Software Architecture for Developers

@simonbrown

Five things every developer should know about software architecture

1. Software architecture isn't about big design up front

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

“Waterfall”

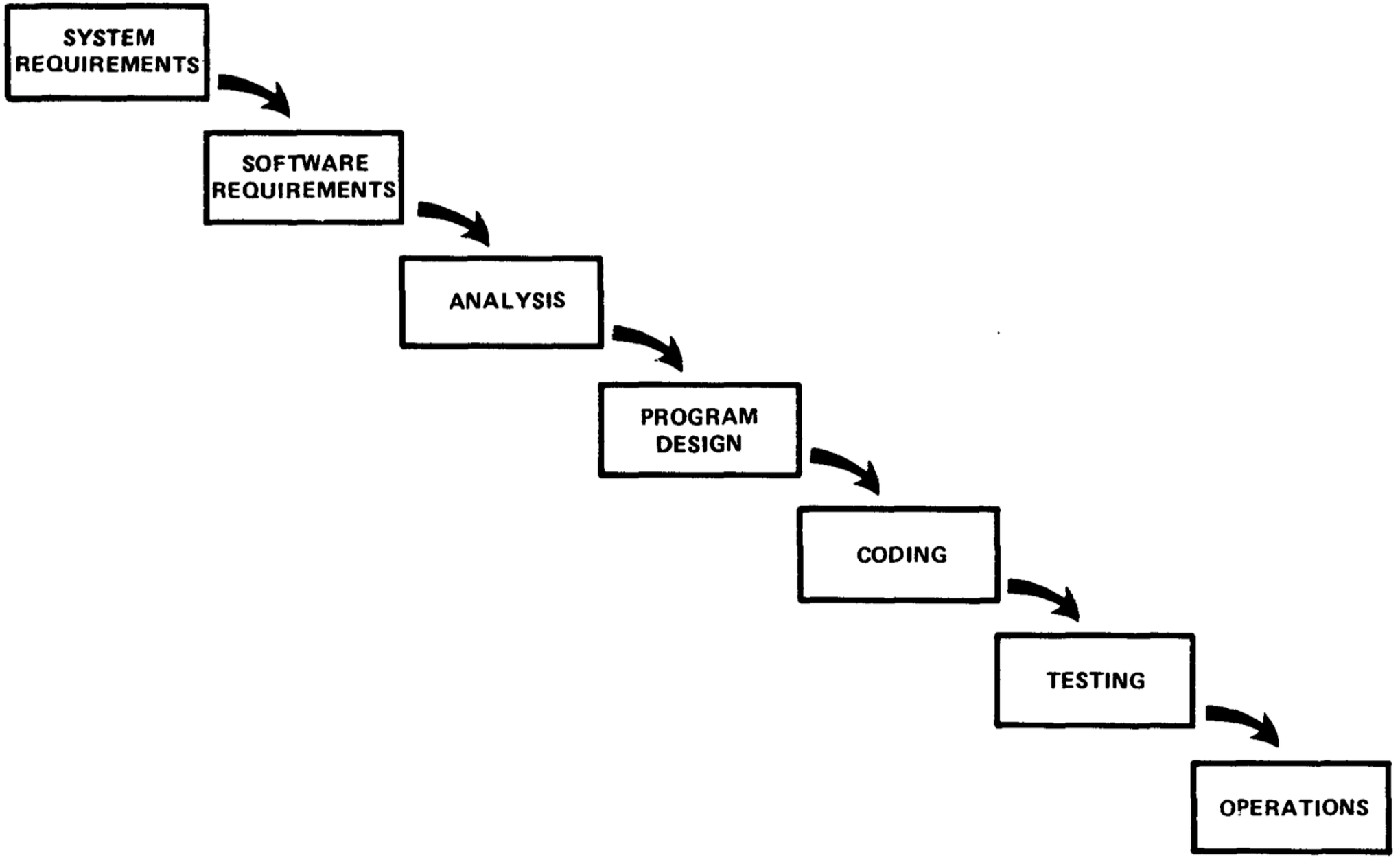


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

I believe in this concept, but the implementation described above is risky and invites failure.

Managing the development of large software systems

Dr Winston W. Royce

Responding to change
over
following a plan

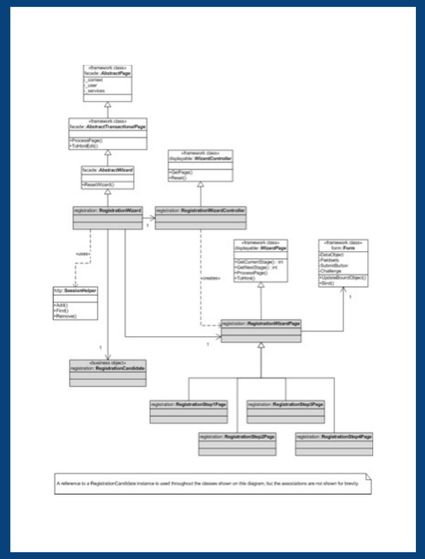
Moving fast, embracing change,
delivering value early, getting feedback

VS

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

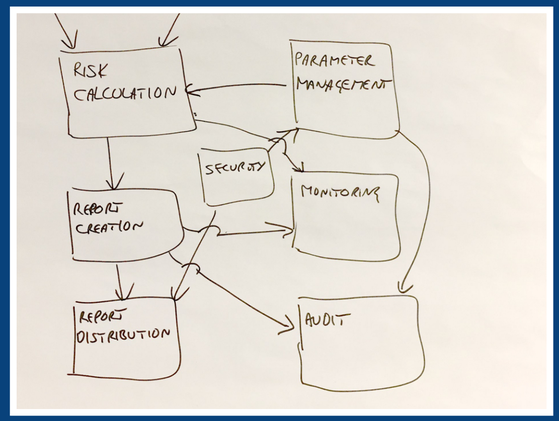
Big design up front

Software Architecture Document



vs

No design up front



Big design up front is dumb.
Doing no design up front
is even dumber.

Dave Thomas

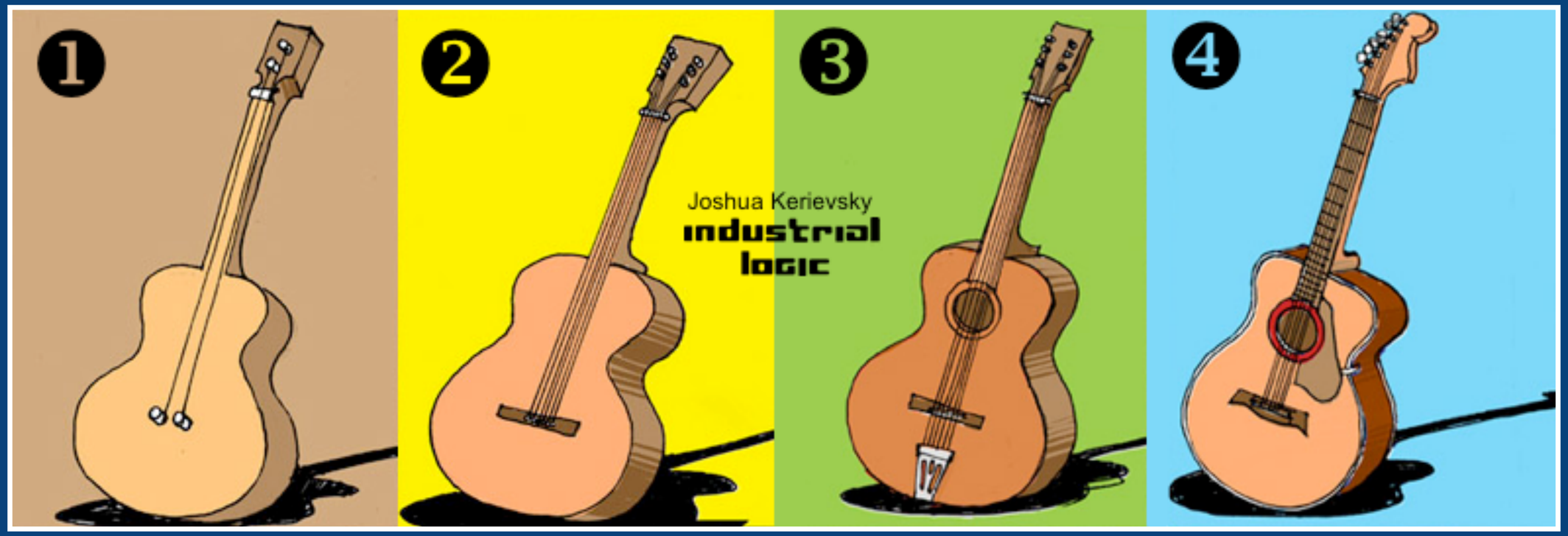
How much **up front design** should you do?

0% 100%

just enough

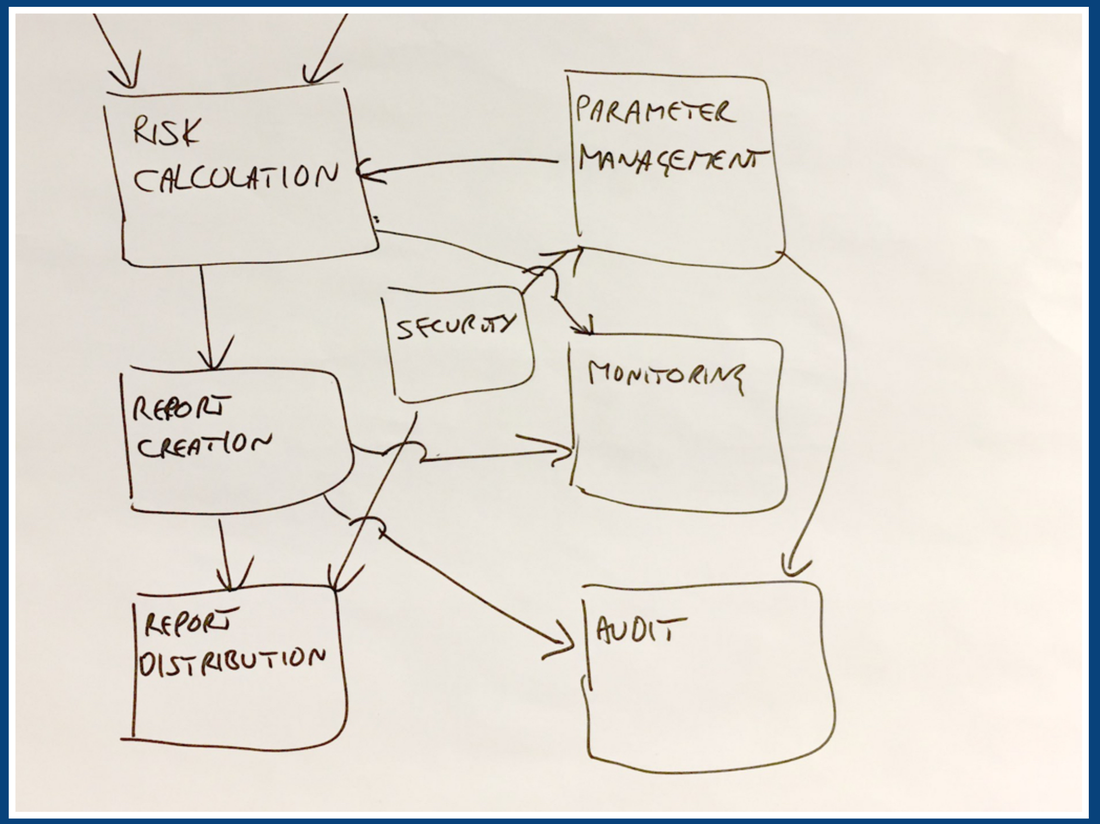
It’s not about creating a perfect end-state or complete architecture

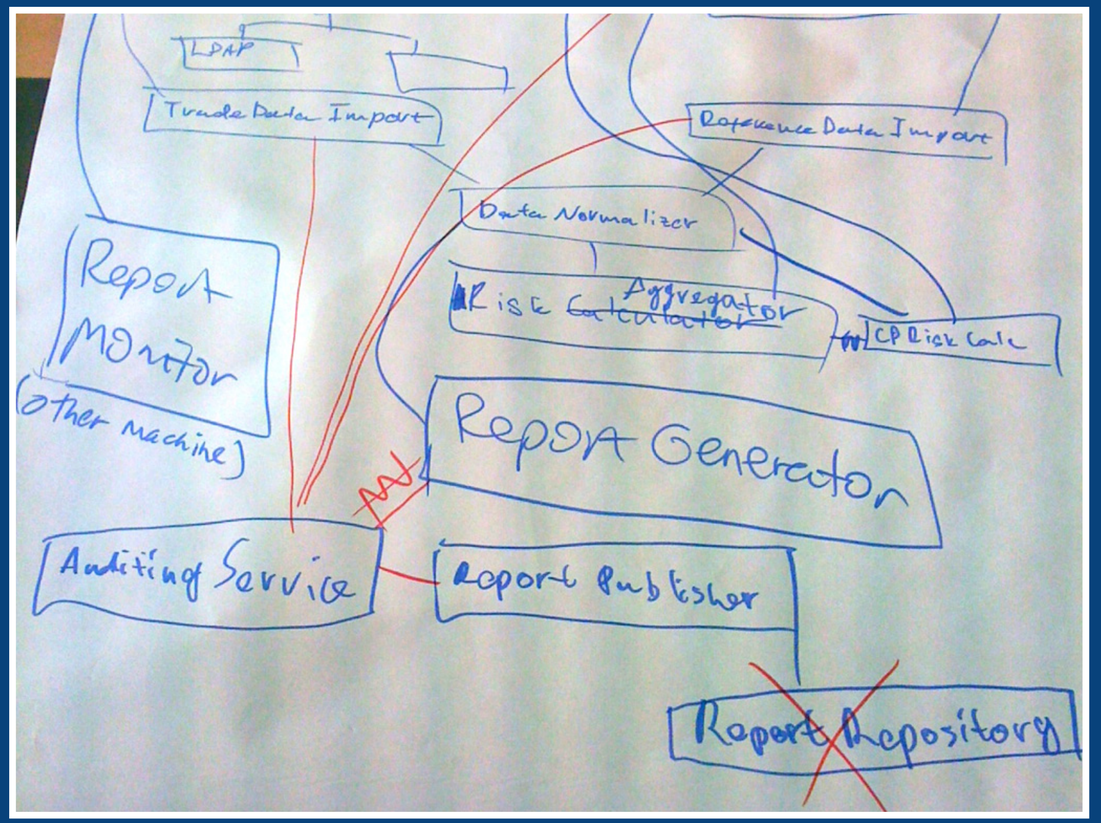
You need a starting point



Evolutionary Design

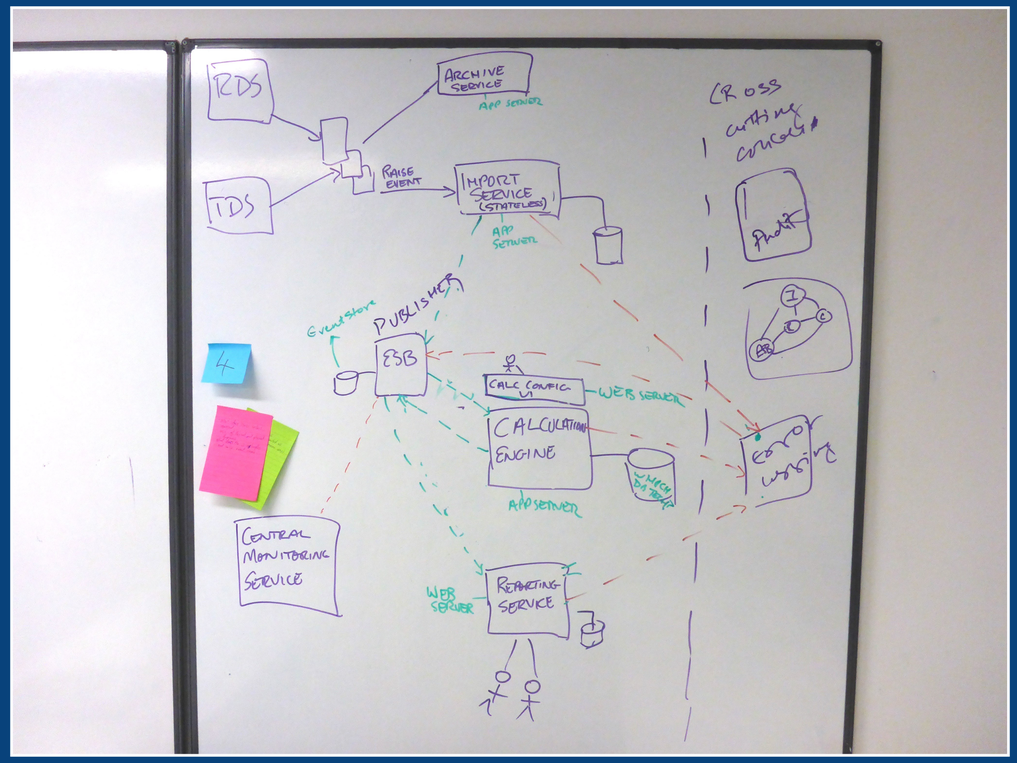
Beginning With A Primitive Whole





If you don’t **engage** in the problem, you end up with a very simplified and superficial view of the solution

1. Is that what we're going to **build**?



2. Is it going to **work**?

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

Grady Booch

Architecture

Programming language
Monolith, microservices or hybrid approach

Design

Implementation

Curly braces on the same or next line
Whitespace vs tabs

Identify and **mitigate**
your **highest priority risks**

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

Agile Architecture: Strategies for Scaling Agile Development

Scott Ambler

Concrete experiment

Proof of concept, prototype, spike, tracer, vertical slice, walking skeleton, executable reference architecture, …

Just enough up front design to create **firm and sufficient foundations**

Thinking about
software architecture lets you
stack the odds of success
in your favour

2. Every software team needs to consider software architecture

What happens if a software development team **doesn’t think about architecture**?

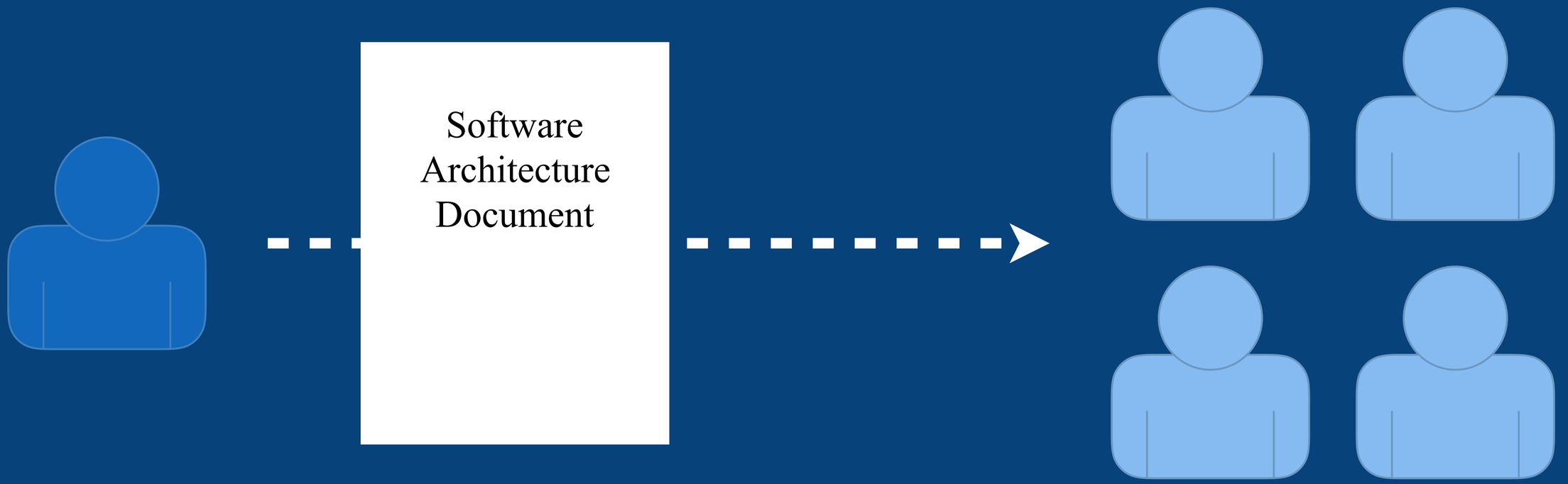
Chaos

Big ball of mud, spaghetti code, inconsistent approaches to solving the same problems, quality attributes are ignored, deployment problems, maintenance issues, etc

Every team needs
technical leadership

3. The software architecture role is about coding, coaching and collaboration

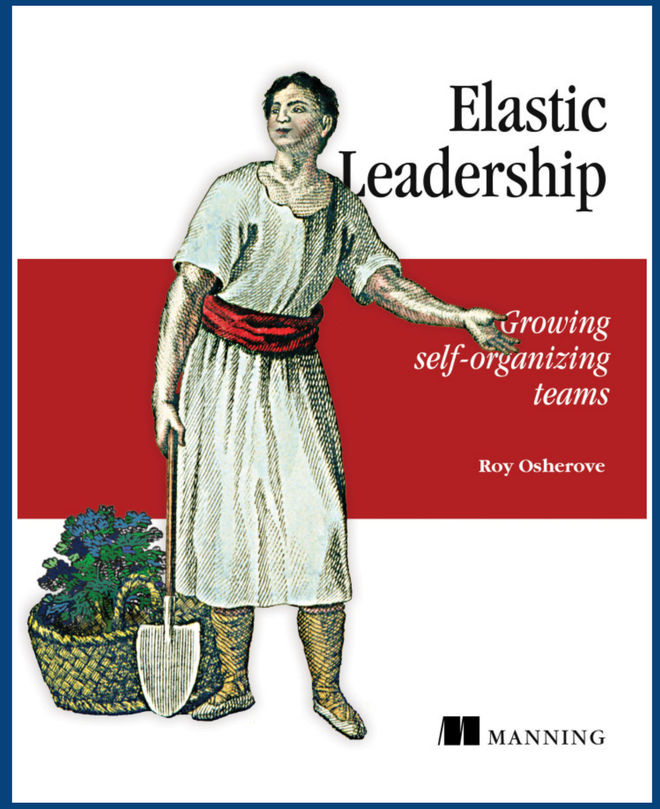
Software development is not a relay sport



AaaS

Architecture as a Service

Continuous technical leadership

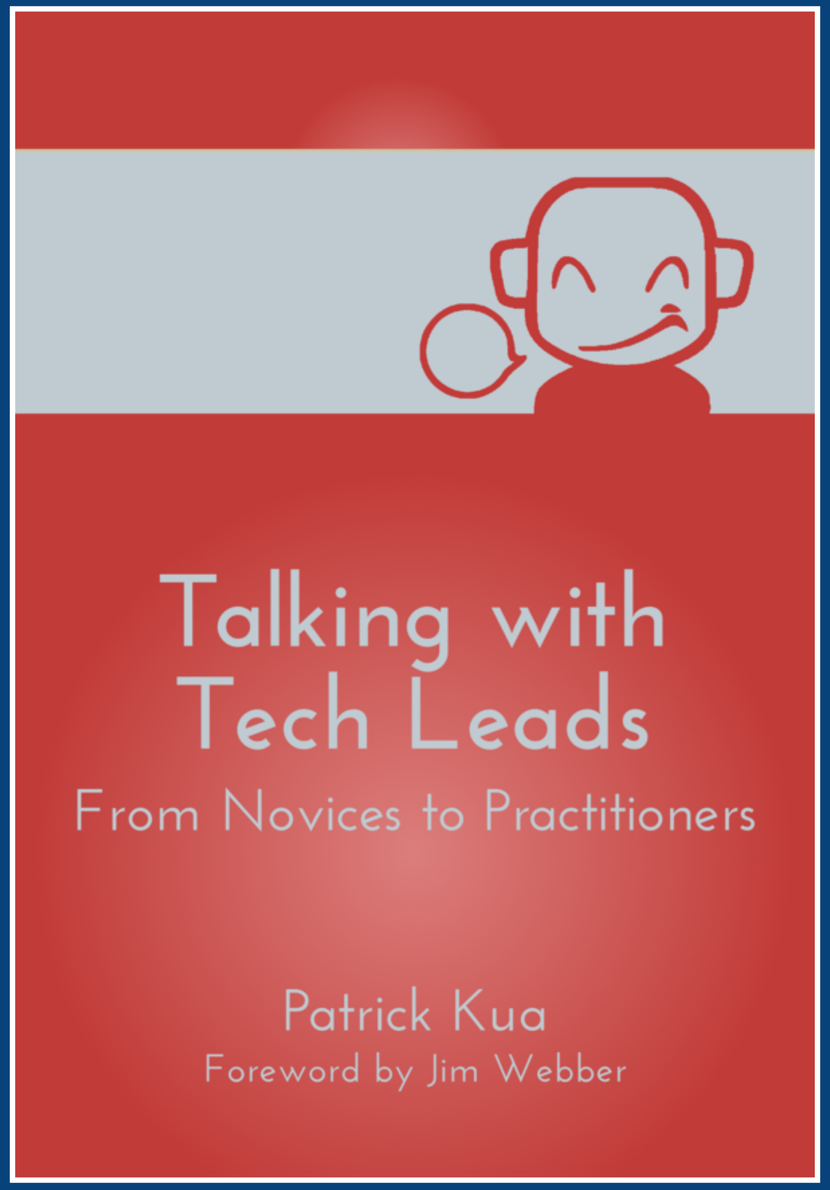


Different types of teams need different leadership styles

Pair architecting

Soft skills

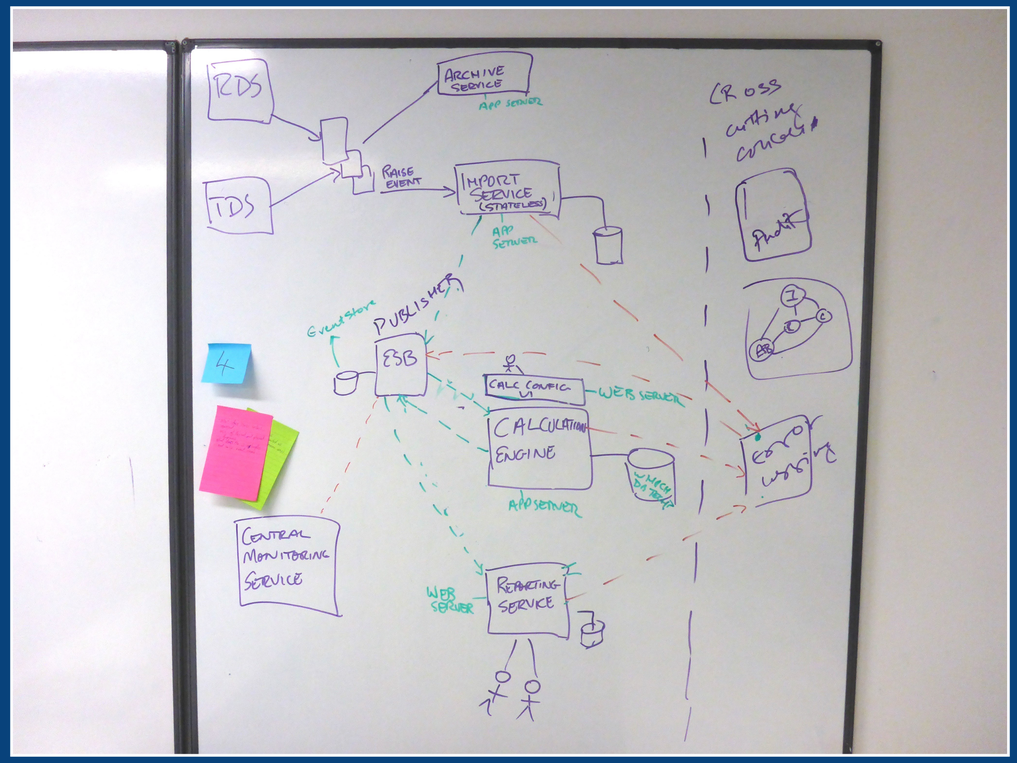
(leadership, communication, presentation, influencing, negotiation, collaboration, coaching and mentoring, motivation, facilitation, political, etc)


Talking with
Tech Leads
From Novices to Practitioners
Patrick Kua
Foreword by Jim Webber

Good software architects
are typically
good software developers

The people designing software must understand technology …
all decisions involve trade-offs

1. Is that what we're going to **build**?



2. Is it going to **work**?

Should software architects write **code**?

Production code, prototypes, frameworks, foundations, code reviews, experimenting, etc

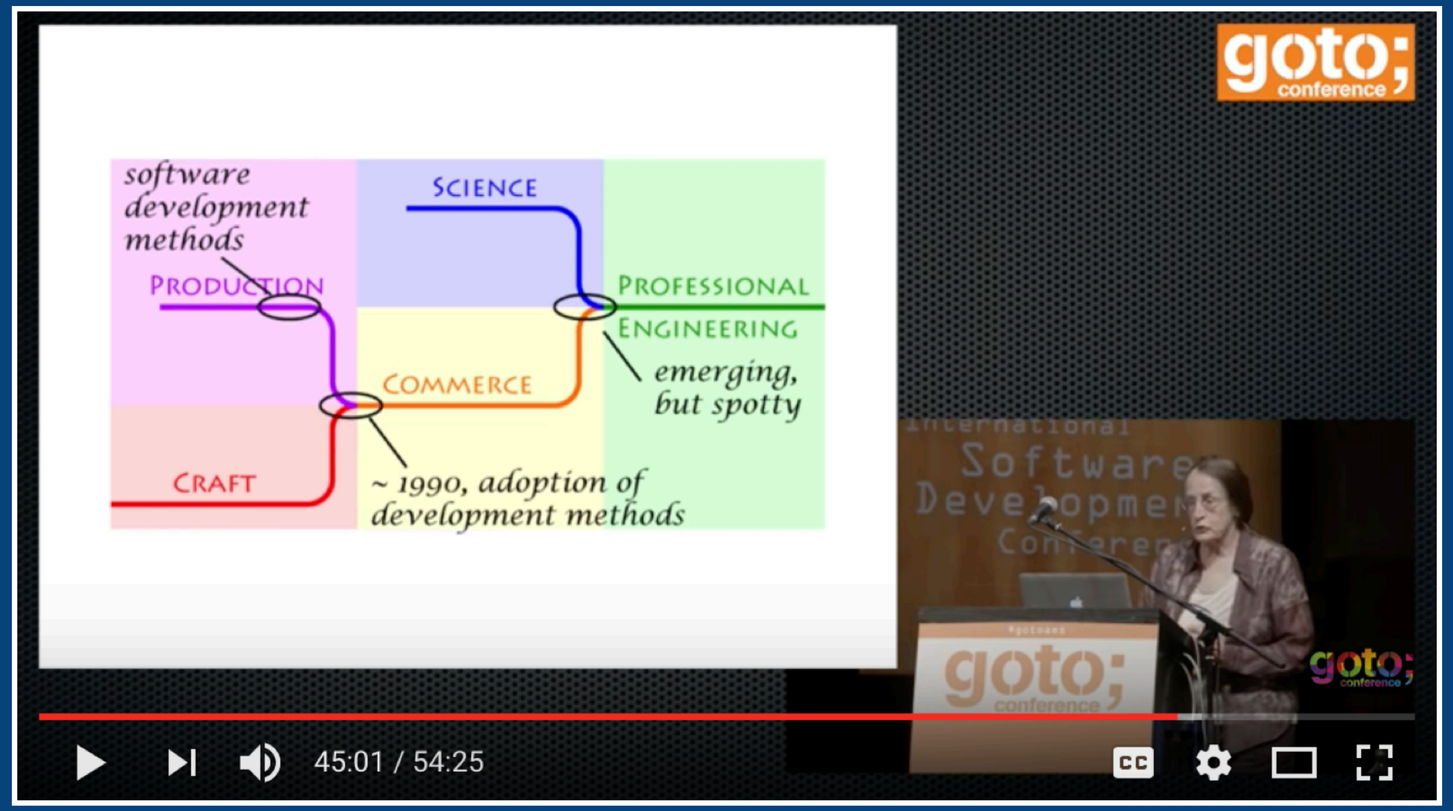
Don’t code all of the time!

Software architects
should be
master builders

I am a senior developer. Recently, I was promoted to the position as architect. Could anyone please let me know which tools/software an architect should master/be familiar with. Thank you

Experience is important ...

software architecture is not a rank!



Progress Toward an Engineering Discipline of Software

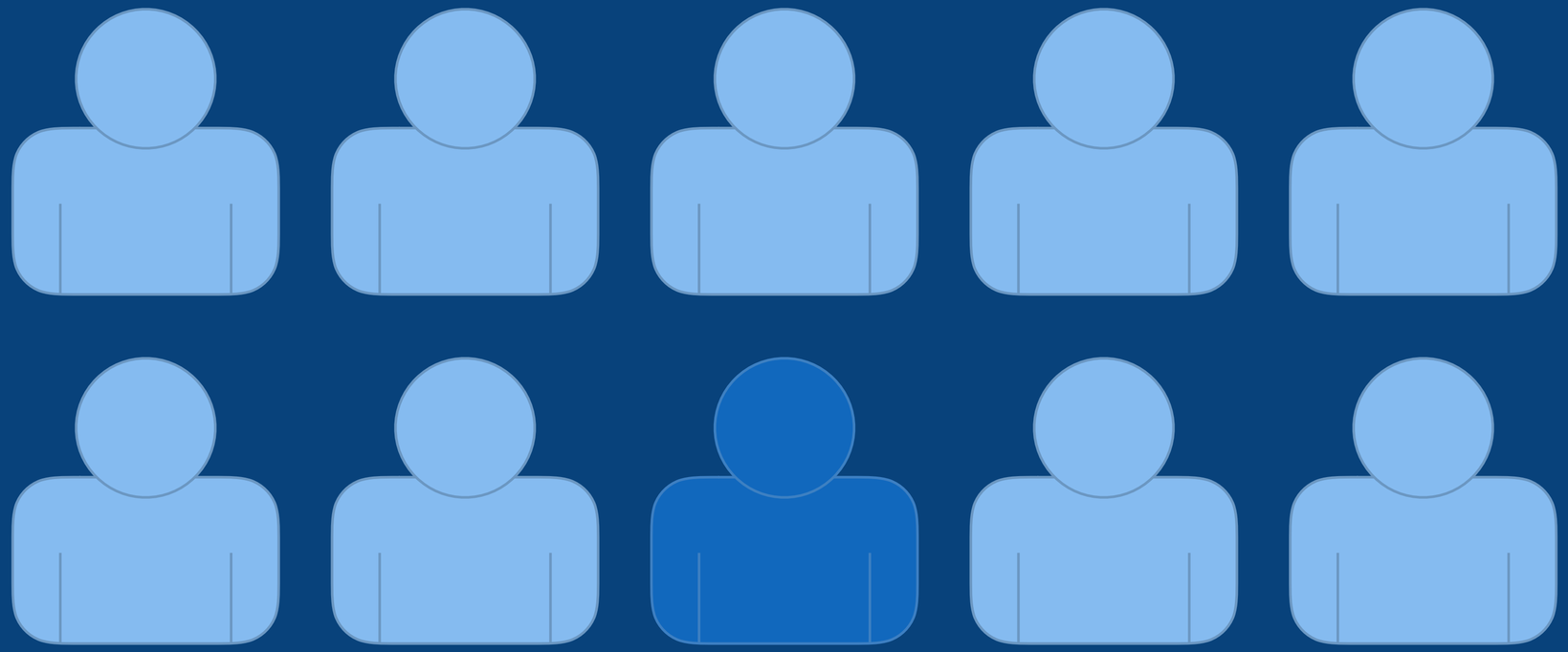
Mary Shaw

The software architecture role is **multi-faceted**

(technical depth, technical breadth, soft skills)

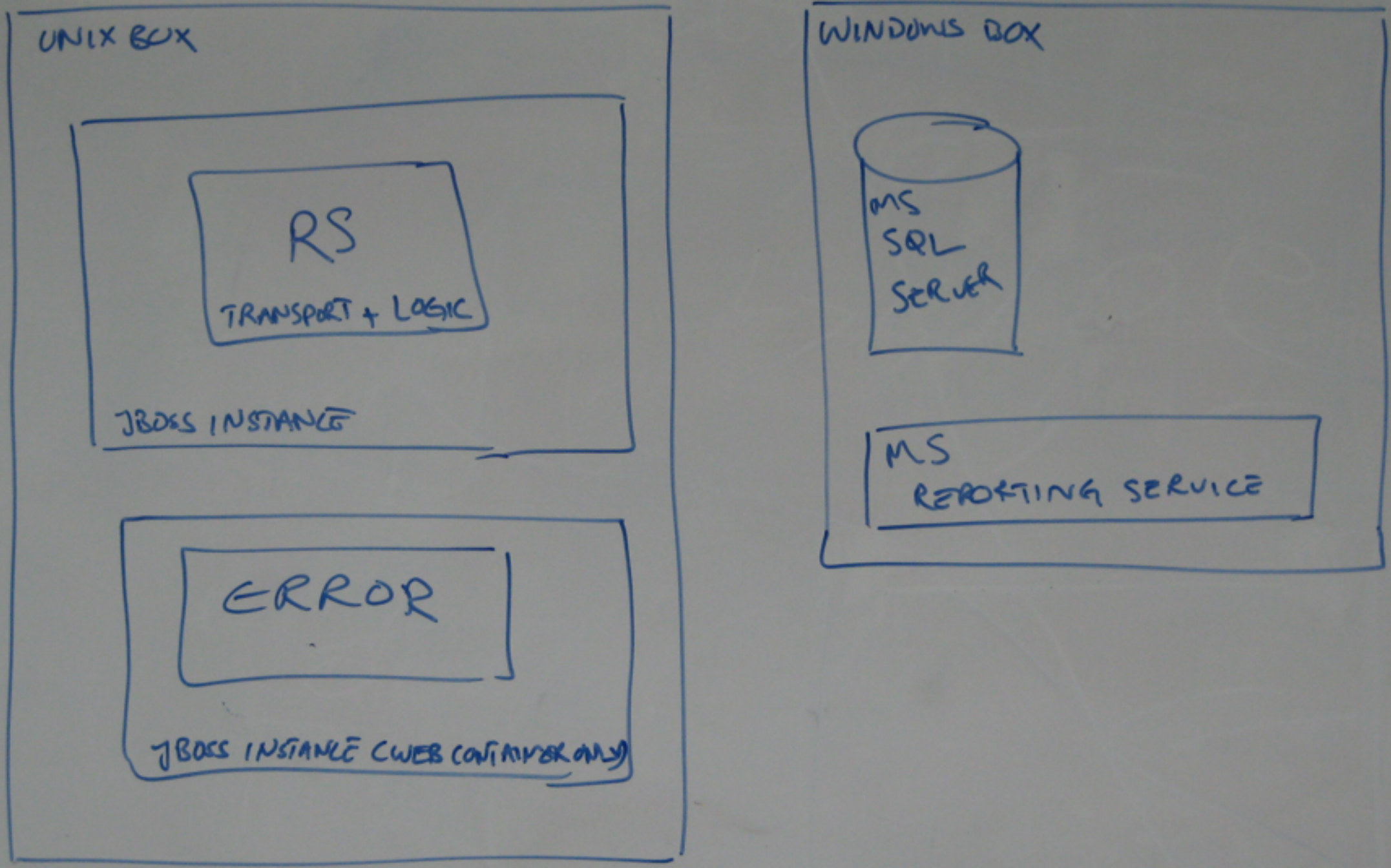
4. You don't need to use UML

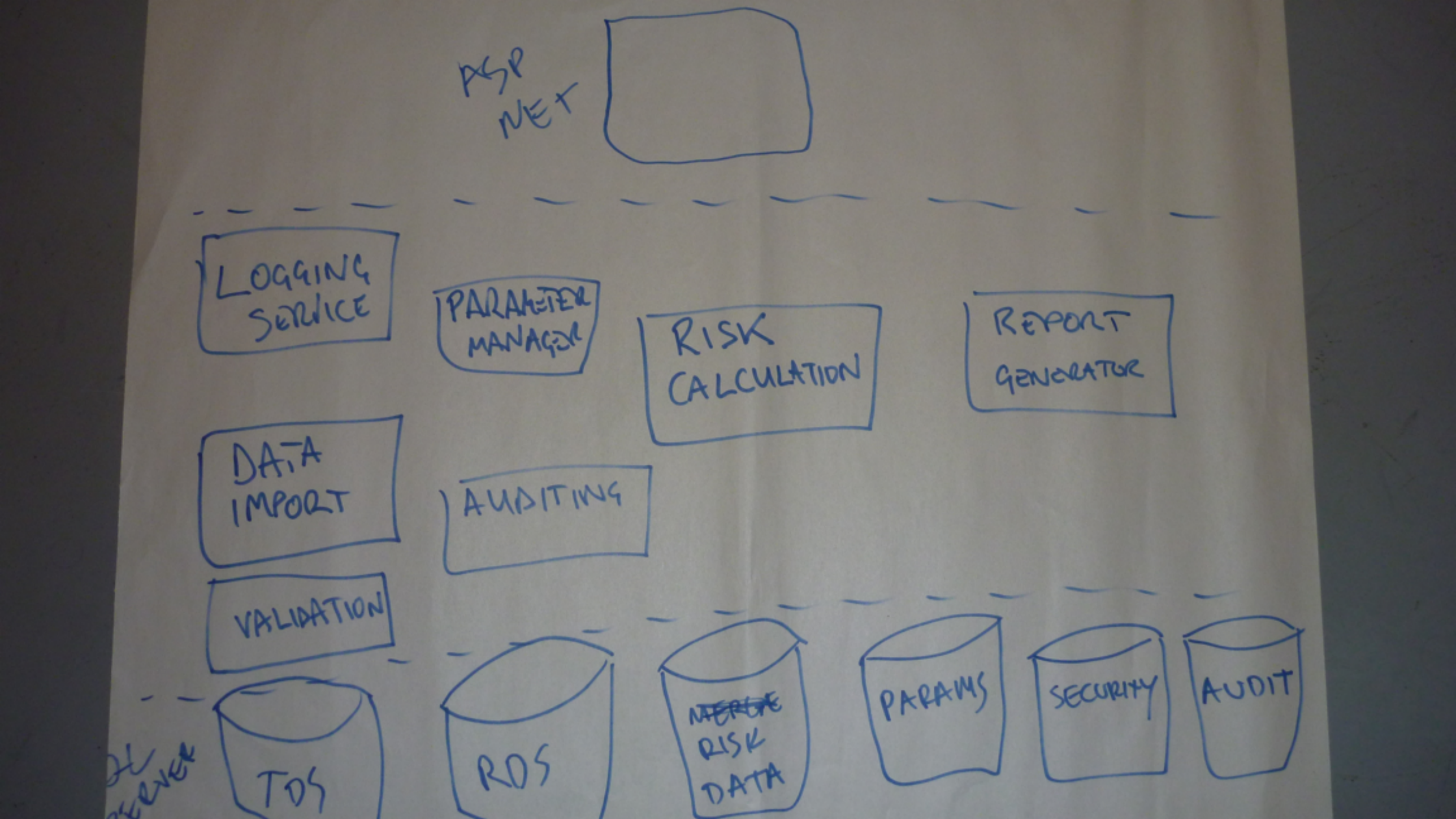
Do **you** use UML?



In my experience, optimistically,

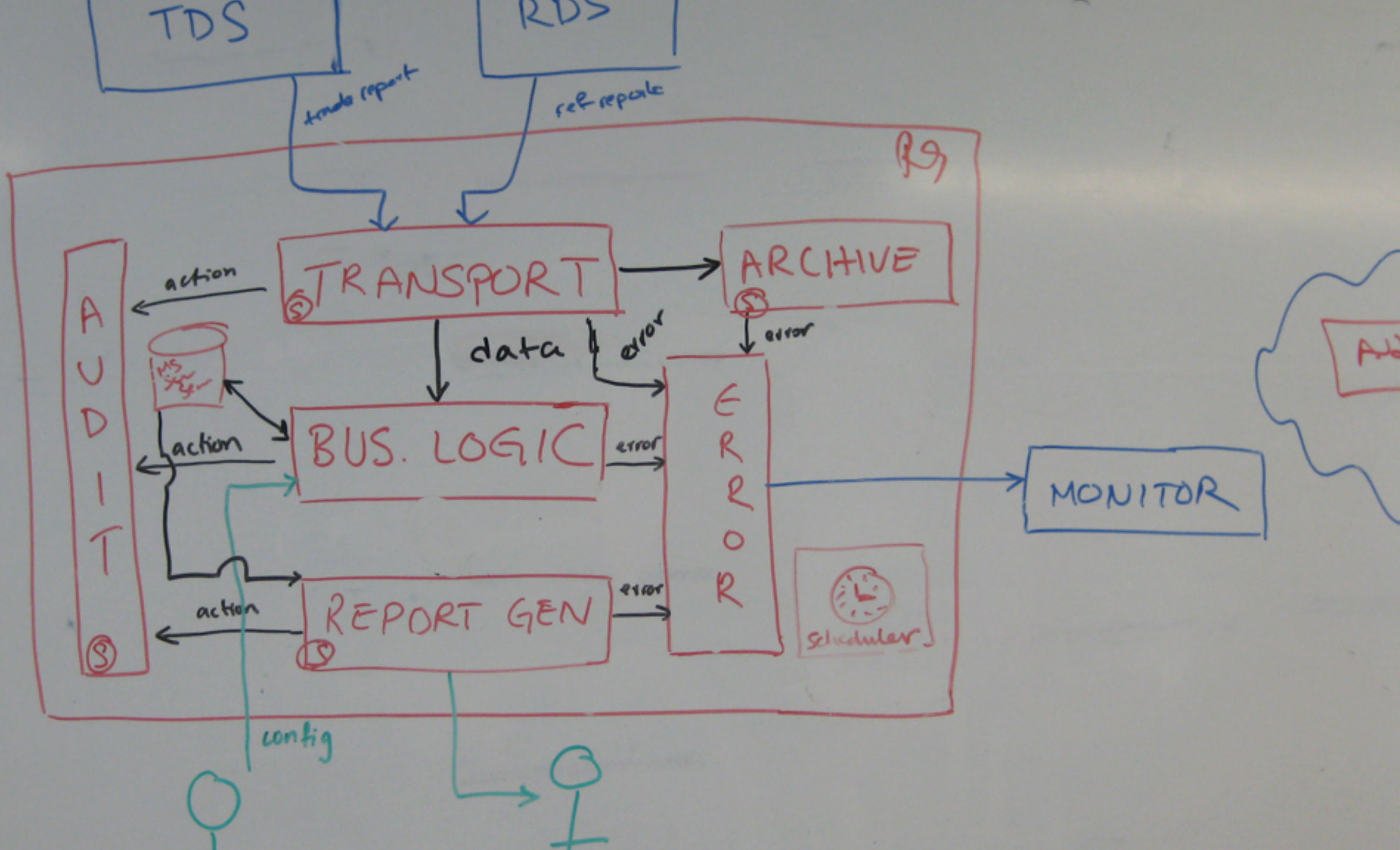
1 out of 10 people use UML

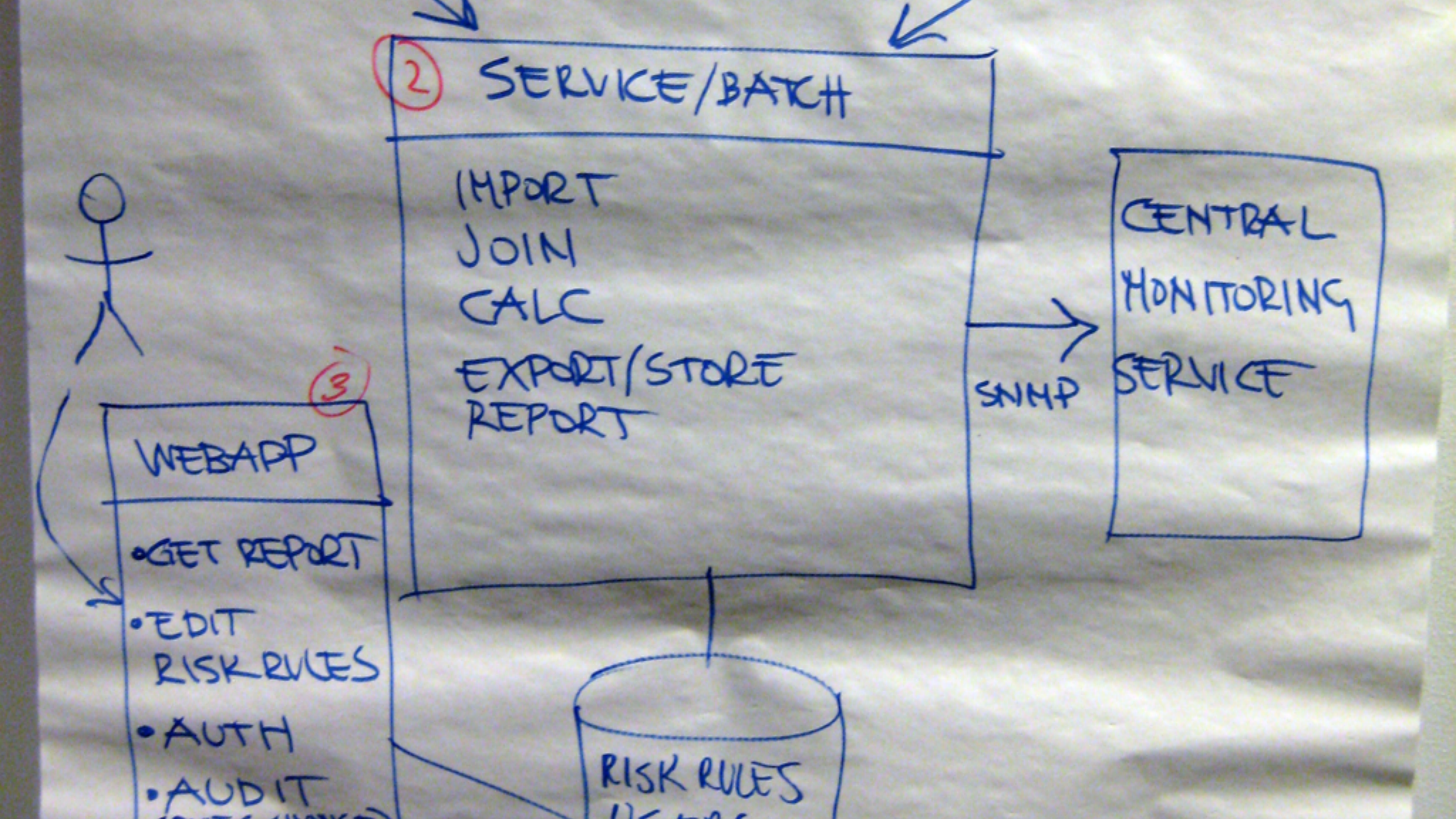

UNIX BOX
RS
TRANSPORT + LOGIC
JBOSS INSTANCE
ERROR
JBOSS INSTANCE (WEB CONTAINER ONLY)
WINDOWS BOX
MS
SQL
SERVER
MS
REPORTING SERVICE


ASP NET
LOGGING SERVICE
PARAMETER MANAGER
RISK CALCULATION
REPORT GENERATOR
DATA IMPORT
AUDITING
VALIDATION
TDS
RDS
RISK DATA
PARAMS
SECURITY
AUDIT

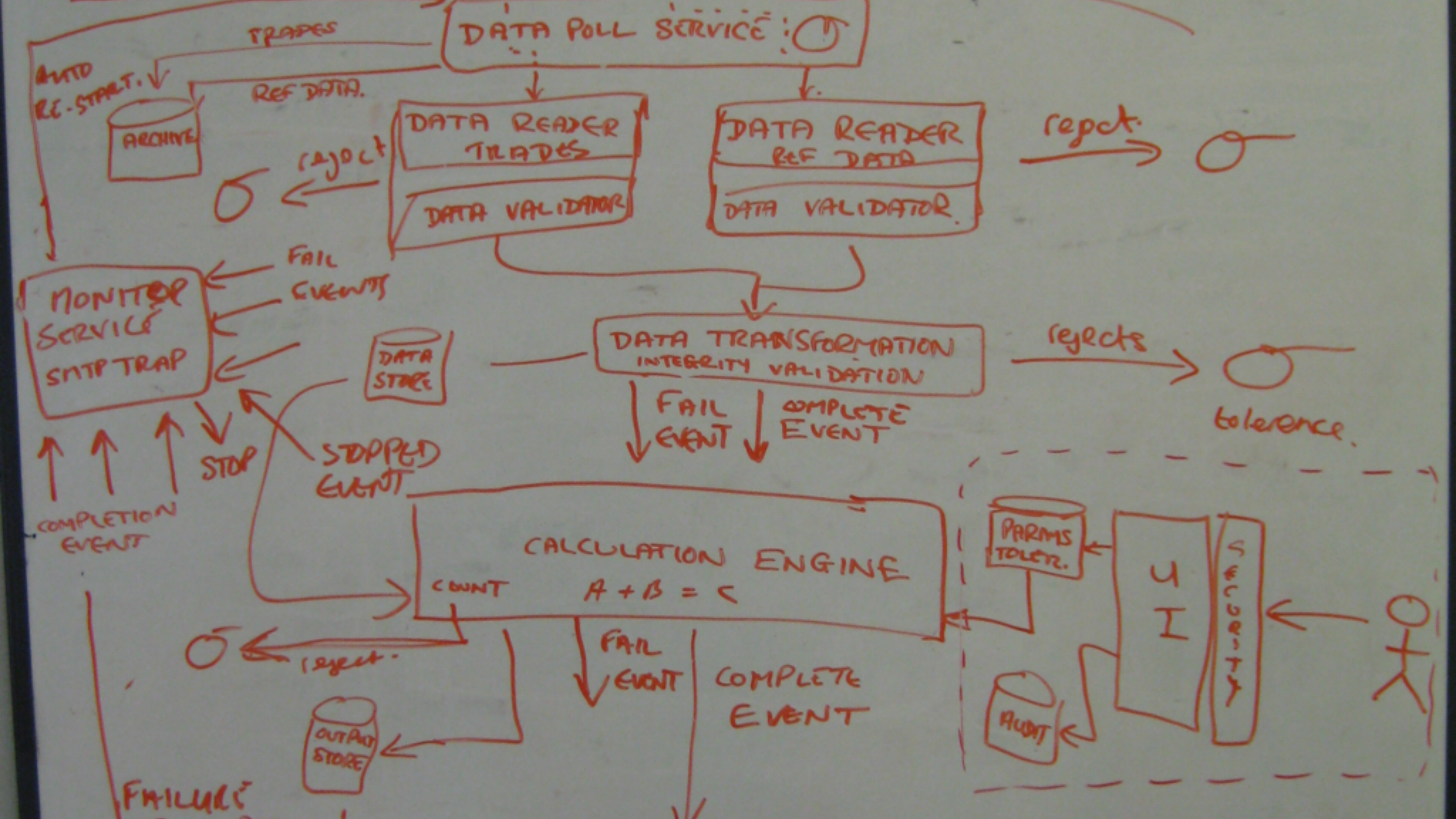
FUNCTIONAL VIEW

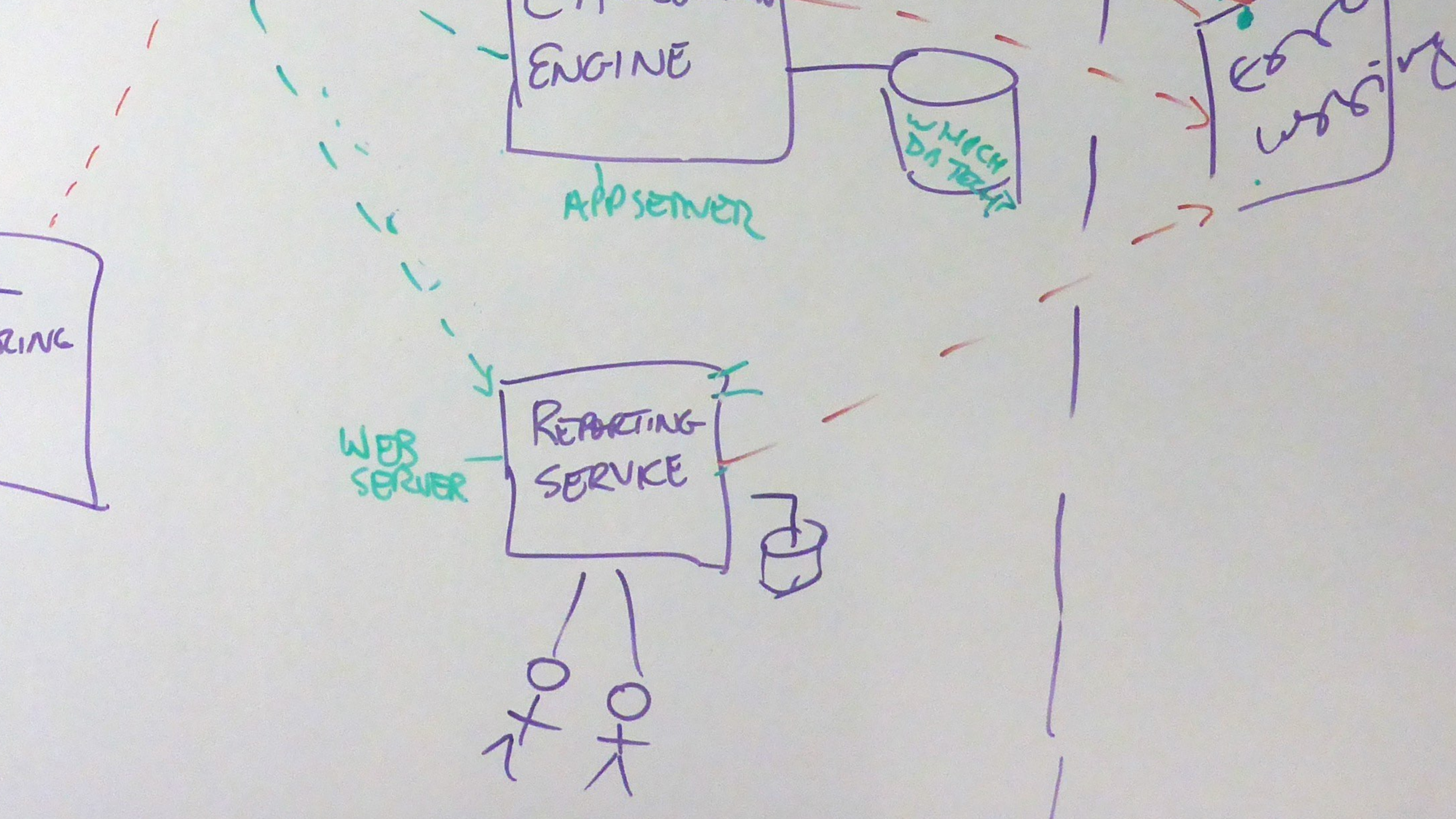

Import
mixed
storage
RISK
USER
CALCULATION
RAW
config
ORM
REPORT
AUDIT
purging


TDS
RDS
trade report
ref repos
R9
TRANSPORT
ARCHIVE
action
data
error
error
AUDIT
BUS. LOGIC
action
error
ERROR
MONITOR
action
REPORT GEN
error
Scheduler
config


2
SERVICE/BATCH
IMPORT
JOIN
CALC
EXPORT/STORE
REPORT
SNMP
CENTRAL
MONITORING
SERVICE
3
WEBAPP
•GET REPORT
•EDIT
RISK RULES
•AUTH
•AUDIT
RISK RULES

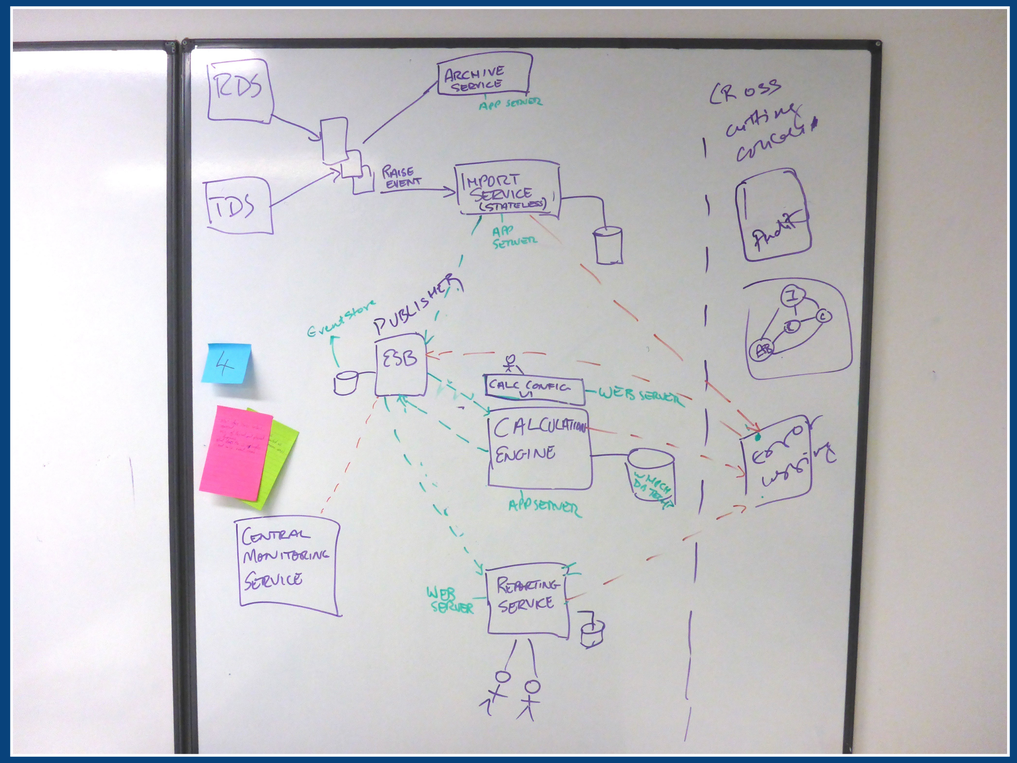
WIN SERVER
APPLICATION
CONSOLE - C#
SCHEDULER
CONSOLE - C#
DATA LOADER
C#
CALCULATION MANAGER
C#
NOTIFICATION MANAGER
C#
EXTERNAL SYSTEMS
SNMP SERVICE
DATA MERGE C#
XML READER C#
RDS
TDS
XML
RISK CALCULATION C#
NOTIFICATION SERVICE C#
SMTP
parameters
LOGGER
nLog
SNMP TRAP GENERATOR
DATA ACCESS
SQL SERVER


TRADES
DATA POLL SERVICE
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
OUTPUT STORE
AUDIT
FAILURE


ENGINE
APP SERVER
RING
WEB SERVER
REPORTING SERVICE

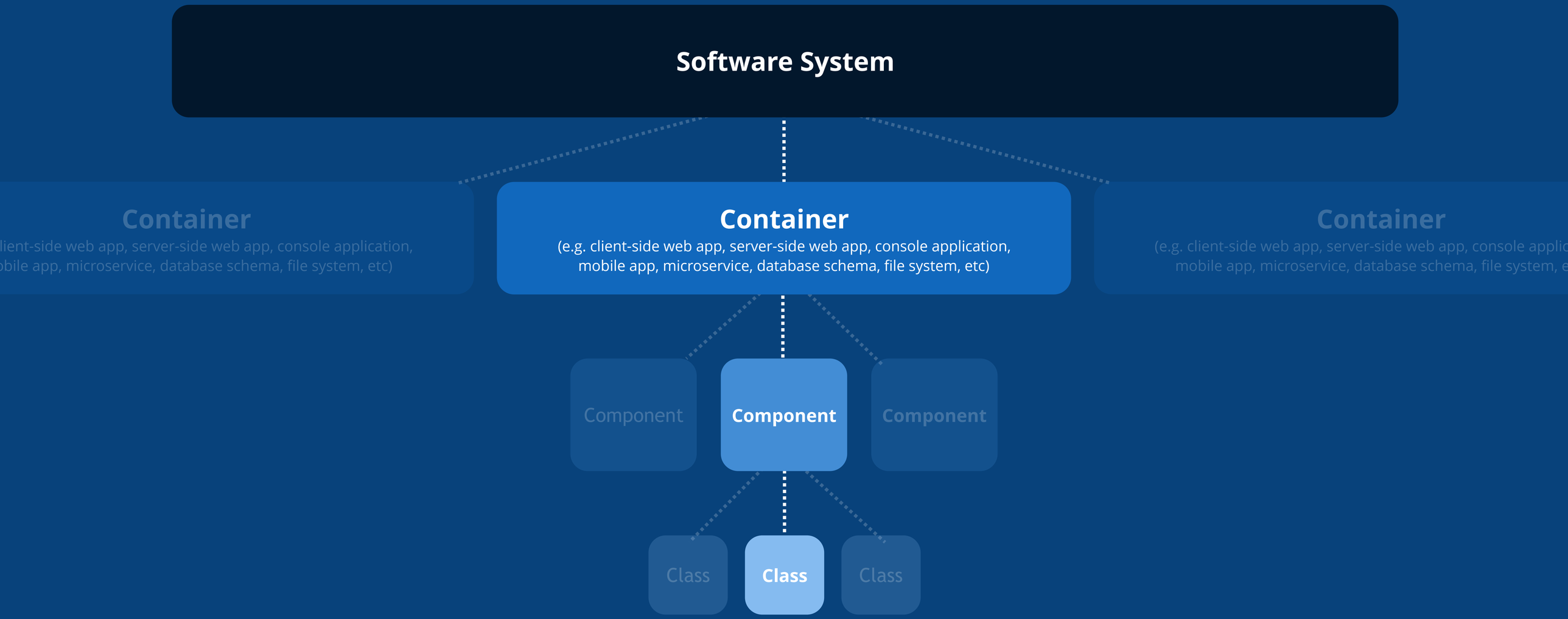

Params
Calcs
EH?
Batch
Date
- RiskCol
- calcs
- Par

1. Is that what we're going to **build**?



2. Is it going to **work**?

A **common set of abstractions**
is more important
than a common notation



A **software system** is made up of one or more **containers**,
each of which contains one or more **components**,
which in turn are implemented by one or more **classes** (or **code**).

C4

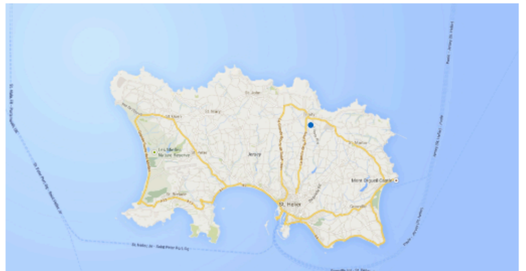
c4model.com

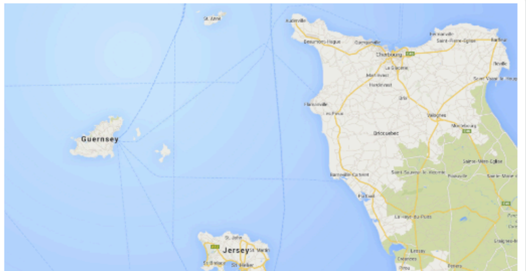
The C4 model for software architecture

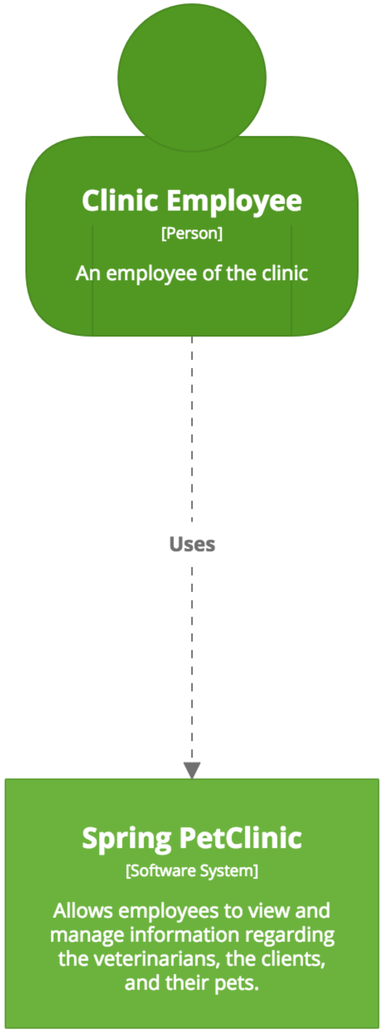
Context, Containers, Components and Classes (or Code)

Maps of your code

The C4 model was created as a way to help software development teams describe and communicate software architecture, both during up-front design sessions and when retrospectively documenting an existing codebase. It's a way to create **maps of your code**, at various levels of detail, in the same way you would use something like Google Maps to zoom in and out of an area you are interested in.



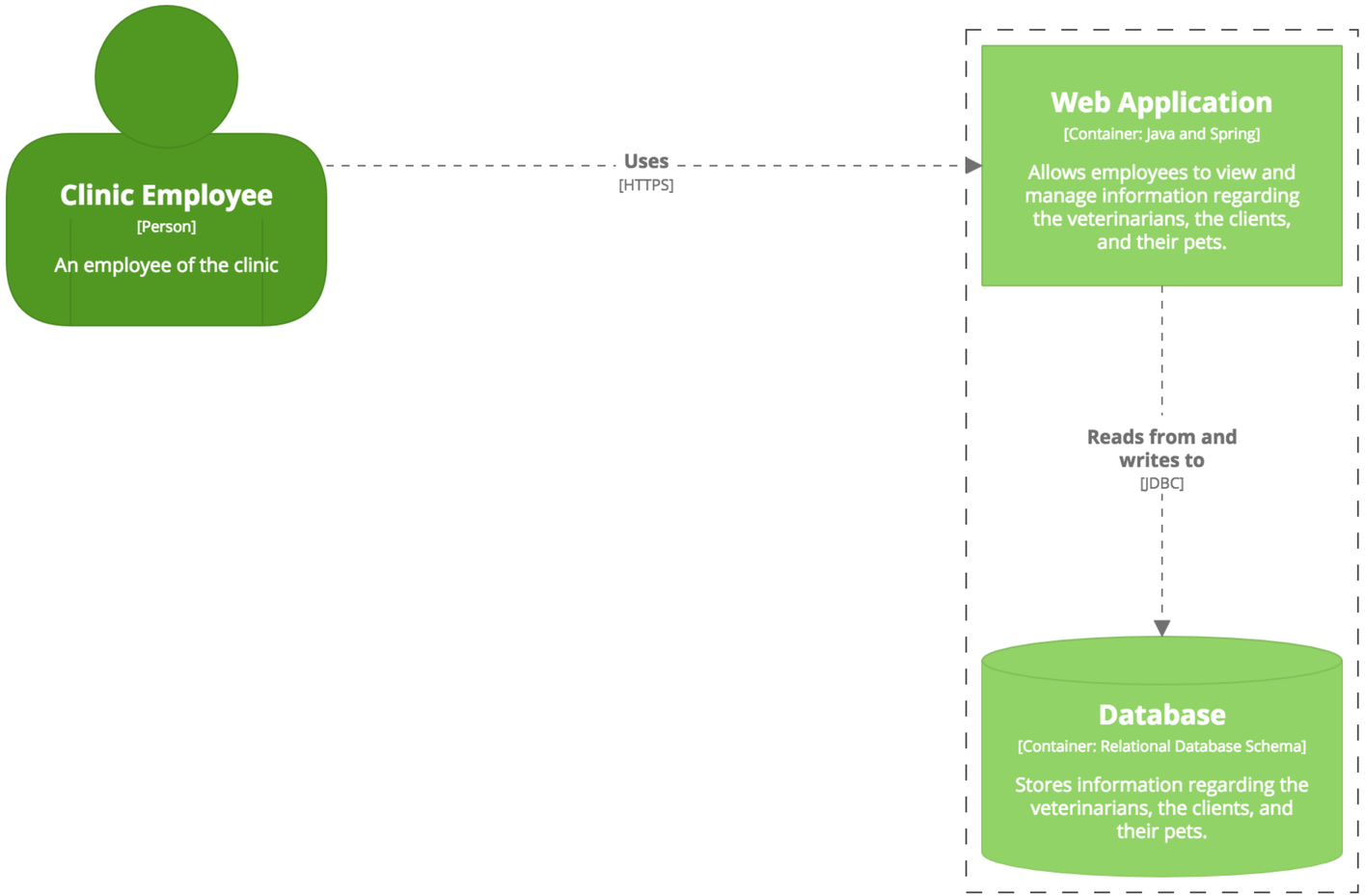




System Context diagram for Spring PetClinic

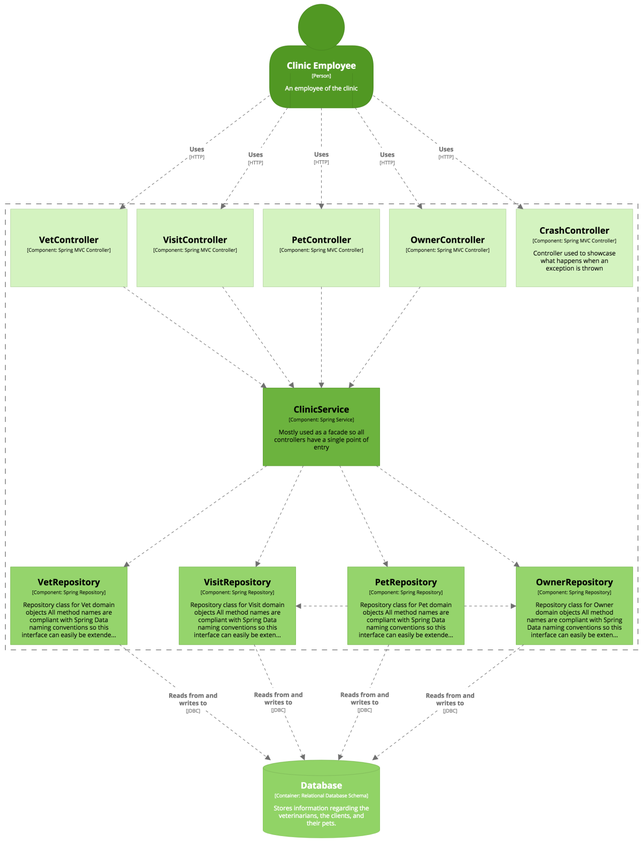
The System Context diagram for the Spring PetClinic system.

Last modified: Monday 26 June 2017 12:59 UTC | Version: 95de1d9f8bf63560915331664b27a4a75ce1f1f6



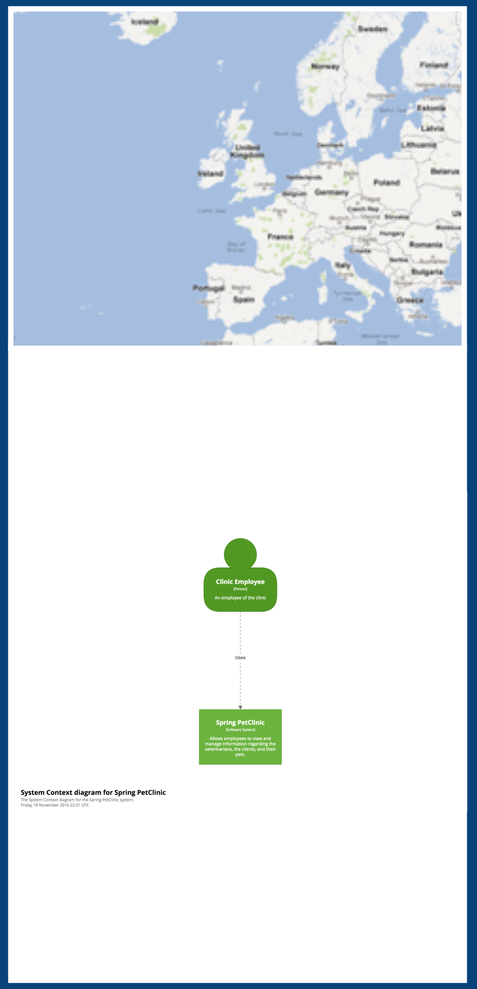
Container diagram for Spring PetClinic

The Containers diagram for the Spring PetClinic system.
Last modified: Monday 26 June 2017 12:59 UTC | Version: 95de1d9f8bf63560915331664b27a4a75ce1f1f6



Component diagram for Spring PetClinic - Web Application

The Components diagram for the Spring PetClinic web application.
Last modified: Monday 26 June 2017 12:59 UTC | Version: 95de1d9f8bf635609153311664b27a4a75ce1f1f6

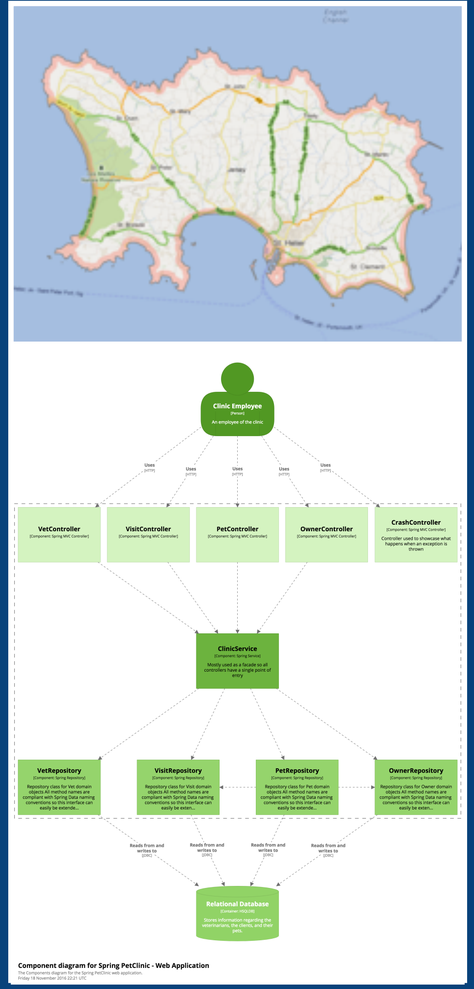


System Context diagram for Spring PetClinic

The System Context diagram for the Spring PetClinic system.
Friday 18 November 2016 22:21 UTC

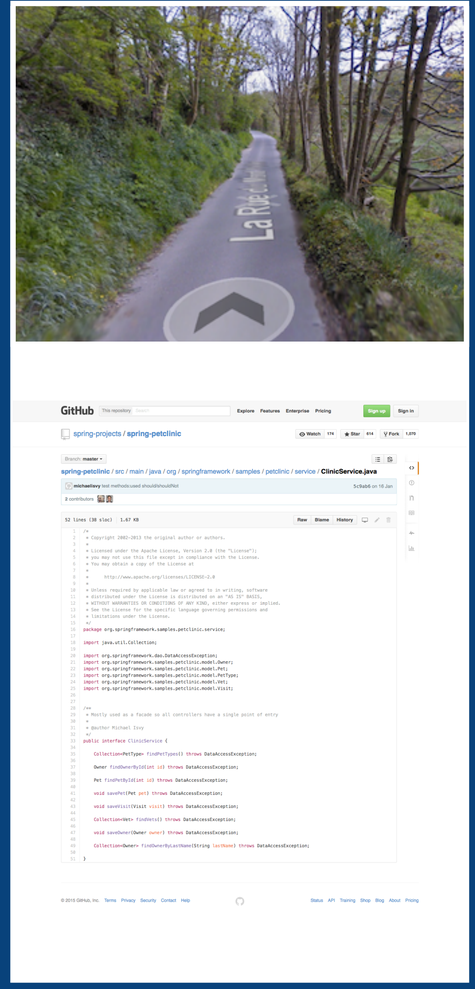
Container diagram for Spring PetClinic

The Containers diagram for the Spring PetClinic system.
Friday 18 November 2016 22:21 UTC



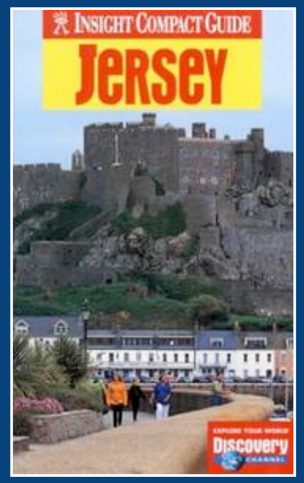
Component diagram for Spring PetClinic - Web Application

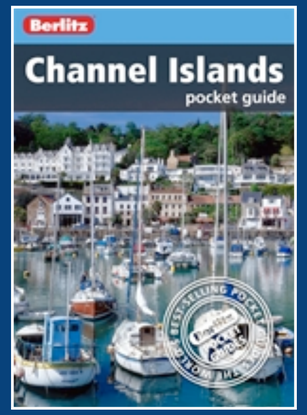
The Components diagram for the Spring PetClinic web application.
Friday 18 November 2016 22:21 UTC

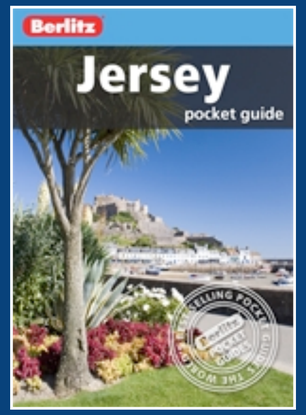


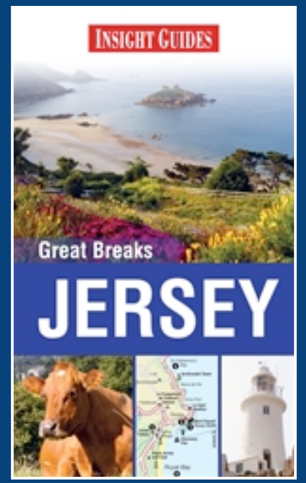
Diagrams are maps

that help software developers navigate a large and/or complex codebase









Software Guidebook

(maps, points of interest, sights, itineraries, history, culture, practical information, etc)

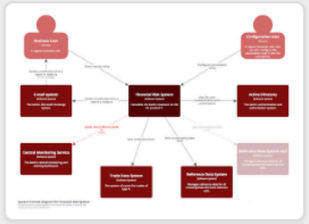
What **tools** do you recommend?

```
public static void main(String[] args) throws Exception {
    Workspace workspace = new Workspace("Getting Started", "This is a model of
    Model model = workspace.getModel();

    Person user = model.addPerson("User", "A user of my software system.");
    SoftwareSystem softwareSystem = model.addSoftwareSystem("Software System",
    user.uses(softwareSystem, "Uses");

    ViewSet views = workspace.getViews();
    SystemContextView contextView = views.createSystemContextView(softwareSyst
    contextView.addAllSoftwareSystems();
    contextView.addAllPeople();

    Styles styles = views.getConfiguration().getStyles();
    styles.addElementStyle(Tags.SOFTWARE_SYSTEM).background("#1168bd").color("
    styles.addElementStyle(Tags.PERSON).background("#08427b").color("#ffffff")
}
```

```
static void Main()
{
    Workspace workspace = new Workspace("Getting Started", "This is a model of
    Model model = workspace.Model;

    Person user = model.AddPerson("User", "A user of my software system.");
    SoftwareSystem softwareSystem = model.AddSoftwareSystem("Software System",
    user.Uses(softwareSystem, "Uses");

    ViewSet viewSet = workspace.Views;
    SystemContextView contextView = viewSet.CreateSystemContextView(softwareSy
    contextView.AddAllSoftwareSystems();
    contextView.AddAllPeople();

    Styles styles = viewSet.Configuration.Styles;
    styles.Add(new ElementStyle(Tags.SoftwareSystem) { Background = "#1168bd",
    styles.Add(new ElementStyle(Tags.Person) { Background = "#08427b", Color =
}
```

Structurizr

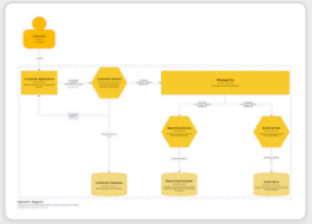
Visualise, document and explore your software architecture



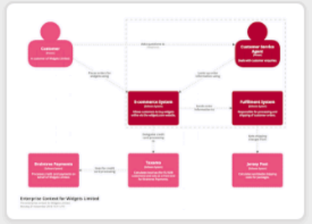
Create system context diagrams using Java code.

Extract components from your Java codebase, using static analysis and reflection.

Client-side encrypt your workspace for additional security ('password').

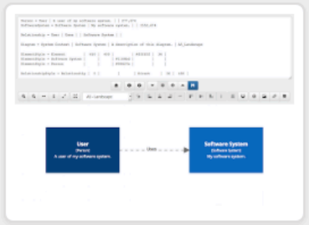


Create animated diagrams to describe dynamic behaviour.

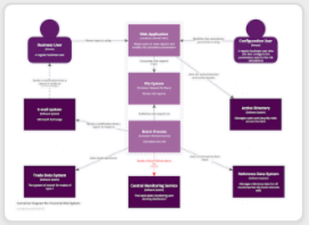


Create enterprise context diagrams using code.

Document your software with Markdown and AsciiDoc, effortlessly embed diagrams.

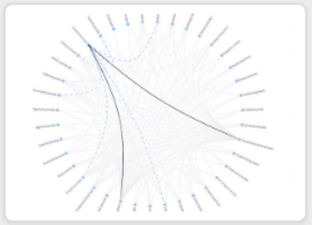


Create diagrams using text with Structurizr Express.

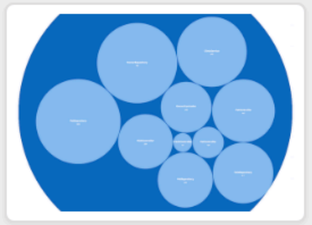


Create diagrams using C# code.

Extract components from your .NET codebase, using static analysis and reflection.

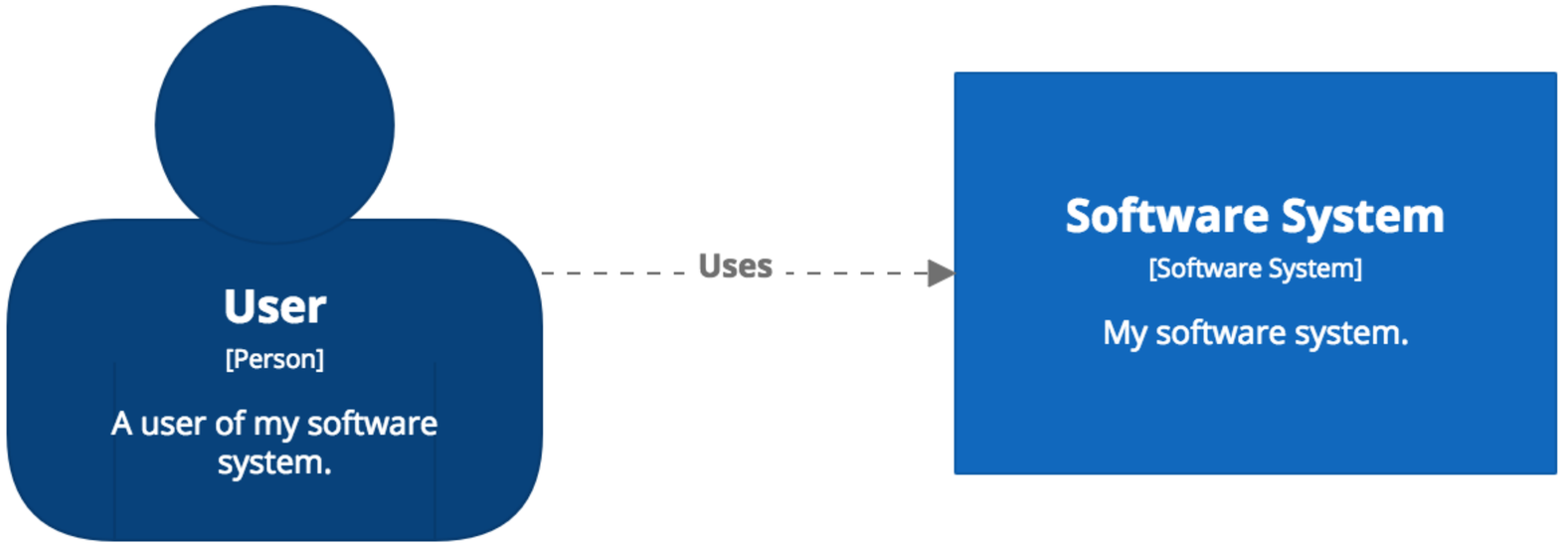


Explore inbound, outbound and cyclic dependencies between components.

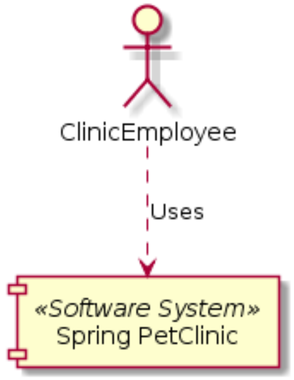


Explore component size.

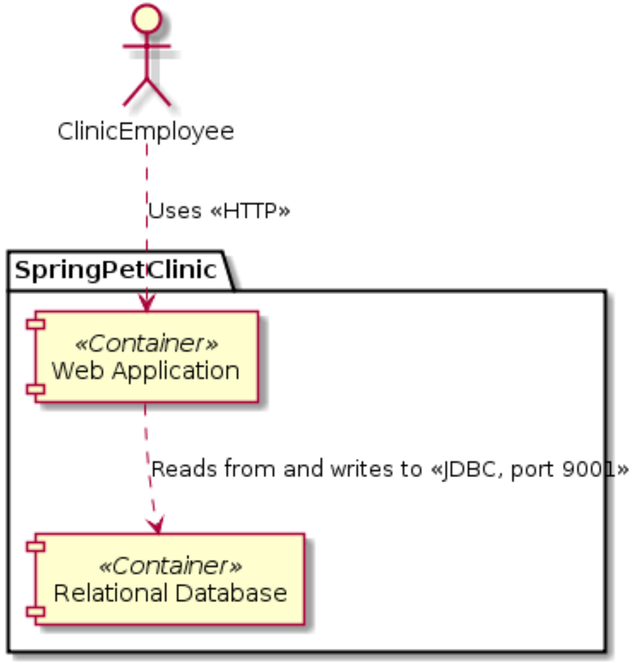
Embed diagrams into Atlassian Confluence with prebuilt macros.


User
[Person]
A user of my software system.
Uses
Software System
[Software System]
My software system.

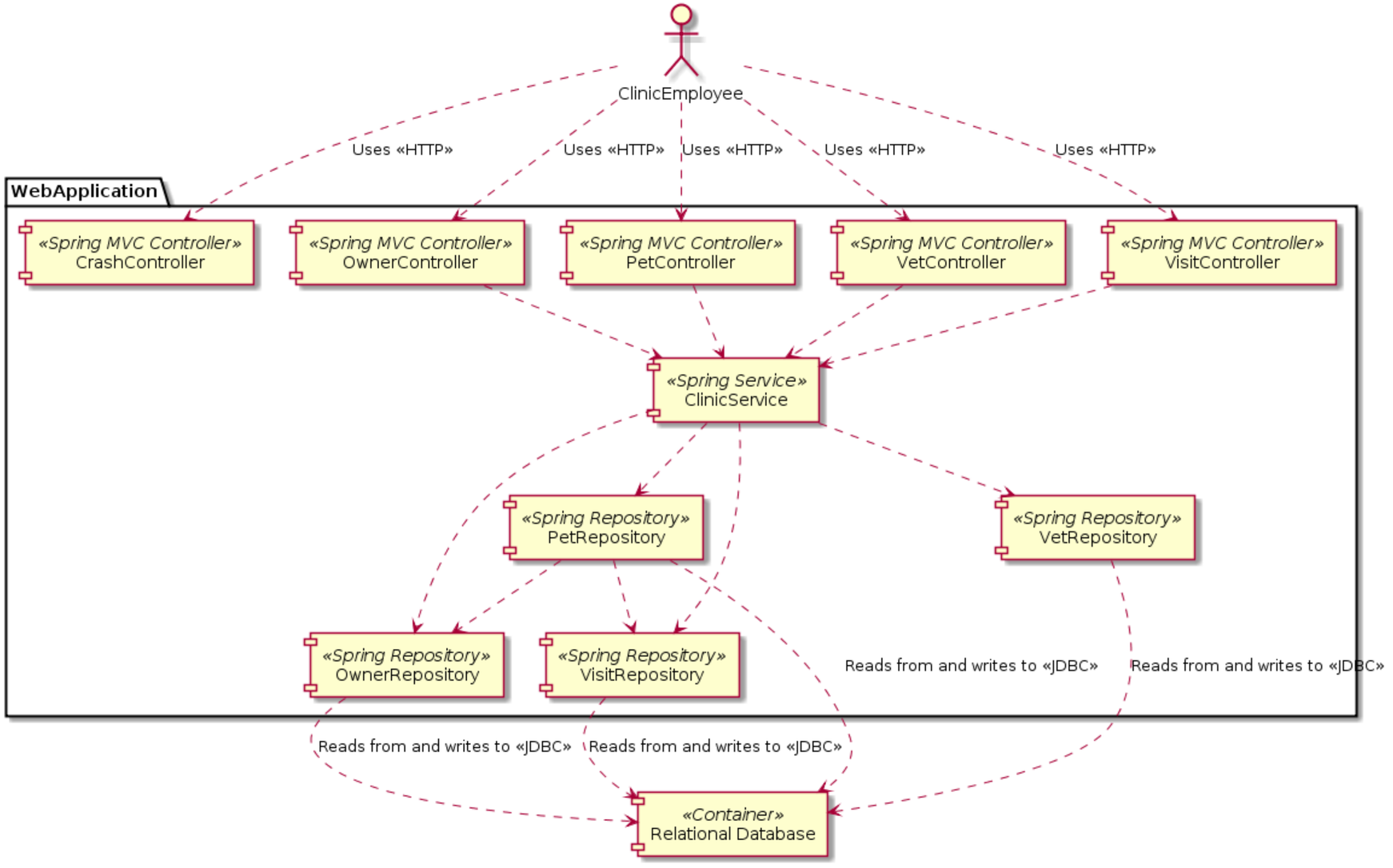
Spring PetClinic - System Context

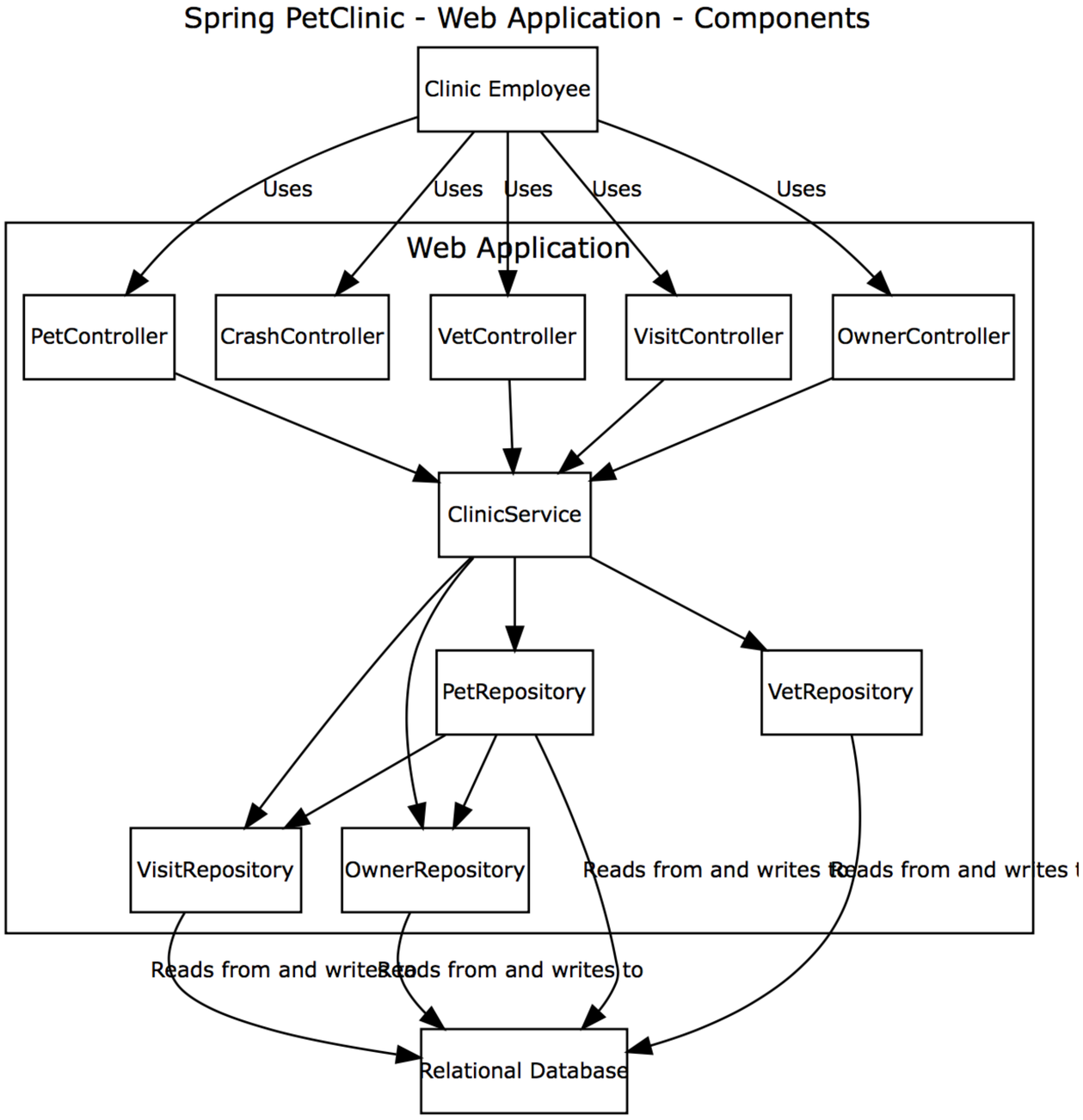


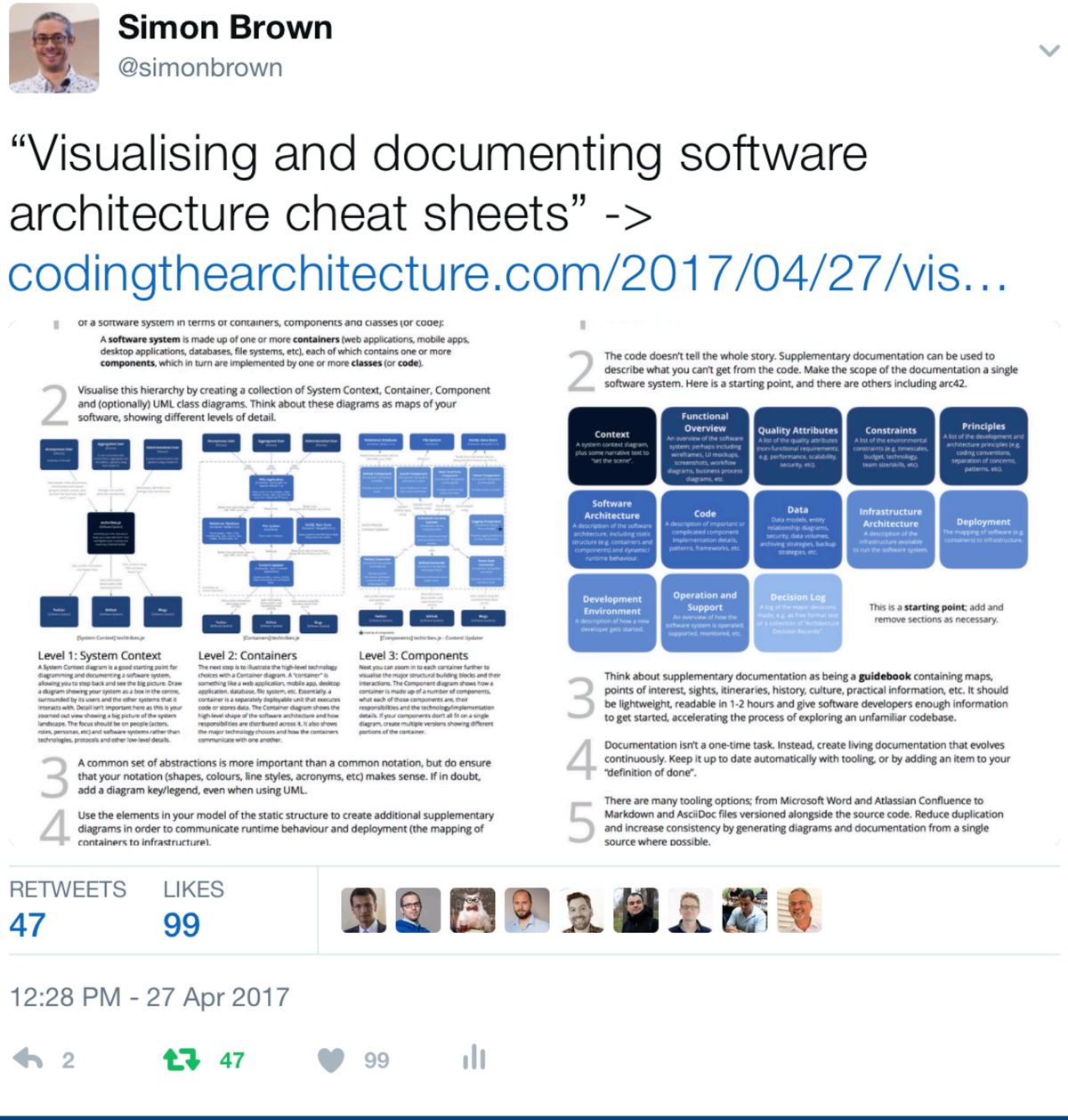
Spring PetClinic - Containers



Spring PetClinic - Web Application - Components




Spring PetClinic - Web Application - Components
Clinic Employee
Uses
Uses
Uses
Uses
Uses
Web Application
PetController
CrashController
VetController
VisitController
OwnerController
ClinicService
PetRepository
VetRepository
VisitRepository
OwnerRepository
Reads from and writes to
Reads from and writes to
Reads from and writes to
Reads from and writes to
Relational Database



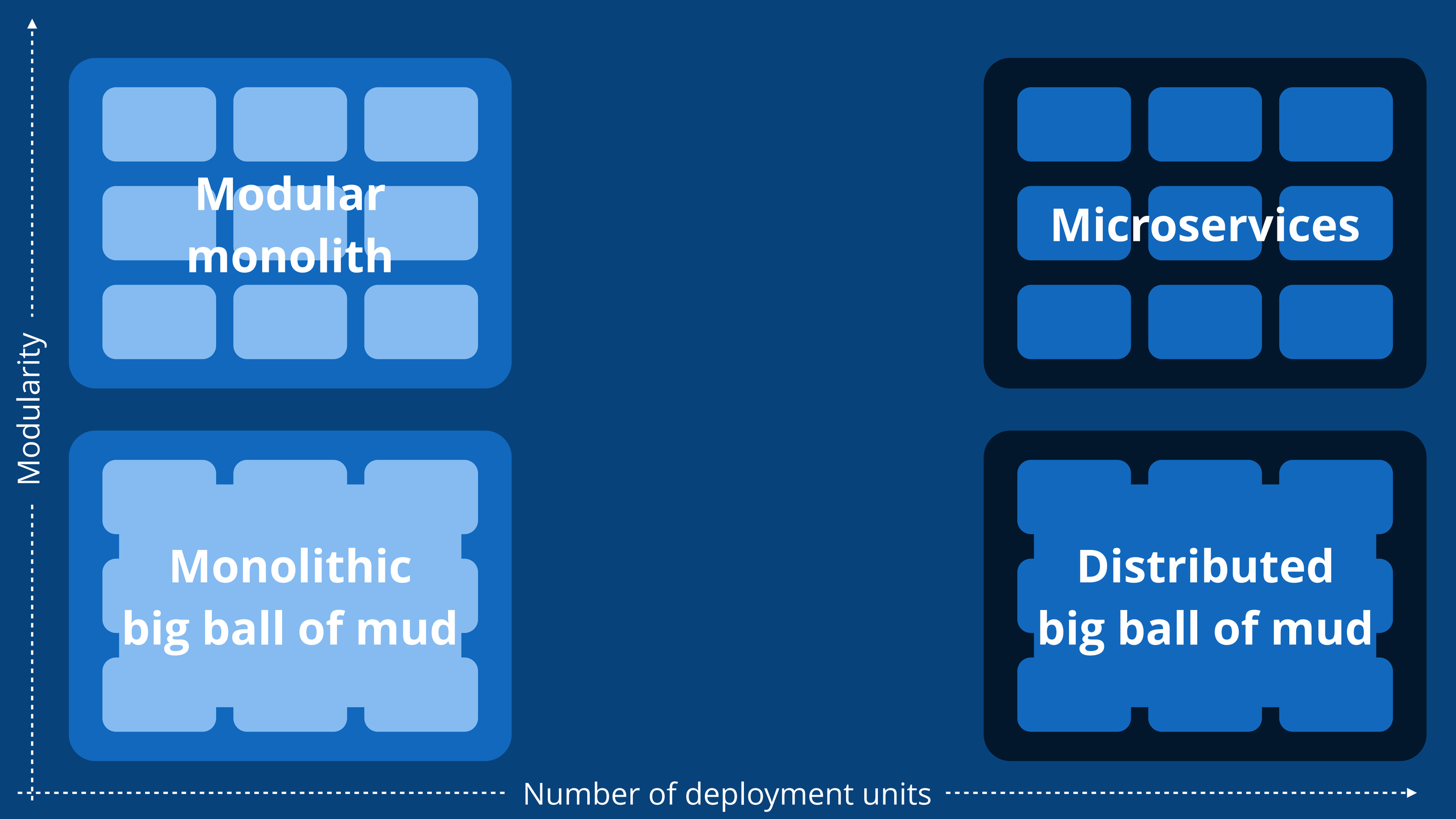
For more information about software architecture diagrams and documentation...

5. A good software architecture enables agility

Agile is about moving fast, embracing change, releasing often, getting feedback, ...

Agile is about a mindset of
continuous improvement

A good architecture enables agility


Modularity
Modular monolith
Microservices
Monolithic big ball of mud
Distributed big ball of mud
Number of deployment units

Agility is a

quality attribute

A good architecture rarely happens through **architecture-indifferent design**

Simon Brown

@simonbrown

I'll keep saying this ... if people can't build monoliths properly, microservices won't help. #qconlondon #DesignThinking #Modularity

Retweets	Likes
258	109

10:49 am - 4 Mar 2015

Architect Clippy

@architectclippy

I see you have a poorly structured monolith. Would you like me to convert it into a poorly structured set of microservices?

RETWEETS 4,441

LIKES 2,743

12:59 AM - 24 Feb 2015

Five things every developer should know about software architecture

1. Software architecture isn't about big design up front
2. Every software team needs to consider software architecture
3. The software architecture role is about coding, coaching and collaboration
4. You don't need to use UML
5. A good software architecture enables agility

simon.brown@codingthearchitecture.com
@simonbrown