Usability

Stephen Colebourne
Member of technical staff
OpenGamma Ltd

The great frontier

- Lots of new APIs
- Lots of new languages
- But not all are very *usable*
- Why is that?

Usability affects everyone!

Everyone

- Lets assume we're all clever
- Lets assume we're all highly capable
- If so, then we can learn and deal with complexity
- But it takes time and effort
- Wouldn't simple and usable be even better?

Git

- I don't especially like Git
- Consensus that it is generally good technically
- Consensus that it is generally hard to use
- For me, the (lack of) usability is more important than the technical ability
- At OpenGamma, we pay for SmartGit GUI
  - much better usability (though not perfect)

Why is Git hard?

- Names of commands
- Error messages
- Complex underlying model

Mental models

- Mental models are how we view the world
- Once we've built one its hard to change
- Relate new concepts to what we know already
- eg. version control (CVS/SVN)
  - work locally
  - when ready, commit to shared repo
  - then everyone can see the work
- Git isn't really like this
  - it has a much larger set of concepts

Git makes me feel stupid!

Coping strategies

- We sometimes develop strategies to cope
- Allow us to work with problem system
- eg. for Git
  - "always commit before pulling"
  - "use a pretty GUI"
  - "treat it like SVN with extra steps"
- But never get to understand Git's model
  - non-standard tasks become major problem
  - eg. merging two forks

Coping strategies

- What if you didn't understand anything deeply?
- What if each task was a coping strategy?
- More common than you might think?

Generics

- Explain what this means:

```
public class Enum<E extends Enum<E>>
 implements Comparable<E> {
}
```

Generics

- Explain what this means:

```
public static
 <T> void copy(List<? super T> dest,
               List<? extends T> src) {
}
```

Generics

- Explain what this means:

```
public static
 <T extends Object & Comparable<? super T>>
    T min(Collection<? extends T> coll) {
}
```

Why is generics hard?

- No wildcards in Generics until last minute
- Implementation proven too complex
- Life without wildcards is simpler

```
Number[] nums = new Number[3];
Object[] objs = nums;
Integer[] ints = nums;
// get runtime error if necessary
```

Who understands generics?

- How many people on the entire planet understand them fully?
- Very, very few
- I find that concerning, do you?

- The rest of us use coping strategies
- Mine is trial and error

Generics makes me feel stupid!

So how did generics happen?

- The experts that designed them were clever
- They understood the problem
- They could discuss and trade off the issues
- But who asked the awkward question
  - "is it usable by people outside this room"?
- If it was asked, they got the wrong answer

Its not that developers are stupid!

- No programmer is stupid
- We've all got skills and ability
- But we are frequently lazy
- We only learn what we need to
- Given a wildcard/variance problem, it is possible to explain it to a developer, and they will understand
- But they will typically forget within a few hours as its not interesting to them, or its not relevant to their main assigned task

Scala

- What does this code do?

```
def foo[A](list : List[A]): List[A] =
 list.foldLeft(List[A]())((r,c) => c :: r)
```

Scala

- What does this code do?

```
def foo(val : Int): Int = {
  val *= 30
  val = (val / 2) + 7
   - (6 * 2)
}
```

Scala

```
def bar[A](list : List[A]): List[(A,int)] =
 list.foldLeft(List[(A,int)]()) {(r,c) =>
  r match {
   case (v,t) :: tail =>
    if (v == c) (c,t+1) :: tail
    else (c,1) :: r
   case Nil =>
    (c,1) :: r
  }
}.reverse
```

Scala

- I don't like Scala
- Its a big missed opportunity
- Good technical stuff
- Consistent reports of being hard to use
- Also hard to write an IDE for

Scala

- Collection library very complex
- Method signatures so complex that documentation had to be extended to give simpler examples
- Very different style to Java
  - Functional emphasised
  - But being sold by some as next Java

Scala

- Scala tackles technical challenges
  - parallel collections
  - functional
  - define a boolean using the language
  - turing complete generics
- Fails to tackle basic productivity issues
  - modules
  - built in immutability
  - pain of handling null

Fantom

- What does this code do?

```
Str:Int foo(Str[] strs) {
 res := Str:Int[:]
 strs.each {
  res[it] = res[it]?.increment ?: 1
 }
}
```

Fantom

- Tackles real developer challenges
  - Modules
  - Immutability
  - No shared state
  - Nullable types
- Not quite enough power or use of type system for my taste

New language

- Take a piece of reasonably complex code
- Give it to a mid-level developer
- Someone not interested in new languages
- Can they understand the code
- No training, no help

Examine some Java

```
/**
 * Copyright 2009 FooBar Ltd.
 *
 * Licensed under the Apache License v2
 */
...
```

Examine some Java

```
header {
   copyright: FooBar Ltd.
   copyrightYears: 2009
   licenseName: ApacheLicense
   licenseVersion: 2
}
...
```

Examine some Java

```
/**
 * Creates a money.
 * @param currency  the currency, not null
 * @param amount  the amount, null means 0
 * @return the currency, not null
 */
Money create(Currency c, Decimal bd)
```

Examine some Java

```
/**
 * Creates a money.
 */
Money create(Currency c, Decimal? bd = 0)
```

Examine some Java

```
List<String> filt(Map<String, Integer> m) {
 List<String> res=new ArrayList<String>();
 for (Entry<String, Integer> e : m) {
  if (e.getValue() > 20) {
   res.add(e.getKey());
  }
 }
 return res;
}
```

Examine some Java

```
String[] filt(Integer[String] m) {
 var res = String[]();
 loop (key, val : m) if (val > 20) {
  res.add(key);
 }
 return res;
}
```

Questions

Stephen Colebourne
Member of technical staff
OpenGamma Ltd