aRven Technologies

G'OO'D Behaviour

Steve Love

ACCU London, March 19th 2009

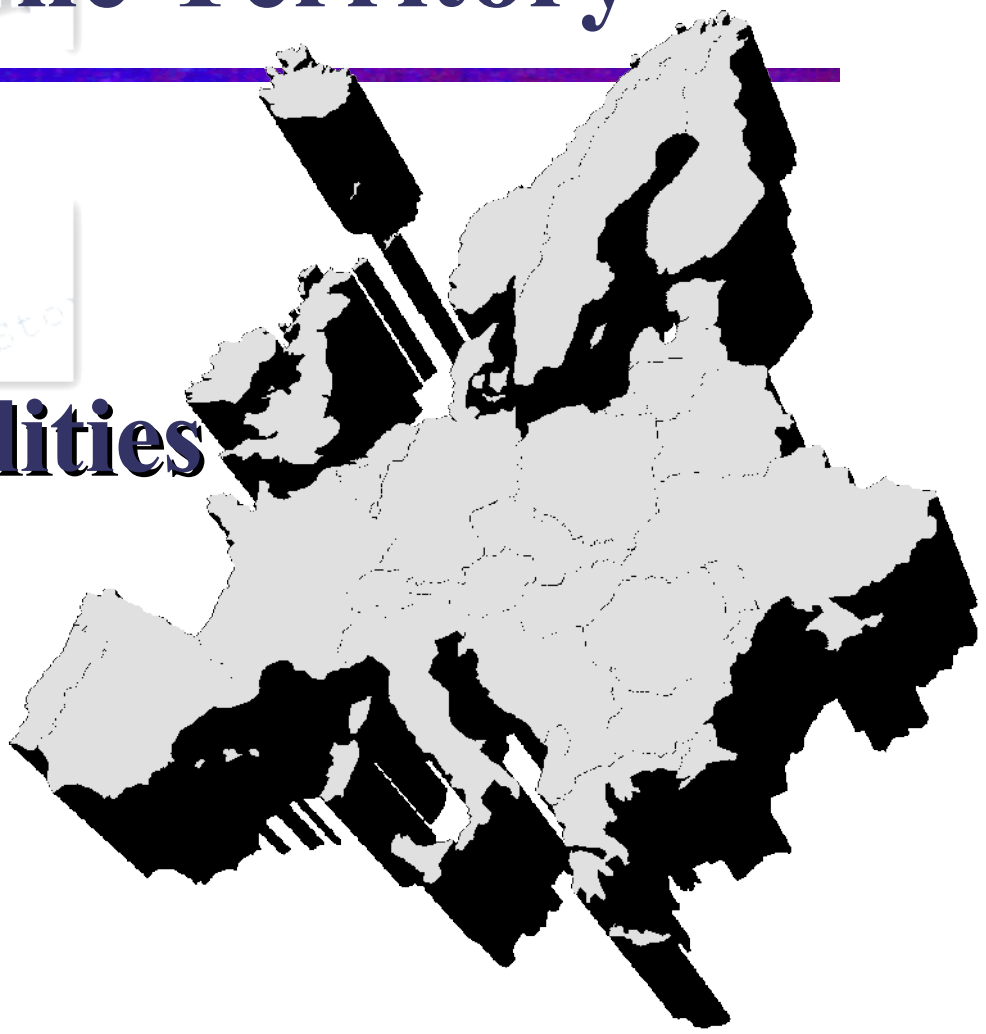
G'OO'D Behaviour

Without requirements or design, programming is the art of adding bugs to an empty text file.

Louis Srygley

The Map Is Not The Territory

- **Simplicity**
- **Roles and Responsibilities**
- **Equal Rights**
- **The Value of Values**



Simplicity

There are two ways of constructing a software design.
One way is to make it so simple that there are
obviously no deficiencies.
And the other way is to make it so complicated that there are
no obvious deficiencies.

C.A.R. Hoare

Simplicity

Don't get carried away

Baroque design leads to arcane usage

Simple is not stupid – or lazy...
...but laziness has its advantages too.

Simplicity

- Don't speculate on what you'll need in v2.0
- Concentrate on the immediate problem
  - Don't over-generalise
  - Don't be over-flexible
- Some apparent simplicity is fake
  - Implementation Inheritance
  - Copy and Paste Code

Simplicity

Laziness takes a lot of effort

but the result is often (always?) worth it.

Roles and Responsibilities

The open secrets of good design practice include the importance of knowing what to keep whole, what to combine, what to separate, and what to throw away.

Kevlin Henney

Roles and Responsibilities

- Model concepts with interfaces
- One Concept, One Interface
- One concept is not the same as one realisation
- Abstractions never depend on implementations

Roles and Responsibilities

- Encapsulation is not a cause – it's an effect
- Cohesive and Decoupled
- Clear specification leads to well-behaved interfaces
- Don't Make Spurious Relationships

Roles and Responsibilities

How do you want to use it?

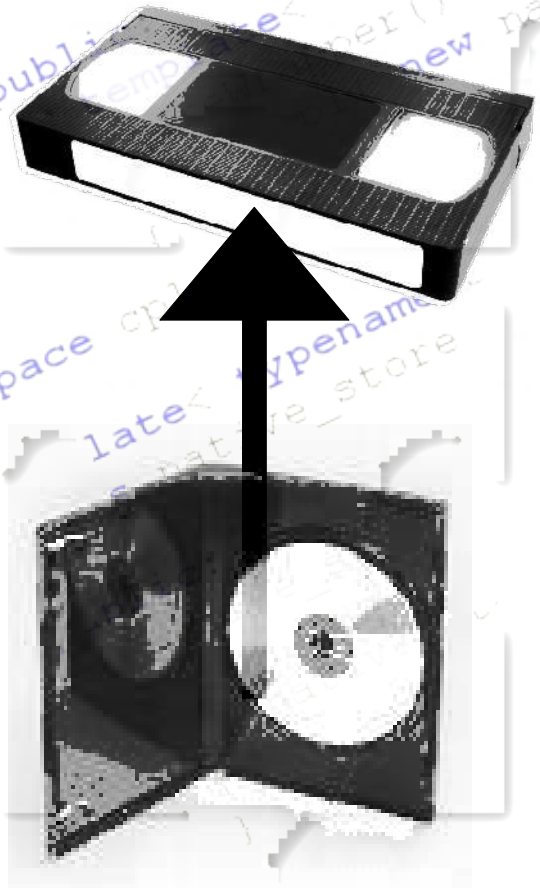
Unit Tests can help

Equal Rights

If it looks like a duck, swims like a duck
and quacks like a duck, it probably is a duck.

The Duck Test

Equal Rights



- Model substitutability correctly with IS-A
- Use Interfaces to represent Types
- Type != Class

Equal Rights

- Inheritance is not the right tool for all jobs
  - Duck Typing
  - Overloading
  - Coercion
- Don't break models to fit local customs (e.g. STL)
  - Containers of polymorphic things store references

Equal Rights

You Can't Inherit From Leaf Classes

Substitutability is about decoupling

The Value of Value

I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail.

Abraham Maslow

The Value of Values

- Don't mix value types with polymorphic models
- Understand the 3 variation points
  - Behaviour
  - State
  - Identity
- Exceptions are exceptions
- Whole Value represents more than just the value

The Value of Values

- Polymorphism is more than inheritance
- Keep an eye open for pessimisation and false optimisation
- Clearly define responsibilities of values and concepts

The Value of Values

Objects Aren't Always Polymorphic

But...don't treat those that are
as if they are values

Conclusions

- The map is not the territory
- There is no substitute for careful design
- Don't be tempted by clever techniques or methods

BUT

- There's more to life – and design – than OO
- At the end of the day it's all just 1's and 0's