# Lessons Learned Adopting Scala

Where we spend a year working in Scala
and still can't make up our minds

some things are fundamental,
some will change

JULY 2012

# Who?

- Richard Care
- Duncan McGregor

# What?
# The Uplink Management System

- A Java, Wicket, Hibernate, Spring webapp to schedule transmission of digital films to cinemas via satellite
- EU funded project, prototype into full app
- Algorithmically interesting
- Eclipse and TDD
- Maven and Git
- 1 man year of code when we started Scala

Now 38kLOC
25% Scala
35% test code

# What is Scala?

- An statically-typed OO/Functional hybrid language running on the JVM
- A hell of a compiler
- An impressive runtime
- Still evolving

Other platforms are in development

# Why Did We Migrate?

- Because we could
- Exploration

So you don't have to!

# Why Might You Migrate?

Assuming that you are starting from Java

Of course there are lots of reasons not to migrate - and lots of those will come up!

- Productivity
  - FP benefits in algorithm design
  - Immutable throughput for multi-core
  - Expressiveness
  - The Python Paradox

"if a company chooses to write its software in a comparatively esoteric language, they'll be able to hire better programmers, because they'll attract only those who cared enough to learn it." Paul Graham

# Why Might You Migrate?

- Fun
- Learning
- CVtastic

# How Did We Migrate?

- The same way you would...
  - Get Scala building
  - New classes in Scala
  - Migrate existing classes case-by-case

# Building

- Mix Java and Scala source in the same tree
- maven-scala-plugin deals with compilation
- Scala compiler can parse .java files, so circular dependencies OK
- Scala-IDE aka Eclipse Scala plug-in
- maven-eclipse-plugin can be configured to generate projects

# Other Build Tools are Available

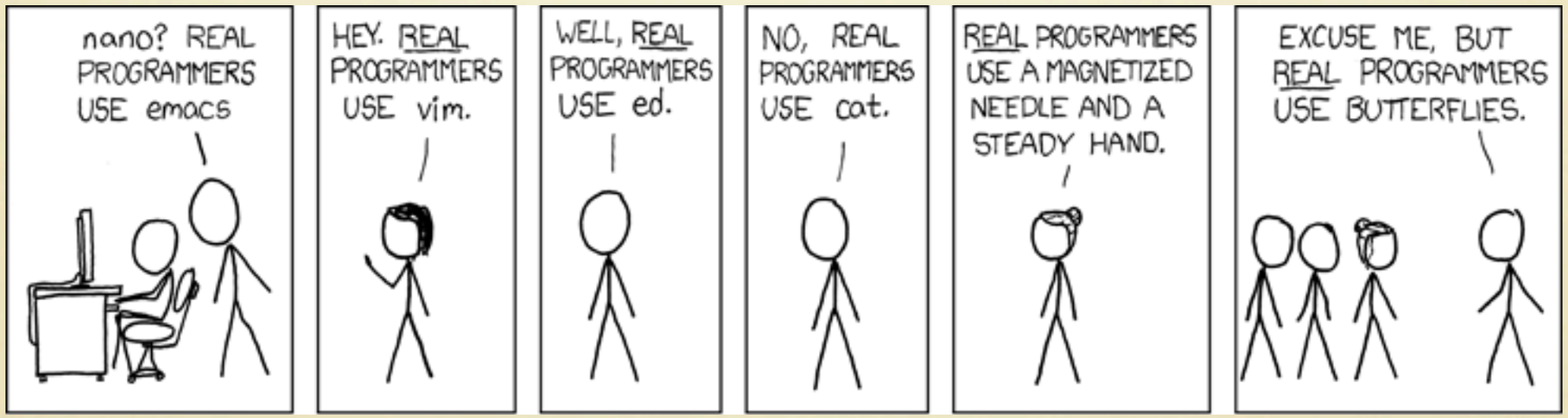
- Maven is slow for Scala, SBT faster
- Eclipse, ah Eclipse

The cool kids all use SBT, but changing a working build never made it high enough on our list of priorities

# Eclipse

- Eclipse support isn't great, and won't be this year
- Type and call hierarchy missing
- Virtually no refactoring

Pretty much just a syntax highlighting editor with click through to types and methods and build errors.

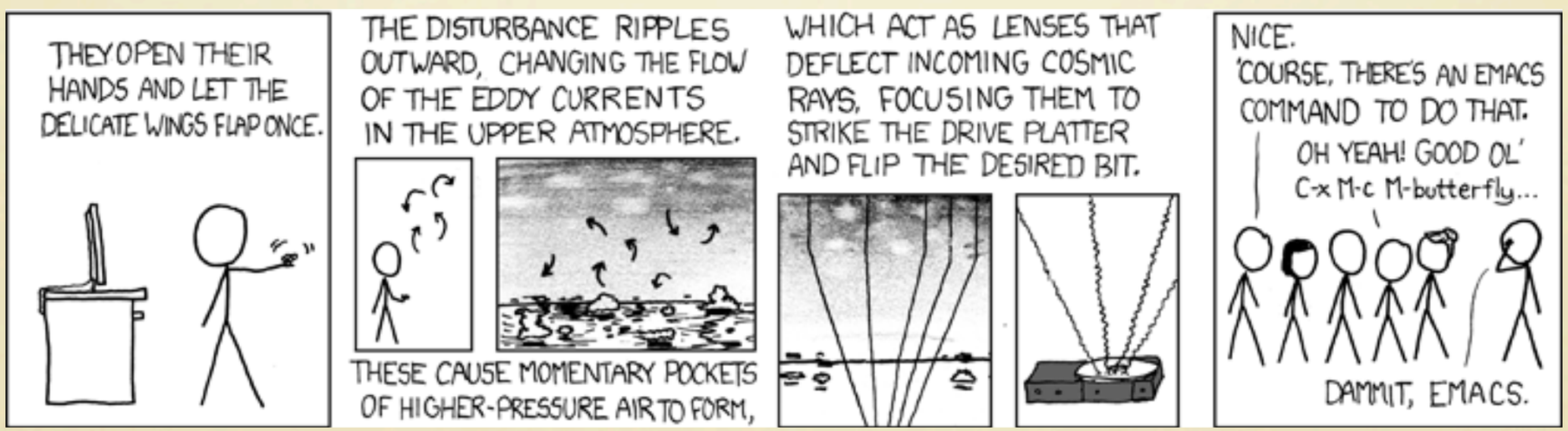


# Ah Eclipse

- Clean to see some breaking changes
- Compile failure breaks build
- It is good enough
- IDEA is apparently better

Think JBuilder in 2000

best of the IDEs by all accounts, but letting go of our 3rd point of contact with the rock didn't seem like a good, erm, idea



# New Classes

but first, new tests

# ScalaTest

- Test framework embracing many styles

In particular JMock etc

- Recommended as a good route to trialling Scala, but this carries its own risks
- Scala's good support for literal lists, sets, maps, strings and xml is a greater advantage in test than production code
- We can still use Java to test Scala, and vice versa

# New Classes

- With a few exceptions interop really is seamless
  - call Java methods
  - implement Java interfaces
  - extend Java classes
- OO or functional style - Java or Scala collections
- REPL to trial use of APIs

Generics is the headache - Scala will almost certainly improve your understanding of Java

# Migrating Existing Classes

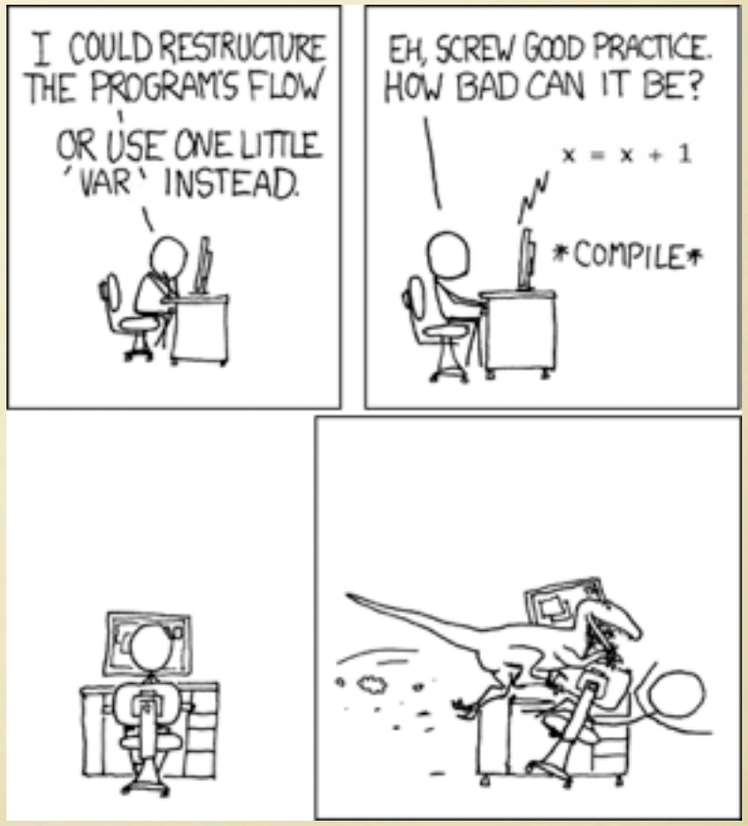
- Leave algorithm and types alone - move to Scala syntax

Regex help

- Move to Scala collections, accessors, immutability

StackOverflow
Based Development

- These also applies to learning Scala



# So Far So Good

- At this point you just have a better Java
  - More consistent
    everything appears to be an object
  - Default immutability
  - Better collections
  - Less noisy
    semi-colons, fields as constructor arguments, type inference
  - More bang per line of code
    closures, list comprehensions, pattern matching
  - Fast
    Our test runs often faster for migrated code

# Except


- The build is much slower
- Tool support is worse
- Debugging is harder
- Error messages are sometimes cryptic
- Scala runtime source is opaque
- Scala is evolving

Lots of synthetic variables and methods, Step Into practically useless

Java compiler is much better at diagnosing the root error

CanBuildFrom, implicits

Idiomatic Scala is a moving target

Once we have migrated our Java idioms

# The Next Step

- Scala tools for making code more expressive
  - Mixin traits
  - Pattern matching
  - Implicit conversions / parameters
  - Operator overloading
  - AST Macros

    Coming soon!
  - Compiler plugins

# Trip Hazard

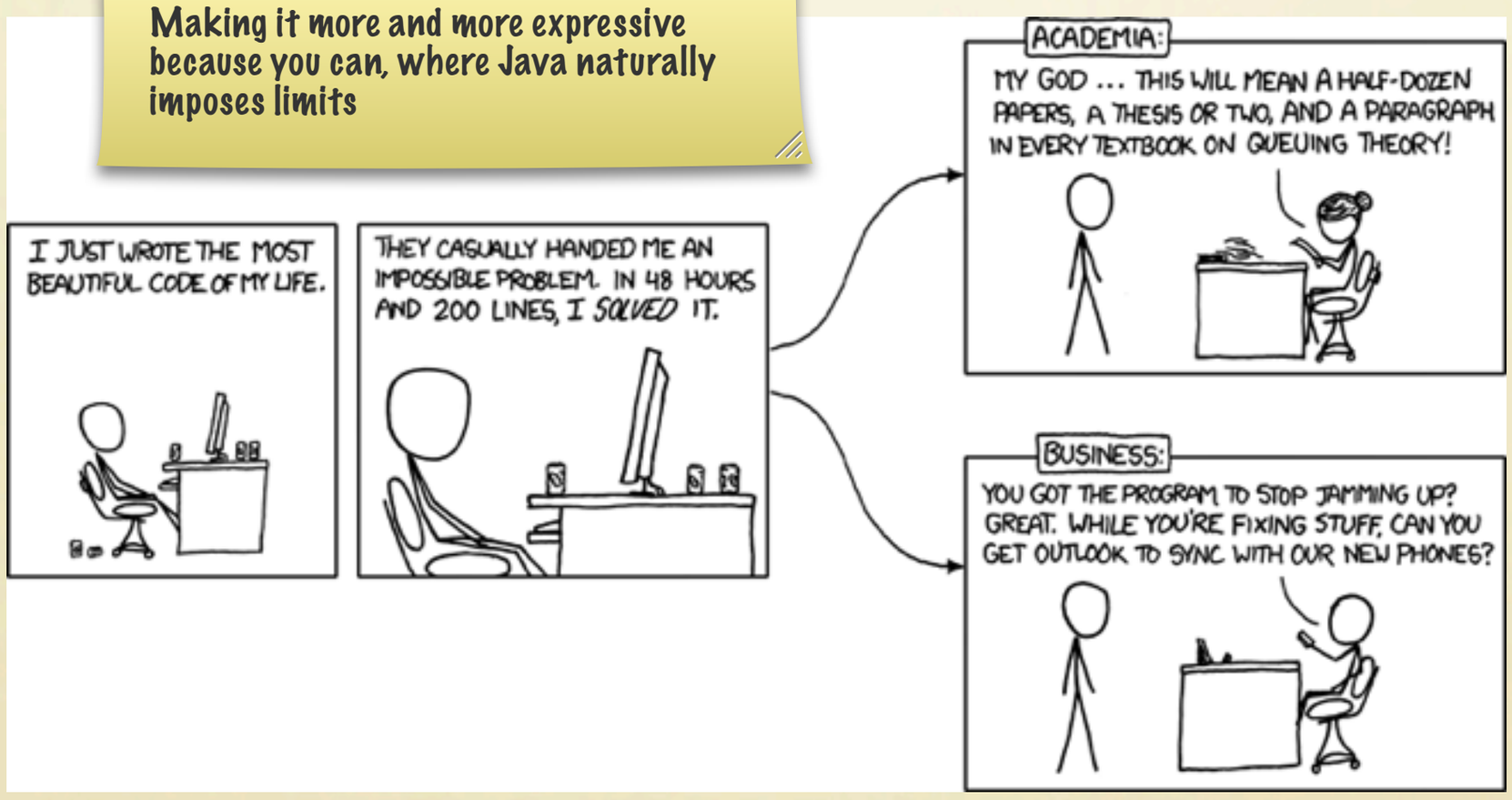
Worse still for macros and compiler plugins

- Implicits make it practically impossible to predict what code will be executed with what arguments
- “Running one’s brain like a compiler”
- “First language I’ve used that I felt I had to be a computer scientist”

# Rabbit Holes

- Increasing expressiveness drags you in

Making it more and more expressive because you can, where Java naturally imposes limits



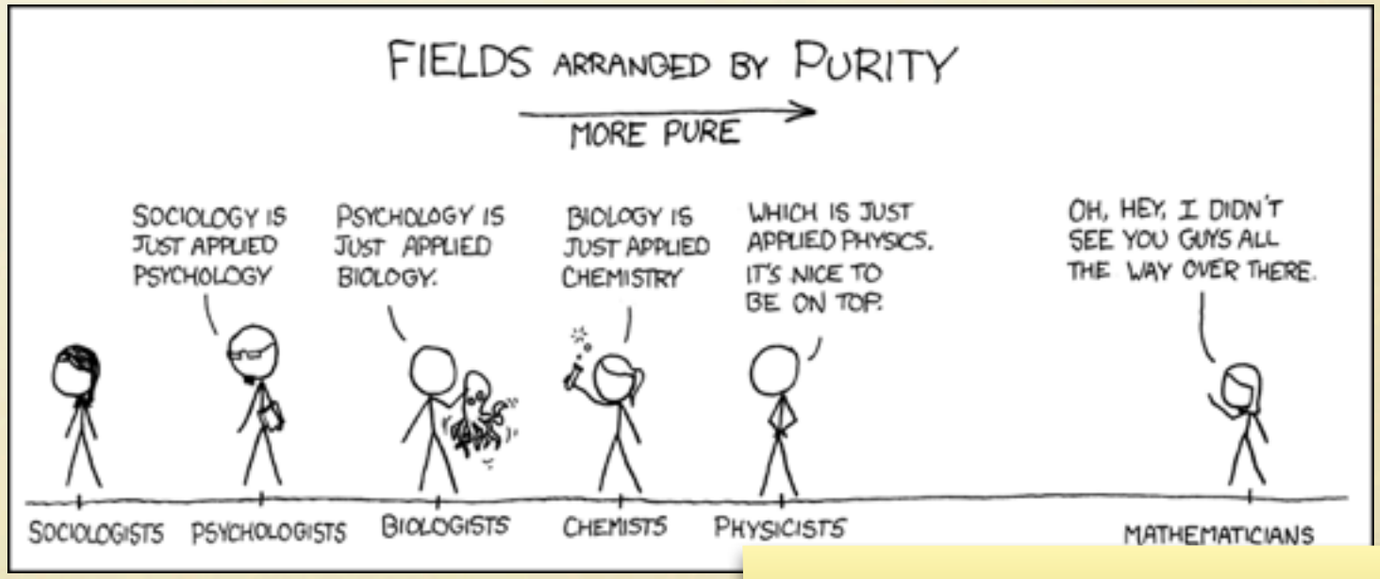
# Functional Style

- Recursive definition
- Lazy evaluation
- Monoids and Monads
- Continuations
- Type classes

We've only dabbled, but mention these here so as not to forget that they are available when useful.

A problem is that many of the Scalarati see the world in these terms rather than OO

"All discussions on the Scala mailing list will eventually descend into Category Theory"




FP programmer is 2m to the right --->

# If it Doesn't Work Out

- Java is still available
- Interop is so good that you could deprecate Scala and return to Java leaving Scala impl alone
- Walled gardens of Scala
- Informed our Java style - Guava etc

# Summary

Like Coplien's purple book C++

- Simultaneously impressed and horrified
- Addictive
- I wouldn't hesitate to take a Scala contract

# Summary

We may be saying that programmers should adopt Scala, but projects should not?

- Starting a Scala project is less clear-cut
- You can live without cutting edge IDE support (?)
- Suitable for a bleeding edge team

or attracting a cutting edge team

- Is something simpler trying to get out?


OH, HI; I'M HERE FROM THE INTERNET.
WHAT ARE YOU DOING!?
GLUING CAPTIONS TO YOUR CATS.
PRRR