

Mettre en œuvre des tests structurels en C++

Fabrice AMBERT – fabrice.ambert@femto-st.fr
Fabrice BOUQUET – fabrice.bouquet@femto-st.fr
Fabien PEUREUX – fabien.peureux@femto-st.fr
Ivan ENDERLIN, Jean-Marie GAUTHIER
Cédric JOFFROY, Alexandre VERNOTTE

Outils mis en oeuvre



googlemock

Google C++ Mocking Framework



googletest

Google C++ Testing Framework


Cucumber

Anatomie d'un test unitaire

Déclaration d'une classe de test



```
#include <cppunit/extensions/HelperMacros.h>
```

```
class MaClasseDeTest : public CPPUNIT_NS::TestFixture {
```

```
    CPPUNIT_TEST_SUITE(MaClasseDeTest);
```

```
    CPPUNIT_TEST(monTest); // ajout de la méthode dans la suite de tests
```

```
    CPPUNIT_TEST_SUITE_END();
```

```
public:
```

```
    MaClasseDeTest();
```

```
    virtual ~ MaClasseDeTest();
```

```
    void setUp(); // met la classe dans un état particulier avant l'exécution du test monTest()
```

```
    void tearDown(); // met la classe dans un état particulier après l'exécution du test monTest()
```

```
private:
```

```
    void monTest();
```

```
};
```

Anatomie d'un test unitaire

Utilisation d'une classe de test



```
#include "MaClasseDeTest.h"

CPPUNIT_TEST_SUITE_REGISTRATION(MaClasseDeTest);

MaClasseDeTest::MaClasseDeTest() { }

MaClasseDeTest::~~ MaClasseDeTest() { }

void MaClasseDeTest::setUp() {
    // code exécuté avant monTest()
}

void MaClasseDeTest::tearDown() {
    // code exécuté après monTest()
}

void MaClasseDeTest::monTest() {
    // On place la ou les assertions dans cette méthode
}
```

Quelques assertions cppunit...

CPPUNIT_ASSERT(condition) *condition est une expression booléenne*

CPPUNIT_ASSERT_MESSAGE(message, condition) *affiche un message en cas d'erreur*

CPPUNIT_ASSERT_EQUAL(expected, actual) *expected: oracle, actual: valeur réelle*

CPPUNIT_ASSERT_THROW(expression, exceptionType) *test le renvoi d'une exception*

Toutes les assertions ici : http://cppunit.sourceforge.net/doc/cvs/group_assertions.html

Les Mocks en c++: déclaration

Plusieurs librairies de mock disponibles: googlemock, mockpp, mockitopp, etc.

```
#include <gmock/gmock.h>
```

```
class MonMock : public MonObjetAMocker {
```

```
public:
```

```
/*
```

Classe MonMock qui hérite de la classe MonObjetAMocker

MOCK_METHOD0 = mocker une méthode avec 0 paramètre

MOCK_METHOD0(m, type(param)) m = méthode à mocker, type = retour de la méthode à mocker

param = paramètre(s) de la méthode à mocker (0 ou N)

```
*/
```

```
MOCK_METHOD0(maMethodeAMocker, int());
```

```
};
```

Les Mocks en c++ : utilisation



```
void MaClasseDeTest::setUp() {  
    MonMock m = new MonMock();  
    MonObjet o = new MonObjet(m);  
}
```

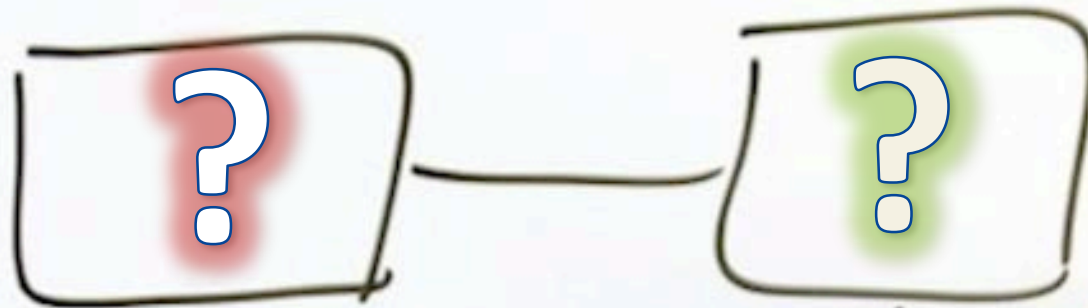
```
void MaClasseDeTest::tearDown() {  
    delete(m);  
}
```

```
void MaClasseDeTest::monTest() {  
    EXPECT_CALL(*m, uneMethode()).WillOnce(Return(unResultat));  
    CPPUNIT_ASSERT(o->maMethode == unResultat);  
}
```

// Ici la méthode maMéthode de l'objet o fait appel à la méthode uneMéthode de l'objet mocké.

Plus d'info ici: <http://code.google.com/p/googlemock/>, <http://code.google.com/p/googlemock/wiki/ForDummies>,
<http://code.google.com/p/googlemock/wiki/CookBook>, <http://code.google.com/p/googlemock/wiki/CheatSheet>

Merci pour votre attention...



"Testing is always model-based!"
Robert Binder

