Why is Microsoft investing in Functional Programming?

Don Syme

With thanks to Leon Bambrick, Chris Smith and the puppies

All opinions are those of the author and not necessarily those of Microsoft
Simplicity
Economics
Fun
What Investments?

• C#
  – C# 2.0 (generics)
  – C# 3.0 (Language Integrated Queries - LINQ)
  – These represent a major industry shift towards functional programming

• F#
  – Bringing F# to product quality

• Haskell
  – Strongly supporting Haskell research

• VB, Python, Ruby
  – These incorporate many functional features and overlap with the functional programming ethos
Who?

- Microsoft Research ("MSR")
  - F#
  - Haskell

- Microsoft Developer Division ("DevDiv"), Visual Studio Languages Group
  - C#  
  - Visual Basic
  - F#
  - Python
  - Ruby
F#: Influences

OCaml

F#

C#/.NET

Similar core language

Similar object model
Simplicity
//F#
open System
let a = 2
Console.WriteLine a

//C#
using System;

namespace ConsoleApplication1
{
    class Program
    {
        static int a()
        {
            return 2;
        }
        static void Main(string[] args)
        {
            Console.WriteLine(a);
        }
    }
}
More Code!

```fsharp
// F#
open System
let a = 2
Console.WriteLine a
```

```csharp
using System;

namespace ConsoleApplication1
{
    class Program
    {
        static int a;
        {
            return 2;
        }
        static void Main(string[] args)
        {
            Console.WriteLine(a);
        }
    }
}
```

More Noise Than Signal!
// Use first-order functions as commands
let BreakCommand = Command(fun rover -> rover.Accelerate(-1.0))
let TurnLeftCommand = Command(fun rover -> rover.Rotate(-5.0<degrees>))
Pleasure

let rotate(x,y,z) = (z,x,y)

let reduce f (x,y,z) = f x + f y + f z

Pain

Tuple<V,T,U> Rotate(Tuple<T,U,V> t)
{
    return new Tuple<V,T,U>(t.Item3,t.Item1,t.Item2);
}

int Reduce(Func<T,int> f,Tuple<T,T,T> t)
{
    return f(t.Item1) + f(t.Item2) + f(t.Item3);
}
Friendly, Approachable.

Orthogonal & Unified Constructs

• Let “let” simplify your life...

let data = (1,2,3)

let f(a,b,c) =
    let sum = a + b + c
    let g(x) = sum + x*x
    g(a), g(b), g(c)

Bind a static value
Bind a static function
Bind a local value
Bind a local function

Type inference. The safety of C# with the succinctness of a scripting language
Simplicity

Processing 200 images in parallel
Microsoft is investing in functional programming because....

*It enables simple, compositional and elegant problem solving in data-rich, control-rich and symbolic domains*
Case Study

Ad Ranking,
MSR Cambridge Online Services and Advertising Group
The adCenter Problem

• Selling "web space" at www.live.com and www.msn.com
• "Paid Search" (prices by auctions)
• The internal competition focuses on Paid Search.
OSA Machine Learning

• Internal Competition

• Use F# for major adCenter and Xbox Live projects
  – 4 week project, 4 machine learning experts
  – 100 million probabilistic variables
  – Processes 6TB of training data
  – Real time processing

“F# was absolutely integral to our success”

“We delivered a robust, high-performance solution on-time.”

“We couldn’t have achieved this with any other tool given the constraints of the task”

“F# programming is fun – I feel like I learn more about programming every day”
Observations
- Quick Coding
- Agile Coding
- Scripting
- Performance
- Memory-Faithful
- Succinct
- Symbolic
- .NET Integration

F#’s type inference means less typing, more thinking
Type-inferred functional/ OO code is easily factored and re-used
Interactive “hands-on” exploration of algorithms and data over smaller data sets. Used in combination with Excel
Immediate scaling to massive data sets

Live in the domain.
Schema compilation and efficient “Schedule” representations key
Especially Excel, SQL Server
The Team’s Summary

– “F# was absolutely integral to our success”

– “We delivered a robust, high-performance solution on-time.”

– “We couldn’t have achieved this with any other tool given the constraints of the task”

– “F# programming is fun – I feel like I learn more about programming every day”
Some Code Highlights

• Type-safe Schema Bulk Import

BulkImporter<'Schema>:
    database:string * prefix:string -> BulkImport<'Schema>

- Written as part of the team’s toolchain
- Schema in F# types
- Compiled using F# “schema compilation” techniques
- 800 lines

- Enabled team to clean and insert entire data set over 3 day period
Some Code Highlights

```csharp
/// Create the SQL schema
let schema = BulkImporter<PageView> ("cpidssdm18", "Cambridge", "June10")

/// Try to open the CSV file and read it pageview by pageview
File.OpenTextReader "HourlyRelevanceFeed.csv"
|> Seq.map (fun s -> s.Split [|',',|])
|> Seq.chunkBy (fun xs -> xs.[0])
|> Seq.iteri (fun i (rguid,xss) ->
    /// Write the current in-memory bulk to the Sql database
    if i % 10000 = 0 then
        schema.Flush ()
    
    /// Get the strongly typed object from the list of CSV file lines
    let pageView = PageView.Parse xss

    /// Insert it
    pageView |> schema.Insert
)
/// One final flush
schema.Flush ()
```
Some Code Highlights

```ocaml
/// A schedule of computation in a factor graph
/// (i.e., a series of update functions
/// and variables that should be updated)
type Schedule =
  | ScheduleStep of (IFactor * int)
  | ScheduleSeq of Schedule[]
  | ScheduleLoop of Schedule * float

/// Runs a schedule
member schedule.Run() =
  match schedule with
  | ScheduleStep (fac,idx) -> fac.UpdateMessage idx
  | ScheduleSeq sseq -> sseq |> Seq.maxOn (fun s -> s.Run())
  | ScheduleLoop (s,maxDelta) ->
    let delta = s.Run()
    if delta > maxDelta then schedule.Run() else delta
```

Expressing and evaluating “Approximation Schedules” was crucial to this work.

Functional programming made this beautiful.
(Aside: Units Of Measure)

type acceleration = float<\text{m} / \text{s}^2>

let fast = 3.0e6<\text{m/s}>
let gravity = -9.81<\text{m/s}^2>)

The F# September CTP includes “units of measure” Inference and checking

/// Computes the absolute difference between two Gaussians
let AbsoluteDifference (a:Gaussian<'u>,b:Gaussian<'u>) =
    max (abs(a.PrecisionMean - b.PrecisionMean))
    (sqrt(abs(a.Precision - b.Precision))))
Re-Ranking the History of Chess

Search for “TrueSkill Through Time” (MSR Cambridge Online Services and Advertising Group)
Control Rich

Async.Run

(Async.Parallel
  [ Async.GetHttp "www.google.com";
    Async.GetHttp "www.live.com";
    Async.GetHttp "www.yahoo.com"; ]
)
Why learn F#?

Moore’s Law, but no speed increase
Parallelism

• The Economics of the Hardware Industry are Changing

• Functional programming is a crucial tool in parallel and asynchronous programming
  – For architecture
  – For implementation

• Good synergies, e.g. with Parallel Extensions for .NET
Economics
Economies of Scale at Microsoft

• Have .NET
• Have .NET Libraries
• Have Visual Studio, Silverlight, .NET CF, ASP.NET, XNA GameStudio, RoboticsStudio
• Have Tools (profilers, debuggers, designers)

• Given this basis, the opportunities for low-cost, value-add investments are enormous:
  – Dynamic Languages
  – Functional Languages
  – Web programming (client, server, services)
  – Business programming
  – Parallel programming
  – Game programming
  – Data mining programming

• Cost: low, Value: high
Economics for Users

• Learn .NET
• Can use the tools right for the job
• Can reuse much knowledge from tool to tool
Economics

Microsoft is investing in functional programming because....

*It is a sensible, relatively low-cost investment that adds real value to Visual Studio and the .NET Framework*
Fun
This is fun

```light
#light

open System

Console.Beep

Console.Beep(int frequency, int duration) : unit
Console.Beep() : unit
Plays the sound of a beep of a specified frequency and duration through the console speaker.

Exceptions:
- System.ArgumentOutOfRangeException
- System.Security.HostProtectionException
```
This is fun

```csharp
#light

open System
open System.IO

let ratedMovies = Directory.GetFiles @"NetFlixPrizeData\"

ratedMovies
|> Array.map processRatingsFile
|> Seq.concat
|> Seq.iter combinedLog.WriteLine
```
This is not fun
This is fun
Async.Run

(Async.Parallel

[ GetWebPage "http://www.google.com";
  GetWebPage "http://www.live.com";
  GetWebPage "http://www.yahoo.com";
]

Async.Run

(Async.Parallel

[ for i in 1 .. numImages -> ProcessImage(i) ]
)
This is fun too!

#r "Microsoft.ManagedDirectX.dll"

#r "System.Parallel.dll"

#r "System.Xml.dll"

#r "Xceed.Charting.dll"

#r "NUnit.Framework.dll"

#r "ExtremeOptimization.Math.dll"
Community fun

19 February 2008

Why I Love F#: Pattern Matching

Greetings fellow F#-philes! Today we're definitely completely infatuated with the F# language. Pattern matching is a simple idea. Essentially, a set of rules for comparing a value and a set of possible matches. It's the fastest genome assembly viewer I've ever seen and only 500 lines of F#. It's really an incredible language...

Subject: Microsoft F#

Hi,

Microsoft Research have a functional language F# that is built on the .NET environment. It's pretty powerful but is not yet (as far as I know) part of the Microsoft pipeline of products. We'd like to raise this with them at a senior level, to register our interest in them providing this as a supported product at some point in the future. Do you know who/how we can best raise this?

Subject: Thank You Don

Don,

I am excited by F# and anticipate many years of exploration and pragmatic productivity.

I enjoy the surprises that come from working with F# when unfamiliarity melts away to reveal the patterns of underlying consistency.

I appreciate the language, it's documentation and your support for the community.

Why I Love F#: The Interactive Environment

I'm starting a brand new series of short articles about F#. The plan is to describe features that, for me, make F# a compelling and enjoyable .NET language. So far, I have 10-15 articles in mind, but I'm open to suggestions. If you have any ideas for...
A Fantastic Team

- Developers
- QA
- Research/Architecture
- Program Managers
- Oversight

- Joe, Santosh, James, Baofa, Sean, Luca, Tim, Mike, Matteo

The decision to bring F# to product quality was made and informed by a collective process involving:

- Vice Presidents, Research leaders, Architects, Technical fellows, CTOs, Product Unit Managers, Developers, Testers, Researchers...
Fun

Microsoft is investing in functional programming because....

People want it
People like it
People are (in certain important domains) more productive with it
Summary

• Functional Programming Brings Simplicity

• Functional Programming with .NET makes Business Sense

• And it’s fun!