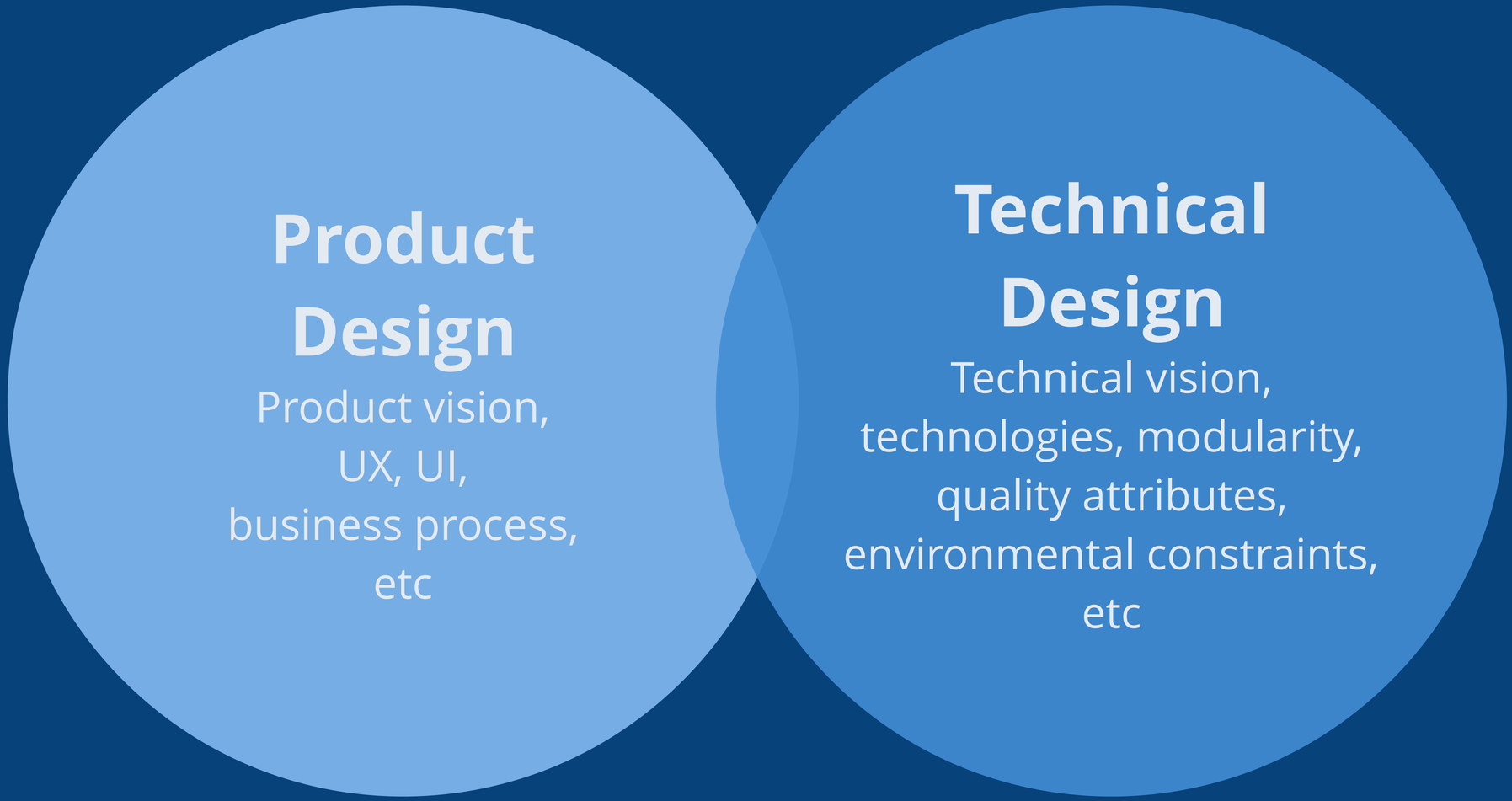
The lost art of software design

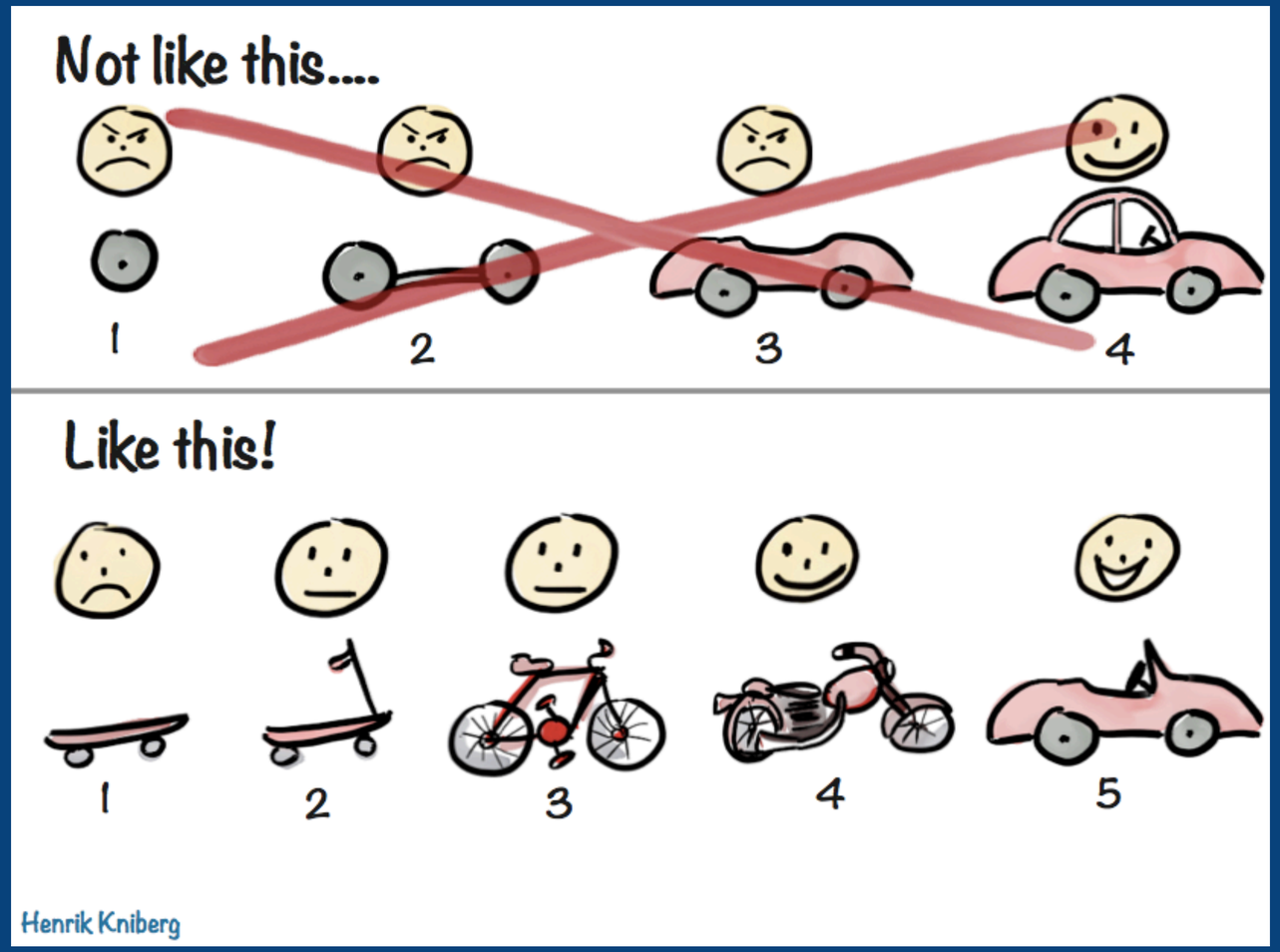
@simonbrown

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

Dave Thomas



I’m referring to **technical design** rather than product design



Iteration (via prototyping and experimentation) is great for product design but...

you don’t just “build the car”

How to draw an owl

1.

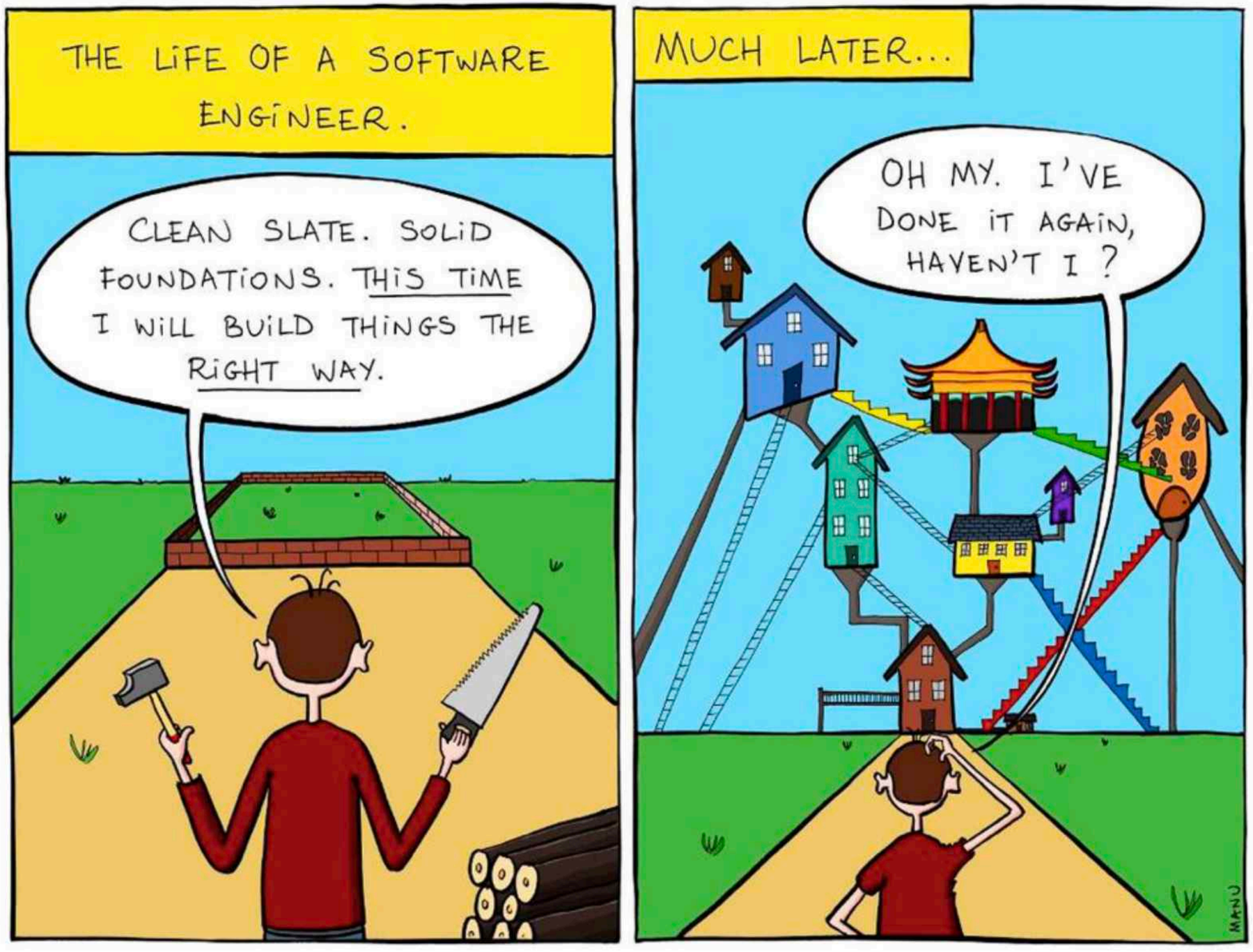


2.



1. Draw some circles

2. Draw the rest of the 🤬🤬🤬🤬 owl


THE LIFE OF A SOFTWARE ENGINEER.
CLEAN SLATE. SOLID FOUNDATIONS. THIS TIME I WILL BUILD THINGS THE RIGHT WAY.
MUCH LATER...
OH MY. I'VE DONE IT AGAIN, HAVEN'T I ?
MANU

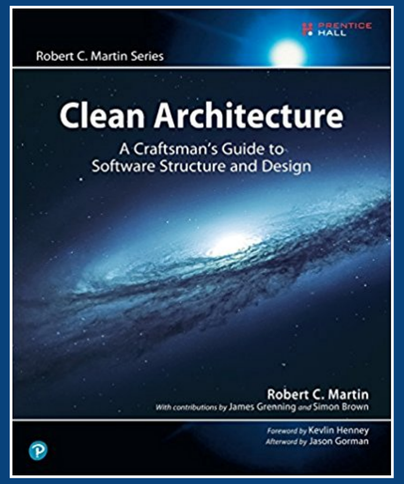
Simon Brown

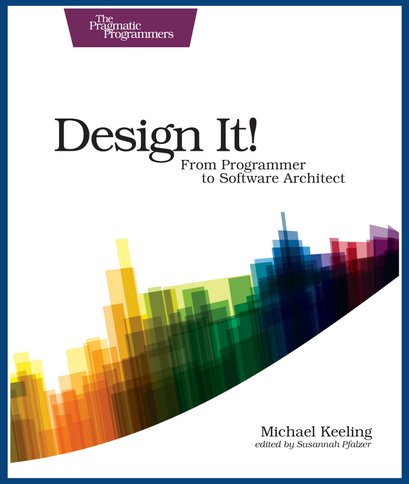
Independent consultant specialising in software architecture

@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.

CC BY NC SA

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

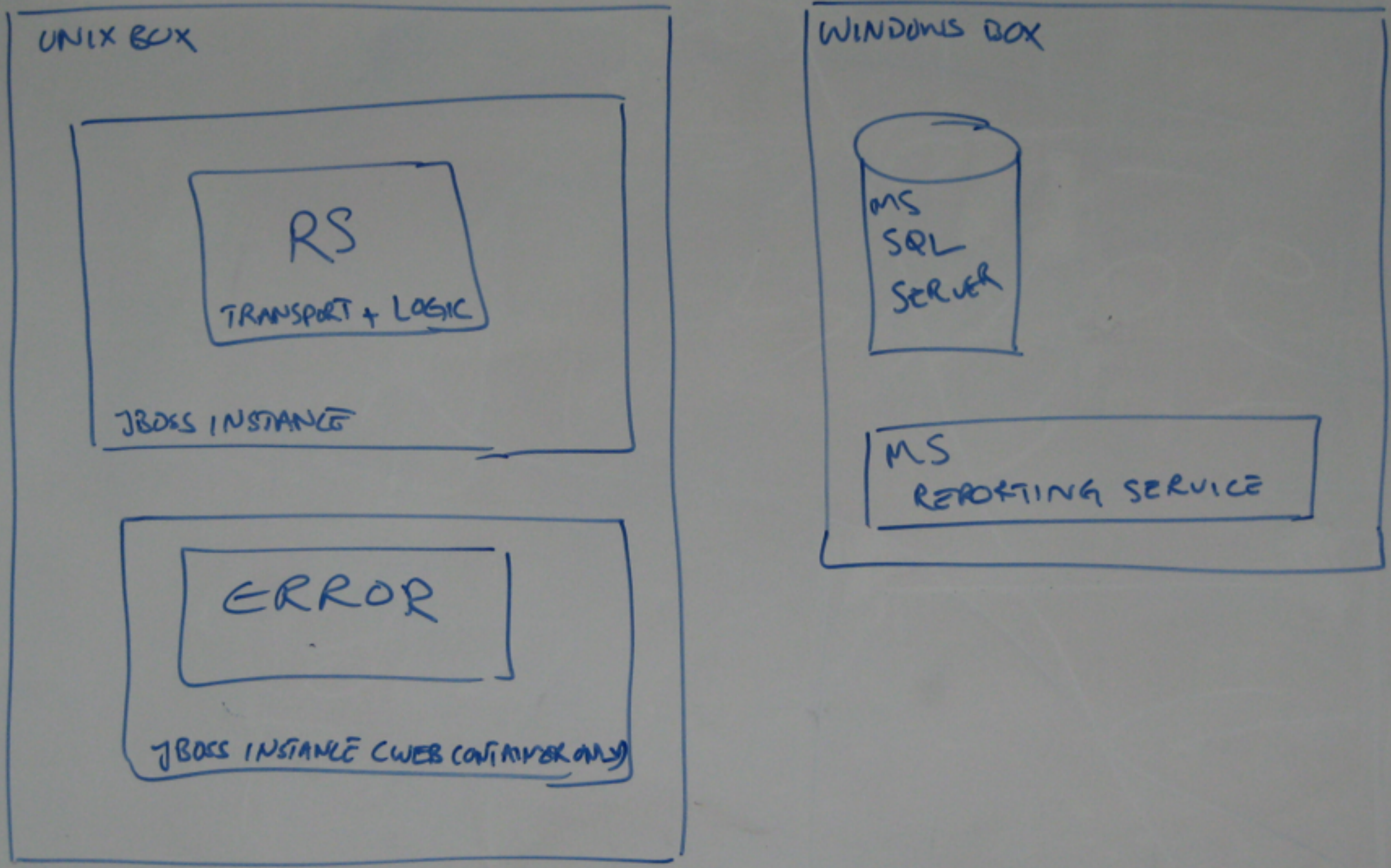
- Interfaces with existing data systems should conform to and use existing data formats.

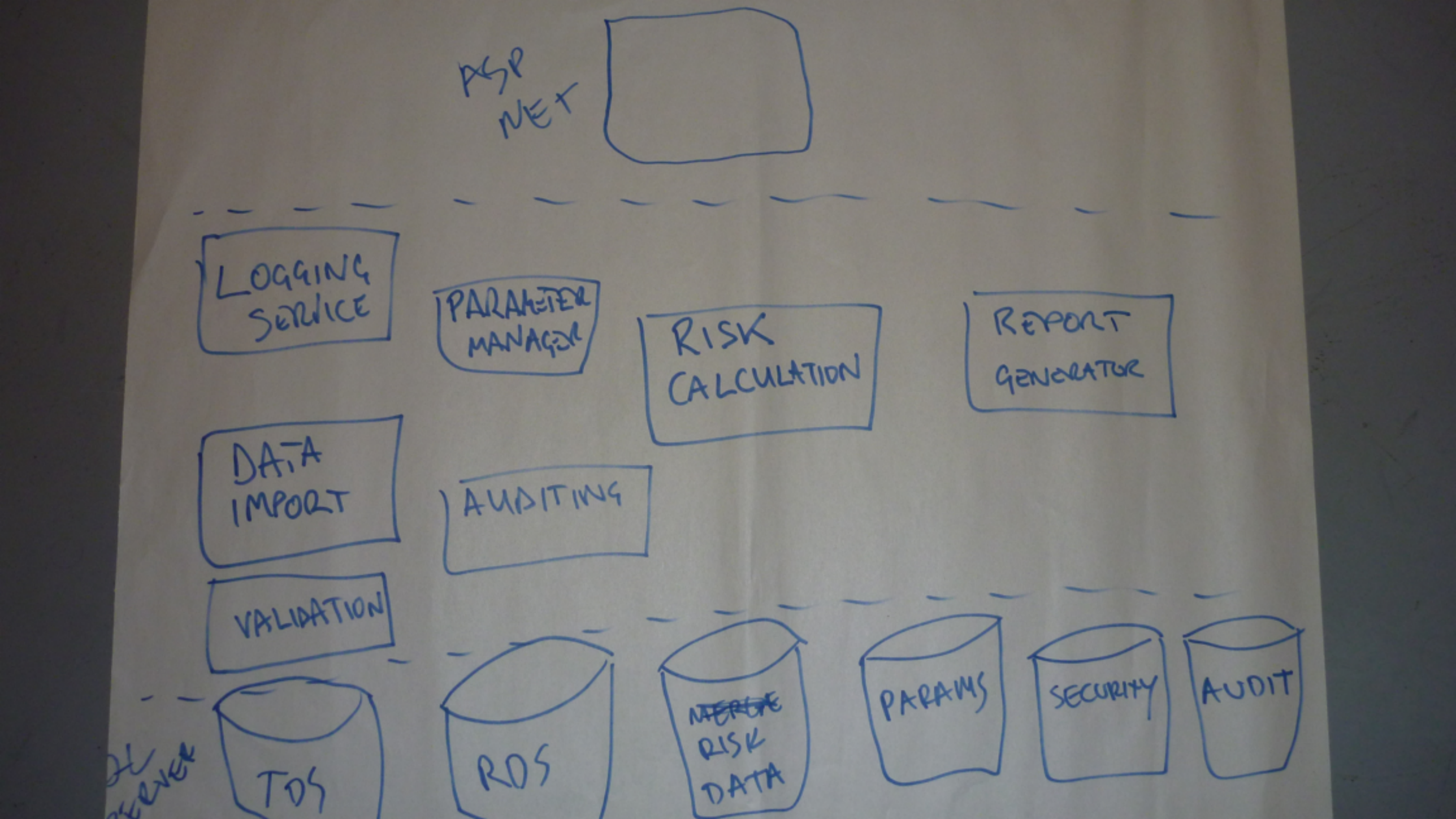
“Financial Risk System” architecture kata
Simon Brown | @simonbrown

CC BY NC SA

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

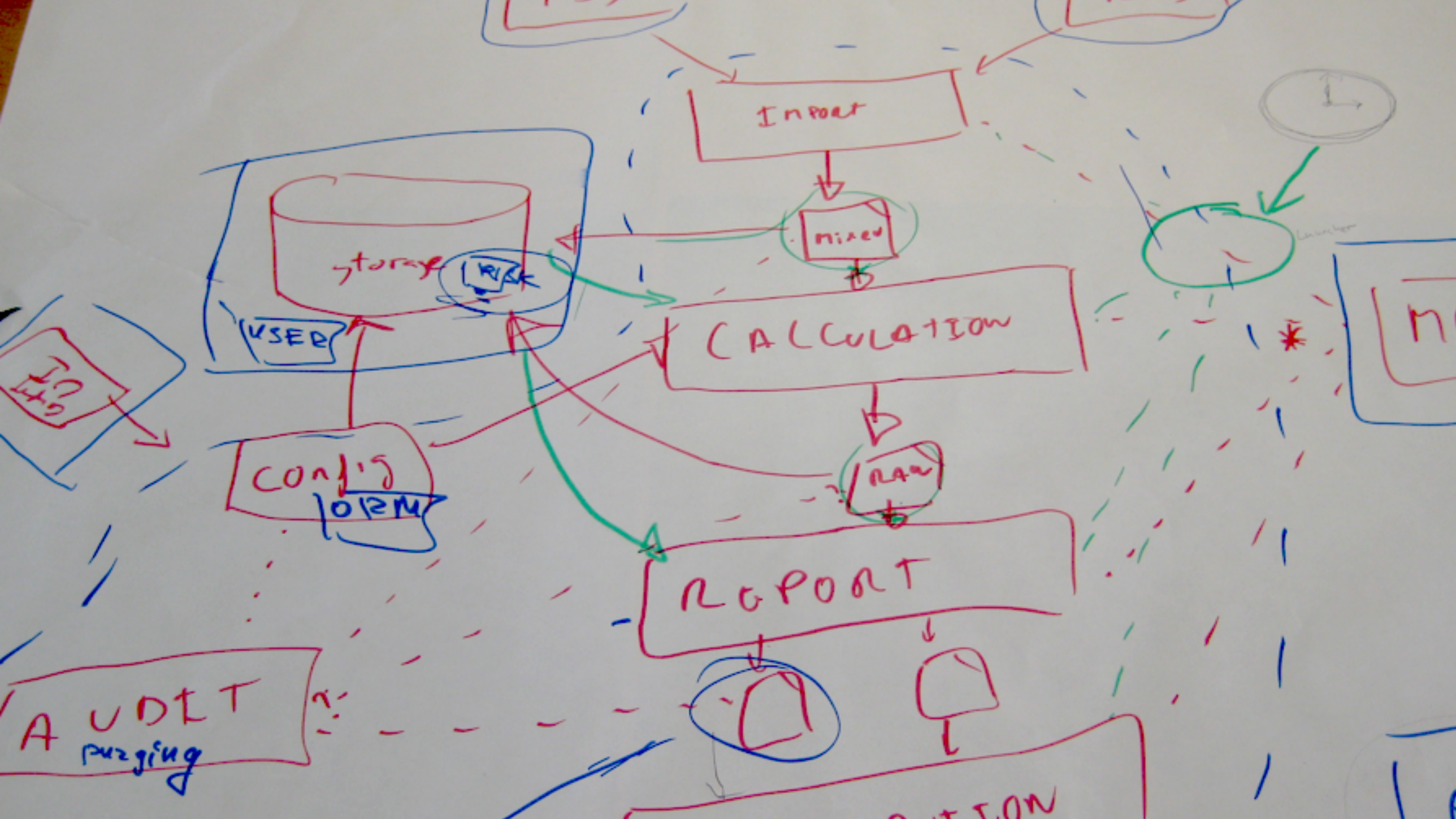
60-90 minutes

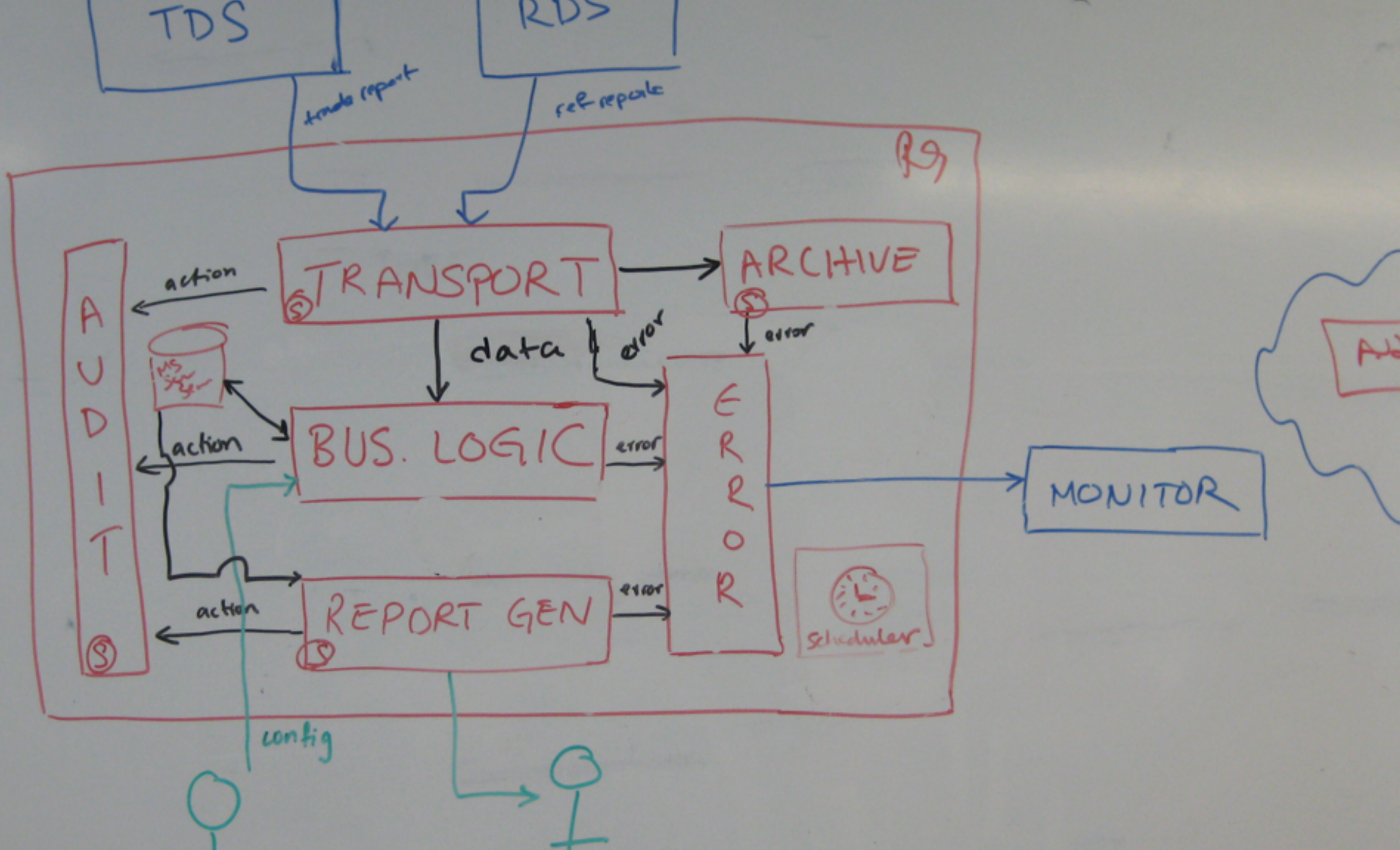

UNIX BOX
RS
TRANSPORT + LOGIC
JBOSS INSTANCE
ERROR
JBOSS INSTANCE (WEB CONTAINER ONLY)
WINDOWS BOX
MS
SQL
SERVER
MS
REPORTING SERVICE

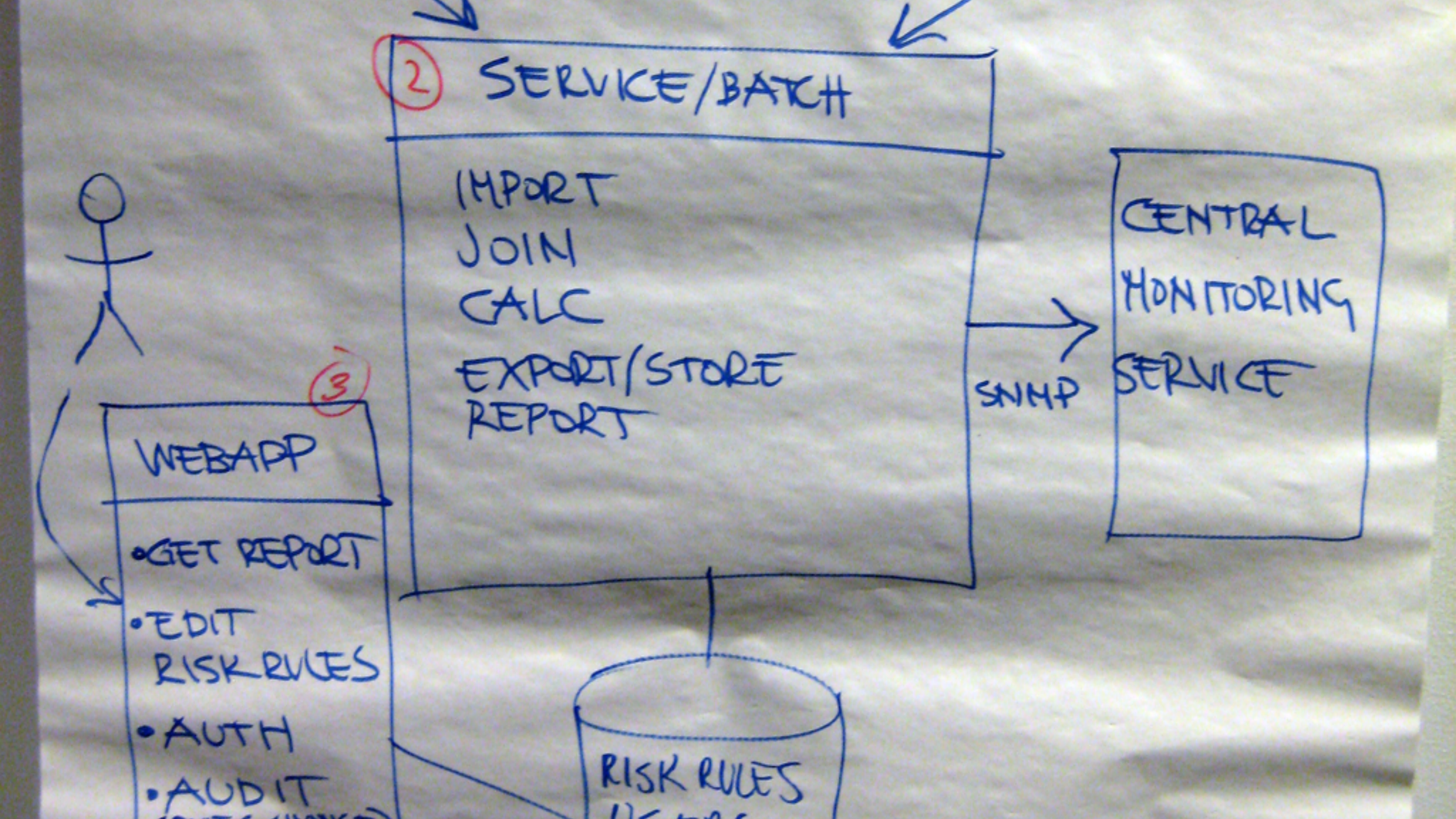

ASP NET
LOGGING SERVICE
PARAMETER MANAGER
RISK CALCULATION
REPORT GENERATOR
DATA IMPORT
AUDITING
VALIDATION
TDS
RDS
RISK DATA
PARAMS
SECURITY
AUDIT

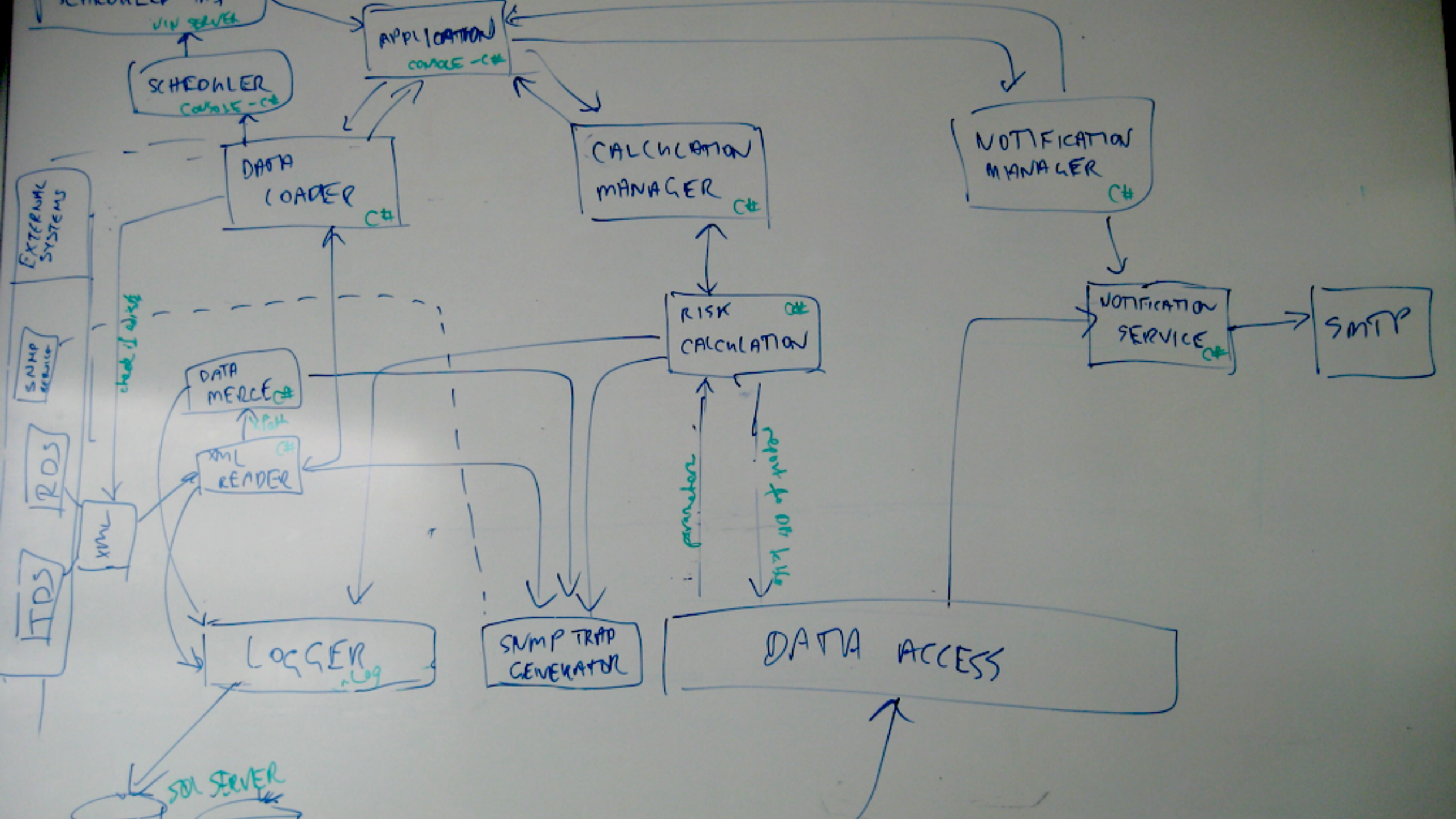
FUNCTIONAL VIEW

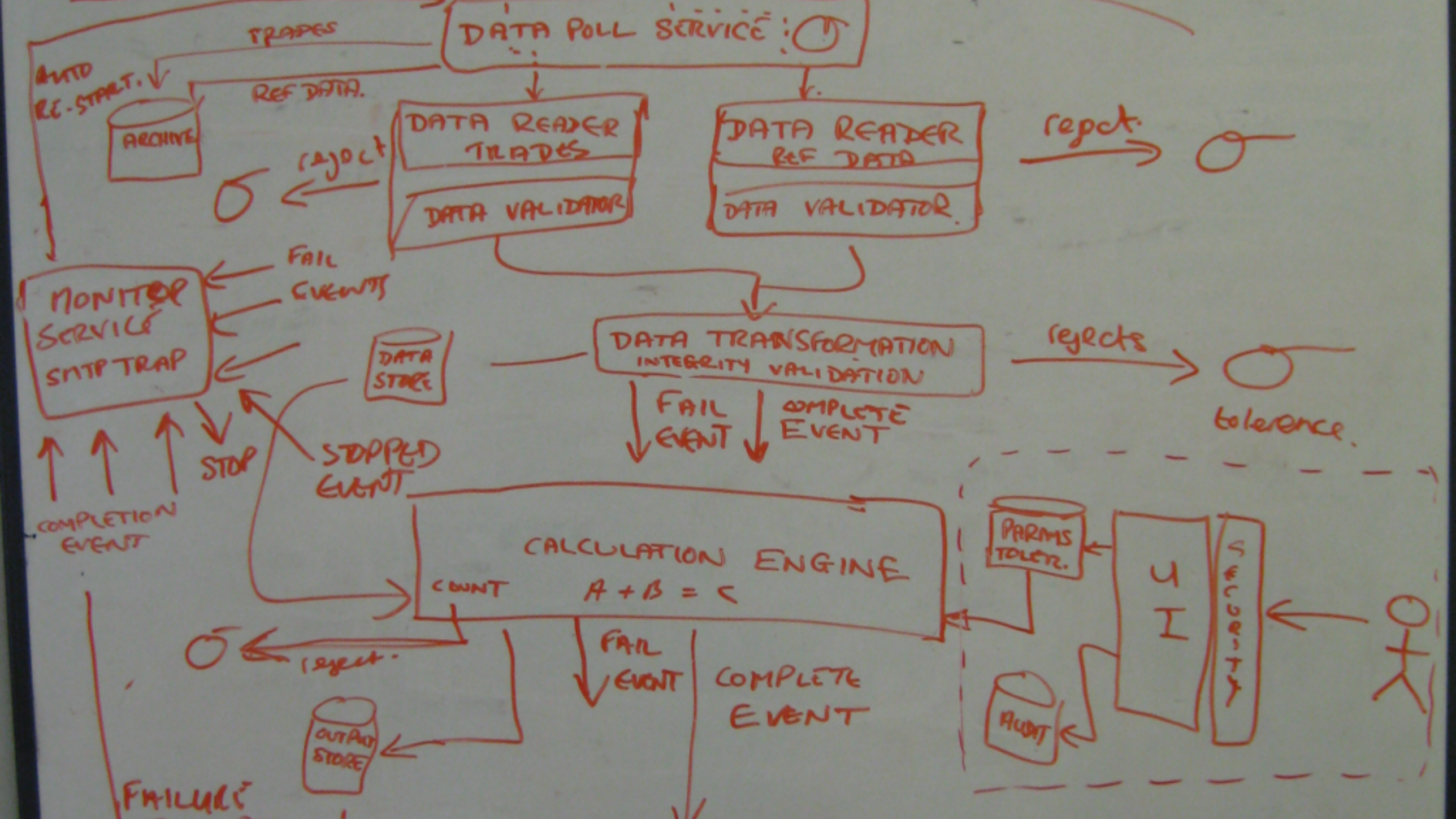


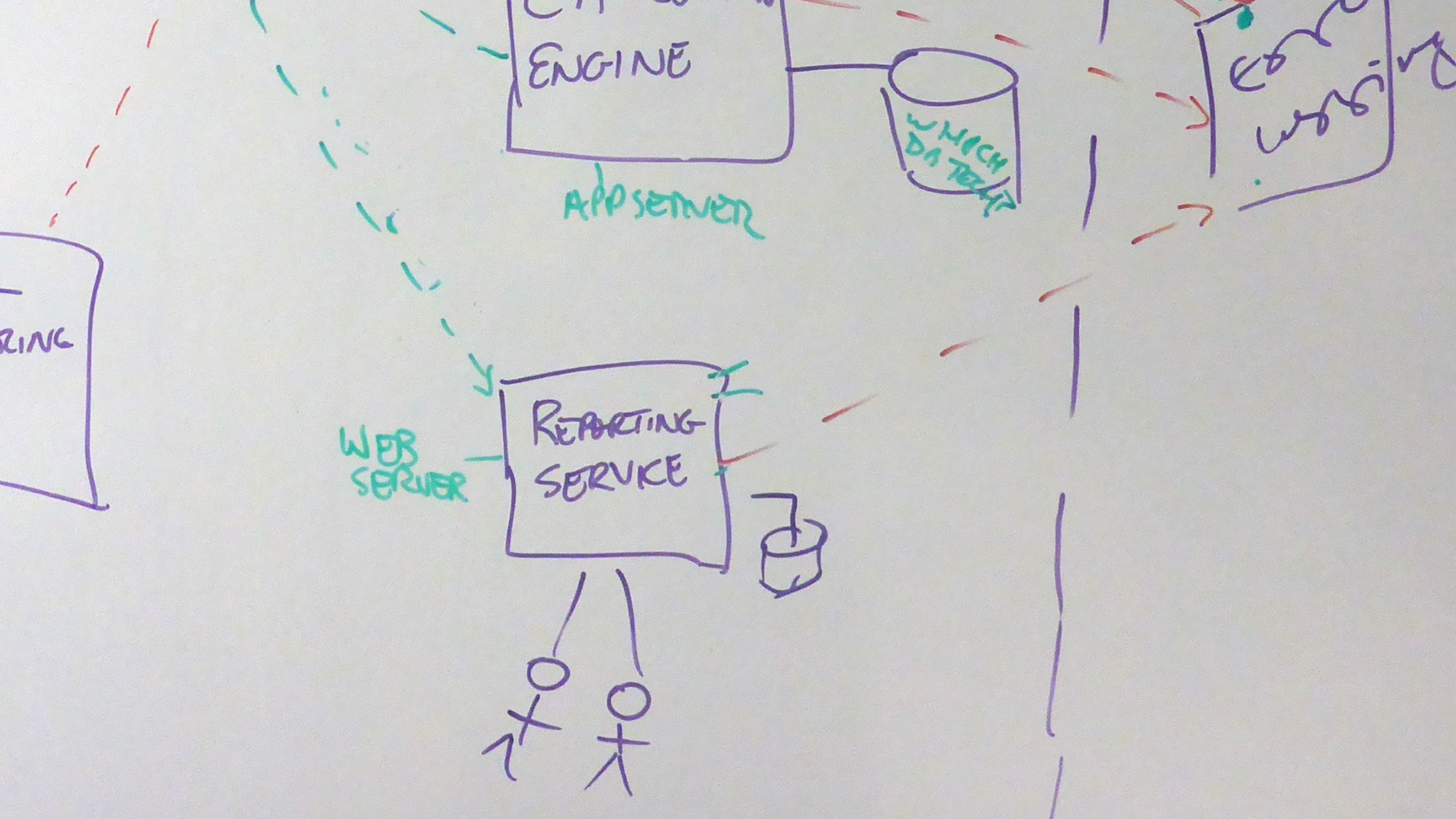

Import
storage
risk
USER
config
mixed
CALCULATION
plan
REPORT
AUDIT
purging


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

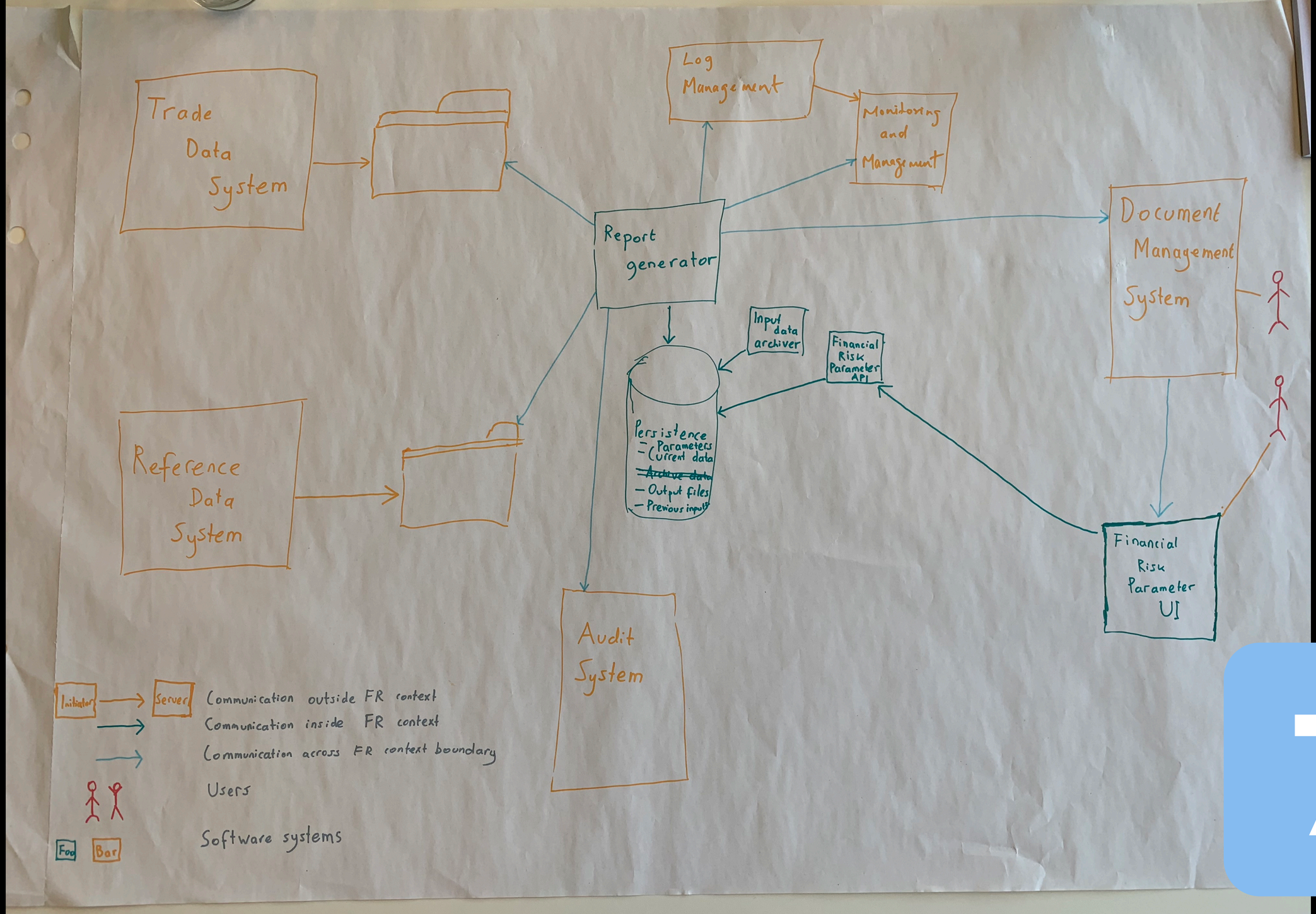

2
SERVICE/BATCH
IMPORT
JOIN
CALC
EXPORT/STORE
REPORT
SNMP
CENTRAL
MONITORING
SERVICE
3
WEBAPP
•GET REPORT
•EDIT
RISK RULES
•AUTH
•AUDIT
RISK RULES


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

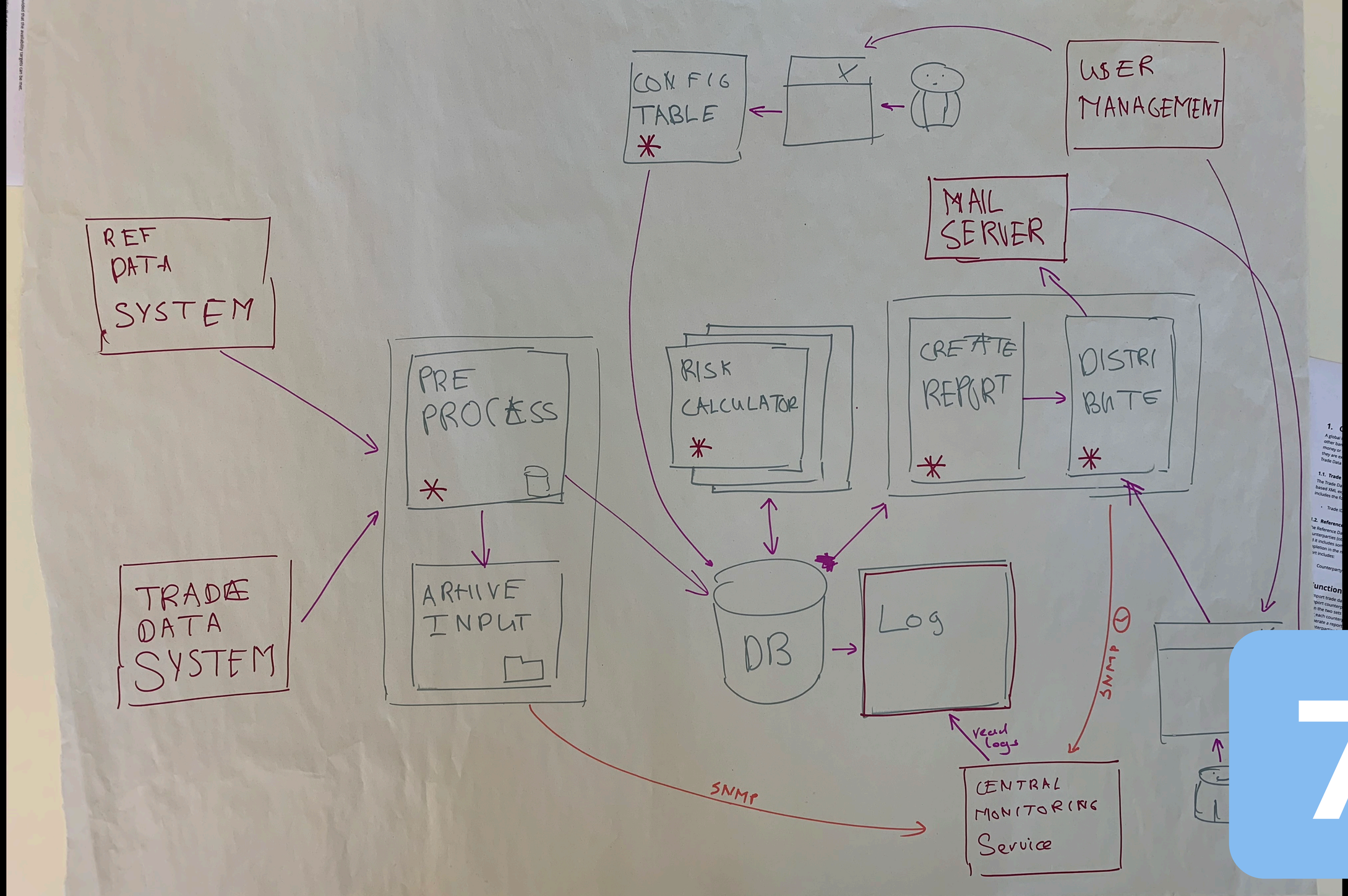

DATA POLL SERVICE
TRADES
REF DATA.
AUTO RE-START.
ARCHIVE
DATA READER TRADES
DATA VALIDATOR
reject
DATA READER REF DATA
DATA VALIDATOR
reject
MONITOR SERVICE SMTP TRAP
FAIL EVENTS
DATA STORE
DATA TRANSFORMATION
INTEGRITY VALIDATION
rejects
tolerance
FAIL EVENT
COMPLETE EVENT
STOP
STOPPED EVENT
COMPLETION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

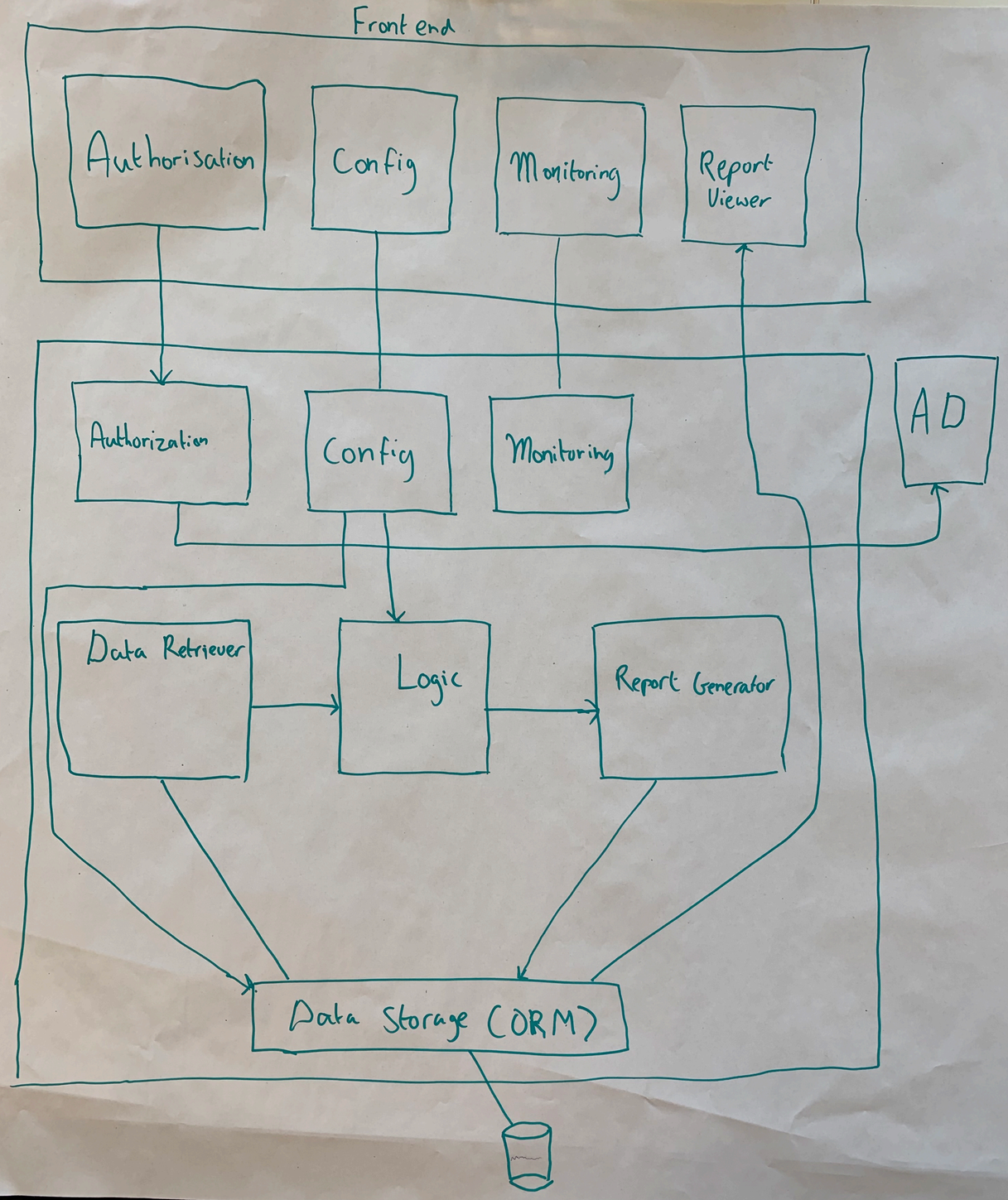

Params
Calcs
EH?
Batch
Date
- RiskCal
- calcs
- Par


Trade Data System
Log Management
Monitoring and Management
Report generator
Document Management System
Input data archiver
Financial Risk Parameter API
Persistence
- Parameters
- Current data
- Archive data
- Output files
- Previous inputs
Reference Data System
Audit System
Financial Risk Parameter UI
Initiator
Server
Communication outside FR context
Communication inside FR context
Communication across FR context boundary
Users
Foo
Bar
Software systems

7


CONFIG TABLE
X
USER MANAGEMENT
MAIL SERVER
REF DATA SYSTEM
PRE PROCESS
RISK CALCULATOR
CREATE REPORT
DISTRI BUTE
TRADE DATA SYSTEM
ARHIVE INPUT
DB
Log
SNMP
read logs
SNMP
CENTRAL MONITORING Service

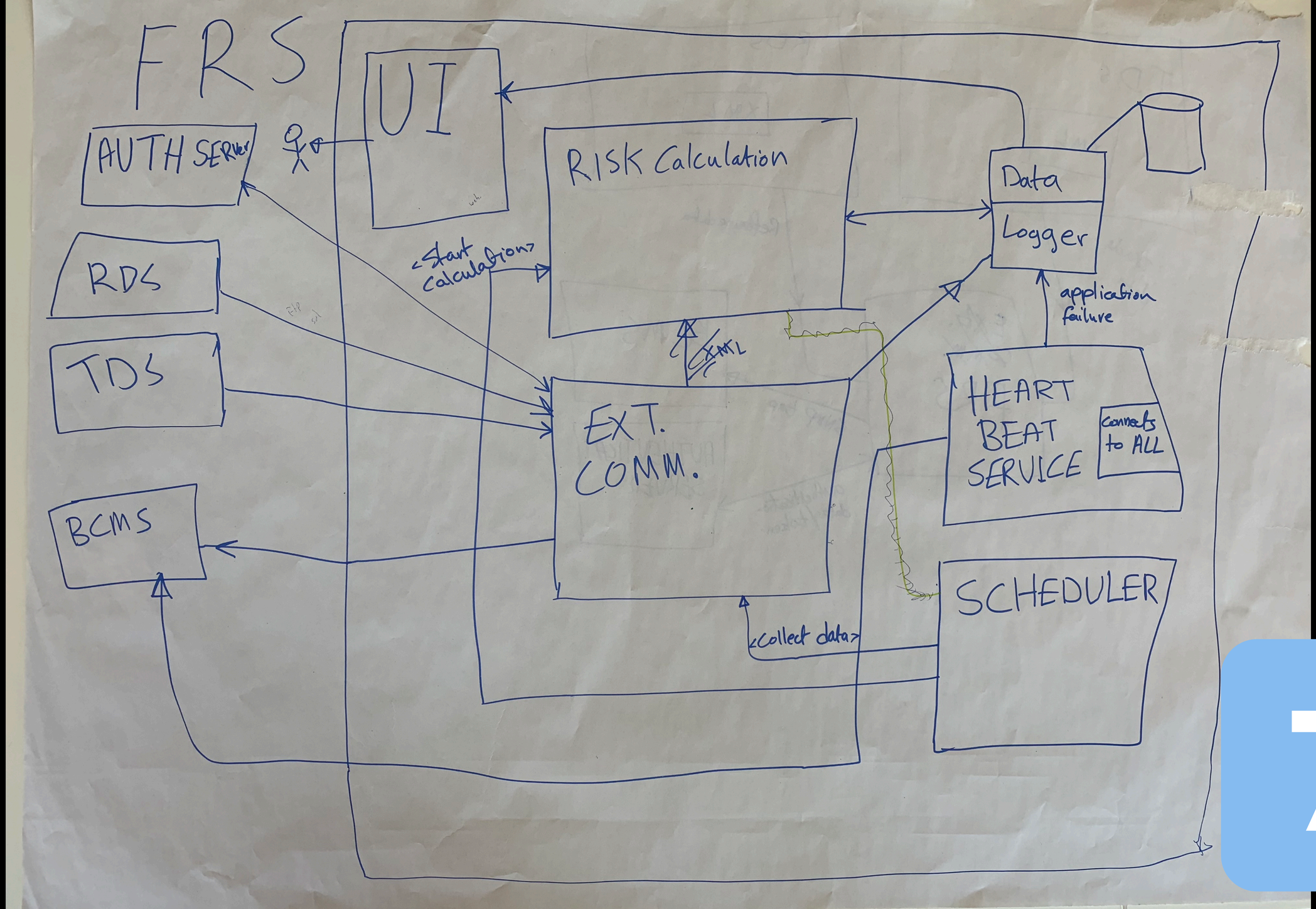
7

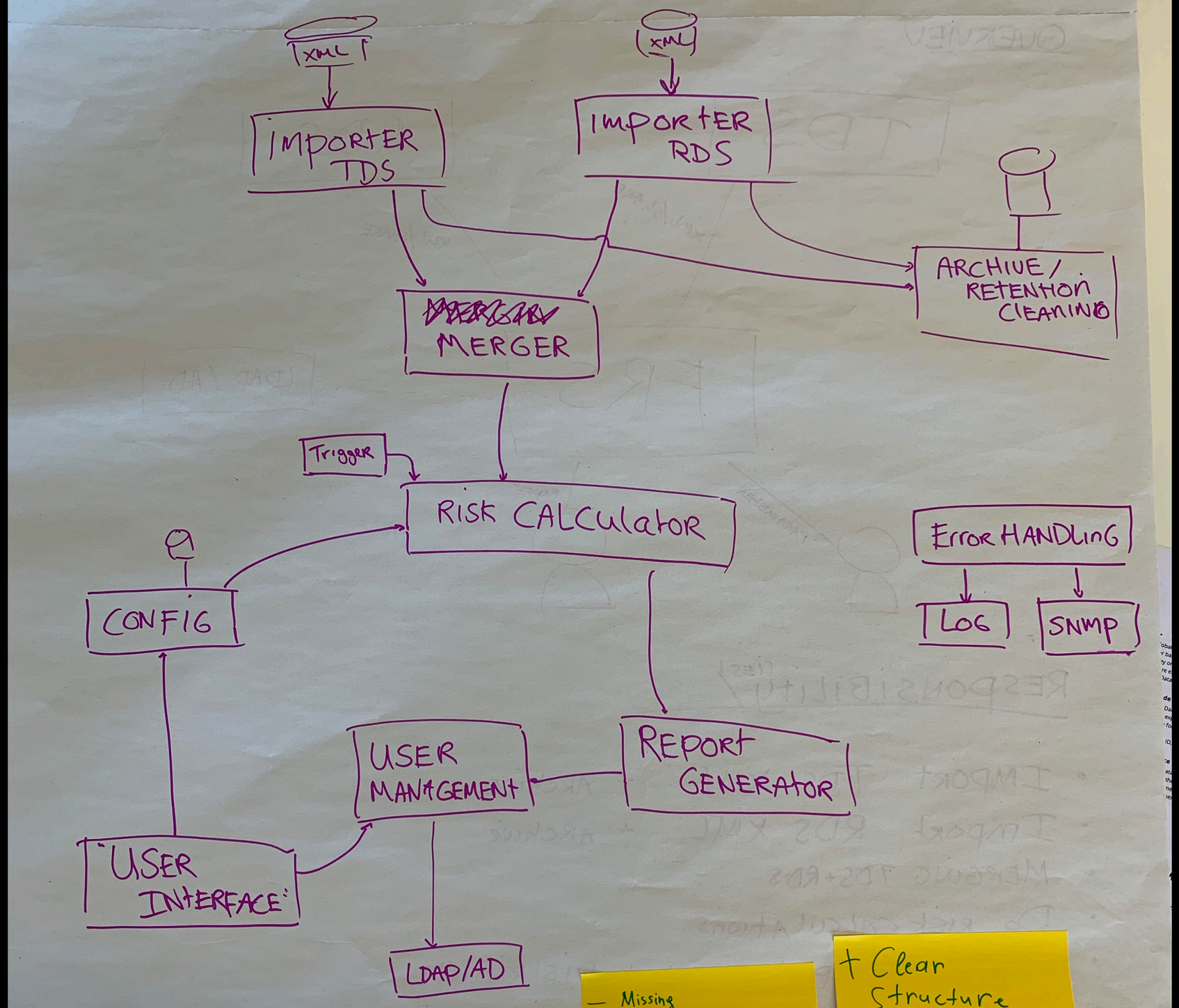


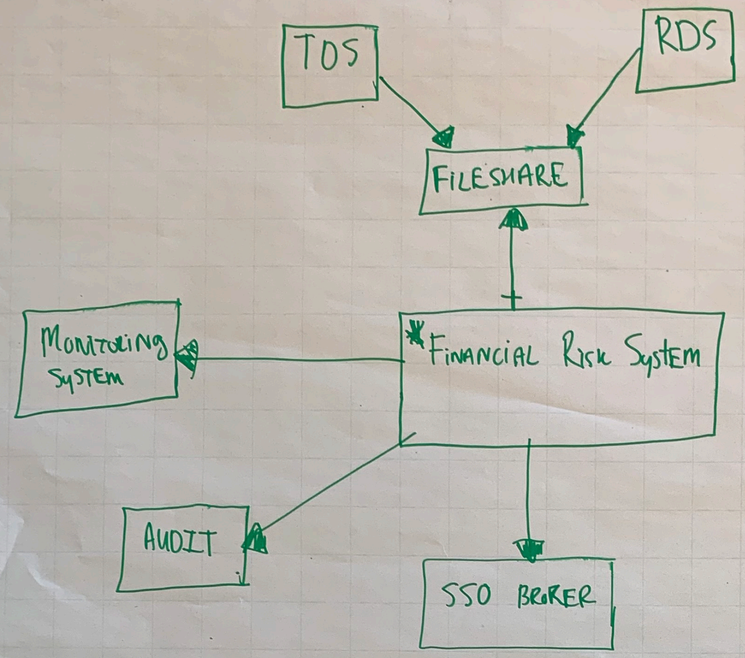
Significant decisions

- F/E <> B/E
- Make use of OS' watchdog mechanism
- Data storage ORM-framework: Entity
- ASP .NET B/E
- Angular F/E

7


FRS
UI
AUTH SERVER
RISK Calculation
Data
Logger
RDS
TDS
<start calculation>
XML
application failure
EXT. COMM.
HEART BEAT SERVICE
connects to ALL
BCMS
SCHEDULER
<collect data>


XML
XML
IMPORTER TDS
IMPORTER RDS
ARCHIVE / RETENTION CLEANING
MERGER
Trigger
RISK CALCULATOR
ERROR HANDLING
CONFIG
LOG
SNMP
USER MANAGEMENT
REPORT GENERATOR
USER INTERFACE
LDAP/AD
+ Clear Structure

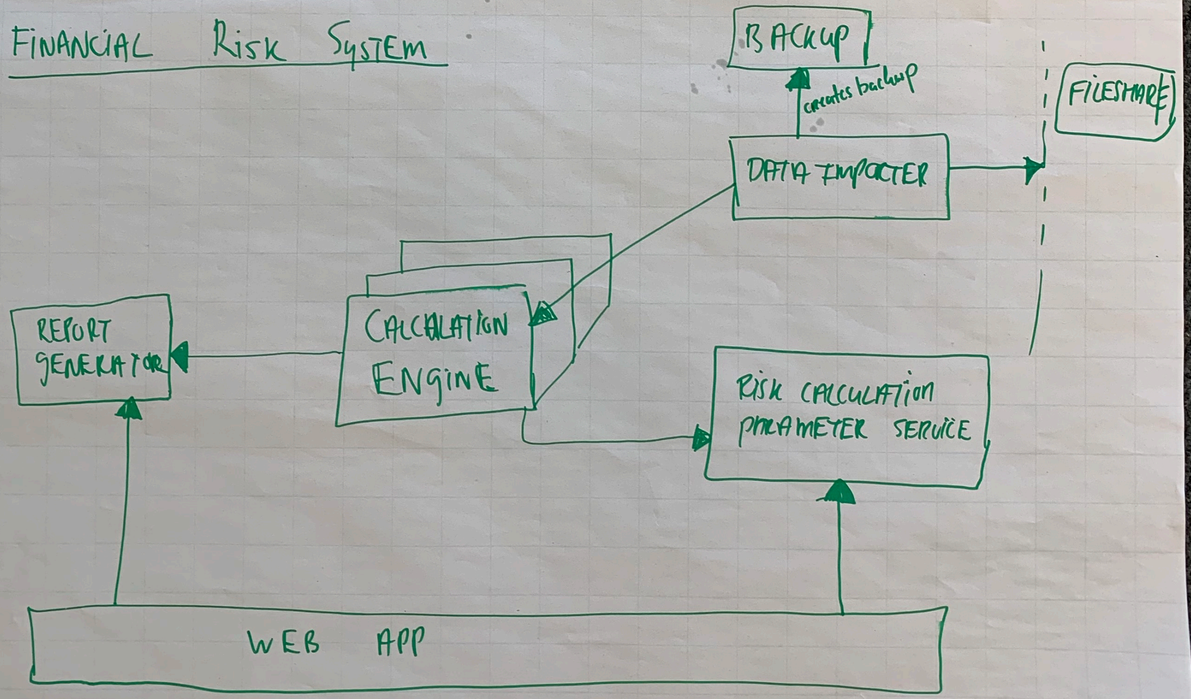


BACKUP

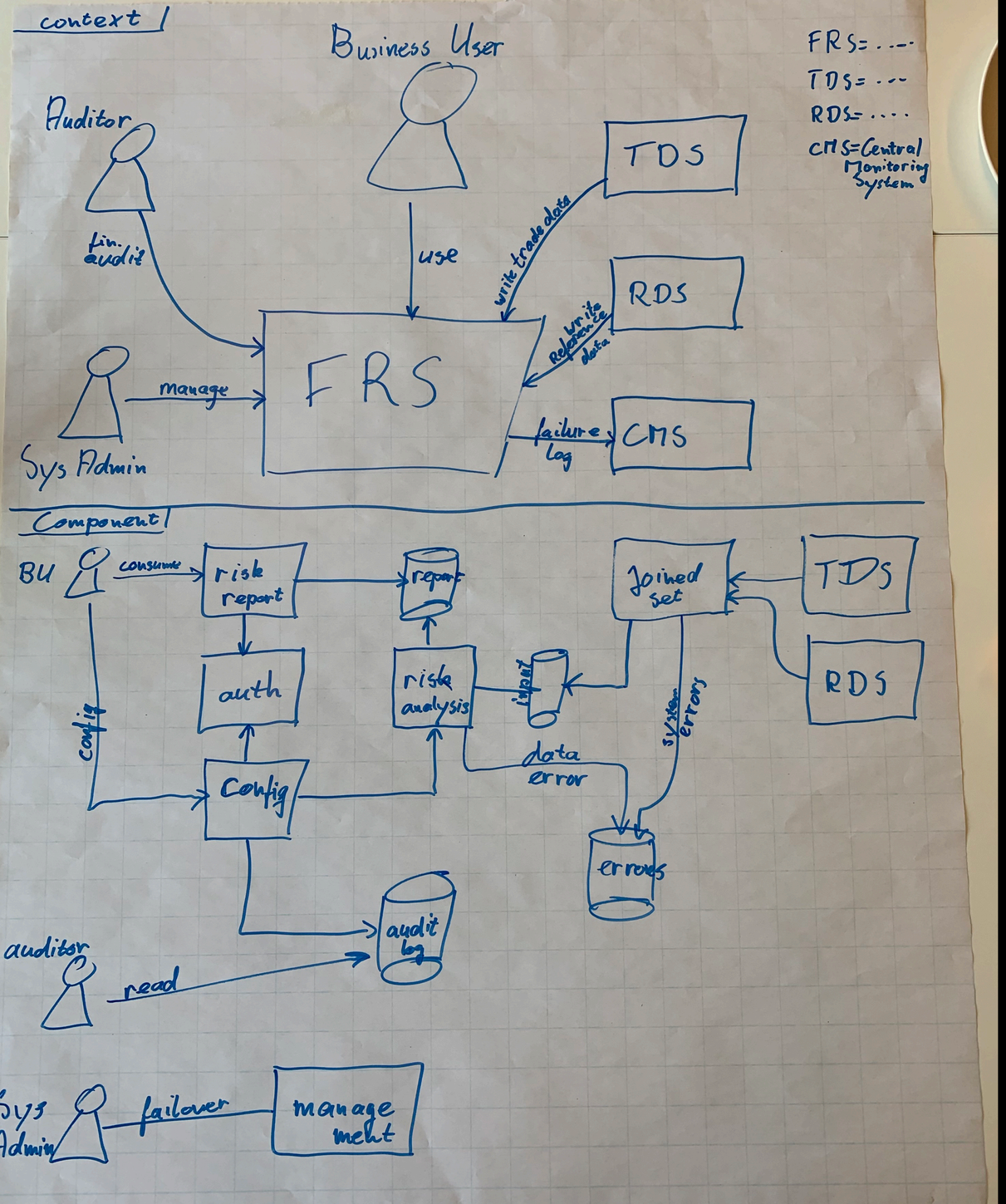
DECISIONS:

* CALCULATIONS ARE A JOB TRIGGERED ON SCHEDULE
* EXECUTE CALCULATIONS IN PARALLEL FOR EACH COUNTERPARTY
* WEBUI FOR VIEWING REPORTS AND MODIFYING RISK PARAMETERS.
* AUTHENTICATE AND AUTHORIZE USERS BASED ON SSO
* SINGLE POINT OF ENTRY

* FINANCIAL RISK SYSTEM



7


context
Business User
Auditor
fin. audit
use
write trade data
TDS
RDS
write Reference data
FRS
manage
Sys Admin
failure log
CMS
FRS= ...
TDS= ...
RDS= ....
CMS=Central Monitoring System
Component
BU
consume
risk report
report
joined set
TDS
RDS
auth
risk analysis
input
config
Config
data error
system errors
errors
audit log
auditor
read
Sys Admin
failover
manage ment

What’s wrong these diagrams?

Swap and **review** your diagrams

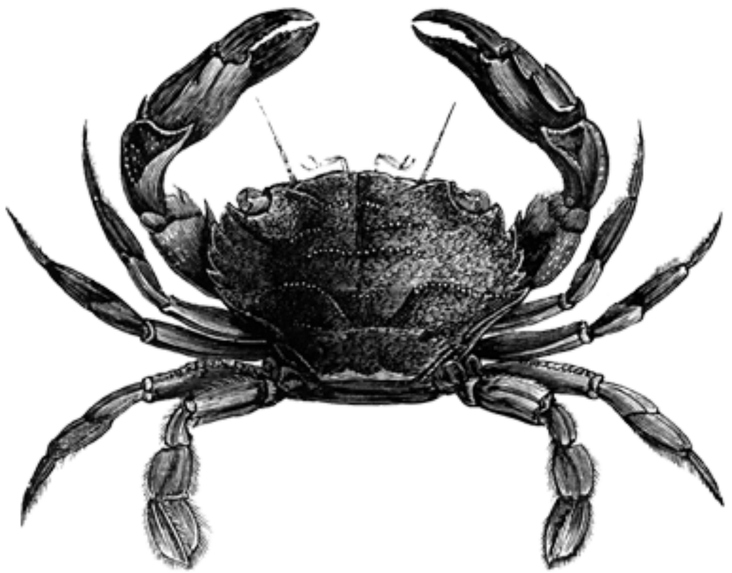
1. Do the solutions satisfy the architectural drivers?
2. If you were the bank, would you buy this solution?



10 minutes

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

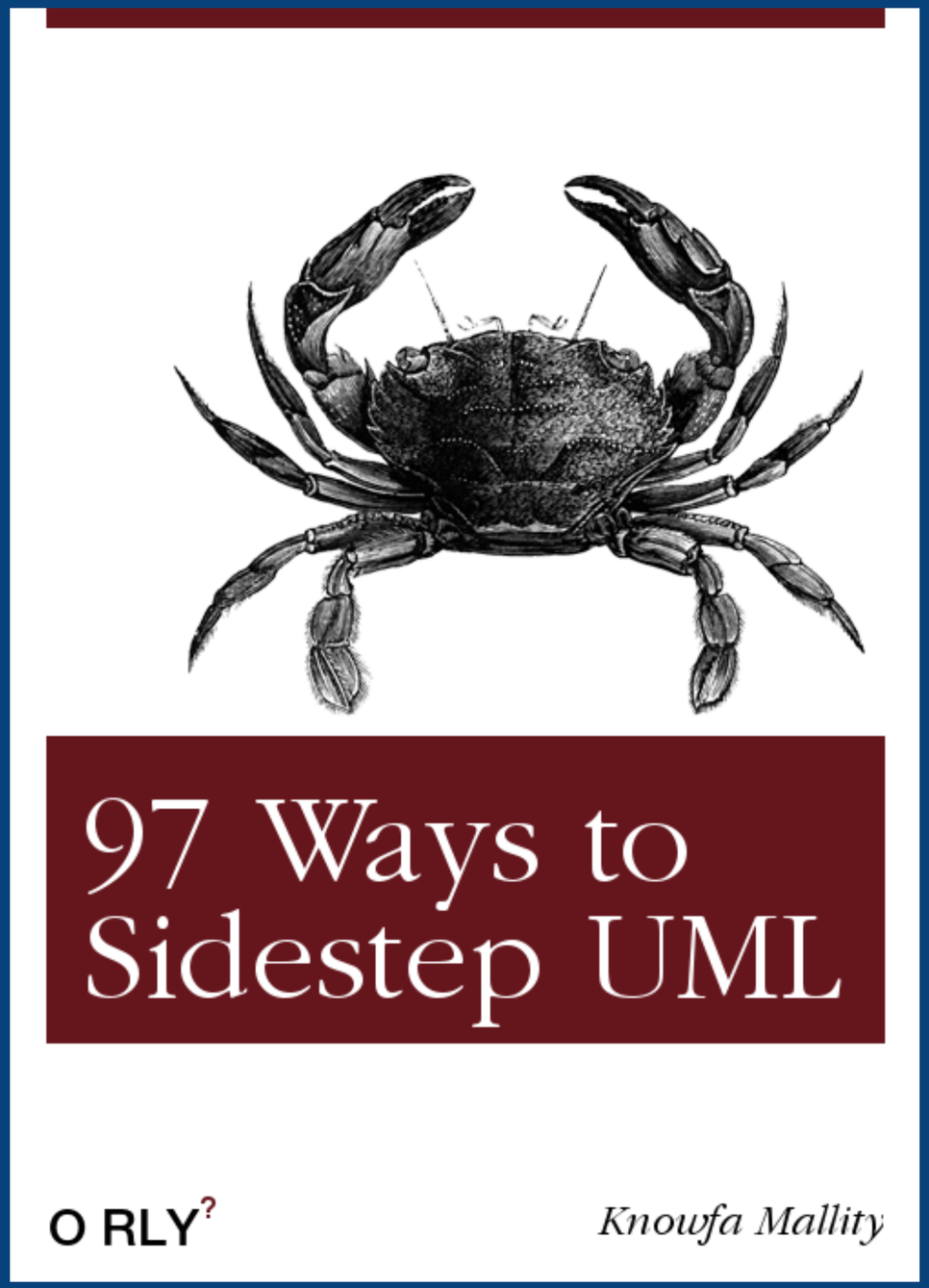
UML



97 Ways to Sidestep UML

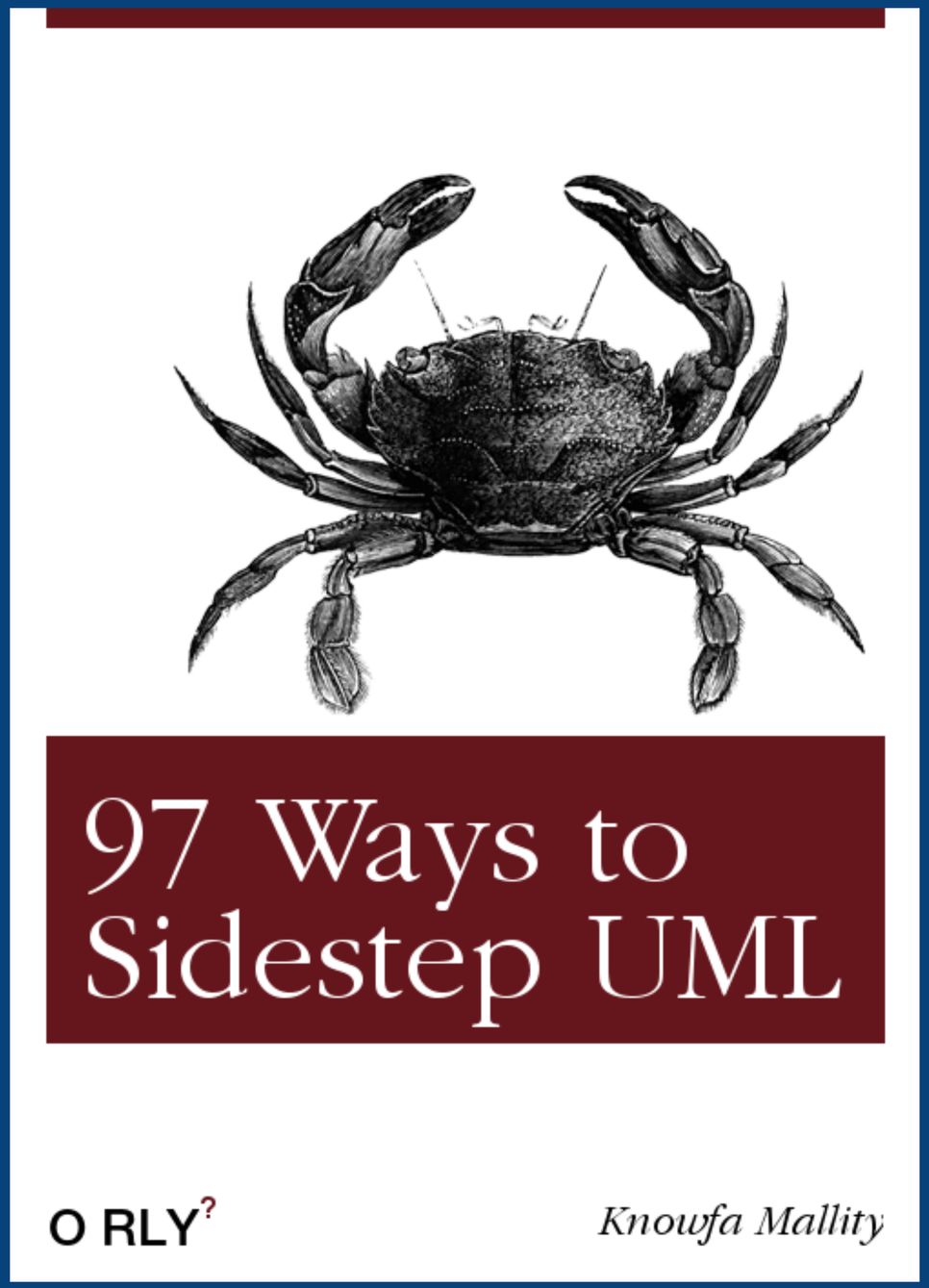
O RLY?

Knowfa Mallity



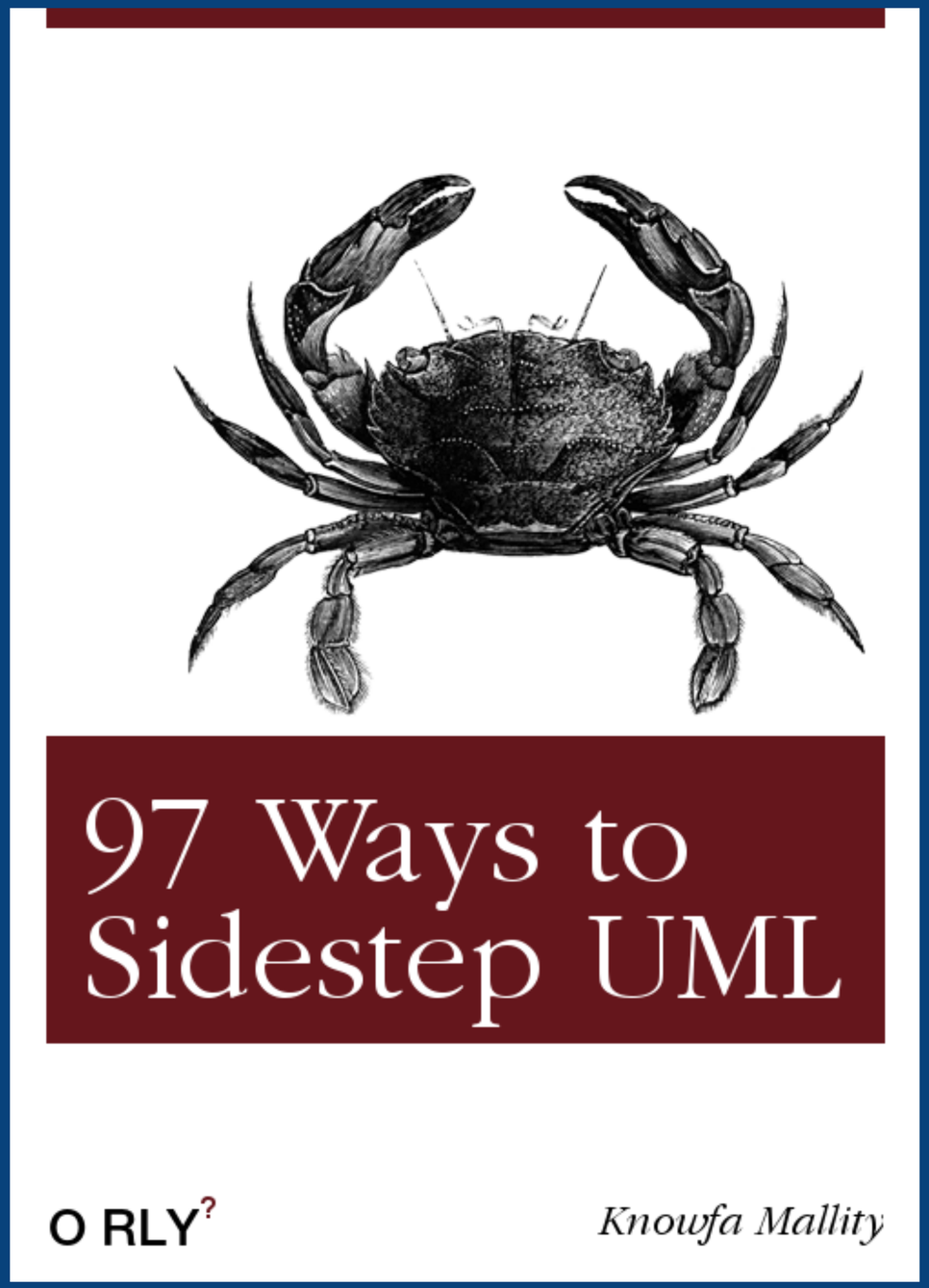
#1

“I don’t know it.”



#2

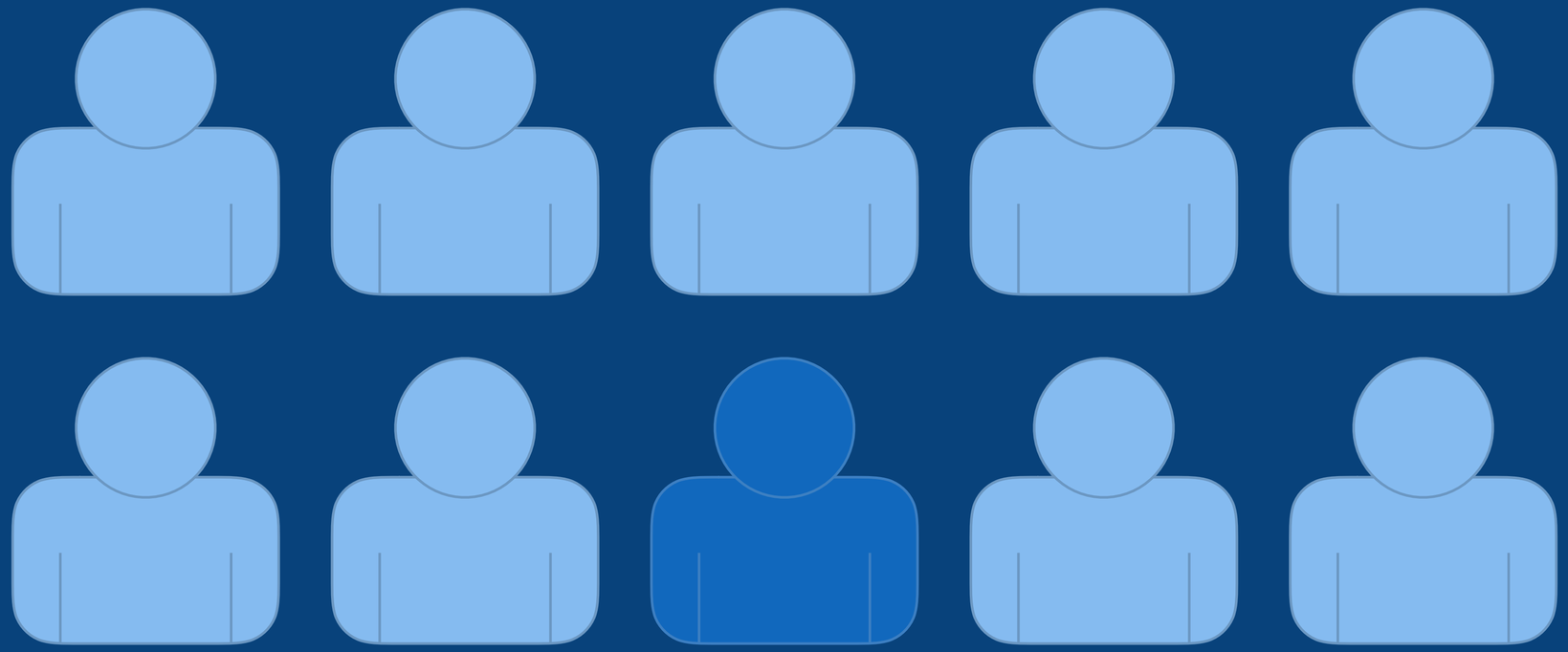
“Not everybody on the team knows it.”



#3

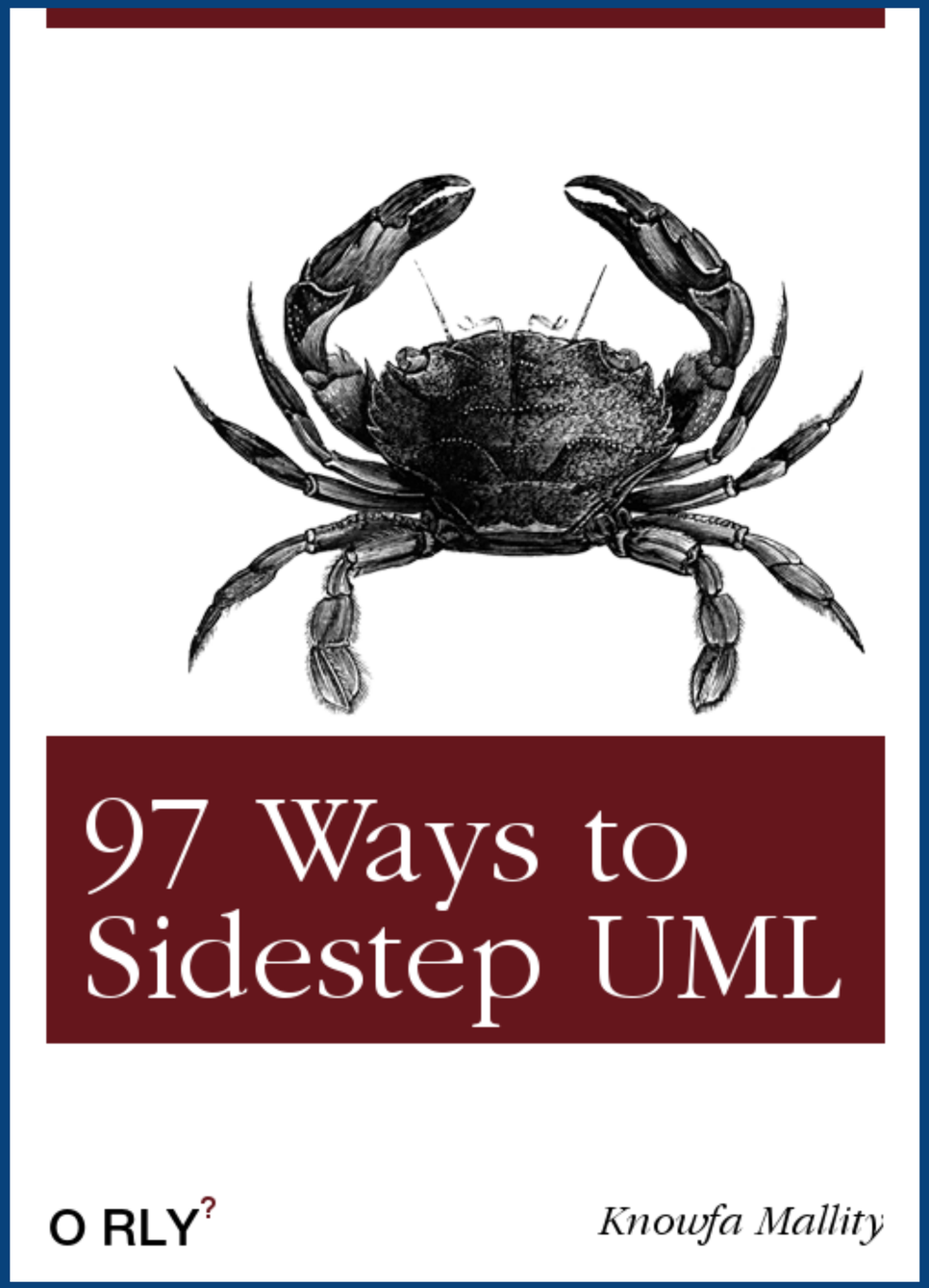
“I’m the only person on the team who knows it.”

Do **you** use UML?



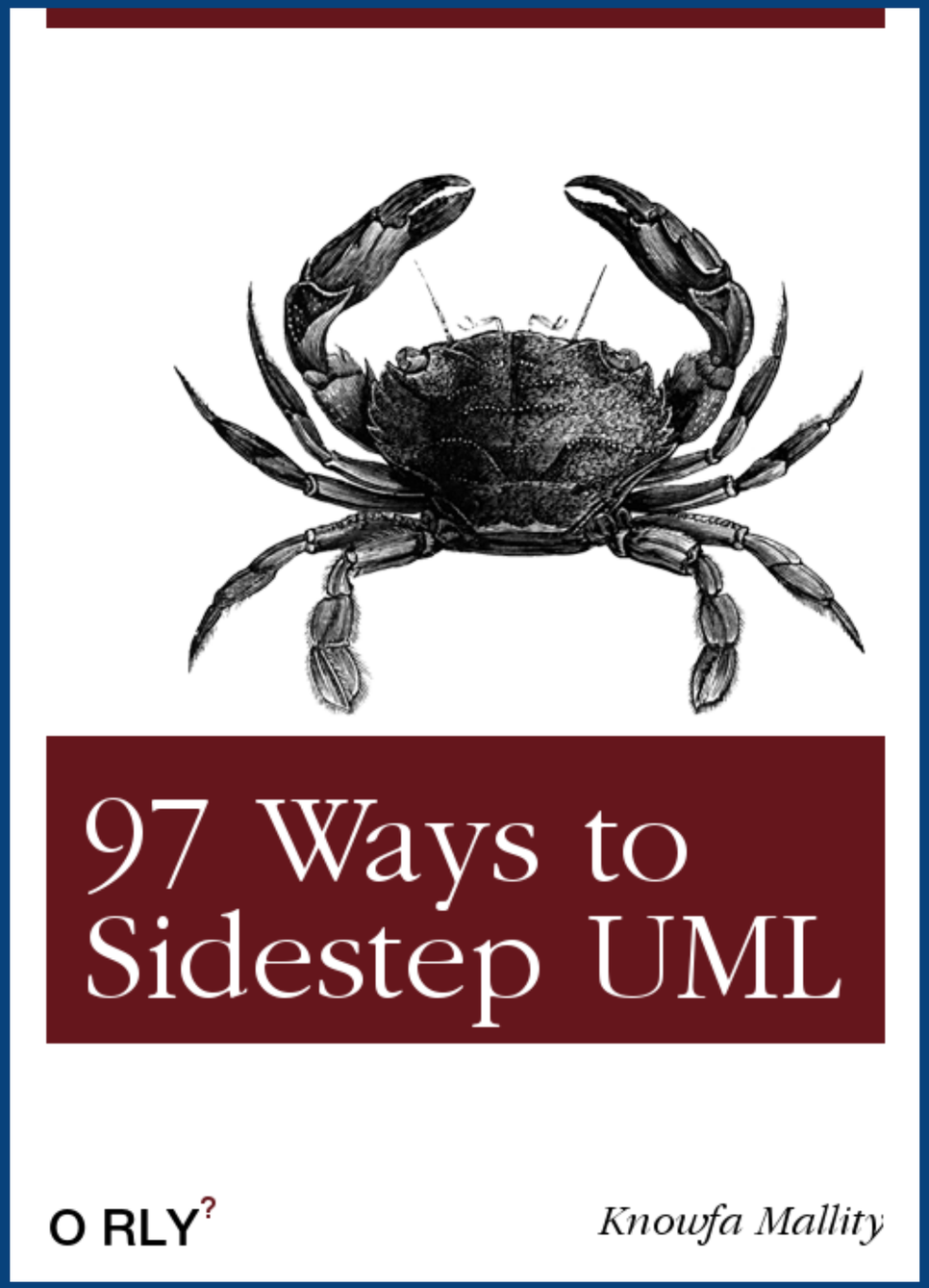
In my experience, optimistically,

1 out of 10 people use UML



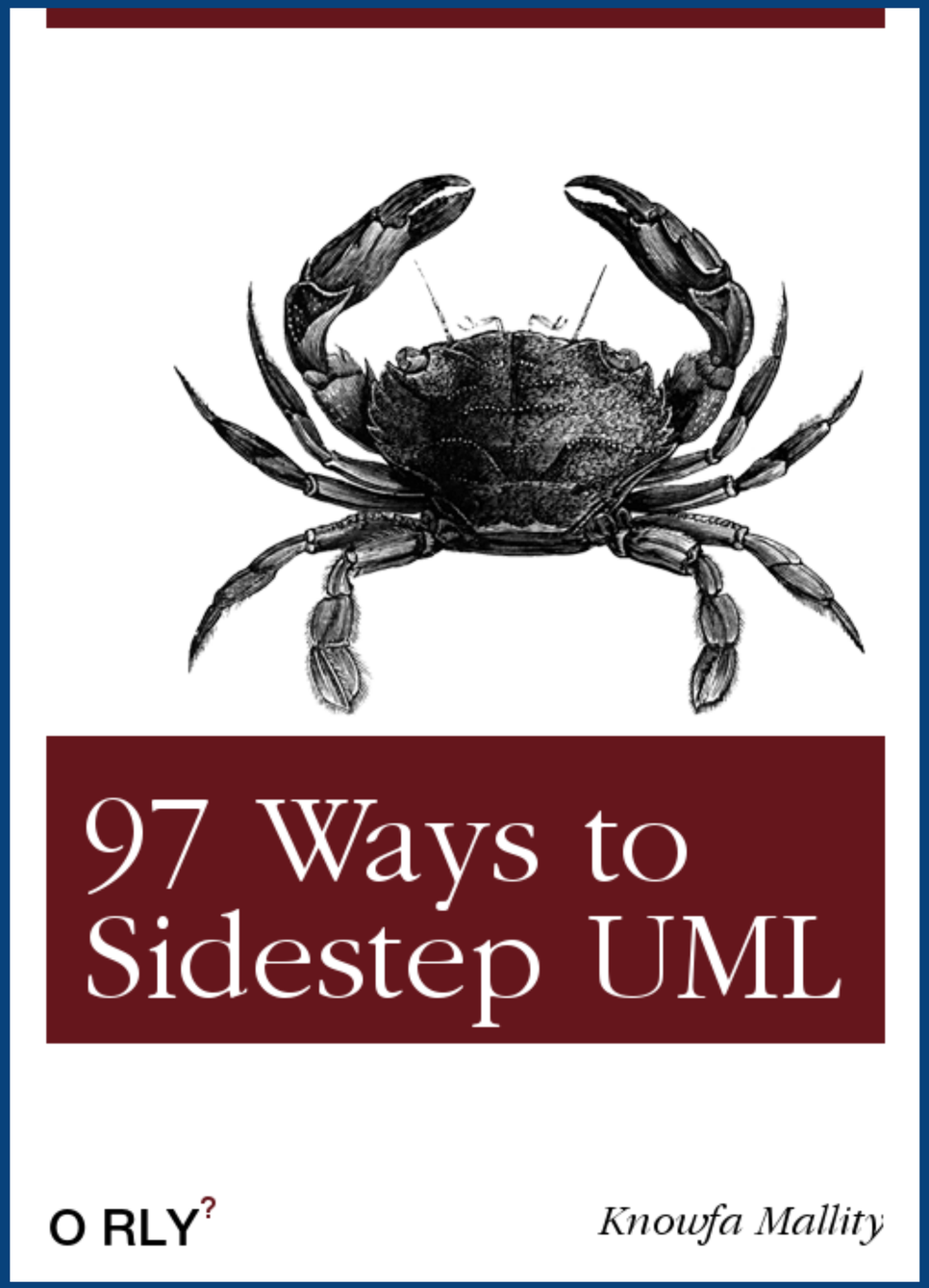
#36

“You’ll be seen as old.”



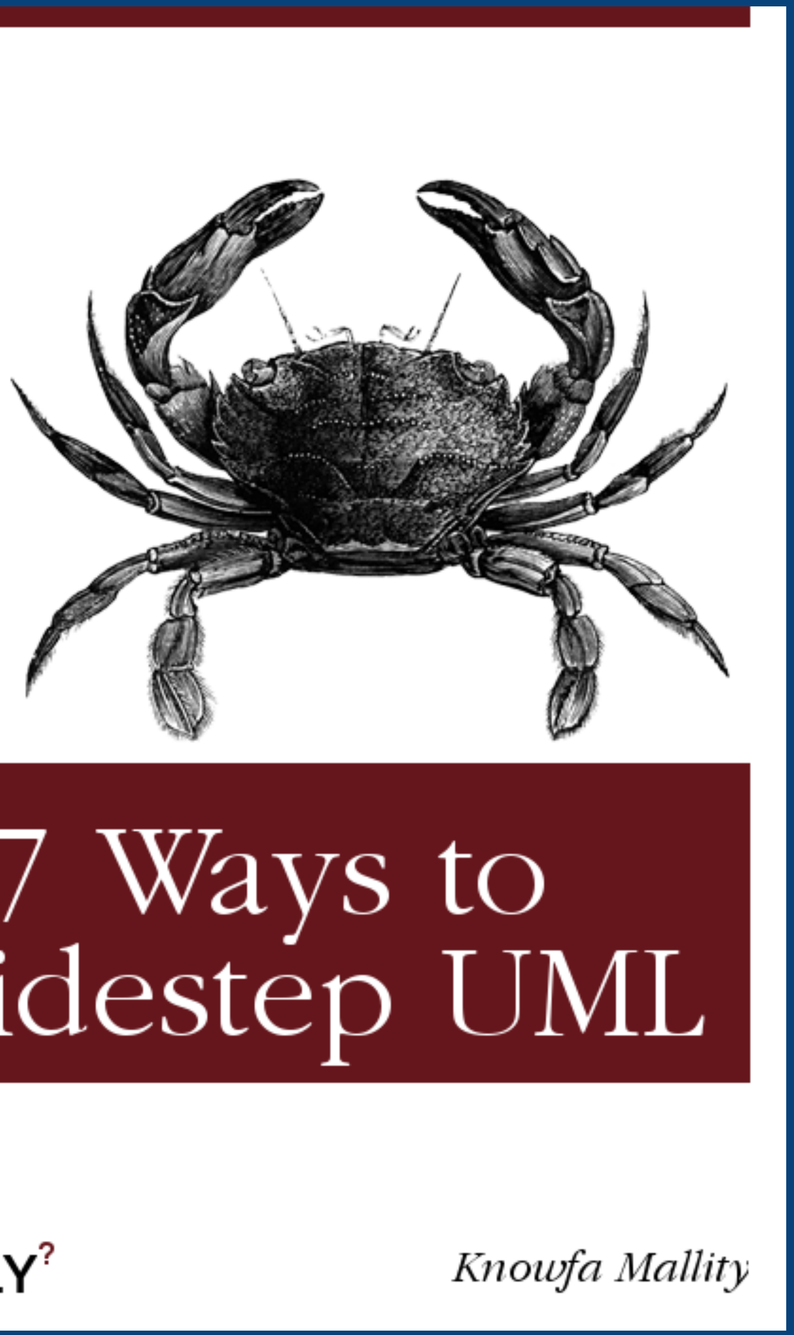
#37

“You’ll be seen as old-fashioned.”



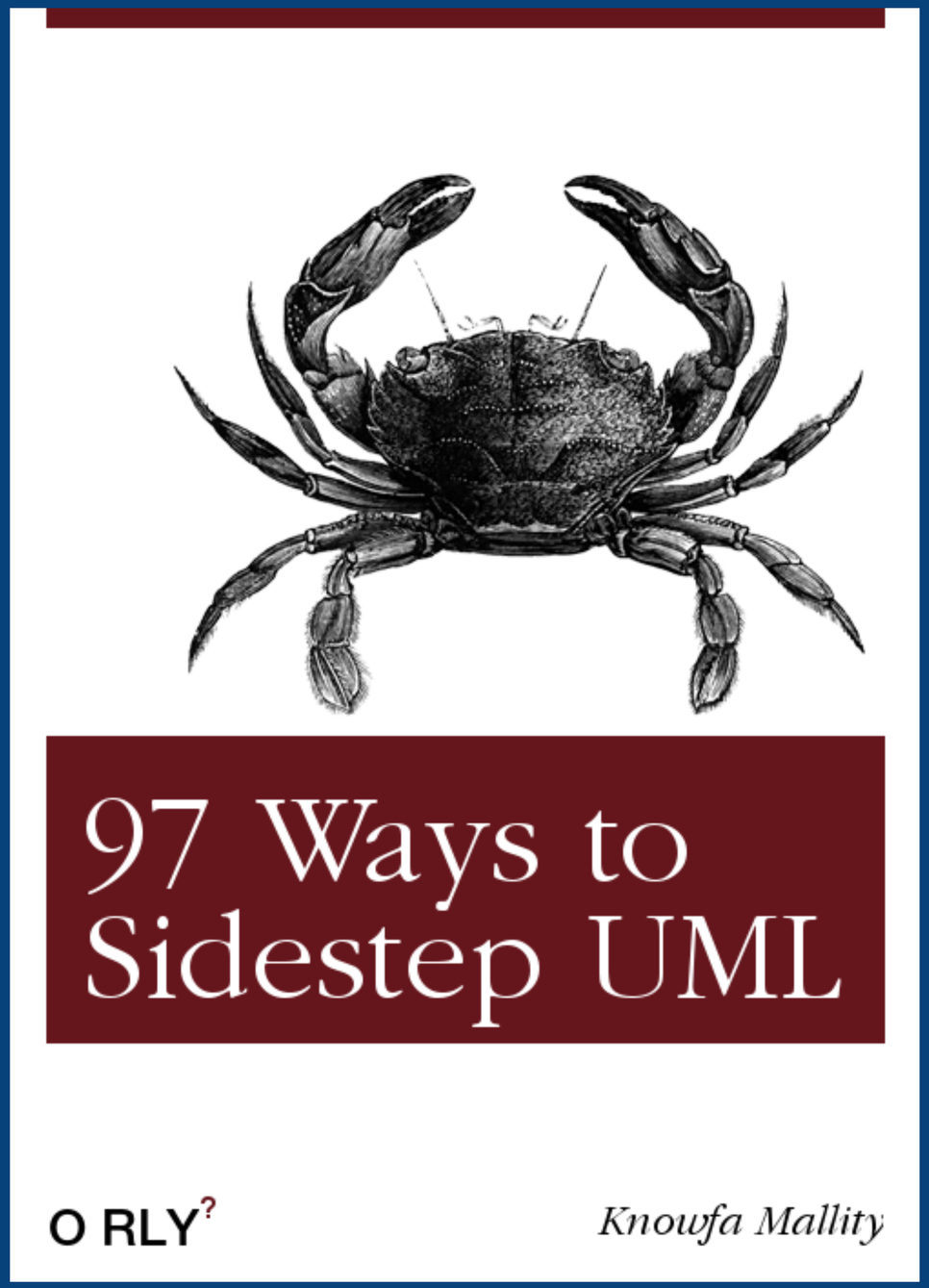
#46

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



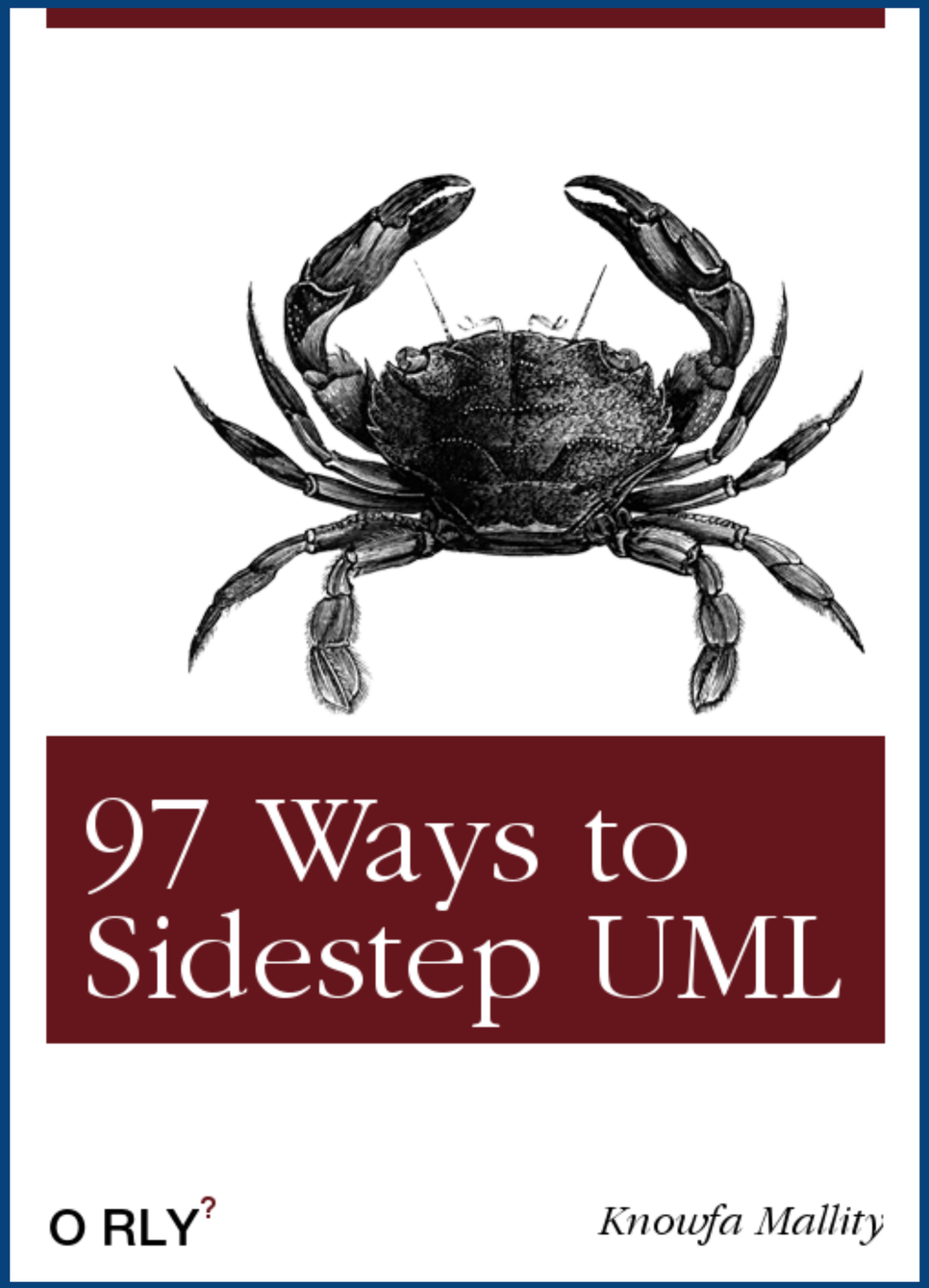
#66

“The tooling sucks.”



#80

“It’s too detailed.”

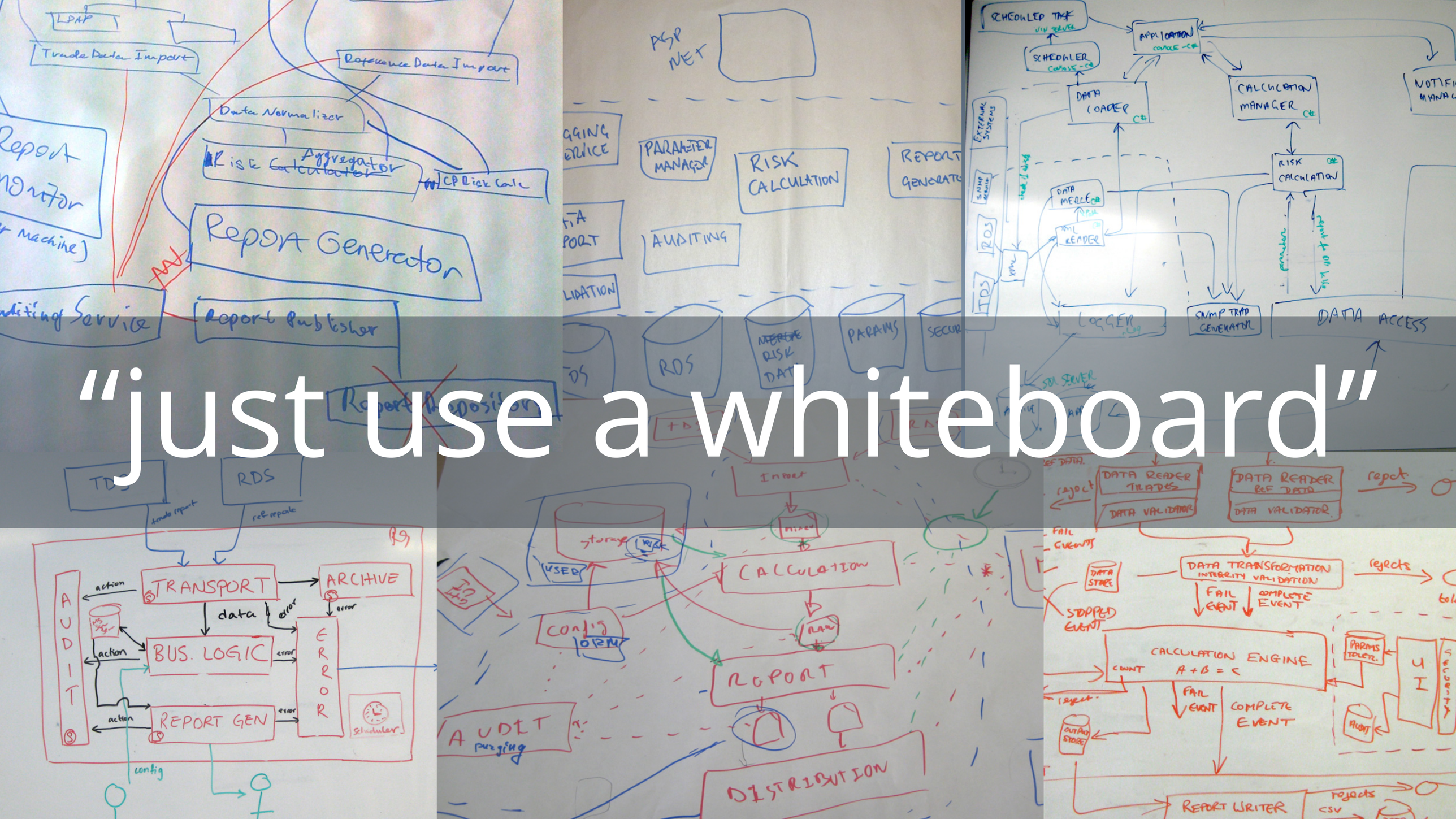


#81

“It’s a very elaborate waste of time.”

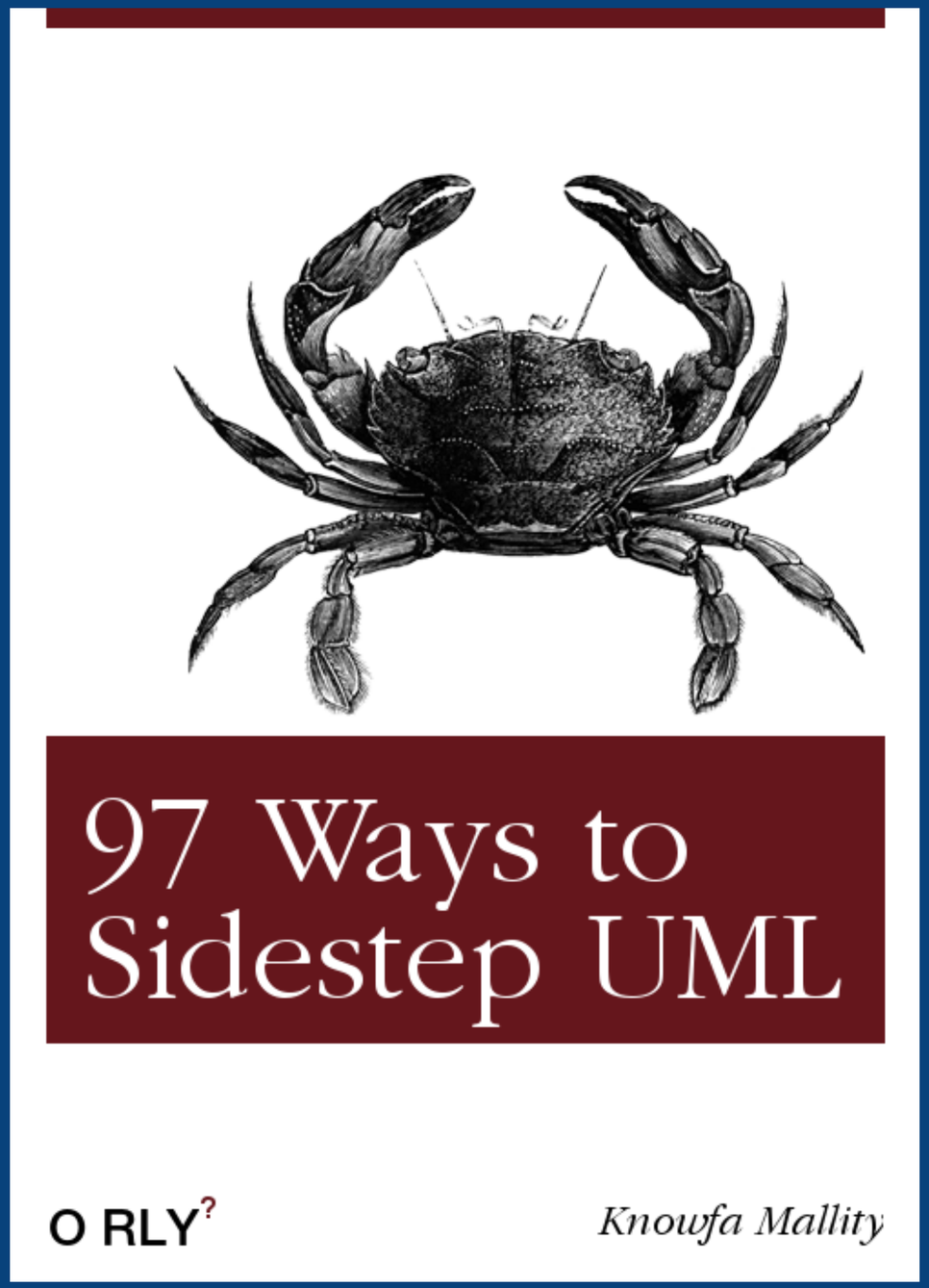

UML, Cucumber and modeling reality - MPJ's Musings - Fun Fun Function
Very elaborate waste of time
1:16 / 12:48


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



“just use a whiteboard”

Engineers,
not artists!



#92

“It’s not expected in agile.”

Would it be better if we used a CASE tool to lay out the design?

No, it wouldn't. The design is more readily expressed, changed, and understood when done less formally, with CRC or on the whiteboard or a bar napkin.

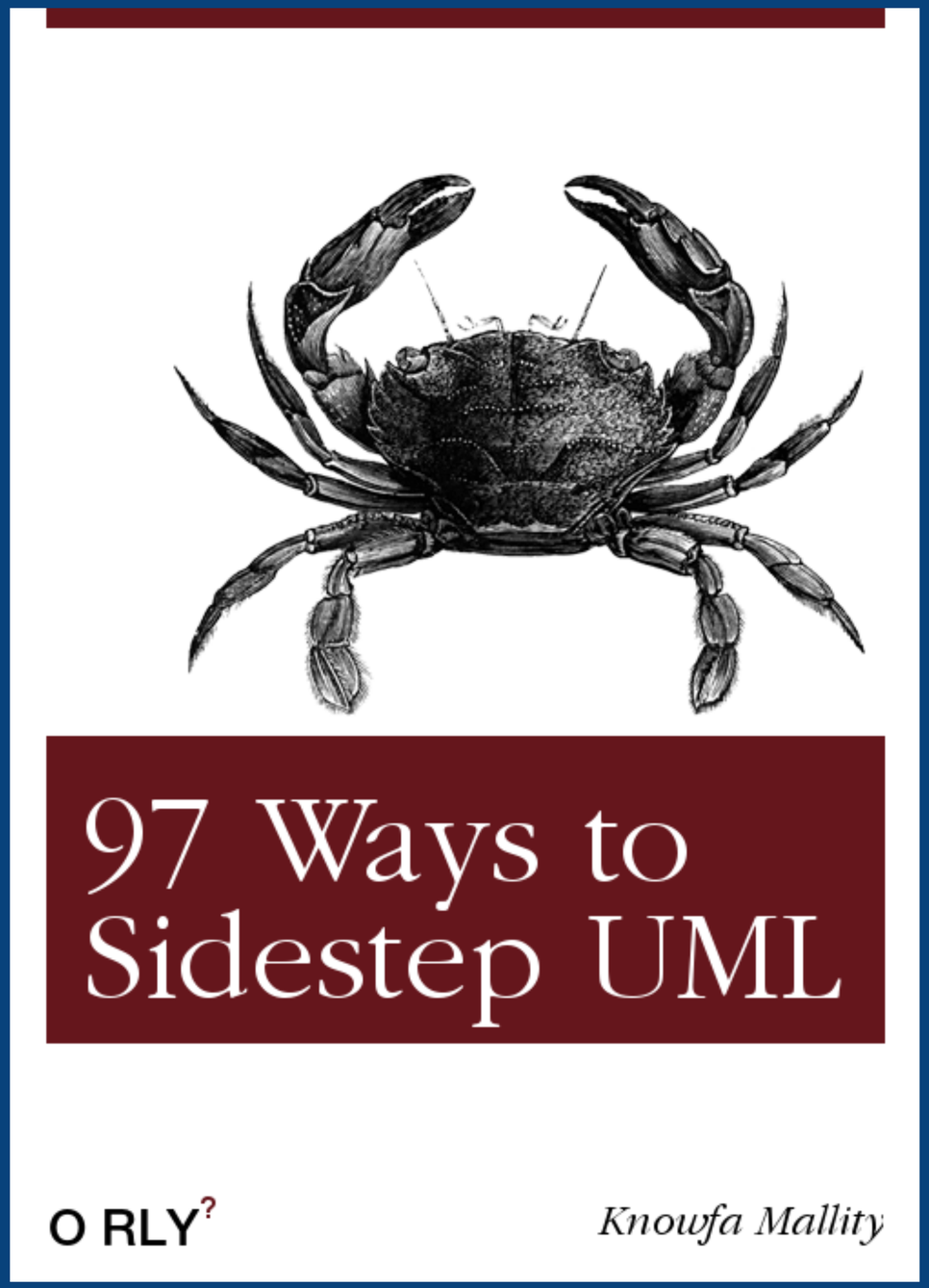
Ron Jeffries

https://ronjeffries.com/xprog/articles/fussaboutdocumentation/

Now don’t get me wrong (again). **You may well need some nicely formatted UML for your project**, or you may need to print out Javadoc when you distribute your code to other users, or you may need to document the requirements for management or as part of a contract. If and when you really need these things, then by all means you should do them. **But inside your collocated Whole Team, you most probably will not need them, because the information you need will be communicated through the more effective medium of conversation.**

Ron Jeffries

https://ronjeffries.com/xprog/articles/fussaboutdocumentation/



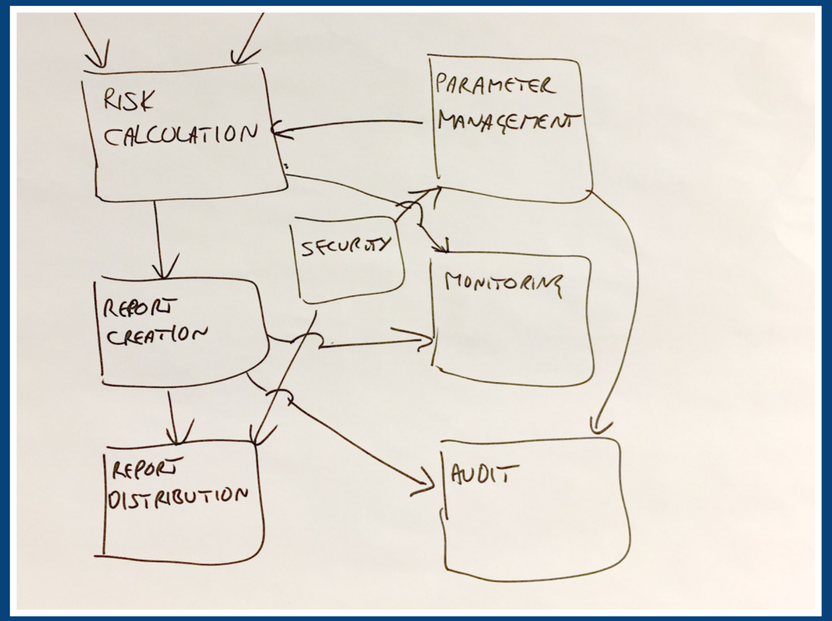
#97

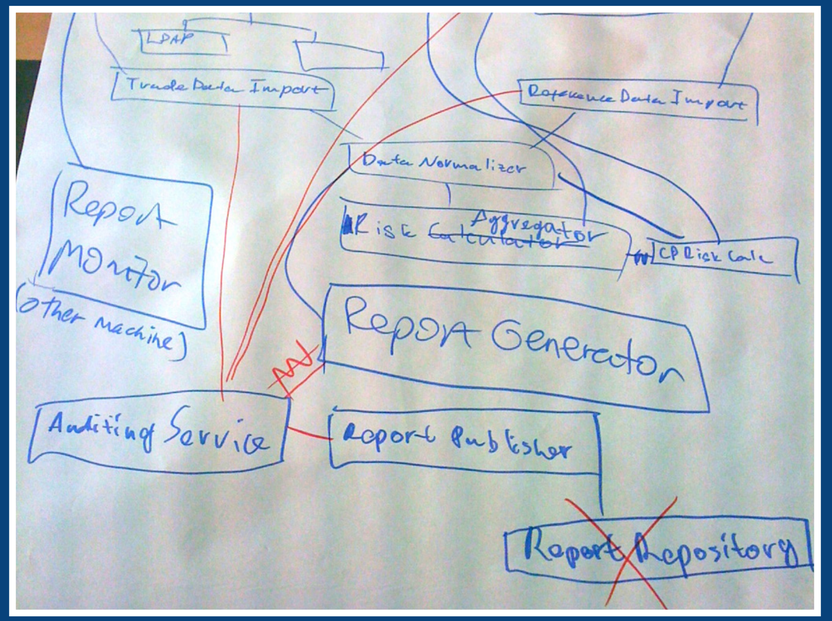
“The value is in the conversation.”

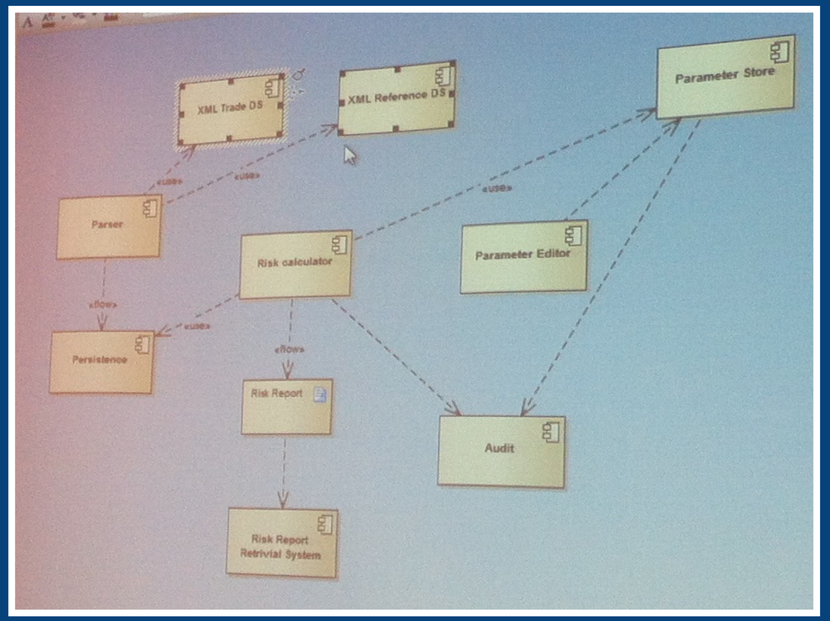
They are all excellent, as long as there is a conversation about their meaning and intent. It's the accompanying conversation that matters.

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

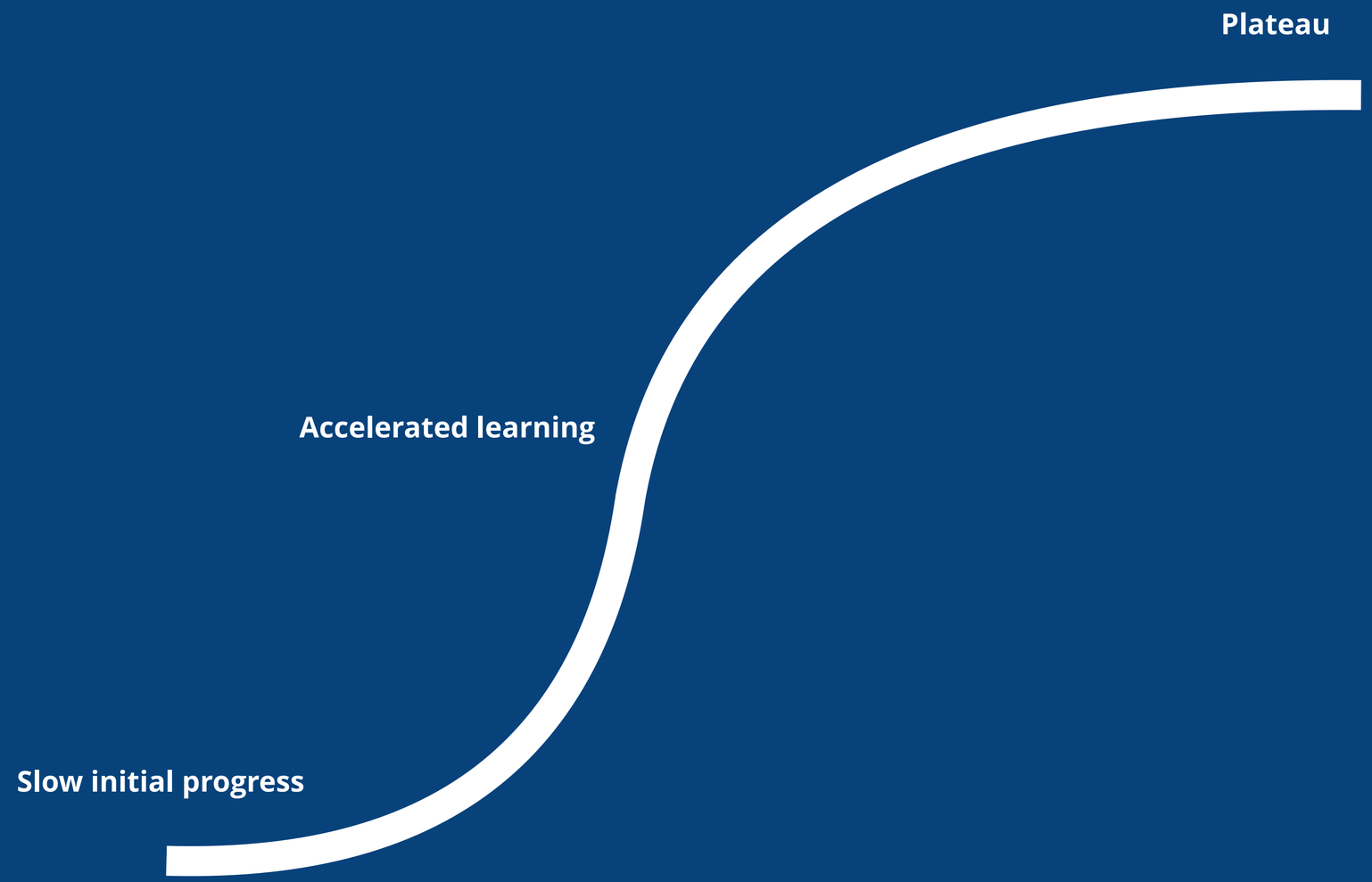
Superficial up front design







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



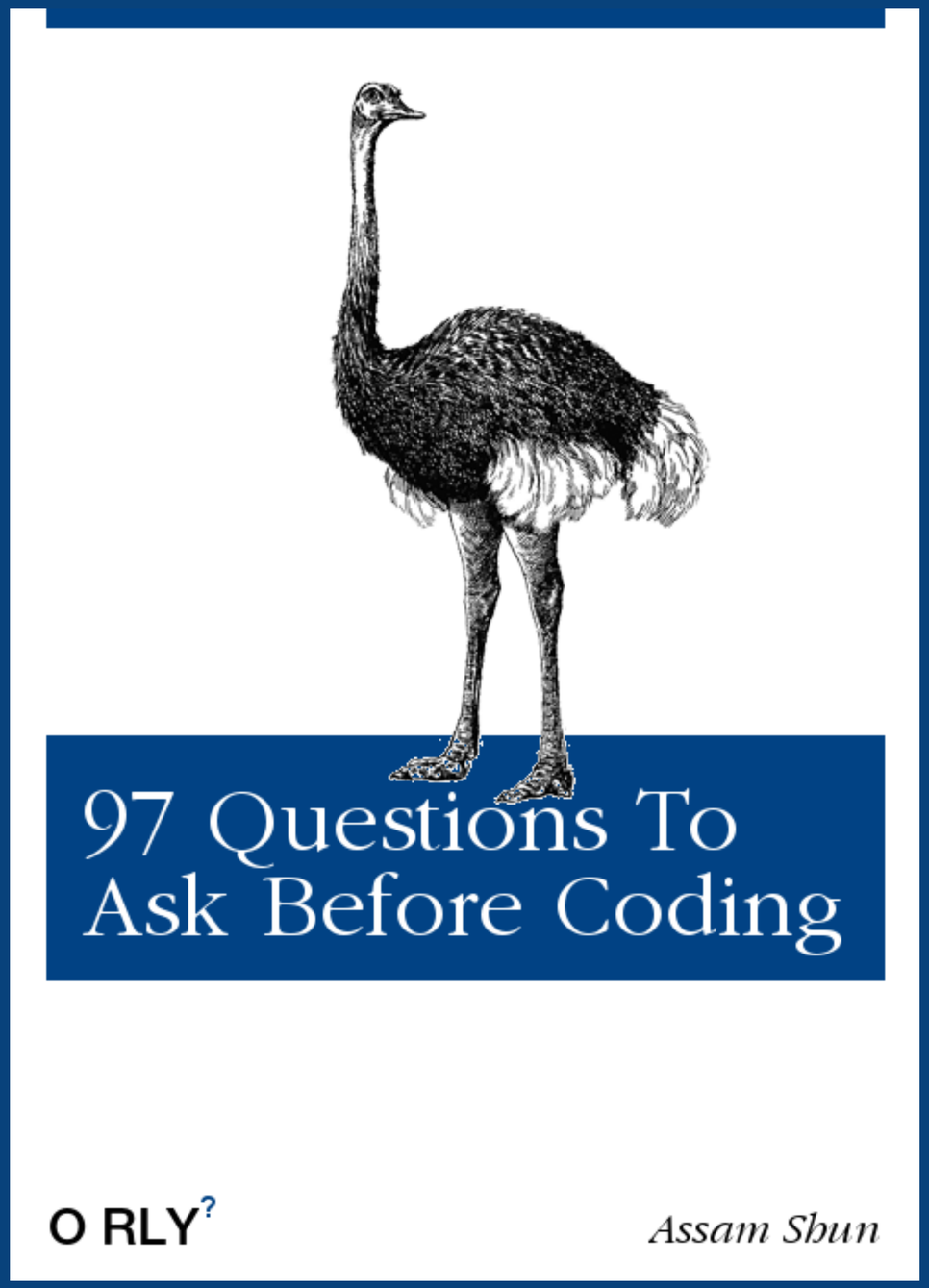
The typical s-curve of learning



97 Questions To Ask Before Coding

O RLY?

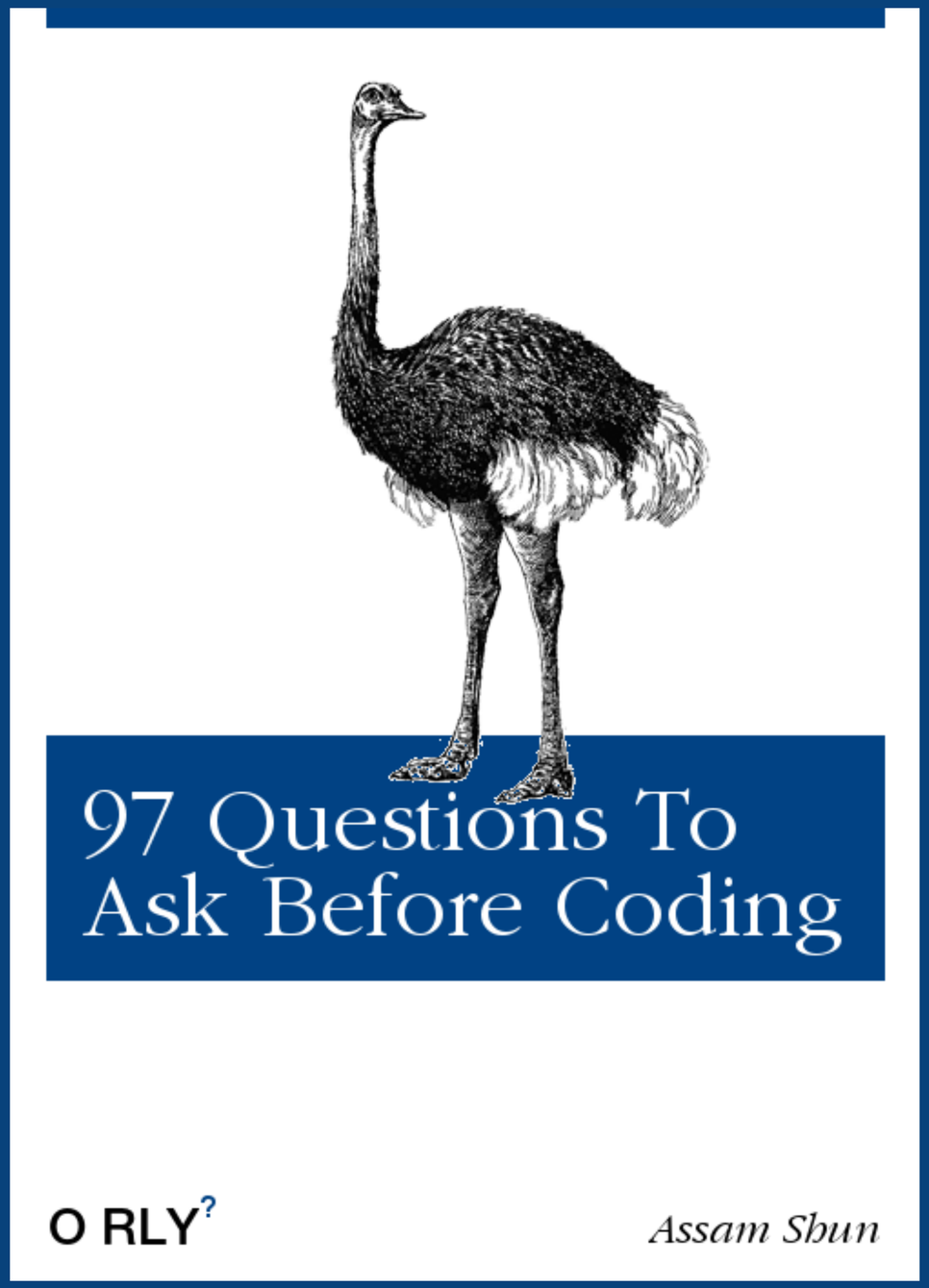
Assam Shun



#1

“What programming language should we use?”

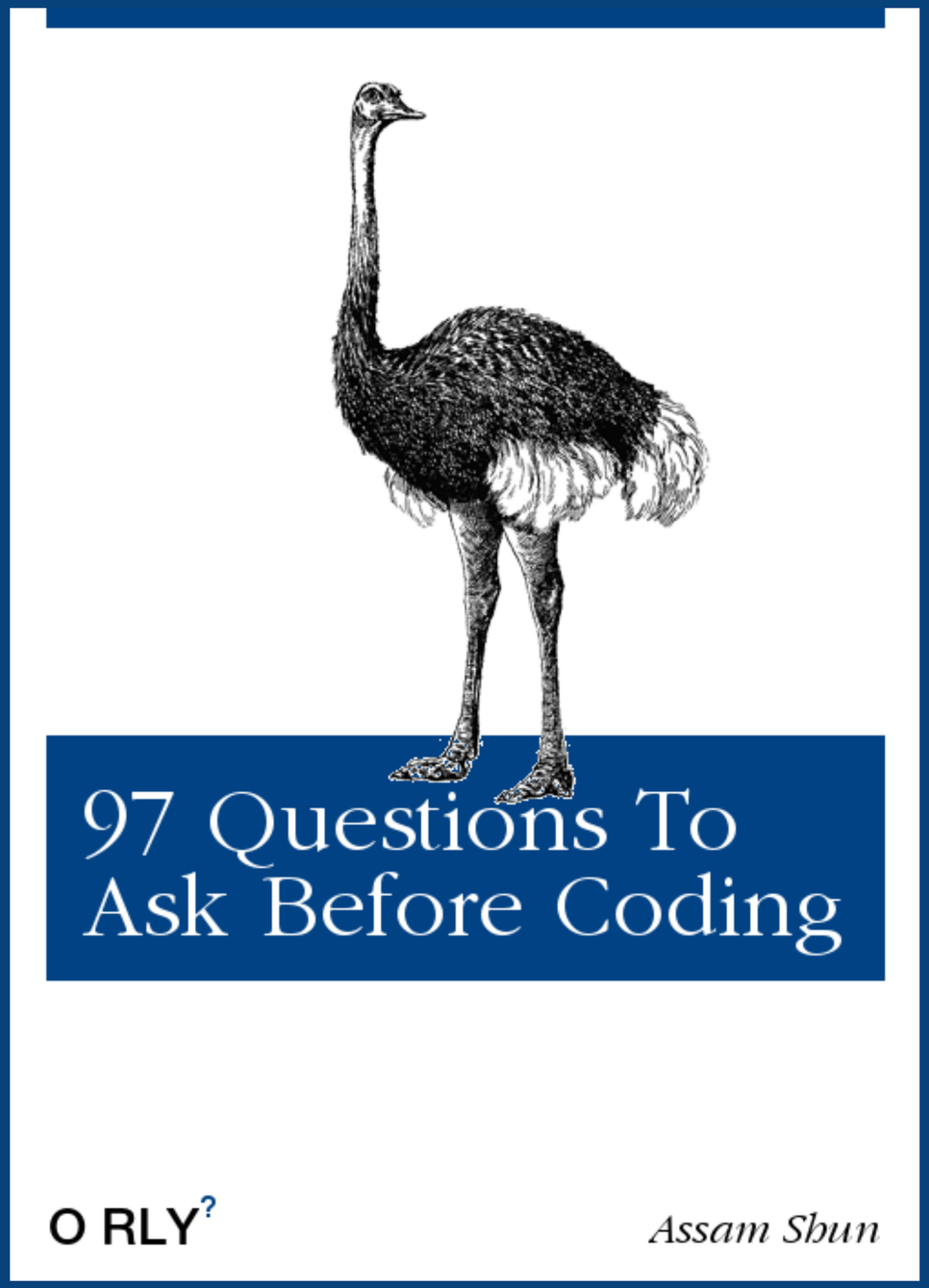
I didn’t realise this was a [Java|.NET|Python|etc] solution.



#4

“Is this a microservices architecture?”

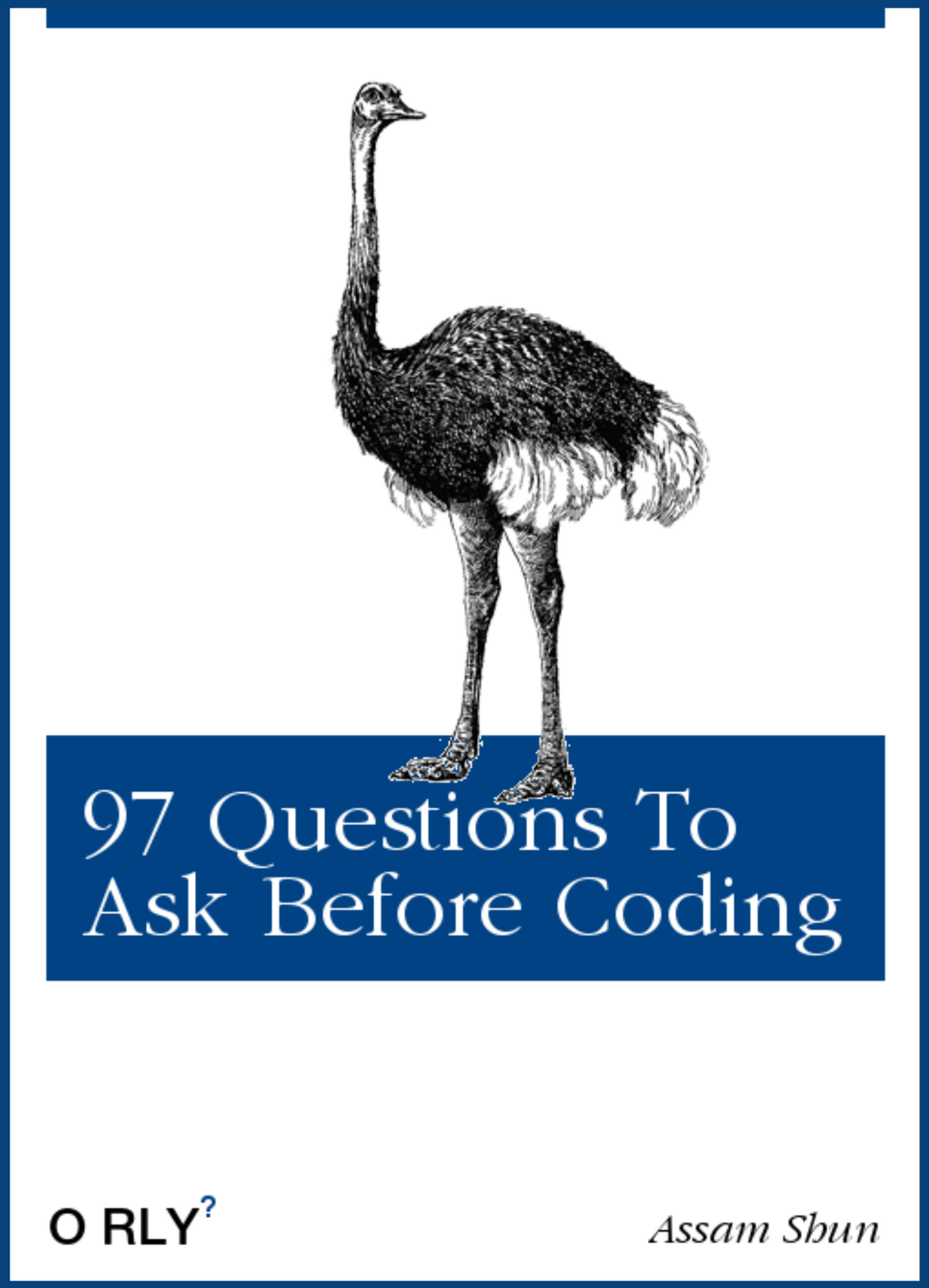
I didn’t realise this was a collection of microservices.



#5

“Is this a monolithic application?”

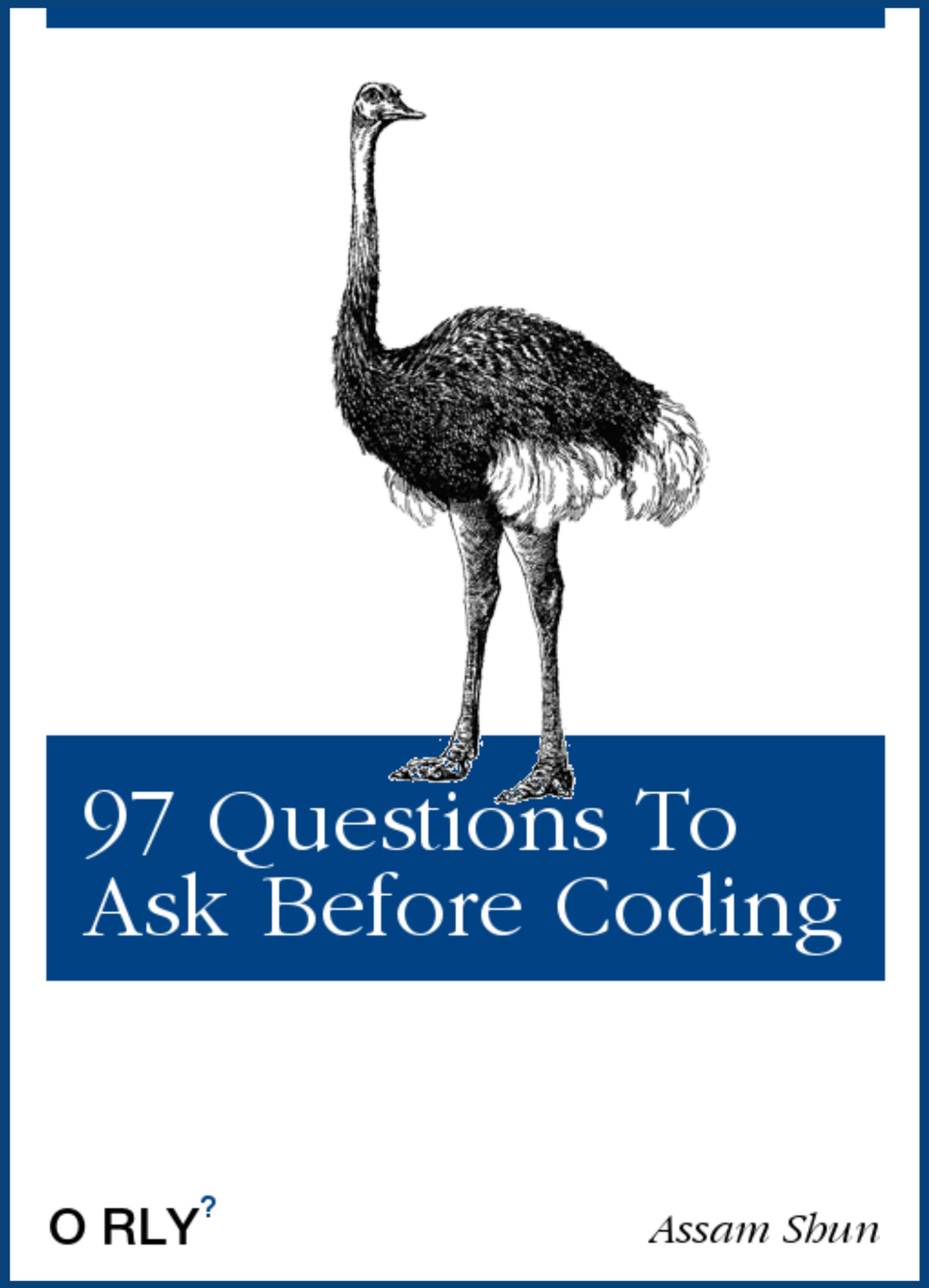
I thought this was a collection of microservices.



#18

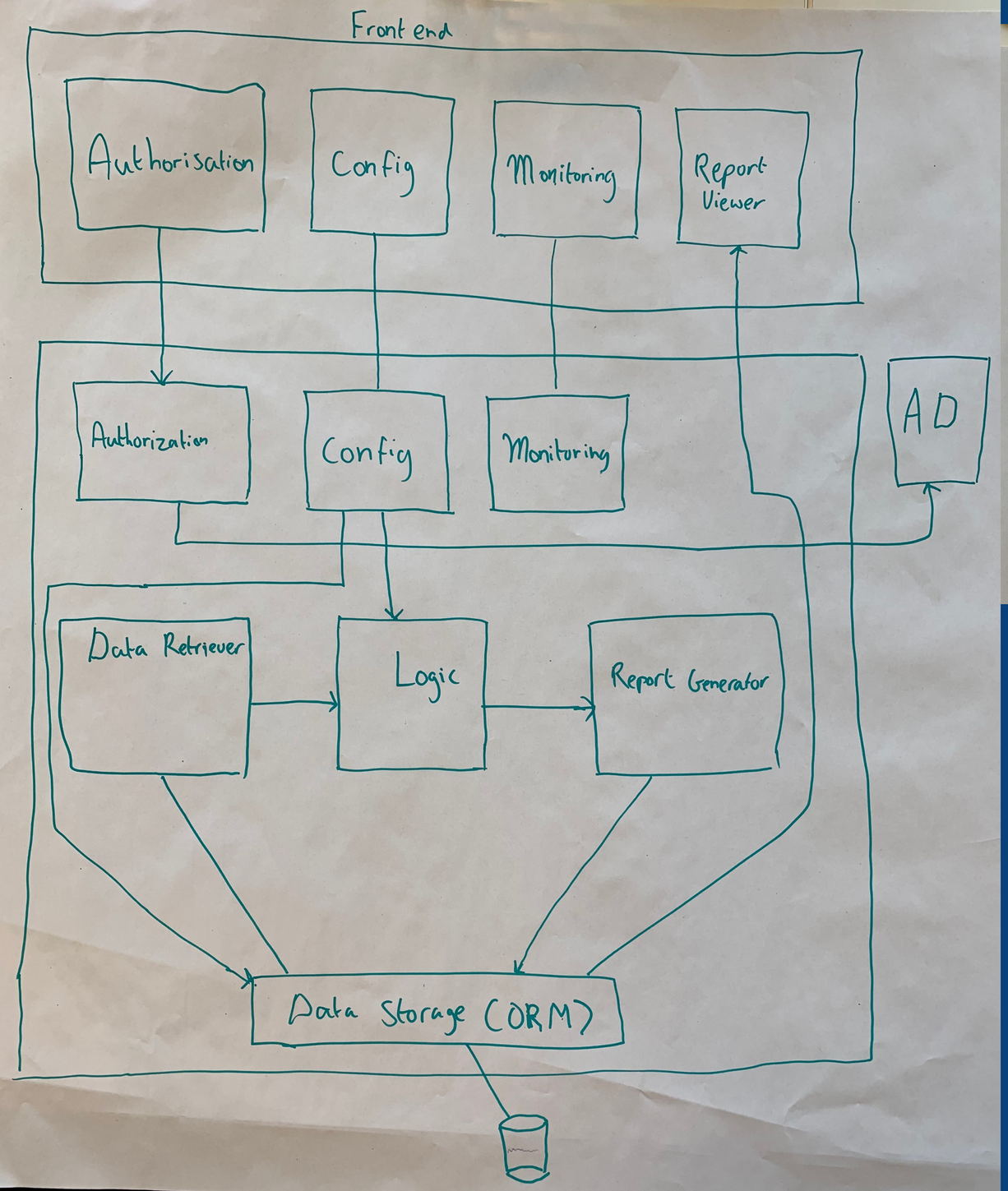
“What does the cylinder shape mean?”

I didn’t think that we were using a database.



#73

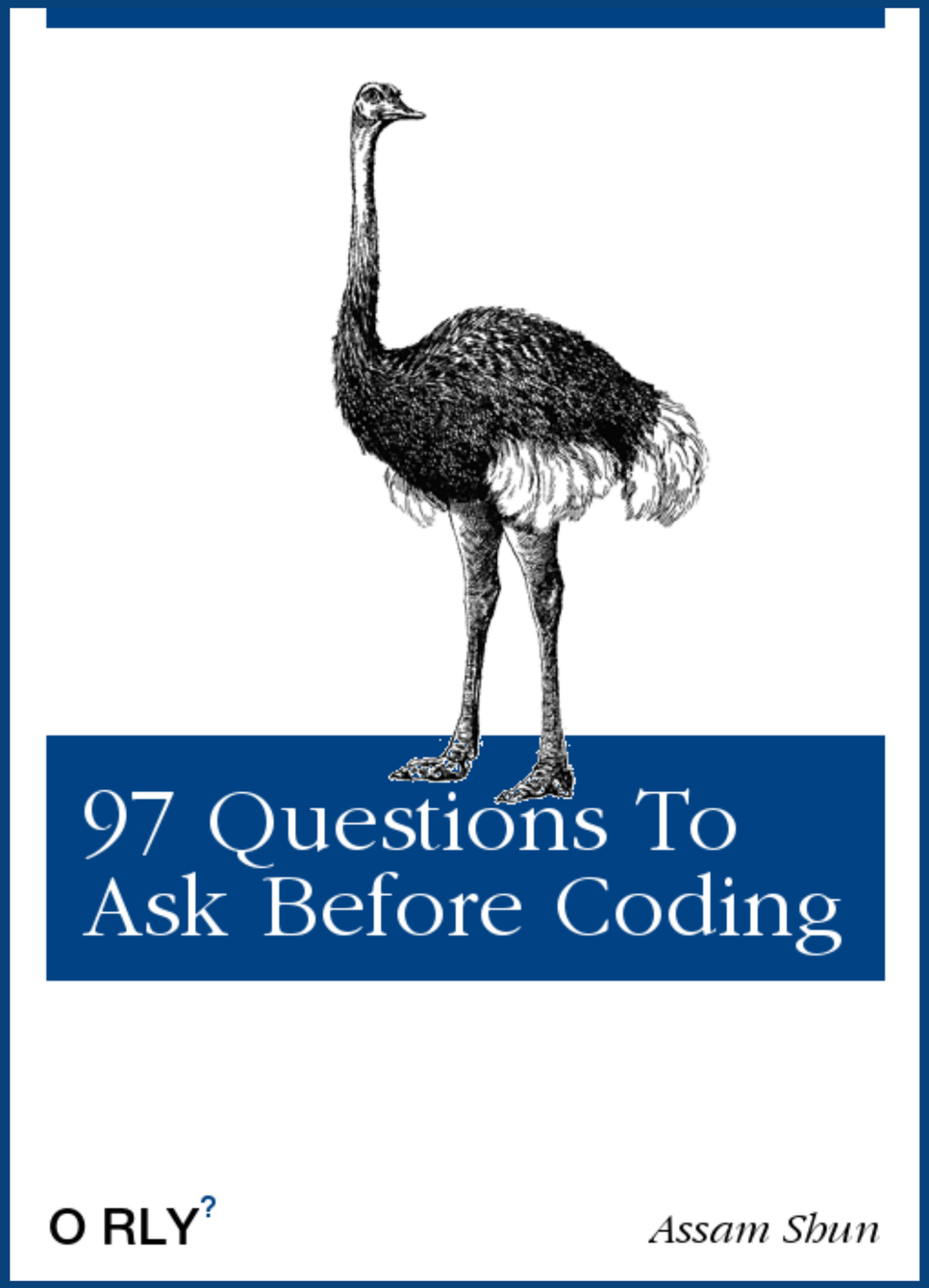
“Why is the ORM directly connected to the Angular front-end?”



Significant decisions

- F/E <> B/E
- Make use of OS' watchdog mechanism
- Data storage ORM-framework: Entity
- ASP .NET B/E
- Angular F/E

Why is the ORM directly connected to the Angular front-end?



#76

“Is the web UI getting data from Amazon S3?”

Yes, the web UI gets data from Amazon S3.

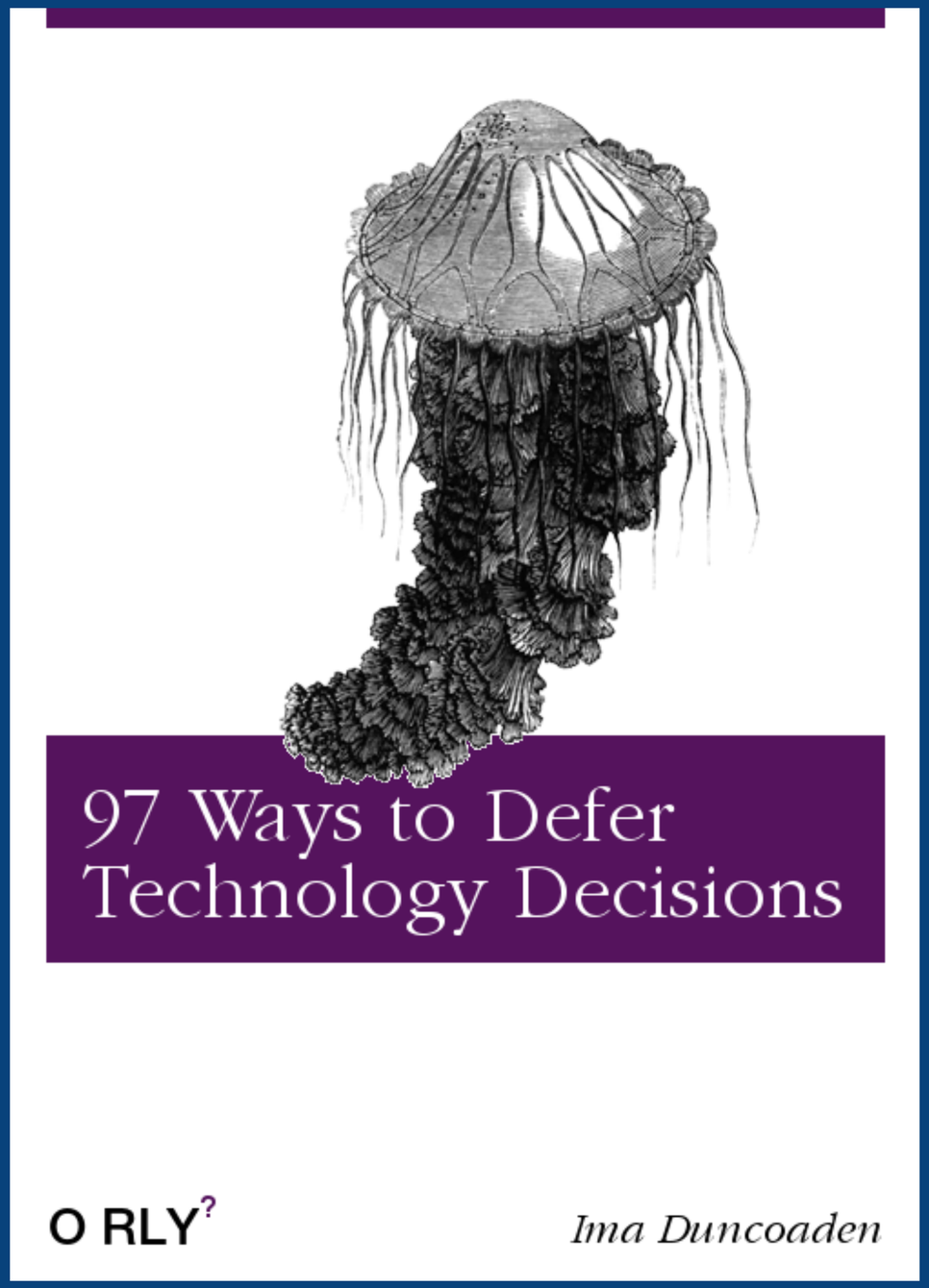
Technology Decisions



97 Ways to Defer Technology Decisions

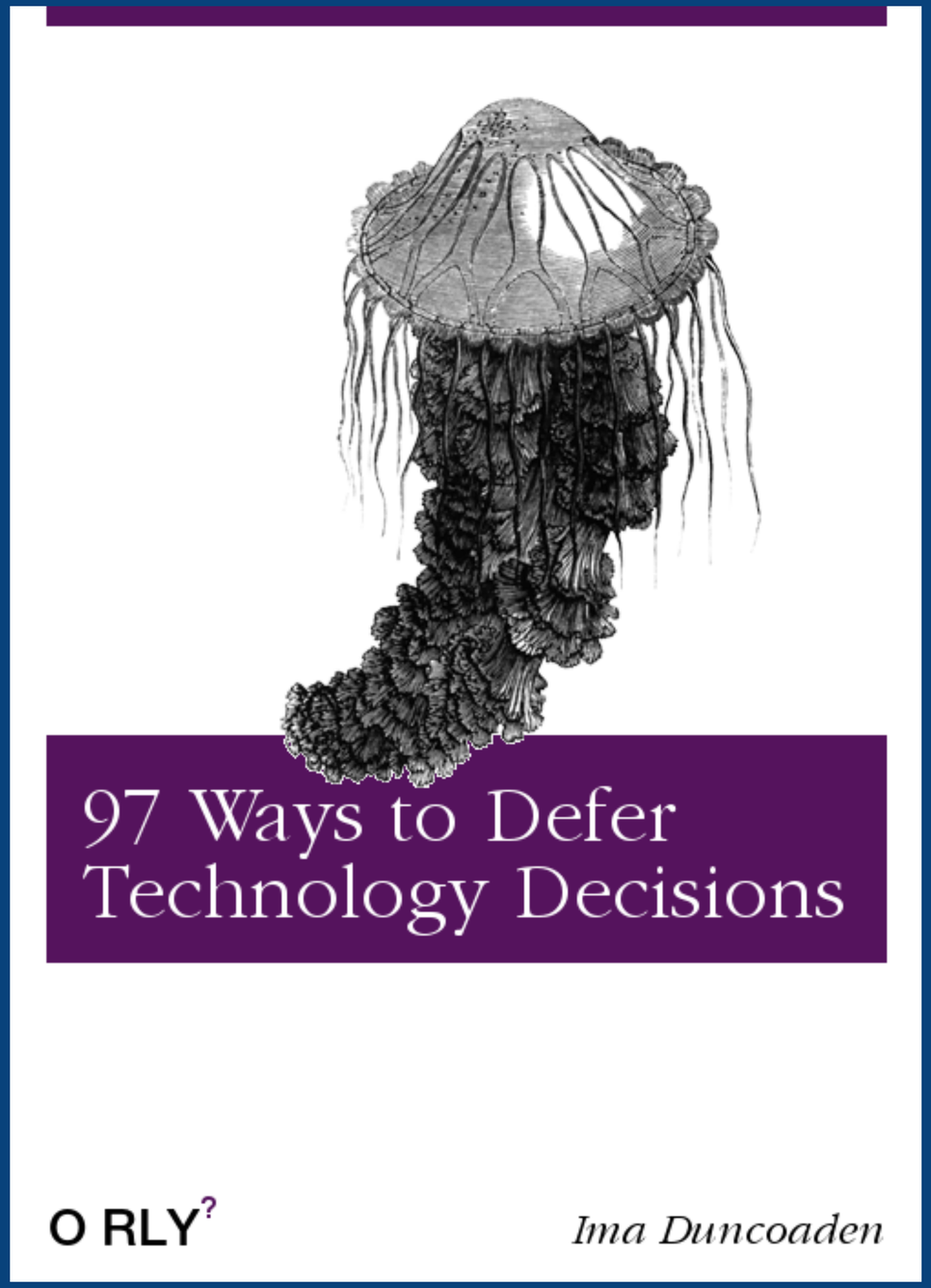
O RLY?

Ima Duncoaden



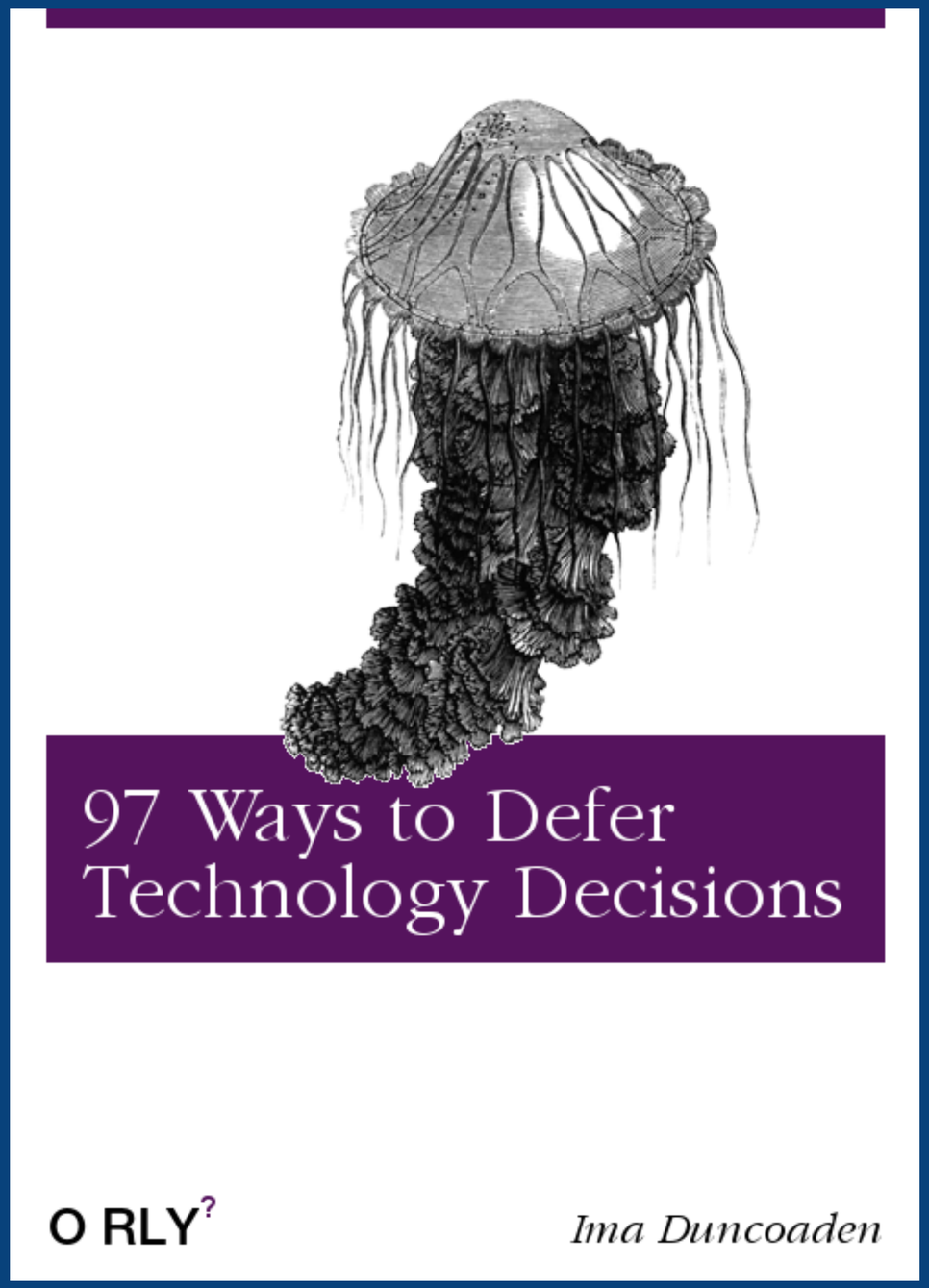
#1

“We don’t have enough detail.”



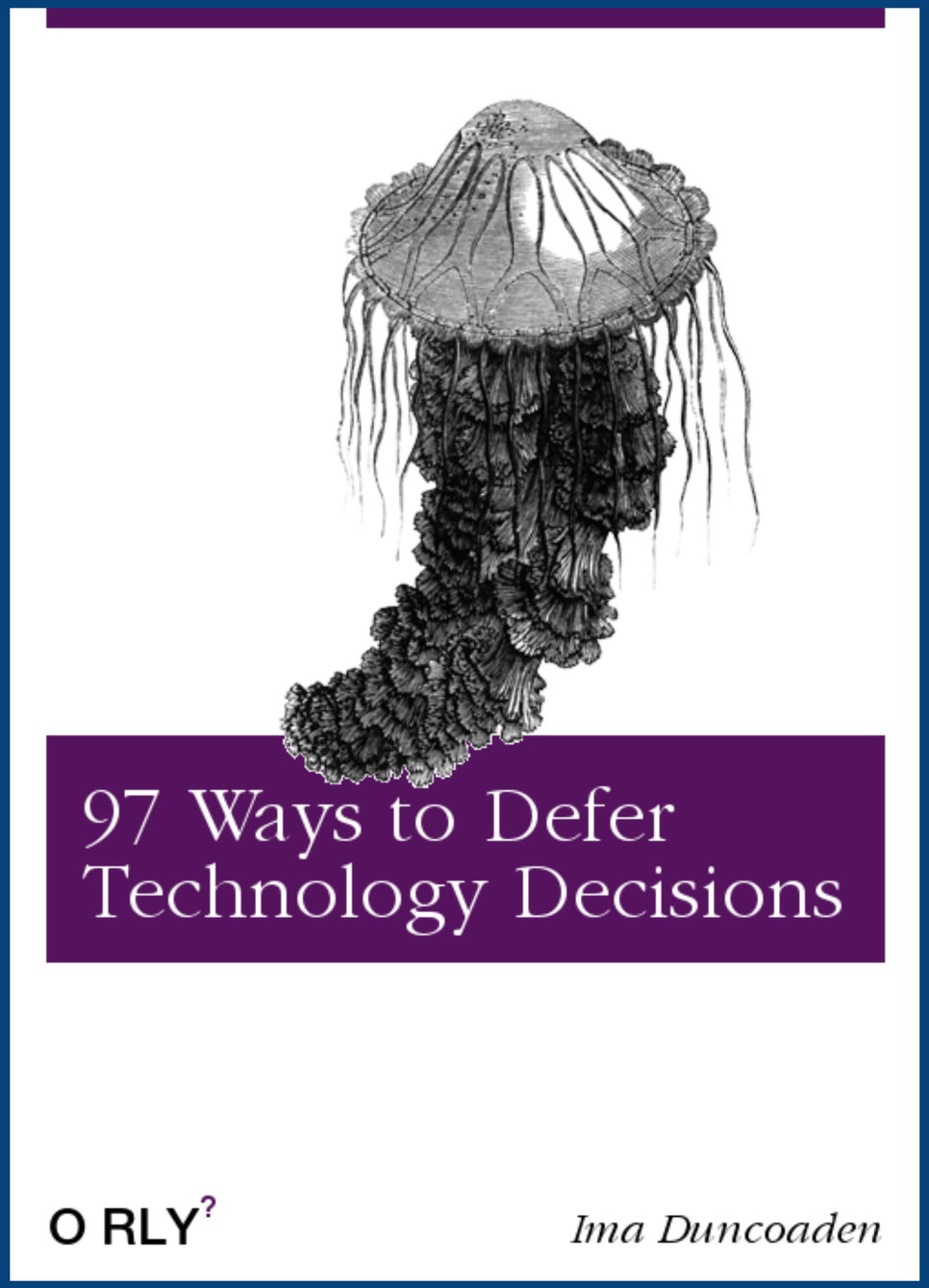
#8

“We don’t solutionize.”



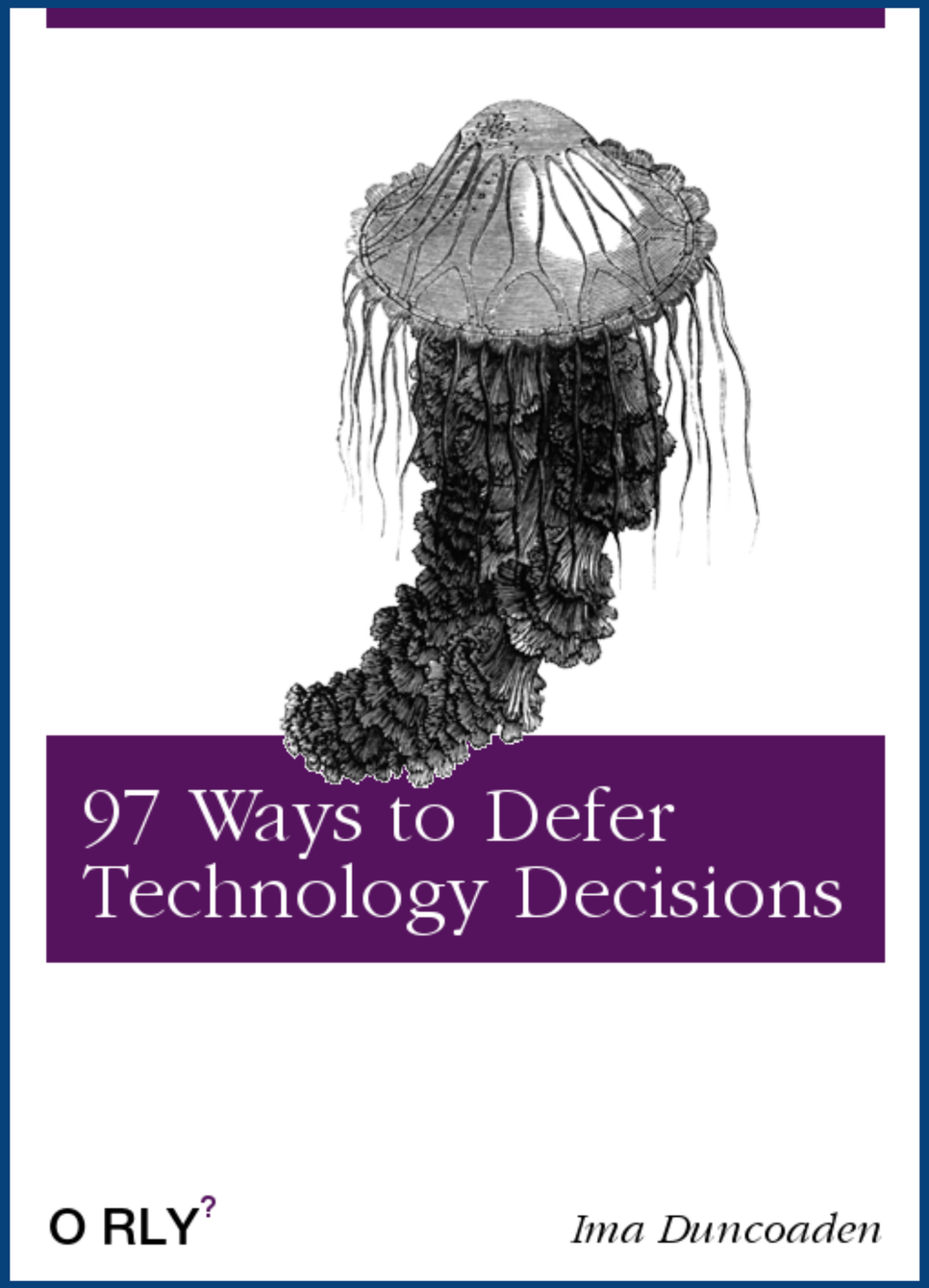
#9

“We don’t solutionize this early.”



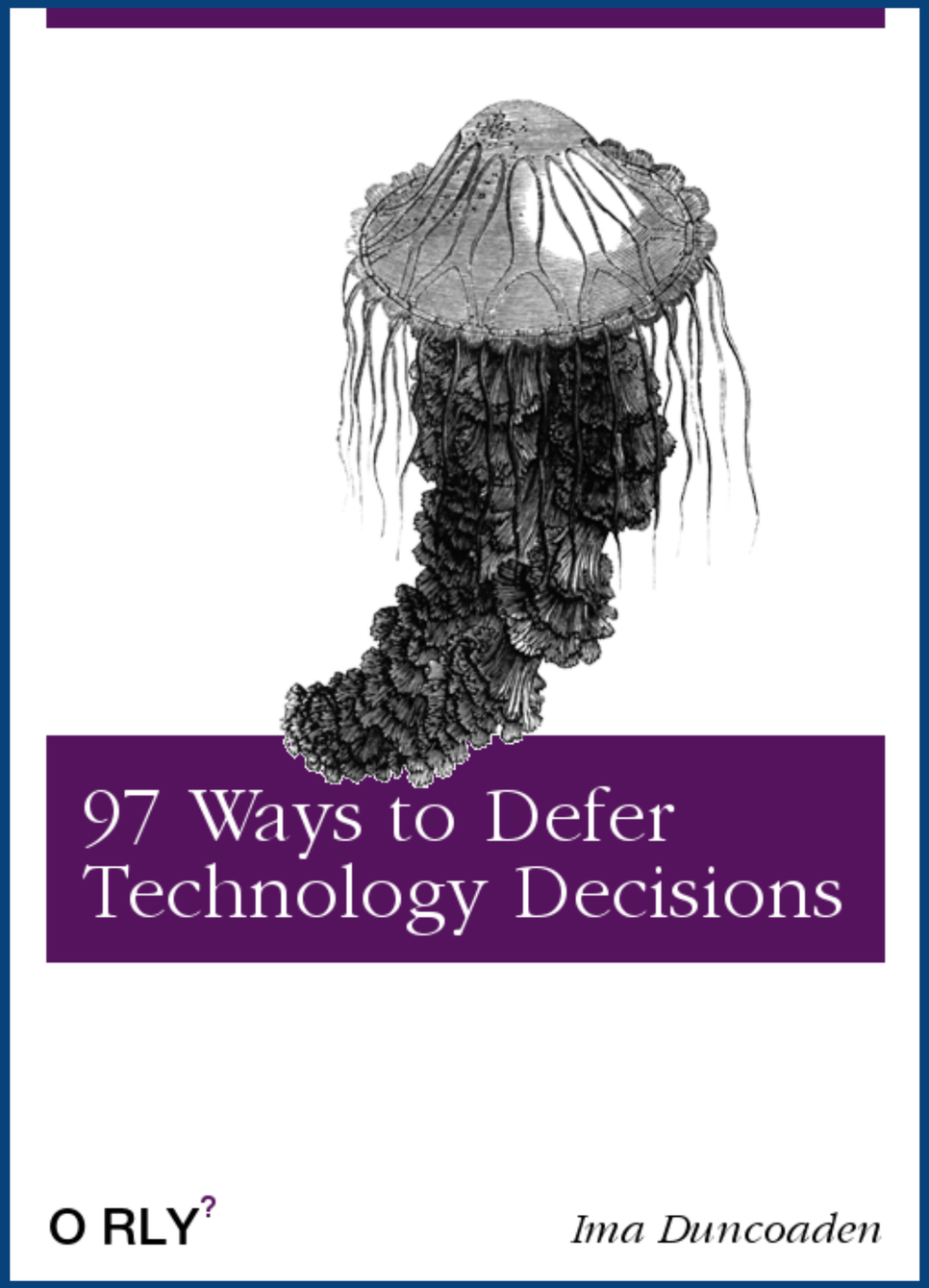
#10

“Our architects are not allowed to do solutioneering.”



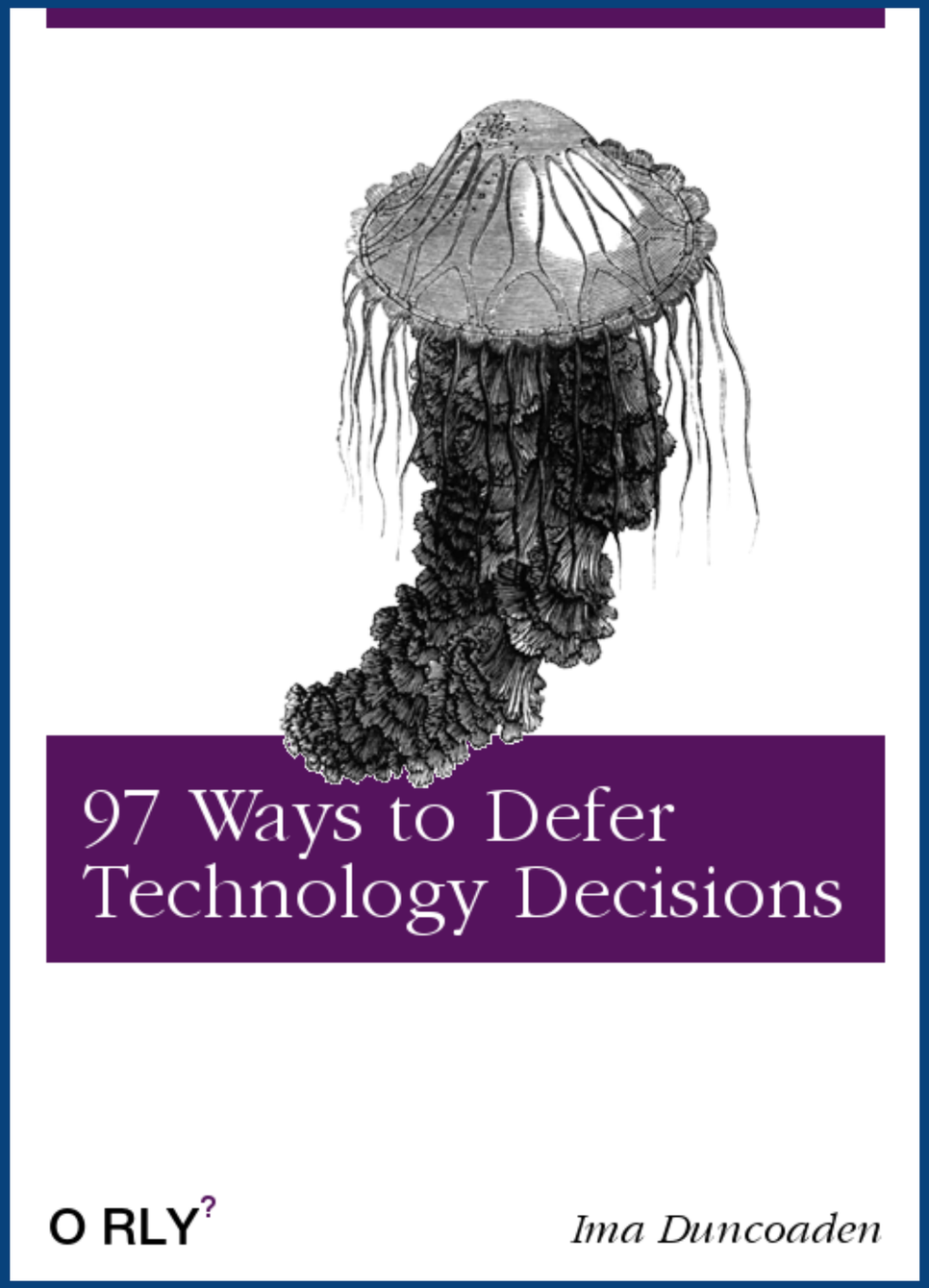
#25

“I don’t know how to do that.”



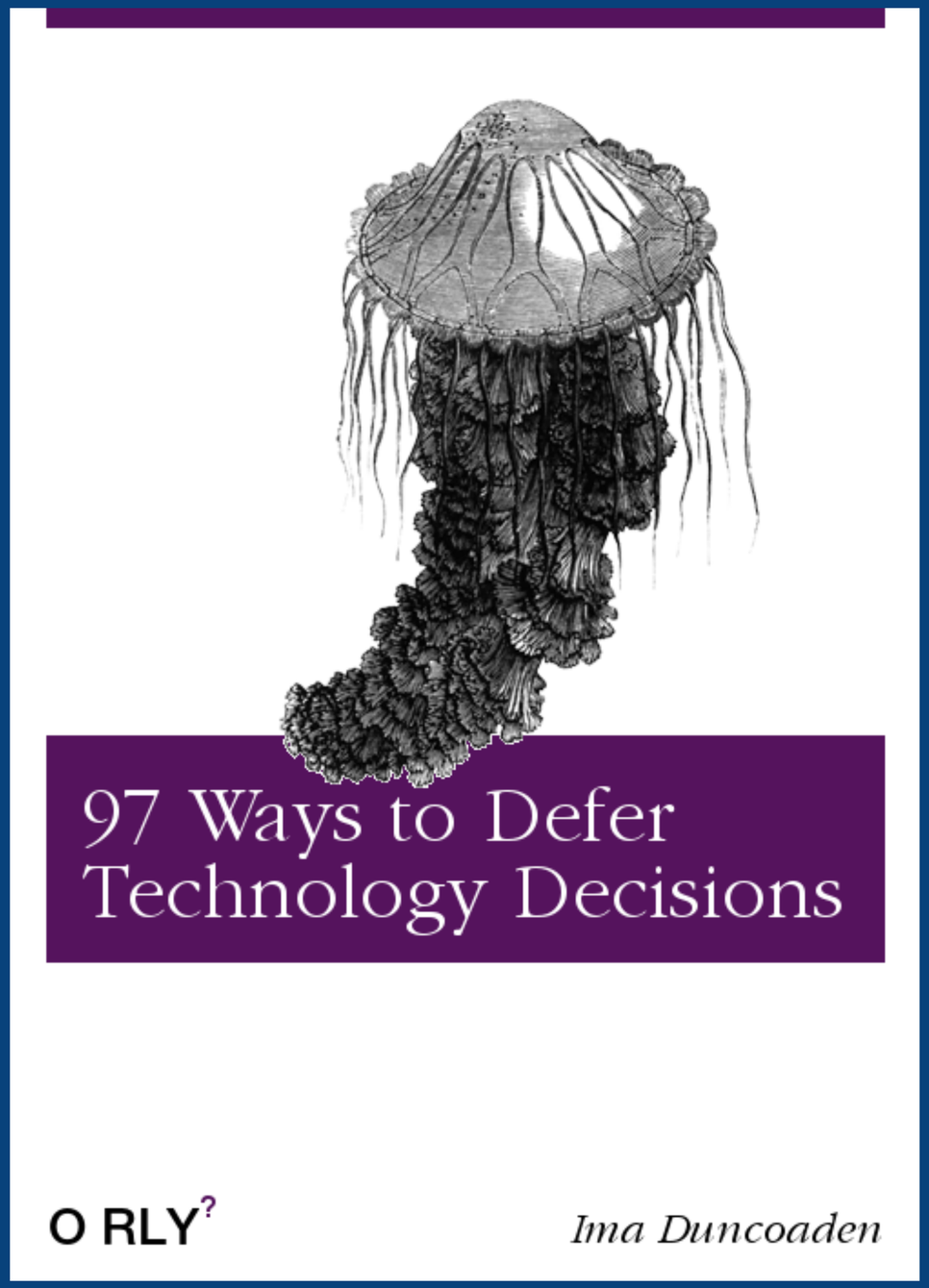
#39

“We don’t want to impose a solution upon the development team.”



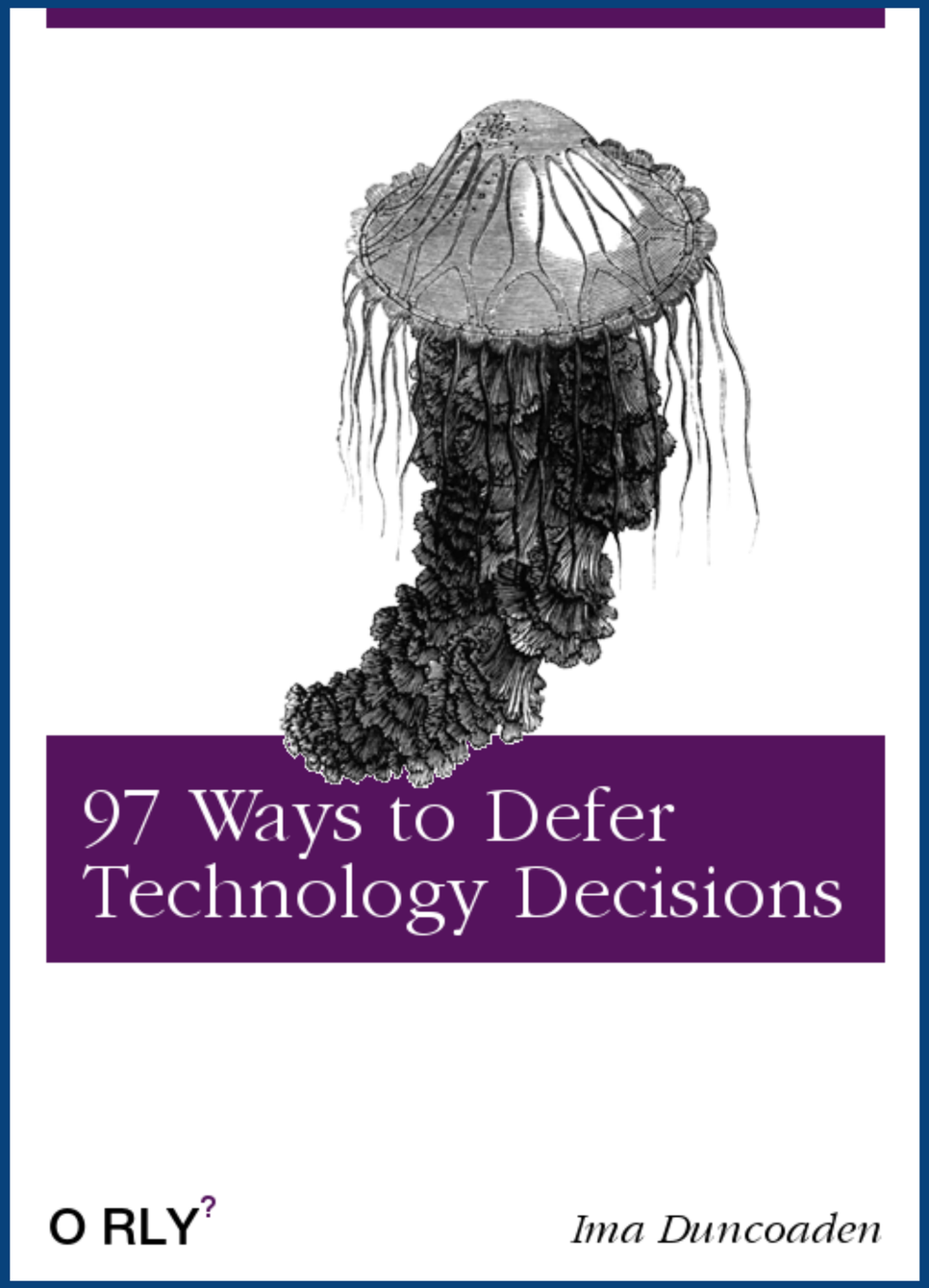
#40

“We leave developers to choose the implementation details.”



#41

“The team should choose the implementation strategy.”

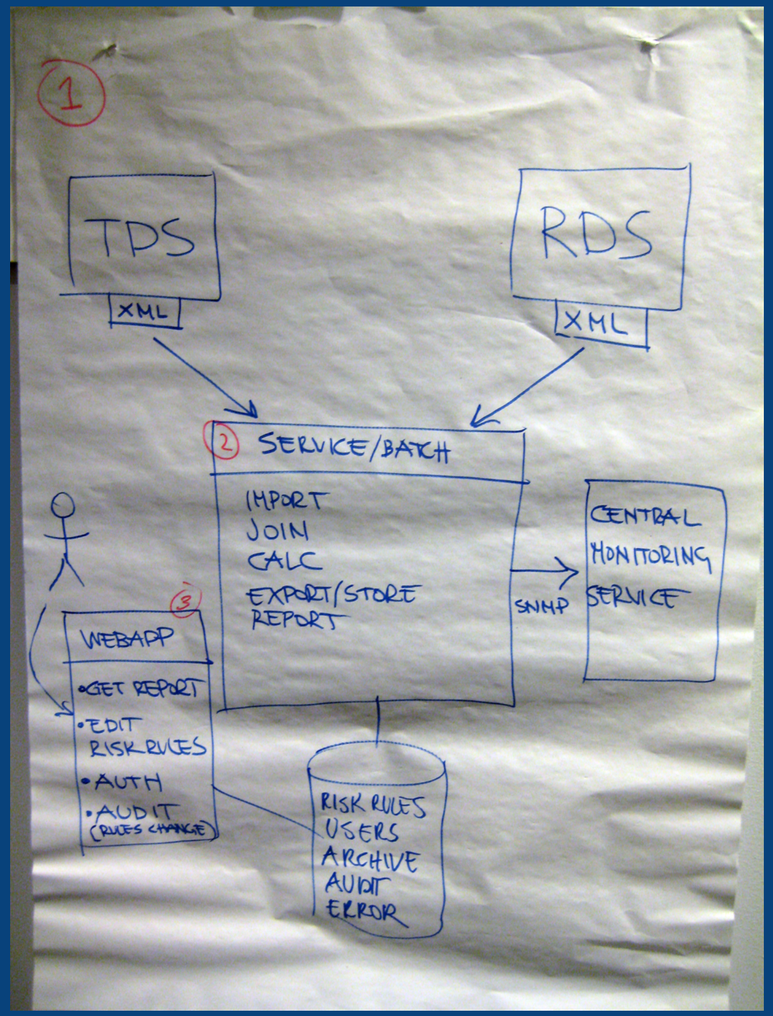


#42

“We’re a Java team, so it’s obviously a Java solution.”

The producer-consumer conflict of software architecture diagrams

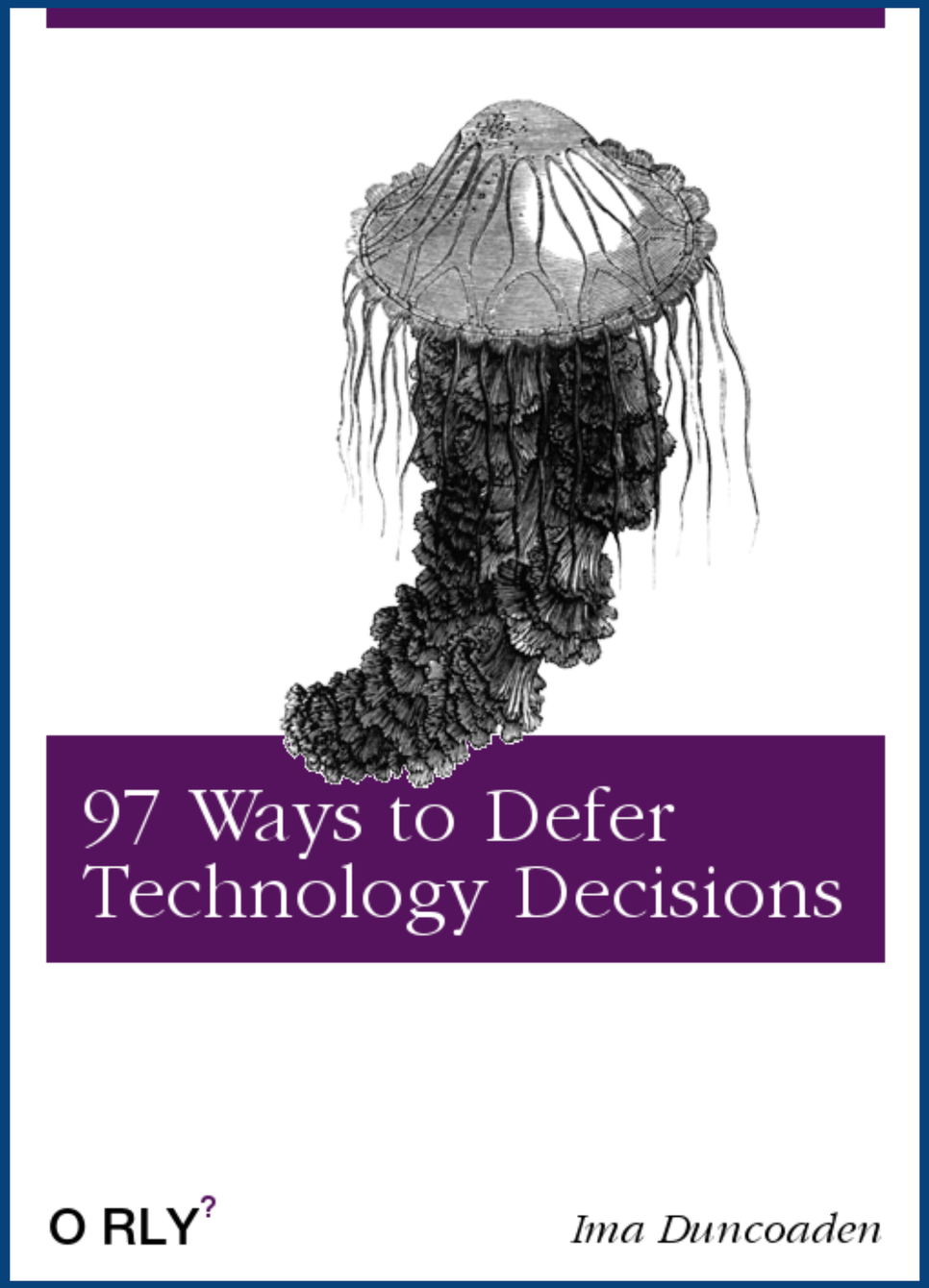
I don’t want to put technology choices on the diagrams…



I wish these diagrams included technology choices…

Producer

Consumer



#97

“We wait until the last responsible moment.”

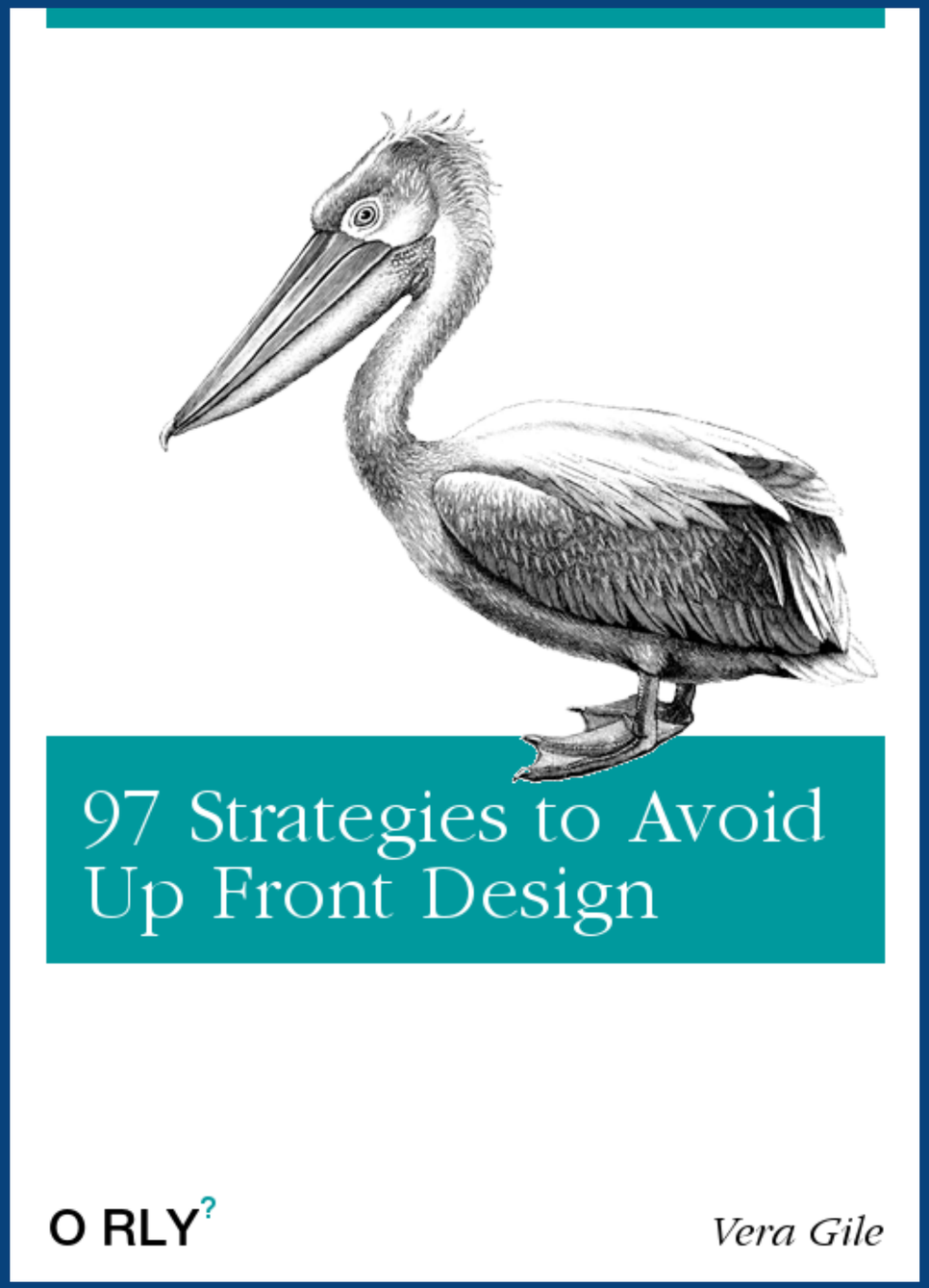
Up Front Design



97 Strategies to Avoid Up Front Design

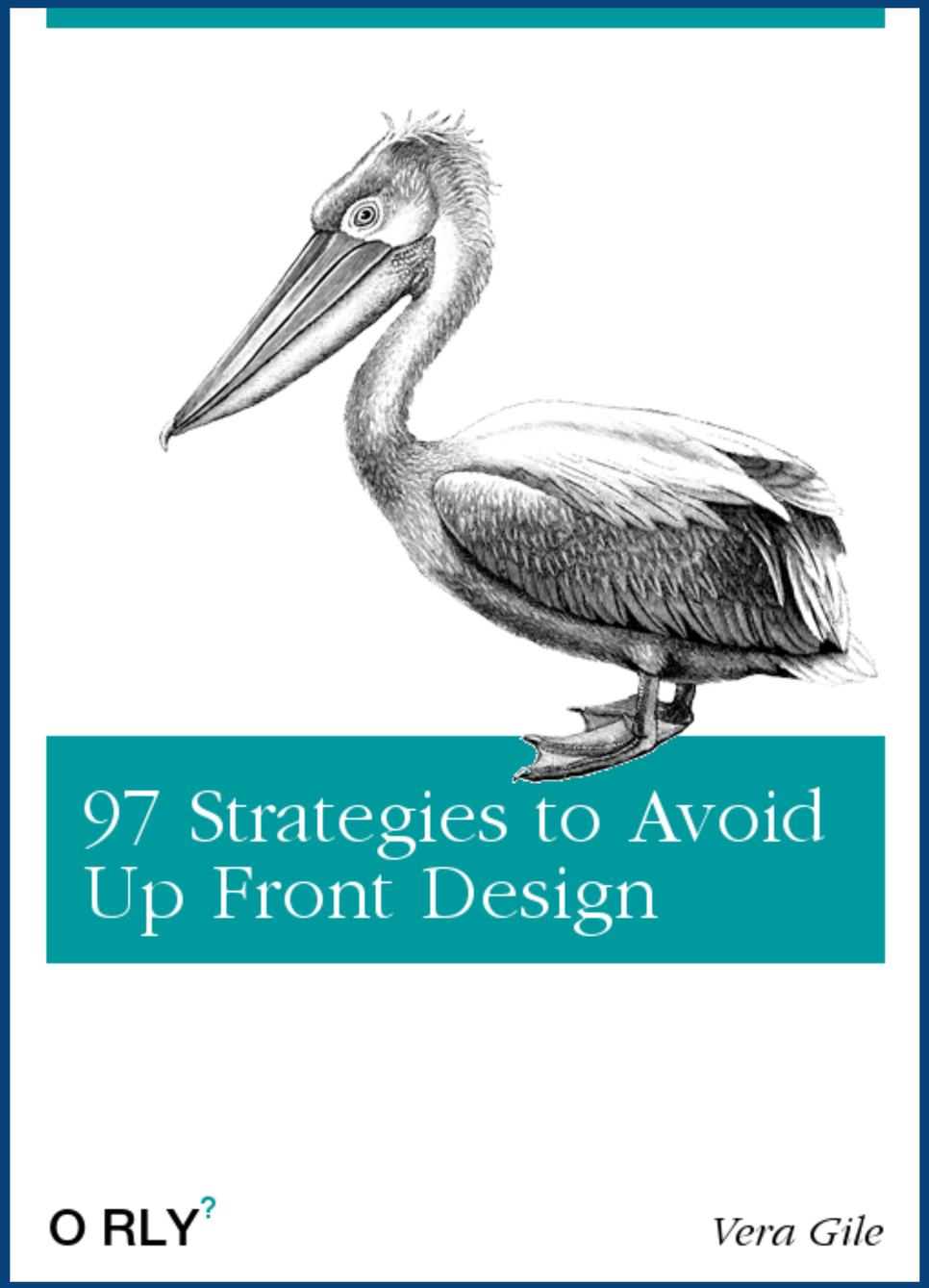
O RLY?

Vera Gile



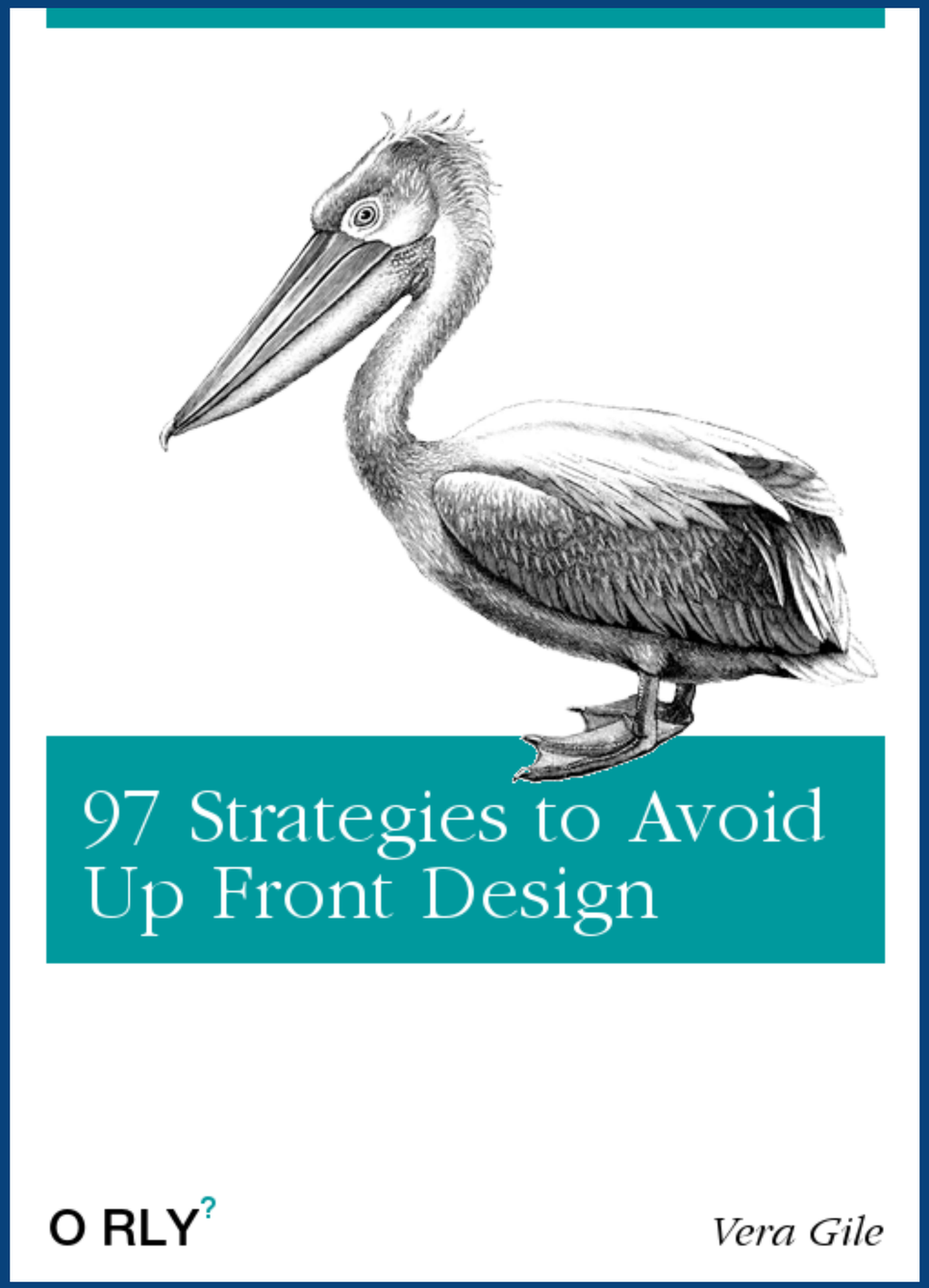
#1

“Are we allowed to do up front design?”



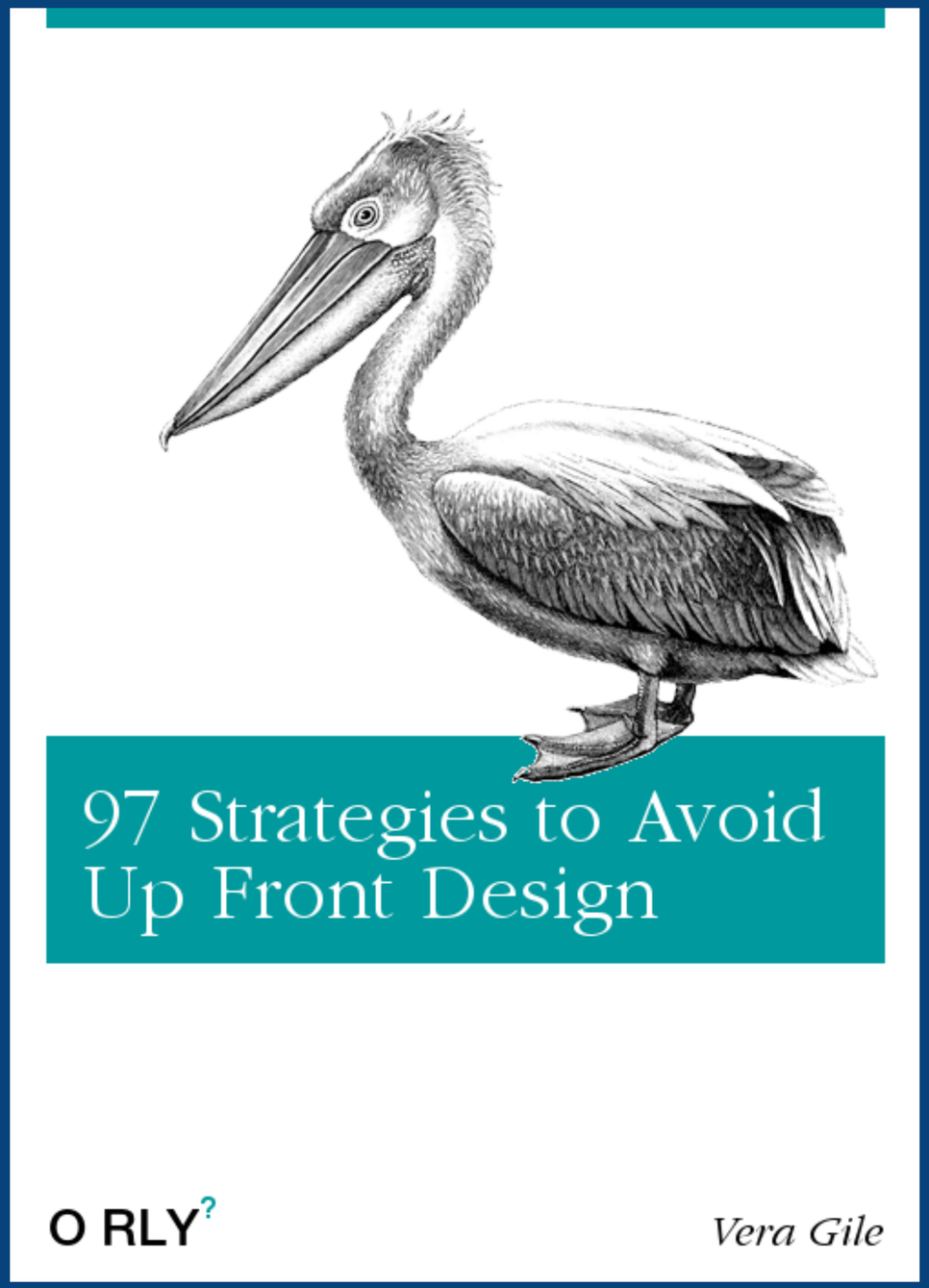
#12

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



#13

“We’re agile.”



#17

“It’s not expected in agile.”

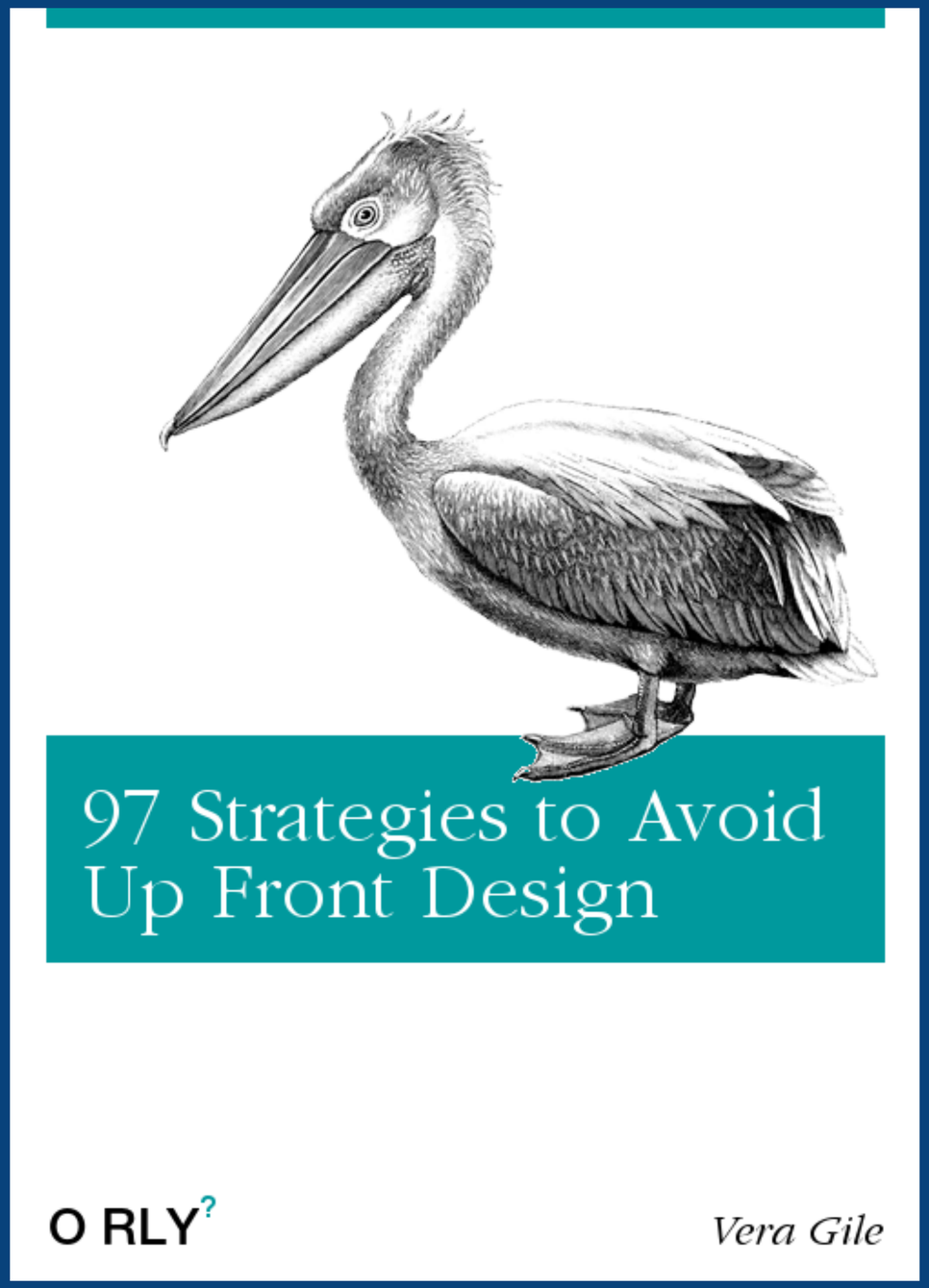
There is no Big Design Up Front. Most of the design activity takes place on the fly and incrementally, starting with "the simplest thing that could possibly work" and adding complexity only when it's required by failing tests.

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

What role does an architecture play when you are using evolutionary design? Again XPs critics state that XP ignores architecture, that XP's route is to go to code fast and trust that refactoring that will solve all design issues. Interestingly they are right, and that may well be weakness. **Certainly the most aggressive XPers** - Kent Beck, Ron Jeffries, and Bob Martin - **are putting more and more energy into avoiding any up front architectural design**. Don't put in a database until you really know you'll need it. Work with files first and refactor the database in during a later iteration.

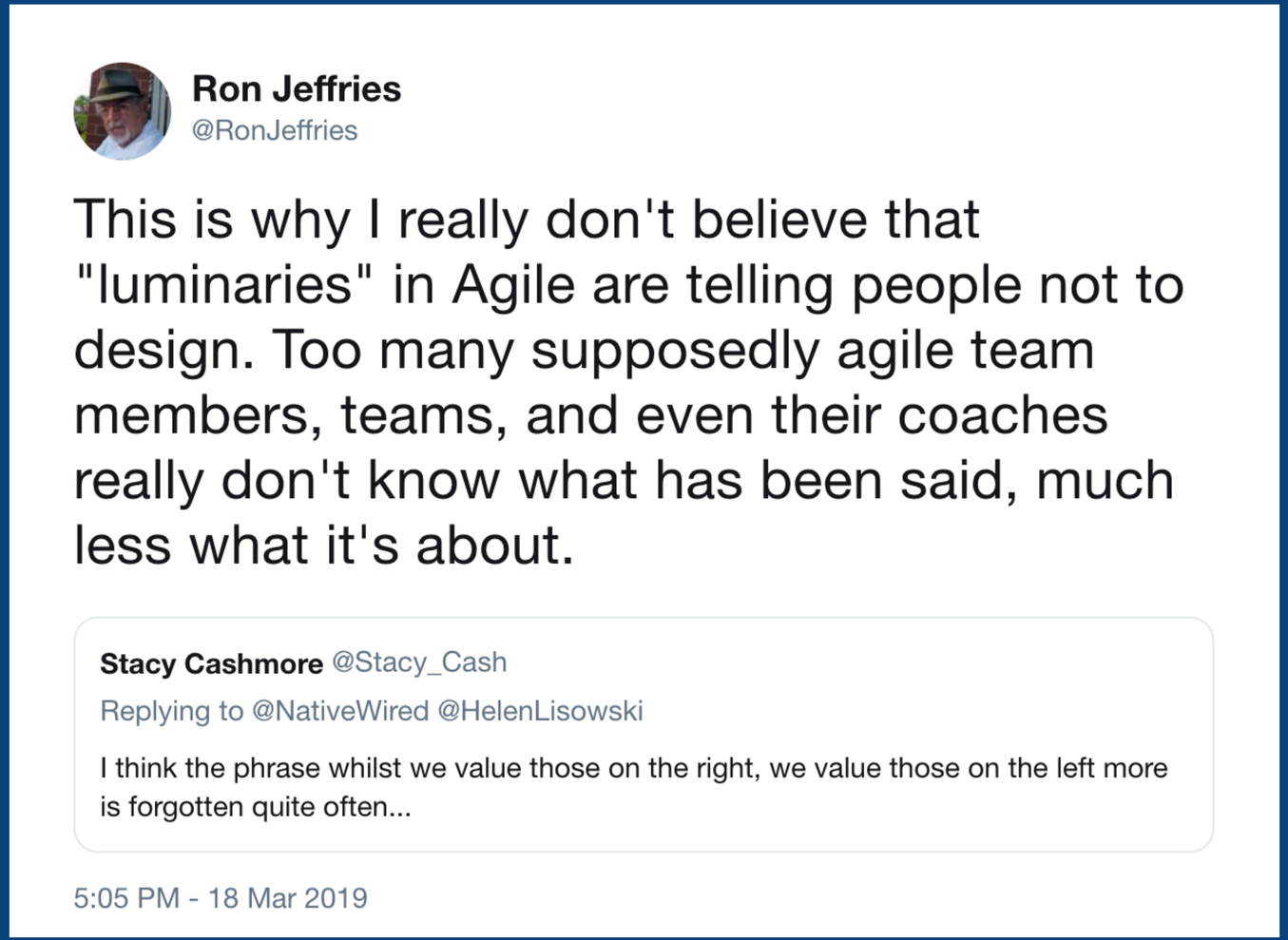
Martin Fowler

https://martinfowler.com/articles/designDead.html



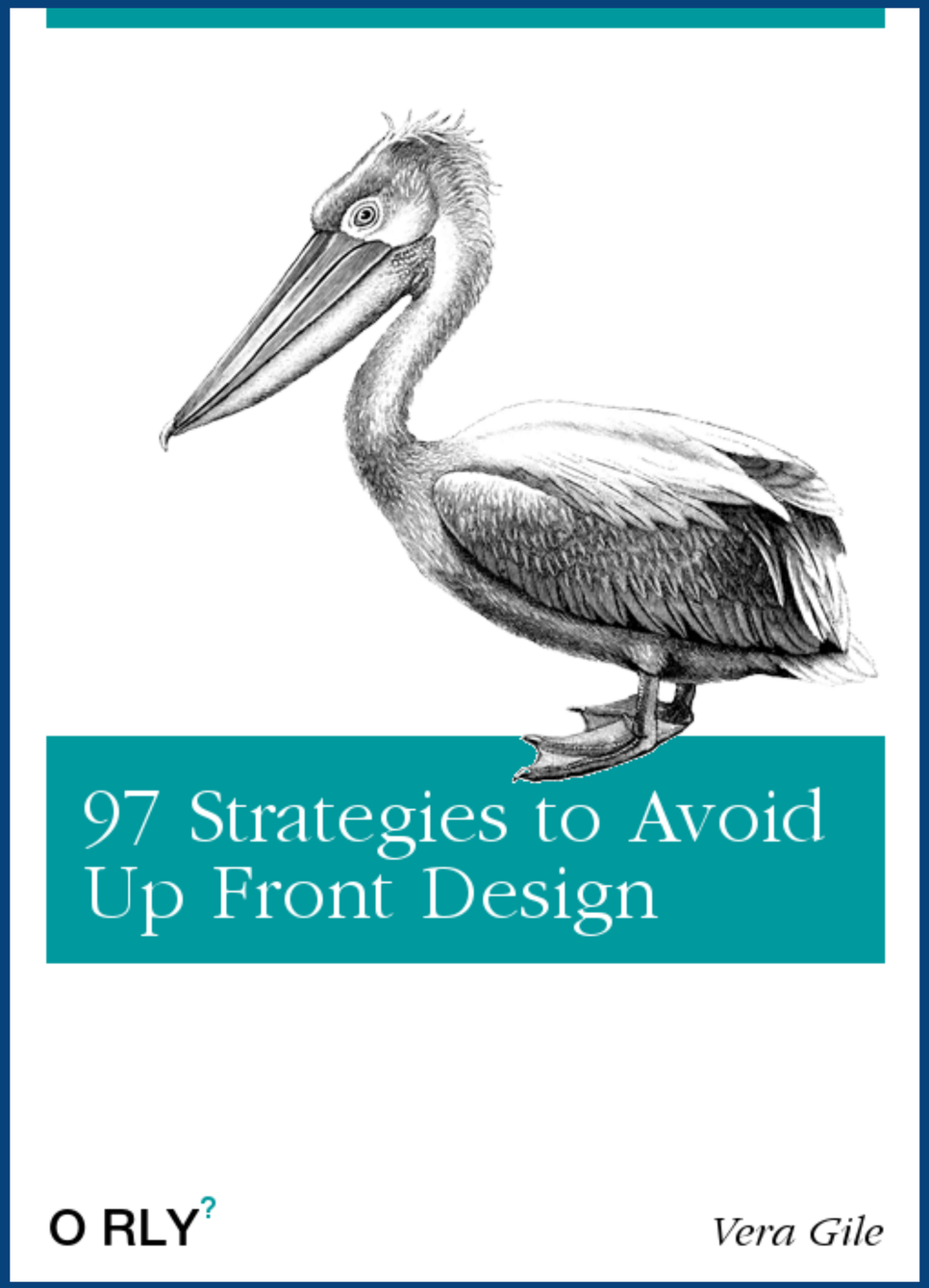
#52

“I’m good with maybe a day for a one-year effort.”



The “luminaries” in Agile are not telling people to do design either

(it’s easy to make assumptions about what's **not** being said)



#88

“I'm guessing you are not a programmer.”

Talking about it doesn’t make you better.
Doing does.



I do programming things too. 🙃

Remember that the folks behind the agile manifesto have a lot of experience.

Most teams likely don’t have that same level of experience.



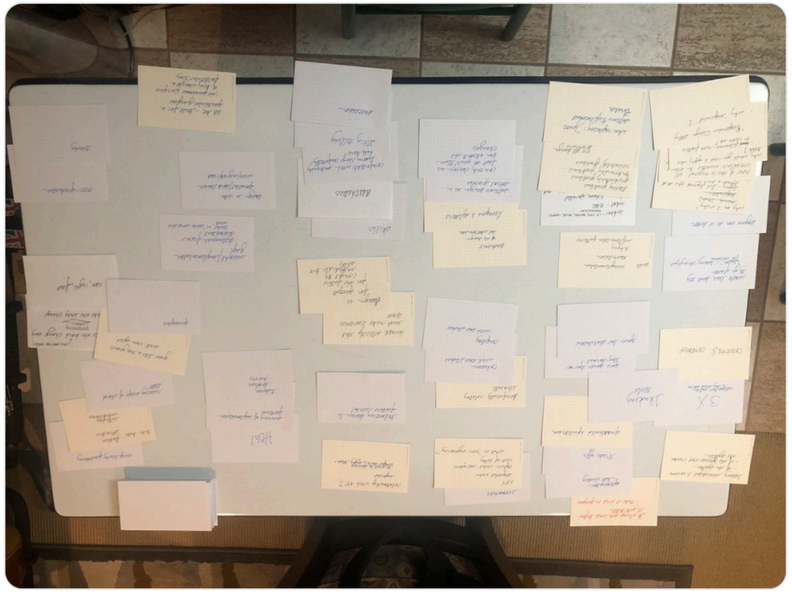
Simon Brown @simonbrown · Mar 18

"To write a book I need to see a whole, in part just to reduce my anxiety." ... I'm the same when I'm building software. 😉 #SomeDesignUpFront

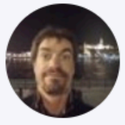


Kent Beck ✔
@KentBeck

I unexpectedly have two weeks free before I start my new job (more on that later). I decided to time-box write the book on software design that's been ripening in my head for a decade or two. Here's the outline:



1:24 AM - 18 Mar 2019



John Hearn @johnhearnbcn · Mar 18

Replying to @KentBeck

Interesting that you write an outline first. My daughter has discovered that writing with flow, keeping notes and refactoring as she goes gives her better results

2 5



Kent Beck ✔ @KentBeck · Mar 18

I've done ~20,000 words on software design (maybe more—not going to stop to check...not). To write a book I need to see a whole, in part just to reduce my anxiety.

1 17

Agility requires a toolbox of techniques and practices but many people don’t have them, and we’ve stopped teaching them

How do you design software?

“we use a whiteboard”

“we draw boxes and lines”

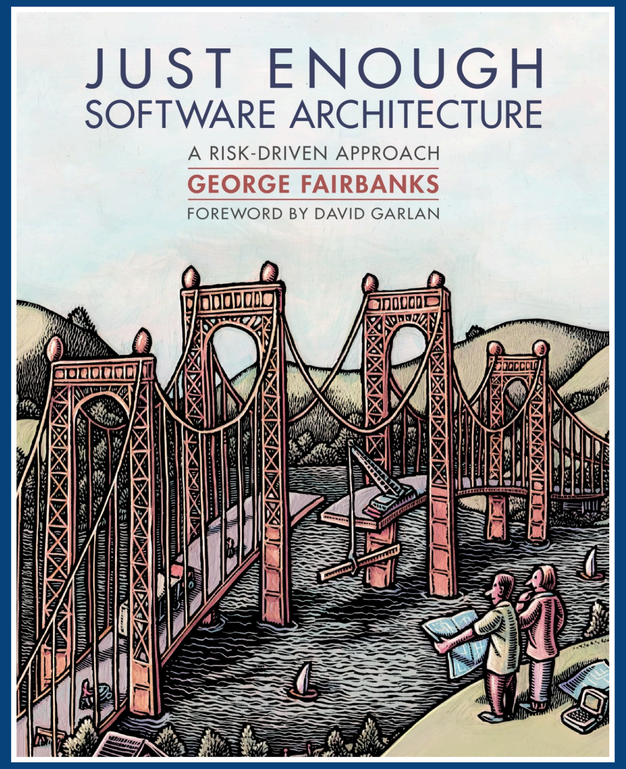
“ the boxes
represent components ”

we use our

experience

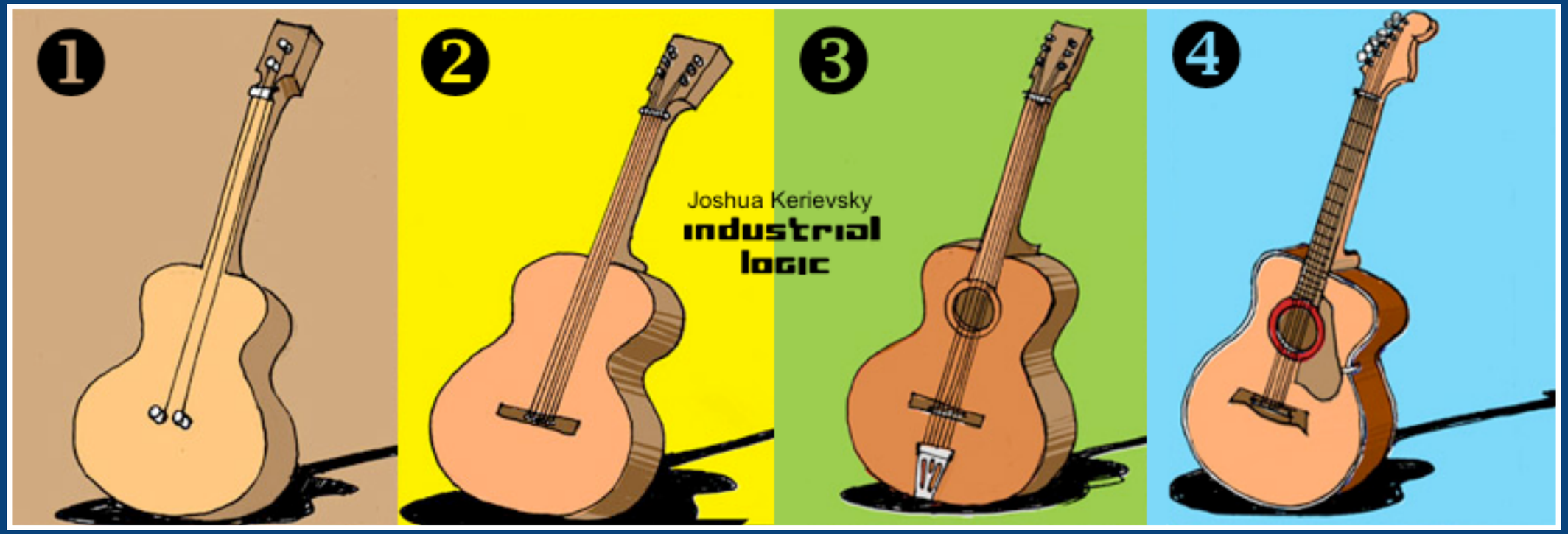
> Continuous attention to technical excellence and good design enhances agility.

Principle 9 of the Manifesto for Agile Software Development



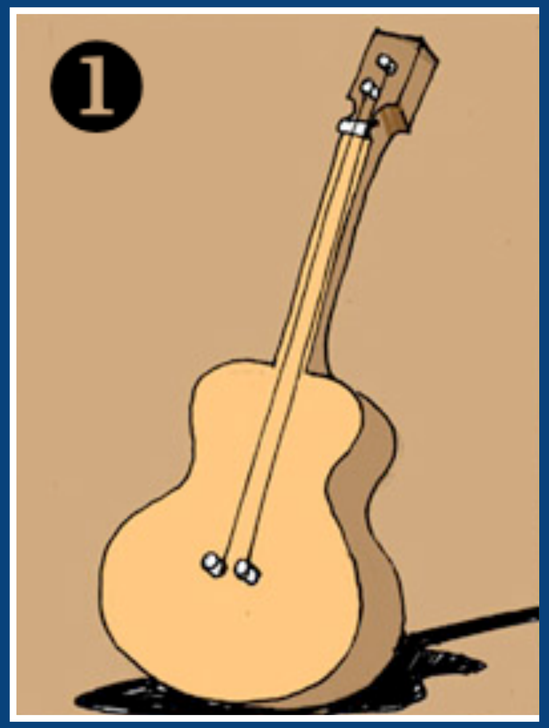
A good architecture rarely happens through **architecture-indifferent design**

Up front design is not necessarily about creating a perfect end-state or complete architecture



Evolutionary Design

Beginning With A Primitive Whole



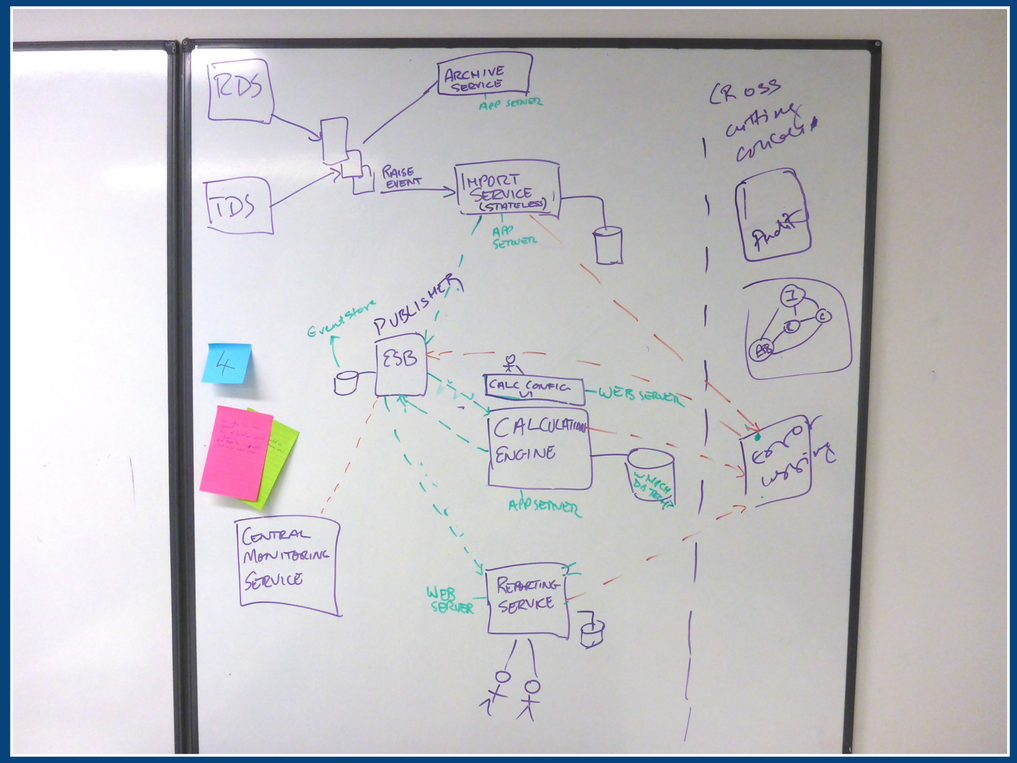
Evolutionary Design

Beginning With A Primitive Whole

Enough up front design
to create a good
starting point and **direction**

Part of the design activity is about discovering “unknown unknowns”

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



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

We’re not trying to
make every decision

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

Grady Booch

Architecture

Programming languages
Technologies and platforms
Monolith, microservices or hybrid approach

Design

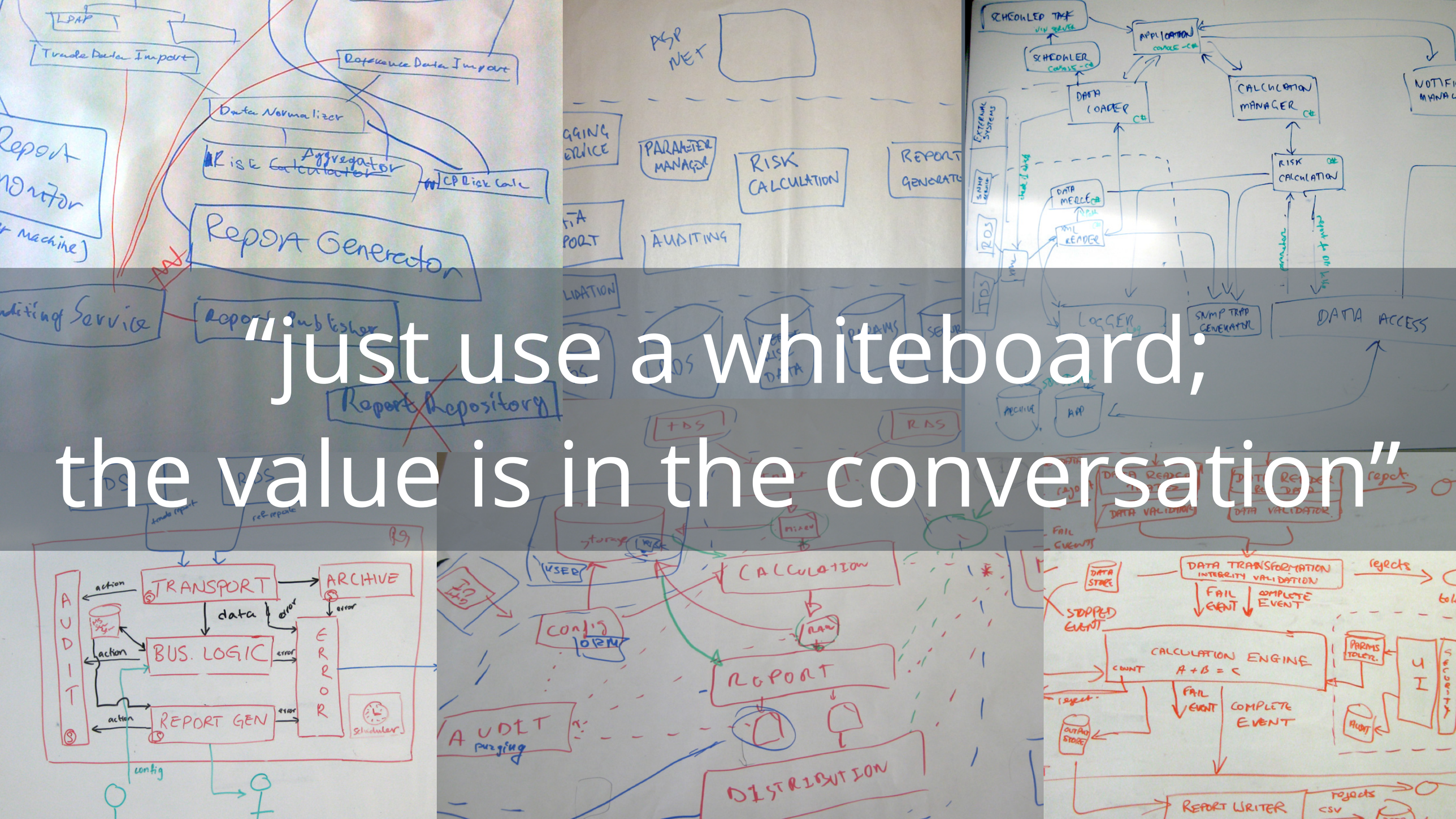
Implementation

Curly braces on the same or next line
Whitespace vs tabs

I think there is a role for a broad starting point architecture. Such things as stating early on how to layer the application, how you'll interact with the database (if you need one), what approach to use to handle the web server.

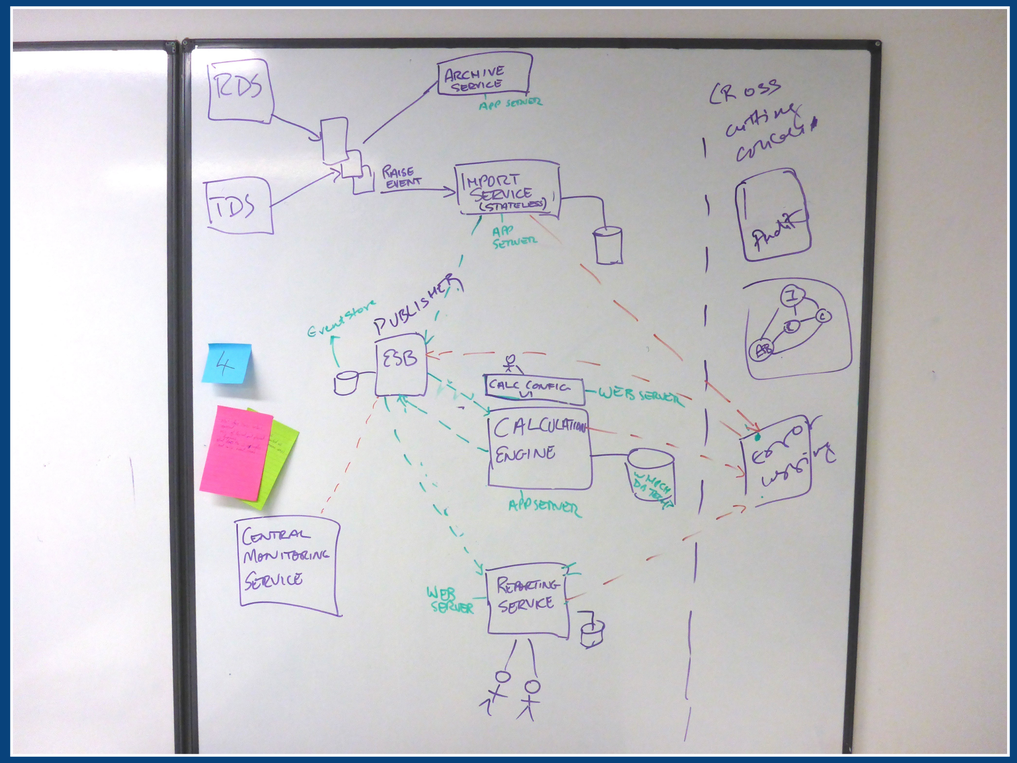
Martin Fowler

https://martinfowler.com/articles/designDead.html



“just use a whiteboard; the value is in the conversation”

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



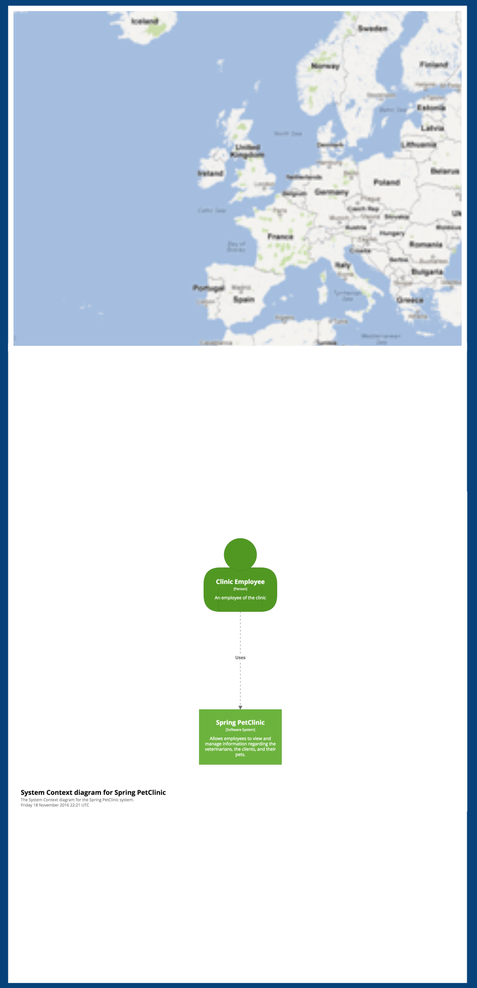
2. Is it going to **work**?

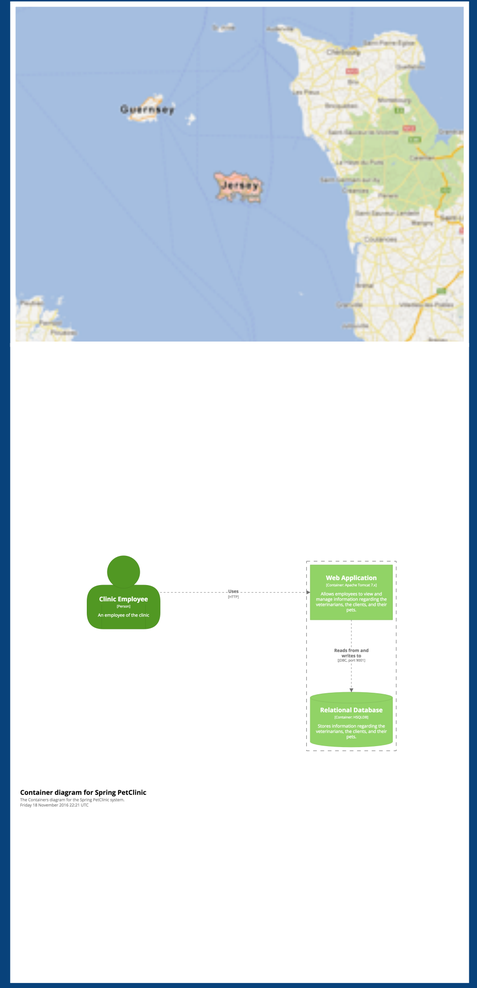
Teams need a **ubiquitous language** to communicate effectively

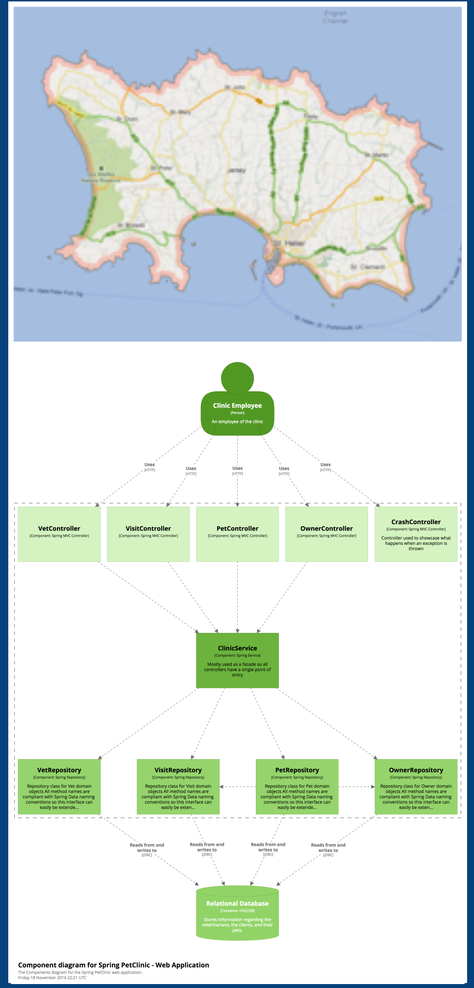
C4

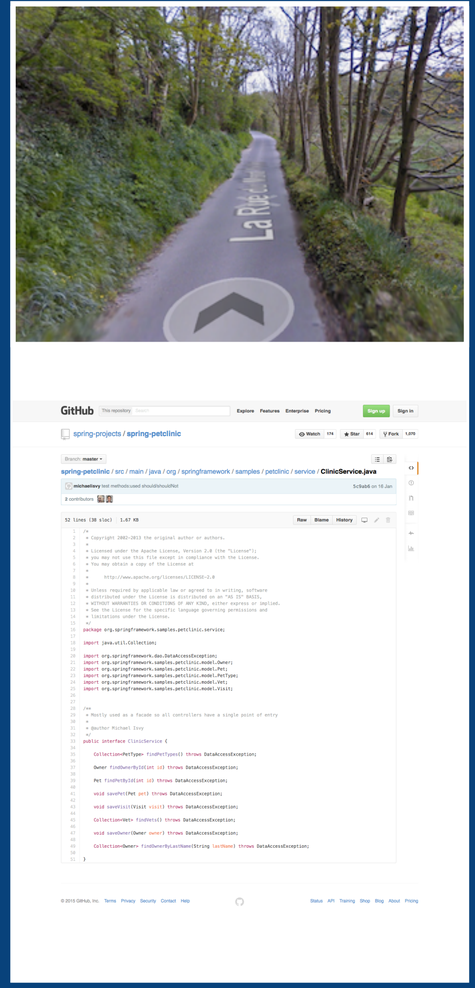
Context, Containers, Components and Code

(c4model.com for more information)





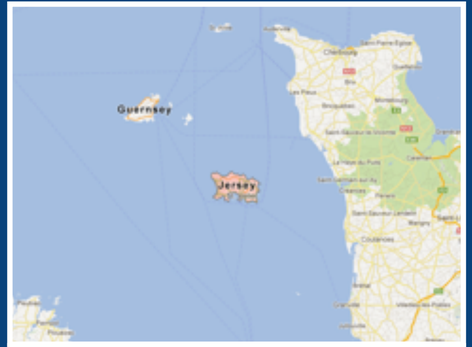




Diagrams are maps

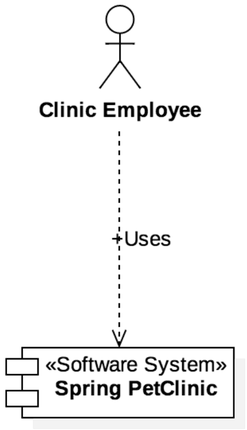
that help software developers navigate a large and/or complex codebase

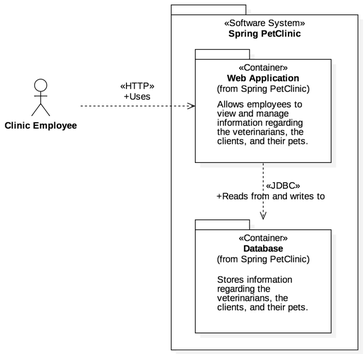


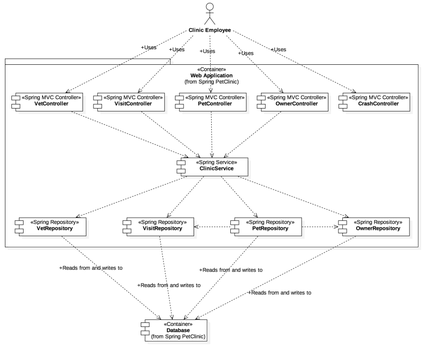


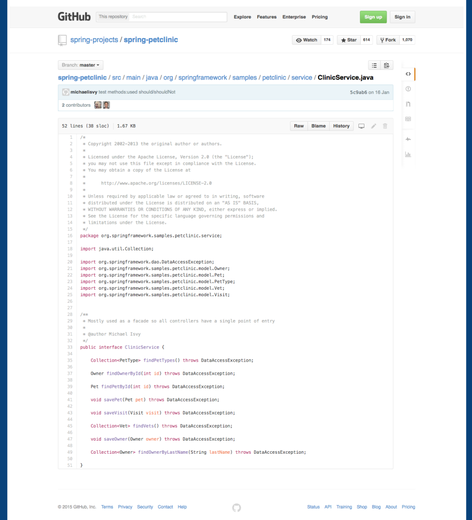












Diagrams are maps

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

Diagrams are a visual checklist for design decisions

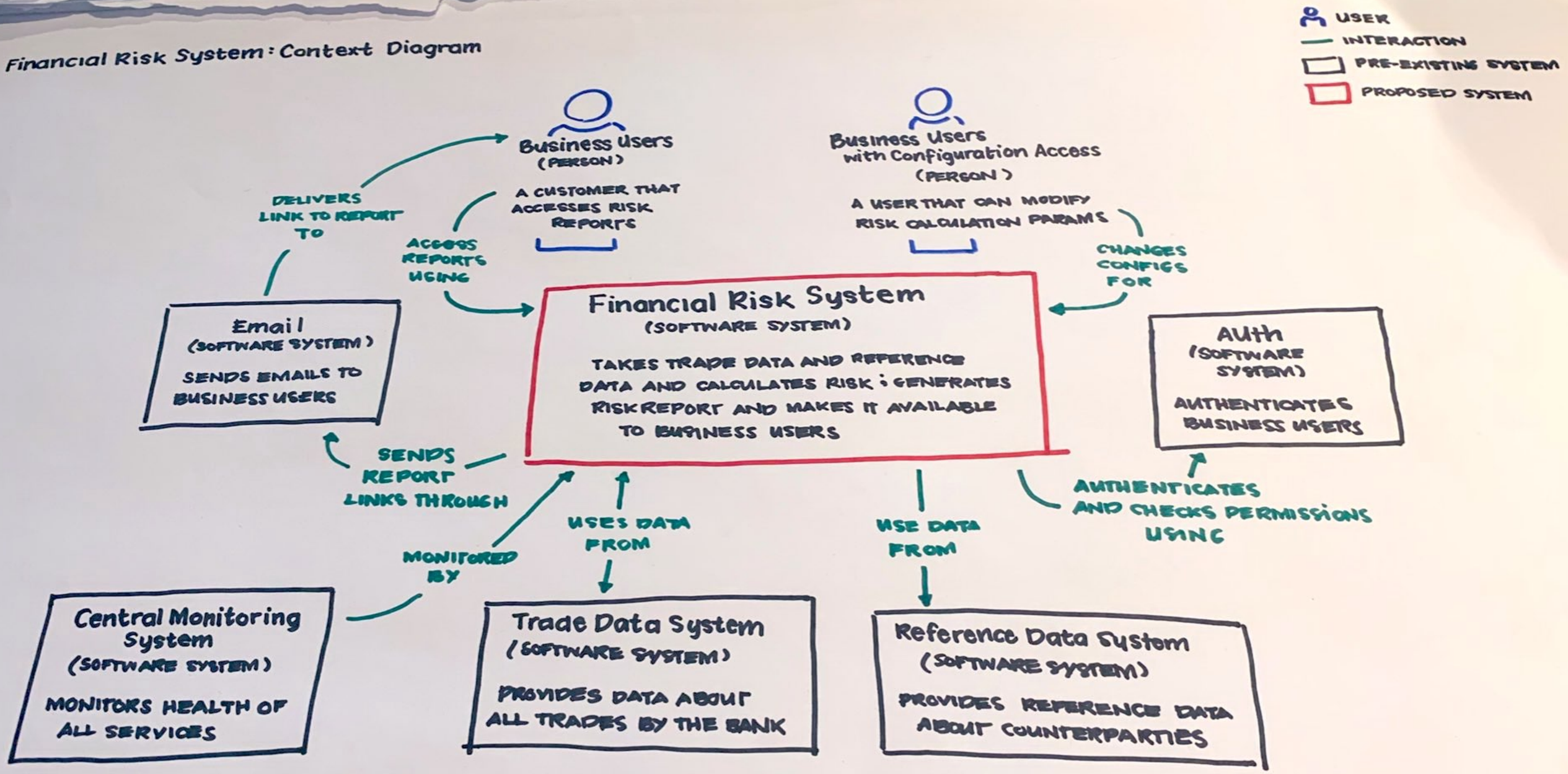
System Context diagram

What is the scope of the software system we’re building?

Who is using it? What are they doing?

What system integrations does it need to support?

Financial Risk System: Context Diagram

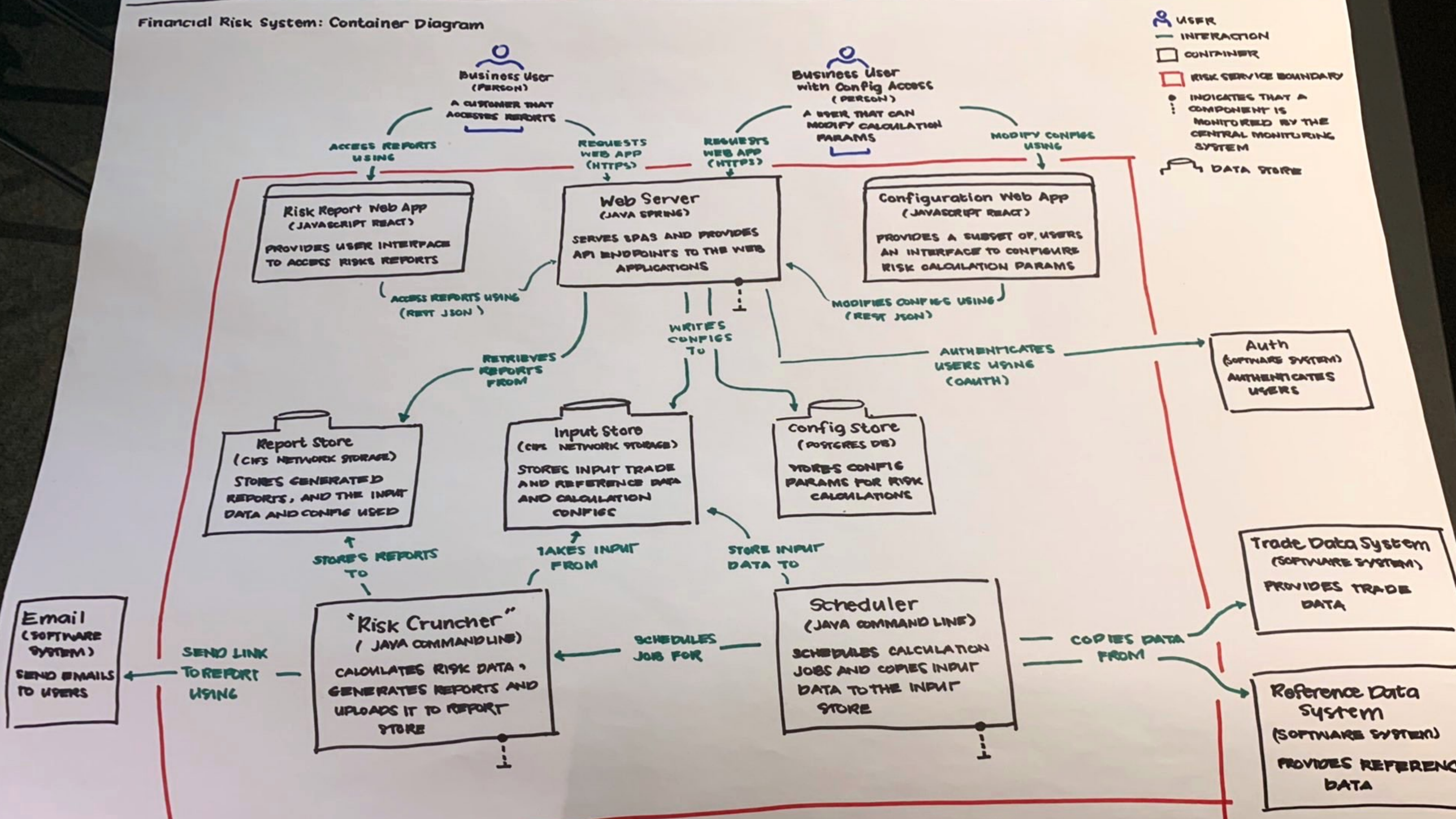


Container diagram

What are the major technology building blocks?

What are their responsibilities?

How do they communicate?


Financial Risk System: Container Diagram
USER
INTERACTION
CONTAINER
RISK SERVICE BOUNDARY
INDICATES THAT A COMPONENT IS MONITORED BY THE CENTRAL MONITORING SYSTEM
DATA STORE
Business User
(PERSON)
A CUSTOMER THAT ACCESSES REPORTS
Business User with Config Access
(PERSON)
A USER THAT CAN MODIFY CALCULATION PARAMS
ACCESS REPORTS USING
REQUESTS WEB APP (HTTPS)
REQUESTS WEB APP (HTTPS)
MODIFY CONFIGS USING
Risk Report Web App
(JAVASCRIPT REACT)
PROVIDES USER INTERFACE TO ACCESS RISK REPORTS
Web Server
(JAVA SPRING)
SERVES SPAS AND PROVIDES API ENDPOINTS TO THE WEB APPLICATIONS
Configuration Web App
(JAVASCRIPT REACT)
PROVIDES A SUBSET OF USERS AN INTERFACE TO CONFIGURE RISK CALCULATION PARAMS
ACCESS REPORTS USING (REST JSON)
MODIFIES CONFIGS USING (REST JSON)
WRITES CONFIGS TO
RETRIEVES REPORTS FROM
AUTHENTICATES USERS USING (OAUTH)
Auth
(SOFTWARE SYSTEM)
AUTHENTICATES USERS
Report Store
(CIFS NETWORK STORAGE)
STORES GENERATED REPORTS, AND THE INPUT DATA AND CONFIG USED
Input Store
(CIFS NETWORK STORAGE)
STORES INPUT TRADE AND REFERENCE DATA AND CALCULATION CONFIGS
Config Store
(POSTGRES DB)
STORES CONFIG PARAMS FOR RISK CALCULATIONS
STORES REPORTS TO
TAKES INPUT FROM
STORE INPUT DATA TO
Email
(SOFTWARE SYSTEM)
SEND EMAILS TO USERS
SEND LINK TO REPORT USING
"Risk Cruncher"
(JAVA COMMANDLINE)
CALCULATES RISK DATA, GENERATES REPORTS AND UPLOADS IT TO REPORT STORE
SCHEDULES JOB FOR
Scheduler
(JAVA COMMAND LINE)
SCHEDULES CALCULATION JOBS AND COPIES INPUT DATA TO THE INPUT STORE
COPIES DATA FROM
Trade Data System
(SOFTWARE SYSTEM)
PROVIDES TRADE DATA
Reference Data System
(SOFTWARE SYSTEM)
PROVIDES REFERENC
DATA

The diagrams should spark **meaningful questions**

No

“What does that arrow mean?”

“Why are some boxes red?”

“Is that a Java application?”

“Is that a monolithic application, or a collection of microservices?”

“How do the users get their reports?”

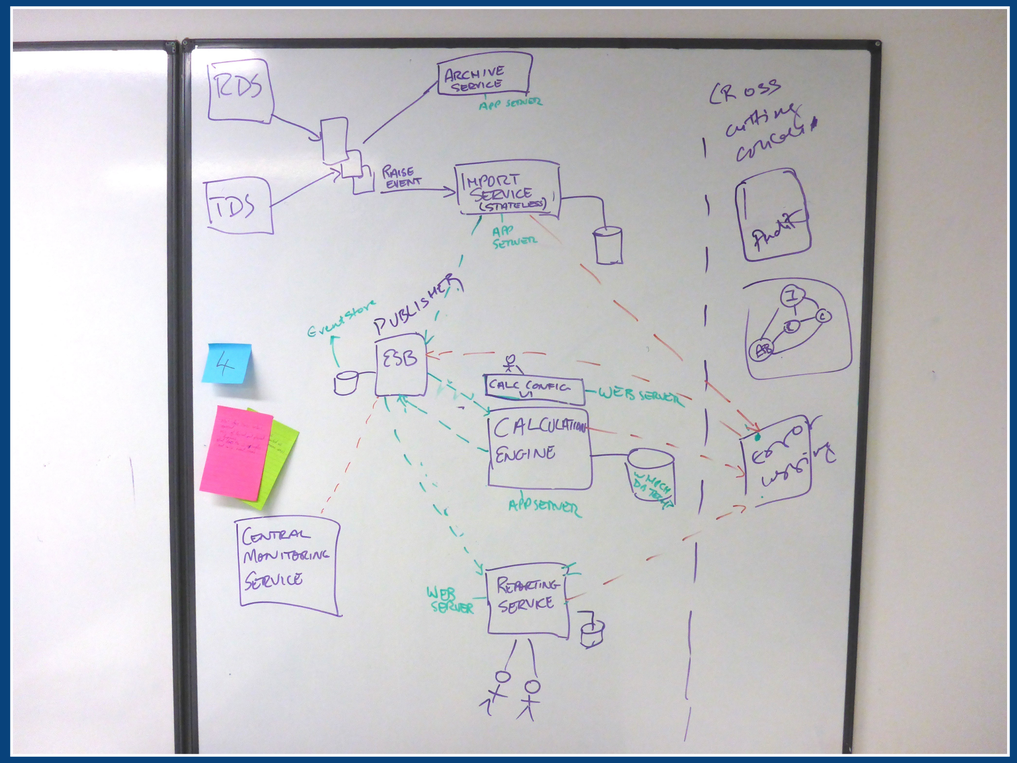
Yes

“What protocol are your two Java applications using
to communicate with each other?”
“Why do you have two separate C# applications instead of one?”
“Why are you using MongoDB?”
“Why are you using MySQL when our standard is Oracle?”
“Should we really build new applications with .NET Framework
rather than .NET Core?”

The diagrams should provide **meaningful feedback**

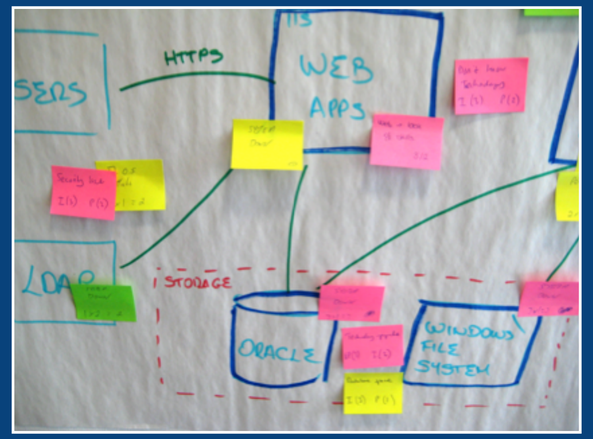
We’re trying to diagram a [microservices|serverless] architecture, but the diagram is getting complicated.

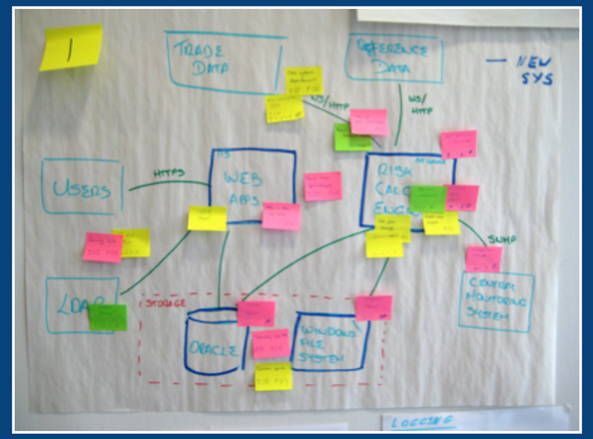
1. Is that what we're going to **build**?

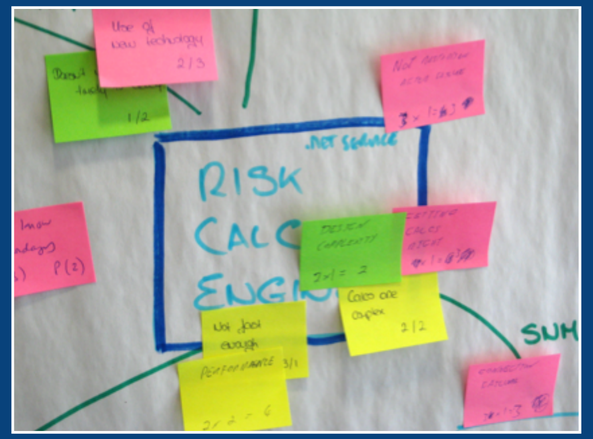


2. Is it going to **work**?

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







Risk-storming

A visual and collaborative technique for identifying risk

How much up front design should you do?

Up front design is an iterative and incremental process; stop when:



You understand the significant architectural drivers.



You understand the context and scope of what you're building.



You understand the significant design decisions (i.e. technology, modularity, etc).

You have a way to communicate your technical vision to other people.



You are confident that your design satisfies the key architectural drivers.



You have identified, and are comfortable with, the risks associated with building the software.



Thank you!

simon@architectis.je

@simonbrown