coding {the} architecture

Just enough
software architecture

Simon Brown

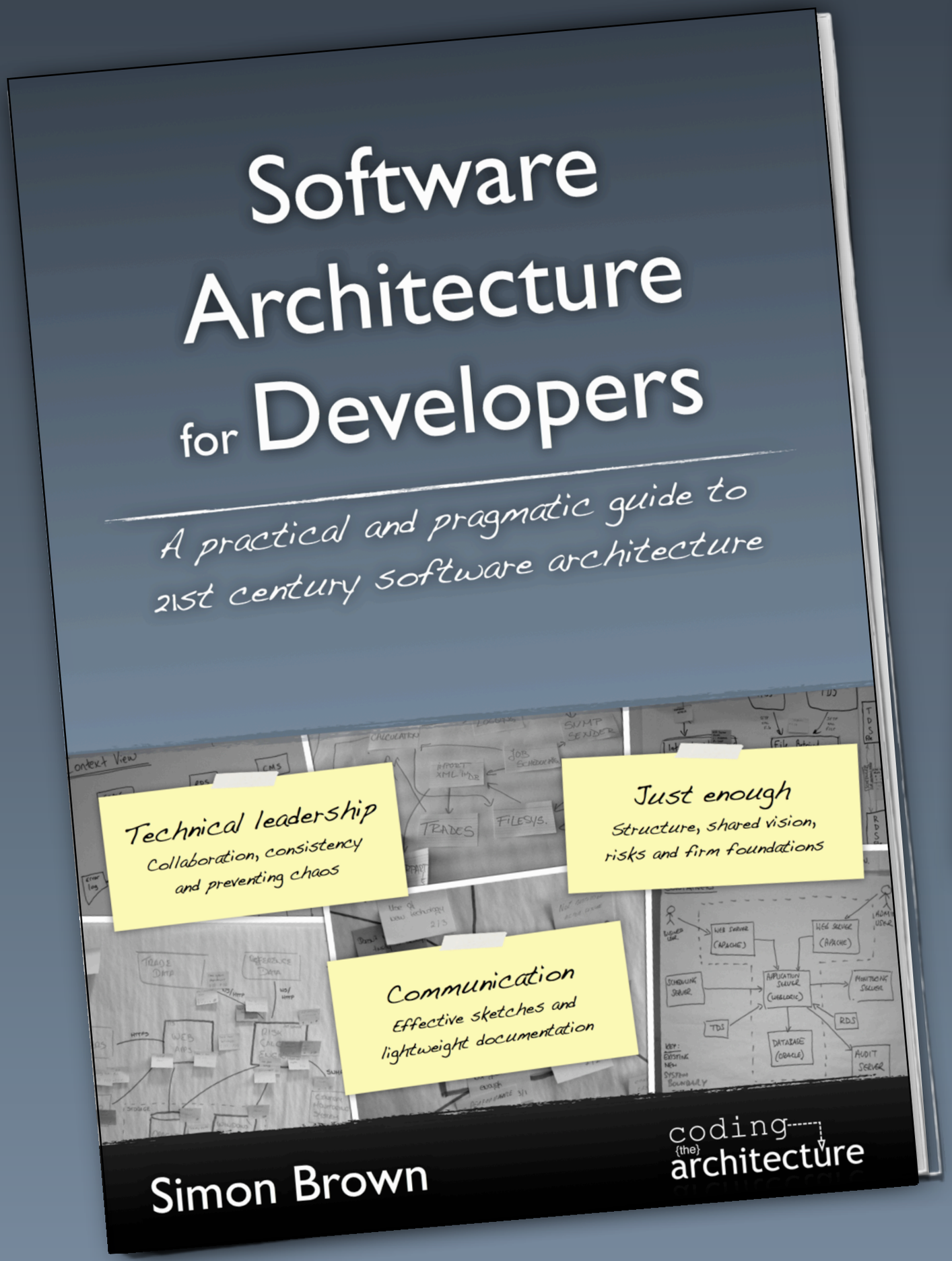
Jersey, Channel Islands

simon.brown@codingthearchitecture.com

@simonbrown on Twitter

Writing

Training and consulting



Published incrementally

Variable pricing

Buy now and get free updates

Leanpub

leanpub.com

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

What is design?

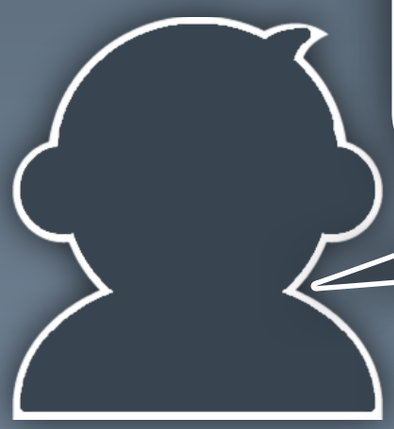
As a noun, design is the named structure or behaviour of a system whose presence resolves … a force on that system.

A design thus represents one point in a potential decision space.

Grady Booch
http://www.handbookofsoftwarearchitecture.com/index.jsp?page=Blog&part=2006

What is design?

All architecture is design, but not all design is architecture.

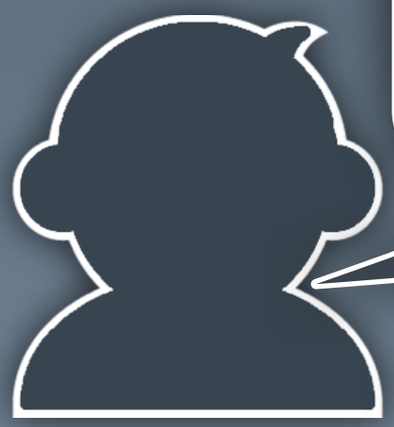


Grady Booch

http://www.handbookofsoftwarearchitecture.com/index.jsp?page=Blog&part=2006

What is design?

Architecture represents the
significant decisions,
where significance is measured
by **cost of change**.



Grady Booch
http://www.handbookofsoftwarearchitectu…log&part=2006

*Can you **refactor** it in an afternoon?*

Team structure

A difference 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

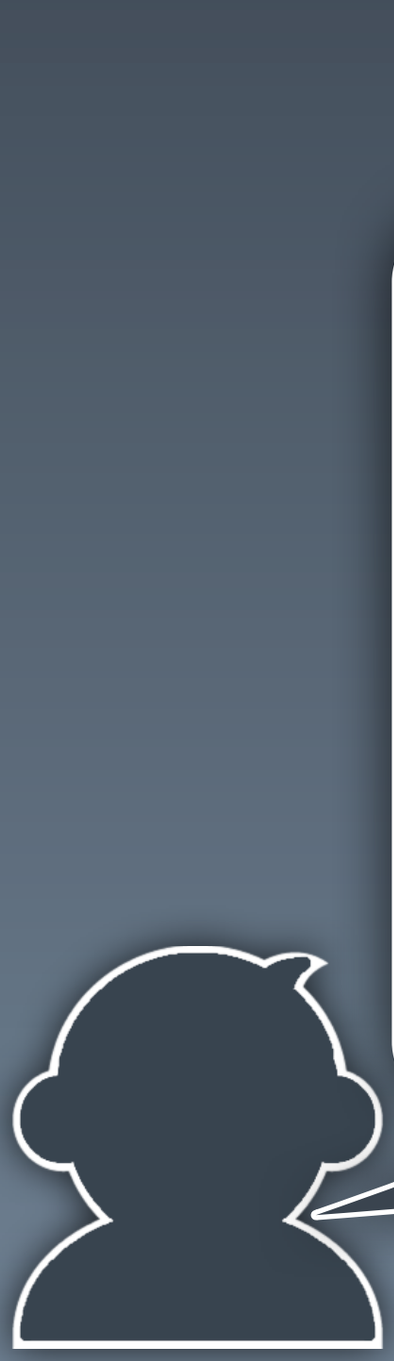
and analysis paralysis

Waterfall

UML

I'm a

software architect



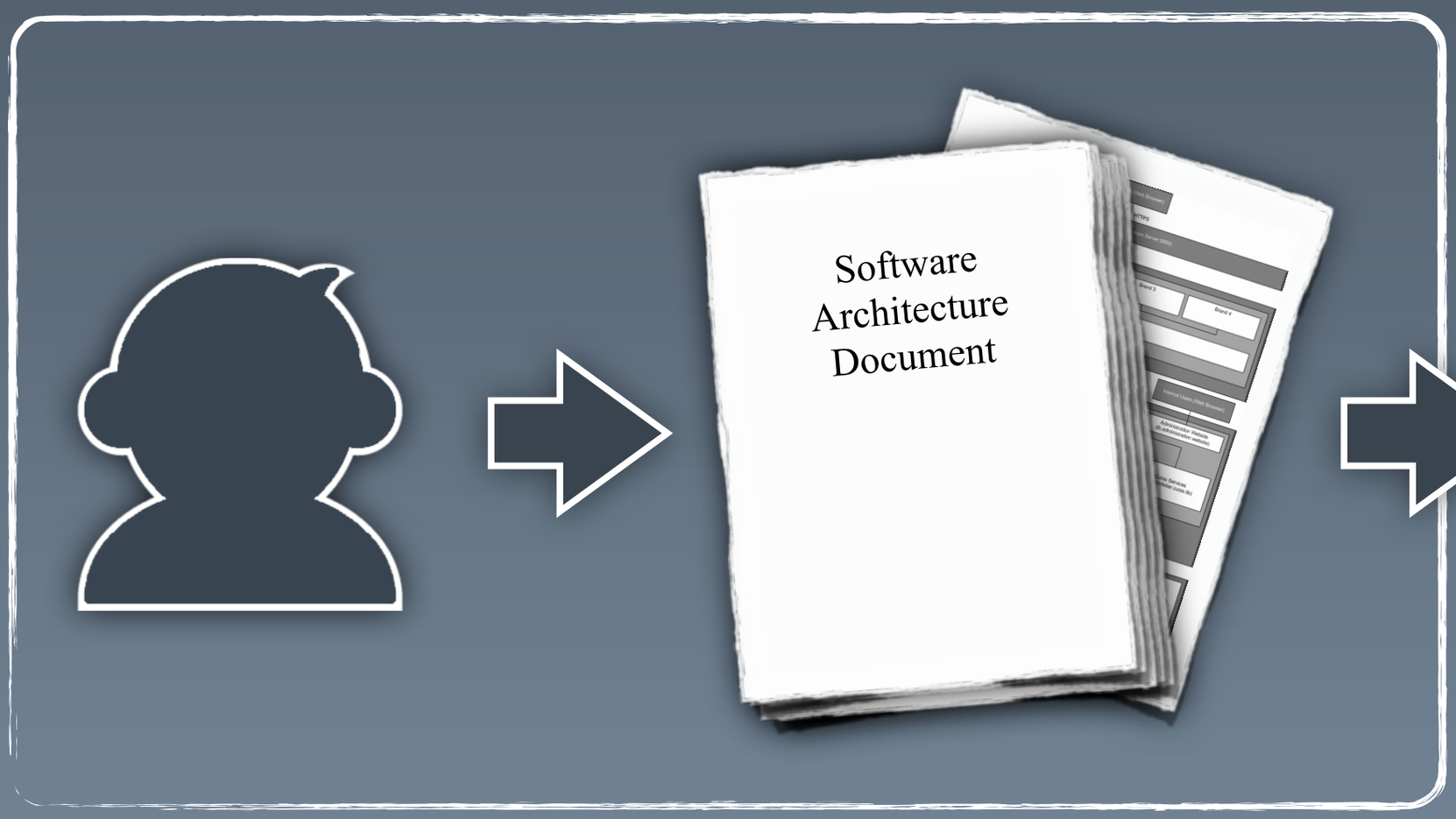
Ivory Tower

PowerPoint Architect

Architecture Astronaut

Software development is not a

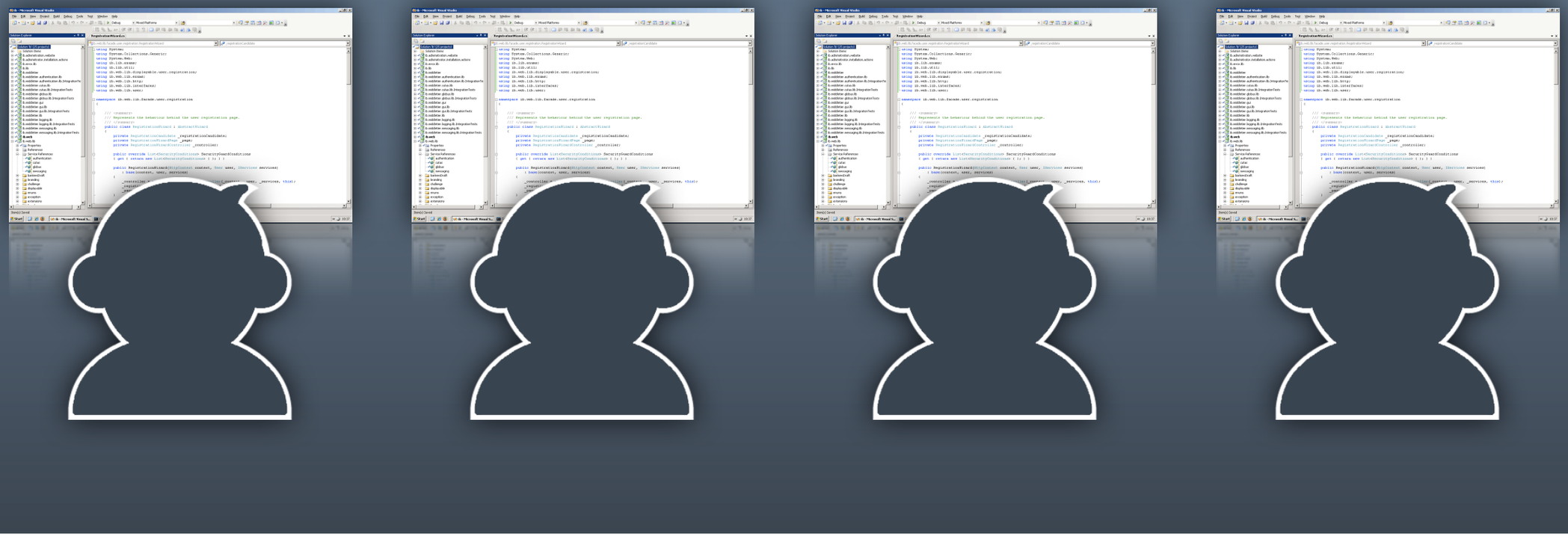
relay sport



AaaS ... architecture as a service

Shared vision of

TL;DR



Architect

Sits in an ivory tower

Gap

Focusses on the low level detail

Developer Developer Developer Developer Developer

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

Process

A difference 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

Approach

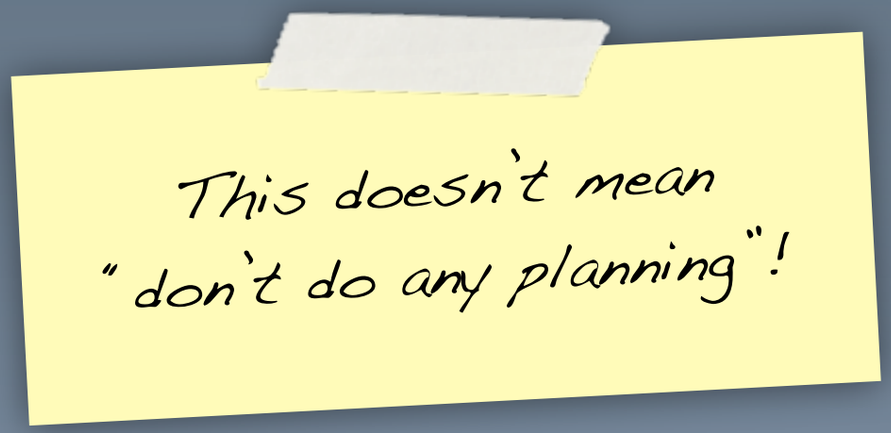
Moving fast, embracing change, delivering value early, getting feedback

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

What's the context of your current software project?

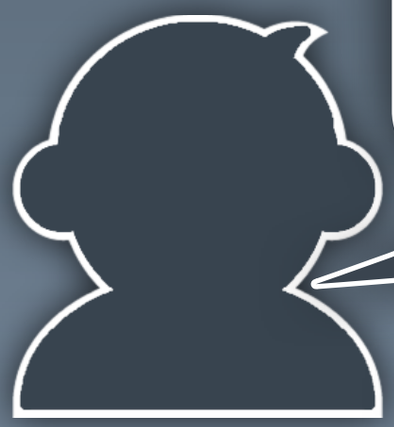
1. Who are the key users/roles/personas?
2. What are the key system dependencies?
3. What are the key constraints you're working within?

No

design up front

We don't need
software architecture;

we do TDD



Agile software team

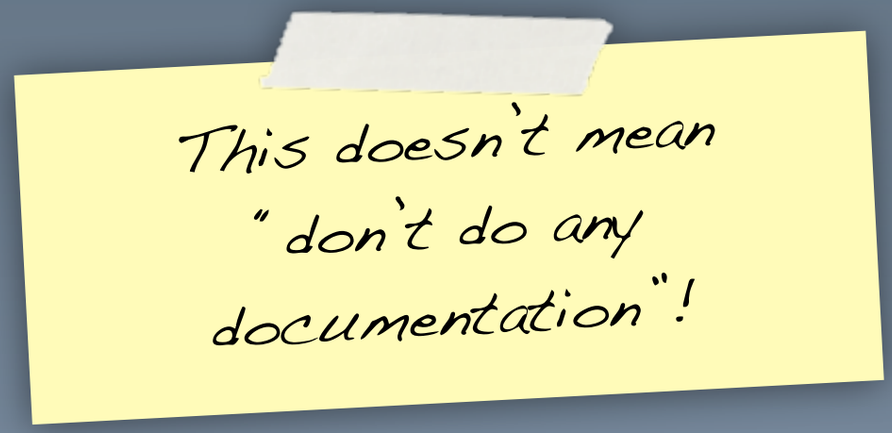
Working software

over

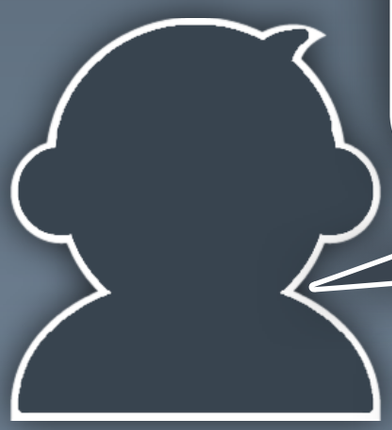
comprehensive

documentation

Manifesto for Agile Software Development, 2001



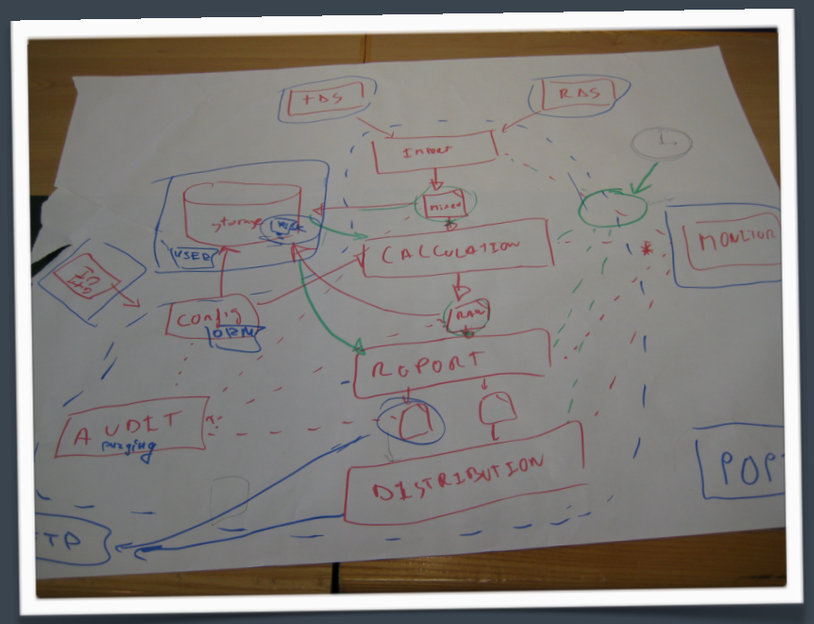
We're kind of agile



Most software teams?

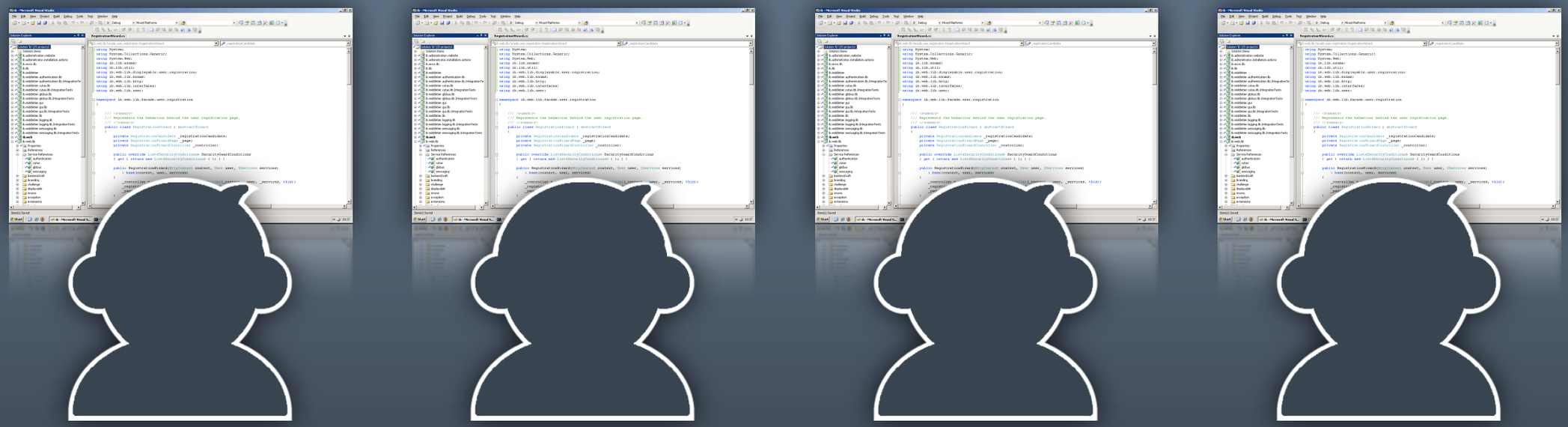
Many teams never did any design up front or documentation anyway! :-/

The result of the differences?

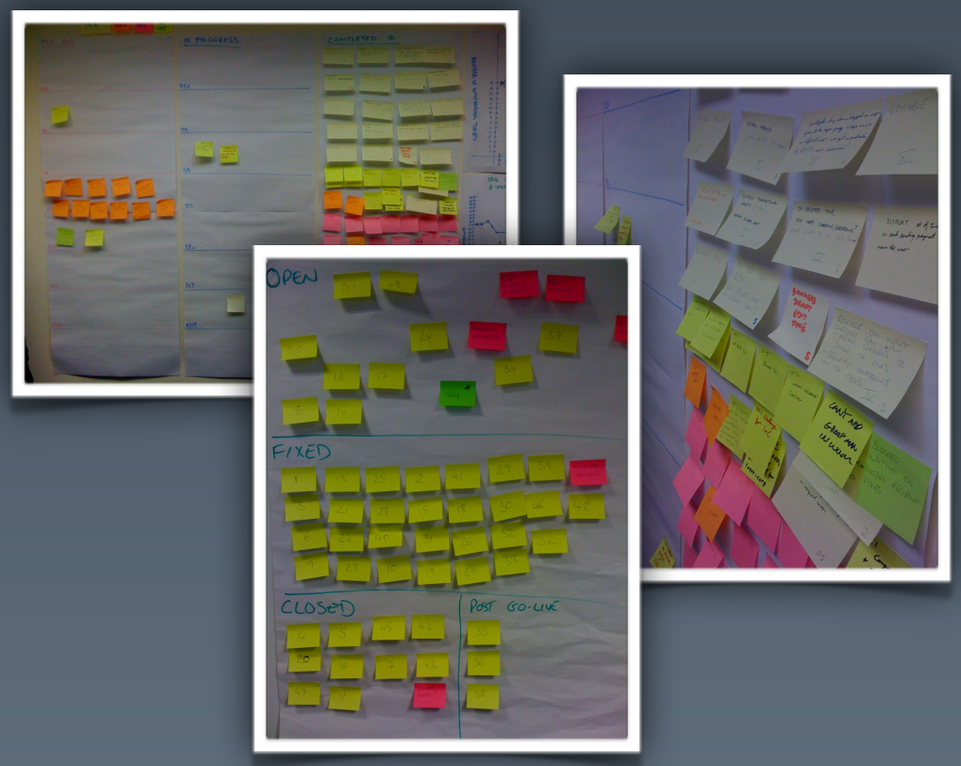


Shared vision of

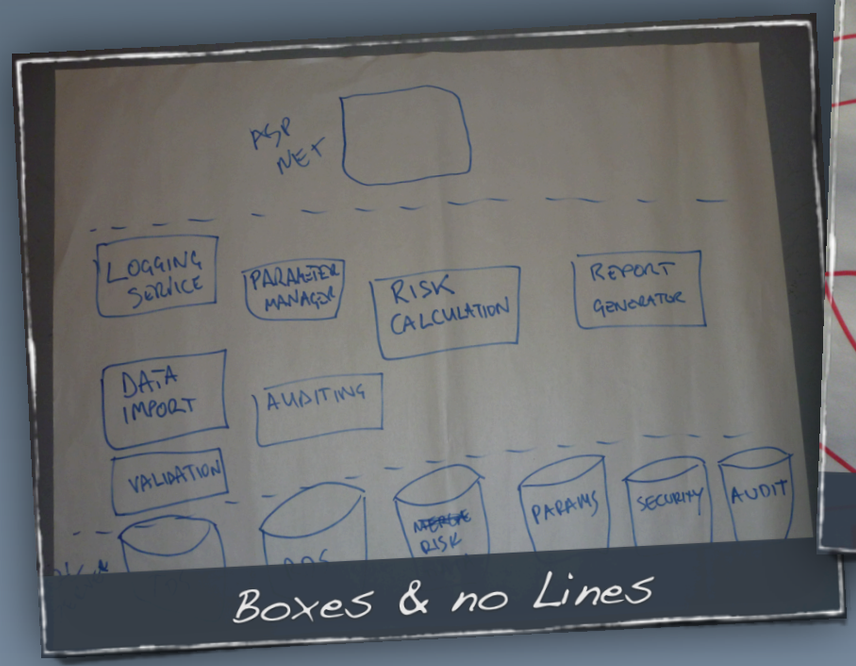
WTF?!



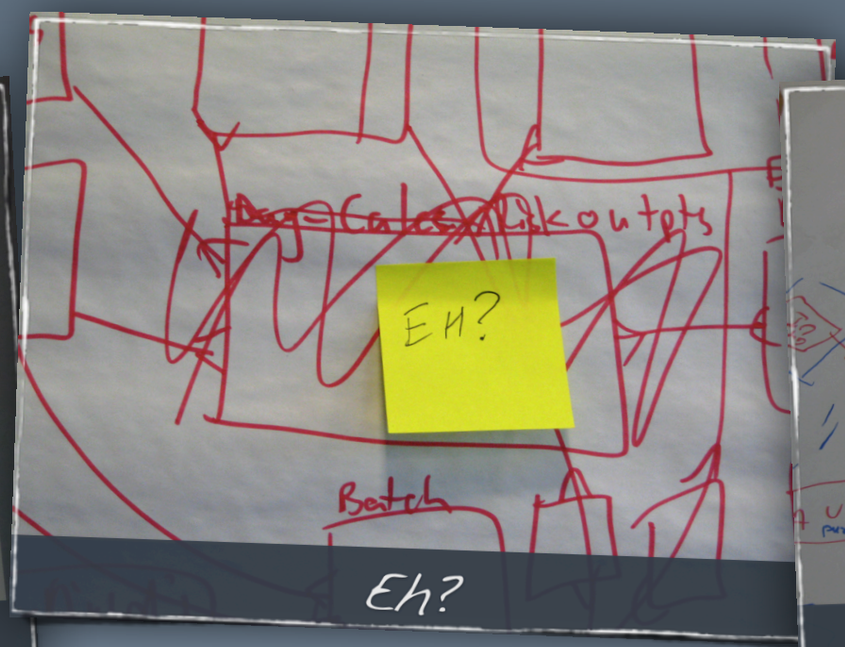
We can visualise our process…



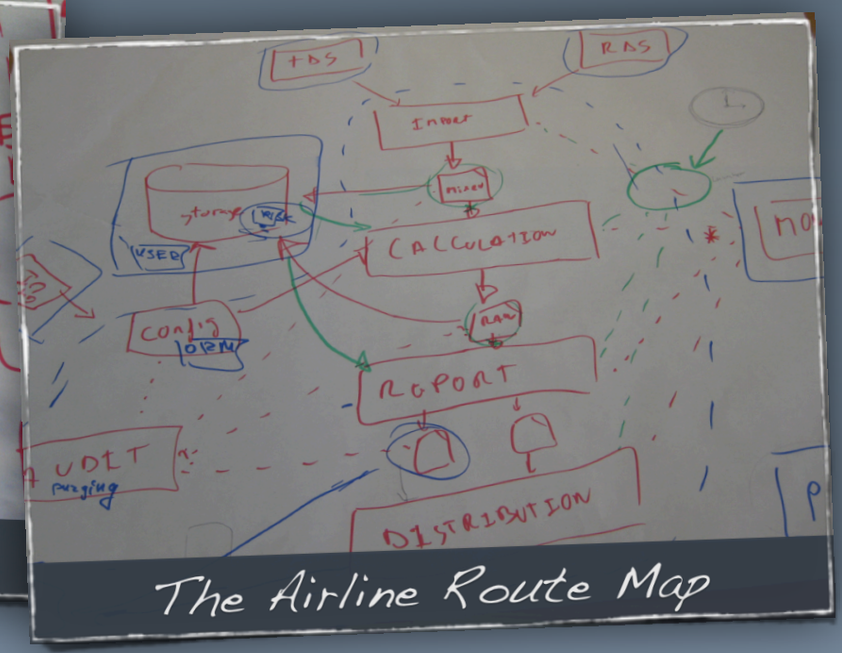
…but not our software!



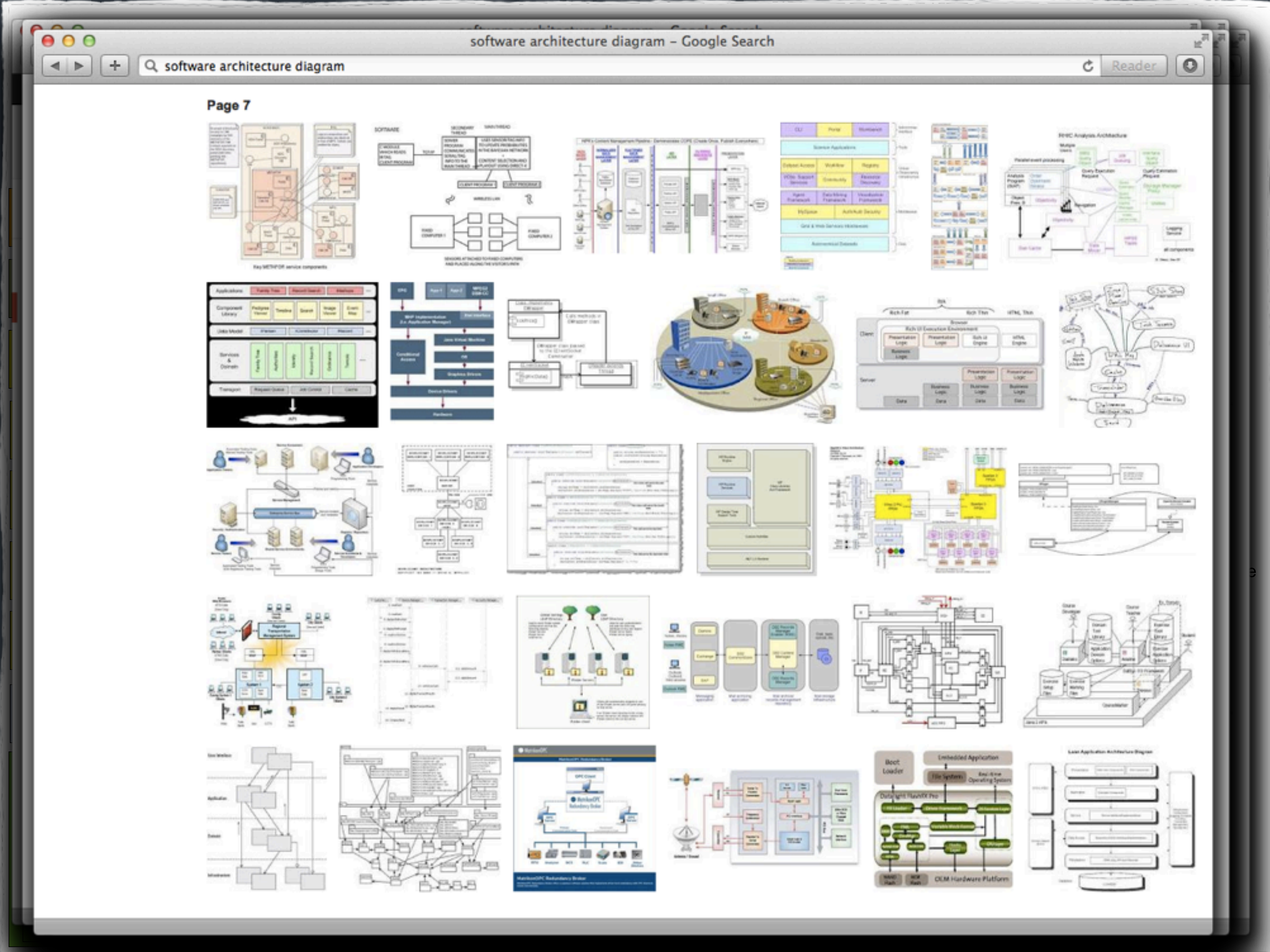
Boxes & no Lines



Eh?



The Airline Route Map


software architecture diagram – Google Search
software architecture diagram
Reader
Page 7

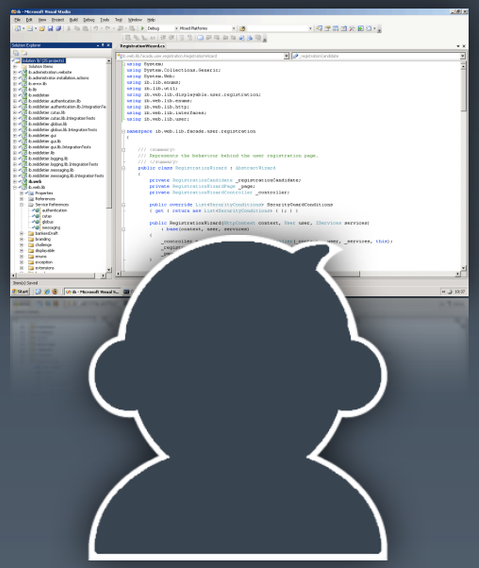
Coffee break challenge #1

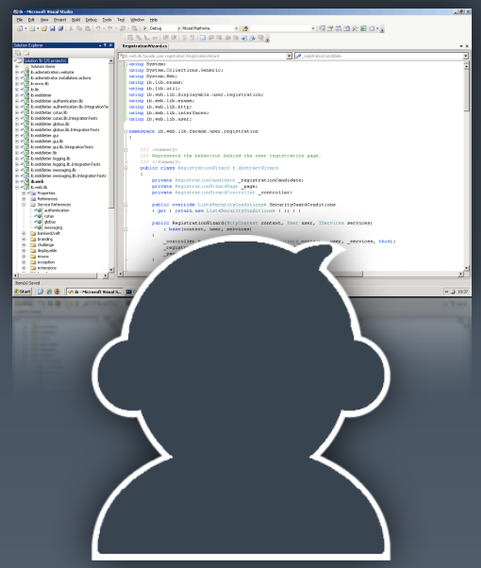
Can your team draw one or more diagrams to visualise your current software project?

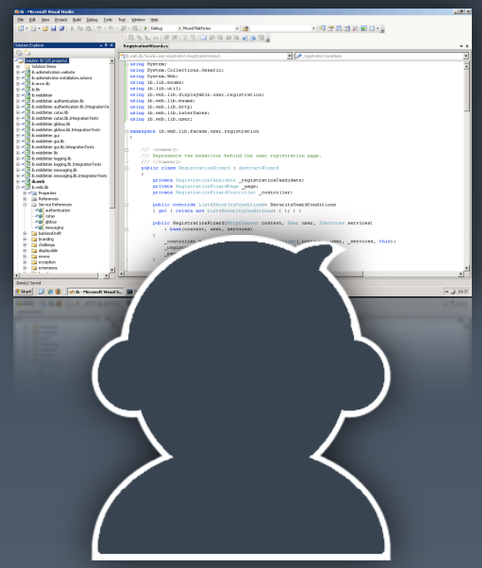
Coffee break challenge #2

Can your team explain how they would design a new software system?

(blank piece of paper + requirements -> solution)

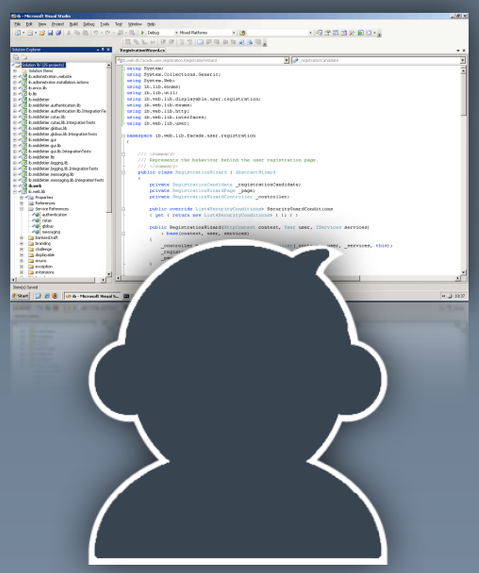


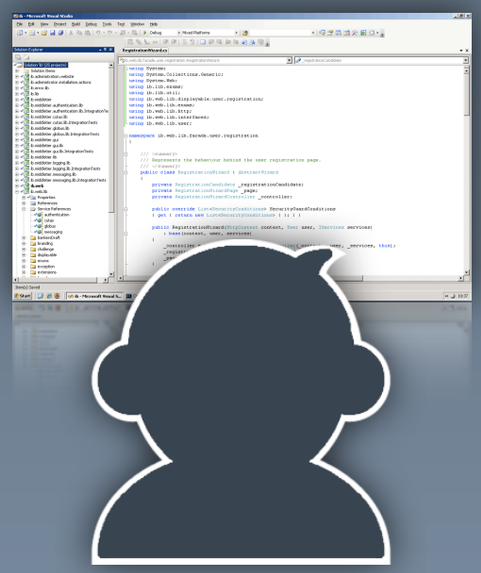


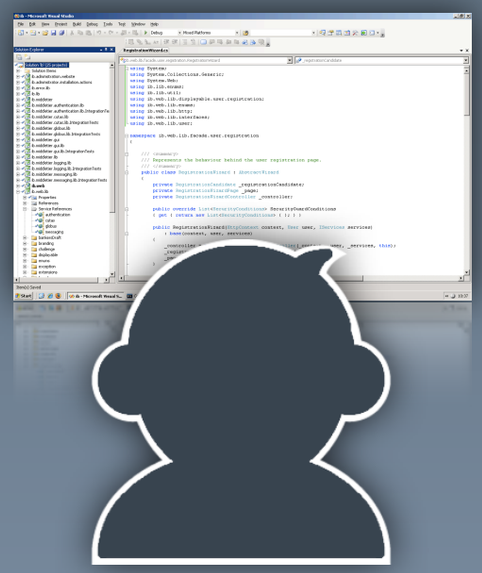


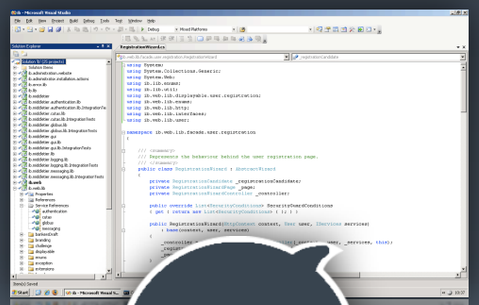
Chaos!

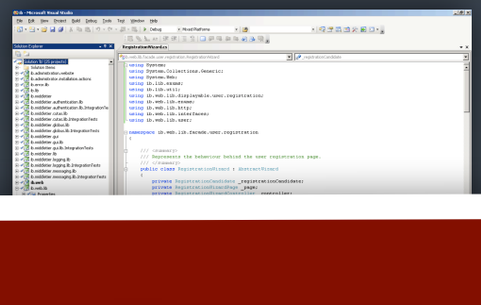
Does the team understand what they are building and how they are building it?

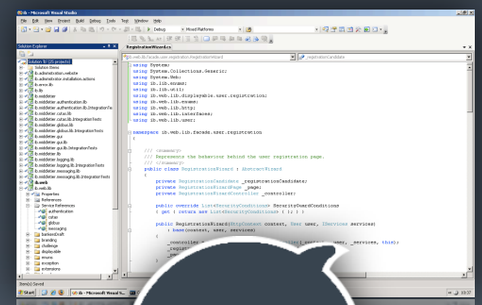








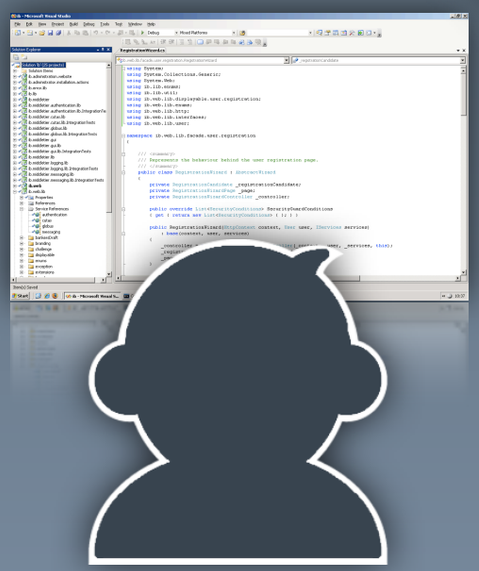


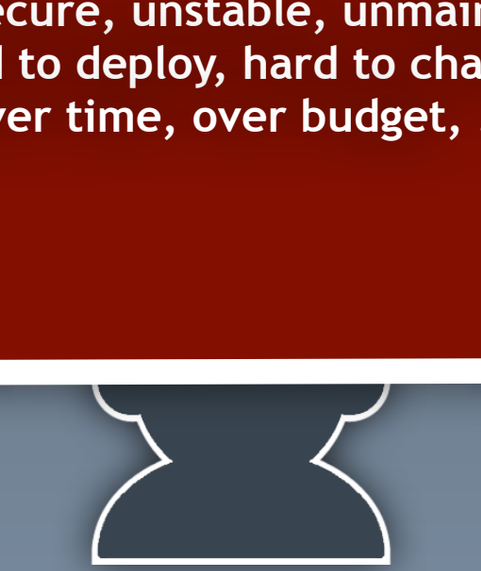


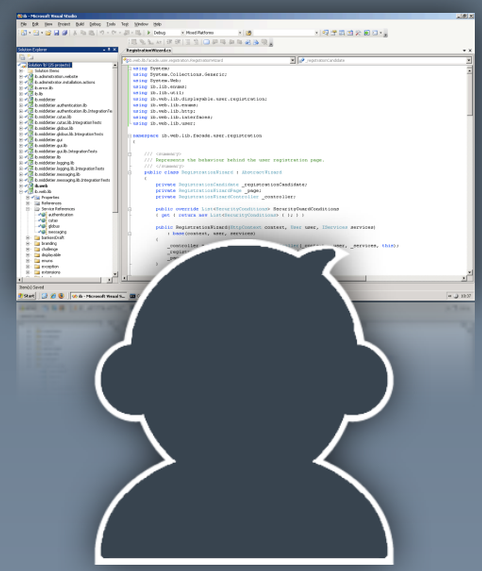
No defined structure,
inconsistent approaches,
big ball of mud,
spaghetti code, ...

STOP

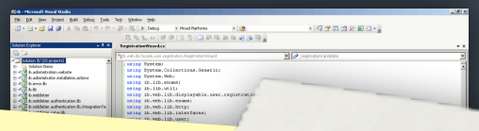
Slow, insecure, unstable, unmaintainable,
hard to deploy, hard to change,
over time, over budget, ...



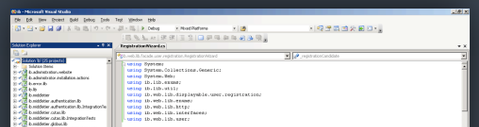




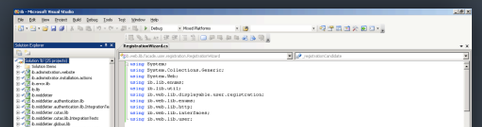
Software architecture in the 21st century



Let`s agree
on some things



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

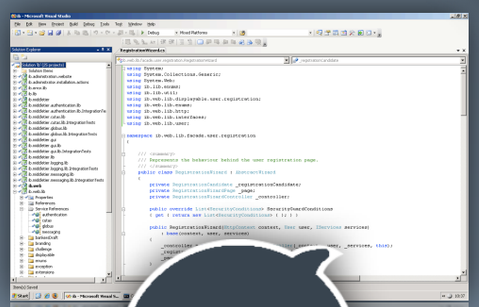


Let`s make the implicit,
explicit

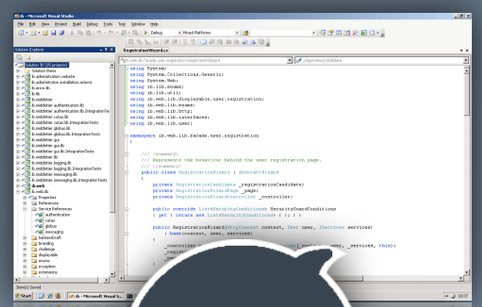
STOP

Doe g it?

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



Put some boundaries
and guidelines in place



A difference 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

Breadth

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

Awareness of options and trade-offs

Depth

Deep hands-on technology skills and knowledge

Good software architects are master-builders

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

A difference 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”

Sounds like something
an agile approach
would advocate... :-)

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

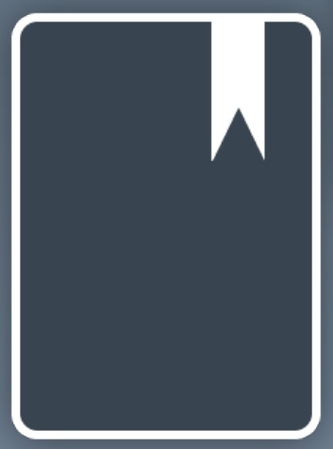
Try this with your team...

Does your software project have tricky

non-functional requirements

or environmental constraints?

Base your architecture on requirements, travel light and prove your architecture with concrete experiments.



Scott Ambler
http://www.agilemodeling.com/essays/agileArchitecture.htm

You need to

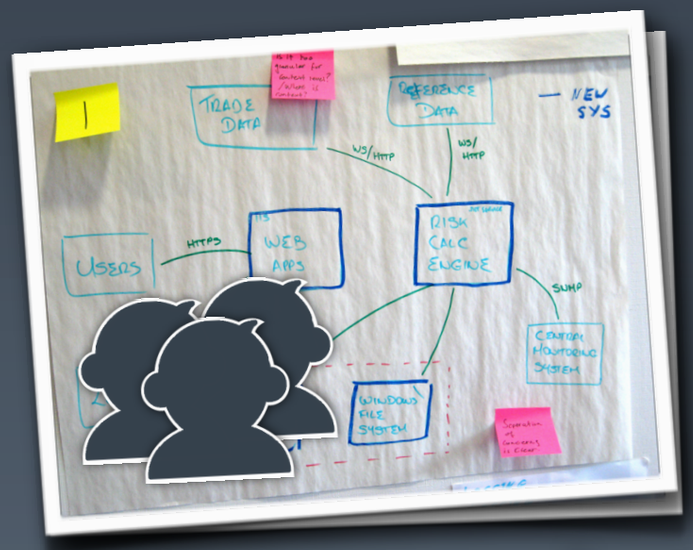
identify and mitigate your highest priority risks

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

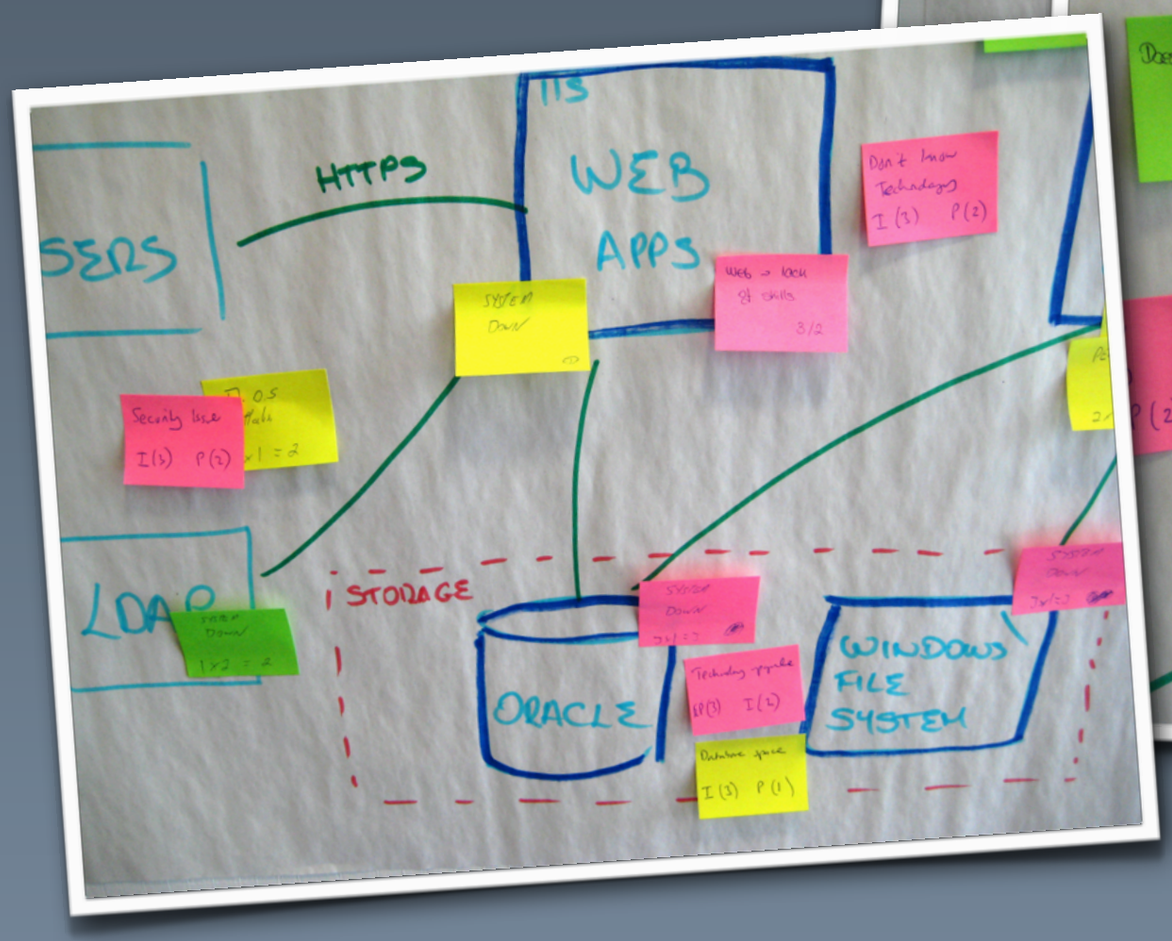
Agile software projects do have risks, right? :-)

Probability

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



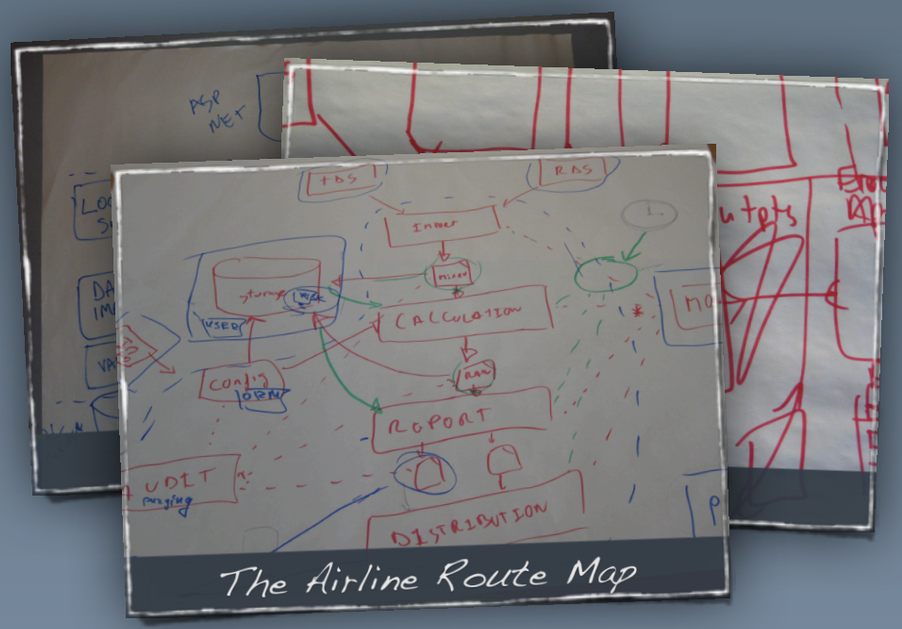
Risk-storming

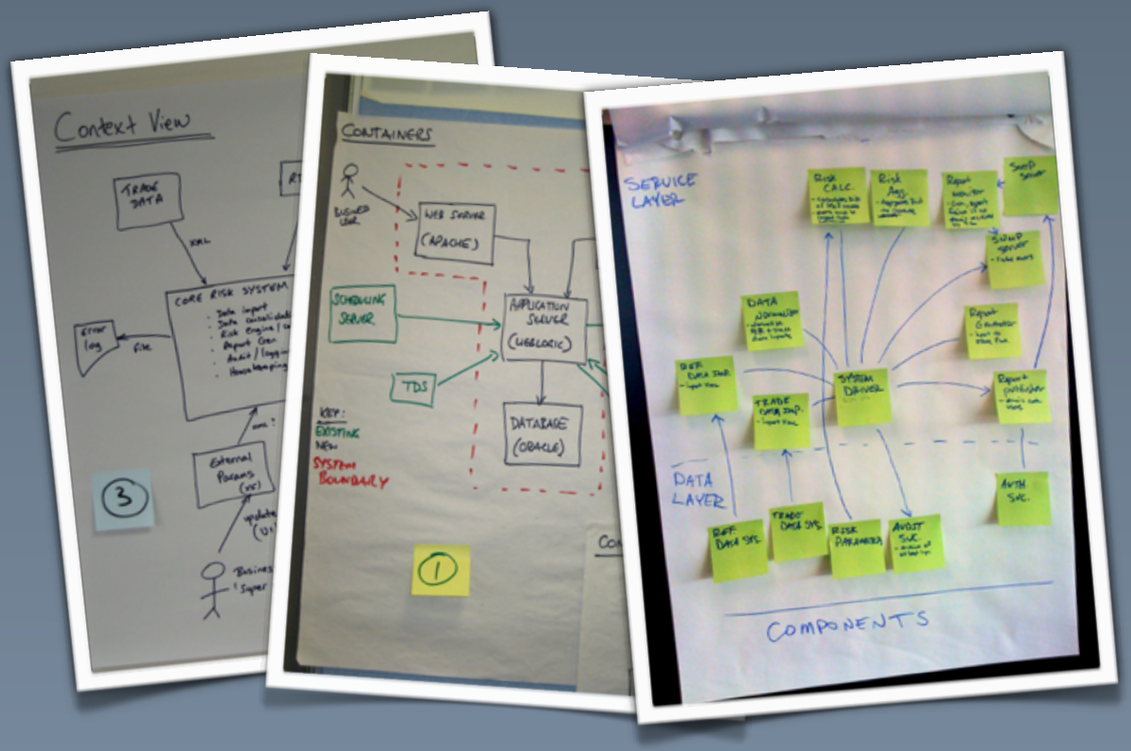


A collaborative and visual technique for identifying risk

Moving fast requires

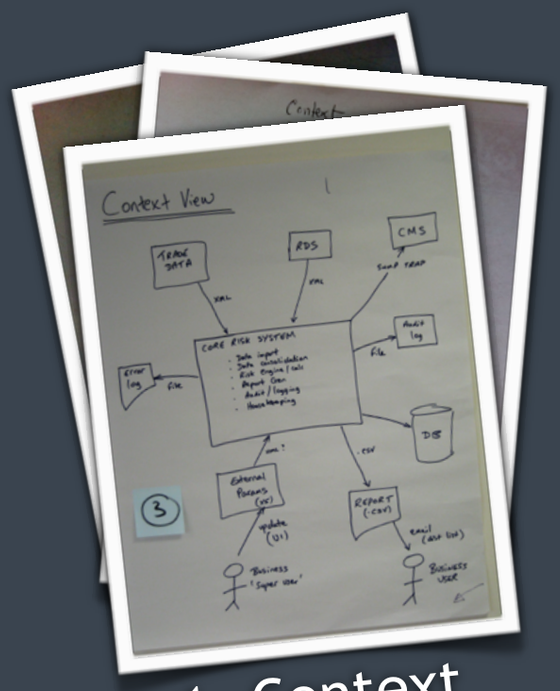
good communication



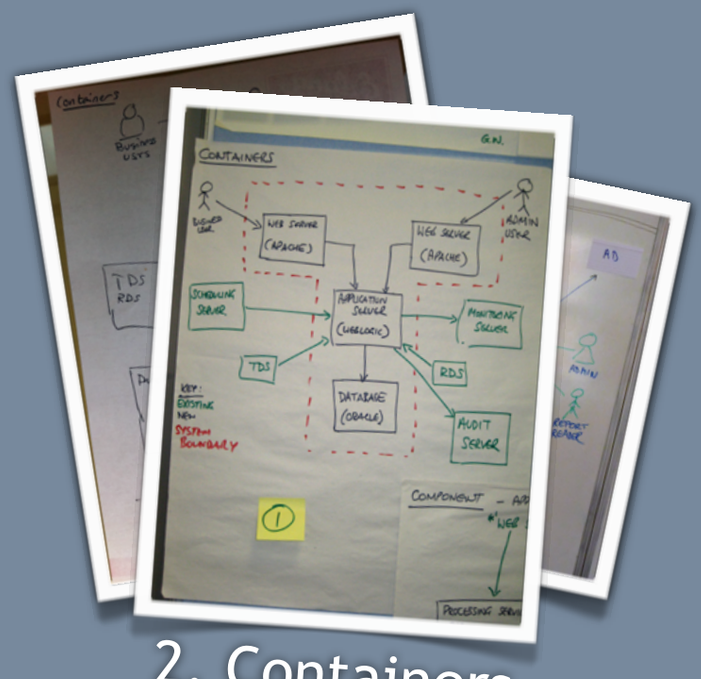


Software System
└ Containers
└ Components
└ Classes

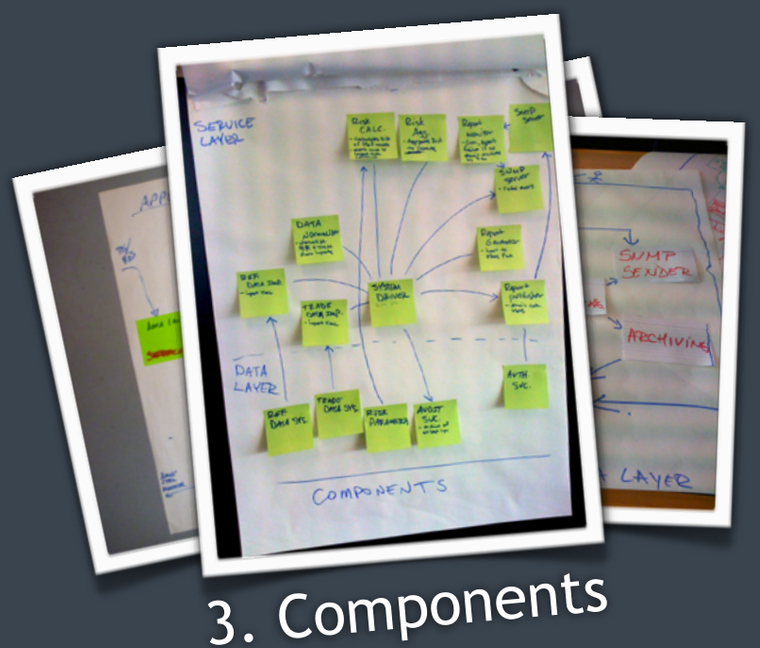
Agree on a simple abstraction that the whole team can use to communicate



1. Context



2. Containers



3. Components

C4

- Context
- Containers
- Components
- Classes

This only covers
the static structure
(runtime, infrastructure,
deployment, etc are also important)

4. Classes
(optionally)

Thinking inside the box

The role

21st century software architecture

"just enough"

Understand how the significant elements fit together

Identify and mitigate the key risks

Provide firm foundations and a vision to move forward

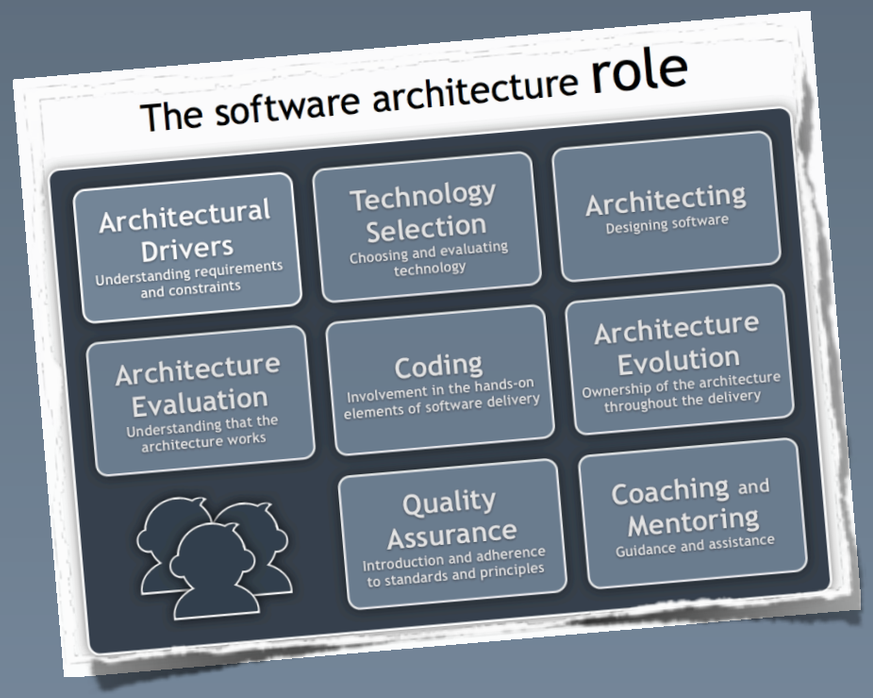
The process

```
/// <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**

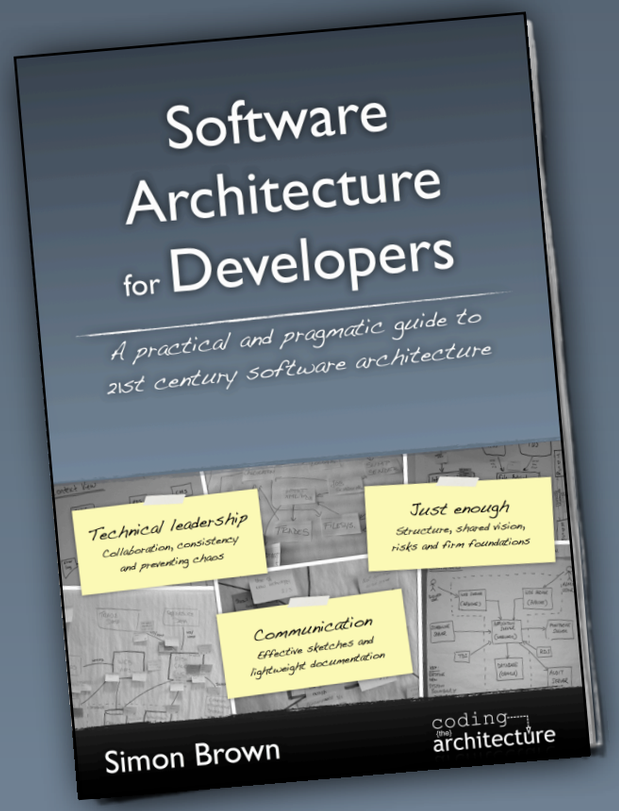
of the **jigsaw**?



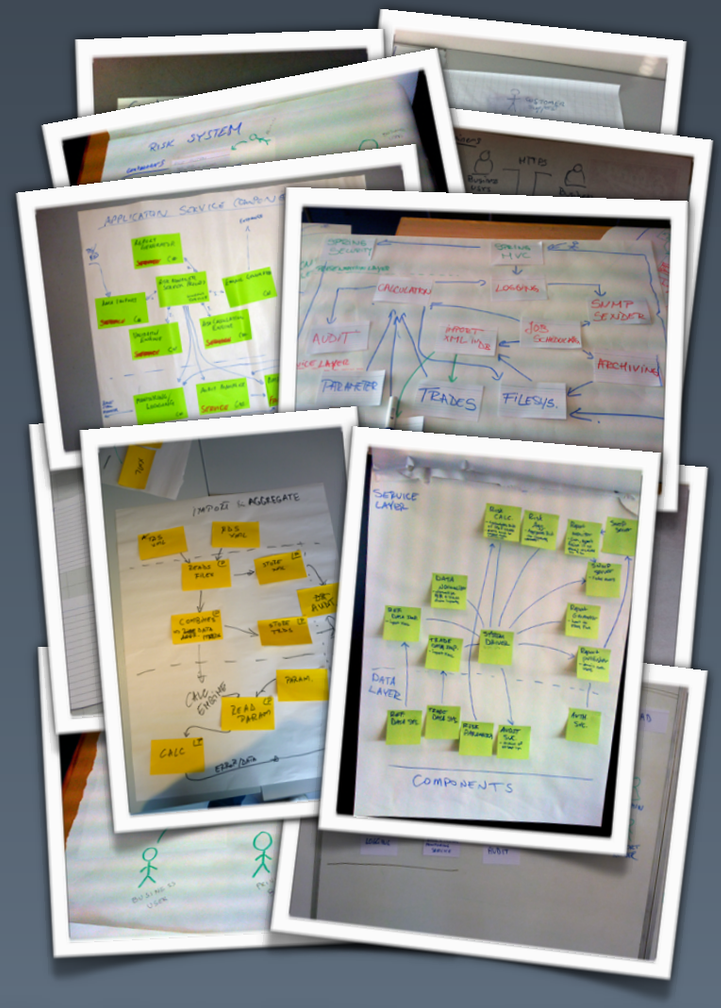
Do whatever works for

you

codingthearchitecture.com



Leanpub leanpub.com



On-site training and consulting

simon.brown@codingthearchitecture.com
@simonbrown on Twitter