Simplicity

The Way of the Unusual Architect

Dan North, DRW

In the beginning…

…the software was without form, and void

The Architects said "Let there be light," and they separated the light from the darkness

And they called the light Architecture and the darkness Hacking

And that was the first project

On the second project…

The Architects used all the technologies of the heavens and the earth they hadn't got round to the first time

The simple new() was replaced by a Factory

- which was replaced by Dependency Injection
- which was replaced by an IoC Container
- which was augmented by XML configuration
- which was supplemented by @nnotations

But they were not done yet...

The simple save() was replaced by a DAO

- which was replaced by a Unit of Work pattern
- which was replaced by a custom ORM
- which was replaced by Hibernate
  - which is called NHibernate by the Redmondites
- which was (partly) replaced by iBatis
- which was replaced by EJB 3
- which was (not) replaced by Active Record

And still they toiled…

The simple compile was replaced by a Makefile

- which was replaced by an Ant build.xml
  - which is called NAnt by the Redmondites
- which was replaced by many build.xml files
- which were generated by an XSLT transform
- which was replaced by Maven

And Maven brought forth a Plague of Apache Commons, and there was a flood of all the Libraries of the Internet as a judgement upon the people

And that was the Second System

Architects were fruitful and multiplied

They decided to build an Architecture that would reach to the heavens, to show how clever and wise they were, and remote invocation would be its name

But it came to pass that they were scattered to the four winds and began to speak in different tongues

Some spoke in CORBA, which was called DCOM by the Redmondites. The Sunnites spoke the language of JNDI, of the EJBites, which was XMLish and verbose

And there was a plague of standards to test the people

These are the generations of RPC

RPC begat RMI

- which begat COM and Object Brokers

COM begat DCOM, which begat WCF

Object Brokers begat Web Services

Web Services married XML

- and they begat SOAP and WSDL

SOAP begat the twelve (hundred) tribes of WS-*

WSDL begat Code Generated Stubs

And the people wrung their hands and wept

On the seventh day they RESTed

Here's what I know about remoting

1. You usually don't need remoting
2. HTTP works, and it's really simple
3. Pub/Sub works, and it's really simple
4. Systems share **data**, not objects

"Transfer Object" is an oxymoron

The same story happens over and over

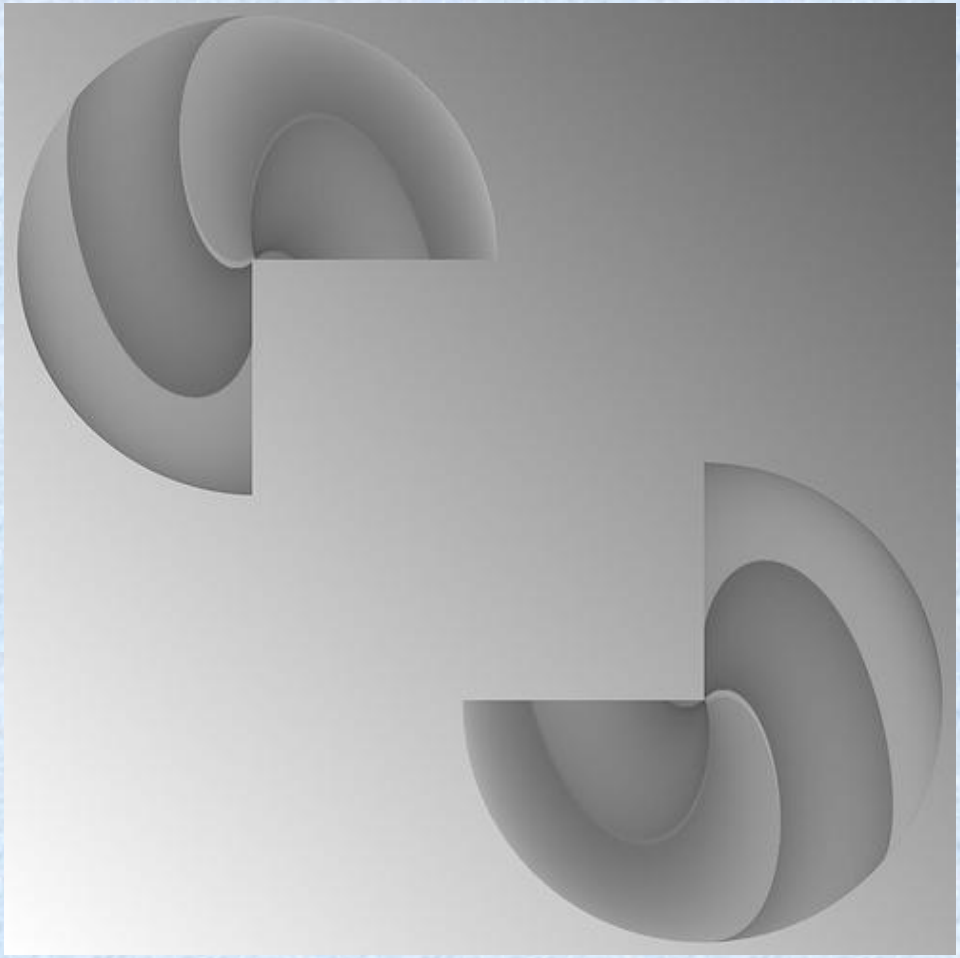
1. We observe a pattern
2. We create abstractions and generalisations
3. We turn the abstractions into a framework
4. The framework becomes a Golden Hammer

5. *People start to subvert the framework*

6. Finally, sometimes, simplicity grows out of adversity

Why do we keep doing this?

This is a pair of three-quarter circles



http://www.flickr.com/photos/davidjoyner/2491859887/

We are programmed to see structure

...even where none exist

We *distort*, *delete* and *generalise*

Vendors exploit this to sell us the Next Big Thing

We *complify* where we should *simplicate*

We choose to optimise for *Generality*

or *Flexibility*

or *Reusability*

or *Return-on-Investment*

"Some people, when confronted with a problem, think "I know, I'll use regular expressions." *Now they have two problems."*

– Jamie Zawinski, 1997

How can we get out of this mess?

"My name is Dan, and I'm a Complexaholic"

Identify complexity traps (aka gumption traps)

- Chunk up: *What for?*

Use time boxes to expose yak-shaving

Are you focusing on the bike shed?

- Chunk down: *How?*

"If I were going to Dublin..."

Question every dependency

Pull value rather than pushing a solution

- What is the shortest route between here and Done?

Ask: *What is actually slowing me down?*

Get a pair, or a bath duck

We tend to solve the wrong problem

What are we really trying to do when...

1. Clustering an application server
2. Deploying portlets in a Portal Server
3. Using a CASE tool
4. Using Maven
5. Using an Object-Relational mapper

Solving the right problem

Optimise for *Ease of Change*

Create options

Create a roadmap

“Maximise the work not done”

Deliberate Discovery

Systemically reducing your ignorance

Technology

Domain

People & Process

Conclusion

We are programmed to *complify*

The real goal is to *simplicate*

Deliberate Discovery can shine a light

"I would not give a fig for the simplicity this side of complexity, but I would give my life for the simplicity on the other side of complexity."

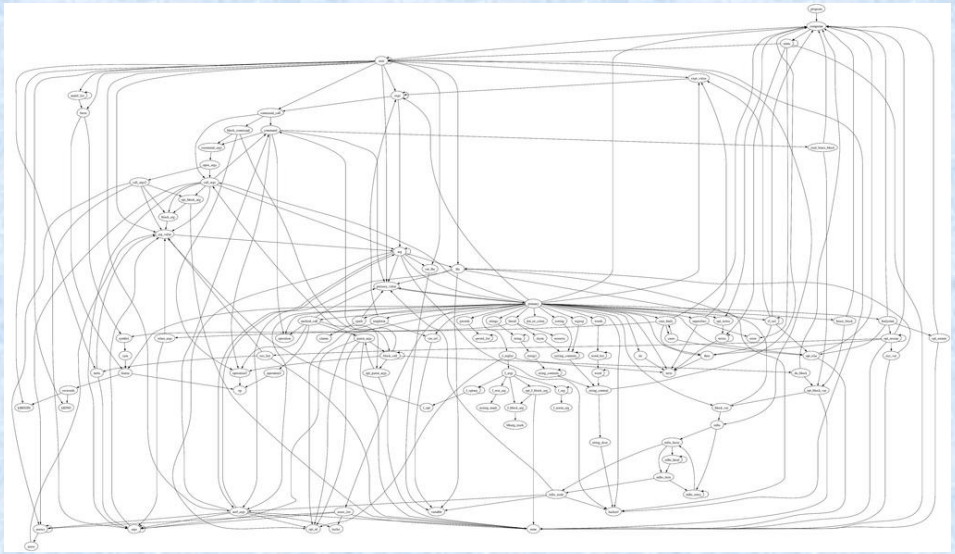
– Oliver Wendell Holmes

Thank you

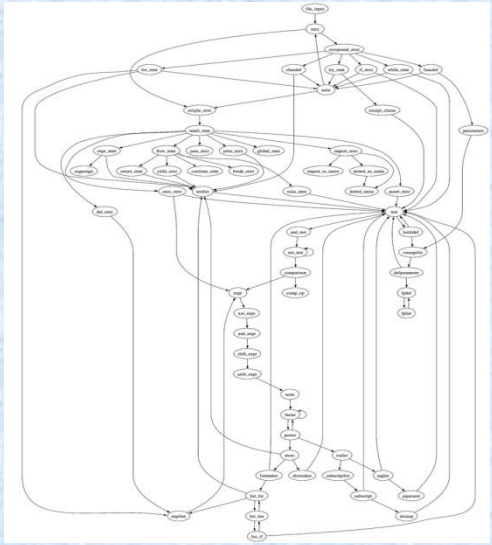
dnorth@drwuk.com

@tastapod

http://dannorth.net



http://www.flickr.com/photos/nicksieger/280661836/



http://www.flickr.com/photos/nicksieger/281055485/