Unit tests & test paradigms

Dmitry Pinaev
pinaev@in.tum.de

TUM CSE

November 12, 2012

1. Introduction
2. Testing in software development workflow
3. Unit testing
4. Example

What is software tesing?

Questions:

- Indeed, why do we need to test software?
- How?
- What types (approaches) do you know?

Typical workflow in software development

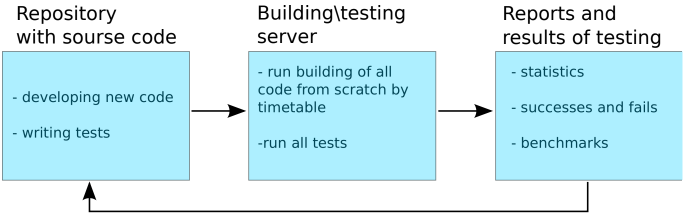
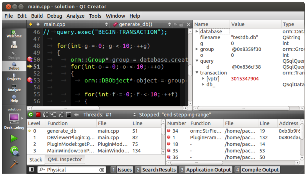


Figure: Stages of software development

Types and approaches in software testing

- **Debugging**



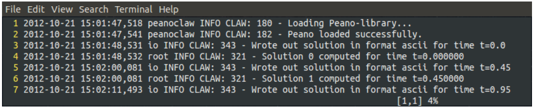
```
//  query.exec("BEGIN TRANSACTION");

    for(int g = 0; g < 10; ++g)
    {
        orm::Group* group = database.creat
        for(int o = 0; o < 10; ++o)
        {
            orm::DBObject* object = group-

            for(int f = 0; f < 10; ++f)
            {
```

Name	Value	Type
▸ database		orm::Data
filename	"testdb.db"	QString
g	0	int
▸ group	@0x8359f30	orm::Gro
o	0	int
▾ query		QSqlQuer
d	@0x836cf38	QSqlQuer
▾ transaction		orm::Tran
▸ [vptr]	3015347904	
▸ db_		QSqlData

Level	Function	File	Line
0	generate_db	main.cpp	51
1	DBViewerPlugin::g...	main.cpp	82
2	PluginModel::getP...	PluginMod...	75
3	MainWindow::onP...	MainWindo...	134

Number	Function	File	Line	Address
34	orm::StrFie...	/home/pac...	19	0xb3b9f6
1	PluginFram...	/home/pac...	132	0x804da
18	-	/home/pac...	14	
35	-	/home/pac...	53	
36	-	/home/pac...	50	

- **Logging** (output some values during the execution of program)



```
2012-10-21 15:01:47,518 peanoclaw INFO CLAW: 180 - Loading Peano-library...
2012-10-21 15:01:47,541 peanoclaw INFO CLAW: 182 - Peano loaded successfully.
2012-10-21 15:01:48,531 io INFO CLAW: 343 - Wrote out solution in format ascii for time t=0.0
2012-10-21 15:01:48,532 root INFO CLAW: 321 - Solution 0 computed for time t=0.000000
2012-10-21 15:02:00,081 io INFO CLAW: 343 - Wrote out solution in format ascii for time t=0.45
2012-10-21 15:02:00,081 root INFO CLAW: 321 - Solution 1 computed for time t=0.450000
2012-10-21 15:02:11,493 io INFO CLAW: 343 - Wrote out solution in format ascii for time t=0.95
                                                                      [1,1] 4%
```

- **Graphical** User Interface testing (just for applications with GUI)
  You click the buttons, enter data to text fields etc. and check the behaviour of program

- **Unit testing** (writing mini programs chiecking some features your software)
  Discussed on next slides

What is unit test? Example

We decided to write super efficient mathematical library.
It computes different trigonometric functions.
To compute sin() we use Tailor series and expand the function around $x_0 = 0.0$ point

$$f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots . \tag{1}$$

$$\sin(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}. \tag{2}$$

```cpp
#include <iostream>
#include <math.h>
// takes x in radians
double mysin(double x)
{
    const int iterations = 5;

    double s = x;
    double result = x;
    for(int i = 1; i < iterations;  ++i)
    {
        s *= - x * x /(2*i+1);
        result += s;
    }

    return result;
}


int main()
{
    const double pi = 3.1415926535;
    std::cout << mysin(0.2) << "␣␣␣␣␣" << sin(0.2) << std::endl;
    std::cout << mysin(pi * 0.2) << "␣␣␣␣␣" << sin(pi * 0.2) << std::endl;
    std::cout << mysin(20.0) << "␣␣␣␣␣" << sin(20.0) << std::endl;
}
```

Using Google unit-test framework

```cpp
#include <iostream>
#include <math.h>

#include "mysin.h"
#include "gtest/gtest.h"


// Declare a test
TEST(TestSuite, testCase1)
{
    EXPECT_EQ(mysin(0.0), 0.0);
}

TEST(TestSuite, testCase2)
{
    EXPECT_DOUBLE_EQ(mysin(M_PI * 0.5), sin(M_PI * 0.5));
}


// Run all the tests that were declared with TEST()
int main(int argc, char **argv)
{
    testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

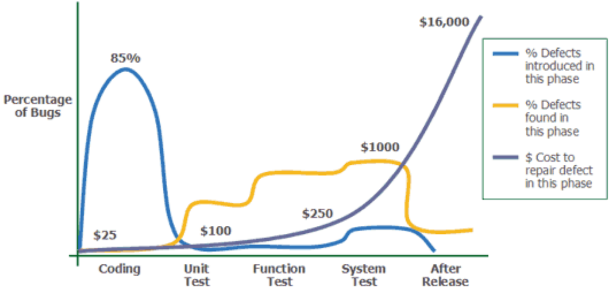
Aspects of unit testing

Seems that we can compare two numbers without Unit-test library.
But:

- Unified approach for tests
- Standard output format
- Building cases with initialisation, finalisation
- Testing not just functions but classes
- Easy to run automatically
- Very useful with rapid development

Aspects of unit testing

- Extreme programming. First test, second implementation
- Unit-test require good code coverage
- Fixing a bug in QA can cost 100 times more than fixing it in development



Useful links

http://www.guru99.com/software-testing.html
http://en.wikipedia.org/wiki/Software_testing
http://en.wikipedia.org/wiki/Test-driven_development
http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks
http://en.wikipedia.org/wiki/Unit_testing
http://www.agitar.com/solutions/why_unit_testing.html