coding {the} architecture

Effective software architecture sketches

Simon Brown
@simonbrown

simon.brown@codingthearchitecture.com

@simonbrown on Twitter

Jersey

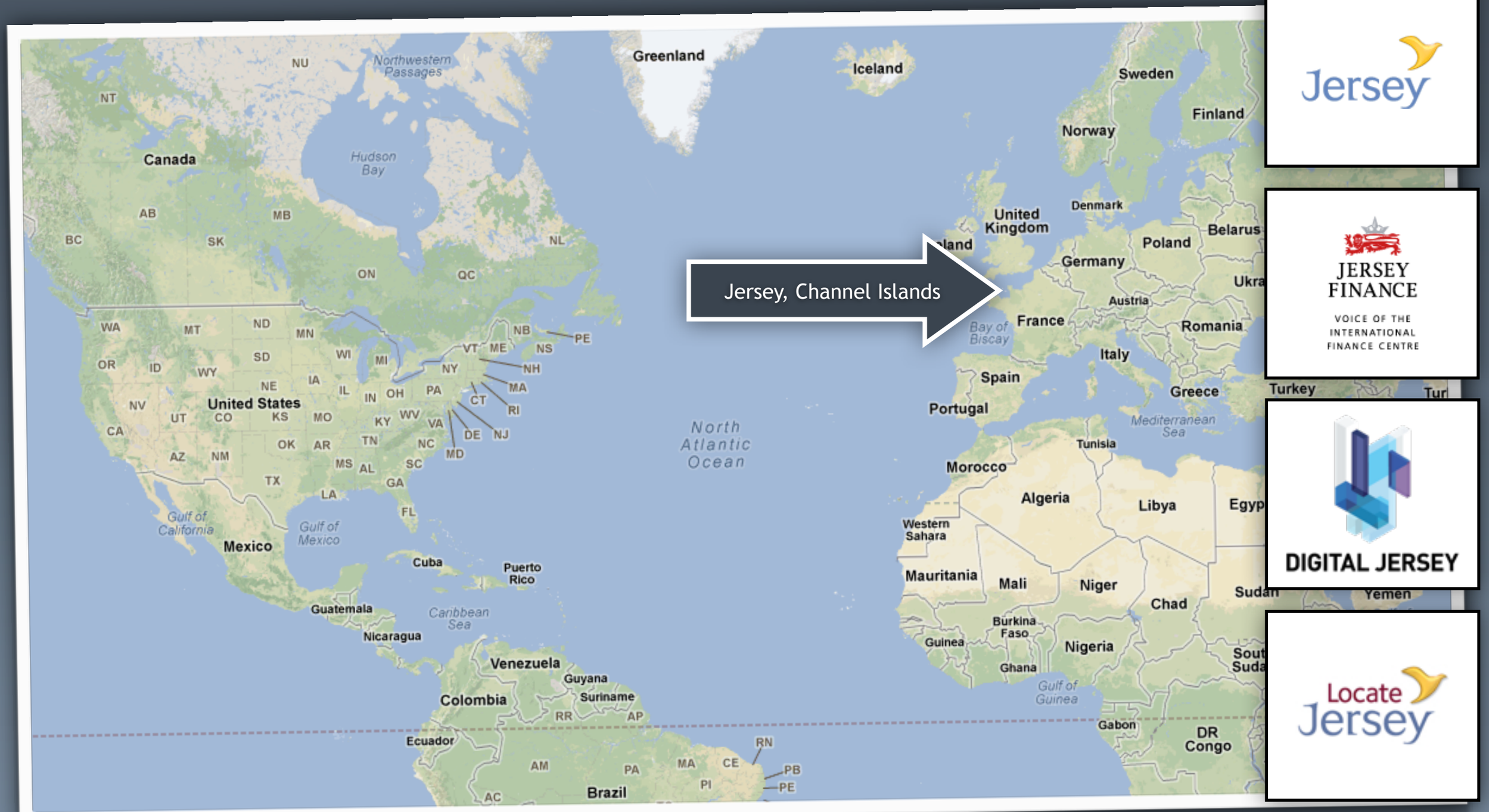
JERSEY FINANCE
VOICE OF THE INTERNATIONAL FINANCE CENTRE

DIGITAL JERSEY

Locate Jersey

simon.brown@codingthearchitecture.com

@simonbrown on Twitter



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

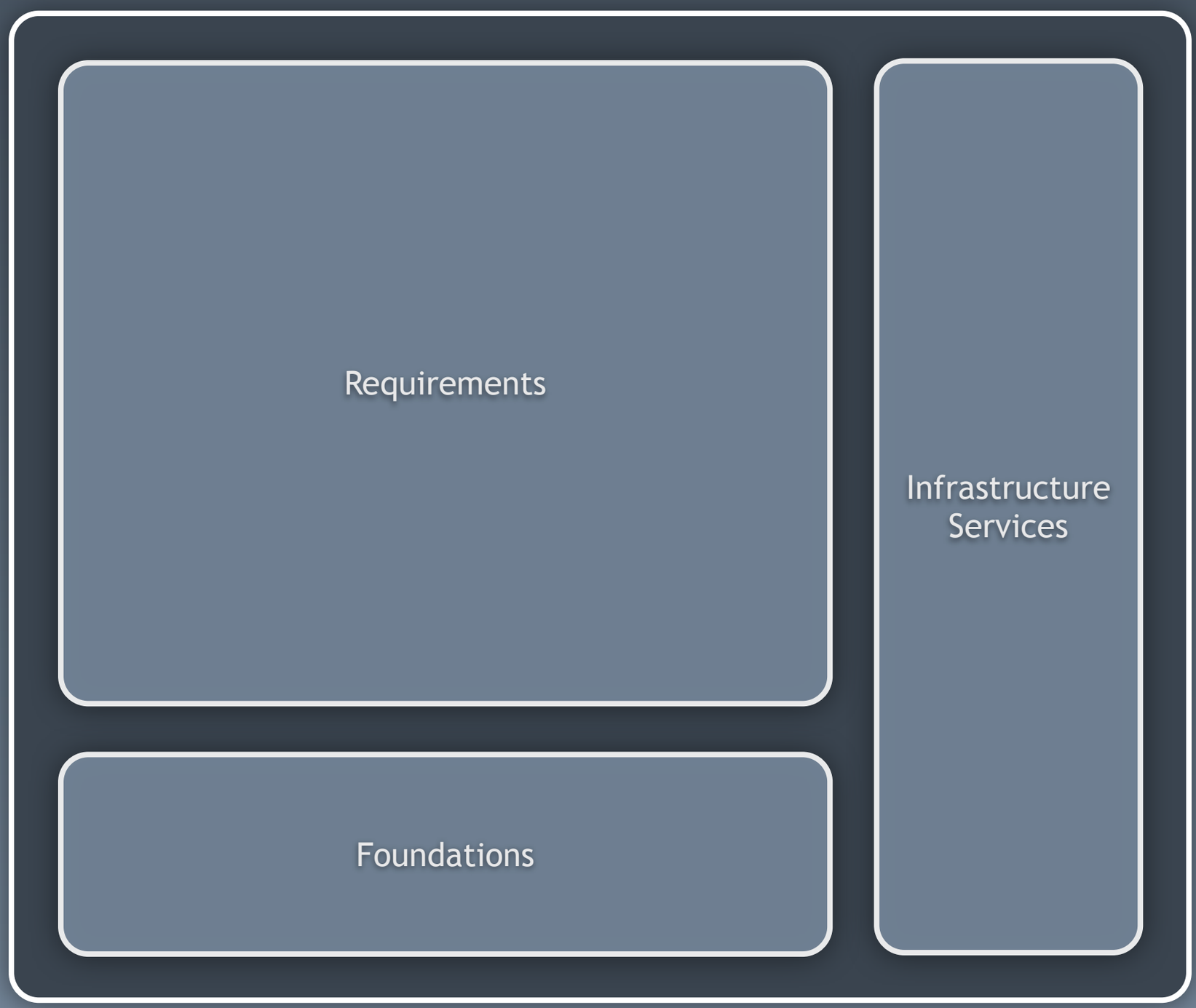
(I code too)

Training

Book

Speaking

What is architecture?



As a noun...

Structure

The definition of something in terms of its components and interactions

As a verb...

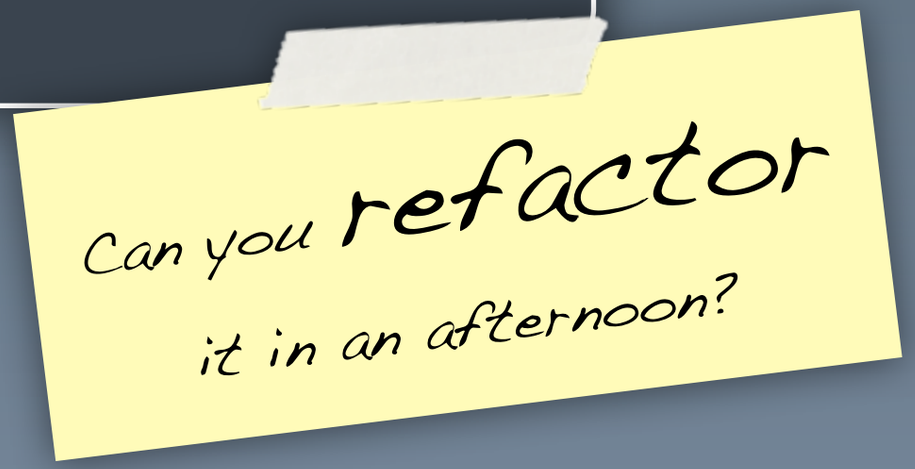
Vision

The process of architecting, making design decisions, providing guidance, etc

Architecture vs design

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

Grady Booch



5 minutes

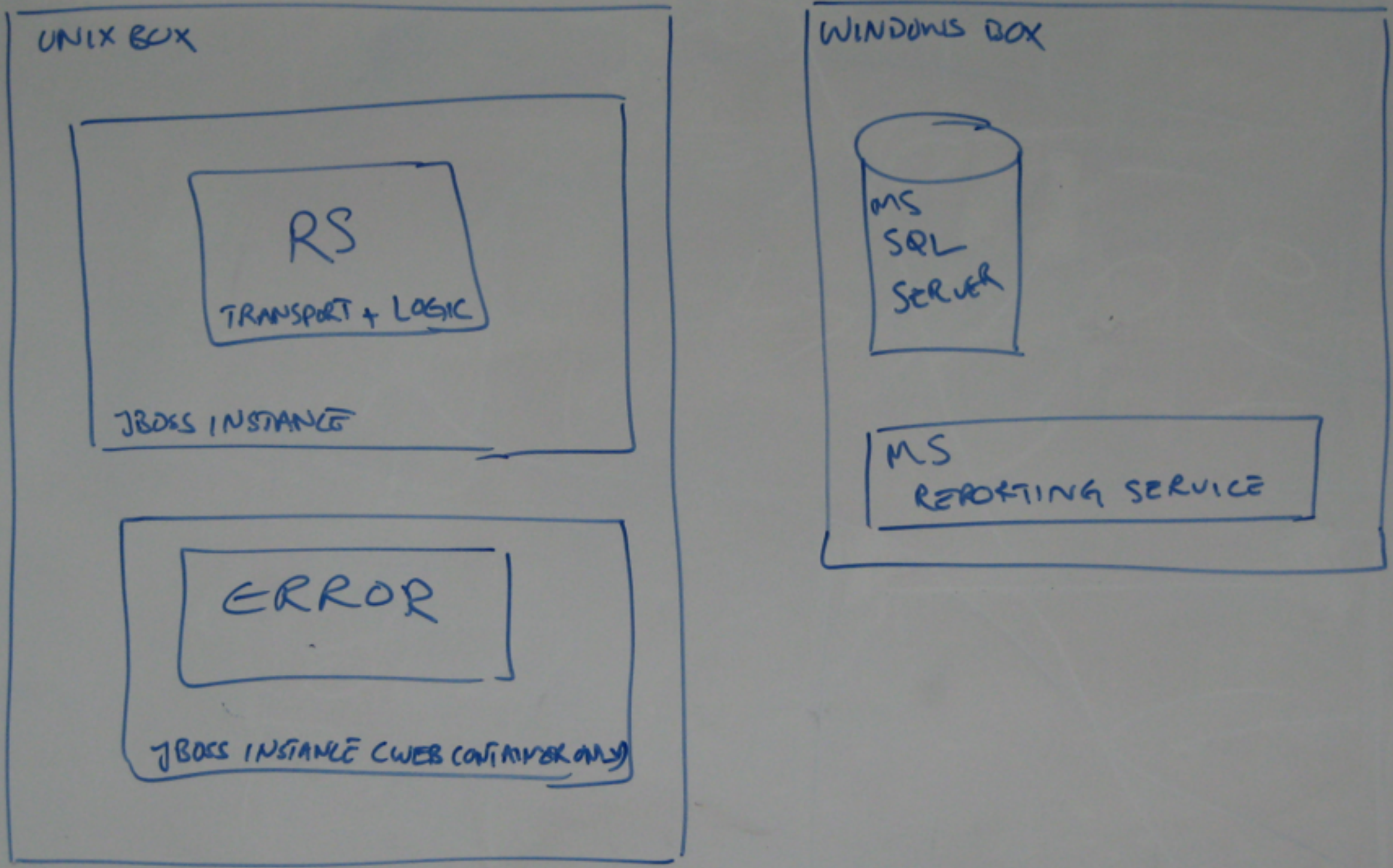
Sketch one or more diagrams to describe the software architecture of a software system you've worked on

Was this easy?

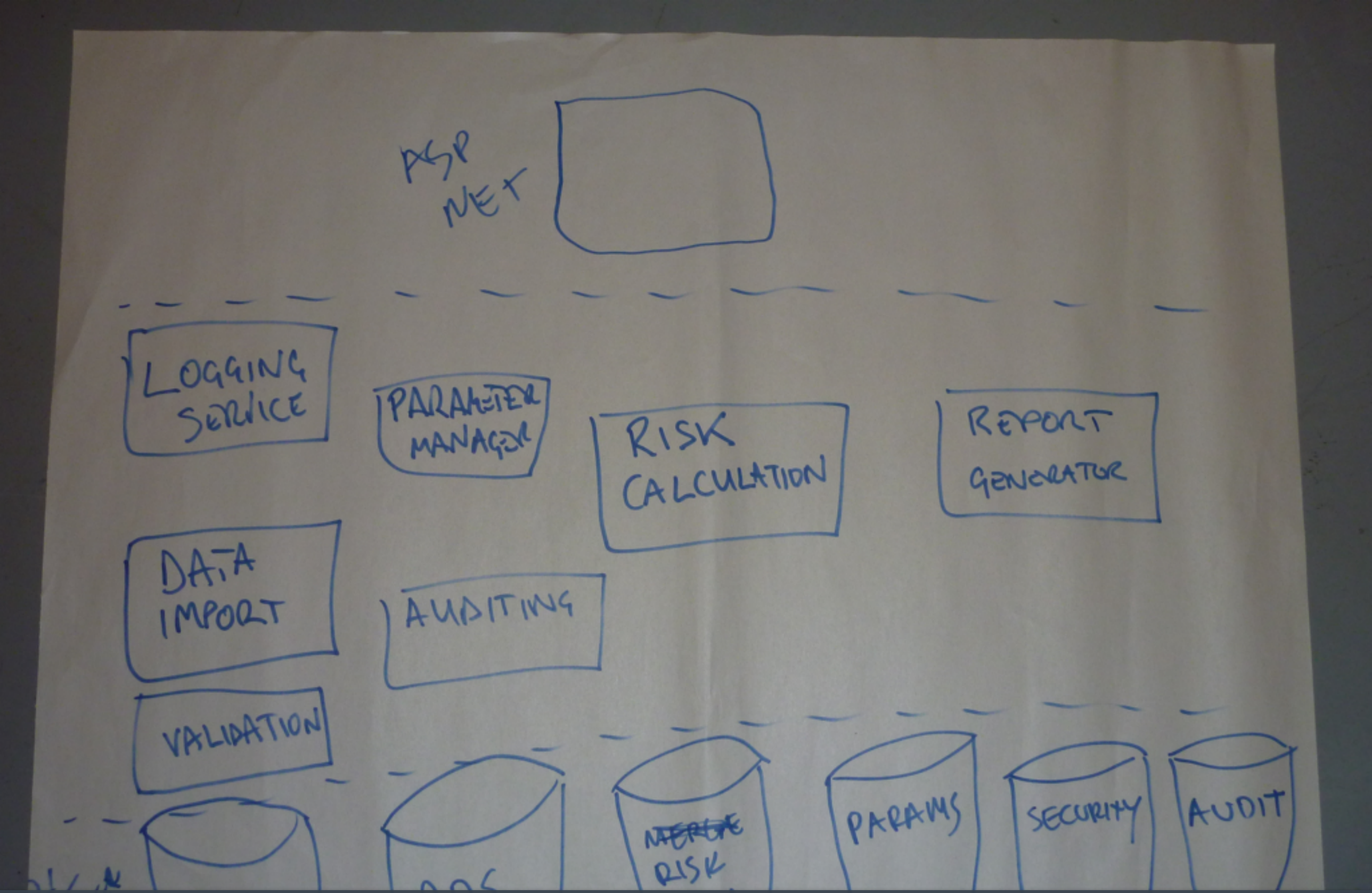
Are these

effective

sketches?



The Shopping List



Boxes & No Lines

FUNCTIONAL VIEW

File Retriever

Scheduler

Auditing

Reference Archiver

Risk Assesment Processor

Risk Parameter Configuration

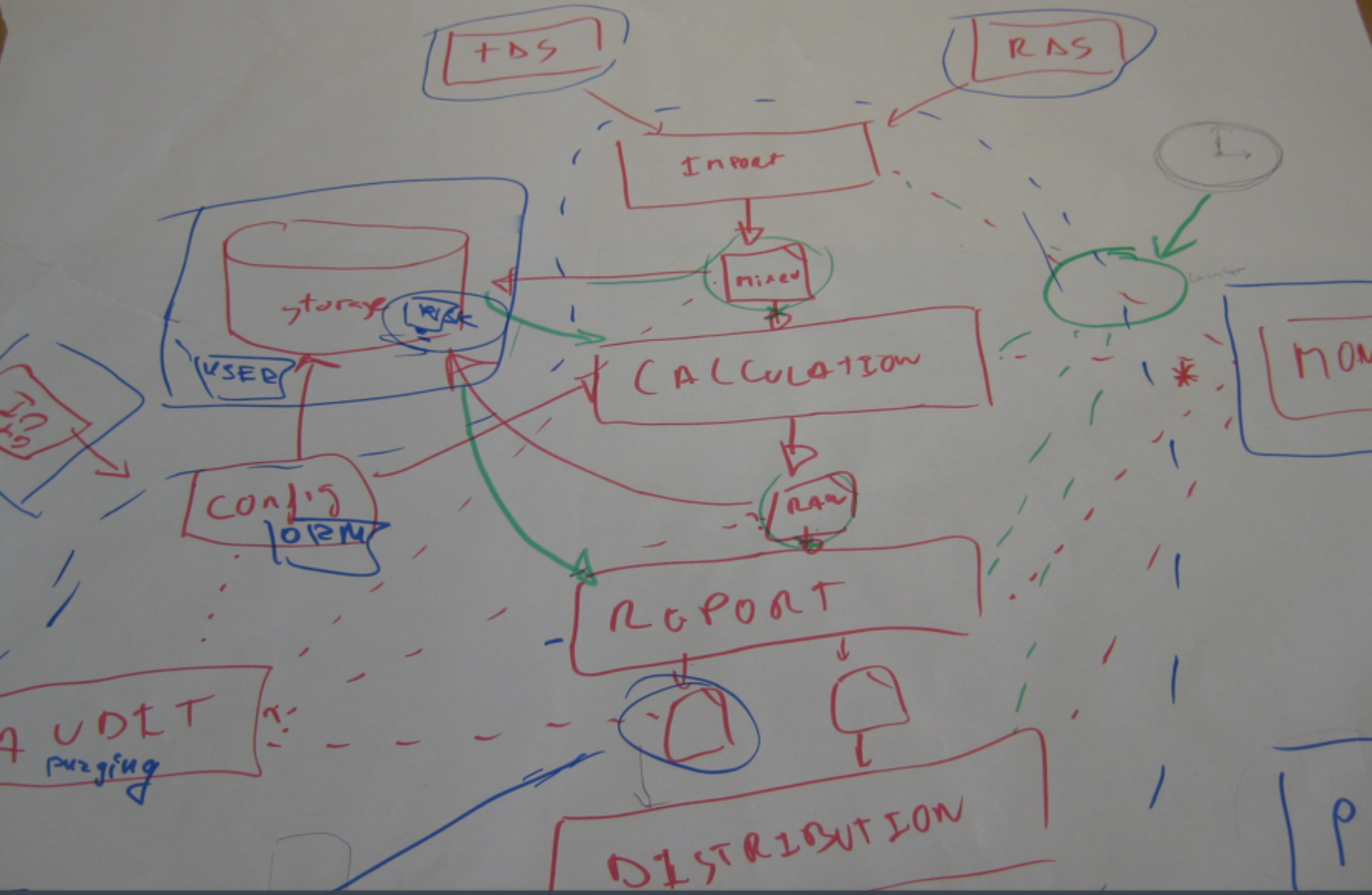
Trade Archiver

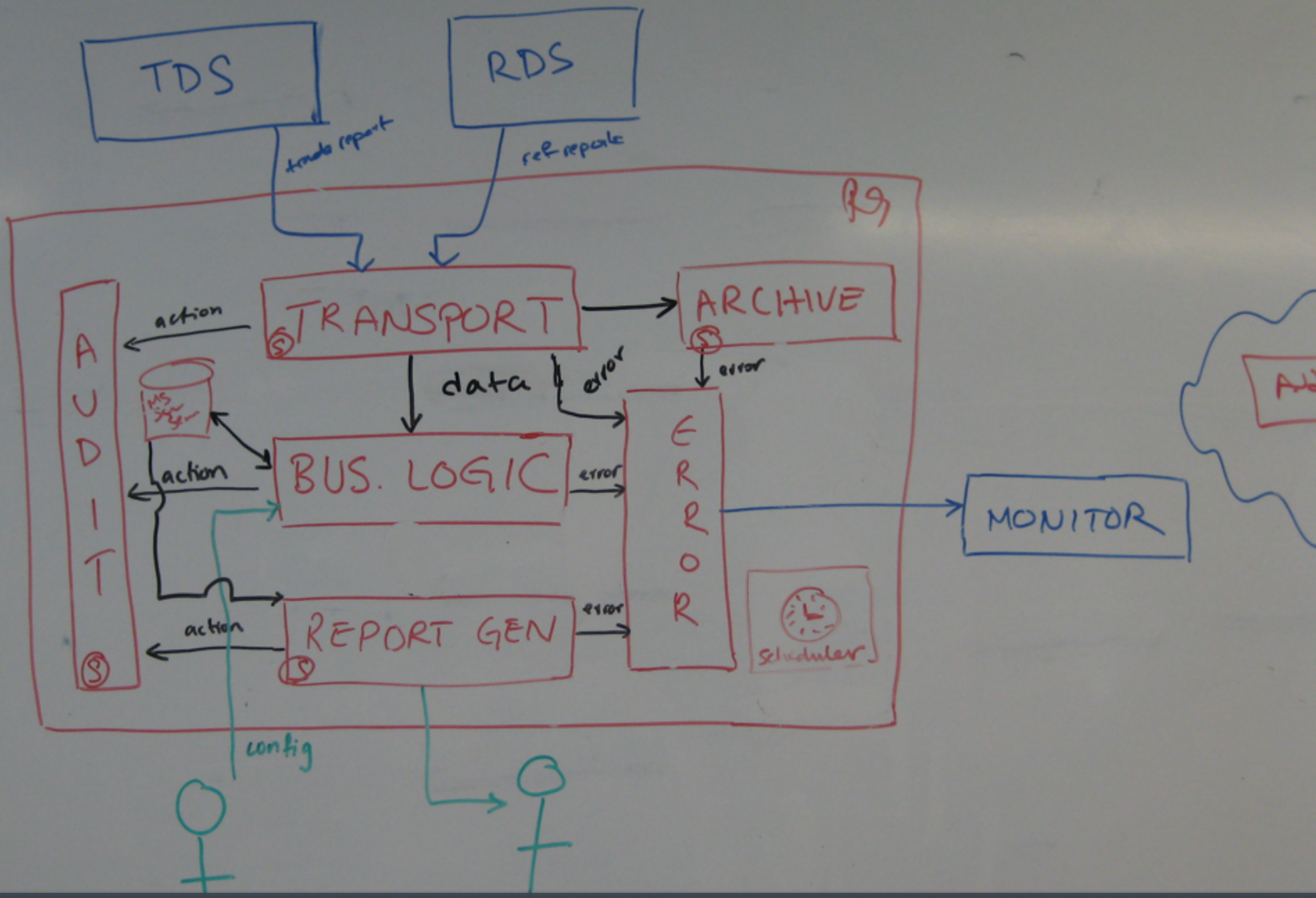
Report Generator

Report Distributor

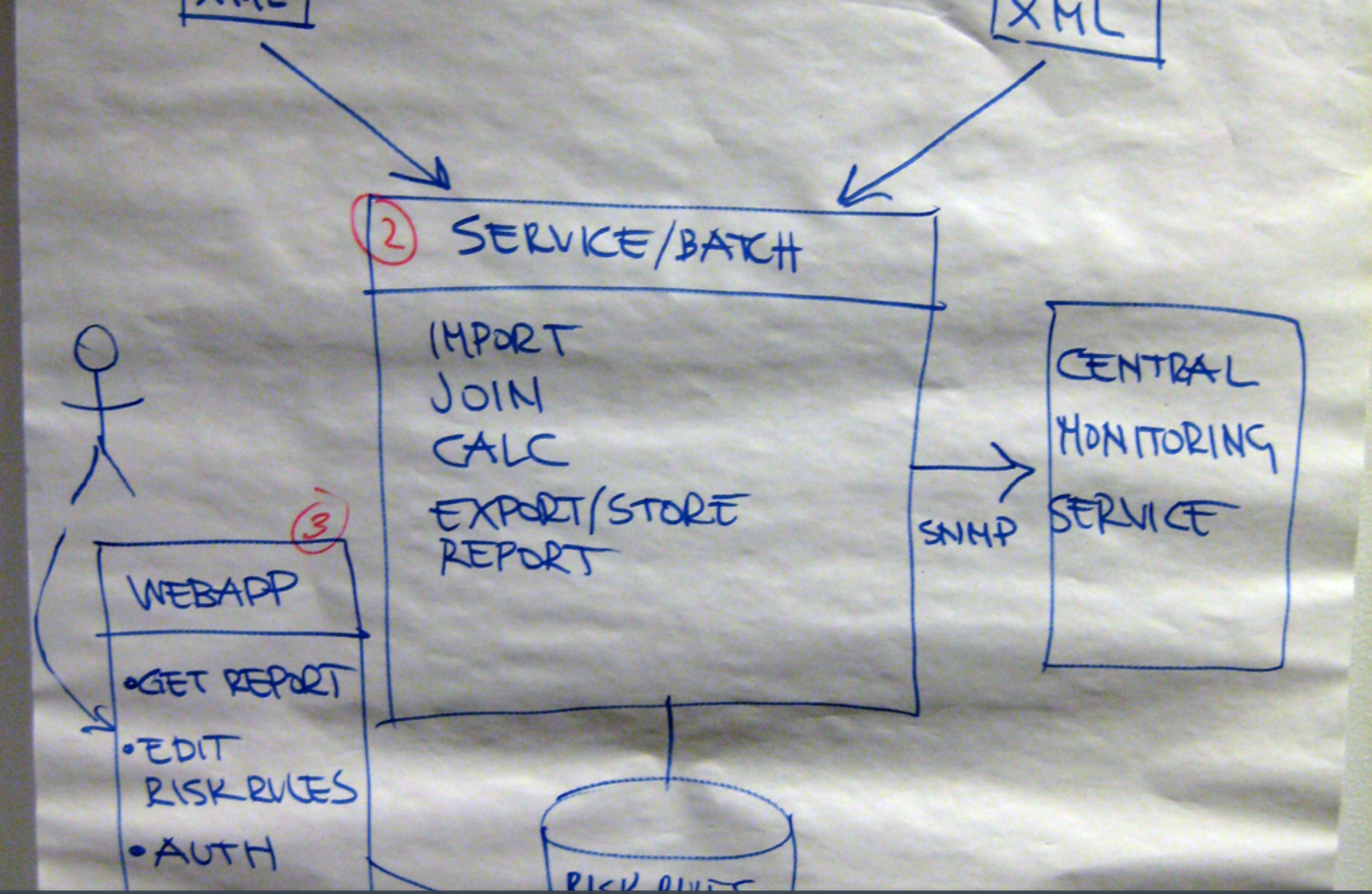
The Functional View

The Airline Route Map

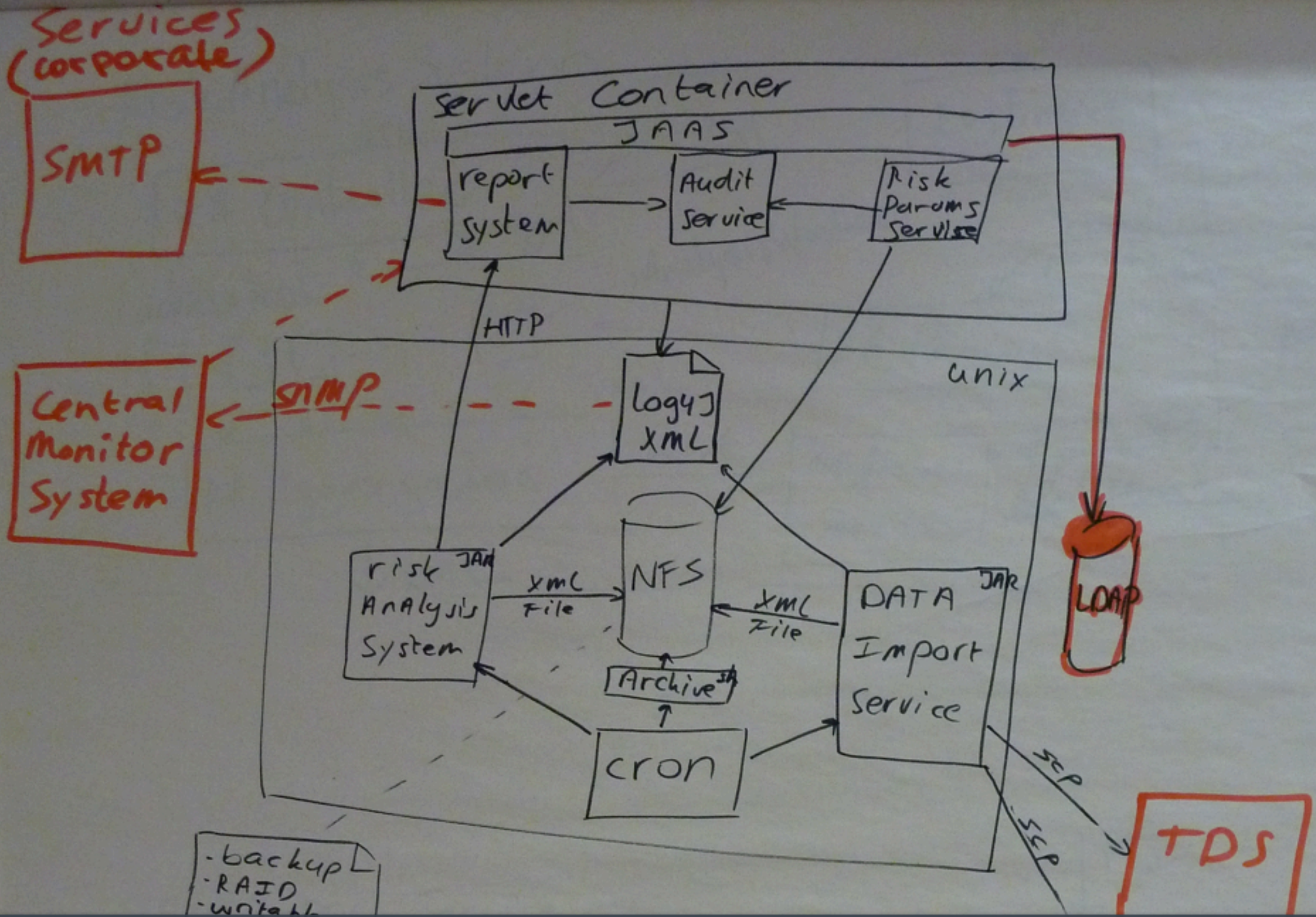




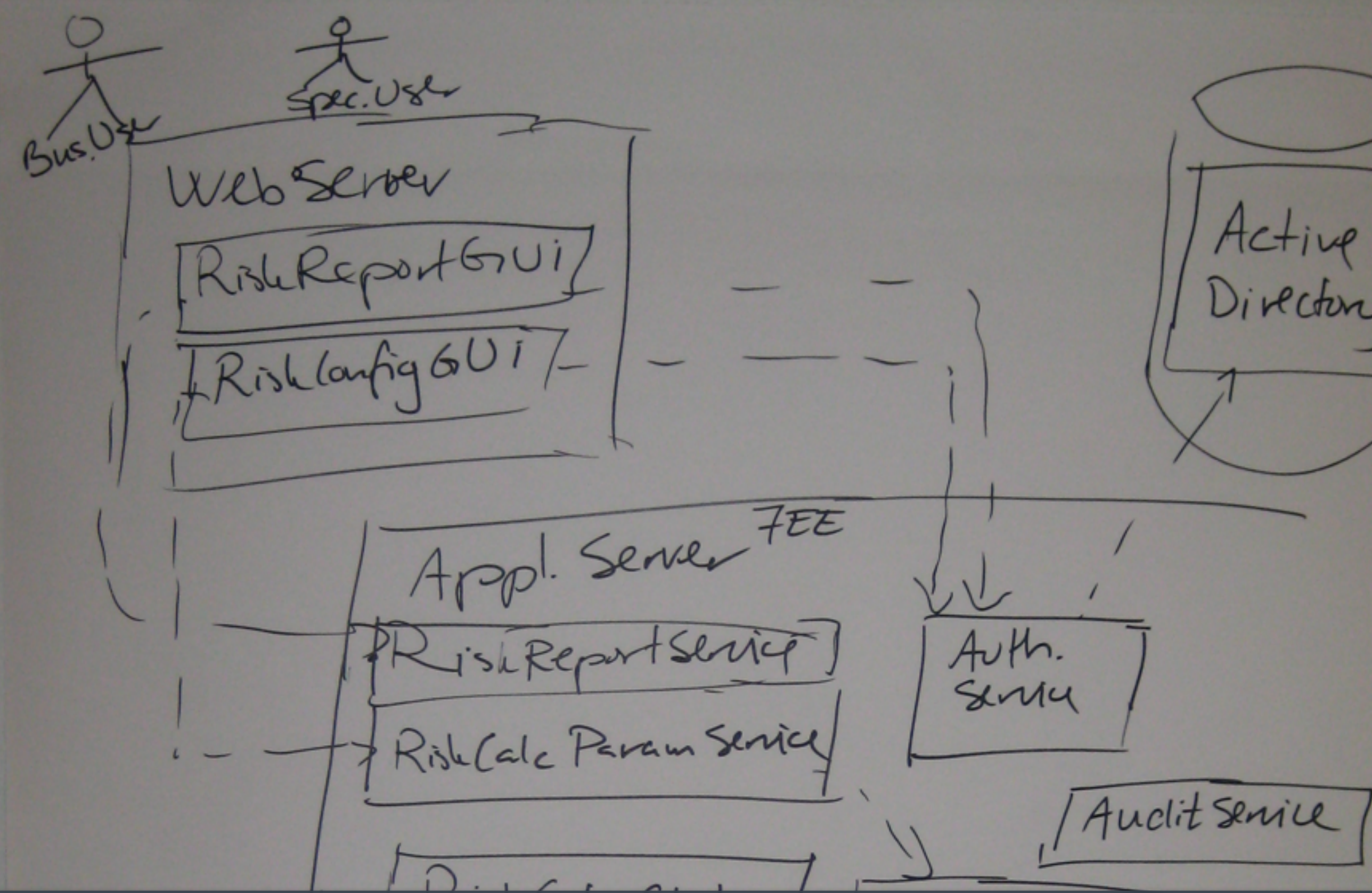
Generically True



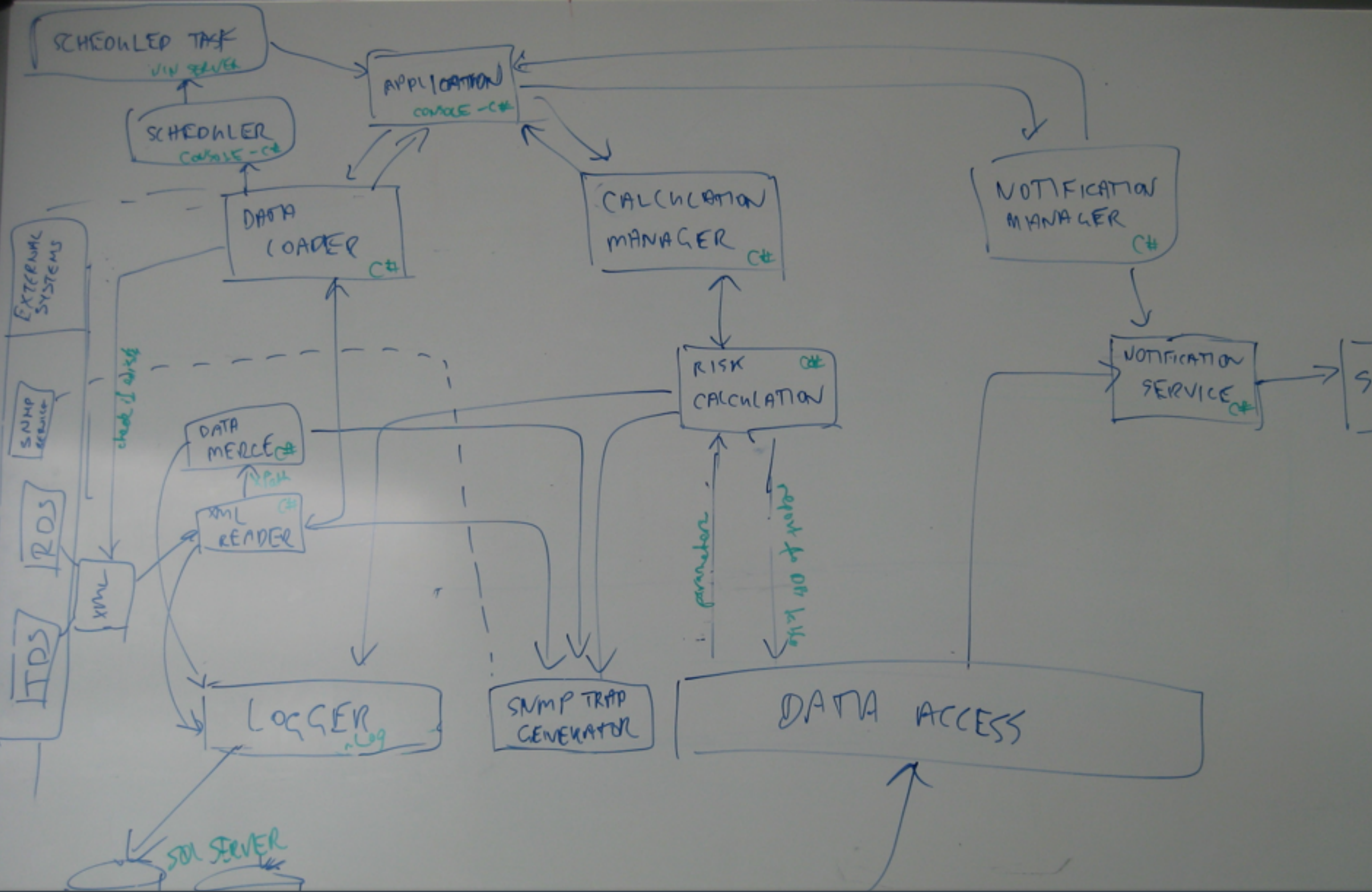
The Technology Deferral



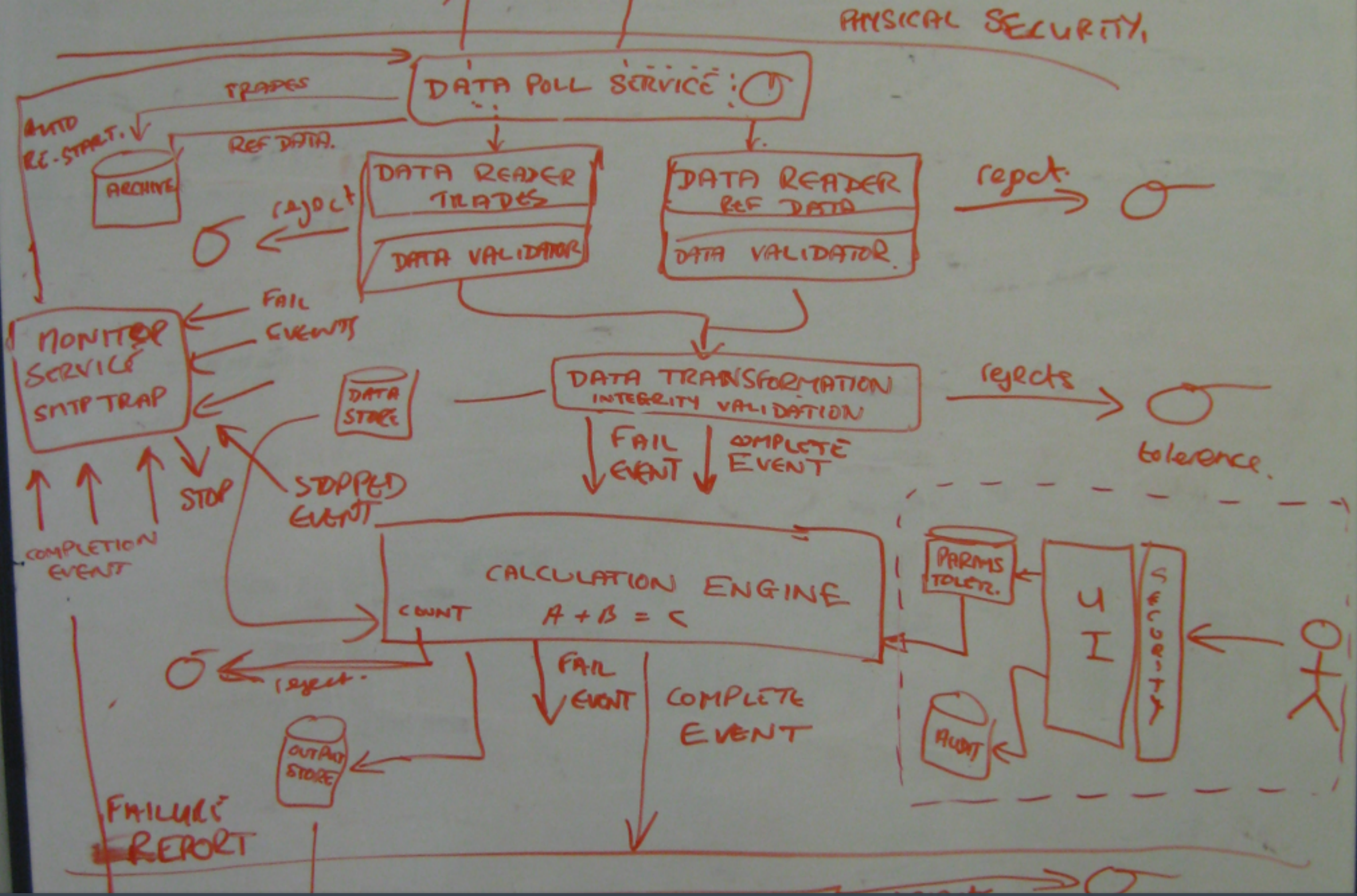
Missing Details



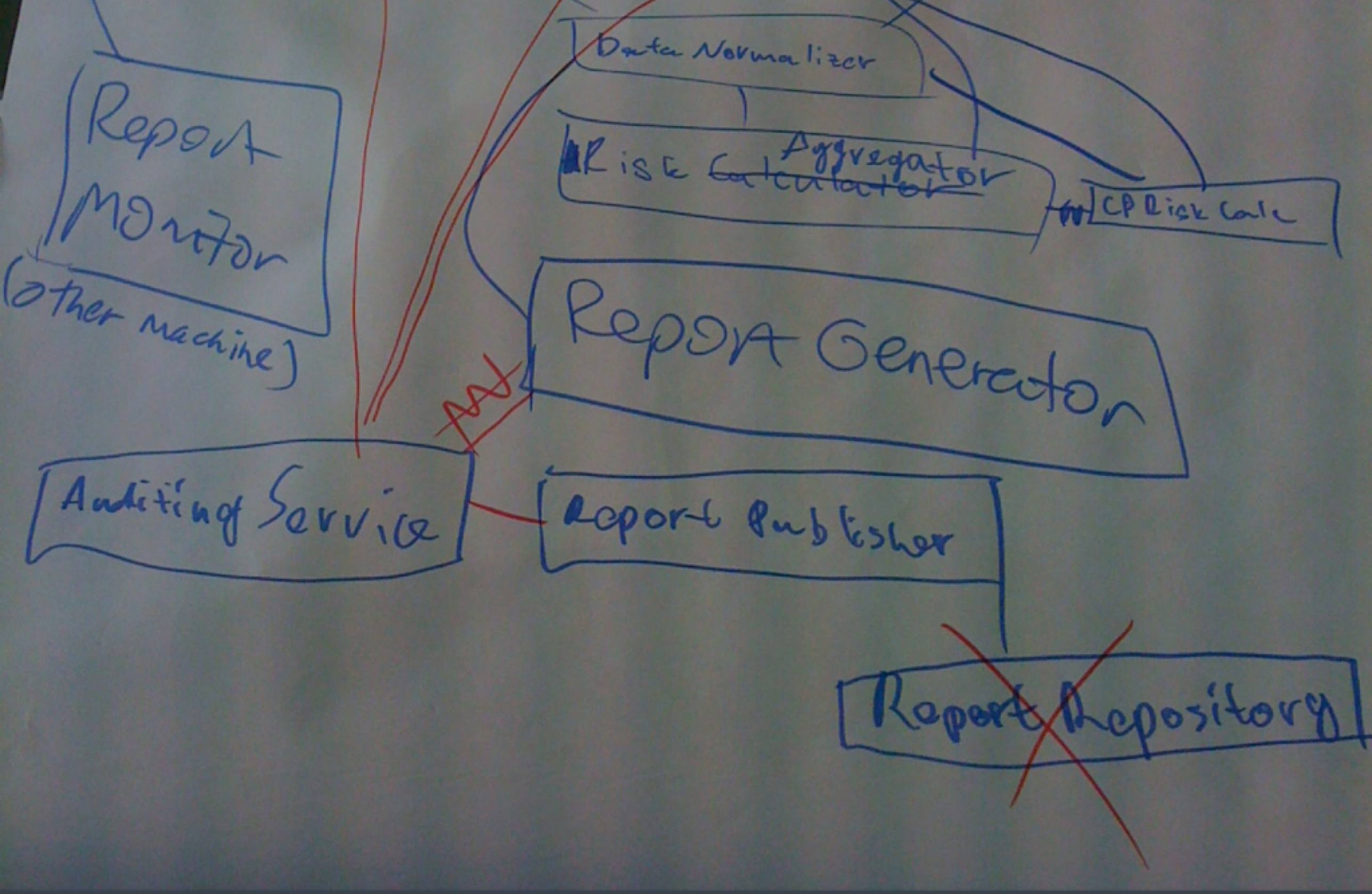
Assumptions are the mother of all ...



Homeless Old C# Object (HOCO)



Choose your own adventure



Should have used a whiteboard!

Params
Calcs
Trades
ref_current
Ref_hist
Logger
EH?
Batch
Date
- calcs

Eh?

Review the diagrams

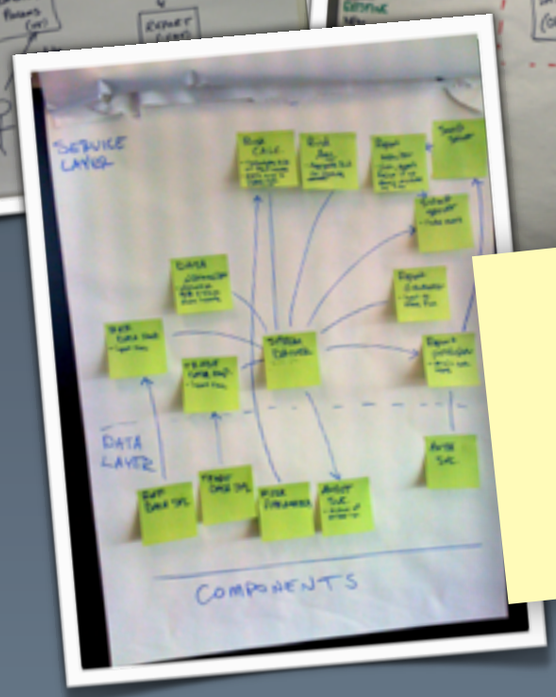
3+ things we like about the diagrams

3+ things we think would improve the diagrams

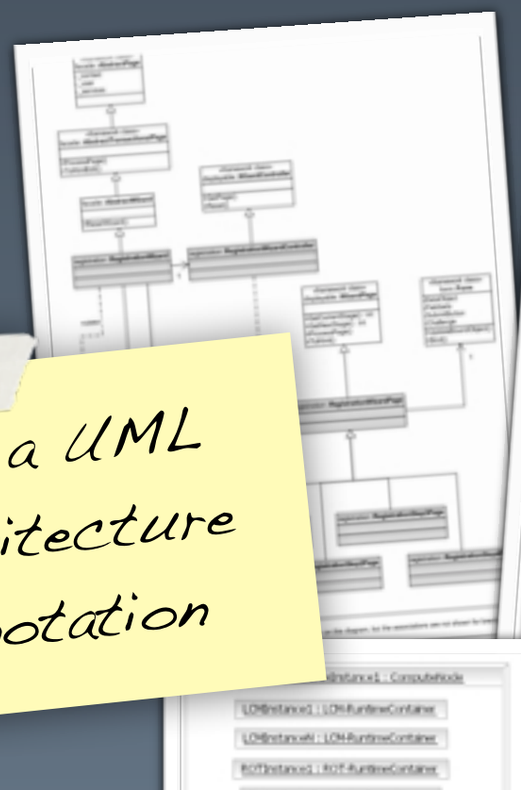
People expect to present their designs and therefore

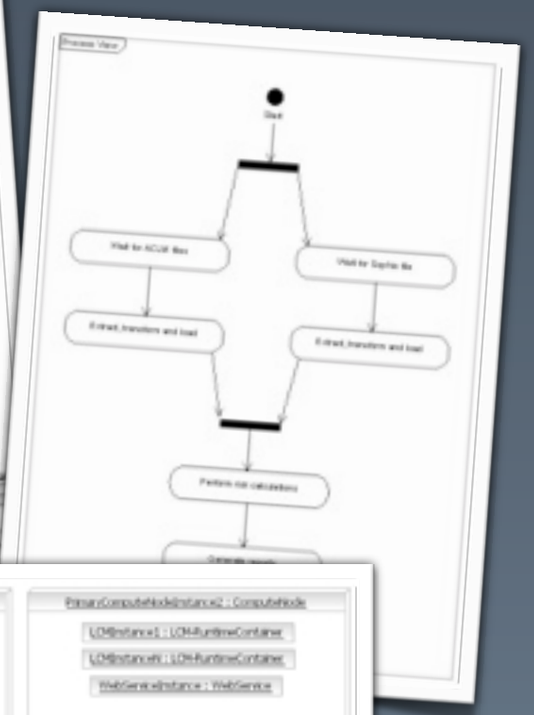
information is still stuck in their heads

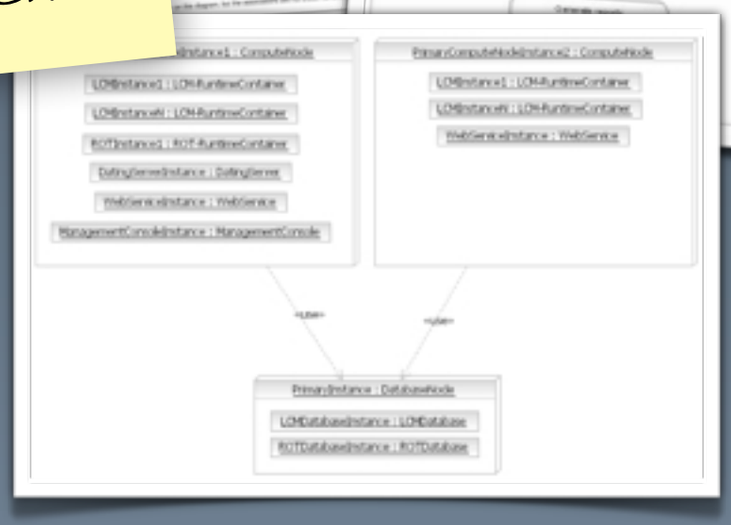
UML tool?



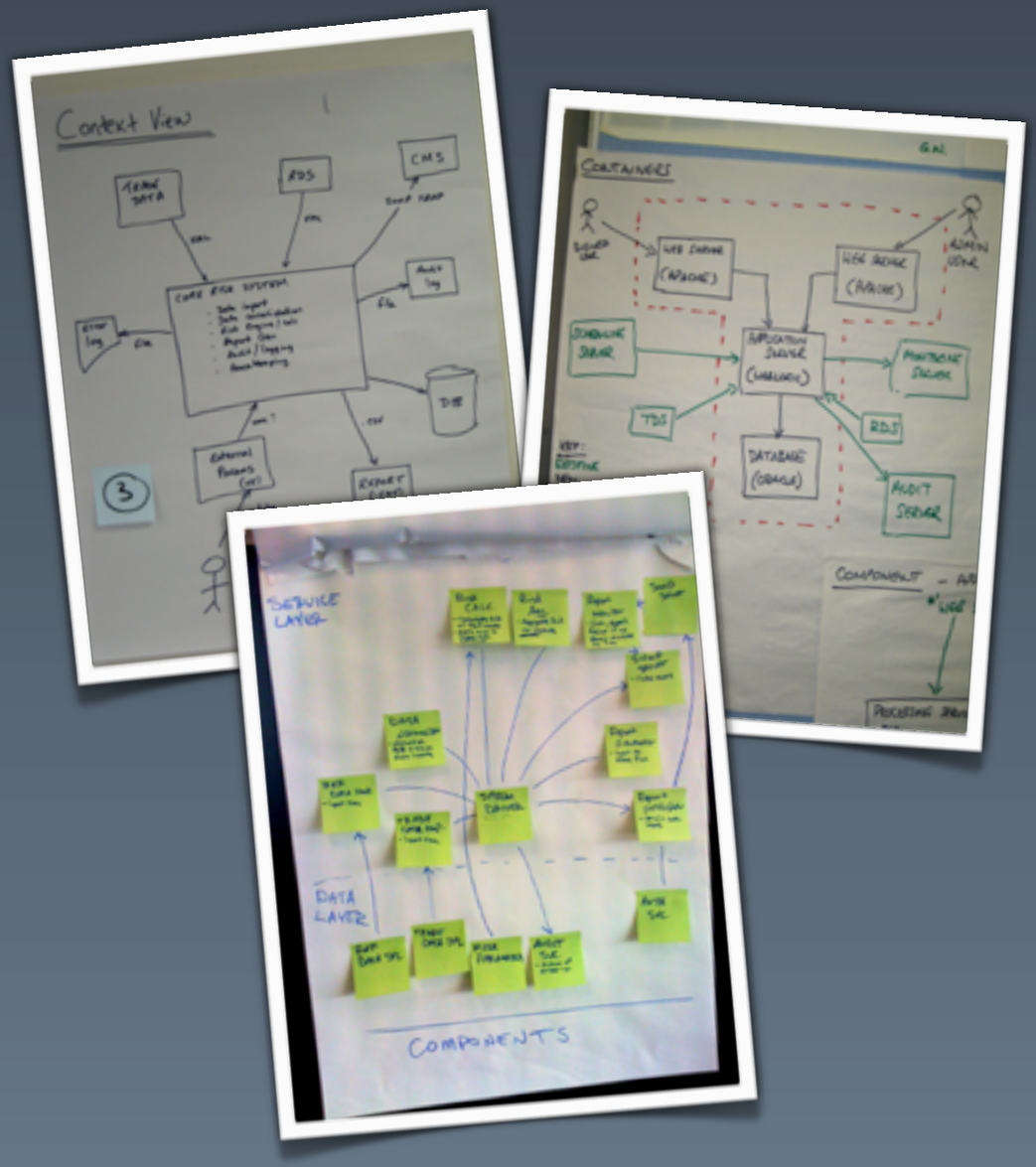
You don't need a UML tool to do architecture but agree on notation







Whiteboard or flip chart?

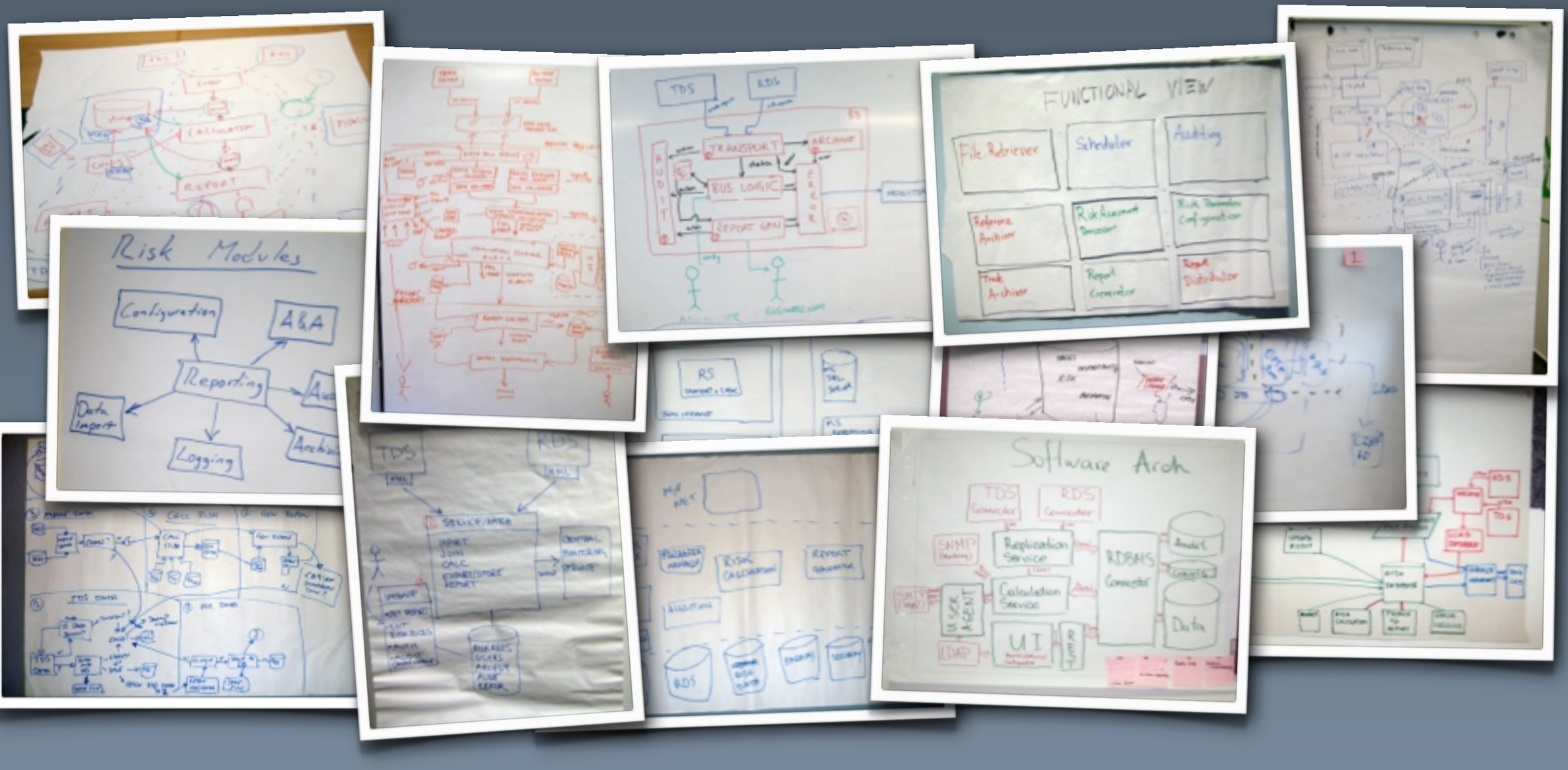


Collaborative design

(e.g. pair architecting)

NoUML

diagrams?



Challenging?

Level of detail
- where to stop

Who is the audience / different backgrounds

Implementation
- easy to get bogged down in detail

Type of diagrams

Notation

Documenting assumptions

(7) Challenging

Needed to ask questions / make assumptions

Temptation to focus on detail
- when do we stop?

How much detail?

Talked about more than the diagrams

What notation?
- boxes
- arrows

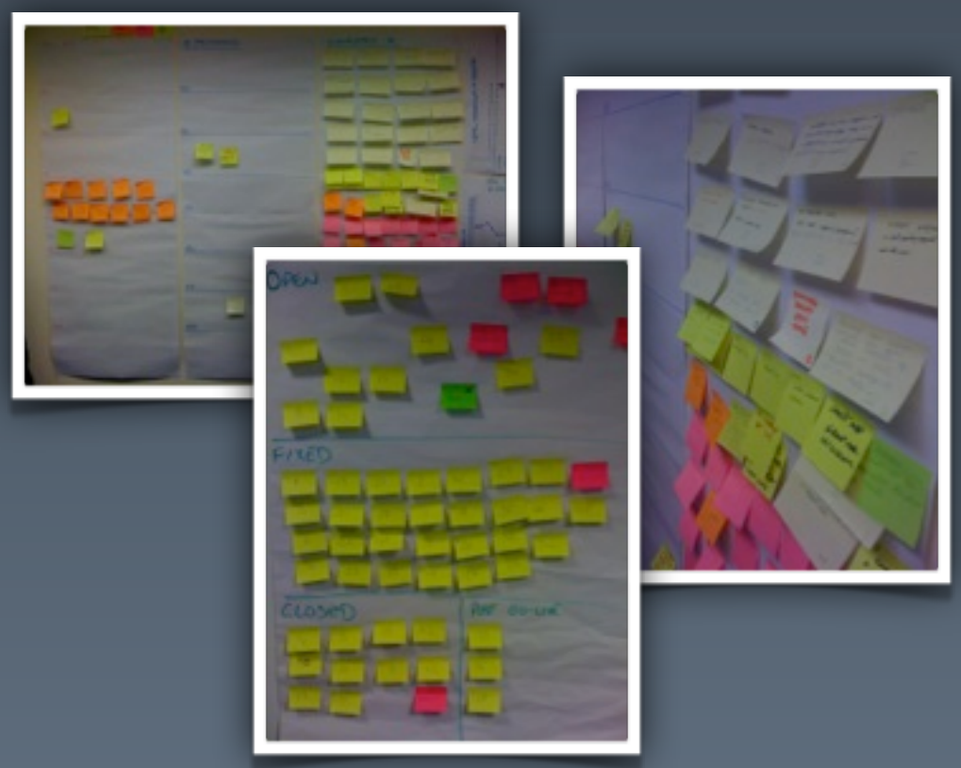
(10) Challenging?

Verifying our own assumptions

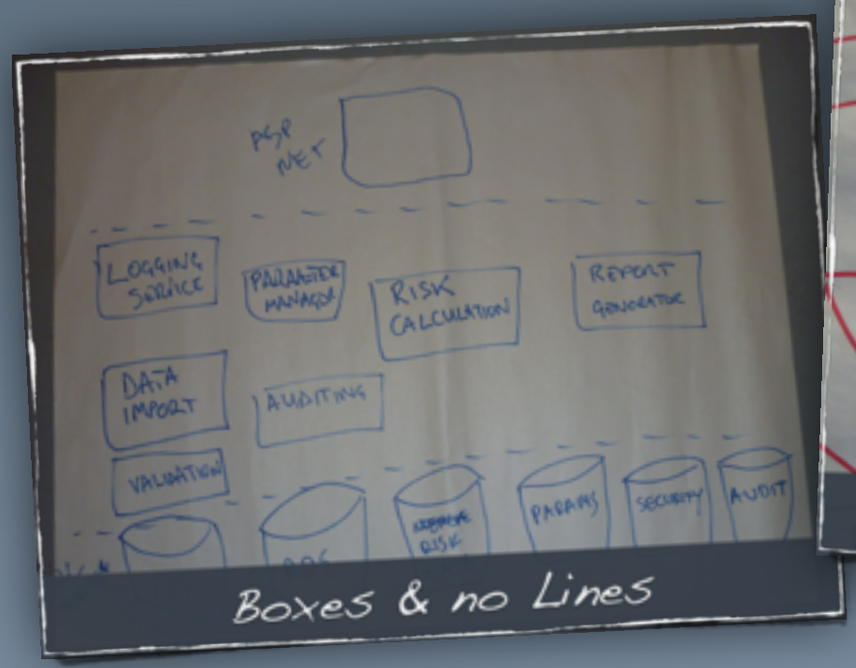
Expressing the solution
- communicating it in a clear way
- use of notation
- easy to mix levels of abstraction
- how much detail?

What's been challenging about the exercise?

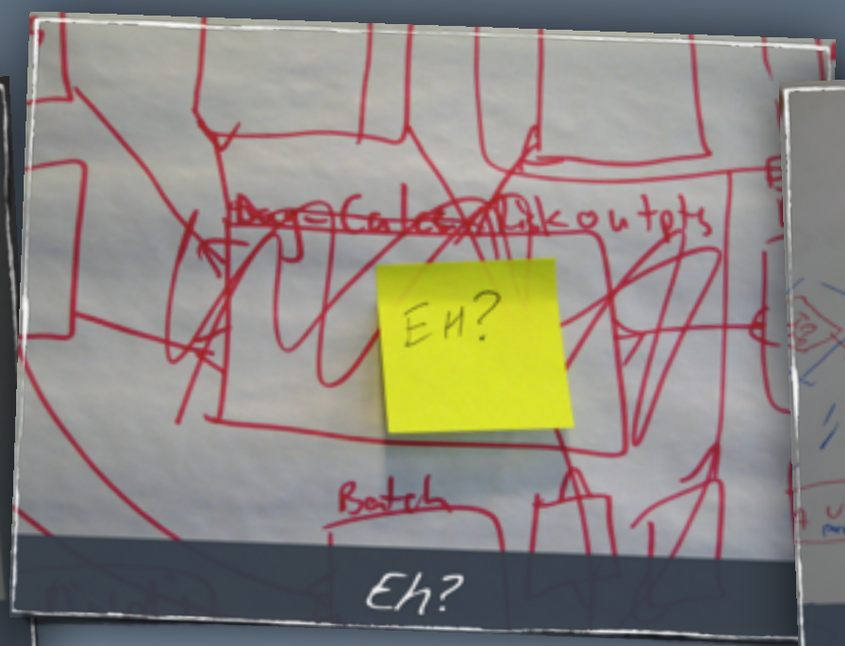
We can visualise our process…



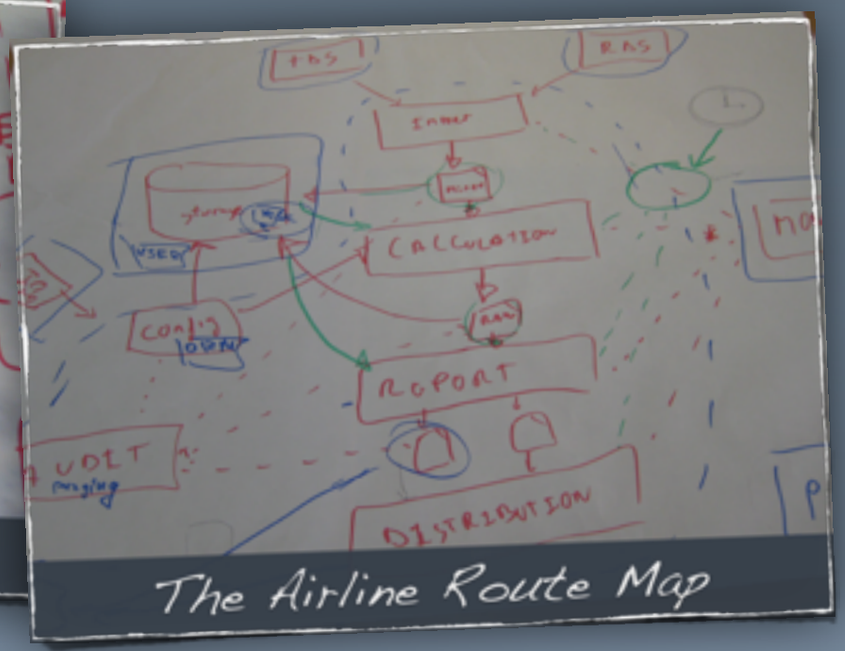
…but not our software!



Boxes & no Lines

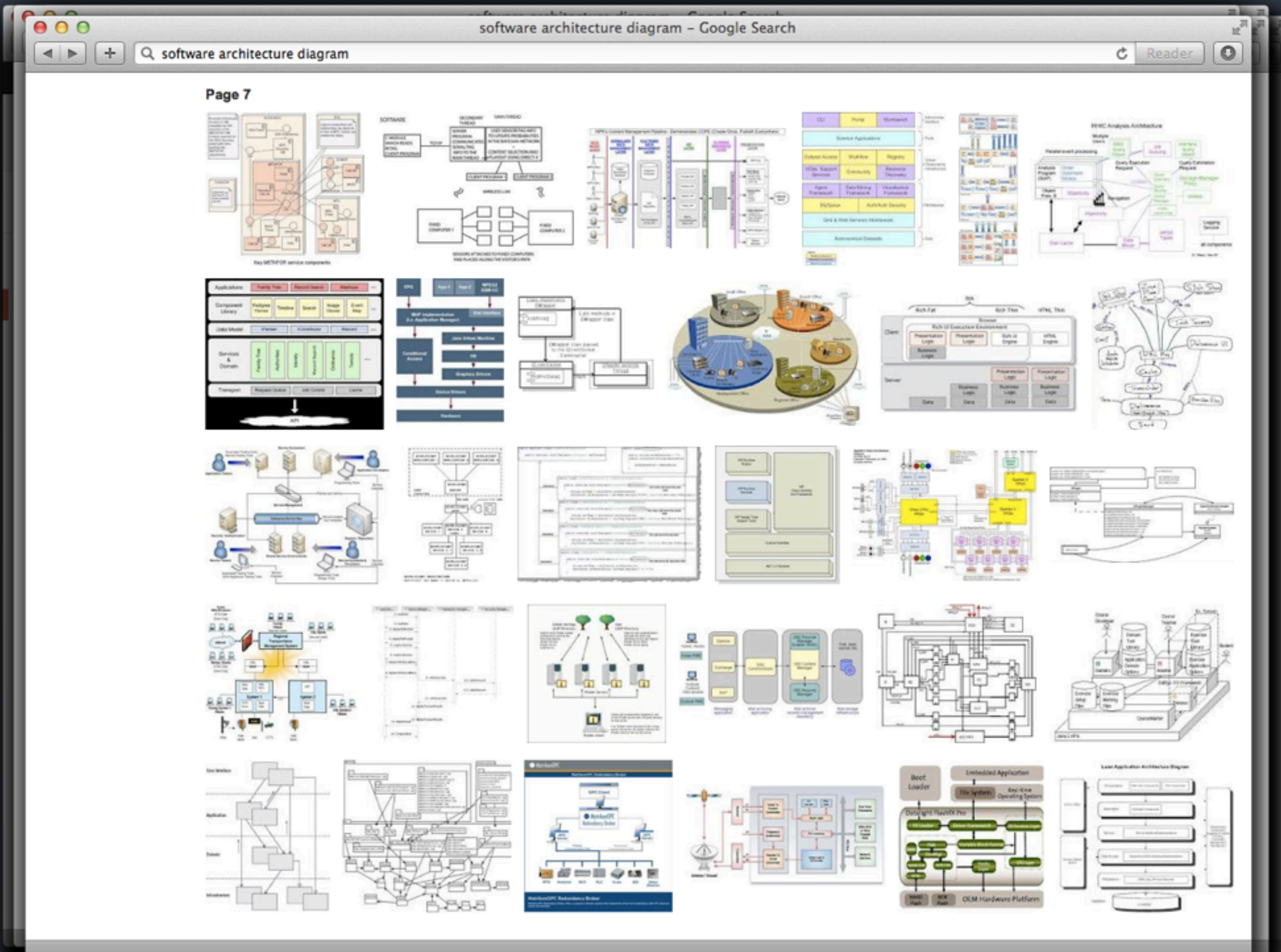


Eh?



The Airline Route Map

Why?


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

Video over IP

Capturing video frames from Camera

Format conversion and Pre-processing

H.264 Encoder

NALU Fragmentation

Audio/Video sync module

Format conversion and Post-processing

H.264 Decoder

Reconstruction of NALU

RTP/ RTCP

Voice over IP

Acoustic echo cancellation (AEC)

LEC (G.168/ G.165)

AGC - Gain Control

VAD

Speech Encoder

Audio Interface layer

Sound card Interface (PCI / USB)

FXS/TDM Interface

Telephony Interface layer - Tone detection/generation, CPT/DTMF, MF-R1/R2, and Voice Fax Modem Discrimination Unit

Administration UI

Reservation Manager UI

AutoShell

Packaging

Process Automation

Configuration Management / Compliance

Help Desk

Reporting (Jasper)

Agent Remote Deployment

Service Response Monitoring

CA IT Client Manager

CA Patch Manager

CA Process Automation

CA Configuration Automation

CA Service Desk

Chargeback

Agent Policy Configuration

Server Monitoring (Local & Remote)

Web Services and Reliable Transport (Active MQ)

Storage

Platform Management

Microsoft HyperV

Provisioning

Rapid Server Imaging

Automation Infrastructure Platform

AOM Data Service

Policy, Performance, and Resource Management

Authentication / Security (CA EEM)

State and Event Management

Scheduler (Windows)

Discovery (CA Network Discovery Gateway)

Management DB

Performance DB

Service Controller

Rich UI

Web UI

Controls

Service Interface

Activity

Business Rules

Human Workflow

Workflow

Service Agents

DAL

E-Publish

EAI

ECM

DW

OLTP

Signing

Authentication

Authorization

Monitoring

Log & Trace

Exception Management

Configuration

Presentation Sublayer

Application Sublayer

Project Editing Service

Project Management Service

Computational Analysis Service

e-AIRSmesh

e-AIRSview

Mesh Converter

CFD Solver

Output Converter

Remote Experiment Service

Exp. Exec.

Exp. Info.

Aerodynamic Exp.

PIV Exp.

Parametric Study

Validation

Analysis

Collaboration

Session Reservation

Resource Configuration Service

Resource Monitoring Service

Authentication

P / HTTP

ASP (Scientific Application Service Provider)

Basic Execution Service

Parameter Sweeper

Global Scheduler

Resource Catalog

GRAM adapter

Plotting Service

eAIRS Plot

Conv. Graph

Data Transfer Service

Metadata Management Service

AG + VNC

Visualization Sharing

Mobile Service Component

A/V Conference

Chat

File Editor

Library

ASCII Files

SCADA Develop. Environ

Export / Import

Commercial DB

Project Editor

Commercial Develop tools

Driver Toolkit

HMI

Alarm Display

Log Display

Active X Controls

3rd Party Applic.

Active X Container

Client - Server - Publish / Subscribe - TCP/IP

SCADA Server

Recipe DB

Recipe Managt

Ref. DB

Data Proces-sing

RT DB

Report Gener.

SQL

Alarm

Alarm DB

Log

Log DB

Archive

Archive DB

ODBC

DDE

API DLL

Data R/W

Driver

OPC

Private Application

EXCEL

VME

PLC

PLC

Typical generic software architecture of SCADA systems

ote administration interface

Output device manager

Output devices

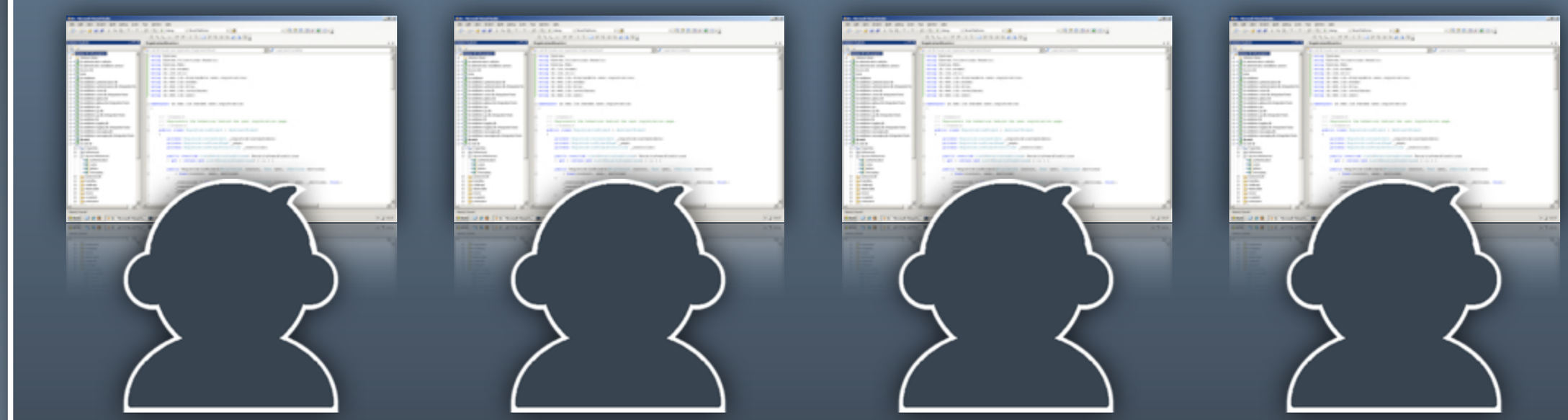
User manager

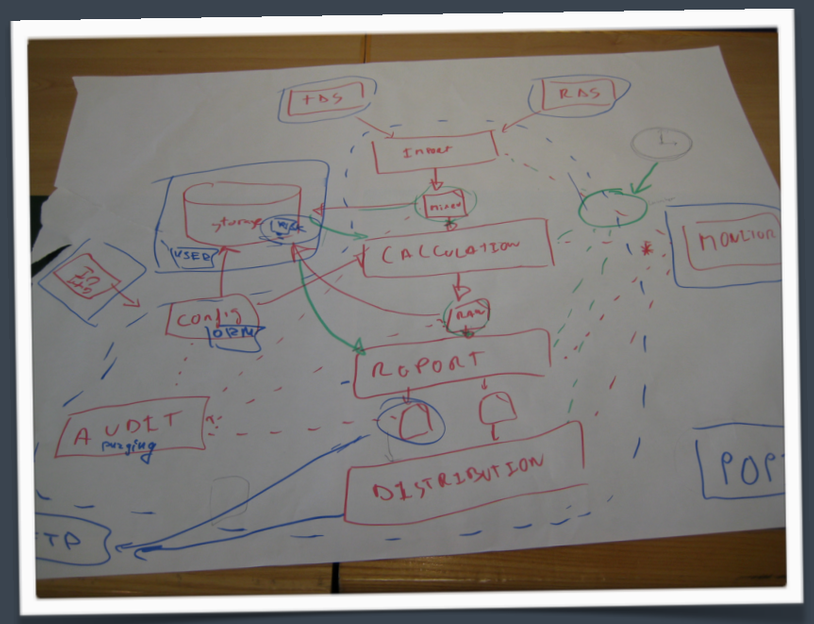
DB-Server

Analysis

Shared vision of

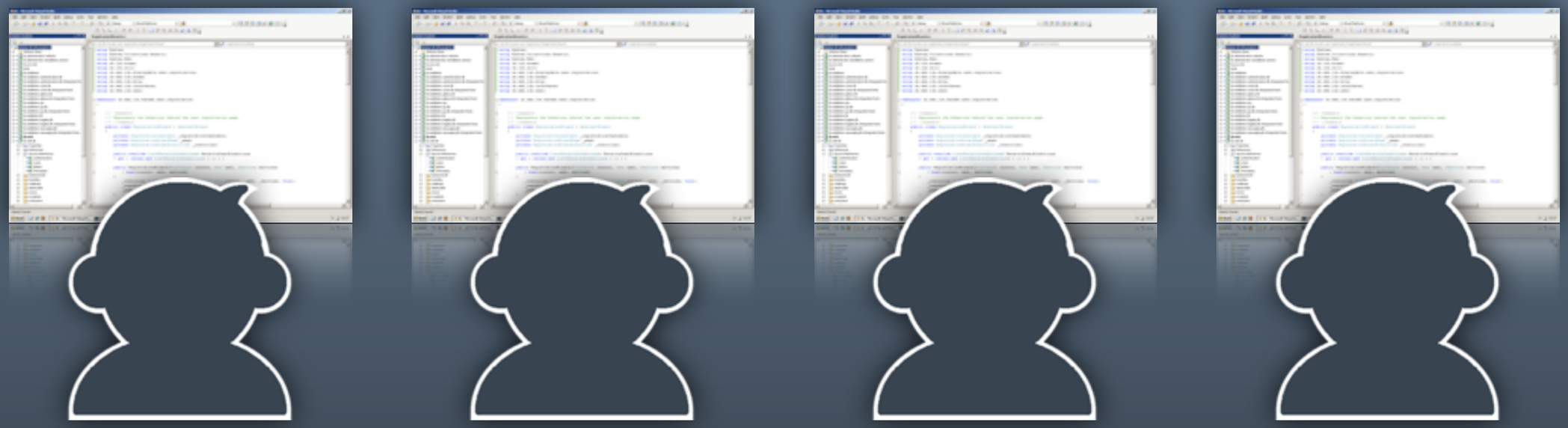
TL; DR

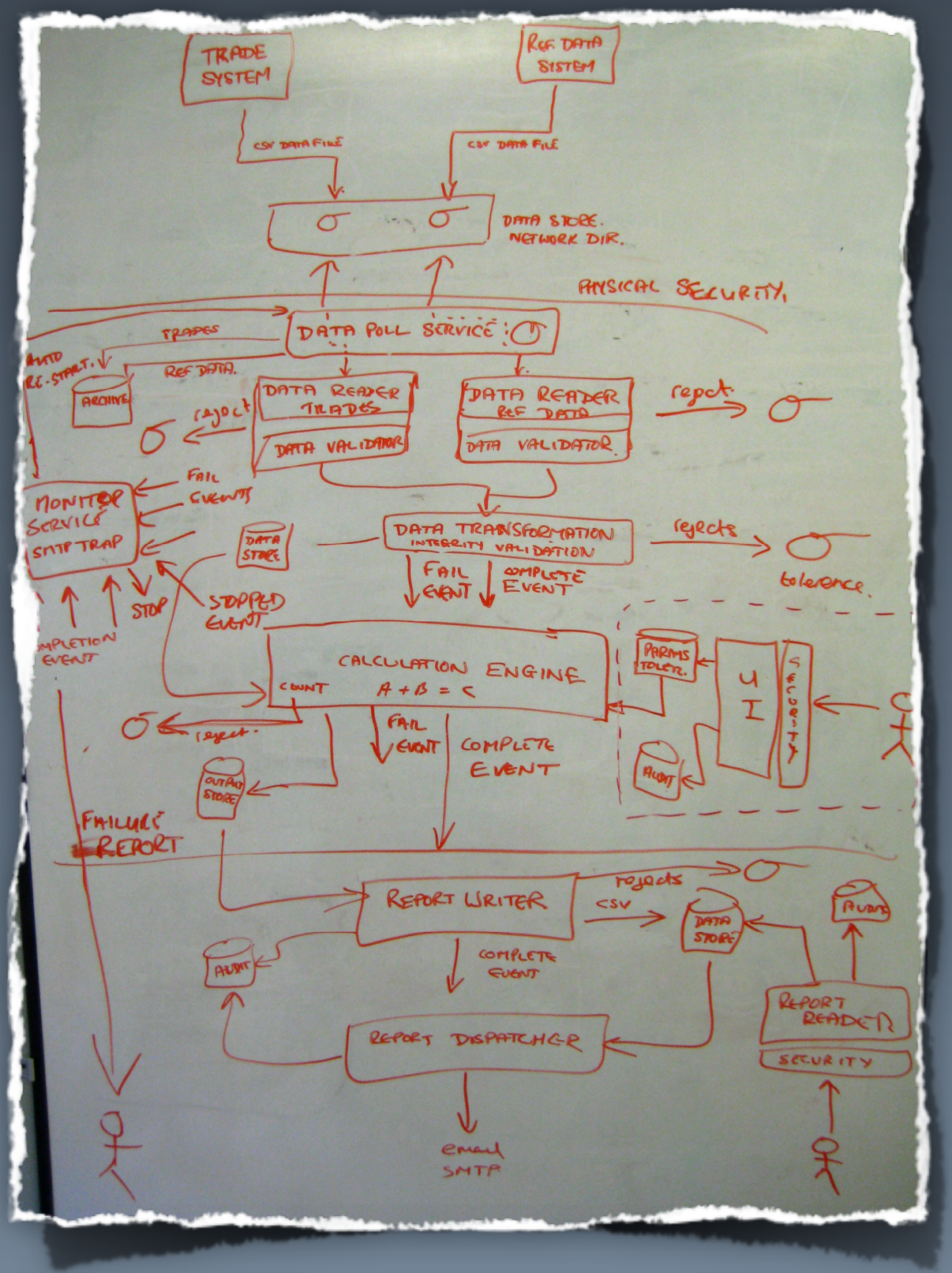




Shared vision of

WTF?!





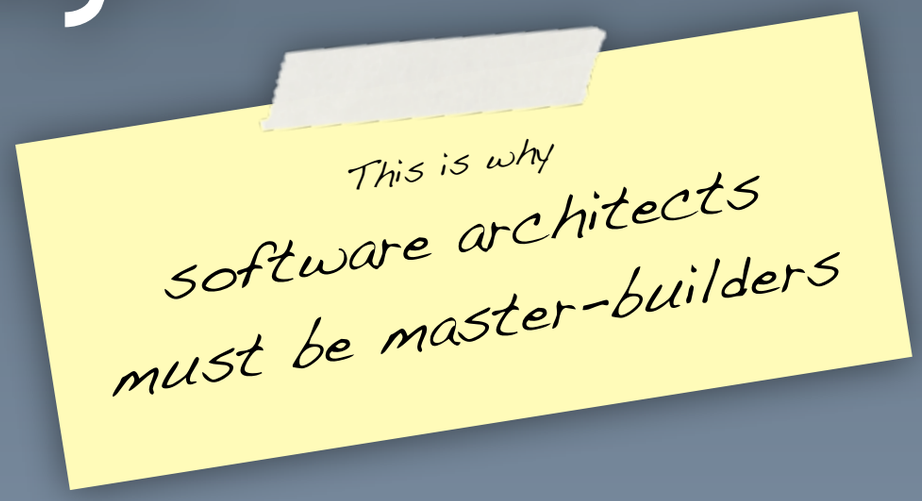
It's usually difficult to show the entire design on a **single** diagram

Different **views** of the design can be used to manage complexity and highlight different aspects of the solution

Would you

code

it that way?



Software System

iner
tion server, standalone
e, file system, etc)

Container
(e.g. web server, application server, standalone application, browser, database, file system, etc)

Cont
(e.g. web server, applic
application, databa

Component

Component

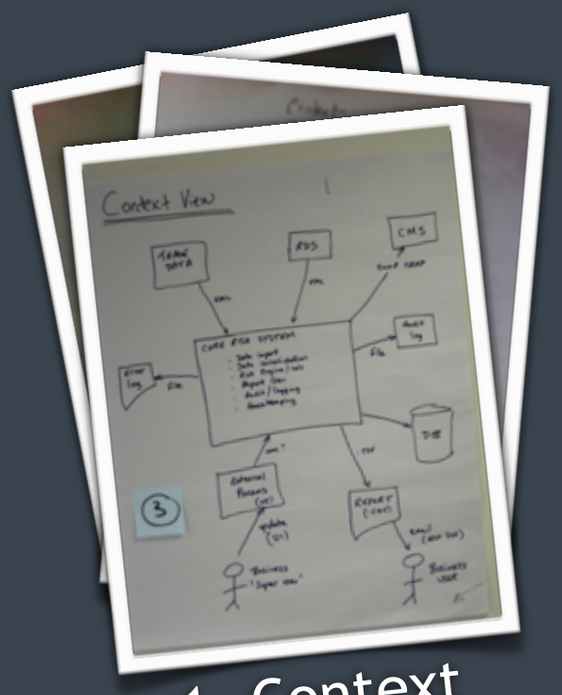
Component

Class

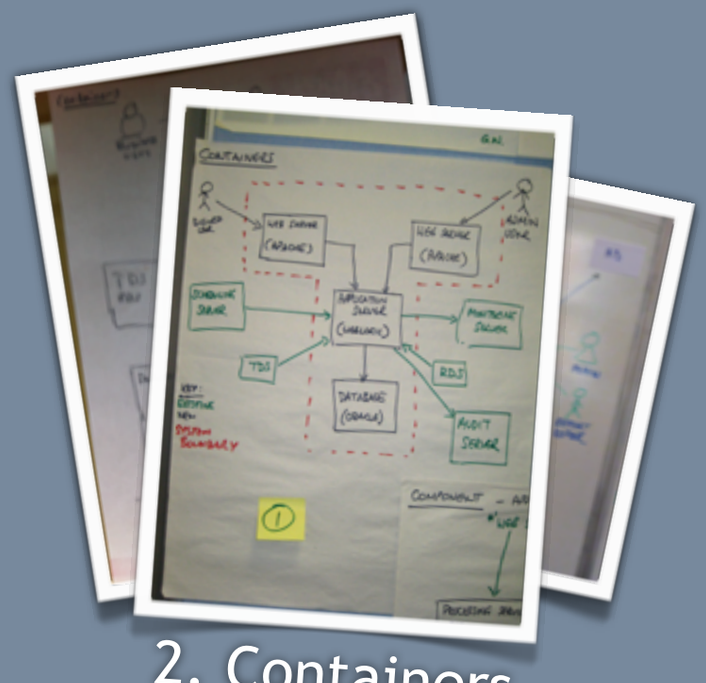
Class

Class

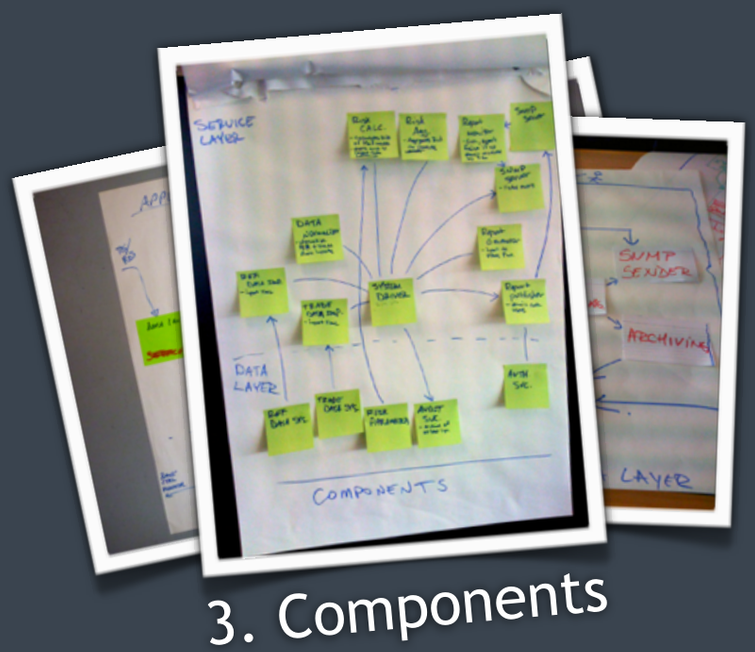
Agree on a simple set of abstractions that the whole team can use to communicate



1. Context



2. Containers



3. Components

C4

- Context
- Containers
- Components
- Classes

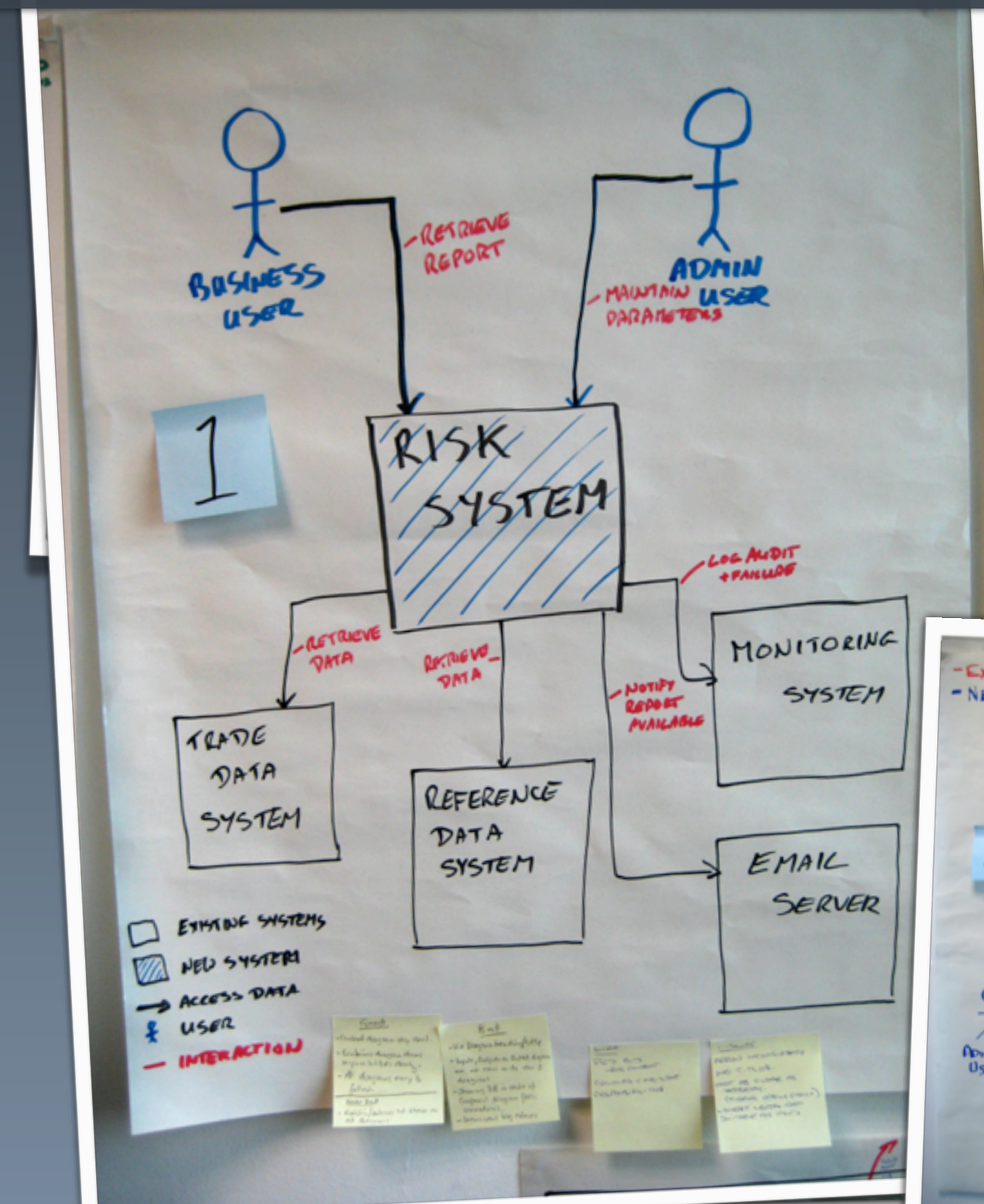
This only covers the static structure (runtime, infrastructure, deployment, etc are also important)

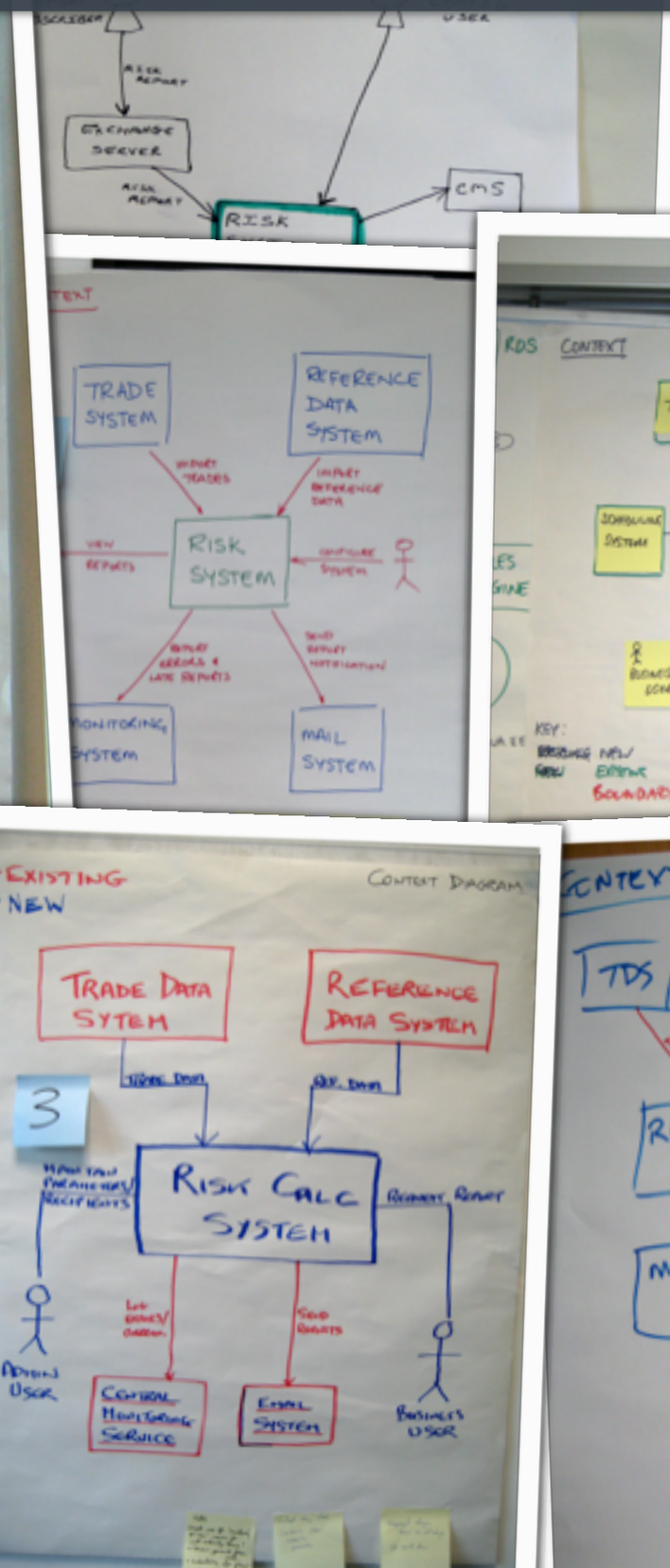
… and, optionally,
4. Classes

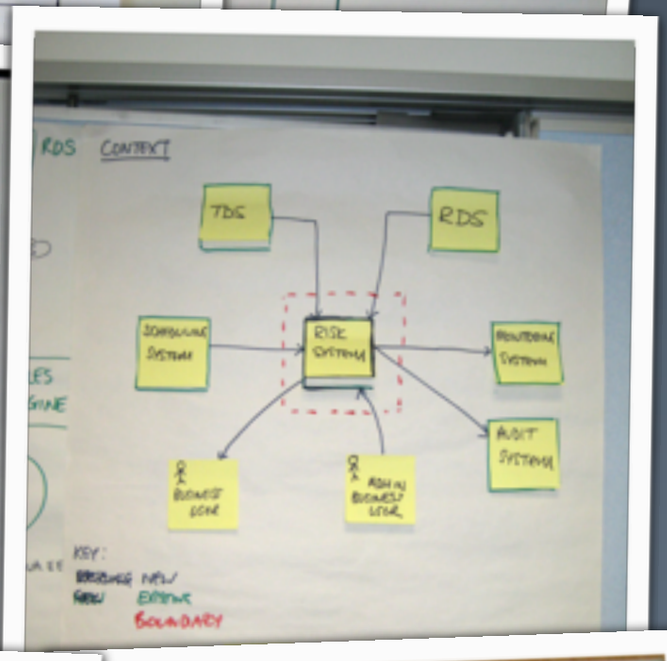
Thinking inside the box

Context

- What are we building?
- Who is using it? (users, actors, roles, personas, etc)
- How does it fit into the existing IT environment?

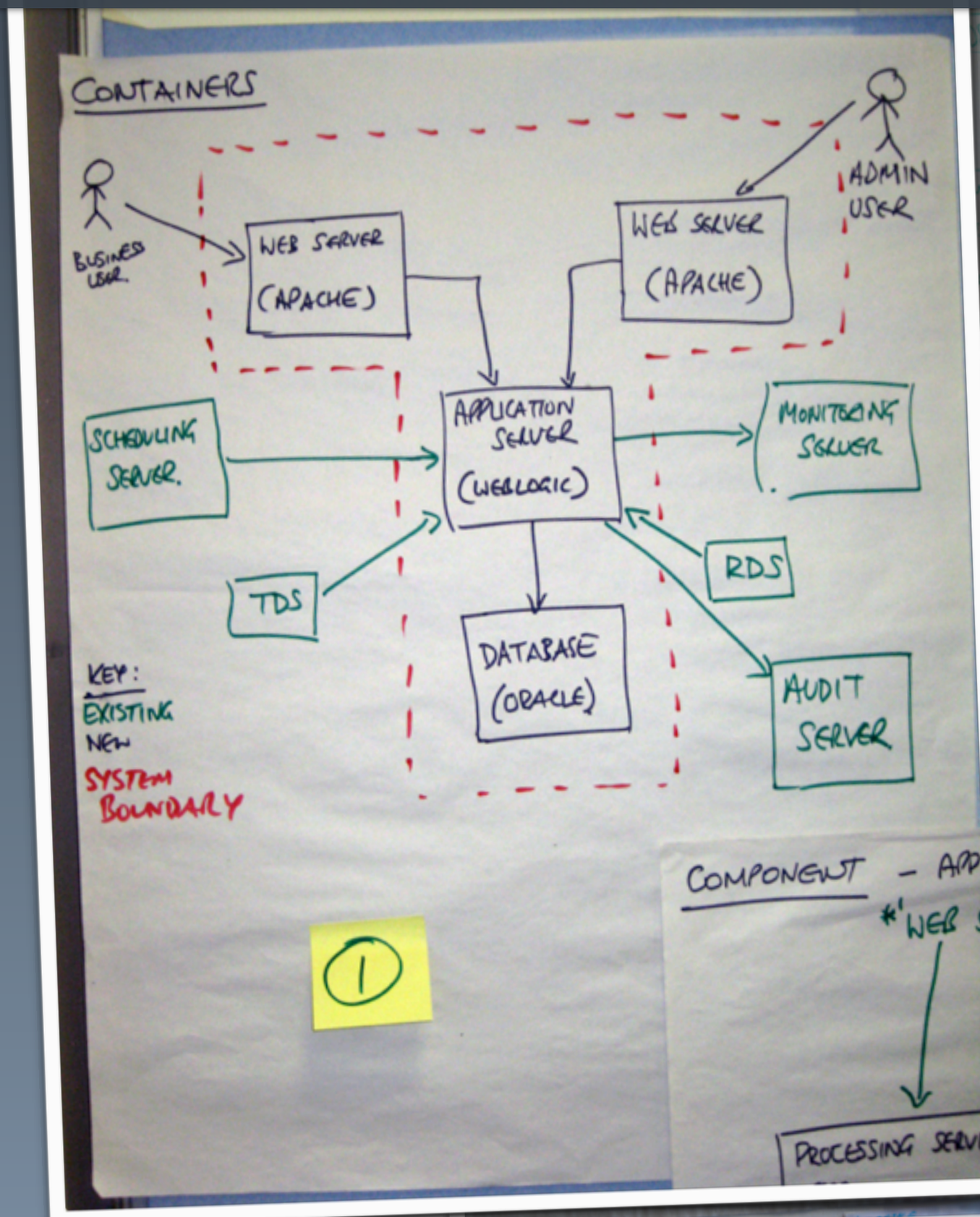


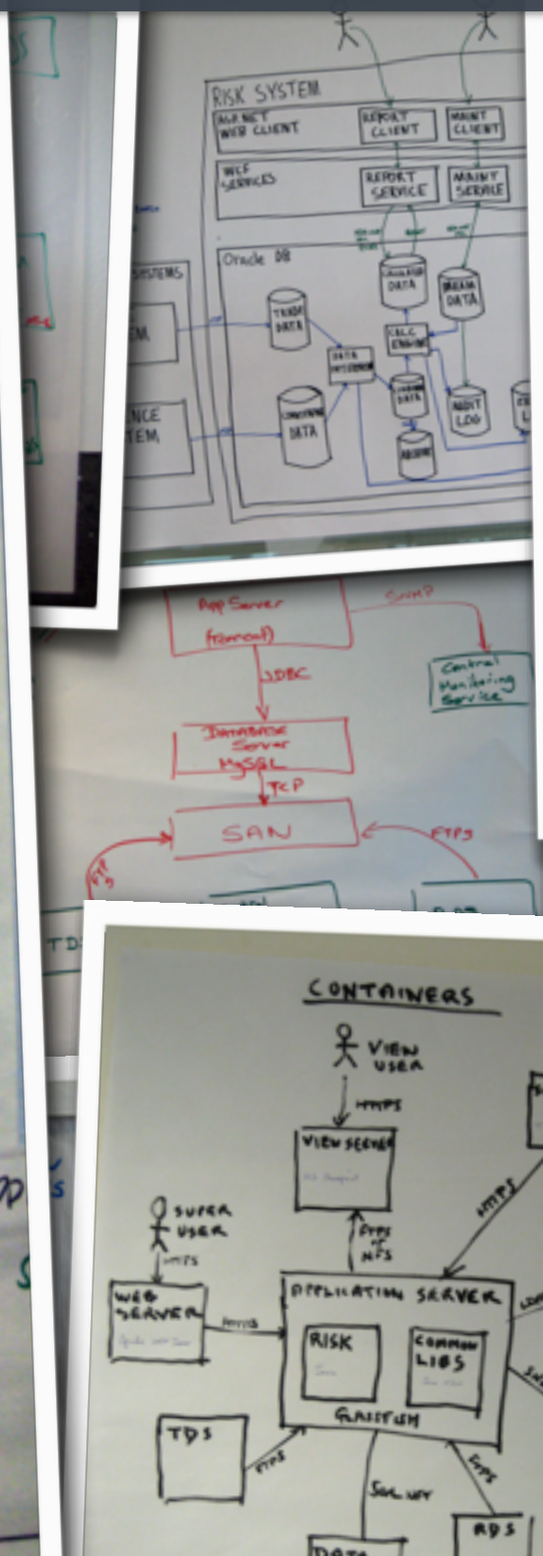


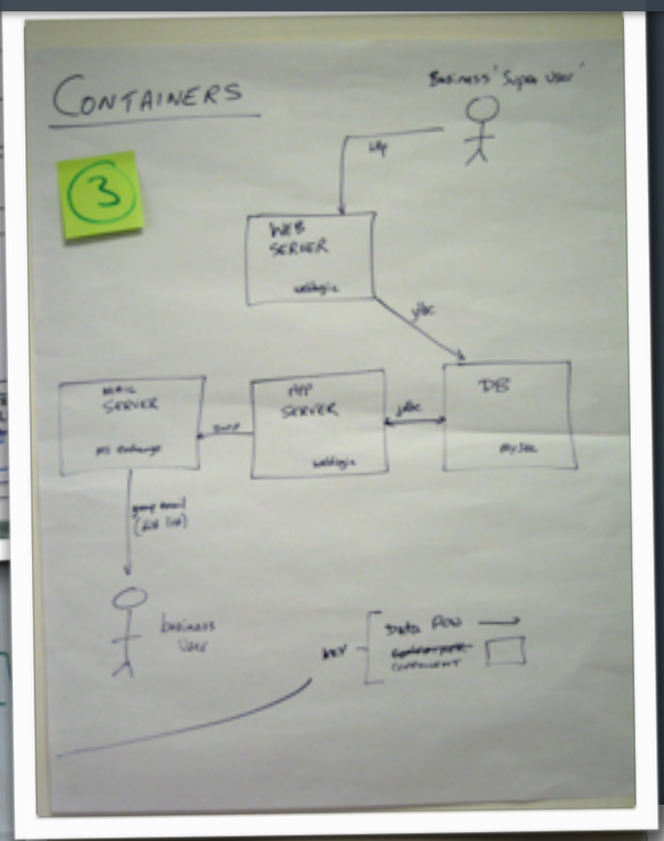


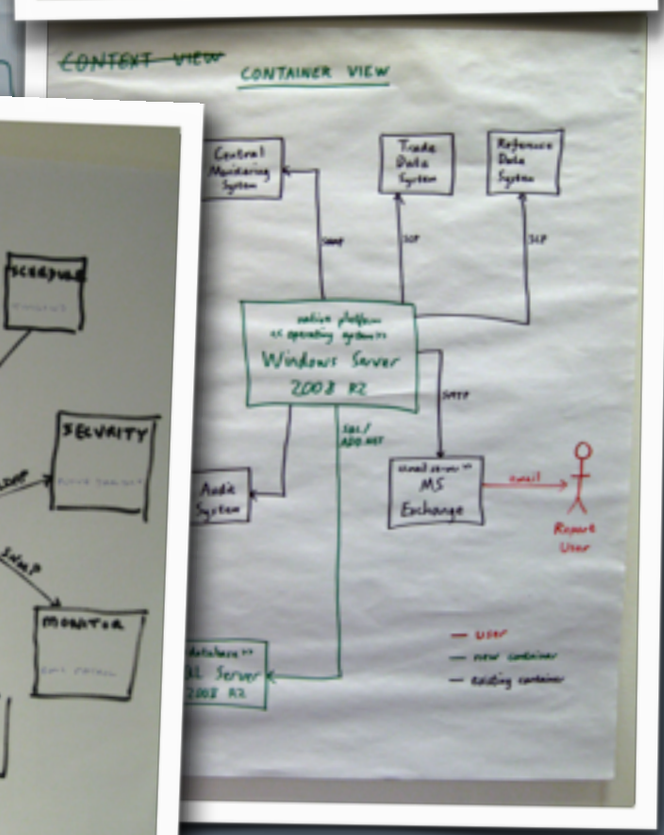
Containers

- What are the high-level technology decisions?
- How do containers communicate with one another?
- As a developer, where do I need to write code?



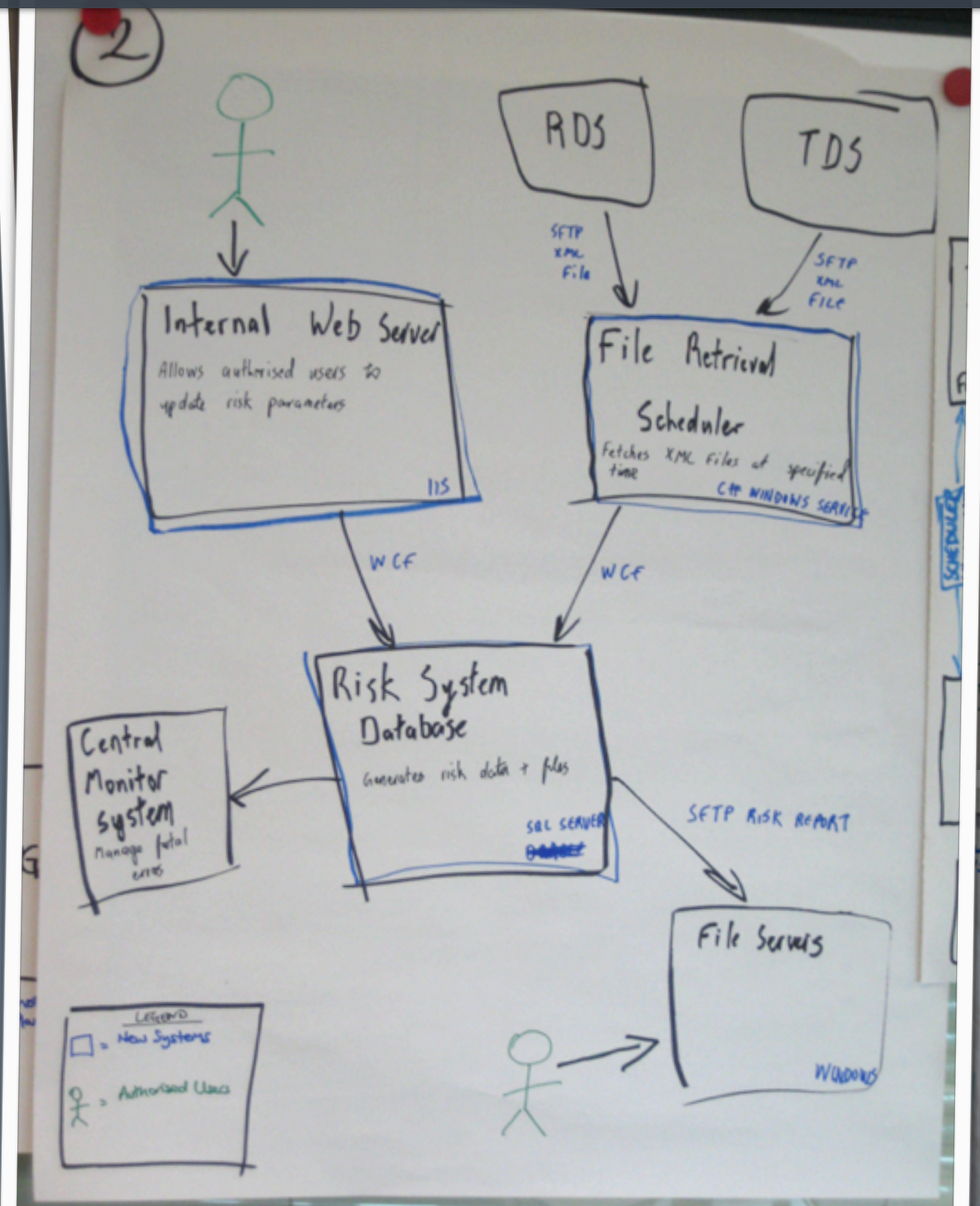


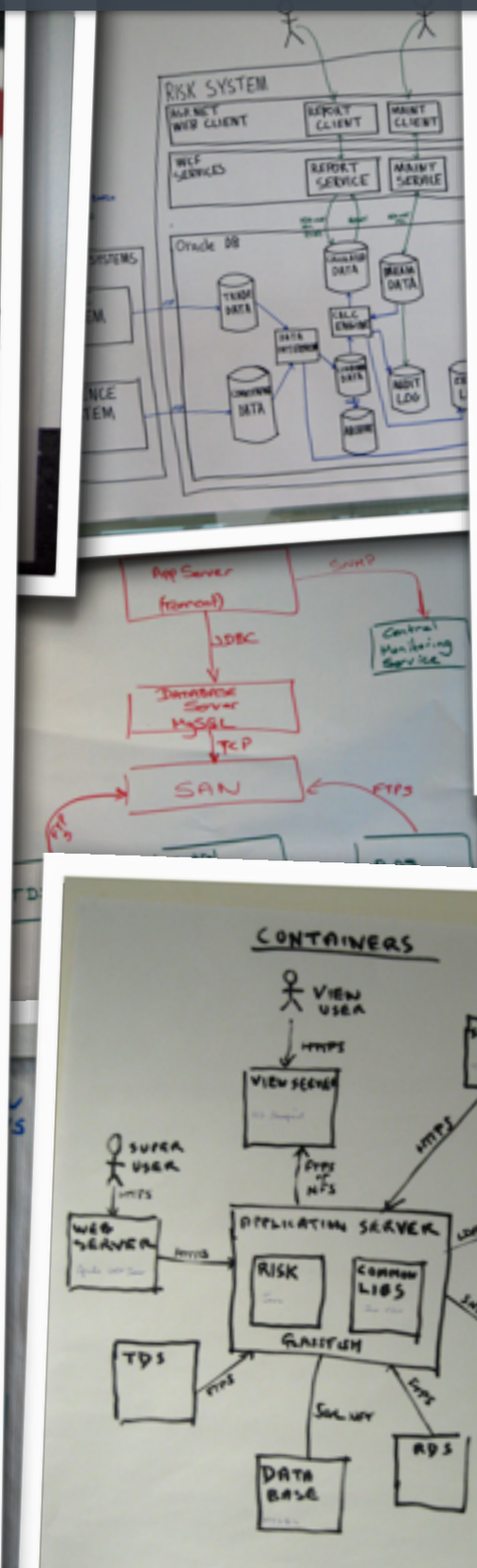


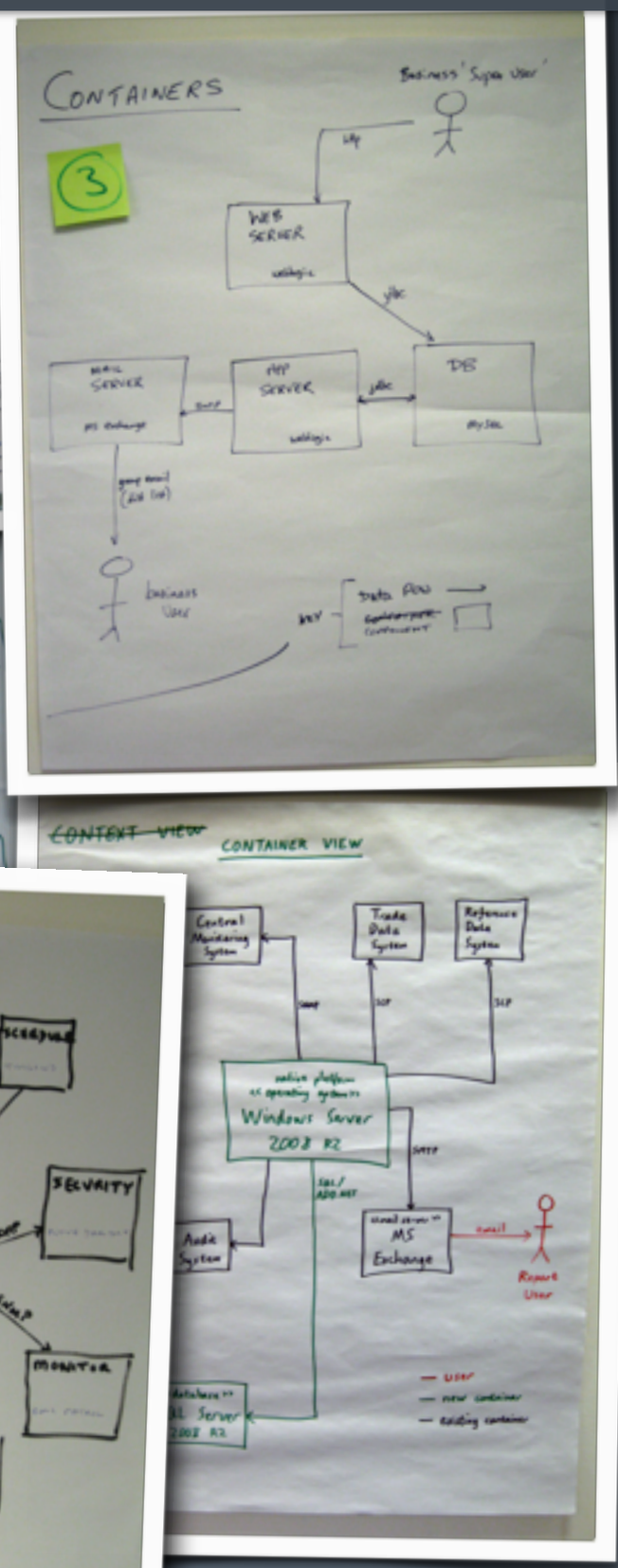


Containers

- What are the high-level technology decisions?
- How do containers communicate with one another?
- As a developer, where do I need to write code?

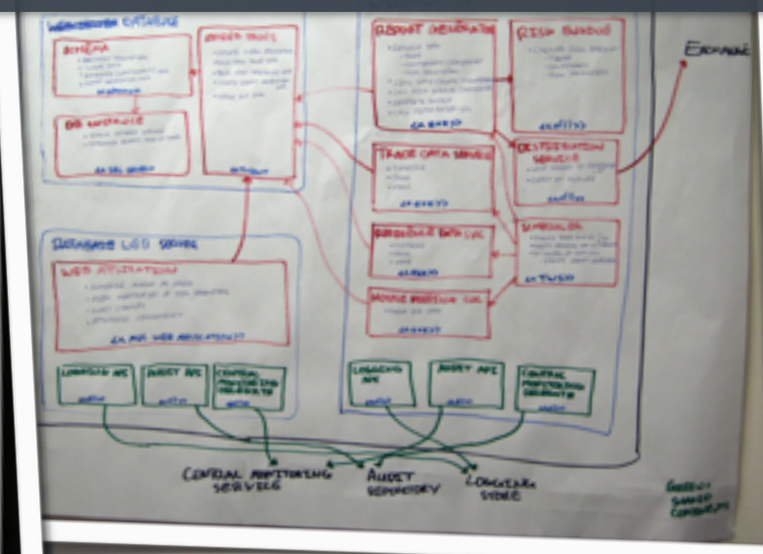


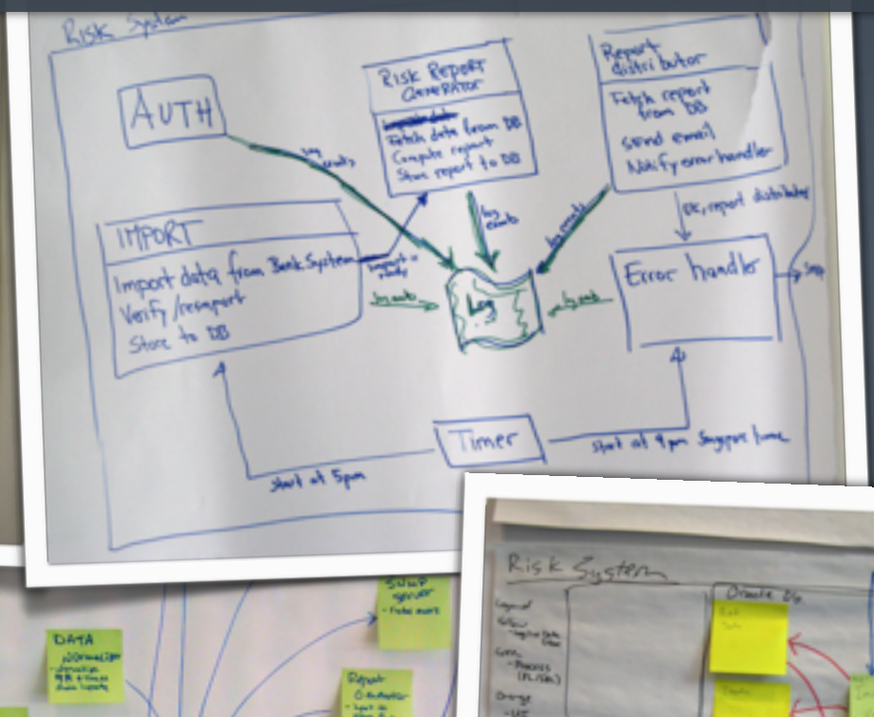


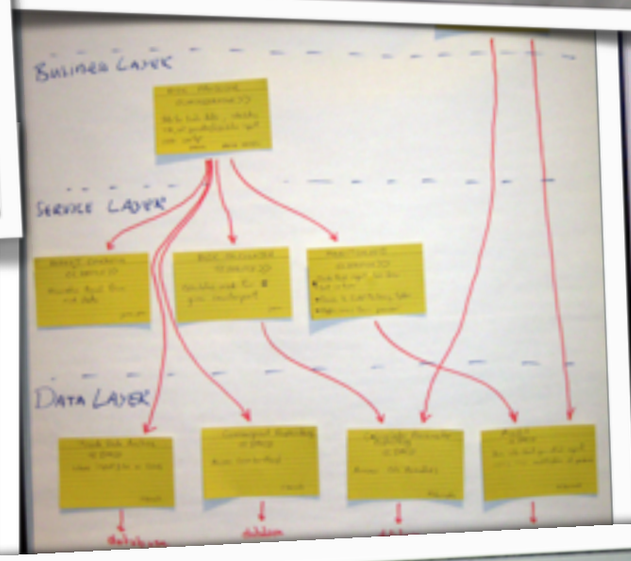


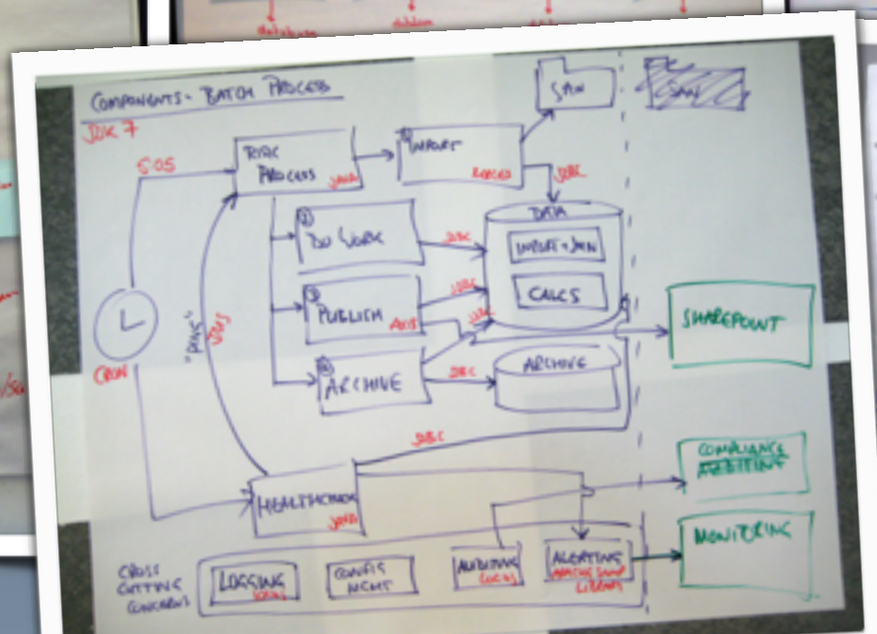
Components

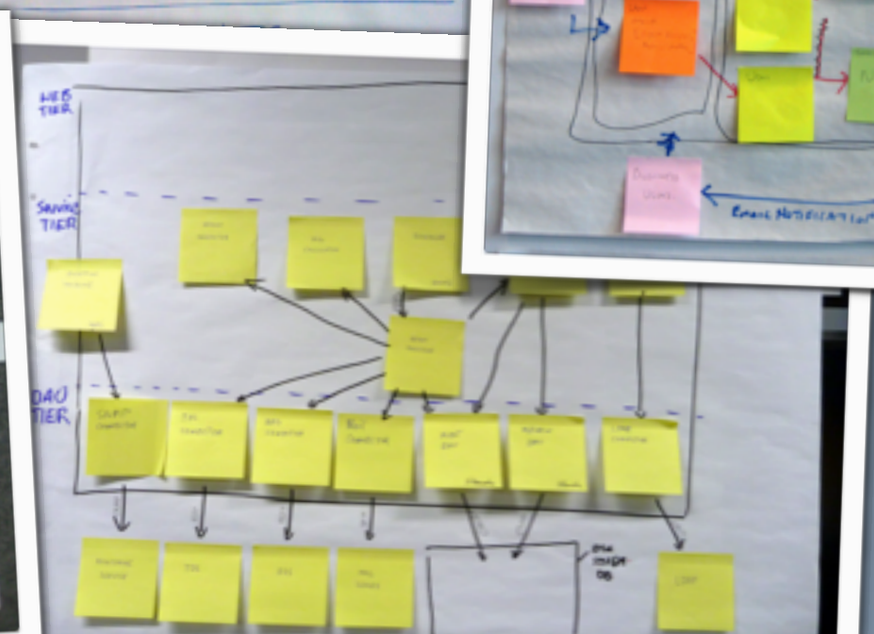
- What components/services is the system made up of?
- Is it clear how the system works at a high-level?
- Do all components have a home (a container)?





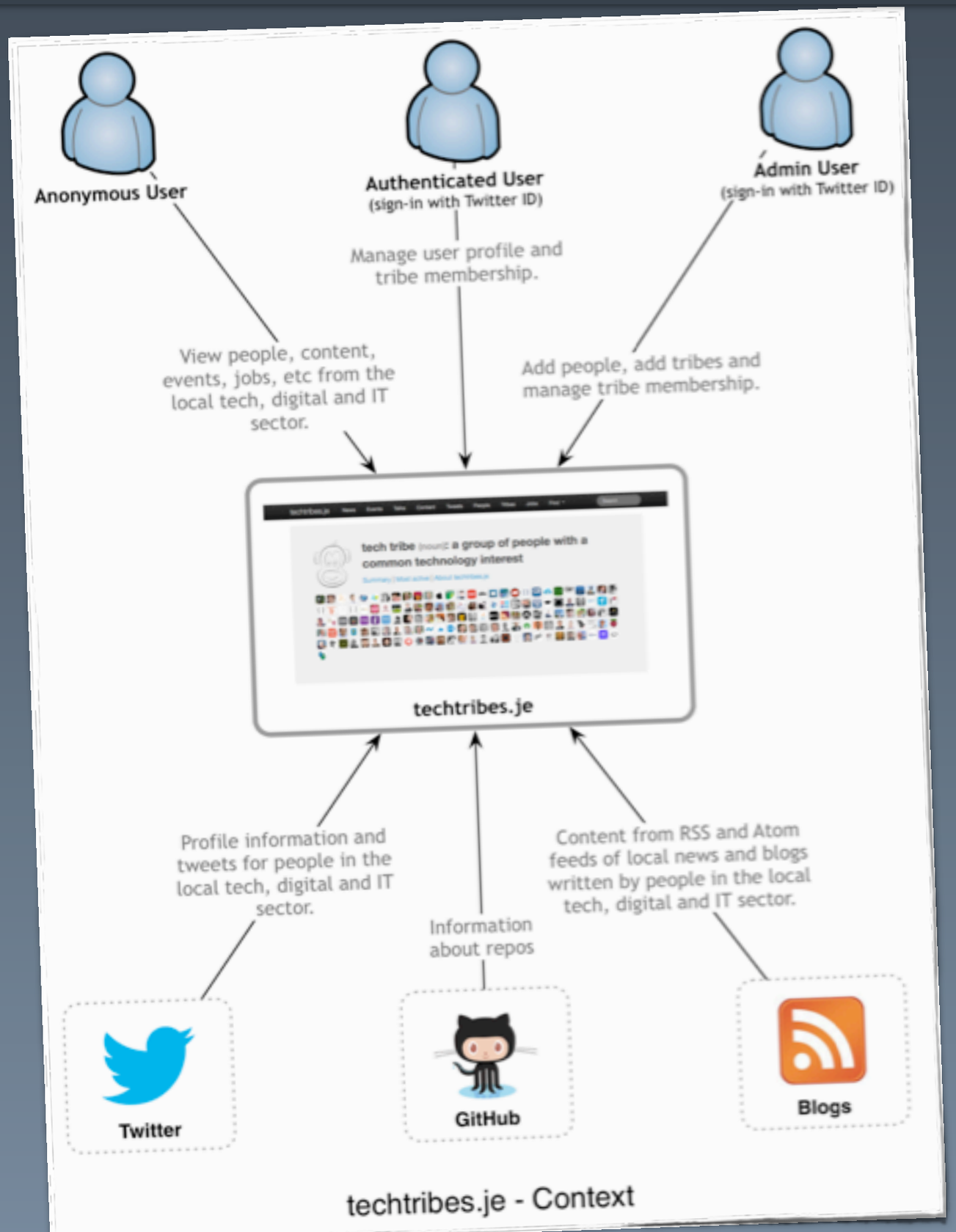






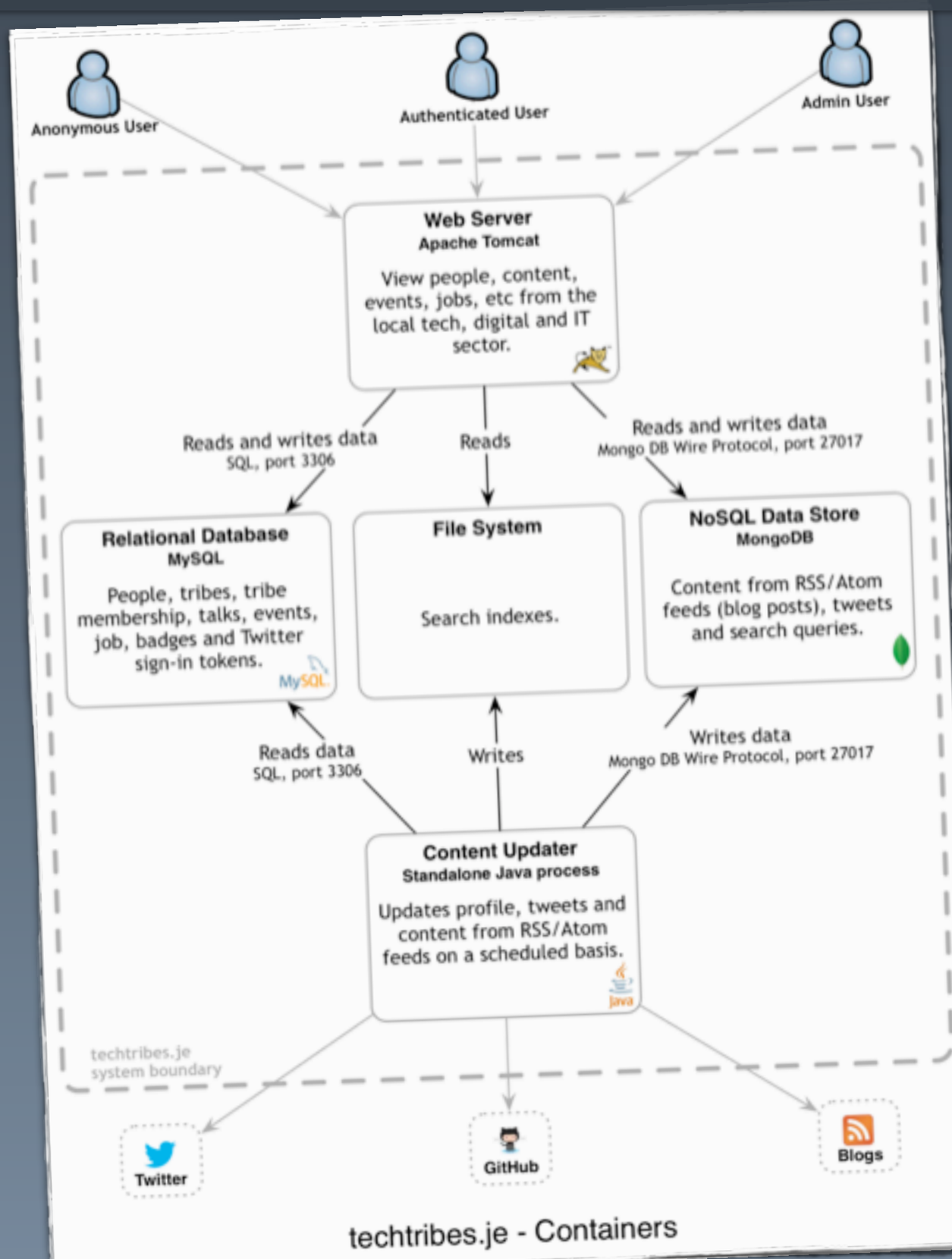
Context

- What are we building?
- Who is using it? (users, actors, roles, personas, etc)
- How does it fit into the existing IT environment?



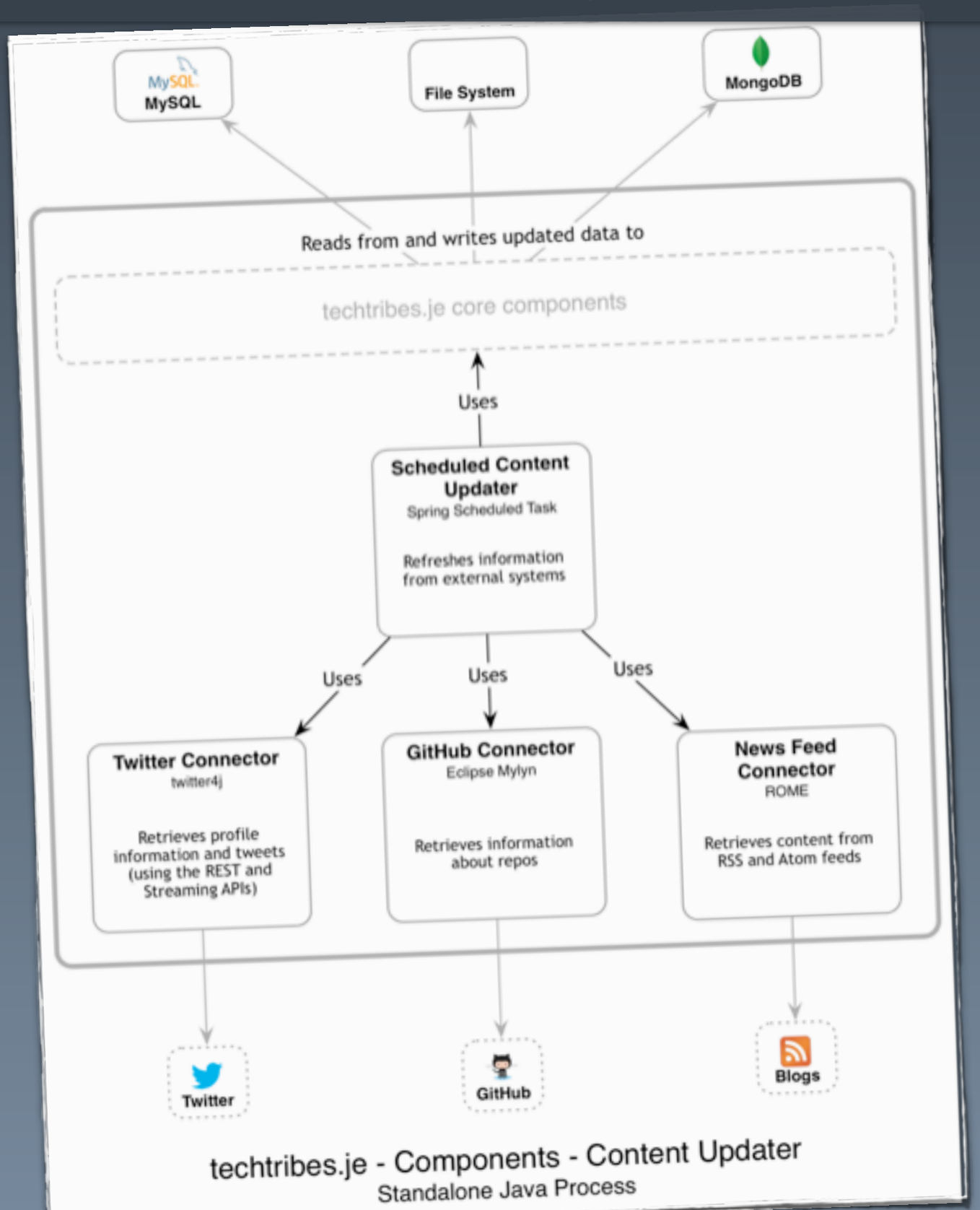
Containers

- What are the high-level technology decisions?
- How do containers communicate with one another?
- As a developer, where do I need to write code?



Components

- What components/services is the system made up of?
- Is it clear how the system works at a high-level?
- Do all components have a home (a container)?



Titles
Short and meaningful, numbered if diagram order is important

Lines
Make line style and arrows explicit, add annotations to lines to provide additional information

Layout
Sticky notes and index cards make a great substitute for drawn boxes, especially early on

Labels
Be wary of using acronyms

Colour
Ensure that colour coding is made explicit

Orientation
Users at the top and database at the bottom? Or perhaps “upside-down”?

Shapes
Don’t assume that people will understand what different shapes are being used for

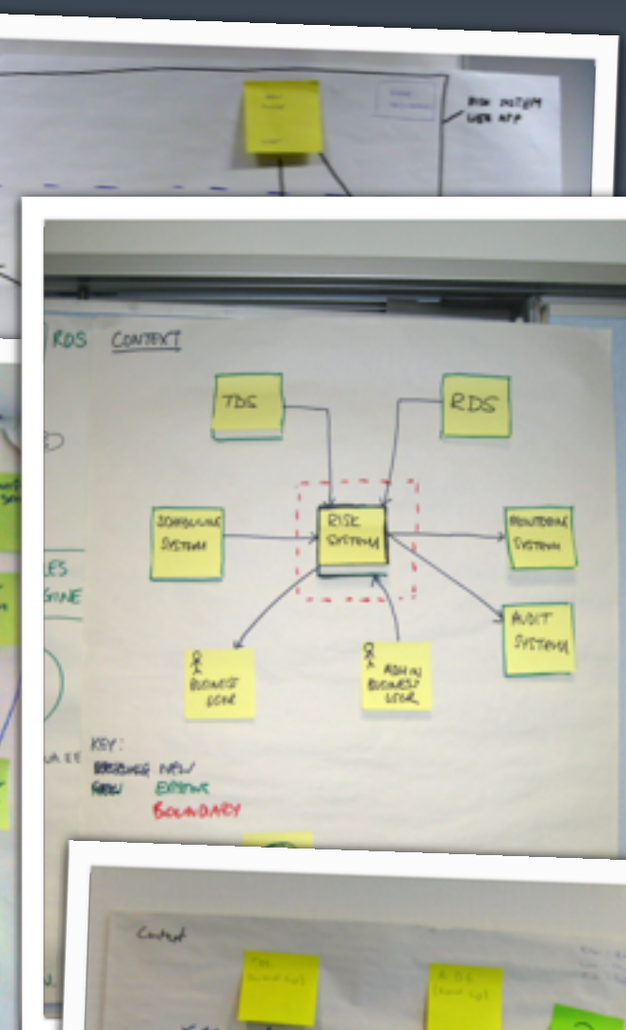
Borders
Use borders to provide emphasis or group related items, but ensure people know why

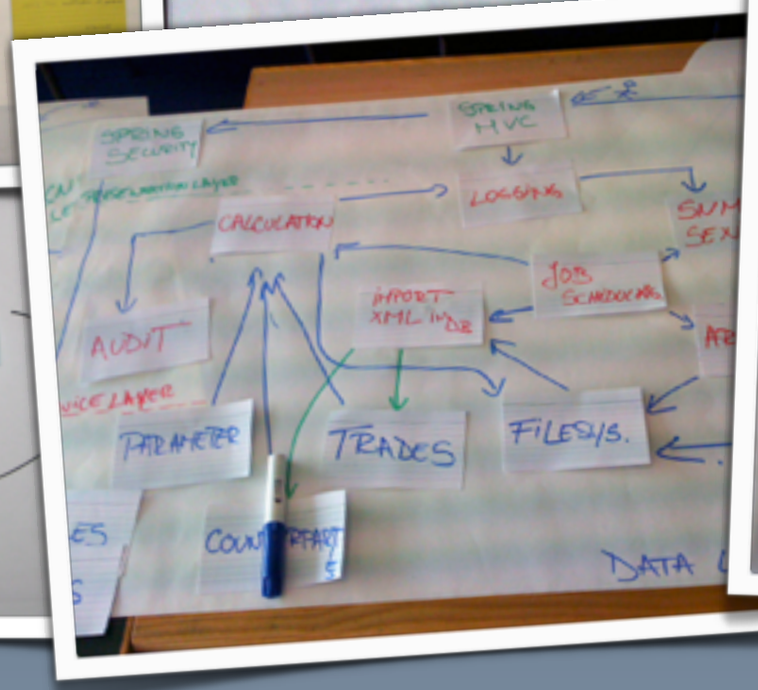
Keys
Explain shapes, lines, colours, borders, acronyms, etc

Responsibilities
Adding responsibilities to boxes can provide a nice “at a glance” view

Some tips for *effective sketches*

Use sticky notes or index cards instead of drawing boxes





Structure

- [ ] I can see the solution from multiple levels of abstraction
- [ ] I understand the big picture (context)
- [ ] I understand the logical containers
- [ ] I understand the major components used to satisfy the important user stories/features
- [ ] I understand the notation, colour coding, etc used on the diagrams
- [ ] I can see the traceability between diagrams

Vision

- [ ] I understand the major technology decisions
- [ ] I understand the implementation strategy (frameworks, libraries, APIs, etc)
- [ ] I can visualise the code structure

Start from the top and work down into the detail

Make some technology decisions!

This isn't about creating a standard

It's about providing you some organisational ideas

Think about the target audience

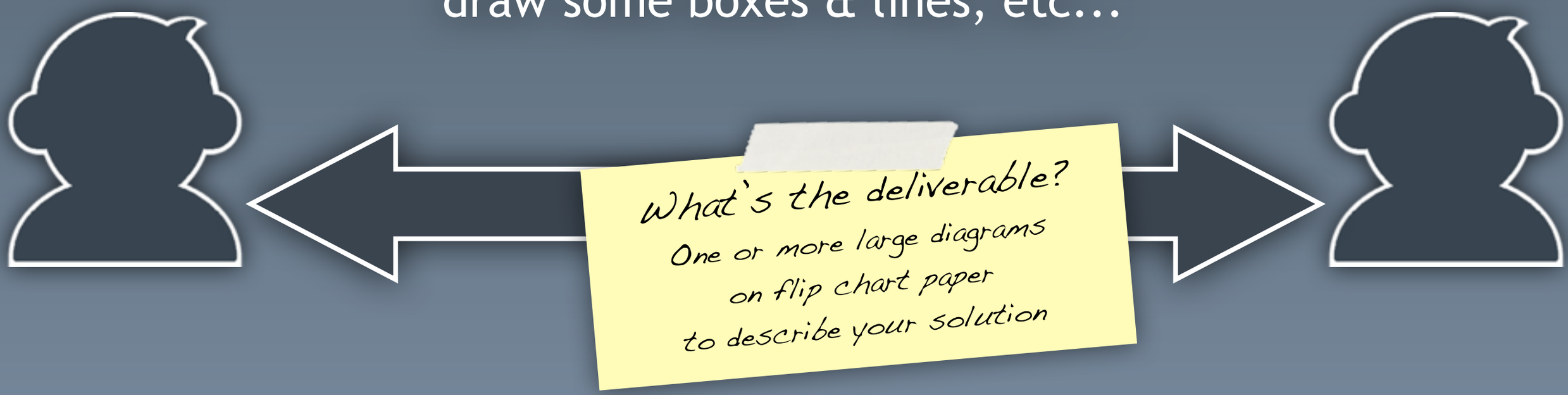
Non-technical Semi-technical Very technical

Do whatever works for

you

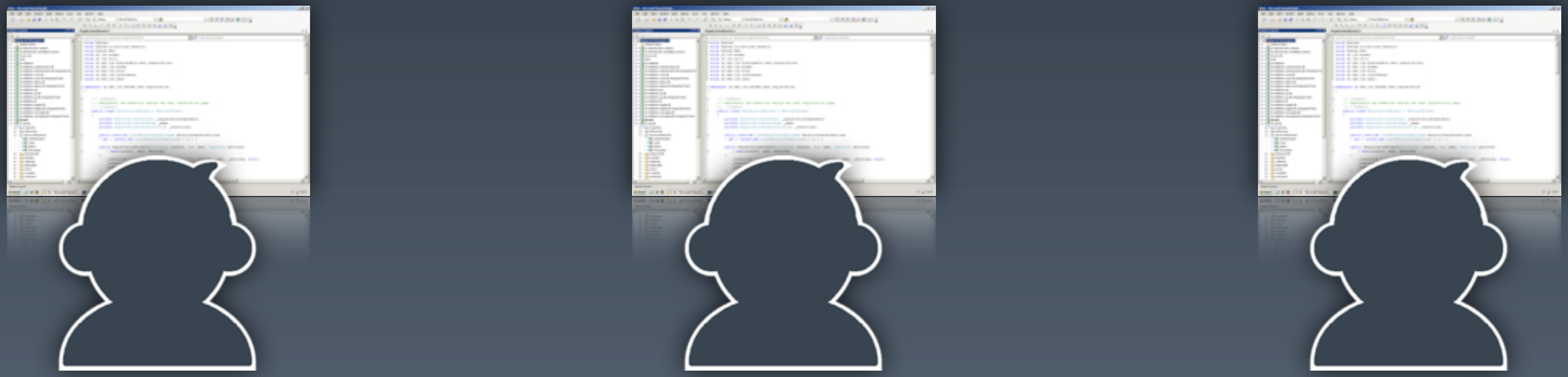
Design a software solution for the financial risk system

Do some design, choose some technologies,
draw some boxes & lines, etc...



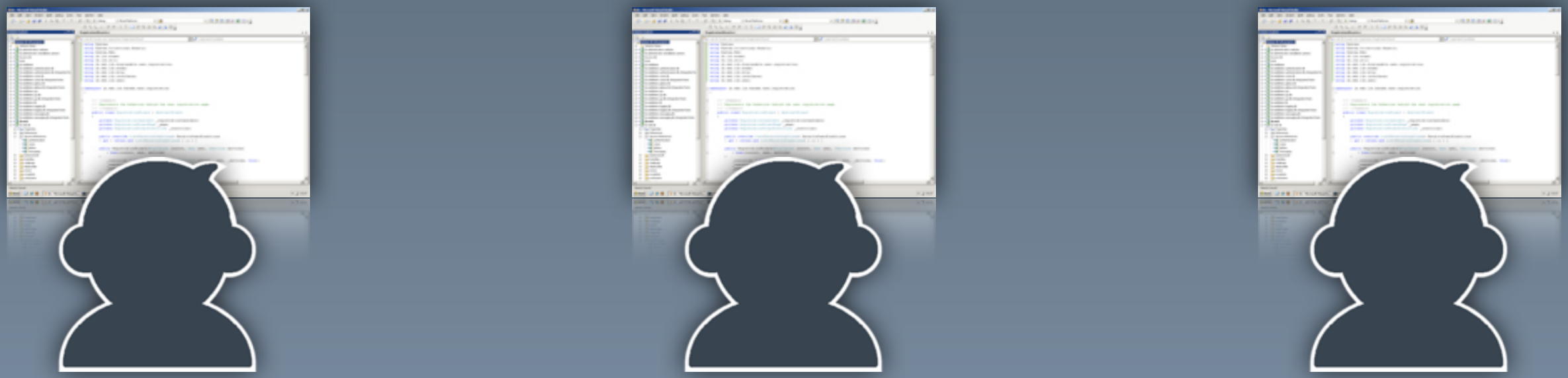
Sketches

in context

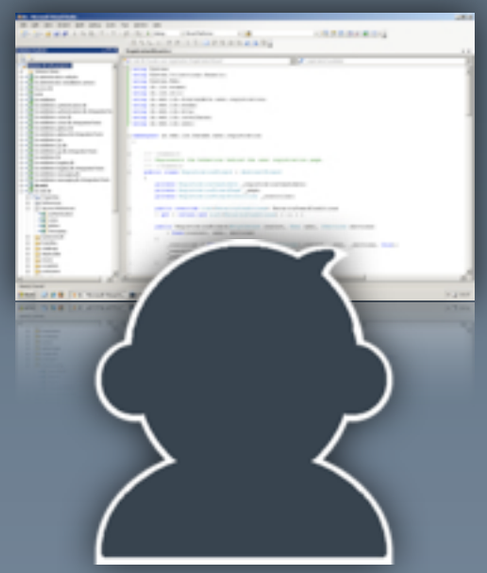


Chaos!

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

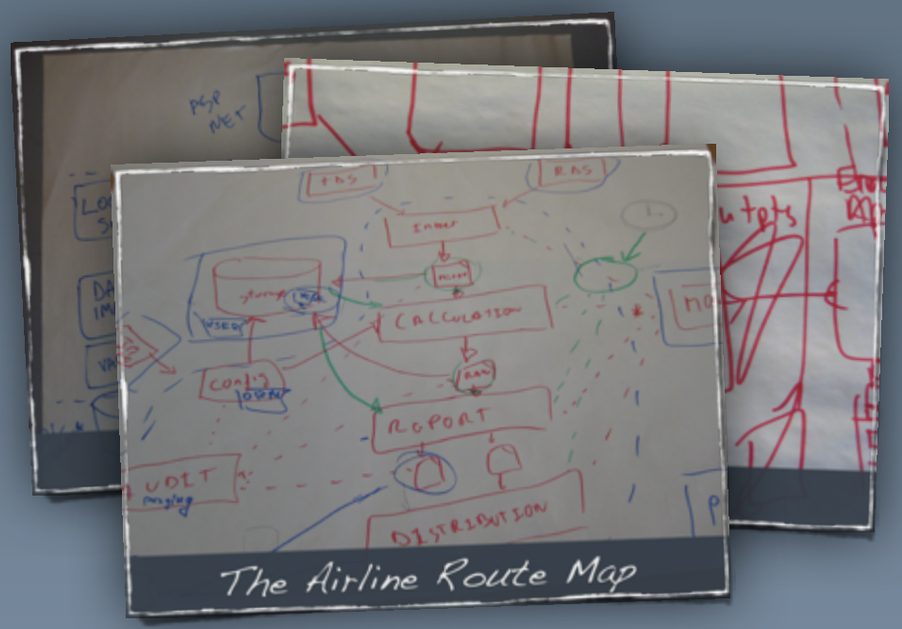


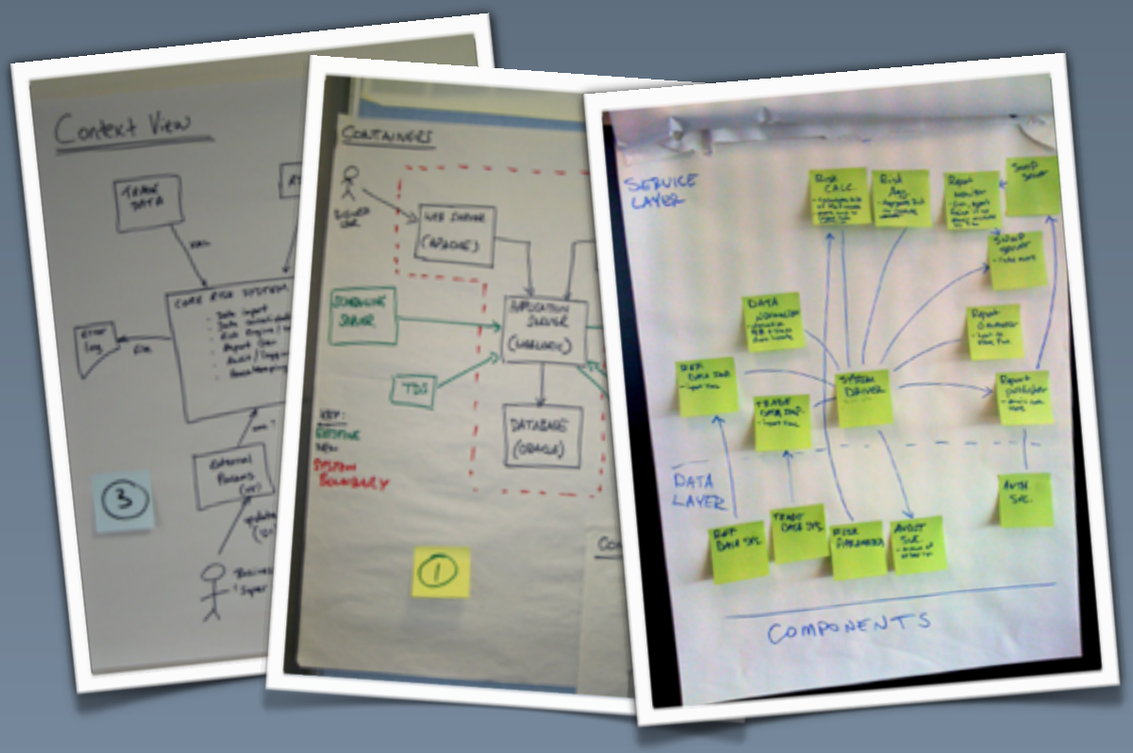
The code doesn't tell the *whole* story, but it does *a* story



Moving fast (agility) requires

good communication





Every software developer should know how to sketch

It allows you to visualise a solution and communicate it quickly

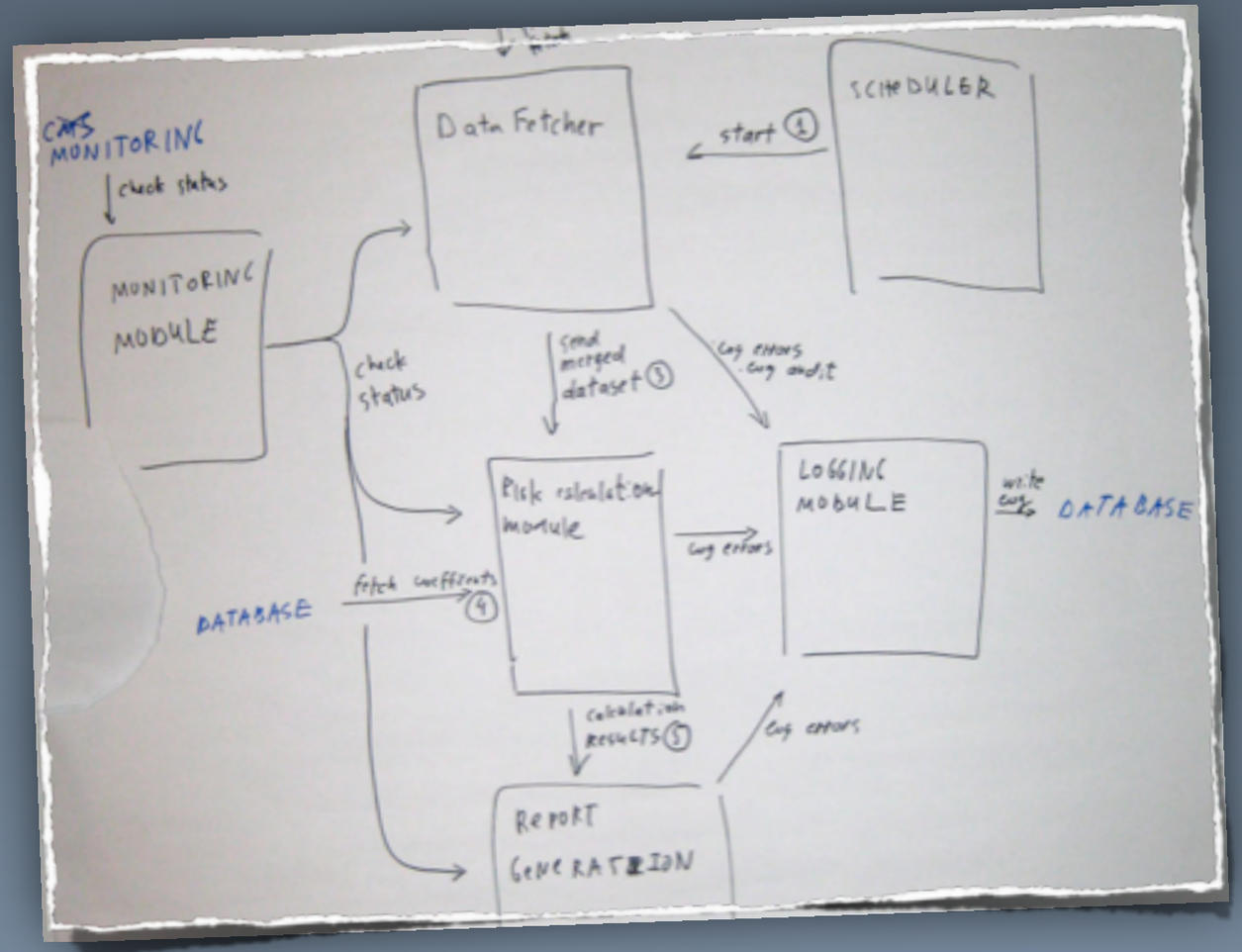
It paves the way for collaborative design and collective code ownership

Sketches are not art or comprehensive models

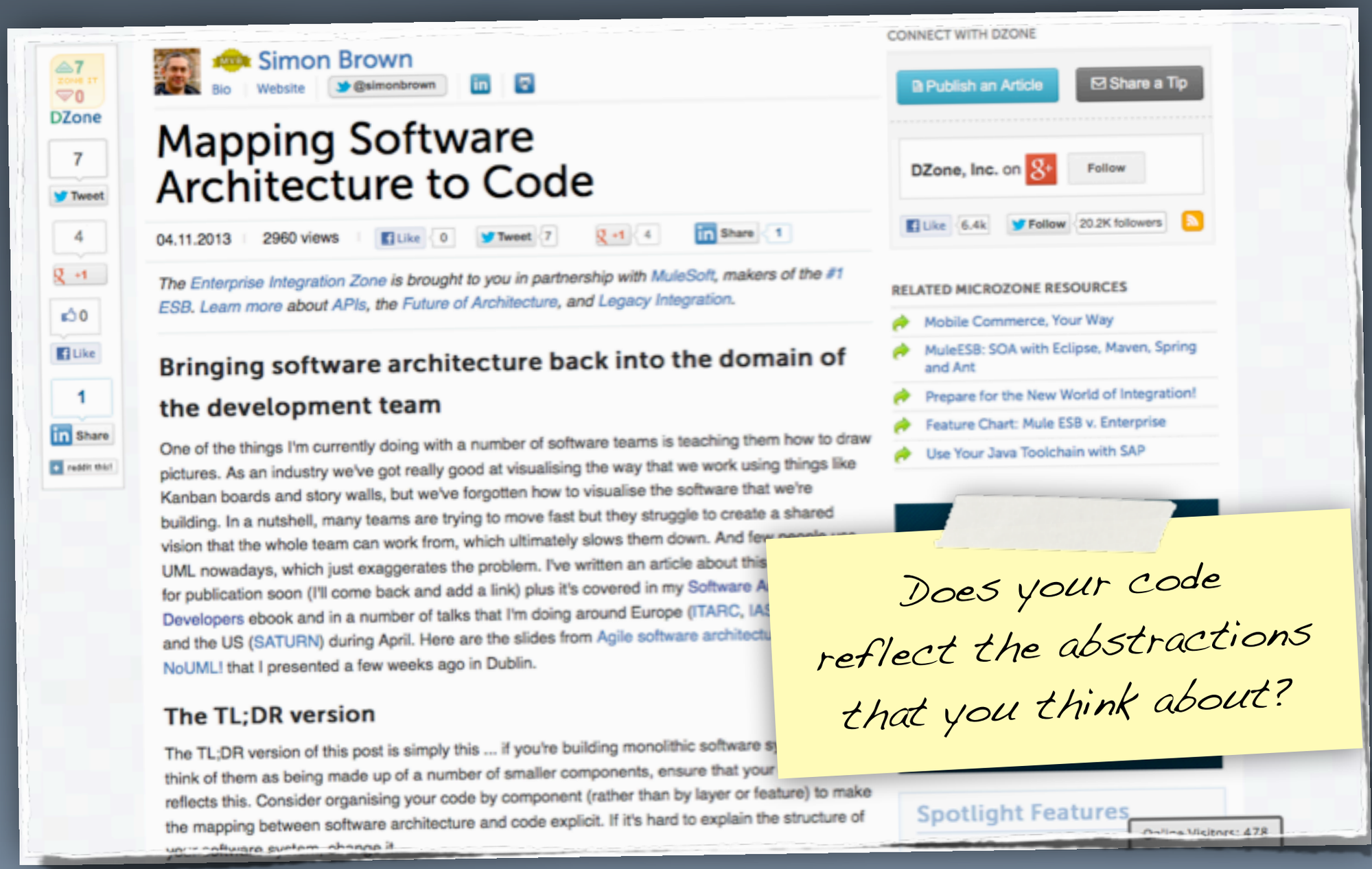
Abstraction

is about reducing detail

rather than creating a different representation

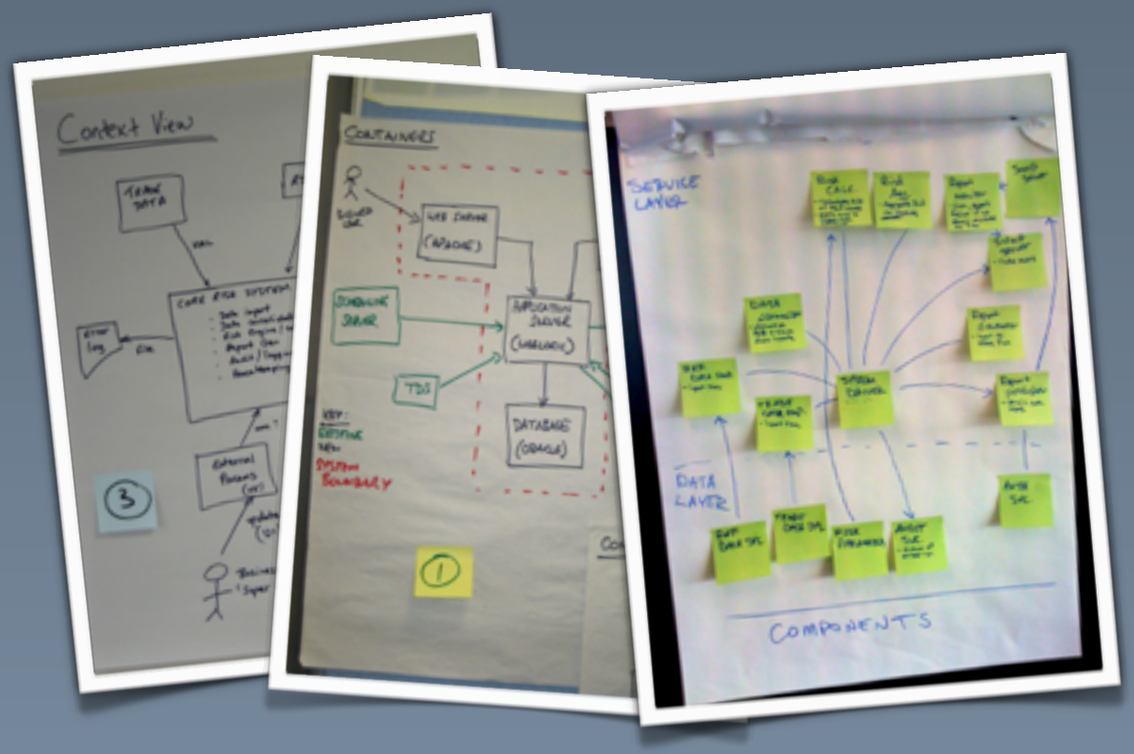


We think in components but write classes



Pictures

are the simplest form of documentation



Leave your *sketches* on the wall...

A point of reference for technical discussions
(something to point at)

A map to help the team navigate a complex codebase

Sketches

can form the basis for

just enough
up front design

Just enough up front design to

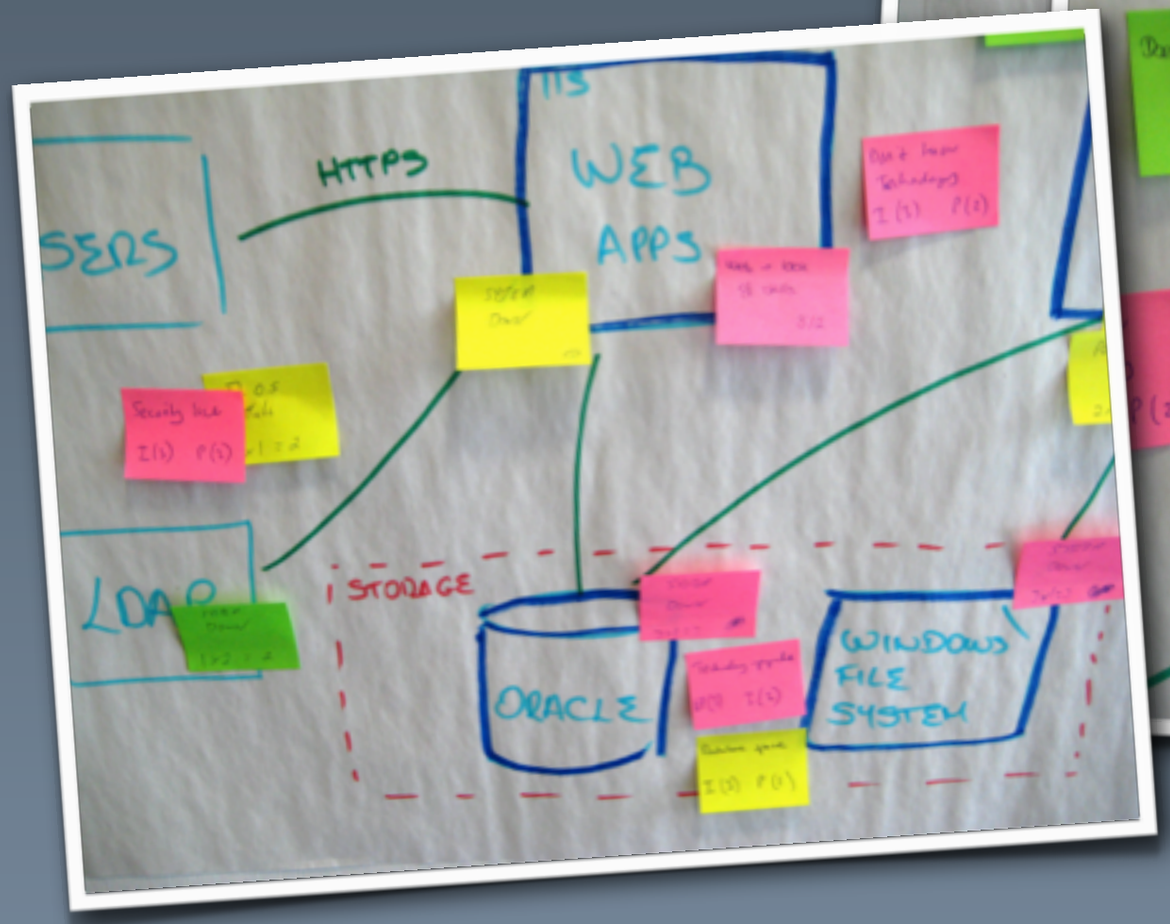
understand the structure

of the software and

create a shared vision

for the team

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

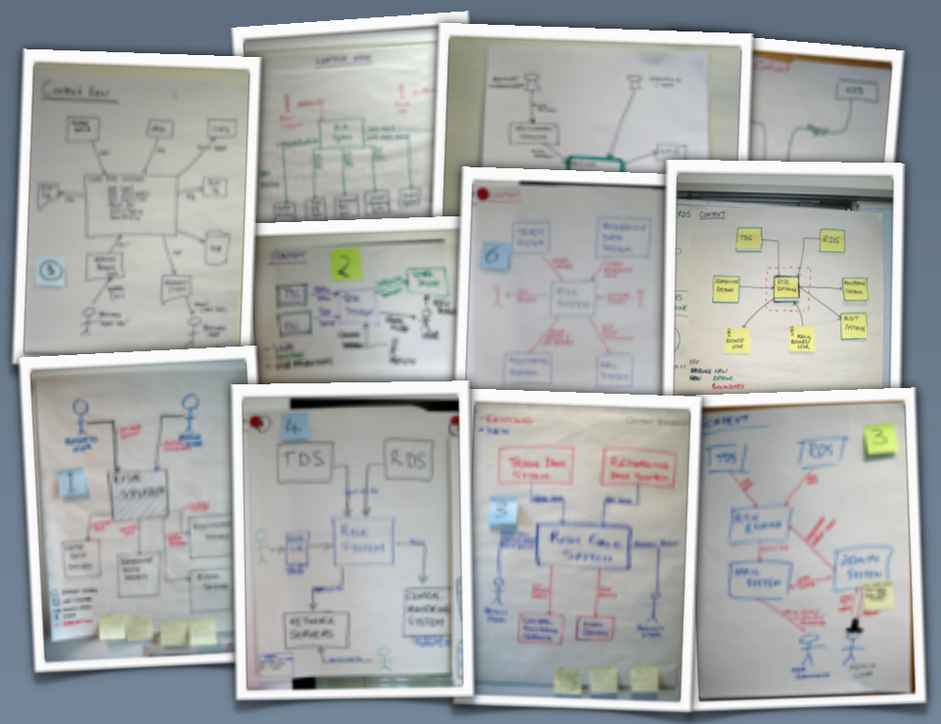
Provide firm foundations and a vision to move forward

The process

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

    public SomeWizard()
    {
    }

    ...

}
```

Thanks

and happy sketching!



simon.brown@codingthearchitecture.com
@simonbrown on Twitter