coding {the} architecture

The conflict between agile and architecture

Myth or reality?

Simon Brown

simon.brown@codingthearchitecture.com

@simonbrown on Twitter

Jersey, Channel Islands

Jersey

JERSEY FINANCE

VOICE OF THE INTERNATIONAL FINANCE CENTRE

DIGITAL JERSEY

Locate Jersey

I help software teams understand

software architecture, technical leadership and the balance with agility

(I code too)

Training

Book

Speaking

What is

agile?

What is architecture?

As a noun...

Structure

The definition of something in terms of its components and interactions

and

As a verb...

Vision

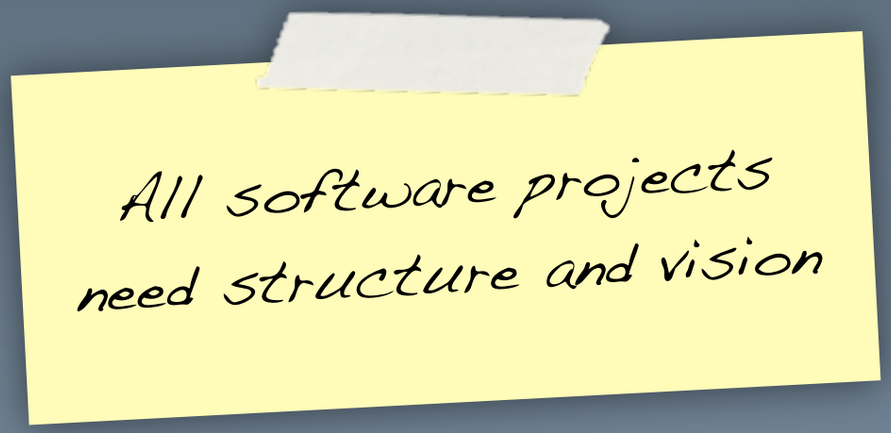
The process of architecting, making (significant) design decisions, etc

The conflict between agile and architecture

Myth or reality?

Myth

There is no conflict between agile and architecture



1. A conflict in team structure

vs

Dedicated
software architect

Single point of responsibility for the technical aspects of the software project

Everybody is a
software architect

Joint responsibility for the technical aspects of the software project

Big up front design

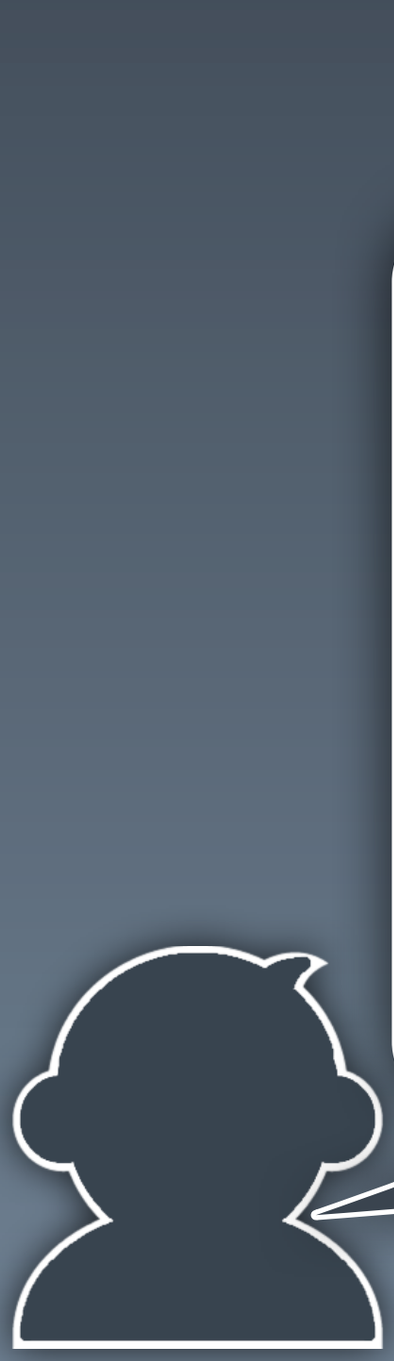
and analysis paralysis

Waterfall

UML

I'm a

software architect



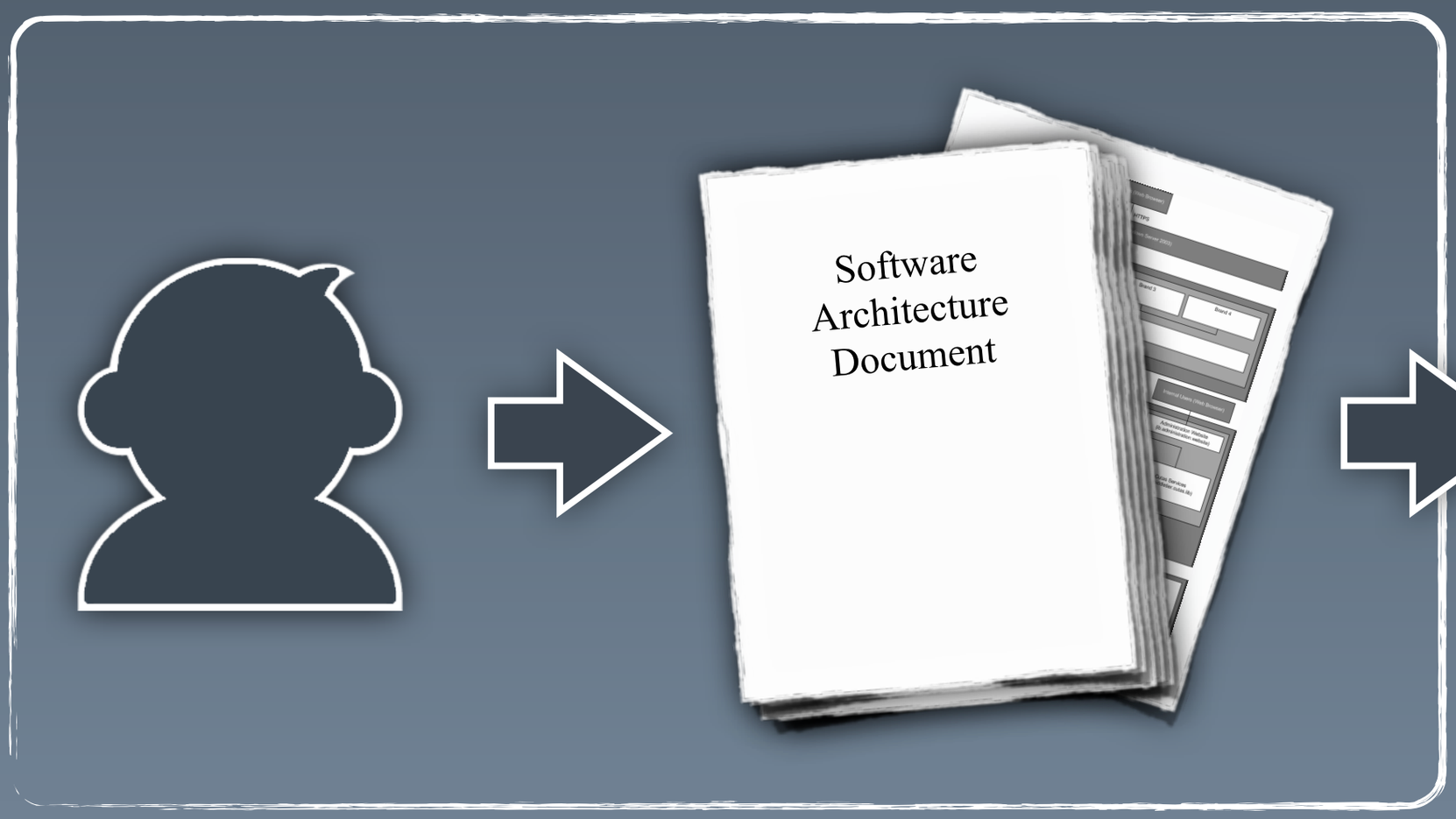
Ivory Tower

PowerPoint Architect

Architecture Astronaut

Software development is not a

relay sport



AaaS ... architecture as a service

Architects?

We don't need no stinkin’ architects!

Small teams of generalising specialists, everybody does everything

With agile, there is often a perception that you must have self-organising teams

2. A conflict in process

VS

```
/// <summary>
/// Represents the behaviour behind the ...
/// </summary>
public class SomeWizard : AbstractWizard
{
    private DomainObject _object;
    private WizardPage _page;
    private WizardController _controller;

    public SomeWizard()
    {
    }

    ...

}
```

Big up front design

Requirements capture, analysis and design complete before coding starts

Evolutionary architecture

The architecture evolves secondary to the value created by early regular releases of working software

The conflict relates to the desired

approach

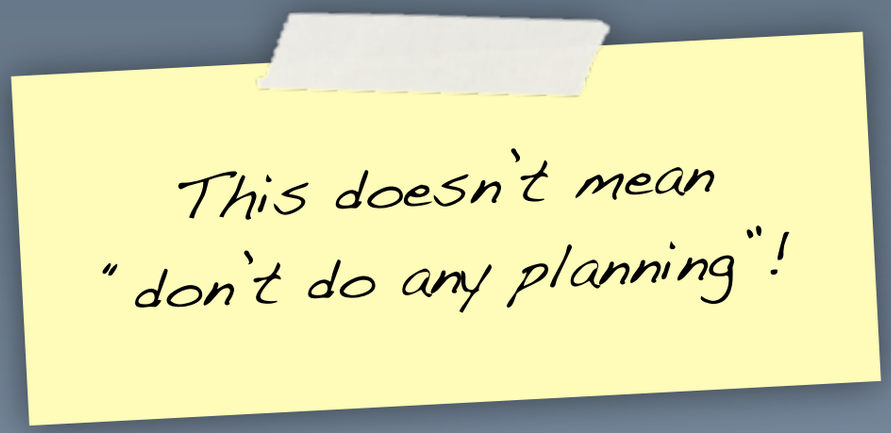
Moving fast, embracing change, delivering value early, getting feedback

vs

Understanding everything up front, defining a blueprint for the team to "follow"

Responding to change over following a plan

Manifesto for Agile Software Development, 2001



Modern software development teams often seem afraid of doing

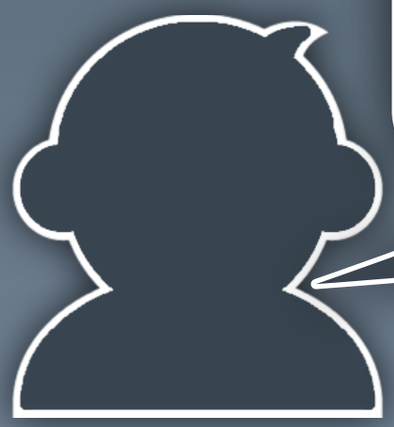
analysis

No

design up front

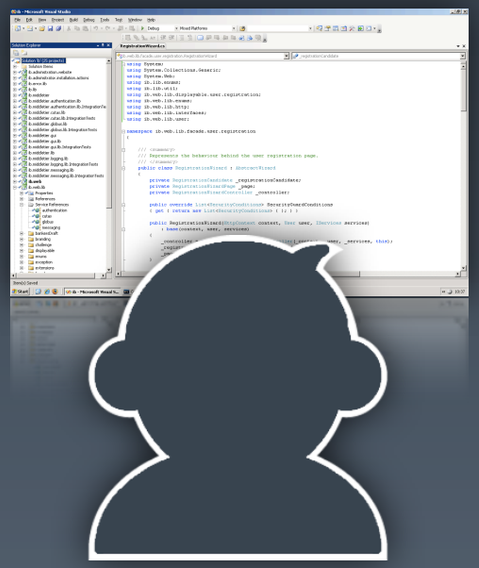
We don't need
software architecture;

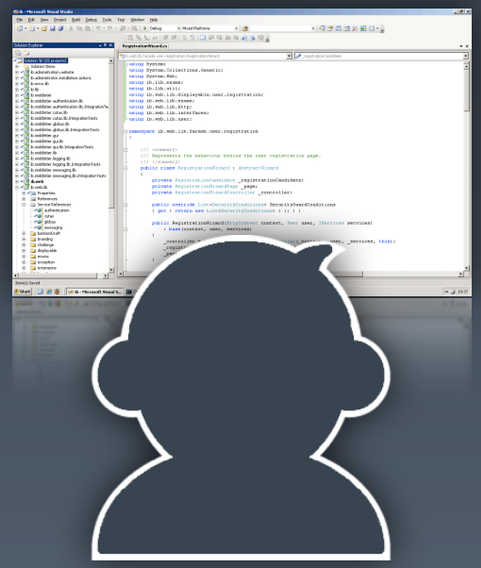
we do TDD

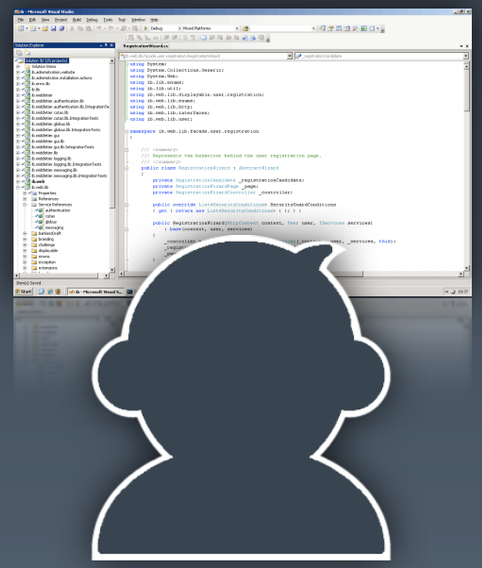


Agile software team

The result of the conflicts?

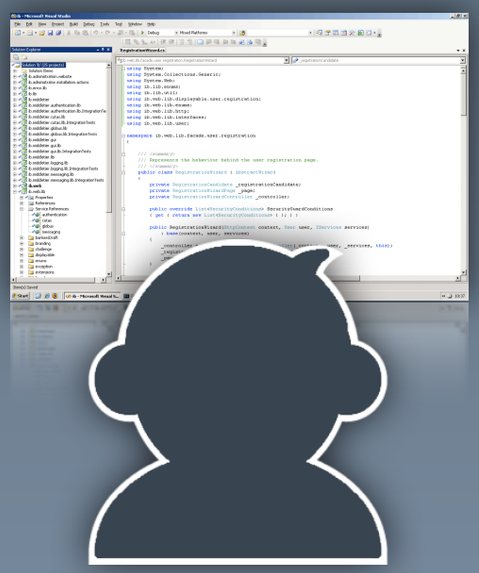


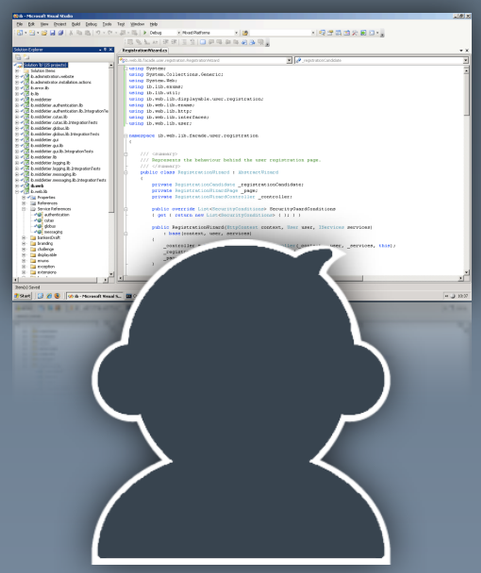


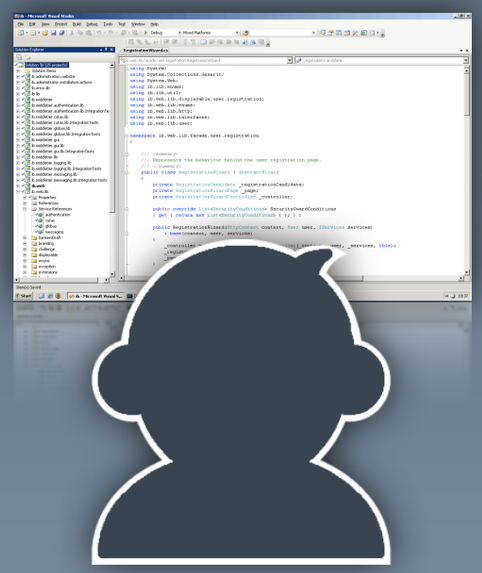


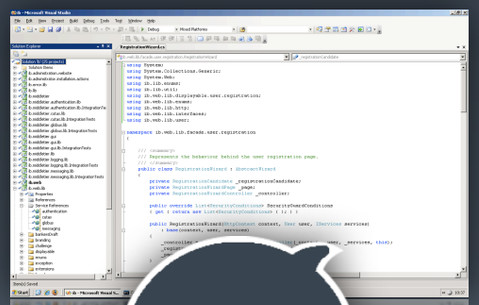
Chaos!

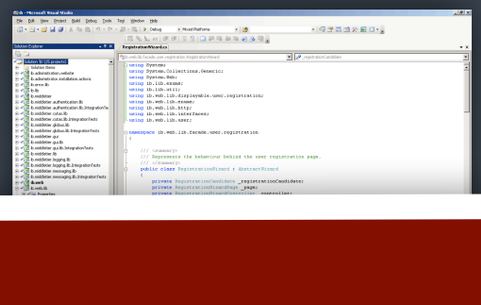
Does the team understand what they are building and how they are building it?

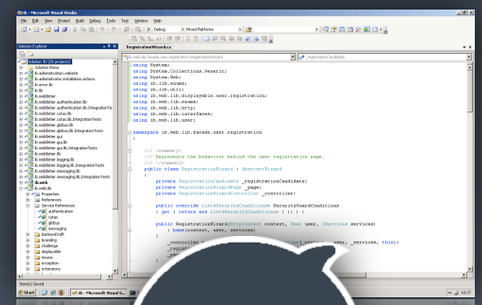








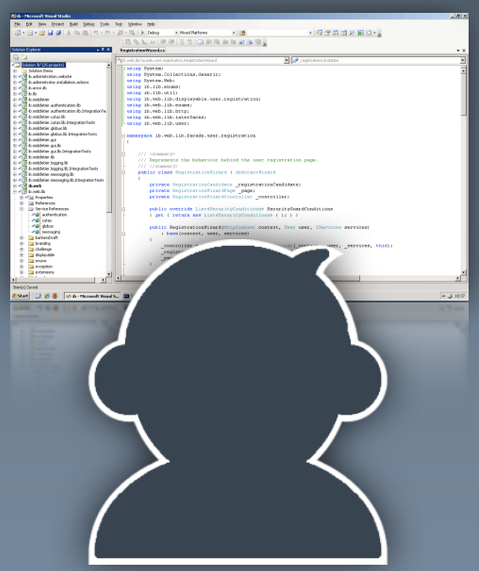


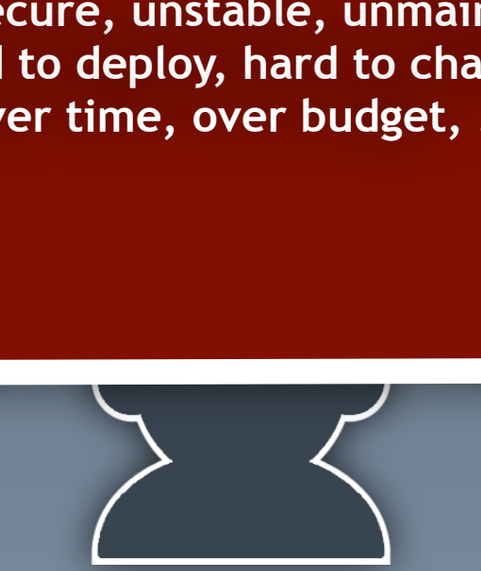


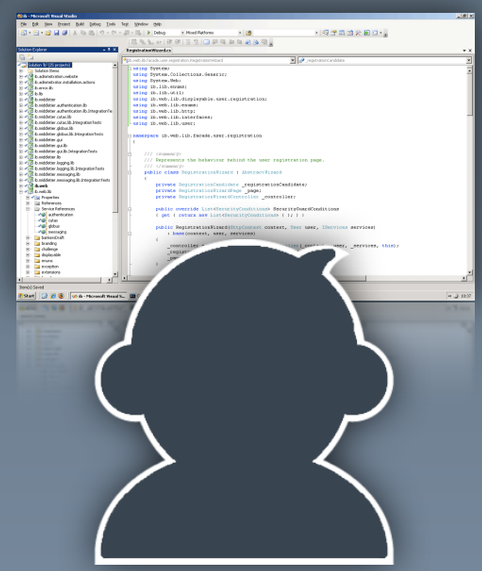
No defined structure,
inconsistent approaches,
big ball of mud,
spaghetti code, ...

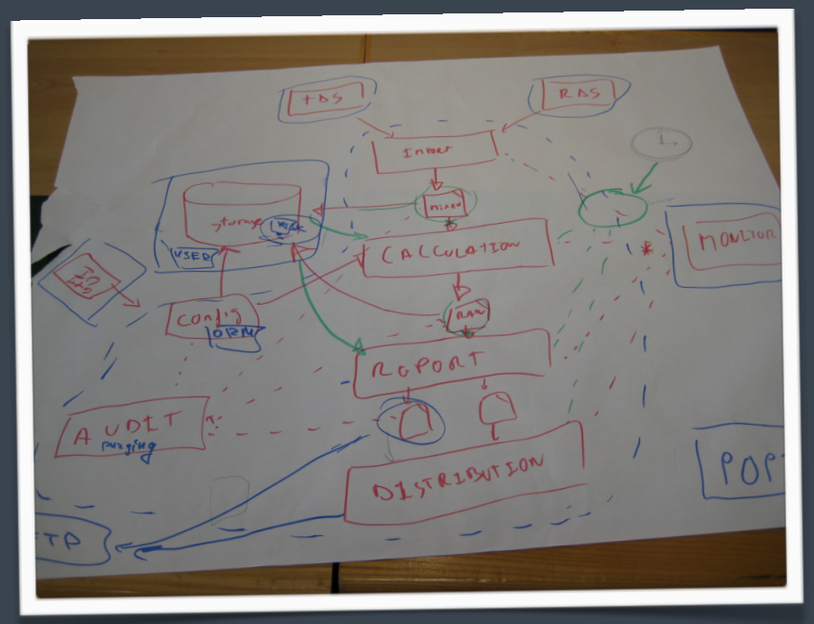
STOP

Slow, insecure, unstable, unmaintainable,
hard to deploy, hard to change,
over time, over budget, ...



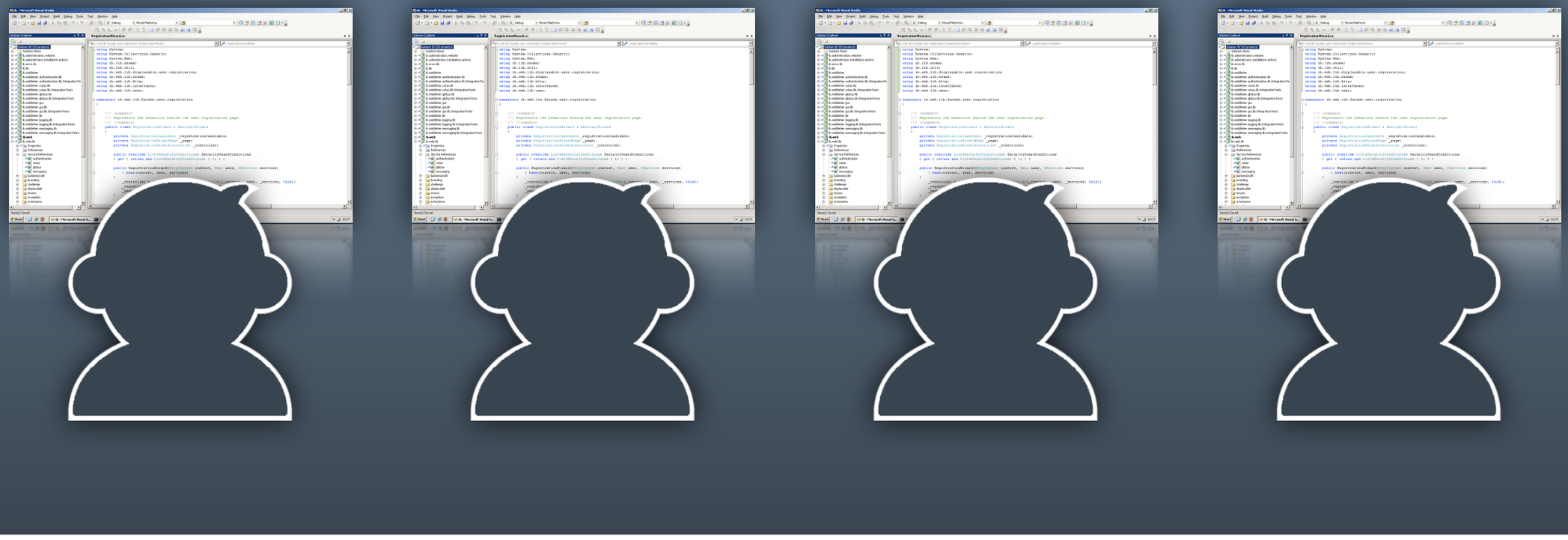




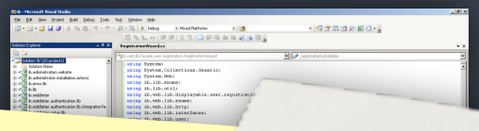


Shared vision of

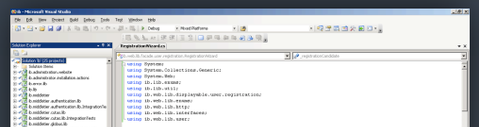
WTF?!



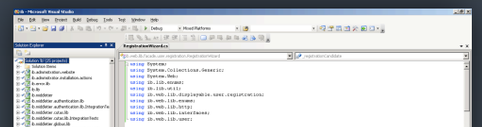
Software architecture in the 21st century



Let`s agree
on some things



No defined structure,
inconsistent approaches,
big ball of mud,
spaghetti code, ...

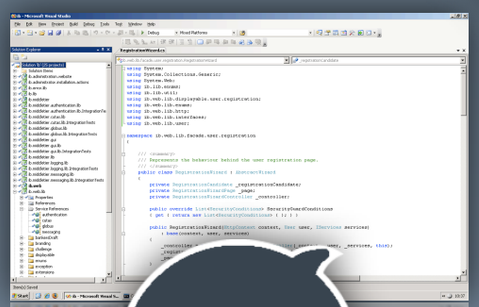


Let`s make the implicit,
explicit

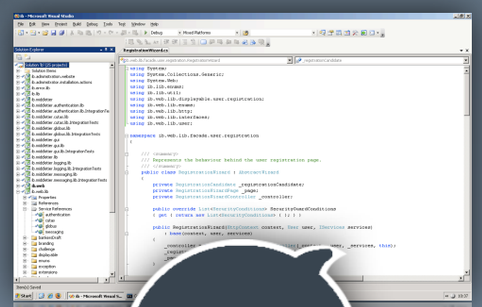
STOP

Doe g it?

Slow, insecure, unstable, unmaintainable,
hard to deploy, hard to change,
over time, over budget, ...



Put some boundaries
and guidelines in place



1. A conflict in team structure

vs

Dedicated
software architect

Single point of responsibility for the technical aspects of the software project

Everybody is a
software architect

Joint responsibility for the technical aspects of the software project

Every software development team needs a master builder

Generalising

Specialist

Depth

Deep hands-on technology skills and knowledge

Good software architects are master-builders

Breadth

Broad knowledge of patterns, designs, approaches, technologies, non-functional requirements

...

Awareness of options and trade-offs

The software architecture role

From chaos to self-organising

Dedicated software architect

Single point of responsibility for the technical aspects of the software project

Elastic Leadership (Roy Osherove)
Chaos (command and control),
learning (coaching),
self-organising (facilitation)

Everybody is a software architect

Joint responsibility for the technical aspects of the software project

2. A conflict in process

VS

```
/// <summary>
/// Represents the behaviour behind the ...
/// </summary>
public class SomeWizard : AbstractWizard
{
    private DomainObject _object;
    private WizardPage _page;
    private WizardController _controller;

    public SomeWizard()
    {
    }

    ...

}
```

Big up front design

Requirements capture, analysis and design complete before coding starts

Evolutionary architecture

The architecture evolves secondary to the value created by early regular releases of working software

How much up front design should you do?

Big design up front?

Emergent design?
(or none, depending on your viewpoint!)

Waterfall

RUP®
Rational Unified Process®

Extreme Programming

DSDM
Atern

ScrumAlliance®
transforming the world of work

Something in between?

You should do

“just enough”

Isn't agile about being flexible and adapting to the context? :-)

What's important?

Significant decisions

Low-level details

Understanding the significant elements and how they fit together

Class and sequence diagrams covering every user story

Understanding how security will work

Defining the length of all database columns

Just enough up front design to

understand the structure

of the software and

create a shared vision

for the team

You need to

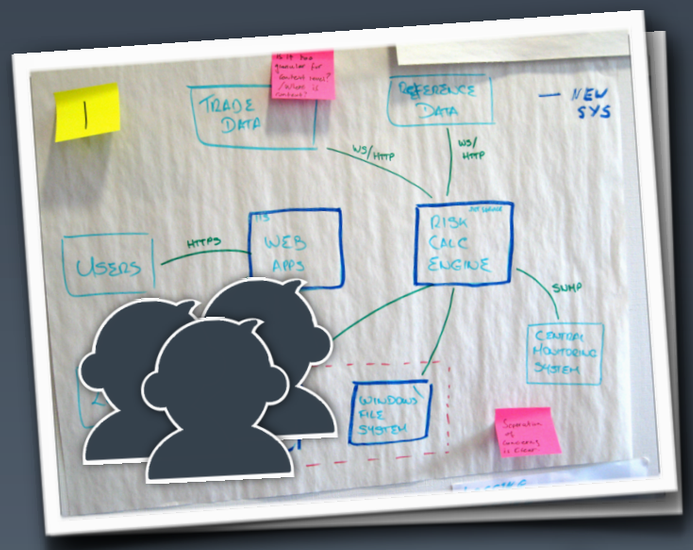
identify and mitigate your highest priority risks

Things that will cause your project to fail or you to be fired!

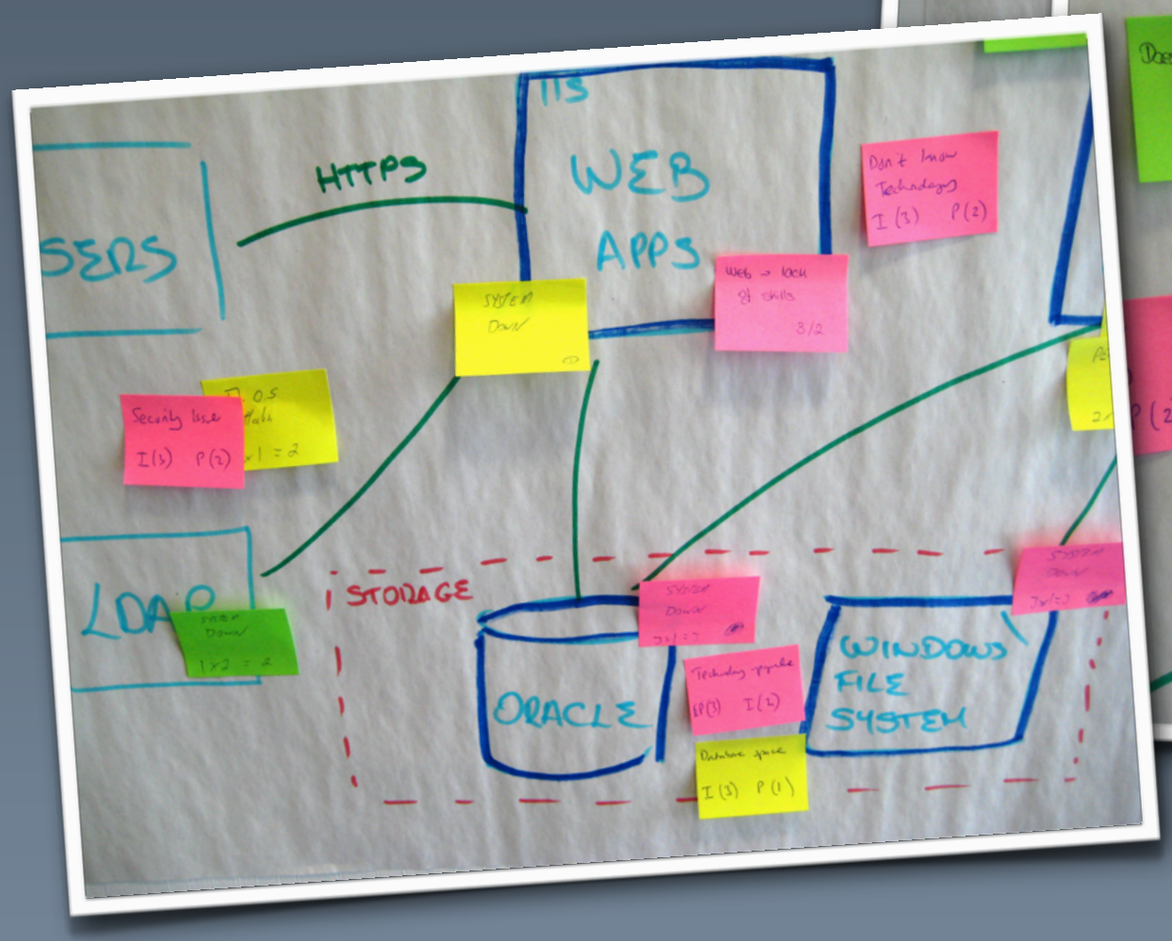
Agile software projects do have risks, right? :-)

Probability

Impact	Low (1)	Medium (2)	High (3)
Low (1)	1	2	3
Medium (2)	2	4	6
High (3)	3	6	9



Risk-storming



A collaborative and visual technique for identifying risk

The role

“just enough”
software architecture

Understand how the significant elements fit together

Identify and mitigate the key risks

Provide firm foundations and a vision to move forward

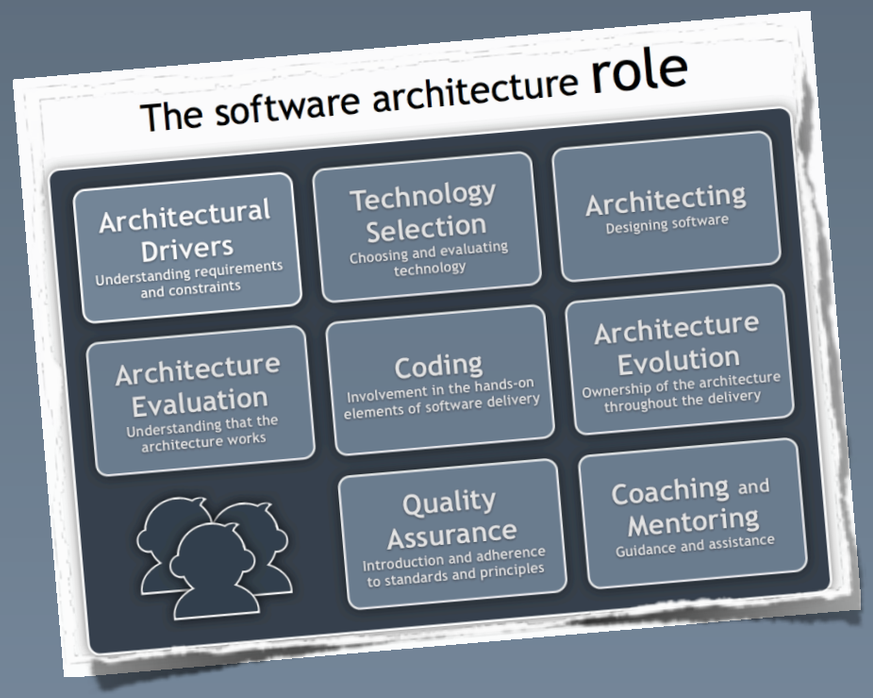
The process

```
/// <summary>
/// Represents the behaviour behind the ...
/// </summary>
public class SomeWizard : AbstractWizard
{
    private DomainObject _object;
    private WizardPage _page;
    private WizardController _controller;

    public SomeWizard()
    {
    }

    ...

}
```

Is a collaborative and lightweight approach to software architecture

the **missing piece**

of the **jigsaw**?



Do whatever works for YOU

simon.brown@codingthearchitecture.com
@simonbrown on Twitter

Buy the ebook
for only
$10
1GfwZBLaUKAM
(code expires 8th May 2013)