The Not So Short Introduction to Magellan

The Simple xUnit Implements in Modern C++11

GuangCong Liu

ZTE Corporation

2015.08

Contents

Motivation

Lots of Unit Test Frameworks

Who was always a favorite of mine?

1. SUnit
2. JUnit/TestNG
3. CppUnit/GoogleTest/BoostTest
4. RSpec/Shoulda
5. ScalaTest

A Simple Example

```
struct RobotCleanerTest : testing::Test
{
protected:
    RobotCleaner robot;
};

TEST_F(RobotCleanerTest, at_the_beginning_the_robot_should_be_in_at_the_initial_position)
{
    ASSERT_EQ(Position(0, 0, NORTH), robot.getPosition());
}

TEST_F(RobotCleanerTest, should_be_face_west_after_turn_left)
{
   robot.turnLeft();
   ASSERT_EQ(Position(0, 0, WEST), robot.getPosition());
}
```

Override Setup/Teardown

```
numberbychapter
struct RobotCleanerTest : testing::Test
{
    RobotCleaner robot;

    virtual void setUp()
    { ... }

    virtual void tearDown()
    { ... }
};

TEST_F(RobotCleanerTest, should_be_face_west_after_turn_left_1_times)
{
    robot.turnLeft();
    ASSERT_EQ(Position(0, 0, WEST), robot.getPosition());
}

TEST_F(RobotCleanerTest, should_be_face_south_after_turn_left_2_times)
{
    robot.turnLeft();
    robot.turnLeft();
    ASSERT_EQ(Position(0, 0, SOUTH), robot.getPosition());
}
numberbychapter
```

Global Setup/Teardown

```
numberbychapter
struct GlobalEnvironment : testing::Environment
{
    virtual void SetUp()
    { ... }

    virtual void TearDown()
    { ... }
};

int main(int argc, char** argv)
{
    testing::AddGlobalTestEnvironment(new GlobalEnvironment);
    testing::InitGoogleTest(&argc, argv);

    return RUN_ALL_TESTS();
}
numberbychapter
```

Features

Length Calculator

1. 1 FEET == 12 INCH
2. 1 YARD == 3 FEET
3. 1 MILE == 1760 YARD

First Test Case

```
#include "magellan/magellan.hpp"
#include "quantity/Length.h"

USING_HAMCREST_NS

FIXTURE(LengthTest)
{
    TEST("1 FEET == 12 INCH")
    {
        ASSERT_THAT(Length(1, FEET), eq(Length(12, INCH)));
    }
};
```

include/quantity/Length.h

```
numberbychapter
#ifndef SDHS_DFJE_3747_CNDHE_37473_VNNFHEB_DHEYE
#define SDHS_DFJE_3747_CNDHE_37473_VNNFHEB_DHEYE

#include "quantity/Amount.h"
#include "quantity/LengthUnit.h"

struct Length
{
    Length(Amount amount, LengthUnit unit);

    bool operator==(const Length& rhs) const;

private:
    const Amount amountInBaseUnit;
};

#endif
numberbychapter
```

include/quantity/LengthUnit.h

```
numberbychapter
#ifndef ERTYT_CNNDH_ENBAW_648294_NVGDGWE_57VNDHA_CH3
#define ERTYT_CNNDH_ENBAW_648294_NVGDGWE_57VNDHA_CH3

enum LengthUnit
{
    INCH = 1,
    FEET = 12 * INCH
};

#endif
numberbychapter
```

include/quantity/LengthUnit.h

```
#ifndef SDFJCN_E6438C_CNDJ866_VNA001223_VNNDHHE3CHD
#define SDFJCN_E6438C_CNDJ866_VNA001223_VNNDHHE3CHD

using Amount = unsigned int;

#endif
```

src/quantity/Length.cpp

```
#include "quantity/Length.h"

Length::Length(Amount amount, LengthUnit unit)
  : amountInBaseUnit(unit * amount)
{
}

bool Length::operator==(const Length& rhs) const
{
    return amountInBaseUnit == rhs.amountInBaseUnit;
}

#endif
```

Pure OO

```
FIXTURE(RobotCleanerTest)
{
    RobotCleaner robot;

    TEST("at the beginning, the robot should be in at the initial position")
    {
        ASSERT_THAT(robot.getPosition(), is(Position(0, 0, NORTH)));
    }

    TEST("the robot should be faced west after turn left")
    {
        robot.exec(left());
        ASSERT_THAT(robot.getPosition(), is(Position(0, 0, WEST)));
    }
};
```

Extract Method

```
FIXTURE(RobotCleanerTest)
{
    RobotCleaner robot;

    void WHEN_I_send_instruction(Instruction* instruction)
    {
        robot.exec(instruction);
    }

    void THEN_the_robot_cleaner_should_be_in(const Position& position)
    {
        ASSERT_THAT(robot.getPosition(), is(position));
    }

    TEST("at the beginning")
    {
        THEN_the_robot_cleaner_should_be_in(Position(0, 0, NORTH));
    }

    TEST("left instruction: 1-times")
    {
        WHEN_I_send_instruction(left());
        THEN_the_robot_cleaner_should_be_in(Position(0, 0, WEST));
    }
};
```

Assertion and Hamcrest

```
FIXTURE(StartsWithTest)
{
    TEST("case sensitive")
    {
        ASSERT_THAT("ruby-cpp", starts_with("ruby"));
        ASSERT_THAT("ruby-cpp", is(starts_with("ruby")));

        ASSERT_THAT(std::string("ruby-cpp"), starts_with("ruby"));
        ASSERT_THAT("ruby-cpp", starts_with(std::string("ruby")));
        ASSERT_THAT(std::string("ruby-cpp"), starts_with(std::string("ruby")));
    }
};
```

Matcher

1. anything
2. eq/ne/lt/gt/le/lt
3. is/is_not
4. nil
5. be_true/be_false
6. contains_string/starts_with/ends_with
7. close_to/nan
8. and so on...

Scope of Setup/Teardown

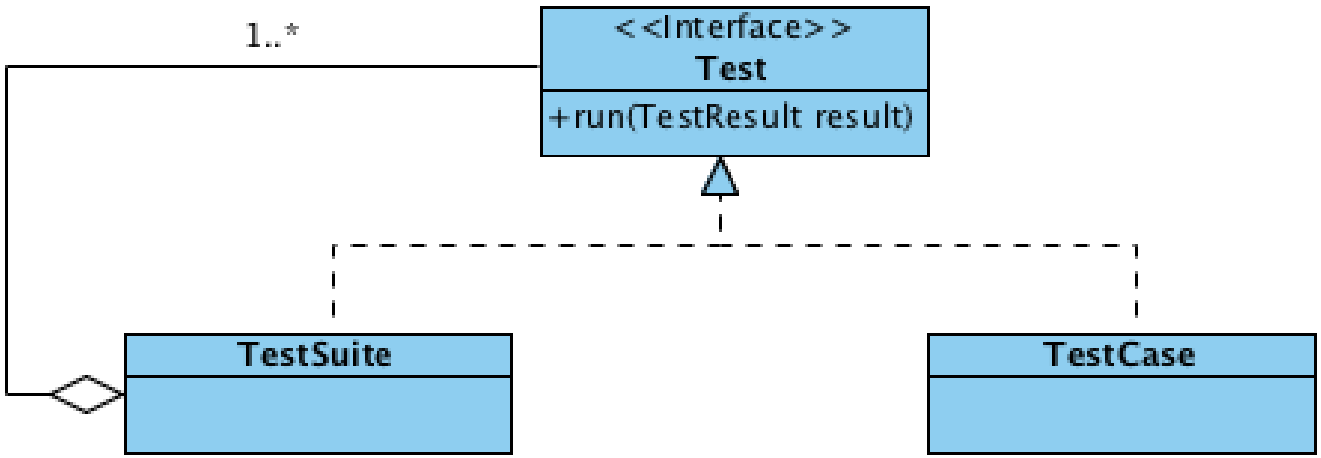
```
BEFORE_ALL("try load global configure")
{ ... }

AFTER_ALL("try load global configure")
{ ... }

FIXTURE(ObjectTest)
{
    BEFORE_CLASS()
    { ... }

    AFTER_CLASS()
    { ... }

    BEFORE()
    { ... }

    AFTER()
    { ... }
};
```

Options

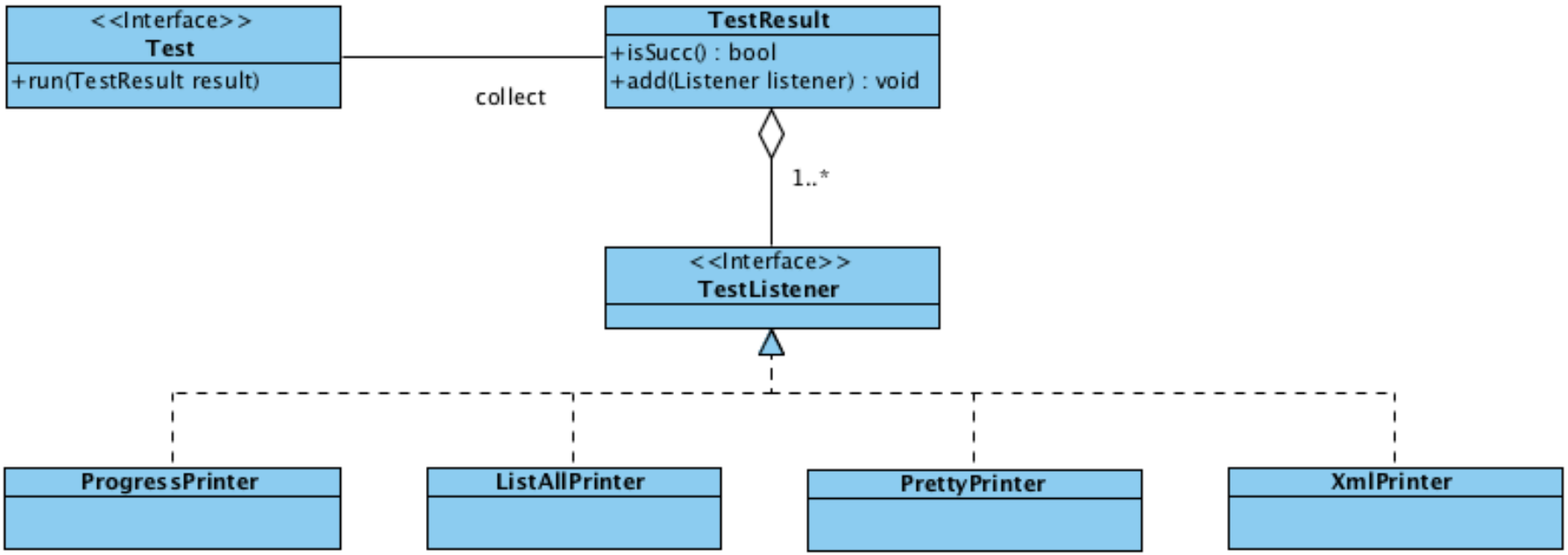
```
TestOptions::TestOptions() : desc("magellan")
{
    desc.add({
        {"help,     h",   "help message"},
        {"filter,   f",   "--filter=pattern"},
        {"color,    c",   "--color=[yes|no]"},
        {"xml,      x",   "print test result into XML file"},
        {"list,     l",   "list all tests without running them"},
        {"progress, p",   "print test result in progress bar"},
        {"verbose,  v",   "verbosely list tests processed"},
        {"repeat,   r",   "how many times to repeat each test"}
    });

    // default value
    options["color"]  = "yes";
    options["repeat"] = "1";
}
```

Design

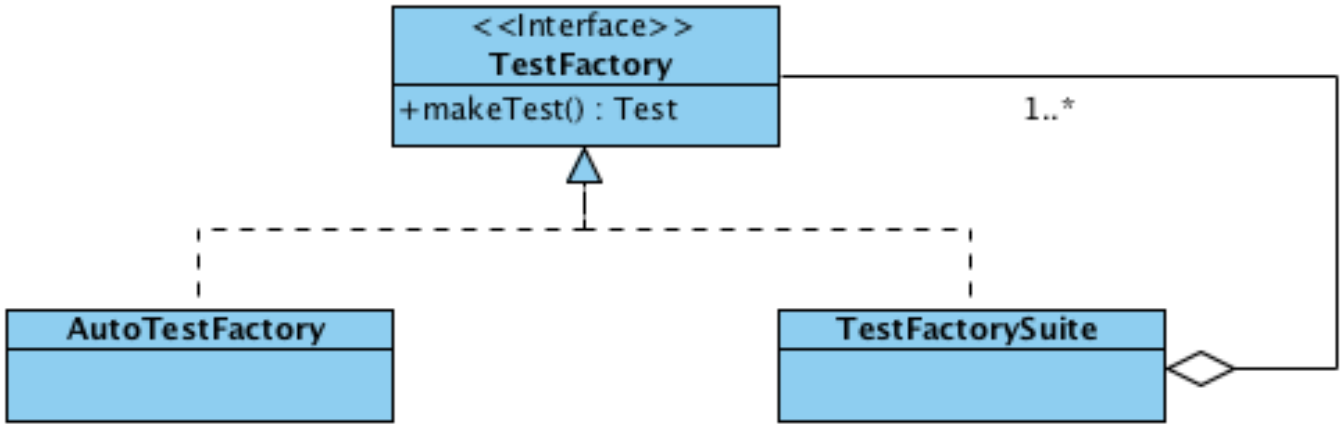
Core Domain


1..*
<<Interface>>
Test
+run(TestResult result)
TestSuite
TestCase

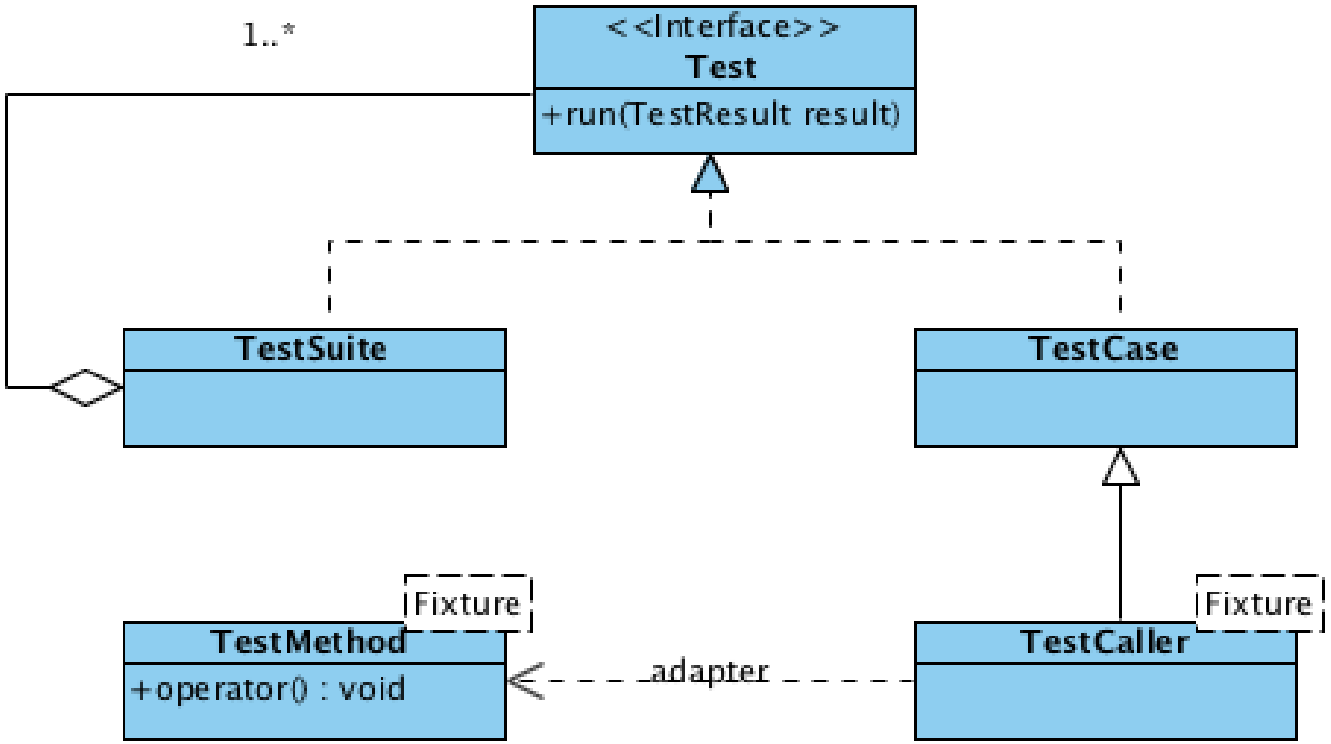
Pretty Output



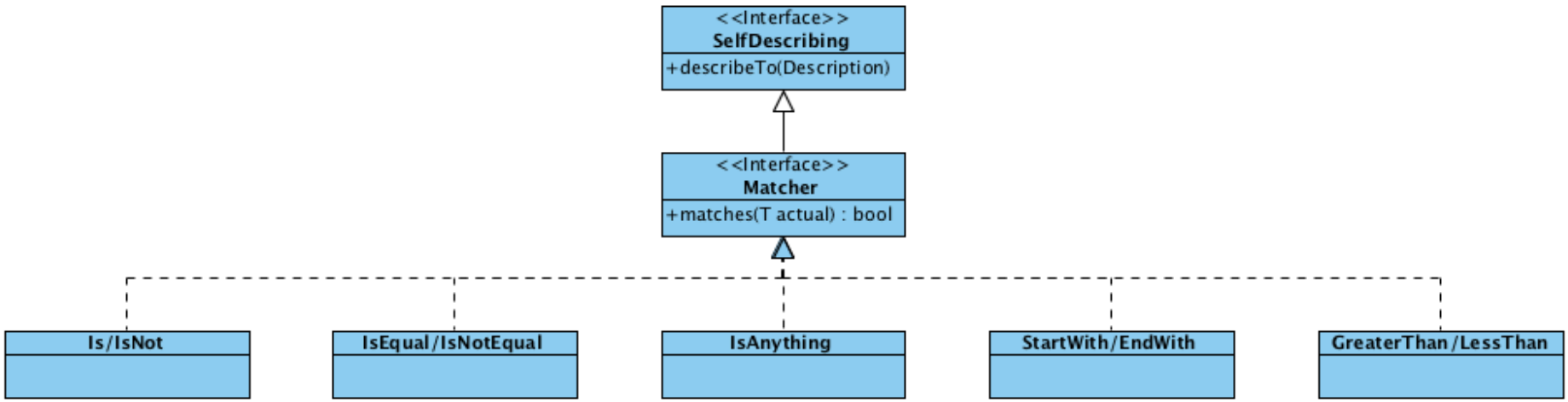
Auto Discovery



Pefect Adapter



Hamcrest


<<Interface>>
SelfDescribing
+describeTo(Description)
<<Interface>>
Matcher
+matches(T actual) : bool
Is/IsNot
IsEqual/IsNotEqual
IsAnything
StartWith/EndWith
GreaterThan/LessThan

Evolution

BDD Style

```
numberbychapter
describe("RobotCleaner")
{
    RobotCleaner robot;

    before()
    { ... }

    after()
    { ... }

    describe("left")
    {
        it("should be faced west after turn left")
        {
            robot.exec(left());
            expect_that(robot.getPosition(), is(Position(0, 0, WEST)));
        }
    }

    describe("right")
    {
        it("should be faced east after turn right")
        {
            robot.exec(right());
            expect_that(robot.getPosition(), is(Position(0, 0, EAST)));
        }
    }
};
```

Notation

```
describe("RobotCleaner")
{
    RobotCleaner robot;

    it("{id=1-left} should be faced west after turn left")
    {
        robot.exec(repeat(left(), 2));
        expect_that(robot.getPosition(), is(Position(0, 0, SOUTH)));
    }

    it("{id=2-left, depend=1-left} should be faced south after turn left with 2 times")
    {
        robot.exec(left());
        expect_that(robot.getPosition(), is(Position(0, 0, SOUTH)));
    }
};
```

Reference

GitHub

- Magellan: https://github.com/horance-liu/magellan
- Reference: http://horance-liu.github.io/magellan/

Thanks