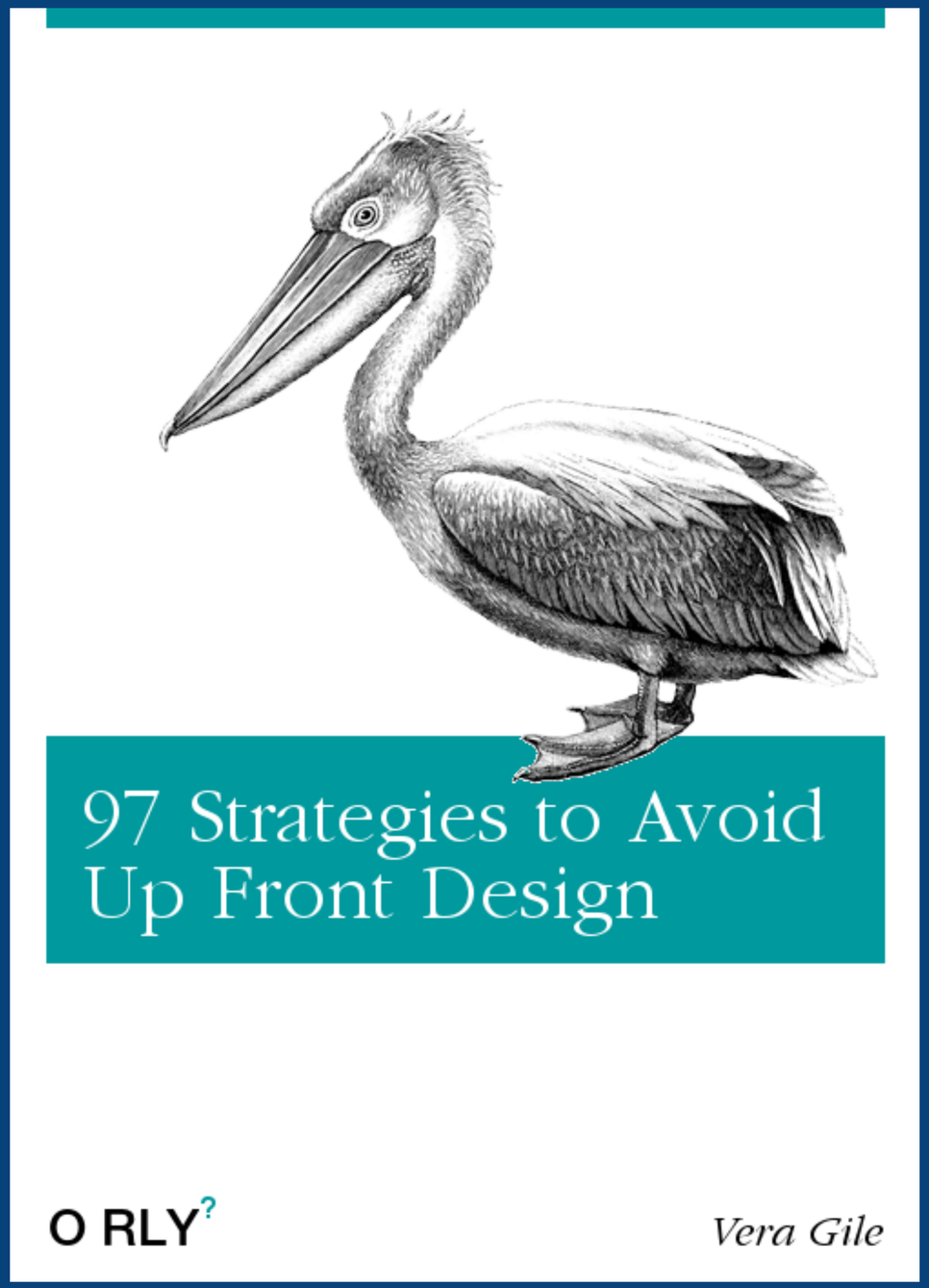
The lost art of software modelling

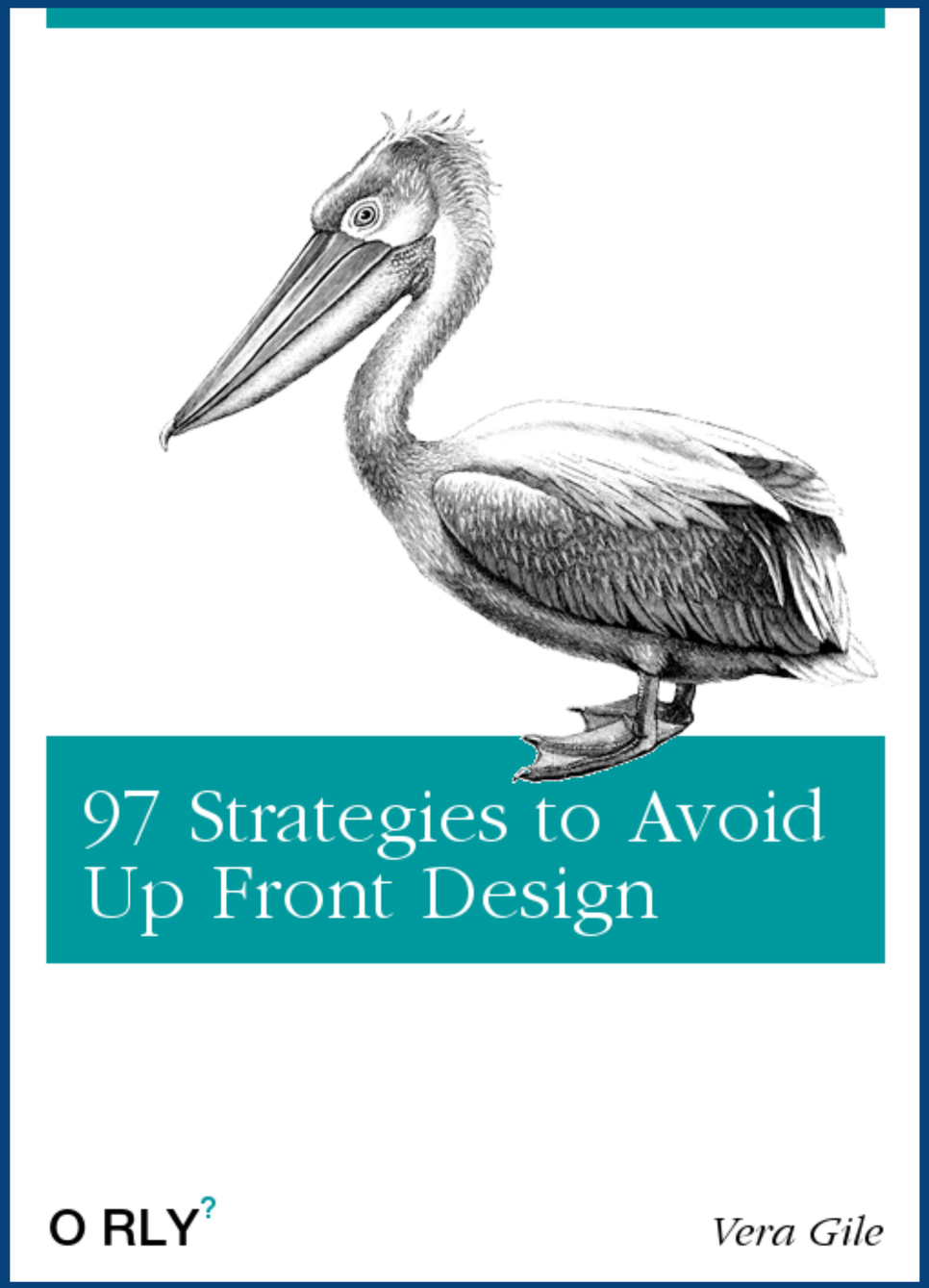
Simon Brown
@simonbrown

Over the past decade, many teams have thrown away big design up front



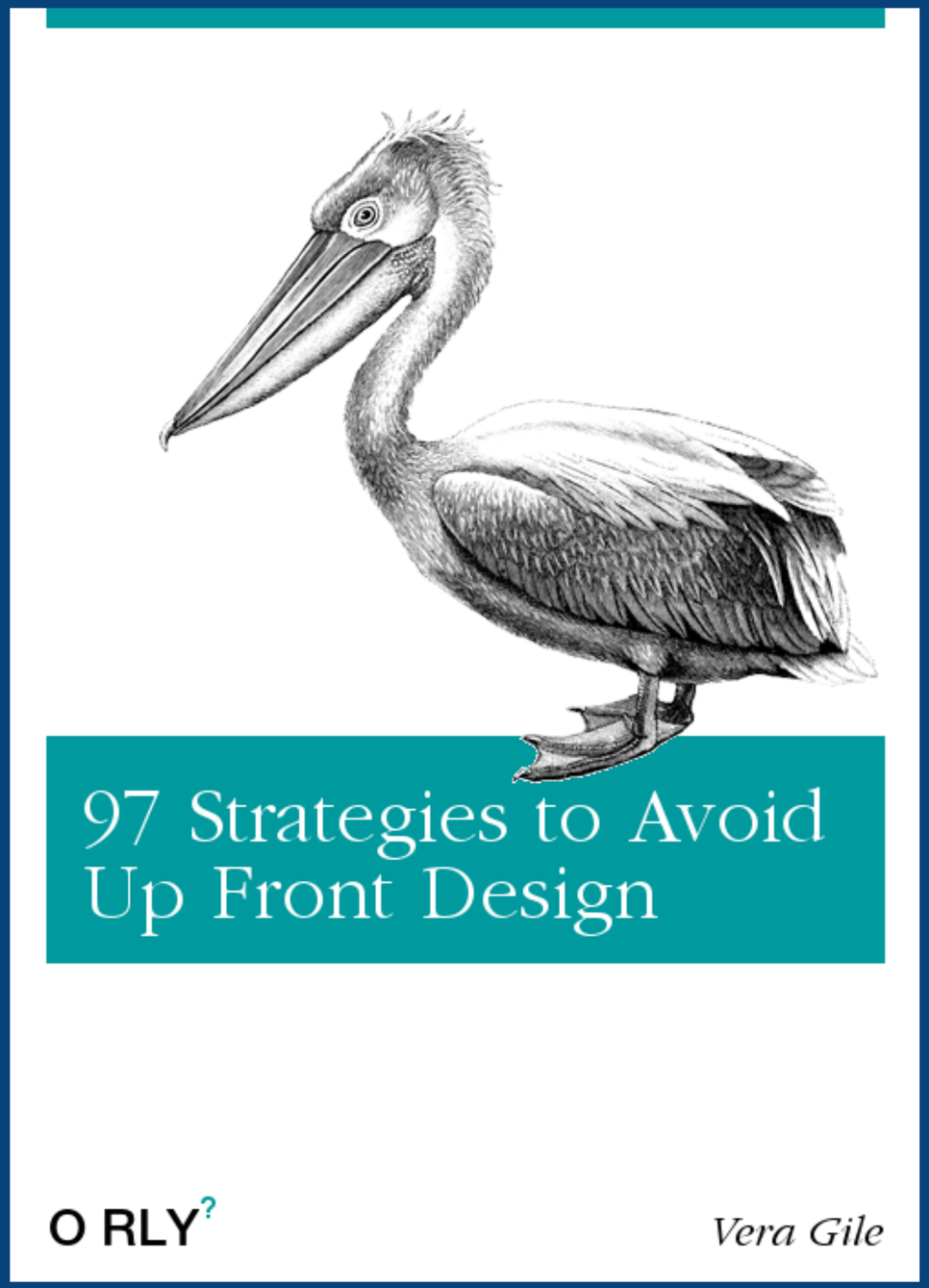
#13

“We’re agile.”



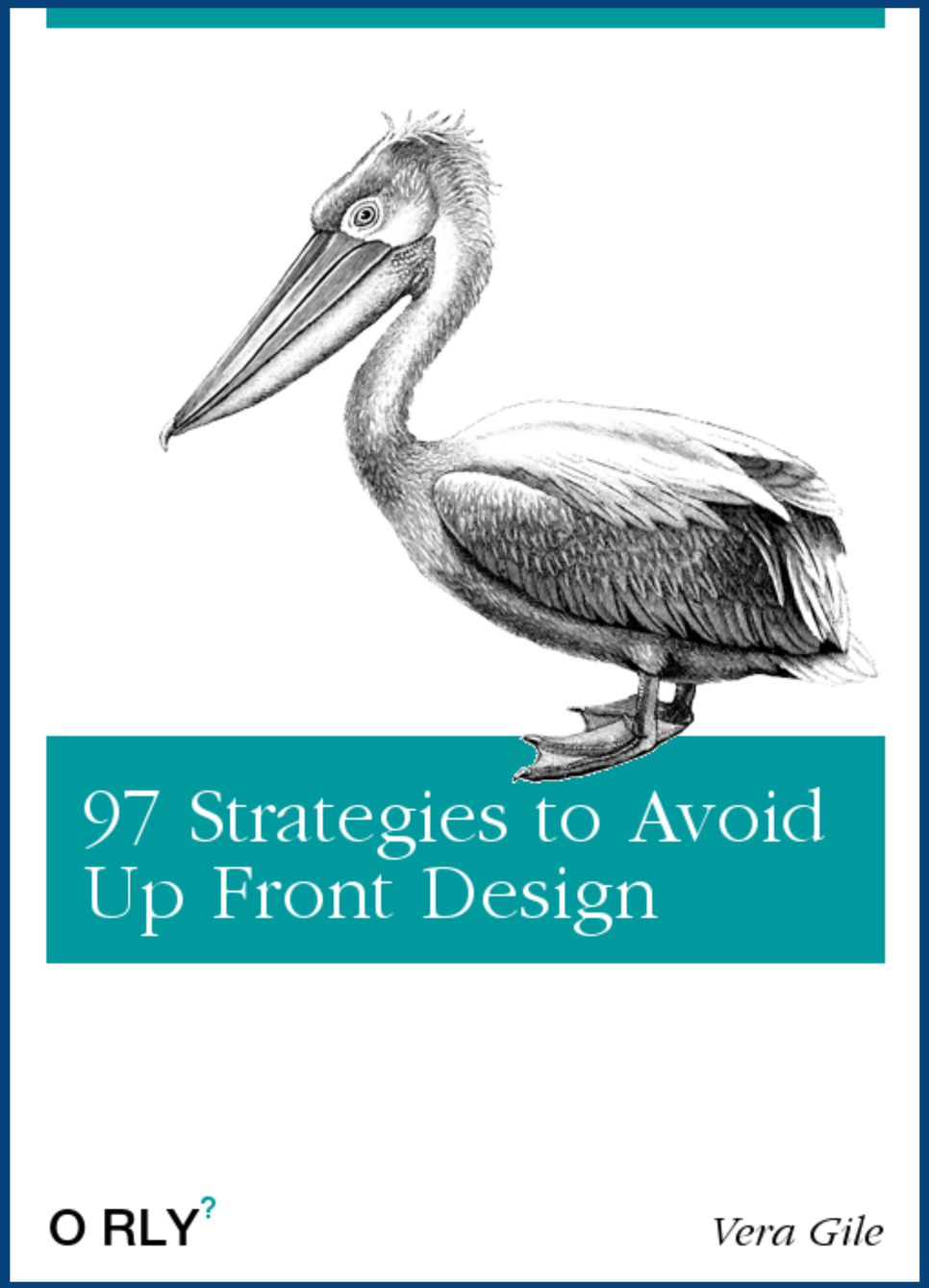
#17

“It’s not expected in agile.”



#1

“Are we allowed to do up front design?”



#12

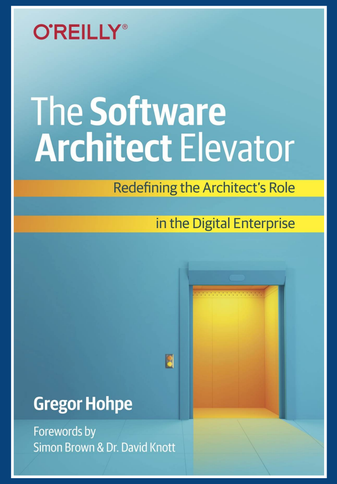
“We don't do up front design because we do XP.”

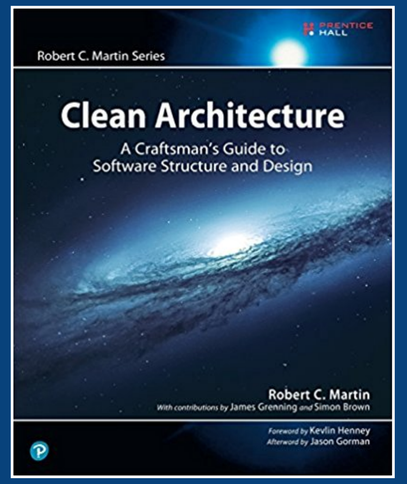
Unfortunately, architectural thinking, documentation, diagramming, and modelling were also often discarded

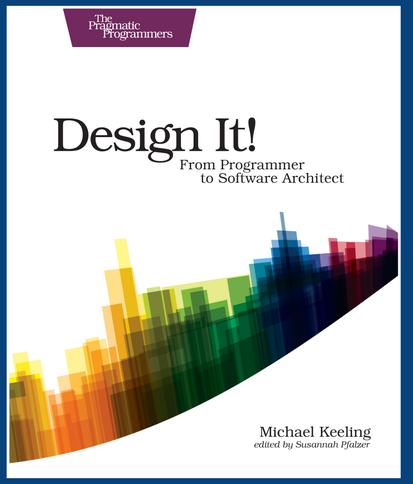
Simon Brown

Independent consultant specialising in software architecture,
plus the creator of the C4 model and Structurizr

@simonbrown







Financial Risk System

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 of trade data to a network share at the close of business at 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 stores all of the reference data needed by the bank. This includes information about counterparties (other banks). A file-based XML export is also generated to a network share at 5pm in New York, and it includes some basic information about each counterparty. A new reference data system is due for completion in the next 3 months, and the current system will eventually be decommissioned. The current data export includes:

- Counterparty ID, Name, Address, etc...

2. Functional Requirements

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.

3. Non-functional Requirements

a. Performance

- Risk reports must be generated before 9am the following business day in Singapore.

b. Scalability

- The system must be able to cope with trade volumes for the next 5 years.
  - The Trade Data System export includes approximately 5000 trades now and it is anticipated that there will be slow but steady growth of 10 additional trades per day.
  - The Reference Data System export includes approximately 20,000 counterparties and growth will be negligible.
- There are 40-50 business users around the world that need access to the report.

c. Availability

- Risk reports should be available to users 24x7, but a small amount of downtime (less than 30 minutes per day) can be tolerated.

d. Failover

- Manual failover is sufficient, provided that the availability targets can be met.

e. Security

- This system must follow bank policy that states system access is restricted to authenticated and authorised users only.
- Reports must only be distributed to authorised users.
- Only a subset of the authorised users are permitted to modify the parameters used in the risk calculations.
- Although desirable, there are no single sign-on requirements (e.g. integration with Active Directory, LDAP, etc).
- All access to the system and reports will be within the confines of the bank’s global network.

f. Audit

- The following events must be recorded in the system audit logs:
  - Report generation.
  - Modification of risk calculation parameters.

g. Fault Tolerance and Resilience

- The system should take appropriate steps to recover from an error if possible, but all errors should be logged.
- Errors preventing a counterparty risk calculation being completed should be logged and the process should continue.

h. Internationalization and Localization

- All user interfaces will be presented in English only.
- All reports will be presented in English only.
- All trading values and risk figures will be presented in US dollars only.

i. Monitoring and Management

- A Simple Network Management Protocol (SNMP) trap should be sent to the bank’s Central Monitoring Service in the following circumstances:
  - When there is a fatal error with the system.
  - When reports have not been generated before 9am Singapore time.

j. Data Retention and Archiving

- Input files used in the risk calculation process must be retained for 1 year.

k. Interoperability

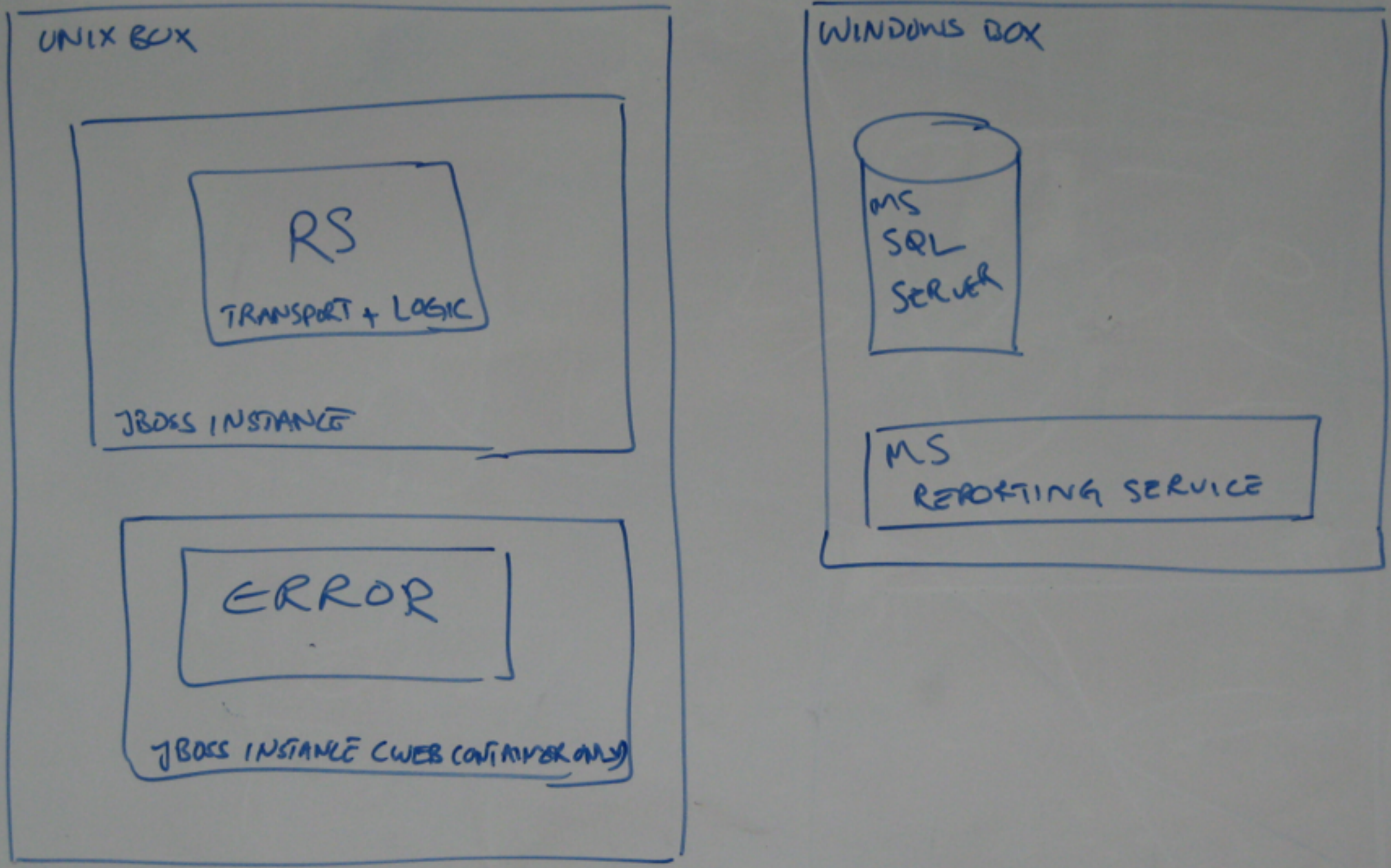
- Interfaces with existing data systems should conform to and use existing data formats.

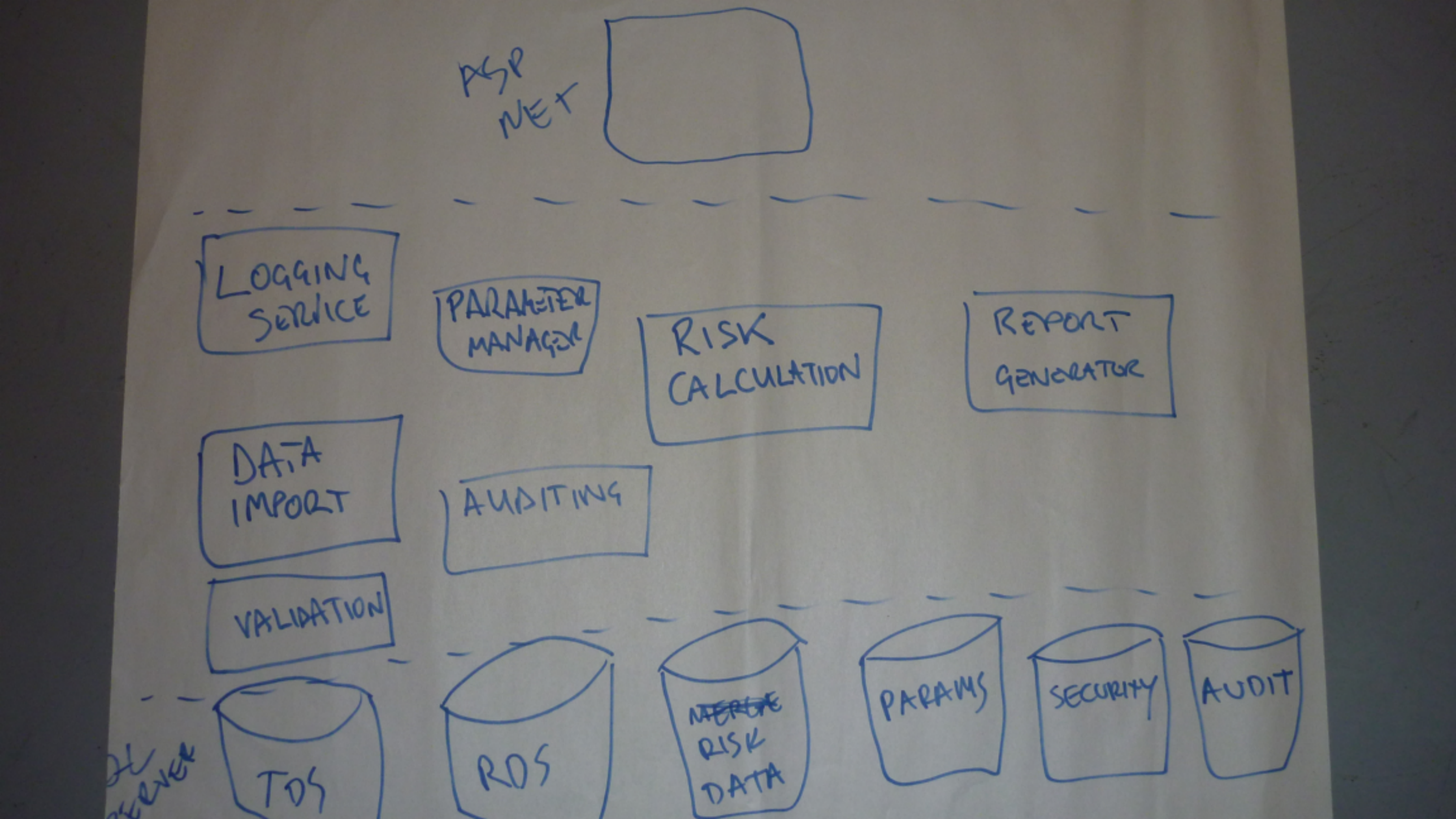
“Financial Risk System” architecture kata
Simon Brown | @simonbrown

CC BY NC SA

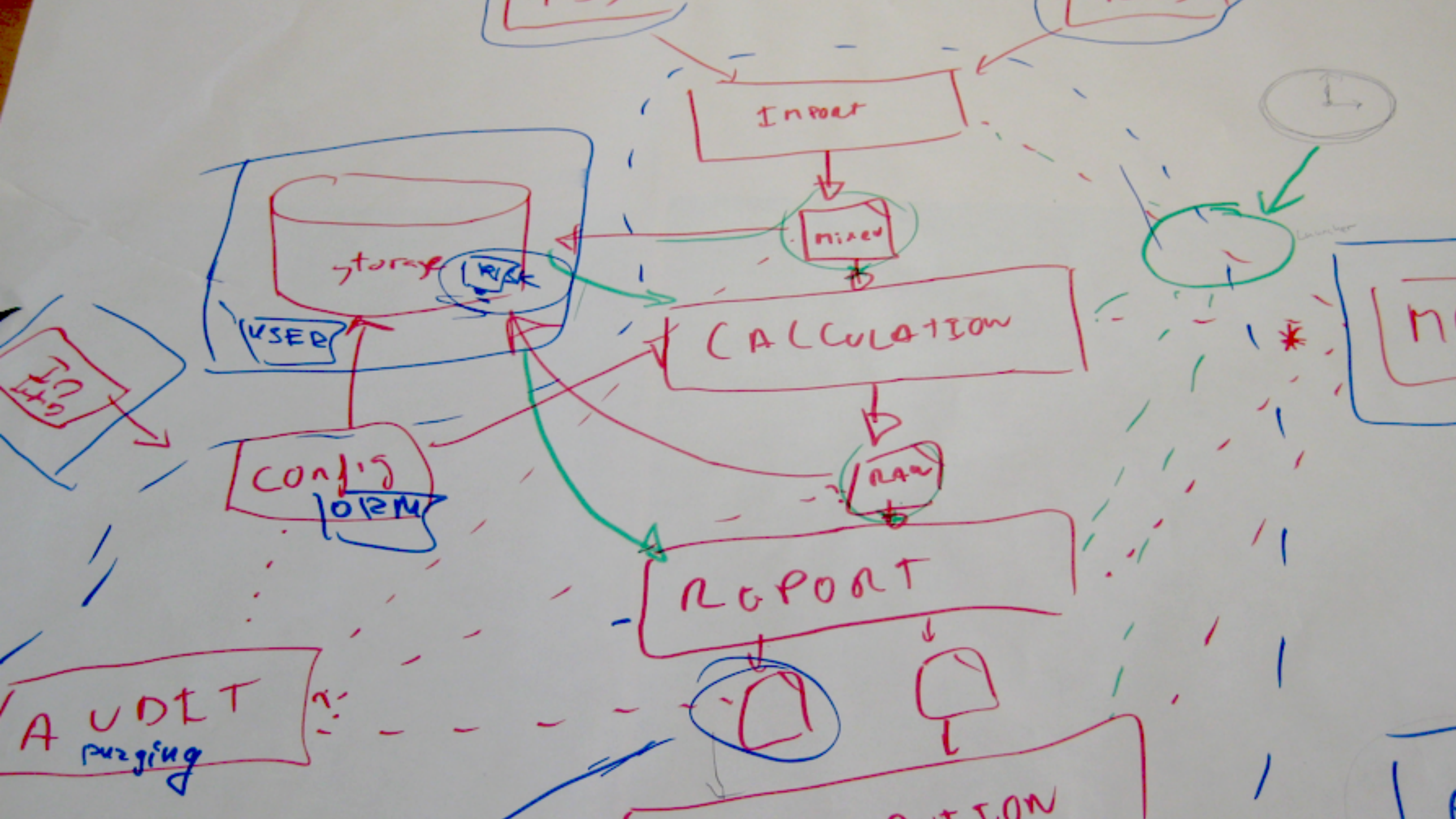
Design a software solution for the ”Financial Risk System”, and **draw** one or more architecture diagrams to describe your solution

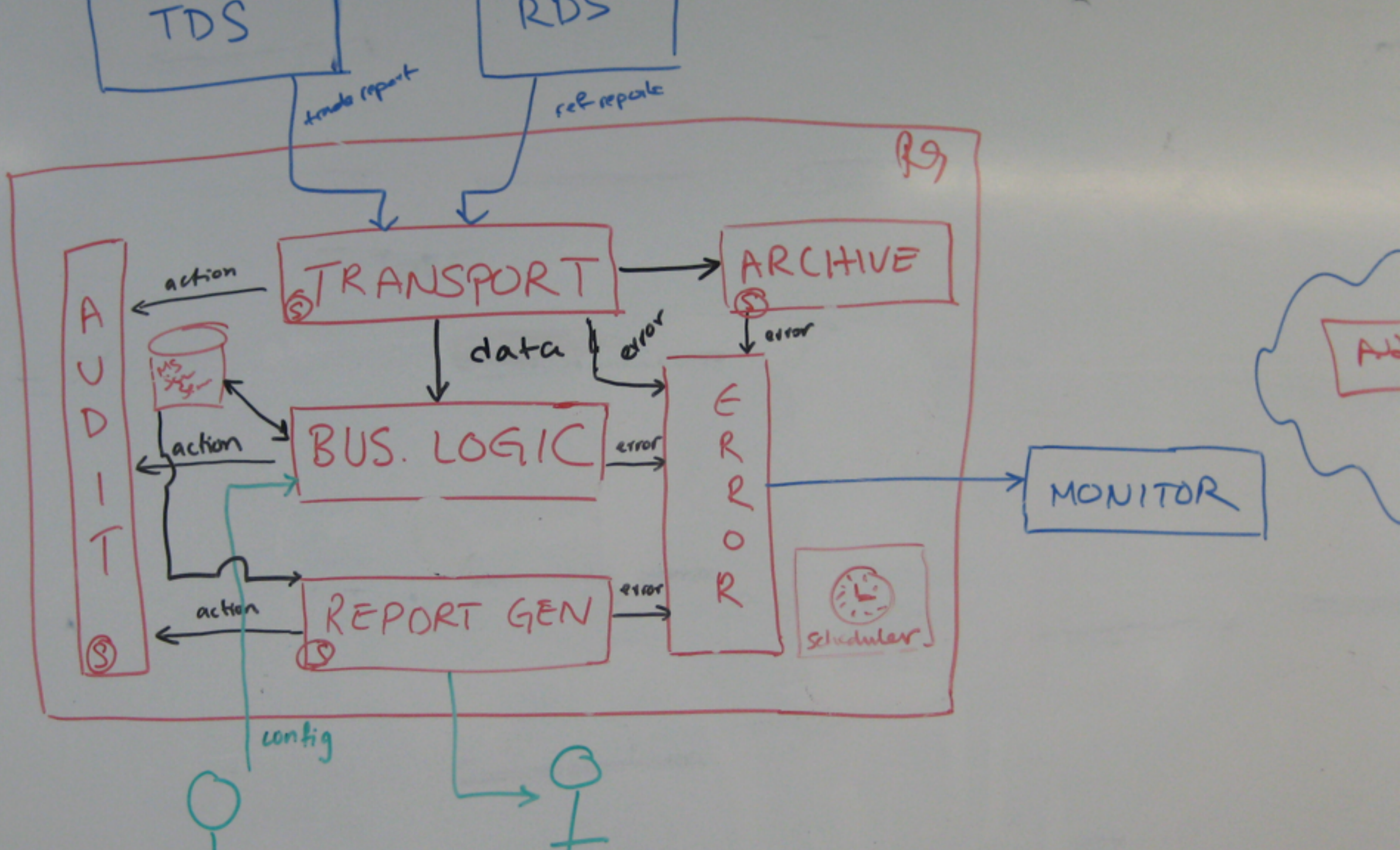
60-90 minutes

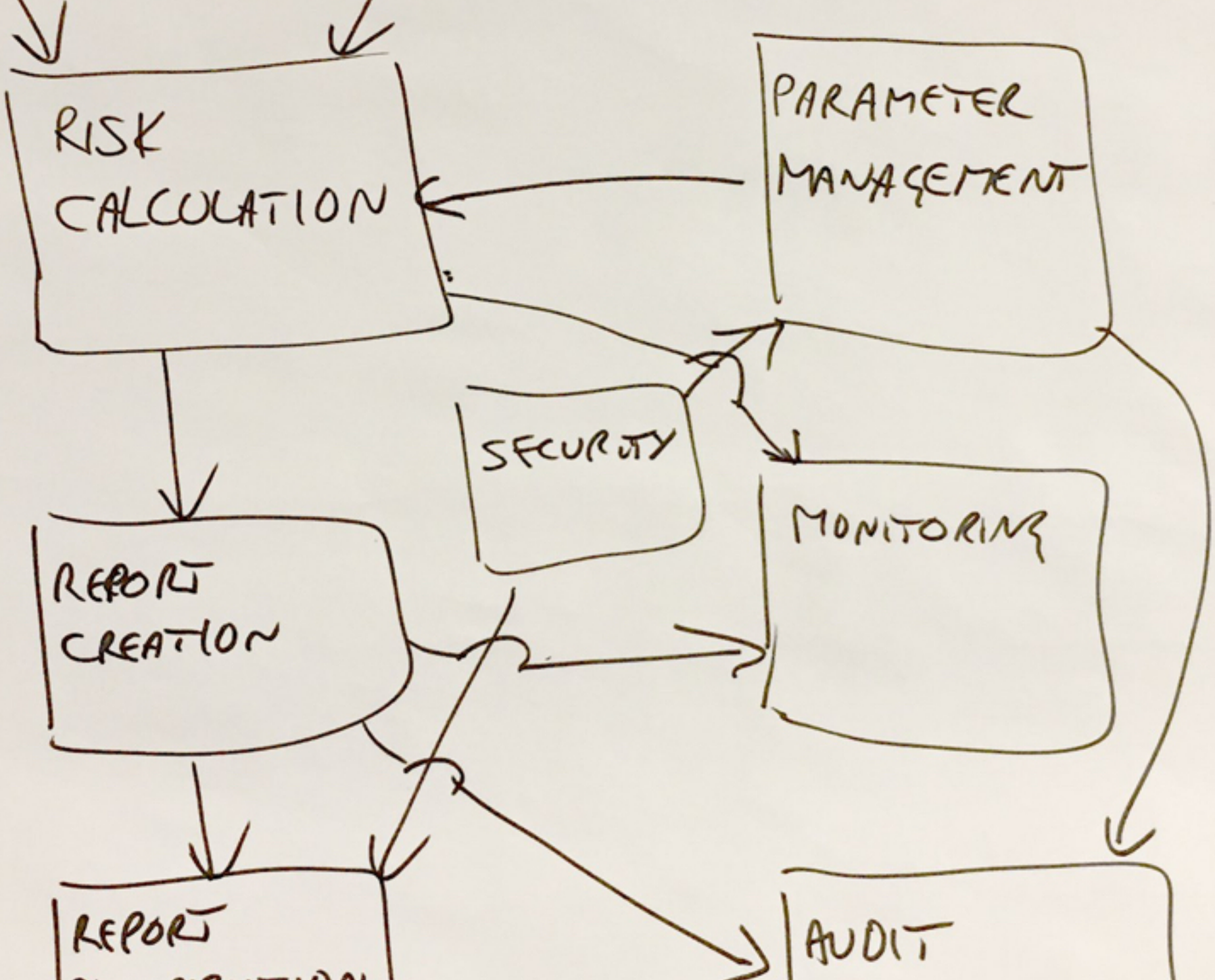

UNIX BOX
RS
TRANSPORT + LOGIC
JBOSS INSTANCE
ERROR
JBOSS INSTANCE (WEB CONTAINER ONLY)
WINDOWS BOX
MS
SQL
SERVER
MS
REPORTING SERVICE

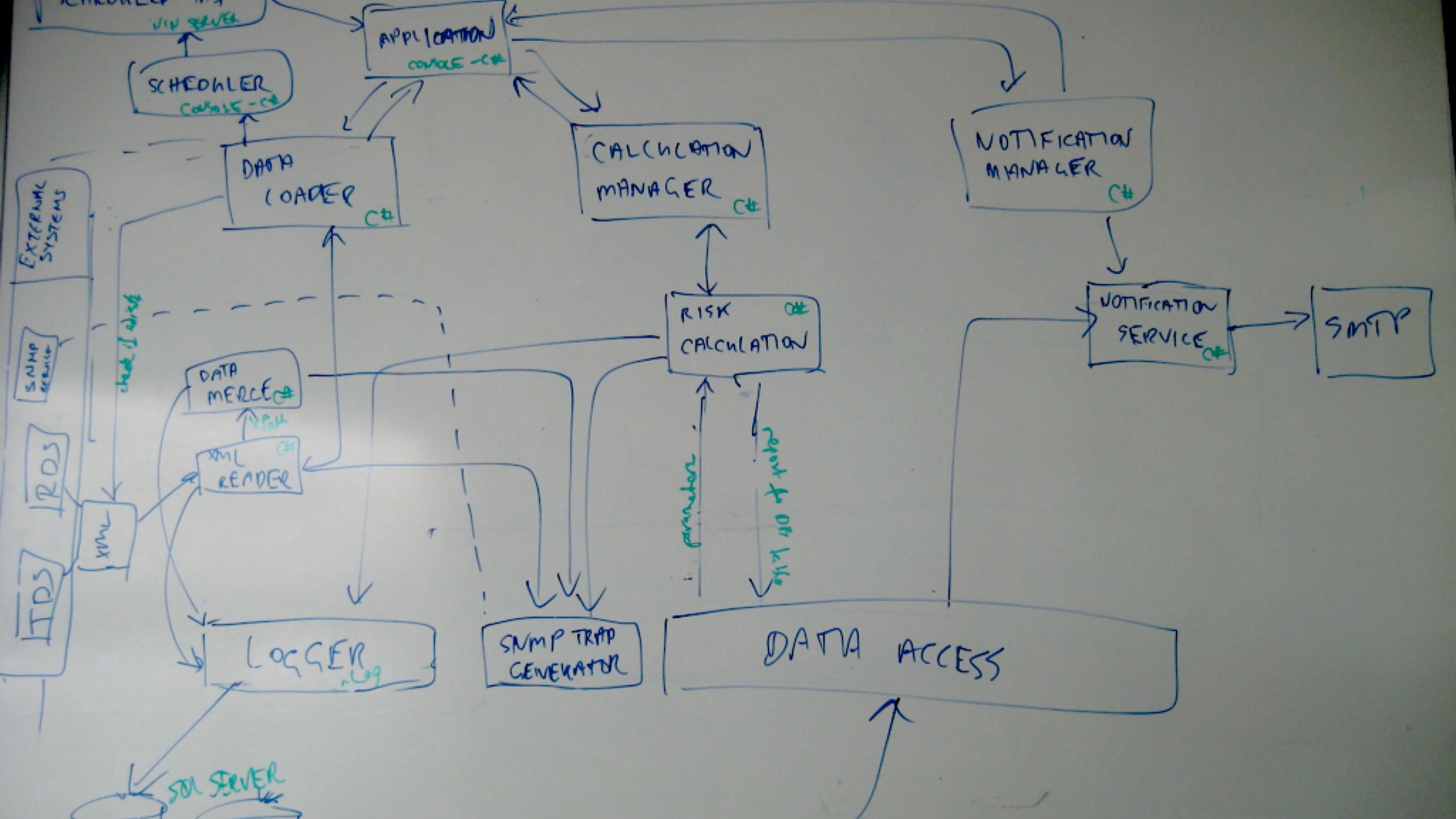

ASP NET
LOGGING SERVICE
PARAMETER MANAGER
RISK CALCULATION
REPORT GENERATOR
DATA IMPORT
AUDITING
VALIDATION
TDS
RDS
RISK DATA
PARAMS
SECURITY
AUDIT

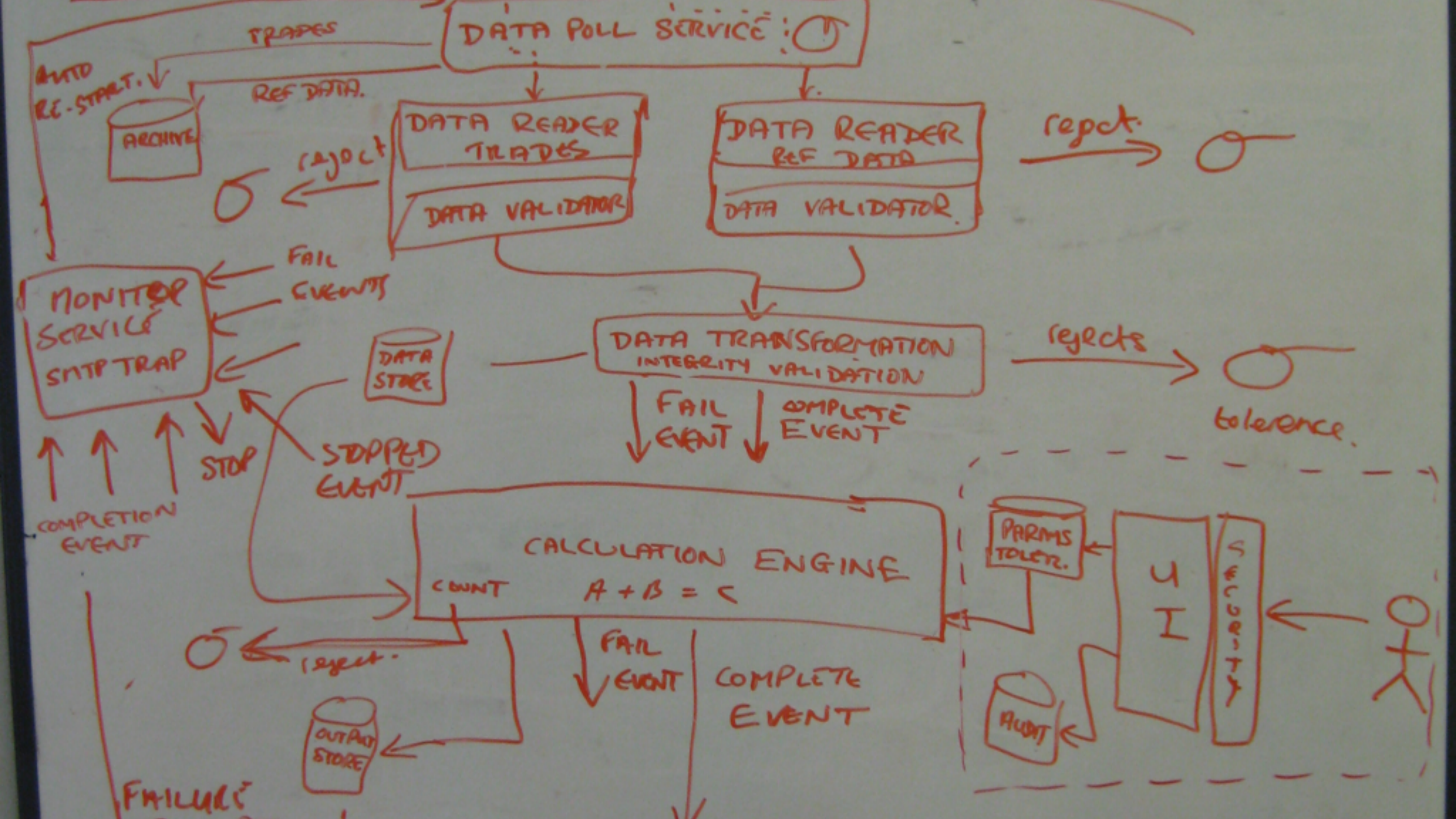
FUNCTIONAL VIEW

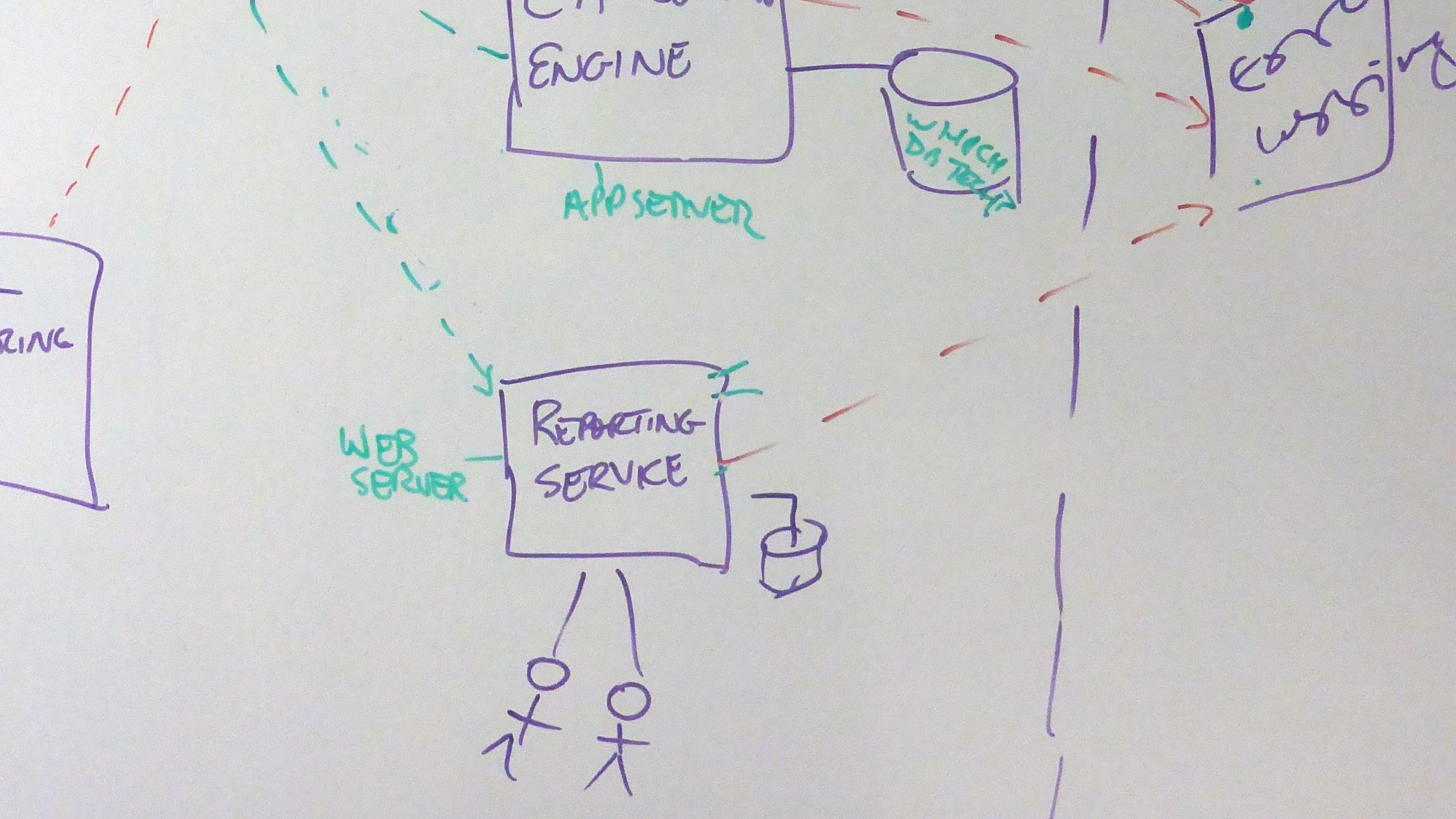

Import
storage
risk
USER
config
mixed
CALCULATION
plan
REPORT
AUDIT
purging


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


RISK
CALCULATION
PARAMETER
MANAGEMENT
SECURITY
MONITORING
REPORT
CREATION
AUDIT


APPLICATION
console - C#
SCHEDULER
console - C#
DATA LOADER
C#
CALCULATION MANAGER
C#
NOTIFICATION MANAGER
C#
EXTERNAL SYSTEMS
SNMP
IRO3
ITD3
check if exist
DATA MERGE
C#
XPath
XML READER
C#
XML
RISK CALCULATION
C#
parameters
save to DB
NOTIFICATION SERVICE
C#
SMTP
LOGGER
.log
SNMP TRAP GENERATOR
DATA ACCESS
SQL SERVER

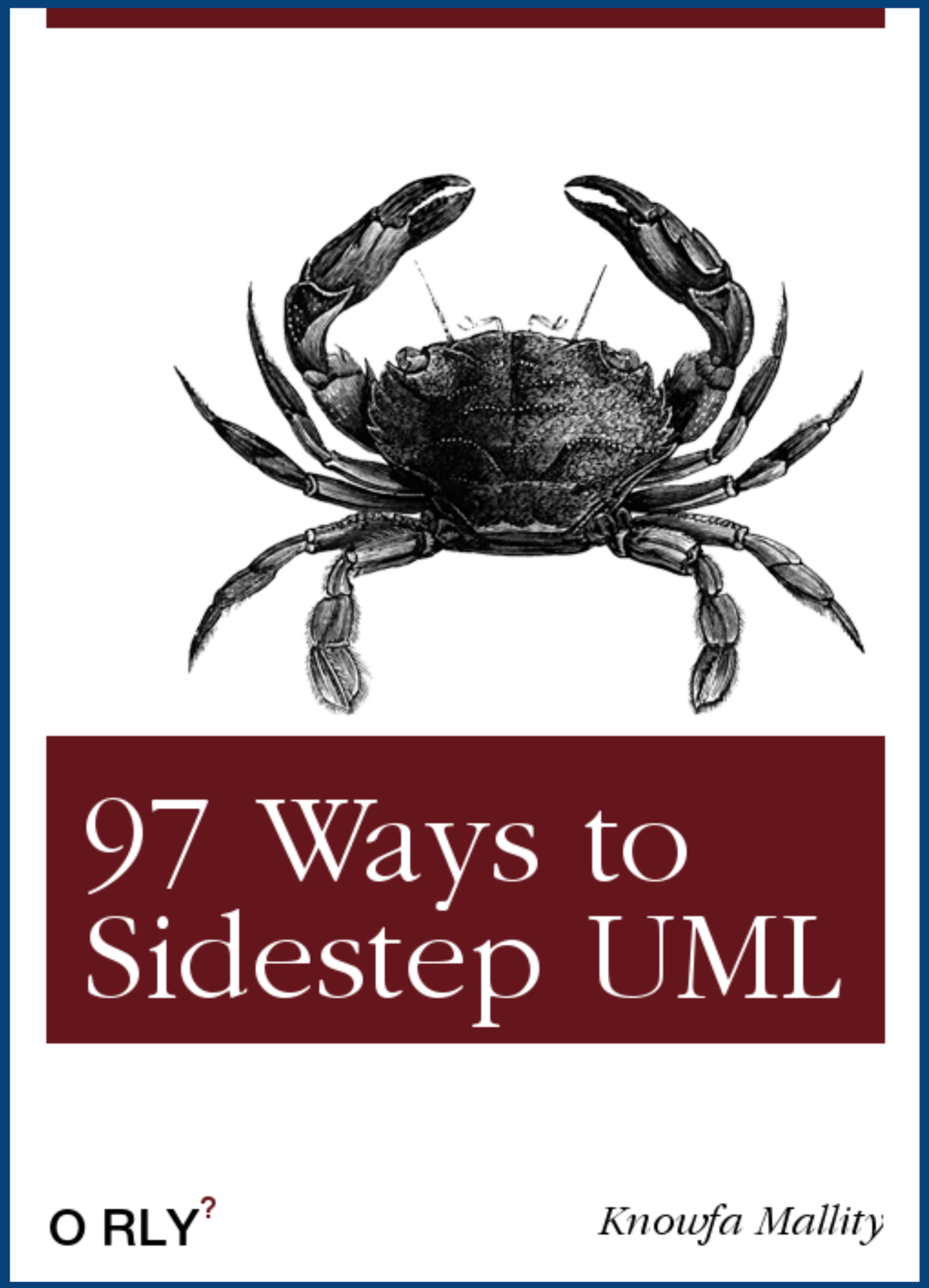

DATA POLL SERVICE
TRADES
REF DATA.
AUTO RE-START.
ARCHIVE
DATA READER TRADES
DATA VALIDATOR
reject
DATA READER REF DATA
DATA VALIDATOR
reject
MONITOR SERVICE SMTP TRAP
FAIL EVENTS
DATA STORE
DATA TRANSFORMATION
INTEGRITY VALIDATION
rejects
tolerance
FAIL EVENT
COMPLETE EVENT
STOP
STOPPED EVENT
COMPLETION 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


Report Monitor
(other machine)
Data Normalizer
Risk Calculator
Aggregator
CP Risk Calc
Report Generator
Auditing Service
Report Publisher
Report Repository


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

UML?

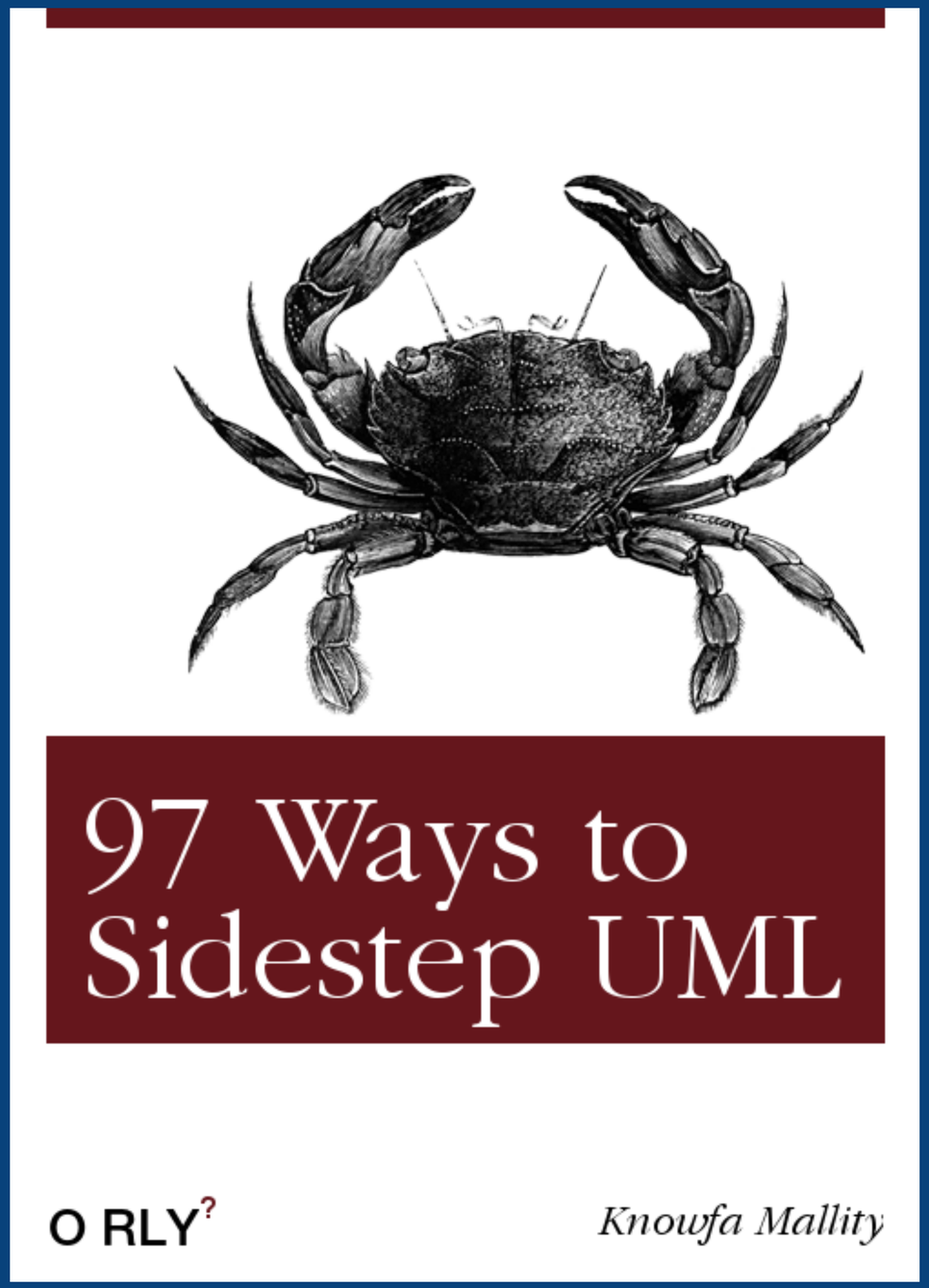


#3

“I’m the only person on the team who knows it.”

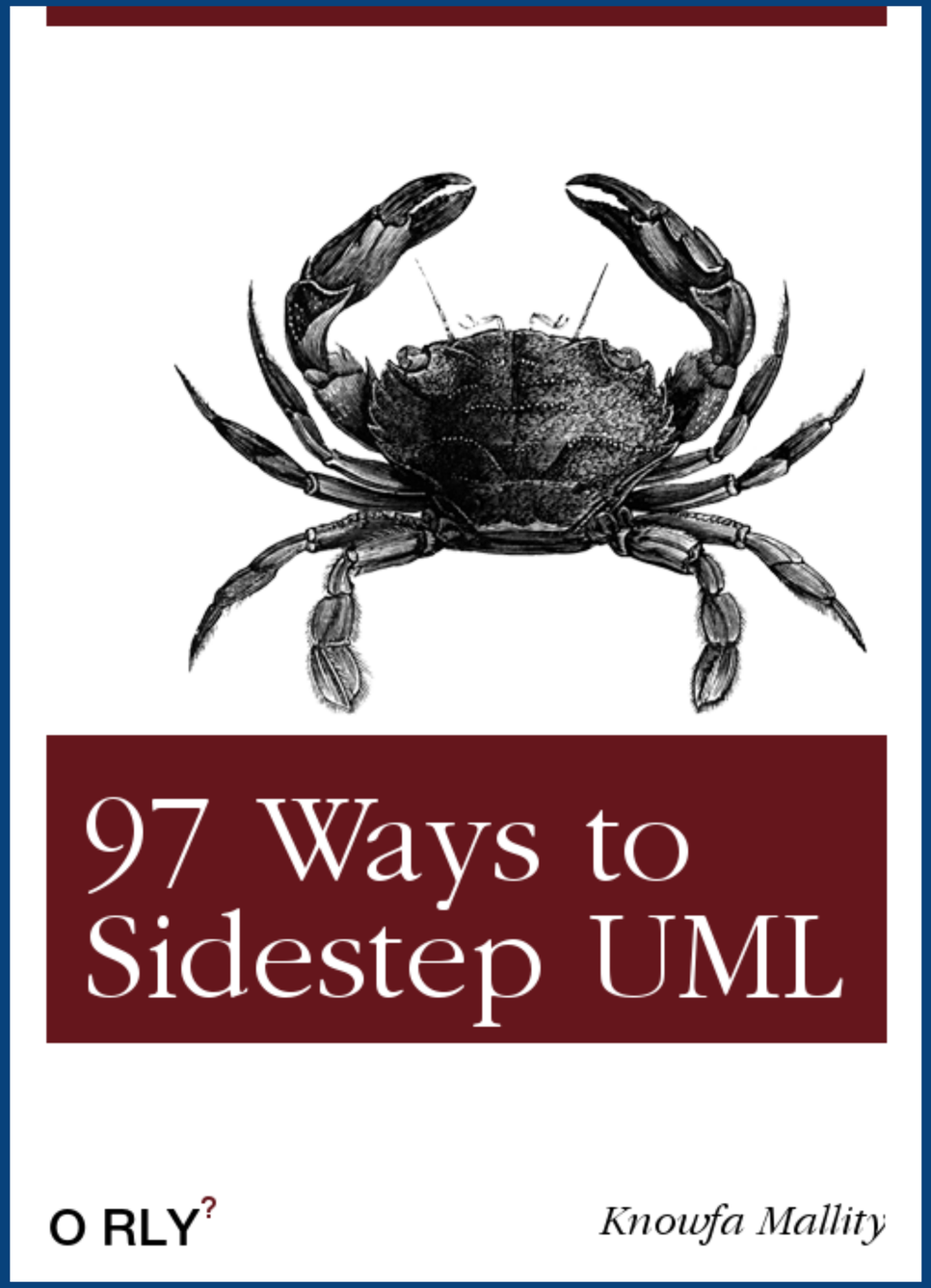
In my experience, optimistically,

1 out of 10 people use UML



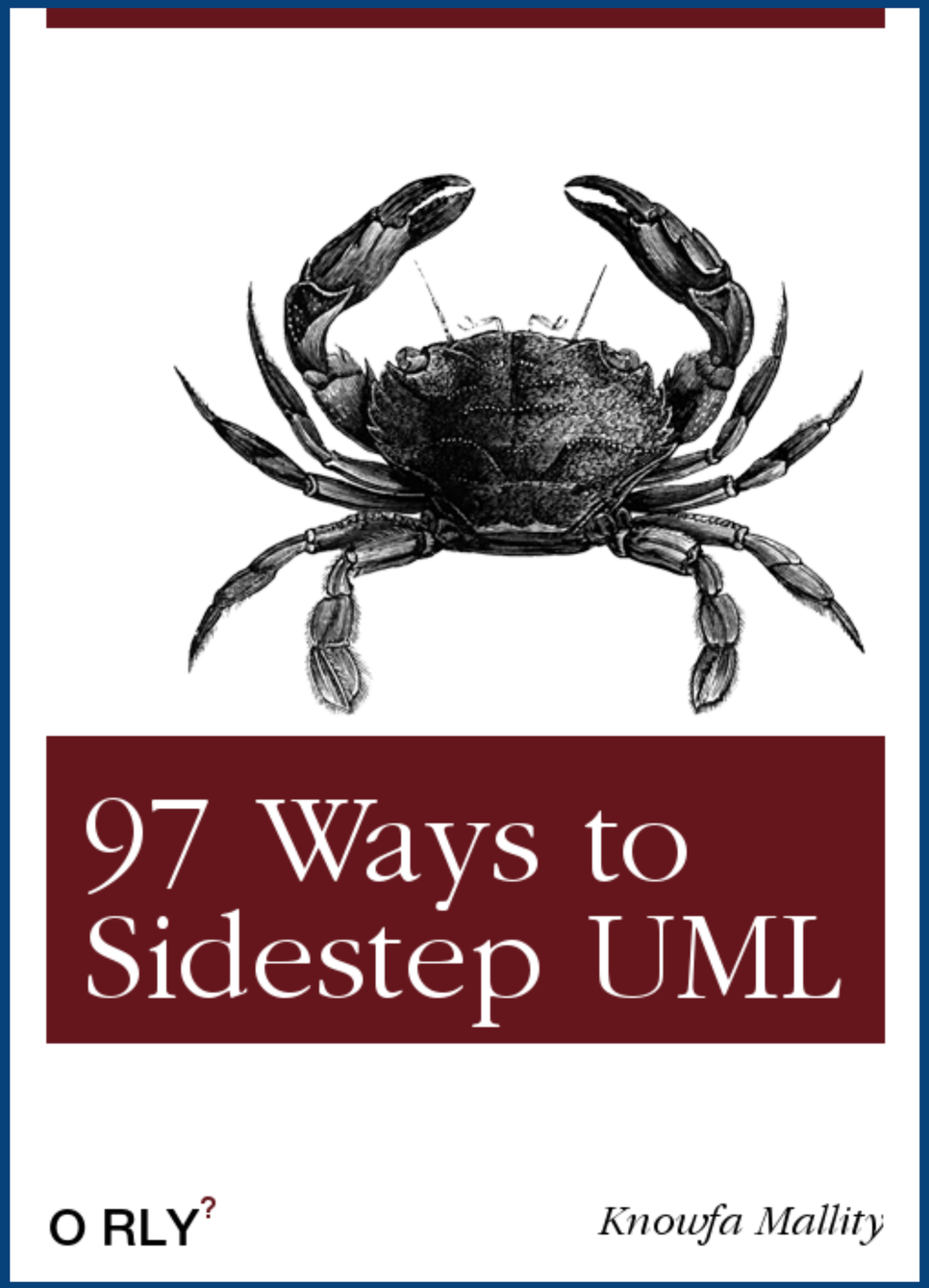
#36

“You’ll be seen as old.”



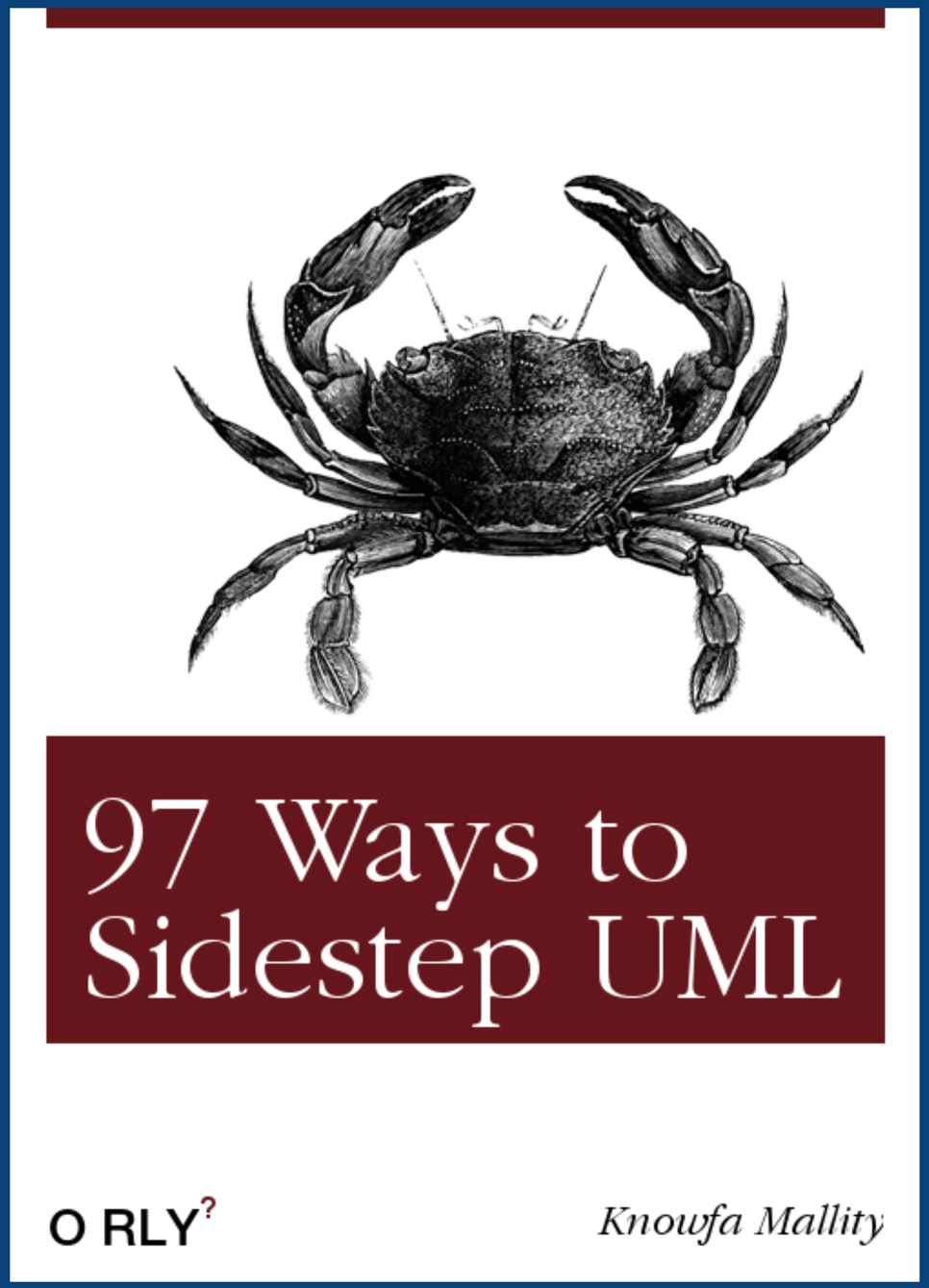
#37

“You’ll be seen as old-fashioned.”



#46

“We don’t want to tell developers what to do.”

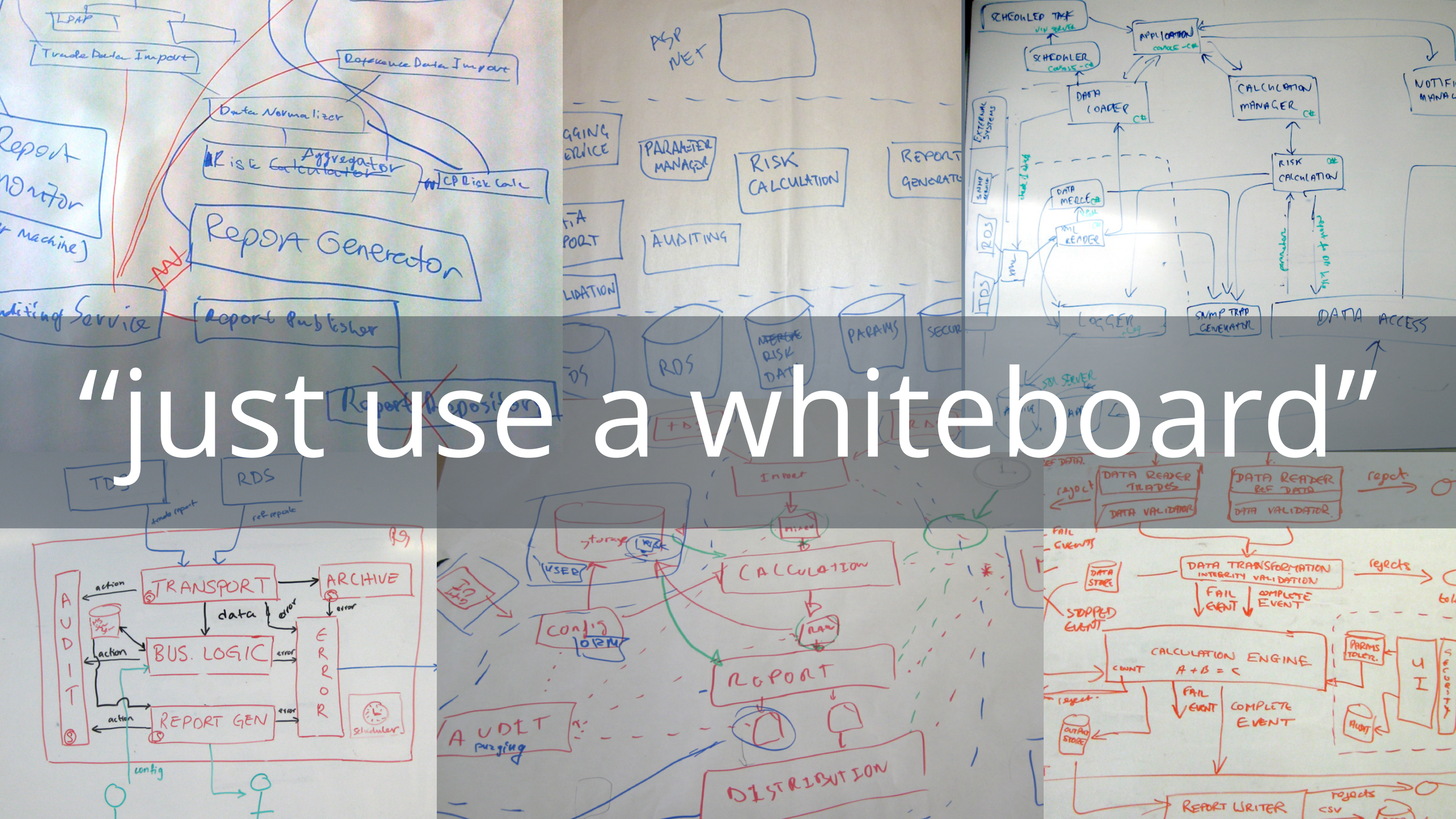


#80

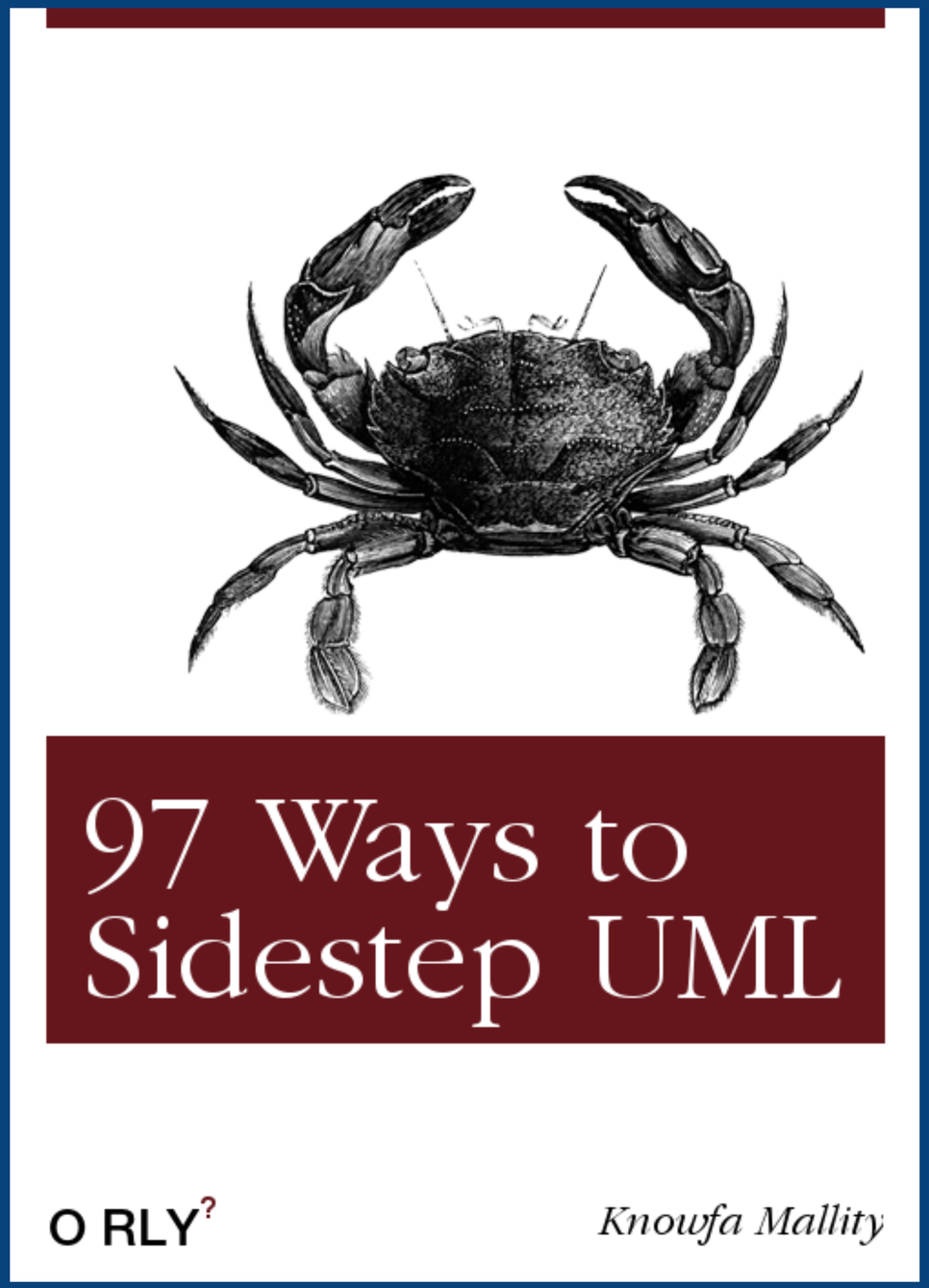
“It’s too detailed.”


UML, Cucumber and modeling reality - MPJ's Musings - Fun Fun Function
Very elaborate waste of time
1:16 / 12:48


UML, Cucumber and modeling reality - MPJ's Musings - Fun Fun Function
Just use a whiteboard!
1:42 / 12:48



“just use a whiteboard”



#97

“The value is in the conversation.”

They are all excellent, as long as there is a conversation about their meaning and intent. It's the accompanying conversation that matters.

“the value is in the conversation”
only works if you’re having
the right conversations

What’s wrong these diagrams?

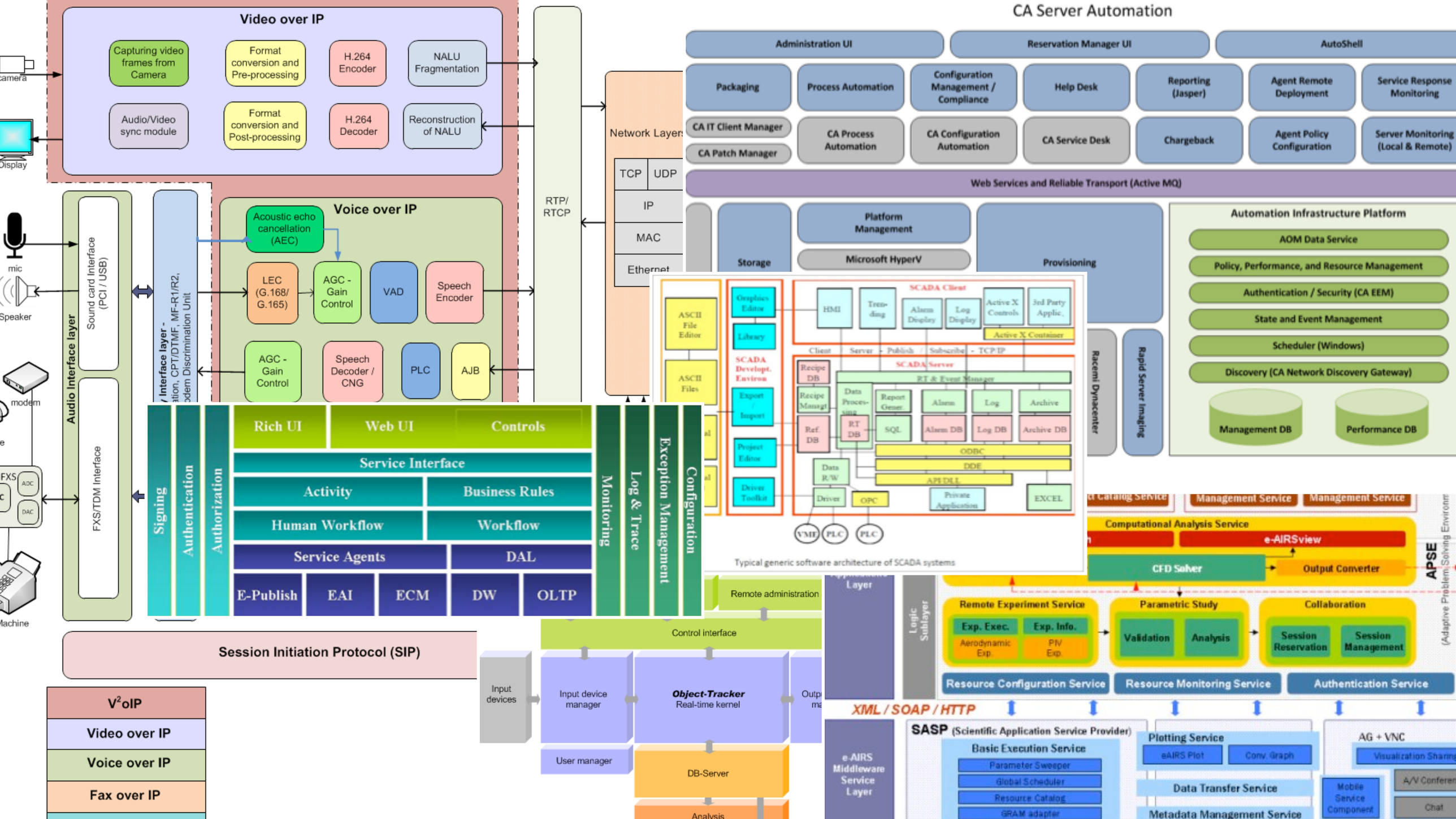
Swap and **review** your diagrams

1. Do the solutions satisfy the architectural drivers?
2. If you were the bank, would you buy this solution?

10 minutes

It’s impossible to
answer those questions

If you can’t see and understand a solution, you can’t evaluate it


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
camera
Display
Voice over IP
Acoustic echo cancellation (AEC)
LEC (G.168/ G.165)
AGC - Gain Control
VAD
Speech Encoder
AGC - Gain Control
Speech Decoder / CNG
PLC
AJB
mic
Speaker
modem
Machine
Audio Interface layer
Sound card Interface (PCI / USB)
FXS/TDM Interface
Interface layer - tion, CPT/DTMF, MF-R1/R2, odem Discrimination Unit
RTP/ RTCP
Network Layers
TCP
UDP
IP
MAC
Ethernet
Signing
Authentication
Authorization
Rich UI
Web UI
Controls
Service Interface
Activity
Business Rules
Human Workflow
Workflow
Service Agents
DAL
E-Publish
EAI
ECM
DW
OLTP
Monitoring
Log & Trace
Exception Management
Configuration
Session Initiation Protocol (SIP)
V²oIP
Video over IP
Voice over IP
Fax over IP
CA Server Automation
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
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
Racemi Dynacenter
Rapid Server Imaging
ASCII File Editor
ASCII Files
Graphics Editor
Library
SCADA Develop. Environ
Export
Import
Project Editor
Driver Toolkit
SCADA Client
HMI
Trending
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 Process-ing
RT & Event Manager
Report Gener.
Alarm
Log
Archive
RT DB
SQL
Alarm DB
Log DB
Archive DB
ODBC
DDE
API DLL
Data R/W
Driver
OPC
Private Application
EXCEL
VME
PLC
PLC
Typical generic software architecture of SCADA systems
Management Service
Management Service
Computational Analysis Service
e-AIRSview
CFD Solver
Output Converter
APSE
(Adaptive Problem Solving Environment)
Remote Experiment Service
Exp. Exec.
Exp. Info.
Aerodynamic Exp.
PIV Exp.
Parametric Study
Validation
Analysis
Collaboration
Session Reservation
Session Management
Resource Configuration Service
Resource Monitoring Service
Authentication Service
XML / SOAP / HTTP
Logic Sublayer
e-AIRS Middleware Service Layer
SASP (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 Conferen
Chat
Remote administration
Control interface
Input devices
Input device manager
Object-Tracker
Real-time kernel
User manager
DB-Server
Analysis

If you’re going to use “boxes & lines”, at least do so in a **structured way**, using a **self-describing notation**

To describe a software architecture, we use a model composed of multiple views or perspectives.

Architectural Blueprints - The “4+1” View Model of Software Architecture

Philippe Kruchten

The description of an architecture—the decisions made—can be organized around these four views, and then illustrated by a few selected *use cases*, or *scenarios* which become a fifth view. The architecture is in fact partially evolved from these scenarios as we will see later.

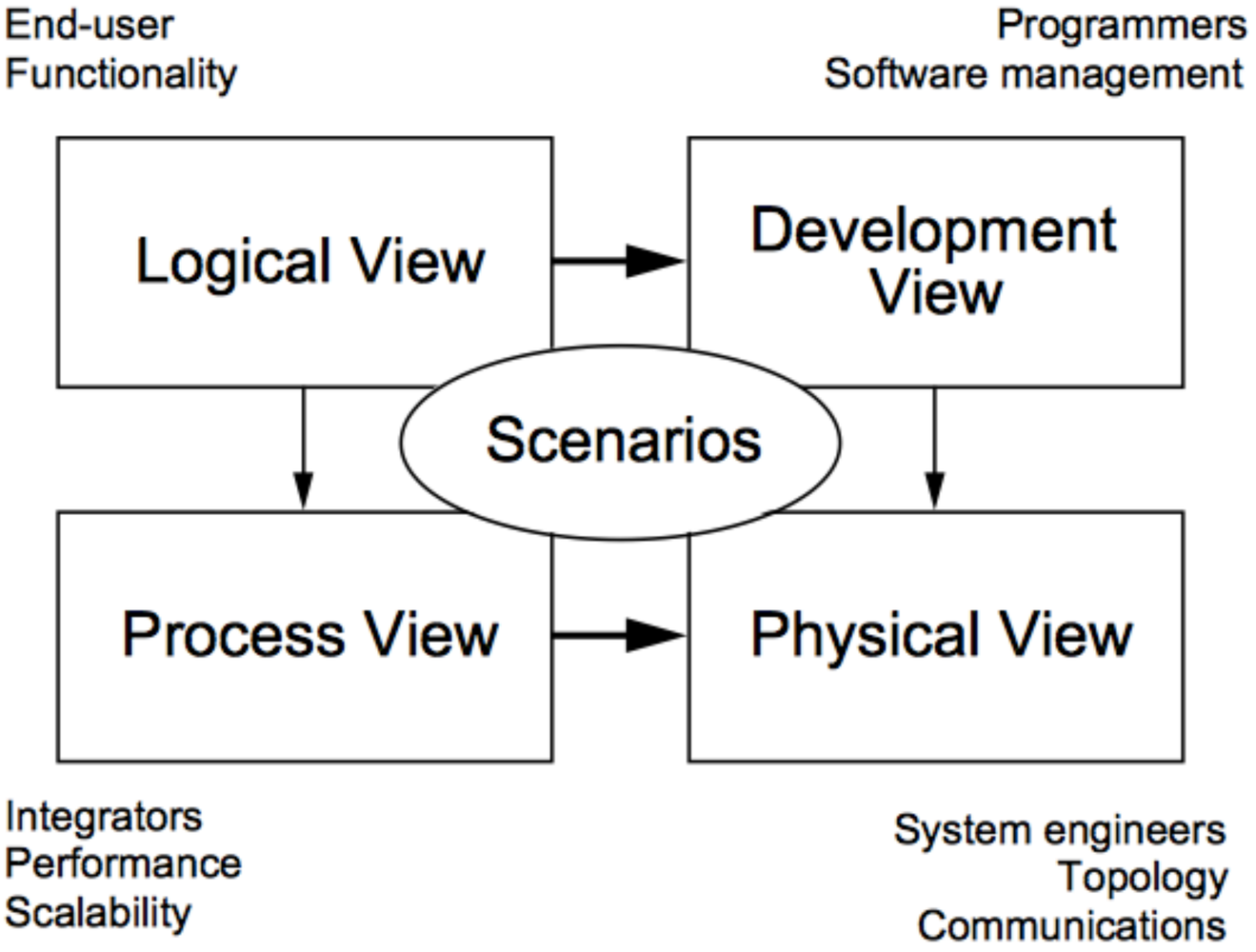
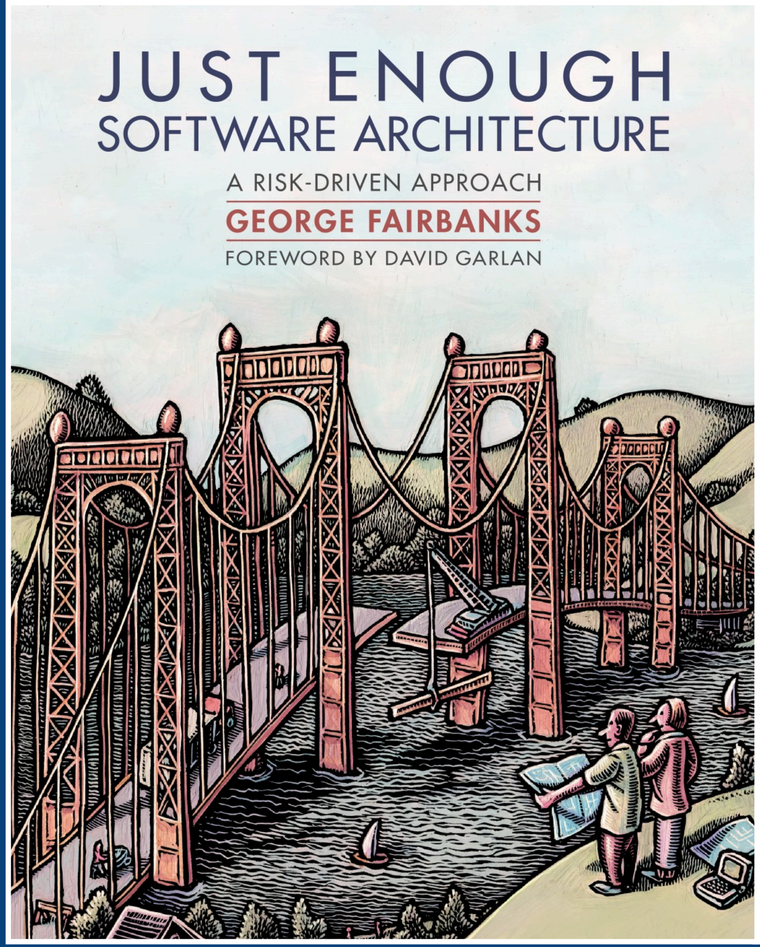


Figure 1 — The "4+1" view model

Why is there a separation between the **logical** and **development** views?

Our architecture diagrams don’t match the code.



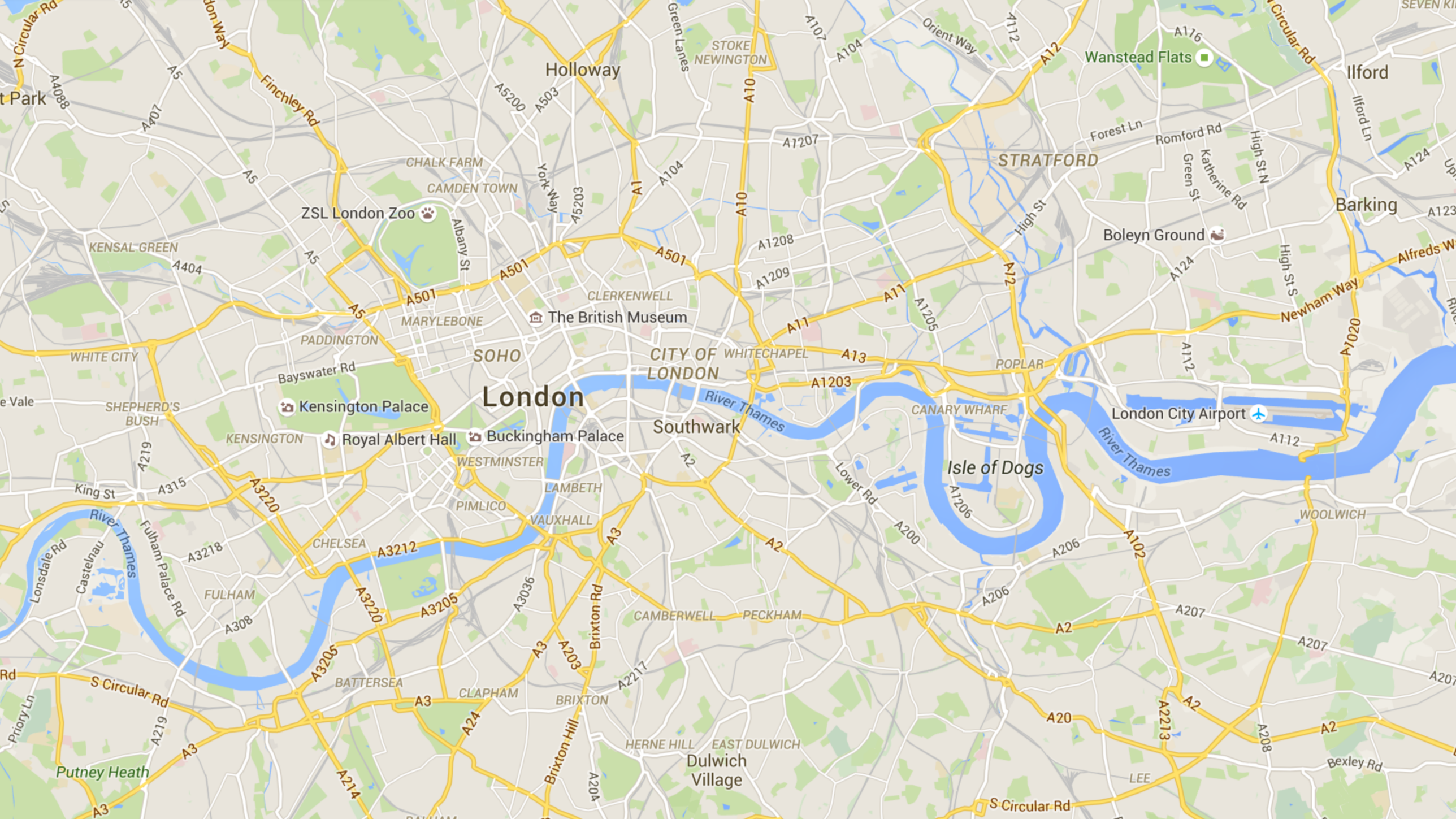
Model-code gap. Your architecture models and your source code will not show the same things. The difference between them is the *model-code gap*. Your architecture models include some abstract concepts, like components, that your programming language does not, but could. Beyond that, architecture models include intensional elements, like design decisions and constraints, that cannot be expressed in procedural source code at all.

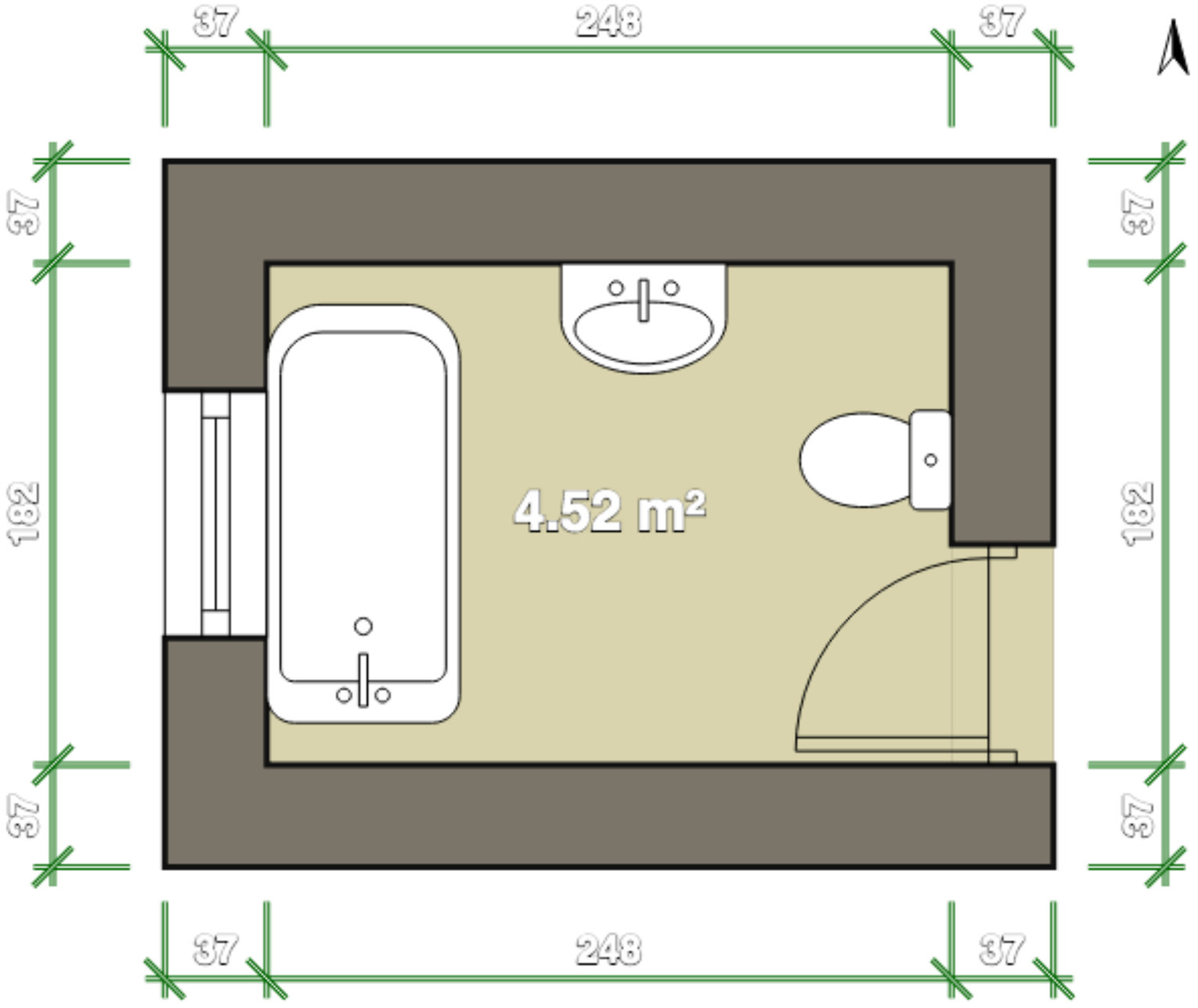
Consequently, the relationship between the architecture model and source code is complicated. It is mostly a refinement relationship, where the extensional elements in the architecture model are refined into extensional elements in source code. This is shown in Figure 10.3. However, intensional elements are not refined into corresponding elements in source code.

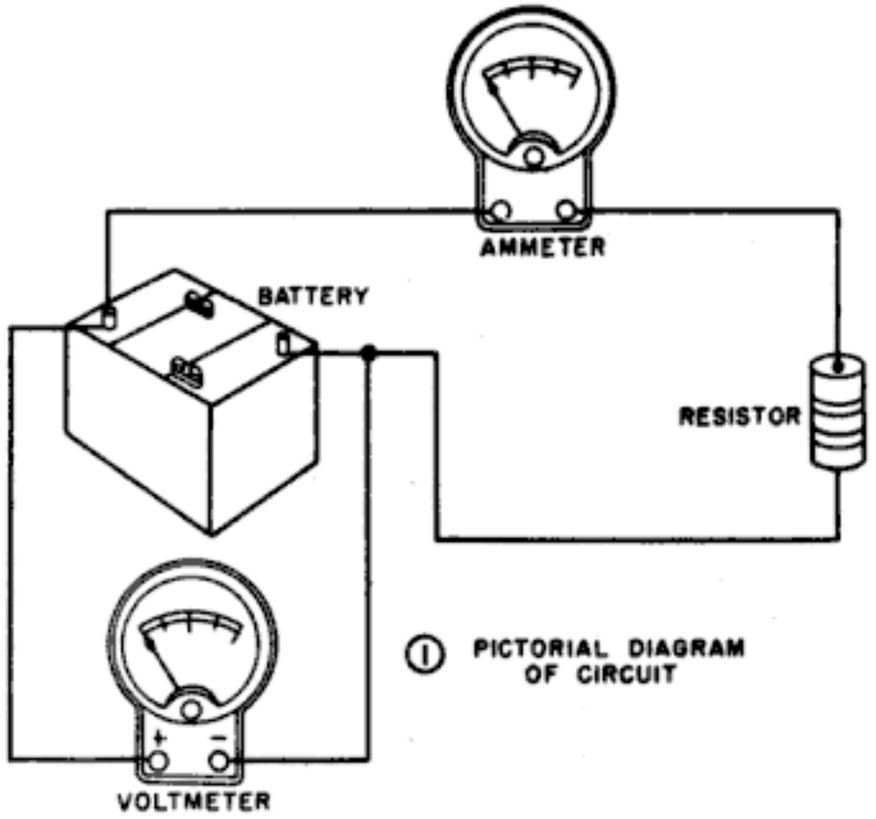
Upon learning about the model-code gap, your first instinct may be to avoid it. But reflecting on the origins of the gap gives little hope of a general solution in the short term: architecture models help you reason about complexity and scale because they are abstract and intensional; source code executes on machines because it is concrete and extensional.

“model-code gap”

We lack a **common vocabulary**
to describe software architecture


London
Holloway
STOKE NEWINGTON
Wanstead Flats
Ilford
Barking
STRATFORD
CHALK FARM
CAMDEN TOWN
ZSL London Zoo
Boleyn Ground
KENSAL GREEN
MARYLEBONE
The British Museum
CLERKENWELL
PADDINGTON
WHITE CITY
SOHO
CITY OF LONDON
WHITECHAPEL
POPLAR
Bayswater Rd
Kensington Palace
SHEPHERD'S BUSH
KENSINGTON
Royal Albert Hall
Buckingham Palace
WESTMINSTER
Southwark
River Thames
CANARY WHARF
London City Airport
Isle of Dogs
LAMBETH
PIMLICO
VAUXHALL
CHELSEA
FULHAM
CAMBERWELL
PECKHAM
BATTERSEA
CLAPHAM
BRIXTON
HERNE HILL
EAST DULWICH
Dulwich Village
Putney Heath
WOOLWICH
LEE
Bexley Rd
Finchley Rd
Albany St
York Way
Green Lanes
Orient Way
Forest Ln
Romford Rd
Green St
Katherine Rd
High St
High St N
High St S
Ilford Ln
Newham Way
Alfreds W
Lower Rd
Brixton Rd
Brixton Hill
Fulham Palace Rd
Lonsdale Rd
Castelnau
Priory Ln
King St
S Circular Rd
N Circular Rd
A1
A2
A3
A5
A10
A11
A12
A13
A20
A24
A40
A102
A104
A107
A112
A116
A123
A124
A200
A201
A202
A203
A204
A206
A207
A208
A214
A219
A308
A315
A404
A407
A501
A503
A1020
A1203
A1205
A1206
A1207
A1208
A1209
A2213
A2217
A3036
A3205
A3212
A3218
A3220
A4088
A5200
A5203


37
248
37
37
182
37
37
182
37
4.52 m²
37
248
37



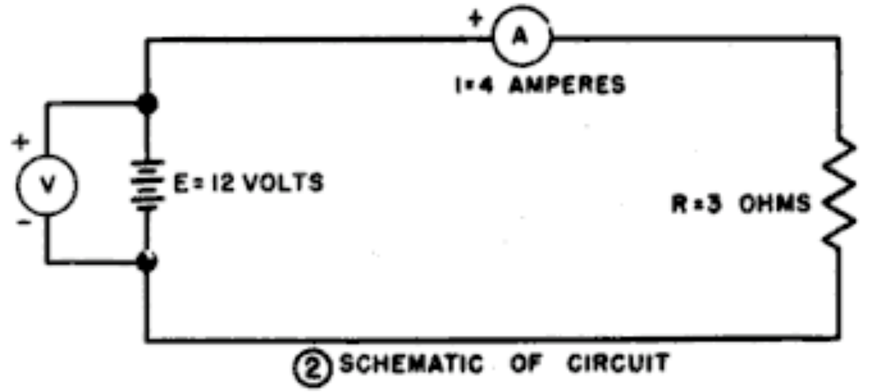
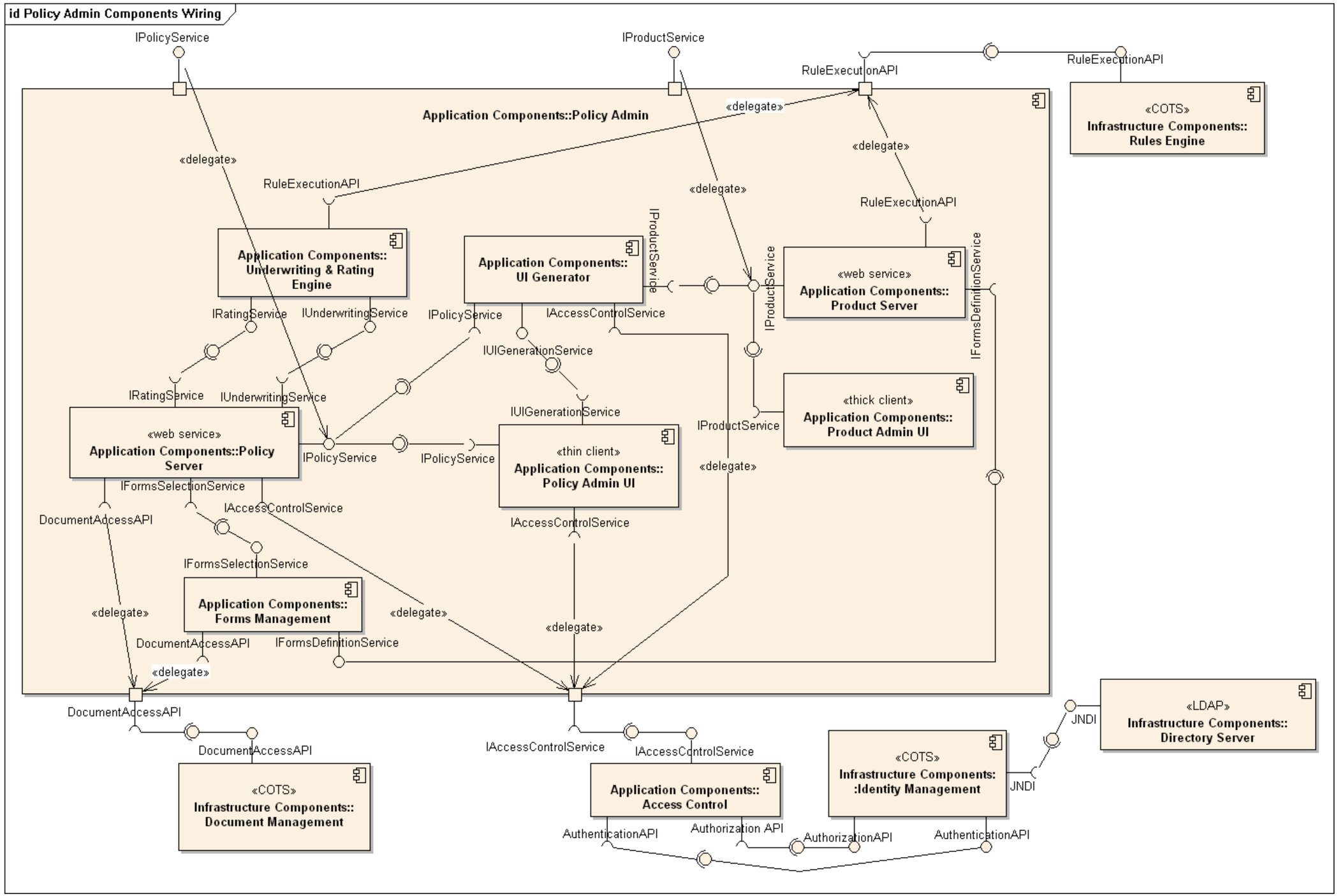
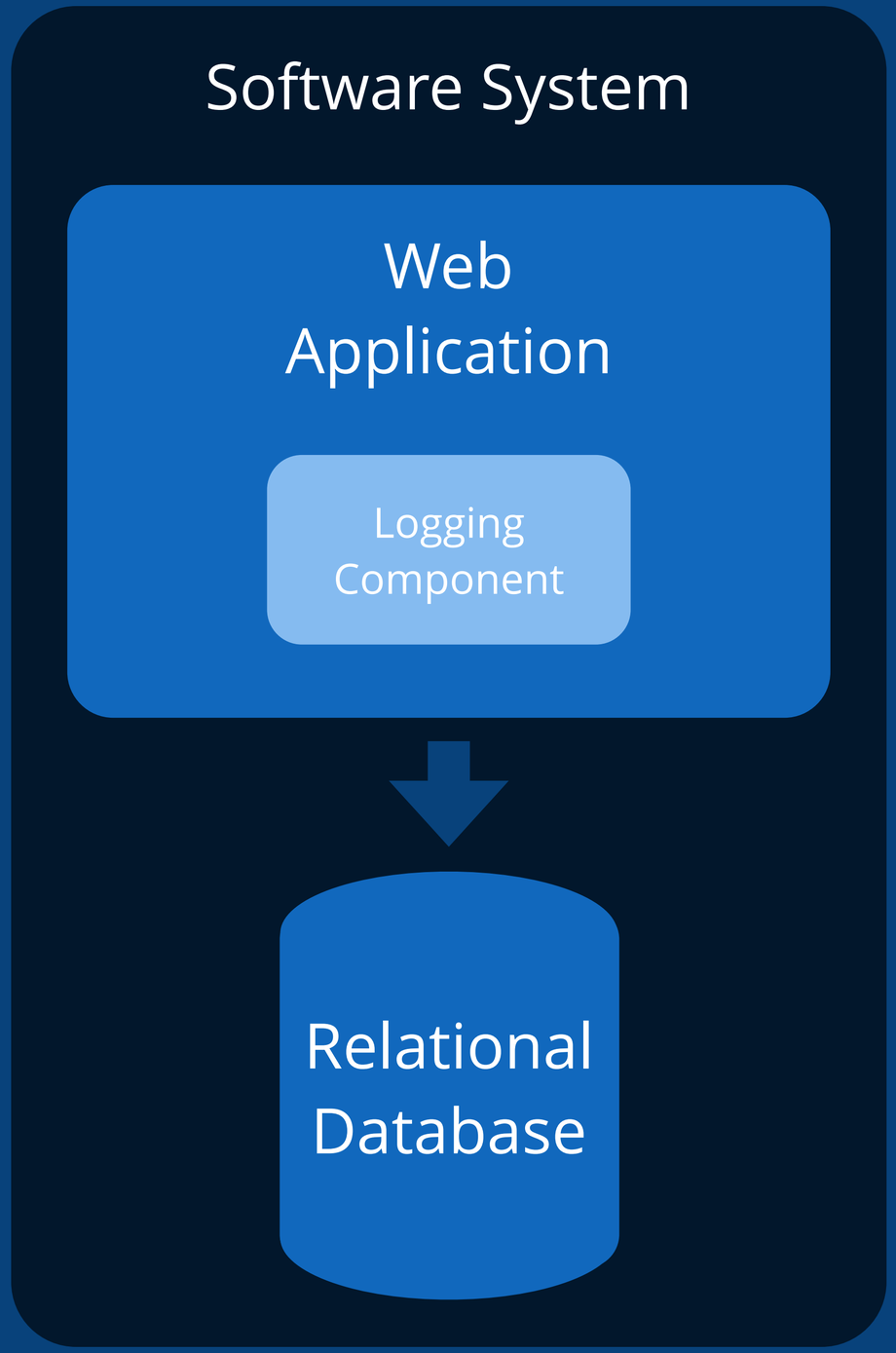


Figure 48. Diagram of a basic circuit.

https://en.wikipedia.org/wiki/Circuit_diagram



https://en.wikipedia.org/wiki/Component_diagram



component

noun | com·po·nent | \kəm-ˈpō-nənt, ˈkäm-ˌ, käm-ˈ\

Simple Definition of COMPONENT

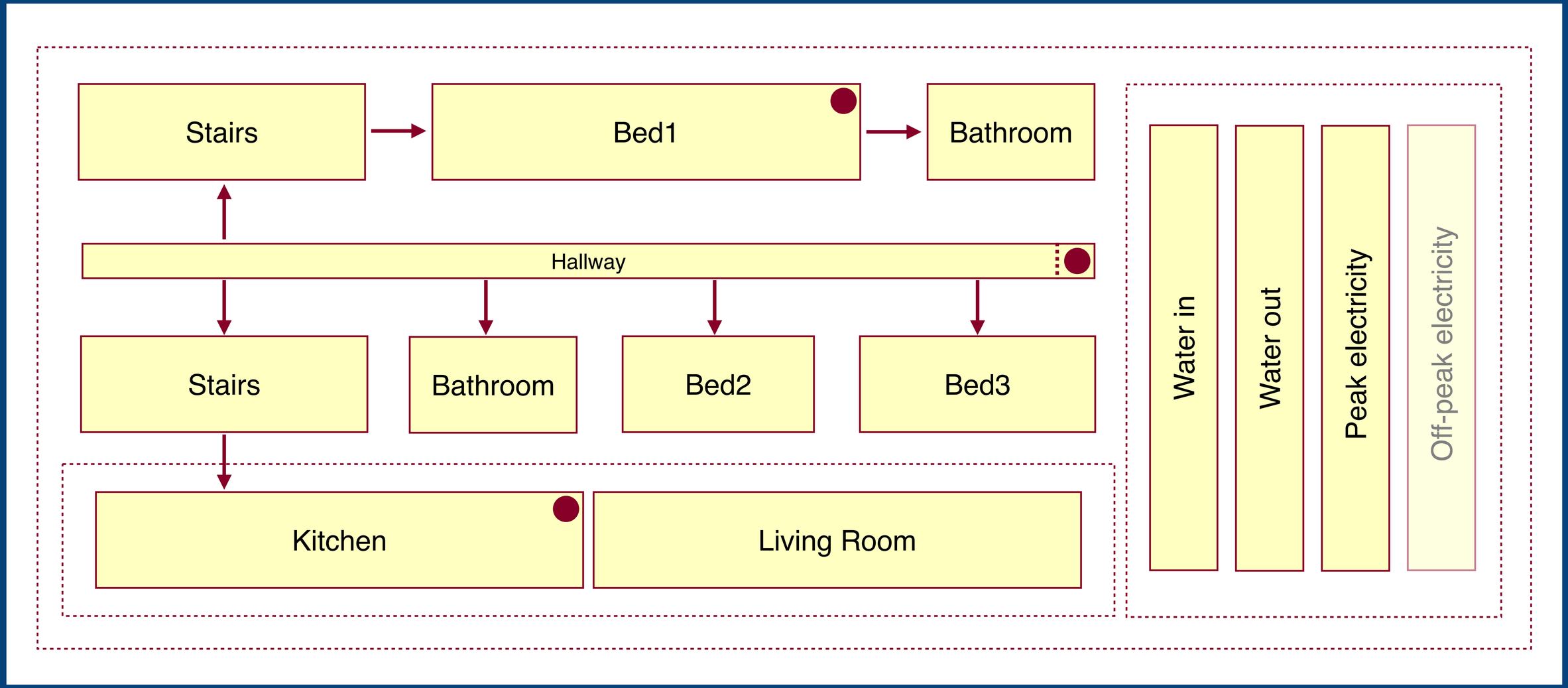
Popularity: Top 30% of words

: one of the parts of something (such as a system or mixture) : an important piece of something

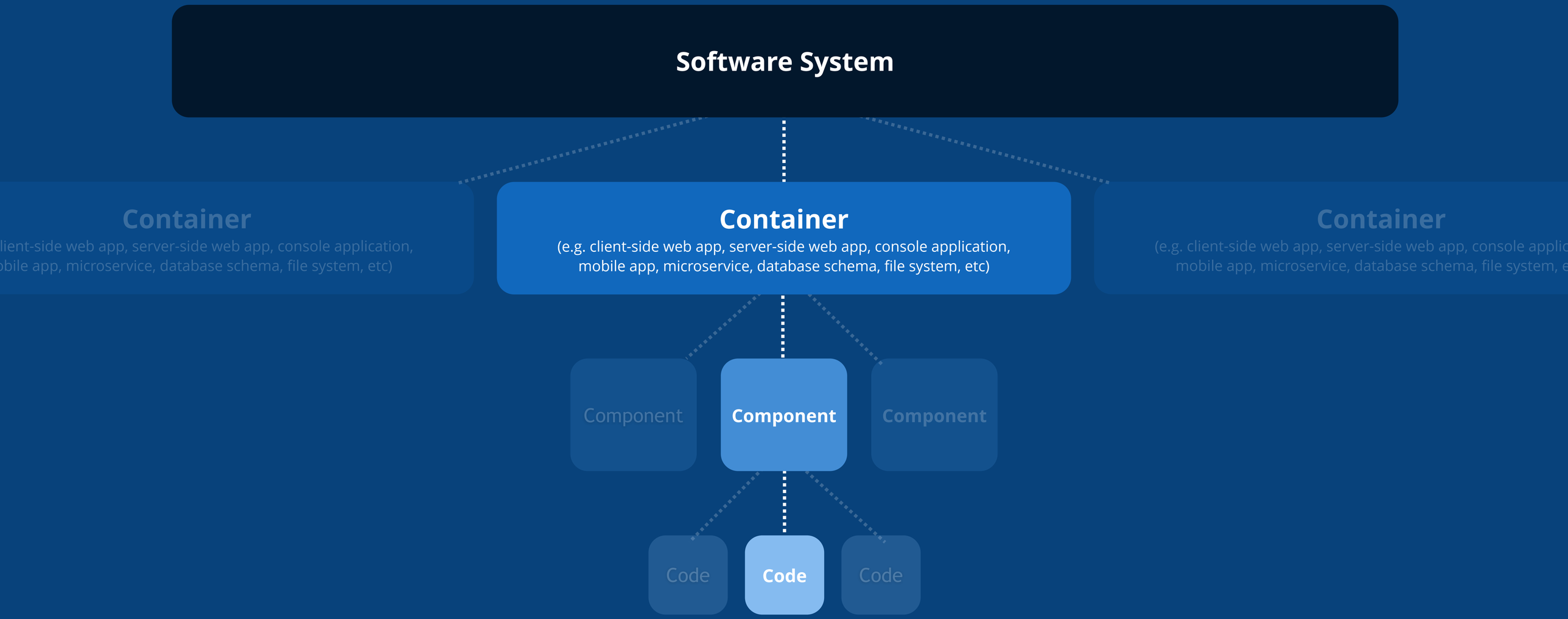
Source: Merriam-Webster's Learner's Dictionary

When drawing software architecture diagrams, think like a software developer

If software developers created building architecture diagrams...



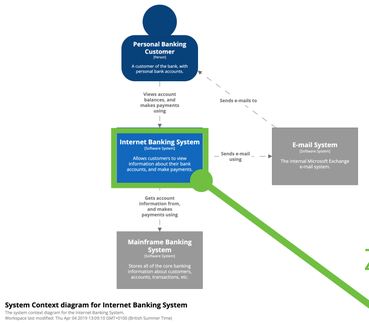
A **common set of abstractions**
is more important
than a common notation



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 **code elements**.

The C4 model for visualising software architecture

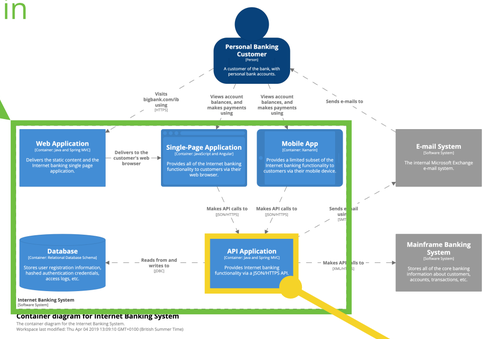
c4model.com



System Context diagram for Internet Banking System

The system context diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

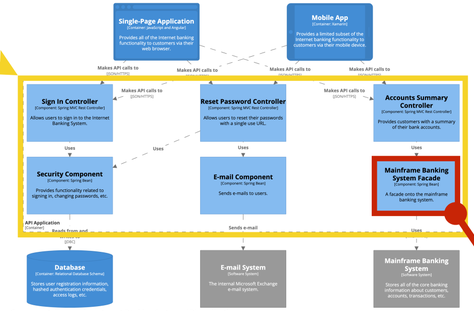
Zoom in



Container diagram for Internet Banking System

The container diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

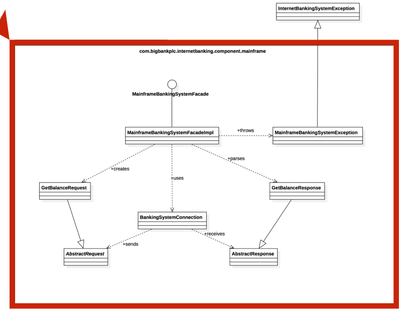
Zoom in



Component diagram for Internet Banking System - API Application

The component diagram for the API Application.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

Zoom in



Level 1
System Context

Level 2
Containers

Level 3
Components

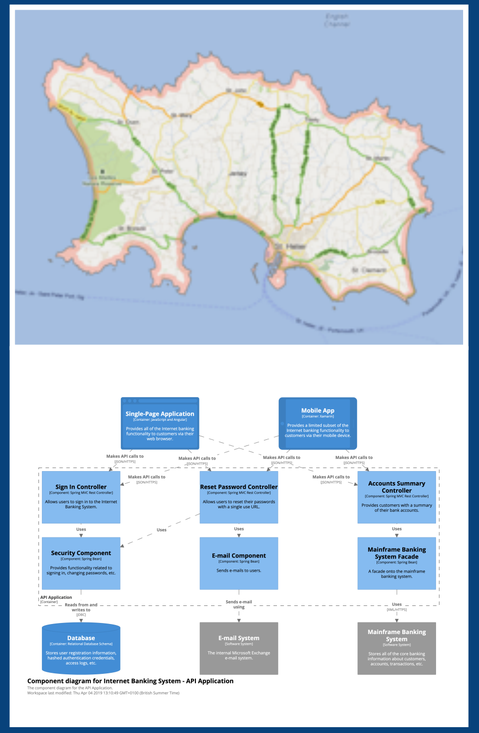
Level 4
Code

System Context diagram for Internet Banking System

The system context diagram for the Internet Banking System.
Workspace last modified: Thu, Apr 04 2019 13:15:48 GMT+0100 (British Summer Time)

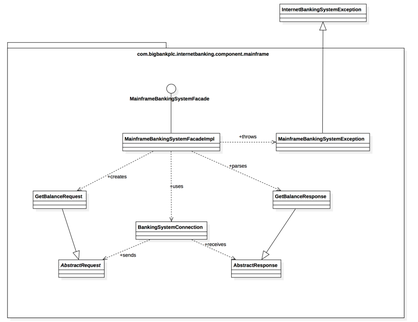
Container diagram for Internet Banking System

The container diagram for the Internet Banking System.
Workspace last modified: Thu, Apr 04 2019 13:15:48 GMT+0100 (British Summer Time)



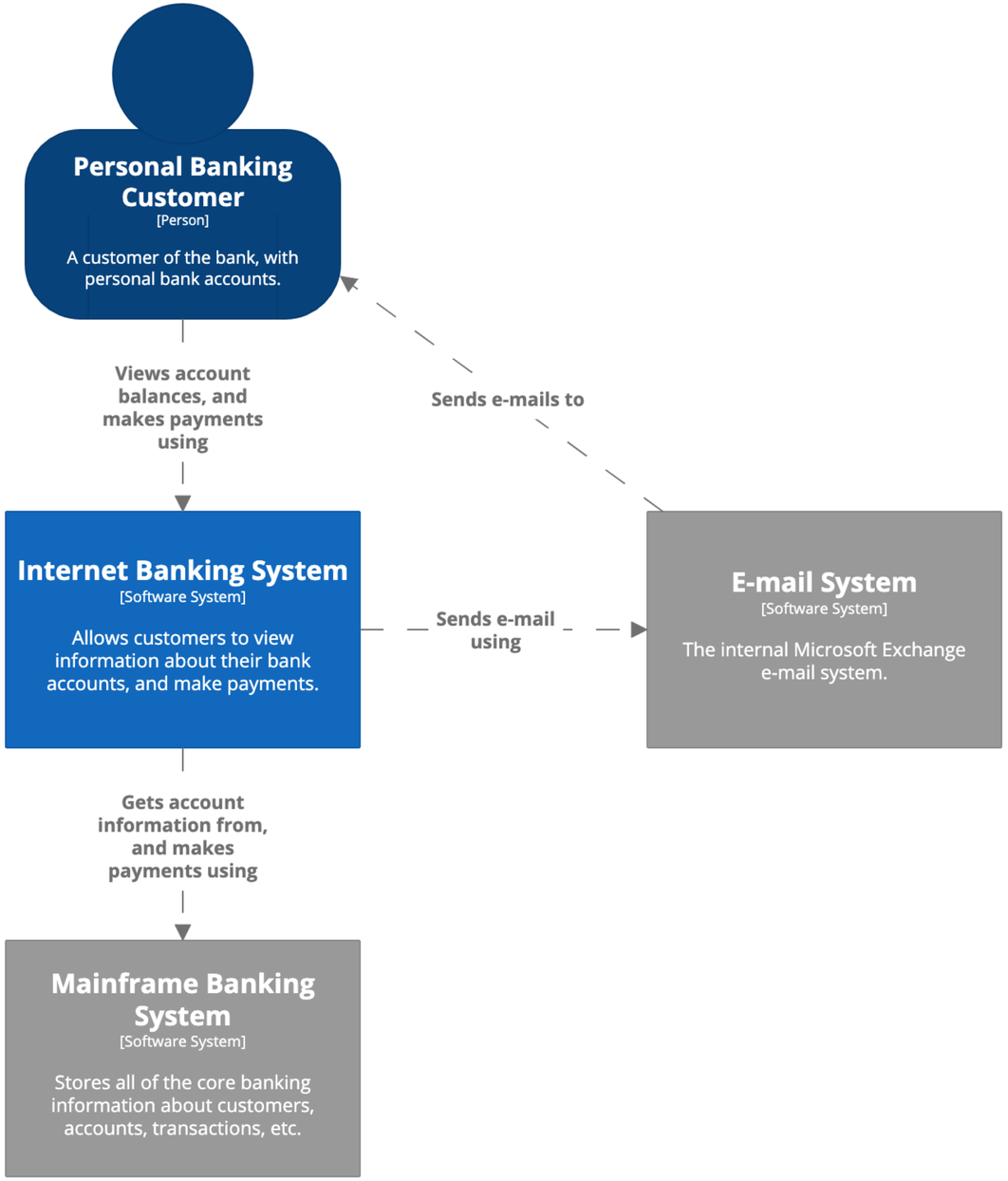
Component diagram for Internet Banking System - API Application

The component diagram for the API Application.
Workspace last modified: Thu, Apr 04 2019 13:15:48 GMT+0100 (British Summer Time)



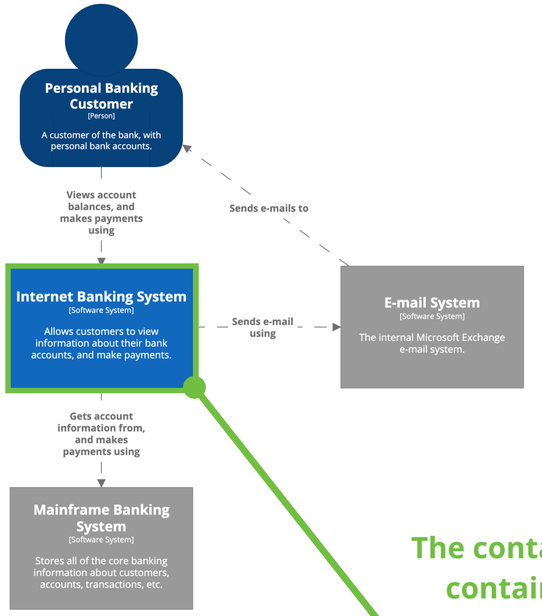
Diagrams are maps

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



System Context diagram for Internet Banking System

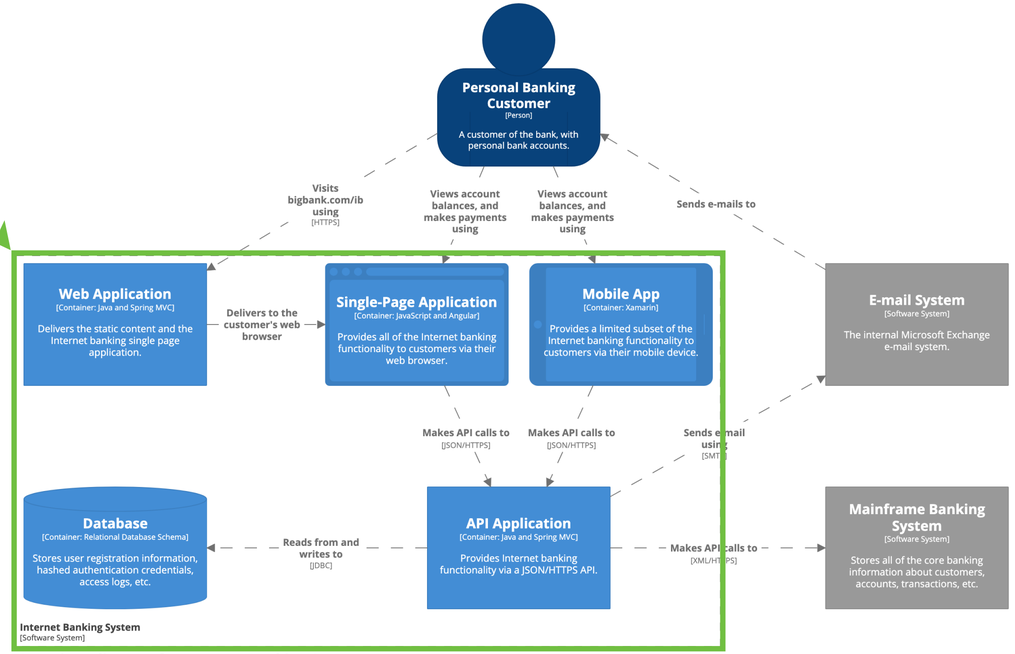
The system context diagram for the Internet Banking System.
Workspace last modified: Wed Feb 05 2020 09:33:36 GMT+0100 (Central European Standard Time)



System Context diagram for Internet Banking System

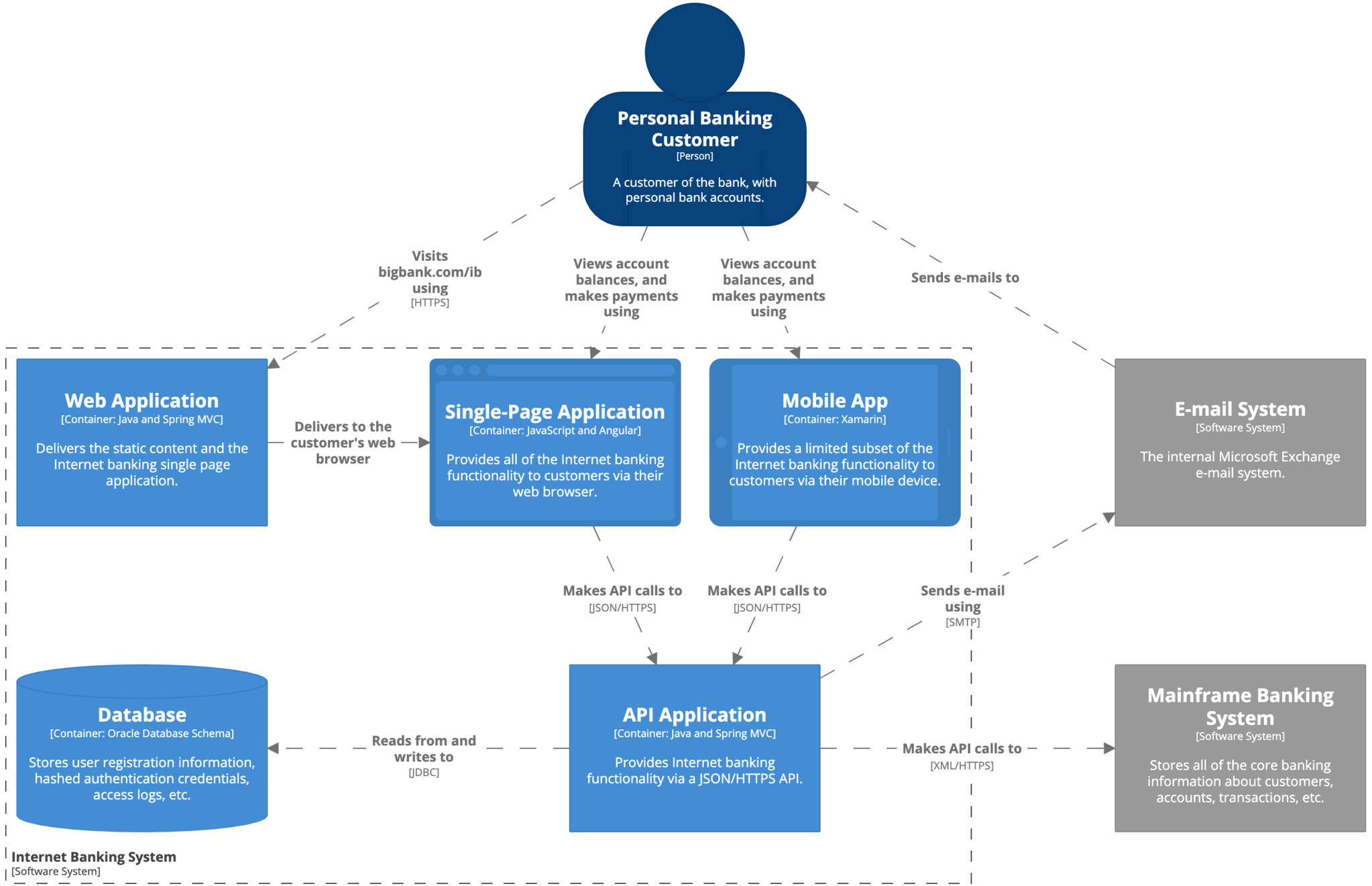
The system context diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

The container diagram shows the containers that reside inside the software system boundary



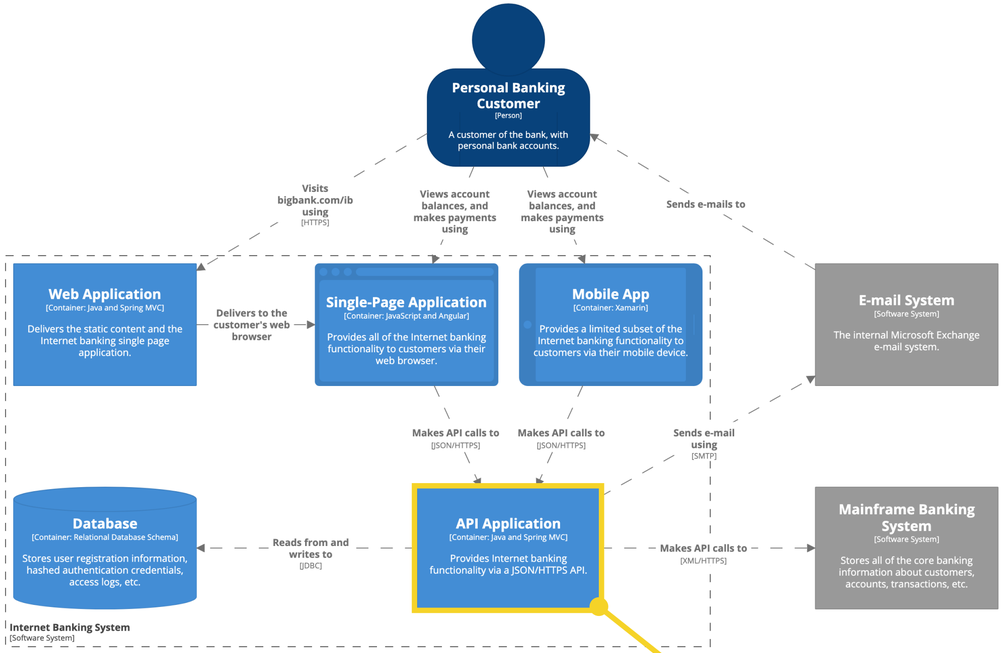
Container diagram for Internet Banking System

The container diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)



Container diagram for Internet Banking System

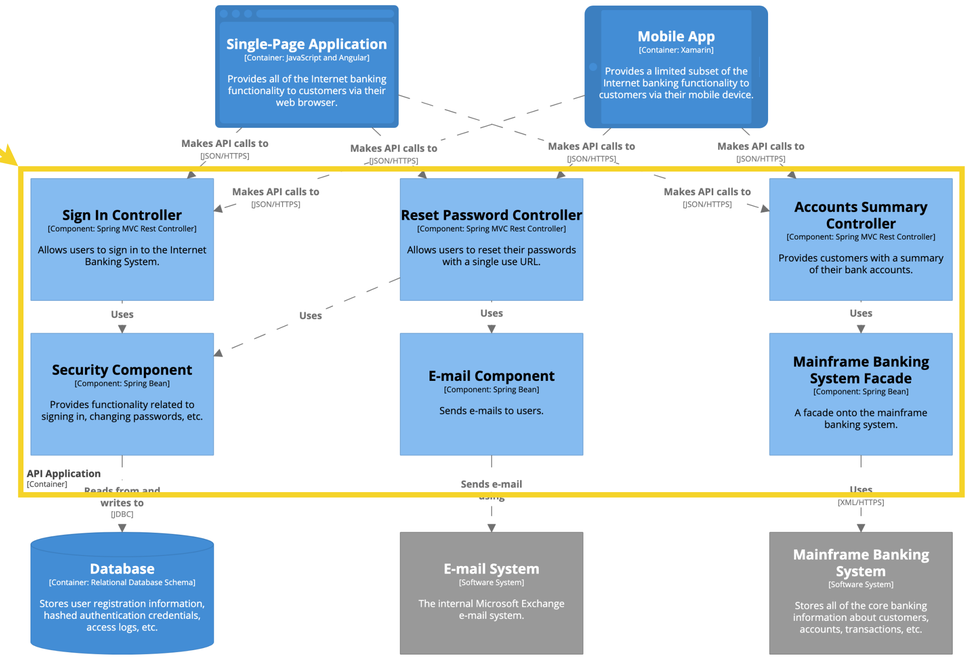
The container diagram for the Internet Banking System.
Workspace last modified: Wed Feb 05 2020 09:33:36 GMT+0100 (Central European Standard Time)



Container diagram for Internet Banking System

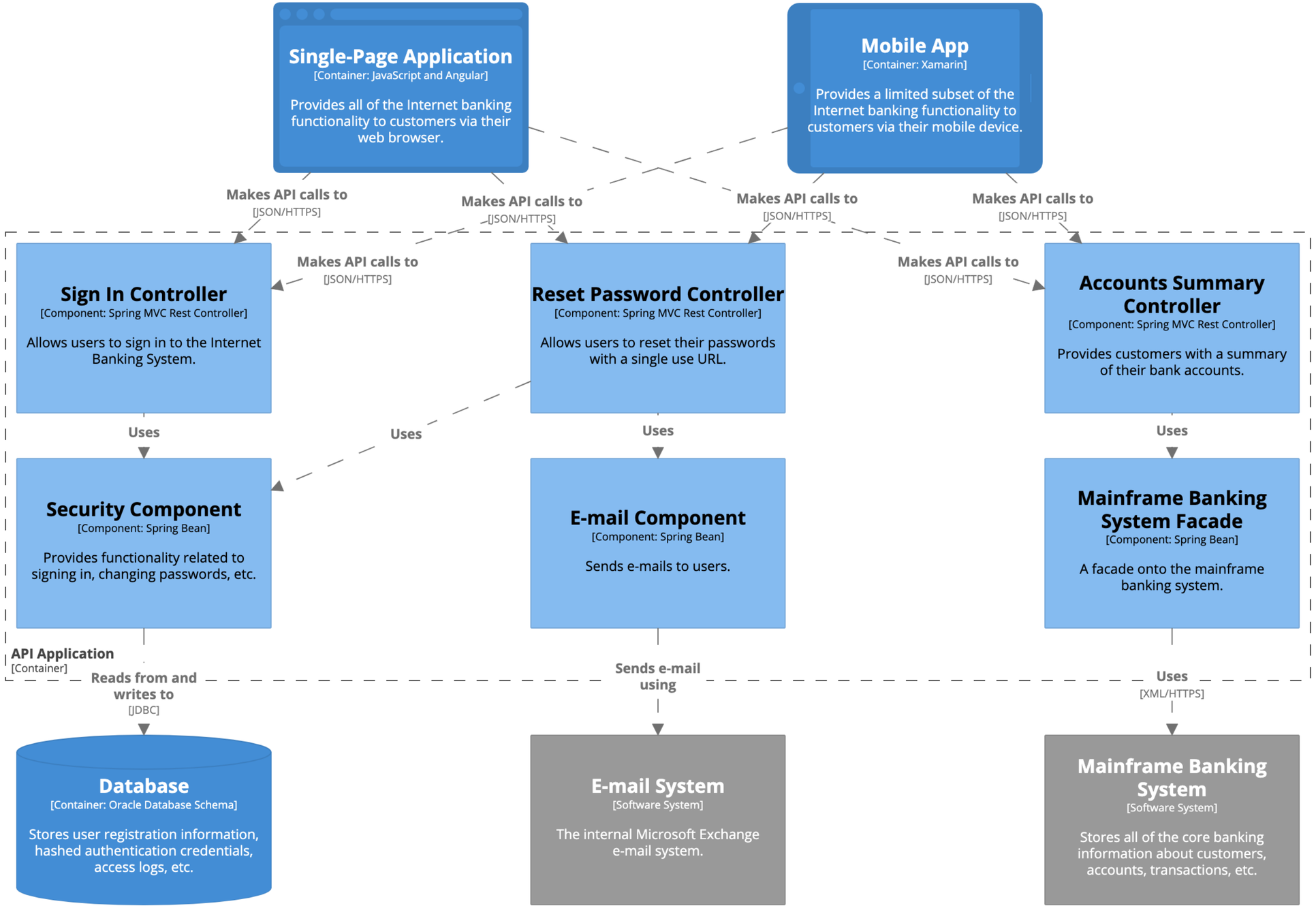
The container diagram for the Internet Banking System.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

The component diagram shows the components that reside inside an individual container



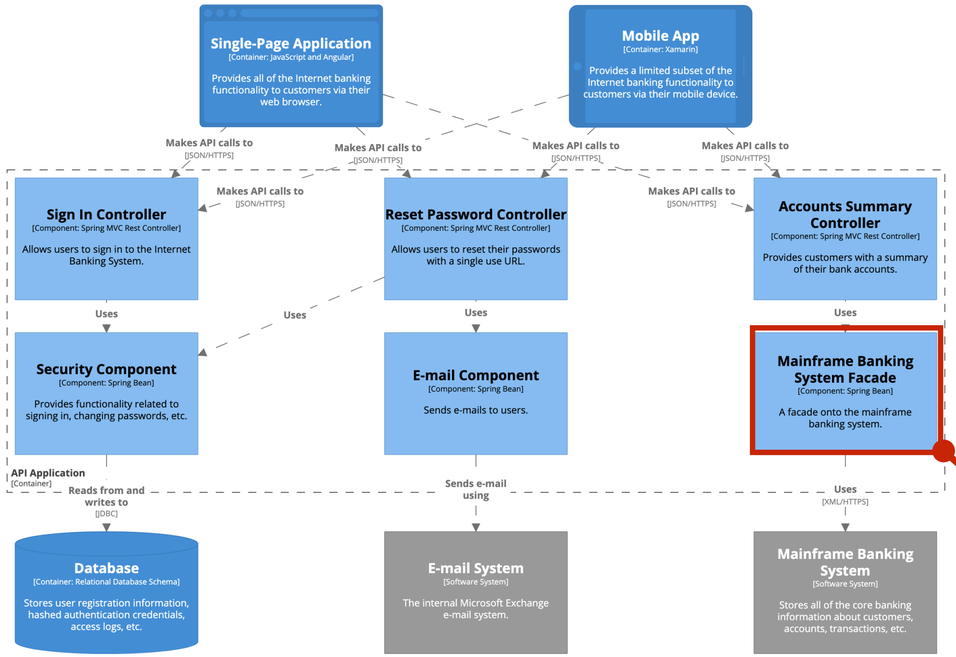
Component diagram for Internet Banking System - API Application

The component diagram for the API Application.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)



Component diagram for Internet Banking System - API Application

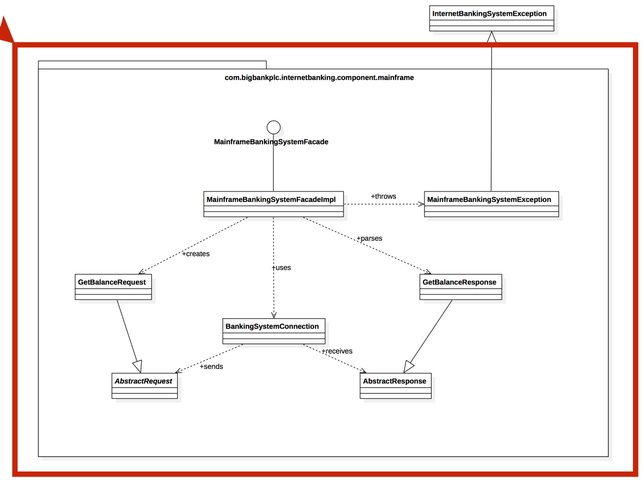
The component diagram for the API Application.
Workspace last modified: Wed Feb 05 2020 09:33:36 GMT+0100 (Central European Standard Time)

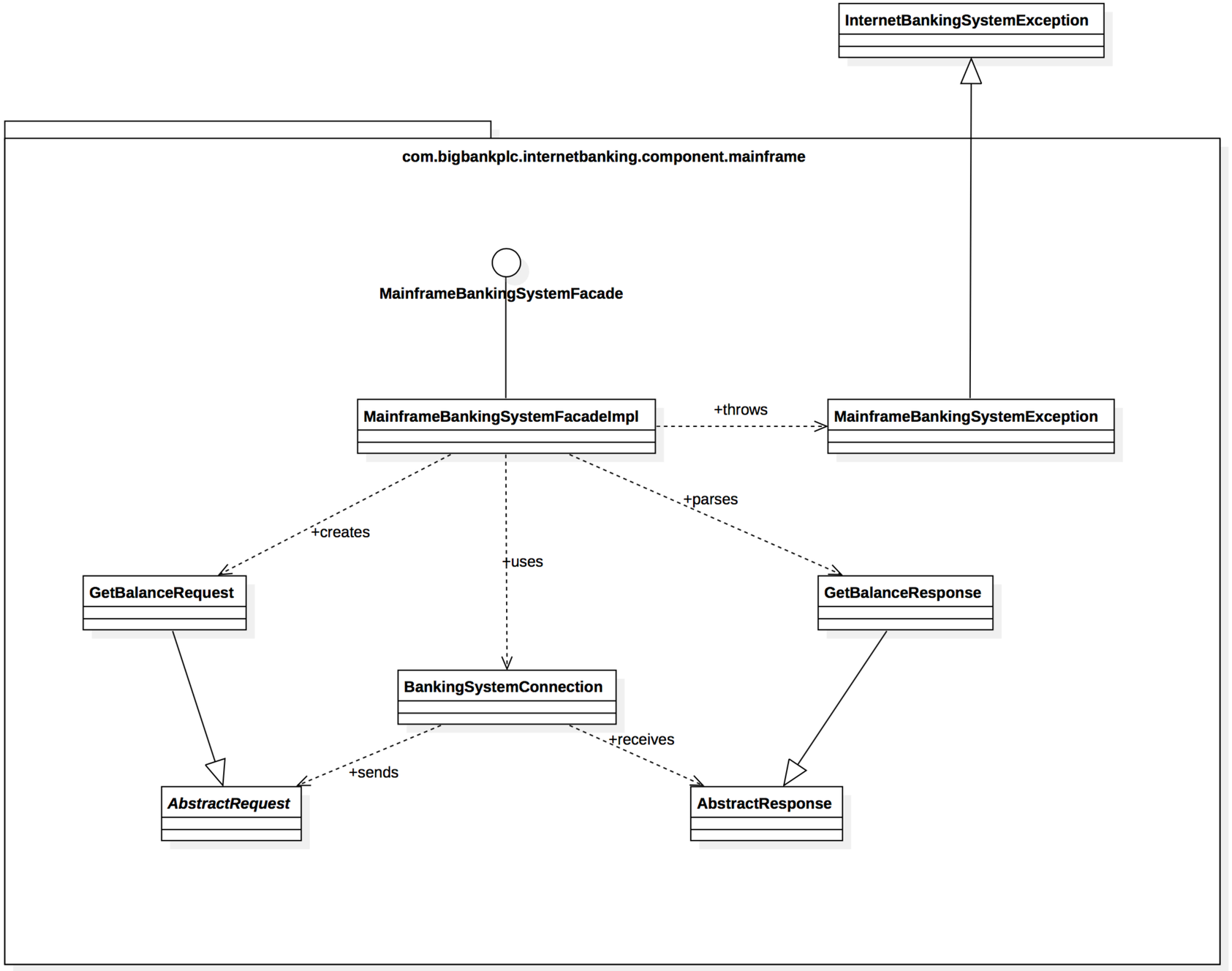


Component diagram for Internet Banking System - API Application

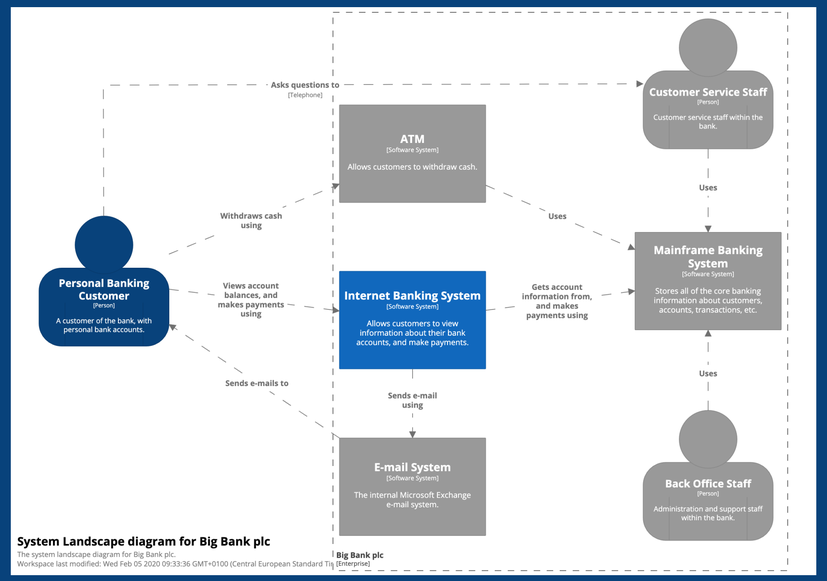
The component diagram for the API Application.
Workspace last modified: Thu Apr 04 2019 13:09:10 GMT+0100 (British Summer Time)

The code level diagram shows the code elements that make up a component

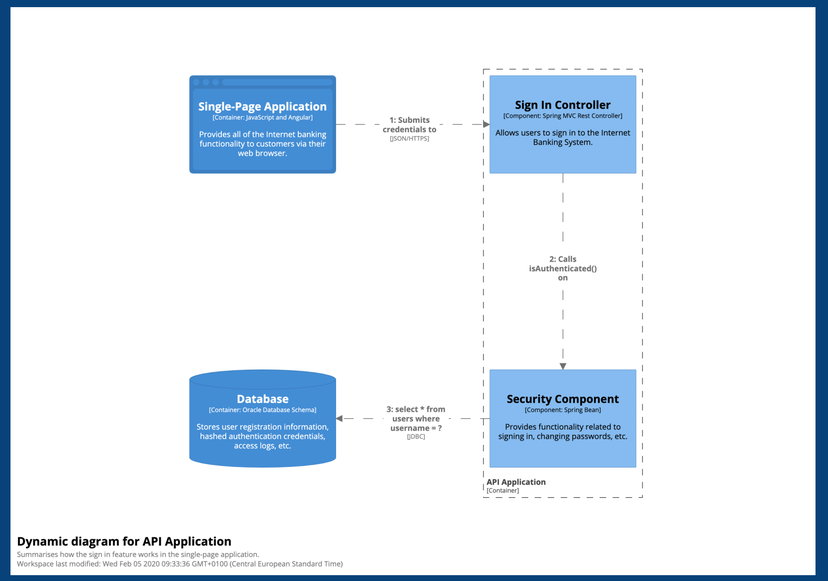



InternetBankingSystemException
com.bigbankplc.internetbanking.component.mainframe
MainframeBankingSystemFacade
MainframeBankingSystemFacadeImpl
+throws
MainframeBankingSystemException
+creates
+uses
+parses
GetBalanceRequest
GetBalanceResponse
BankingSystemConnection
+receives
+sends
AbstractRequest
AbstractResponse

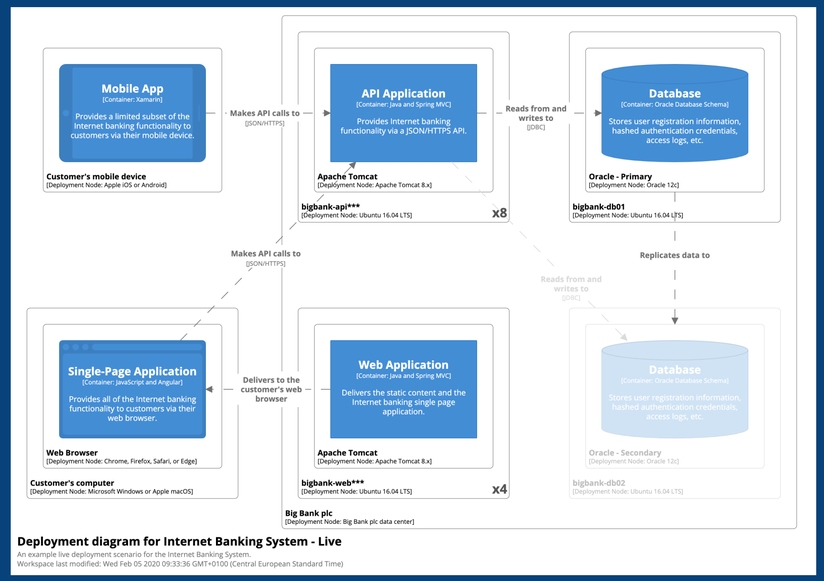
Plus some supplementary diagrams...



System Landscape



Dynamic

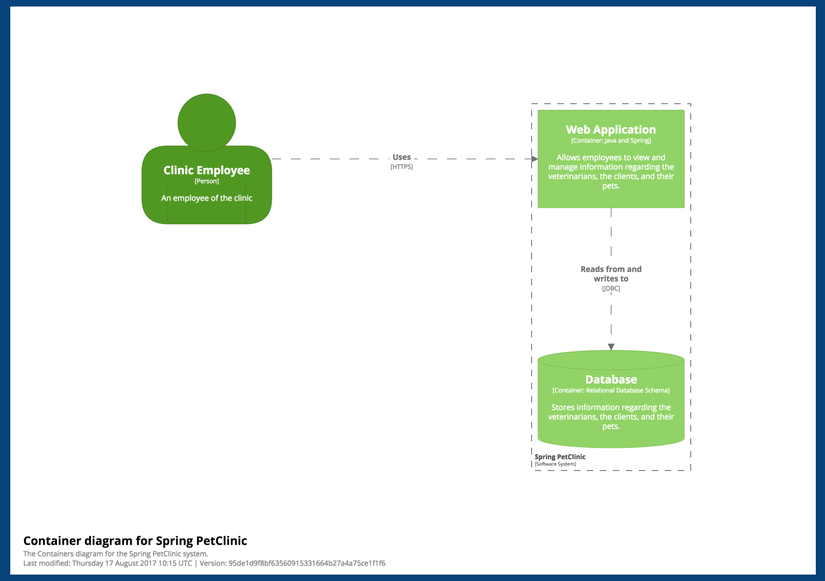


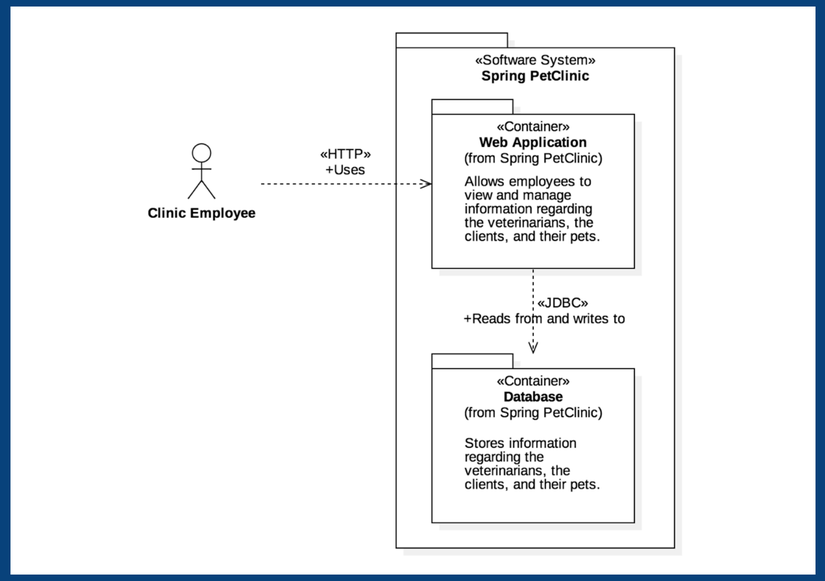
Deployment

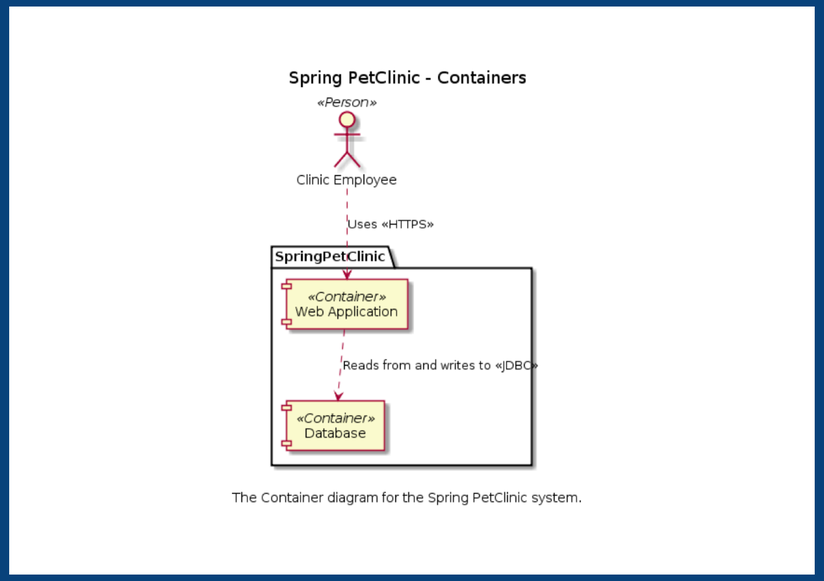
Abstractions first, notation second

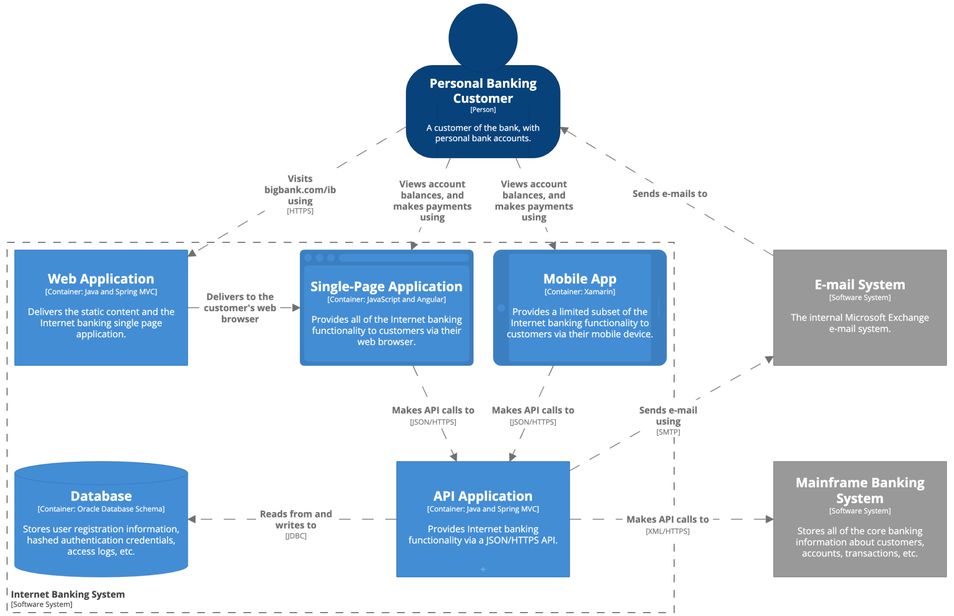
Ensure that your team has a ubiquitous language to describe software architecture

The C4 model is **notation independent**



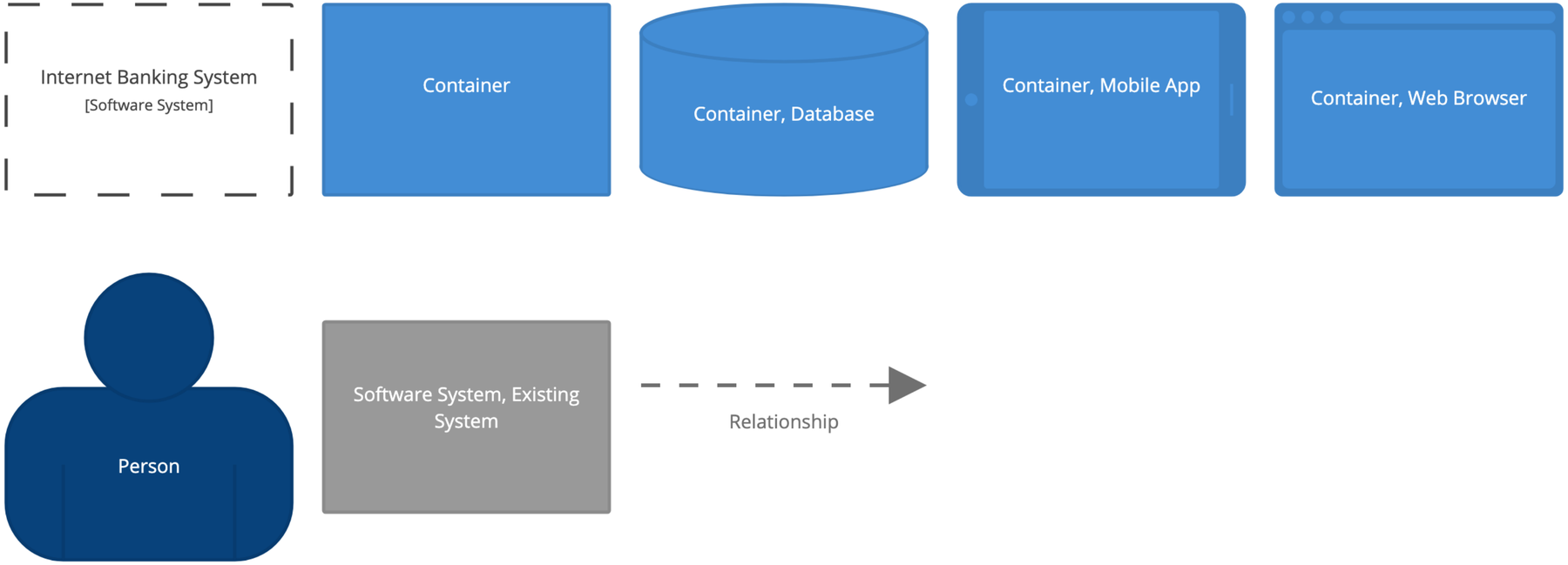






Container diagram for Internet Banking System

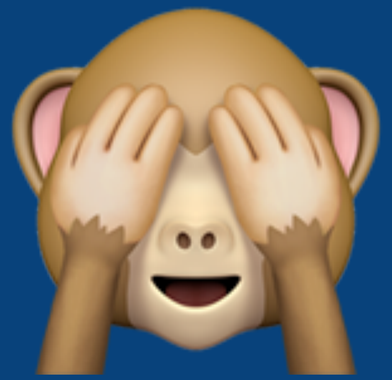
The container diagram for the Internet Banking System.
Workspace last modified: Tue Oct 29 2019 15:41:33 GMT+0000 (Greenwich Mean Time)



The lost art of software modelling?

How can we avoid copy-pasting elements across diagrams?

Stop using Visio!



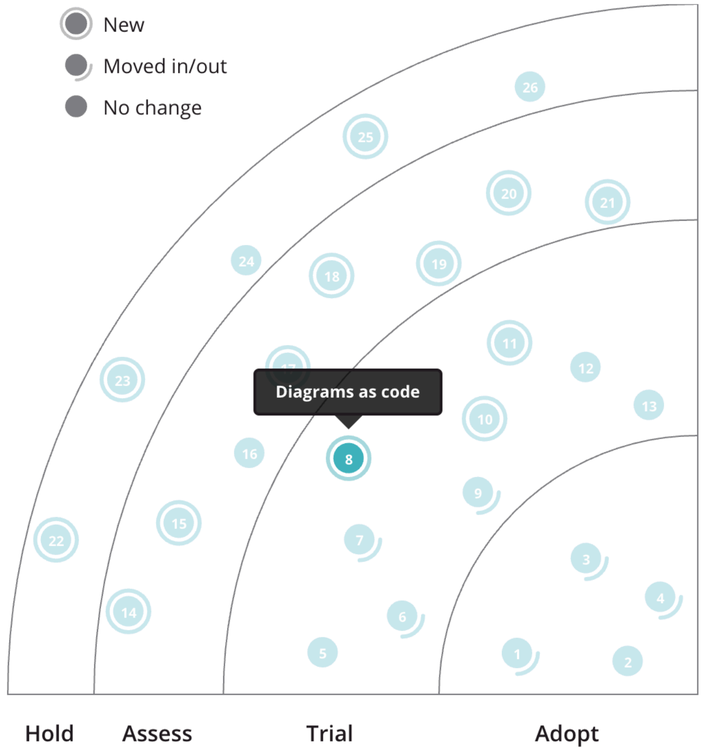
TECHNOLOGY RADAR

Techniques

Trial

5. Continuous delivery for machine learning (CD4ML)
6. Data mesh
7. Declarative data pipeline definition
8. **Diagrams as code**

We're seeing more and more tools that enable you to create software architecture and other **diagrams as code**. There are benefits to using these tools over the heavier alternatives, including easy version control and the ability to generate the DSLs from many sources. Tools in this space that we like include **Diagrams**, **Structurizr DSL**, **AsciiDoctor Diagram** and stables such as **WebSequenceDiagrams**, **PlantUML** and the venerable **Graphviz**. It's also fairly simple to generate your own SVG these days, so don't rule out quickly writing your own tool either.



Unable to find something you expected to see?

Each edition of the radar features blips reflecting what we came across during the previous six months. We might have covered

“Diagrams as code” is easy to author, diff, version control, collaborate on, integrate into CI/CD, etc

```
@startuml
title Software System - System Context

top to bottom direction

hide stereotype

rectangle "==User\n<size:10>[Person]</size>" <<User>> as User
rectangle "==Software System\n<size:10>[Software System]</size>" <<SoftwareSystem>> as SoftwareSystem

User ..> SoftwareSystem : "Uses"
@enduml
```

Domain language of diagramming

(no rules, no guidance)

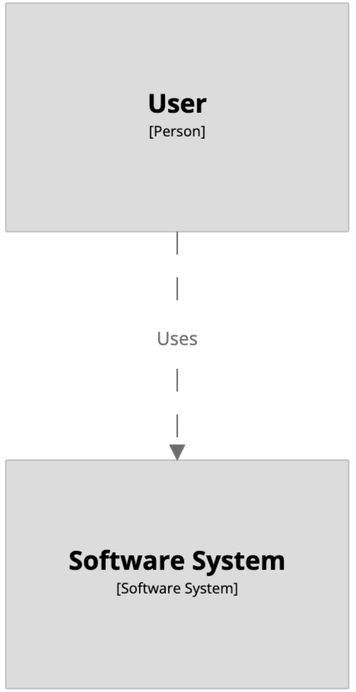
“Diagrams as code 2.0”
makes this model based,
separating content from presentation

```
workspace {

    model {
        user = person "User"
        softwareSystem = softwareSystem "Software System"

        user -> softwareSystem "Uses"
    }

    views {
        systemContext softwareSystem {
            include *
            autoLayout
        }
    }

}
```

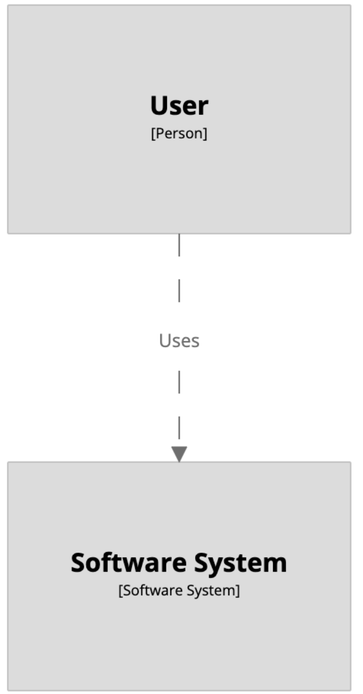
Domain language of software architecture

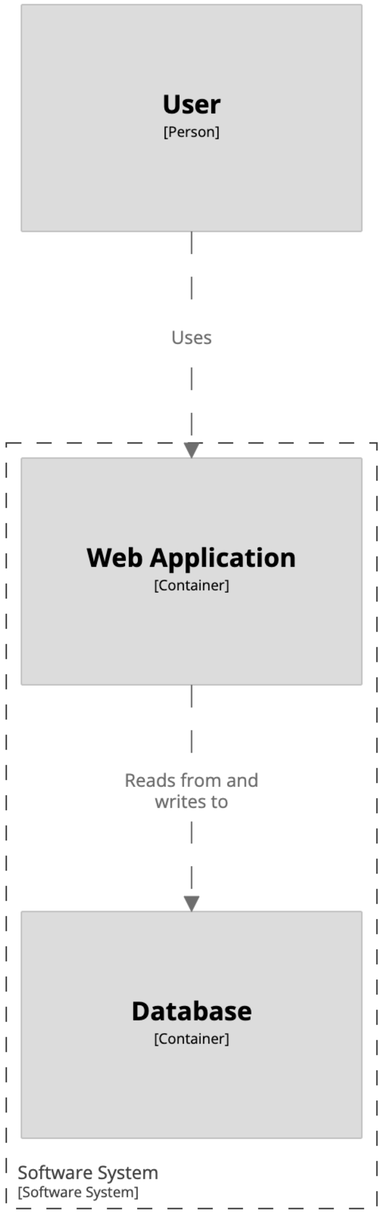
(metamodel and rules)

```
workspace {

    model {
        user = person "User"
        softwareSystem = softwareSystem "Software System"



        user -> softwareSystem "Uses"

    }

    views {
        systemContext softwareSystem {
            include *
            autoLayout
        }



    }

}
```

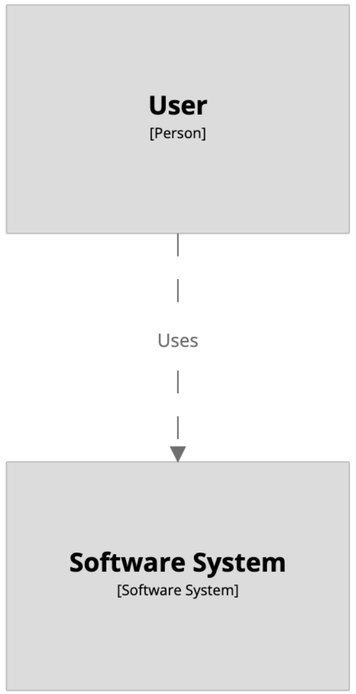


```
workspace {

    model {
        user = person "User"
        softwareSystem = softwareSystem "Software System" {
            webapp = container "Web Application"
            database = container "Database"
        }

        user -> webapp "Uses"
        webapp -> database "Reads from and writes to"
    }

    views {
        systemContext softwareSystem {
            include *
            autoLayout
        }

        container softwareSystem {
            include *
            autolayout
        }
    }

}
```

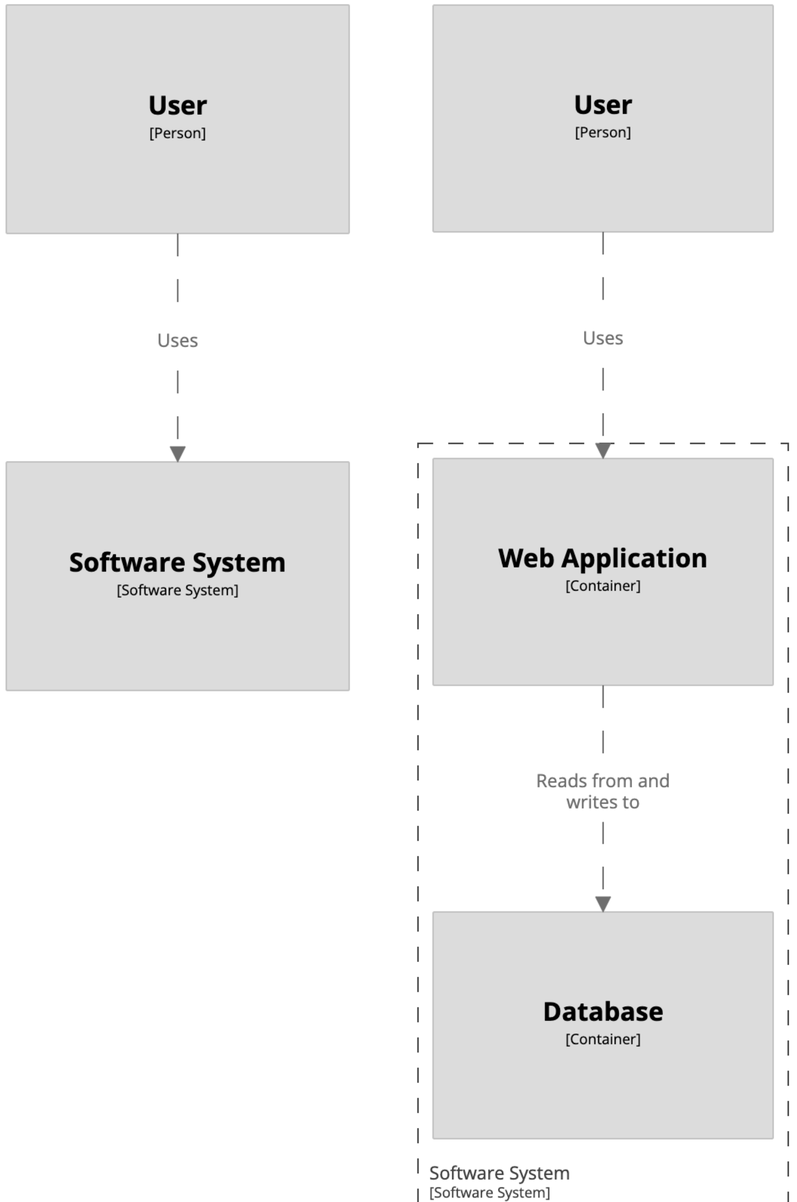




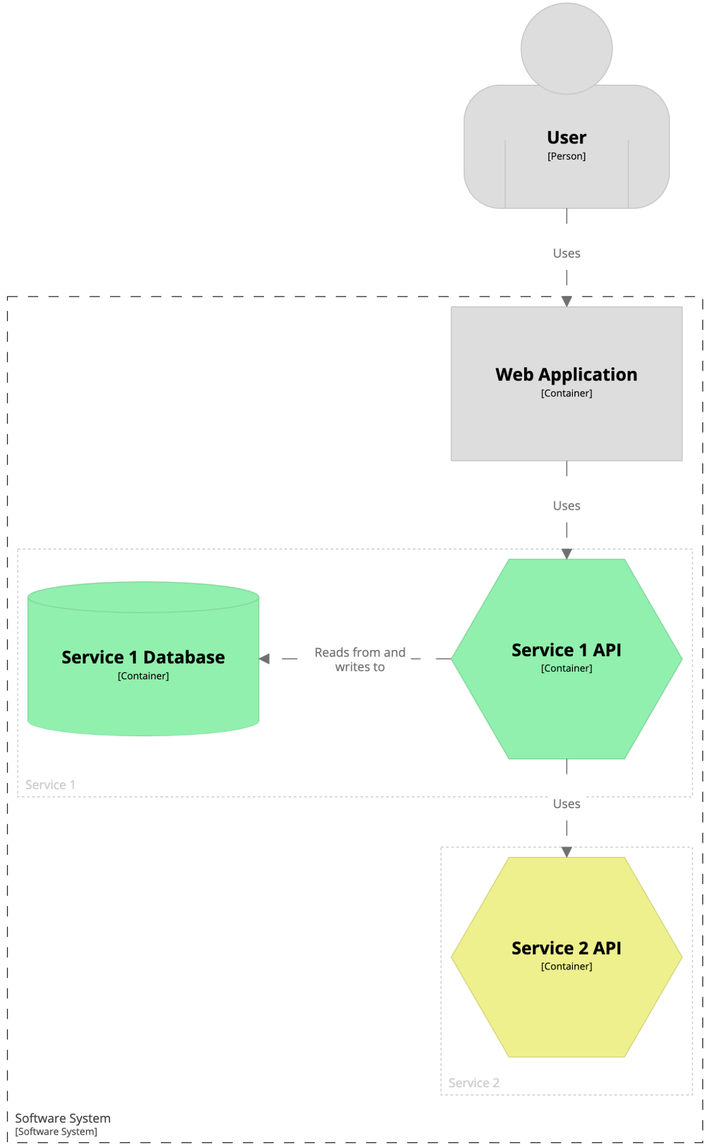
```
user -> softwareSystem "Uses"
```

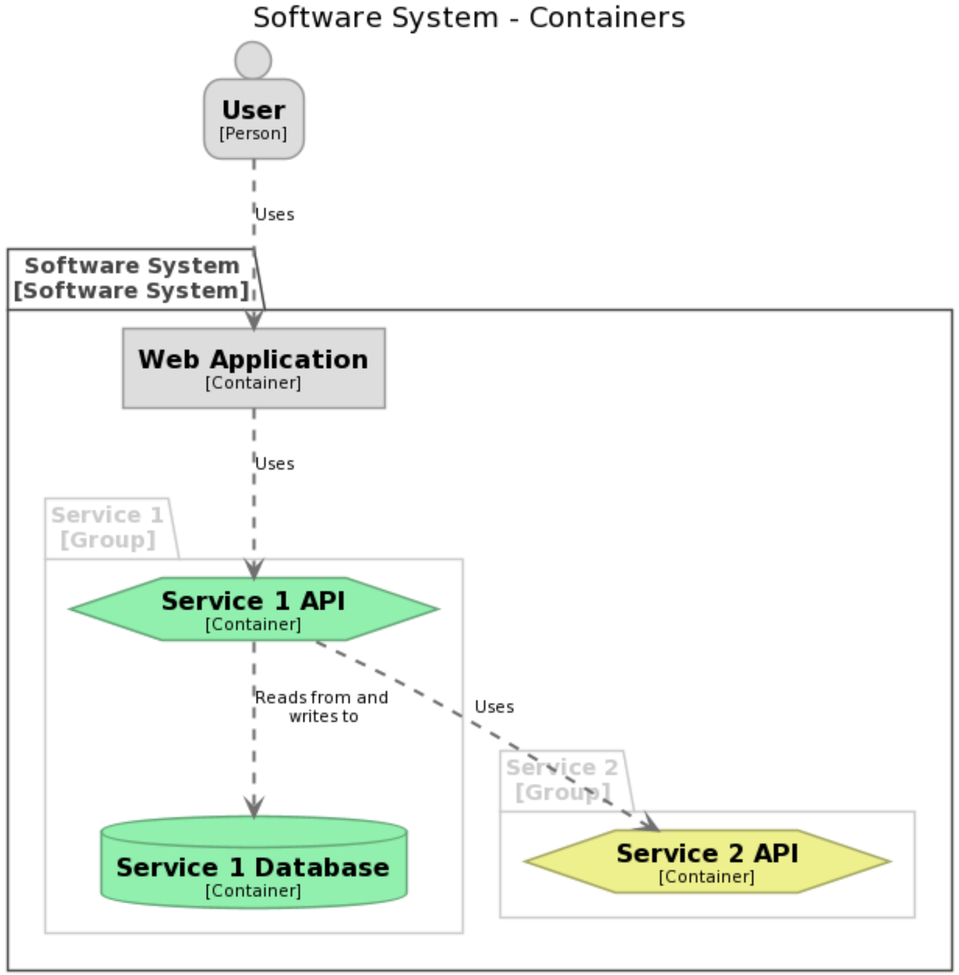



```
user -> webapp "Uses"
webapp -> database "Reads from and writes to"
```



```
container softwareSystem {
    include user ->service1->
    autolayout
}
```





How can we avoid our diagrams becoming out of sync when we make changes to our code?

Authoring tool

Create diagrams as code (Java, .NET, TypeScript, Python, PHP, etc) or text (DSL, YAML) via a number of different authoring tools.

Creates

Workspace

A workspace is the wrapper for a software architecture model and views, described using the C4 model and an open JSON data format.

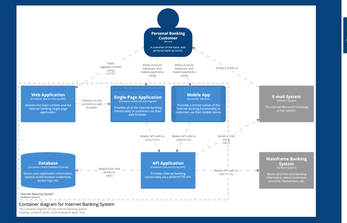
Renders

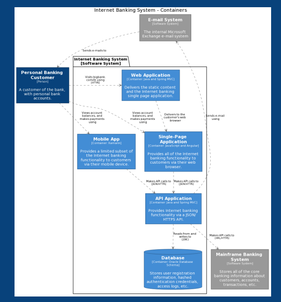
Rendering tool

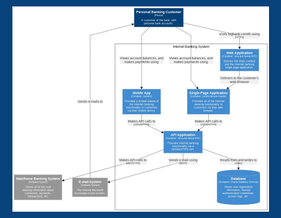
Render views using multiple diagramming tools and formats (Structurizr cloud service/on-premises installation/Lite, PlantUML, Mermaid, WebSequenceDiagrams, Ilograph, etc).

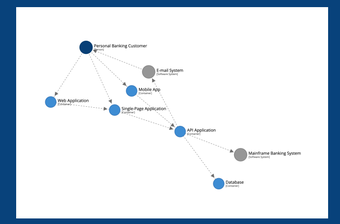
```
private static final long WORKSPACE_ID = 25441;
private static final String API_KEY = "";
private static final String API_SECRET = "";

public static void main(String[] args) throws Exception {
    // all software architecture models belong to a workspace
    Workspace workspace = new Workspace("Getting Started", "This is a model of my software system.");
    Model model = workspace.getModel();

    // create a model to describe a user using a software system
    Person user = model.addPerson("User", "A user of my software system.");
    SoftwareSystem softwareSystem = model.addSoftwareSystem("Software System", "My software system.");
    user.uses(softwareSystem, "Uses");

    // create a system context diagram showing people and software systems
    ViewSet views = workspace.getViews();
    SystemContextView contextView = views.createSystemContextView(softwareSystem, "SystemContext", "An example of
    contextView.addAllSoftwareSystems();
    contextView.addAllPeople();

    // add some styling to the diagram elements
    Styles styles = views.getConfiguration().getStyles();
    styles.addElementStyle(Tags.SOFTWARE_SYSTEM).background("#1168bd").color("#ffffff");
    styles.addElementStyle(Tags.PERSON).background("#08427b").color("#ffffff").shape(Shape.Person);

    // upload to structurizr.com (you'll need your own workspace ID, API key and API secret)
    StructurizrClient structurizrClient = new StructurizrClient(API_KEY, API_SECRET);
    structurizrClient.putWorkspace(WORKSPACE_ID, workspace);
}
```

```
workspace "Getting Started" "This is a model of my software system." {

    model {
        user = person "User" "A user of my software system."
        softwareSystem = softwareSystem "Software System" "My software system."

        user -> softwareSystem "Uses"
    }

    views {
        systemContext softwareSystem "SystemContext" "An example of a System Context diagram." {
            include *
        }

        styles {
            element "Software System" {
                background #1168bd
                color #ffffff
            }
            element "Person" {
                shape person
                background #08427b
                color #ffffff
            }
        }
    }

}
```









Consumes

Custom tool

Your own tooling to parse the model and views; for integration into other rendering tools, dashboards, service catalogs, etc.

Structurizr

Diagramming tools are still the first choice for most teams, but some are starting to adopt modelling tools to improve consistency and enable diagram automatic generation

Abstractions first, notation second

Thank you!

Simon Brown
@simonbrown