coding {the} architecture

Agile and architecture

(finally friends after 15 years)

Simon Brown
@simonbrown

coding {the} architecture

Software Architecture for Developers

Volume 1

Technical leadership by **coding**, coaching, collaboration and just enough up front design

Simon Brown

coding {the} architecture

Software Architecture for Developers

Volume 2

Visualise, document and explore your software architecture

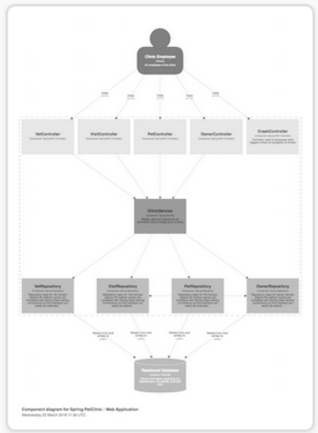
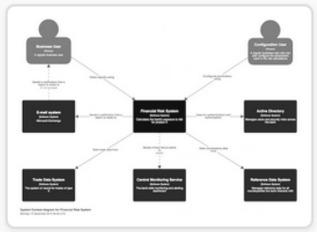
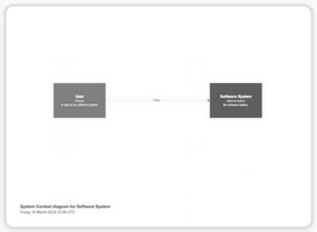
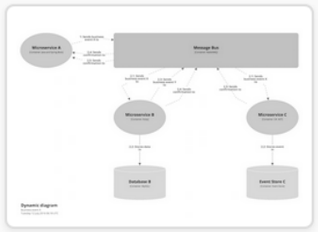
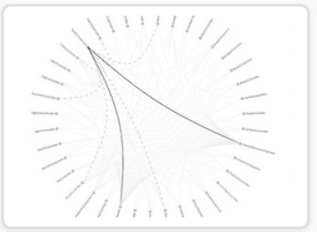
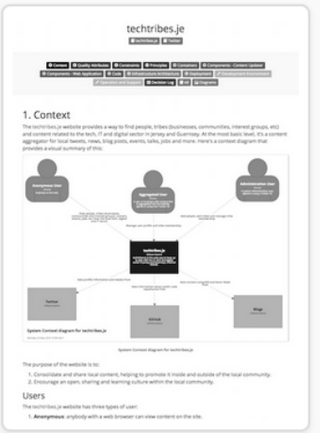
Simon Brown

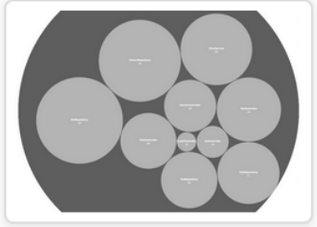
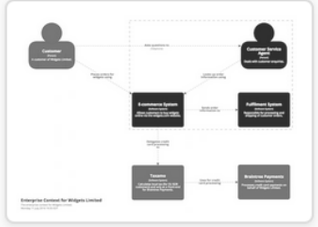
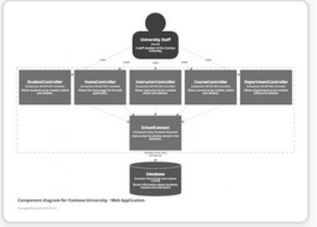
I help software teams understand

software architecture, technical leadership and the balance with agility

Structurizr

Visualise, document and explore your software architecture

Structurizr Express Create software architecture diagrams using text with Structurizr Express - free, no sign up required.

Visualise, document and explore your software architecture

(use promotional code “ASAS 2016” before 30th Sept to get 3 months of the Standard Plan for free)

Structured Programming

(Software Craftsmanship v1.0)

“Waterfall”

time spent early in the software production cycle can reduce costs at later stages

Wikipedia

A “structured approach” with an emphasis on comprehensive documentation

SSADM

(Structured Systems Analysis and Design Method)

a systems approach to the analysis and design of information systems

Wikipedia

Long feedback loops often resulted in the wrong thing being delivered

Iterative and incremental development

RUP

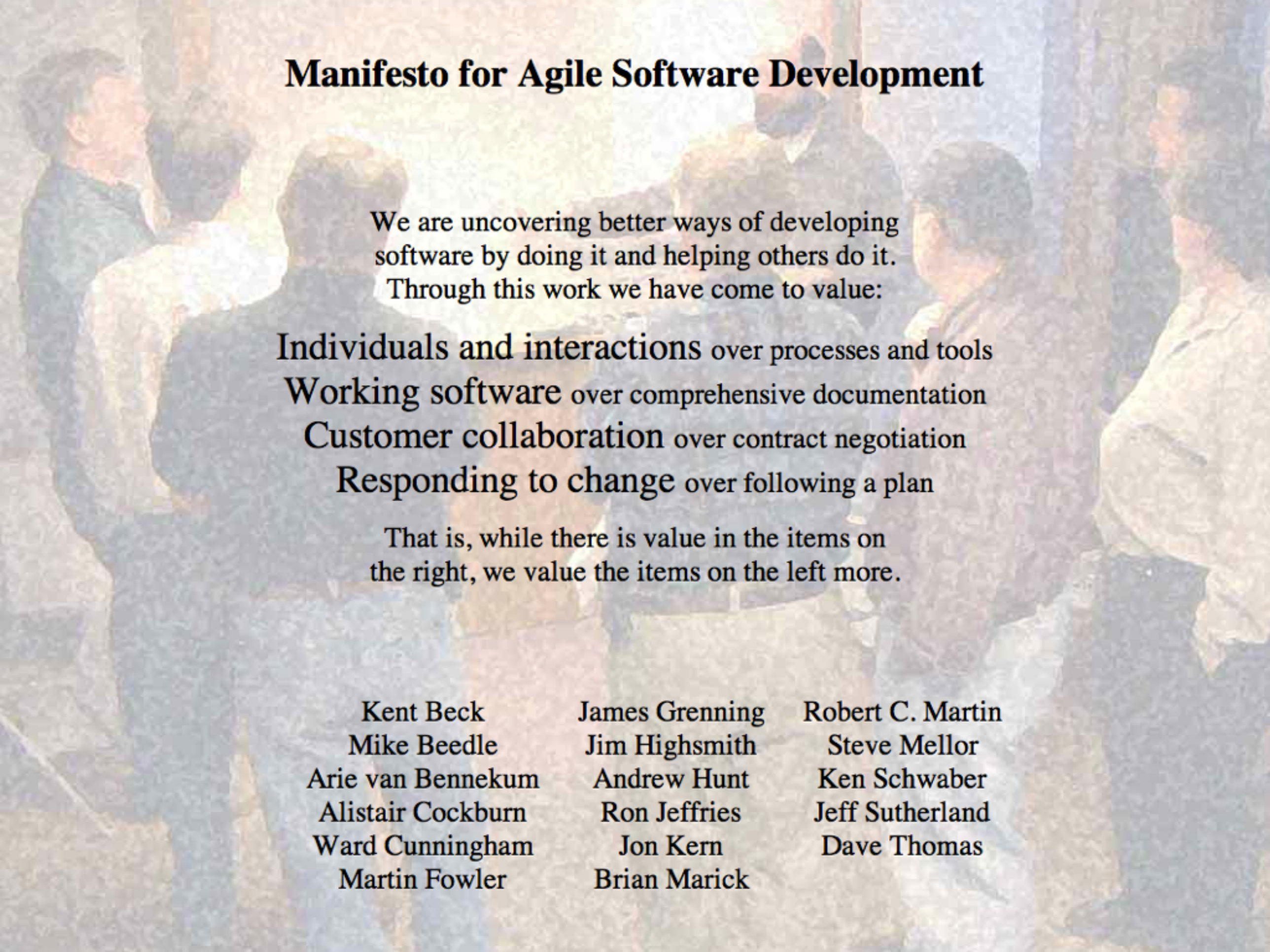
(Rational Unified Process)

Roles and artifacts

Rapid Application Development

("prototype-driven design")

Agile



Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

The 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

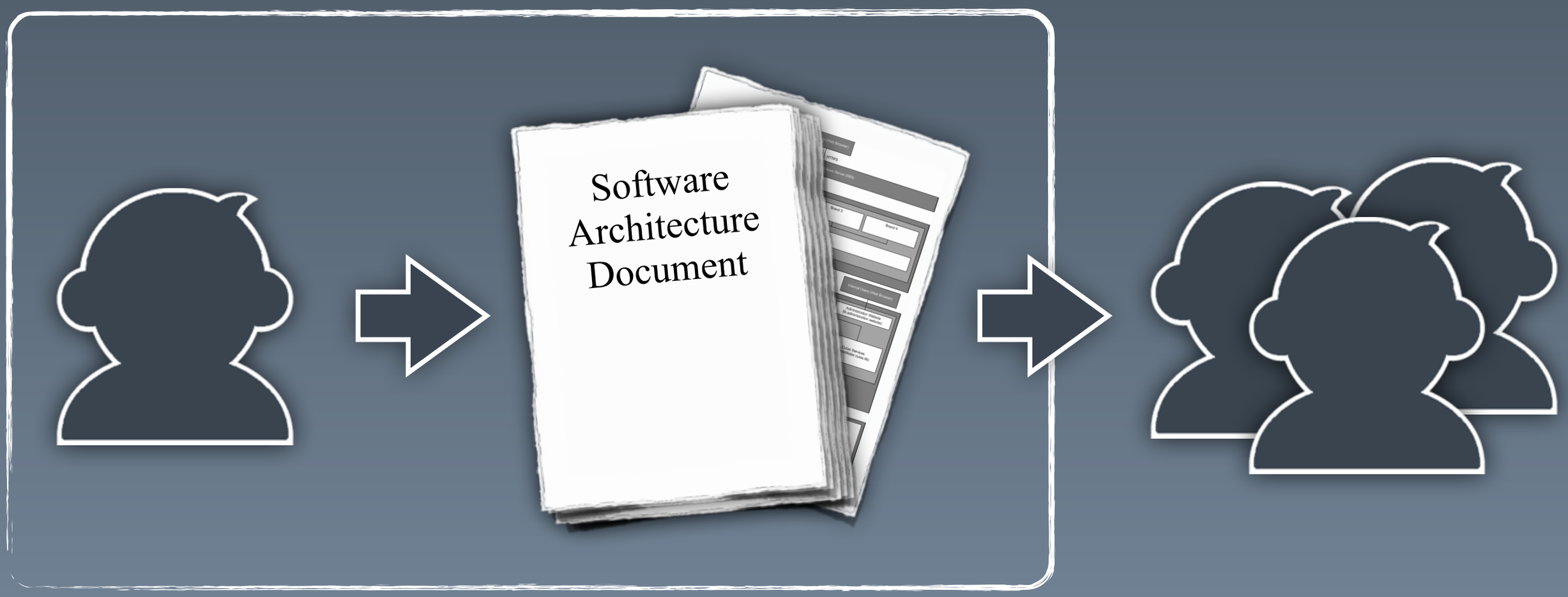
architect

Ivory Tower

PowerPoint Architect

Architecture Astronaut

Software development is not a relay sport

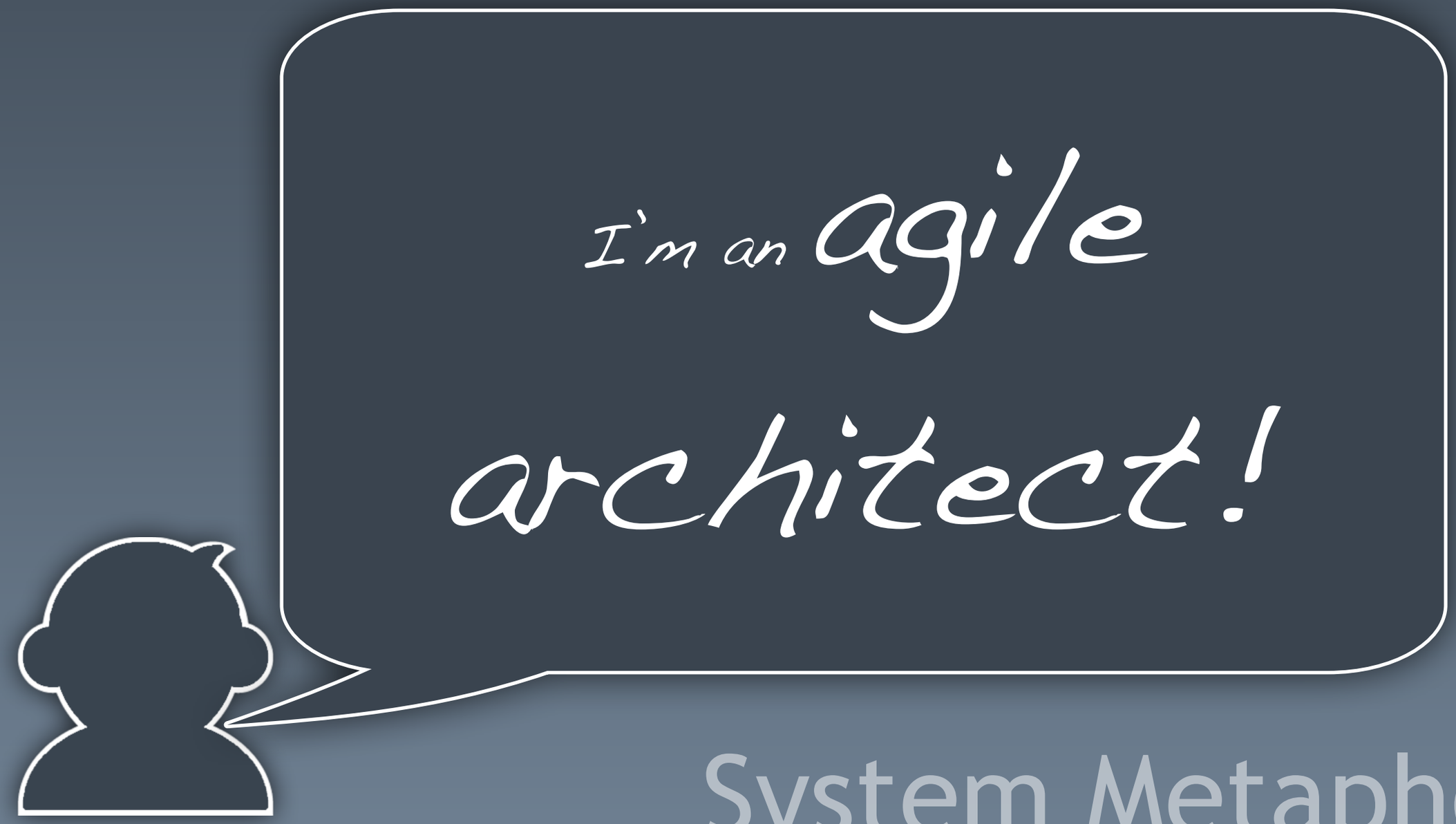


AaaS ... architecture as a service

Evolutionary Architecture
and Emergent Design

YAGNI

Defer until the
last responsible
moment



System Metaphor

Refactoring

Spikes, stripes and tracers

The best architectures, requirements, and designs emerge from self-organizing teams.

Principle 11 of the Manifesto for Agile Software Development

Architects?

We don't need no stinkin' architects!

Small, self-organising teams
of generalising specialists,
often where everybody is
jointly responsible

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

Historically there's been a tendency towards **big design up front**

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

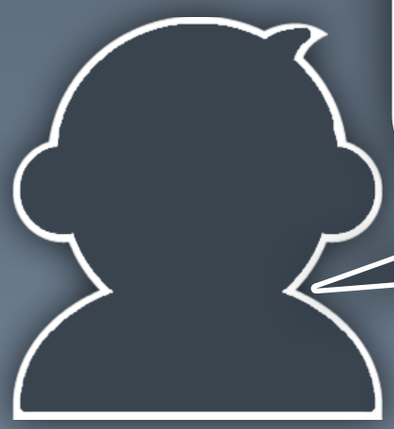
No

design up front

Evolutionary architecture

We don’t need
software architecture;

we do TDD



Agile software team

Last

responsible

moment

Are we allowed to do up front design if we're agile?

“Big design up front
is dumb.

Doing no design up front
is even dumber.”

(YOW! Dave Thomas)

What is

"agile"?

Moving fast, embracing change, releasing often, getting feedback, etc?

Lightweight?

A mindset/culture of continuous improvement?

Robert Martin

1 hr ·

Nice recounting of the event. I would add that it was Martin Fowler and I working together who came up with the initial invitation list; which Alistair made many additions to. The original email (does anyone still have it?) had the subject line: Lightweight Process Summit". IIRC.

I went back to snowbird a couple of years ago and snapped this picture of the original meeting room.

https://www.facebook.com/robertcecilmartin/posts/1503470873001857

Alistair.Cockburn.us | How I saved Agile and the Rest of the World

This is not about me saying anything. It is to remind you

Side note about selecting the name 'aglie': I facilitated the name-finding session, so I have a fairly detailed recall of how that went: "Agile", for example, was not a sure winner for a name at any time, "Adaptive" tied it at the final round. We filled pages with brainstormed names. We selected a handful and said why we did ***not*** *like them, odd though that procedure sounds. The dominant meme in that discussion was "I don't want to have to wear pink tights and a tutu when I say it"! We voted down to five, then voted for a winner but found a tie between agile and adaptive, then selected "agile".*

http://alistair.cockburn.us/How+I+saved+Agile+and+the+rest+of+the+world

My son, who is now in the workforce full-time as a software developer, asked me what "agile software development" is supposed to mean, in a nutshell. I said it meant taking a step, assessing the outcome, learning from that outcome, adjusting our practices, and then taking the next step. He thought that made sense.

Another way to say it is that "agility" basically means an organization has fostered a culture of continual improvement. The alternative is an organization in which people simply learn a defined process and follow it by rote, over and over again. It makes no difference if their process has the word "agile" in its name.

http://www.leadingagile.com/2016/09/reclaiming-agile/

We do Scrum

as described

in the book.

“ x over y ”

Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development.

Continuous attention to technical excellence and good design enhances agility.

Principle 9 of the Manifesto for Agile Software Development

A good architecture enables

agility

Agility

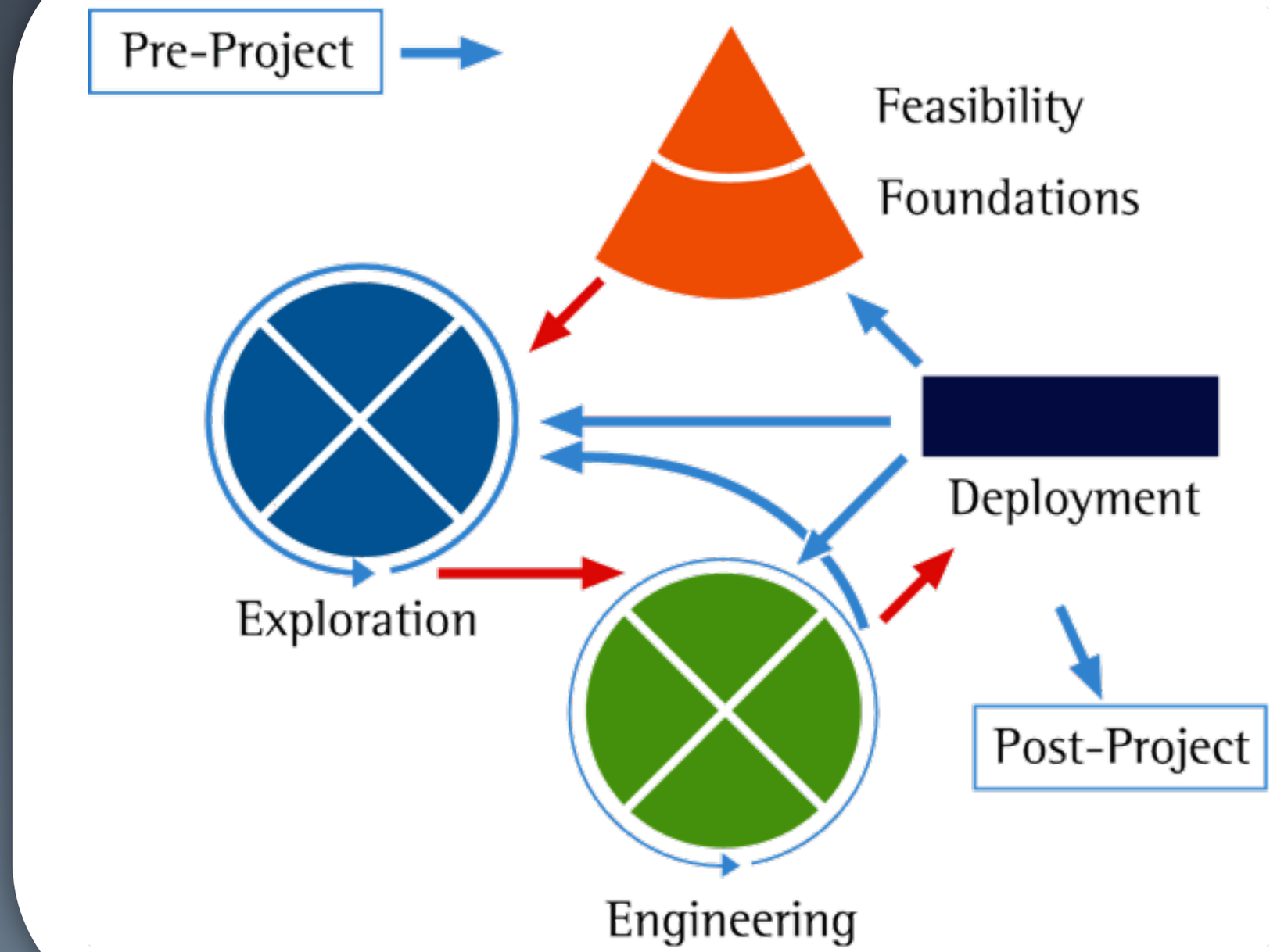
is a

quality attribute

A design revival?

Beyond Scrum

DSDM Atern
Disciplined Agile Delivery (DAD)
Scaled Agile Framework (SAFe)



“The Atern lifecycle process”

(https://www.dsdm.org/content/lifecycle)

The Foundations phase is aimed at establishing

firm and enduring foundations

for the project.

It’s **not** about creating a perfect end-state or complete architecture

You need a starting point

Detailed blueprints

vs

setting a direction

Continuous technical leadership

Add value, eliminate waste

Structure

Understand the significant structural elements and how they fit together, based upon the architectural drivers.

Design and decomposition down to containers and components.

Vision

Create and communicate a vision for the team to work with.

Context, container and component diagrams.

Risks

Identify and mitigate the highest priority risks.

Risk-storming and concrete experiments.

Just enough up front design
to create firm foundations
for the software product and its delivery

The role

Just enough
up front design
to create firm
foundations
for the
software product
and its delivery

(structure, vision, risks)

The process

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

    public SomeWizard()
    {
    }

    ...

}
```

we used to do stuff like this, it worked but we stopped doing it when we became agile

We all came away impressed with the practical, agile approach to software architecture. I'm going to push that we adopt most of these practices as a baseline…

Is agile and software craftsmanship enough?

Software craftsmanship is an approach to software development that emphasizes the coding skills of the software developers themselves.

Wikipedia

Clean code

Are we optimising the wrong thing?

Agility needs a starting point, yet many software teams don't seem to have

a sufficient toolbox of practices

How do you design software?

“we use a whiteboard”

we draw boxes that represent components

“we use our experience”

Decomposition (computer science)

From Wikipedia, the free encyclopedia

Decomposition in computer science, also known as **factoring**, is breaking a complex problem or system into parts that are easier to conceive, understand, program, and maintain.

Overview [edit]

There are different types of decomposition defined in computer sciences:

- In structured programming, *algorithmic decomposition* breaks a process down into well-defined steps.
- Structured analysis breaks down a software system from the system context level to system functions and data entities as described by Tom DeMarco.[1]
- *Object-oriented decomposition*, on the other hand, breaks a large system down into progressively smaller classes or objects that are responsible for some part of the problem domain.
- According to Booch, algorithmic decomposition is a necessary part of object-oriented analysis and design, but object-oriented systems start with and emphasize decomposition into classes.[2]

Decomposition paradigm [edit]

A decomposition paradigm in computer programming is a strategy for organizing a program as a number of parts, and it usually implies a specific way to organize a program text. Usually the aim of using a decomposition paradigm is to optimize some metric related to program complexity, for example the modularity of the program or its maintainability.

Most decomposition paradigms suggest breaking down a program into parts so as to minimize the static dependencies among those parts, and to maximize the cohesiveness of each part. Some popular decomposition paradigms are the procedural, modules, abstract data type and object oriented ones.

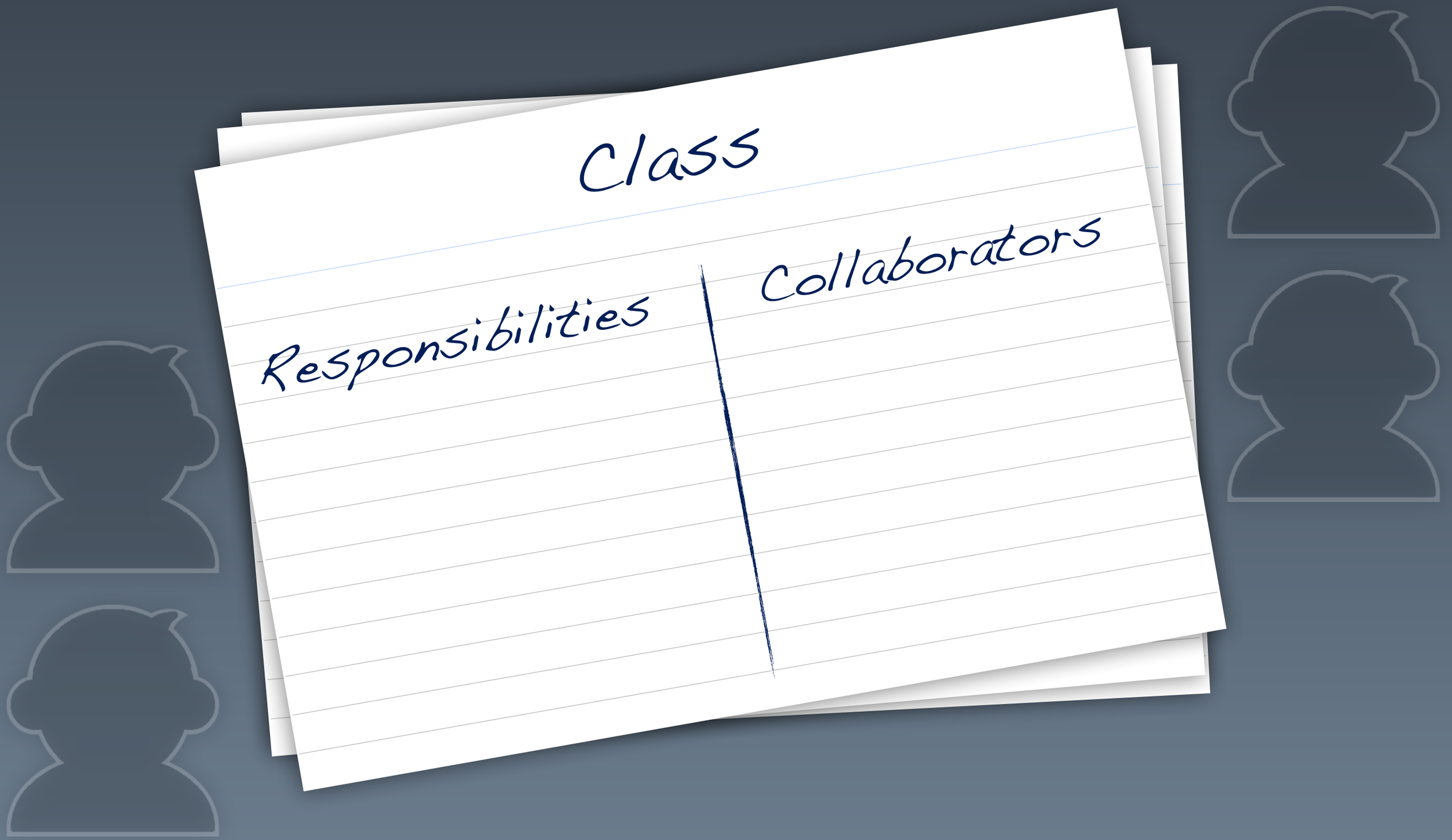
Programming Techniques
R. Morris Editor

On the Criteria To Be Used in Decomposing Systems into Modules

D.L. Parnas
Carnegie-Mellon University

Expected Benefits of Modular Programming

The benefits expected of modular programming are: (1) managerial—development time should be shortened because separate groups would work on each module with little need for communication: (2) product flexibility—it should be possible to make drastic changes to one module without a need to change others; (3) comprehensibility—it should be possible to study the system one module at a time. The whole system can therefore be better designed because it is better understood.



Class-Responsibility-Collaboration

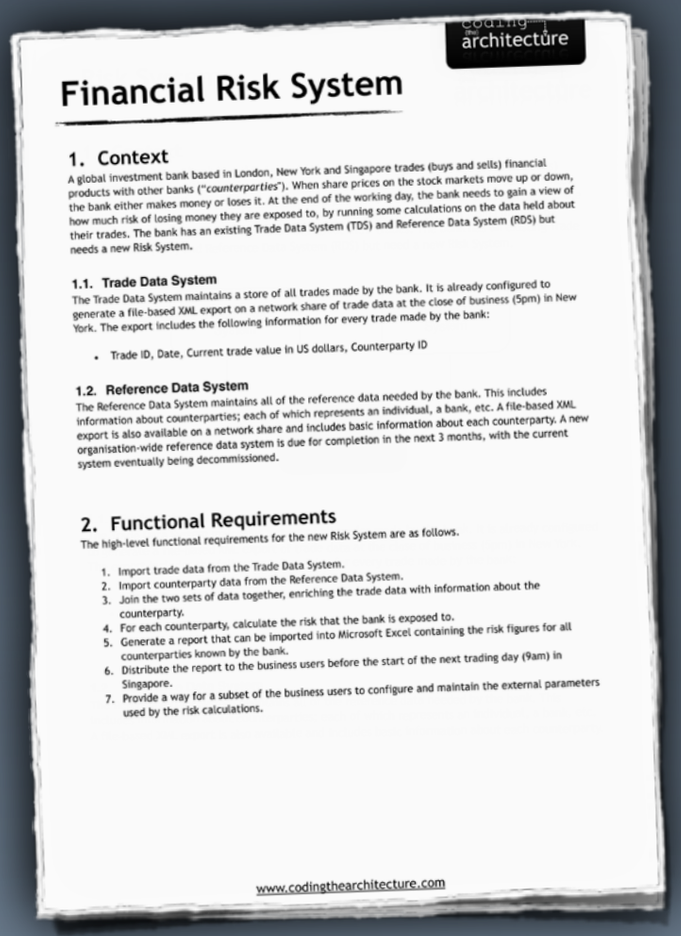
Components

~~Classes,~~

Responsibilities, Collaborations

During my workshops, often groups draw a visual representation of the requirements and say,

“we’re done, we’re agile”



Financial Risk System

coding the architecture

1. Context

A global investment bank based in London, New York and Singapore trades (buys and sells) financial products with other banks ("*counterparties*"). When share prices on the stock markets move up or down, the bank either makes money or loses it. At the end of the working day, the bank needs to gain a view of how much risk of losing money they are exposed to, by running some calculations on the data held about their trades. The bank has an existing Trade Data System (TDS) and Reference Data System (RDS) but needs a new Risk System.

1.1. Trade Data System

The Trade Data System maintains a store of all trades made by the bank. It is already configured to generate a file-based XML export on a network share of trade data at the close of business (5pm) in New York. The export includes the following information for every trade made by the bank:

- Trade ID, Date, Current trade value in US dollars, Counterparty ID

1.2. Reference Data System

The Reference Data System maintains all of the reference data needed by the bank. This includes information about counterparties; each of which represents an individual, a bank, etc. A file-based XML export is also available on a network share and includes basic information about each counterparty. A new organisation-wide reference data system is due for completion in the next 3 months, with the current system eventually being decommissioned.

2. Functional Requirements

The high-level functional requirements for the new Risk System are as follows.

1. Import trade data from the Trade Data System.
2. Import counterparty data from the Reference Data System.
3. Join the two sets of data together, enriching the trade data with information about the counterparty.
4. For each counterparty, calculate the risk that the bank is exposed to.
5. Generate a report that can be imported into Microsoft Excel containing the risk figures for all counterparties known by the bank.
6. Distribute the report to the business users before the start of the next trading day (9am) in Singapore.
7. Provide a way for a subset of the business users to configure and maintain the external parameters used by the risk calculations.

www.codingthearchitecture.com

“the solution is simple and can be built with any technology”

“we don't want to force a solution on developers”

"it's an implementation detail”

“we follow the ‘last responsible moment’ principle”

“software architecture should be technology independent”

Decoupling
technology choices
is not the same as
deferring them

Technology is

not

an “implementation detail”

UNIX BOX

RS

TRANSPORT + LOGIC

JBOSS INSTANCE

ERROR

JBOSS INSTANCE (WEB CONTAINER ONLY)

WINDOWS BOX

MS SQL SERVER

MS REPORTING SERVICE

ASP NET

LOGGING SERVICE

PARAMETER MANAGER

RISK CALCULATION

REPORT GENERATOR

DATA IMPORT

AUDITING

VALIDATION

SQL SERVER

TDS

RDS

~~MERGE~~ RISK DATA

PARAMS

SECURITY

AUDIT

FUNCTIONAL VIEW

File Retriever

Scheduler

Auditing

Reference Archiver

Risk Assesment Processor

Risk Parameter Configuration

Trade Archiver

Report Generator

Report Distributor

APP SERVER

Audit

Event store

PUBLISHER

ESB

CALC CONFIG UI

WEB SERVER

CALCULATION ENGINE

WHICH DATA?

APP SERVER

4

CENTRAL MONITORING SERVICE

WEB SERVER

REPORTING SERVICE

TDS

RAS

Import

storage

KPI

USER

mixed

CALCULATION

config

ORM

raw

REPORT

AUDIT

purging

DISTRIBUTION

TDS

RDS

trade report

ref repos

R9

TRANSPORT

ARCHIVE

action

data

error

error

AUDIT

MS

BUS. LOGIC

action

error

ERROR

MONITOR

REPORT GEN

action

error

Scheduler

config

ADMIN USER

BUSINESS USER

XML

XML

2 SERVICE/BATCH

IMPORT
JOIN
CALC
EXPORT/STORE
REPORT

CENTRAL
MONITORING
SERVICE

SNMP

3 WEBAPP

- GET REPORT
- EDIT RISK RULES
- AUTH
- AUDIT (RULES CHANGE)

RISK RULES
USERS
ARCHIVE

SCHEDULED TASK
VIA SERVER

SCHEDULER
CONSOLE - C#

APPLICATION
CONSOLE - C#

DATA LOADER
C#

CALCULATION MANAGER
C#

NOTIFICATION MANAGER
C#

EXTERNAL SYSTEMS

SNMP

IRDS

ITDS

XML

check if data

DATA MERGE
C#

XML READER
C#

RISK CALCULATION
C#

NOTIFICATION SERVICE
C#

parameters

results of risk

LOGGER
Log

SNMP TRAP GENERATOR

DATA ACCESS

SQL SERVER

ARCHIVE

APP

PHYSICAL SECURITY.

DATA POLL SERVICE

TRADES

AUTO RE-START.

REF DATA.

ARCHIVE

DATA READER TRADES

DATA VALIDATOR

reject

DATA READER REF DATA

DATA VALIDATOR

reject

MONITOR SERVICE SMTP TRAP

FAIL EVENTS

DATA STORE

DATA TRANSFORMATION INTEGRITY VALIDATION

rejects

tolerance.

FAIL EVENT

COMPLETE EVENT

STOP

STOPPED EVENT

COMPLETION EVENT

CALCULATION ENGINE

A + B = C

COUNT

PARMS TOLER.

UI

SECURITY

reject.

FAIL EVENT

COMPLETE EVENT

AUDIT

OUTPUT STORE

FAILURE REPORT

REPORT WRITER

rejects

CSV

DATA STORE

AUDIT

COMPLETE

Report Monitor

(other machine)

Data Normalizer

Risk ~~Calculator~~ Aggregator

CP Risk Calc

Report Generator

Auditing Service

Report Publisher

Report Repository

Params

Calcs

Trades

ref_current

Ref_hist

Discounts

Error App Log

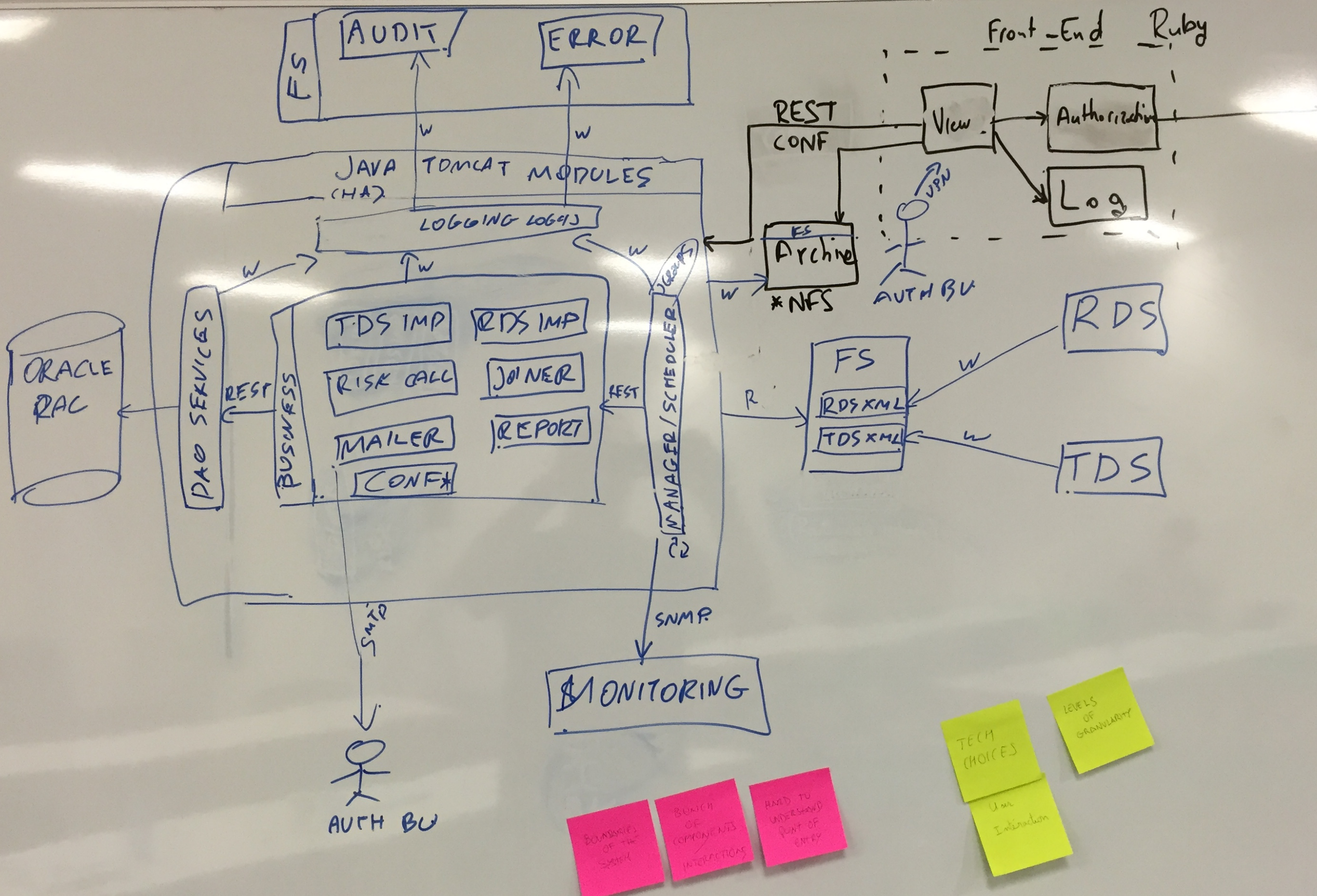
EH?

Batch

Params_Risk_Hist

Detail
B_id:
Outp id
Trade id

App Error Log
Date
- Risk Sel
- calcs
- Par


FS
AUDIT
ERROR
W
W
JAVA TOMCAT MODULES
(HA)
LOGGING LOG4J
W
W
W
PGRAFT
Front_End_Ruby
REST
CONF
View
Authorization
Log
UPN
ES
Archive
*NFS
AUTH BU
W
RDS
FS
RDS XML
TDS XML
W
W
TDS
R
ORACLE
RAC
SERVICES
DAO
REST
BUSINESS
TDS IMP
RDS IMP
RISK CALC
JOINER
MAILER
REPORT
CONF*
REST
MANAGER/SCHEDULER
SMTP
SNMP
MONITORING
AUTH BU
BOUNDARIES OF THE SYSTEM
BUNCH OF COMPONENTS / INTERACTIONS
HARD TO UNDERSTAND POINT OF ENTRY
TECH CHOICES
LEVELS OF GRANULARITY
User Interaction

LDAP

INPUT FILES

RETRIES

UI

F. RISK CALCULATOR

SNMP

LOGGING ALERTS

CONFIG

RISK CALCULATION + INPUTS

CONFIG CHANGE

SIMPLE!

(RELATIVELY) CONSISTENT SYMBOLS

MESSAGE BUS

EMAIL REPORT SYSTEM

ARCHIVING SERVICE

WHAT IS THE ARCHIVING SERVICE ARCHIVING TO?

Good level of detail

Logical System

DB SERVER

RDBMS

JOB RESULTS

SYS PARAMS

UPDATED SYS PARAMS

TDS SYSTEM

EXPORT JOB

FILE EXPORT

RISK APP SERVER X 2

Applications Container

Batch / ETL module

Web module

Health watchdog app/script

TDS IRDS DATA

AUTHENTICATION AUTHORIZATION

HTTP

USERS

Lines Convey Flow

1) Line Complexity

~~4) Colours~~

Colours

RDS SYSTEM

EXPORT JOB

FATAL SNMP

NETWORK STORAGE

NETWORK SHARE

FILE EXPORT

EXPORT (CSV)

MONITORING SYSTEM

SNMP MIB / TRAPPER

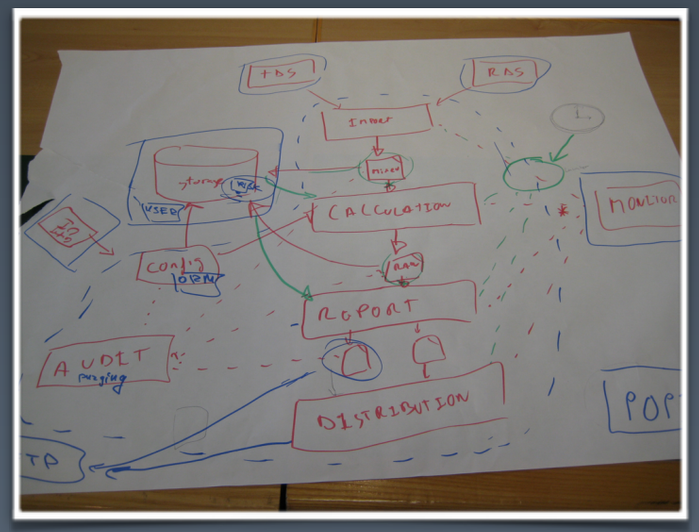
SYSLOG / AUDIT

CENTRAL LOGGING

DIRECTORY SERVER

LDAP / AD

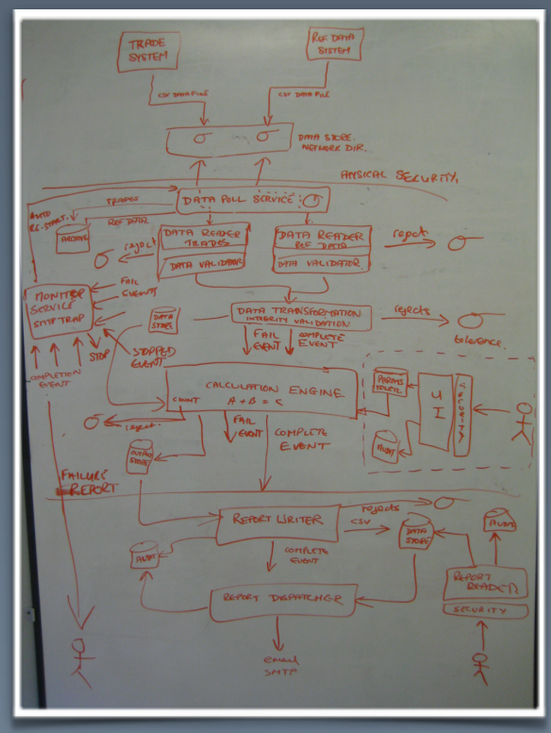
The diagram isn’t self-evident, but we’ll explain it



Team 1

Team 1

Team 2



Team 2

NO ANNOTATION ON FLOWS

SHOULD USE MORE COLORS

POST ITS CAN FALL OFF

what does colour mean?

—

MIXES DIFFERENT LEVELS OF DETAIL

NOT SURE OF TRANSITION BETWEEN DIFFERENT DIAGRAMS -

—

CONFLICTING LEVELS OF DETAIL IN PRESENTATION

~~Meaning of Notation~~

What about the different arrows?

objects vs actions

WHY ARE SOME LINES PINK ?

WHAT DO THE SHAPE MEAN ?

UML is good. But Not Everyone knows it

WHAT DO LINES RE-PRESENT? (DATA? CONTROL DEP.?)

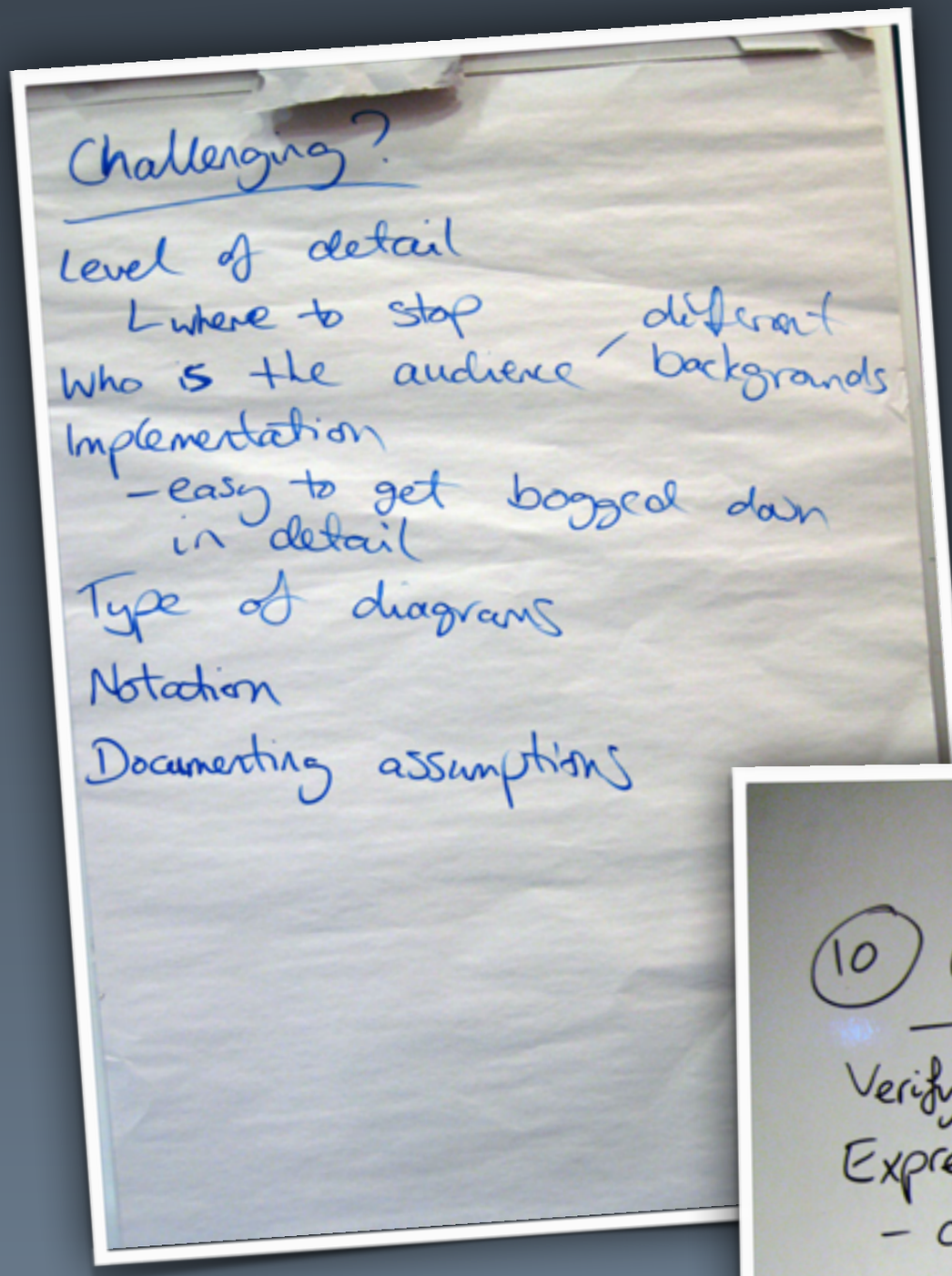
what shapes mean

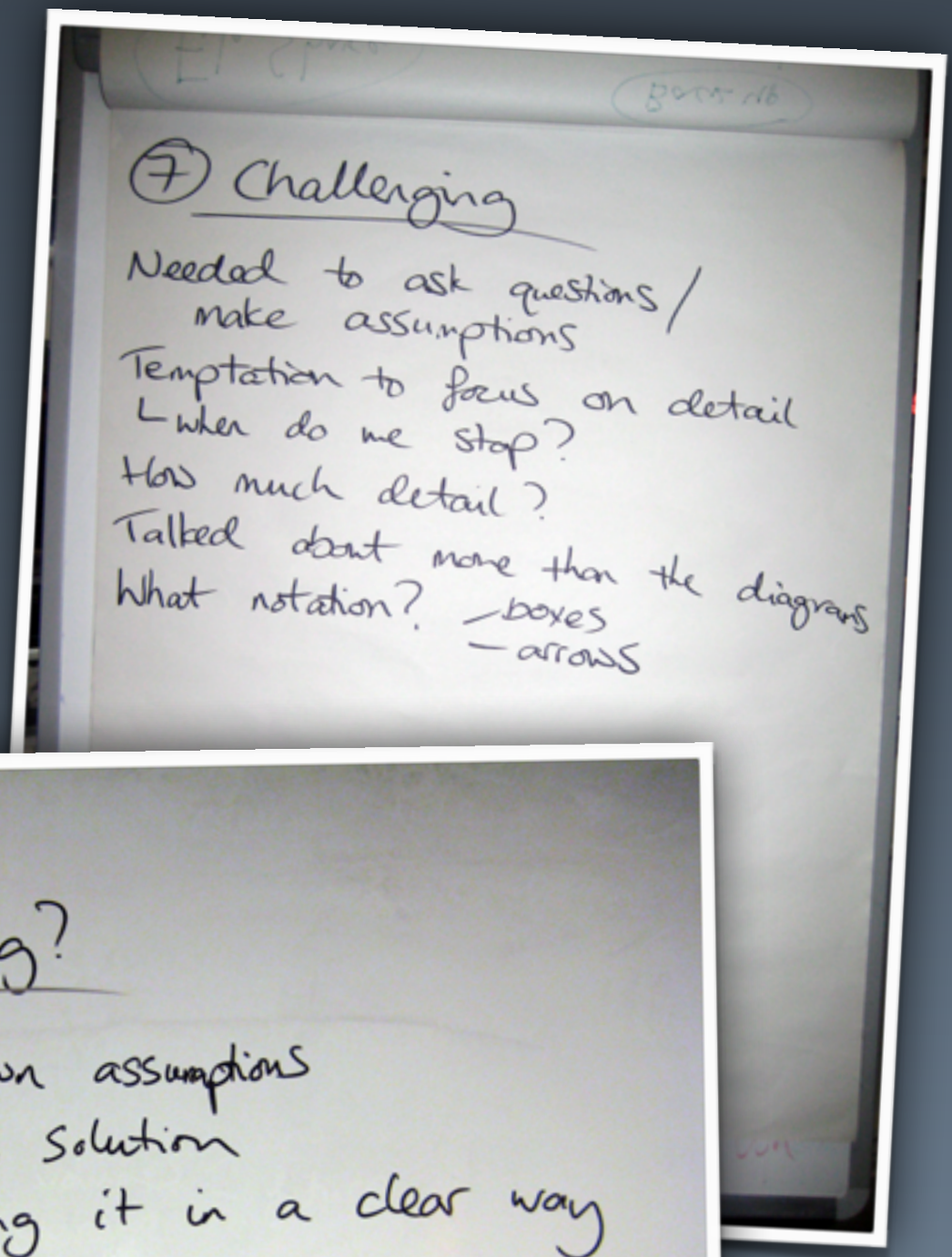
What's the DB-like icon?

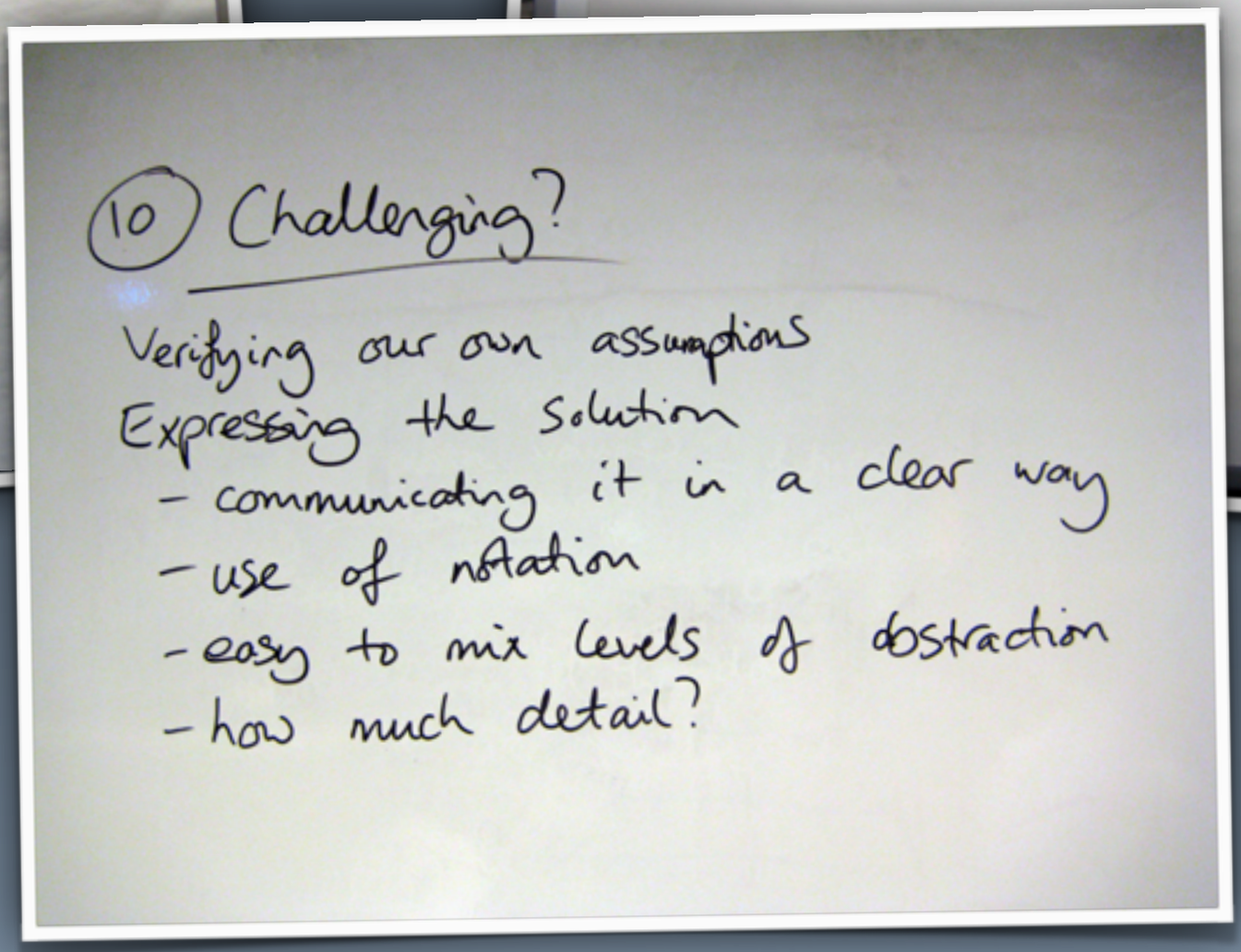
Not sure what this is

DIFFERENT LEVELS IN SAME DIAGRAM

ARE THE ARROWS THE RIGHT WAY ROUND?



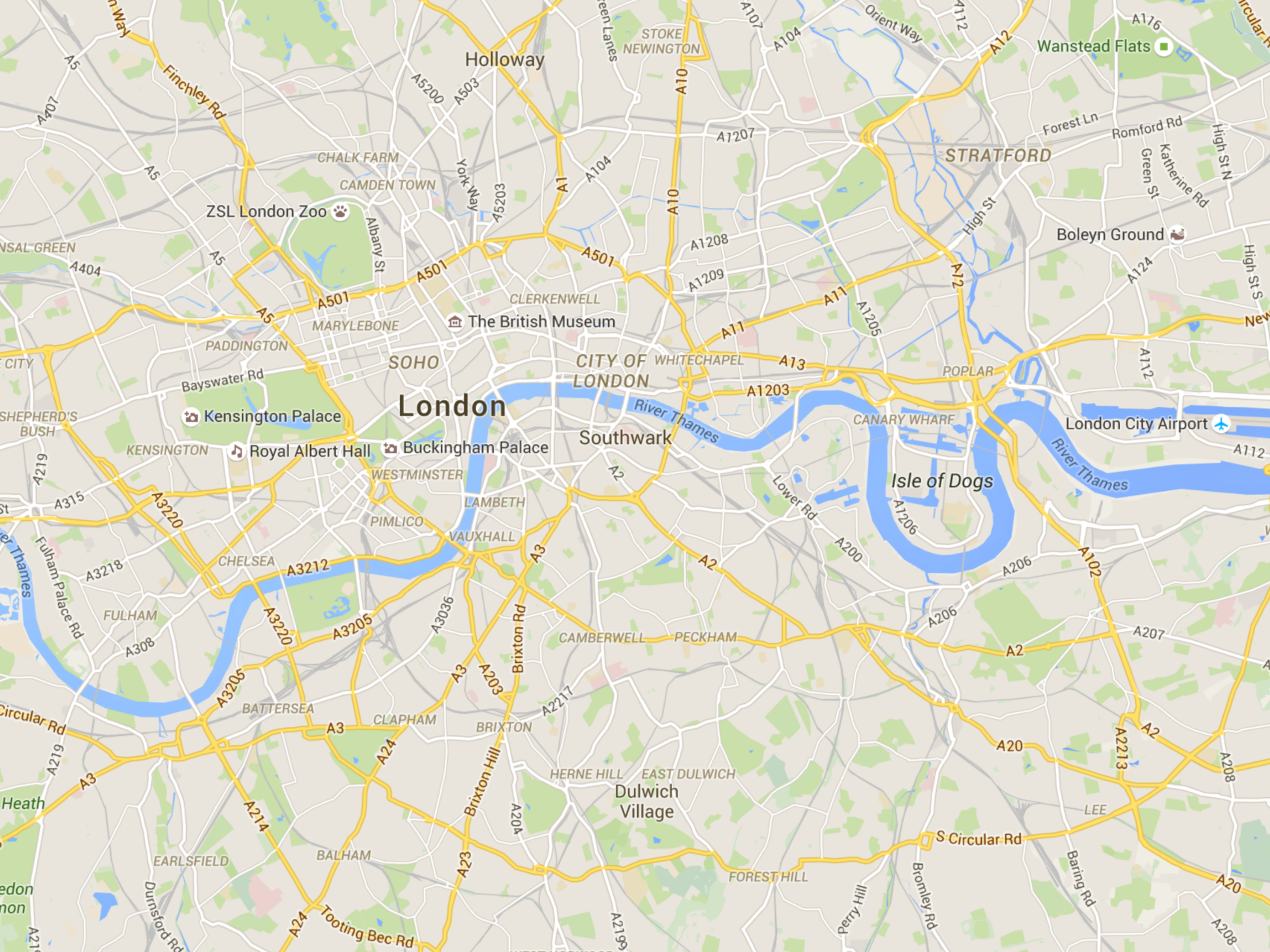


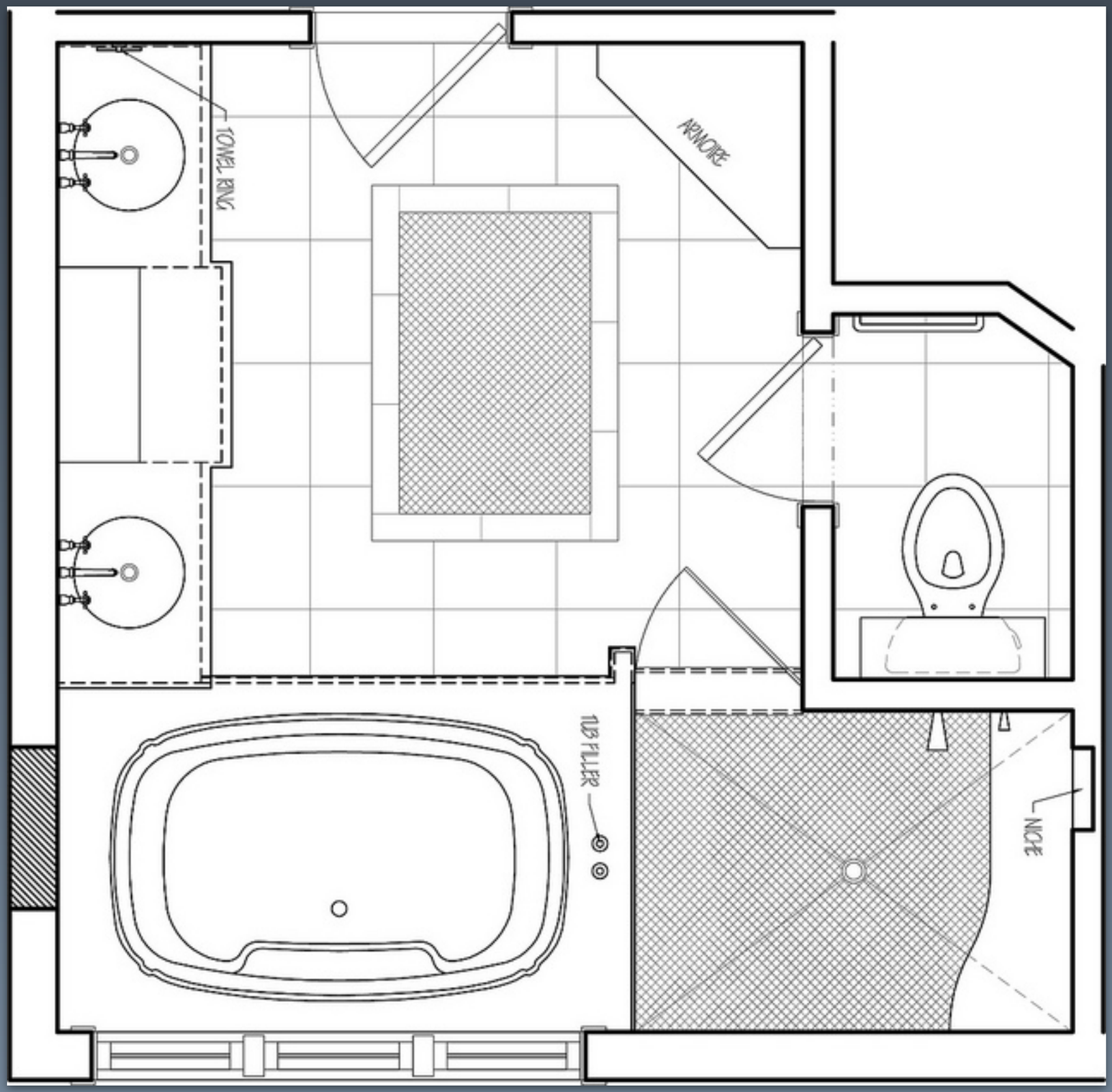


What's been challenging about the exercise?

Moving fast in the same direction requires good communication

As an industry, we lack a common vocabulary with which to think about, describe and communicate software architecture


Holloway
STOKE NEWINGTON
Wanstead Flats
STRATFORD
CHALK FARM
CAMDEN TOWN
ZSL London Zoo
Boleyn Ground
CLERKENWELL
The British Museum
MARYLEBONE
PADDINGTON
SOHO
CITY OF LONDON
WHITECHAPEL
POPLAR
Bayswater Rd
London
Kensington Palace
Royal Albert Hall
Buckingham Palace
Southwark
River Thames
CANARY WHARF
London City Airport
KENSINGTON
WESTMINSTER
LAMBETH
PIMLICO
VAUXHALL
Isle of Dogs
CHELSEA
FULHAM
CAMBERWELL
PECKHAM
BATTERSEA
CLAPHAM
BRIXTON
HERNE HILL
EAST DULWICH
Dulwich Village
LEE
EARLSFIELD
BALHAM
FOREST HILL
Finchley Rd
Forest Ln
Romford Rd
Orient Way
Lower Rd
Fulham Palace Rd
Tooting Bec Rd
S Circular Rd
Brixton Rd
Brixton Hill
Baring Rd
Bromley Rd
Perry Hill
A1
A2
A3
A5
A10
A11
A12
A13
A20
A23
A24
A104
A107
A112
A116
A124
A200
A201
A202
A203
A204
A206
A207
A208
A214
A219
A315
A316
A400
A401
A404
A406
A407
A501
A503
A1200
A1201
A1202
A1203
A1205
A1206
A1207
A1208
A1209
A2199
A2213
A2217
A3036
A3205
A3212
A3218
A3220
A5200
A5203
A308
Albany St
York Way
Green Lanes
High St
High St N
Green St
Katherine Rd



Floor plans

Circuit diagrams
(pictorial or schematic)

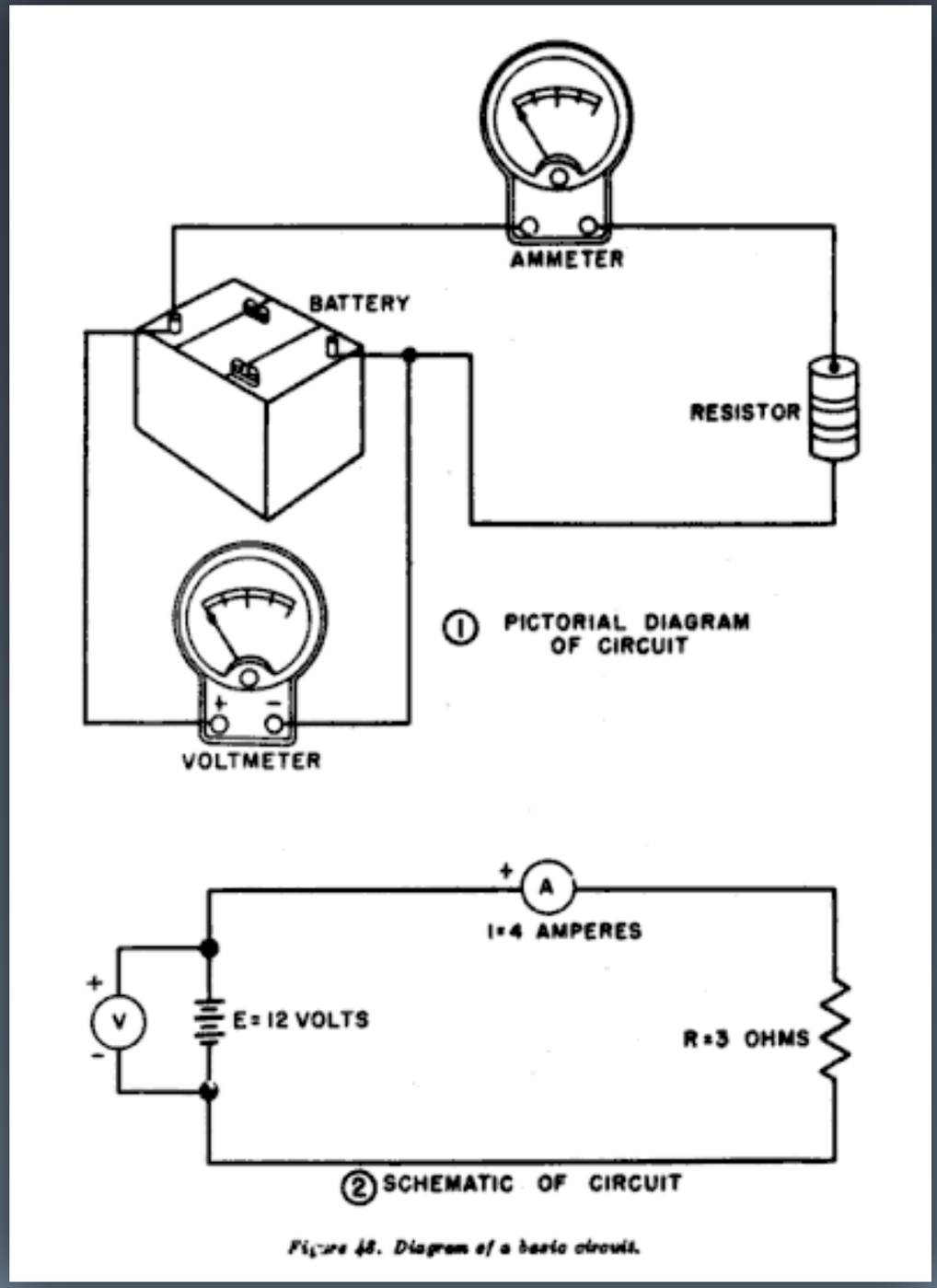
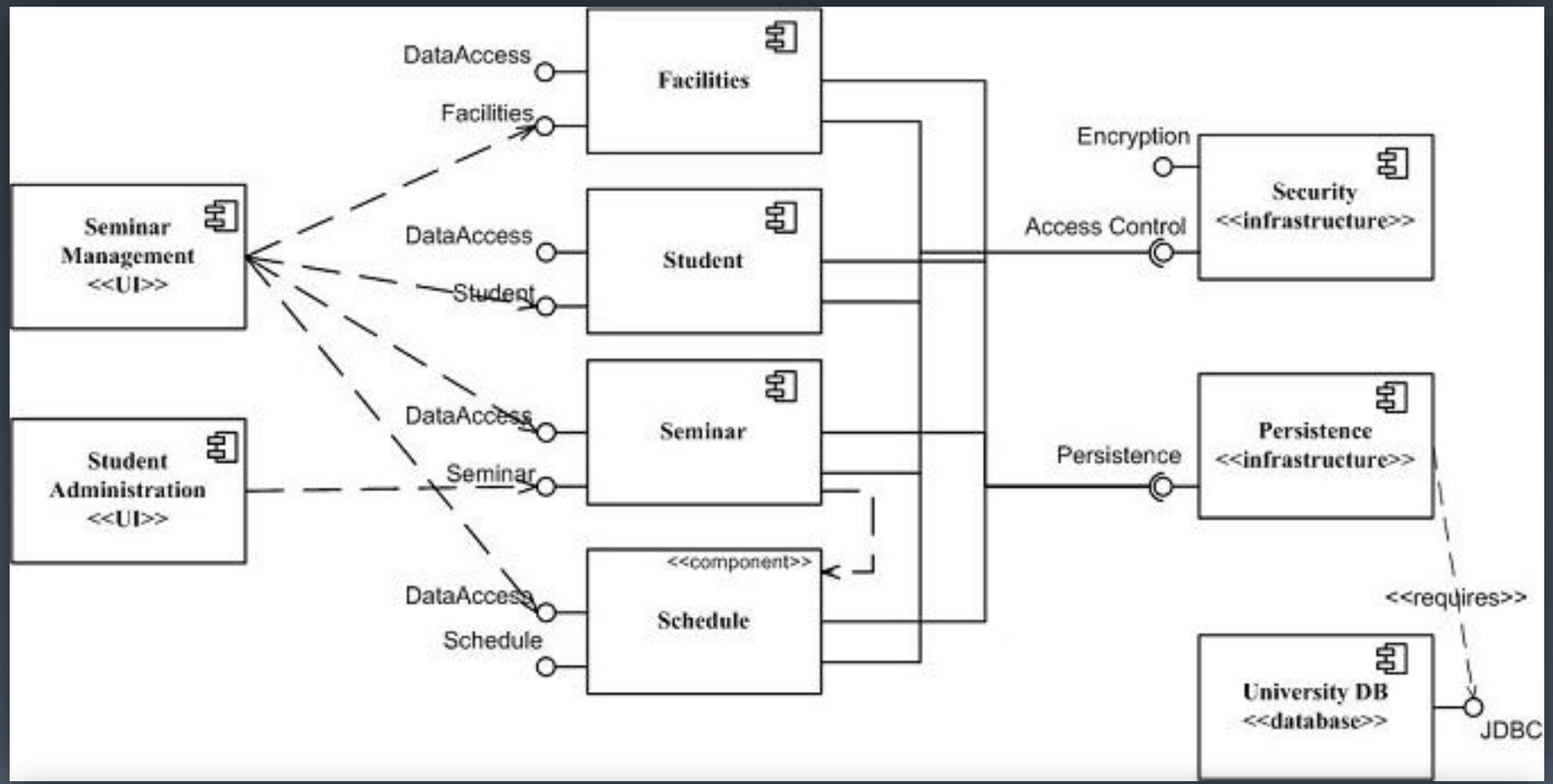
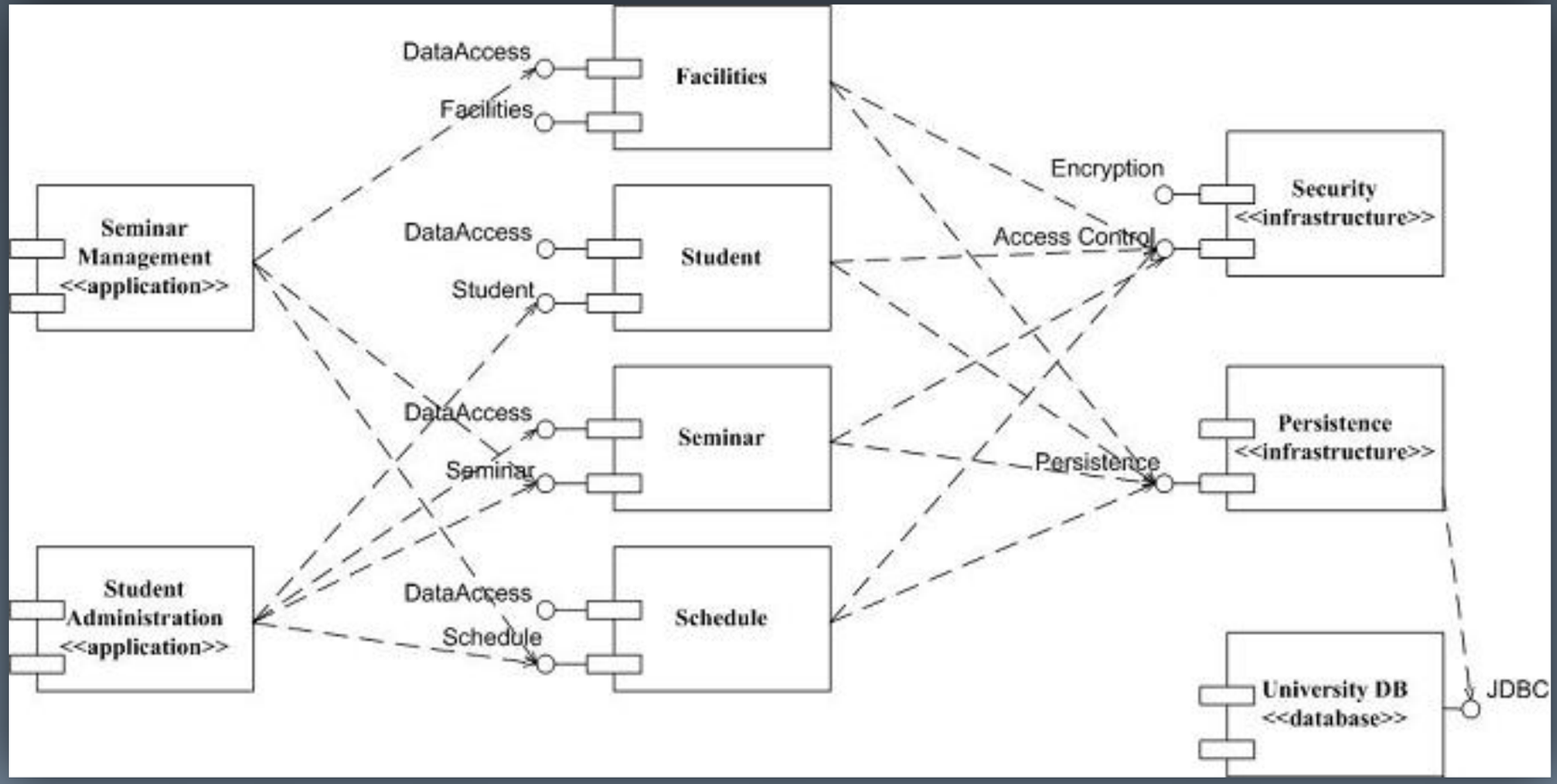
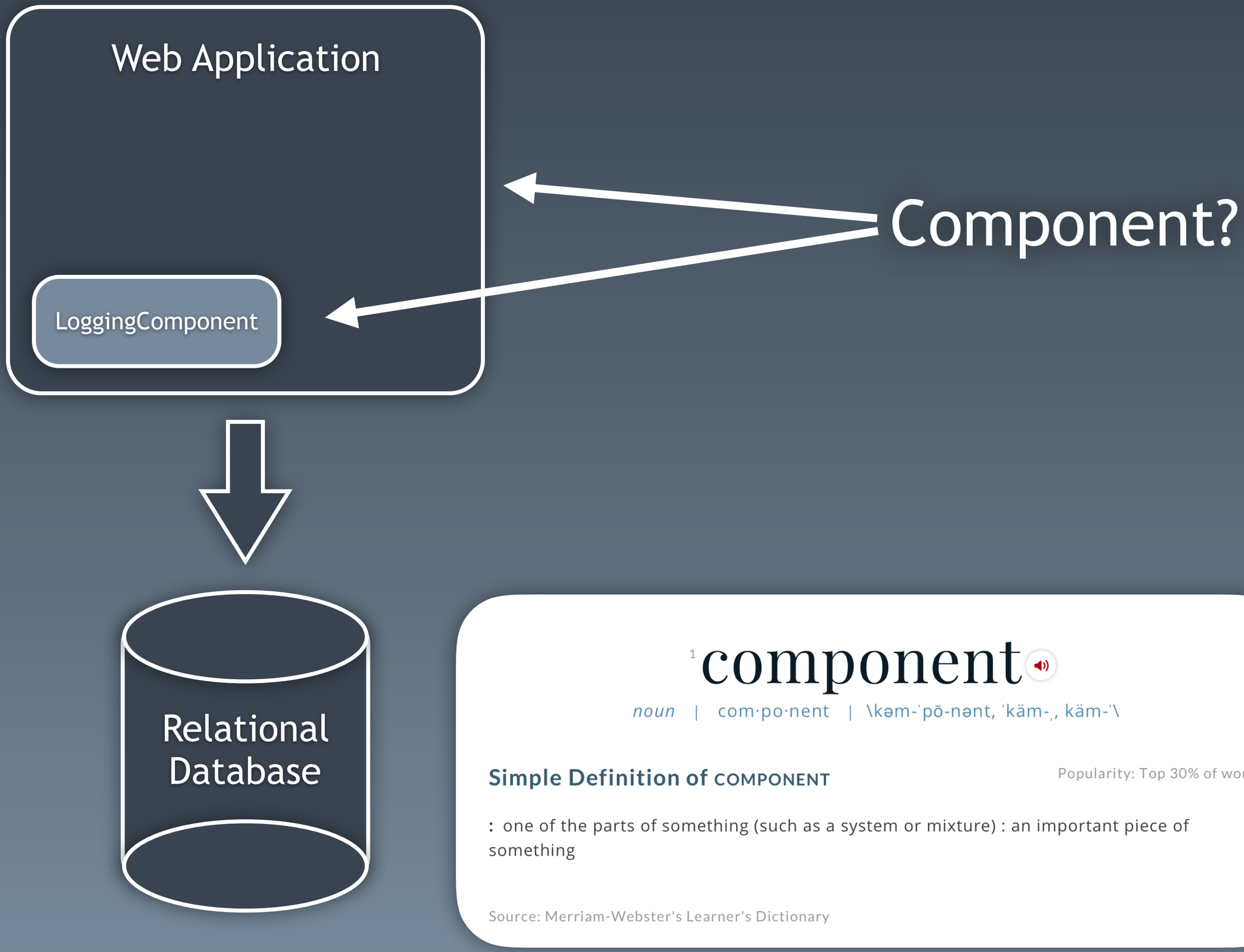


Figure 48. Diagram of a basic circuit.





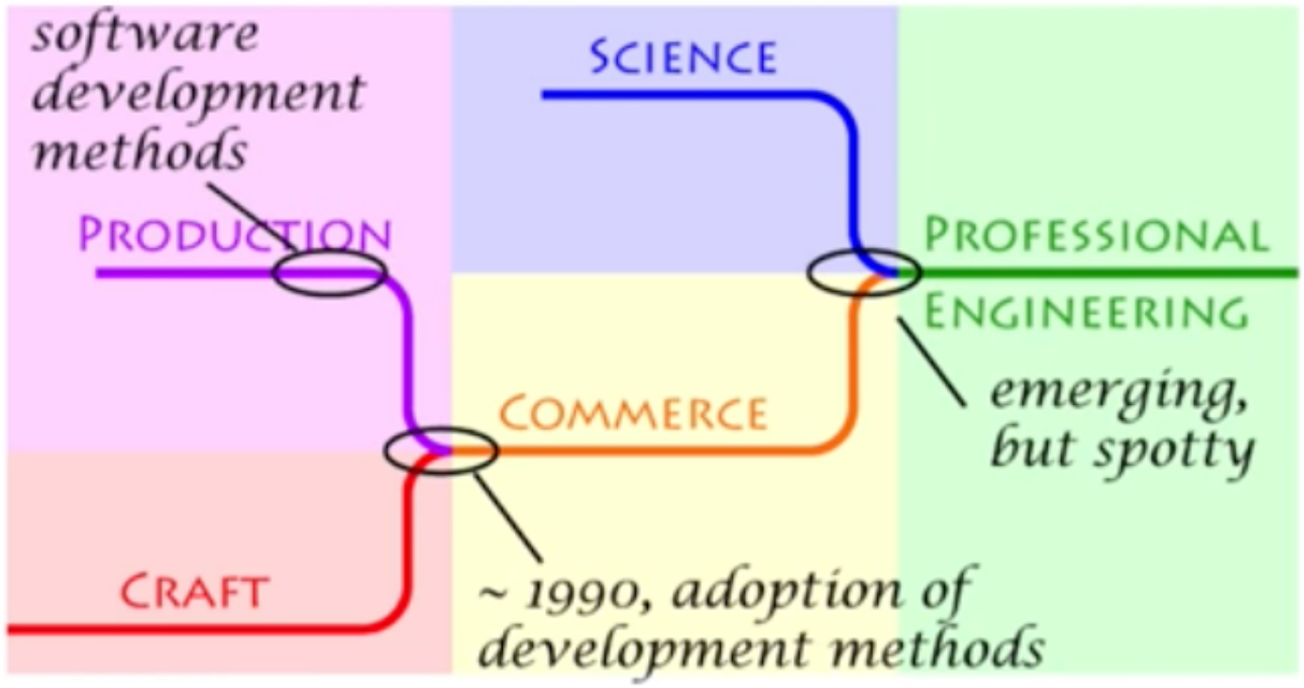
http://agilemodeling.com/artifacts/componentDiagram.htm

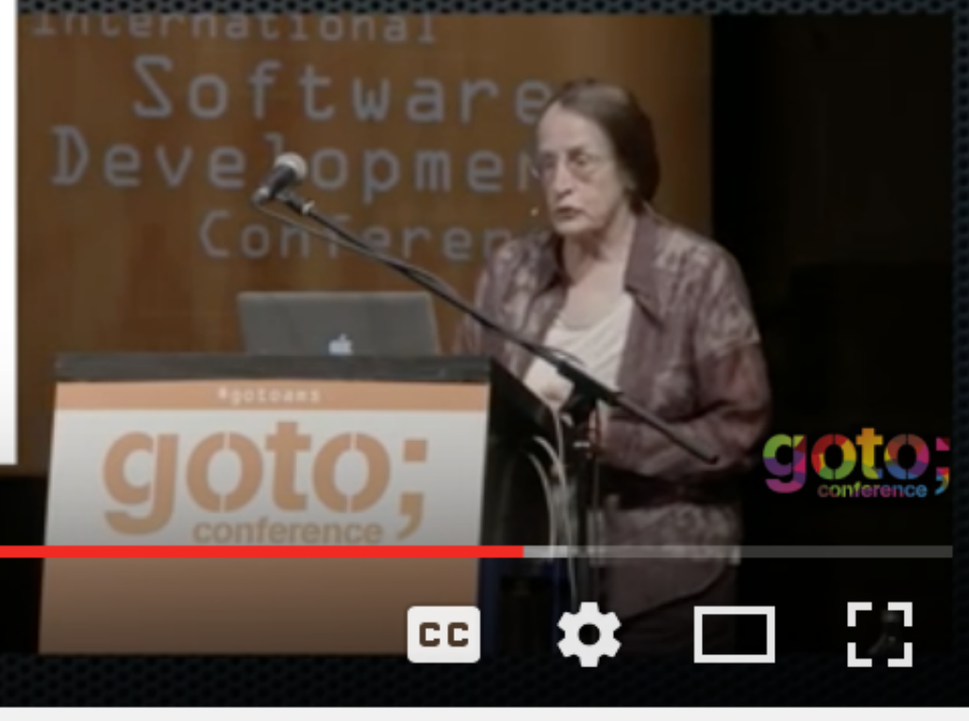

Web Application
LoggingComponent
Component?
Relational Database
1 component
noun | com·po·nent | \kəm-ˈpō-nənt, ˈkäm-ˌ, käm-ˈ\
Simple Definition of COMPONENT
Popularity: Top 30% of words
: one of the parts of something (such as a system or mixture) : an important piece of something
Source: Merriam-Webster's Learner's Dictionary

Ubiquitous language

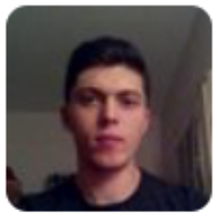
Software development is like

${metaphor}





GOTO 2015 • Progress Toward an Engineering Discipline of Software • Mary Shaw



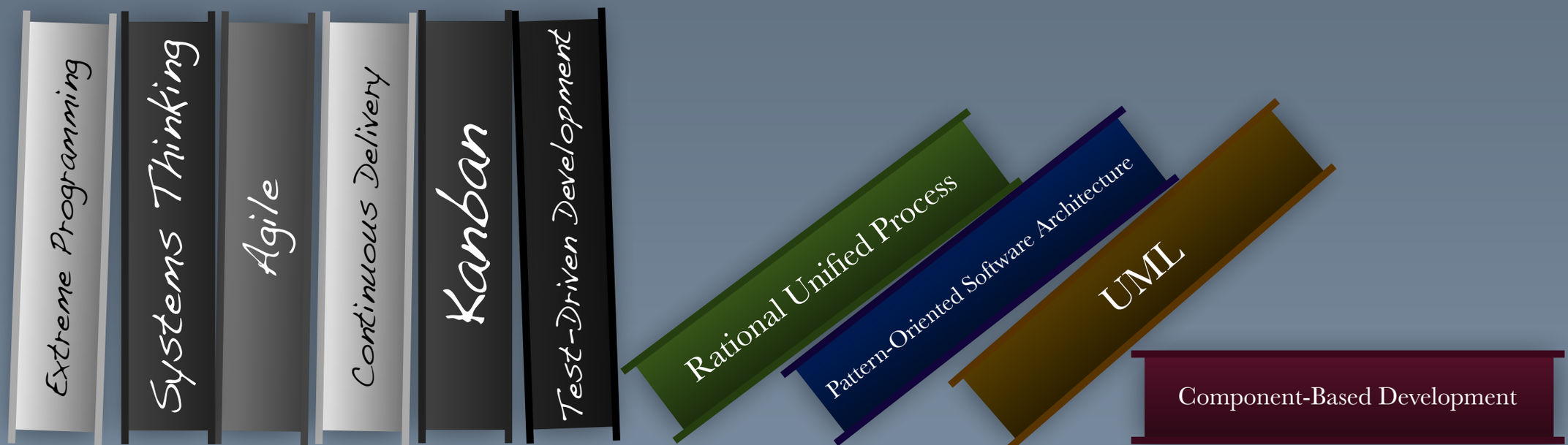
Dominik Földi
@dominikfoldi

I don't know why do we learn about #SSADM methodology at the University. It's a very old waterfall technique with unnecessary documentation.

7:48 PM - 15 May 2016

https://twitter.com/dominikfoldi/status/731919147613949952

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



Ensure you understand

the **value** of

the techniques you use

Do whatever works for

you

Let's learn from past experience rather than ignoring it

simon.brown@codingthearchitecture.com
@simonbrown on Twitter