

coding
{the}
architecture

The Frustrated Architect

Join the conversation
#skillsmatter



simon.brown@codingthearchitecture.com

@simonbrown on Twitter

Hands-on Software Development

Training, Coaching
& Consulting

Perceptions

Big up front design
and analysis paralysis

Waterfall

UML

I'm a

software
architect



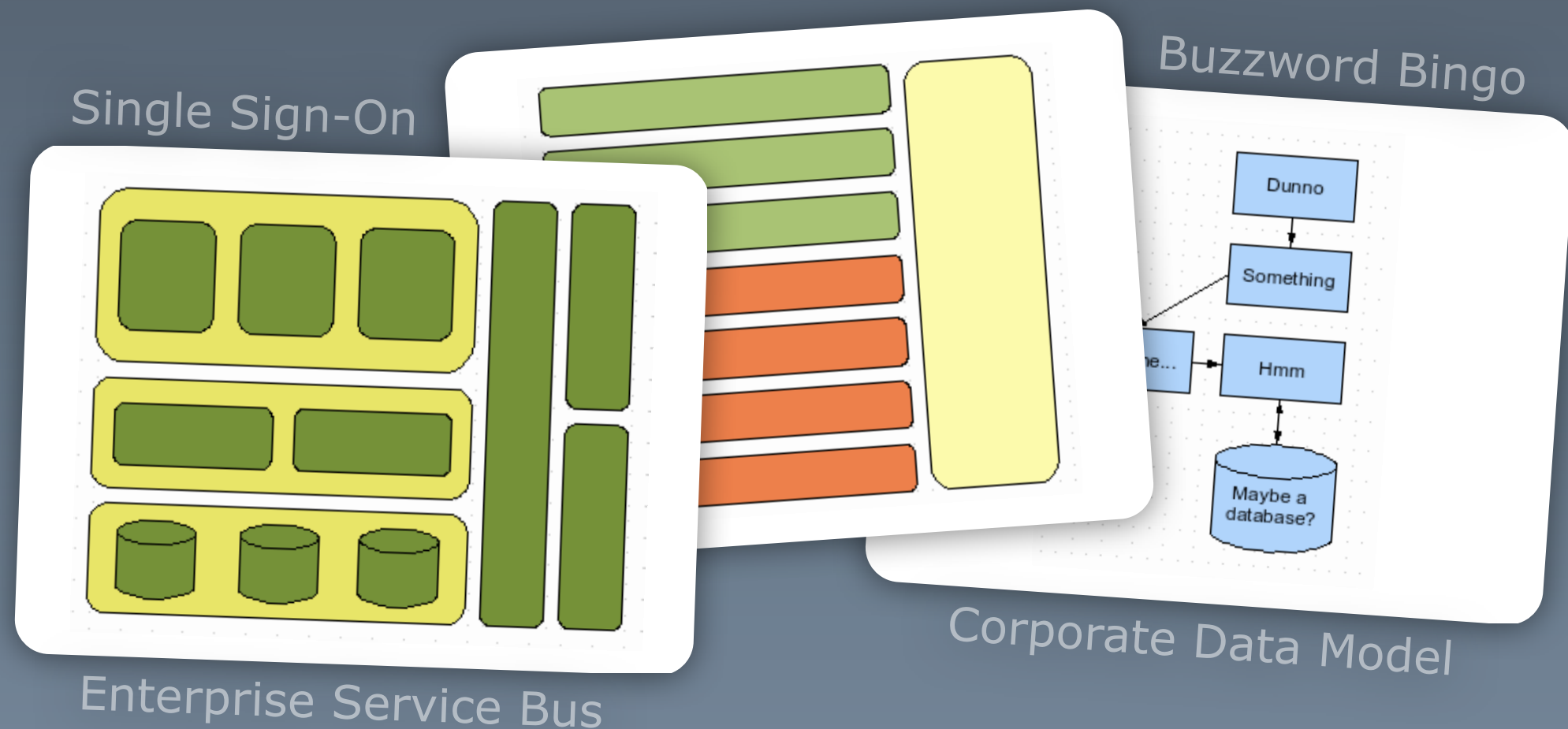
Ivory Tower

PowerPoint Architect

Architecture Astronaut

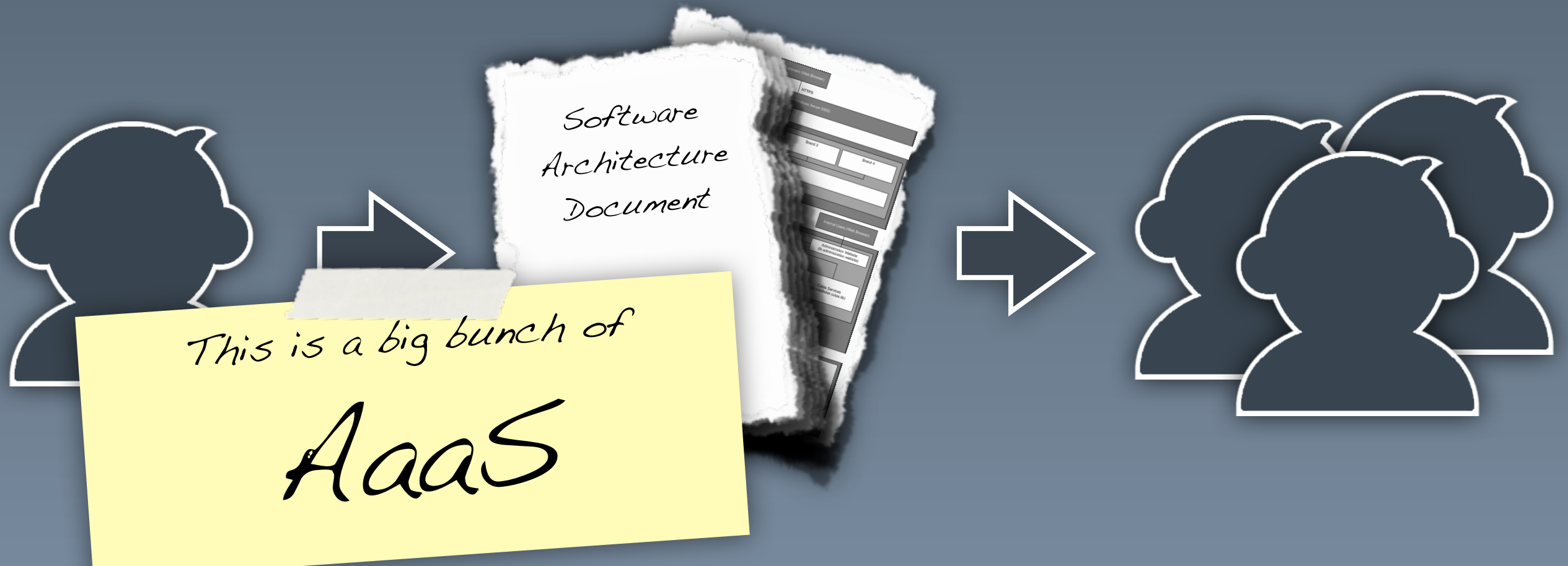
PowerPoint

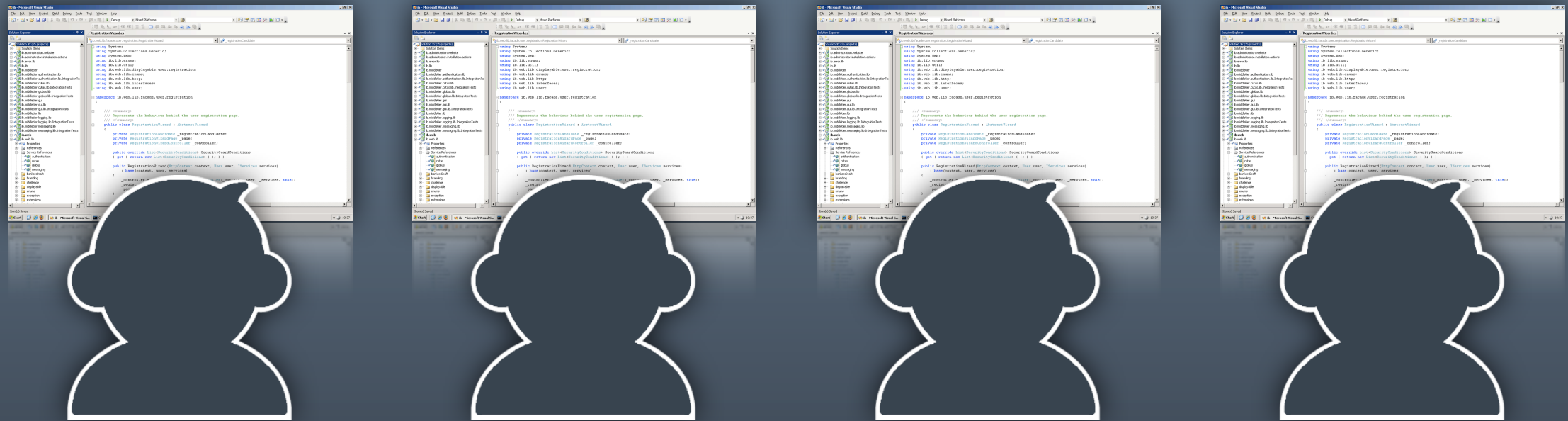
Architecture



yes, I know ... but I'm not using PowerPoint :-P

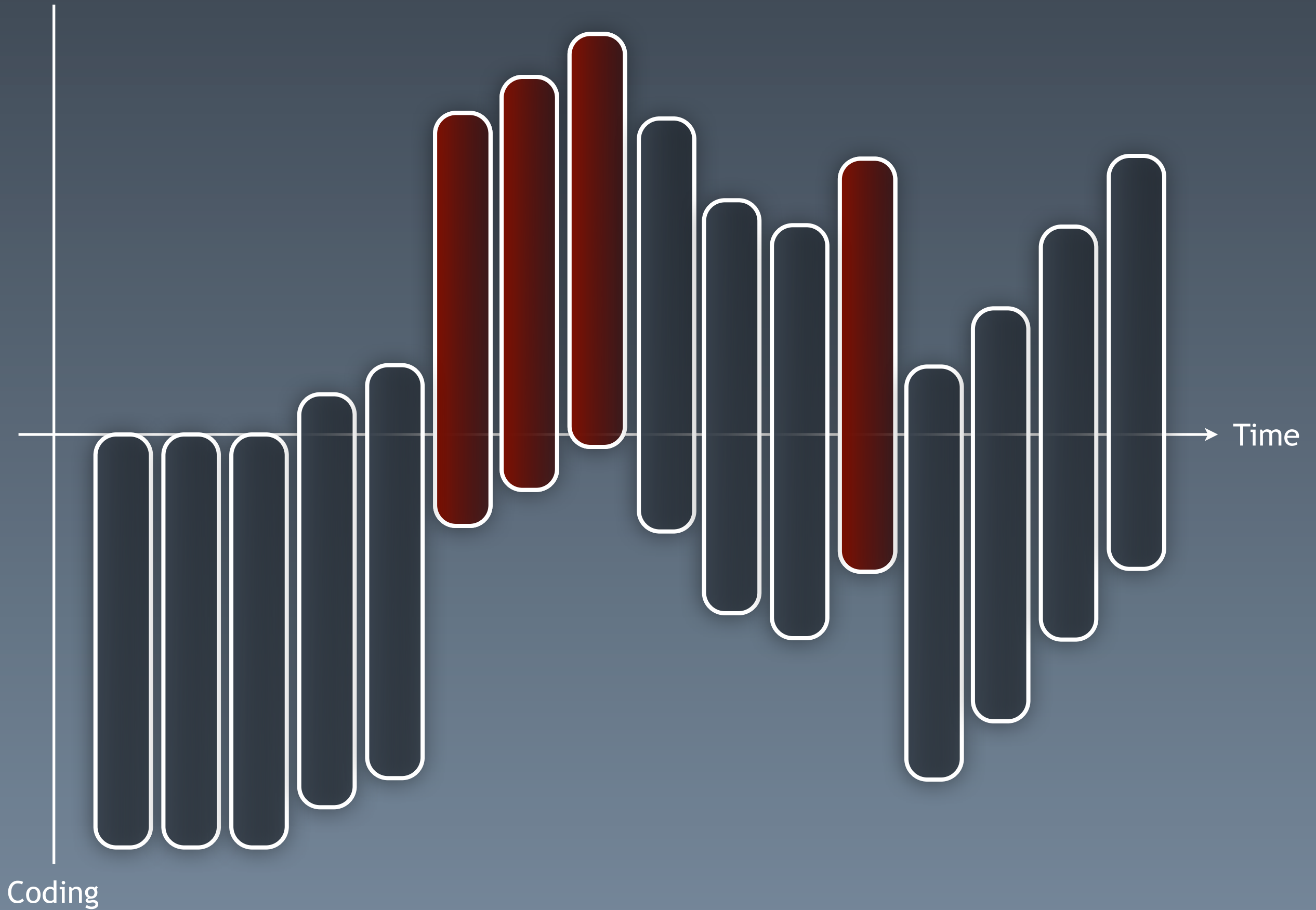
Relay Sport Architecture



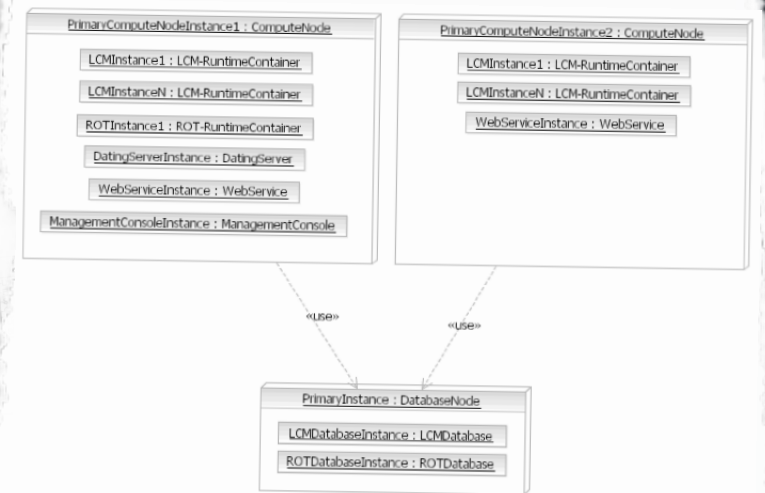
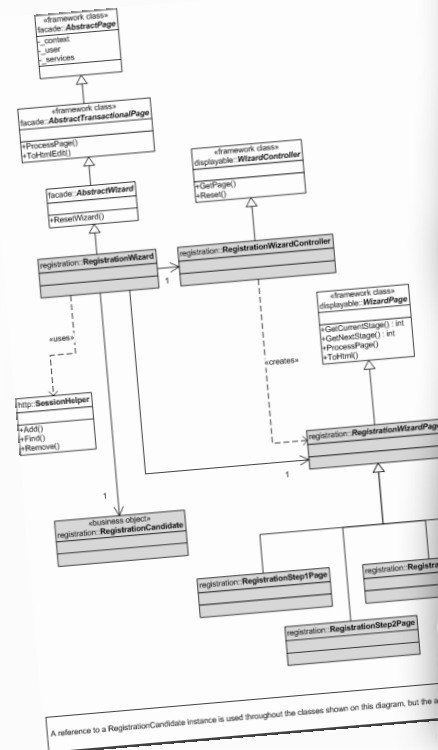
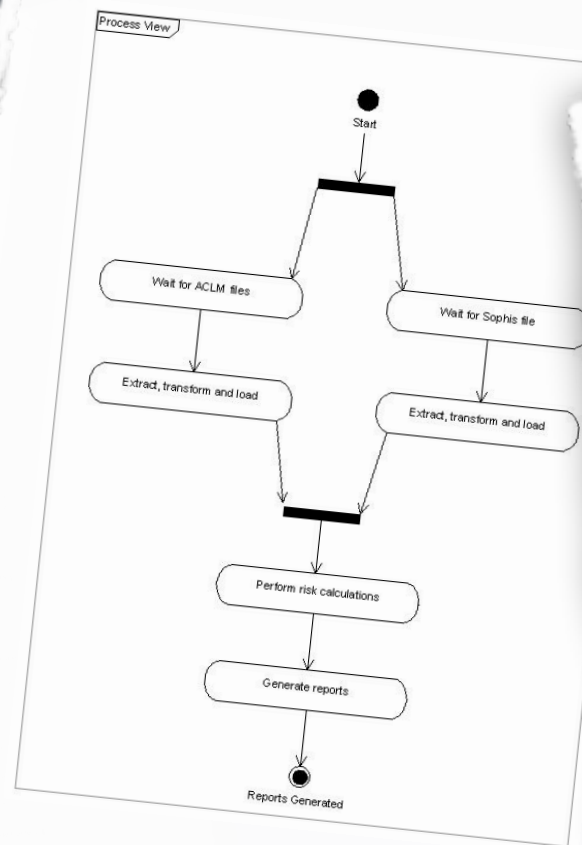
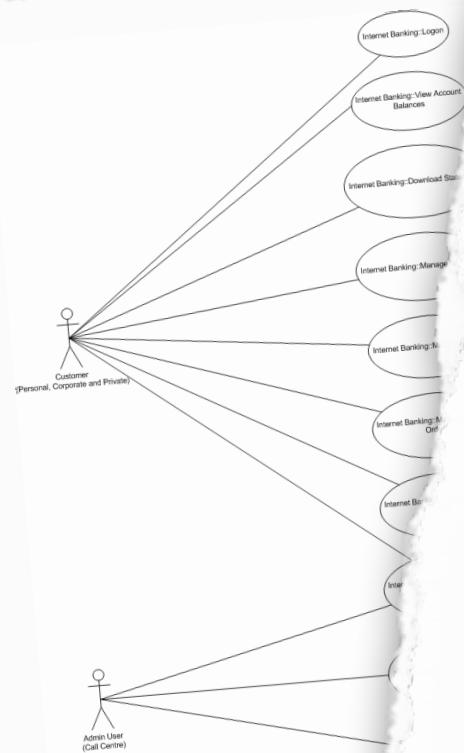


I'm another member of the team
(and I like writing code)

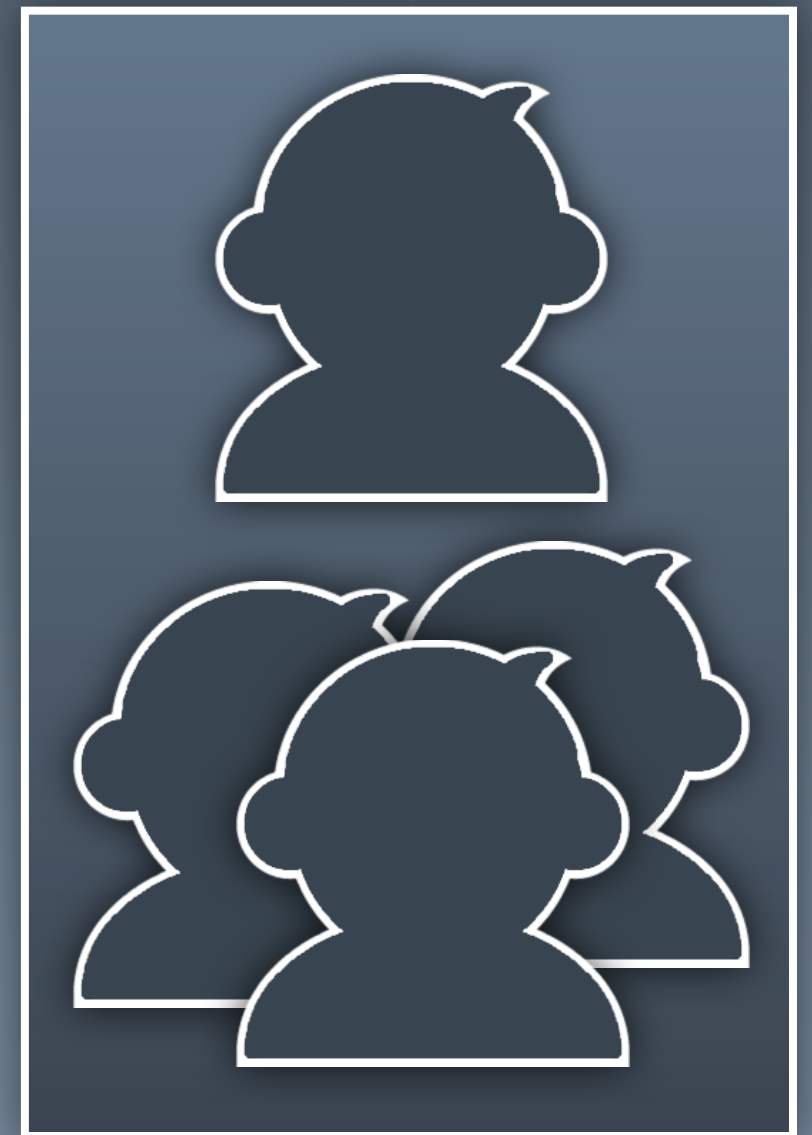
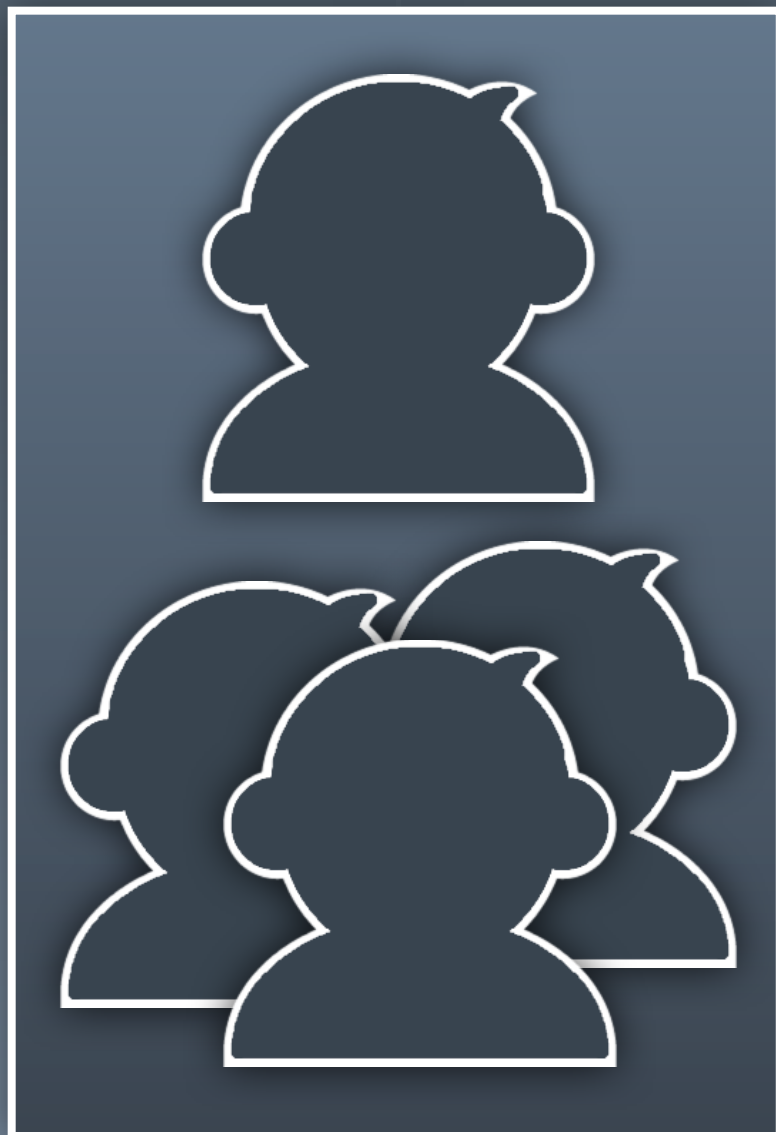
“Architecture”



The UML phase



The Management Consulting phase



Non-technical

The "corporate ladder"



Technical



Our tech lead and mentor
has been "promoted" ...

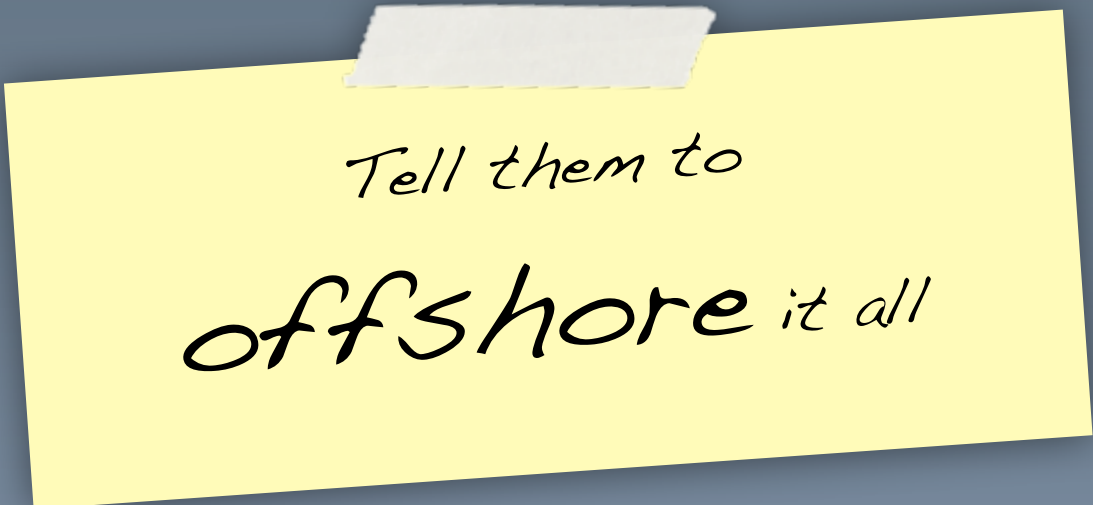
help!

Your management thinks

coding

is a

commodity?



*Tell them to
offshore it all*

coding {the} architecture



<http://www.codingthearchitecture.com>

“Software Architect”

is **not** an

organisational rank

It's a role that you

evolve into

We aspire to be *agile*
and *self-organising*



And that's cool, but aren't you
forgetting something?

Agile

Structure

Self-organising
team

Moving fast,
embracing
change

Test-driven
development

Security

Non-functional
requirements

Availability

Continuous
delivery

Automated
acceptance
testing

Agile

Technical guidance

Performance

Vision

Emergent
design

Kanban

Technical quality

Retrospectives

Scalability

Lean

Agile

*I don't position most of
my content as agile, but...*

What is an

agile architect

anyway?



Evolutionary Architecture
and Emergent Design

Defer until the
last responsible
moment

YAGNI

*I'm an agile
architect!*



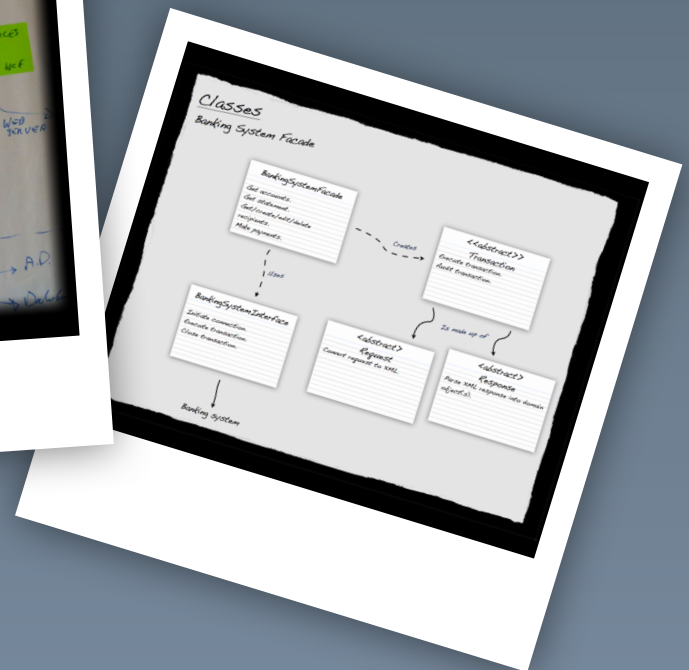
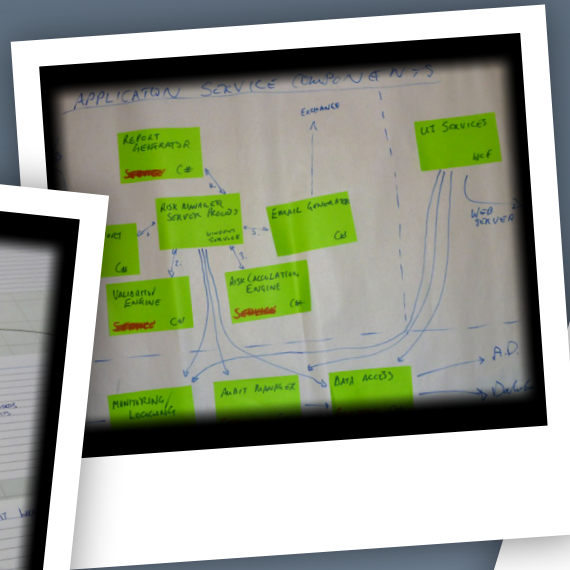
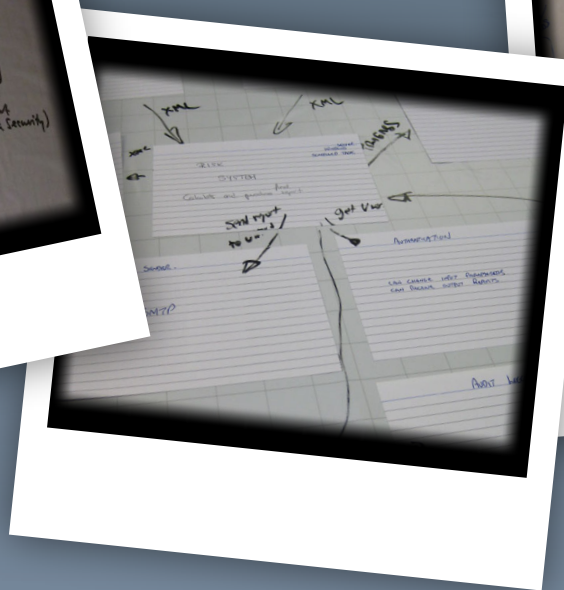
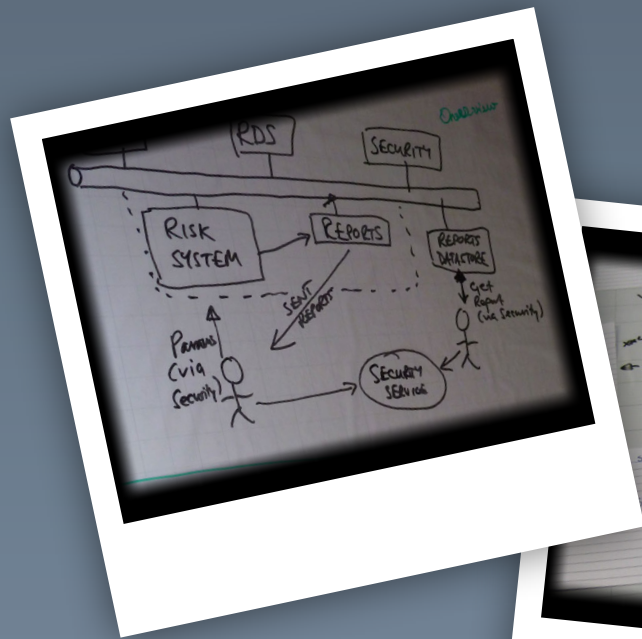
System Metaphor

Refactoring

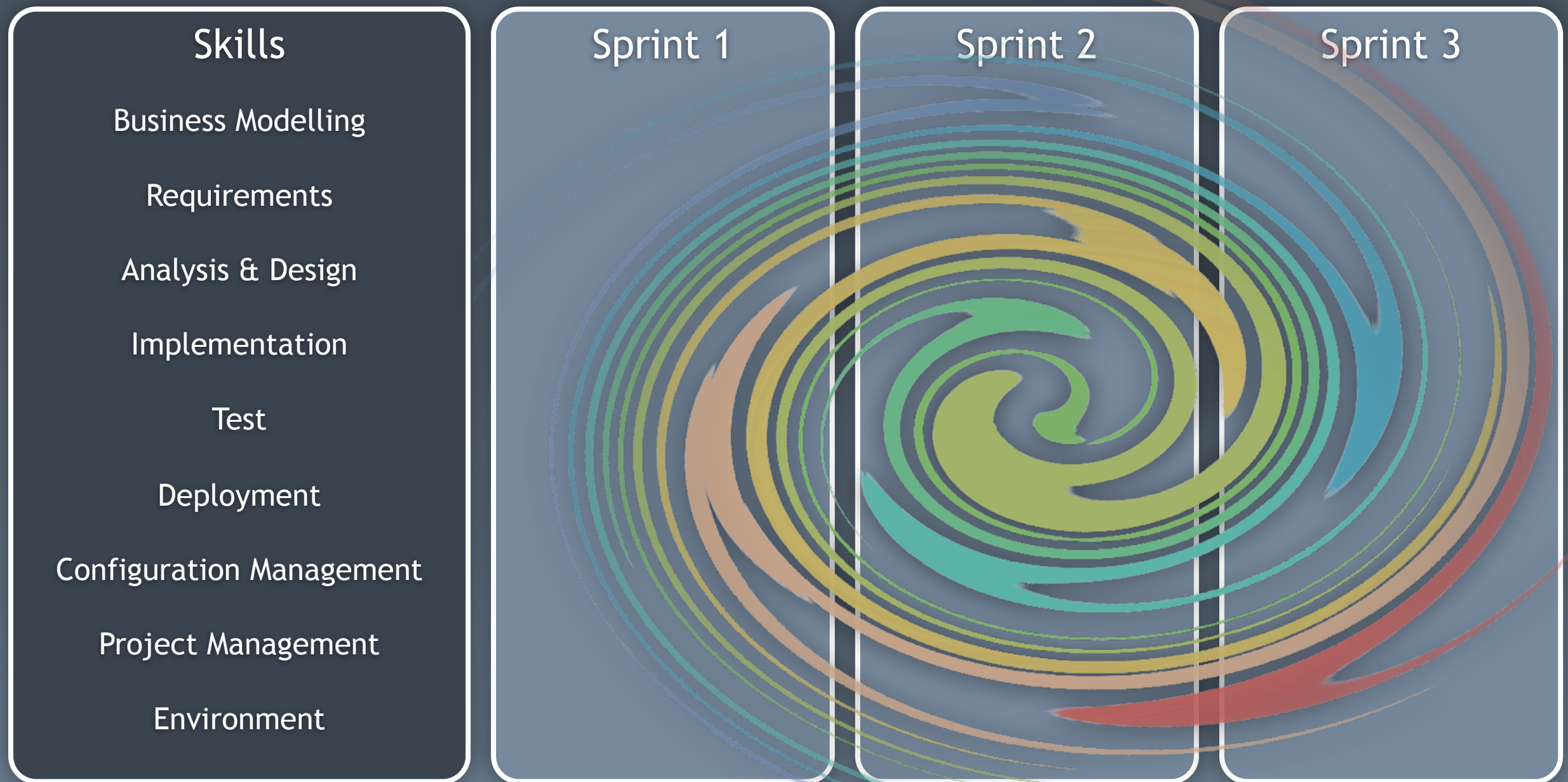
Spikes, stripes and tracers

Agile

architecture?



Agile (e.g. Scrum)



Evolutionary architecture

Foolishly hoping for the best?

We don't need
software architecture;

we do

TDD



Agile software team

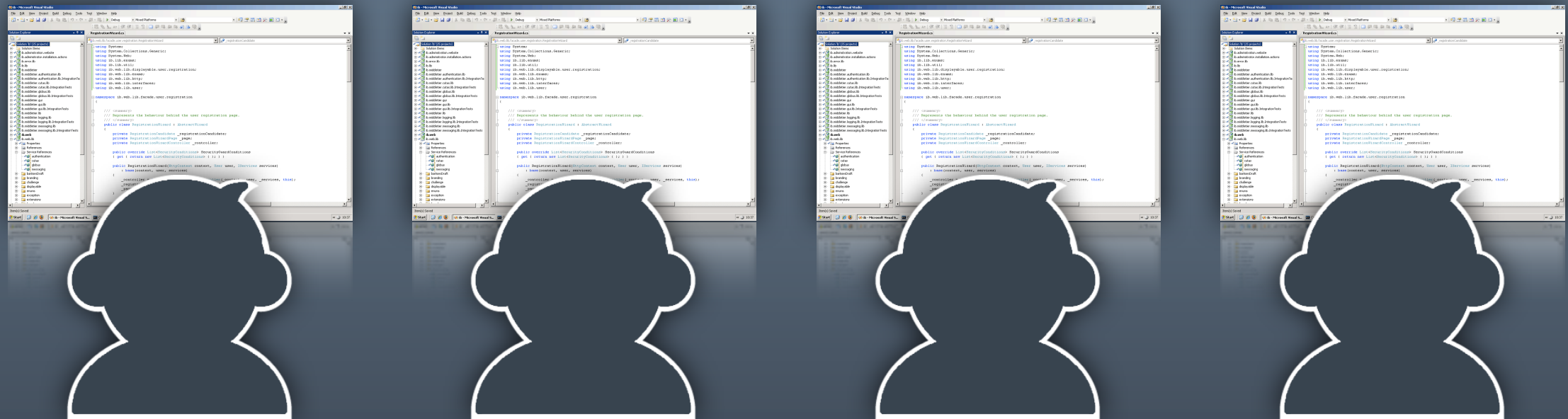
TDD *is about* code

... architecture isn't

(well, it is, but it's also about more than just the code)

Last responsible moment

Most people know roughly
what they're building
so just make some decisions!



Flat, self-organising teams are great but...

...they don't always work

Let's

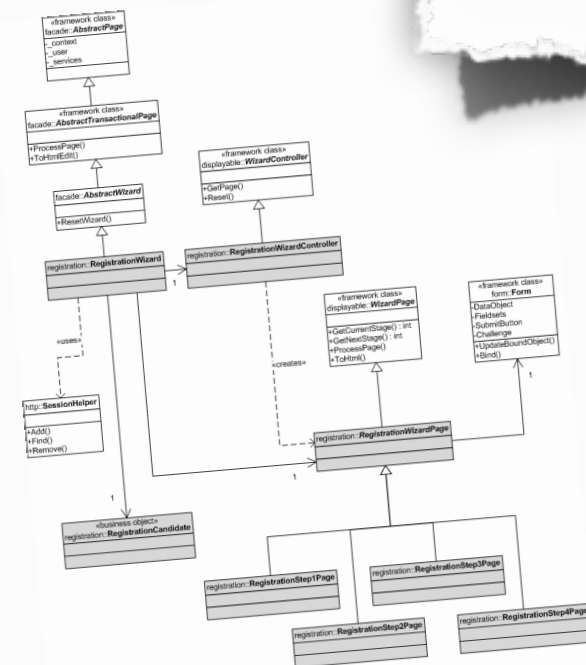
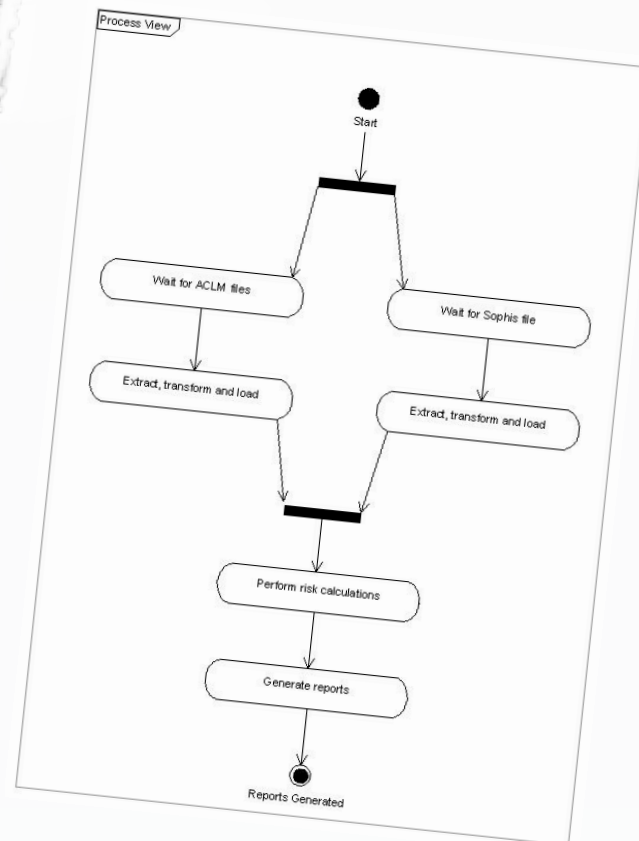
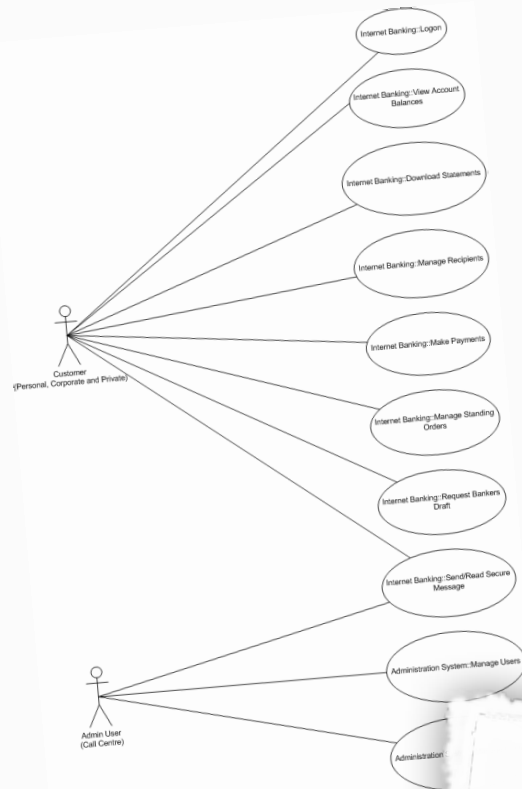
reinvent

something shiny and new...

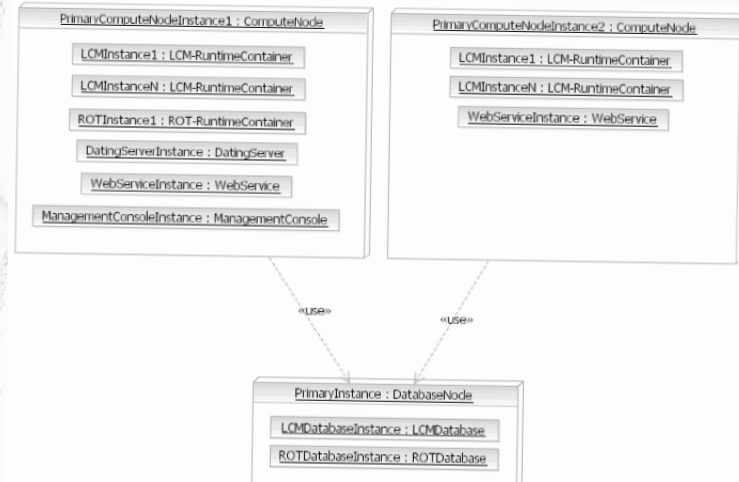
Retrospective

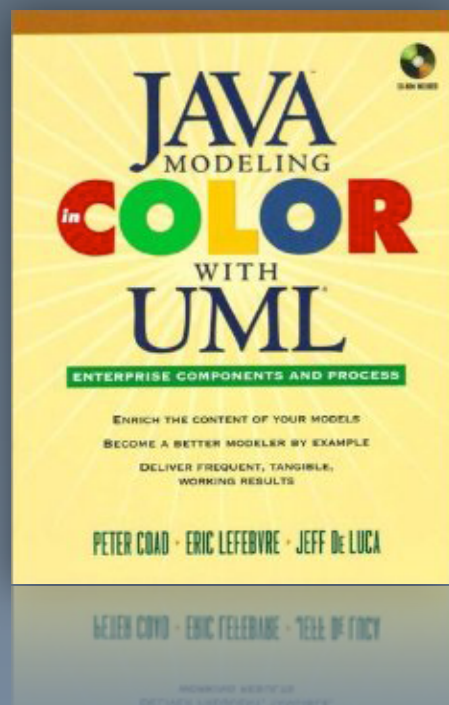
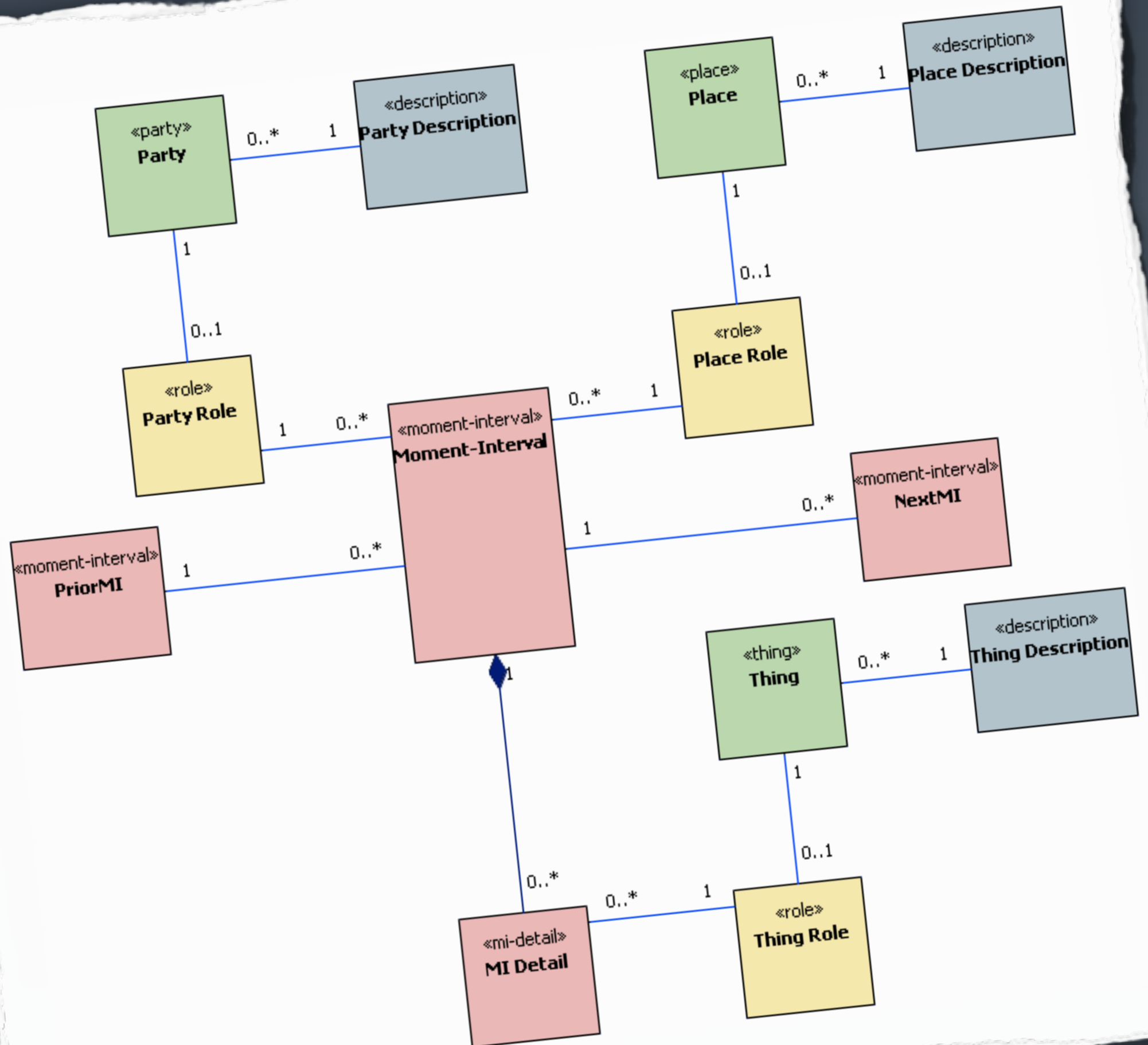
Have we forgotten
more than we've learnt?

UML



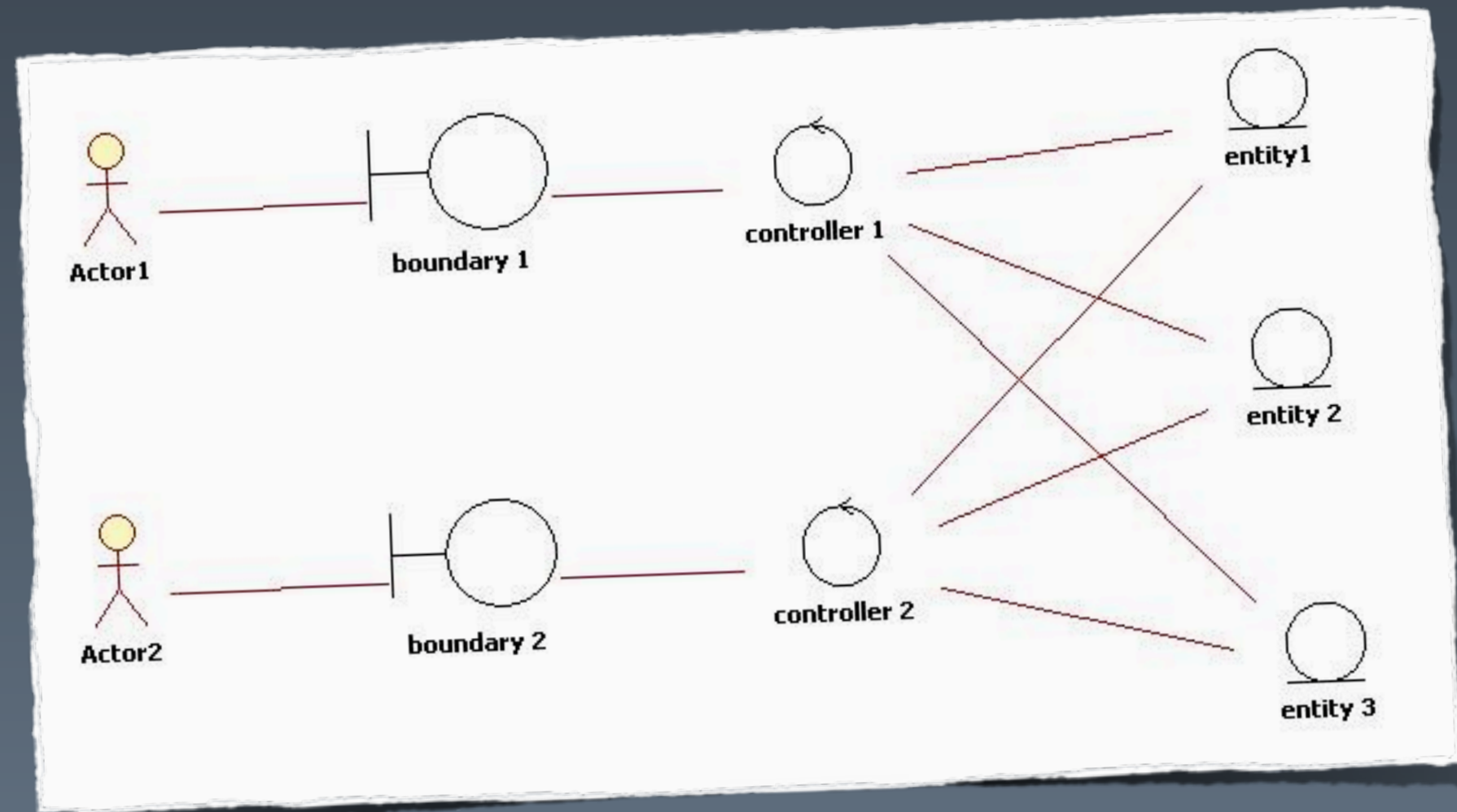
A reference to a RegistrationCandidate instance is used throughout the classes shown on this diagram, but the associations are not shown for brevity.





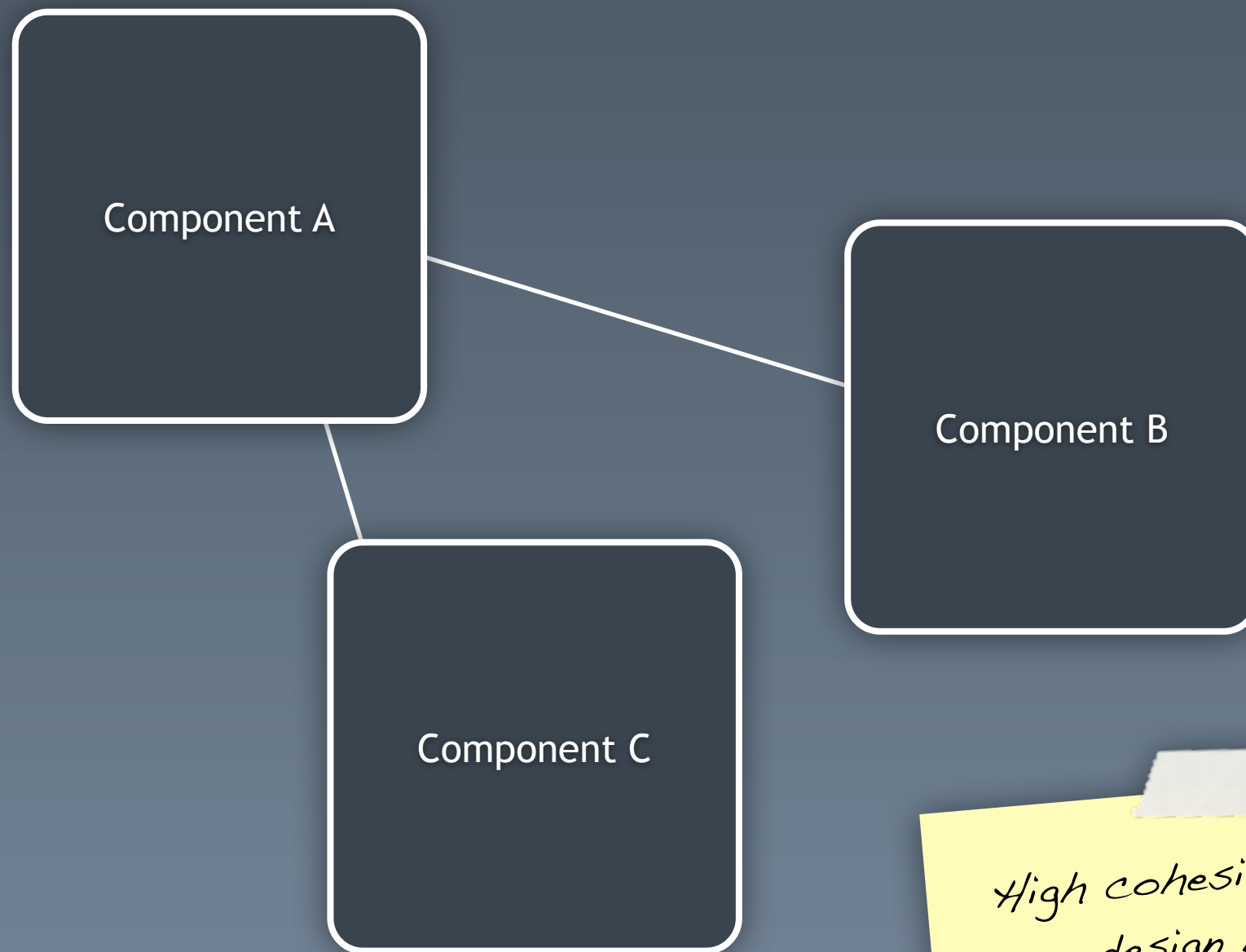
[illegible][illegible][illegible]

Class-Responsibility-Collaboration

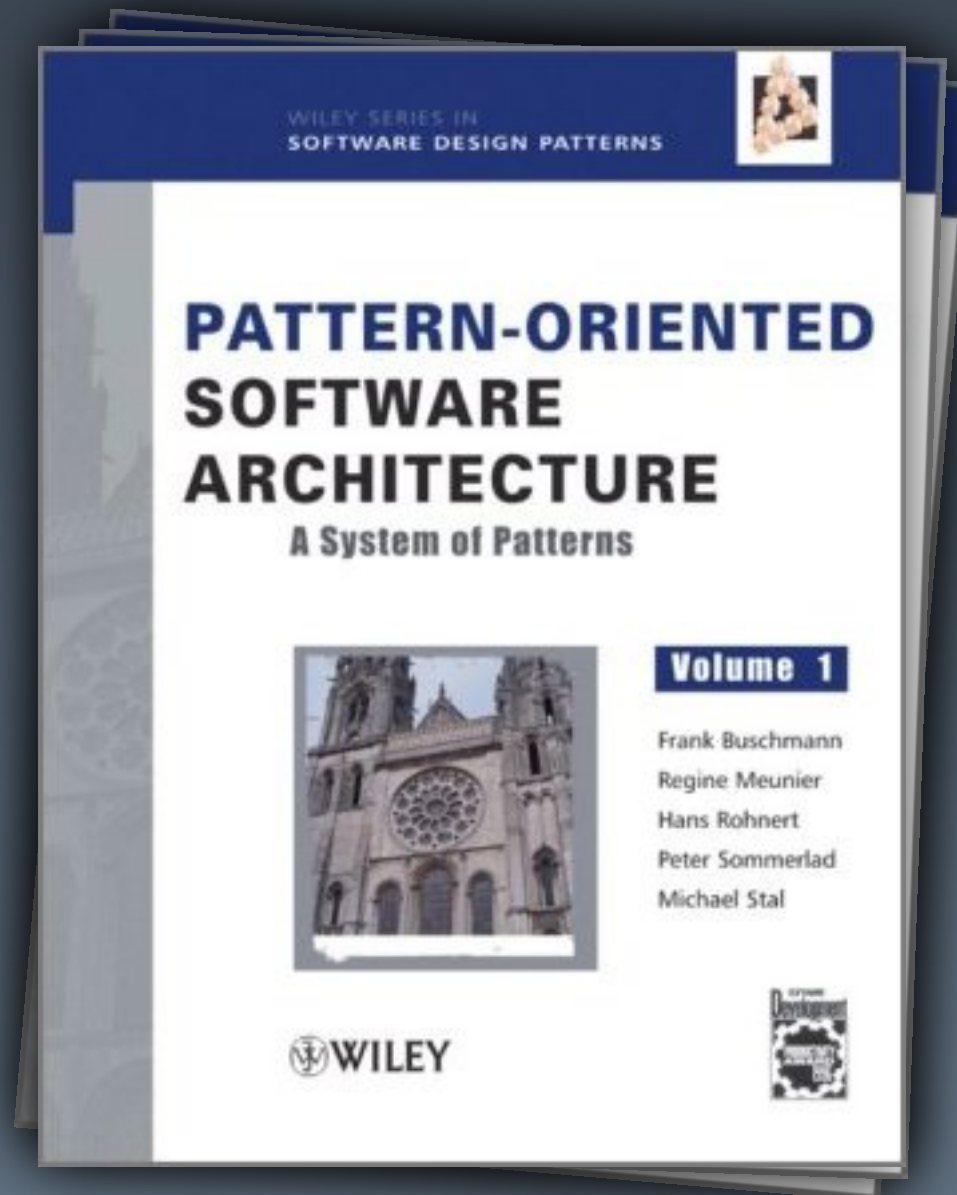


Boundaries, controllers and entities

Component-based development

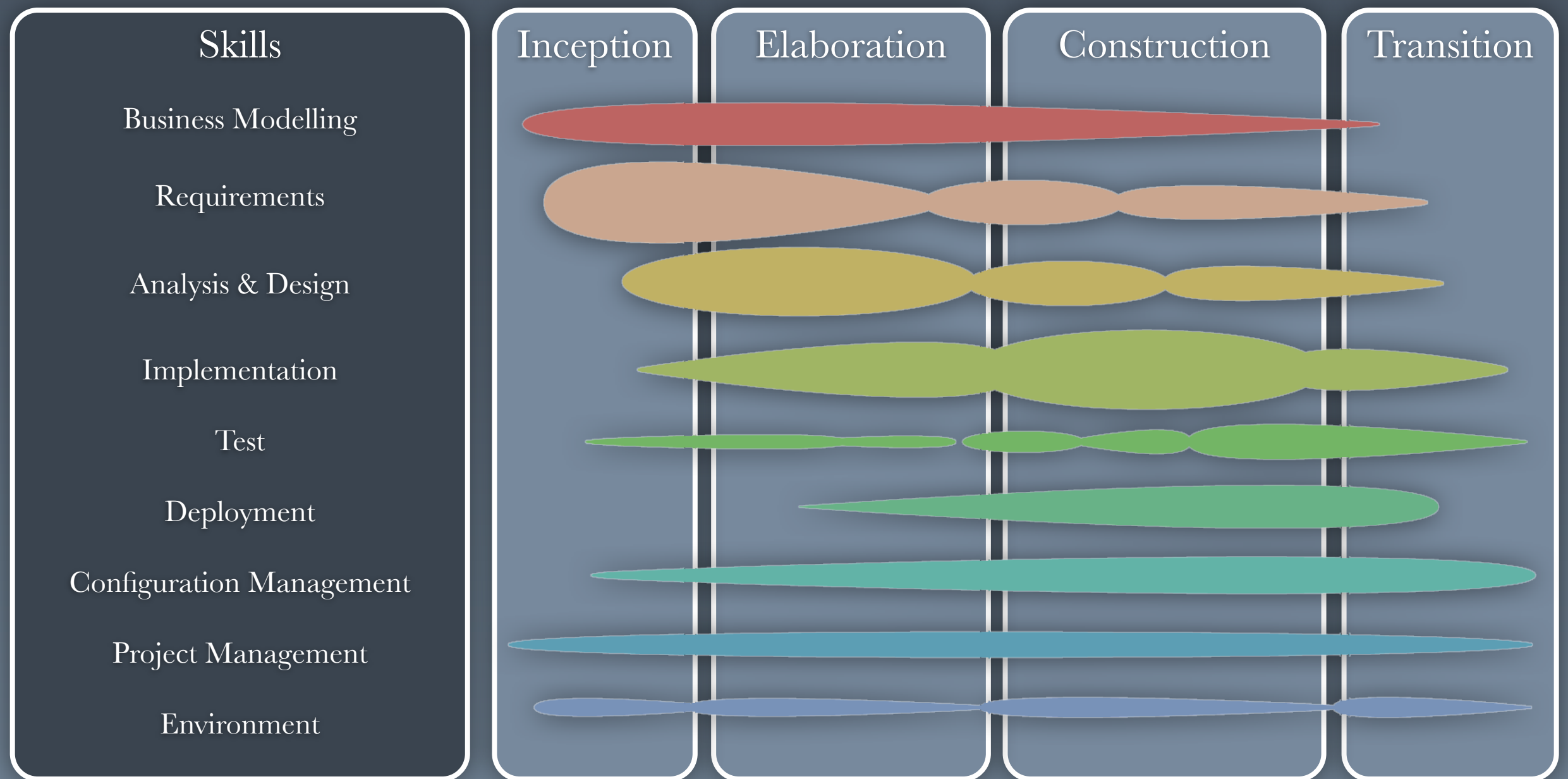


*High cohesion, low coupling
design by contract,
Liskov substitution principle*



Pattern-Oriented Software Architecture

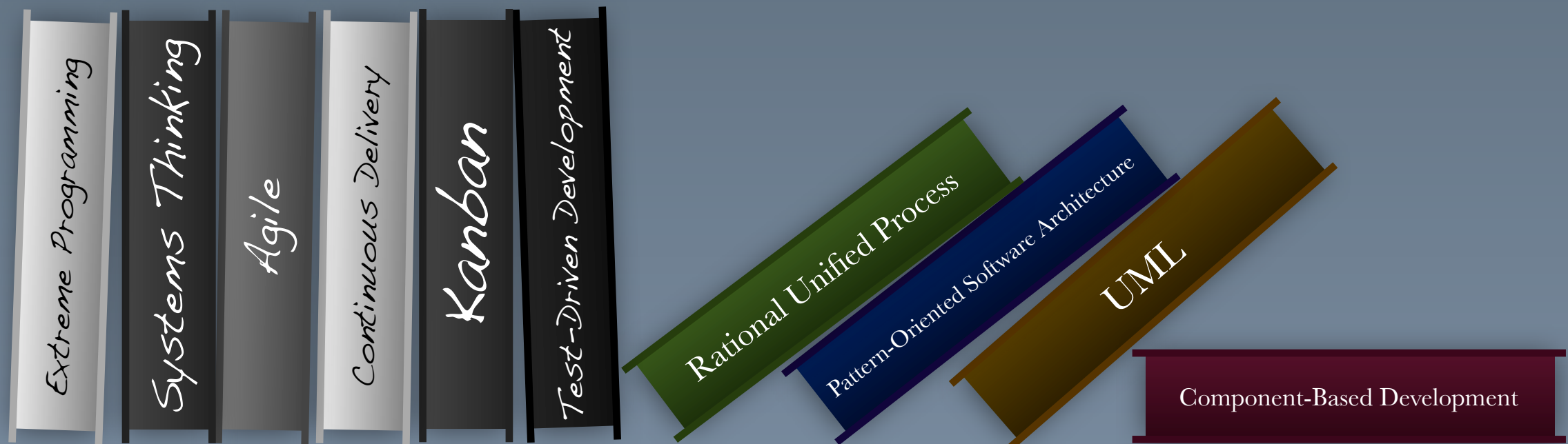
Rational Unified Process (RUP)



Be pragmatic
with this stuff

(if you know it exists, of course)

Who is teaching
the classics of the
pre-*agile* era?



So you **think**
you're an architect

Curriculum Vitae / Resume

Enterprise Architect

A Big Company (2006-date)

I have been responsible for the design and implementation of an enterprise customer solution. I drew some UML diagrams and I wrote some Java code.

I would like a job writing more code please. :-)

Err, no; you're a software architect who just happens to work in a large organisation

(in fact, you're barely a "software architect" either)

Software development is not a
relay sport



Successful software delivery

is **not** an

implementation
detail!

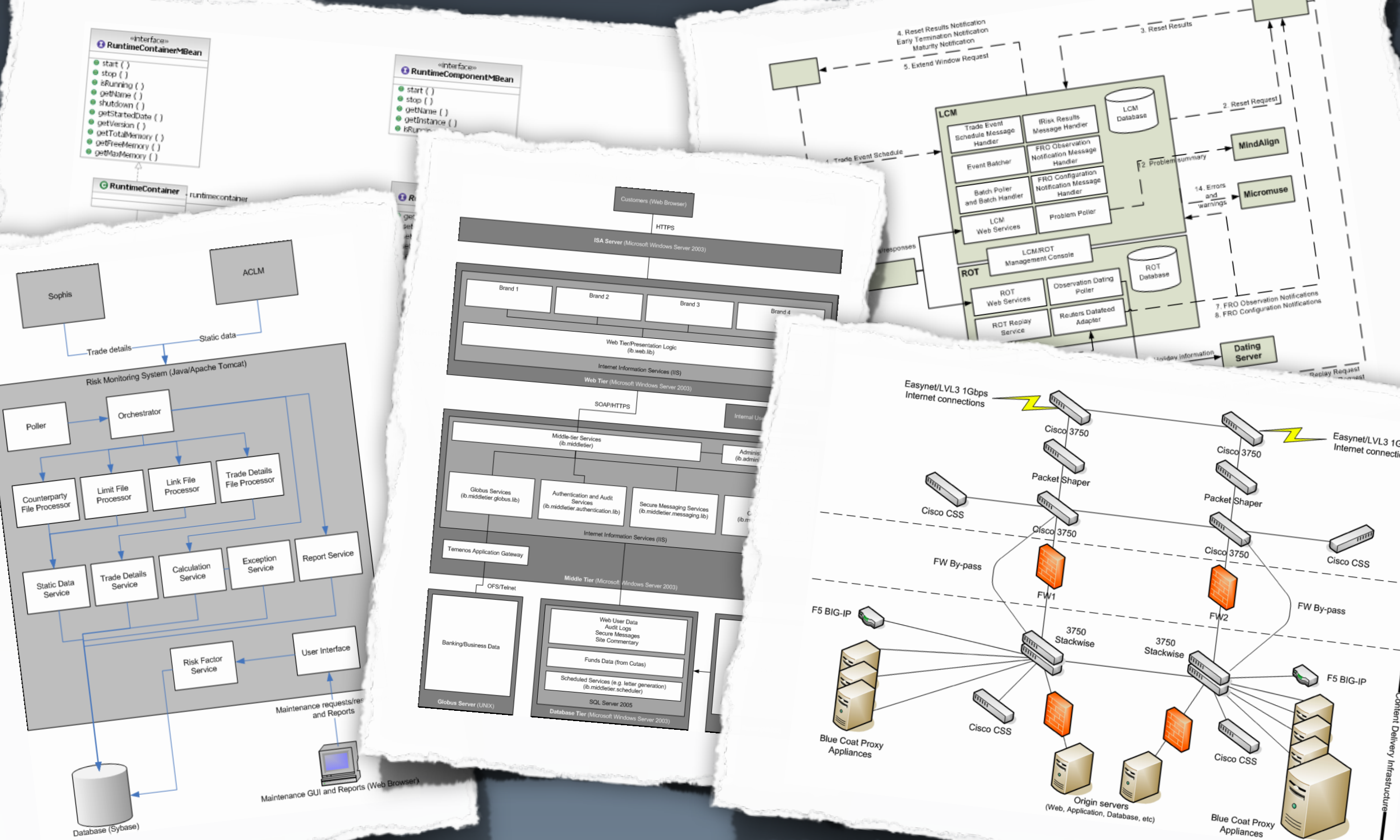
The irresponsible architect

Cross-site scripting attacks possible; weak passwords allowed; HTTP sessions didn't timeout; ...

No non-functional testing (e.g. penetration testing or load testing); ...

Basic functionality errors; little or no quality assurance; rework required late in the project because of assumptions; ...





Foolishly **hoping** for the best?

The irresponsible architect

Cross-site scripting attacks possible; weak passwords allowed; HTTP sessions didn't timeout; ...

No non-functional testing (e.g. penetration testing or load testing); ...

Basic functionality errors; little or no quality assurance; rework required late in the project because of assumptions; ...

No documentation; ...

Oh, did I mention this was supposed to be a "strategic platform"?

Context

What is this all about?

Functional View

What does the system do?

Process View

Does the system implement business processes?

Non-functional View

Are there any significant non-functional requirements influencing the architecture?

Architectural Constraints

Are there any constraints influencing the architecture?

Architectural Principles

Are there any principles influencing the architecture?

Logical View

What does the big picture look like and how is the system structured?

Interface View

Are there internal or external system interfaces?

Design View

Is it clear how system components should be implemented?

Infrastructure View

What does the target deployment environment look like?

Deployment View

How will the system components be deployed onto the target infrastructure?

Operational View

How will people operate and support the system?

Security View

How is security handled across all tiers?

Data View

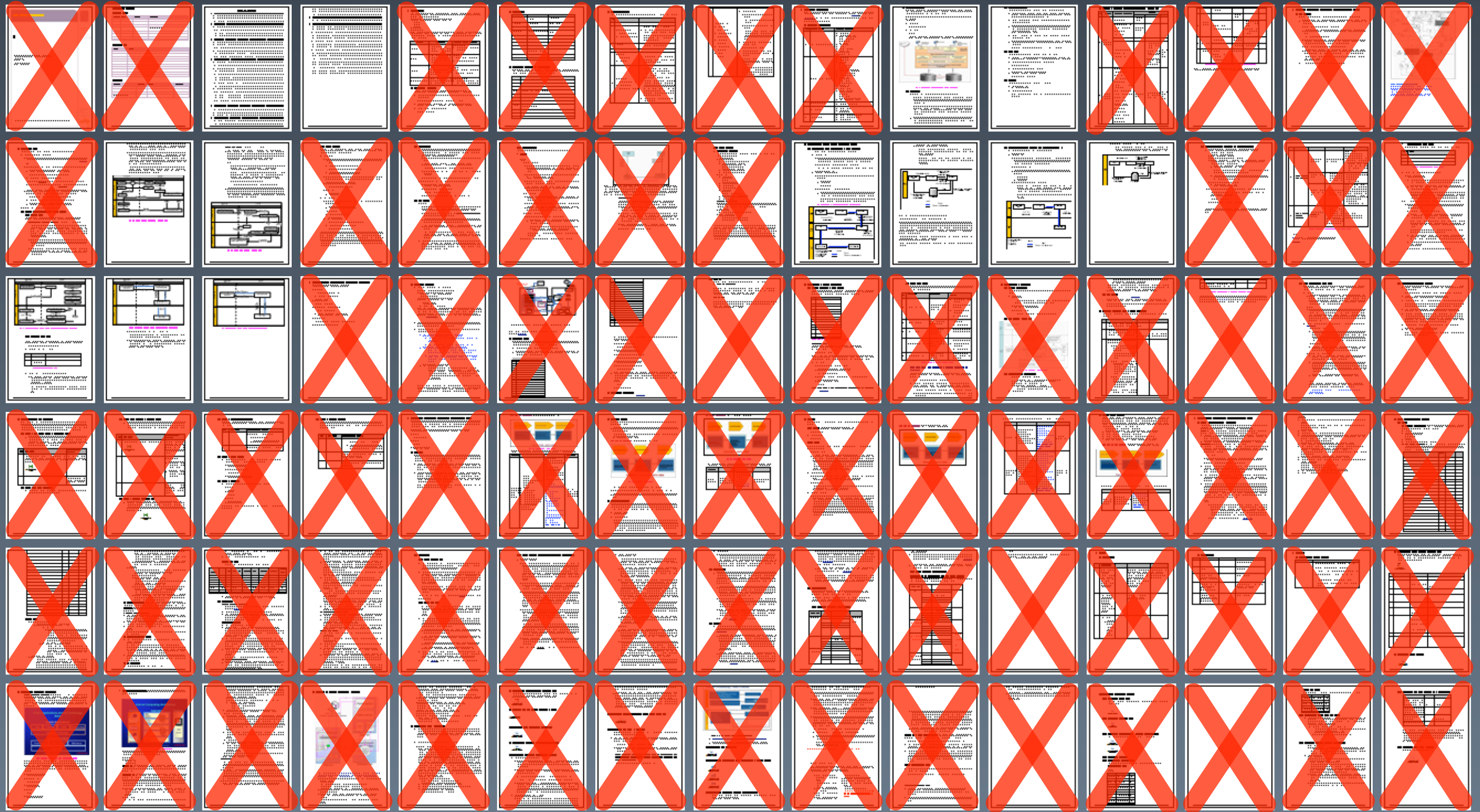
How is data managed, archived, backed-up, etc?

Technology Selection

What led to the selection of the technologies in use?

Architecture Justification

Does the chosen architecture “work”?



Author or architect?

coding
{the}
architecture



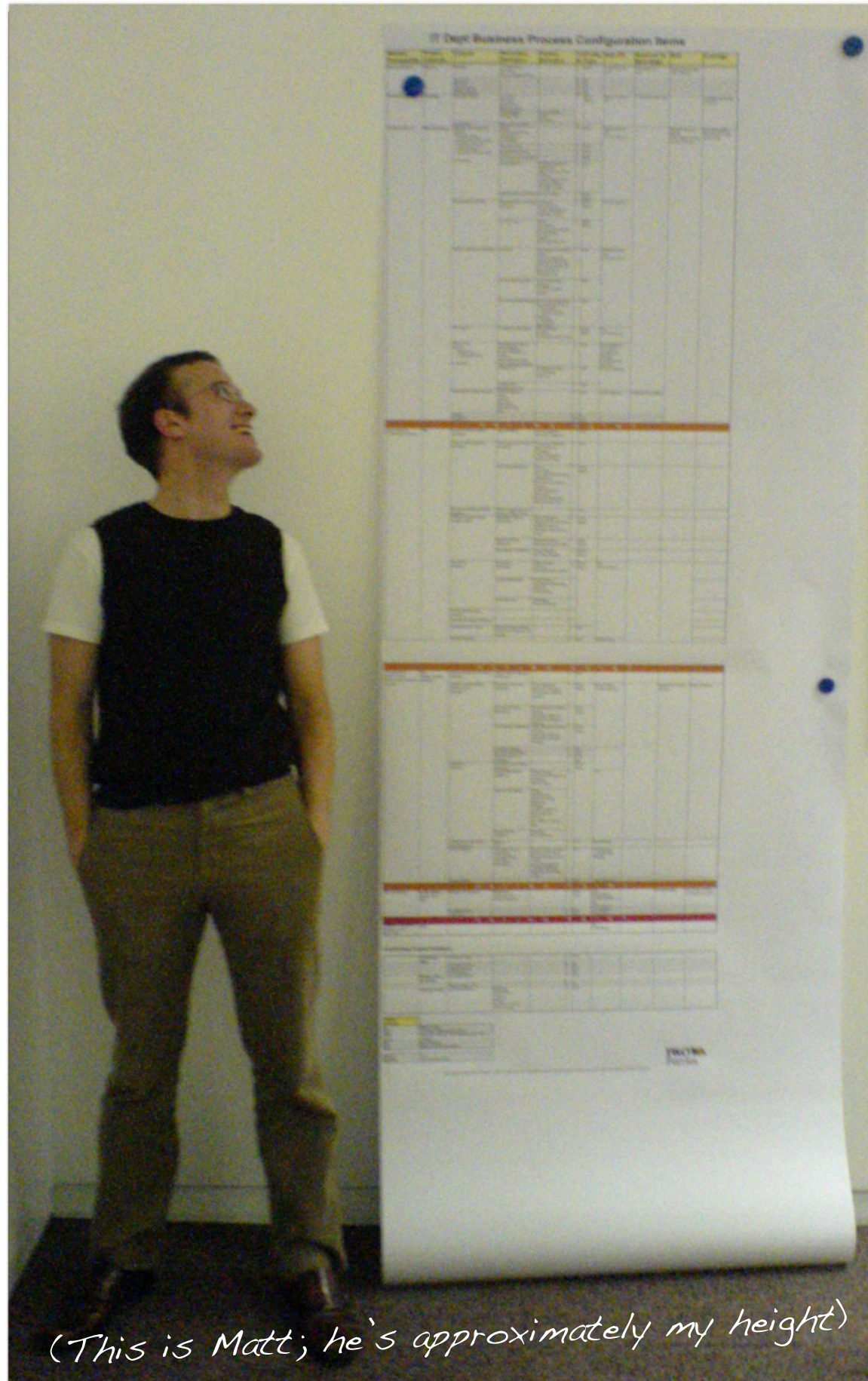
#win

Project Managing

{the}
architecture



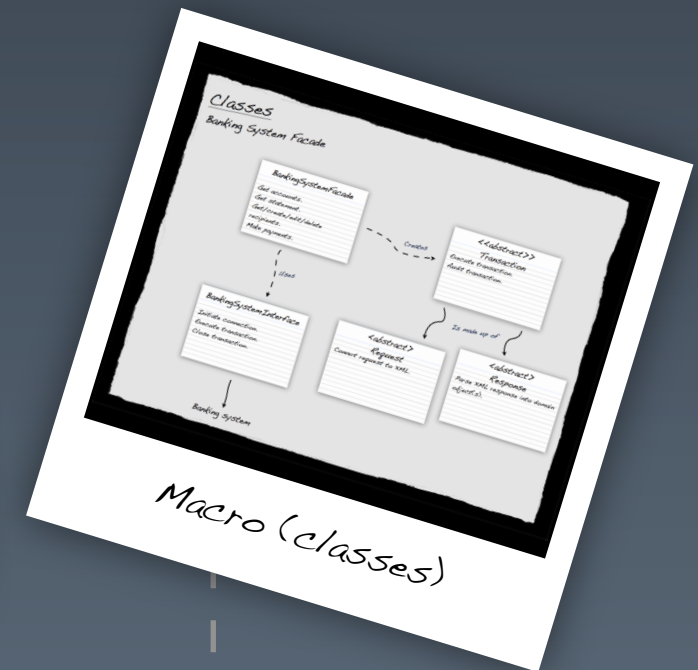
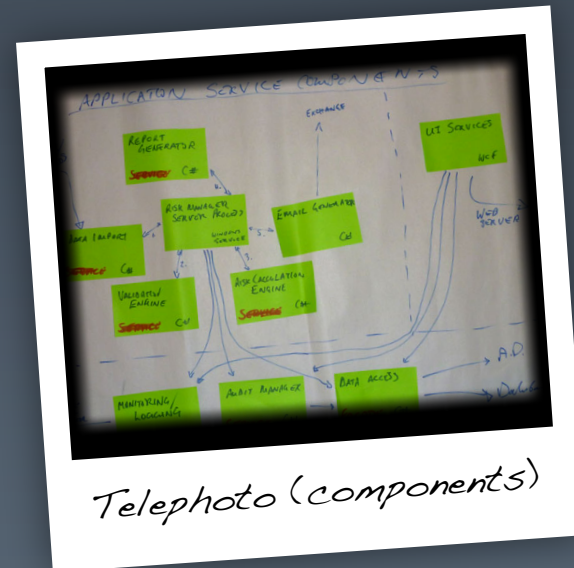
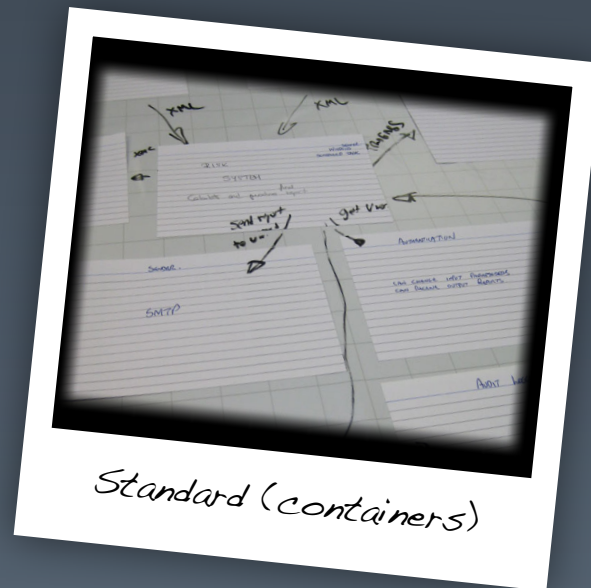
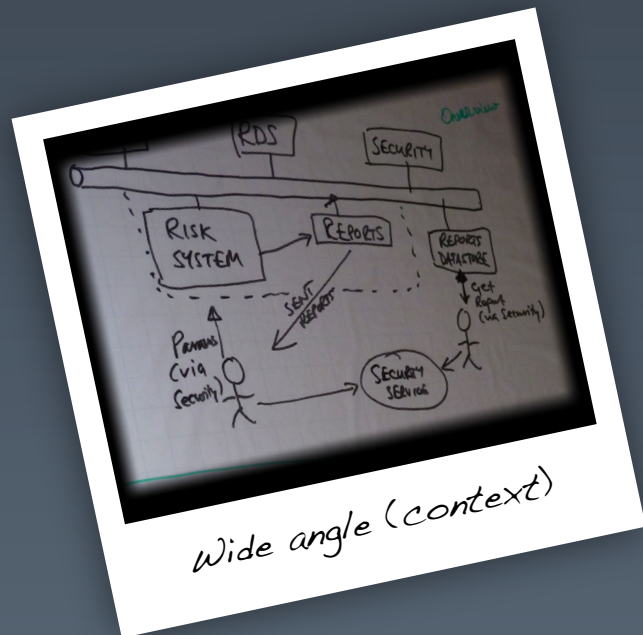
#fail



(This is Matt; he's approximately my height)

Technical
Project
Managers
tend to focus more on
project
management
than technology

Serenity

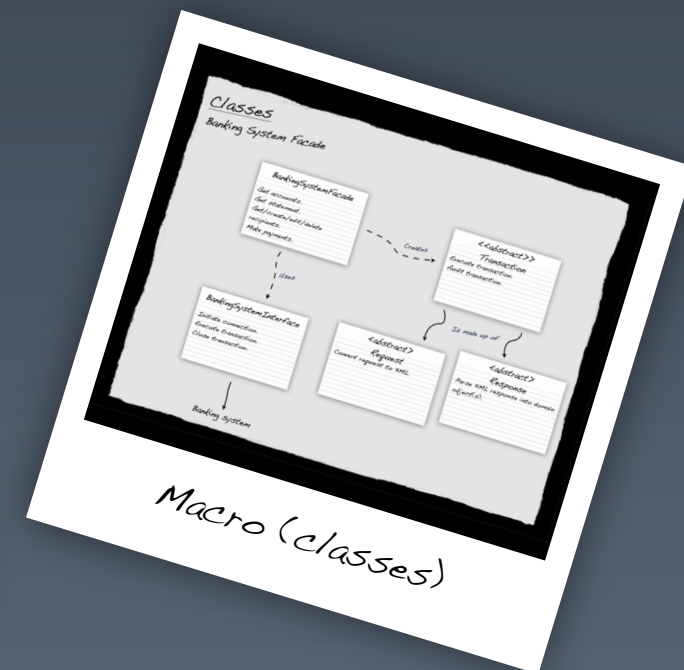
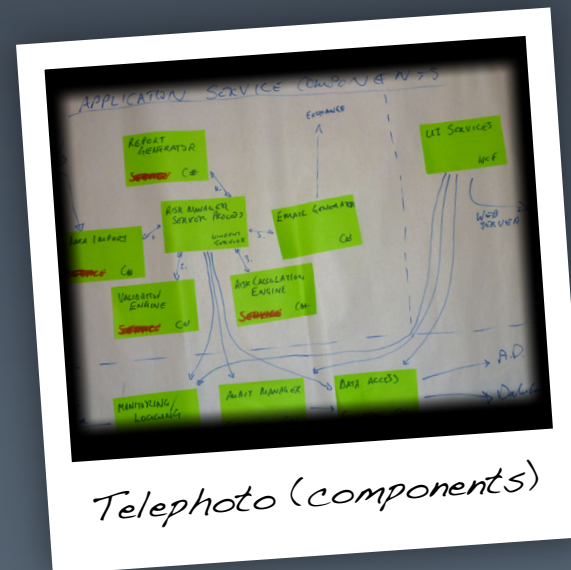
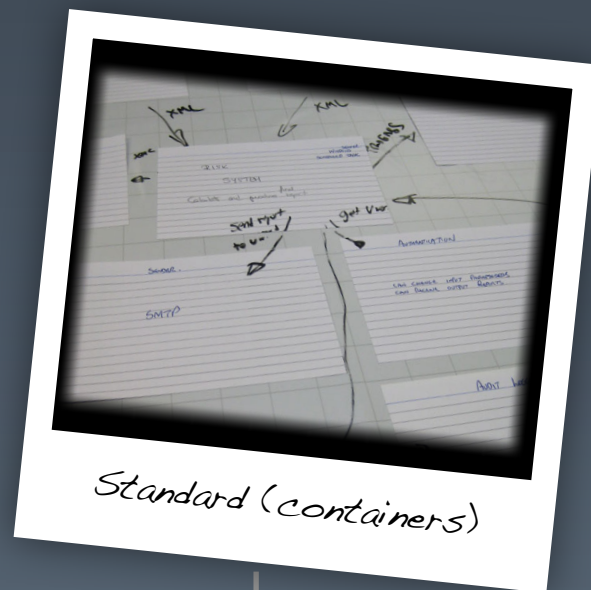
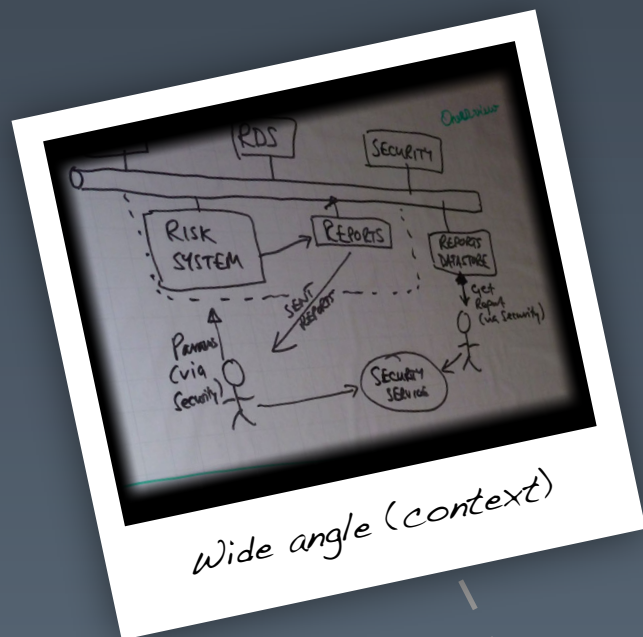


Abstract

Specific

As developers, the **code** is usually our main focus





Abstract

Specific



Sometimes you need to
step back from your IDE

coding—
architecture

We have an existing Internet Banking offering that allows customers to securely view information about their bank accounts held with us via the web. Although we were one of the first to market with such a product, the system itself is a number of years old now and a series of problems has been identified during a consulting exercise that we recently initiated. In summary:

- The system only provides customers with read-only access to information about their bank accounts. This includes account balances, recent transactions and recent statements.
- The information presented to customers is slightly out-of-date, because information from the core banking system is exported to the website on a nightly basis.
- Transactional requests are not possible through the site, with customers instead sending a secure message to the call centre with their request. This process is open to abuse and fraud.
- The number of features supported by the offering is limited.
- The technology is no longer seen as "leading edge", is hard to enhance and costly to alter.
- In addition, the system has "reached 'end of life'" and is no longer proactively supported by the vendor.
- The system doesn't meet current website acceptance standards.

In a recent survey, our Internet Banking system was perceived as poor in terms of the user experience and the level of information available through the website. With our competitors now offering fully transactional systems, there is a risk that we will lose business.

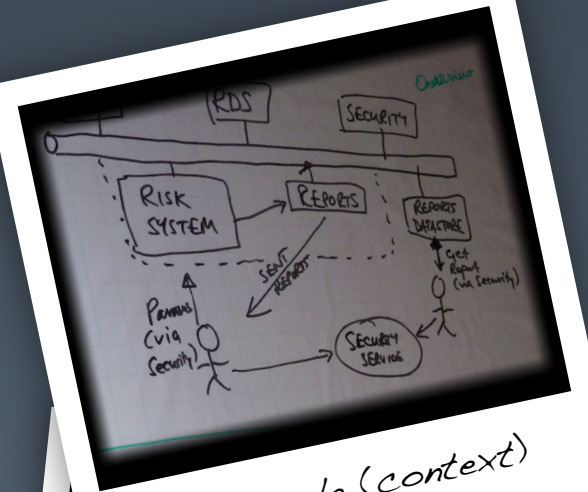
The board have given us the go-ahead to initiate a project to replace the current Internet Banking system, which will need to coincide with the corporate rebranding that will be taking place in 12 weeks. The replacement system should:

- Provide customers with real-time access to information about their bank accounts.
- Provide customers with the ability to perform common transactions through the website. This includes making payments, setting up standing orders, transferring money and so on.
- Provide customers with a rich user experience.
- Meet current website accessibility standards.
- Be developed using the new corporate website design guidelines.

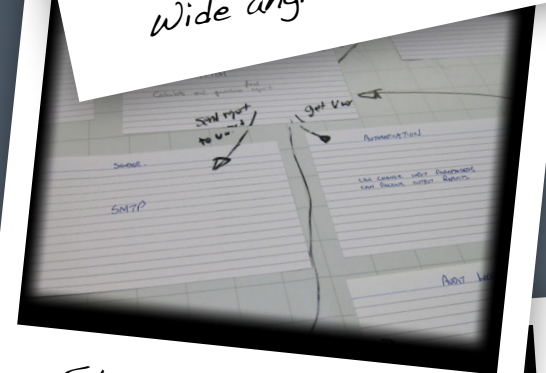
Functional & non-functional requirements

Constraints

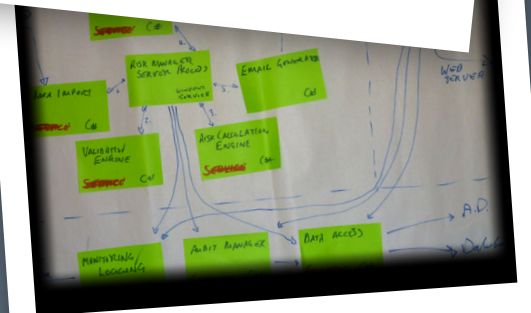
Principles



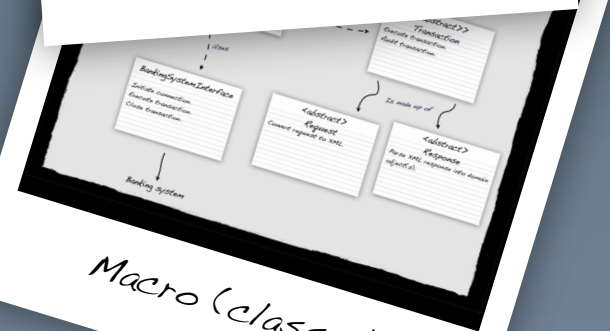
Wide angle (context)



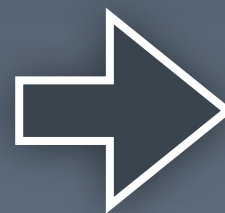
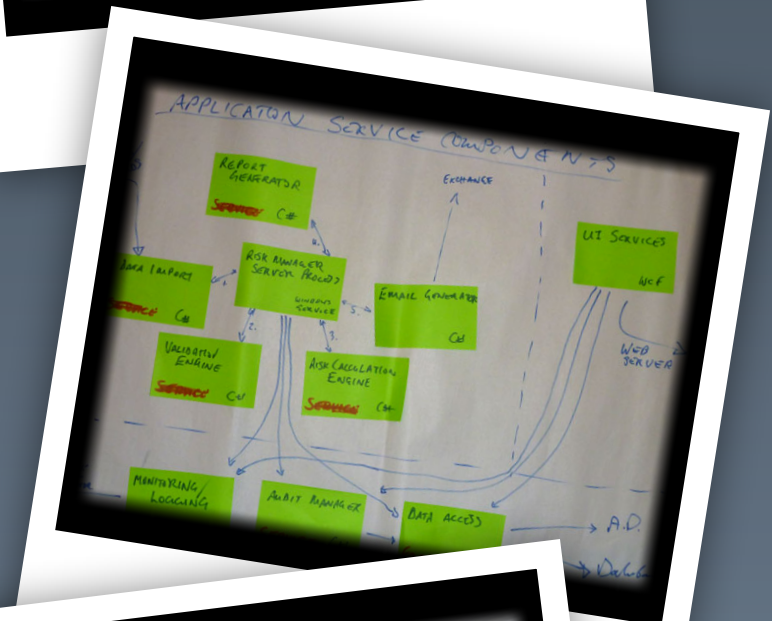
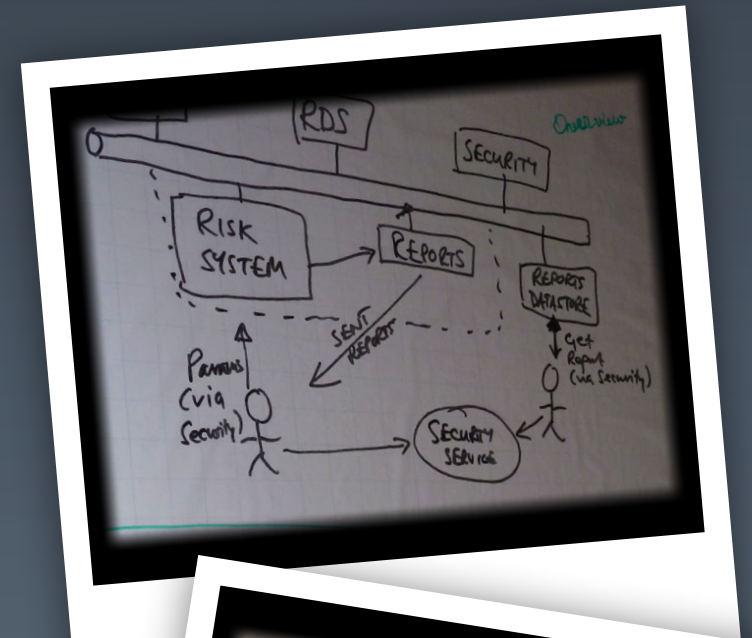
Standard (containers)



Telephoto (components)



Macro (classes)

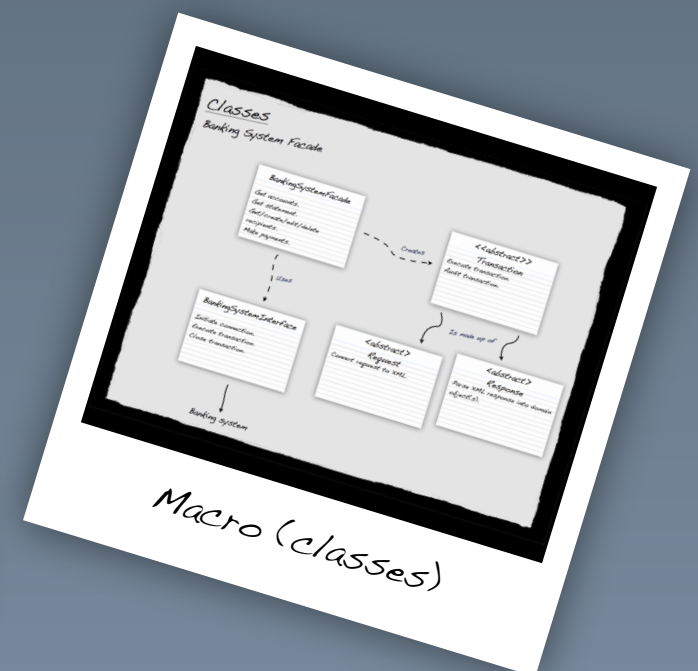
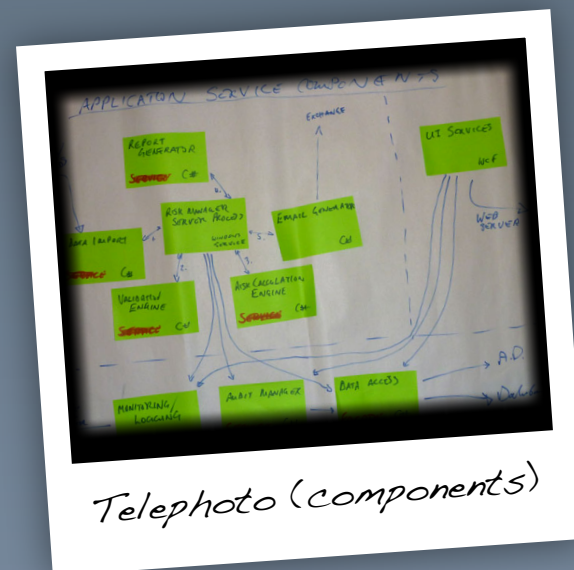
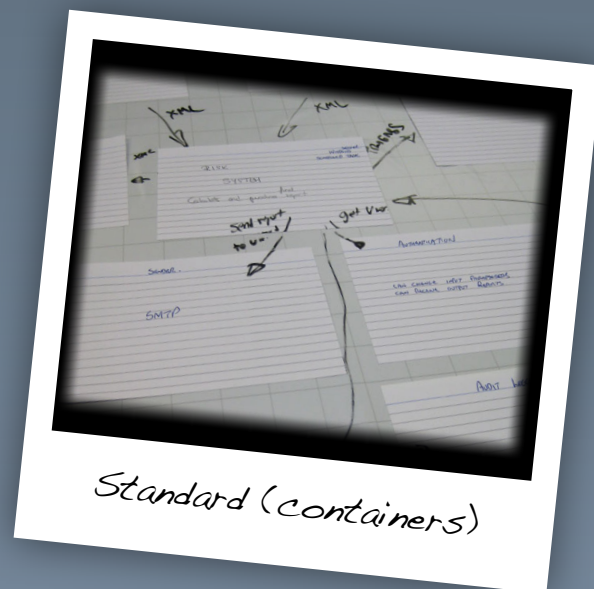
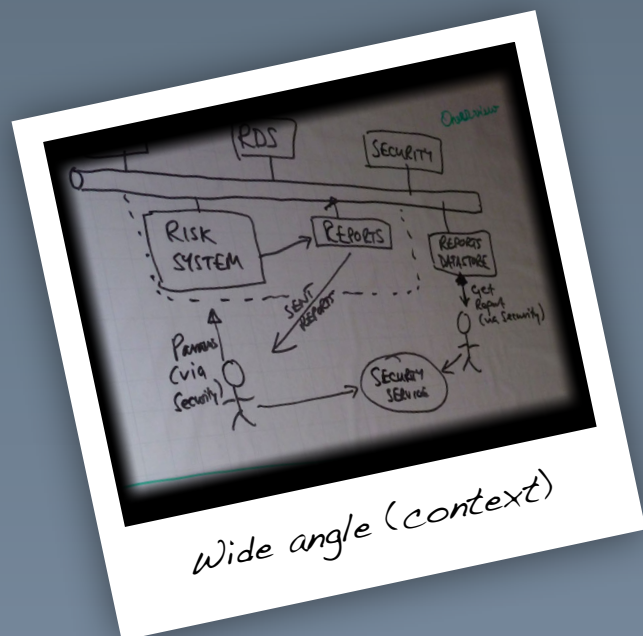


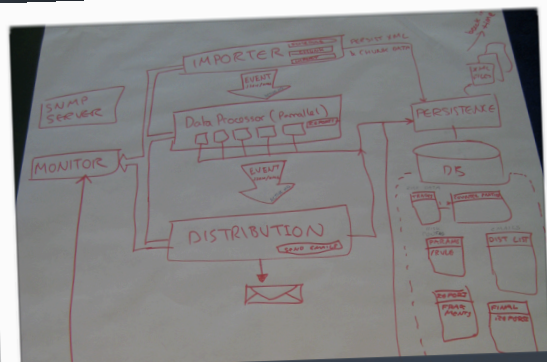
Effective sketches

are an excellent way to

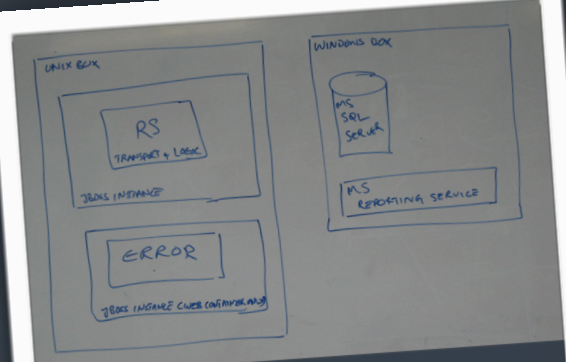
collaborate

on software architecture

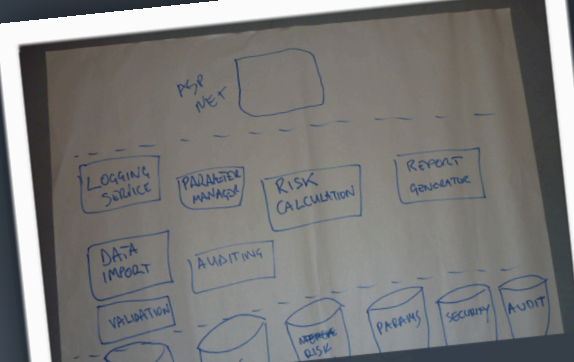




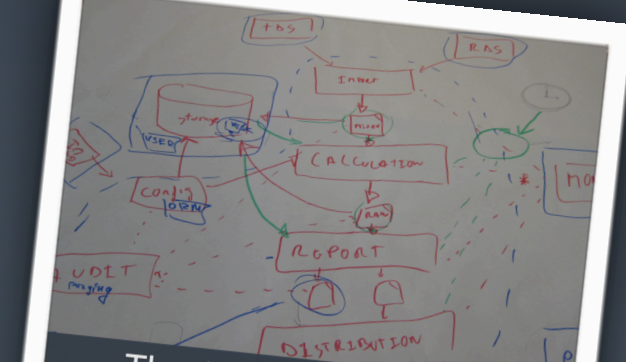
Form over Function



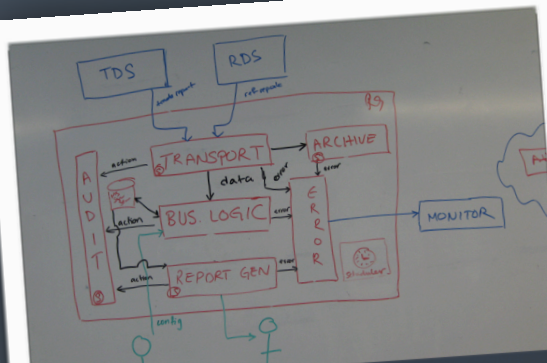
The Shopping List



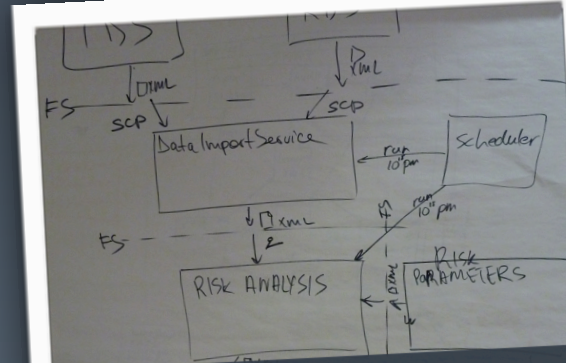
Boxes & no Lines



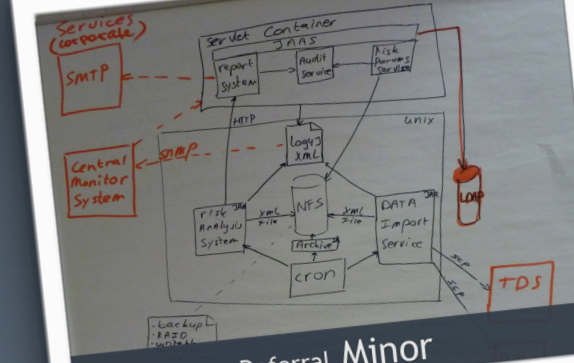
The Airline Route Map



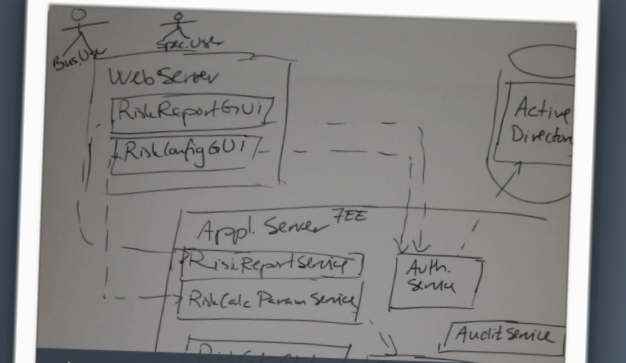
Generically True



Deferral Major



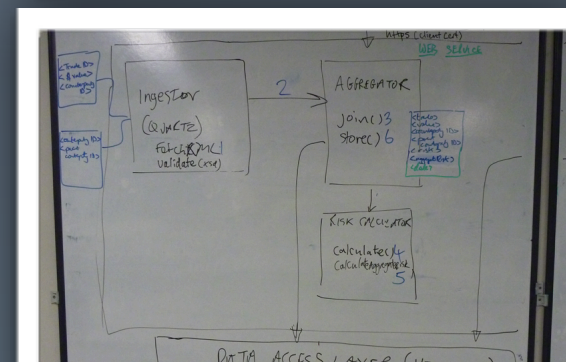
Deferral Minor



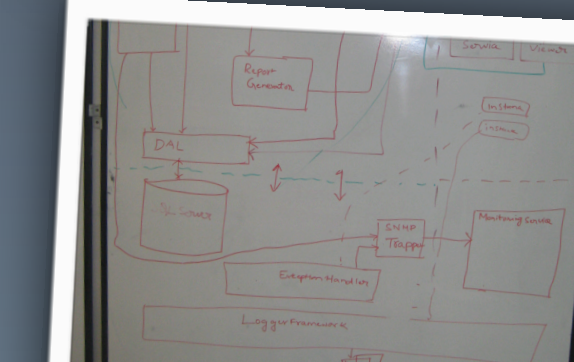
Assumptions are the mother of all ...



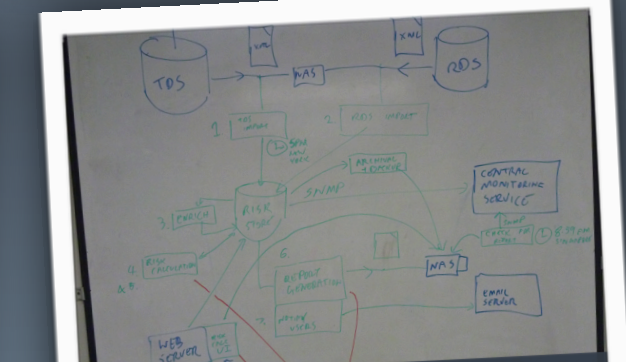
Homeless Old C# Object (HOCO)



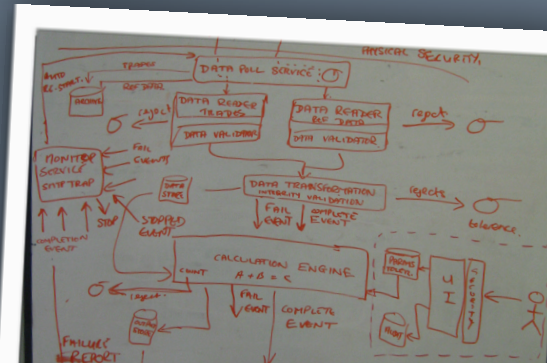
WYCITW?



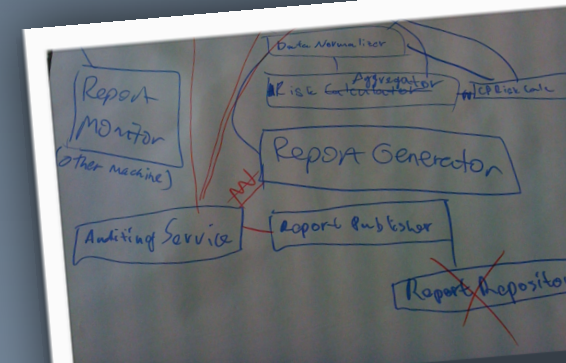
The Pompidou



Mixed Abstractions



The Adventure Book



Should have used a whiteboard!

Drawing diagrams
doesn't make you an
architect

Would you
code
it that way?

*This is why
software architects
must be able to code!*



[@unclebobmartin](#)

Uncle Bob Martin

The architecture of an accounting app
should scream "accounting" not Spring &
Hibernate.

26 Sep via [Twitter for iPhone](#) [★ Favorite](#) [↺ Retweet](#) [↩ Reply](#)

Or preferably **both**; I like software
architectures to be grounded in reality
(and that includes technology choices)



@unclebobmartin
Uncle Bob Martin

A good architecture allows you to defer framework decisions. A good architecture allows frameworks to act as plugins to the app.

26 Sep via [TweetDeck](#) ☆ [Favorite](#) ↺ [Retweet](#) ↻ [Reply](#)

Maybe, maybe not

(remember what I said about technology choices?)



@unclebobmartin
Uncle Bob Martin

I am amazed by the fact that some people actually disagree that a good software architecture allows you to defer framework decisions.

26 Sep via [TweetDeck](#) [★ Favorite](#) [↻ Retweet](#) [↩ Reply](#)

Deferring framework decisions and
isolating them should be a

conscious decision

*This is annoying **detail** and should
be deferred for as long as possible*

The Delivery Mechanism

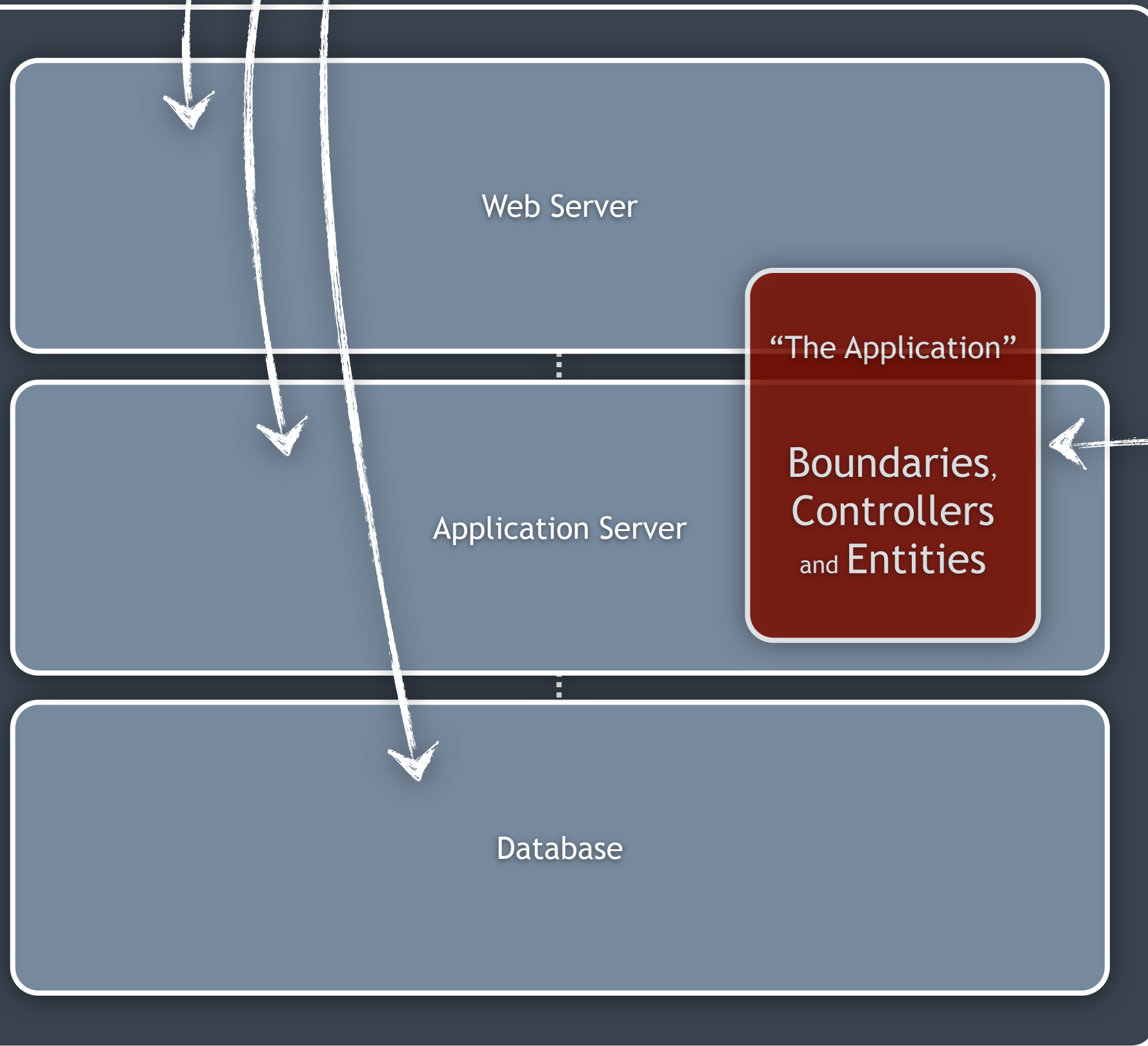
(e.g. web app, console app, SOA, etc)

The Application

Boundaries,
Controllers
and Entities

*This is the good stuff and is
what architecture is all about*

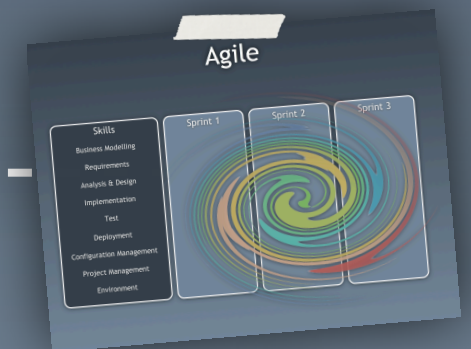
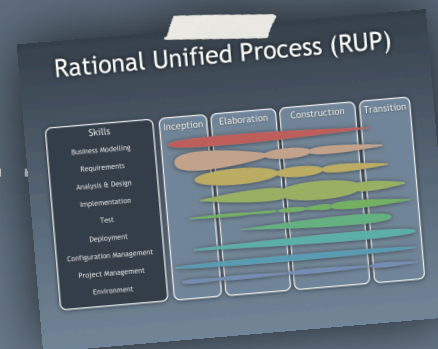
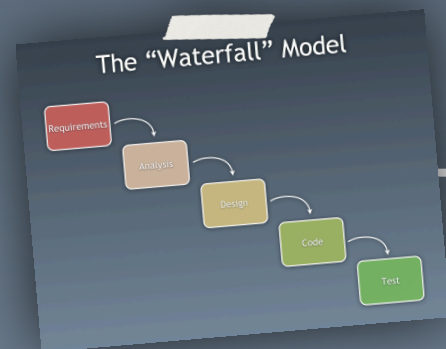
*This annoying detail (the "delivery mechanism")
makes up quite a large part of the big picture*



*This delivery mechanism
independent bit is just
one part of the
big picture*

How much up front design do you need to do?

Big?



None?

You need to do
“just enough”
up front design



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



Scott Ambler

<http://www.agilemodeling.com/essays/agileArchitecture.htm>

If you **flex** functionality,
does the architecture
change?

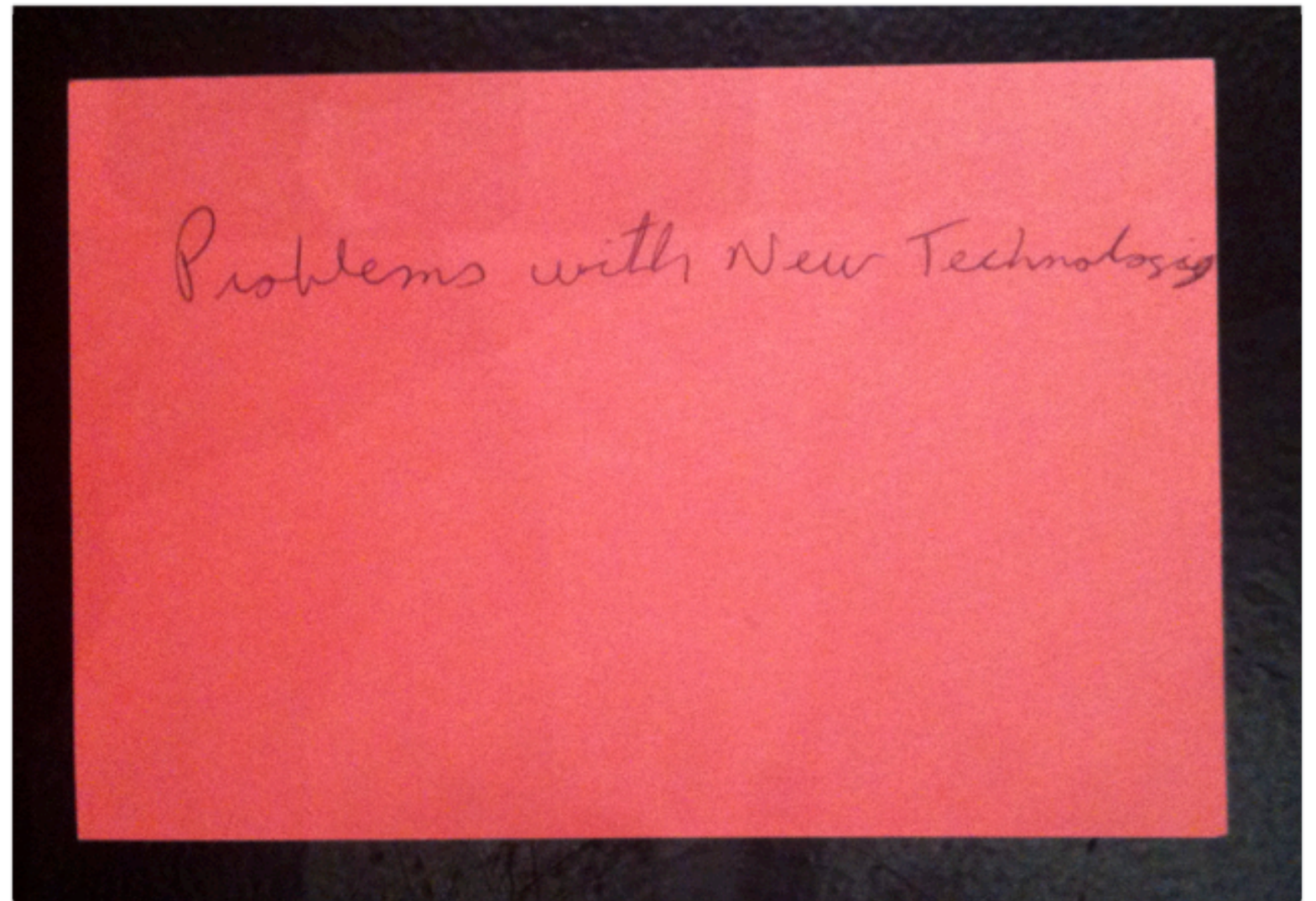
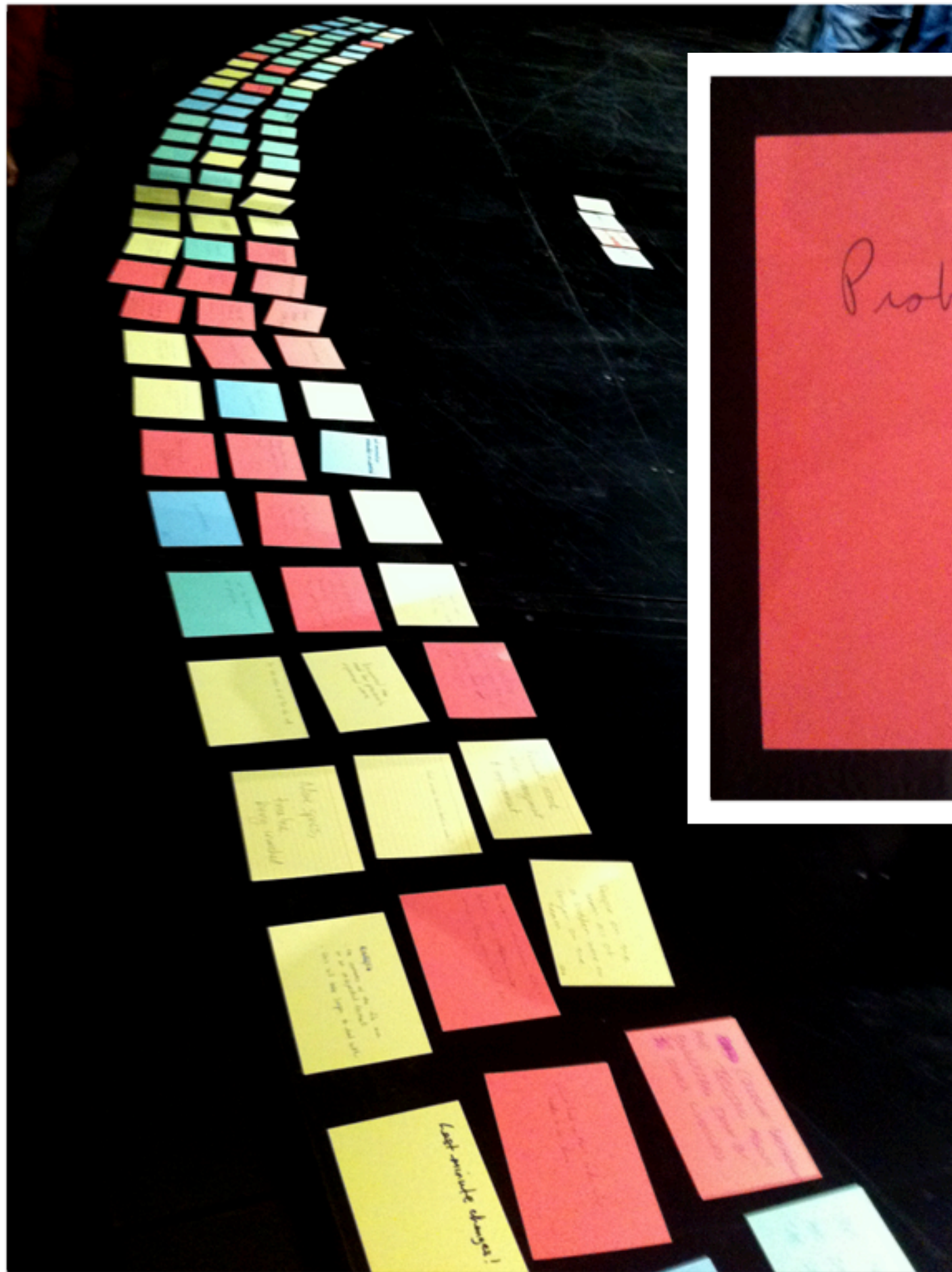
*Probably not, so just make
some decisions...*

What is architecturally significant?

*Costly to change
(can you refactor it
in an afternoon?)*

Complex and risky

New



An example timeline from
“Beyond Retrospectives”
by Linda Rising

#gotocon Aarhus 2011

Just enough

*Understand how the
significant elements
fit together*

*Mitigate the
key risks*

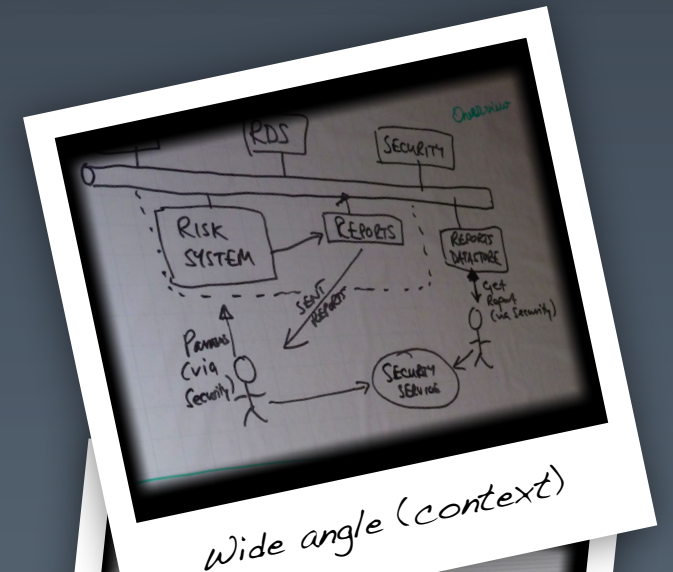
*Provide the foundations and
vision to move forward*

Context and Containers

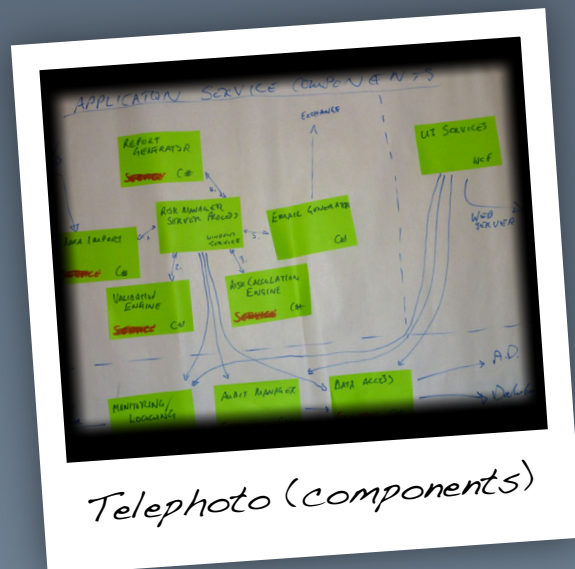
Requirements



1-2 days



Standard (containers)



Components (and Ballpark Estimates)

Doing this collaboratively allows people's separate ideas to meet

The **role** of the software architect and
the **process** of software architecture are

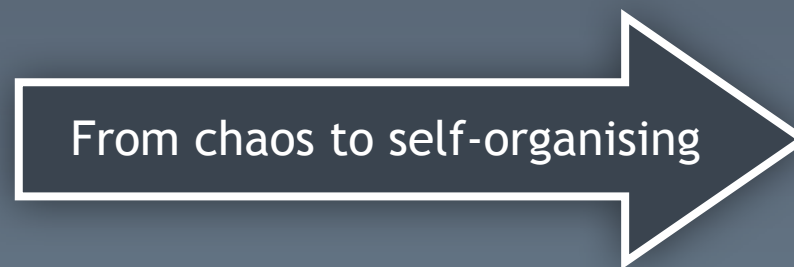
different

The role



Dedicated
software architect

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



Everybody is a
software architect

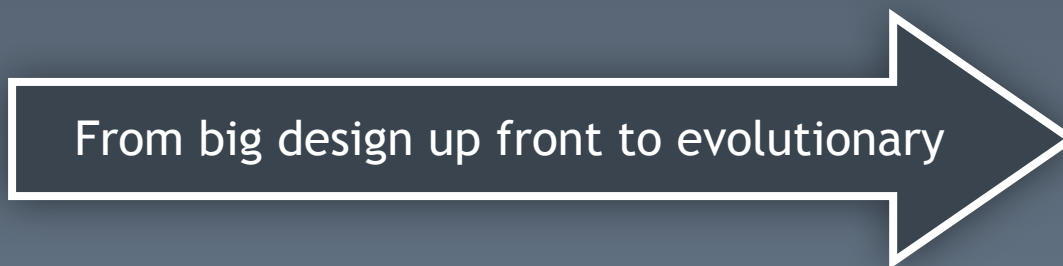
Joint responsibility for the
technical aspects of the
software project

The process



Big up front design

Requirements capture, analysis
and design complete before
coding starts



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

    public SomeWizard()
    {
    }

    ...
}
```

Evolutionary architecture

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



The role



“Just enough”

*Understand how the
significant elements
fit together*

*Mitigate the
key risks*

*Provide the foundations and
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()
    {
    }

    ...
}
```

Let's wrap up...

Does *agile* need architecture?

Yes, architecture provides
structure, firm foundations,
vision and technical leadership

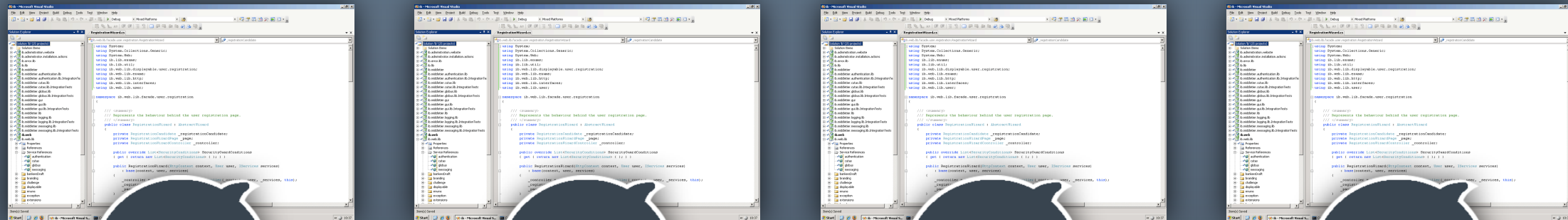
Does architecture need *agile*?

Yes, it helps software teams move away from

big design up front
and analysis paralysis

Define

the software architecture role and
collaborate



Talk about architecture in
your retrospectives

Do you want to

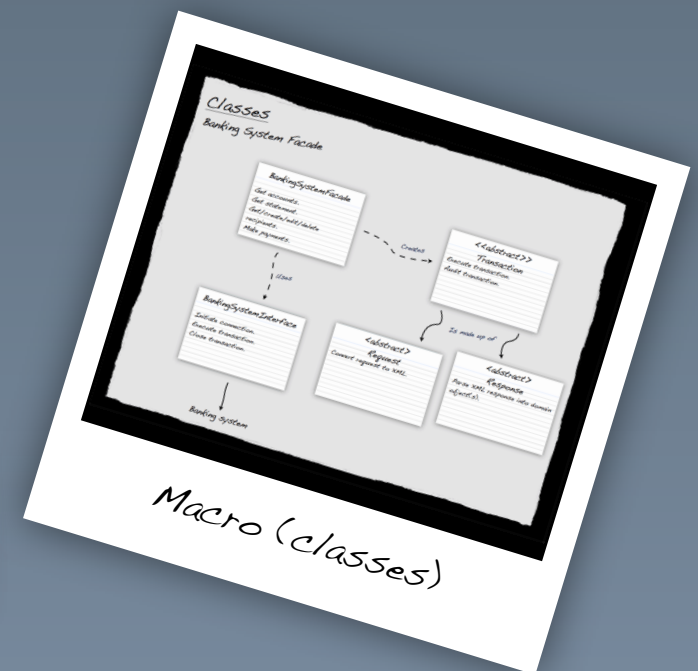
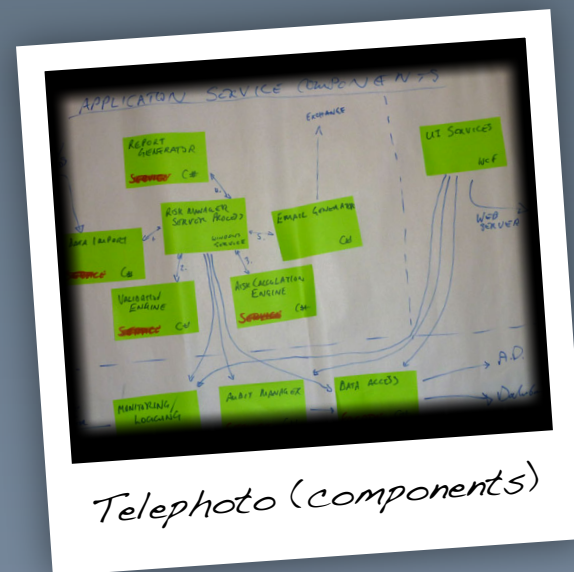
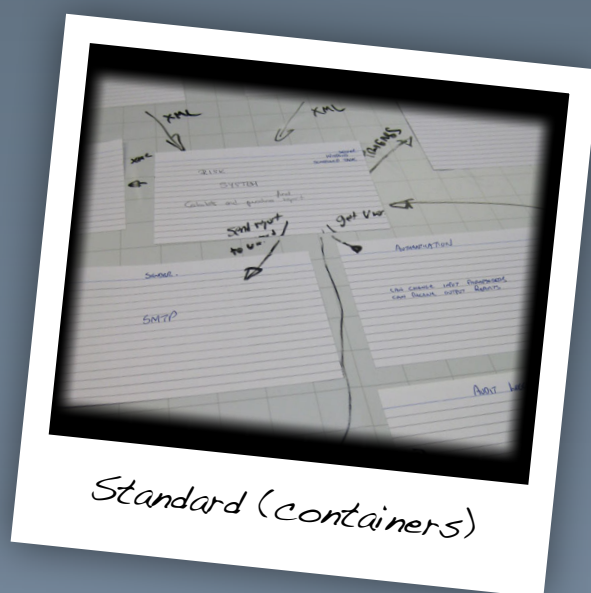
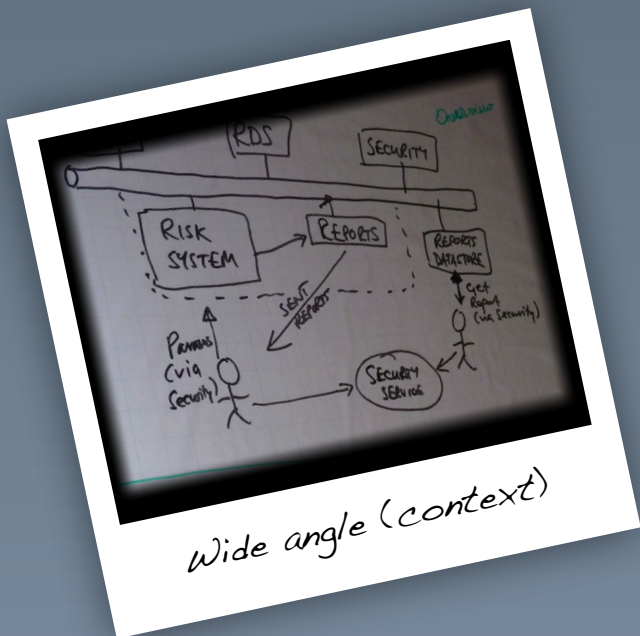
code?

Software architecture documentation should

describe what

the code doesn't

Effective sketches
are an excellent way to
communicate
software architecture



Apply *craftsmanship*
to
software architecture

*Effective yet lightweight
Sketches and documentation*

*Software systems
that actually work*

We need to grow the
architects of
tomorrow

Do whatever works for

you

(just don't get distracted by shiny new things just because they're shinier!)



simon.brown@codingthearchitecture.com

@simonbrown on Twitter