Getting Started Guide of UnitFL

1 Overview

UnitFL is unit testing extension of Visual Studio. It has three main functions.

- NUnit 3.0 Testing Adapter
 UnitFL implements a NUnit 3.0 testing adapter. You can write unit tests with NUnit 3.0
 and run those tests in Test Explorer.
- Coverage Tool
 You can choose to collect coverage information while running unit tests. UnitFL will display them in a tool window.
- Fault Localization Tool
 With coverage information and test results, UnitFL tries to locate the fault that is causing
 the failure. UnitFL displays suspicious program entites in a tool window. You can click
 those entites to navigate to the corresponding source code.

2 UnitFL as A Testing Adapter

UnitFL currently only works with NUnit 3.0. Currently NUnit 3.0 is under prerelease, so to install it you need to include prerelease when you search NUnit in the NuGet package manager (see Fig 1). After building the assembly that contains NUnit tests, UnitFL will display them in the Test Explorer (see Fig 2). You can run the tests in the Explorer.

3 UnitFL as A Coverage Tool

To collect coverage, you need to open the UnitFL result window from View -> Other Windows -> UnitFL Result (see Fig 3). You need to toggle a button to enable profiling (see Fig 4). Now UnitFL will collect coverage information for tests. Collecting coverage costs a significant

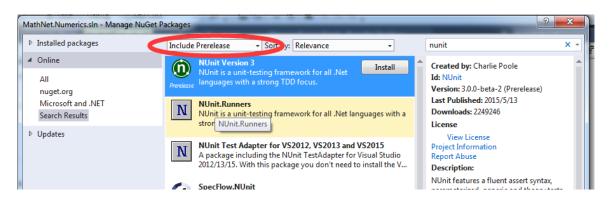


Figure 1: Downloading Nunit 3.0

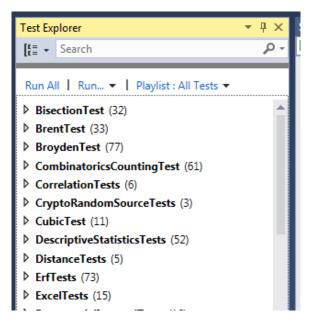


Figure 2: Test Explorer

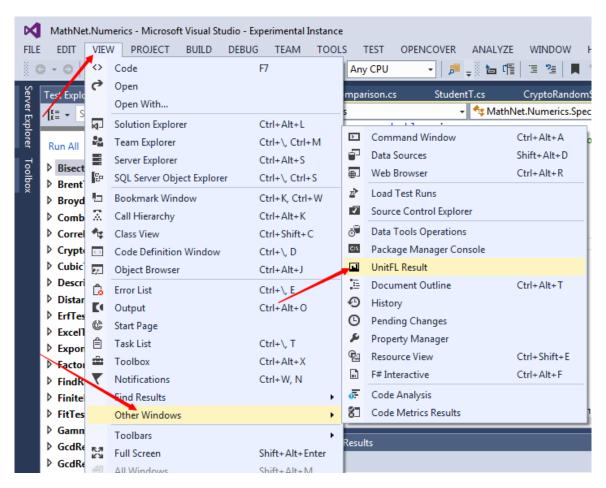


Figure 3: Open Result Window

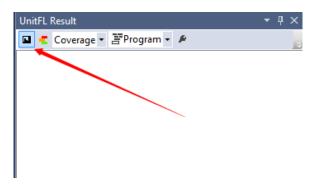


Figure 4: Enable Profiling

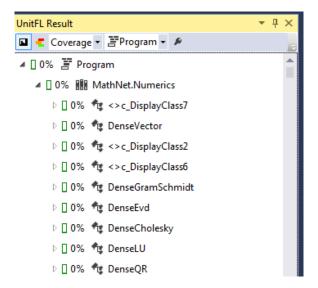


Figure 5: Coverage Result

overhead, so you should toggle the button again to disable profiling when you don't need coverage.

After running tests, coverage information will be displayed in the result window (see Fig 5). In the result window, you can click on a class or a method to navigate to the corresponding source code.

You can toggle a button to enable the coloration of the source code (see Fig 6). Uncovered lines will be marked red and covered lines will be marked green.

4 UnitFL as A Fault Localization Tool

UnitFL can use test reuslts and coverage information to locate faults. To locate fault, you need to enable profiling first and change the combobox from Coverage to Fault Localization (see Fig 7).

After some of the tests failed, UnitFL will calculate the suspiciousness of every program entity and rank them in the result window (see Fig 8). There are five levels of suspiciousness and they are colored from red to green. Again you can navigate to the source code by click an item and you can toggle the coloration button to enable coloration of source code.

```
StudentT.cs
                       CryptoRandomSource.cs
                                                ExponentialIntegral.cs + X
                                                                              UnitFL Result
            🕶 🔩 MathNet.Numerics.SpecialFunctior 🕶 🚳 ExponentialIntegral(double x, int n) 🤊
                                                                              🗖 🗲 Coverage 🕶 👺 Program 🕶 👂
int i, ii;
                                                                                0% 躇 Program
double ndbl = (double)n;
double result;
                                                                                 ▲ [] 0% IIII MathNet.Numerics
double nearDoubleMin = 1e-100; //needs a very small value that is not quite
                                                                                    D 0% 🥞 <>c_DisplayClass7
double factorial = 1.0d;
double del;
                                                                                    D 0% 🥞 <>c_DisplayClass2
double psi;
double a, b, c, d, h; //variables for continued fraction
                                                                                    D 0% 🔩 <>c_DisplayClass6
                                                                                    DenseGramSchmidt
//special cases
                                                                                    DenseEvd
if (n == 0)
{
                                                                                    DenseCholesky
    return Math.Exp(-1.0d*x)/x;
                                                                                    DenseLU
else if (x == 0.0d)
                                                                                    DenseQR
    return 1.0d/(ndbl - 1.0d);
                                                                                    DenseSvd
                                                                                    Daniel Der Der Gram Schmidt
//general cases
//continued fraction for large x
                                                                                    D 0% 🔩 UserCholesky
if (x > 1.0d)
                                                                                    D 0%  UnivariateSliceSampler
     - v + ((double\n\.
                                                                                    DisplayClass4
                                                                                    ⊳ Π 0% 🔩 UserEvd
```

Figure 6: Enable Coloration

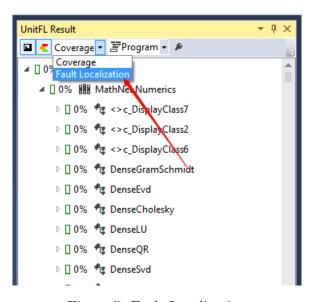


Figure 7: Fault Localization

```
ExponentialIntegral.cs + ×
              🕶 🔩 MathNet.Numerics.SpecialFunctior 🕶 🕸 ExponentialIntegral(double x, int n) 🕶
                                                                                       💶 🗲 Fault Localization 🕶 躇 Program 🕶 🔑
int i, ii;
                                                                                       🗸 F 👺 Program
double ndbl = (double)n;
double result;
                                                                                               III MathNet.Numerics
double nearDoubleMin = 1e-100; //needs a very small value that is not quite
                                                                                               F SpecialFunctions
double factorial = 1.0d;
double del;
                                                                                                   Precision
double a, b, c, d, h; //variables for continued fraction
                                                                                                   <>c_DisplayClass7
                                                                                                   <>c_DisplayClass2
                                                                                                   🕏 <> c_DisplayClass6
if (n == 0)
                                                                                                   t DenseGramSchmidt
    return Math.Exp(-1.0d*x)/x;
                                                                                                   DenseEvd
else if (x == 0.0d)
                                                                                                   t DenseCholesky
    return 1.0d/(ndbl - 1.0d);
                                                                                                   DenseLU
                                                                                                   DenseQR
//general cases
//continued fraction for large x
                                                                                                   DenseSvd
                                                                                                   tuserGramSchmidt
```

Figure 8: Fault Localization Result

5 Filters

You can avoid the profiling some of your assemblies or classes using filters. Filters in UnitFL are very similar to those in OpenCover. Here are some examples:

- +[*]*, -[*Tests]* Exclude those assemblies that end with Tests.
- +[*Sample]*
 Only include those assemblies that end with Sample.
- +[*]*, -[*]*Test Exclude those classes that end with Test.

6 Contact Us

If you have anything to say, please send a email to wangnangg@gmail.com.