Five things every developer should know about **software architecture**

Simon Brown

@simonbrown

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

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

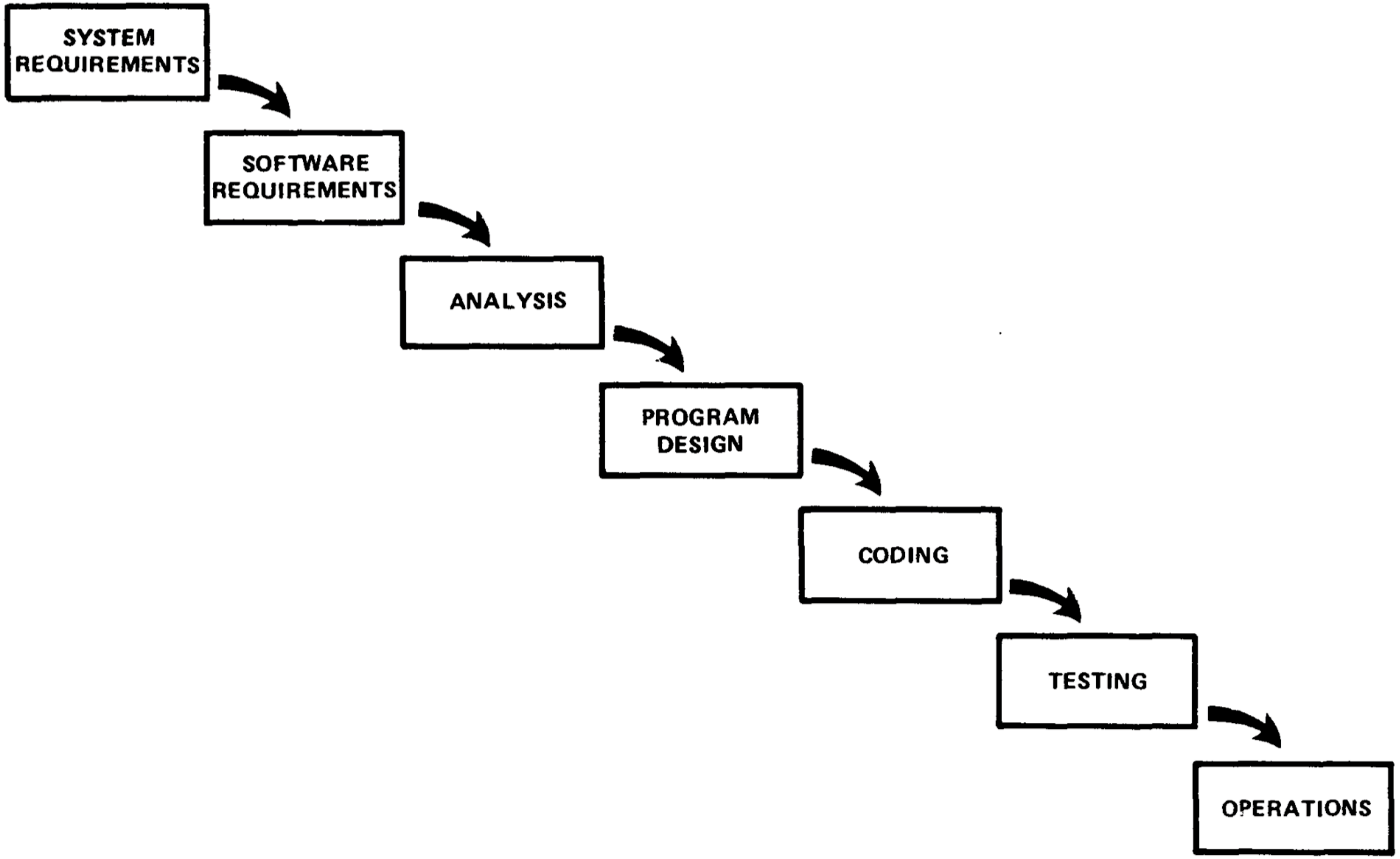


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

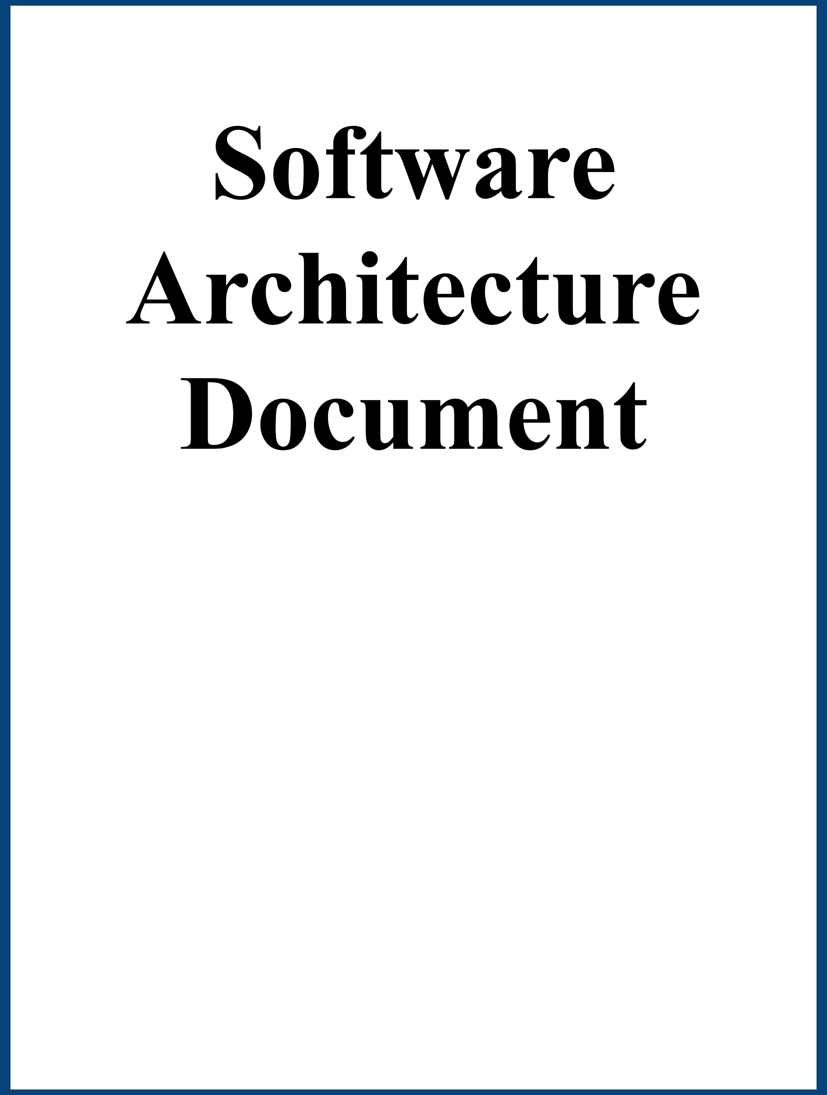
I believe in this concept, but the implementation described above is risky and invites failure.

Managing the development of large software systems

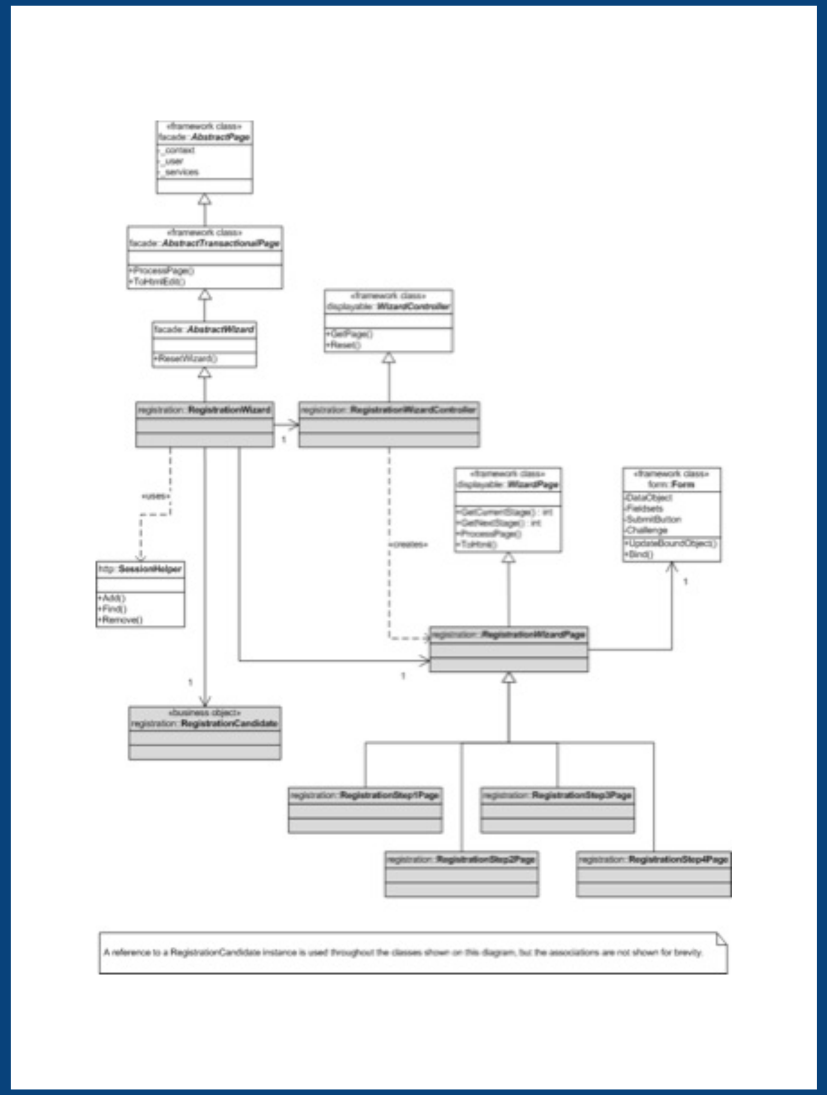
Dr Winston W. Royce

“Responding to change
over
following a plan”

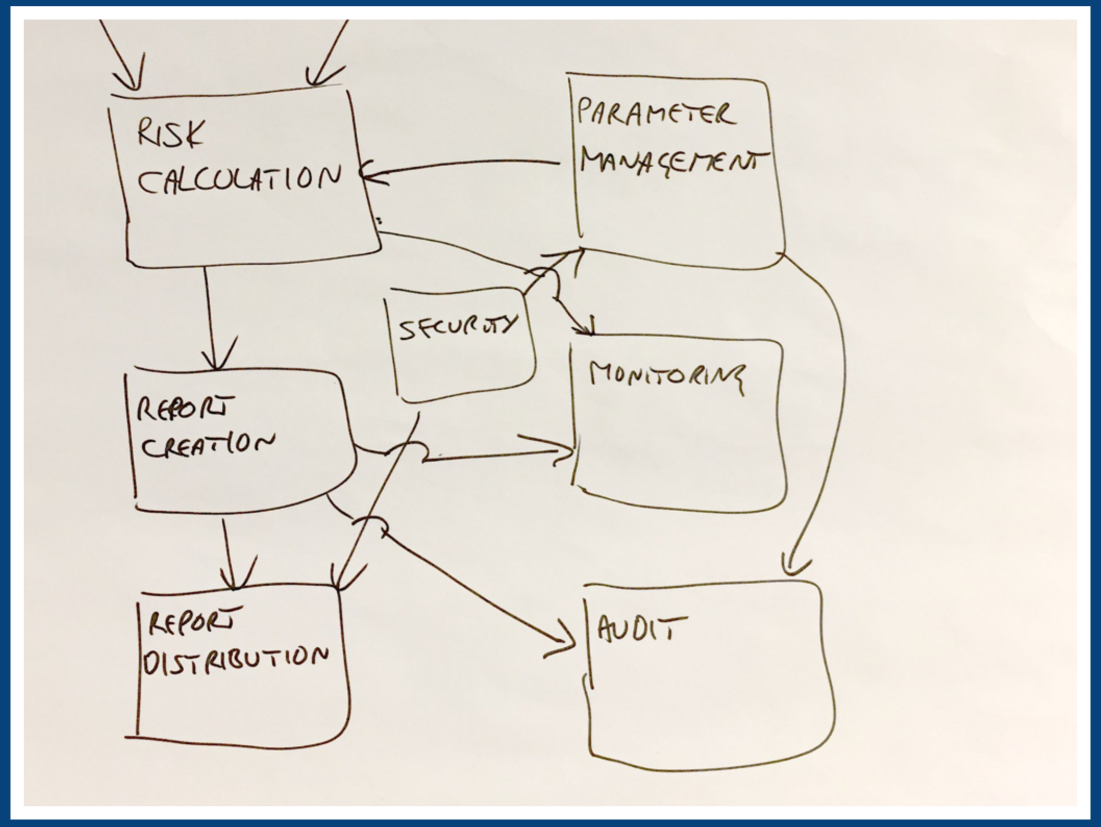
Big design up front



Software Architecture Document



vs



No design up front

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

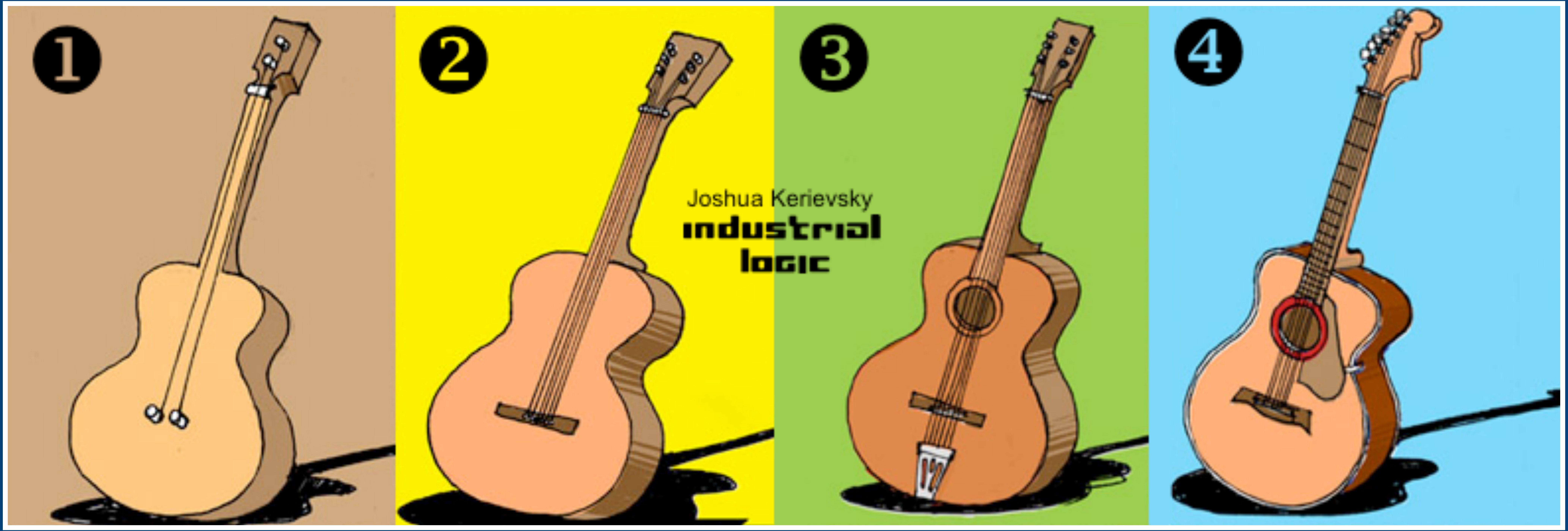
Dave Thomas

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

Sometimes requirements are known, and sometimes they aren’t

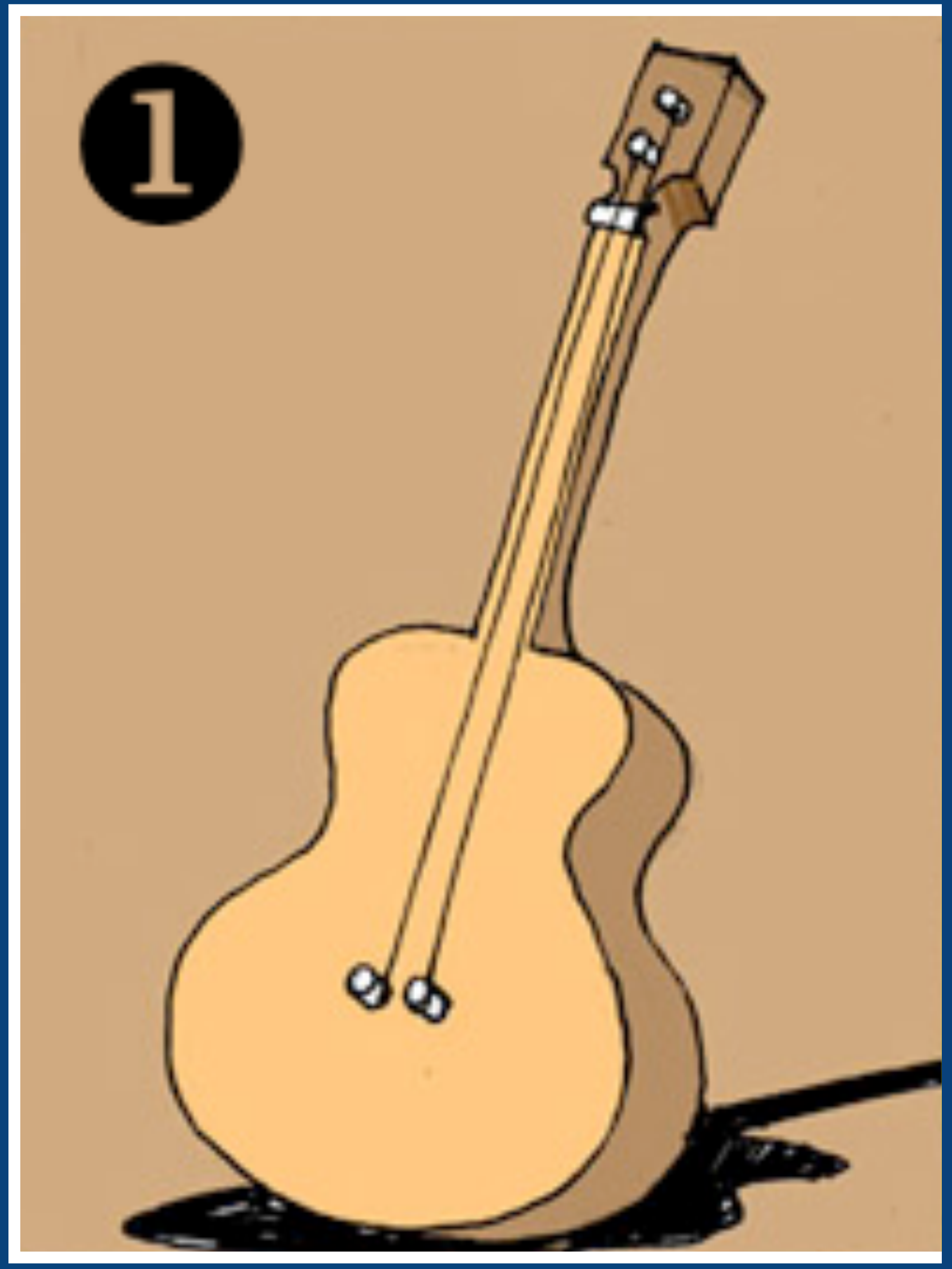
(enterprise software development vs product companies and startups) "just enough"

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



Evolutionary Design

Beginning With A Primitive Whole

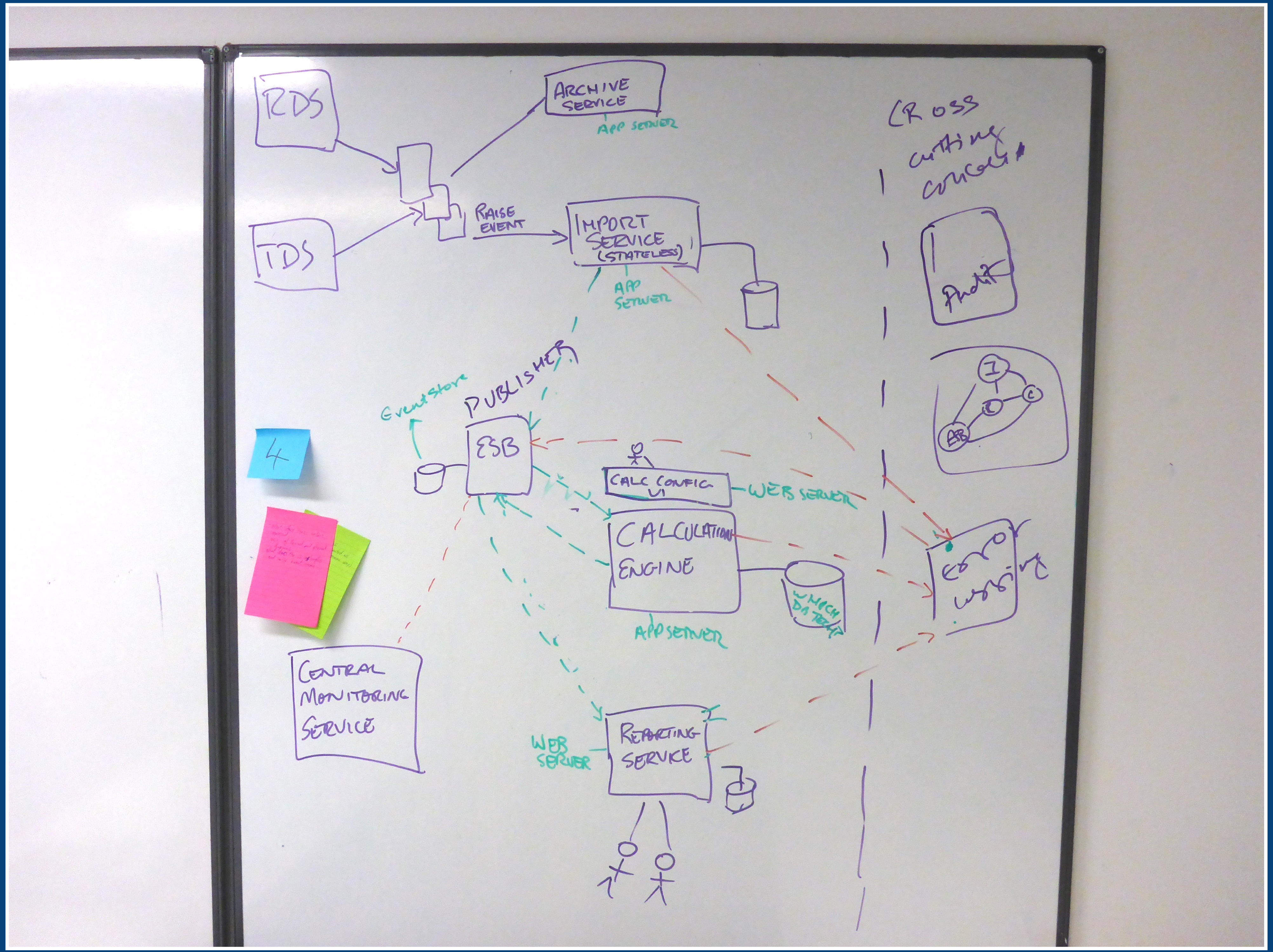


Evolutionary Design

Beginning With A Primitive Whole

A **starting point** adds value

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 languages
Technologies and platforms
Monolith, microservices or hybrid approach

Design

Implementation

Curly braces on the same or next line
Whitespace vs tabs

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

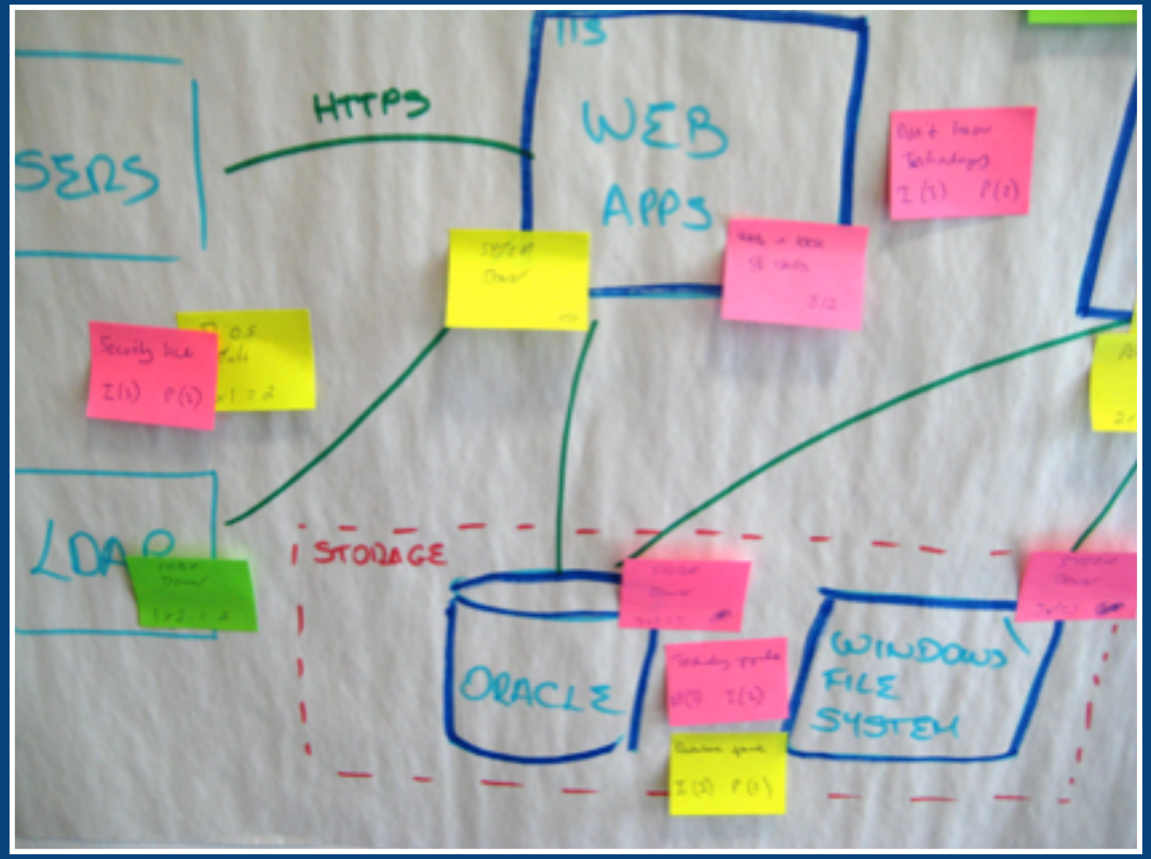
Agile Architecture: Strategies for Scaling Agile Development

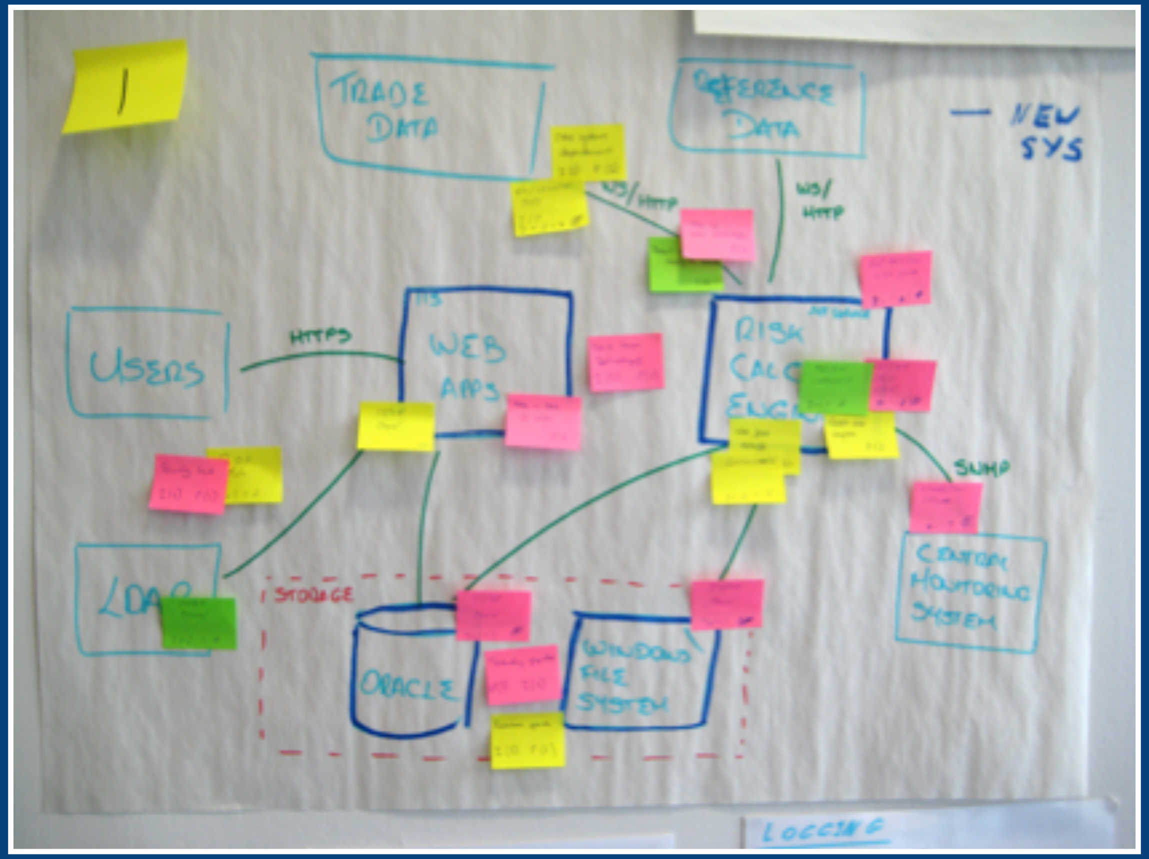
Scott Ambler

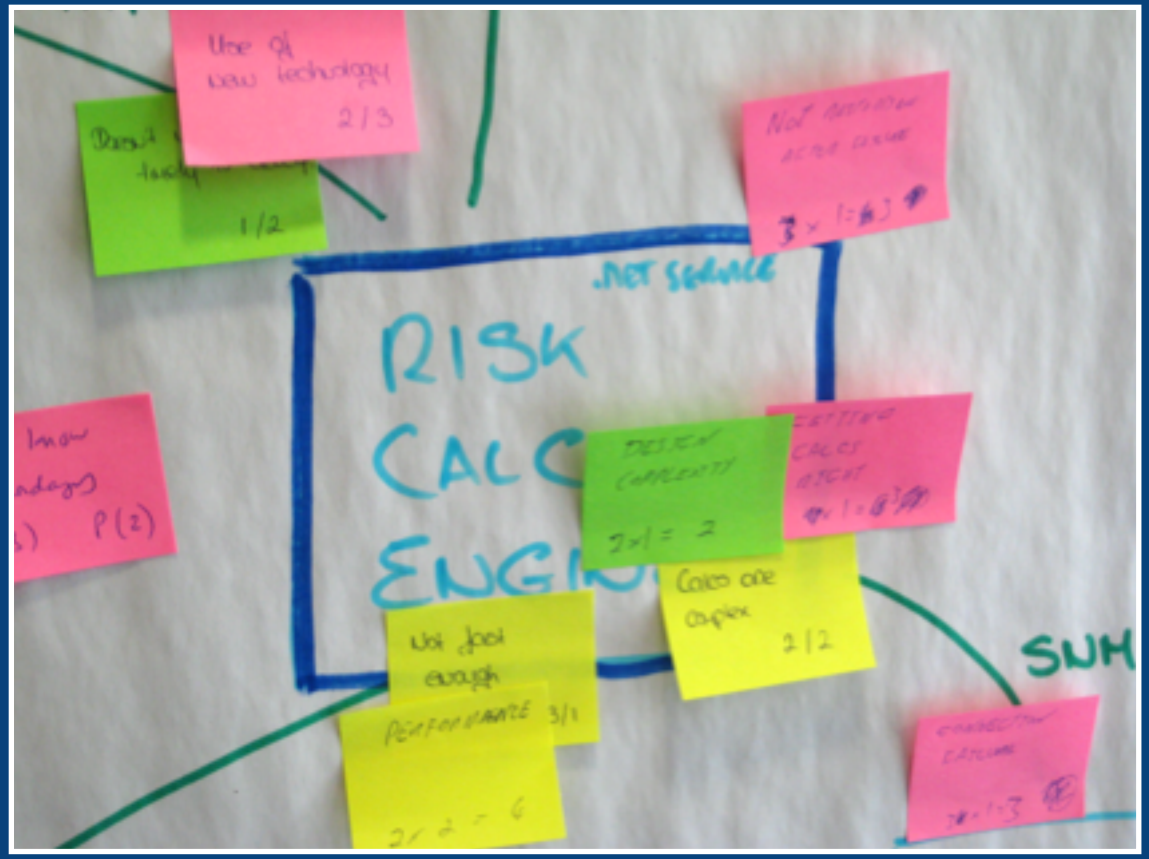
Concrete experiment

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

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?

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

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

- ✓ You understand the significant architectural drivers (requirements, quality attributes, constraints).
- ✓ 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.

Techniques: Workshops, interviews, Event Storming, Impact Mapping, domain modelling, OOAD, CRC, DDD, architecture reviews, ATAM, architecture dry runs, Risk-storming, concrete experiments, C4 model, ADRs, etc.

2. Every software team needs to consider software 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

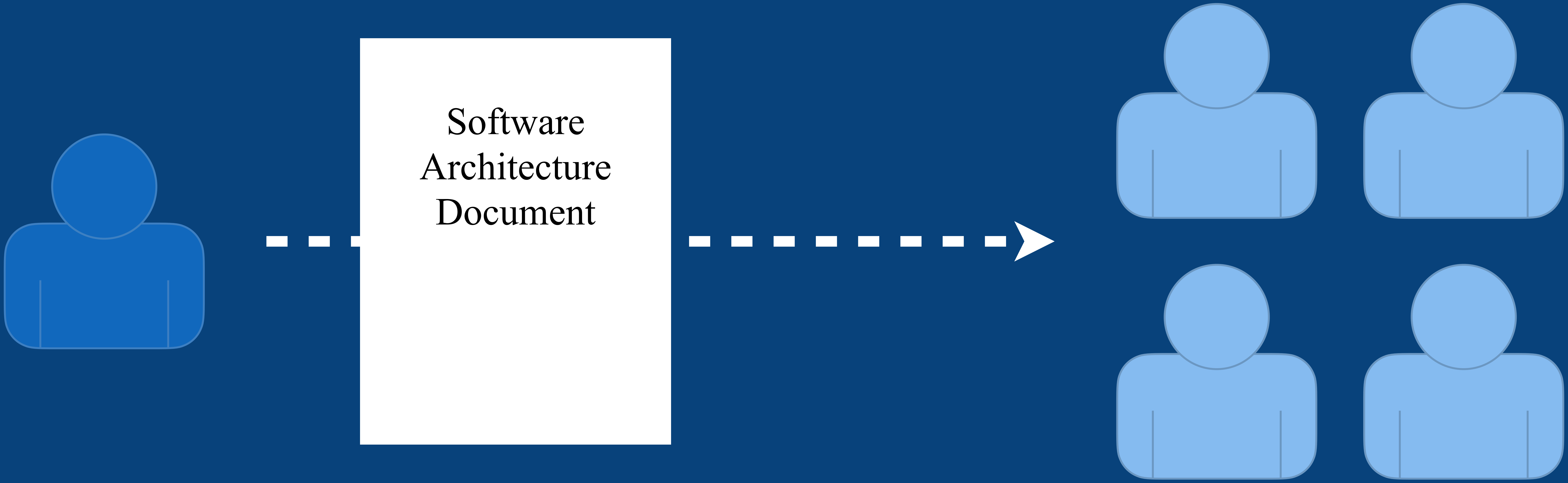
technical leadership

(irrespective of team size)

Every “software system” needs **technical leadership**

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

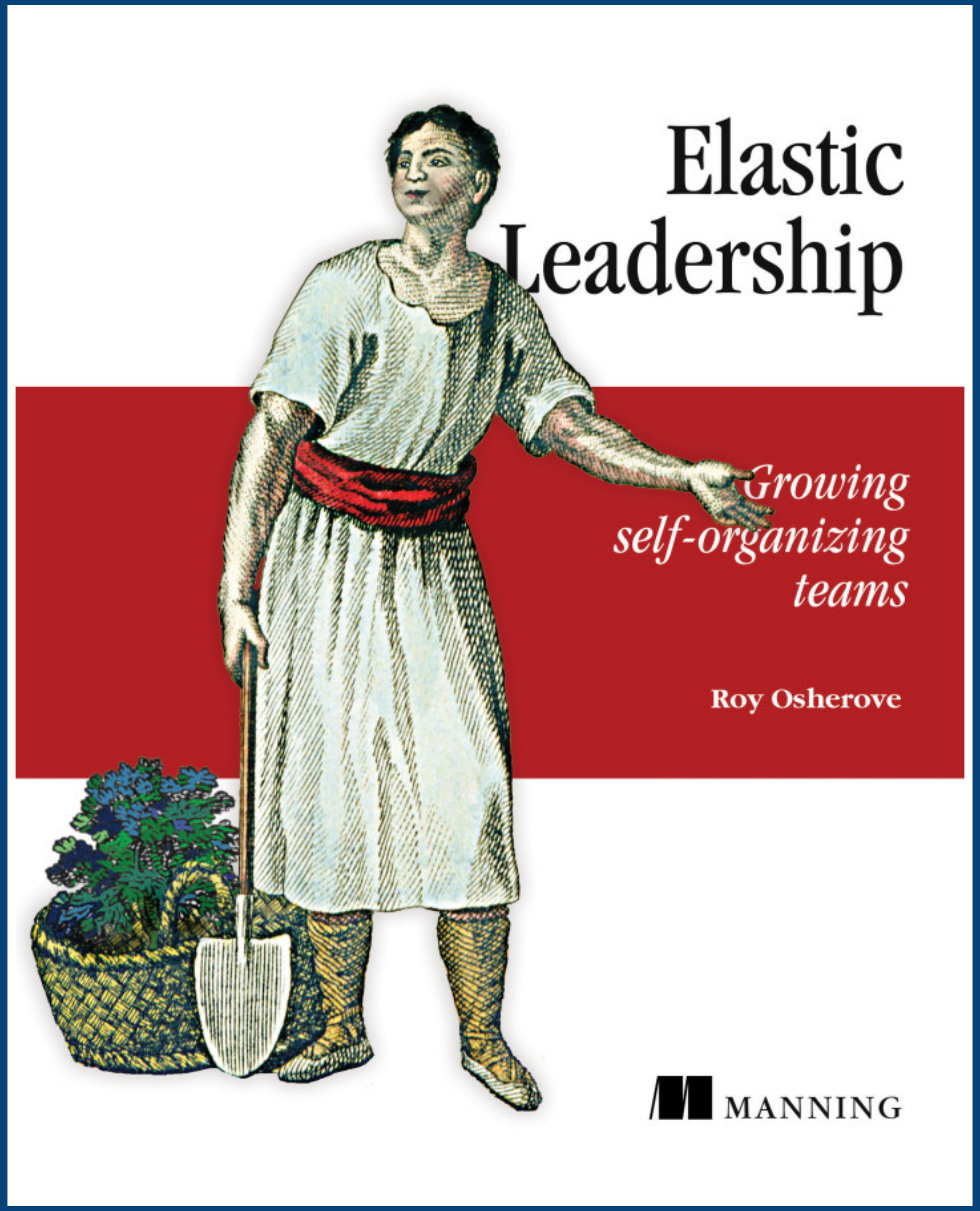
Software development is not a relay sport



AaaS

Architecture as a Service

Continuous technical leadership



Elastic Leadership

Growing self-organizing teams

Roy Osherove

MANNING

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)

Should software architects write **code**?

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

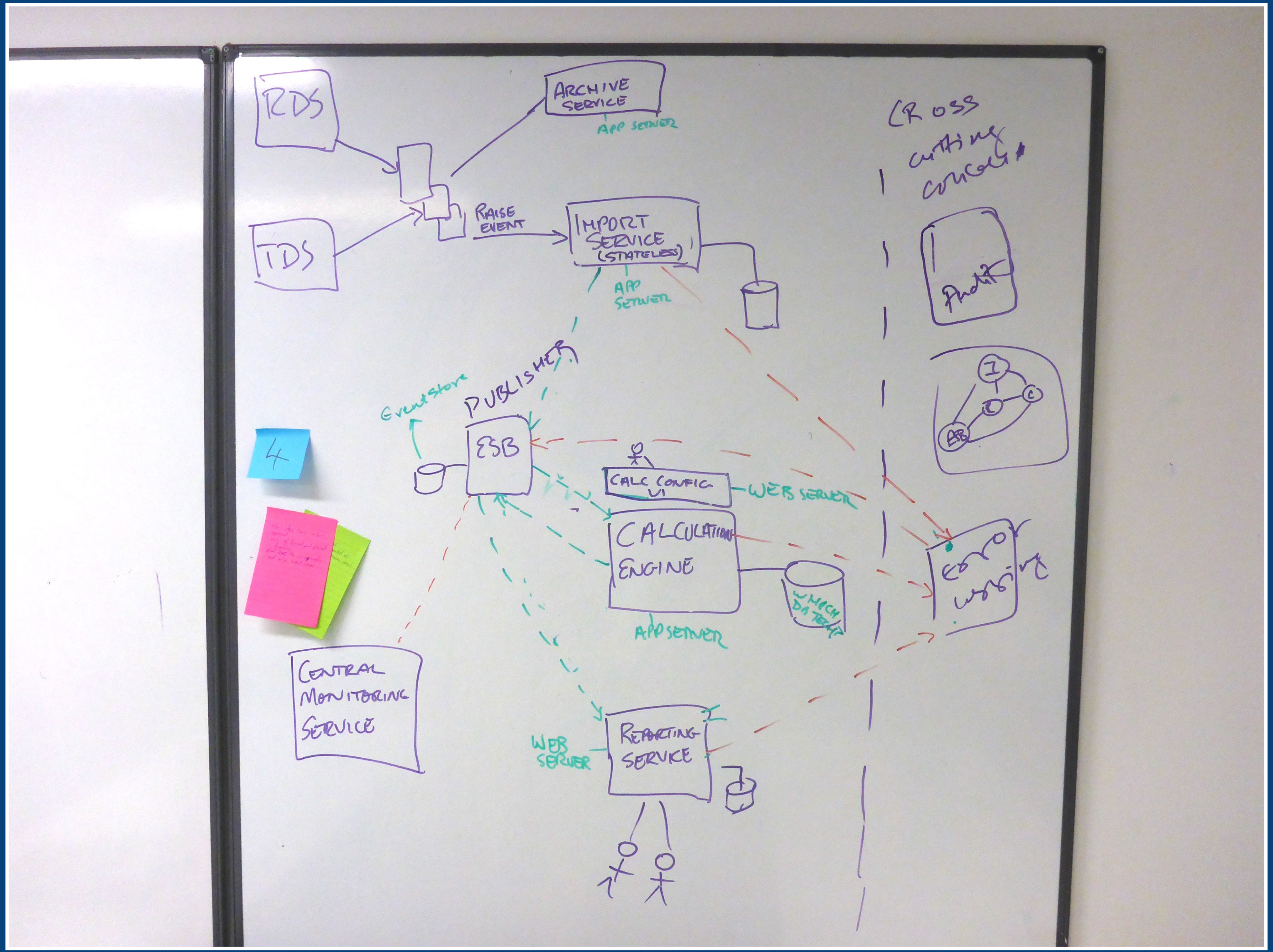
Good software architects

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**?

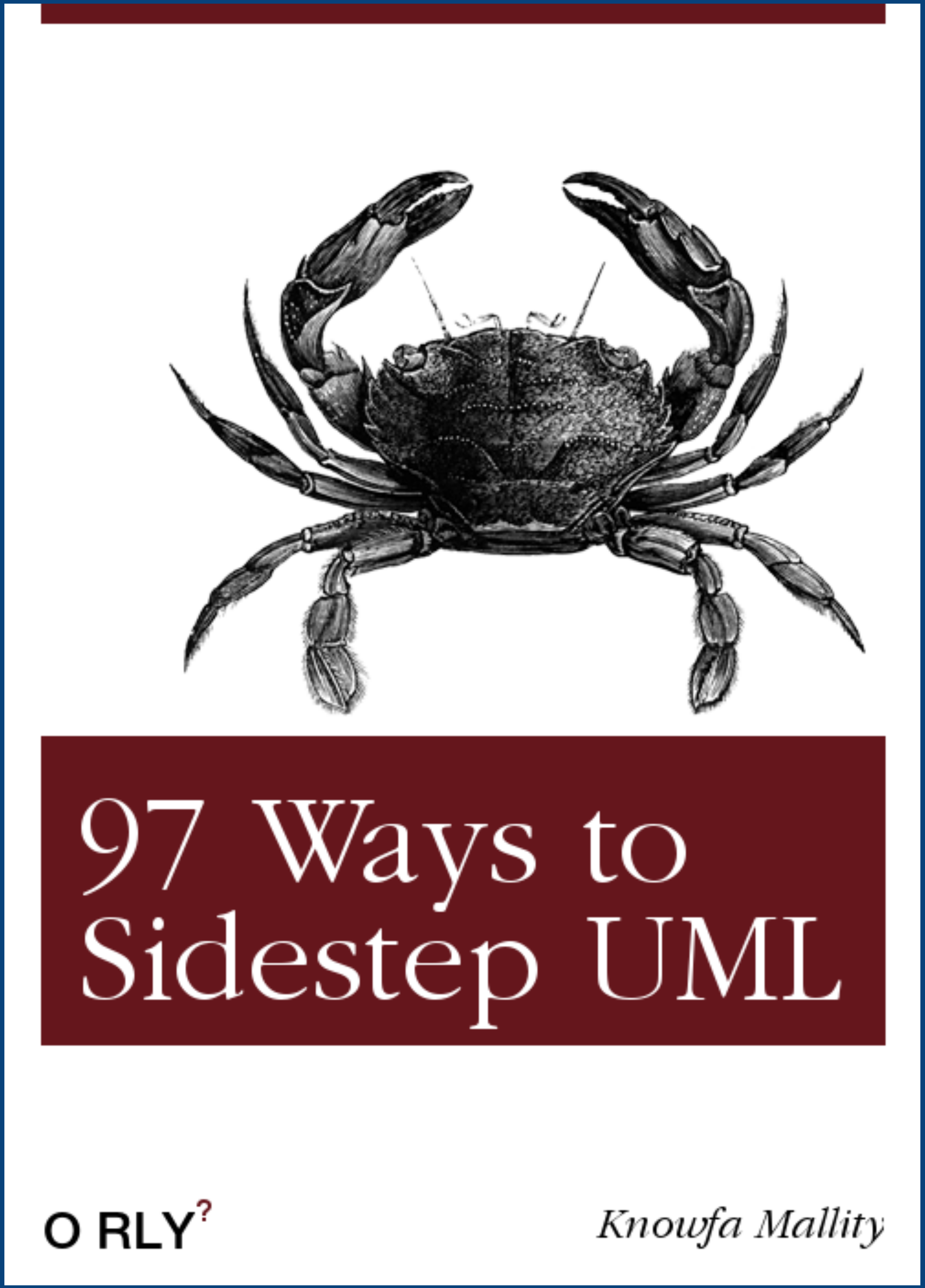
The software architecture role is **multi-faceted**

(technical depth, technical breadth, soft skills)

4. You don't need to use UML

In my experience, optimistically,

1 out of 10 people use UML



#2 “Not everybody else on the team knows it.”

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

#36 “You’ll be seen as old.”

#37 “You’ll be seen as old-fashioned.”

#66 “The tooling sucks.”

#80 “It’s too detailed.”

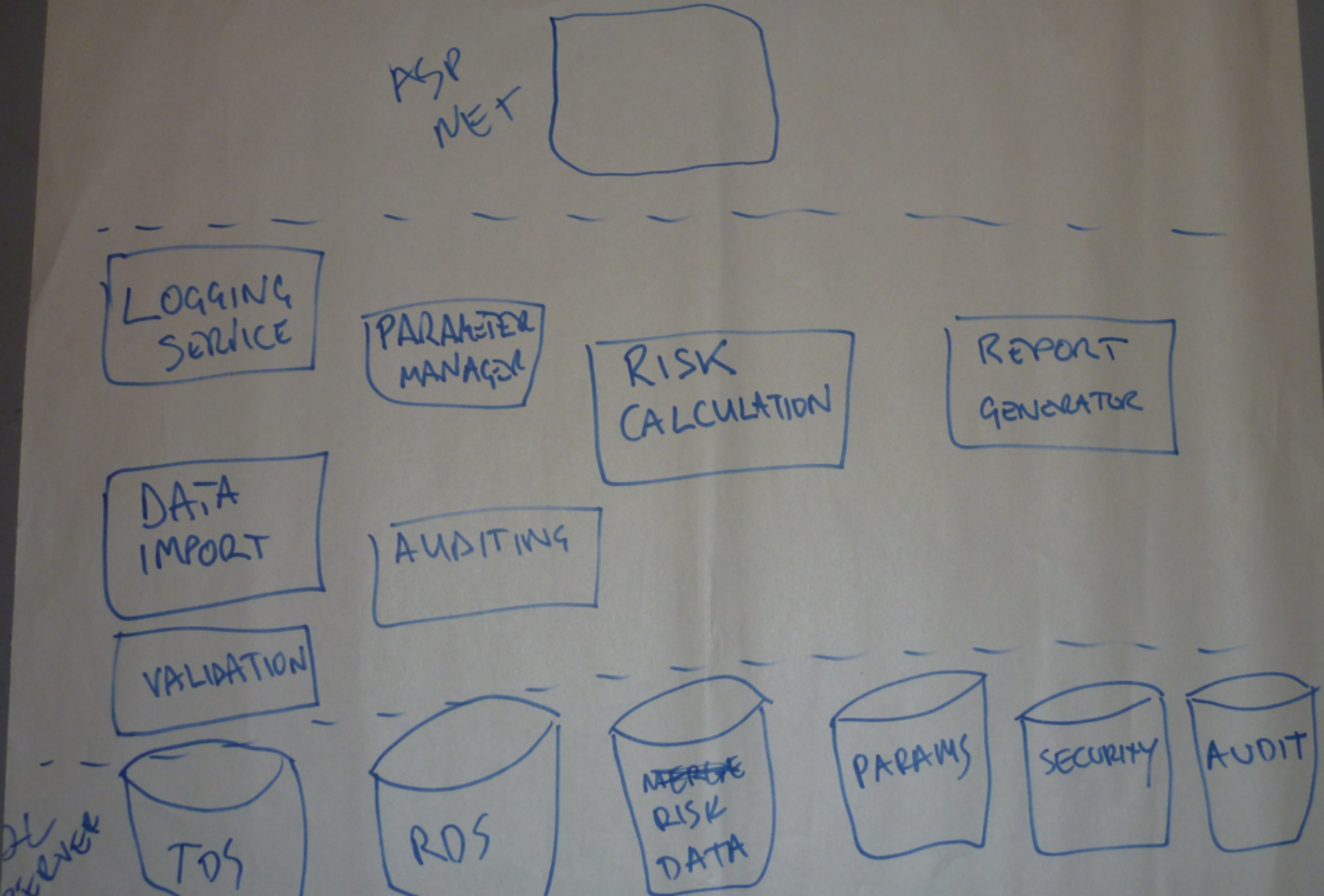
#81 “It’s a very elaborate waste of time.”

#92 “It’s not expected in agile.”

#97 “The value is in the conversation.”

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


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

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

TDS
RDS
trade report
ref report
R9
TRANSPORT
ARCHIVE
AUDIT
action
data
error
error
MS Sql Server
BUS. LOGIC
action
error
REPORT GEN
action
error
ERROR
scheduler
MONITOR
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
check if exists
SNMP service
DATA MERGE
C#
XPath
XML READER
C#
RDS
XML
IPS
RISK CALCULATION
C#
NOTIFICATION SERVICE
C#
SMTP
parameter
report to DB [illegible]
LOGGER
nLog
SNMP TRAP GENERATOR
DATA ACCESS
SQL SERVER

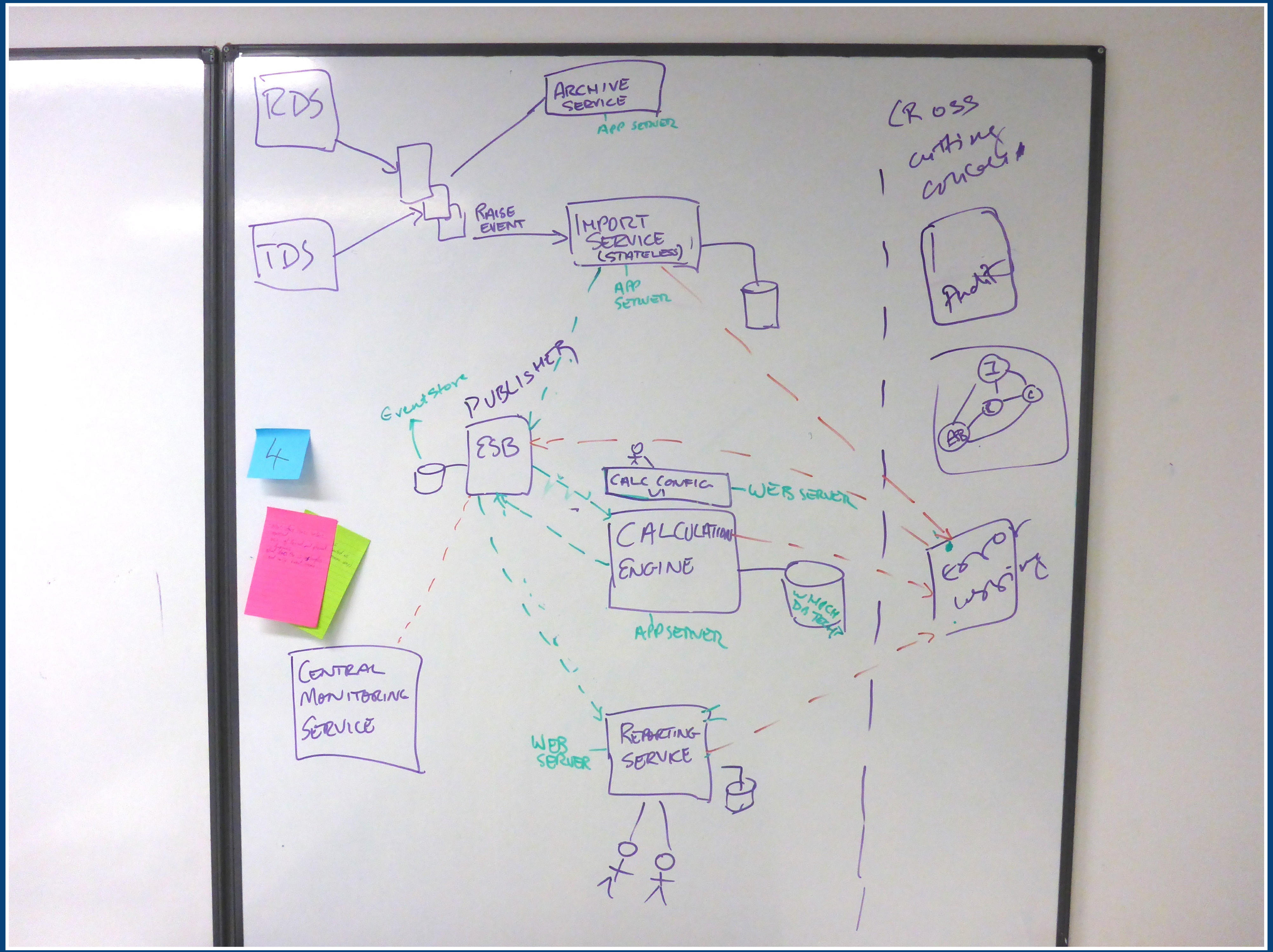
DATA POLL SERVICE
TRADES
AUTO RE-START.
REF DATA.
ARCHIVE
DATA READER TRADES
DATA VALIDATOR
reject
DATA READER REF DATA
DATA VALIDATOR.
reject.
FAIL EVENTS
MONITOR SERVICE SMTP TRAP
DATA STORE
DATA TRANSFORMATION INTEGRITY VALIDATION
rejects
tolerence.
FAIL EVENT
COMPLETE EVENT
STOP
STOPPED EVENT
COMPLETION EVENT
CALCULATION ENGINE
COUNT
A + B = C
PARMS TOLER.
UI
SECURITY
AUDIT
reject.
FAIL EVENT
COMPLETE EVENT
OUTPUT STORE
FAILURE

ENGINE
APPSERVER
WHICH DB TECH?
RING
WEB SERVER
REPORTING SERVICE

Params
Calcs
Risk outputs
EH?
Error
App Log
App Error Log
Date
- Risk sel
- calcs
- Par
Batch
Params_Distrib
Detail
B_id:
Outp id

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
RTP/ RTCP
Network Layer
TCP
UDP
IP
MAC
Ethernet
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
FXS
ADC
DAC
Audio Interface layer
Sound card Interface (PCI / USB)
FXS/TDM Interface
Interface layer - tion, CPT/DTMF, MF-R1/R2, odem Discrimination Unit
Session Initiation Protocol (SIP)
V²oIP
Video over IP
Voice over IP
Fax over IP
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
Remote administration
Control interface
Input devices
Input device manager
Object-Tracker
Real-time kernel
Outp ma
User manager
DB-Server
Analysis
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
Racemi Dynacenter
Rapid Server Imaging
Automation Infrastructure Platform
AOM Data Service
Policy, Performance, and Resource Management
Authentication / Security (CA EEM)
State and Event Management
Scheduler (Windows)
Discovery (CA Network Discovery Gateway)
Management DB
Performance DB
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
RT & Event Manager
Data Processing
Report Gener.
Alarm
Log
Archive
Ref. DB
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
Catalog Service
Management Service
Management Service
Computational Analysis Service
e-AIRSview
CFD Solver
Output Converter
APSE
(Adaptive Problem Solving Environment)
Layer
Logic Sublayer
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
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
A/V Conferen
Chat
Mobile Service Component

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



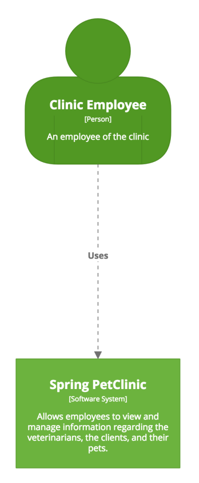
2. Is it going to **work**?

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

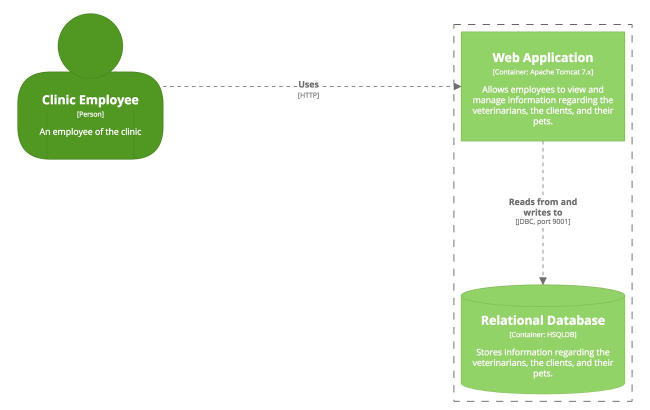
C4

Context, Containers, Components, and Code

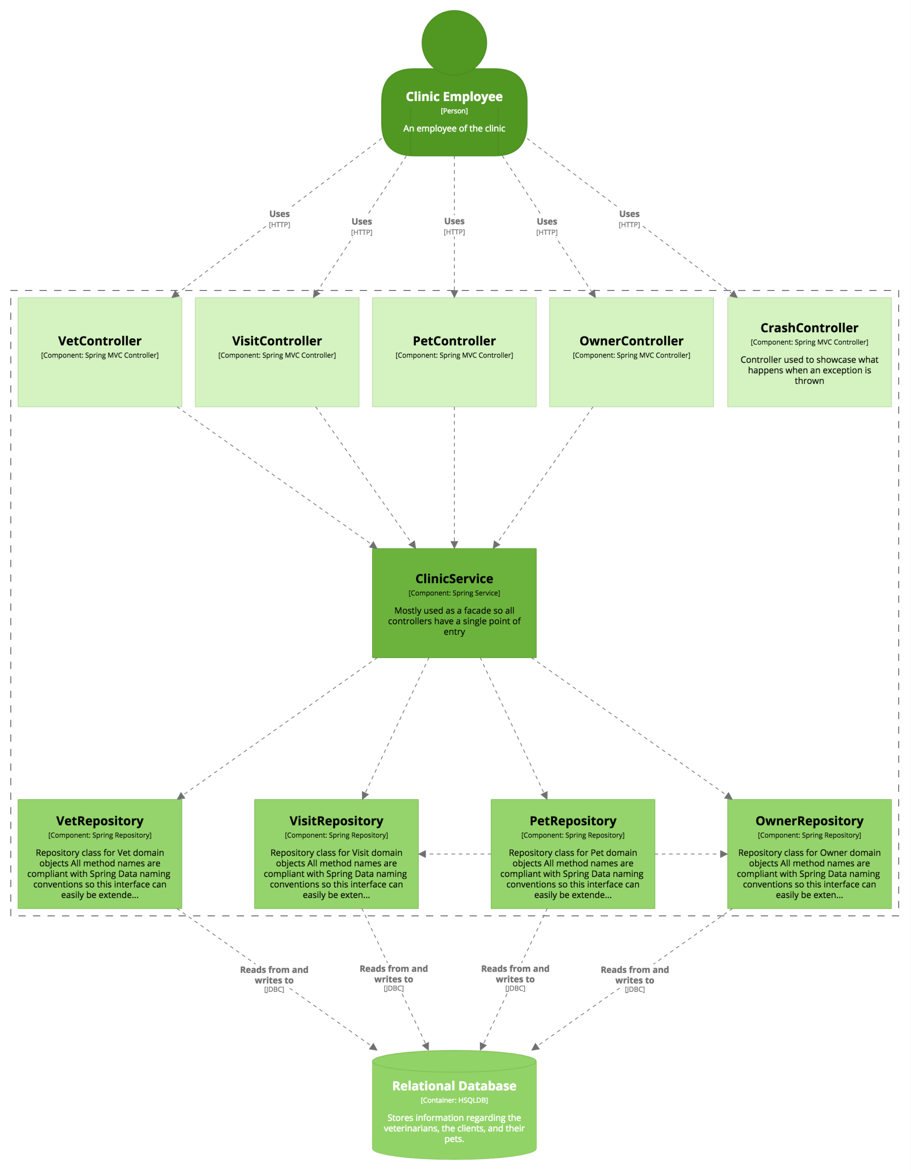
c4model.com


Clinic Employee
[Person]
An employee of the clinic
Uses
Spring PetClinic
[Software System]
Allows employees to view and manage information regarding the veterinarians, the clients, and their pets.

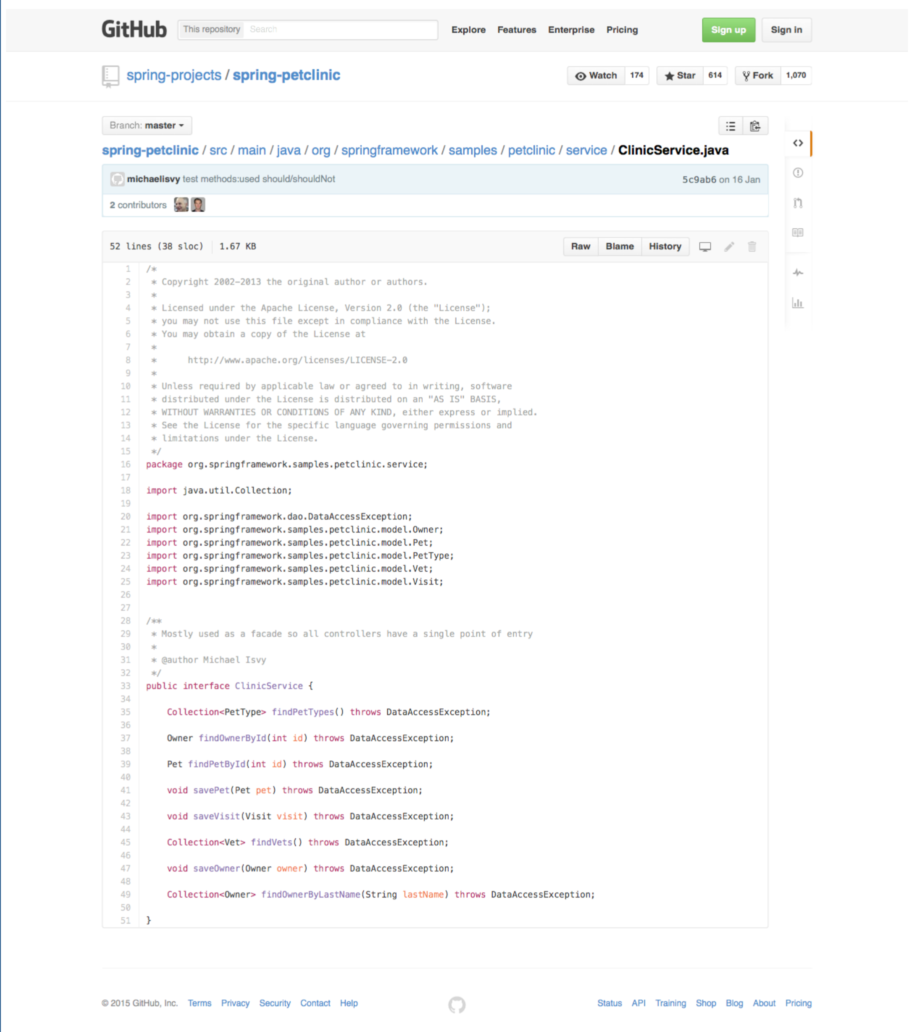
System Context diagram for Spring PetClinic
The System Context diagram for the Spring PetClinic system.
Friday 18 November 2016 22:21 UTC


Clinic Employee
[Person]
An employee of the clinic
Uses
[HTTP]
Web Application
[Container: Apache Tomcat 7.x]
Allows employees to view and manage information regarding the veterinarians, the clients, and their pets.
Reads from and writes to
[JDBC, port 9001]
Relational Database
[Container: HSQLDB]
Stores information regarding the veterinarians, the clients, and their pets.

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


Clinic Employee
[Person]
An employee of the clinic
Uses
[HTTP]
Uses
[HTTP]
Uses
[HTTP]
Uses
[HTTP]
Uses
[HTTP]
VetController
[Component: Spring MVC Controller]
VisitController
[Component: Spring MVC Controller]
PetController
[Component: Spring MVC Controller]
OwnerController
[Component: Spring MVC Controller]
CrashController
[Component: Spring MVC Controller]
Controller used to showcase what happens when an exception is thrown
ClinicService
[Component: Spring Service]
Mostly used as a facade so all controllers have a single point of entry
VetRepository
[Component: Spring Repository]
Repository class for Vet domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be extende...
VisitRepository
[Component: Spring Repository]
Repository class for Visit domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be exten...
PetRepository
[Component: Spring Repository]
Repository class for Pet domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be extende...
OwnerRepository
[Component: Spring Repository]
Repository class for Owner domain objects All method names are compliant with Spring Data naming conventions so this interface can easily be exten...
Reads from and writes to
[JDBC]
Reads from and writes to
[JDBC]
Reads from and writes to
[JDBC]
Reads from and writes to
[JDBC]
Relational Database
[Container: HSQLDB]
Stores information regarding the veterinarians, the clients, and their pets.

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



spring-petclinic / src / main / java / org / springframework / samples / petclinic / service / **ClinicService.java**

michaelisvy test methods:used should/shouldNot 5c9ab6 on 16 Jan

2 contributors

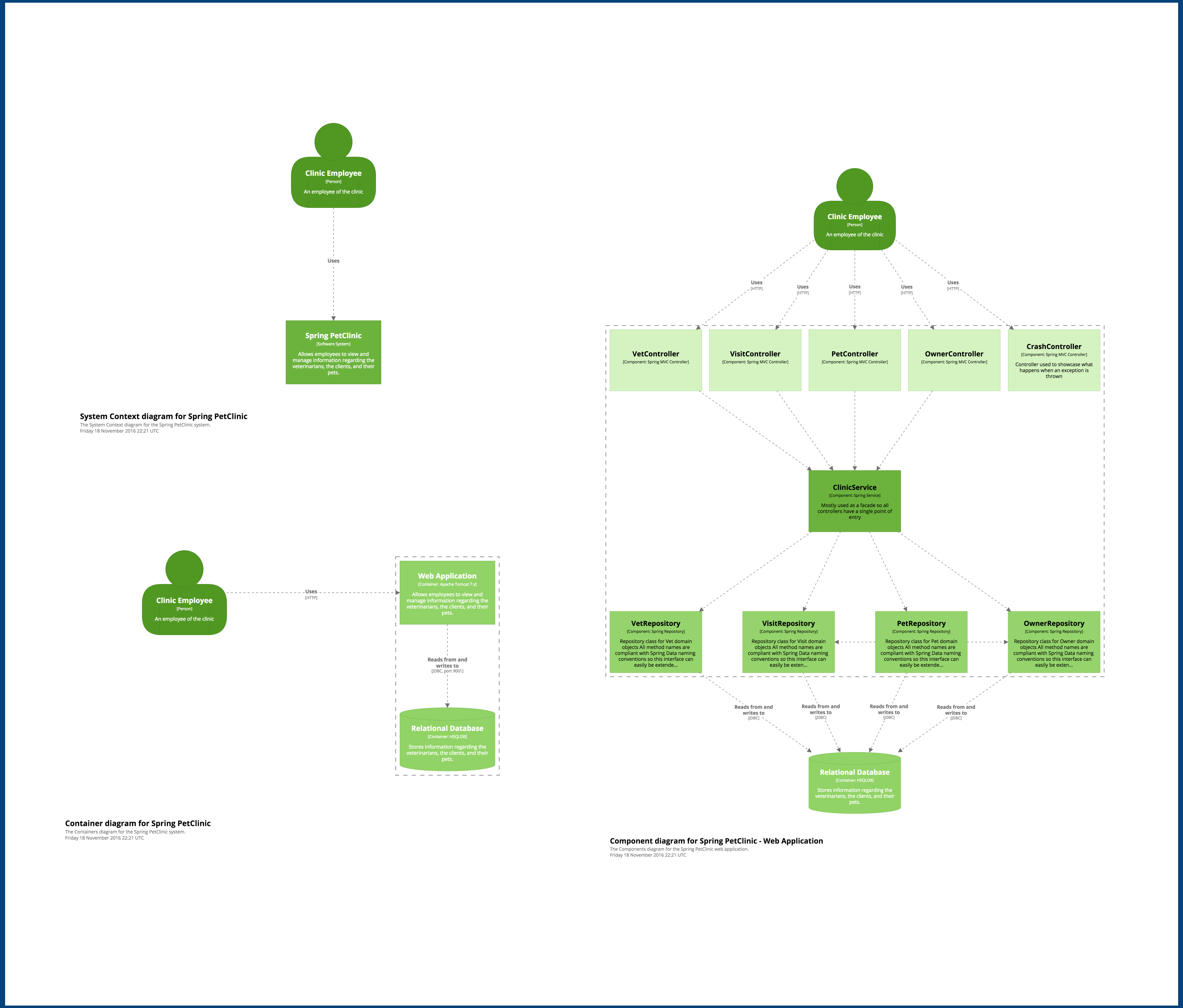
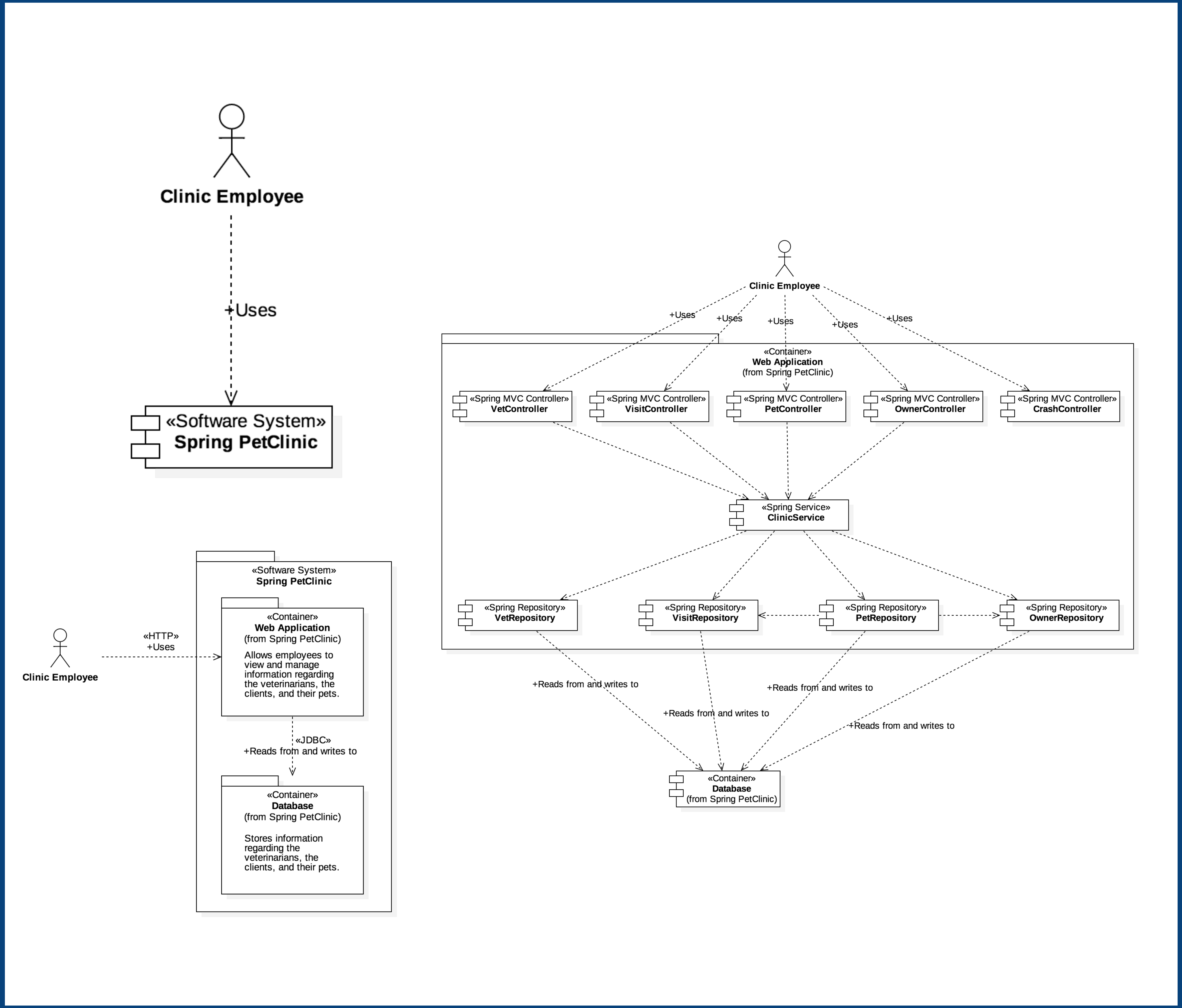
52 lines (38 sloc) 1.67 KB

```java
/*
 * Copyright 2002-2013 the original author or authors.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */
package org.springframework.samples.petclinic.service;

import java.util.Collection;

import org.springframework.dao.DataAccessException;
import org.springframework.samples.petclinic.model.Owner;
import org.springframework.samples.petclinic.model.Pet;
import org.springframework.samples.petclinic.model.PetType;
import org.springframework.samples.petclinic.model.Vet;
import org.springframework.samples.petclinic.model.Visit;


/**
 * Mostly used as a facade so all controllers have a single point of entry
 *
 * @author Michael Isvy
 */
public interface ClinicService {

    Collection<PetType> findPetTypes() throws DataAccessException;

    Owner findOwnerById(int id) throws DataAccessException;

    Pet findPetById(int id) throws DataAccessException;

    void savePet(Pet pet) throws DataAccessException;

    void saveVisit(Visit visit) throws DataAccessException;

    Collection<Vet> findVets() throws DataAccessException;

    void saveOwner(Owner owner) throws DataAccessException;

    Collection<Owner> findOwnerByLastName(String lastName) throws DataAccessException;

}
```

© 2015 GitHub, Inc. Terms Privacy Security Contact Help Status API Training Shop Blog About Pricing

Diagrams are maps

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


Clinic Employee
[Person]
An employee of the clinic
Uses
Spring PetClinic
[Software System]
Allows employees to view and manage information regarding the veterinarians, the clients, and their pets.
System Context diagram for Spring PetClinic
The System Context diagram for the Spring PetClinic system.
Friday 18 November 2016 22:21 UTC
Clinic Employee
[Person]
An employee of the clinic
Uses
[HTTP]
Web Application
[Container: Apache Tomcat 7.x]
Allows employees to view and manage information regarding the veterinarians, the clients, and their pets.
Reads from and writes to
[JDBC, port 9001]
Relational Database
[Container: HSQLDB]
Stores information regarding the veterinarians, the clients, and their pets.
Container diagram for Spring PetClinic
The Containers diagram for the Spring PetClinic system.
Friday 18 November 2016 22:21 UTC
Clinic Employee
[Person]
An employee of the clinic
Uses
VetController
VisitController
PetController
OwnerController
CrashController
ClinicService
VetRepository
VisitRepository
PetRepository
OwnerRepository
Reads from and writes to
Relational Database
Component diagram for Spring PetClinic - Web Application
Friday 18 November 2016 22:21 UTC


The C4 model does not prescribe any particular notation

(boxes and lines, UML, ArchiMate, etc are all options)

A **common set of abstractions**

is more important

than a common notation

c4model.com

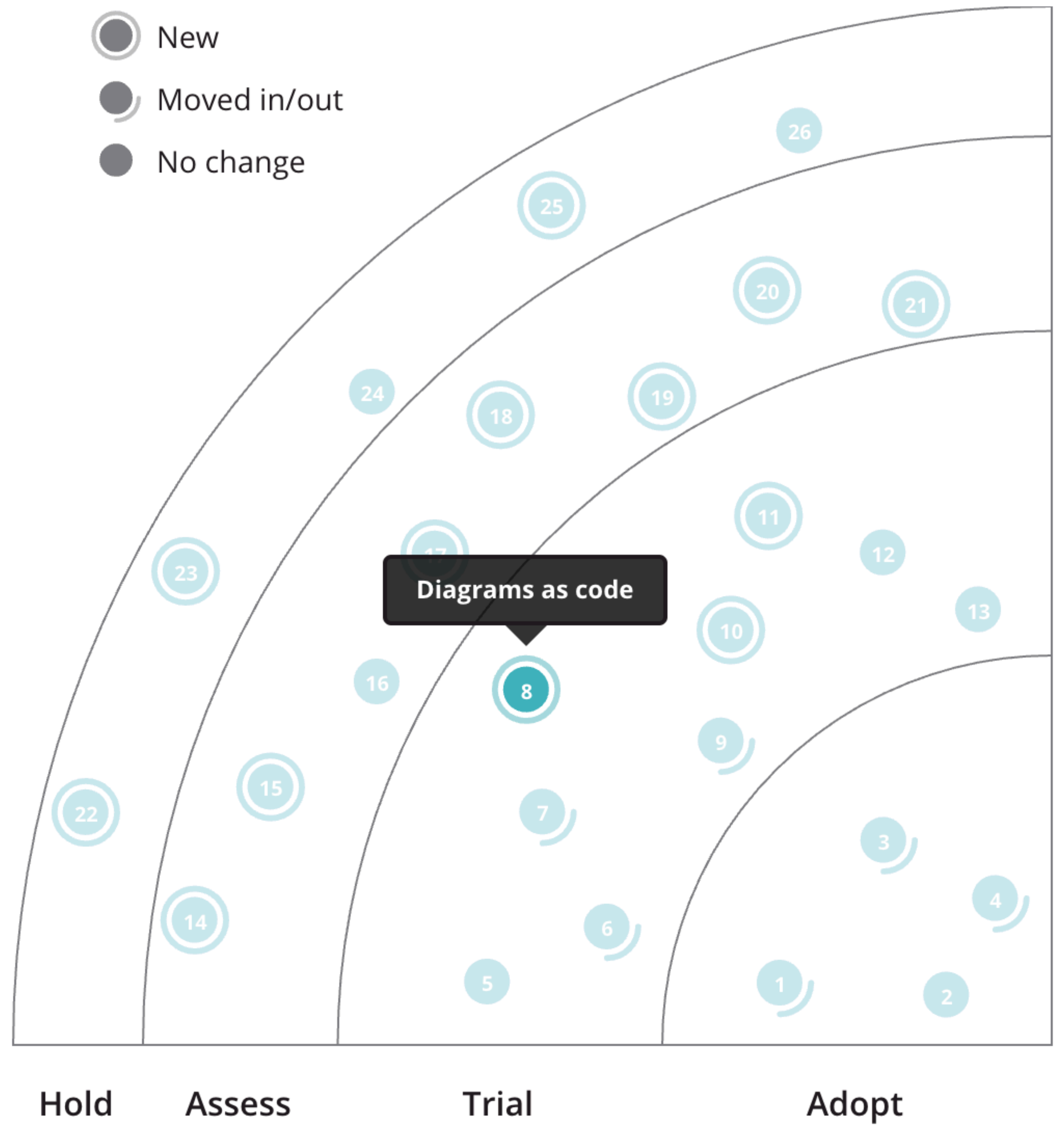
(for more information about software architecture diagrams)

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

Structurizr
Structurizr
PlantUML
Mermaid
WebSequenceDiagrams
Ilograph
[System Context] Software System (#SoftwareSystem-SystemContext)
User
[Person]
A user of my software system.
Uses
Software System
[Software System]
My software system.

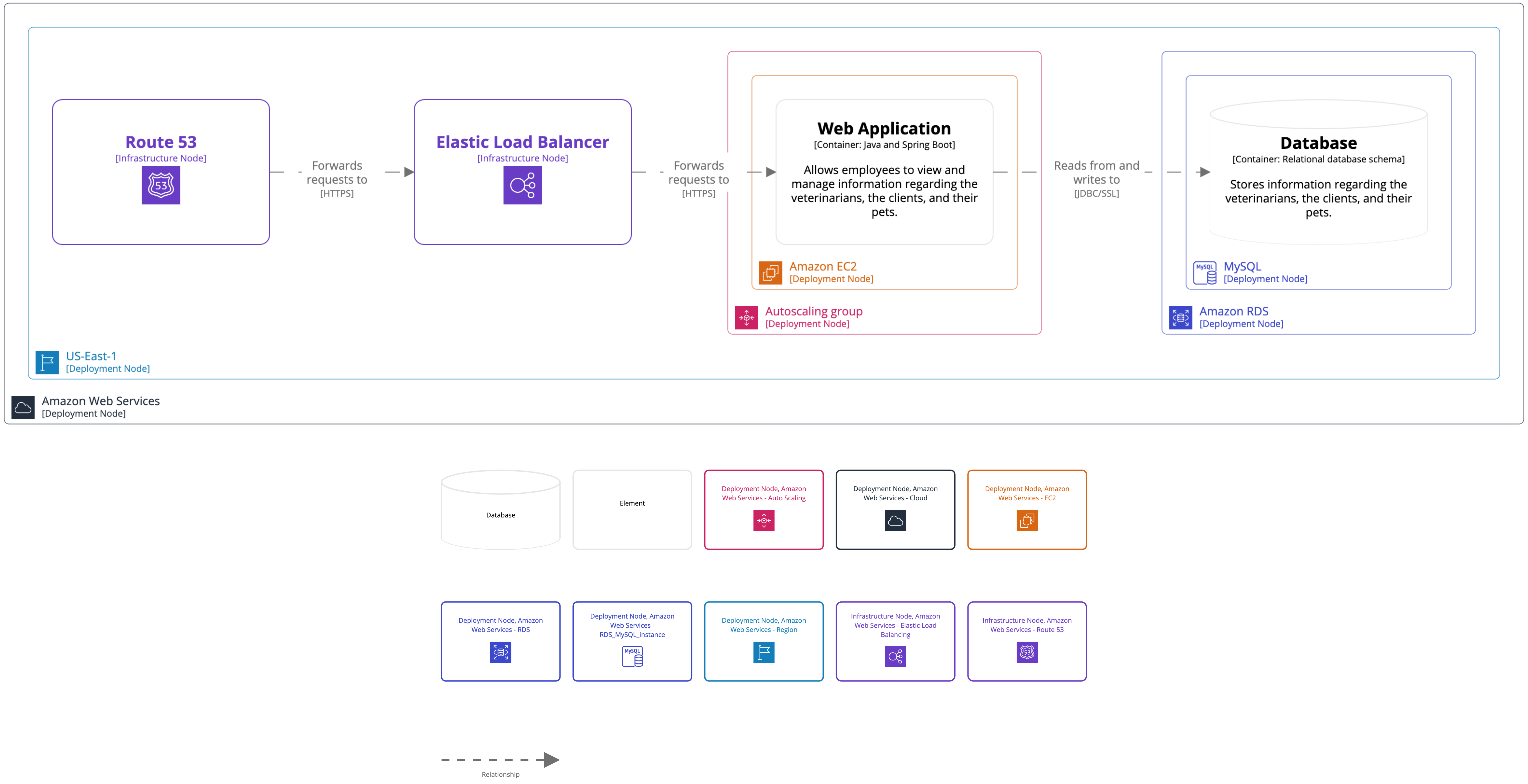
We recommend using the Structurizr CLI to create software architecture models with the DSL.

```
workspace {

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

        user -> softwareSystem "Uses"
    }

}
```

(diagram not editable; automatic layout enabled)

Language reference | Examples: Getting Started - Amazon Web Services - Big Bank plc

https://structurizr.com/dsl


Route 53
[Infrastructure Node]
Forwards requests to
[HTTPS]
Elastic Load Balancer
[Infrastructure Node]
Forwards requests to
[HTTPS]
Web Application
[Container: Java and Spring Boot]
Allows employees to view and manage information regarding the veterinarians, the clients, and their pets.
Amazon EC2
[Deployment Node]
Autoscaling group
[Deployment Node]
Reads from and writes to
[JDBC/SSL]
Database
[Container: Relational database schema]
Stores information regarding the veterinarians, the clients, and their pets.
MySQL
[Deployment Node]
Amazon RDS
[Deployment Node]
US-East-1
[Deployment Node]
Amazon Web Services
[Deployment Node]
Database
Element
Deployment Node, Amazon Web Services - Auto Scaling
Deployment Node, Amazon Web Services - Cloud
Deployment Node, Amazon Web Services - EC2
Deployment Node, Amazon Web Services - RDS
Deployment Node, Amazon Web Services - RDS_MySQL_instance
Deployment Node, Amazon Web Services - Region
Infrastructure Node, Amazon Web Services - Elastic Load Balancing
Infrastructure Node, Amazon Web Services - Route 53
Relationship

https://structurizr.com/dsl

Internet Banking System - Containers
E-mail System
[Software System]
The internal Microsoft Exchange e-mail system
Sends e-mails to
Internet Banking System
[Software System]
Personal Banking Customer
[Person]
A customer of the bank, with personal bank accounts.
Visits bigbank.com/ib using
[HTTPS]
Web Application
[Container: Java and Spring MVC]
Delivers the static content and the Internet banking single page application.
Views account balances, and makes payments using
Delivers to the customer's web browser
Sends e-mail using
Mobile App
[Container: Xamarin]
Provides a limited subset of the Internet banking functionality to customers via their mobile device.
Single-Page Application
[Container: JavaScript and Angular]
Provides all of the Internet banking functionality to customers via their web browser.
Makes API calls to
[JSON/HTTPS]
API Application
[Container: Java and Spring MVC]
Provides Internet banking functionality via a JSON/HTTPS API.
Reads from and writes to
[JDBC]
Makes API calls to
[XML/HTTPS]
Database
[Container: Oracle Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.
Mainframe Banking System
[Software System]
Stores all of the core banking information about customers, accounts, transactions, etc.
Sends e-mail using
[SMTP]
Sign In Controller
Security Component
Reset Password Controller
E-mail Component
Accounts Summary Controller
Mainframe Banking System Facade
Uses
Container diagram for Internet Banking System
The container diagram for the Internet Banking System.
Tuesday, 24 March 2020, 18:29 Greenwich Mean Time

Structurizr DSL + CLI … single model definition, multiple diagrams in multiple output formats

5. A good software architecture enables agility

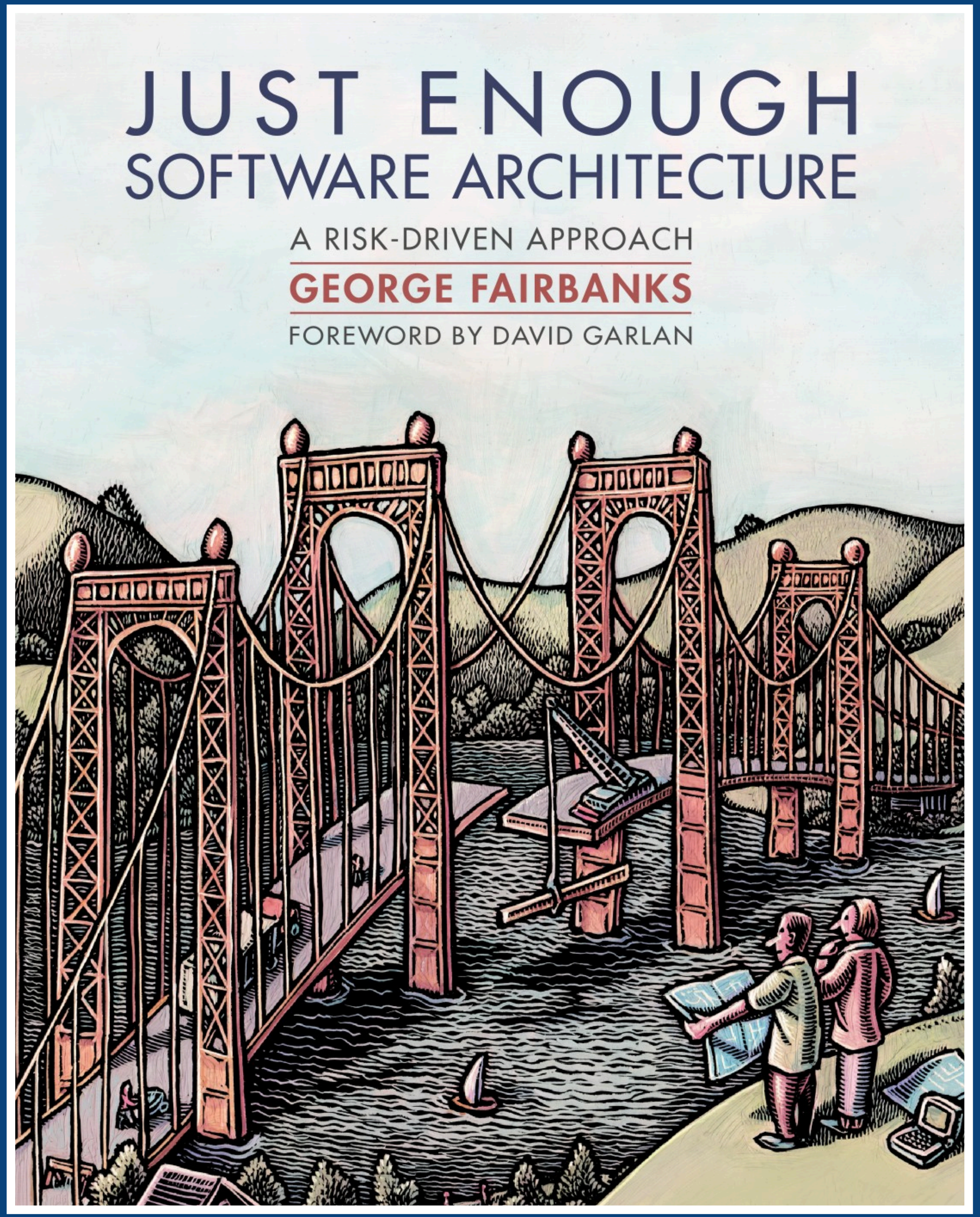
Agile is about moving fast, embracing change, releasing often, getting feedback, …

Agile is about a mindset of
continuous improvement

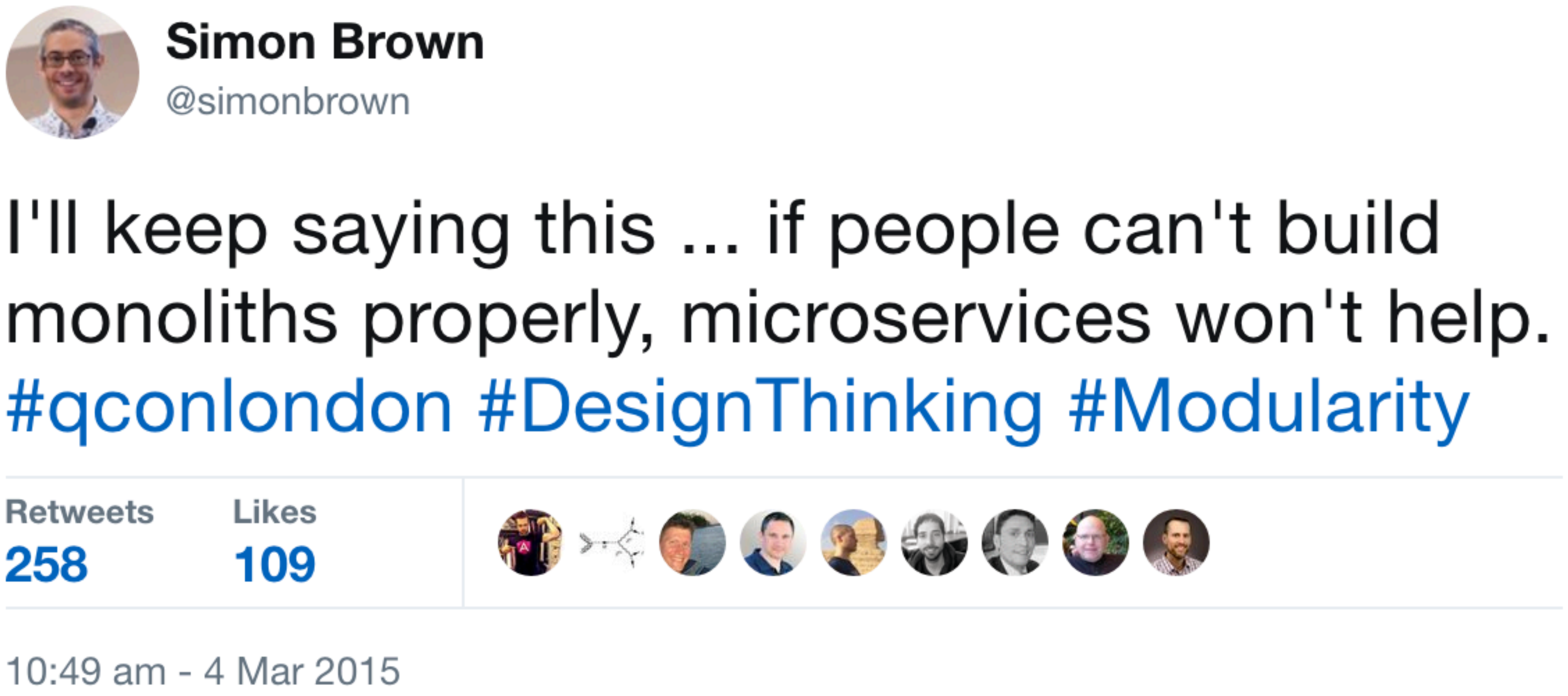
Agility is a

quality attribute

A good architecture enables agility



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

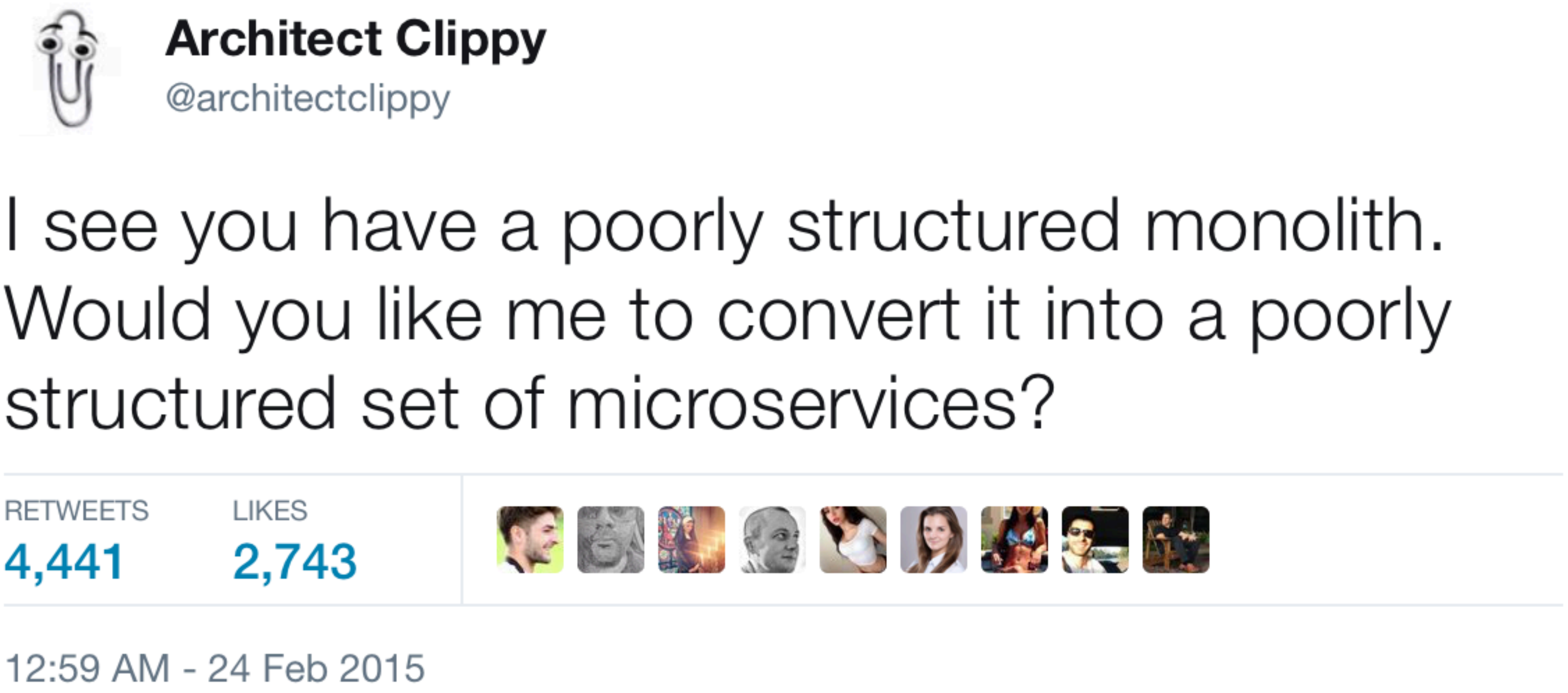


Simon Brown
@simonbrown

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

Retweets 258 | Likes 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
@simonbrown