

F# for Testing And Analysis

at the Community For F#

March 16th 2010

Richard Minerich (F# MVP)

RichardMinerich.com

twitter.com/rickasaurus





- Time and Place -

The first Monday of every month at Microsoft NERD

- User Group Leads -

Talbott Crowell, Michael de la Maza, Rick Minerich

For more information (and video) visit us @ fsug.org

"A programming language is low level when its programs require attention to the irrelevant."

-Alan J. Perlis

What Stops You From Using COP/TDD?

Time

Rigidity

Money

Existing Code

Prototyping

Infrastructure

Complexity

Testing in the Small

Why are idiomatic C#/VB classes difficult to test?

They have...

- Hidden State
- Unknown Side Effects
- Rigid Structure
- Implicit Event Ordering
- Have poorly constrained return values

Why are idiomatic F# functions easy to test?

They have...

- Only one output for one set of outputs
- No side effects
- Flexible Structure
- Continuations instead of events
- The restriction of not returning null and need not throw exceptions

Why do C#/VB classes make for poor testing Infrastructure?

They have...

- Bulky syntax and class overhead
- Stateful but reused resources
- Rigid Structure
- No dynamic object construction
- Dynamic function construction only in terms of an existing delegate
- Implicit type conversion

Why do F# functions make for great testing infrastructure?

They have...

- Very concise syntax and no class overhead
- The tools to avoid reusing stateful resources
- Flexible structure
- Dynamic, anonymous object instantiation on interfaces
- Fully dynamic function construction
- No implicit type conversion

(In my humble opinion)

The winner is clear.

TestDriven.NET and NCover

The screenshot displays the Visual Studio IDE with the following components:

- Code Editor:** Contains F# code for a test titled "Integer Arithmetic". The code defines a function `SampleArithmetic1` and prints the results of arithmetic operations: `x = 19, y = 39, r1 = 6, r2 = 1`.
- Output Window:** Shows the test execution results: "Test started: Assembly: Samples101.exe", followed by the output values and a summary: "1 passed, 0 failed, 0 skipped, took 0.16 seconds."
- Context Menu:** A right-click menu is open over the code, with "Run Test(s)" selected. Other options include "Build", "Test With", "Repeat Test Run", "Send To F# Interactive", "Go To Definition", "Breakpoint", "Run To Cursor", "Go To Reflector", "Cut", "Copy", and "Paste".
- NCover Coverage View:** A tree view on the right shows coverage percentages for various assemblies and classes. The tree includes: "e0.xml (100%)", "sLibrary1 (100%)", "Examples (100%)", "CoverageTests (100%)", "TestCoverageExclude (100%)", "Excluded", "Excluded", and "ExcludedClass".

xUnit.NET

Open Source, No Classes Needed, F# Friendly

```
[<Fact>]
let IsPrime_WithPrimeNumber_ShouldReturnTrue() =
    Assert.True(isPrimeNumber 7)

[<Fact>]
let IsPrime_WithNonPrimeNumber_ShouldReturnFalse() =
    Assert.False(isPrimeNumber 21)
    Assert.False(isPrimeNumber 45)
```

FsCheck

A Framework For Random Property Based Testing

FsCheck

High Reliability Environments

- Automated Trading Systems
- Communication Technologies
- Power Delivery
- Medical Technologies

FsCheck

Missed Edge Cases

- Domain edges are easy to test
- Unexpected edges are always a problem

Reflection Sample

Office Automation Sample

Visualization Sample

Check out our F# User Group @ <http://fsug.org>

Our Next Meeting is **Monday, April 5th**



Thank you for your time.