coding {the} architecture

The Art of Visualising Software Architecture

Simon Brown
@simonbrown

“…the architecture diagrams don’t match the code”

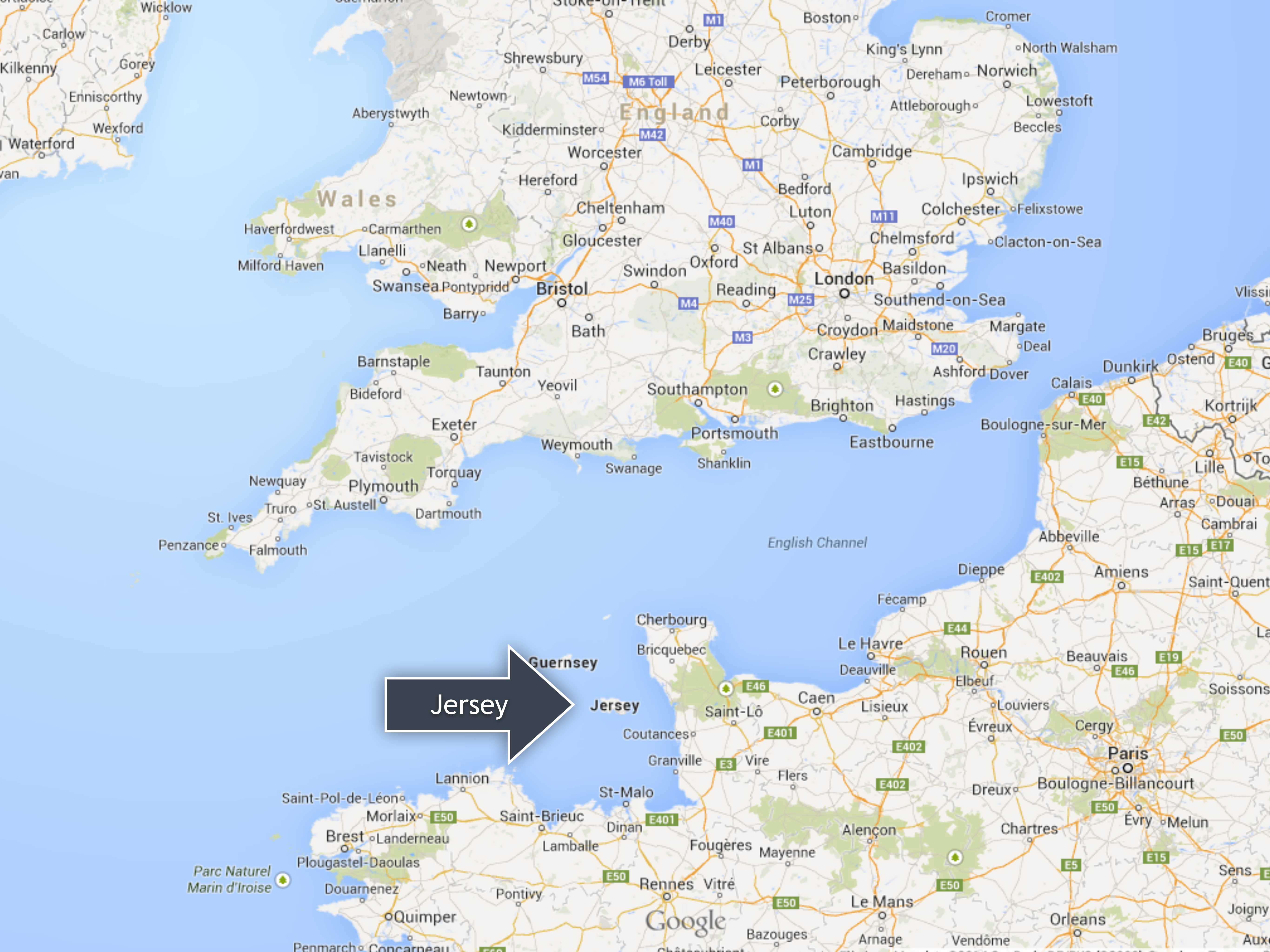
Kristijan | Криштиџн
@Krishtidzn

Follow

Any recommendations for software for drawing software architecture but not MS Visio?

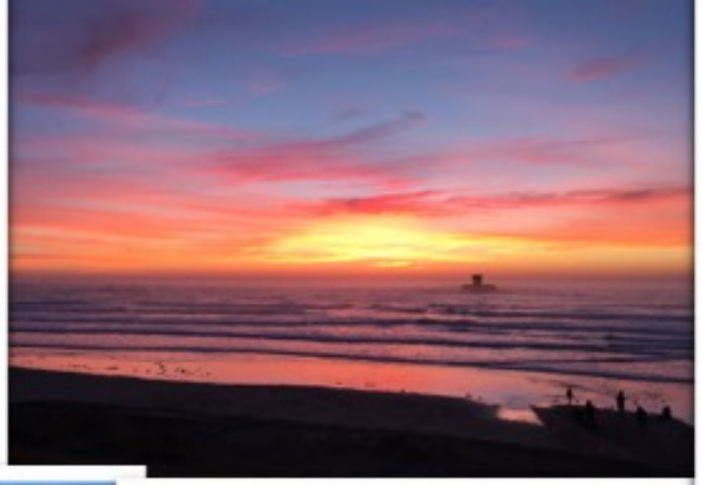
11:11 AM - 16 Apr 2015

Greenland
Iceland
Norwegian Sea
Sweden
Finland
Norway
Denmark
United Kingdom
Ireland
Poland
Germany
Belarus
Jersey
Austria
France
Romania
Italy
Spain
Portugal
Greece
Tunisia
Morocco
Algeria
Libya
Western
North Atlantic Ocean
NL
QC
NB
PE
NS
ME
NH
MA
NY
PA
WV
DE
VA
NC
SC
FL


Jersey
Wicklow
Carlow
Kilkenny
Gorey
Enniscorthy
Wexford
Waterford
Shrewsbury
Derby
Leicester
Boston
Cromer
King's Lynn
North Walsham
Norwich
Dereham
Peterborough
Attleborough
Lowestoft
Beccles
Newtown
Aberystwyth
Kidderminster
England
Corby
Worcester
Cambridge
Hereford
Ipswich
Wales
Bedford
Cheltenham
Luton
Colchester
Felixstowe
Haverfordwest
Carmarthen
Gloucester
Chelmsford
Clacton-on-Sea
Llanelli
St Albans
Milford Haven
Neath
Newport
Swindon
Oxford
Basildon
London
Swansea
Pontypridd
Bristol
Reading
Southend-on-Sea
Barry
Bath
Croydon
Maidstone
Margate
Deal
Crawley
Barnstaple
Taunton
Ashford
Dover
Dunkirk
Ostend
Bruges
Calais
Yeovil
Southampton
Bideford
Brighton
Hastings
Kortrijk
Boulogne-sur-Mer
Exeter
Portsmouth
Eastbourne
Weymouth
Tavistock
Shanklin
Lille
Swanage
Torquay
Béthune
Newquay
Plymouth
Arras
Douai
Truro
St. Austell
Dartmouth
St. Ives
Cambrai
Penzance
Falmouth
English Channel
Abbeville
Dieppe
Amiens
Fécamp
Cherbourg
Le Havre
Bricquebec
Rouen
Beauvais
Guernsey
Deauville
Elbeuf
Soissons
Caen
Lisieux
Saint-Lô
Louviers
Évreux
Cergy
Coutances
Granville
Vire
Paris
Flers
Lannion
St-Malo
Dreux
Boulogne-Billancourt
Saint-Pol-de-Léon
Morlaix
Saint-Brieuc
Dinan
Évry
Melun
Brest
Landerneau
Lamballe
Fougères
Alençon
Chartres
Mayenne
Plougastel-Daoulas
Parc Naturel Marin d'Iroise
Douarnenez
Rennes
Vitré
Sens
Pontivy
Le Mans
Quimper
Joigny
Google
Orleans
Bazouges
Arnage
Vendôme
Penmarch
Concarneau

Guernsey

Jersey












I help software teams understand

software architecture, technical leadership and the balance with agility

Software architecture needs to be more accessible

coding {the} architecture

Software Architecture for Developers

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

Simon Brown

Leanpub

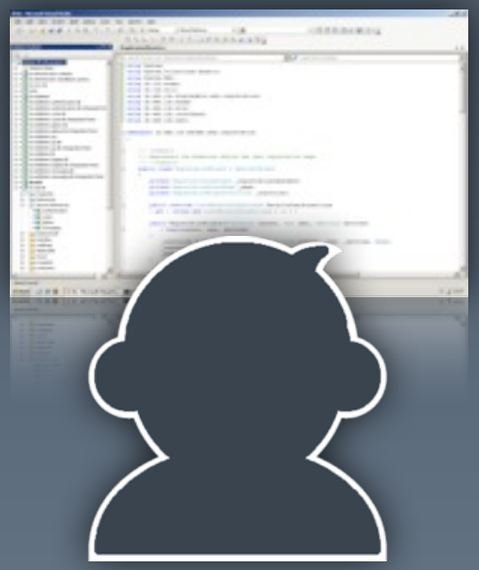
coding {the} architecture

The Art of Visualising Software Architecture

Communicating software architecture with *sketches*, diagrams and the C4 model

Simon Brown

Free!



I code too

⇧ ; - ⇧ 0

The problem

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 (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.

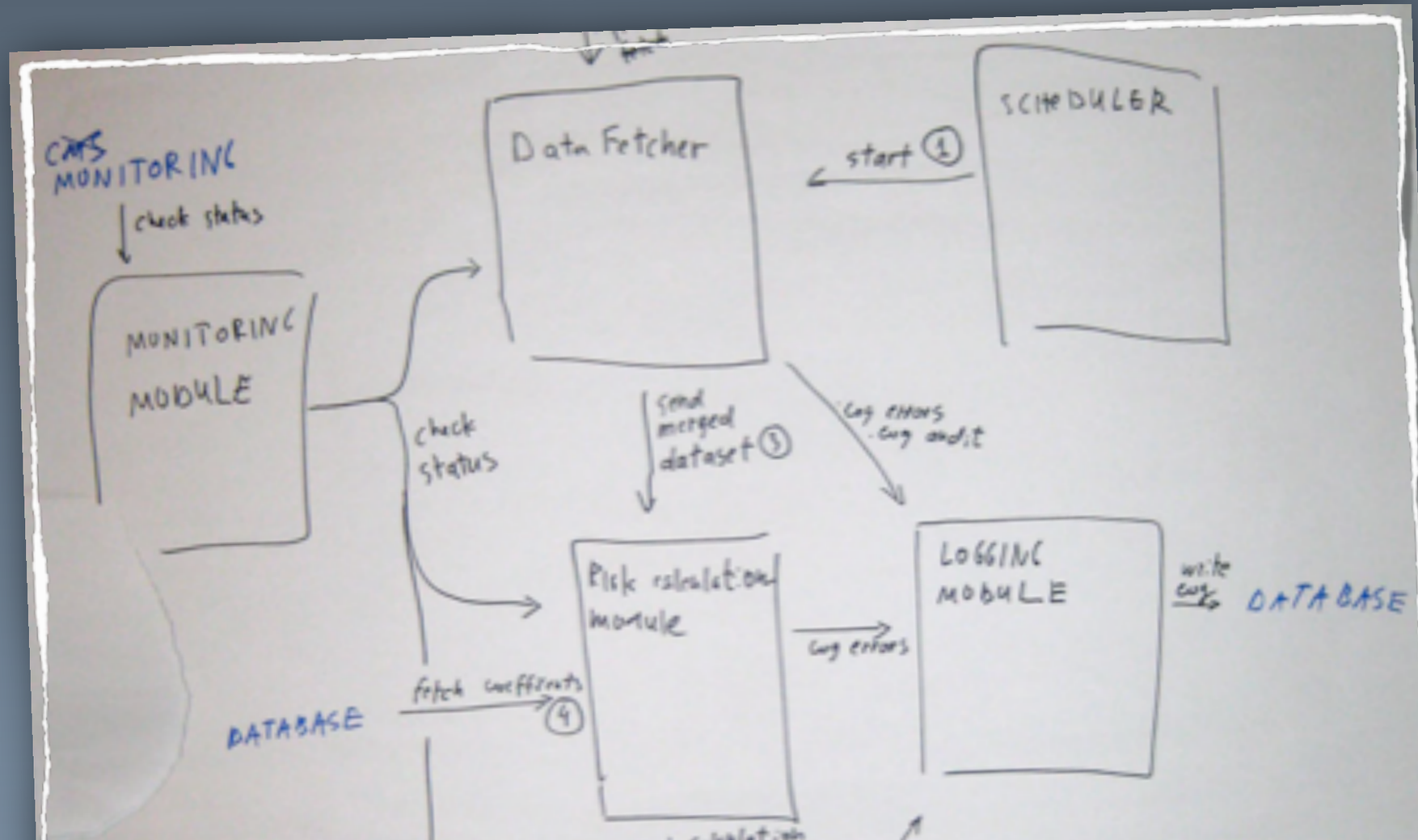
"Financial Risk System" architecture kata
Simon Brown | @simonbrown

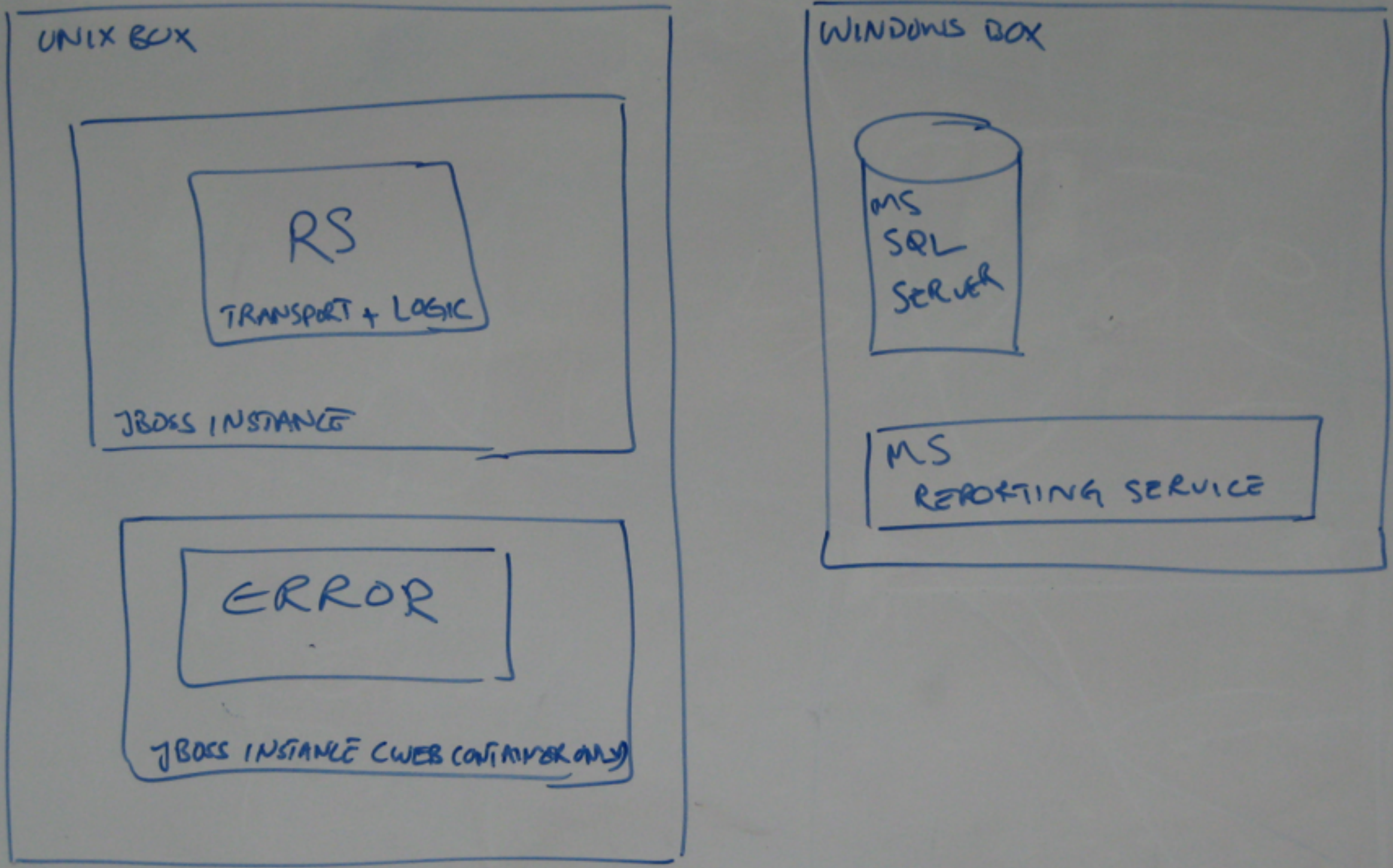
"Design a solution & draw some pictures to describe it"

Abstraction

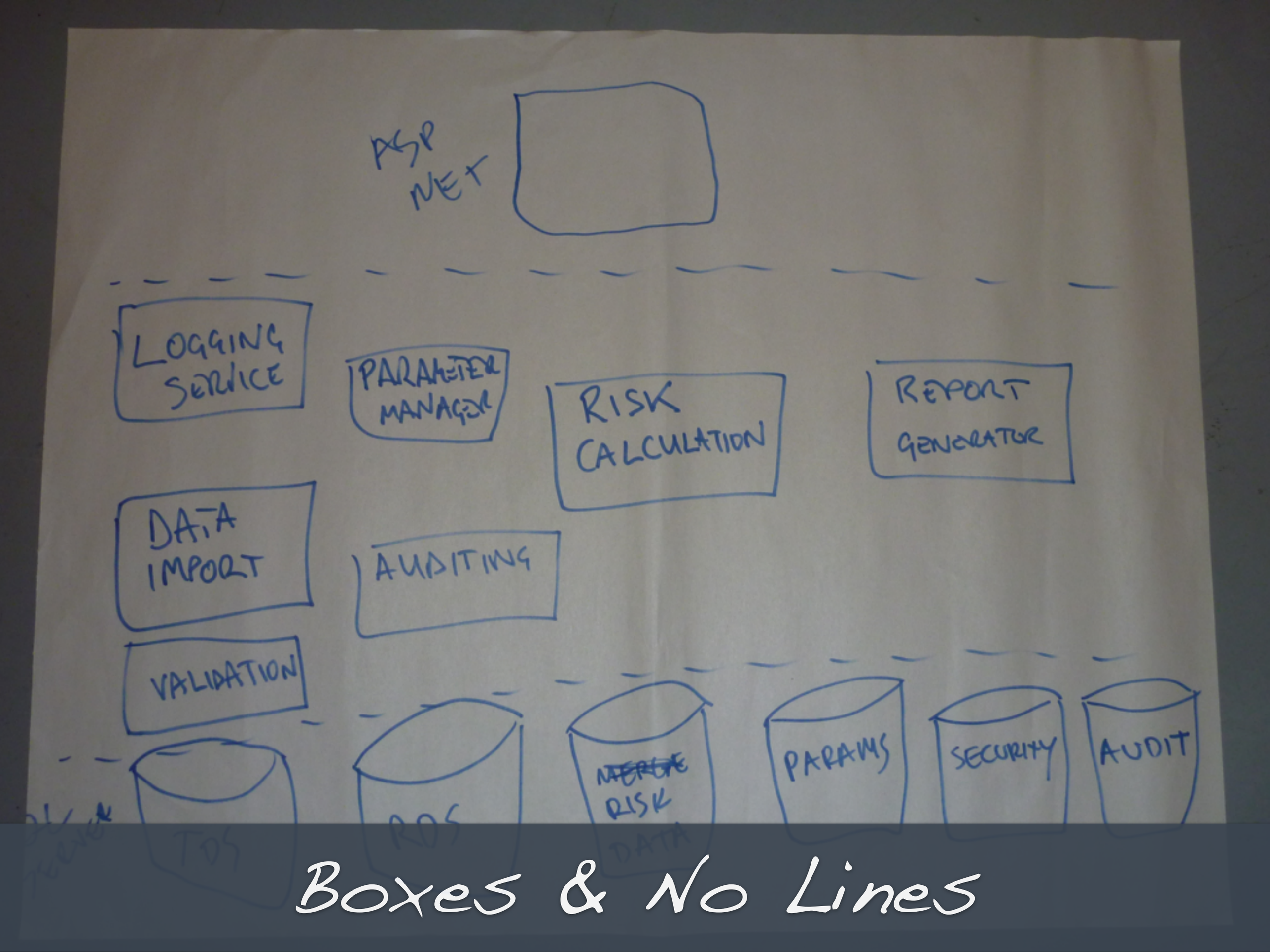
is about reducing detail

rather than creating a different representation





The Shopping List



Boxes & No Lines

FUNCTIONAL VIEW

File Retriever

Scheduler

Auditing

Reference Archiver

Risk Assesment Processor

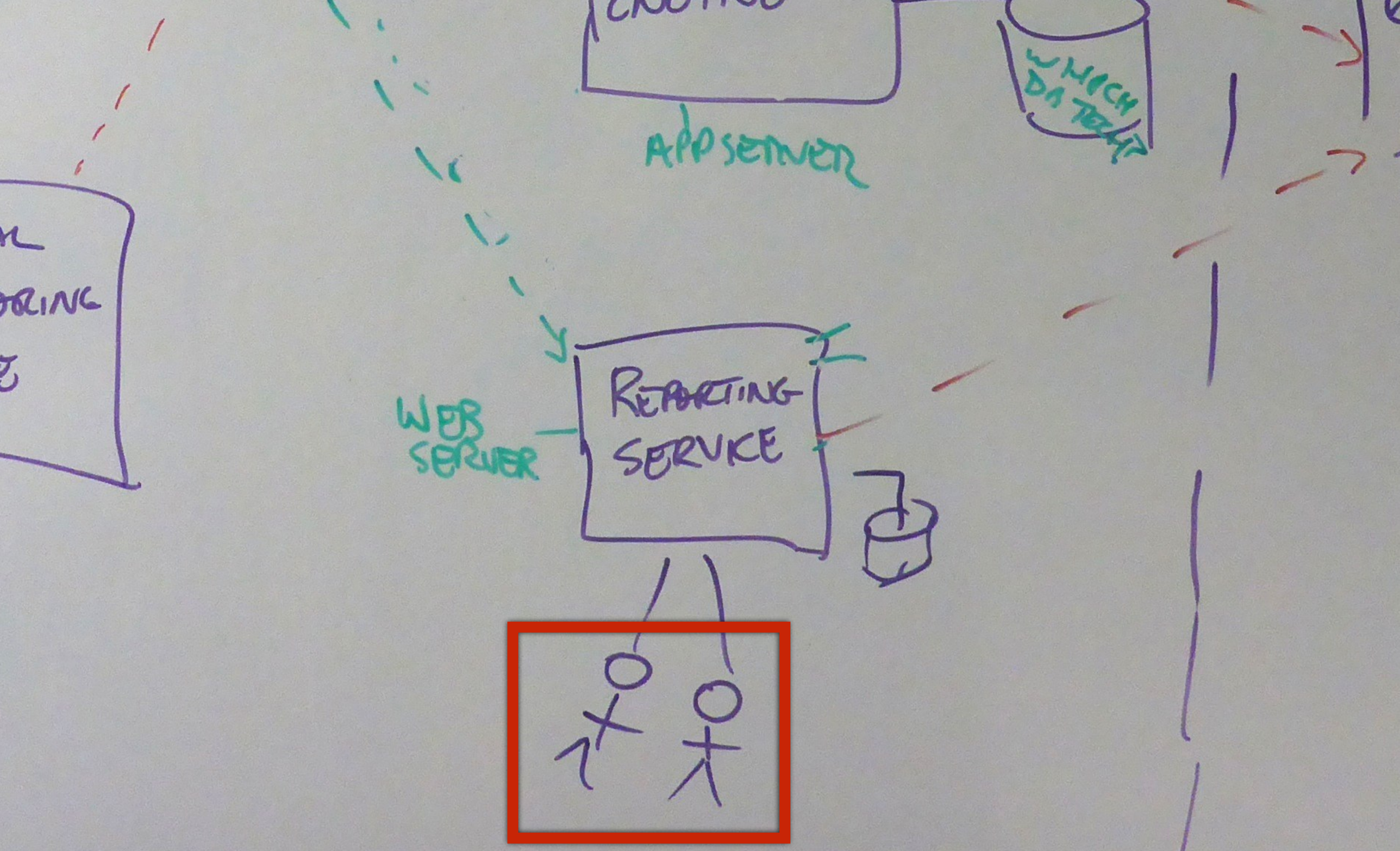
Risk Parameter Configuration

Trade Archiver

Report Generator

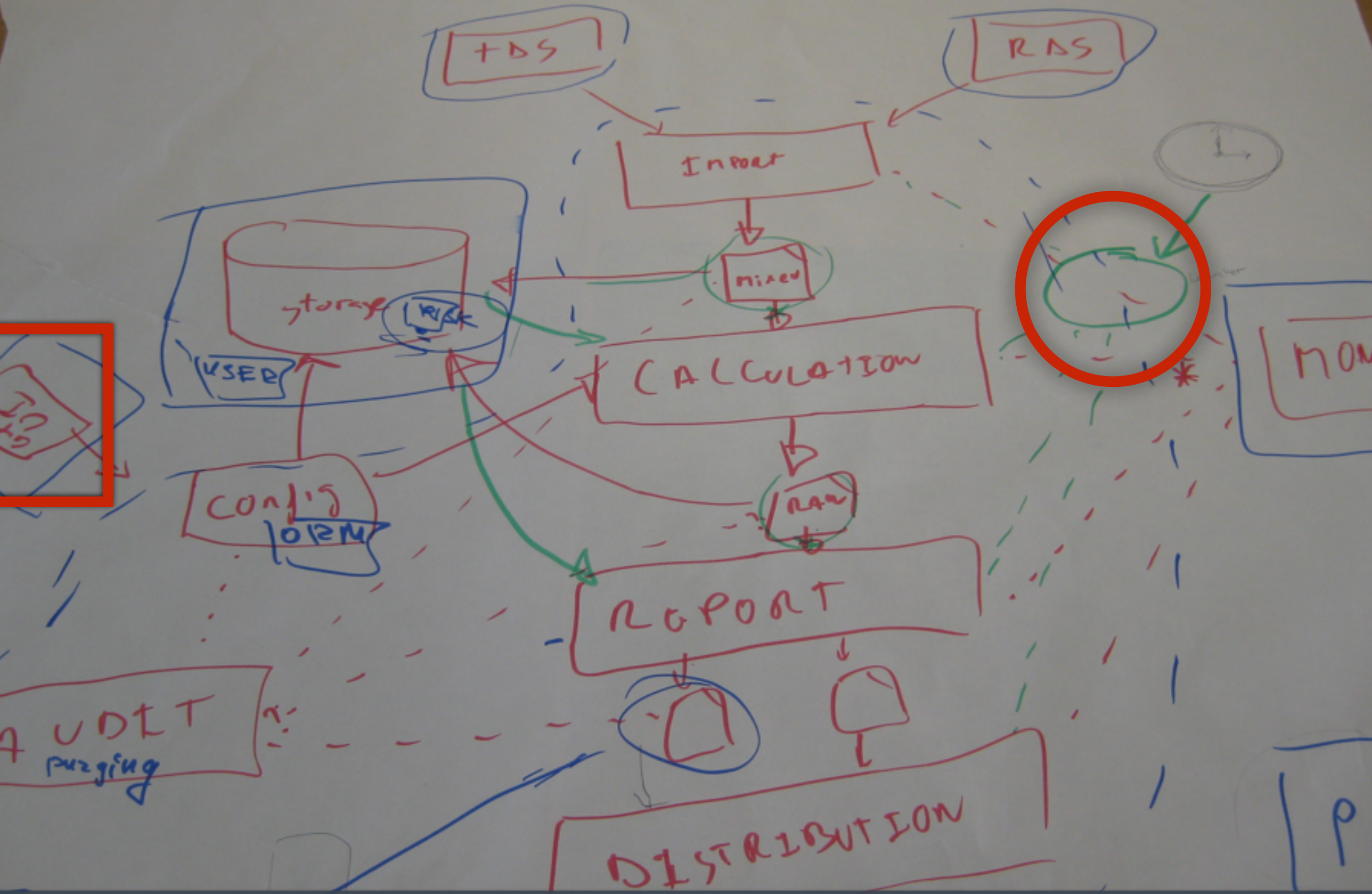
Report Distributor

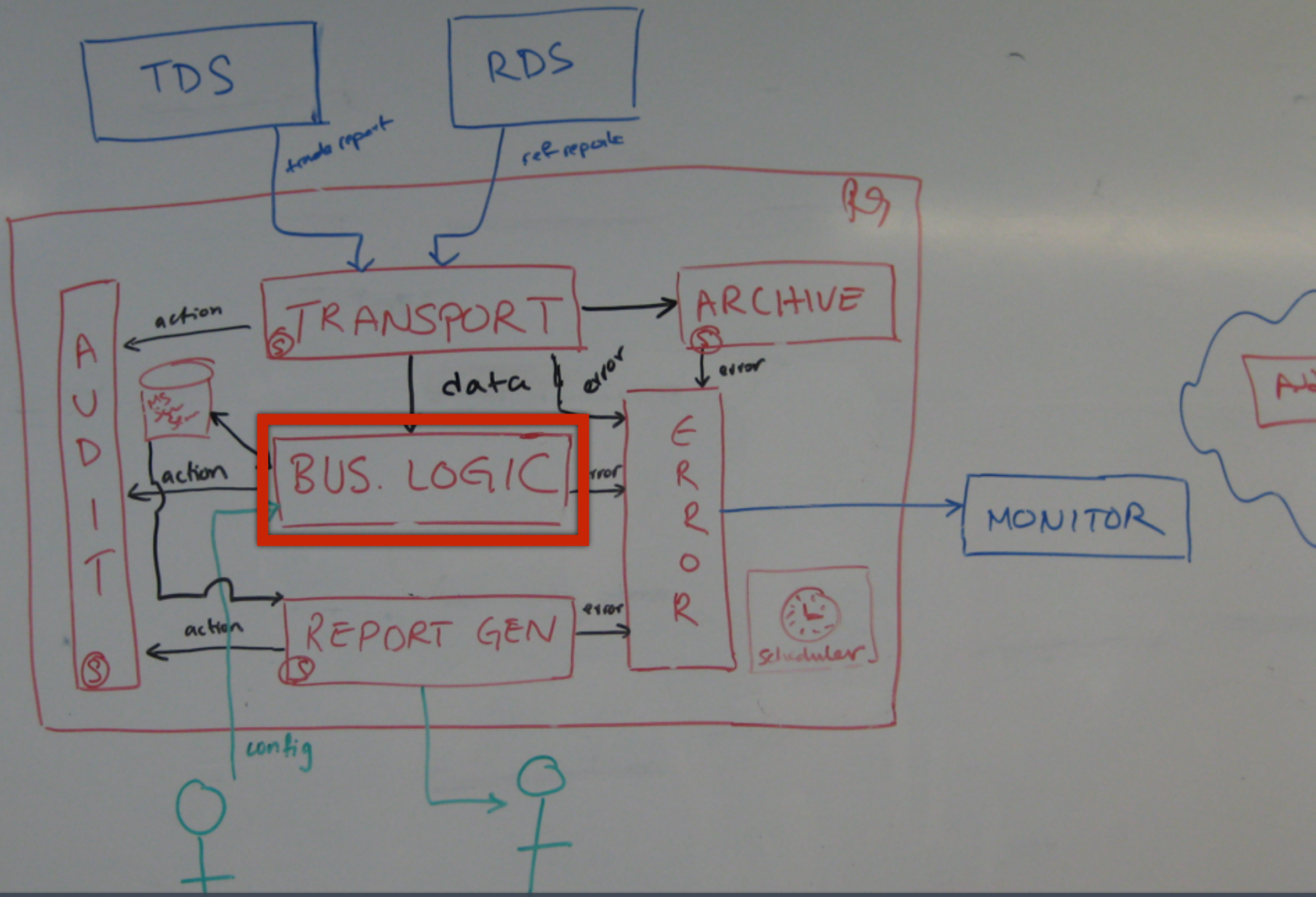
The Functional View



Stormtroopers

The Airline Route Map





Generically True

XML

XML

2 SERVICE/BATCH

IMPORT
JOIN
CALC

EXPORT/STORE
REPORT

CENTRAL
MONITORING
SERVICE

SNMP

3 WEBAPP

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

RISK RULES
USERS
ARCHIVE

The Logical View

Bus. User

Spec. User

Web Server

RiskReportGUI

RiskConfigGUI

Appl. Server J2EE

RiskReportService

RiskCalcParamService

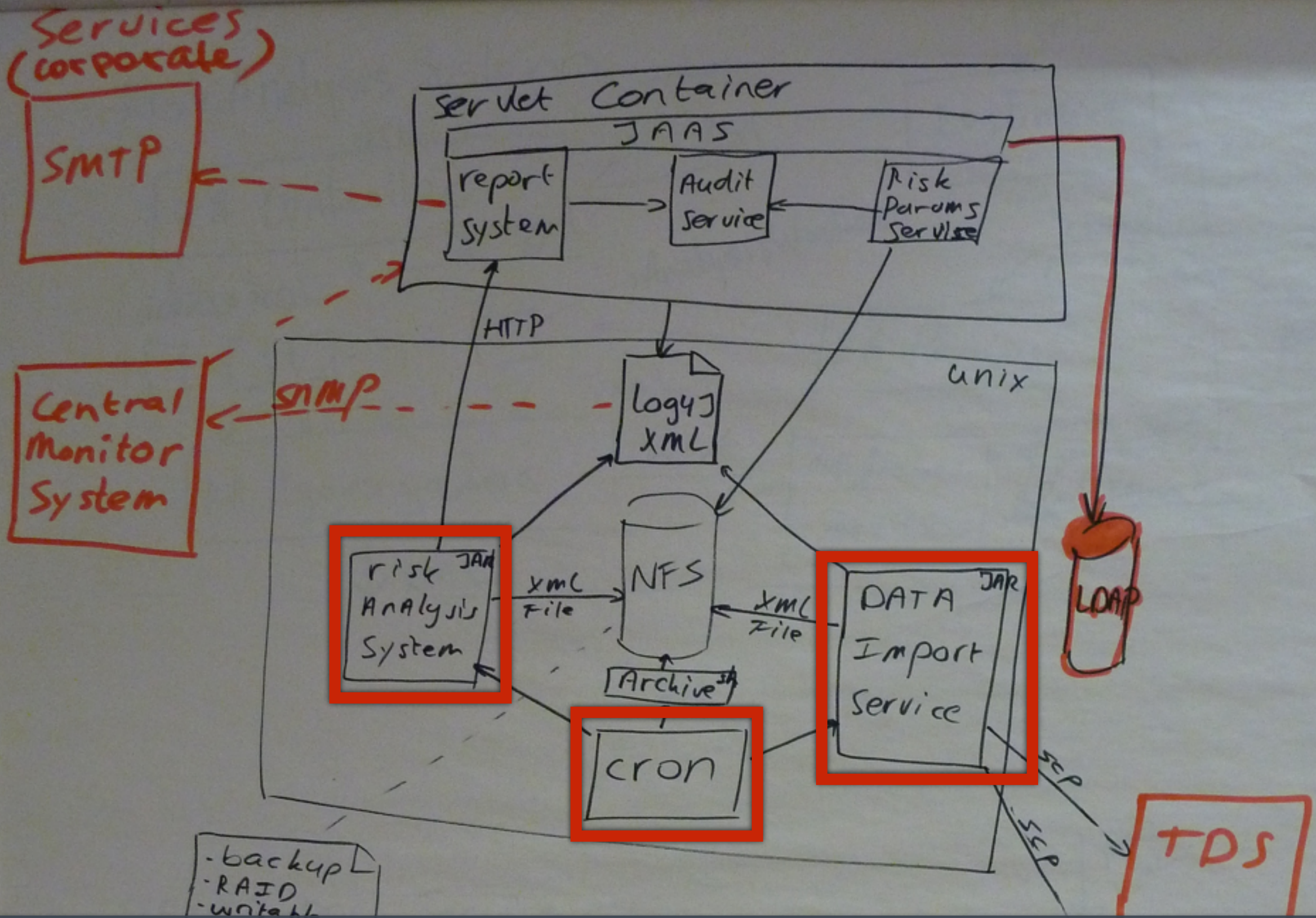
Active Directory

Auth. Service

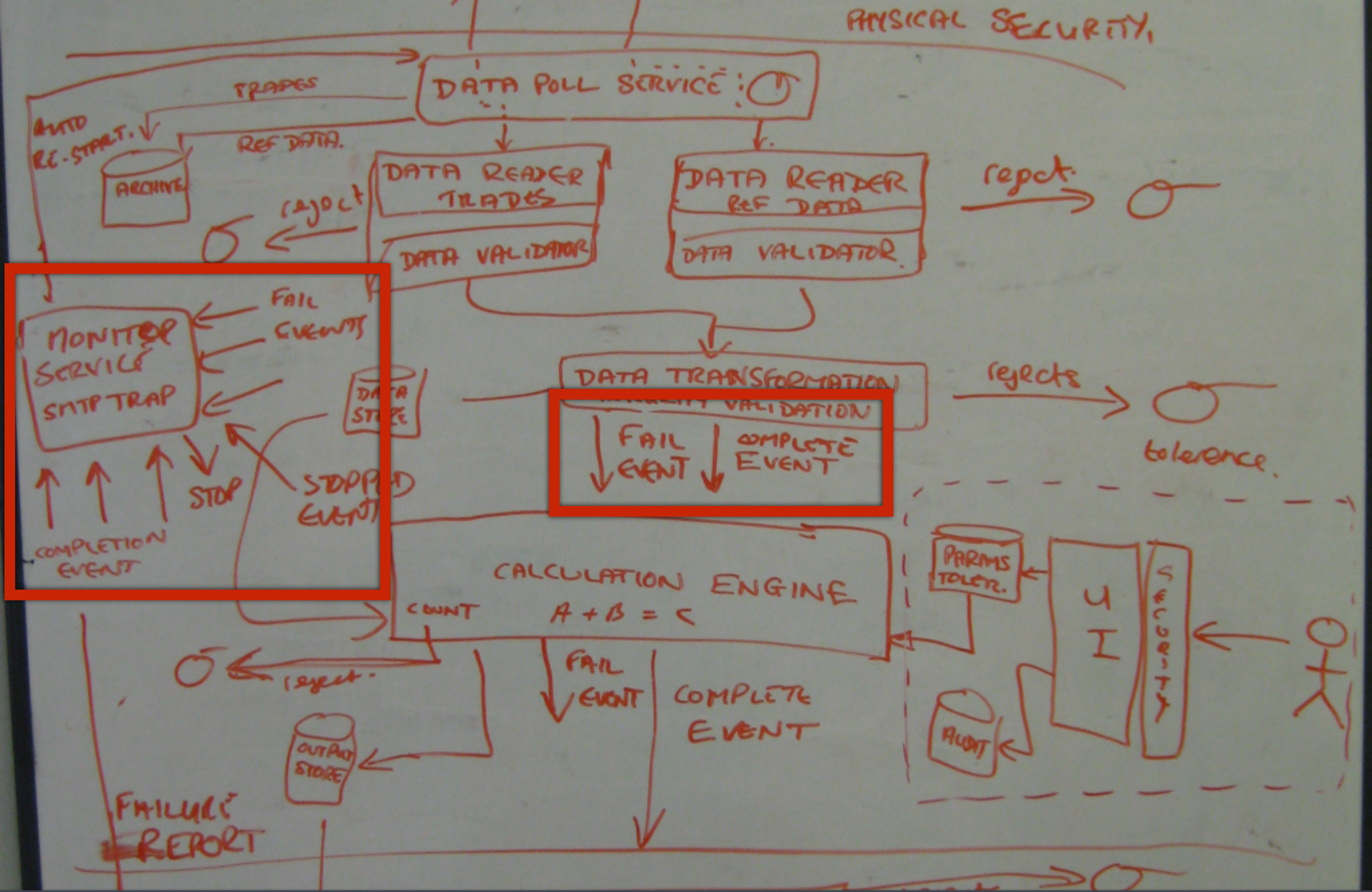
Auclit Service

Missing technology details

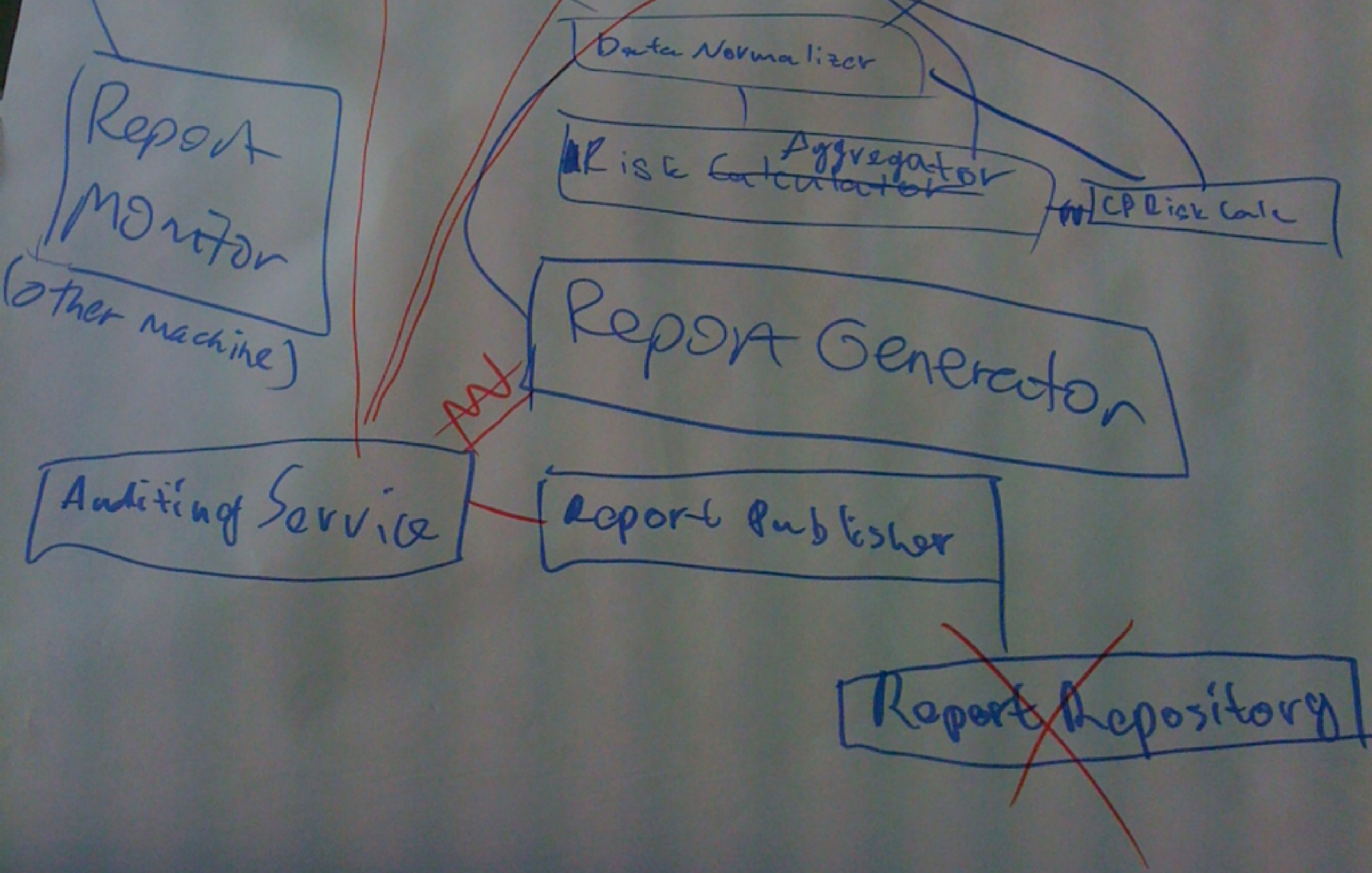
Deployment vs Execution Context



Homeless Old C# Object (HOCO)

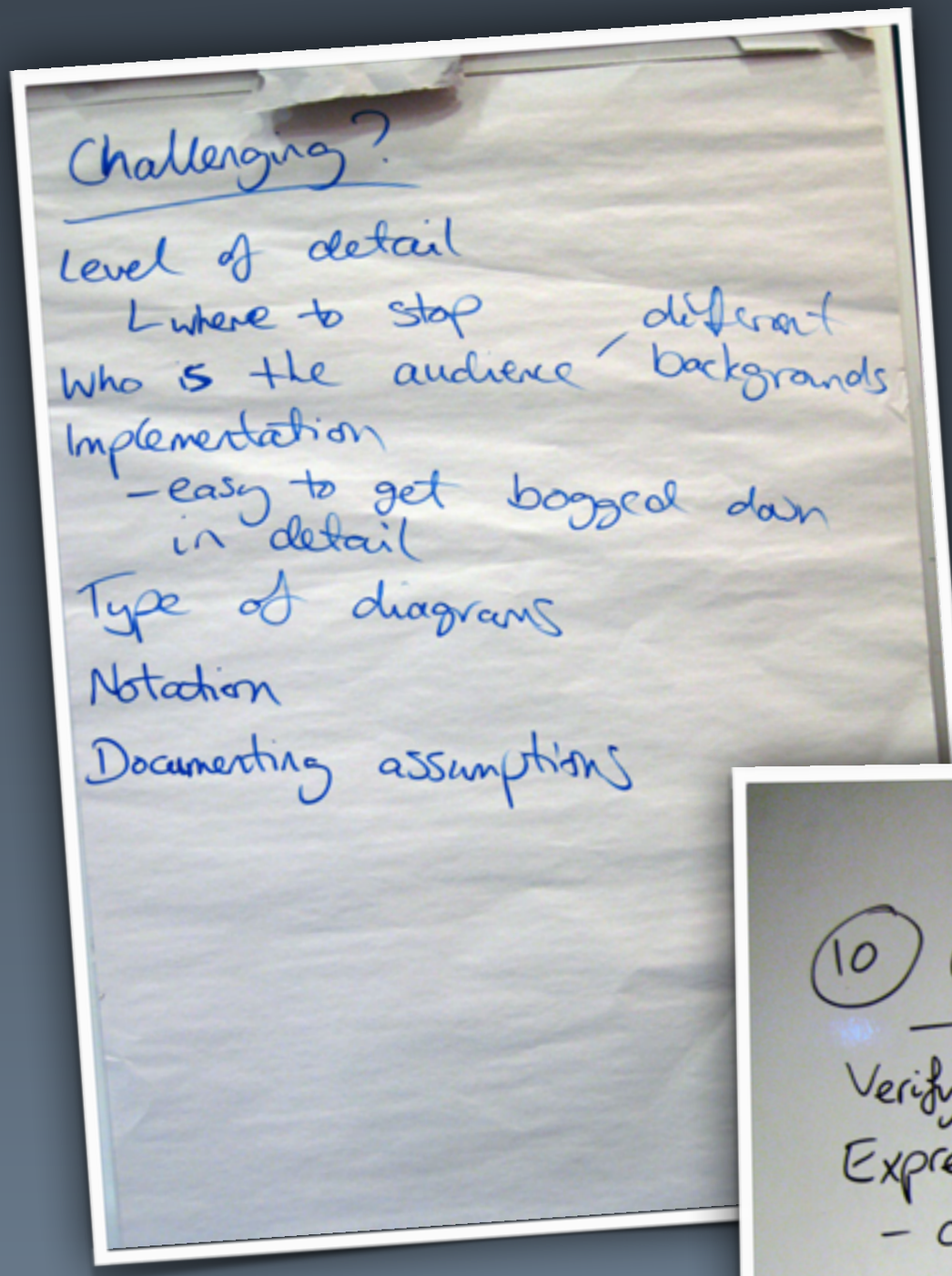


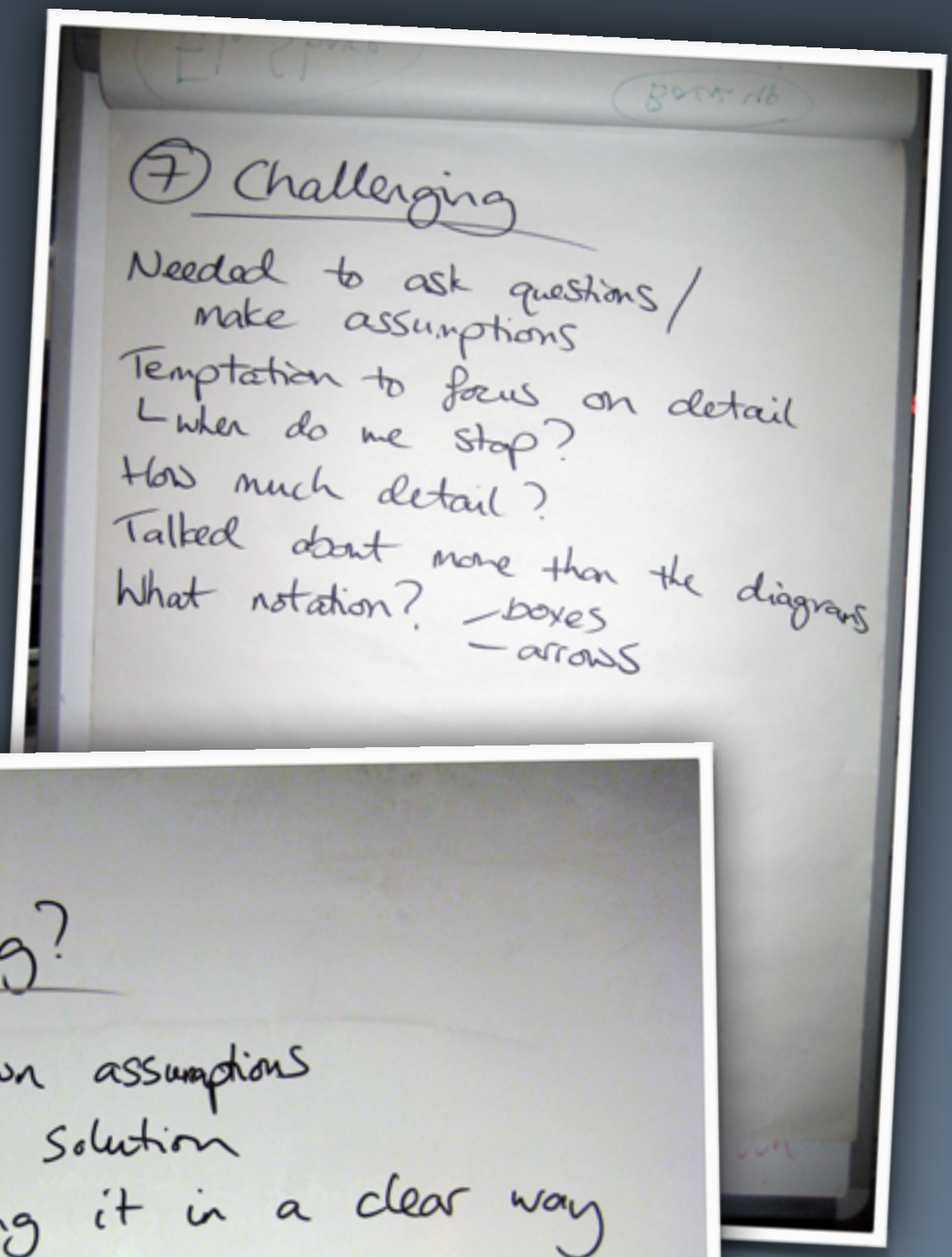
Choose your own adventure

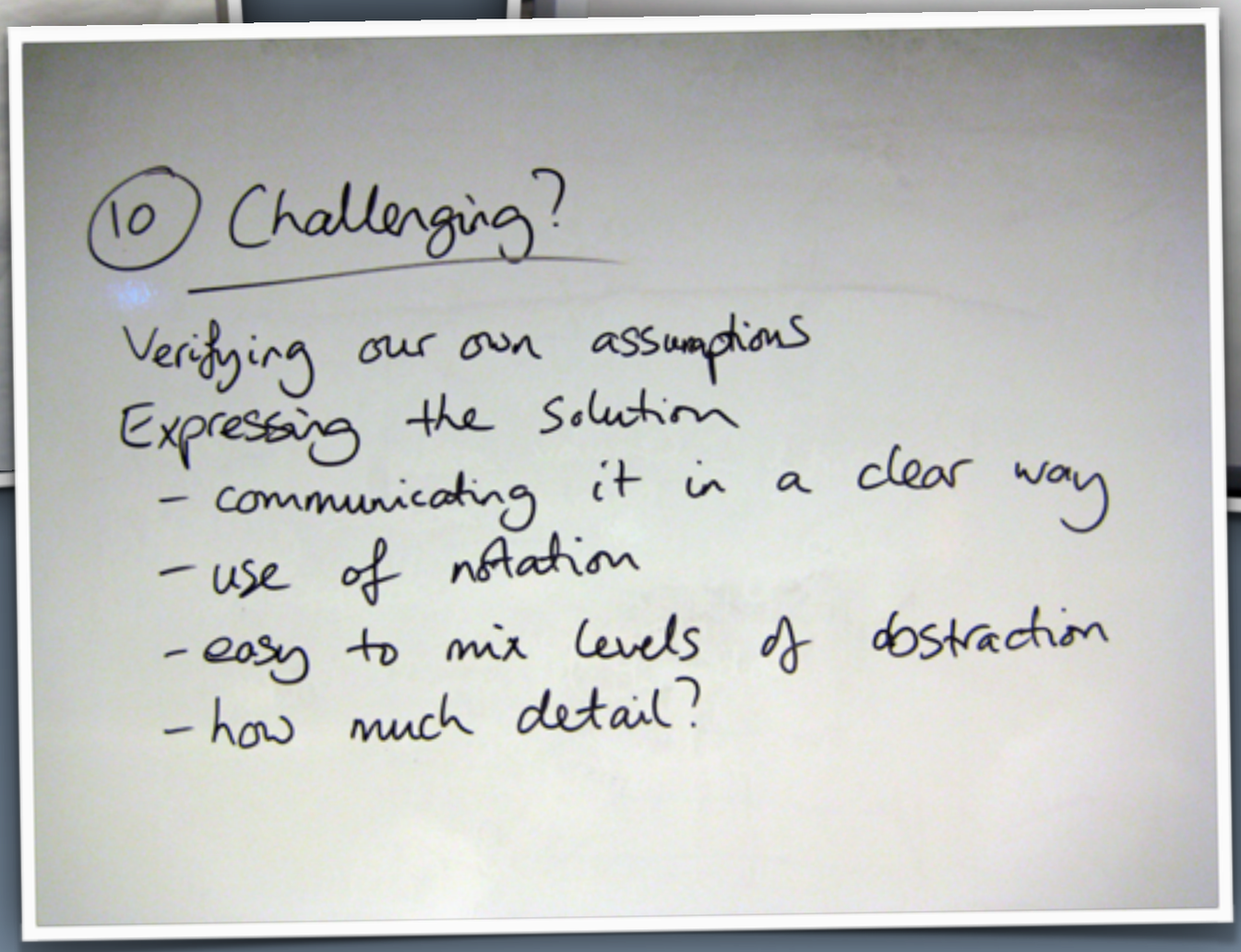


Should have used a whiteboard!

Eh?







What's been challenging about the exercise?

People expect to present their designs and therefore

information is still stuck in their heads

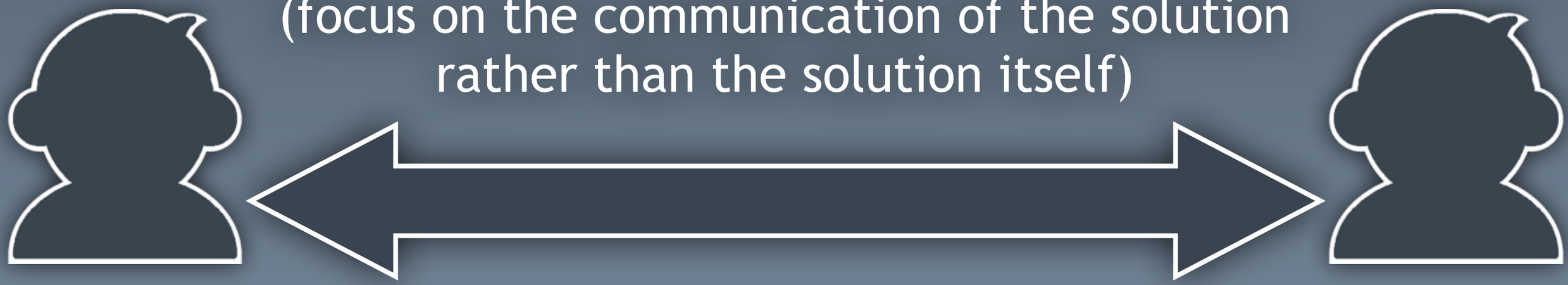
The diagram isn’t self-evident, but we’ll explain it

Review the diagrams

3 things we **like** about the diagrams

3 things we think would **improve** the diagrams

(focus on the communication of the solution rather than the solution itself)



NO ANNOTATION ON FLOWS

SHOULD USE MORE COLORS

POST ITS CAN FALL OFF

what does colour mean?

—

MIXES DIFFERENT LEVELS OF DETAIL

NOT SURE OF TRANSITION BETWEEN DIFFERENT DIAGRAMS -

—

CONFLICTING LEVELS OF DETAIL IN PRESENTATION

~~Meaning of Notation~~

What about the different arrows?

objects vs actions

WHY ARE SOME LINES PINK ?

WHAT DO THE SHAPE MEAN ?

UML is good. But Not Everyone knows it

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

what shapes mean

What's the DB-like icon?

Not sure what this is

DIFFERENT LEVELS IN SAME DIAGRAM

ARE THE ARROWS THE RIGHT WAY ROUND?

Who here uses UML

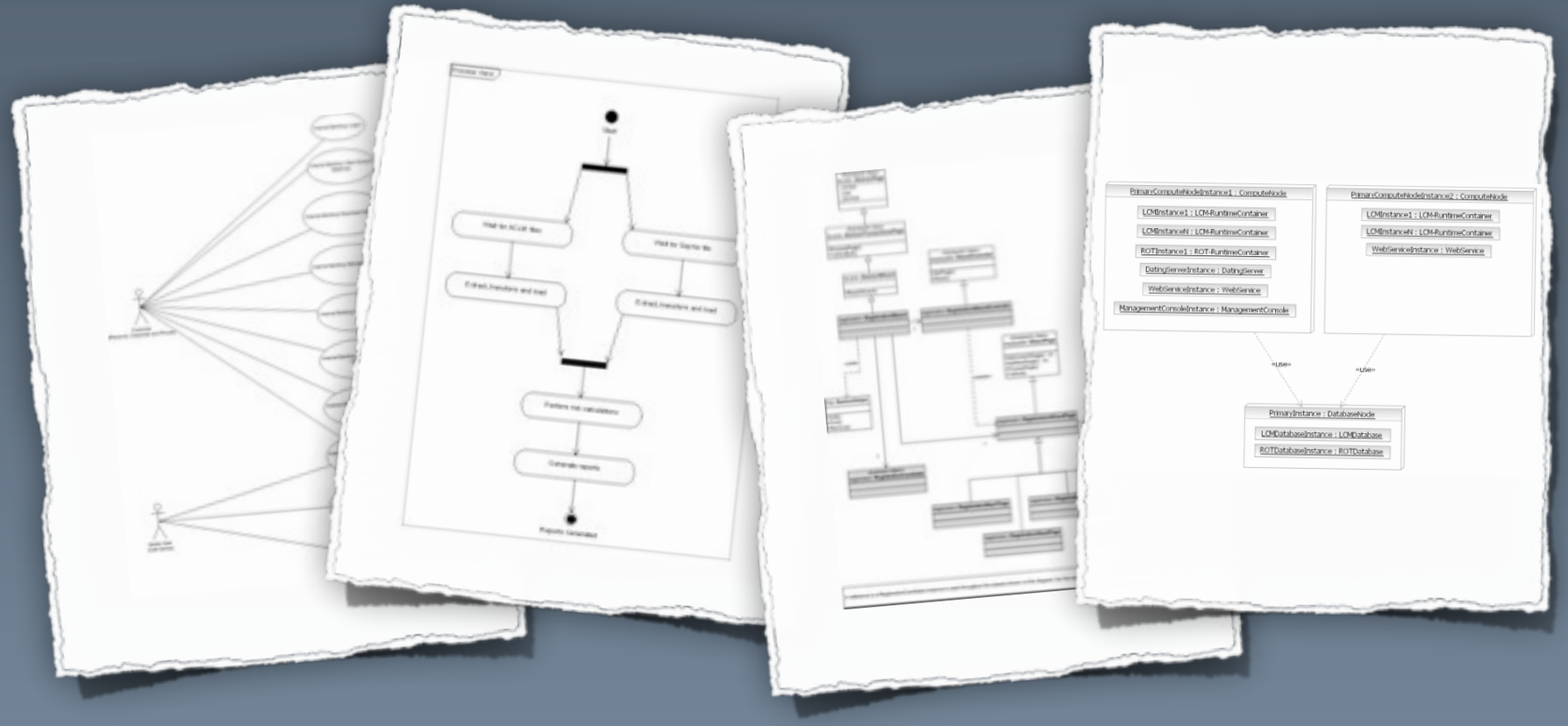
on a regular basis?

9 out of 10 people don't use UML

(in my experience)

I do use UML

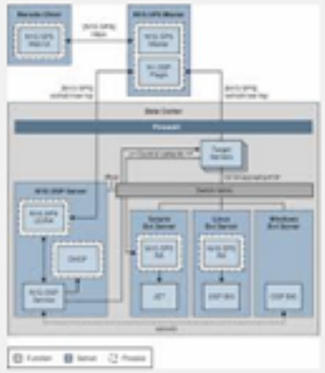
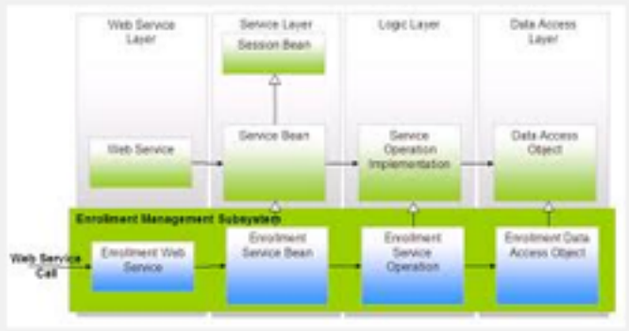
(activity, class, sequence, collaboration, state)

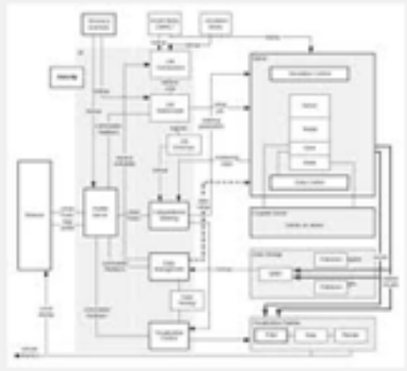
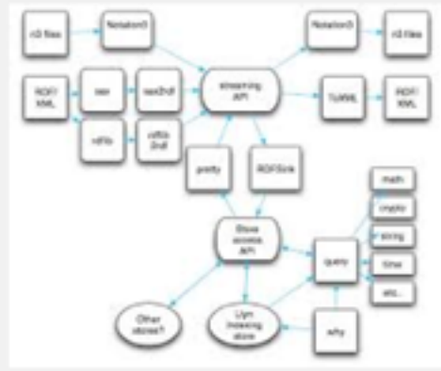


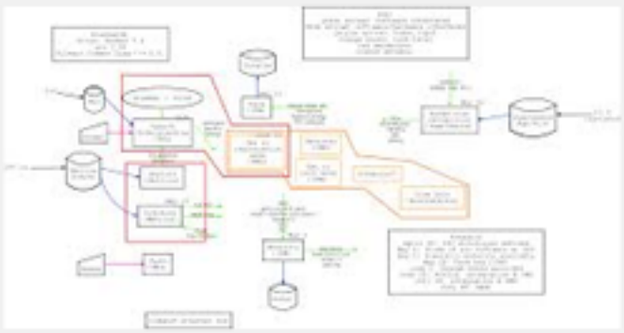
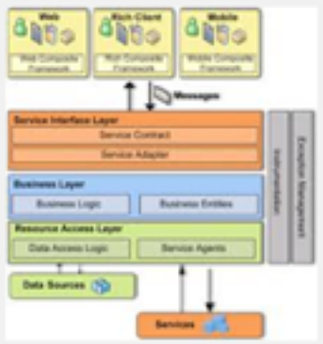
Google software architecture diagram

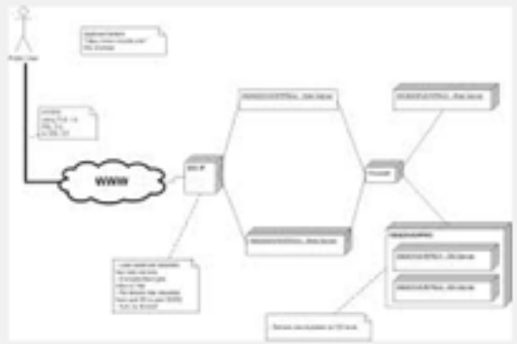
Sign in

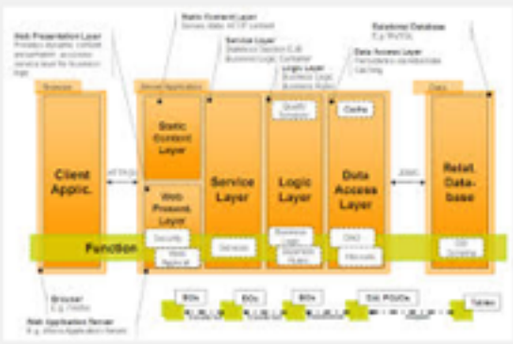
Web Images Videos Shopping News More Search tools SafeSearch

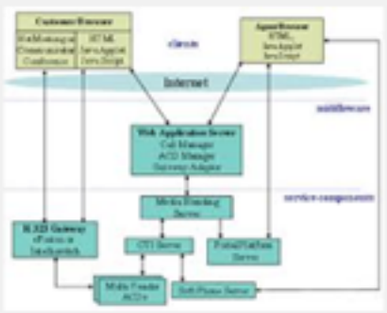




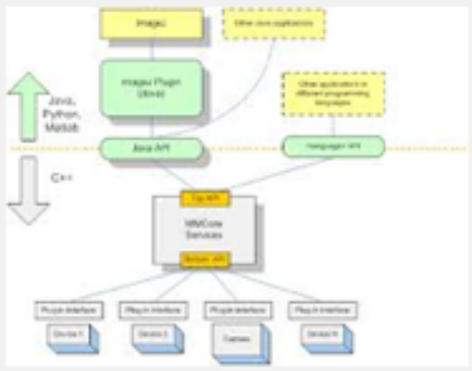



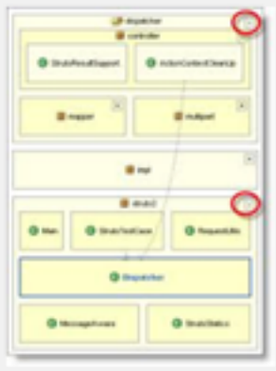



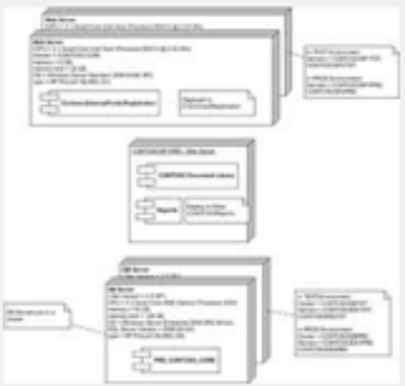
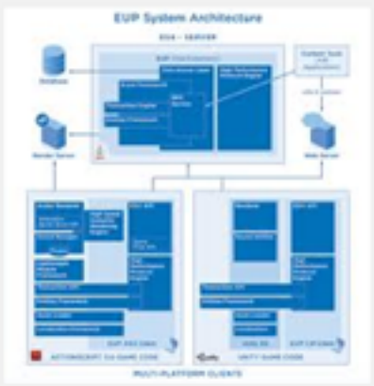






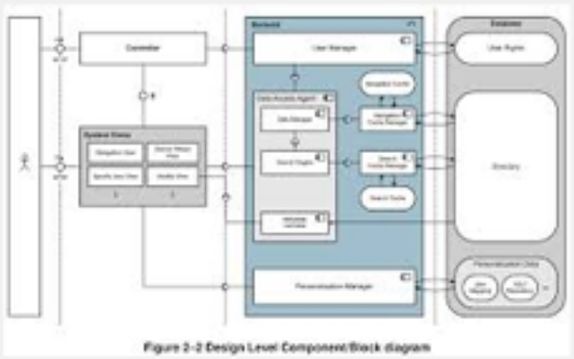
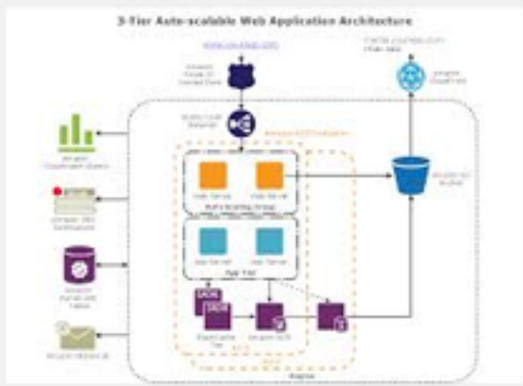
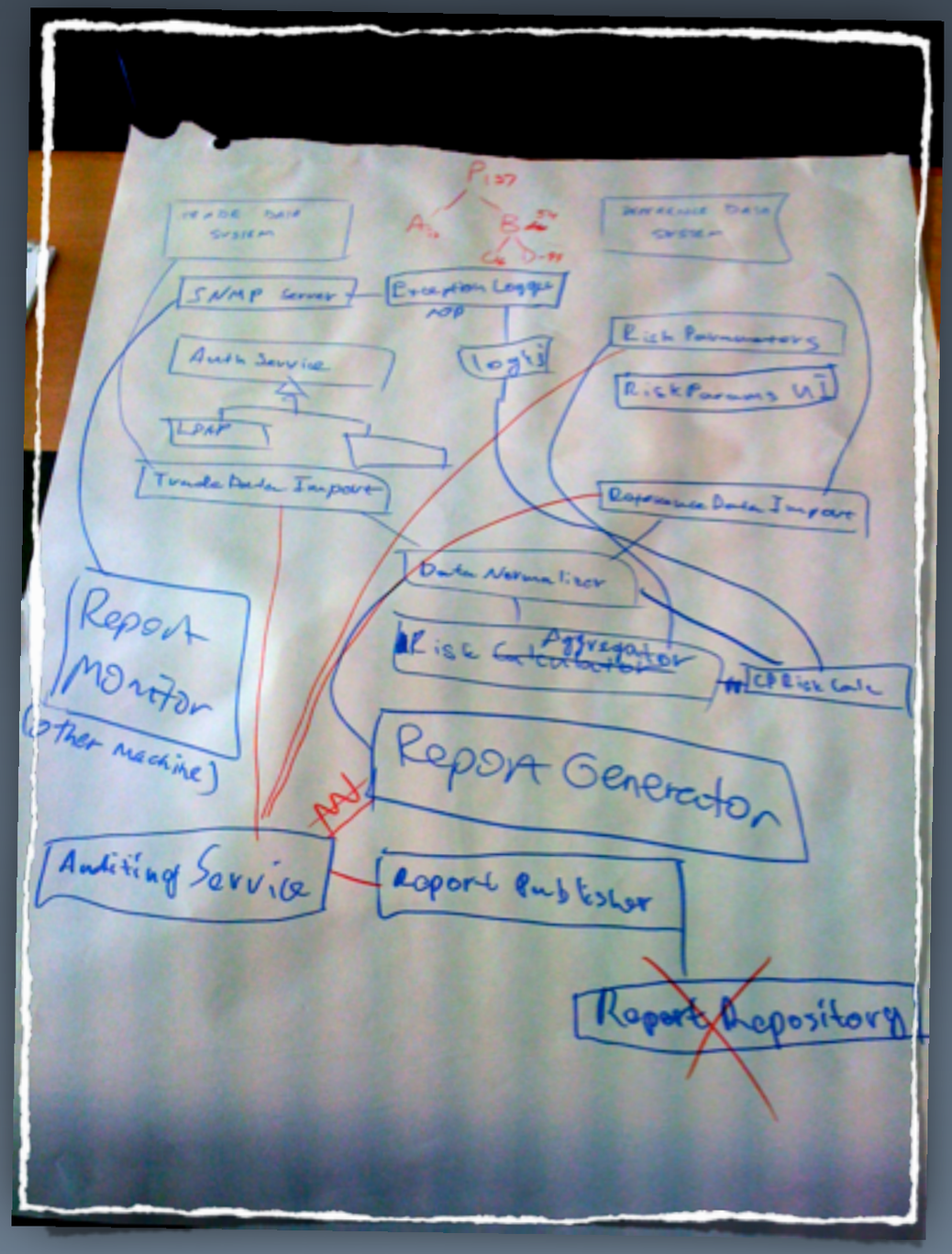


Figure 2-2 Design Level Component/Block diagram


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
RTP/ RTCP
Voice over IP
Acoustic echo cancellation (AEC)
LEC (G.168/ G.165)
AGC - Gain Control
VAD
Speech Encoder
Audio Interface layer
Sound card Interface (PCI / USB)
FXS/TDM Interface
Telephony Interface layer - Tone detection/generation, CPT/DTMF, MF-R1/R2, and Voice Fax Modem Discrimination Unit
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
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
Service Controller
Rapid Server Imaging
Rich UI
Web UI
Controls
Service Interface
Activity
Business Rules
Human Workflow
Workflow
Service Agents
DAL
E-Publish
EAI
ECM
DW
OLTP
Signing
Authentication
Authorization
Monitoring
Log & Trace
Exception Management
Configuration
Presentation Sublayer
Project Editing Service
Computational Analysis Service
e-AIRSmesh
e-AIRSview
Mesh Converter
CFD Solver
Output Converter
Remote Experiment Service
Exp. Exec.
Exp. Info.
Aerodynamic Exp.
PIV Exp.
Parametric Study
Validation
Analysis
Collaboration
Session Reservation
Resource Configuration Service
Resource Monitoring Service
P / HTTP
ASP (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 Conference
Chat
Application
File Editor
ASCII Files
Commercial DB
Commercial Develop tools
Library
SCADA Develop. Environ
Export / Import
Project Editor
Driver Toolkit
Client
Server
Publish
Subscribe
TCP/IP
SCADA Server
Recipe DB
Recipe Managt
Ref. DB
Data Proces-sing
RT DB
Report Gener.
SQL
Alarm
Alarm DB
Log
Log DB
Archive
Archive DB
RT & Event Manager
ODBC
DDE
API DLL
Data R/W
Driver
OPC
Private Application
EXCEL
VME
PLC
PLC
Active X Controls
3rd Party Applic.
Active X Container
Typical generic software architecture of SCADA systems
ote administration interface
Output device manager
Output devices
User manager
DB-Server
Analysis

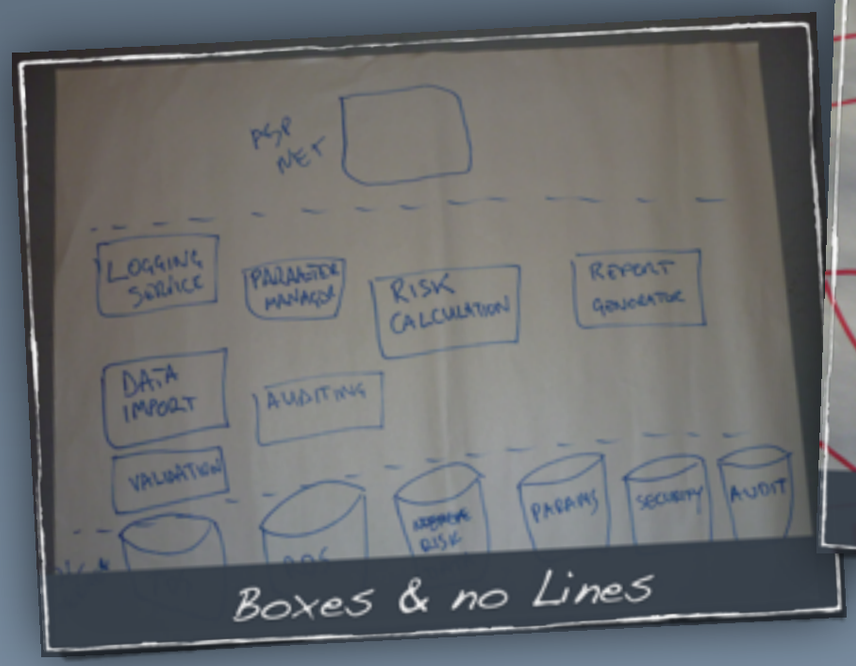
In my experience,

software teams aren't able to effectively visualise the software architecture of their systems

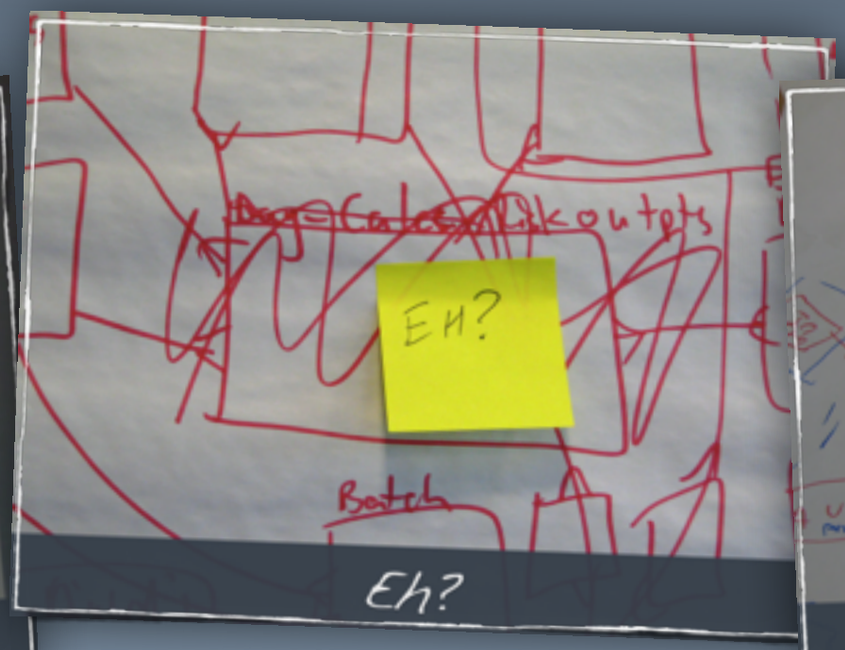


We can visualise our process…

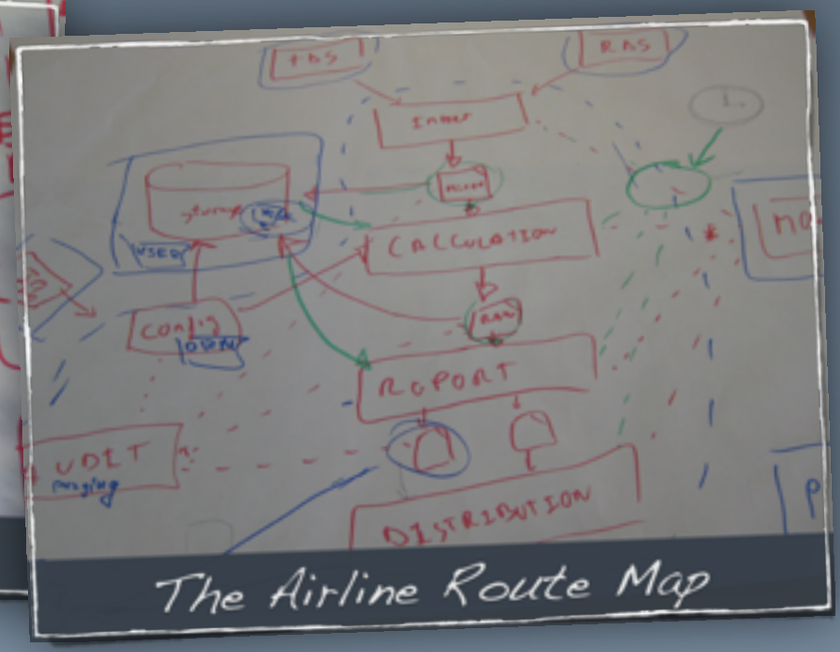
…but not our software!



Boxes & no Lines



Eh?



The Airline Route Map

Moving fast requires good communication

Notation

Titles

Short and meaningful, numbered if diagram order is important

Lines

Make line style and arrows explicit, add annotations to lines to provide additional information

Layout

Sticky notes and index cards make a great substitute for drawn boxes, especially early on

Some tips for

effective sketches

APPLICATION SERVICE COMPONENTS
REPORT GENERATOR
DATA IMPORTS
RISK ANALYSIS SERVICE FACADE
EMAIL GENERATOR
VALIDATION ENGINE
RISK CALCULATION ENGINE
UI SERVICES
WEB SERVER
MONITORING LOGGING
AUDIT MANAGER
DATA ACCESS
FACADE
A.D.
SERVICE
TDS
RDS
CONTEXT
SCHEDULING SYSTEM
RISK SYSTEM
MONITORING SYSTEM
AUDIT SYSTEM
BUSINESS USER
ADMIN BUSINESS USER
KEY:
NEW
EXISTING
BOUNDARY
BUSINESS LAYER
SERVICE LAYER
DATA LAYER
database
SYSTEM DRIVER
DATA LAYER
COMPONENTS
AUTH SVC
Trade Data (XML)
XML processing (parsing + schema)
Persistence
ORACLE
permanent ?
Risk Engine
Monitoring + Alerts
(error Reports?)
report gen.
CREATES
USES
SPRING SECURITY
SPRING MVC
LOGGING
CALCULATION
AUDIT
IMPORT XML IN DB
JOB SCHEDULING
PARAMETER
TRADES
FILESYS.
DATA
Context
XML TRADE DETAIL
HEALTH CHECK
EXCEL REPORT COPY
OPEN REPORT
NOTIFY

Titles

Short and meaningful, numbered if diagram order is important

Lines

Make line style and arrows explicit, add annotations to lines to provide additional information

Layout

Sticky notes and index cards make a great substitute for drawn boxes, especially early on

Labels

Be wary of using acronyms

Colour

Ensure that colour coding is made explicit

Orientation

Users at the top and database at the bottom? Or perhaps “upside-down”?

Shapes

Don’t assume that people will understand what different shapes are being used for

Keys

Explain shapes, lines, colours, borders, acronyms, etc

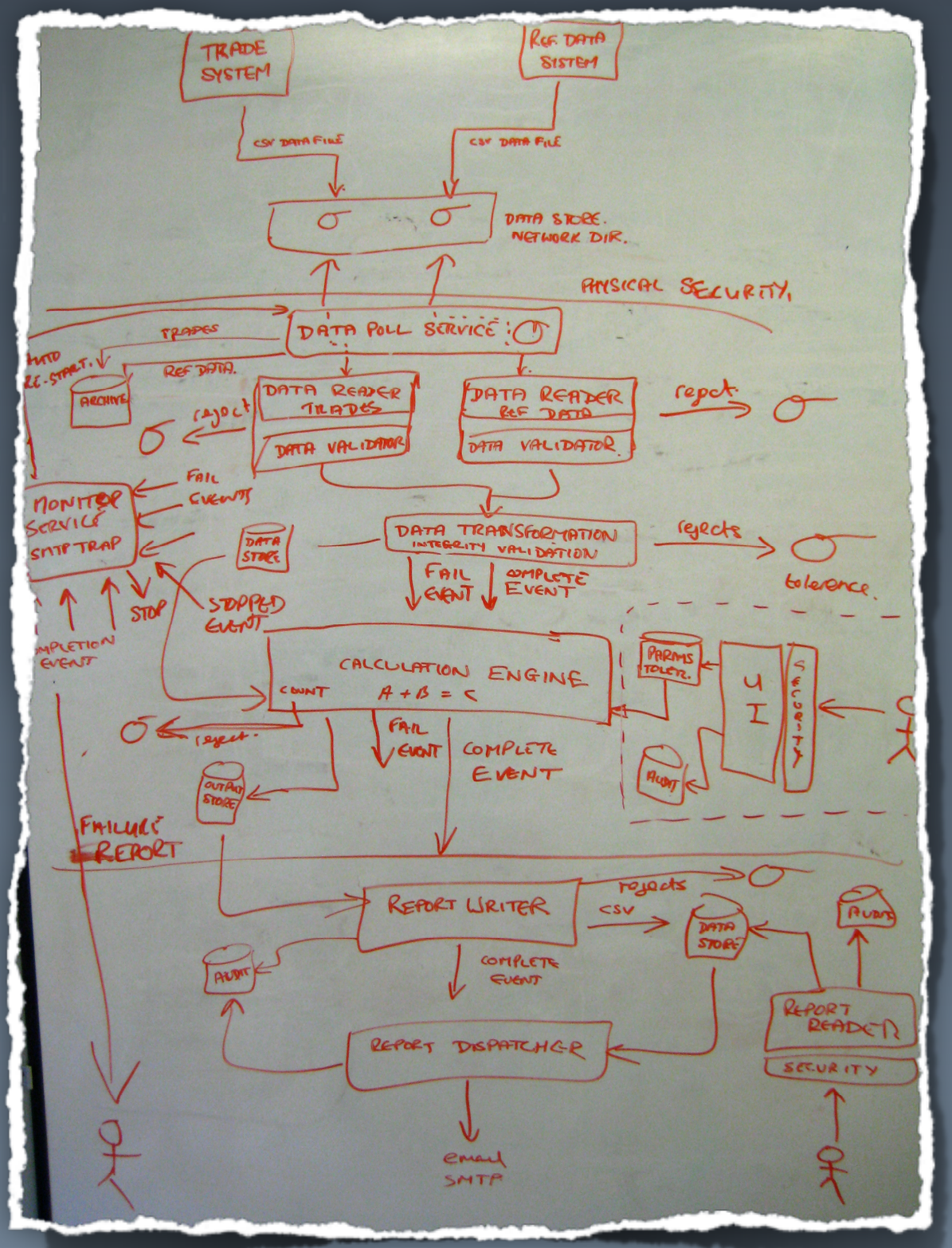
Responsibilities

Adding responsibilities to boxes can provide a nice “at a glance” view (Miller’s Law; 7±2)

Some tips for

effective sketches

Content



It’s usually difficult to show the entire design on a **single** diagram

Different **views** of the design can be used to manage complexity and highlight different aspects of the solution

Software architecture deals with abstraction, with decomposition and composition, with style and esthetics.

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

by 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.

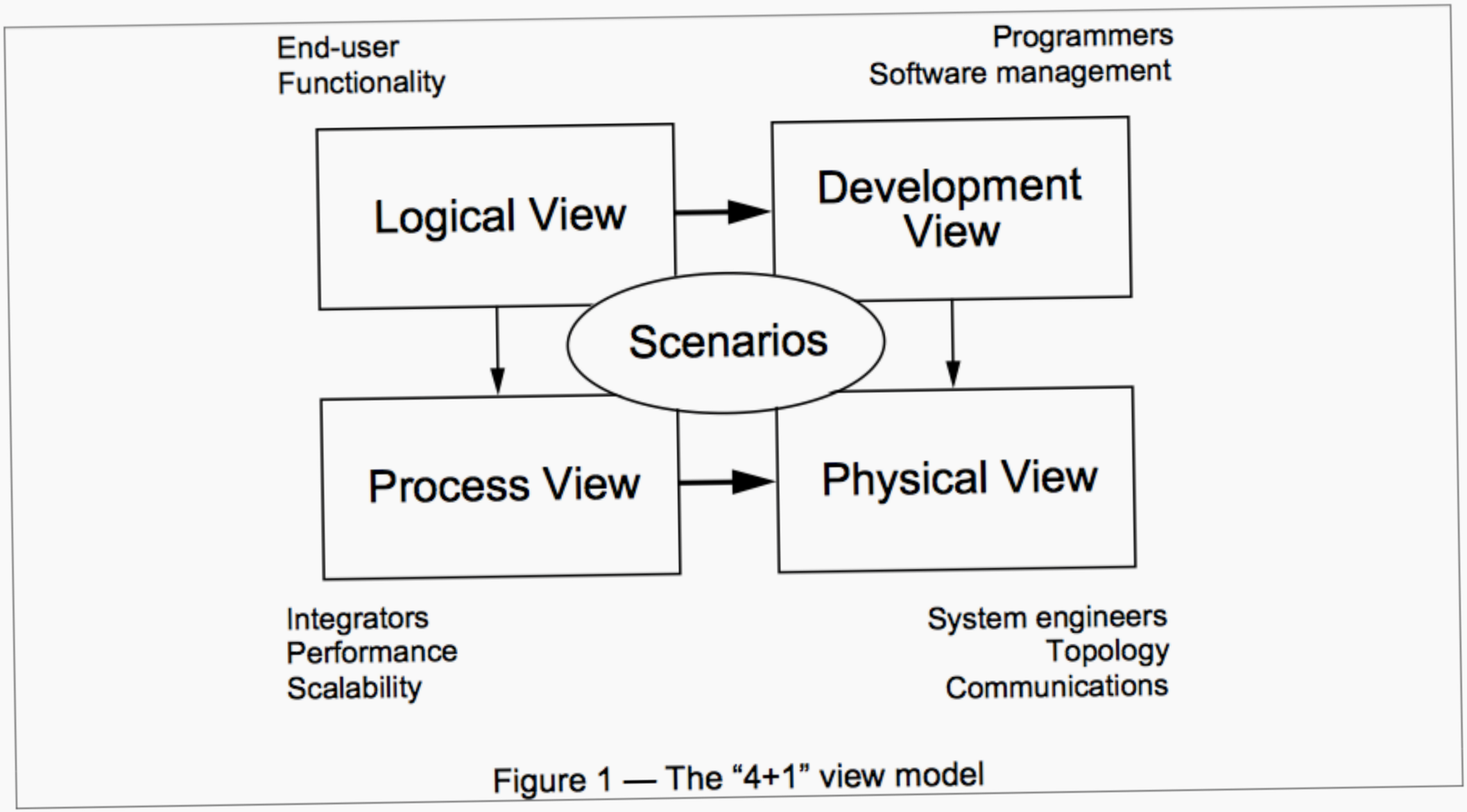
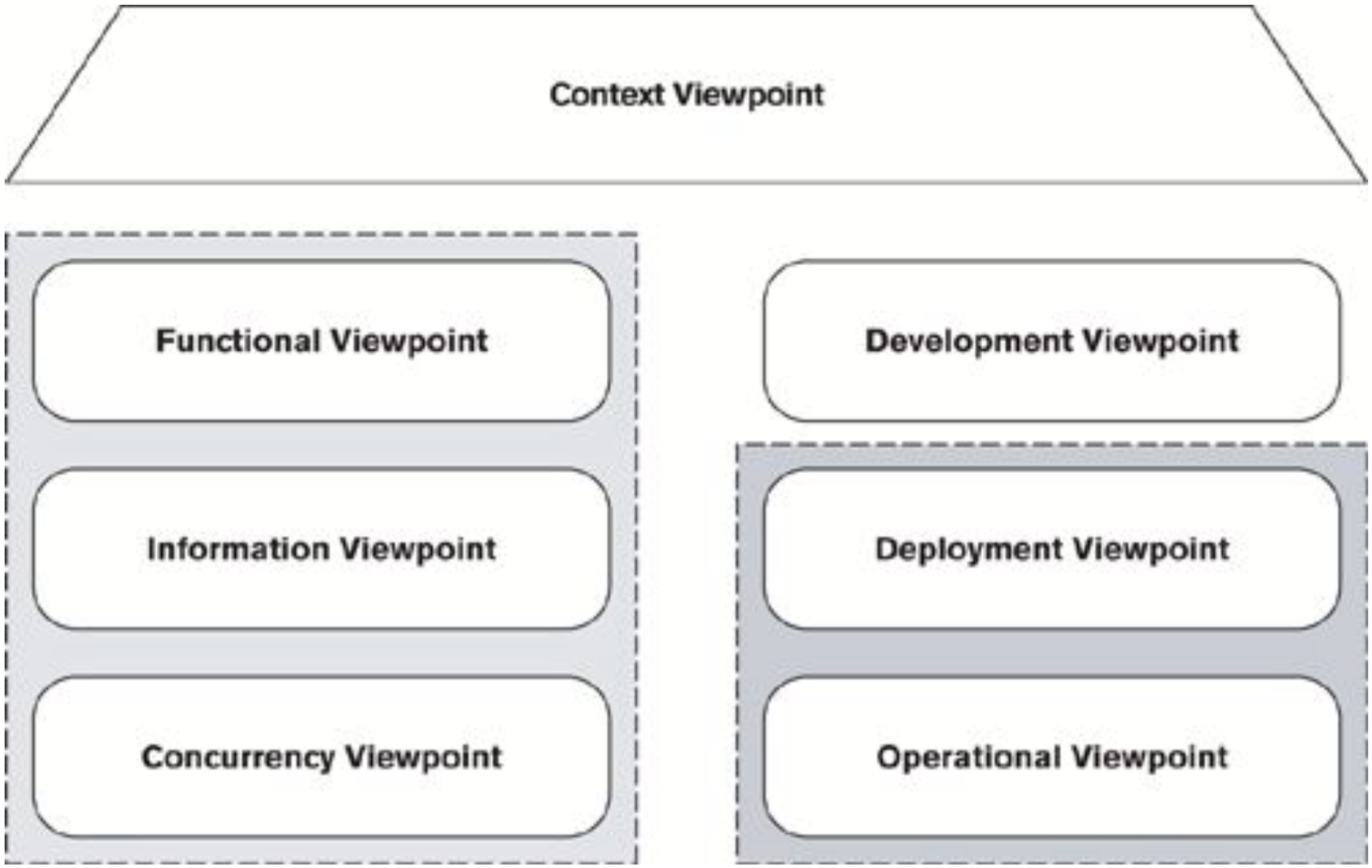
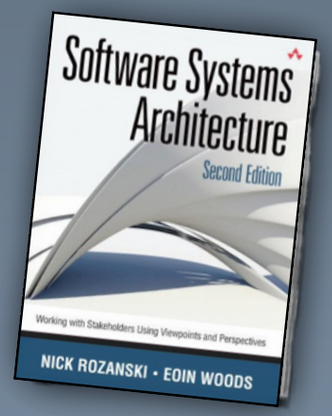


Figure 1 — The "4+1" view model

We apply Perry & Wolf's equation independently on each view, i.e., for each view we define the set of elements to use (components, containers, and connectors) , we capture the forms and patterns that work, and we capture the rationale and constraints, connecting the architecture to some of the requirements.



Viewpoints and perspectives



Viewpoint	Definition
Logical	The logical representation of the system's functional structure, normally presumed to be a class model (in an object-oriented systems development context). Our Functional viewpoint is a development of this "4+1" viewpoint, renamed to make its content clear (because you could have a number of logical aspects to an architecture).
Process	The concurrency and synchronization aspects of the architecture. Our Concurrency viewpoint is a development of this "4+1" viewpoint, renamed to avoid confusion with business process modeling.
Development	The design-time software structure, identifying modules, subsystems, and layers and the concerns directly related to software development. Our Development viewpoint is based on this "4+1" viewpoint.
Physical	The identification of the nodes on which the system's software will be executed and the mapping of other architectural elements to these nodes. Our Deployment viewpoint is a development of this "4+1" viewpoint.

Do the **names**
of those views make sense?

Conceptual vs Logical

Process vs Functional

Development vs Physical

Development vs Implementation

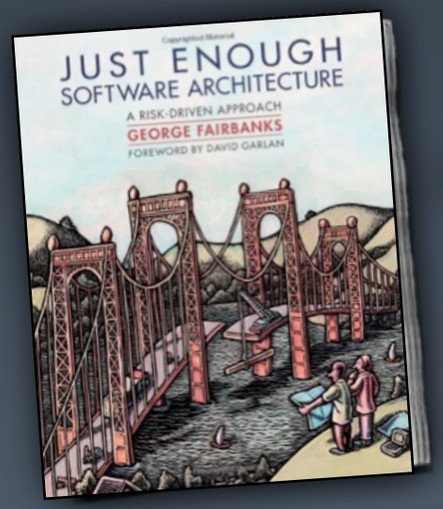
Physical vs Implementation

Physical vs Deployment

Why is there a

separation

between the logical and development views?



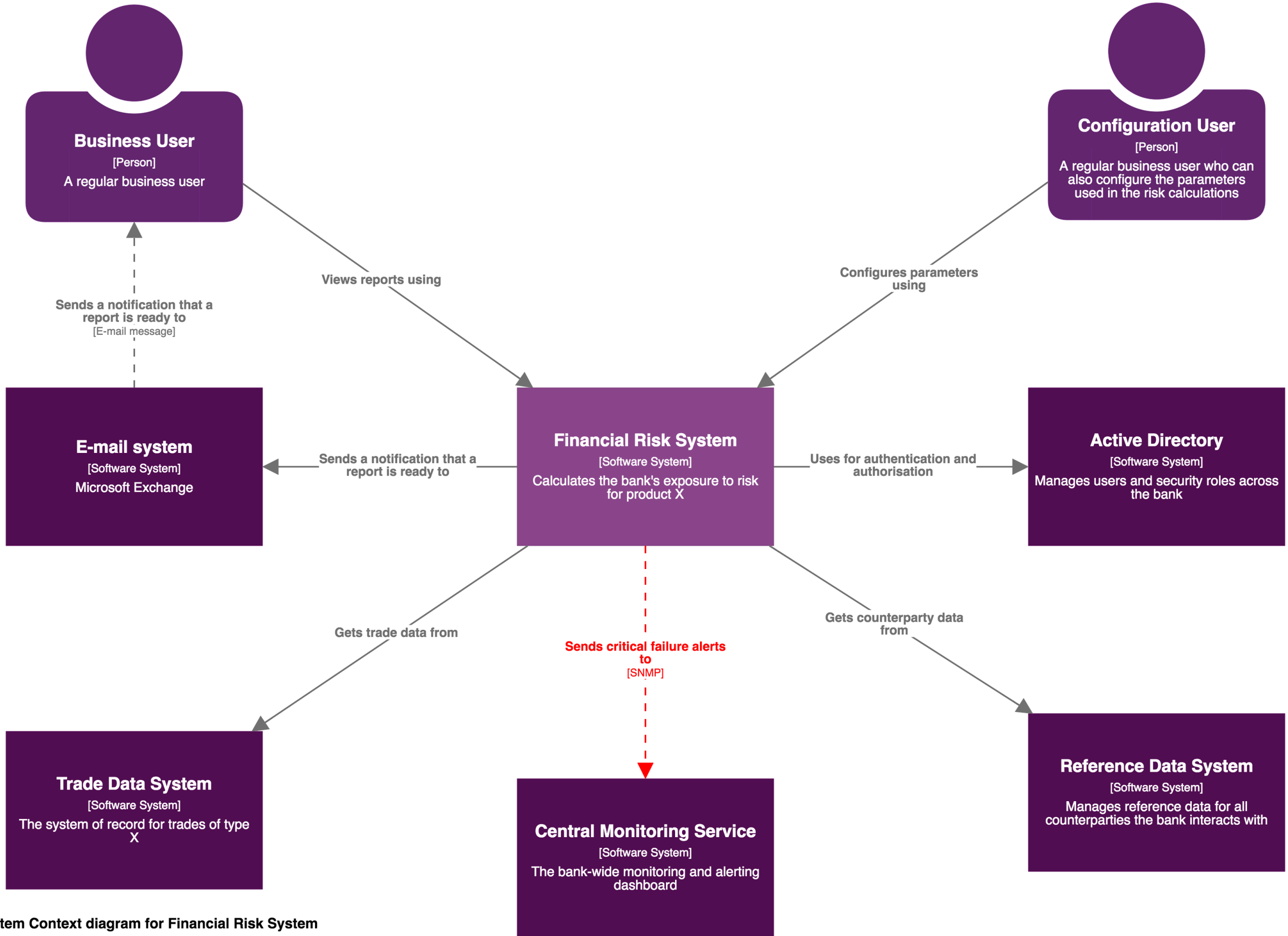
“the model-code gap”

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.

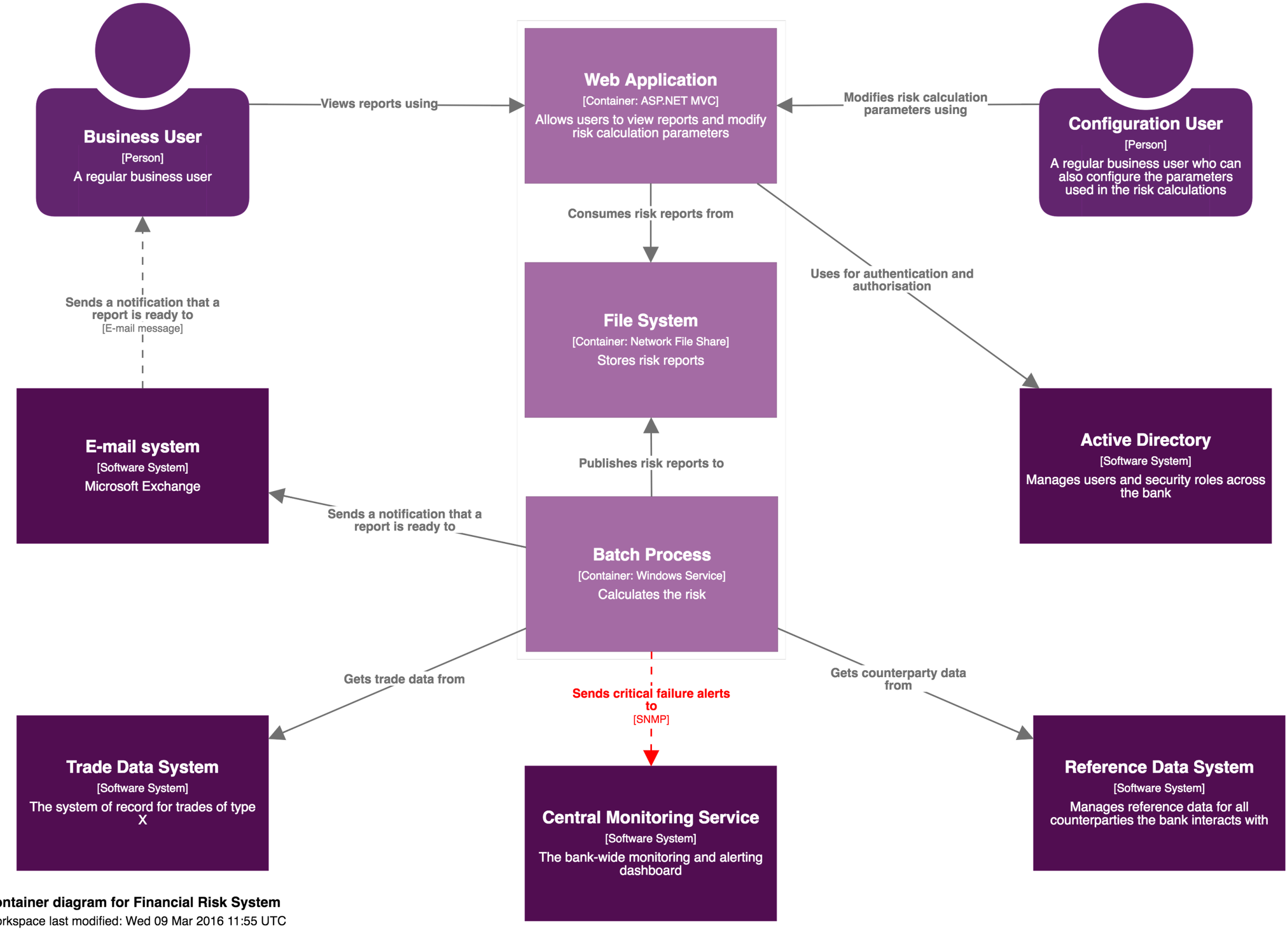
Upon learning about the model-code gap, your first instinct may be to avoid it. But reflecting on the origins of the gap gives little hope of a general solution in the short term: architecture models help you reason about complexity and scale because they are abstract and intensional; source code executes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

Do the diagrams reflect the code?



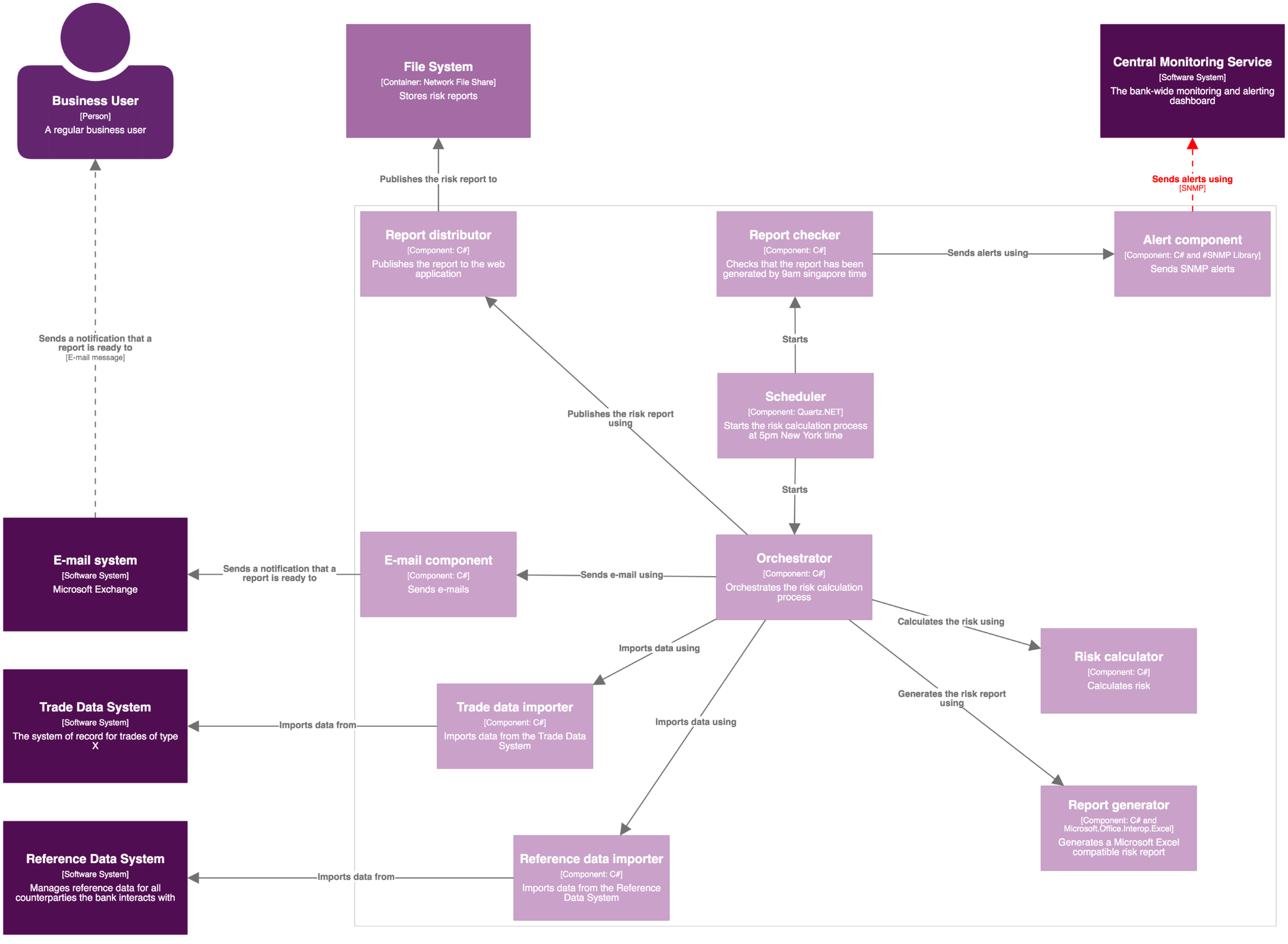
System Context diagram for Financial Risk System

Workspace last modified: Wed 09 Mar 2016 11:55 UTC



Container diagram for Financial Risk System

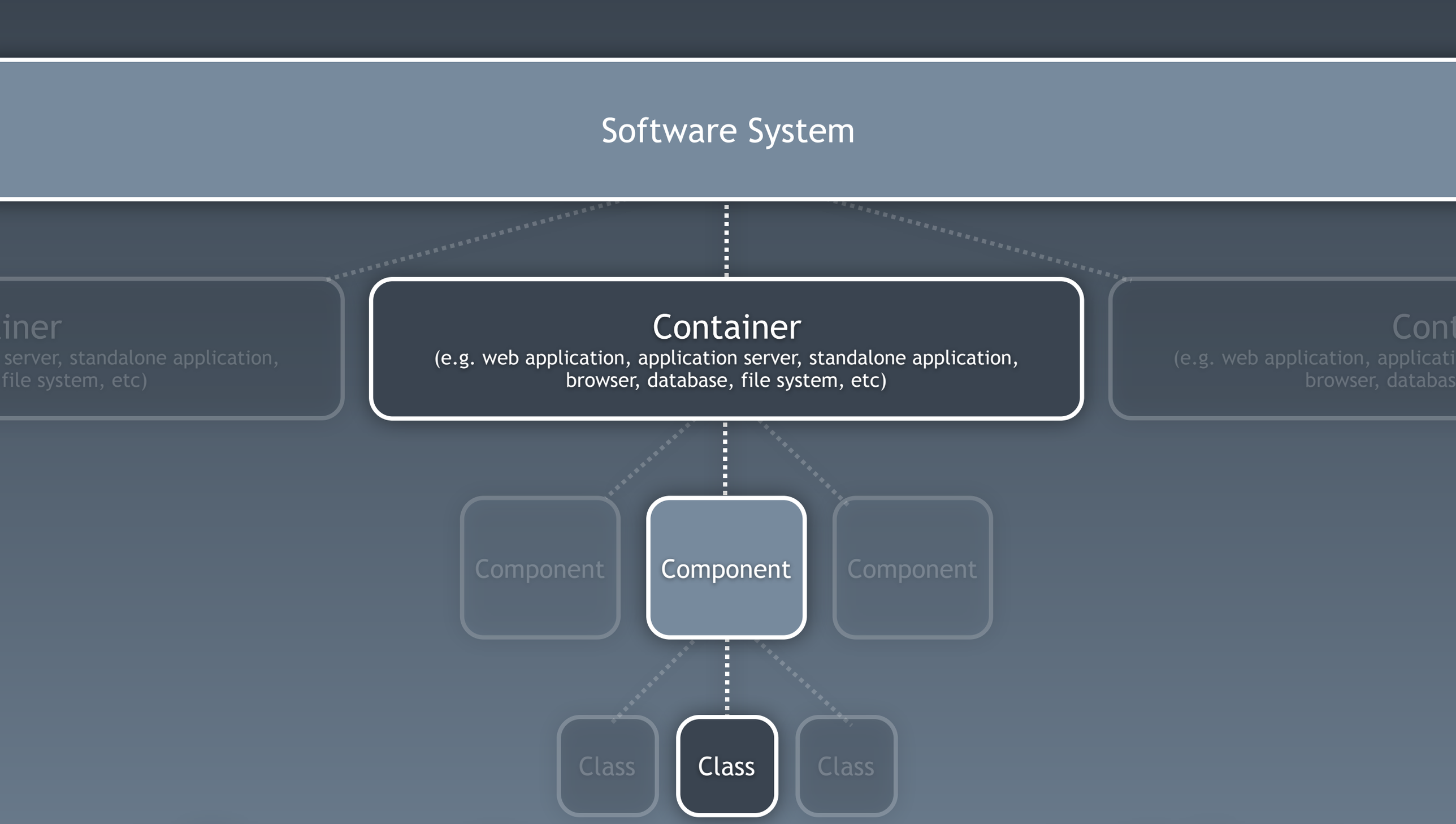
Workspace last modified: Wed 09 Mar 2016 11:55 UTC



Component diagram for Financial Risk System - Batch Process

Workspace last modified: Wed 09 Mar 2016 11:55 UTC

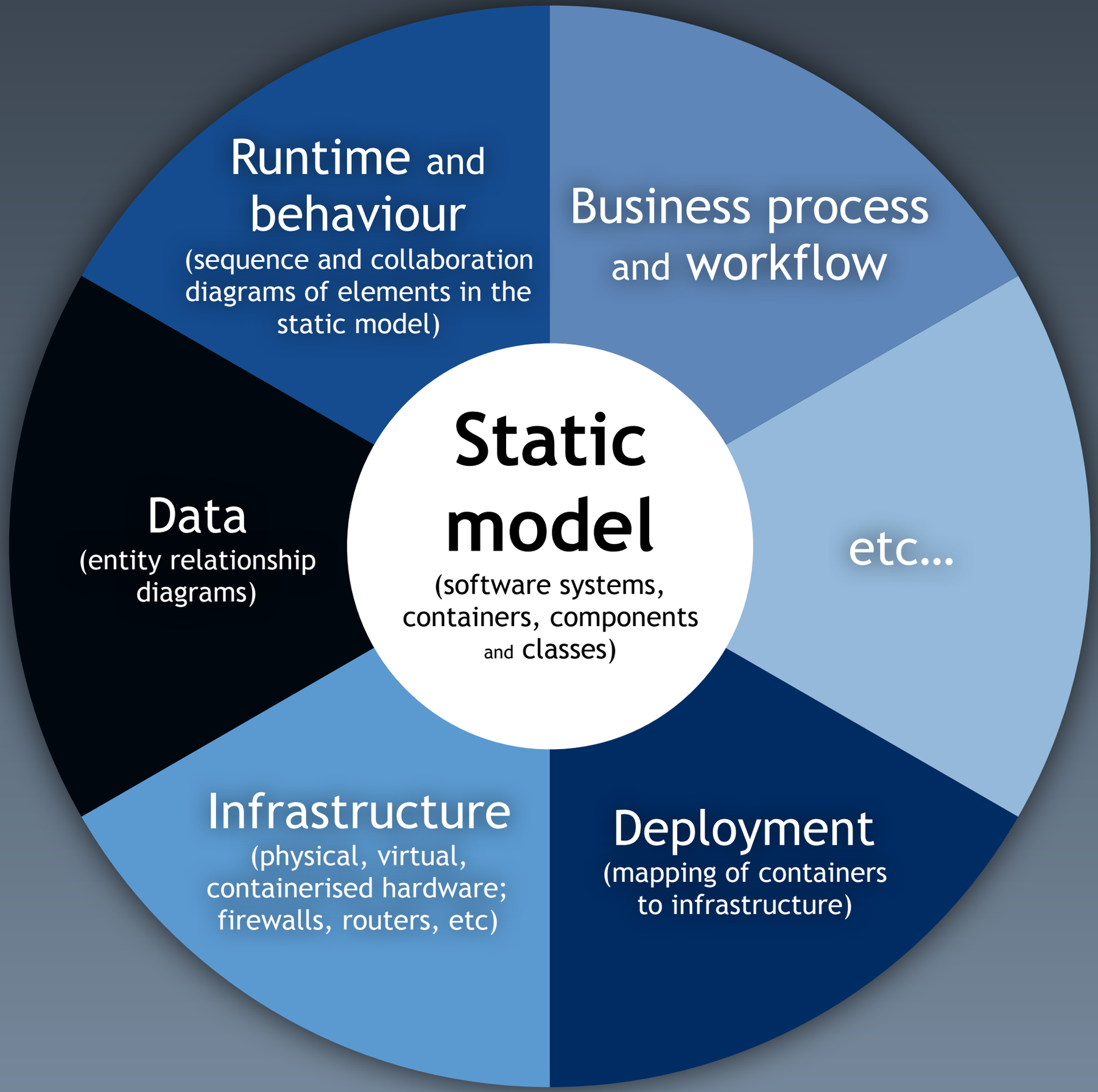
As an industry, we lack a common vocabulary with which to think about, describe and communicate software architecture



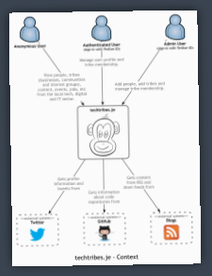
A **software system** is made up of one or more **containers**,

each of which contains one or more **components**,

which in turn are implemented by one or more **classes**.

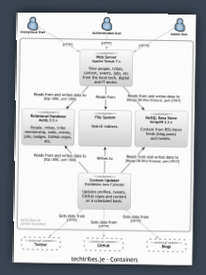

Runtime and behaviour
(sequence and collaboration diagrams of elements in the static model)
Business process and workflow
Static model
(software systems, containers, components and classes)
etc...
Data
(entity relationship diagrams)
Infrastructure
(physical, virtual, containerised hardware; firewalls, routers, etc)
Deployment
(mapping of containers to infrastructure)

The C4 model



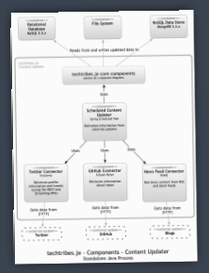
System Context

The system plus users and system dependencies



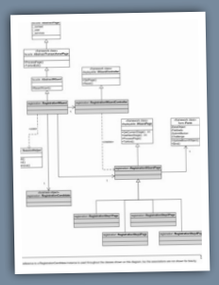
Containers

The overall shape of the architecture and technology choices



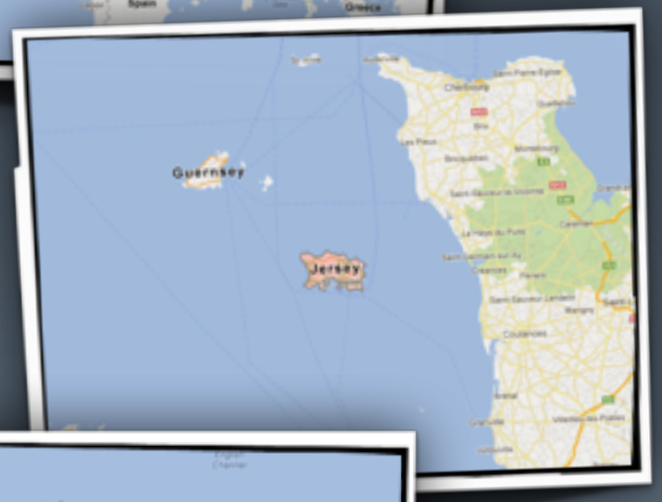
Components

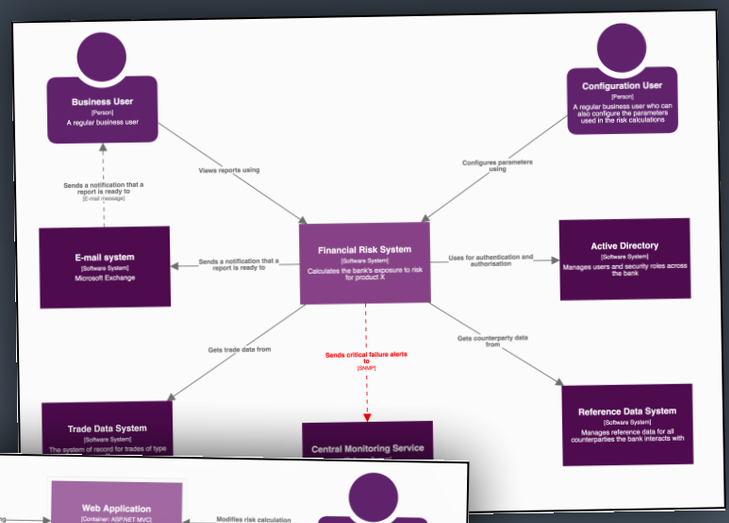
Components and their interactions within a container

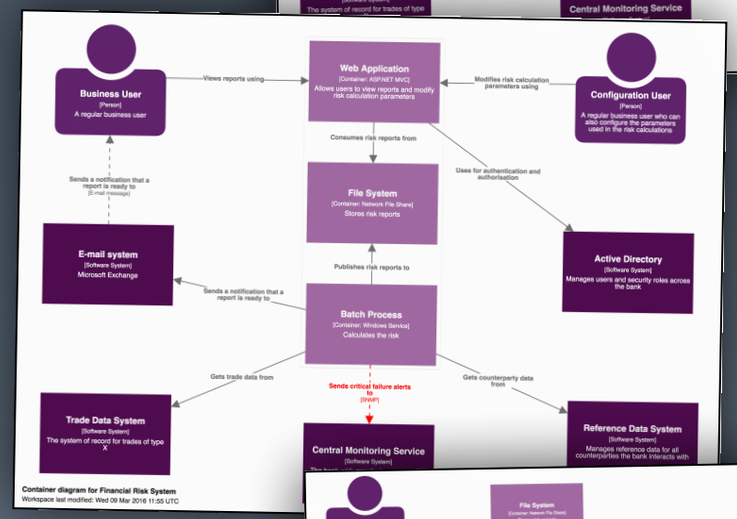


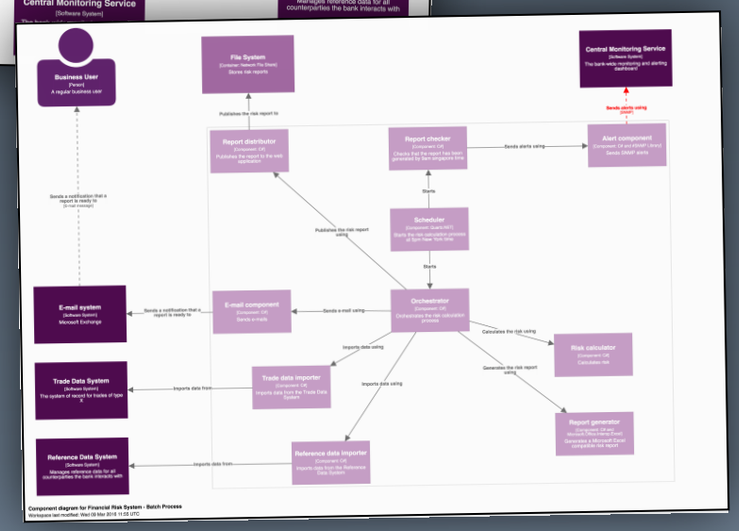
Classes (or Code)

Component implementation details









Diagrams are maps

that help a team navigate a complex codebase

Think about the

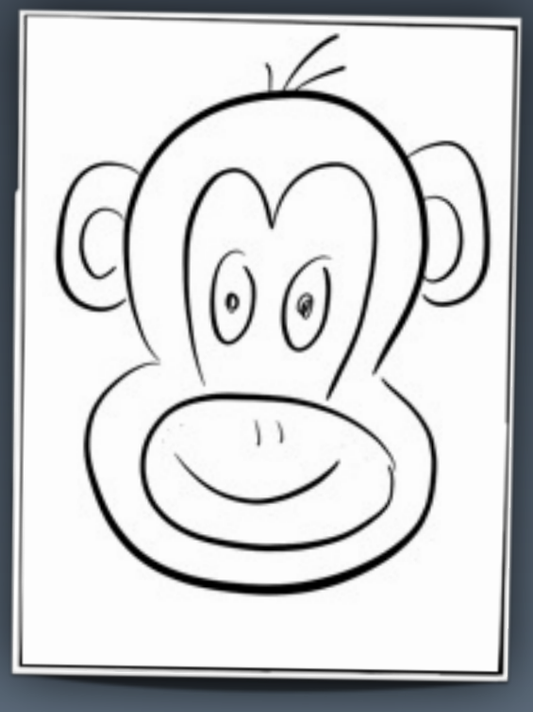
target audience

Non-technical Semi-technical Very technical

A common set of abstractions

is more important than
a common notation

techtribes.je

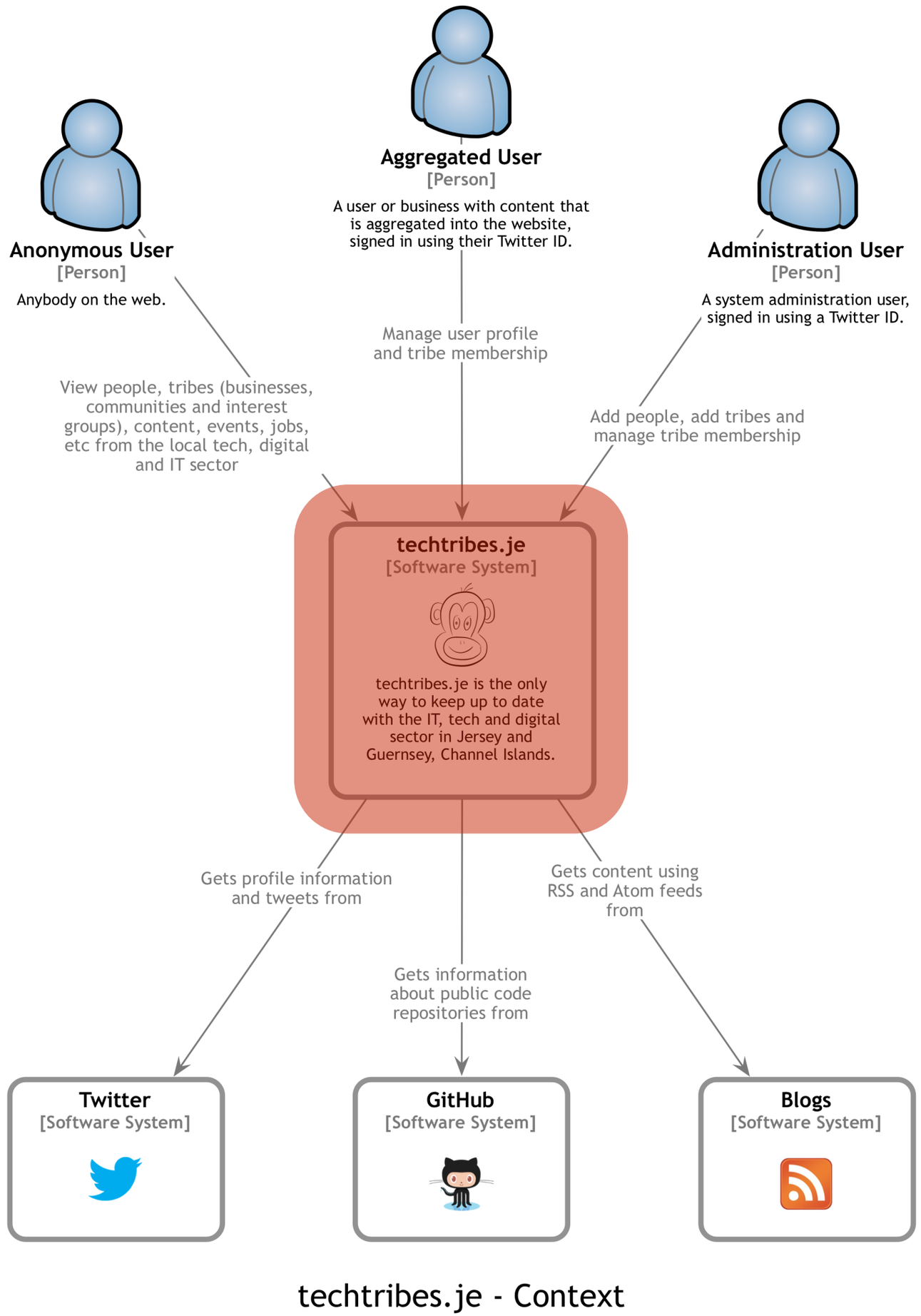


Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)



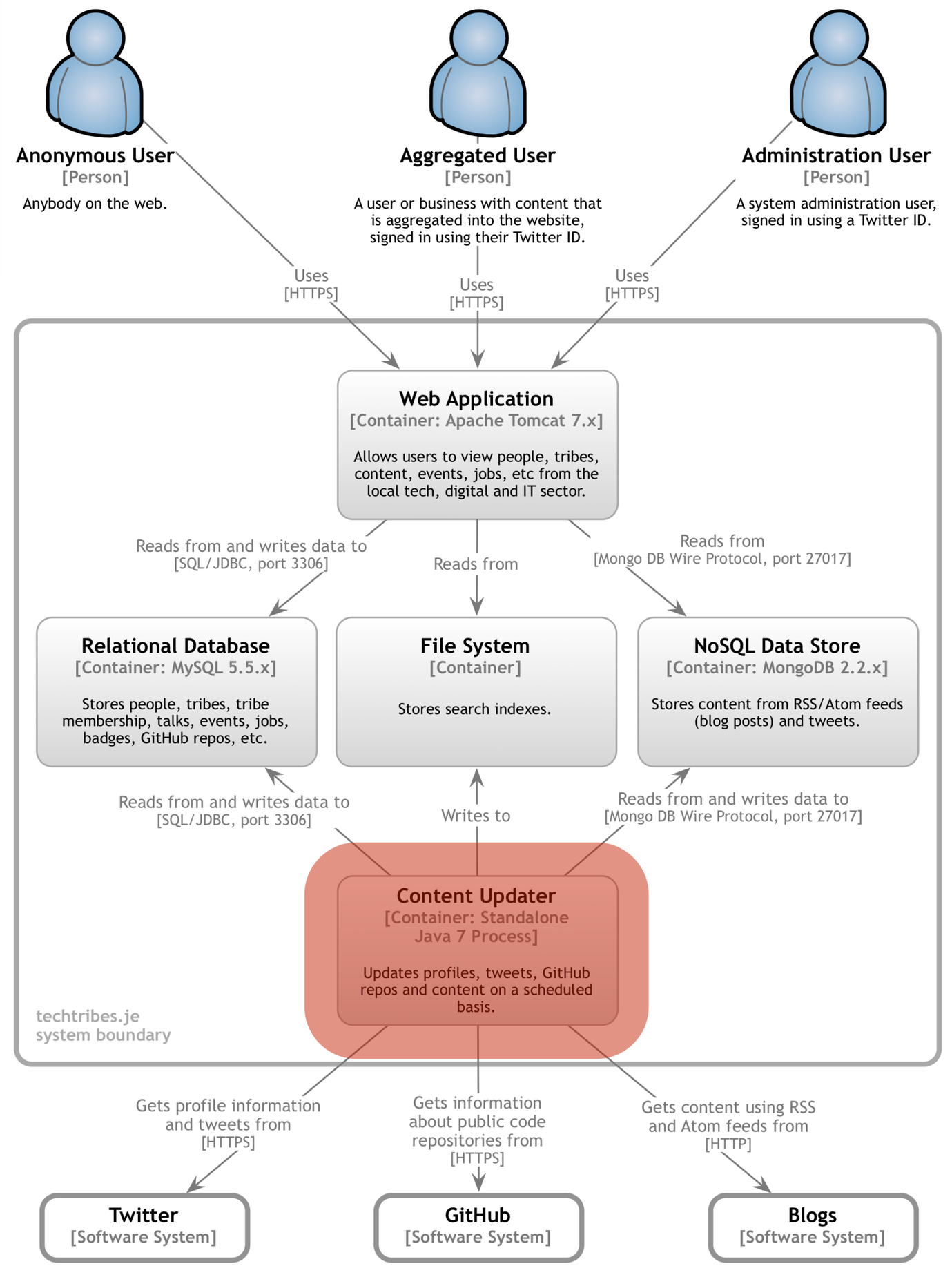
techtribes.je - Context

Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)



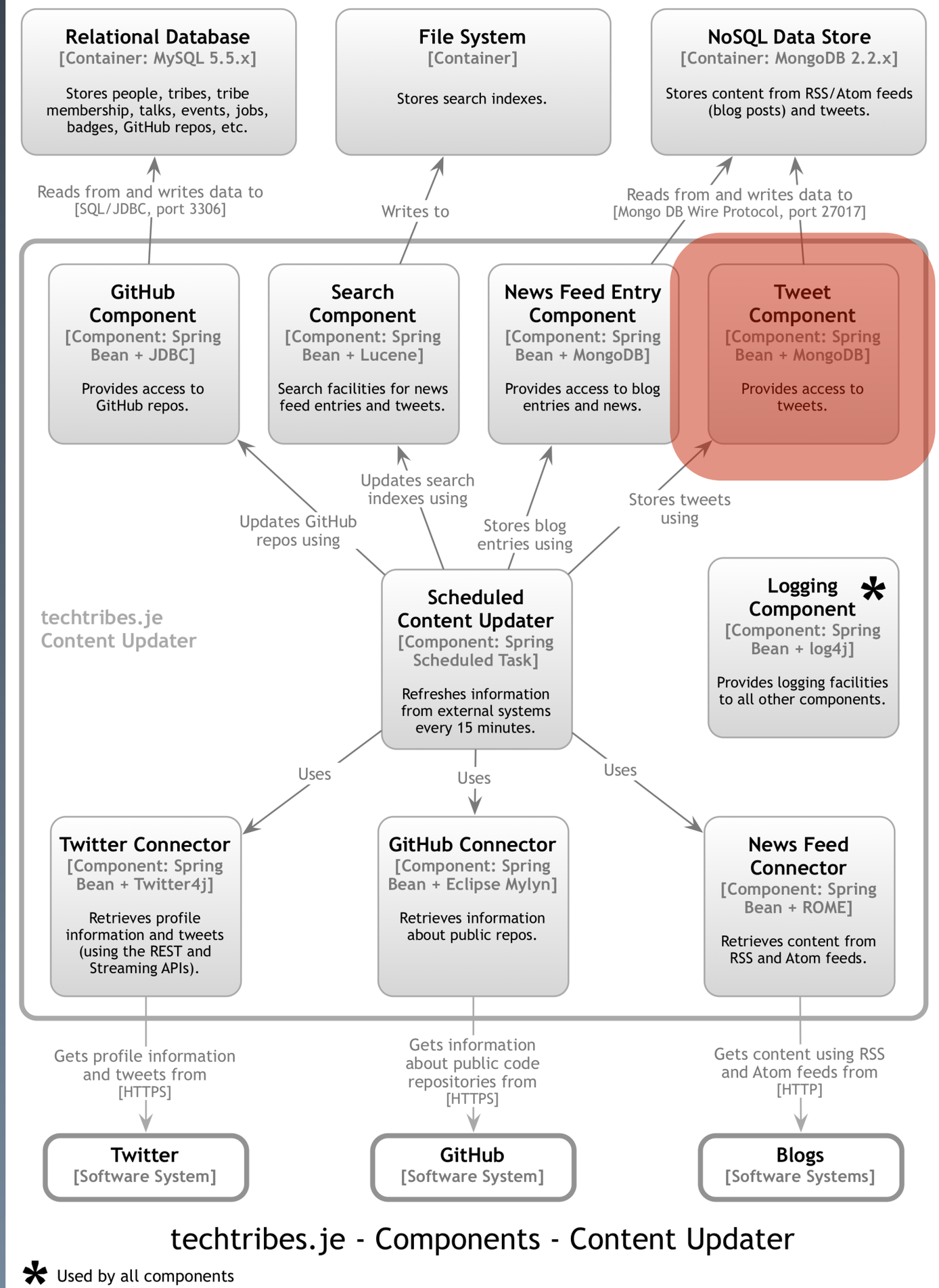
techtribes.je - Containers

Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)



techtribes.je - Components - Content Updater

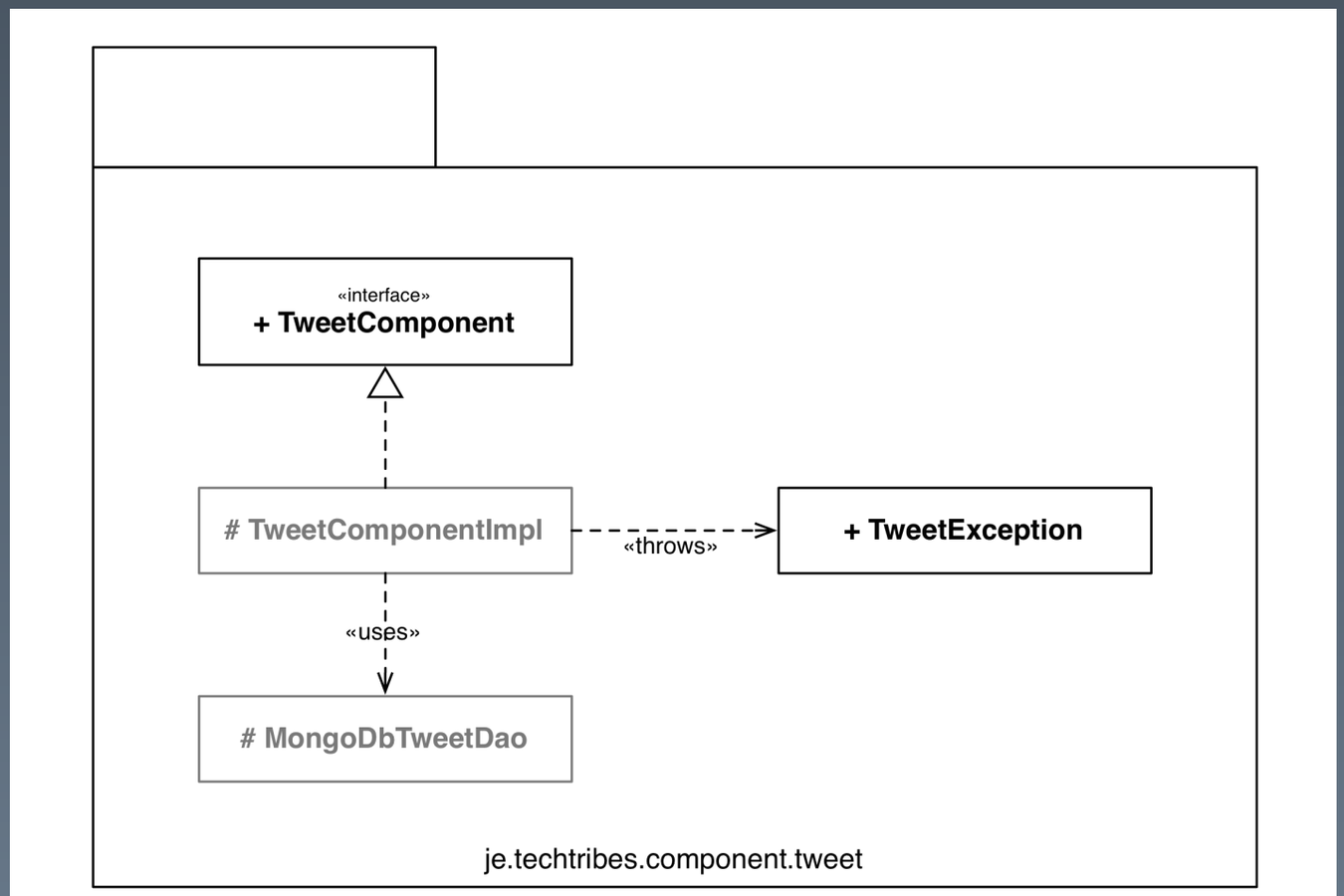
* Used by all components

Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)



A simple notation

(whiteboard and sticky note friendly, supplemented with colour coding)

Anonymous User

[Person]

Anybody on the web.

techtribes.je

[Software System]

techtribes.je is the only way to keep up to date with the IT, tech and digital sector in Jersey and Guernsey, Channel Islands.

Web Application

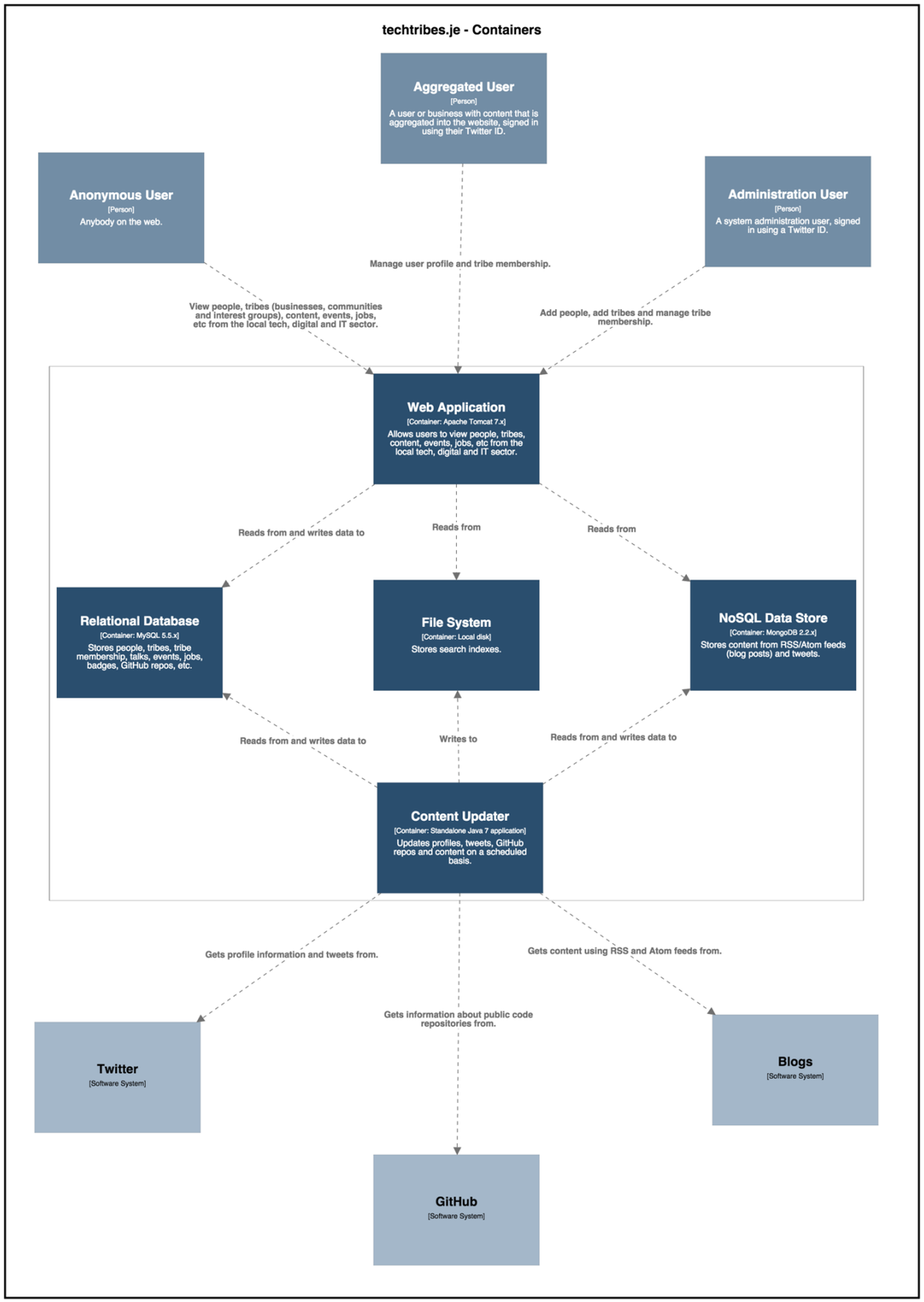
[Container: Apache Tomcat 7.x]

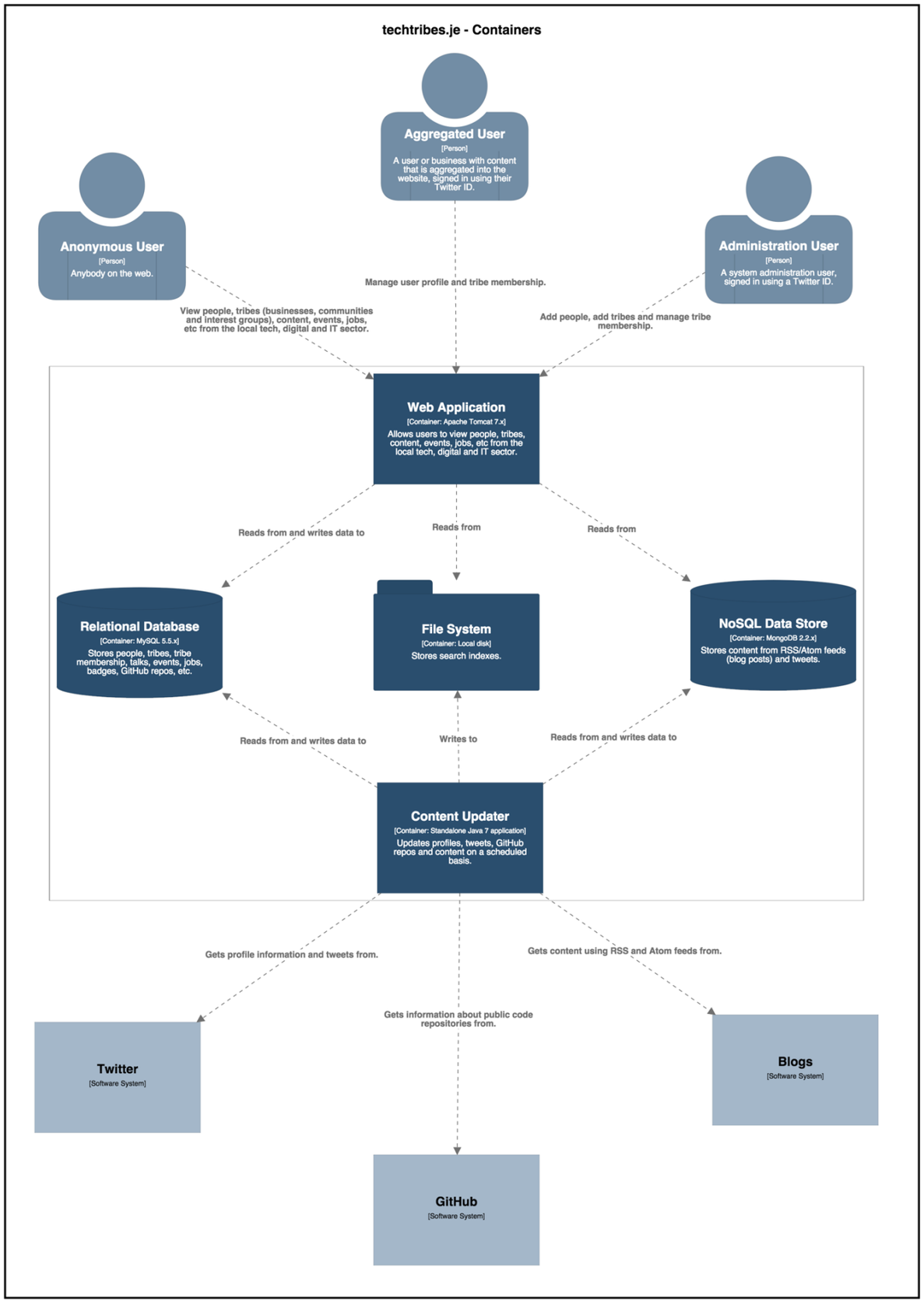
Allows users to view people, tribes, content, events, jobs, etc from the local tech, digital and IT sector.

Twitter Connector

[Component: Spring Bean + Twitter4j]

Retrieves profile information and tweets (using the REST and Streaming APIs).





Shapes and colour can add an additional layer of information

C4++

Enterprise context

User interface mockups and wireframes

Domain model

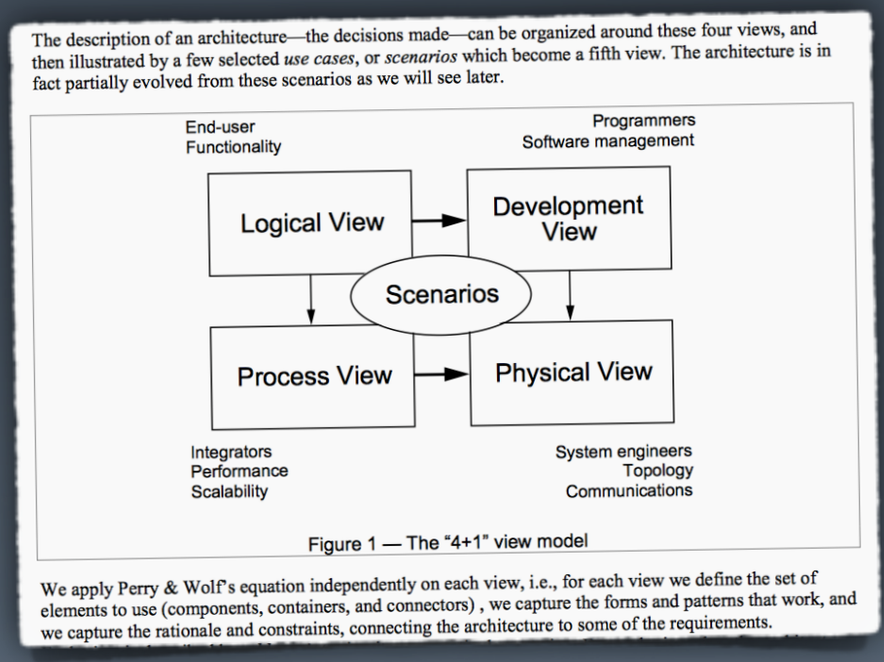
Sequence and collaboration diagrams

Business process and workflow models

Infrastructure model

Deployment model

...



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

We apply Perry & Wolf's equation independently on each view, i.e., for each view we define the set of elements to use (components, containers, and connectors) , we capture the forms and patterns that work, and we capture the rationale and constraints, connecting the architecture to some of the requirements.

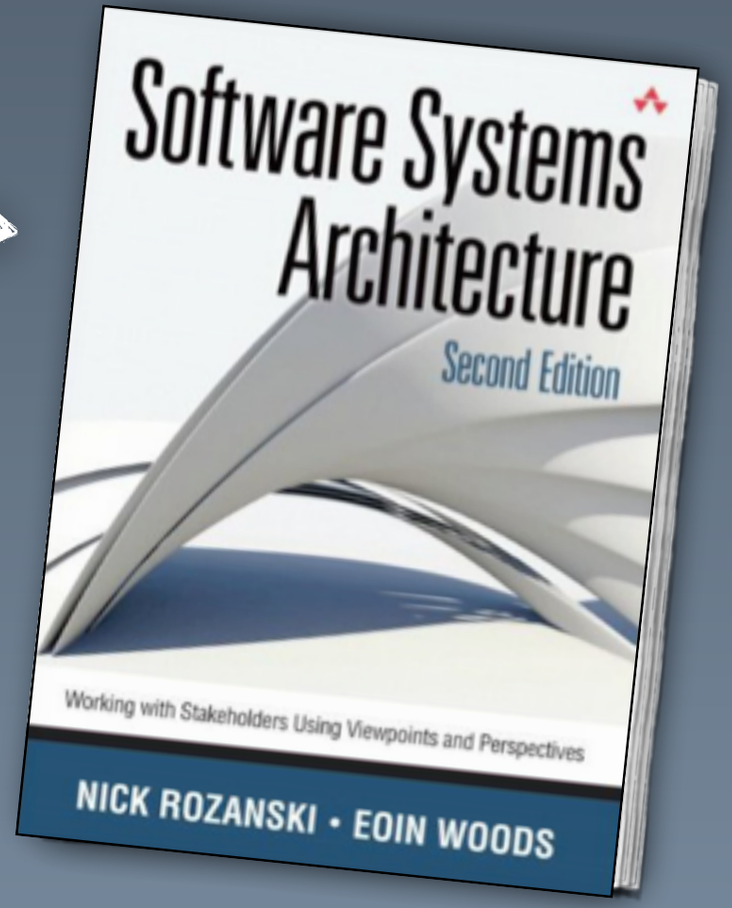
4+1 architectural view model

Philippe Kruchten

Software Systems Architecture

Working with Stakeholders Using Viewpoints and Perspectives (2nd Edition)

Nick Rozanski and Eoin Woods



C4 is **not** a design process

Up front design

vs

retrospectively drawing diagrams

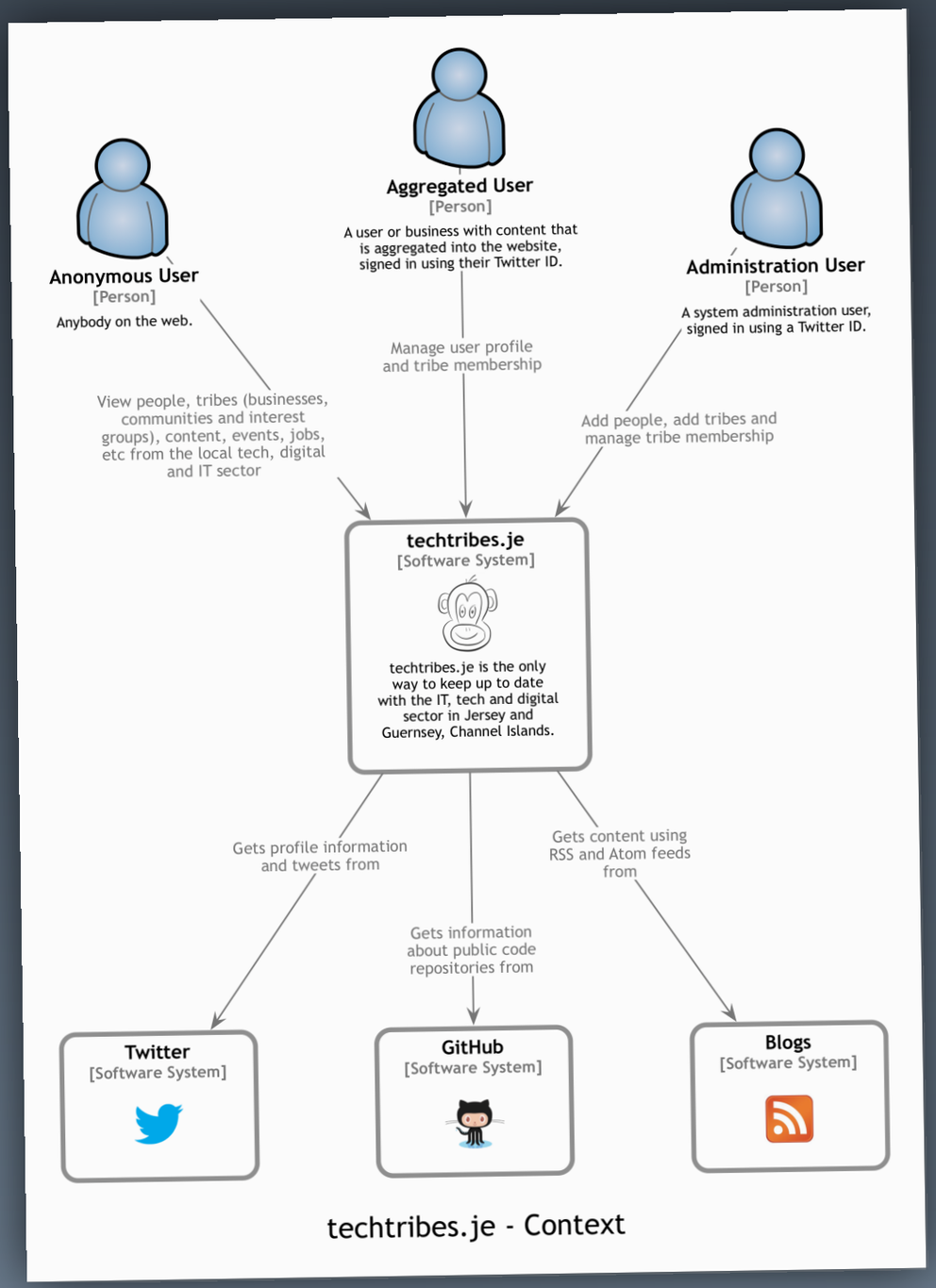
Tooling

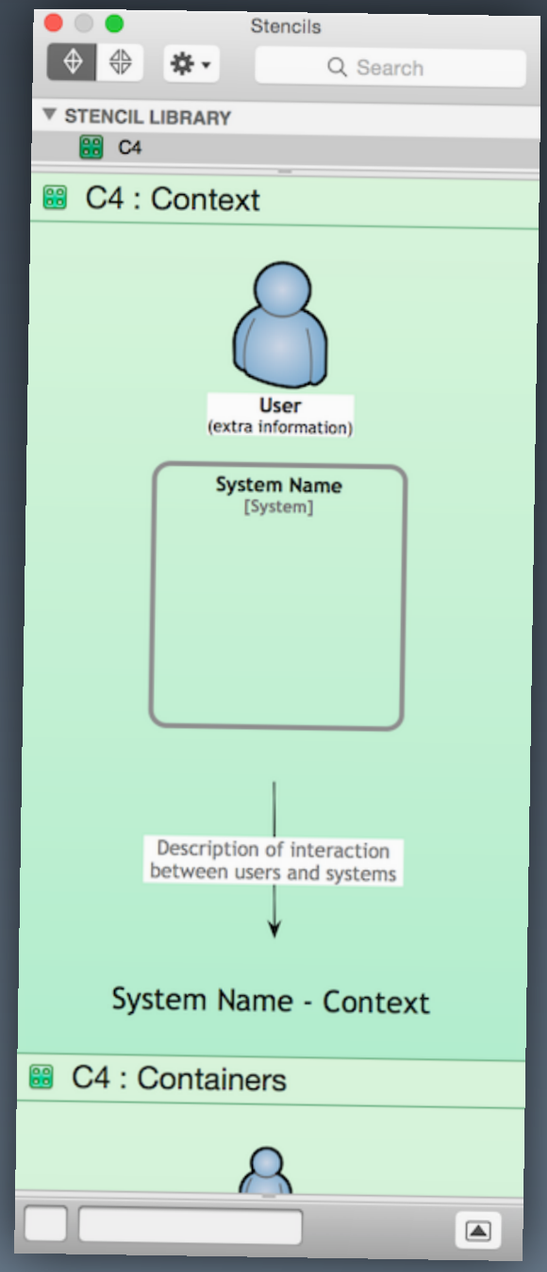
What's the easiest solution to create software architecture diagrams?

I don't want to bother with adhering to UML or finding the proper symbol for the right type of component. I might want a cloud, or 5 boxes in a larger box. Or some other shape. And I'd like a nice variety of arrows to choose from. Something to whip together a diagram fast, but equally important, it should look pretty. Something professional, that could go in a publication or on a company's web site. Certainly free is better, but I think I'm willing to pay once to not have to fight the whatever tool I use to document software anymore.

Any suggestions?

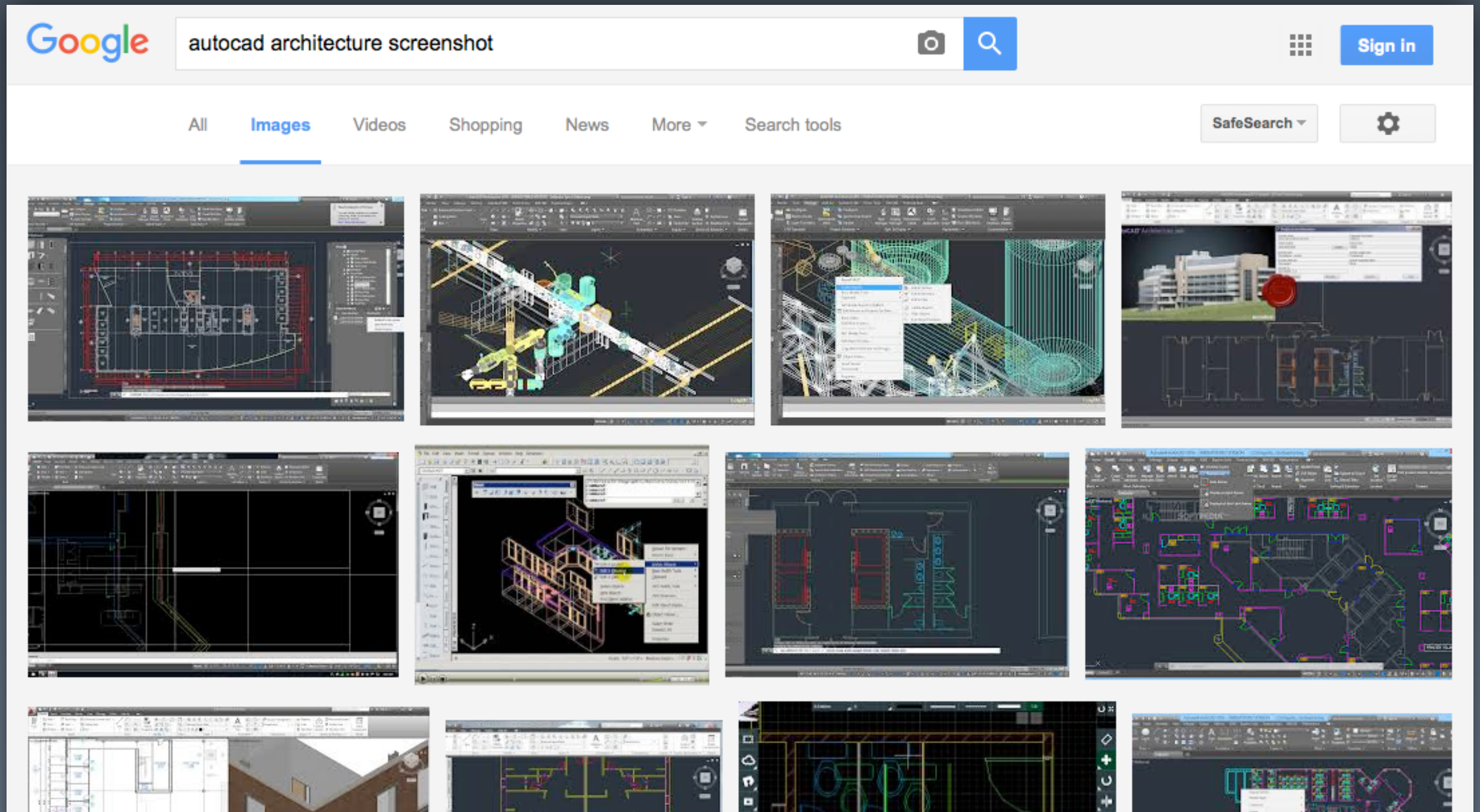
http://quora.com/Whats-the-easiest-solution-to-create-software-architecture-diagrams





Any ***general purpose diagramming tool*** can be used to create software architecture diagrams

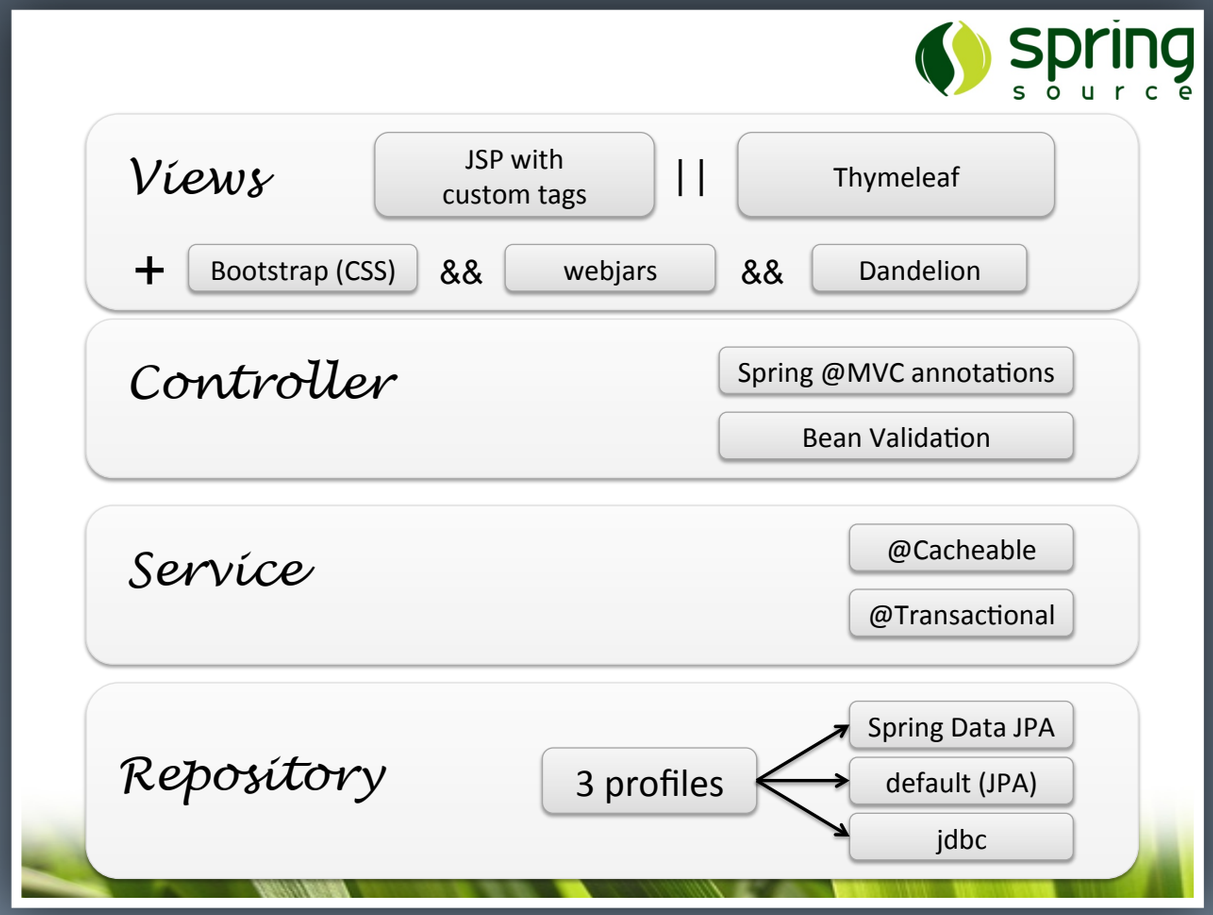
Do building architects use Microsoft Visio?



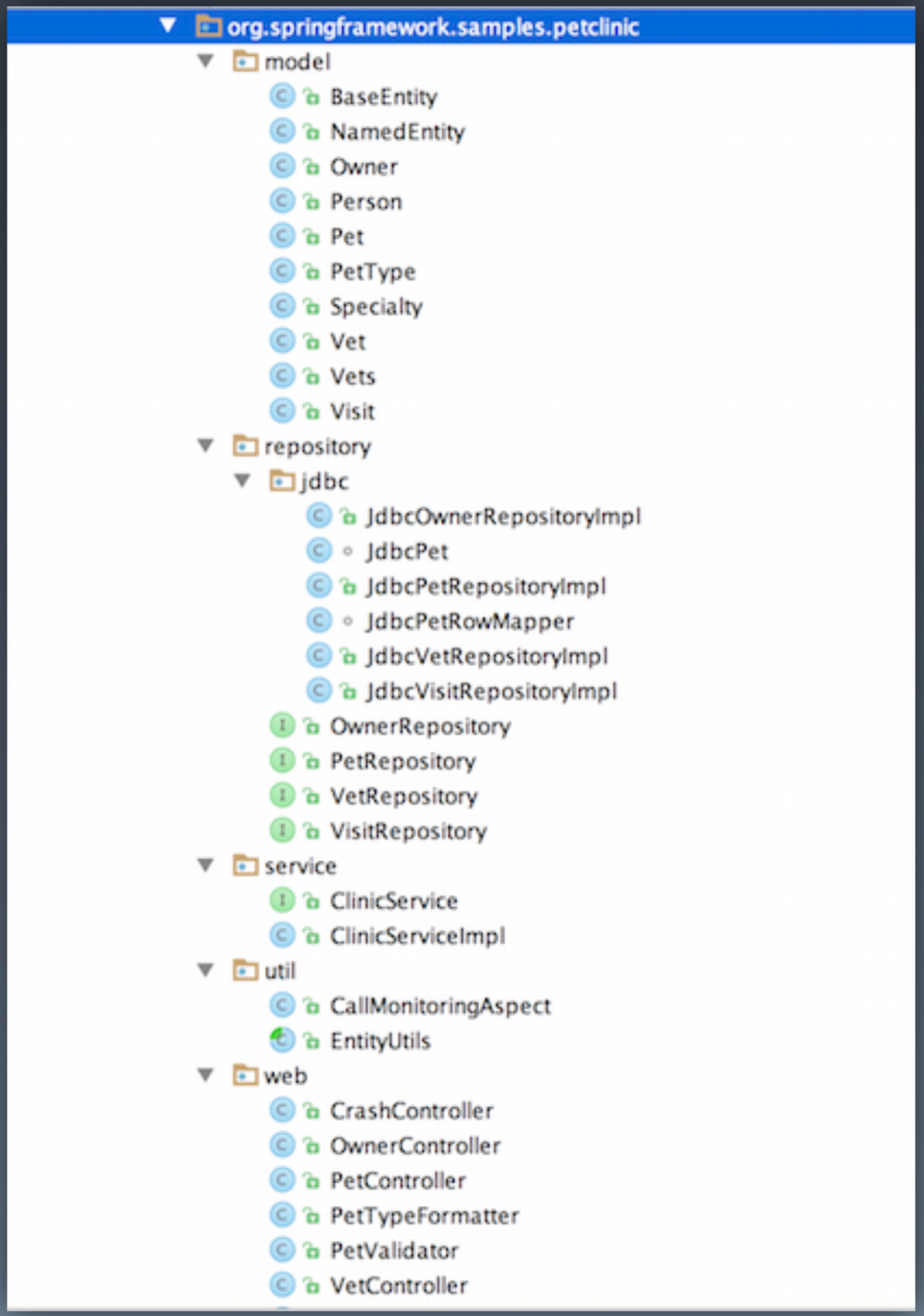
Sketches get out of date,
so why not
auto-generate
the diagrams?

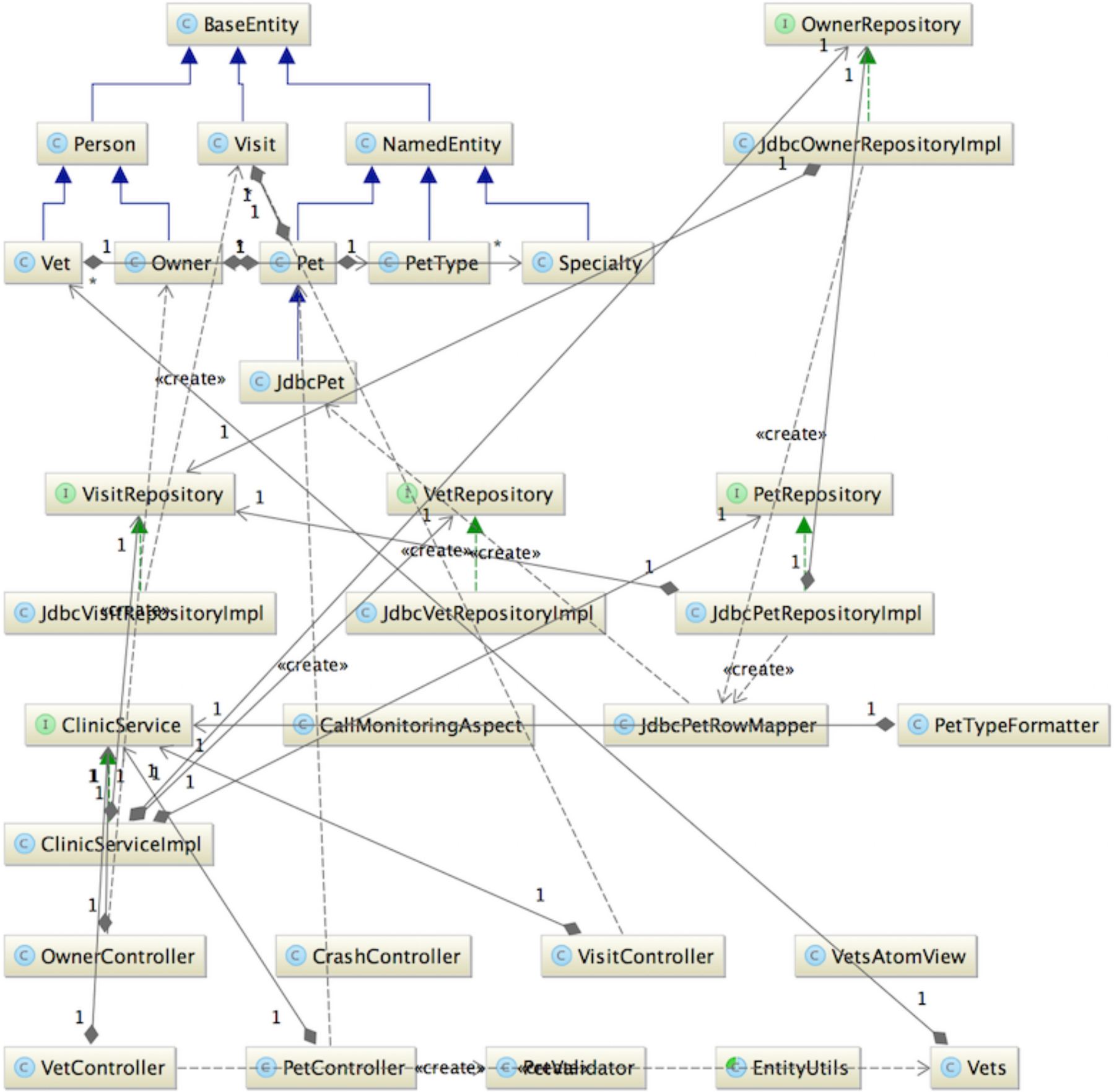
Spring PetClinic

https://github.com/spring-projects/spring-petclinic/



https://speakerdeck.com/michaelisvy/spring-petclinic-sample-application





An auto-generated UML class diagram

Software Reflexion Mo
Bridging the Gap between Source and

Gail C. Murphy and David Notkin

Dept. of Computer Science & Engineering
University of Washington
Box 352350
Seattle WA, USA 98195-2350
{gmurphy, notkin}@cs.washington.edu

Abstract

Software engineers often use high-level models (for instance, box and arrow sketches) to reason and communicate about an existing software system. One problem with high-level models is that they are almost always inaccurate with respect to the system's source code. We have developed an approach that helps an engineer use a high-level model of the structure of an existing software system as a lens through which to see a model of that system's source code. In particular, an engineer defines a high-level model and specifies how the model maps to the source. A tool then computes a software reflexion model that shows where the engineer's high-level model agrees with and where it differs from a model of the source.

The paper provides a formal characterization of reflexion models, discusses practical aspects of the approach, and relates experiences of applying the approach and tools to a number of different systems. The illustrative example used in the paper describes the application of reflexion models to NetBSD, an implementation of Unix comprised of 250,000 lines of C code. In only a few hours, an engineer computed several reflexion models that provided him with a useful, global overview of the structure of the NetBSD virtual memory subsystem. The approach has also been applied to aid in the understanding and experimental reengineering of the Microsoft Excel spreadsheet product.

*This research was funded in part by the NSF grant CCR-8858804 and a Canadian NSERC post-graduate scholarship.

SIGSOFT '95 Washington, D.C., USA
©1995 ACM 0-89791-716-2/95/0010...$3.50

1 Intr

Software e
software sy
Box and ar
are often f
though th
soning abo
can be dan
always ina
source.

Current
high-level
derived m
their very
source. Al
by these re
from the m
ple of this

We have
Figure 1,
sufficiently
ent way. T
of interest.
call graph
source cod
between th
model is th
engineer's
agree with
terprets th
modifies th
tional refle

[1]The old
flexion" from

1 Introduction

Software engineers often think about an existing software system in terms of high-level models. Box and arrow sketches of a system, for instance, are often found on engineers' whiteboards. Although these models are commonly used, reasoning about the system in terms of such models can be dangerous because the models are almost always inaccurate with respect to the system's source.

Current reverse engineering systems derive high-level models from the source code. These derived models are useful because they are, by their very nature, accurate representations of the source. Although accurate, the models created by these reverse engineering systems may differ from the models sketched by engineers; an exam-

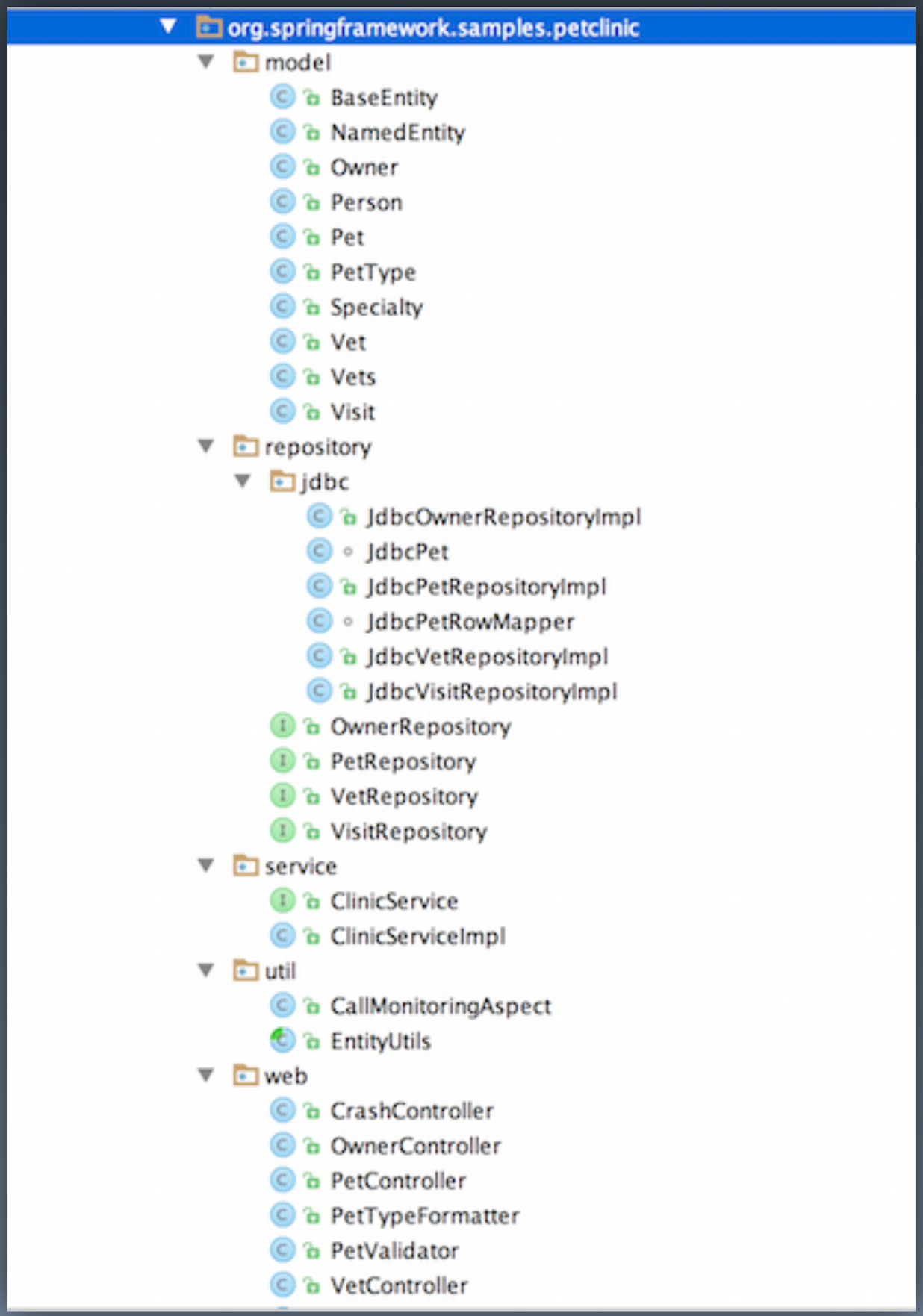
Diagramming tools see

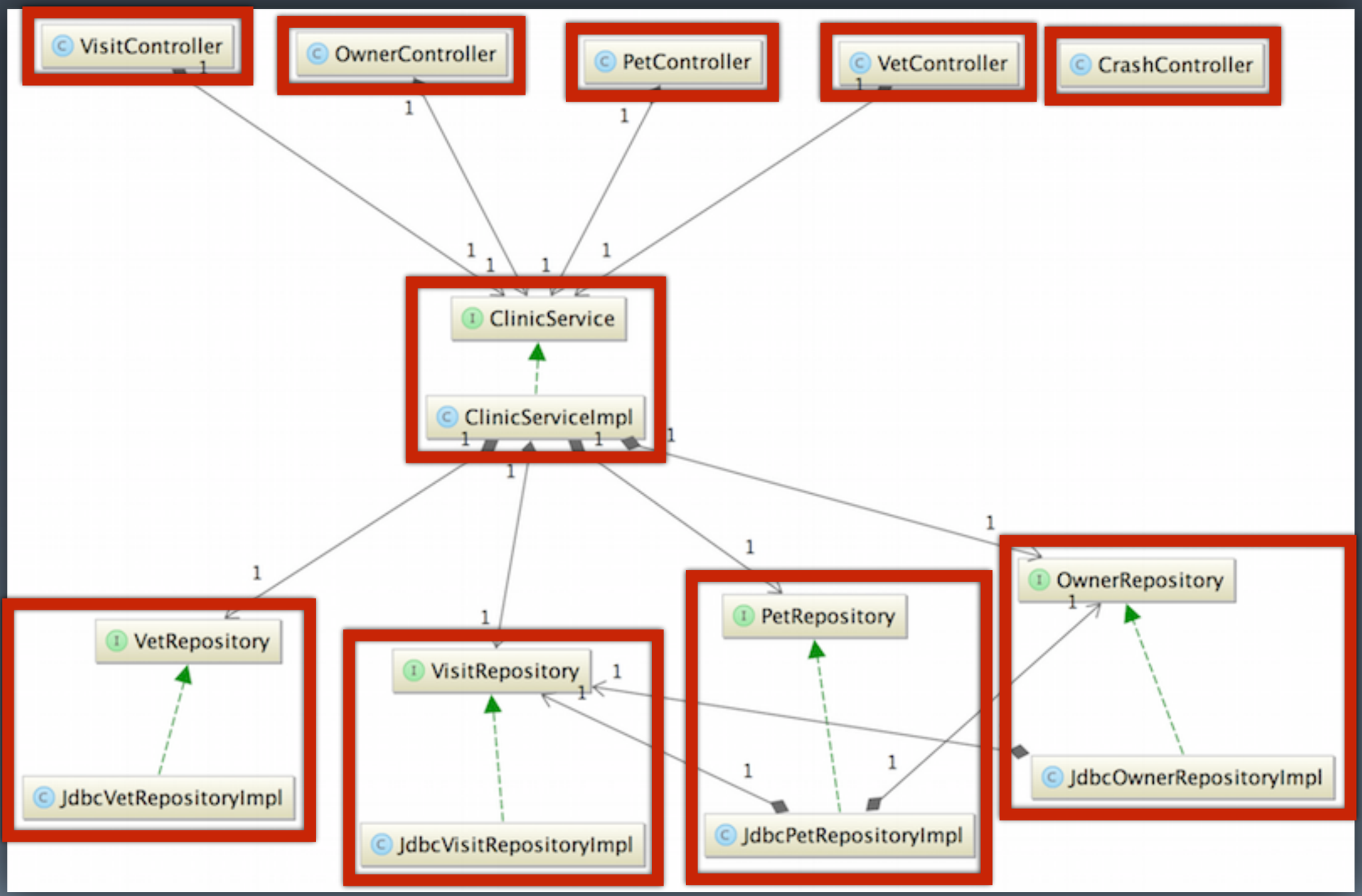
code

rather than components

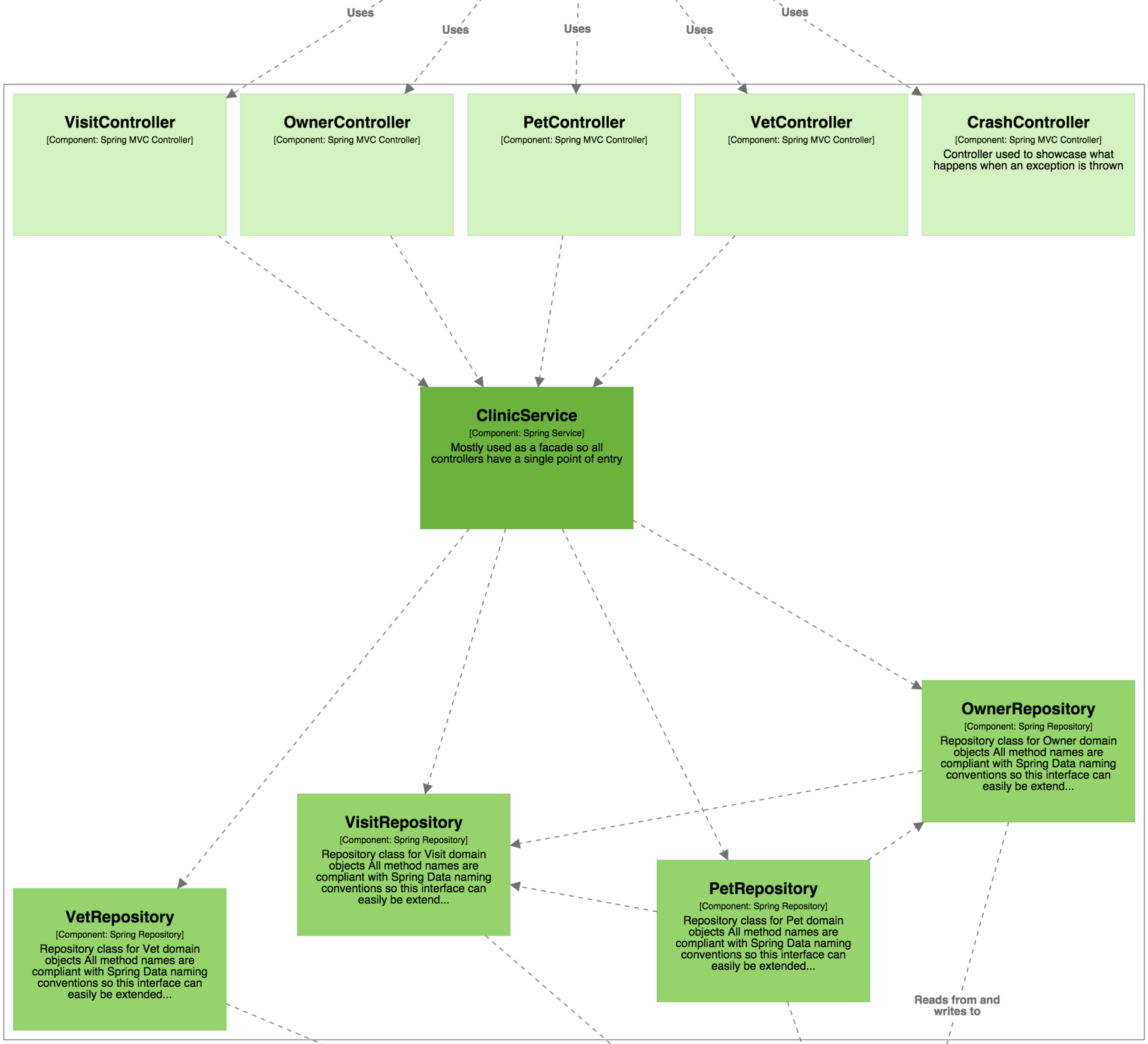
What is a “component”?

What are the architecturally significant elements?





A UML class diagram showing architecturally significant elements



A component diagram, based upon the code

The code is the

embodiment

of the architecture

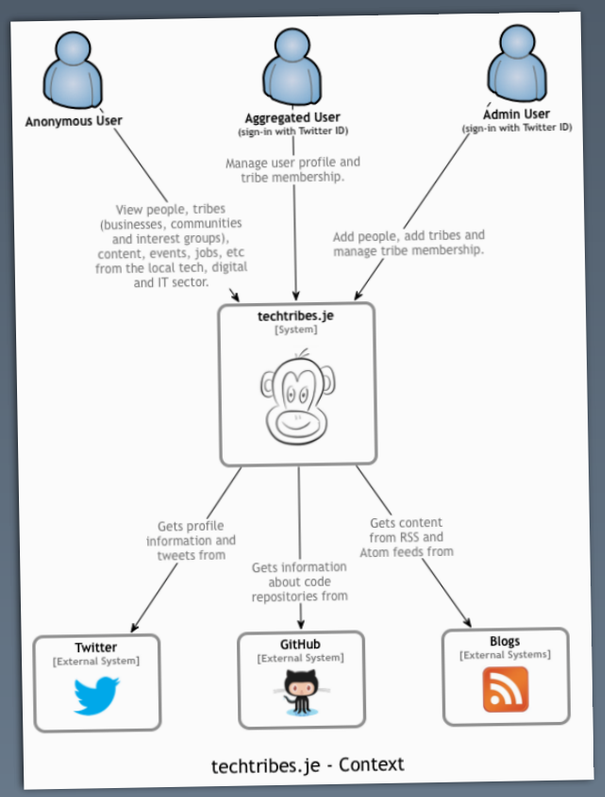
In practice, architecture is embodied and ***recoverable from code****, and many languages provide architecture-level views of the system.*

A Survey of Architecture Description Languages

by Paul C. Clements

Is the architecture *in* the code?

Context



People

Security groups/roles in configuration files, etc.

Software Systems

Integration points, APIs, known libraries, credentials for inbound consumers, etc.

Containers

IDE projects/modules, build output (code and infrastructure), etc.

Components

Extractable from the code if an architecturally-evident coding style has been adopted.

People

Security groups/roles in configuration files, etc.

Software Systems

Integration points, APIs, known libraries, credentials for inbound consumers, etc.

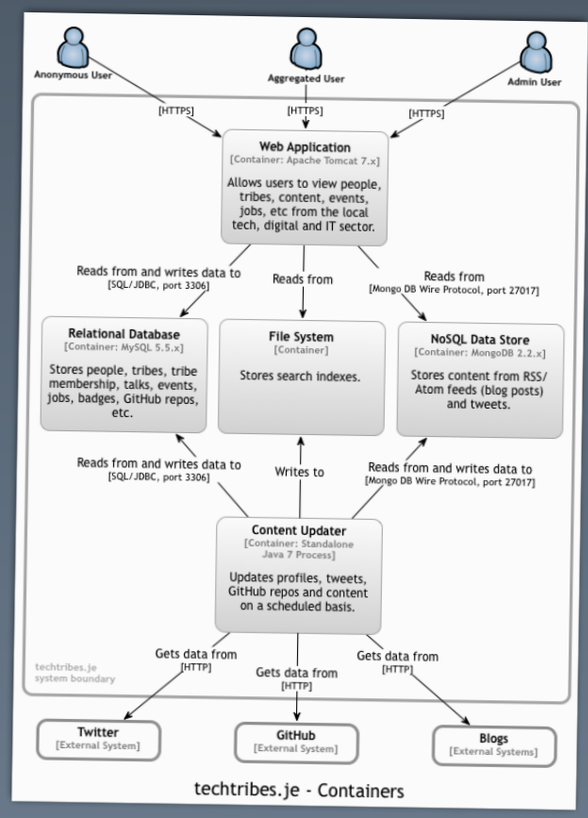
Containers

IDE projects/modules, build output (code and infrastructure), etc.

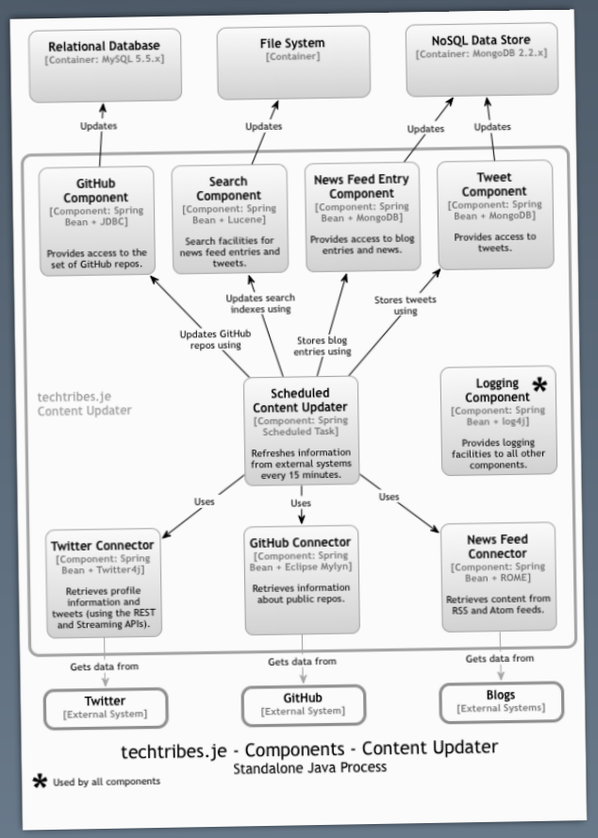
Components

Extractable from the code if an architecturally-evident coding style has been adopted.

Containers



Components



People

Security groups/roles in configuration files, etc.

Software Systems

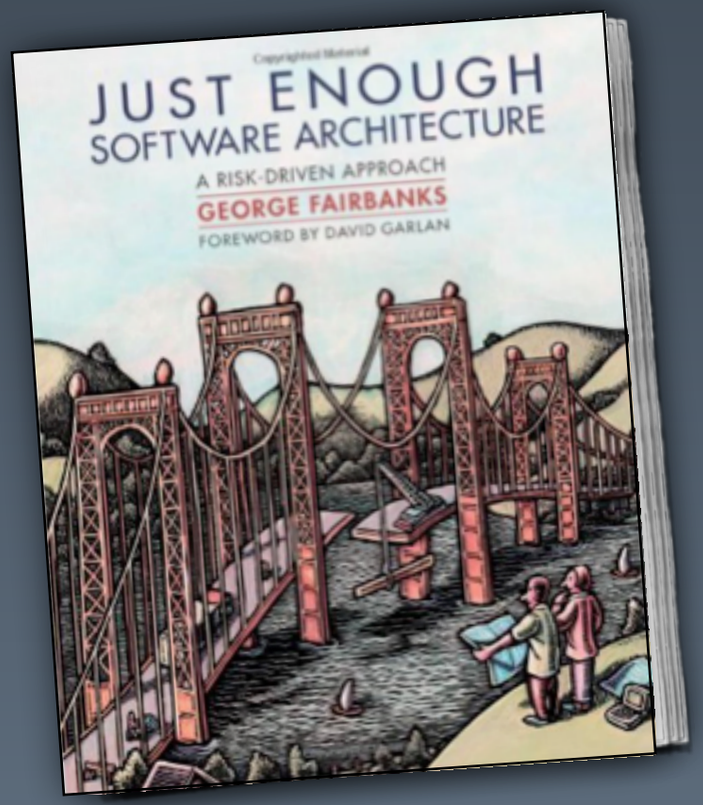
Integration points, APIs, known libraries, credentials for inbound consumers, etc.

Containers

IDE projects/modules, build output (code and infrastructure), etc.

Components

Extractable from the code if an architecturally-evident coding style has been adopted.



“architecturally-evident coding style”

Architecturally-evident coding styles include:

Annotations/attributes (`@Component`, `[Component]`, etc)

Naming conventions (`*Service`)

Namespacing/packaging

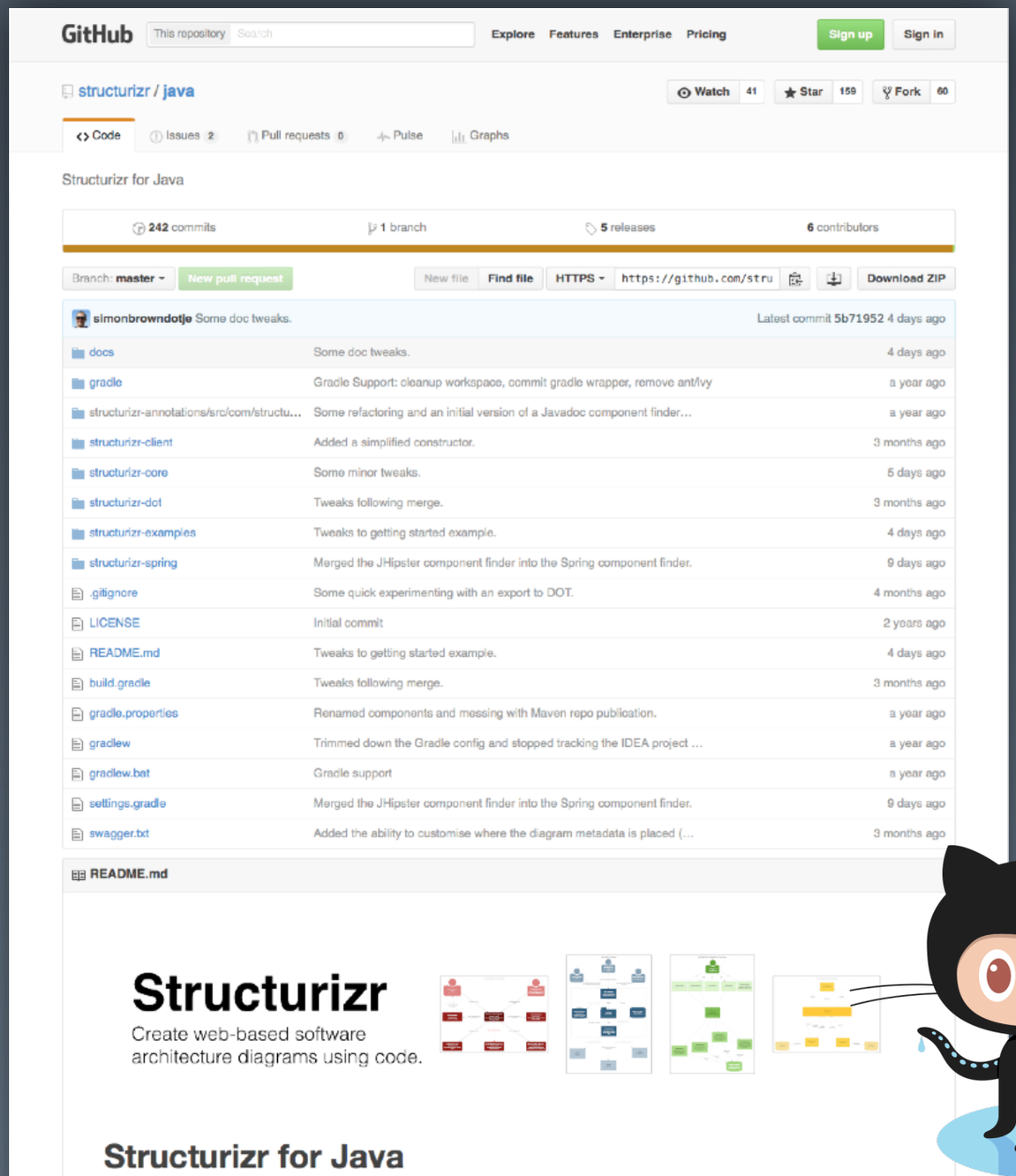
(`com.mycompany.system.components.*`)

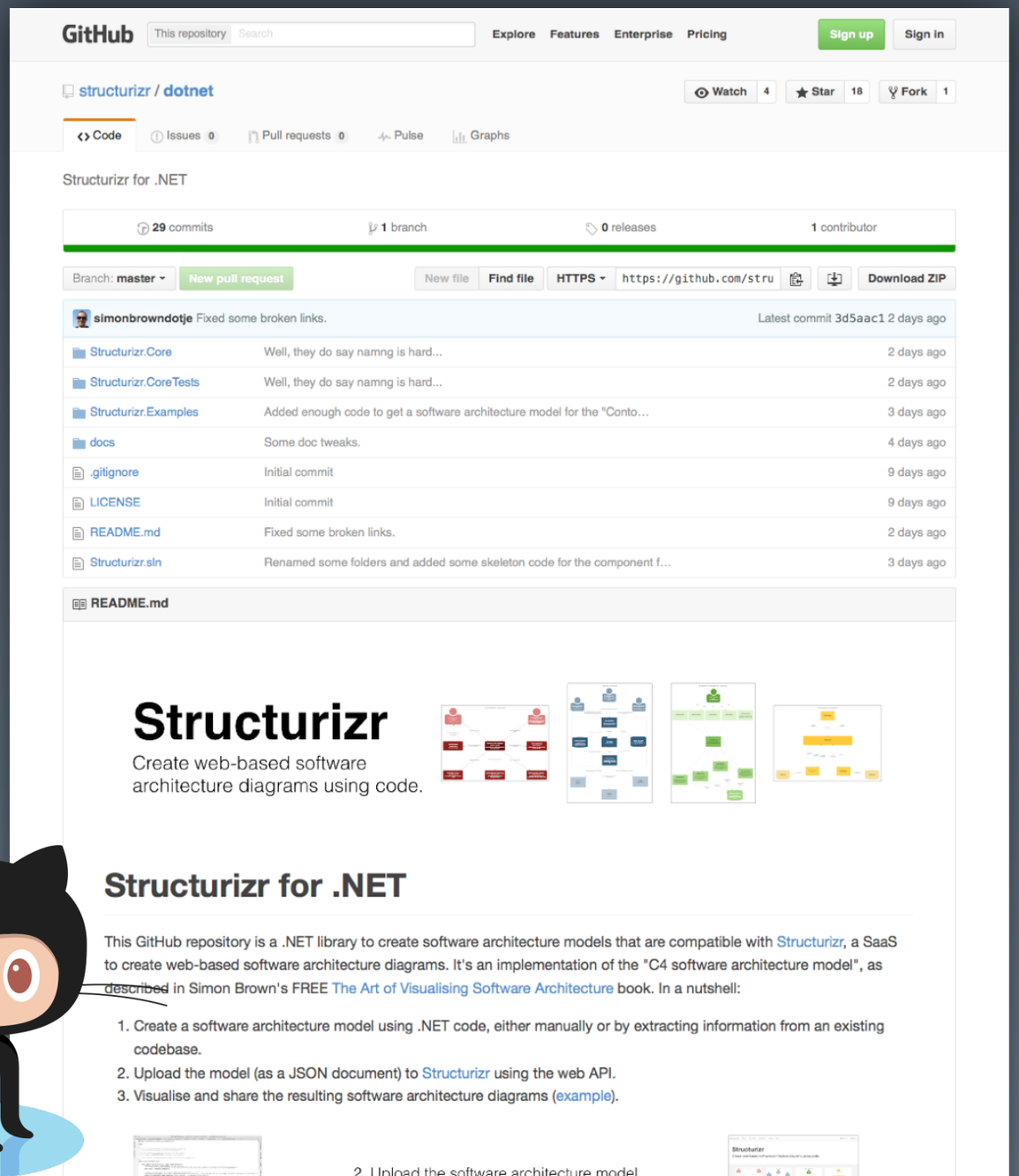
Maven modules, OSGi modules, Java 9 and Jigsaw, JavaScript module patterns, ECMAScript 6 modules, microservices, etc

Extract as much of the software architecture from the code as possible, and supplement where necessary

Create an architecture description language using code

Structurizr for Java
Structurizr for .NET

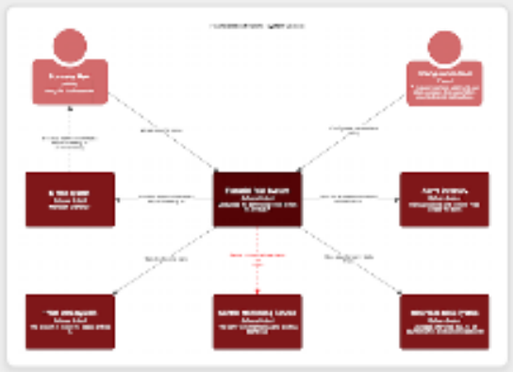


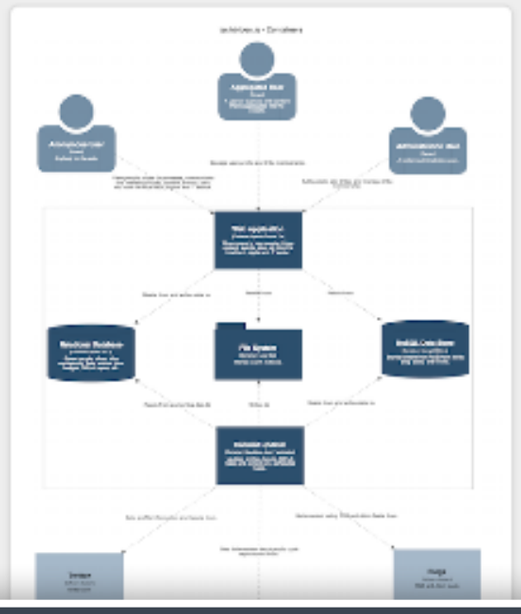


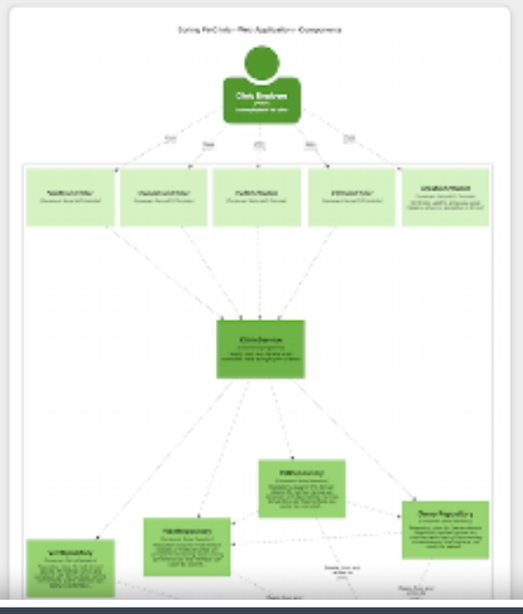
Structurizr

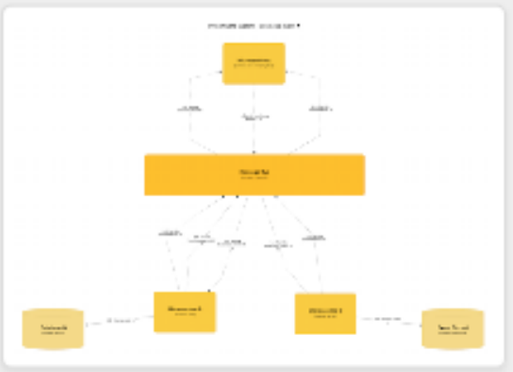
Create web-based software architecture diagrams using code.

Getting Started with Java **Getting started with .NET**





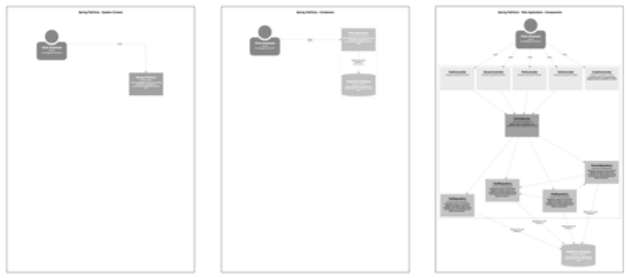




What is Structurizr?

Structurizr Client

2. Upload the software architecture model to Structurizr using the web API



1. Write one or more programs to create a software architecture model of your software system

Manually create model elements using code or extract them from your software system by parsing source code or using reflection and static analysis techniques

Software System

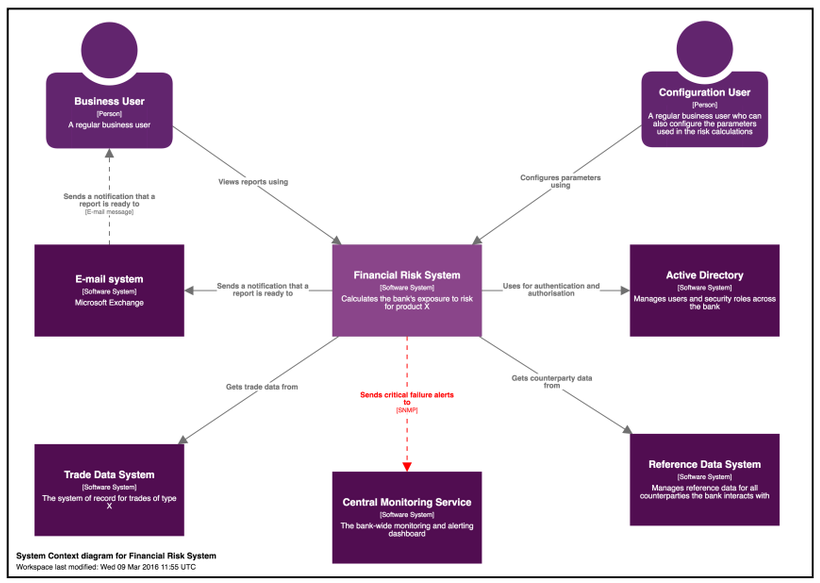
3. Explore, visualise and share your software architecture model with web-based software architecture diagrams

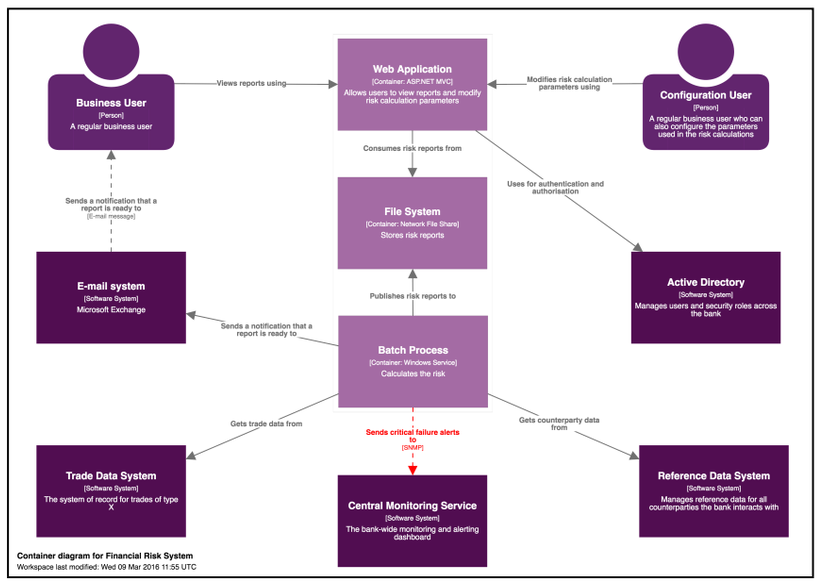
Structurizr is a web-based tool that creates diagrams for you, based upon the software architecture model and views defined in your code

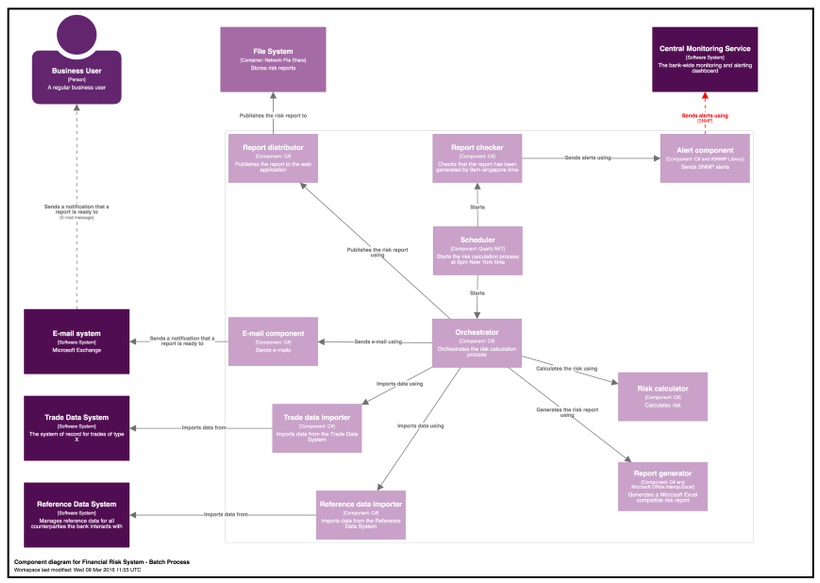
What is Structurizr?

```
class GettingStarted
{

    static void Main(string[] args)
    {
        Workspace workspace = new Workspace("My model", "This is a model of my software system.");
        Model.Model model = workspace.Model;

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

        ViewSet viewSet = workspace.Views;
        SystemContextView contextView = viewSet.CreateContextView(softwareSystem);
        contextView.AddAllSoftwareSystems();
        contextView.AddAllPeople();

        Styles styles = viewSet.Configuration.Styles;
        styles.Add(new ElementStyle(Tags.SoftwareSystem) { Background = "#a4b7c9", Color = "#000000" });
        styles.Add(new ElementStyle(Tags.Person) { Background = "#728da5", Color = "#ffffff" });

        StructurizrClient structurizrClient = new StructurizrClient("key", "secret");
        structurizrClient.PutWorkspace(1234, workspace);
    }

}
```

```
static void Main(string[] args)
{
    Workspace workspace = new Workspace("Financial Risk System", "A simple example C4 model based upon the financial risk system arc
    Model.Model model = workspace.Model;

    // create the basic model
    SoftwareSystem financialRiskSystem = model.AddSoftwareSystem(Location.Internal, "Financial Risk System", "Calculates the bank's

    Person businessUser = model.AddPerson(Location.Internal, "Business User", "A regular business user");
    businessUser.Uses(financialRiskSystem, "Views reports using");

    Person configurationUser = model.AddPerson(Location.Internal, "Configuration User", "A regular business user who can also config
    configurationUser.Uses(financialRiskSystem, "Configures parameters using");

    SoftwareSystem tradeDataSystem = model.AddSoftwareSystem(Location.Internal, "Trade Data System", "The system of record for trade
    financialRiskSystem.Uses(tradeDataSystem, "Gets trade data from");

    SoftwareSystem referenceDataSystem = model.AddSoftwareSystem(Location.Internal, "Reference Data System", "Manages reference data
    financialRiskSystem.Uses(referenceDataSystem, "Gets counterparty data from");

    SoftwareSystem emailSystem = model.AddSoftwareSystem(Location.Internal, "E-mail system", "Microsoft Exchange");
    financialRiskSystem.Uses(emailSystem, "Sends a notification that a report is ready to");
    emailSystem.Delivers(businessUser, "Sends a notification that a report is ready to", "E-mail message", InteractionStyle.Asynchro

    SoftwareSystem centralMonitoringService = model.AddSoftwareSystem(Location.Internal, "Central Monitoring Service", "The bank-wid
    financialRiskSystem.Uses(centralMonitoringService, "Sends critical failure alerts to", "SNMP", InteractionStyle.Asynchronous).Ad

    SoftwareSystem activeDirectory = model.AddSoftwareSystem(Location.Internal, "Active Directory", "Manages users and security role
```

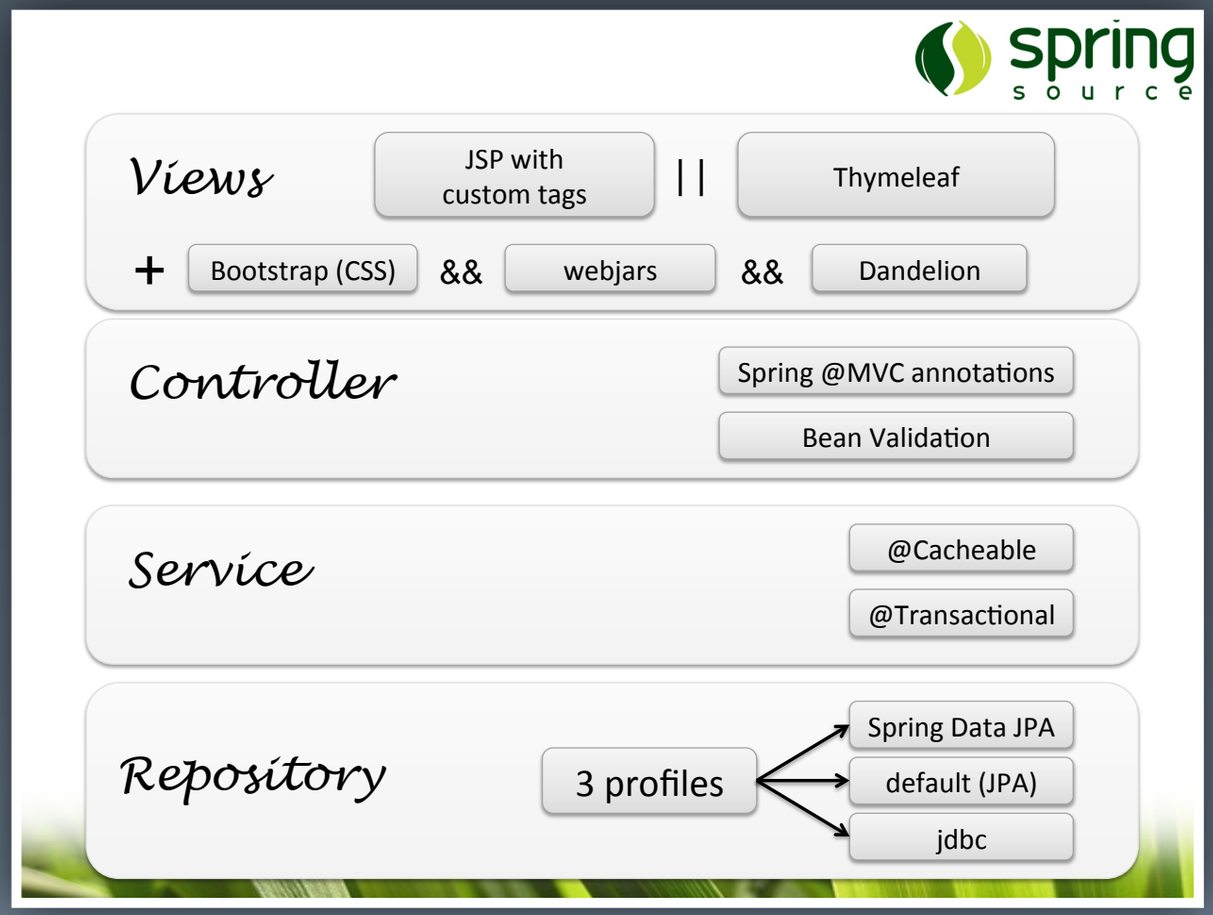




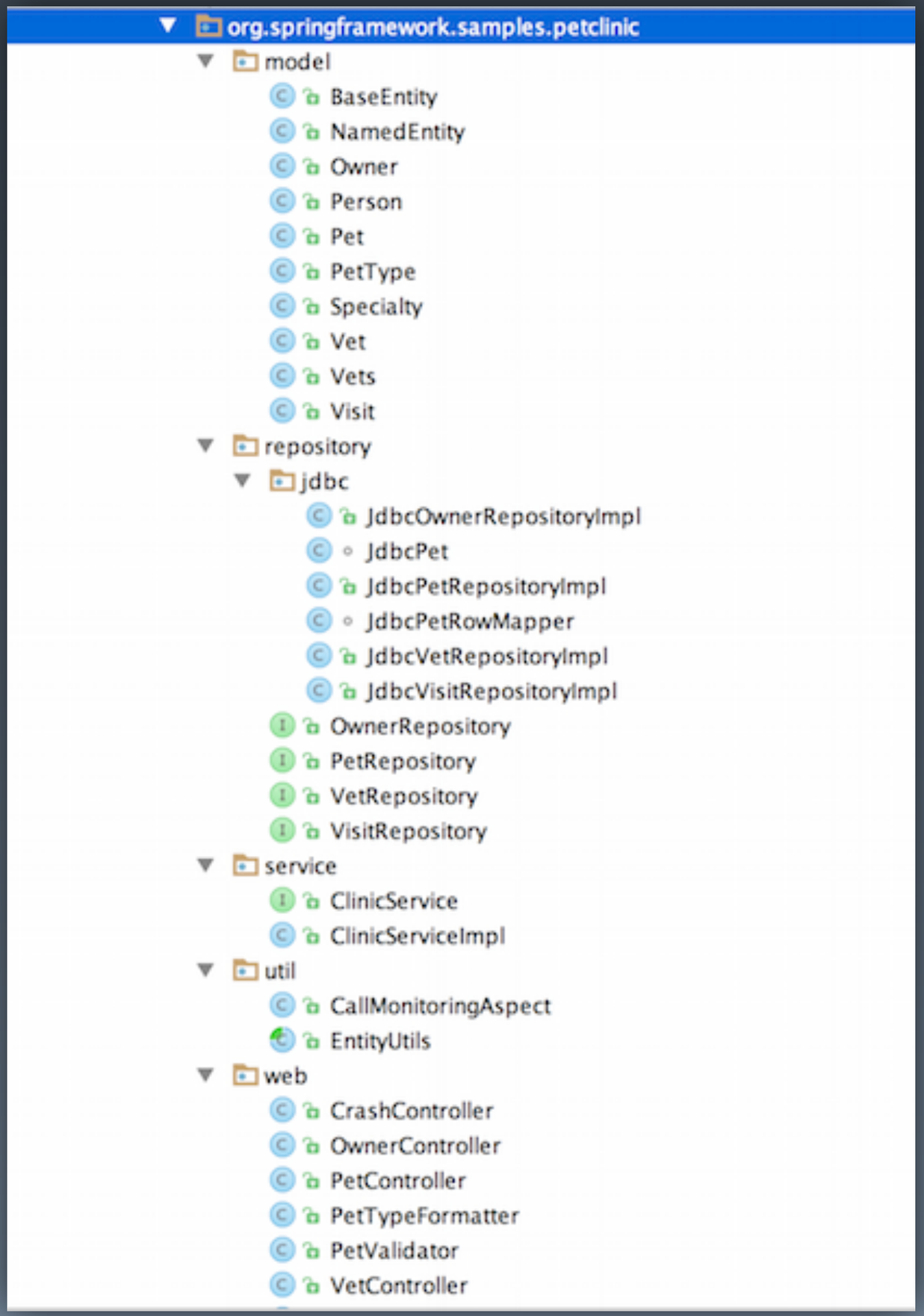


Spring PetClinic

https://github.com/spring-projects/spring-petclinic/



https://speakerdeck.com/michaelisvy/spring-petclinic-sample-application



```java
/**
 * This is a C4 representation of the Spring PetClinic sample app
 * (https://github.com/spring-projects/spring-petclinic/).
 *
 * Use the examples/springpetclinic.sh file to run this example -
 * you'll need a compiled version of the app on the CLASSPATH.
 */
public class SpringPetClinic {

    public static void main(String[] args) throws Exception {
        Workspace workspace = new Workspace("Spring PetClinic",
                "This is a C4 representation of the Spring PetClinic sample app (https://github.com/spring-pro
        Model model = workspace.getModel();

        // create the basic model (the stuff we can't get from the code)
        SoftwareSystem springPetClinic = model.addSoftwareSystem("Spring PetClinic", "Allows employees to view
        Person user = model.addPerson("Clinic Employee", "An employee of the clinic");
        user.uses(springPetClinic, "Uses");

        Container webApplication = springPetClinic.addContainer(
                "Web Application", "Allows employees to view and manage information regarding the veterinarians
        Container relationalDatabase = springPetClinic.addContainer(
                "Relational Database", "Stores information regarding the veterinarians, the clients, and their
        user.uses(webApplication, "Uses", "HTTP");
        webApplication.uses(relationalDatabase, "Reads from and writes to", "JDBC, port 9001");

        // and now automatically find all Spring @Controller, @Component, @Service and @Repository components
        ComponentFinder componentFinder = new ComponentFinder(
                webApplication, "org.springframework.samples.petclinic",
                new SpringComponentFinderStrategy(),
                new JavadocComponentFinderStrategy(new File("/Users/simon/Documents/sandbox/spring/spring-petc
        componentFinder.findComponents();

        // connect the user to all of the Spring MVC controllers
        webApplication.getComponents().stream()
                .filter(c -> c.getTechnology().equals("Spring MVC Controller"))
                .forEach(c -> user.uses(c, "Uses"));

        // connect all of the repository components to the relational database
```

```java
/**
 * This is a C4 representation of the Spring PetClinic sample app
 * (https://github.com/spring-projects/spring-petclinic/).
 *
 * Use the examples/springpetclinic.sh file to run this example -
 * you'll need a compiled version of the app on the CLASSPATH.
 */
public class SpringPetClinic {

    public static void main(String[] args) throws Exception {
        Workspace workspace = new Workspace("Spring PetClinic",
                "This is a C4 representation of the Spring PetClinic sample app (https://github.com/spring-pro
        Model model = workspace.getModel();

        // create the basic model (the stuff we can't get from the code)
        SoftwareSystem springPetClinic = model.addSoftwareSystem("Spring PetClinic", "Allows employees to view
        Person user = model.addPerson("Clinic Employee", "An employee of the clinic");
        user.uses(springPetClinic, "Uses");

        Container webApplication = springPetClinic.addContainer(
                "Web Application", "Allows employees to view and manage information regarding the veterinarians
        Container relationalDatabase = springPetClinic.addContainer(
                "Relational Database", "Stores information regarding the veterinarians, the clients, and their
        user.uses(webApplication, "Uses", "HTTP");
        webApplication.uses(relationalDatabase, "Reads from and writes to", "JDBC, port 9001");

        // and now automatically find all Spring @Controller, @Component, @Service and @Repository components
        ComponentFinder componentFinder = new ComponentFinder(
                webApplication, "org.springframework.samples.petclinic",
                new SpringComponentFinderStrategy(),
                new JavadocComponentFinderStrategy(new File("/Users/simon/Documents/sandbox/spring/spring-petc
        componentFinder.findComponents();

        // connect the user to all of the Spring MVC controllers
        webApplication.getComponents().stream()
                .filter(c -> c.getTechnology().equals("Spring MVC Controller"))
                .forEach(c -> user.uses(c, "Uses"));

        // connect all of the repository components to the relational database
```

“Component Finder”
with pluggable strategies, implemented using reflection & static analysis

(e.g. Java Annotations, .NET Attributes,
type name ends with “`Controller`”,
type extends class `X`, type implements interface `Y`,
supplement model with type-level comments
from source code, etc)

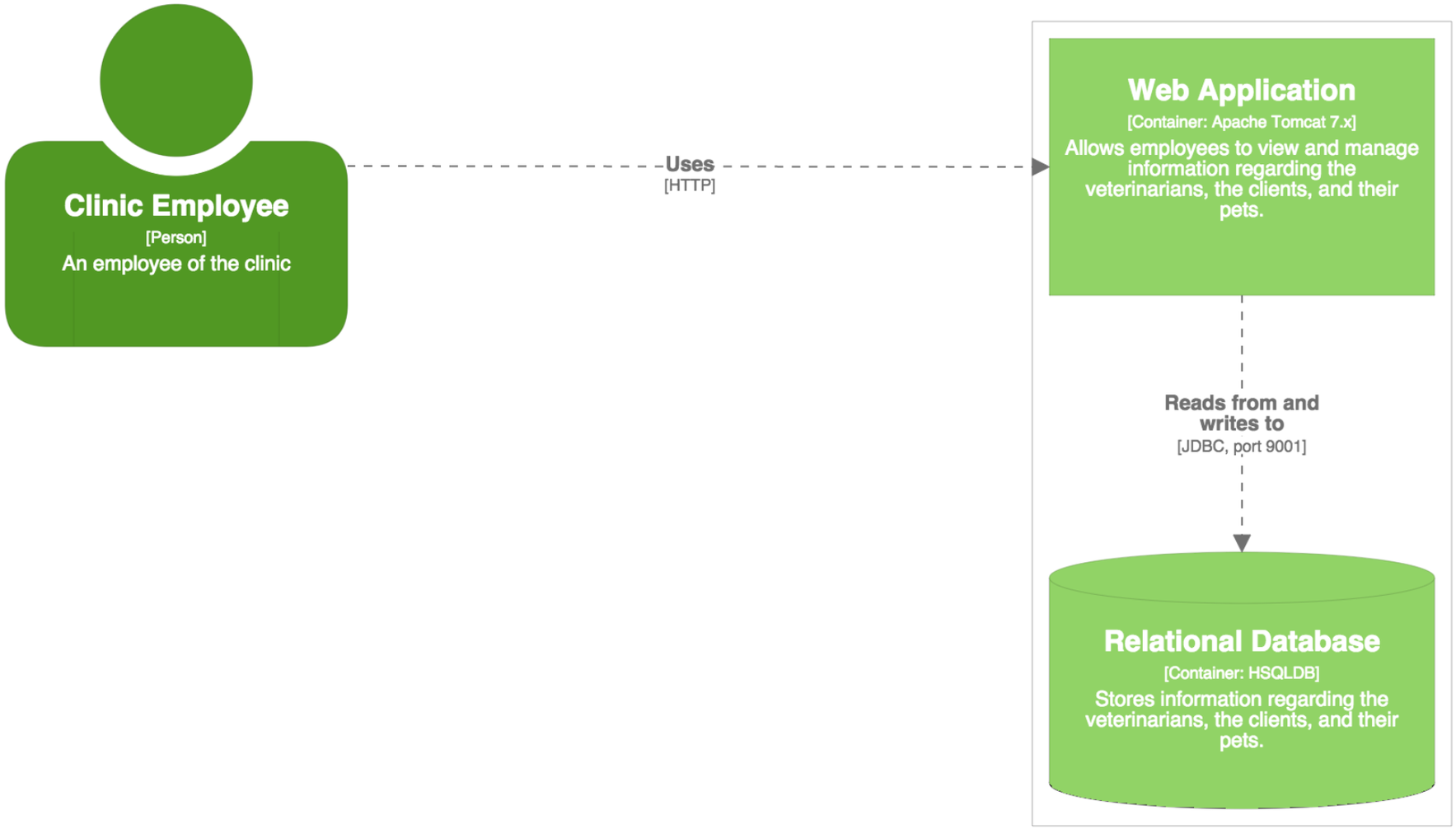
```
// and now automatically find all Spring @Controller, @Component, @Service and @Repository components
ComponentFinder componentFinder = new ComponentFinder(
        webApplication, "org.springframework.samples.petclinic",
        new SpringComponentFinderStrategy(),
        new JavadocComponentFinderStrategy(new File("/Users/simon/Documents/sandbox/spring/spring-petc
componentFinder.findComponents();

// connect the user to all of the Spring MVC controllers
webApplication.getComponents().stream()
        .filter(c -> c.getTechnology().equals("Spring MVC Controller"))
        .forEach(c -> user.uses(c, "Uses"));

// connect all of the repository components to the relational database
webApplication.getComponents().stream()
        .filter(c -> c.getTechnology().equals("Spring Repository"))
        .forEach(c -> c.uses(relationalDatabase, "Reads from and writes to"));

// finally create some views
ViewSet viewSet = workspace.getViews();
SystemContextView contextView = viewSet.createContextView(springPetClinic);
contextView.addAllSoftwareSystems();
contextView.addAllPeople();

ContainerView containerView = viewSet.createContainerView(springPetClinic);
containerView.addAllPeople();
containerView.addAllSoftwareSystems();
containerView.addAllContainers();

ComponentView componentView = viewSet.createComponentView(webApplication);
componentView.addAllComponents();
componentView.addAllPeople();
componentView.add(relationalDatabase);

// link the architecture model with the code
for (Component component : webApplication.getComponents()) {
    if (component.getSourcePath() != null) {
        component.setSourcePath(component.getSourcePath().replace(
                "/Users/simon/Documents/sandbox/spring/spring-petclinic/",
```

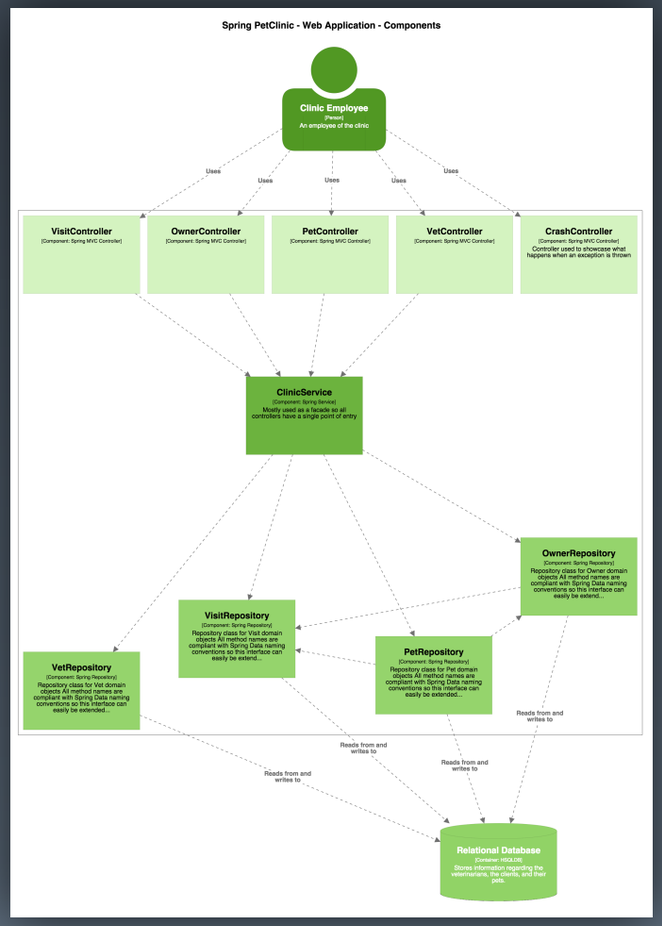
```
// and now automatically find all Spring @Controller, @Component, @Service and @Repository components
ComponentFinder componentFinder = new ComponentFinder(
        webApplication, "org.springframework.samples.petclinic",
        new SpringComponentFinderStrategy(),
        new JavadocComponentFinderStrategy(new File("/Users/simon/Documents/sandbox/spring/spring-petc
componentFinder.findComponents();

// connect the user to all of the Spring MVC controllers
webApplication.getComponents().stream()
        .filter(c -> c.getTechnology().equals("Spring MVC Controller"))
        .forEach(c -> user.uses(c, "Uses"));

// connect all of the repository components to the relational database
webApplication.getComponents().stream()
        .filter(c -> c.getTechnology().equals("Spring Repository"))
        .forEach(c -> c.uses(relationalDatabase, "Reads from and writes to"));

// finally create some views
ViewSet viewSet = workspace.getViews();
SystemContextView contextView = viewSet.createContextView(springPetClinic);
contextView.addAllSoftwareSystems();
contextView.addAllPeople();

ContainerView containerView = viewSet.createContainerView(springPetClinic);
containerView.addAllPeople();
containerView.addAllSoftwareSystems();
containerView.addAllContainers();

ComponentView componentView = viewSet.createComponentView(webApplication);
componentView.addAllComponents();
componentView.addAllPeople();
componentView.add(relationalDatabase);

// link the architecture model with the code
for (Component component : webApplication.getComponents()) {
    if (component.getSourcePath() != null) {
        component.setSourcePath(component.getSourcePath().replace(
                "/Users/simon/Documents/sandbox/spring/spring-petclinic/",
```

```
// finally create some views
ViewSet viewSet = workspace.getViews();
SystemContextView contextView = viewSet.createContextView(springPetClinic);
contextView.addAllSoftwareSystems();
contextView.addAllPeople();

ContainerView containerView = viewSet.createContainerView(springPetClinic);
containerView.addAllPeople();
containerView.addAllSoftwareSystems();
containerView.addAllContainers();

ComponentView componentView = viewSet.createComponentView(webApplication);
componentView.addAllComponents();
componentView.addAllPeople();
componentView.add(relationalDatabase);
```

Spring PetClinic - System Context

Spring PetClinic - Containers

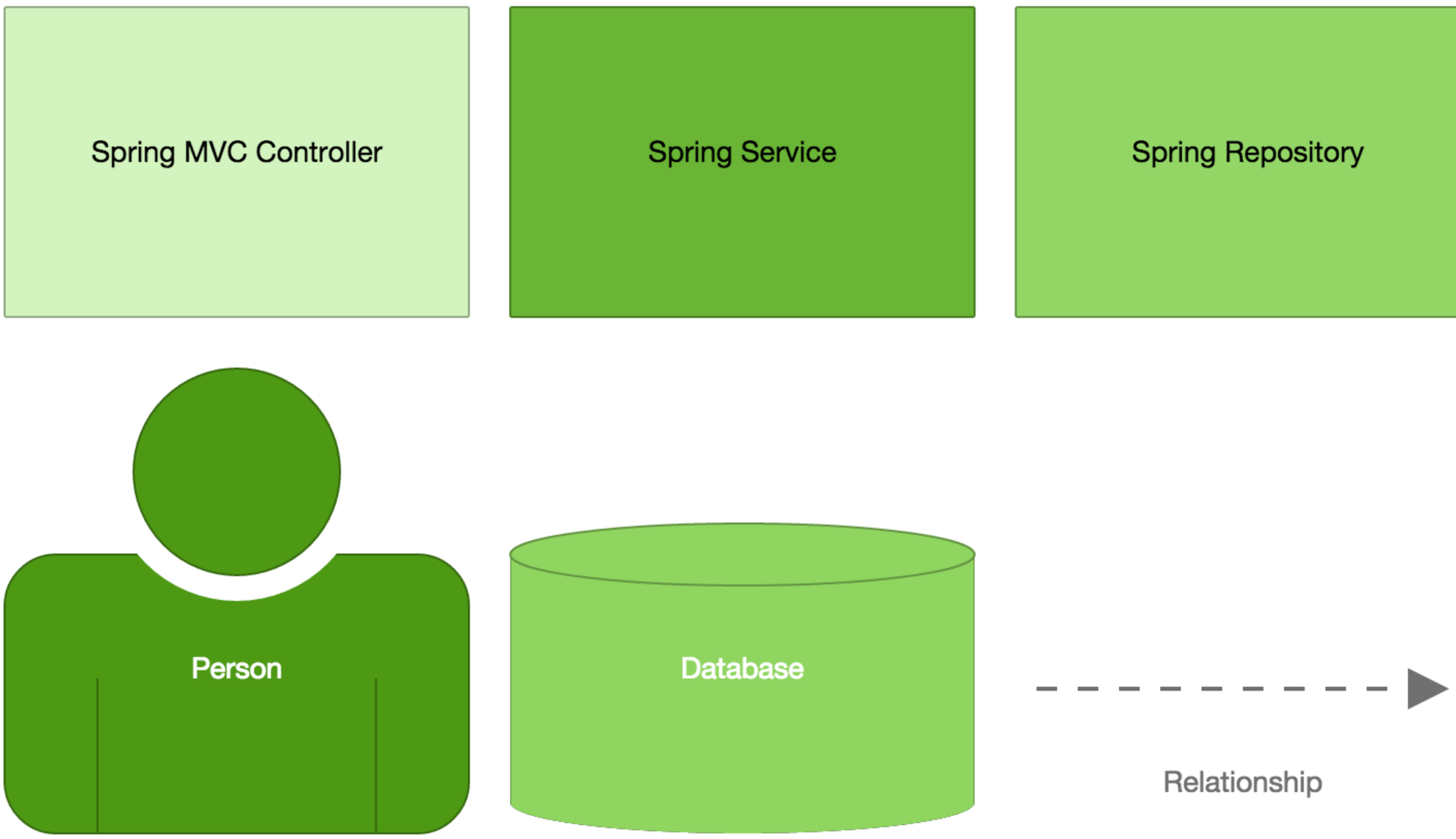




Spring PetClinic - Web Application - Components

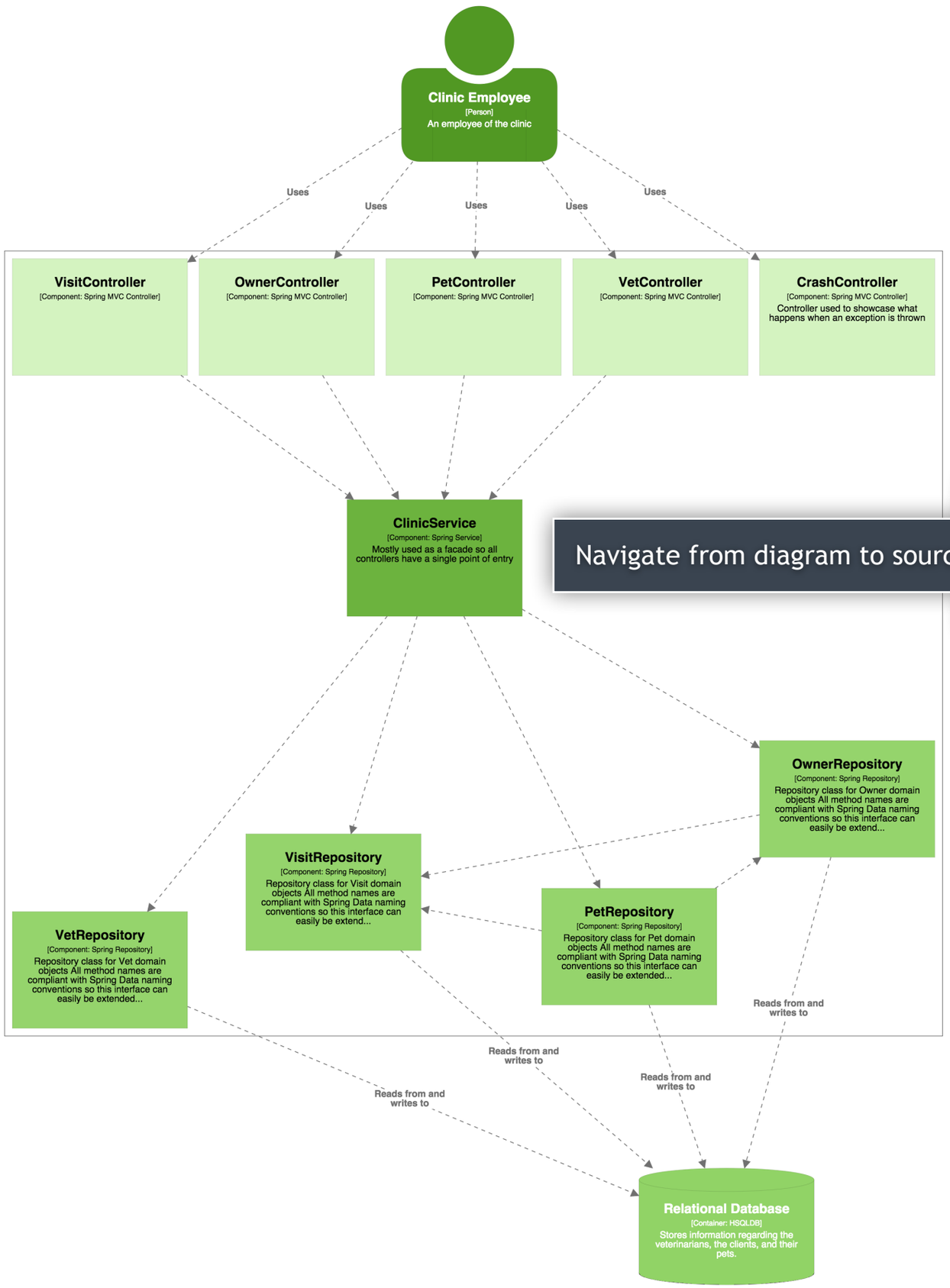
Clinic Employee
[Person]
An employee of the clinic
Uses
VisitController
[Component: Spring MVC Controller]
OwnerController
[Component: Spring MVC Controller]
PetController
[Component: Spring MVC Controller]
VetController
[Component: Spring MVC Controller]
CrashController
[Component: Spring MVC Controller]
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
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 extend...
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 extend...
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 extended...
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 extended...
Reads from and writes to

Spring PetClinic - Web Application - Components



Close

Spring PetClinic - Web Application - Components



Navigate from diagram to source code

GitHub This repository Search Explore Features Enterprise Pricing Sign up Sign in

spring-projects / spring-petclinic Watch 174 Star 614 Fork 1,070

Branch: master

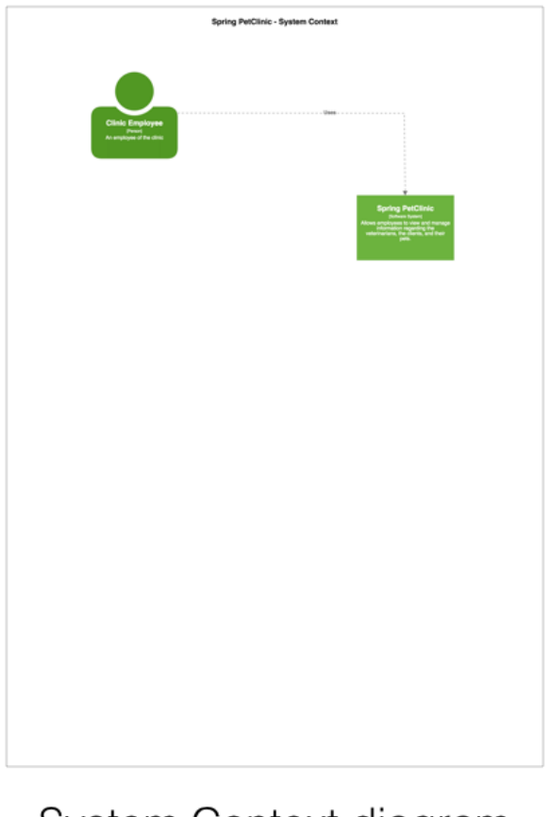
spring-petclinic / src / main / java / org / springframework / samples / petclinic / service / **ClinicService.java**

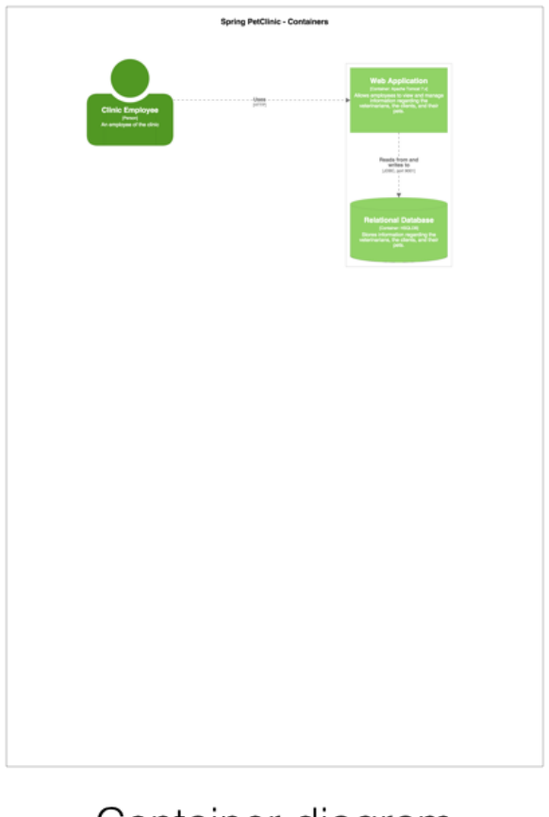
michaelisvy test methods:used should/shouldNot 5c9ab6 on 16 Jan

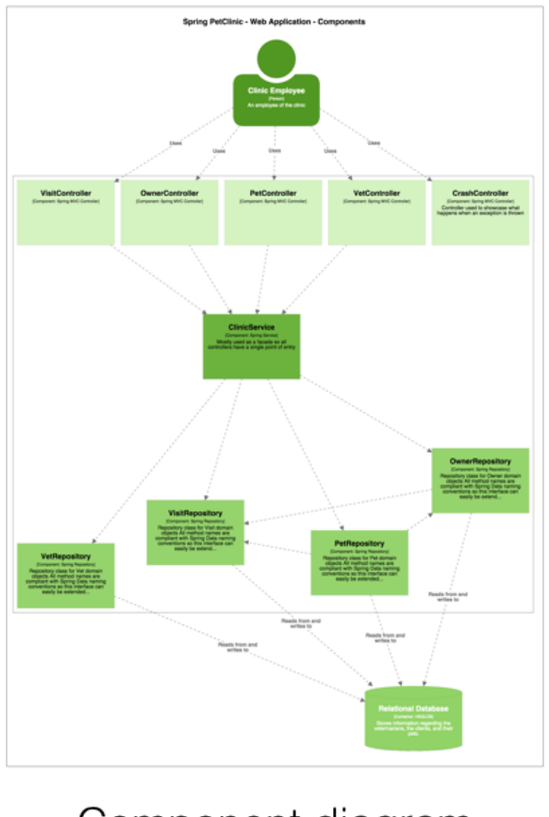
2 contributors

52 lines (38 sloc) 1.67 KB Raw Blame History

```
1  /*
2   * Copyright 2002-2013 the original author or authors.
3   *
4   * Licensed under the Apache License, Version 2.0 (the "License");
5   * you may not use this file except in compliance with the License.
6   * You may obtain a copy of the License at
7   *
8   *      http://www.apache.org/licenses/LICENSE-2.0
9   *
10  * Unless required by applicable law or agreed to in writing, software
11  * distributed under the License is distributed on an "AS IS" BASIS,
12  * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13  * See the License for the specific language governing permissions and
14  * limitations under the License.
15  */
16 package org.springframework.samples.petclinic.service;
17
18 import java.util.Collection;
19
20 import org.springframework.dao.DataAccessException;
21 import org.springframework.samples.petclinic.model.Owner;
22 import org.springframework.samples.petclinic.model.Pet;
23 import org.springframework.samples.petclinic.model.PetType;
24 import org.springframework.samples.petclinic.model.Vet;
25 import org.springframework.samples.petclinic.model.Visit;
26
27
28 /**
29  * Mostly used as a facade so all controllers have a single point of entry
30  *
31  * @author Michael Isvy
32  */
33 public interface ClinicService {
34
35     Collection<PetType> findPetTypes() throws DataAccessException;
36
37     Owner findOwnerById(int id) throws DataAccessException;
38
39     Pet findPetById(int id) throws DataAccessException;
40
41     void savePet(Pet pet) throws DataAccessException;
42
43     void saveVisit(Visit visit) throws DataAccessException;
44
45     Collection<Vet> findVets() throws DataAccessException;
46
47     void saveOwner(Owner owner) throws DataAccessException;
48
49     Collection<Owner> findOwnerByLastName(String lastName) throws DataAccessException;
50
51 }
```







System Context diagram | Container diagram | Component diagram | Source code

Double-click a software system

Double-click a container

Double-click a component

Diagrams are maps

techtribes.je - Web Application - Components - All components

Anonymous User
[Person]
Anybody on the web.

Aggregated User
[Person]
A user or business with content that is aggregated into the website, signed in using their Twitter ID.

Administration User
[Person]
A system administration user, signed in using a Twitter ID.

BooksController
[Component: Spring MVC Controller]
Allows users to view books by local authors.

HomePageController
[Component: Spring MVC Controller]
Shows the home page.

TweetsController
[Component: Spring MVC Controller]
Shows tweets by local people and tribes.

NewsController
[Component: Spring MVC Controller]
Shows recent news.

TalksController
[Component: Spring MVC Controller]
Shows talks by local people.

BadgesController
[Component: Spring MVC Controller]

JobsController
[Component: Spring MVC Controller]

CreationsController
[Component: Spring MVC Controller]
Shows creations (websites, apps, etc) by local people/businesses.

ViewByMonthController
[Component: Spring MVC Controller]
Allows users to view a summary of site content by month.

SignInController
[Component: Spring MVC Controller]
Allows registered users to sign in using their Twitter ID.

PeopleController
[Component: Spring MVC Controller]
Shows a summary of a person; including contents, tweets, talks, etc.

UserProfileController
[Component: Spring MVC Controller]
Allows the user to view and manage their profile (e.g. GitHub ID).

SignOutController
[Component: Spring MVC Controller]
Allows the user to sign out.

AdminController
[Component: Spring MVC Controller]
Provides some basic administration features related to adding people, tribes, etc.

RssController
[Component: Spring MVC Controller]
Provides an RSS feed of recent content.

ActivityController
[Component: Spring MVC Controller]
Shows a summary of the recent activity (number of blog posts, tweets, events, etc).

SearchController
[Component: Spring MVC Controller]
Allows users to search for content and tweets.

AboutPageController
[Component: Spring MVC Controller]
Shows the about page, which includes a summary of the number of people, tribes, blog posts, tweets, etc.

EventsController
[Component: Spring MVC Controller]
Shows local business and community events (e.g. meetups, etc).

ContentController
[Component: Spring MVC Controller]
Show recent content (news and blog posts).

ErrorPageController
[Component: Spring MVC Controller]
Handles all error scenarios (e.g. page not found, internal server error, etc).

CodeController
[Component: Spring MVC Controller]
Shows GitHub repositories owned by local coders.

TribesController
[Component: Spring MVC Controller]
Shows a summary of a tribe; including contents, tweets, talks, etc.

SummaryController
[Component: Spring MVC Controller]
Shows the summary page, which includes a summary of the number of people, tribes, blog posts, tweets, etc.

BookComponent
[Component]
Provides access to information about books by local authors.

CreationComponent
[Component]
Provides access to information about creations by local people/businesses.

GitHubComponent
[Component]
Provides access to the list of GitHub repos associated with content sources.

ContentUpdaterComponent
[Component]
Refreshes the cache of content sources and their activity every five minutes.

LoggingComponent
[Component]
Provides logging facilities to all other components.

JobComponent
[Component]
Provides access to information about recently posted jobs.

ActivityComponent
[Component]
Provides access to information about, and calculates, the recent activity rankings

BadgeComponent
[Component]
Manages the badges that have been awarded to people and tribes.

ContentSourceComponent
[Component]
Provides access to information about people and tribes.

TalkComponent
[Component]
Provides access to information about talks by local speakers.

EventComponent
[Component]
Provides access to information about local events (e.g. meetups, user groups, etc).

SearchComponent
[Component]
Search facilities for news feed entries and tweets.

NewsFeedEntryComponent
[Component]
Provides access to blog entries and news.

TweetComponent
[Component]
Provides access to tweets.

Relational Database
[Container: MySQL 5.5.x]
Stores people, tribes, tribe membership, talks, events, jobs, badges, GitHub repos, etc.

File System
[Container: Local disk]
Stores search indexes.

NoSQL Data Store
[Container: MongoDB 2.2.x]
Stores content from RSS/Atom feeds (blog posts) and tweets.

uses

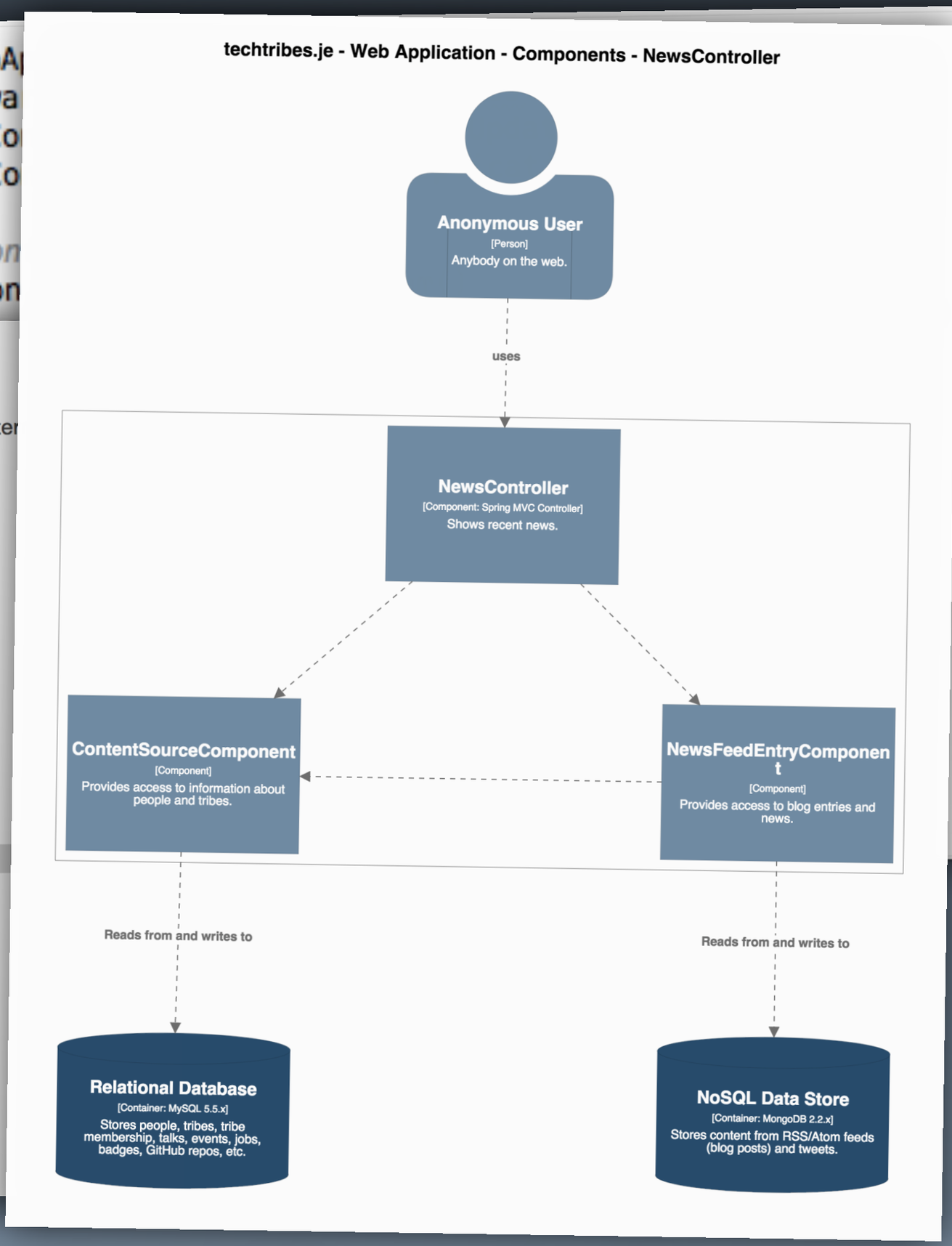
Writes logs messages using

Gets tweets from

Reads from and writes to

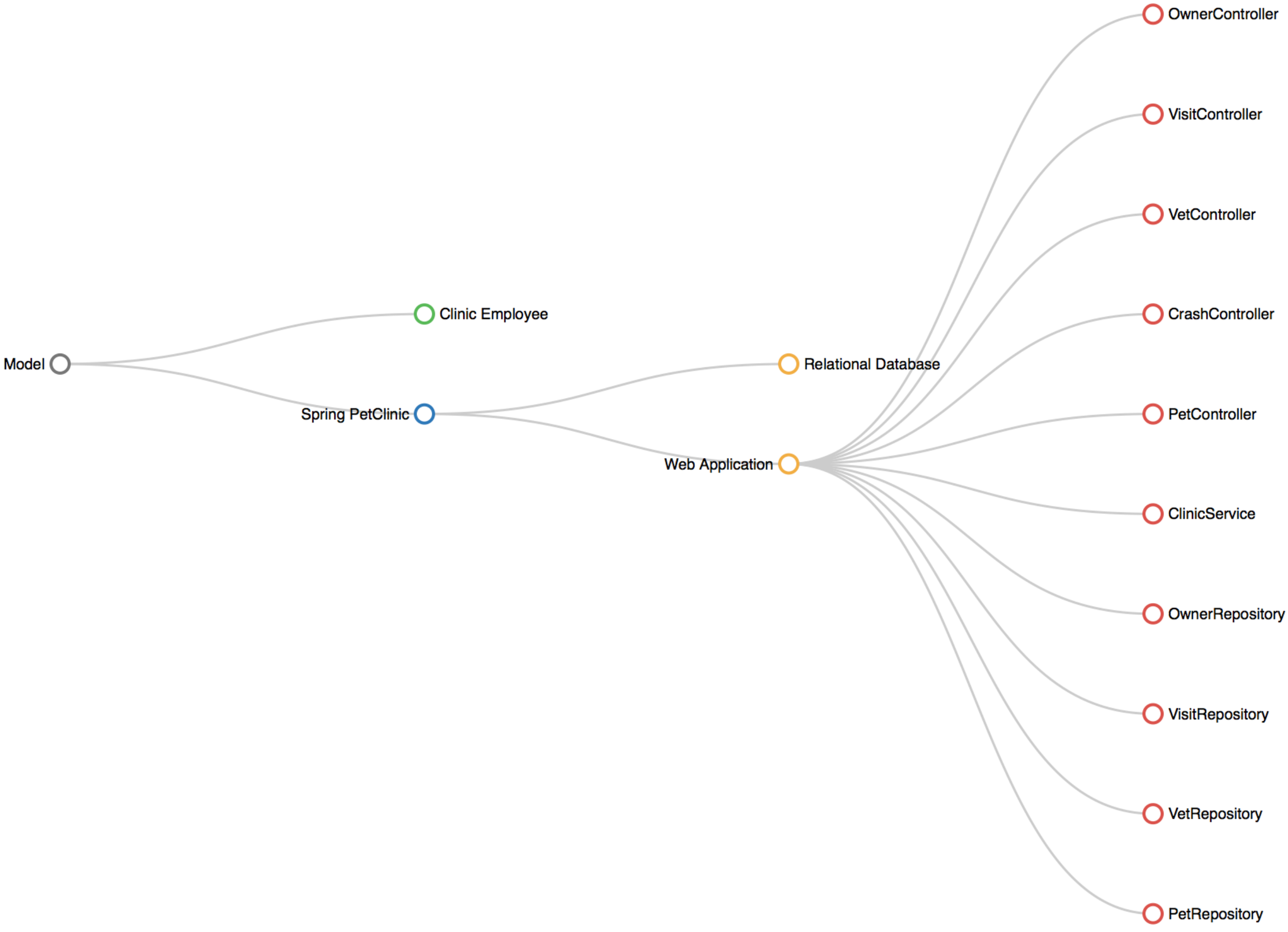
```
private static void createComponentViewsForWebA
    SoftwareSystem techTribes = model.getSoftwa
    Container contentUpdater = techTribes.getCo
    Container webApplication = techTribes.getCo

    // create one component view per Spring con
    Set<Component> controllers = webApplication
```

techtribes.je - System Context
techtribes.je - Containers
techtribes.je - Content Updater - Components - Awarding badges
techtribes.je - Content Updater - Components - Updating information from exter
techtribes.je - Content Updater - Components - Updating recent activity
techtribes.je - Web Application - Components - AboutPageController
techtribes.je - Web Application - Components - ActivityController
techtribes.je - Web Application - Components - AdminController
✓ techtribes.je - Web Application - Components - All components
techtribes.je - Web Application - Components - BadgesController
techtribes.je - Web Application - Components - BooksController
techtribes.je - Web Application - Components - CodeController
techtribes.je - Web Application - Components - ContentController
techtribes.je - Web Application - Components - CreationsController
techtribes.je - Web Application - Components - ErrorPageController
techtribes.je - Web Application - Components - EventsController
techtribes.je - Web Application - Components - HomePageController
techtribes.je - Web Application - Components - JobsController
techtribes.je - Web Application - Components - NewsController
techtribes.je - Web Application - Components - PeopleController
techtribes.je - Web Application - Components - RssController
techtribes.je - Web Application - Components - SearchController
techtribes.je - Web Application - Components - SignInController
techtribes.je - Web Application - Components - SignOutController
techtribes.je - Web Application - Components - SummaryController
techtribes.je - Web Application - Components - TalksController
techtribes.je - Web Application - Components - TribesController
techtribes.je - Web Application - Components - TweetsController
techtribes.je - Web Application - Components - UserProfileController
techtribes.je - Web Application - Components - ViewByMonthController

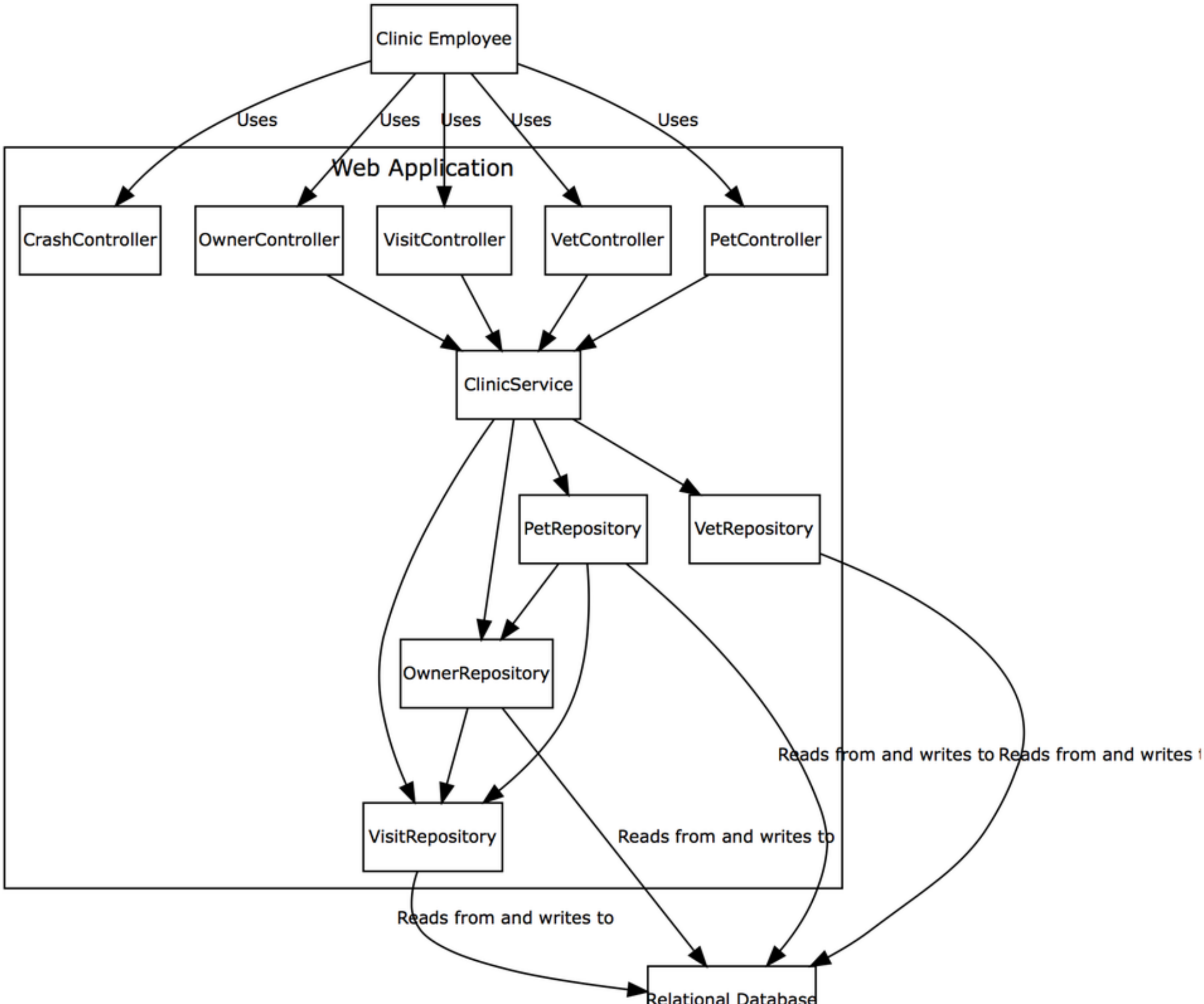


Creating the model as code provides opportunities…

Once you have a model, you can export that model and visualise it however you like…


Model
Clinic Employee
Spring PetClinic
Relational Database
Web Application
OwnerController
VisitController
VetController
CrashController
PetController
ClinicService
OwnerRepository
VisitRepository
VetRepository
PetRepository

Spring PetClinic - Web Application - Components

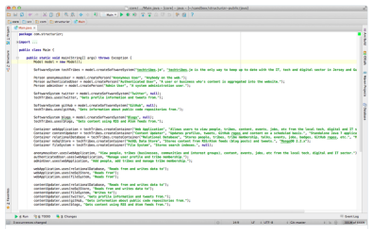


Neo4j GraphGist | Resources | Page Source | GitHub Gist/File / Dropbox URL

Architecture as a Graph (AaaG)

Techtribes.je technical architecture as a graph model

Simon Brown, Robin Bramley and Michael Hunger had a discussion on twitter about architecture modeling using code/a DSL, started by:



Simon Brown
@simonbrown
Follow

I've been messing with this stuff over the past few days … architecture model as code?

8:57 AM - 21 Jun 2014

3 5

Techtribes.je is the only way to keep up to date with the IT, tech and digital sector in Jersey and Guernsey, Channel Islands.

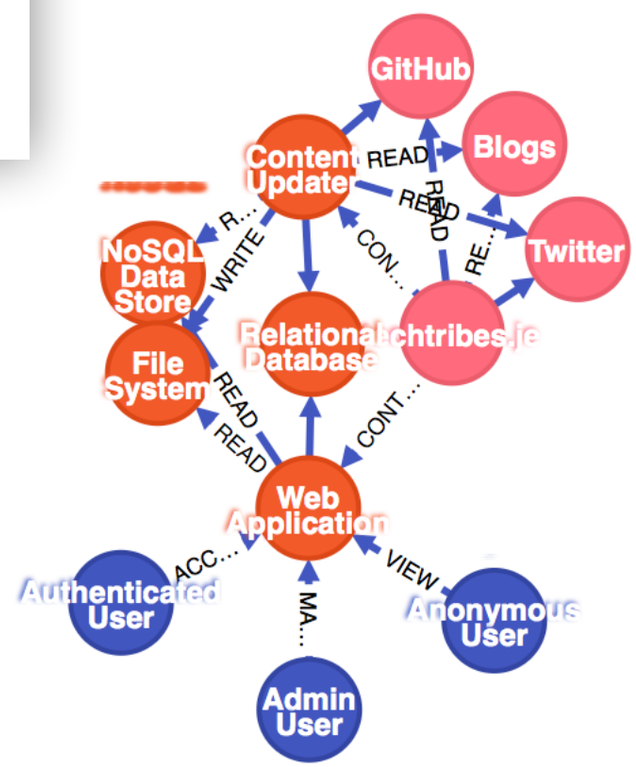
This interactive graph document describes its system architecture, and demonstrates some use-cases.

Setup

No Errors. Examples! G+1 Like 0 Tweet 0 Rate This

nsole

Test run OK



What can the different users do with which software

Query 2

```
MATCH (u:User)-[r]->(s:Software)
RETURN u.name, type(r), r.description, s.name
```

Test run OK

u.name	type(r)	r.description	s.name

TeamCity

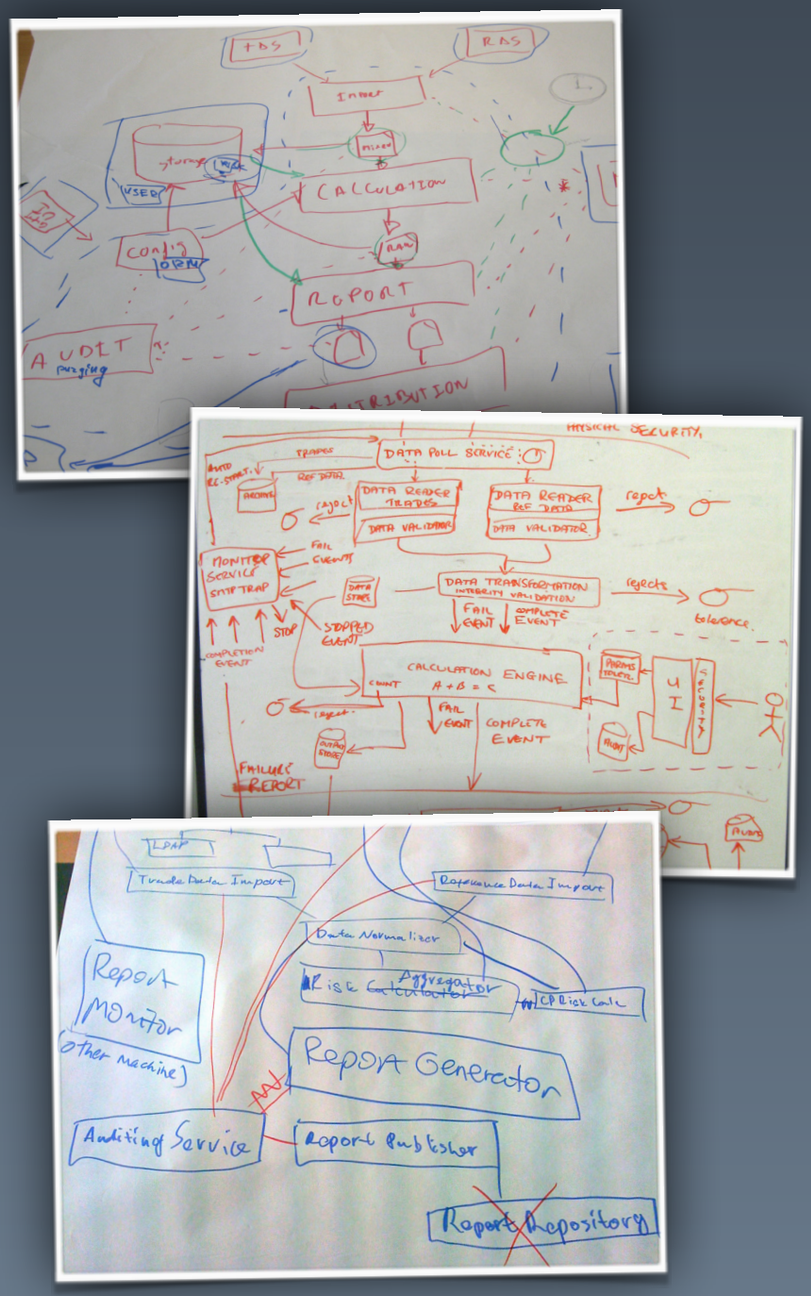
<APACHE ANT>

gradle

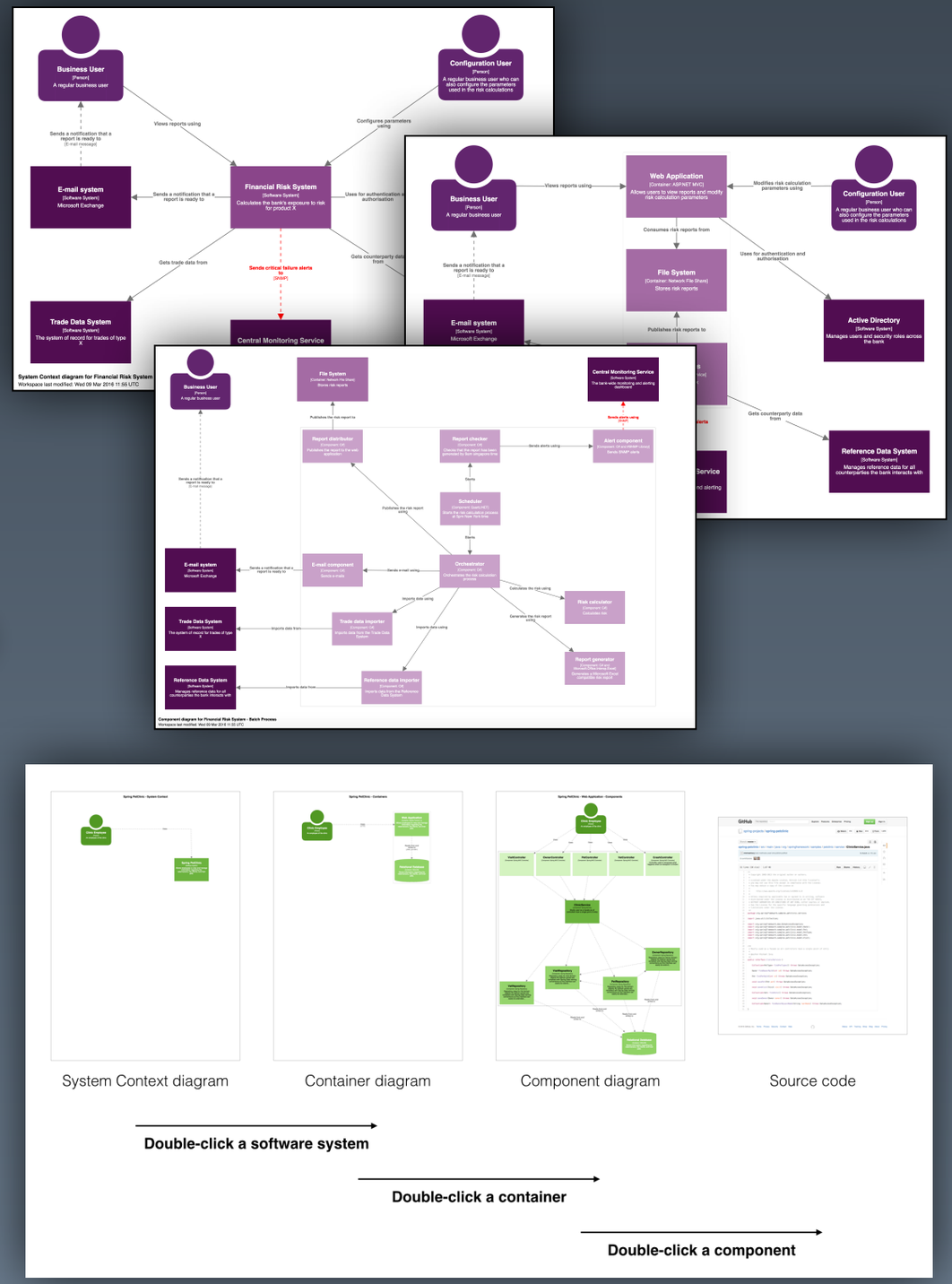
maven

Jenkins

Build pipeline integration keeps software architecture models up-to-date



From static diagrams to maps of the code



Visualising software architecture is still very much an *art*,
but it's 2016 and time to stop using tools like Microsoft Visio!

Thanks!

simon.brown@codingthearchitecture.com
@simonbrown on Twitter