Simplicity

The Way of the Unusual Architect

Dan North

Dan North & Associates

@tastapod

This week I learned...

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 XSL 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

The Architects were fruitful and multiplied

They decided to build an Architecture that would reach to the very Heavens, to show how clever and wise they were, and Distributed Systems 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 – who would one day be swallowed up by the mighty Whale of the Oracle – spoke in the tongue of JNDI, which was XMLish and verbose

And there was a plague of standards to test the people

These are the generations of Distributed Systems...

RPC produced RMI

which produced COM and Object Brokers

COM produced DCOM, which produced WCF

Object Brokers produced Web Services

Web Services married XML

and they had two sons, and SOAP and WSDL were their names

SOAP produced the twelve [hundred] tribes of WS-*

WSDL produced Code Generated Stubs

and the Abstractions did Leak forth upon the Software

And the people wrung their hands and wept

On the seventh day, they RESTed

The same story happens over and over

1. We observe a repeating pattern
2. We create abstractions
3. The abstractions become a framework

4. *People start to subvert the framework*

5. Finally, sometimes, simplicity grows out of adversity

Why do we keep doing this?

The Three Ages of... everything

1. Explore

   - maximize discovery

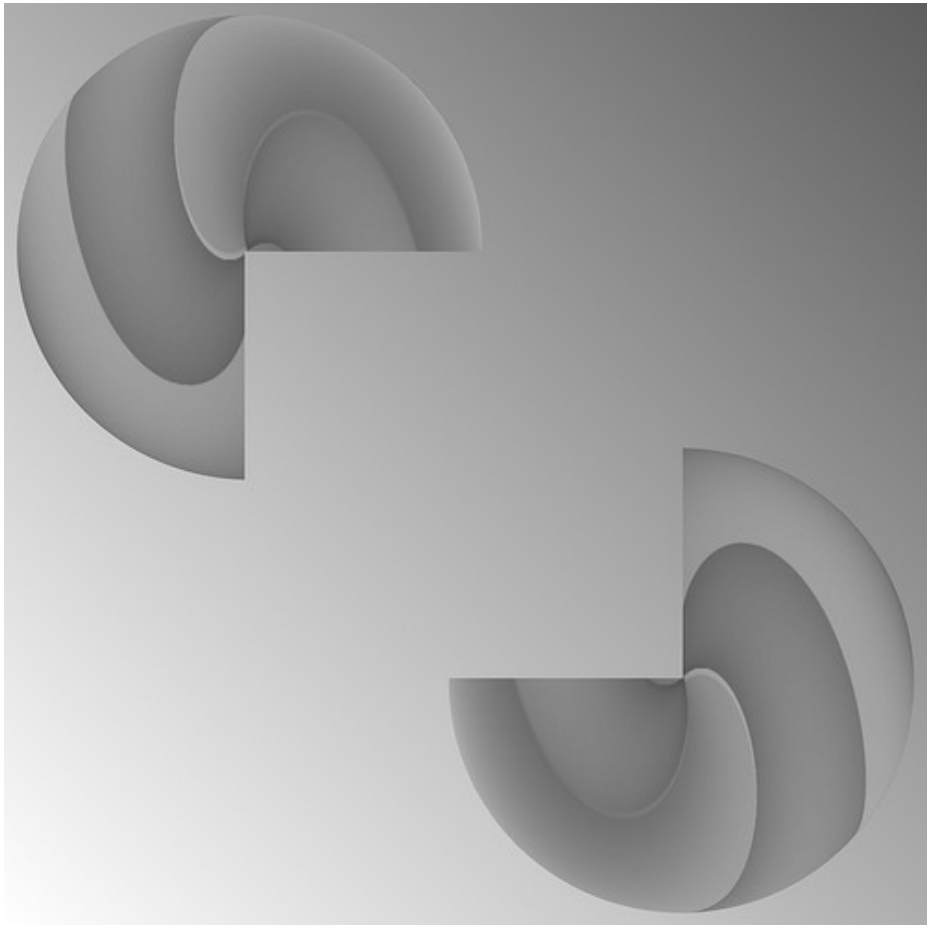
2. Stabilize

   - minimize variance

3. Commoditize

   - minimize cost

This is a pair of three-quarter circles



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

We are programmed to see structure

...even where none exists!

We distort, delete and generalize

Marketers exploit this to sell us the Next Big Thing

We complify where we should simplicate

We choose to optimize for generality

or flexibility

or reusability

or cost-per-use

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 accidental complexity

- Ask: *What is this for?*

Use time-boxes to challenge your progress

- Ask: *How else? Who else?*

Simplicity is different from familiarity

- Ask: *What really matters here?*

“If I were going to Dublin...”

Question every dependency

Pull value rather than pushing a solution

Ask: *What is actually slowing me down?*

Get a pair. Or a bath duck.

We tend to solve the wrong problem

What is the first-order problem when we are:

- clustering state across application instances
- using a business modeling tool
- using Maven
- using an Object-Relational Mapper

Solve the right problem

Simplicity leads to adaptability

Defer decisions to create options

Have a roadmap

“Maximize the work not done” - Kent Beck

Conclusion

We are programmed to complify

The real goal is to simplicate

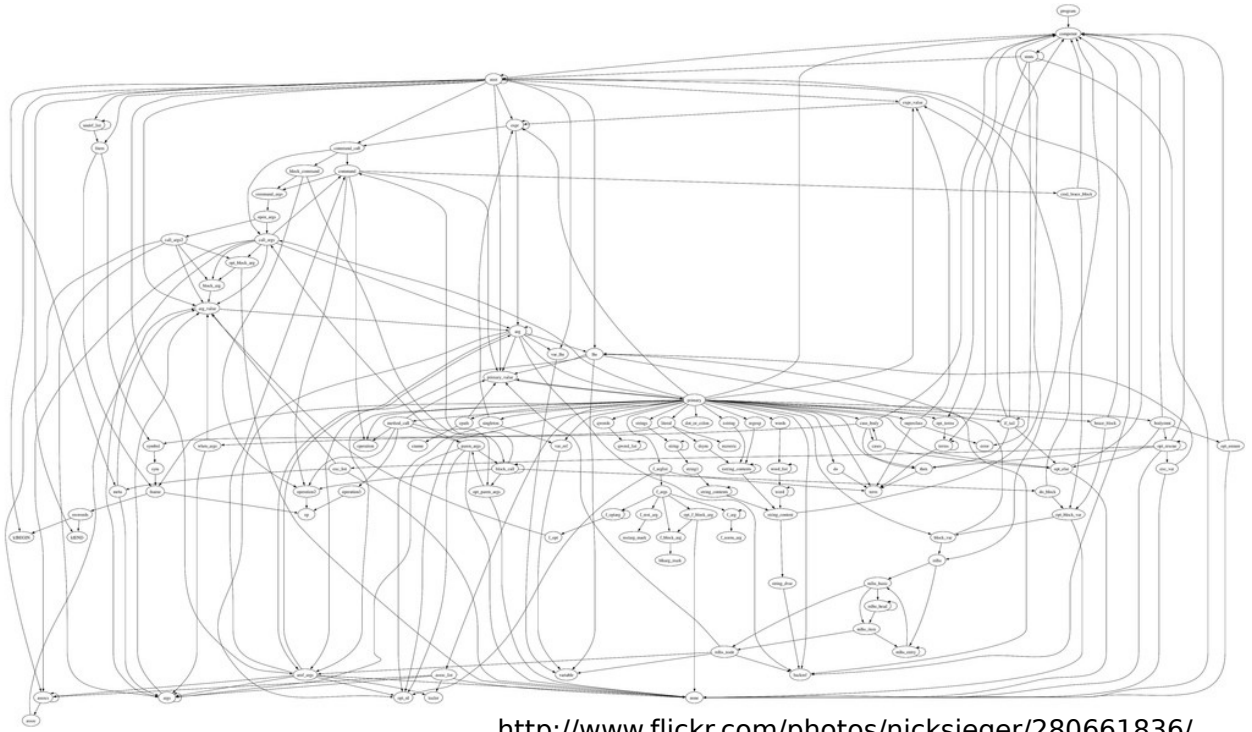
Always assume there is a simpler way

Thank you

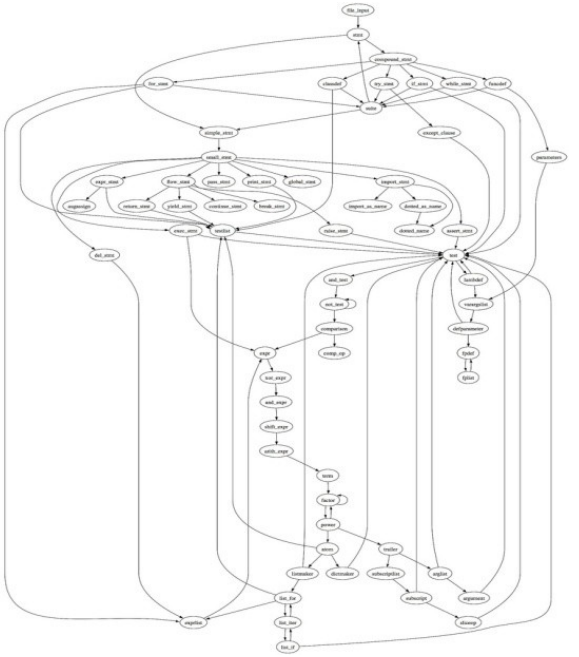
dan@dannorth.net

http://dannorth.net

@tastapod



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



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