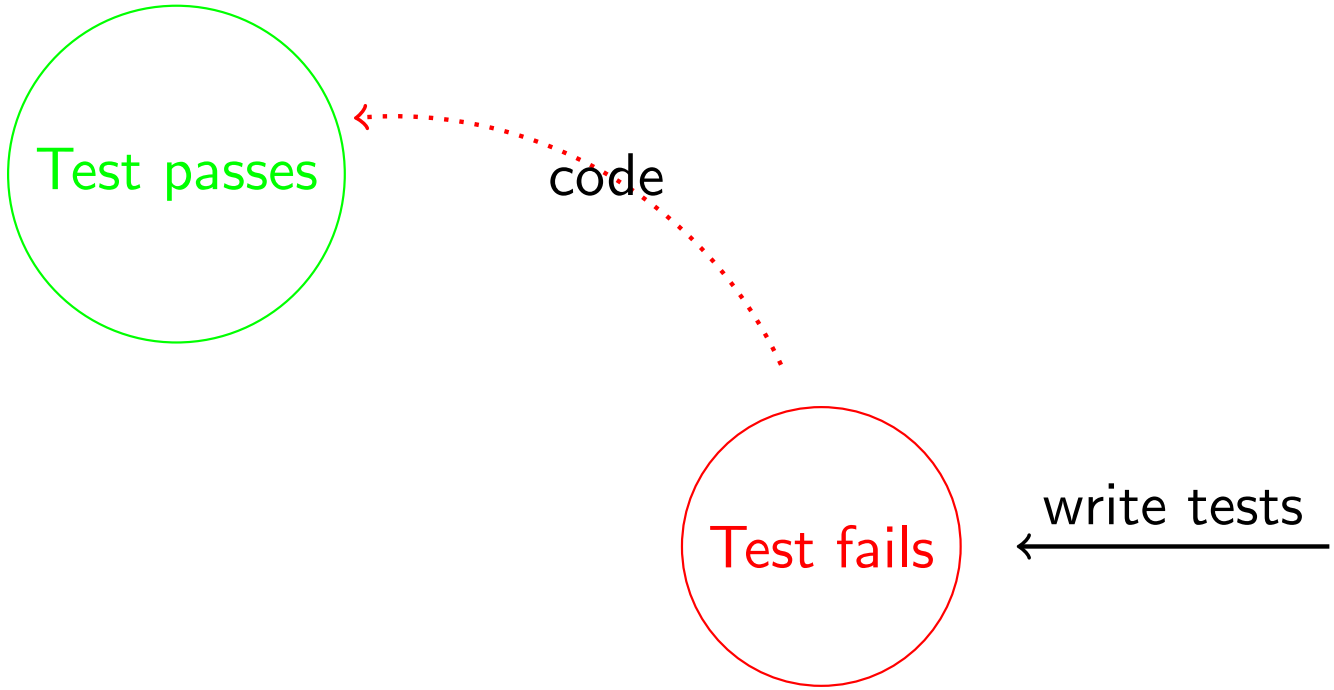
Continous Integration / Continous Delivery



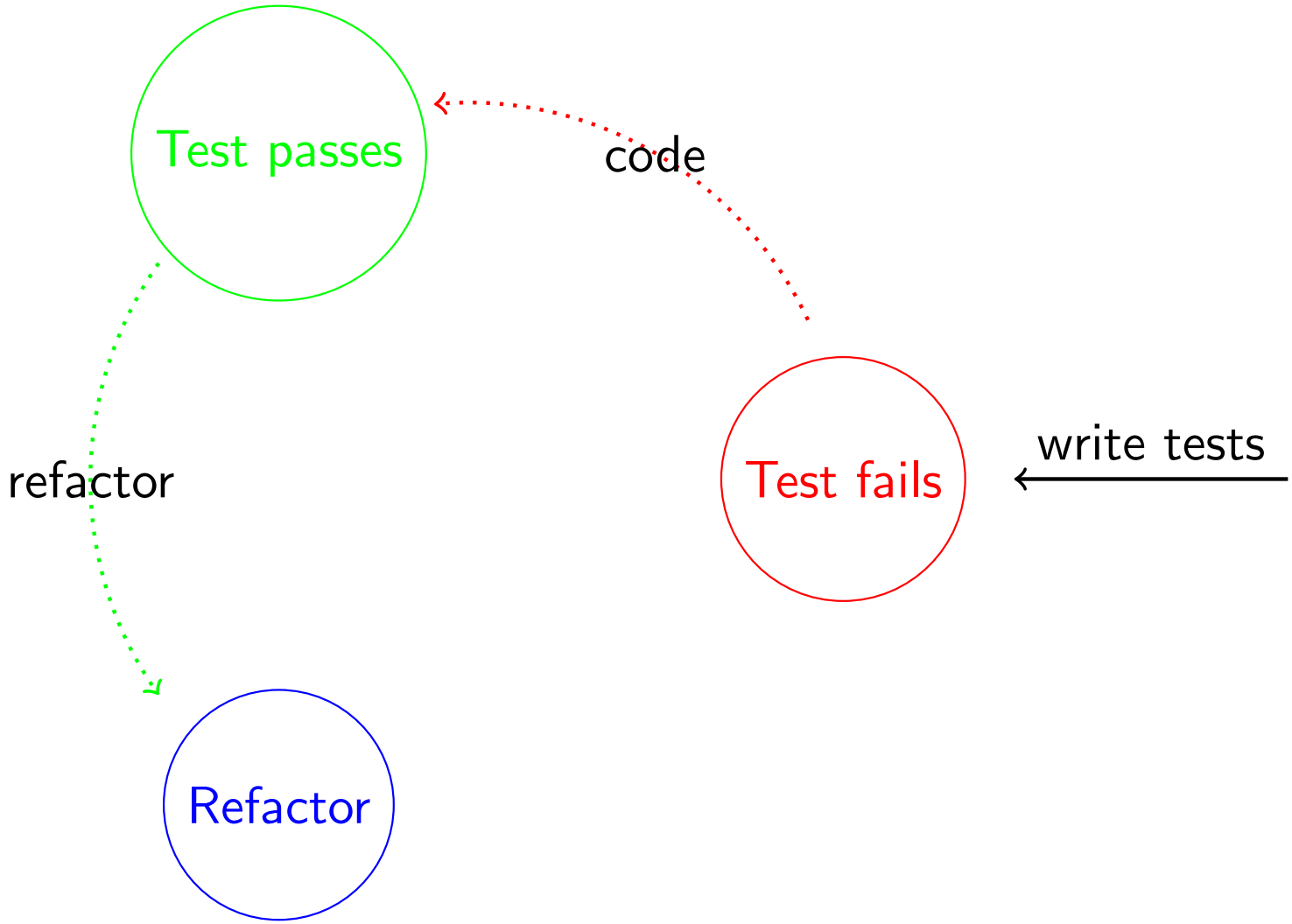
Test driven development



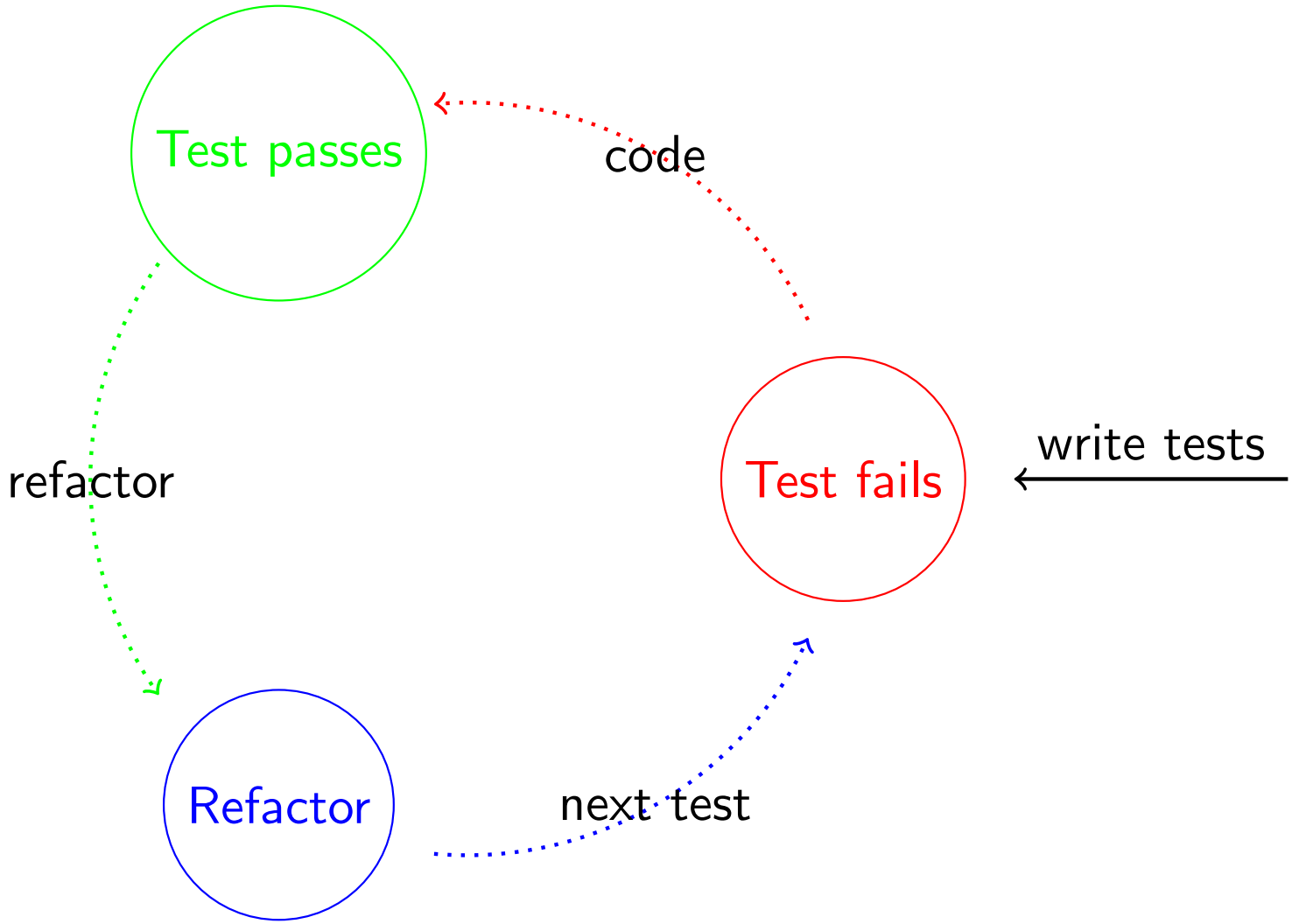
Test driven development



Test driven development



Test driven development



Boost.Test

Structure

```
#include <boost/test/included/unit_test.hpp>

//uncomment if using templates
//#include <boost/test/test_case_template.hpp>

//uncomment if testing in parallel
//#include <boost/mpl/list.hpp>

//stuff

BOOST_AUTO_TEST_SUITE(<testsuite_name>)

    BOOST_AUTO_TEST_CASE(<testcase_name>)
    {
        //Testcase here
    }
    BOOST_AUTO_TEST_CASE(<testcase_name>)
    {
        //Testcase here too
    }

BOOST_AUTO_TEST_SUITE_END()
```

Error Levels

error level	error counter	test continuation
warn		yes
check	++	yes
require	++	no

Basic Checks

- BOOST_<level>(predicate)
- BOOST_<level>_<GE,LE,GT,LT,NE>(left, right)
- BOOST_<level>_EQUAL(left, right)
- BOOST_IS_DEFINED(SYMBOL)

Warn

```
BOOST_AUTO_TEST_CASE(warn){
    BOOST_MESSAGE("showcasing warn");
    BOOST_WARN();
}
```

Check

```
BOOST_AUTO_TEST_CASE(check){
    BOOST_MESSAGE("showcasing check");
    BOOST_CHECK();
}
```

Require

```
BOOST_AUTO_TEST_CASE(require){
    BOOST_MESSAGE("showcasing require");
    BOOST_REQUIRE();
    BOOST_REQUIRE(false);
    BOOST_MESSAGE("this is unreachable");
}
```

Float point comparison

BOOST_<level>_CLOSE(left, right, tolerance)

Float point comparison

Exception handling

- BOOST_<level>_THROW(expression, exception_type)
- BOOST_<level>_NO_THROW(expression)

Exception handling

Fixtures

```
struct UGbase
{
    //Call UGInit before testcase starts
    UGbase()
    {
        ug::UGInit(&framework::master_test_suite().argc,
            &framework::master_test_suite().argv);
    }
    //call UGFinalize after test case ends
    ~UGbase()  {
        ug::UGFinalize();
    }
};
BOOST_AUTO_TEST_SUITE(fixtureshowsuite)
BOOST_AUTO_TEST_CASE(fixtureshowcase, UGbase){
    //your test needing a clean ug
}
BOOST_AUTO_TEST_SUITE_END()
```

Templates

```
typedef boost::mpl::list<int, long, unsigned char>
    test_types;

BOOST_AUTO_TEST_SUITE(templateshowsuite)
    BOOST_AUTO_TEST_CASE_TEMPLATE(templateshowcase, T,
        test_types)
    {
        //your super fancy template test
    }
BOOST_AUTO_TEST_SUITE_END()
```

Testing

Test execution

- add buildflags "–fprofile-arcs –ftest-coverage –fPIC" as well as no optimization for code coverage analysis
- build ug with UGTest and your plugin activated
- your plugin contains tests in a top level folder named "tests"
-
- list of params
- example:

  ug4/bin $ ./ ugtest_unit −−log−level=ALL −−log−format=HRF

Automatization with Jenkins

- Cobertura
- two builds one serial, one parallel
- Code coverage: gcovr can produce xml for cobertura

Automatization with Docker

- Container stuff
- Dockerfile

Additional resources

- Boost.Test documentation
- ugtests github
- Antipatterns

References

- wiki
- Basiswissen Softwaretest