coding {the} architecture

The Art of Visualising Software Architecture

Simon Brown
@simonbrown

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

Kristijan | Криштиџн
@Krishtidzn

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

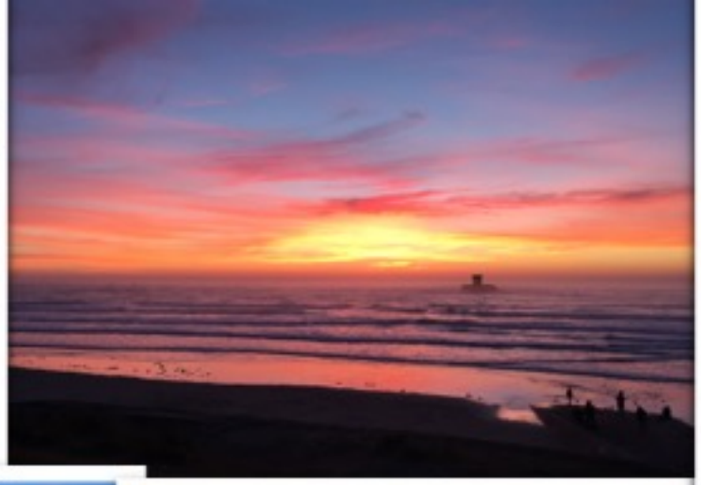
11:11 AM - 16 Apr 2015

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

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

Volume 1

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

Simon Brown

coding {the} architecture

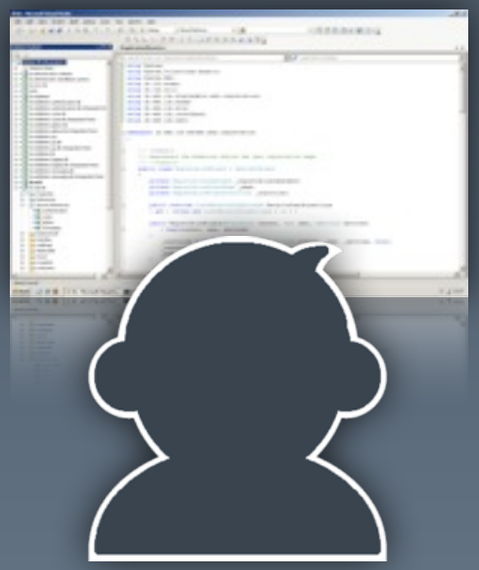
Software Architecture for Developers

Volume 2

Visualise, document and explore your software architecture

Simon Brown

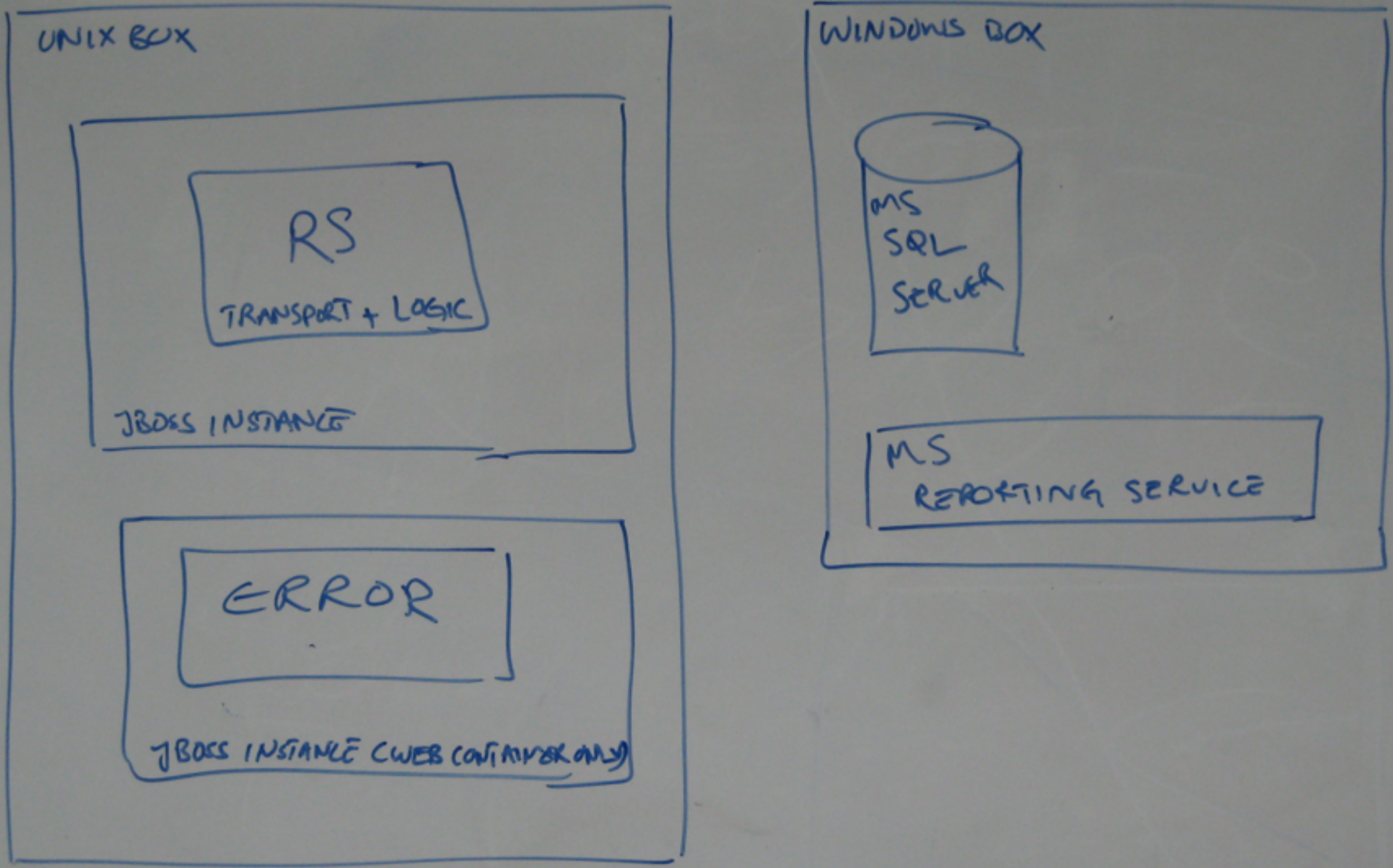
Leanpub



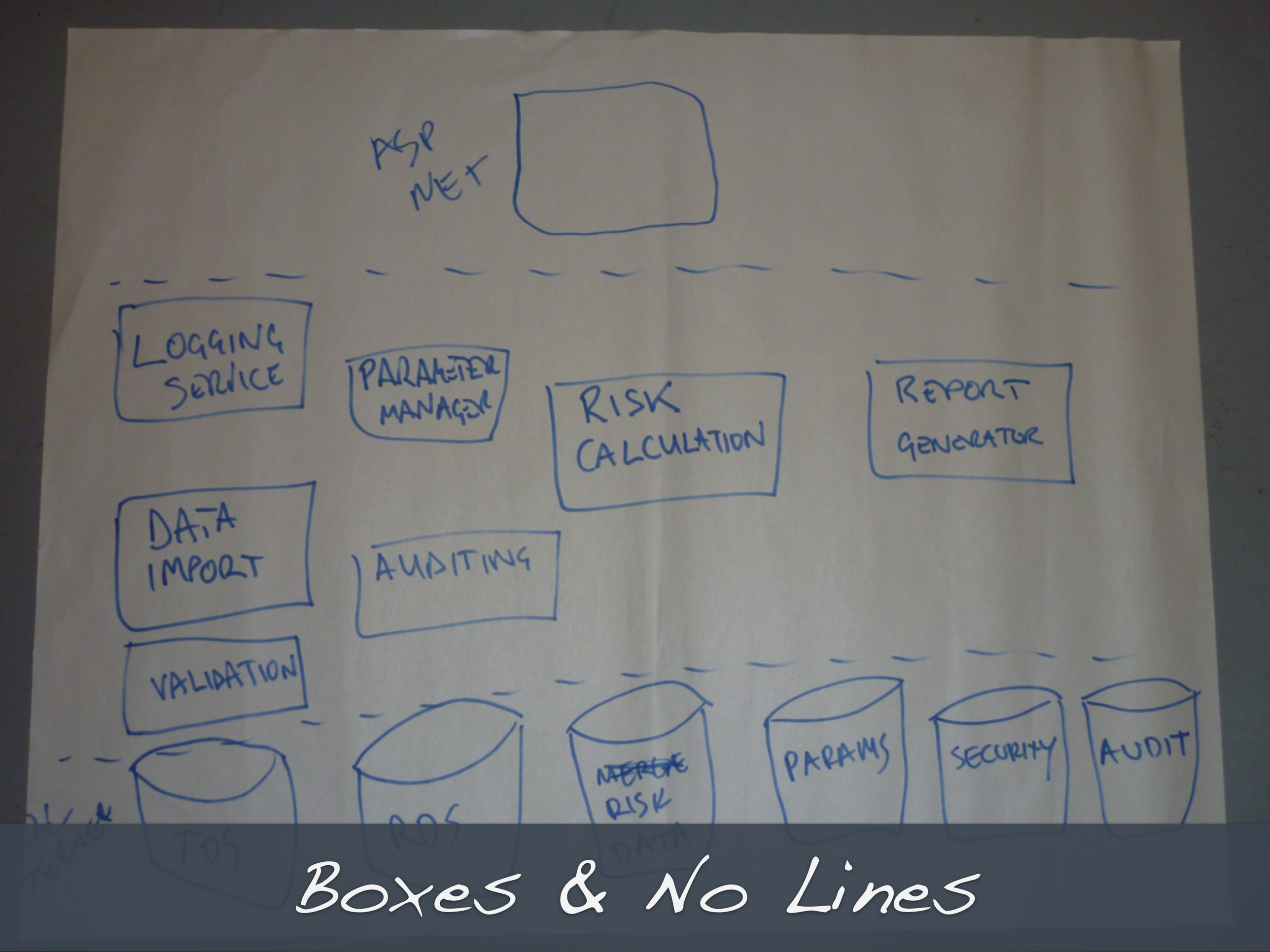
I code too

⇧ ; - ⇧ 0

The problem



The Shopping List



Boxes & No Lines

FUNCTIONAL VIEW

File Retriever

Scheduler

Auditing

Reference Archiver

Risk Assesment Processor

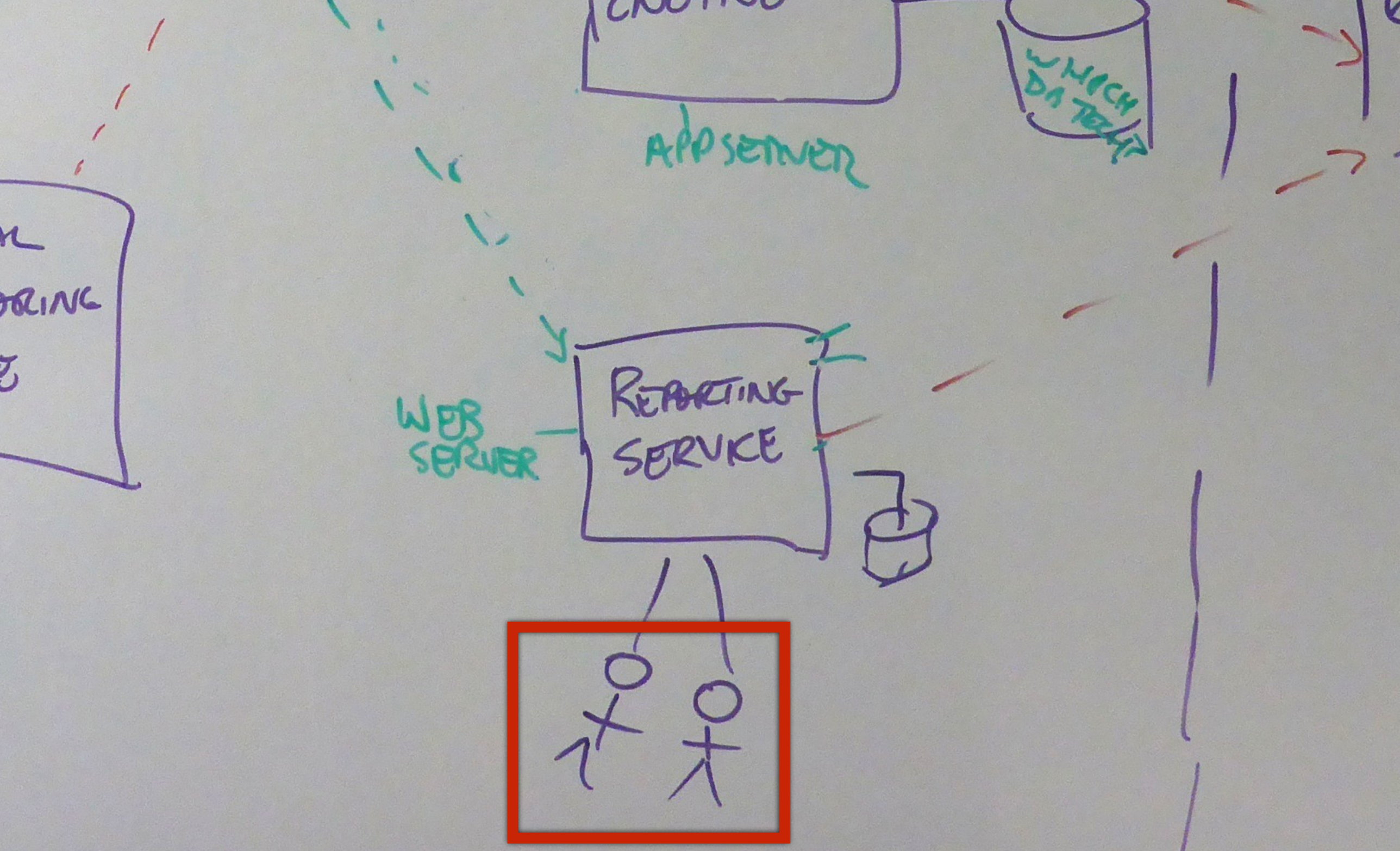
Risk Parameter Configuration

Trade Archiver

Report Generator

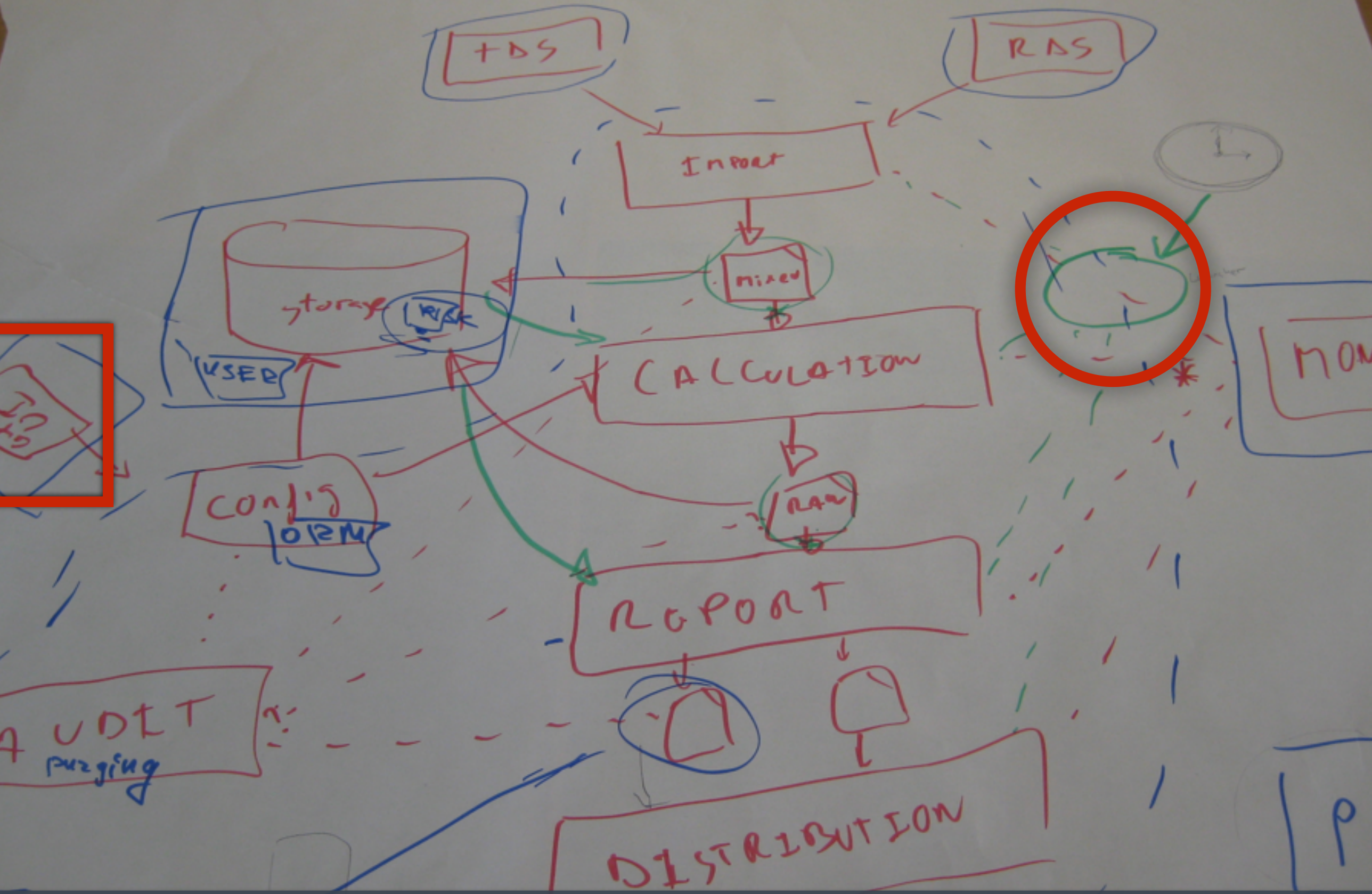
Report Distributor

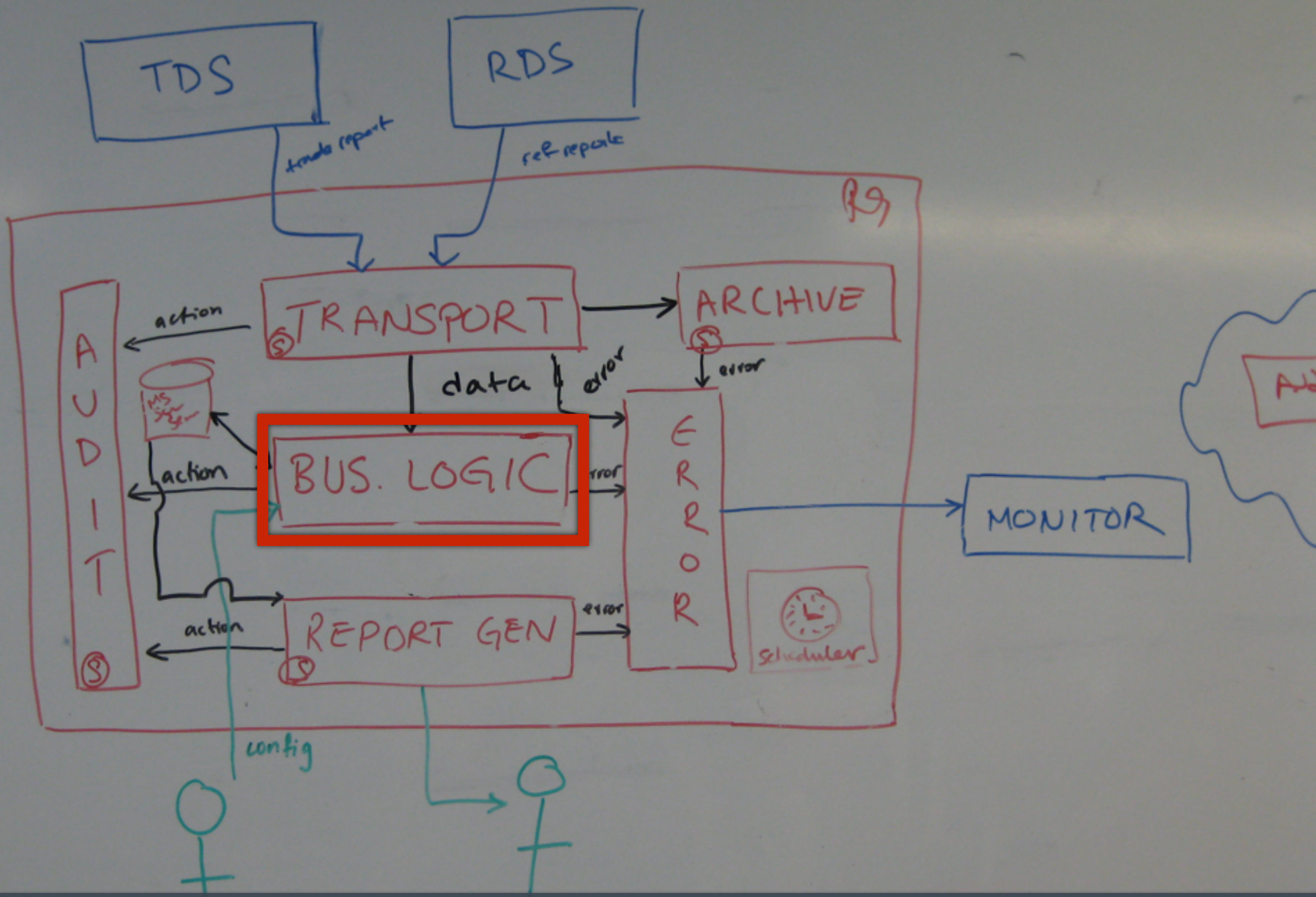
The Functional View



Stormtroopers

The Airline Route Map





Generically True

XML

XML

2 SERVICE/BATCH

IMPORT
JOIN
CALC

EXPORT/STORE
REPORT

CENTRAL
MONITORING
SERVICE

SNMP

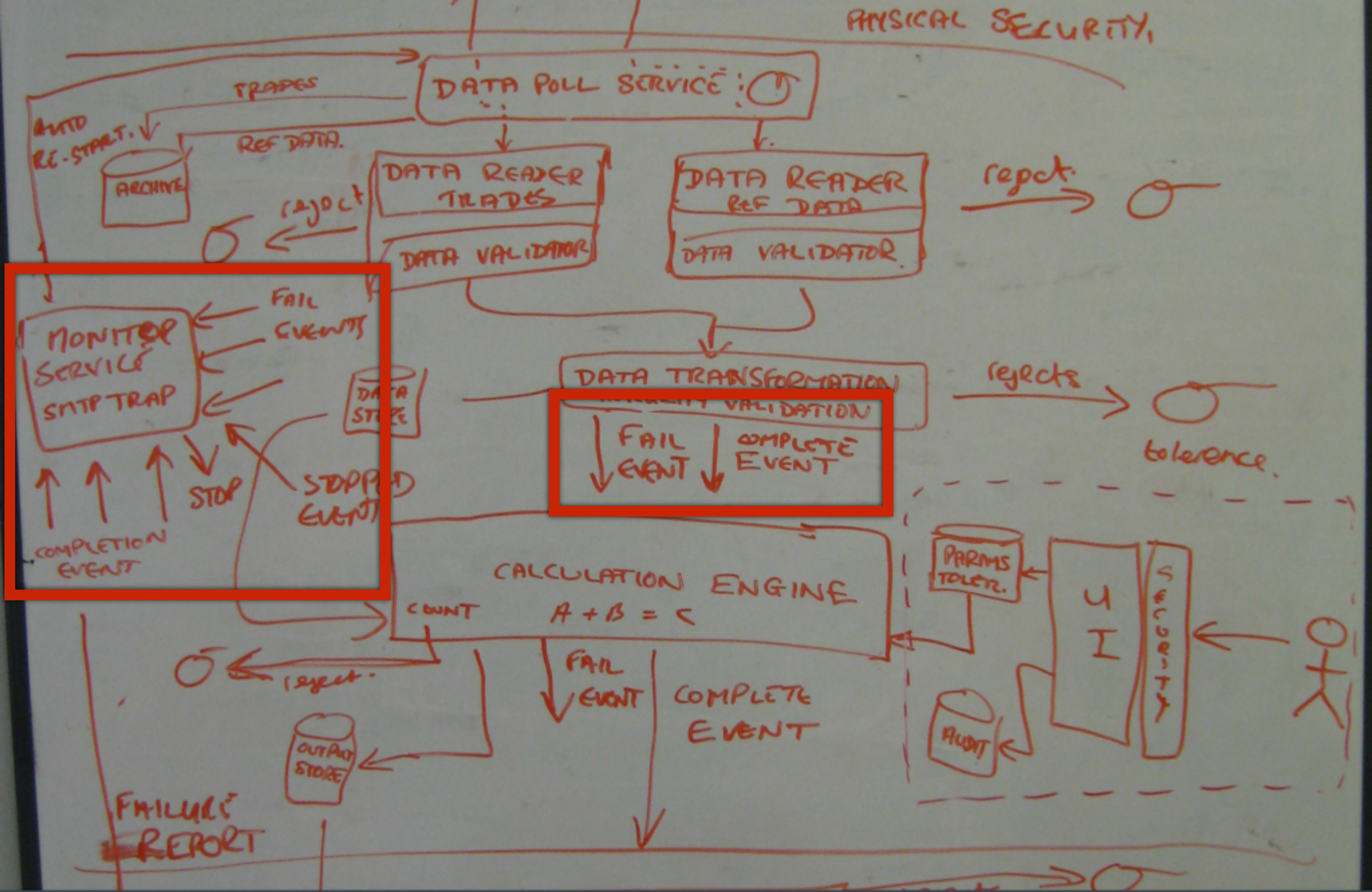
3 WEBAPP

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

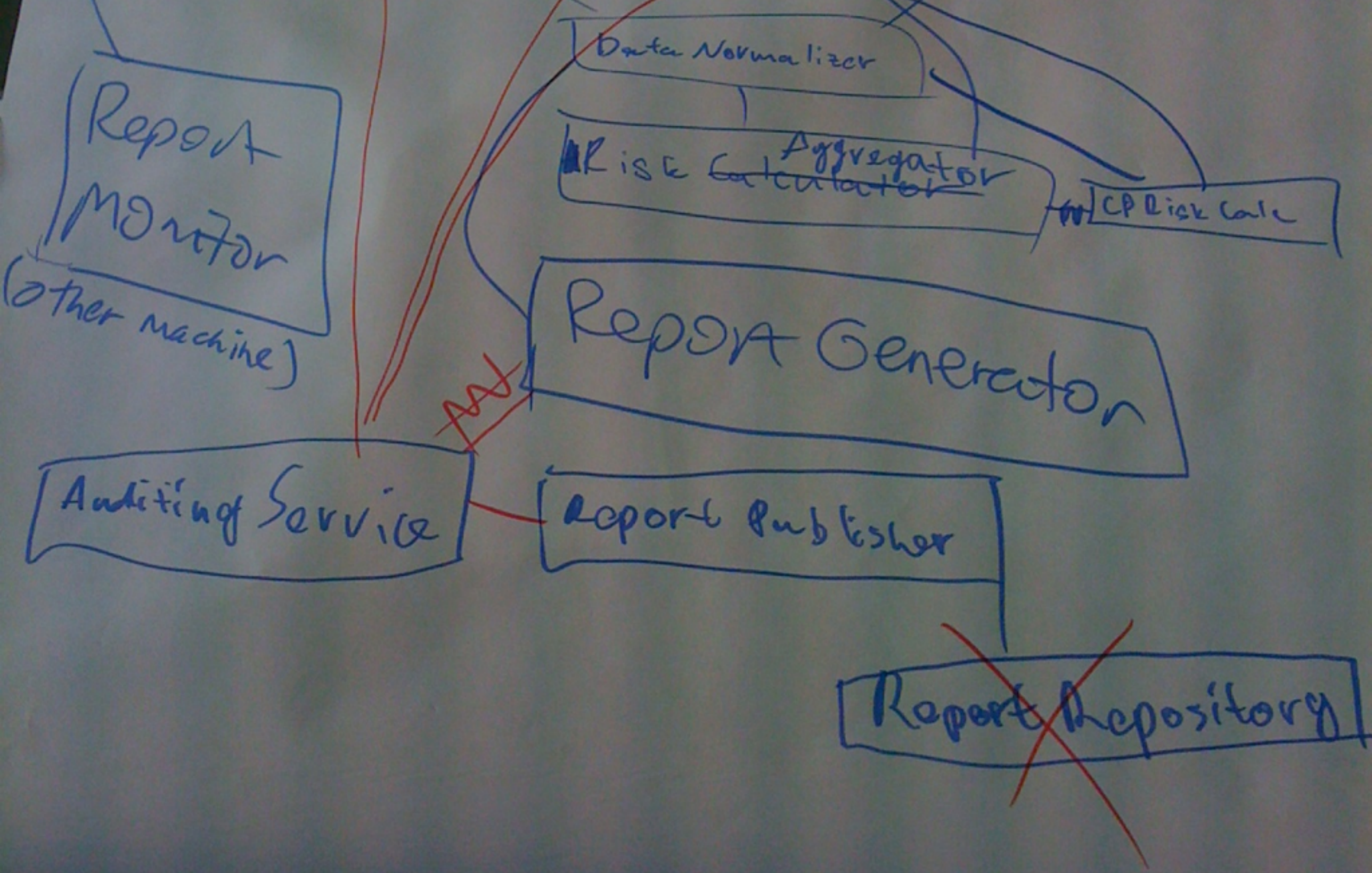
RISK RULES
USERS
ARCHIVE

The Logical View

Homeless Old C# Object (HOCO)



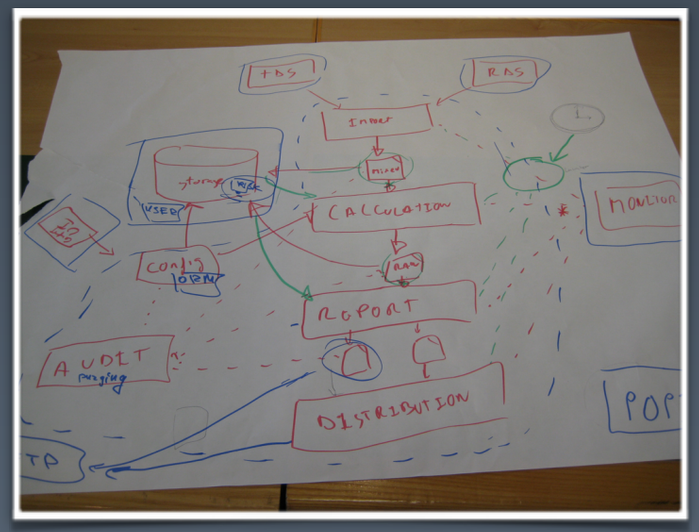
Choose your own adventure



Should have used a whiteboard!

Eh?

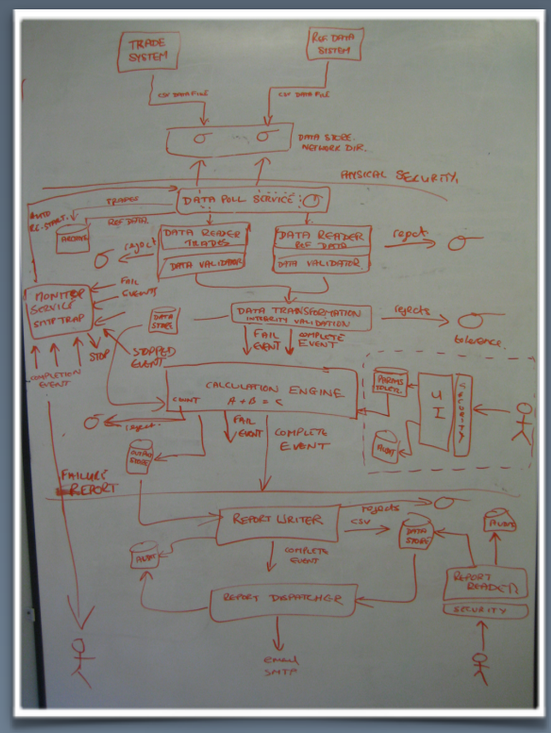
The diagram isn’t self-evident, but we’ll explain it



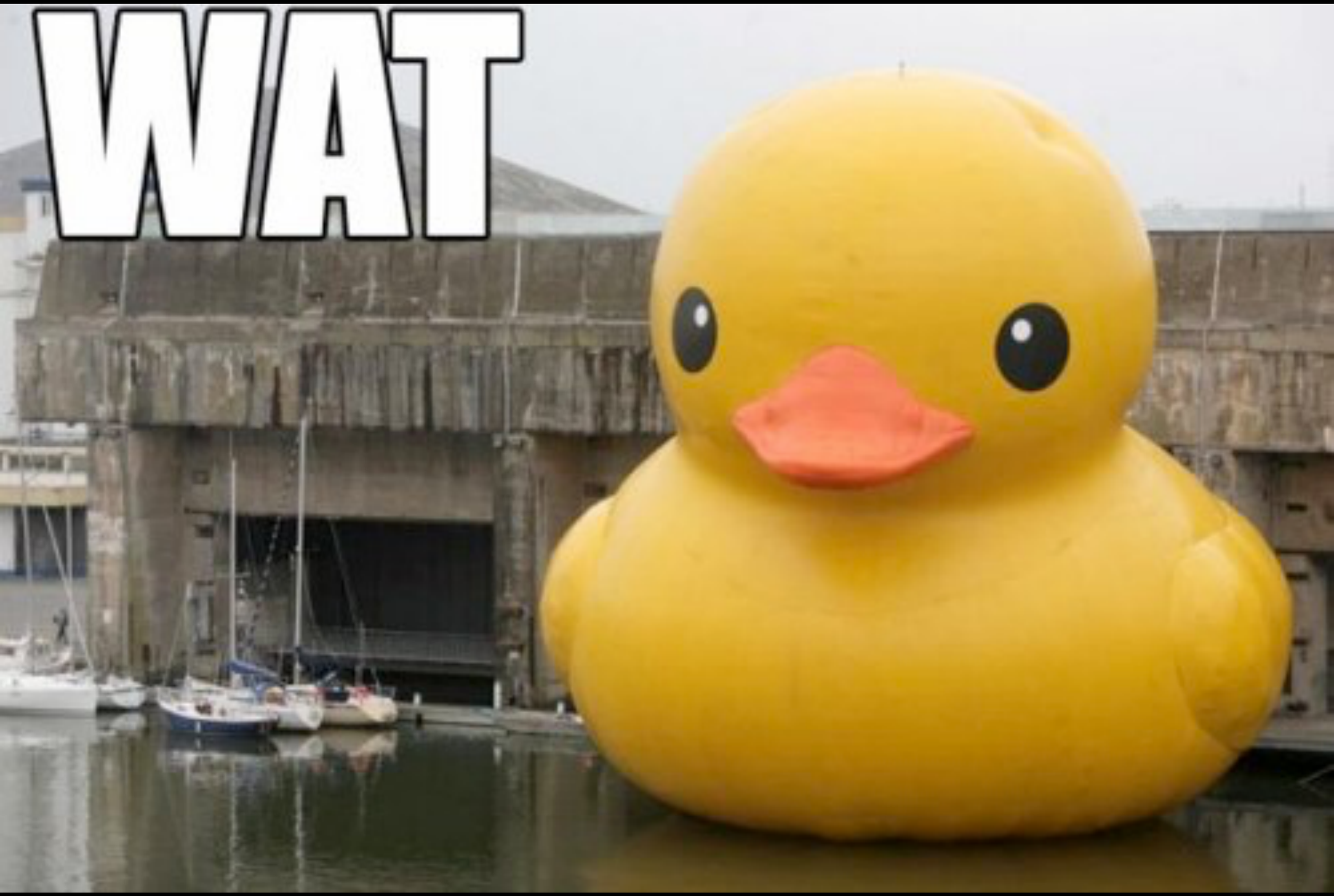
Team 1

Team 1

Team 2



Team 2


WAT

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.?)

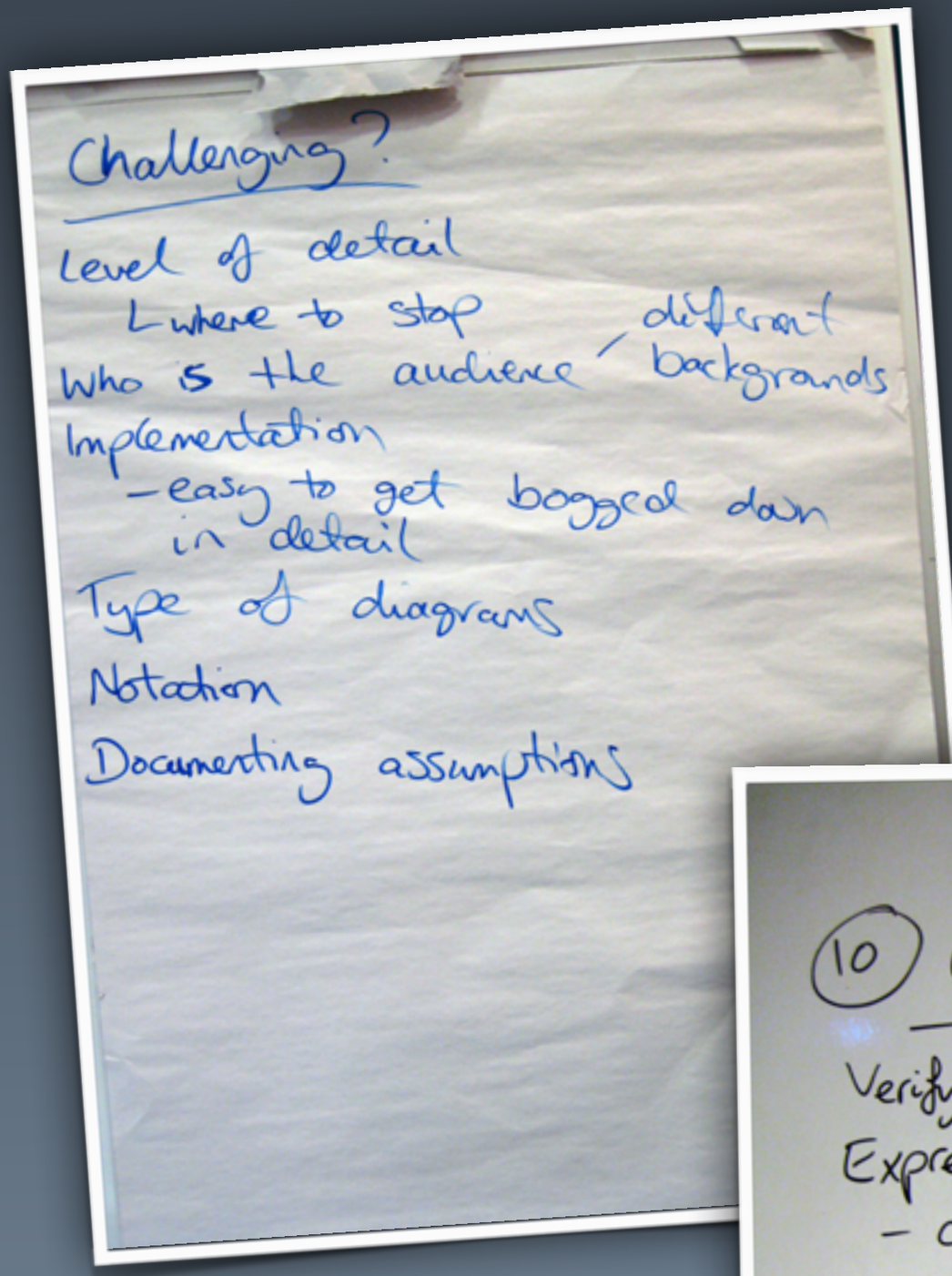
what shapes mean

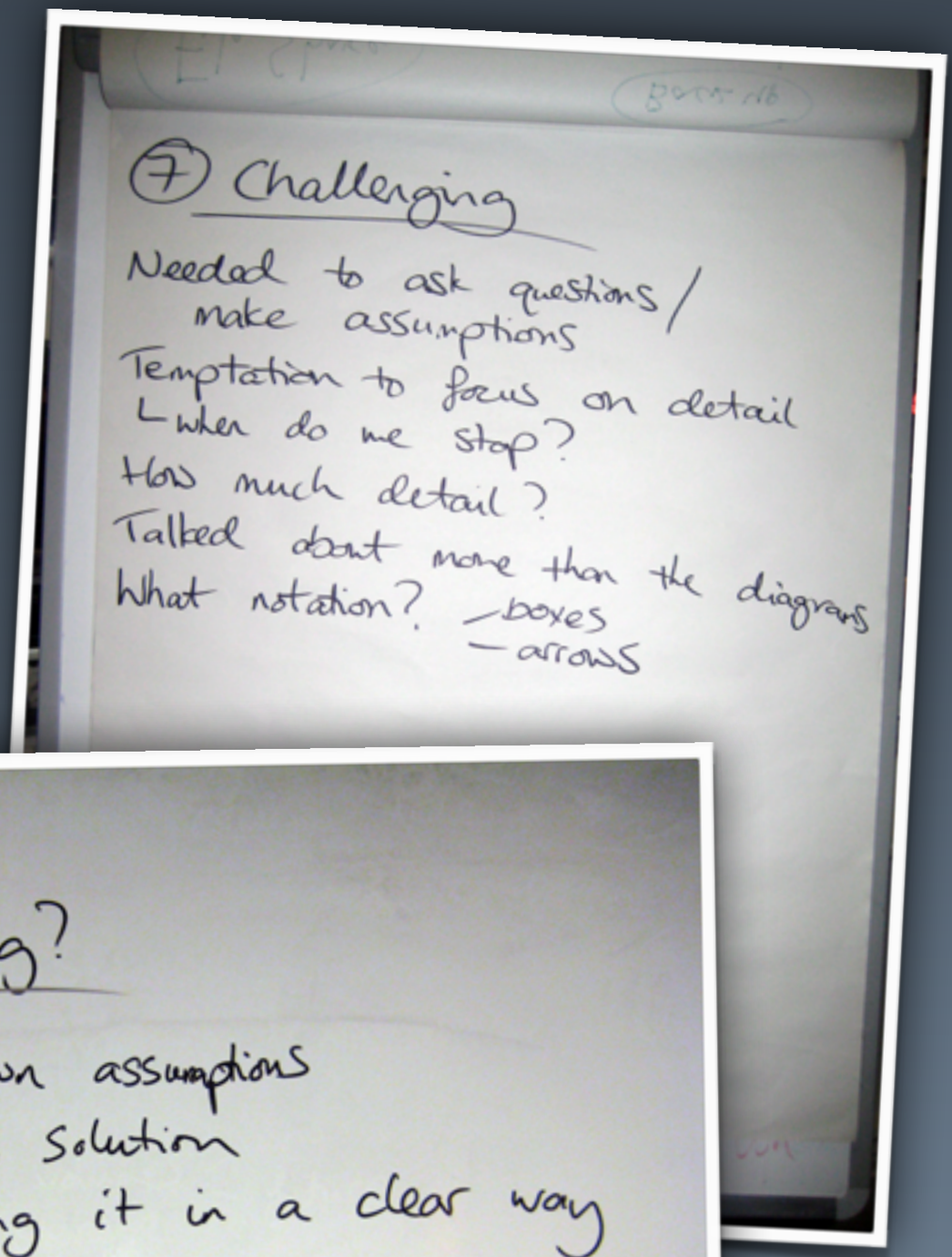
What's the DB-like icon?

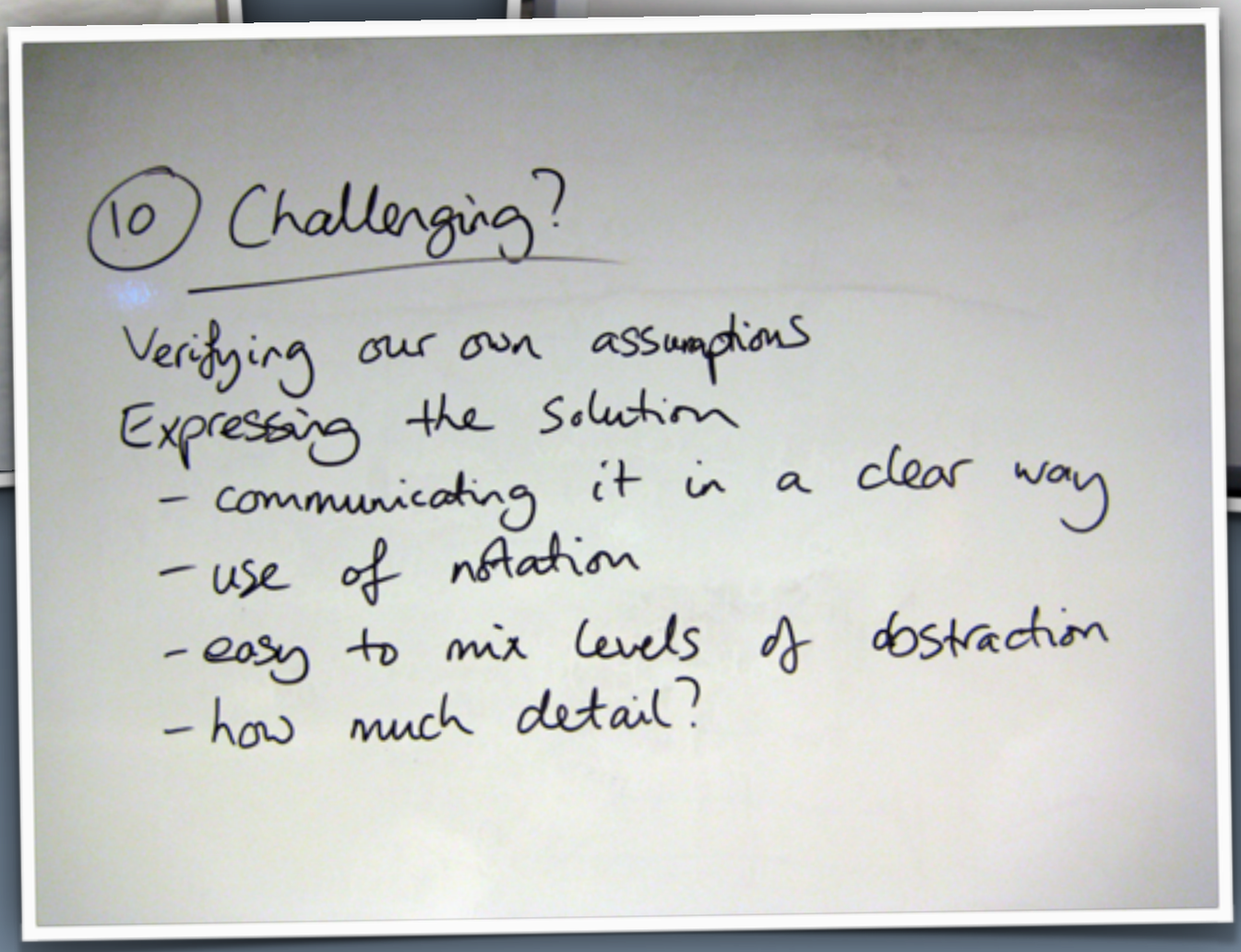
Not sure what this is

DIFFERENT LEVELS IN SAME DIAGRAM

ARE THE ARROWS THE RIGHT WAY ROUND?



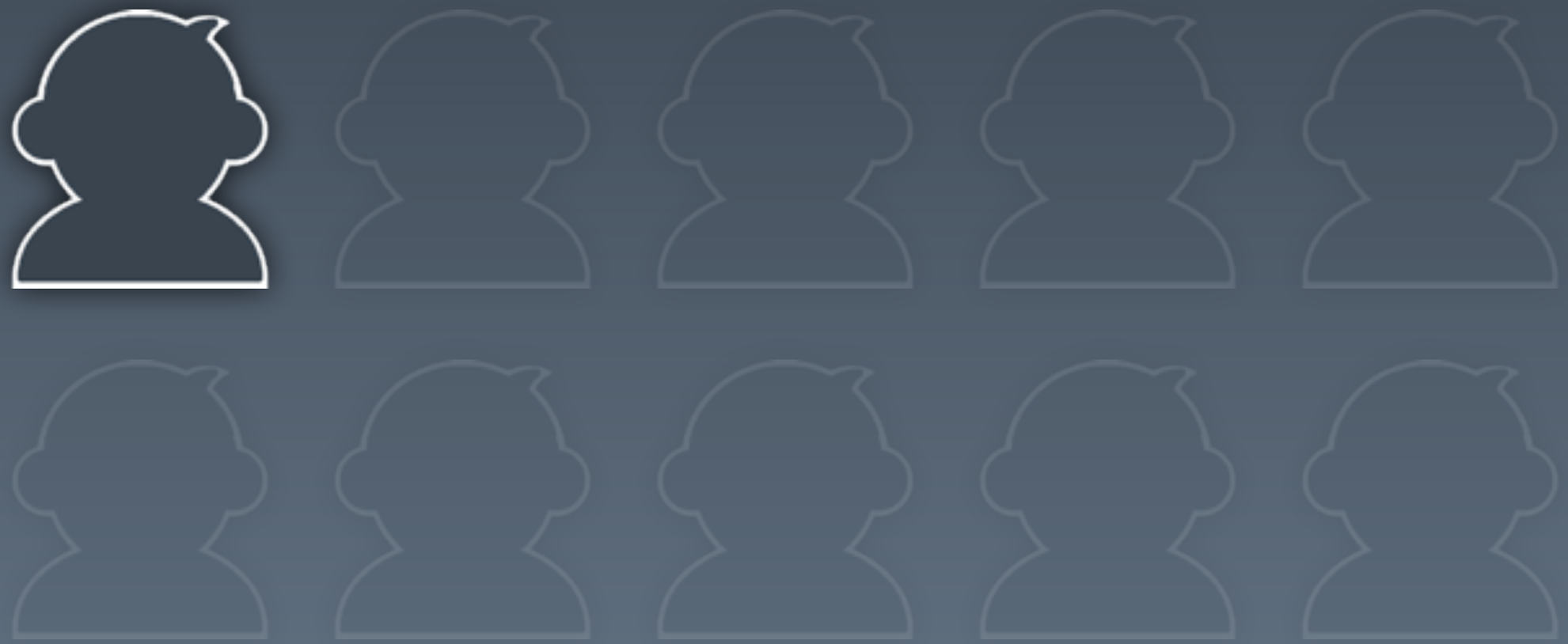




What's been challenging about the exercise?

Who here uses UML

on a regular basis?

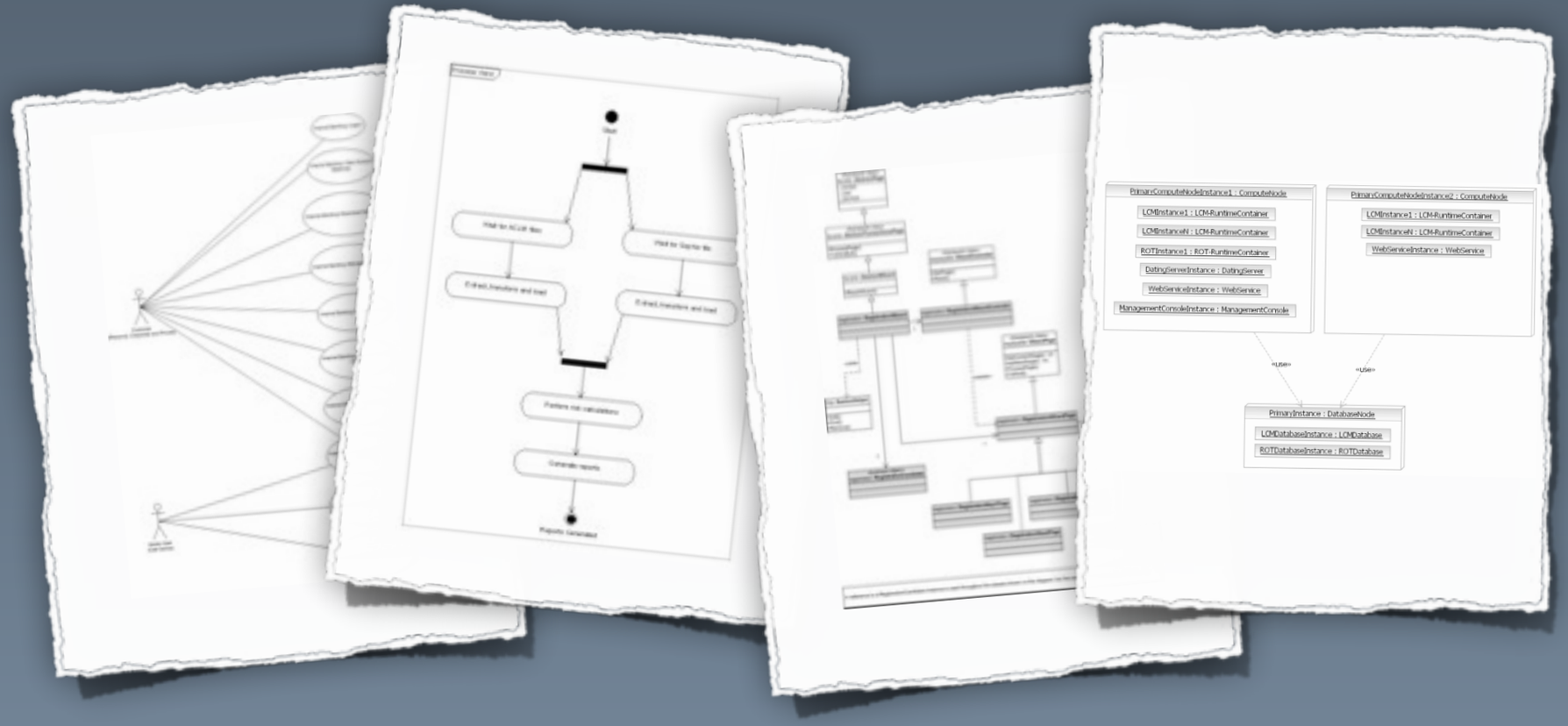


1 out of 10 people use UML

(in my experience)

I do use UML

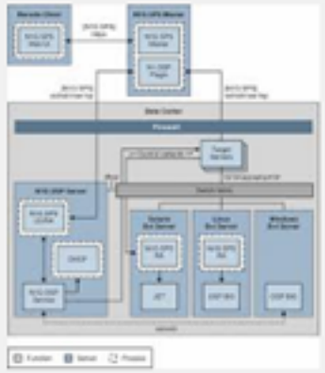
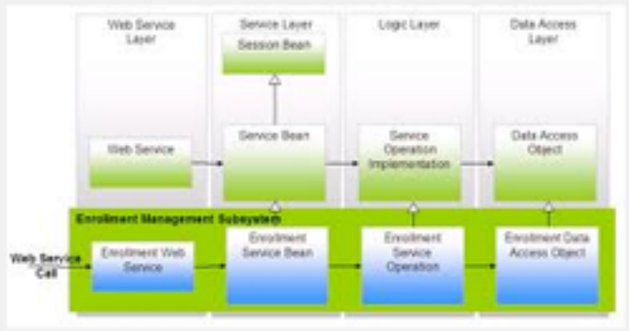
(activity, class, sequence, collaboration, state)

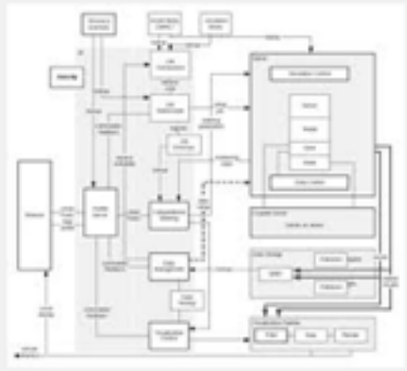
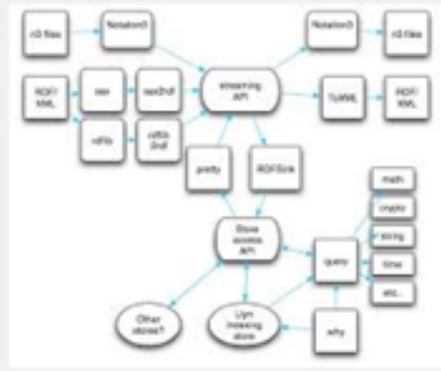


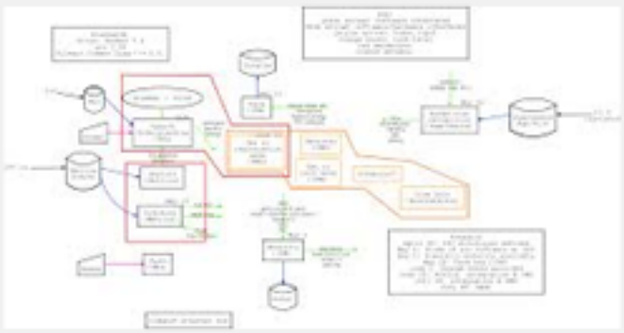
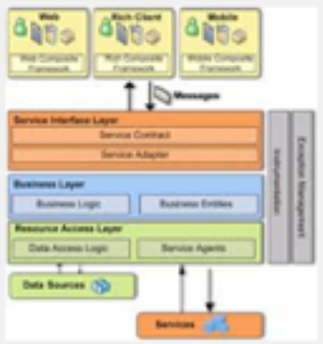
Google software architecture diagram

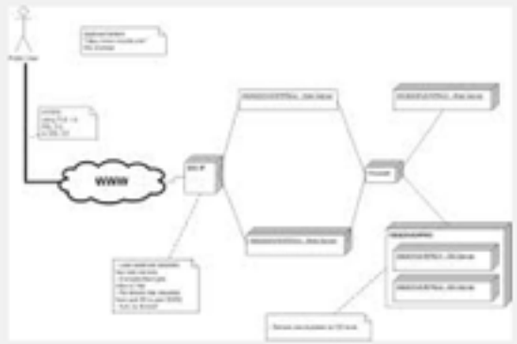
Sign in

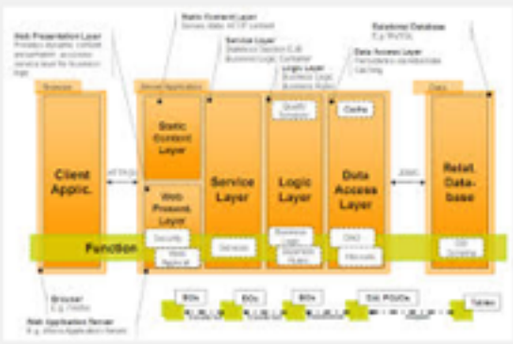
Web Images Videos Shopping News More Search tools SafeSearch

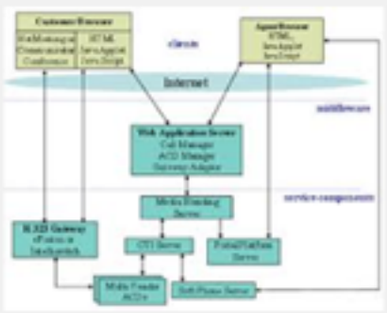




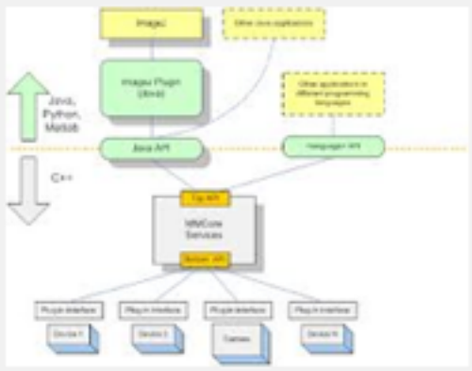



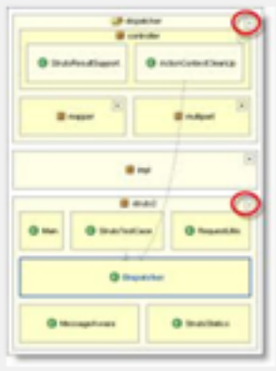



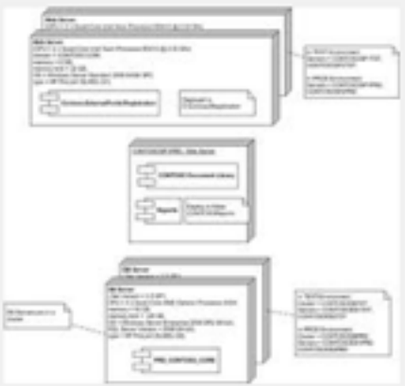
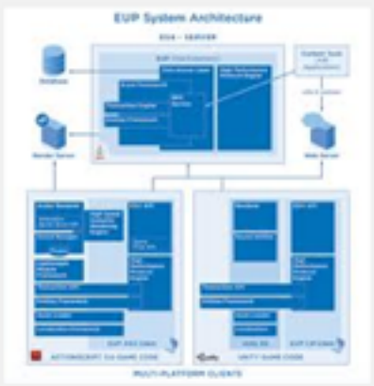






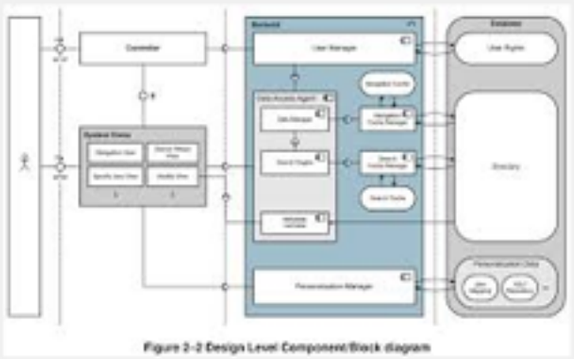
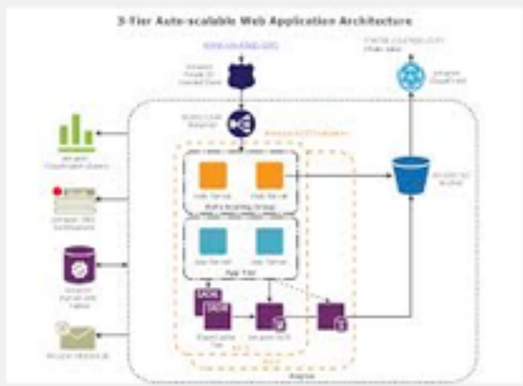
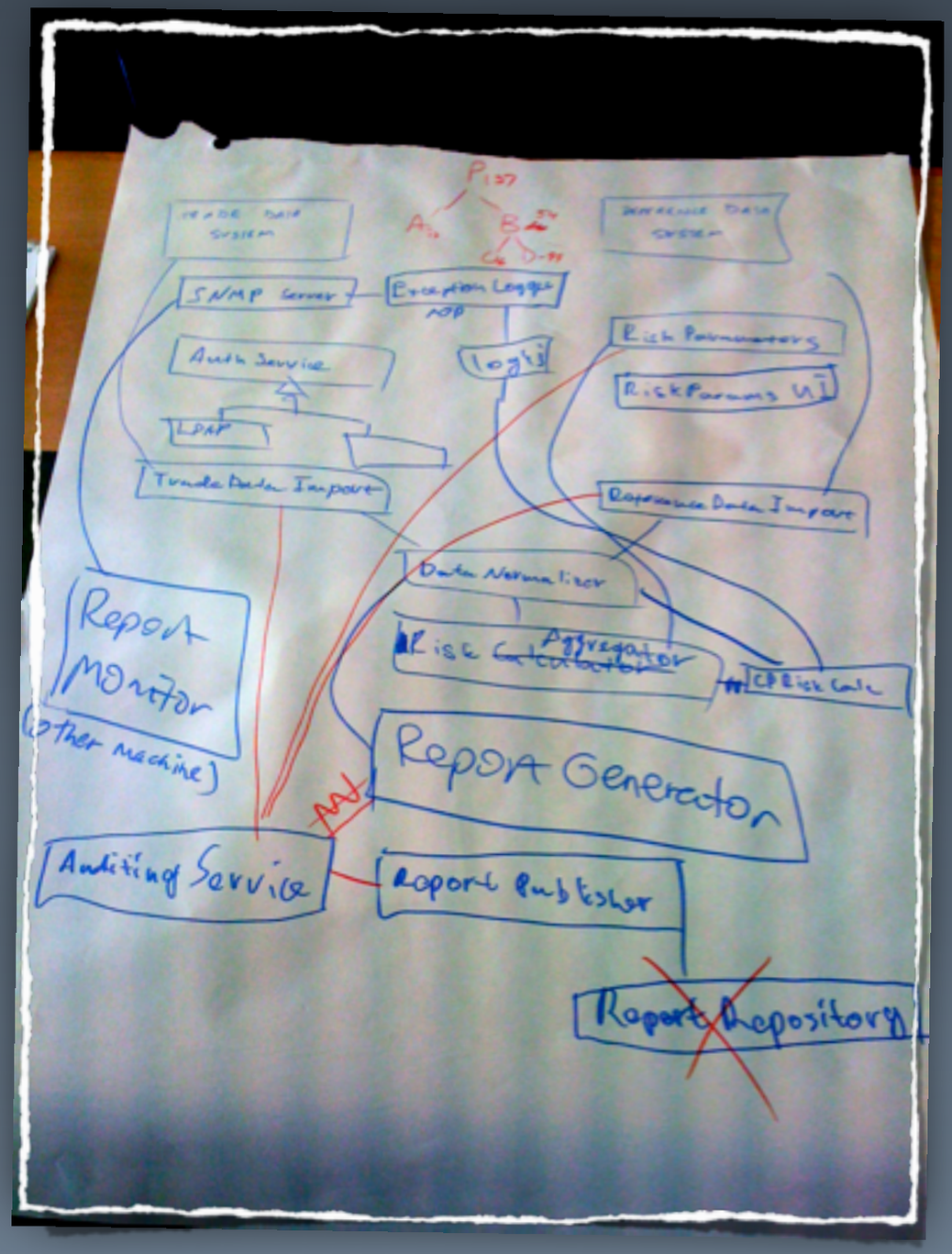


Figure 2-2 Design Level Component/Block diagram


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
Provisioning
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
Service Controller
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
Typical generic software architecture of SCADA systems
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
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
Output device manager
Output devices
User manager
DB-Server
Analysis

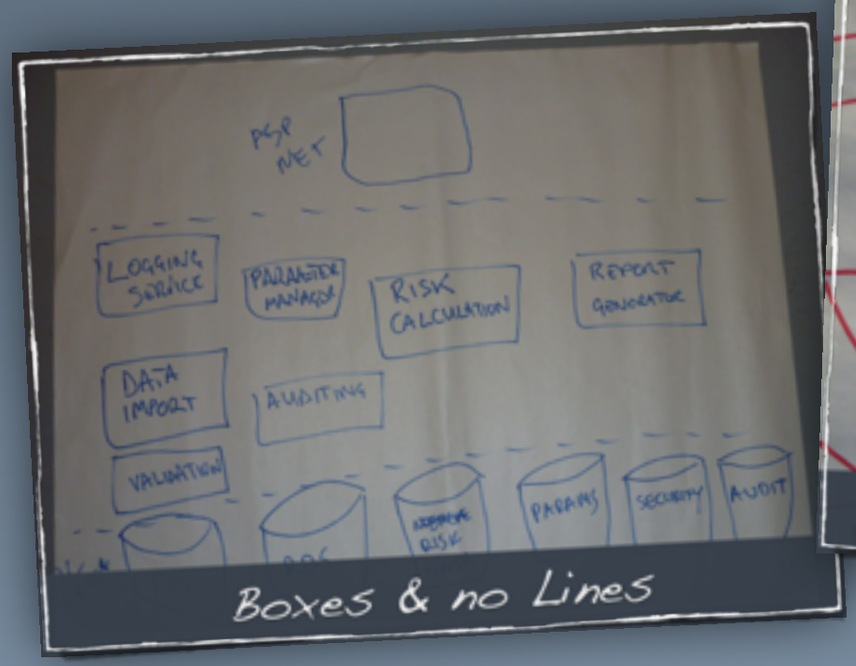
In my experience,

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

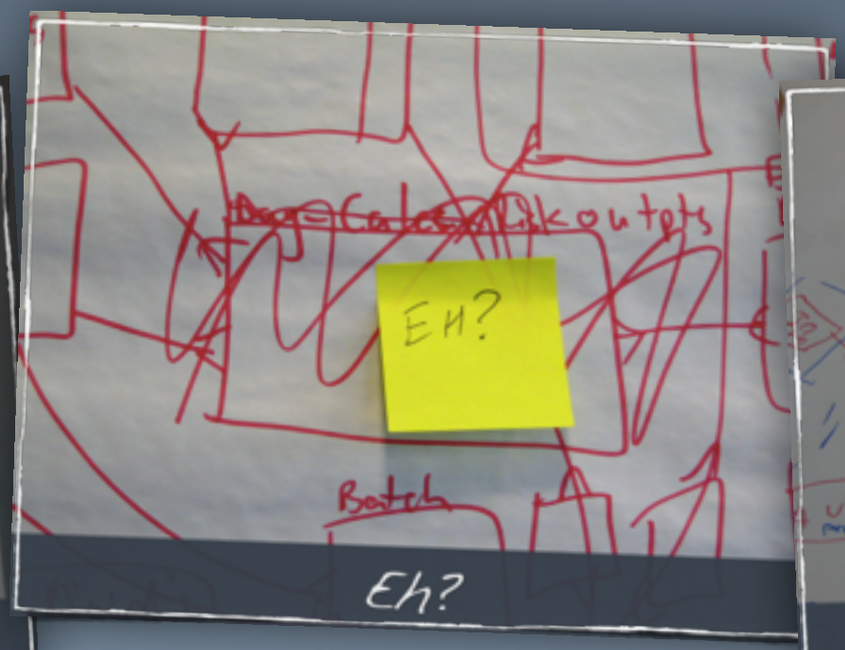


We can visualise our process…

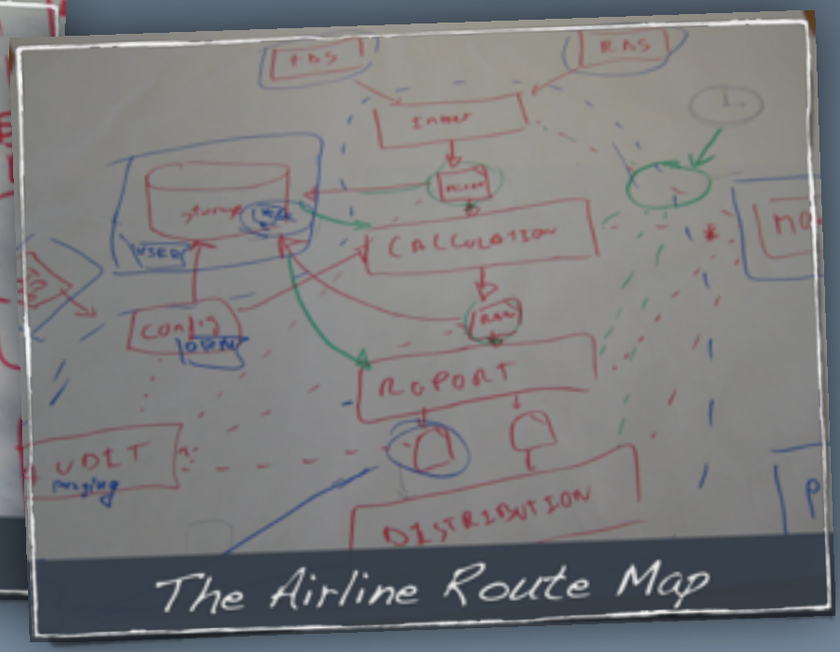
…but not our software!



Boxes & no Lines



Eh?



The Airline Route Map

Moving fast in the same direction requires good communication

Notation

Titles

Short and meaningful, numbered if diagram order is important

Lines

Favour unidirectional arrows, add descriptive text 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, especially those related to the business/domain that you work in

Colour

Ensure that colour coding is made explicit; watch out for colour-blindness and black/white printers

Orientation

Most important thing in the middle; be consistent across diagrams

Shapes

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

Keys

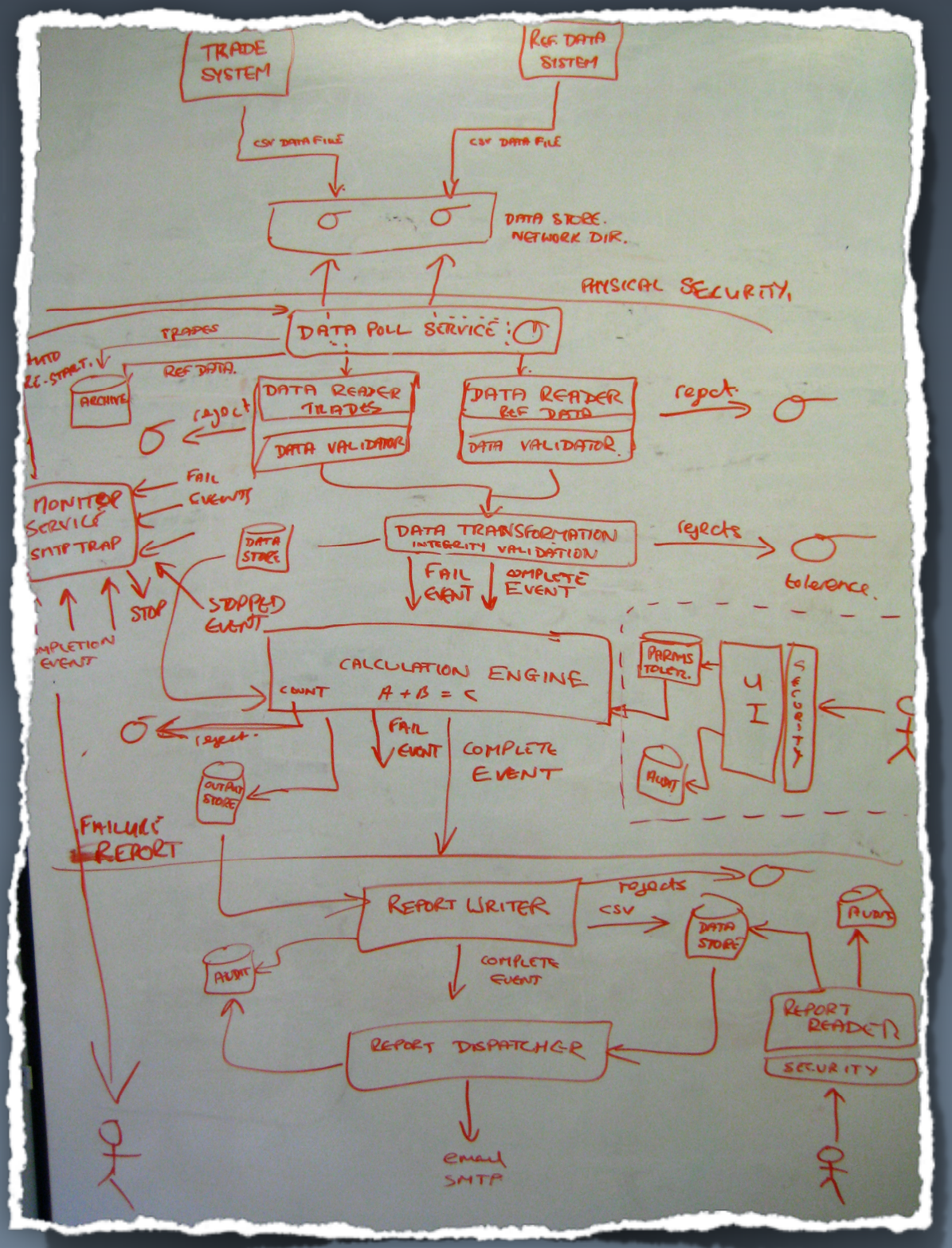
Explain shapes, lines, colours, borders, acronyms, etc

Responsibilities

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

Some notation tips…

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.

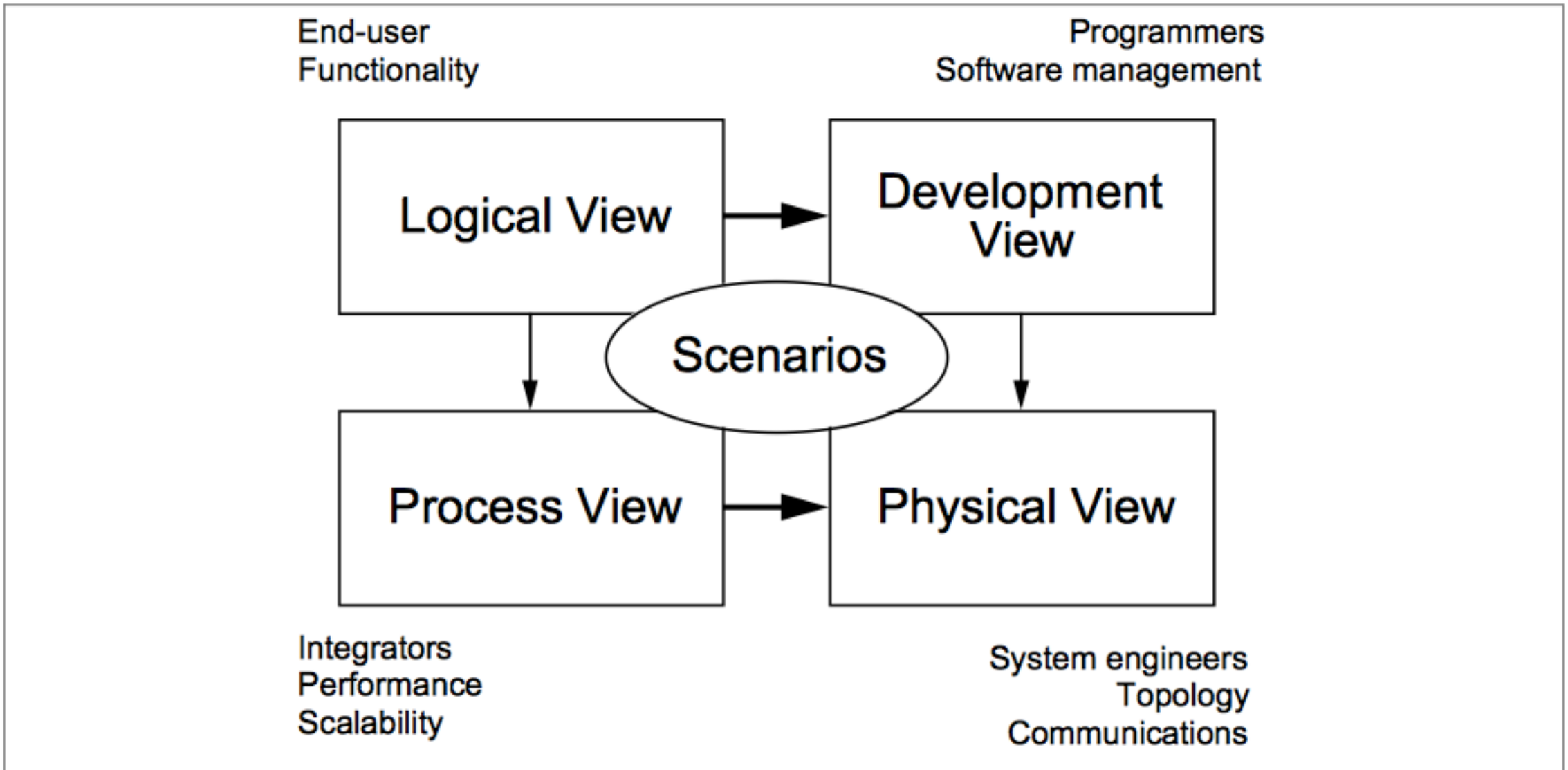
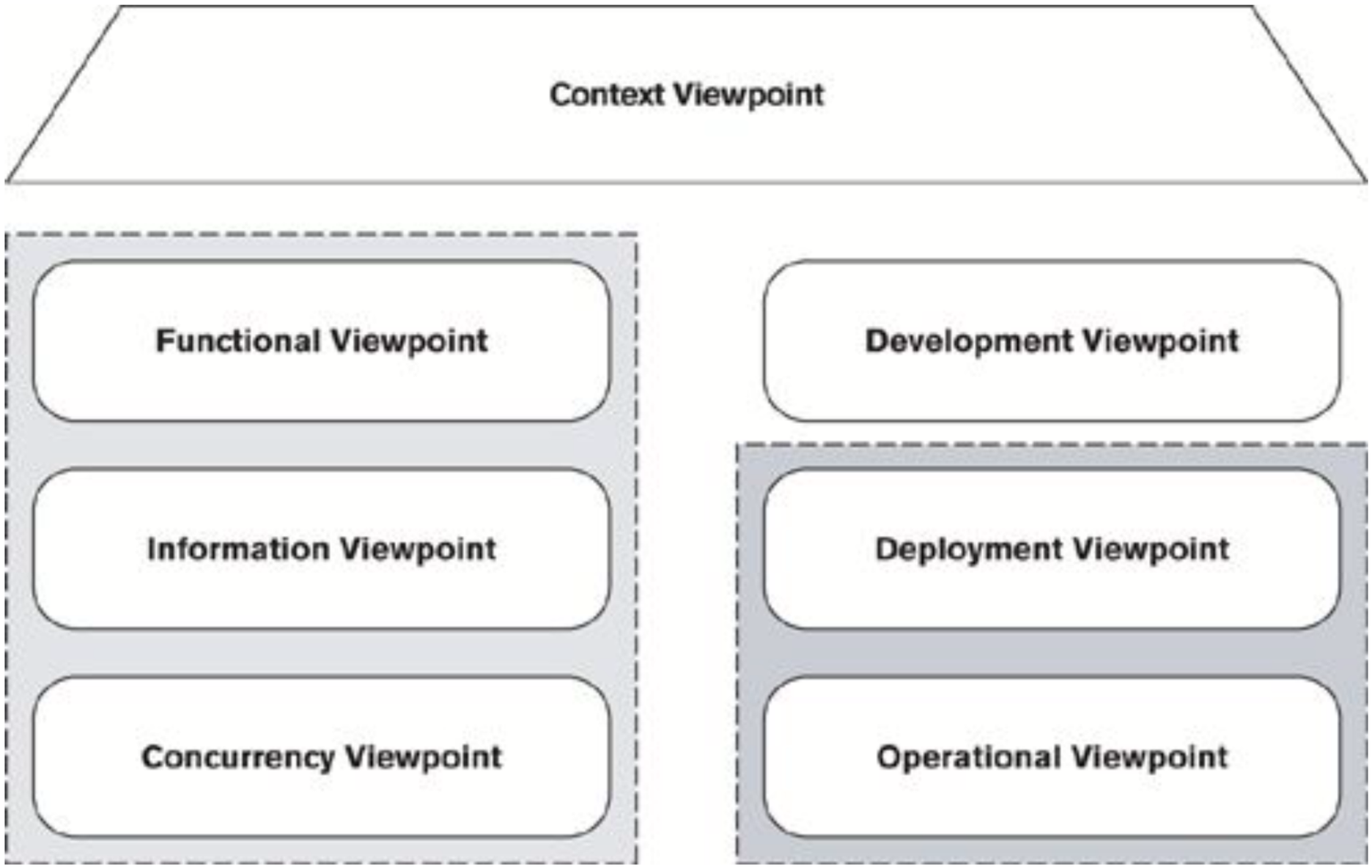
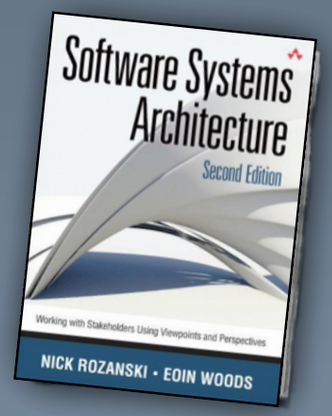


Figure 1 — The "4+1" view model

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.

Does everybody on the team understand the naming?

Conceptual vs Logical

Process vs Functional

Development vs Physical

Development vs Implementation

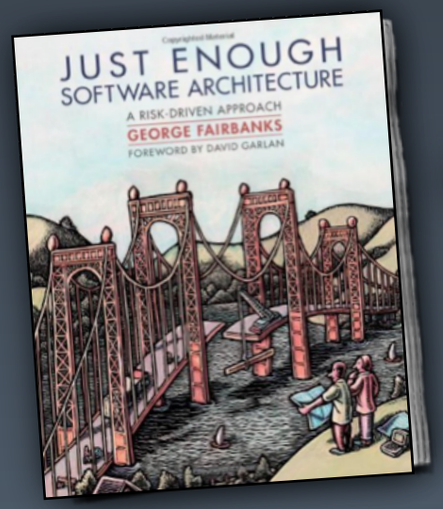
Physical vs Implementation

Physical vs Deployment

Why is there a

separation

between the logical and development views?



“the model-code gap”

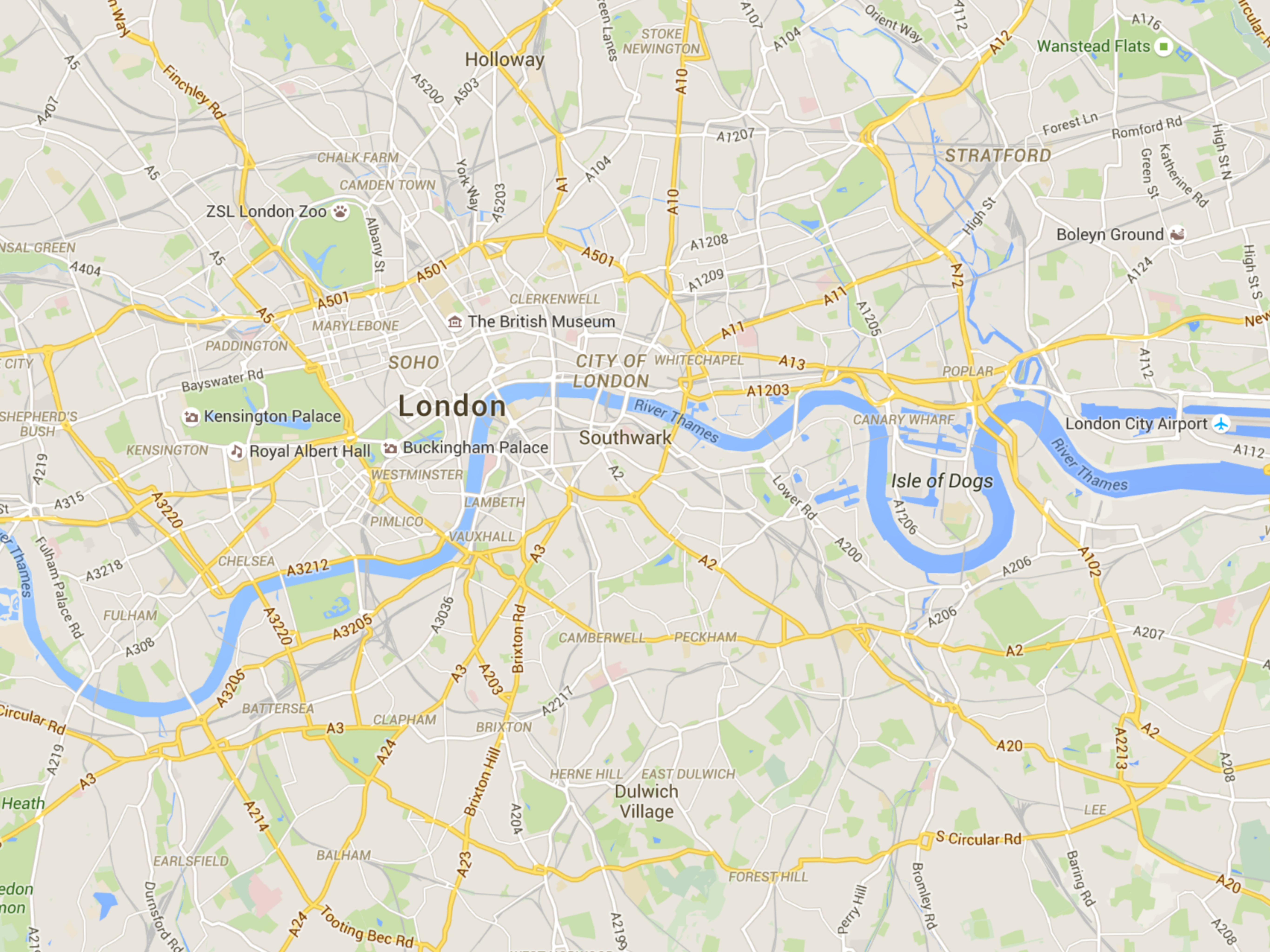
Model-code gap. Your architecture models and your source code will not show the same things. The difference between them is the *model-code gap*. Your architecture models include some abstract concepts, like components, that your programming language does not, but could. Beyond that, architecture models 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.

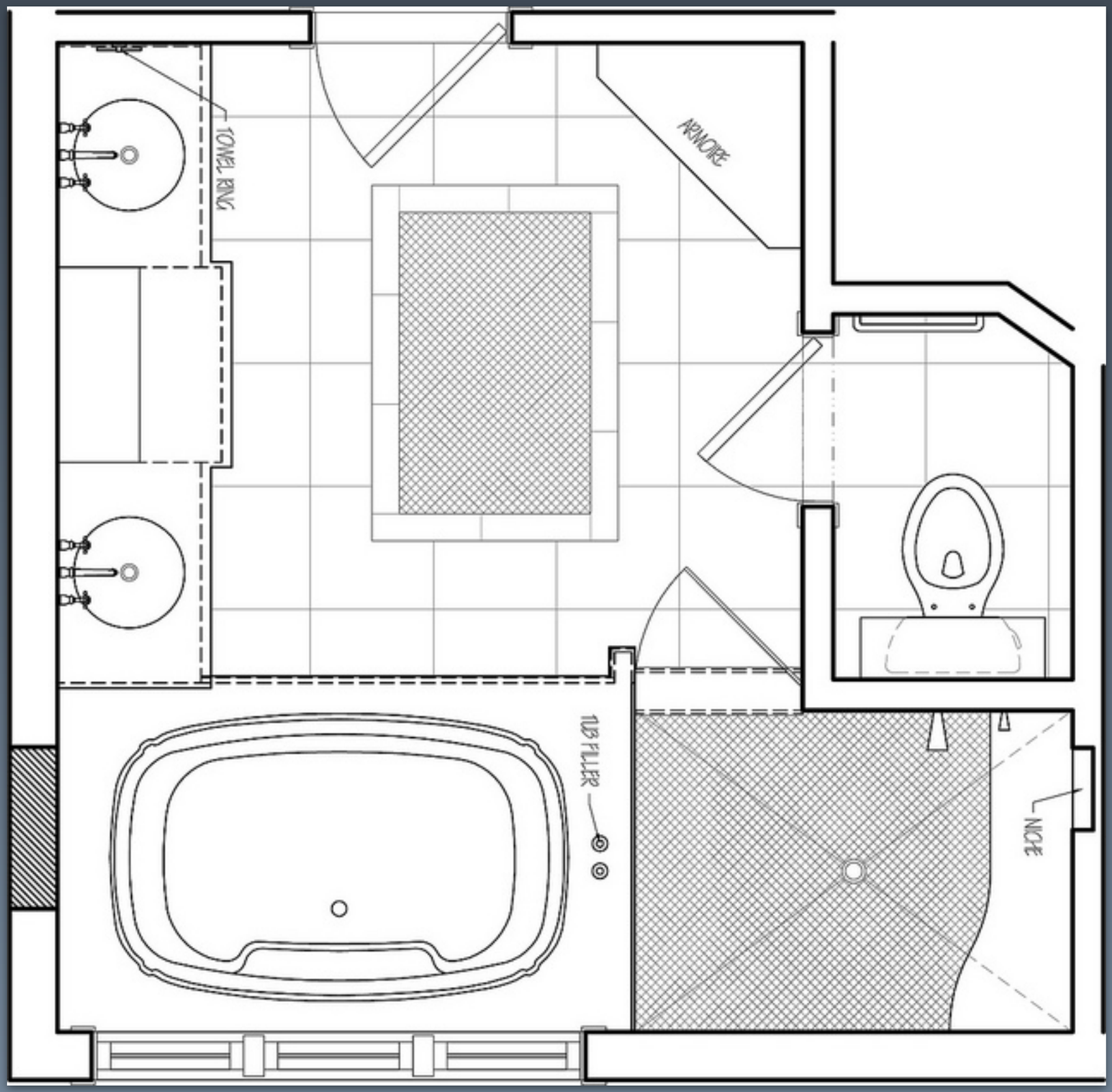
Upon learning about the model-code gap, your first instinct may be to avoid it. But reflecting on the origins of the gap gives little hope of a general solution in the short term: architecture models help you reason about complexity and scale because they are abstract and intensional; source code executes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

Do the diagrams reflect the code?

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



Holloway
STOKE NEWINGTON
Wanstead Flats
STRATFORD
CHALK FARM
CAMDEN TOWN
ZSL London Zoo
Boleyn Ground
CLERKENWELL
The British Museum
MARYLEBONE
PADDINGTON
SOHO
CITY OF LONDON
WHITECHAPEL
POPLAR
London
Kensington Palace
River Thames
CANARY WHARF
London City Airport
SHEPHERD'S BUSH
Southwark
KENSINGTON
Royal Albert Hall
Buckingham Palace
WESTMINSTER
Isle of Dogs
LAMBETH
PIMLICO
VAUXHALL
CHELSEA
FULHAM
CAMBERWELL
PECKHAM
BATTERSEA
CLAPHAM
BRIXTON
HERNE HILL
EAST DULWICH
Dulwich Village
LEE
S Circular Rd
BALHAM
EARLSFIELD
FOREST HILL
Tooting Bec Rd
Finchley Rd
Orient Way
Forest Ln
Romford Rd
Bayswater Rd
Fulham Palace Rd
Brixton Rd
Brixton Hill
Lower Rd
High St
Albany St
York Way
Katherine Rd
Green St
Baring Rd
Bromley Rd
Perry Hill
Durnsford Rd



Floor plans

Circuit diagrams
(pictorial or schematic)

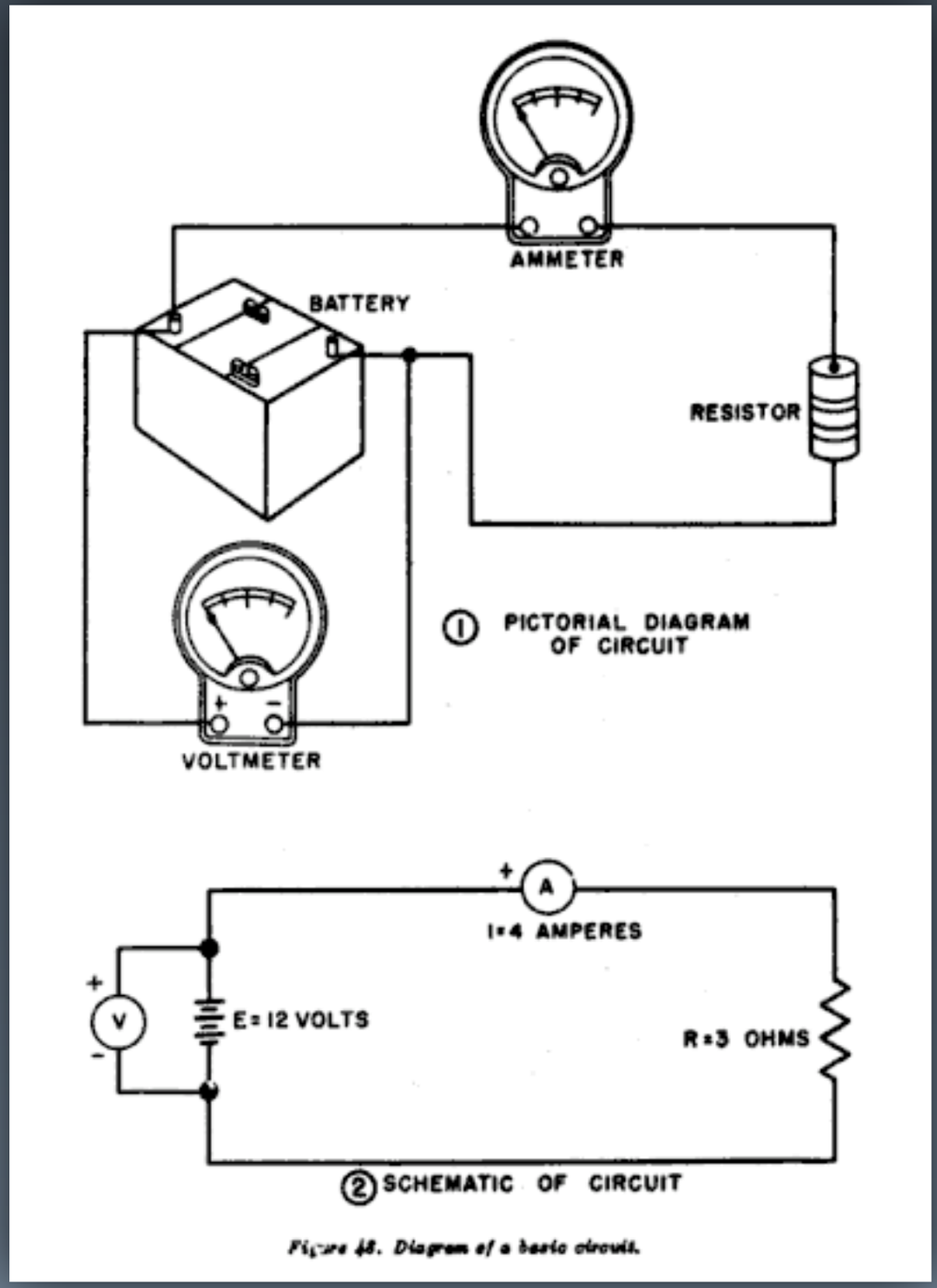
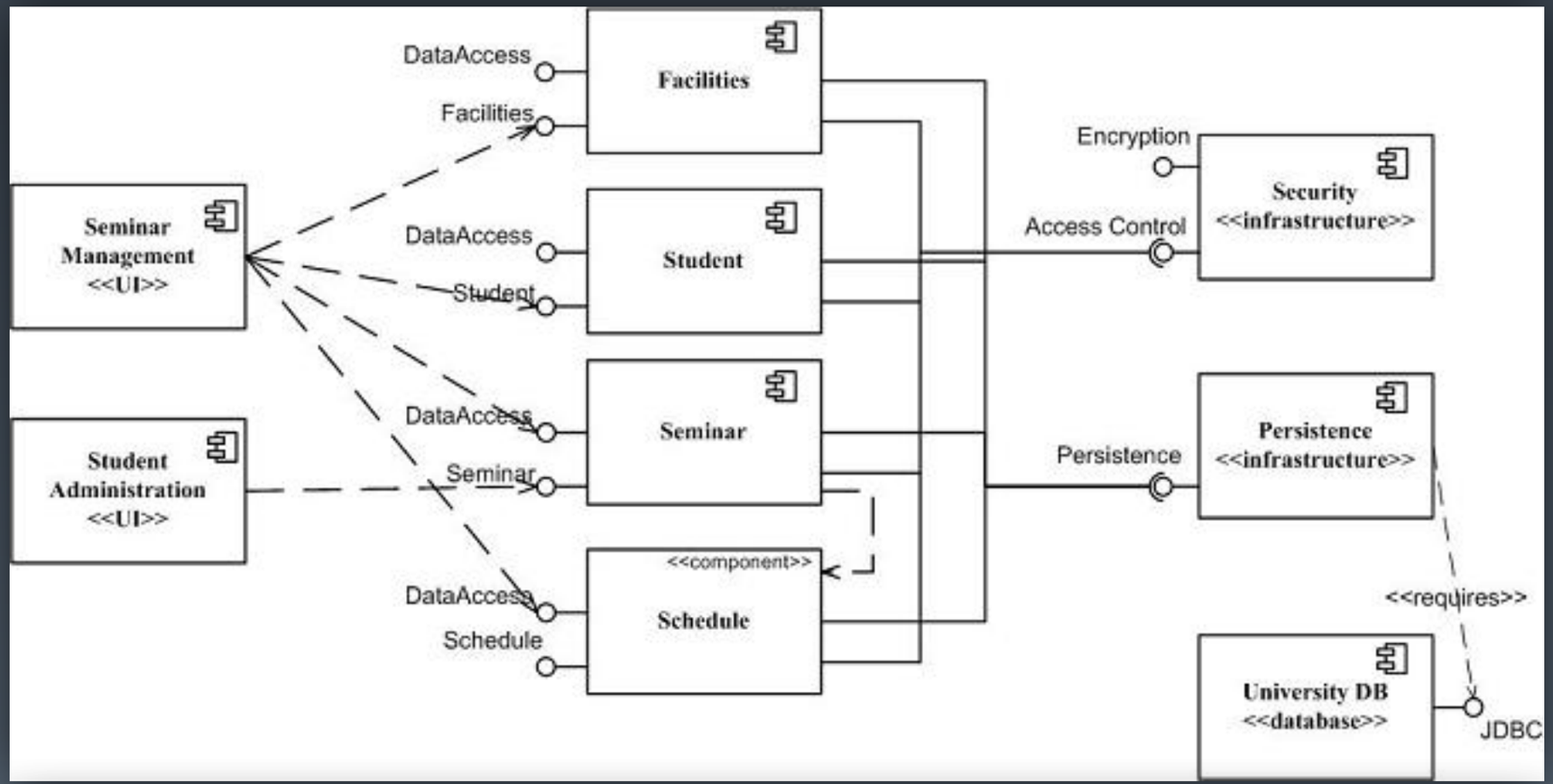
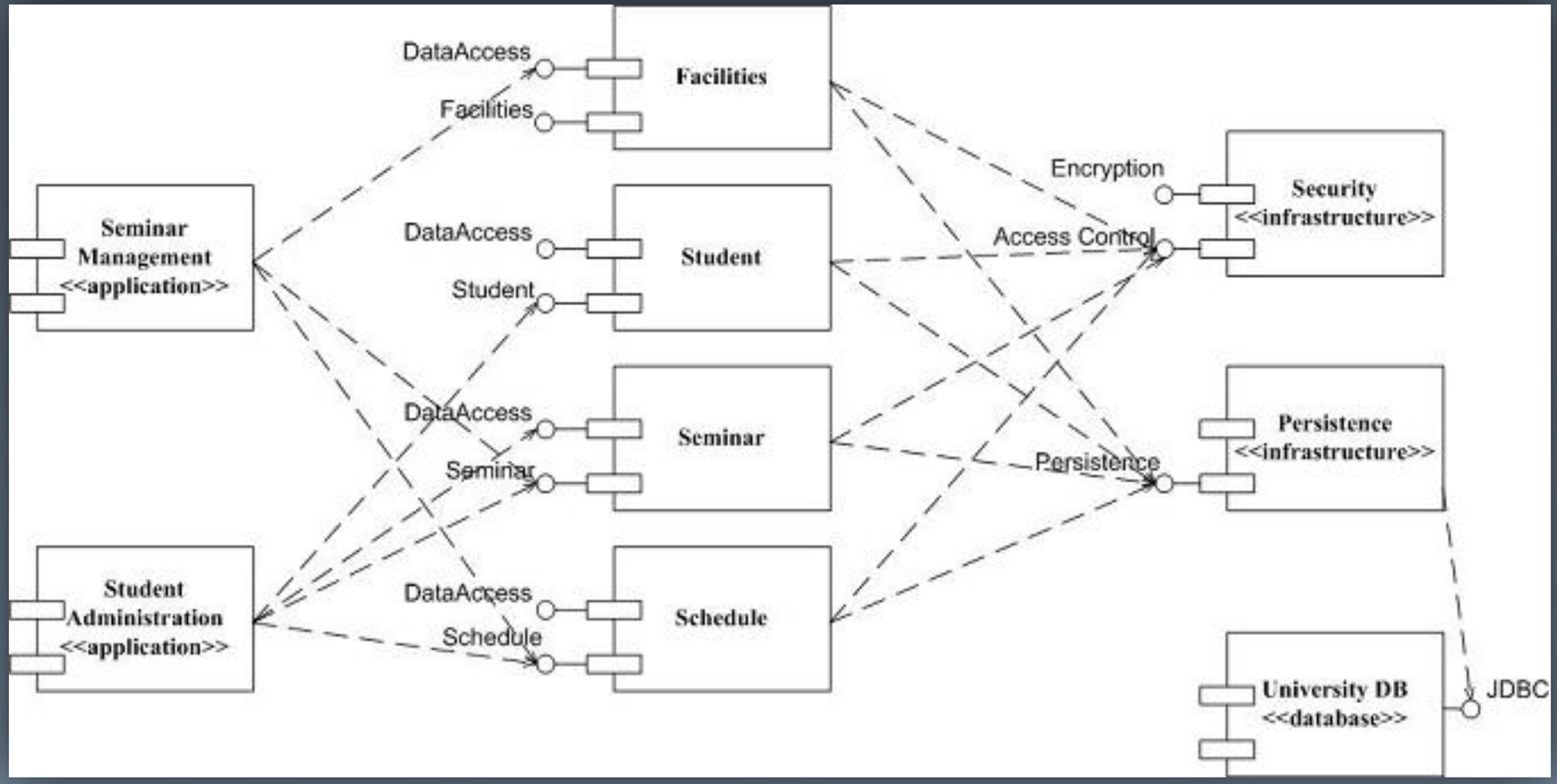
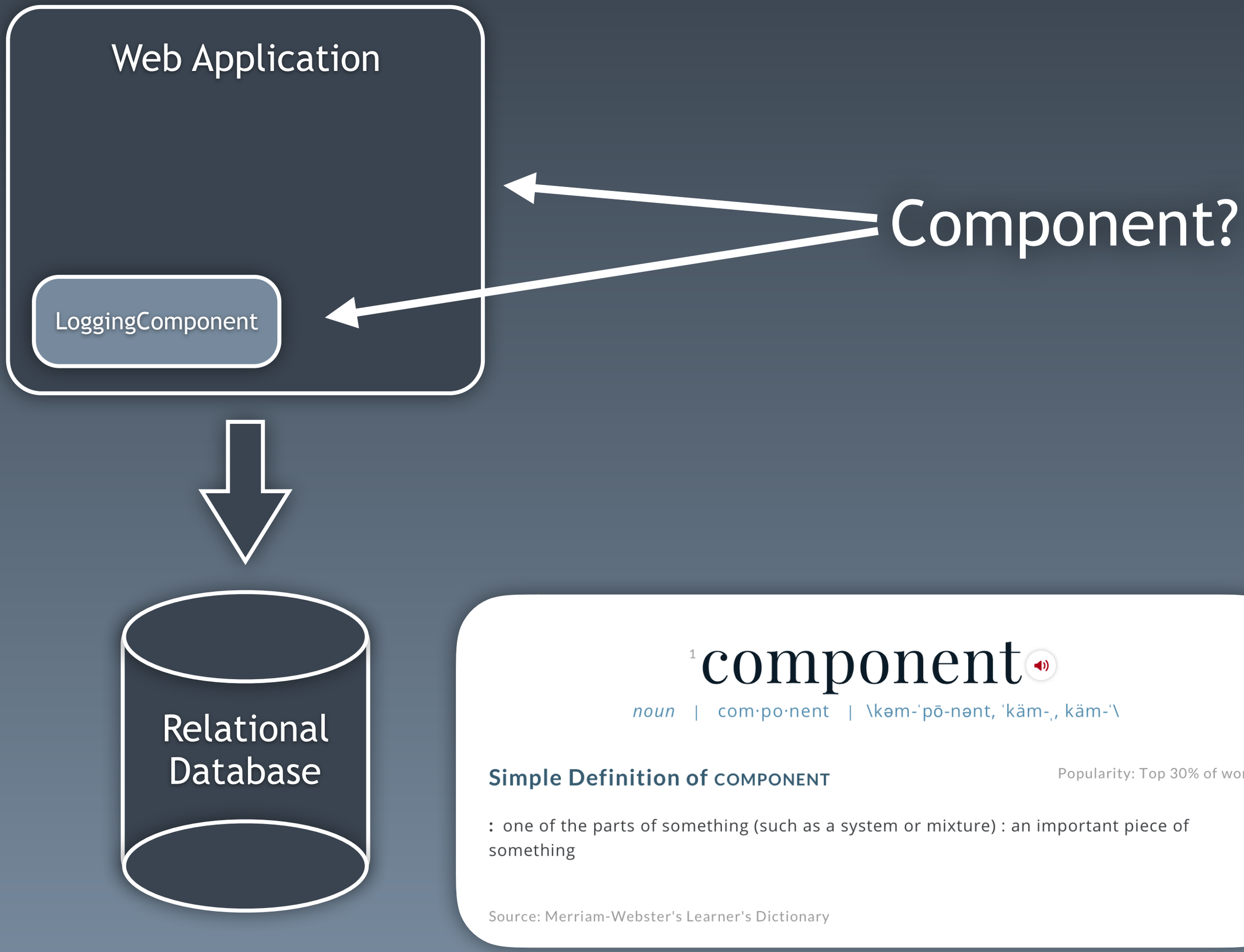


Figure 48. Diagram of a basic circuit.





http://agilemodeling.com/artifacts/componentDiagram.htm

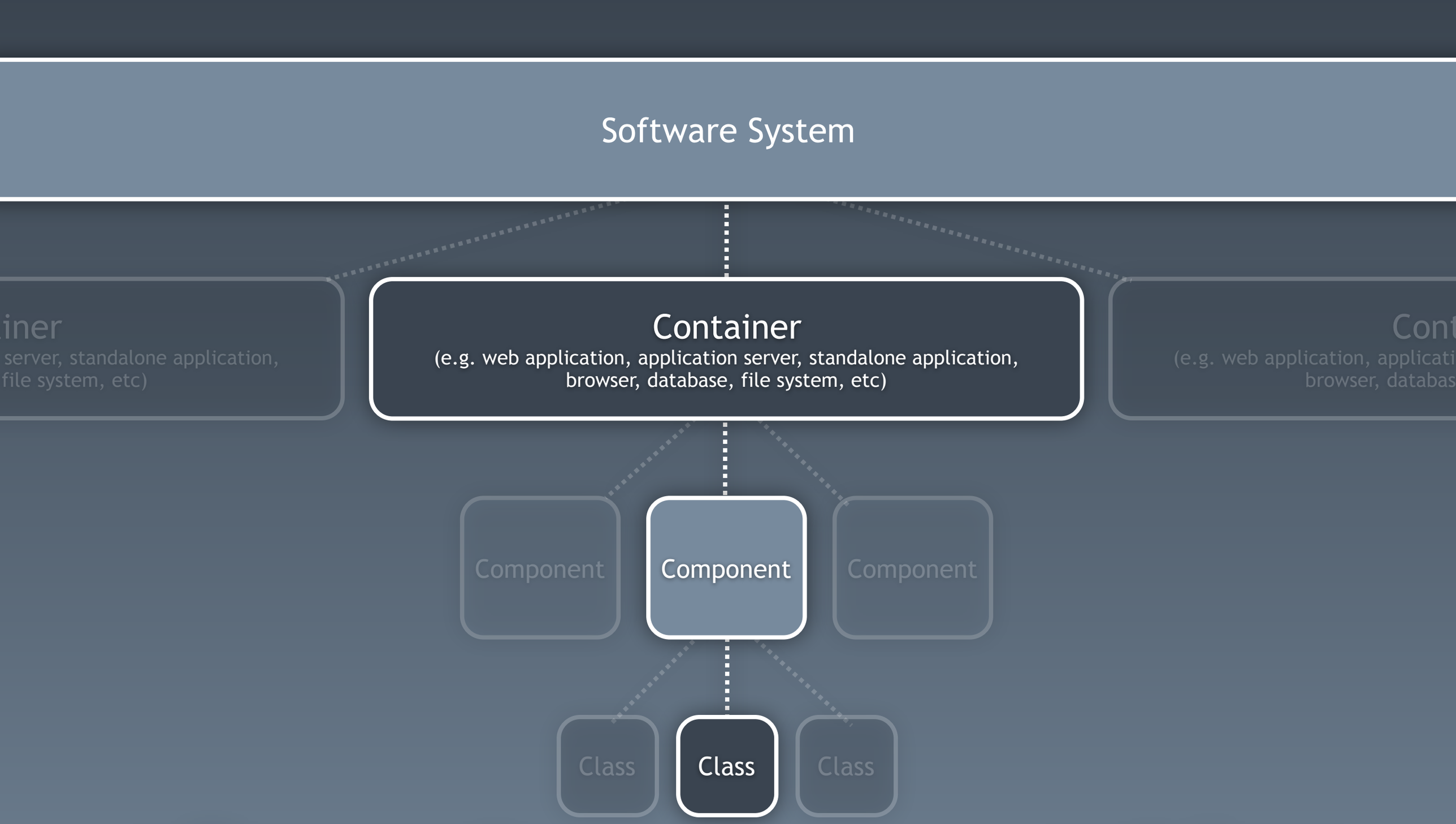

Web Application
LoggingComponent
Component?
Relational Database
1 component
noun | com·po·nent | \kəm-ˈpō-nənt, ˈkäm-ˌ, käm-ˈ\
Simple Definition of COMPONENT
Popularity: Top 30% of words
: one of the parts of something (such as a system or mixture) : an important piece of something
Source: Merriam-Webster's Learner's Dictionary

Ubiquitous language

A common set of abstractions

is more important than

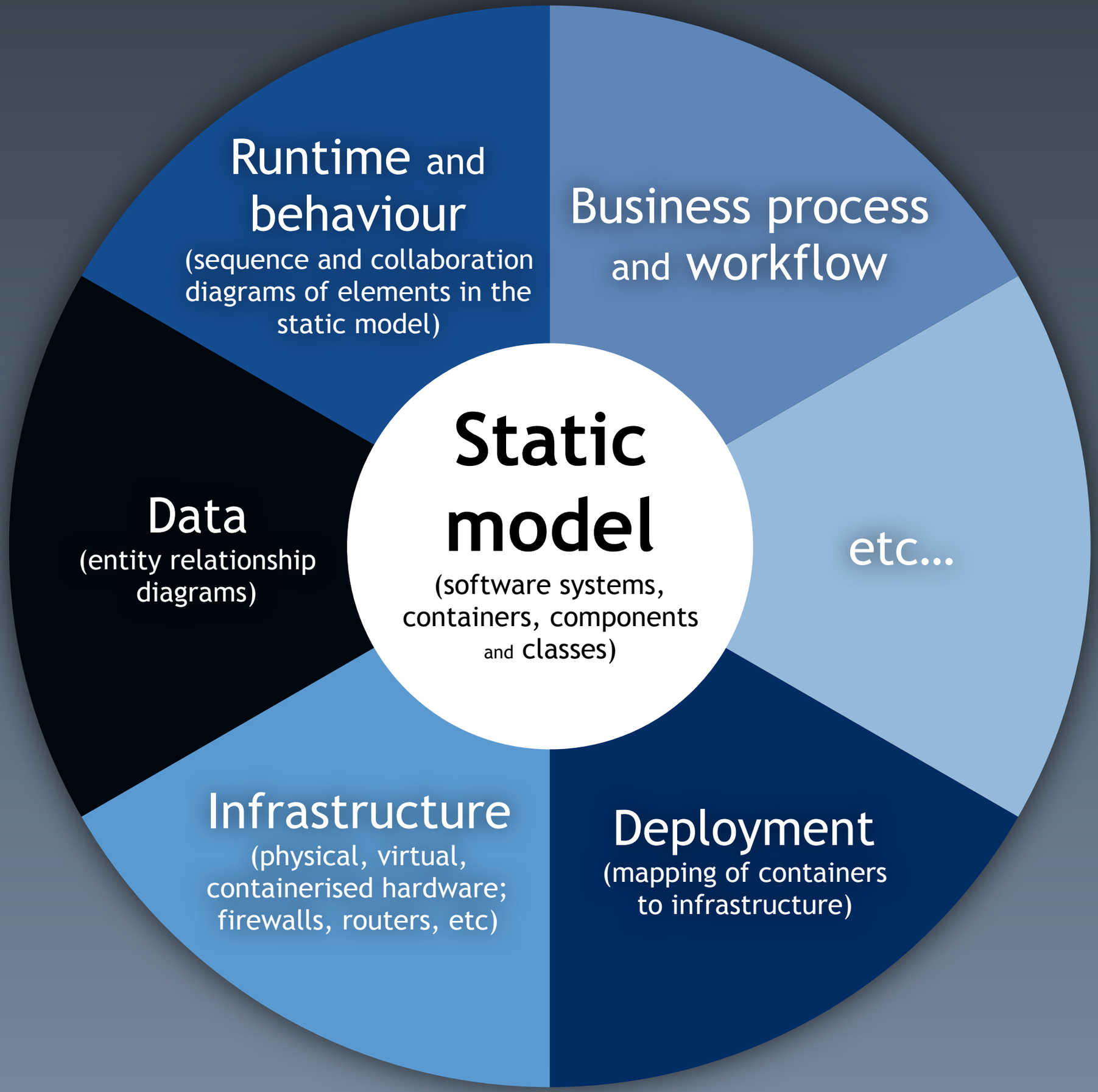
a common notation



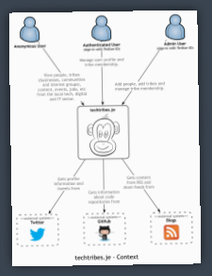
A **software system** is made up of one or more **containers**,

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

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

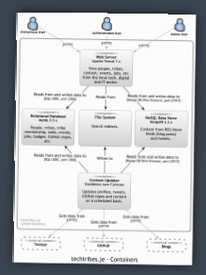

Runtime and behaviour
(sequence and collaboration diagrams of elements in the static model)
Business process and workflow
Static model
(software systems, containers, components and classes)
etc...
Data
(entity relationship diagrams)
Infrastructure
(physical, virtual, containerised hardware; firewalls,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

The C4 model



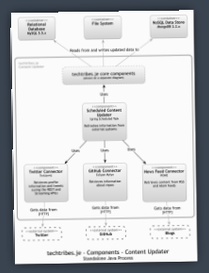
System Context

The system plus users and system dependencies



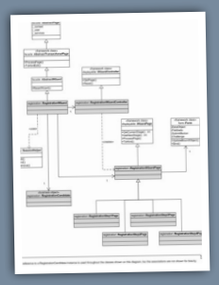
Containers

The overall shape of the architecture and technology choices



Components

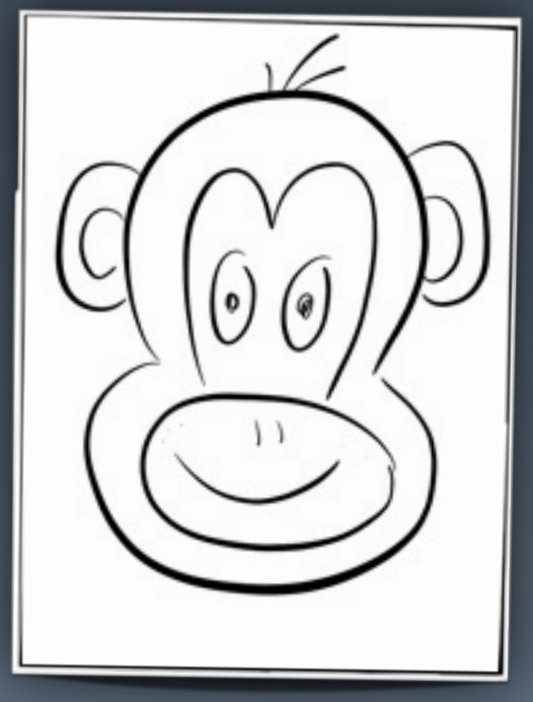
Components and their interactions within a container



Classes (or Code)

Component implementation details

techtribes.je

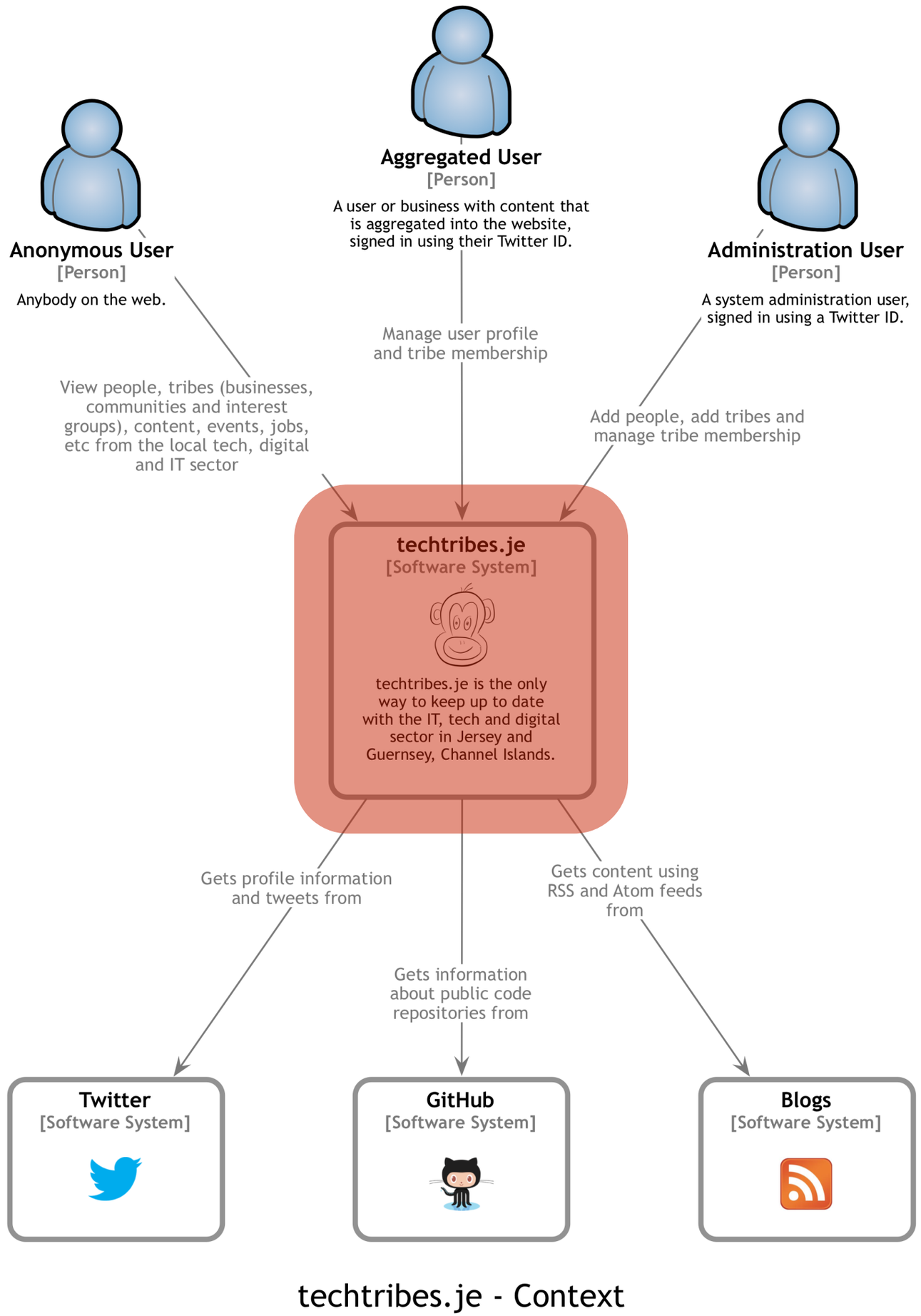


Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)



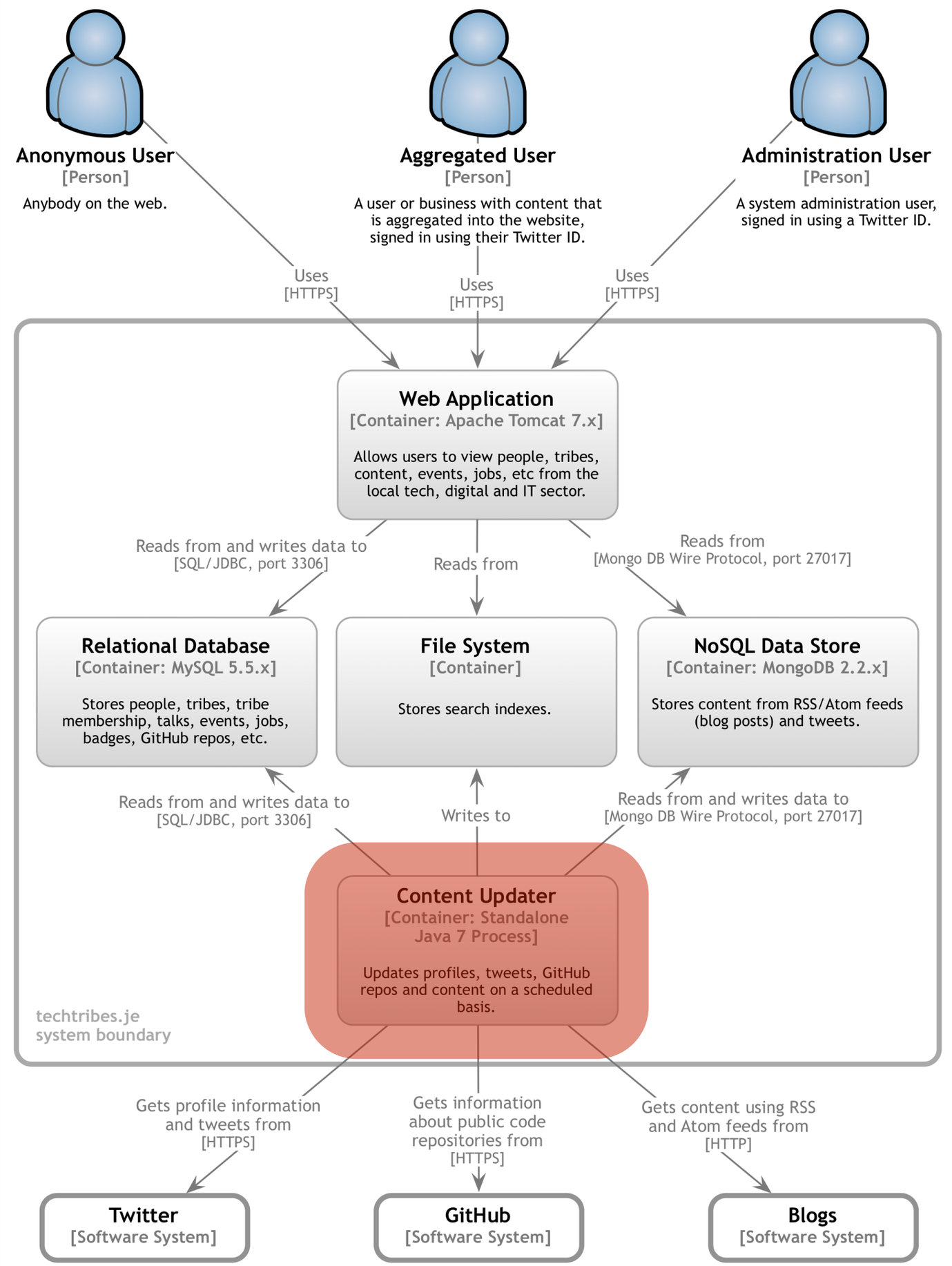
techtribes.je - Context

Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)



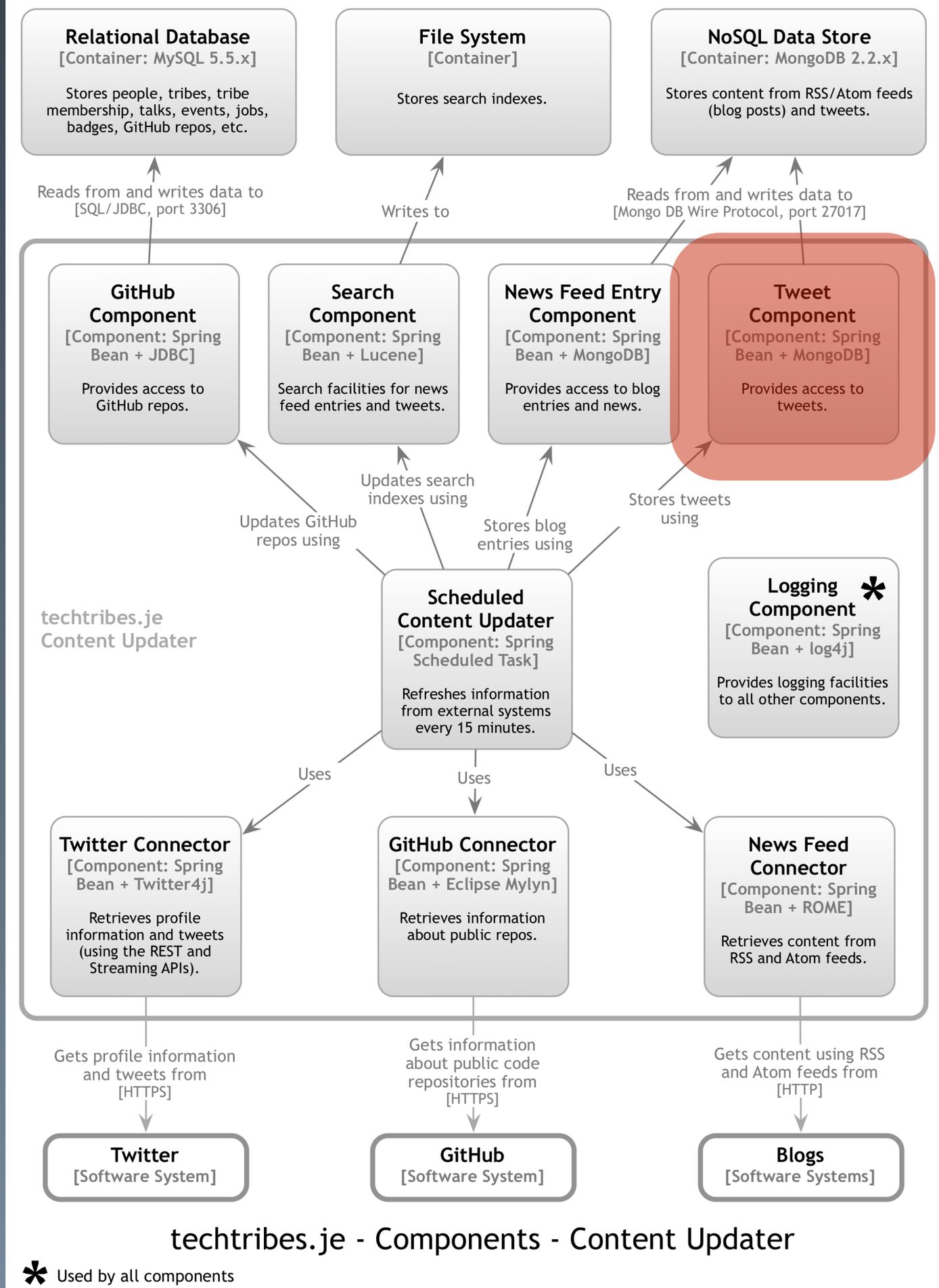
techtribes.je - Containers

Context diagram
(level 1)

Container diagram
(level 2)

Component diagram
(level 3)

Class diagram
(level 4)



techtribes.je - Components - Content Updater

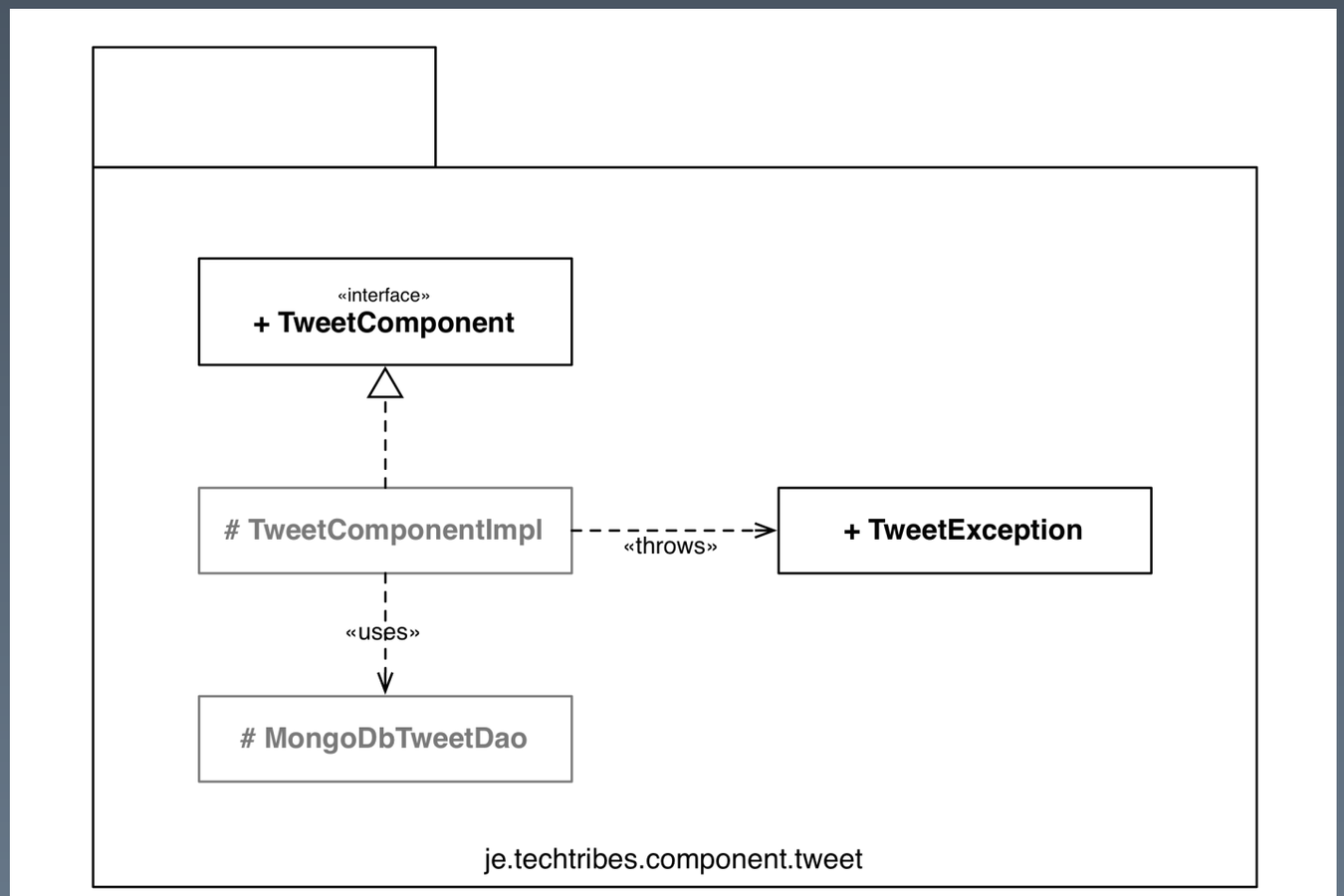
* Used by all components

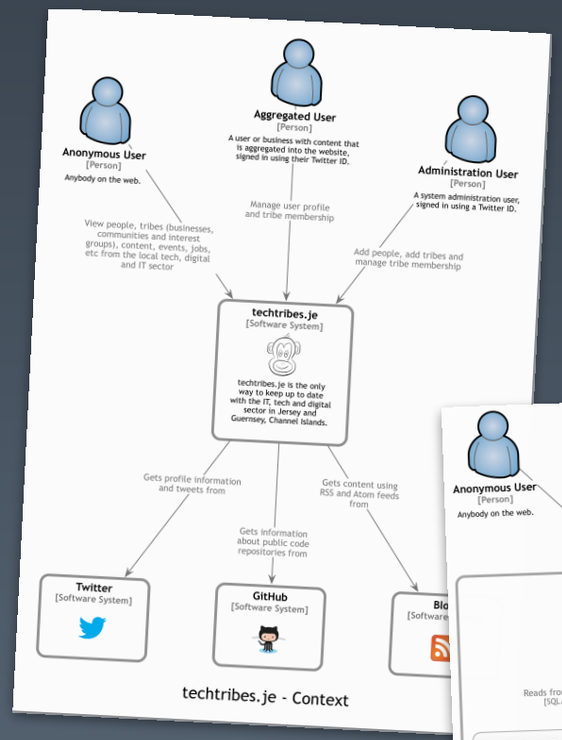
Context diagram
(level 1)

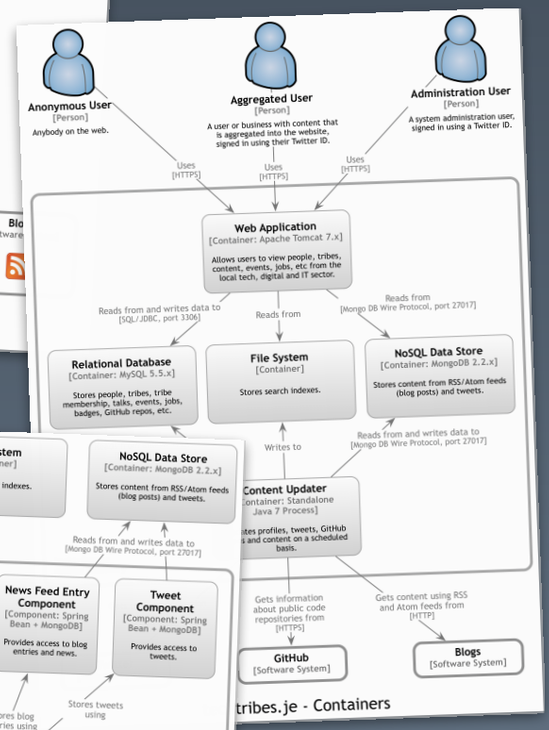
Container diagram
(level 2)

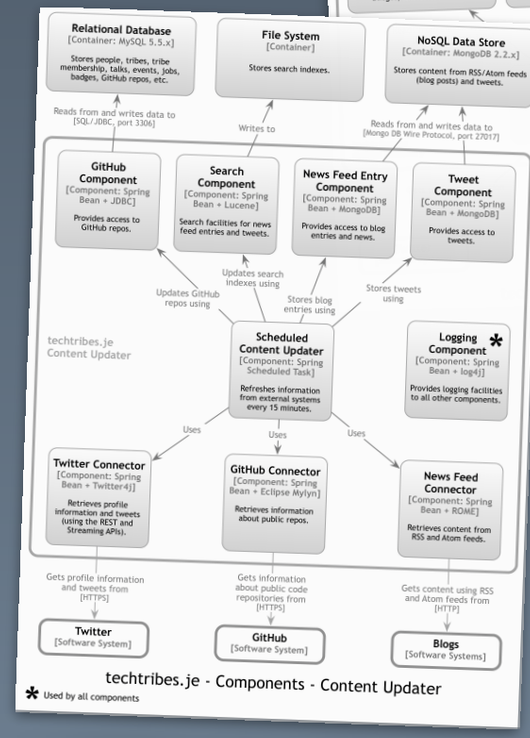
Component diagram
(level 3)

Class diagram
(level 4)









Diagrams are maps

that help a team navigate a complex codebase

Think about the target audience

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

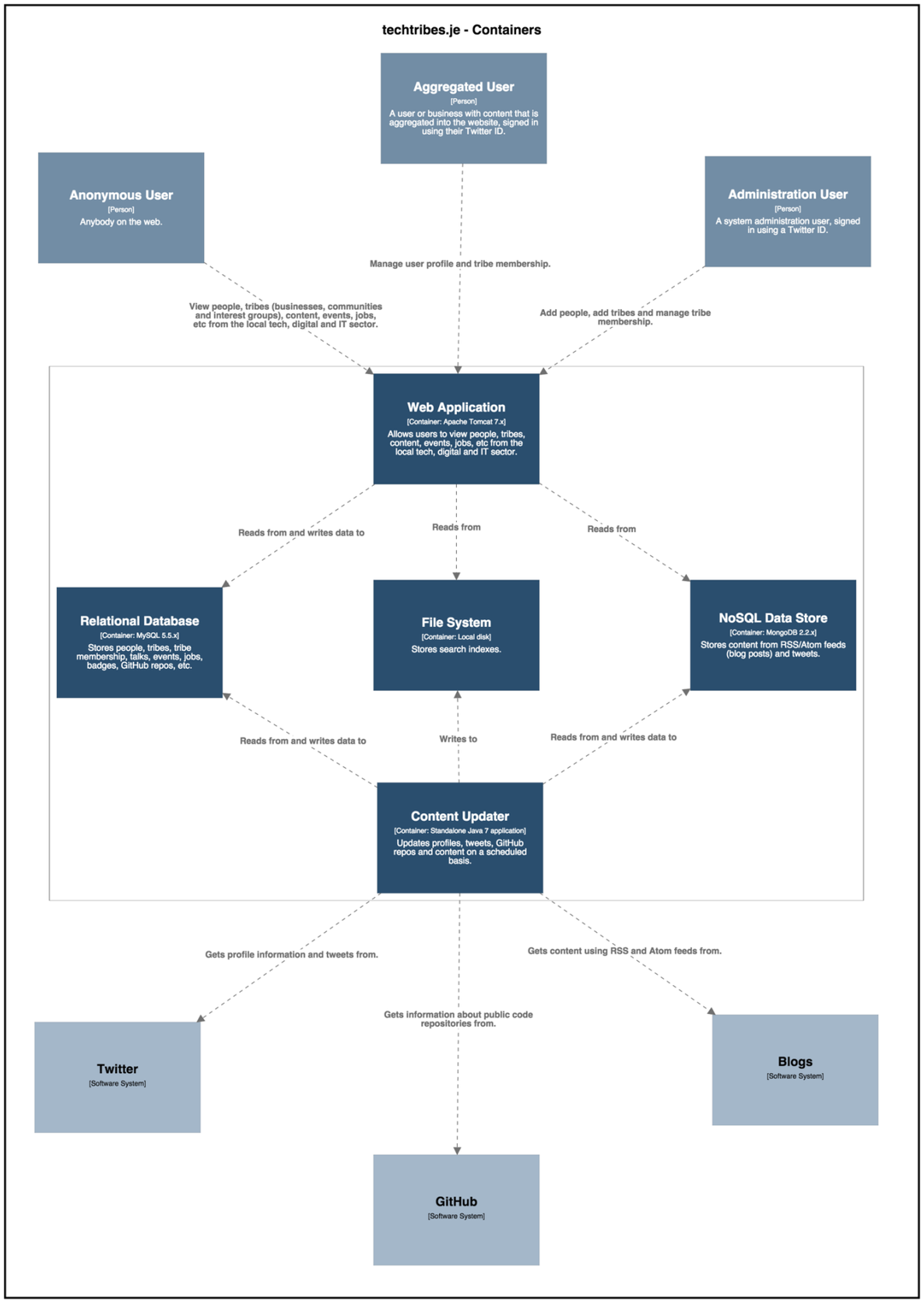
[Container: Apache Tomcat 7.x]

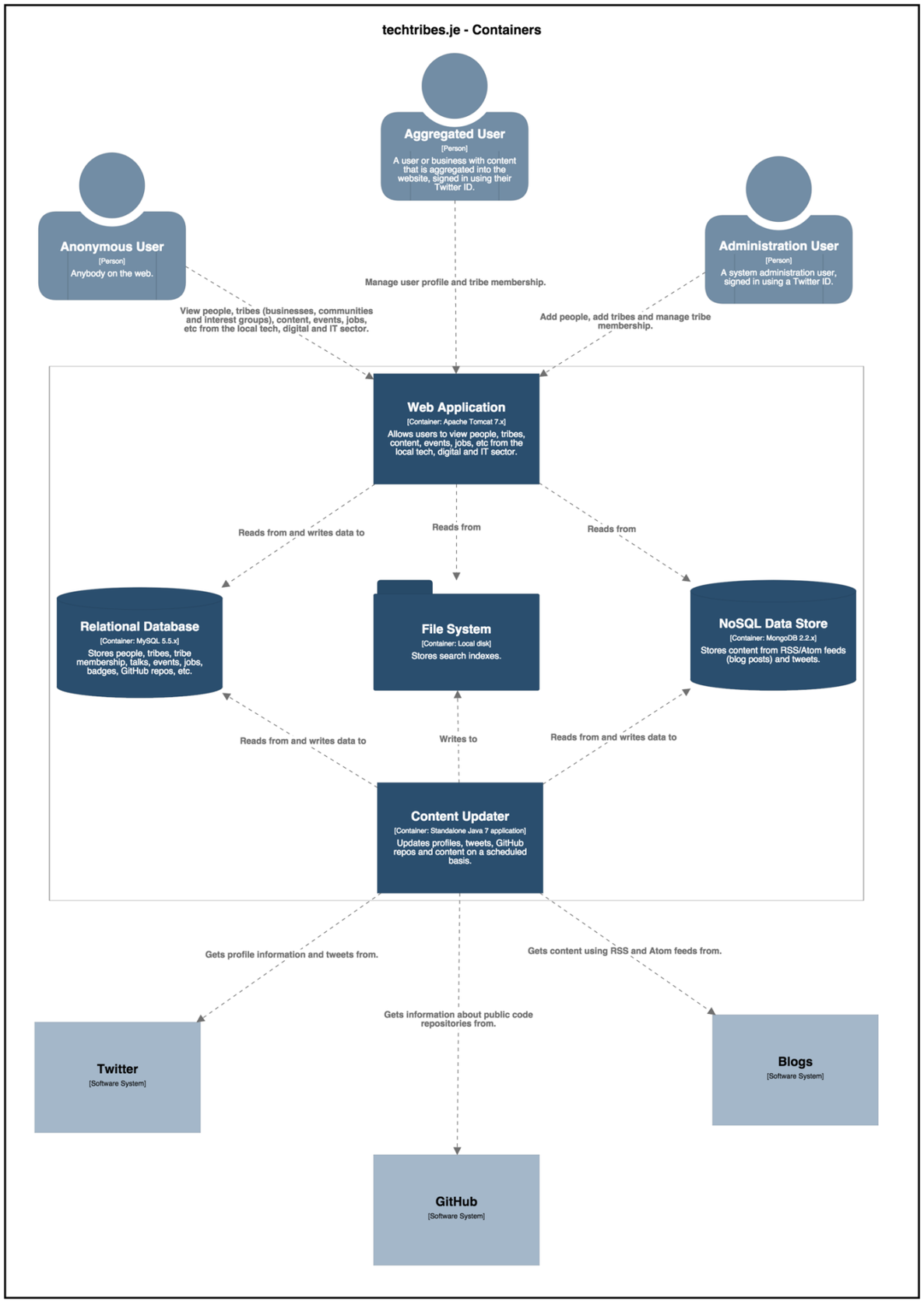
Allows users to view people, tribes, content, events, jobs, etc from the local tech, digital and IT sector.

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

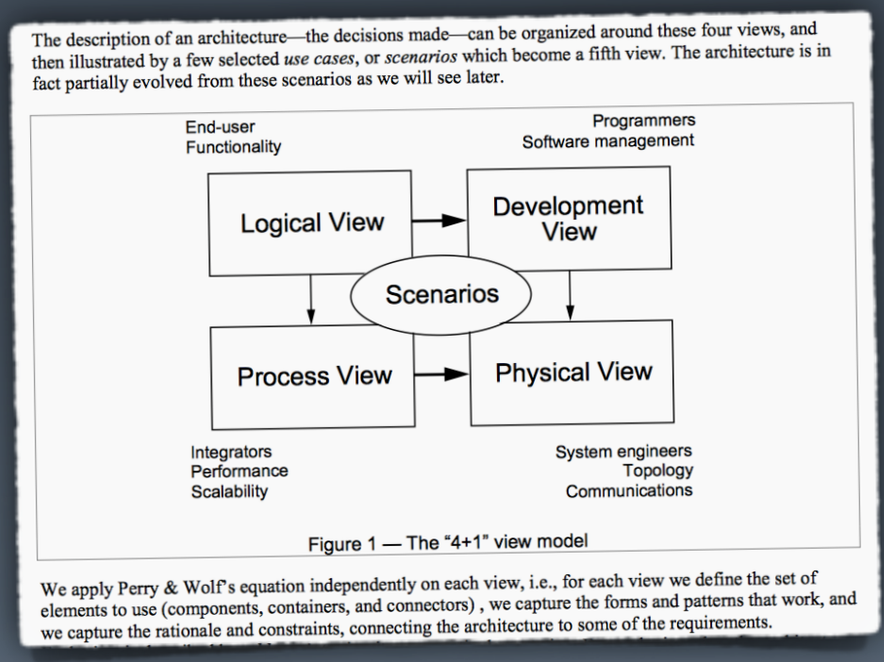
Sequence and collaboration diagrams

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.

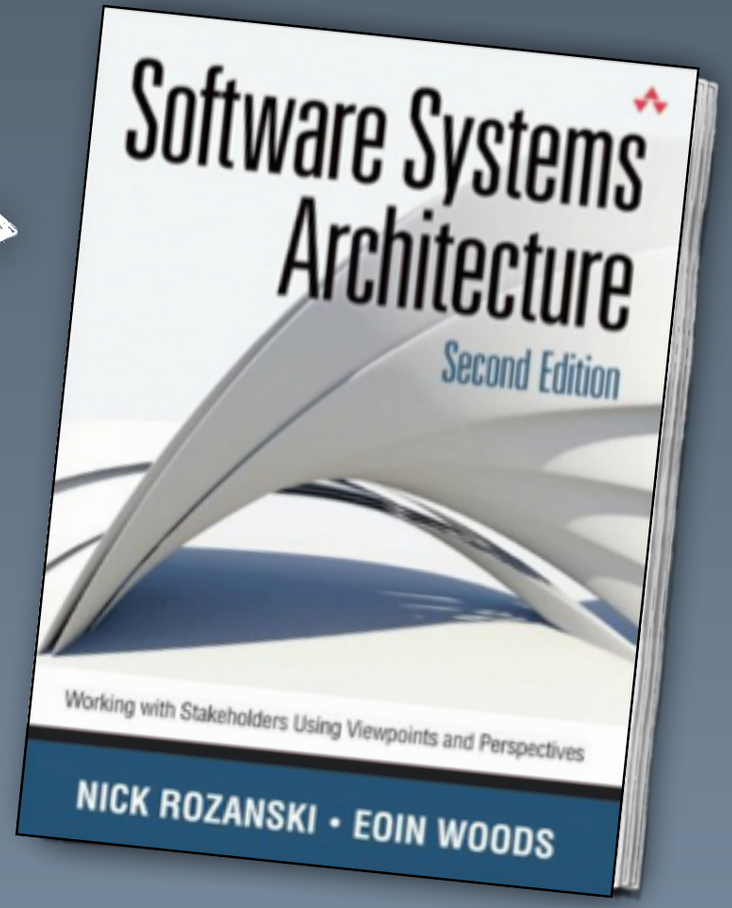
4+1 architectural view model

Philippe Kruchten

Software Systems Architecture

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

Nick Rozanski and Eoin Woods



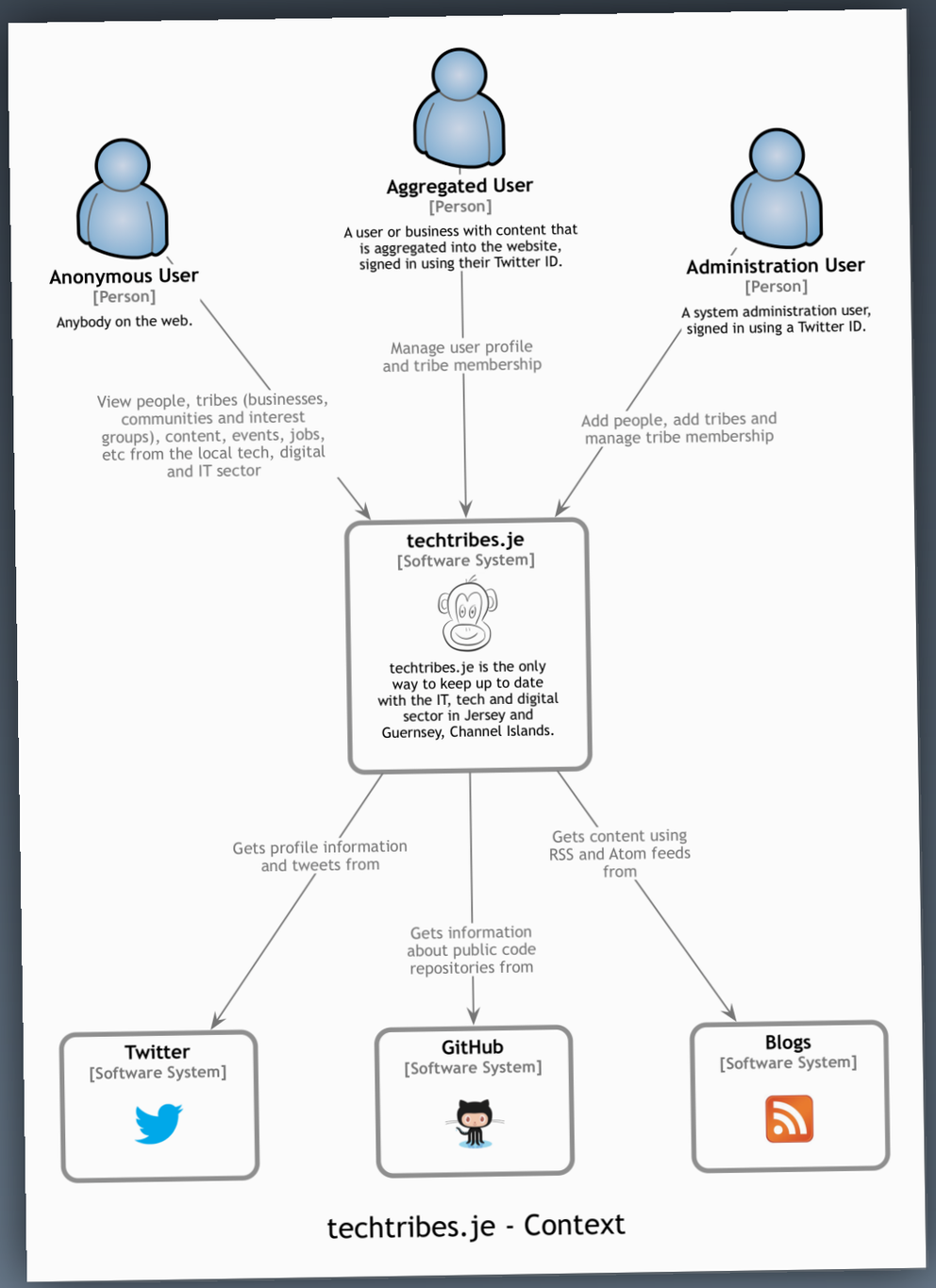
C4 is not a design process

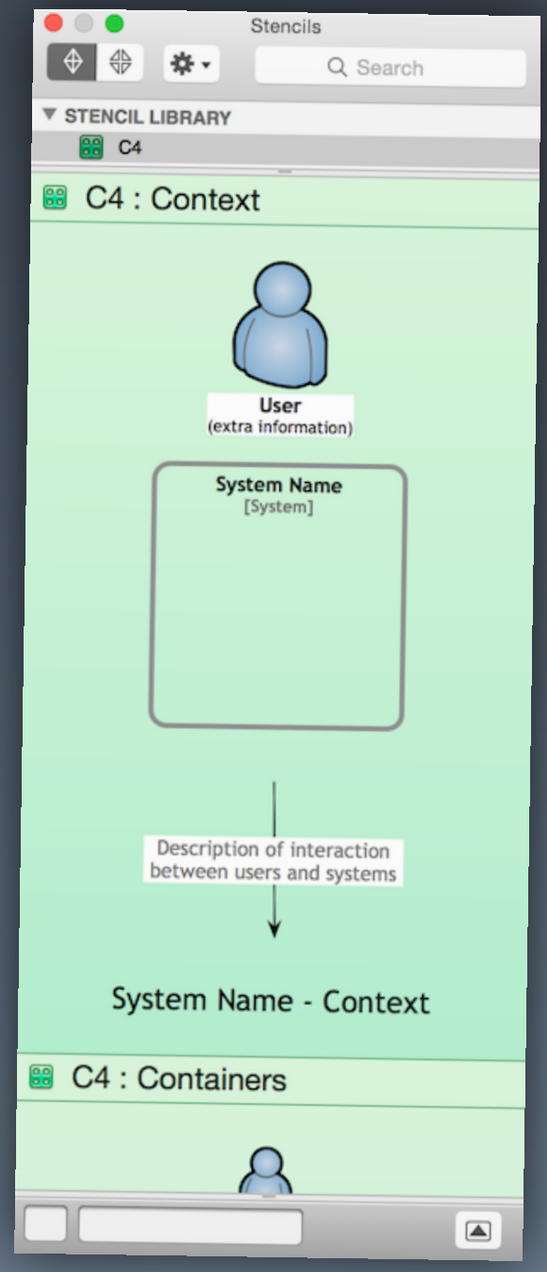
Up front design

vs

retrospectively drawing diagrams

Tooling





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

Simon Brown
@simonbrown

OmniGraffle, Keynote, Gliffy, LucidChart, Google Docs, Inkscape, draw.io, yEd … welcome to software dev in 2016!

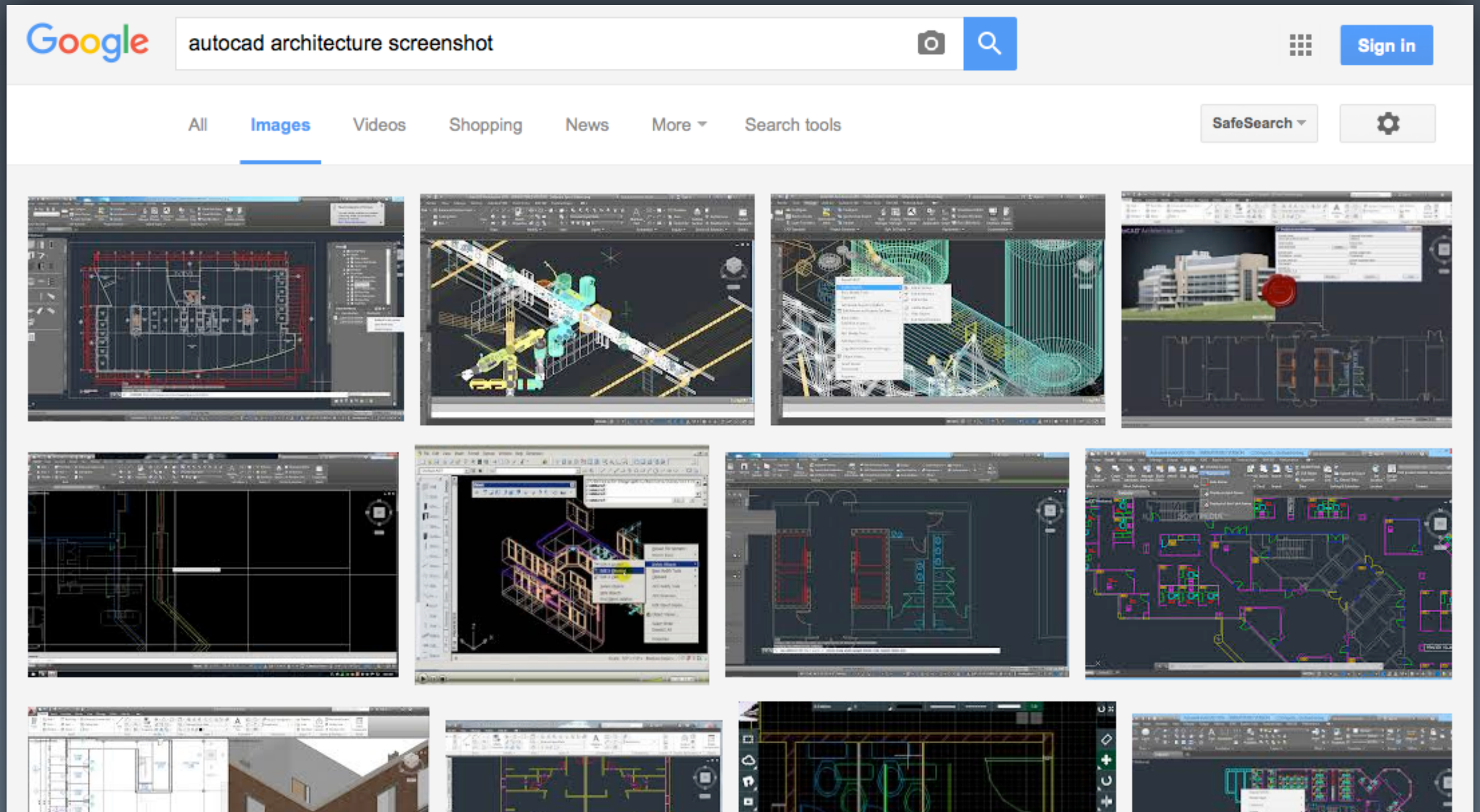
> **Kelly Sommers** @kellabyte
> What's a good Mac app for making architecture slides?

RETWEETS 5 LIKES 14

8:15 AM - 11 Jul 2016

5 14

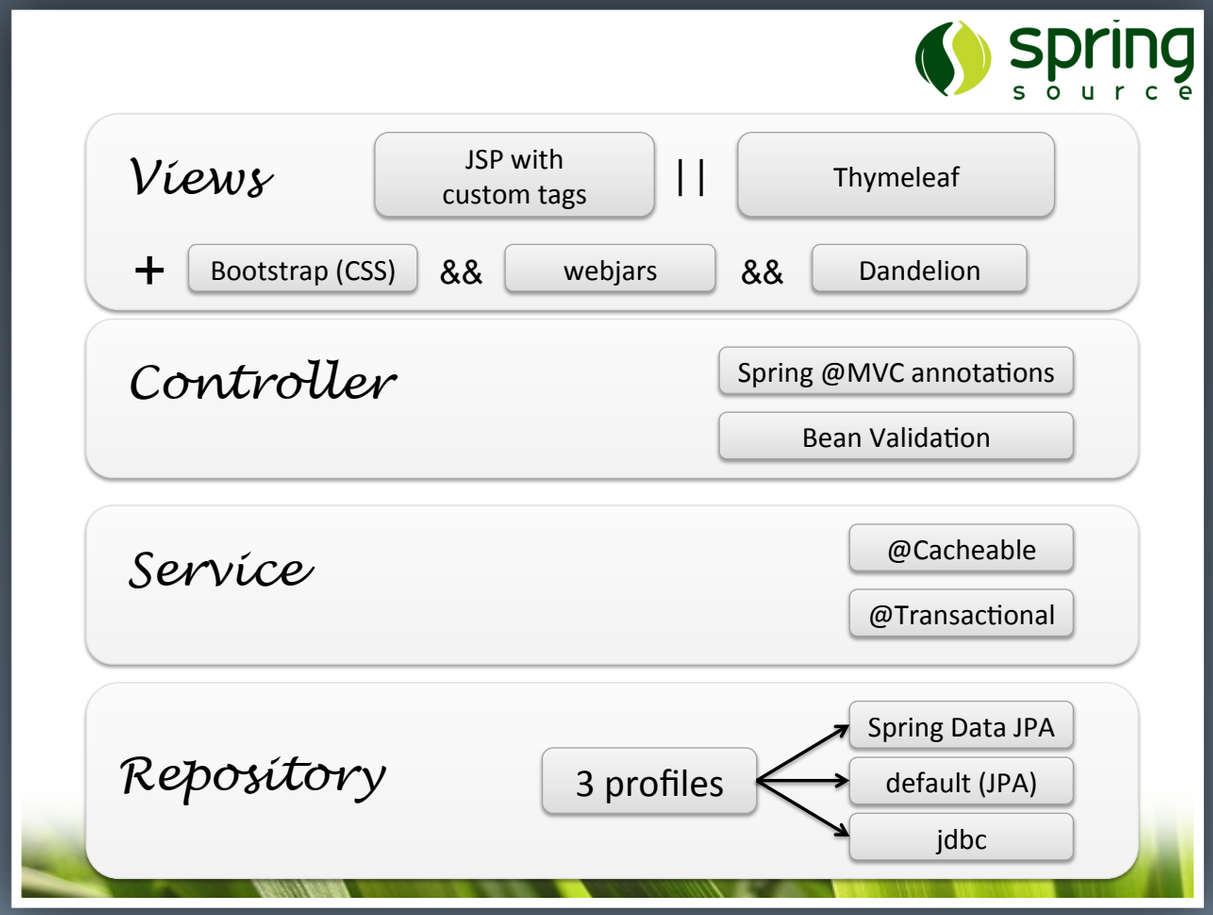
Do building architects use Microsoft Visio?



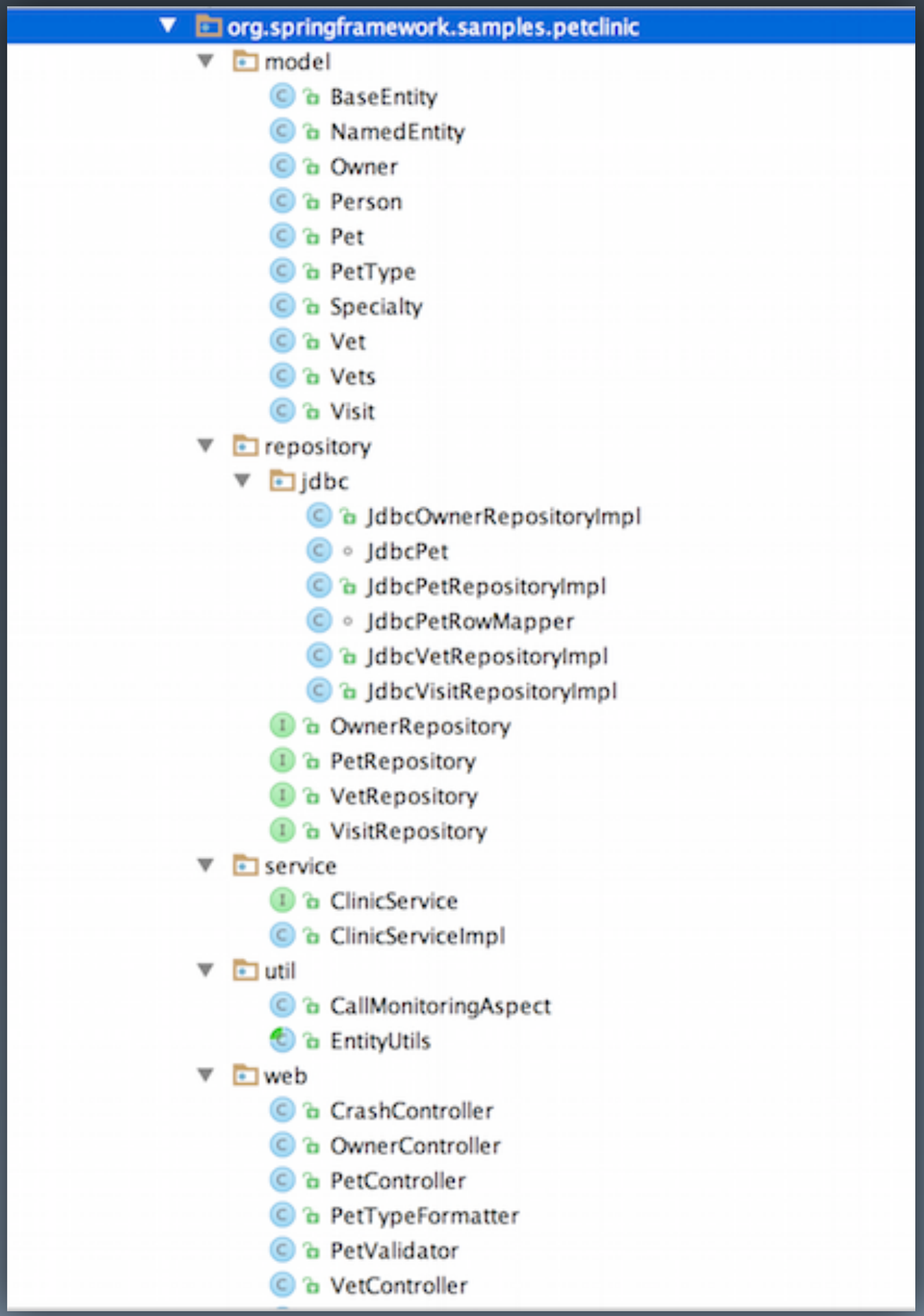
Sketches get out of date,
so why not
auto-generate
the diagrams?

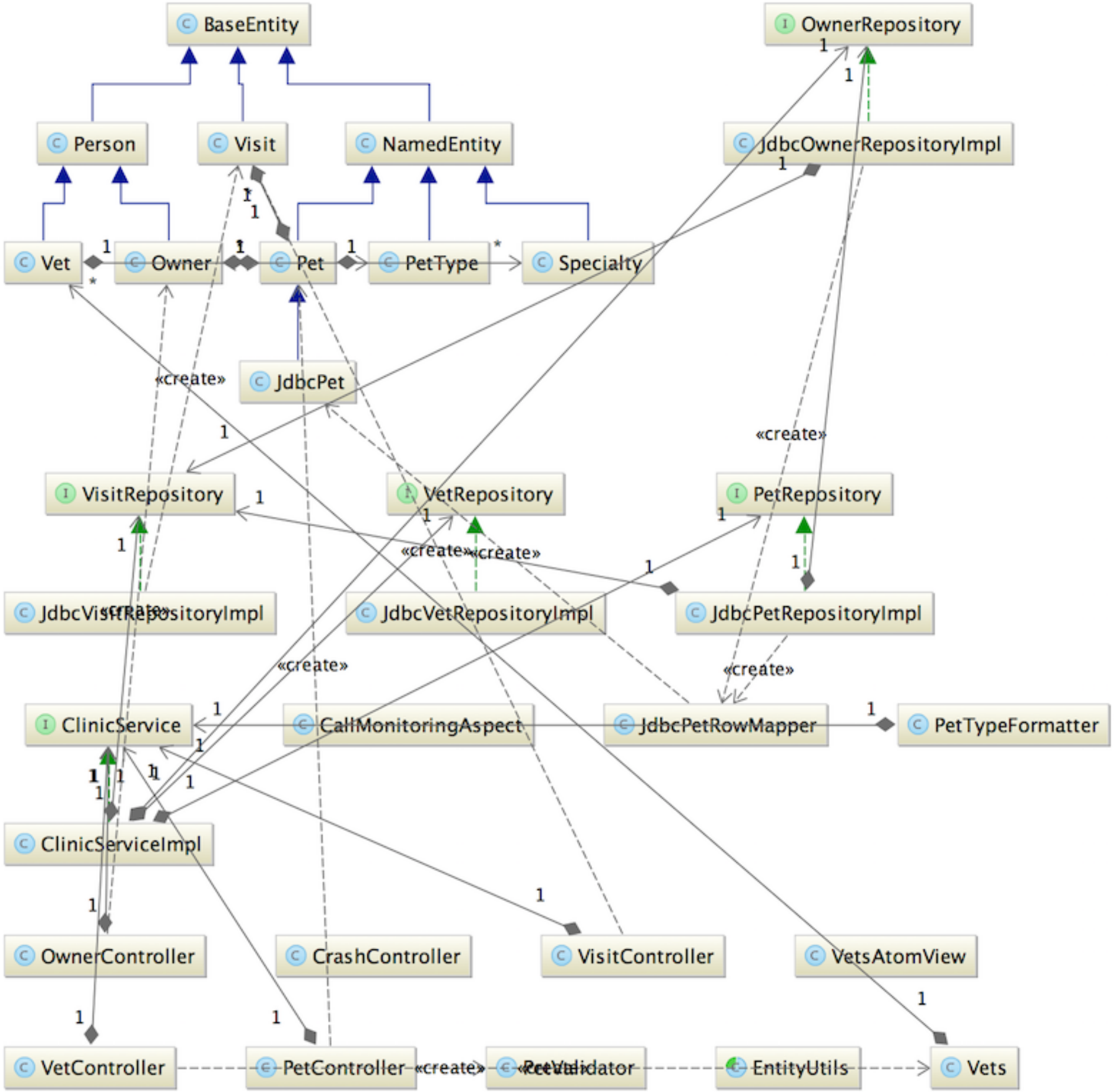
Spring PetClinic

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



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





An auto-generated UML class diagram

Diagramming tools see

code

rather than components

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

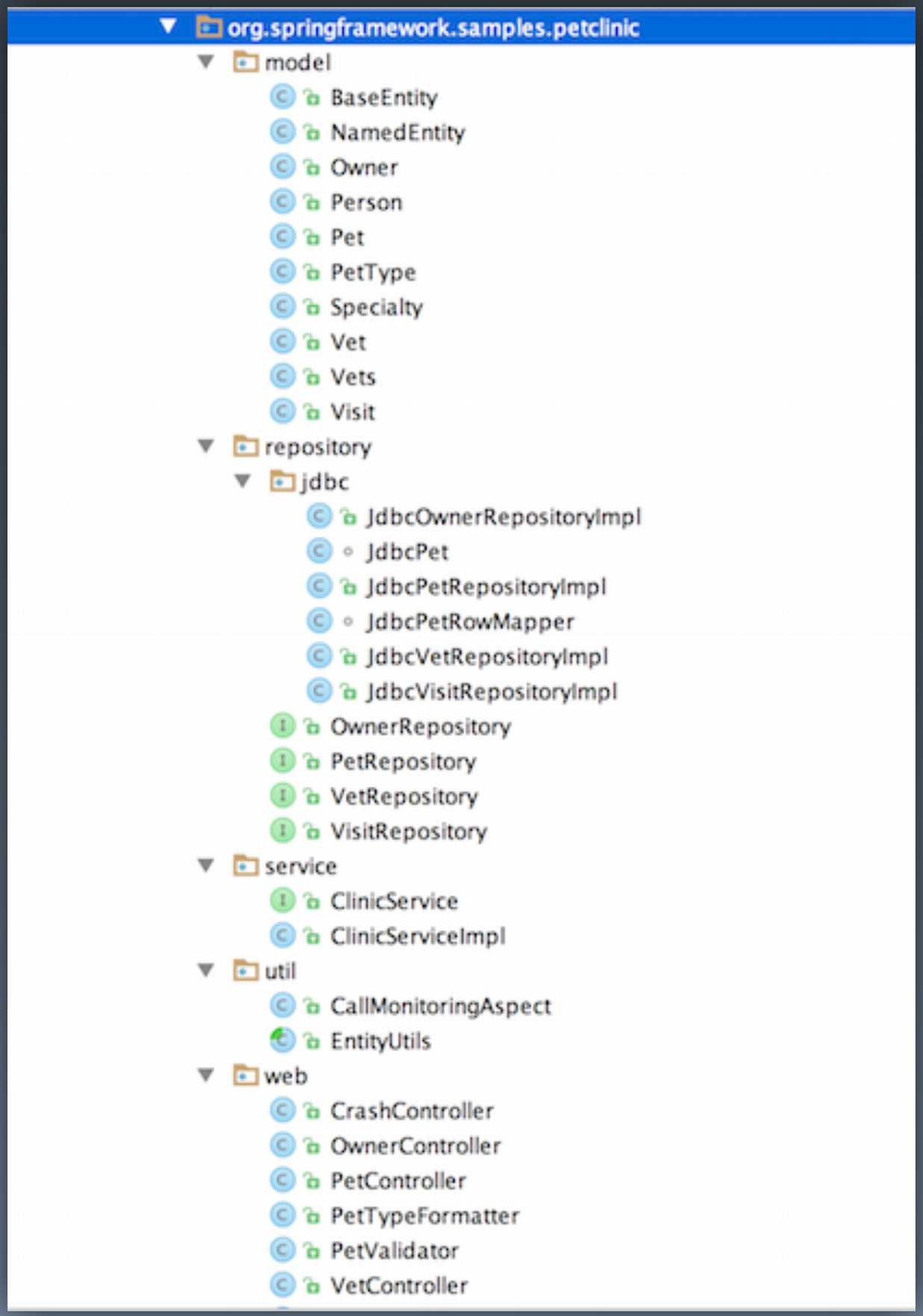
1 Introduction

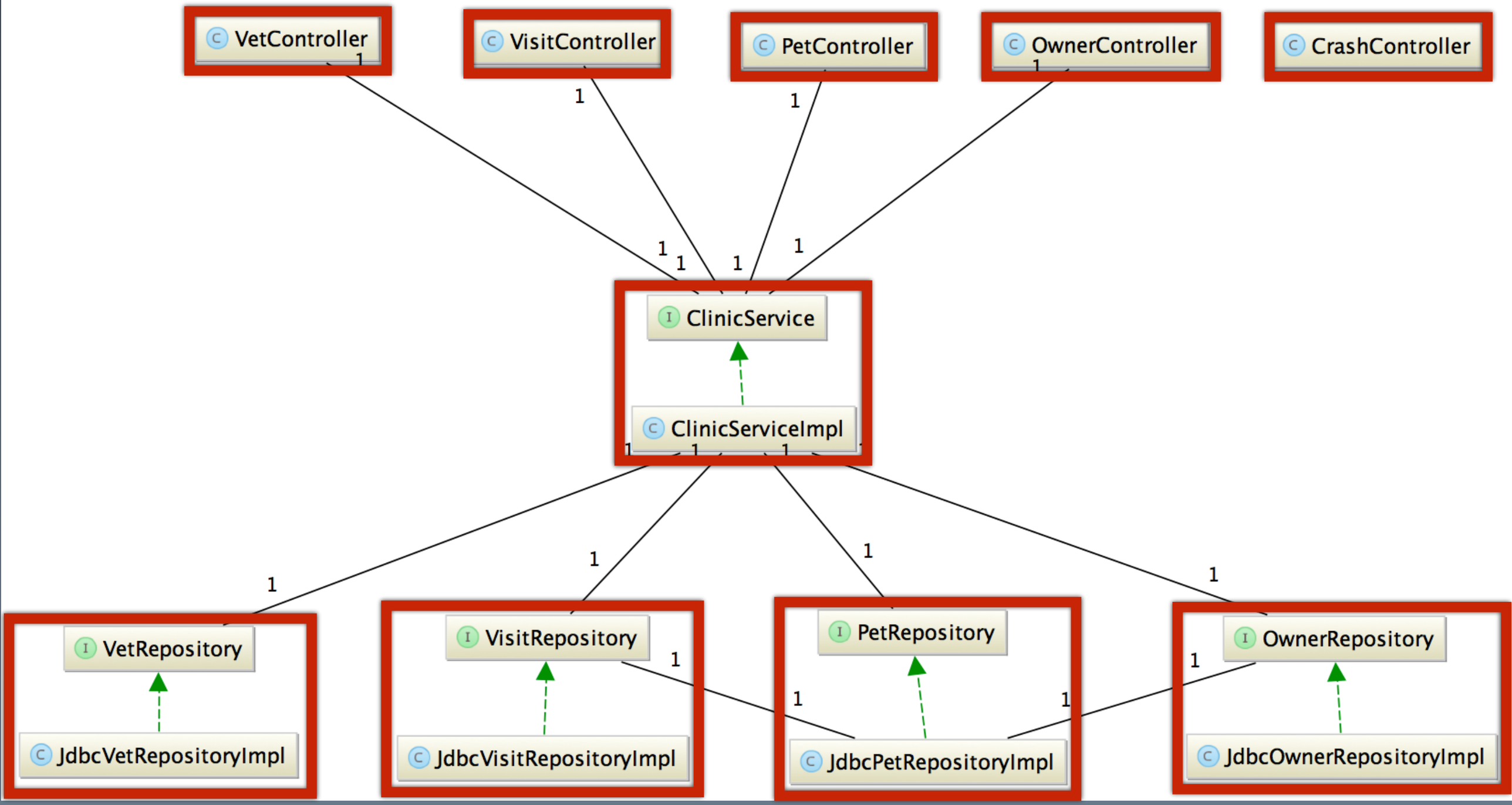
Software engineers often think about an existing software system in terms of high-level models. Box and arrow sketches of a system, for instance, are often found on engineers' whiteboards. Although these models are commonly used, reasoning about the system in terms of such models can be dangerous because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

What is a “component”?

What are the architecturally significant elements?





A UML class diagram showing architecturally significant elements

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 extended...

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...

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...

Reads from and writes to

Reads from and writes to

Reads from and writes to

Reads from and writes to

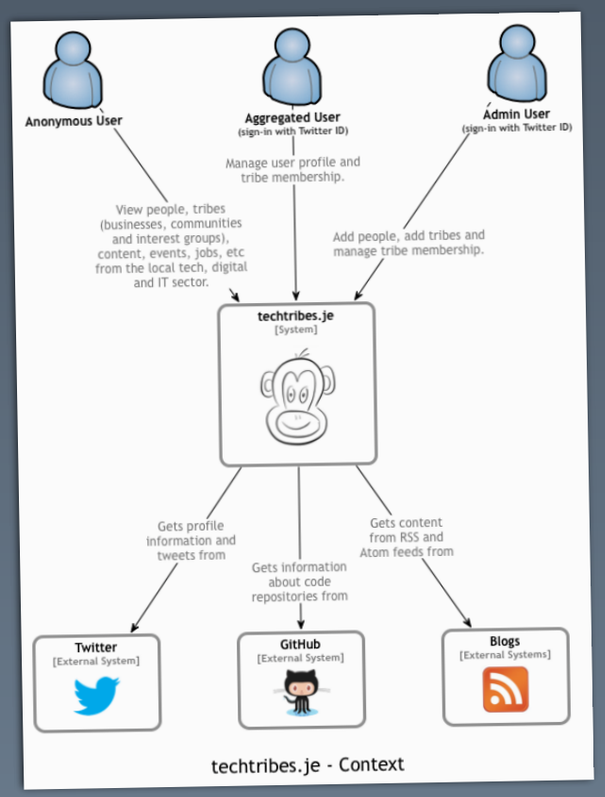
The code is the

embodiment

of the architecture

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.

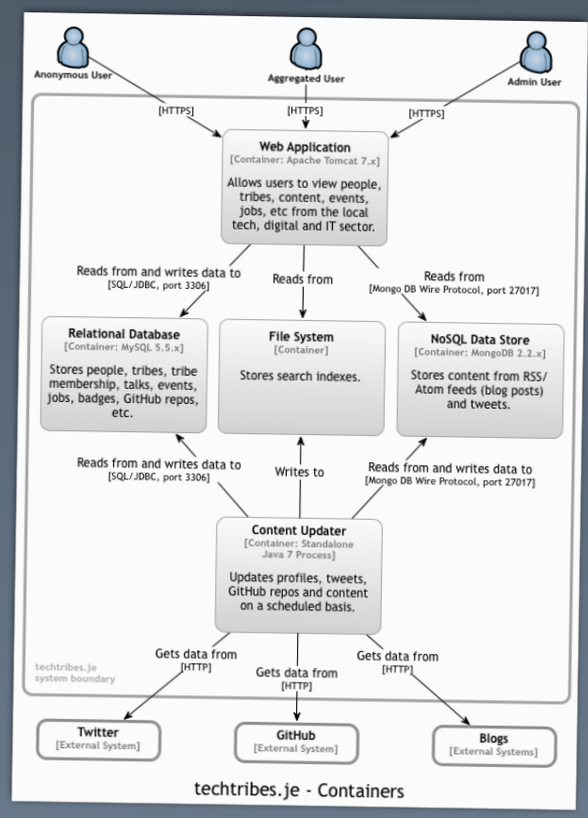
Containers

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

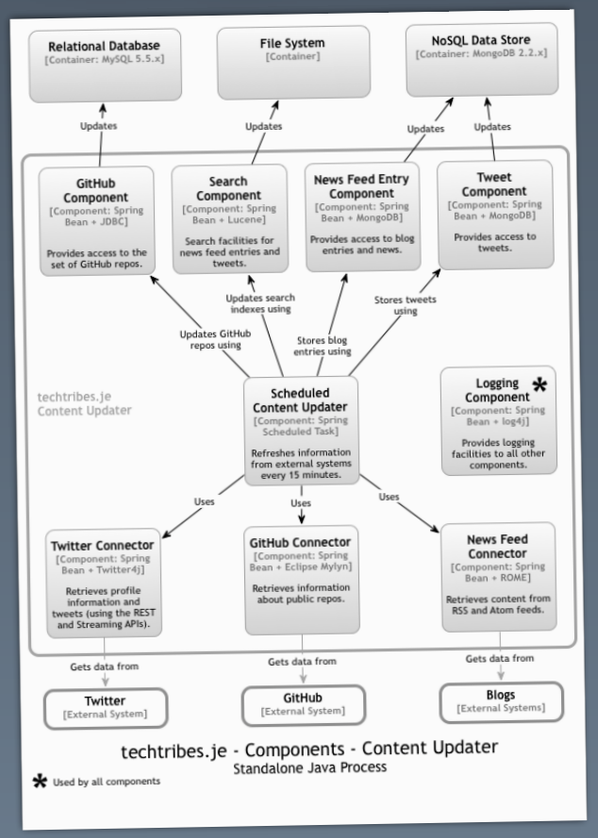
Components

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

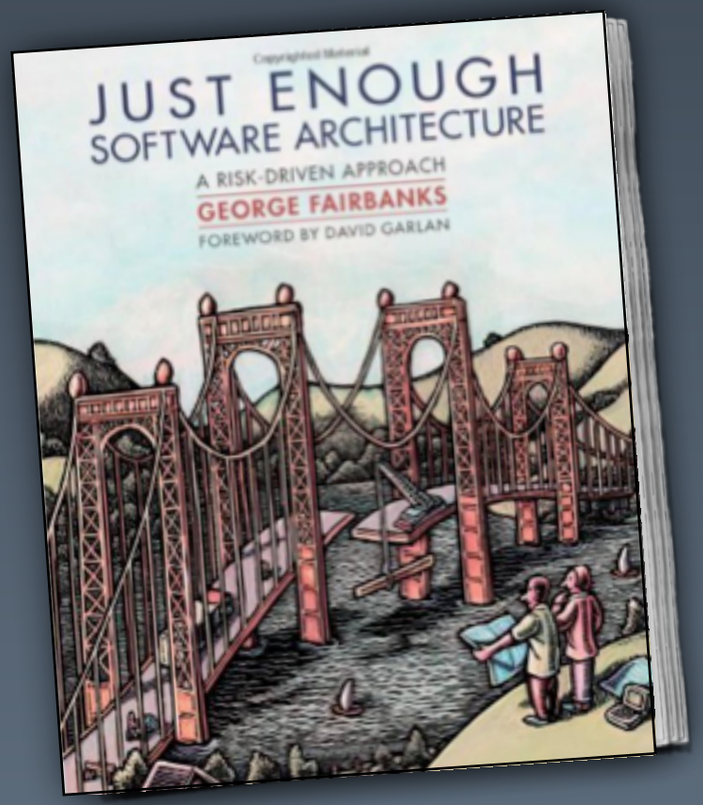
Integration points, APIs, known libraries, credentials for inbound consumers, etc.

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

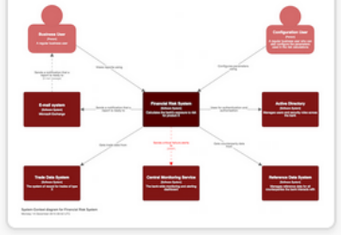
Extract as much of the software architecture from the code as possible, and supplement where necessary

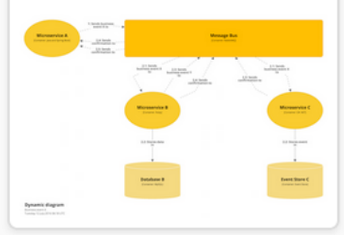
Architecture description languages

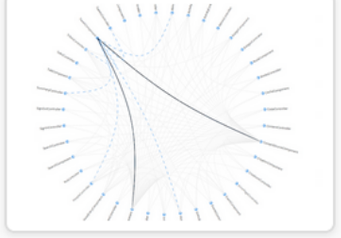
Create an architecture description language using code

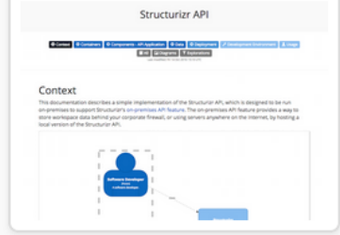
Structurizr

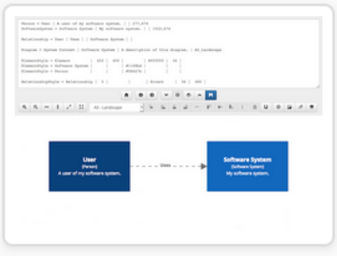
Visualise, document and explore your software architecture

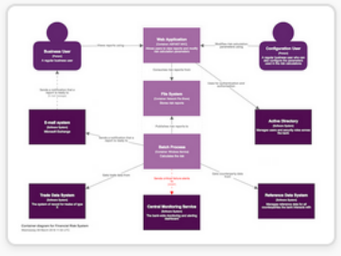


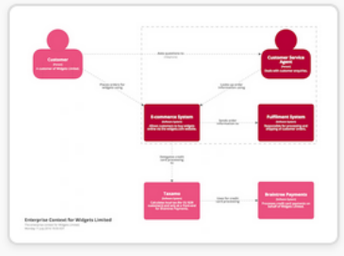


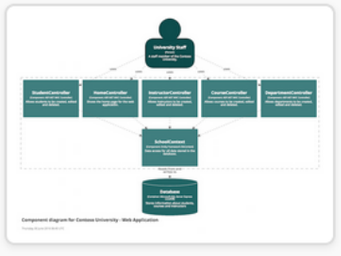


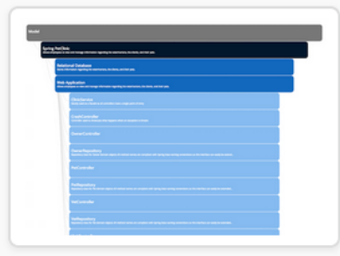


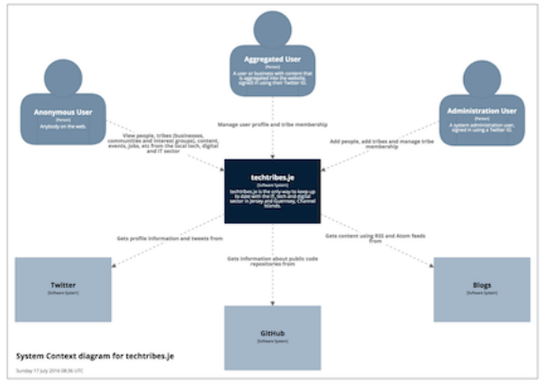


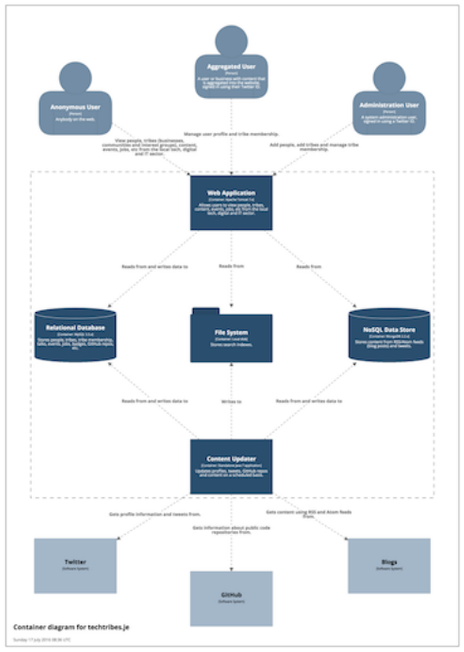


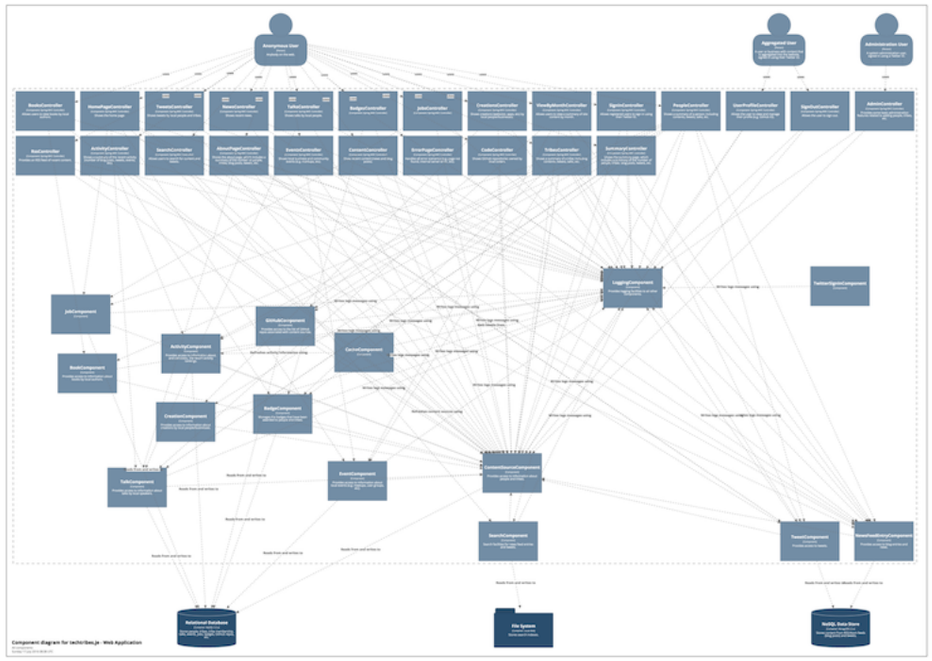


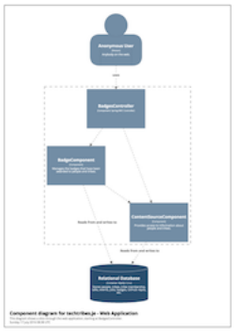


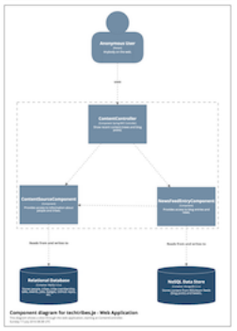


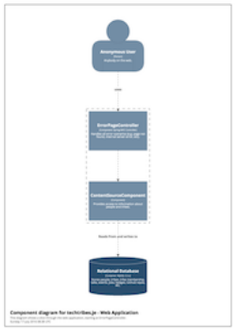


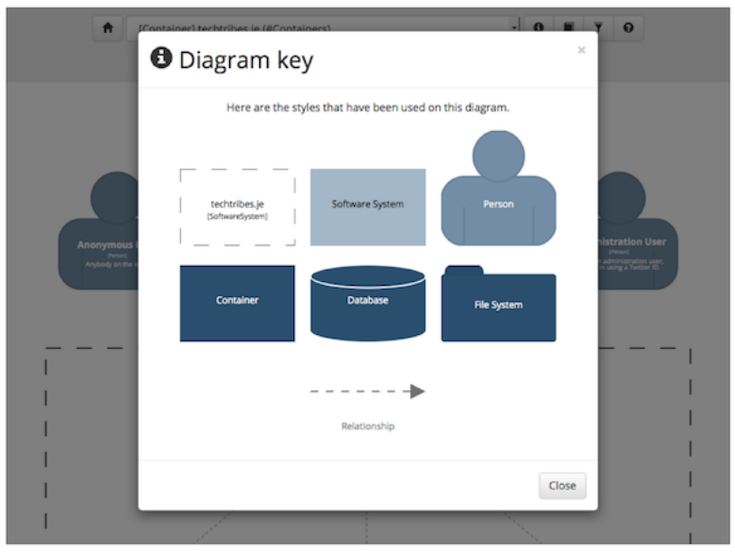


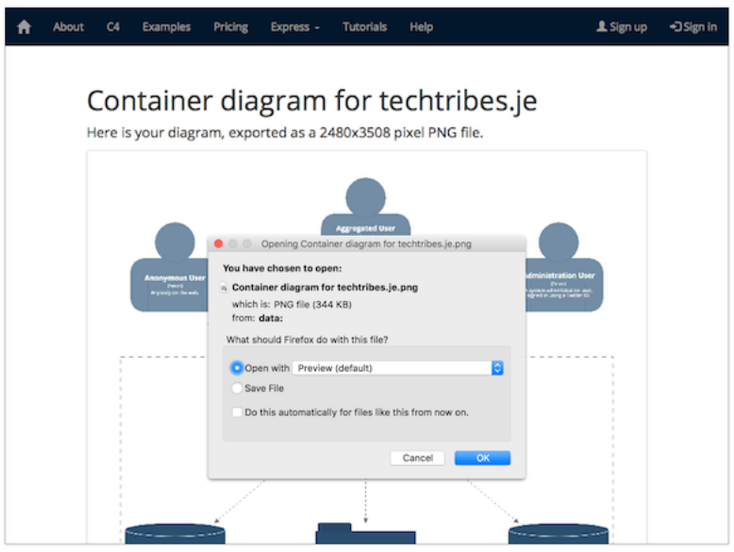












Visualise, document and explore your software architecture

System Context diagram for techtribes.je

Context

The techtribes.je website provides a way to find people, tribes (businesses, communities, interest groups, etc) and content related to the tech, IT and digital sector in Jersey and Guernsey. At the most basic level, it's a content aggregator for local tweets, news, blog posts, events, talks, jobs and more. Here's a context diagram that provides a visual summary of this:

System Context diagram for techtribes.je

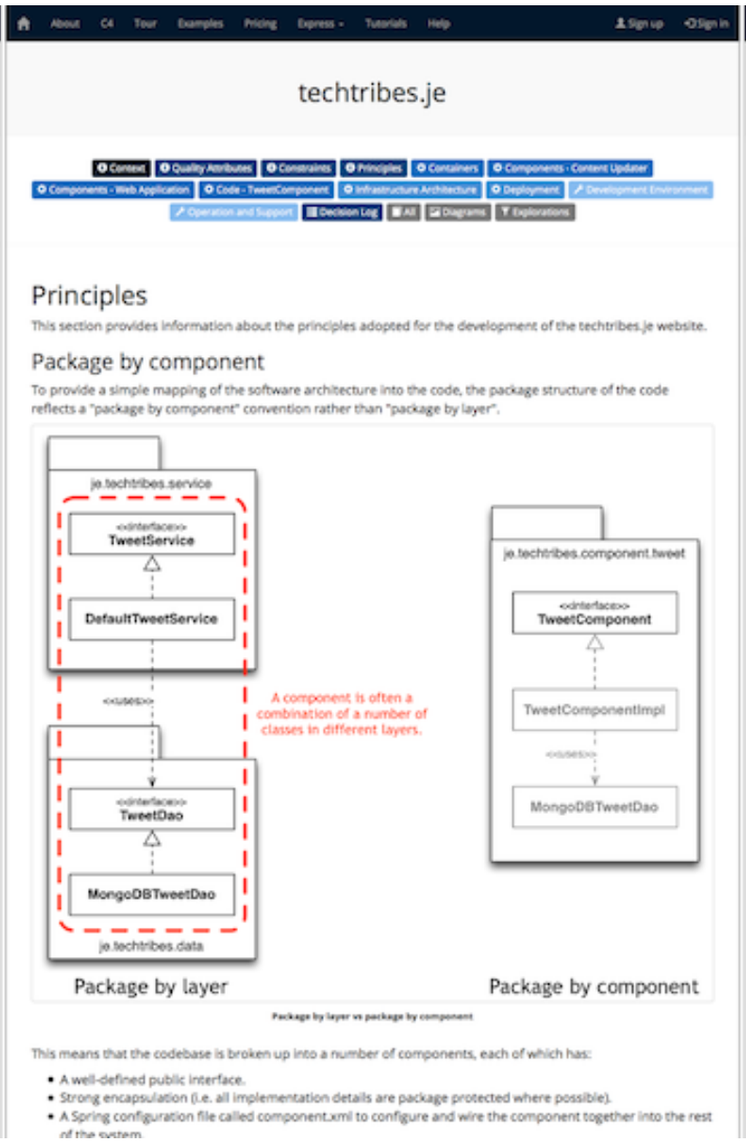
The purpose of the website is to:

1. Consolidate and share local content, helping to promote it inside and outside of the local community.
2. Encourage an open, sharing and learning culture within the local community.

Users

The techtribes.je website has three types of user:

1. **Anonymous**: anybody with a web browser can view content on the site.
2. **Authenticated**: people/tribes who have content aggregated into the website can sign-in to the website using their registered Twitter ID (if they have one) to modify some of their basic profile information.
3. **Admin**: people with administrative (super-user) access to the website can manage the people, tribes and



Package by layer vs package by component

Principles

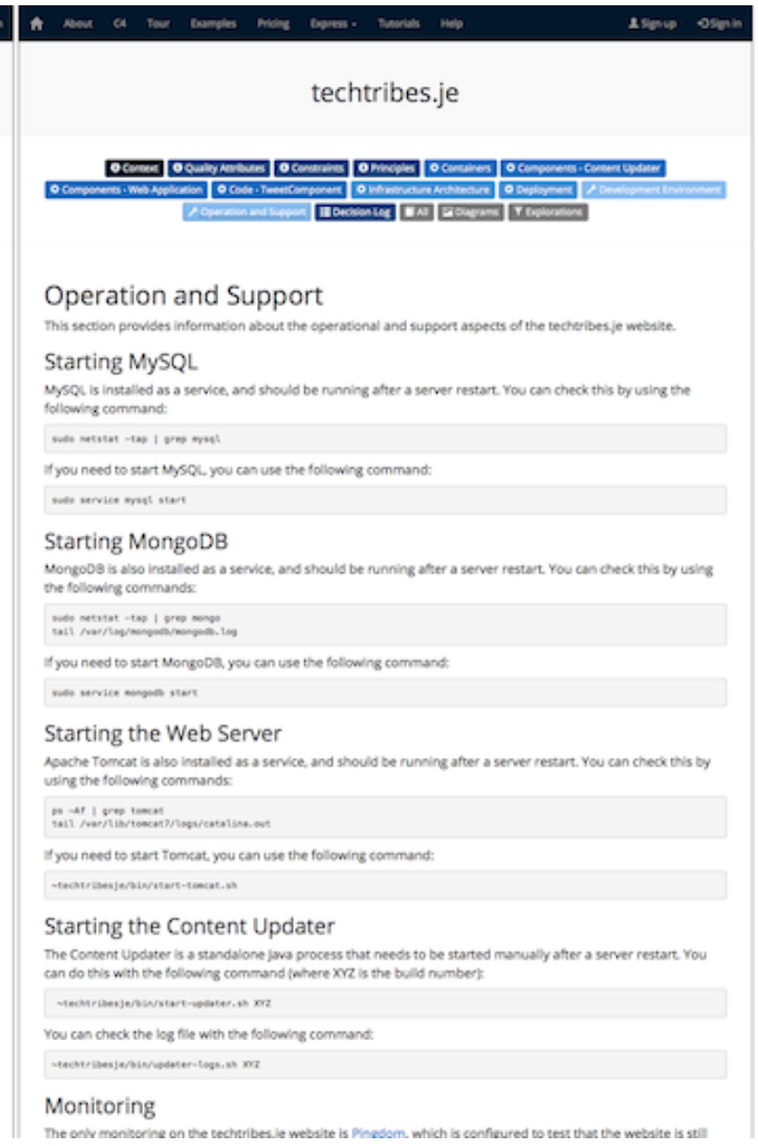
This section provides information about the principles adopted for the development of the techtribes.je website.

Package by component

To provide a simple mapping of the software architecture into the code, the package structure of the code reflects a "package by component" convention rather than "package by layer".

This means that the codebase is broken up into a number of components, each of which has:

- A well-defined public interface.
- Strong encapsulation (i.e. all implementation details are package protected where possible).
- A Spring configuration file called component.xml to configure and wire the component together into the rest of the system.



Operation and Support

This section provides information about the operational and support aspects of the techtribes.je website.

Starting MySQL

MySQL is installed as a service, and should be running after a server restart. You can check this by using the following command:

```
sudo netstat -tap | grep mysql
```

If you need to start MySQL, you can use the following command:

```
sudo service mysql start
```

Starting MongoDB

MongoDB is also installed as a service, and should be running after a server restart. You can check this by using the following commands:

```
sudo netstat -tap | grep mongo
tail /var/log/mongodb/mongodb.log
```

If you need to start MongoDB, you can use the following command:

```
sudo service mongodb start
```

Starting the Web Server

Apache Tomcat is also installed as a service, and should be running after a server restart. You can check this by using the following commands:

```
ps -Af | grep tomcat
tail /var/lib/tomcat7/logs/catalina.out
```

If you need to start Tomcat, you can use the following command:

```
~techtribesje/bin/start-tomcat.sh
```

Starting the Content Updater

The Content Updater is a standalone Java process that needs to be started manually after a server restart. You can do this with the following command (where XYZ is the build number):

```
~techtribesje/bin/start-updater.sh XYZ
```

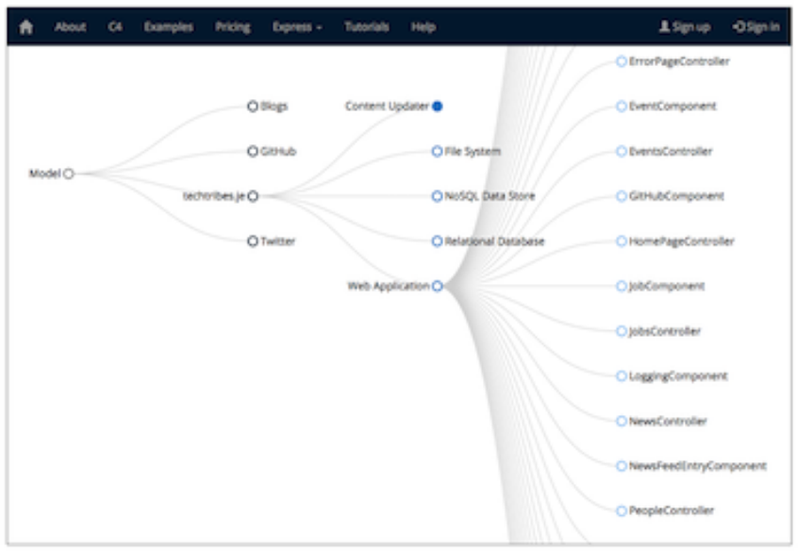
You can check the log file with the following command:

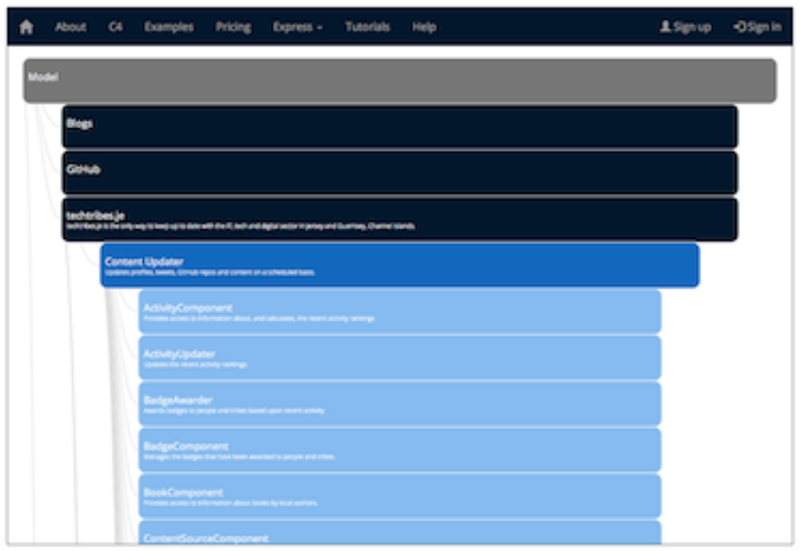
```
~techtribesje/bin/updater-logs.sh XYZ
```

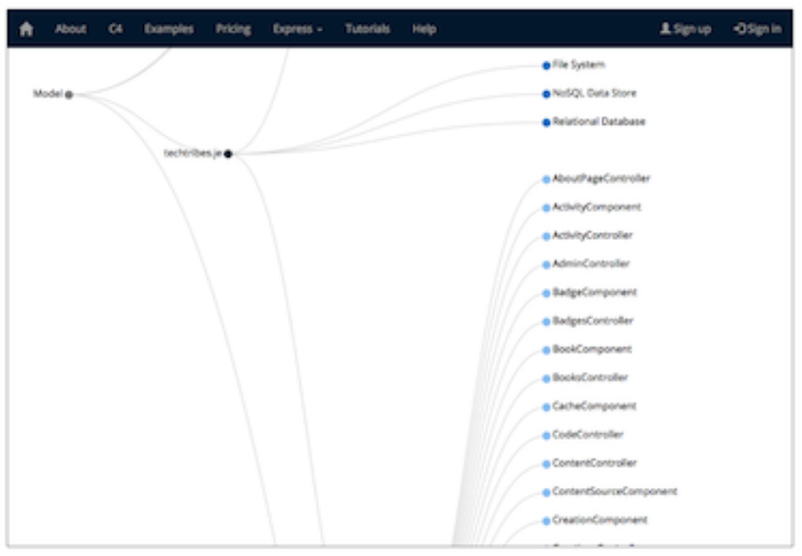
Monitoring

The only monitoring on the techtribes.je website is Pingdom, which is configured to test that the website is still

Visualise, **document** and explore your software architecture







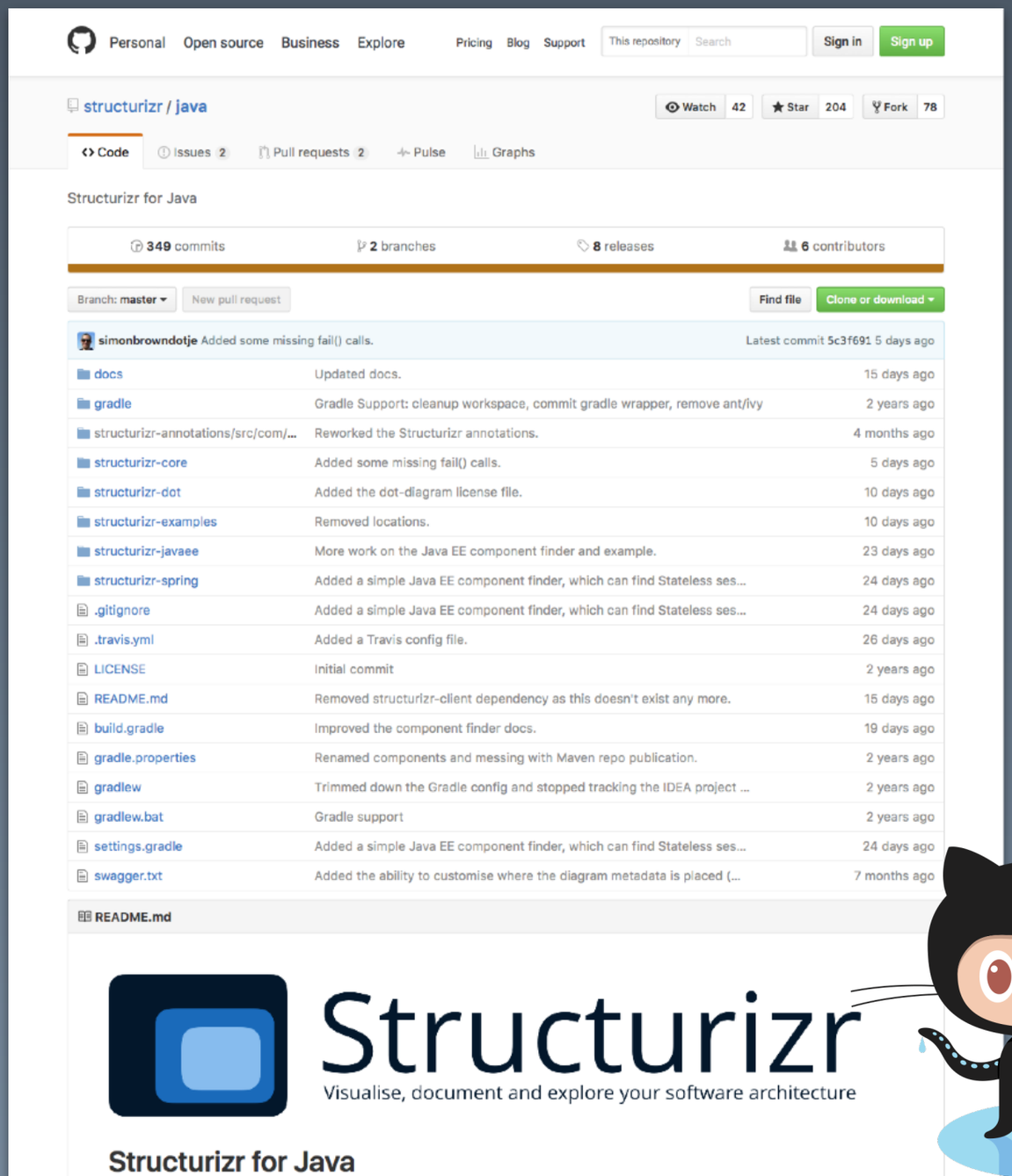
Visualise, document and **explore** your software architecture

```
Person = User | A user of my software system. | | 277,674
SoftwareSystem = Software System | My software system. | | 1552,674

Relationship = User | Uses | | Software System | |

Diagram = System Context | Software System | A description of this diagram. | A5_Landscape

ElementStyle = Element         |  650 |  400 |         | #ffffff |  36 |
ElementStyle = Software System |      |      | #1168bd |         |     |
ElementStyle = Person          |      |      | #08427b |         |     |
```

A5 - Landscape

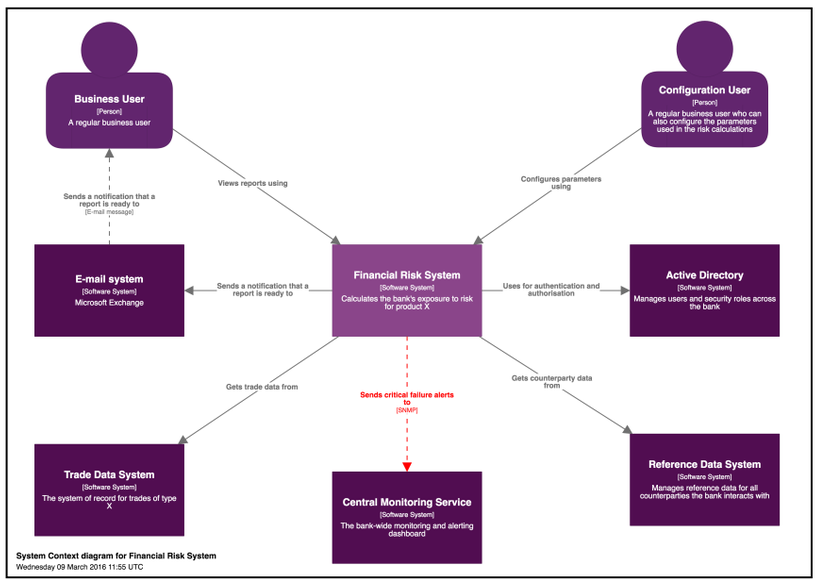
Structurizr for Java

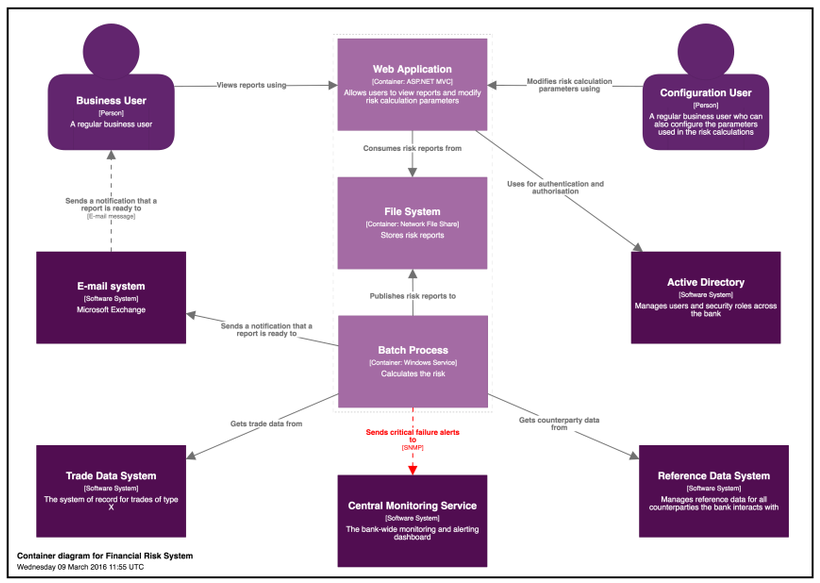
Structurizr for .NET

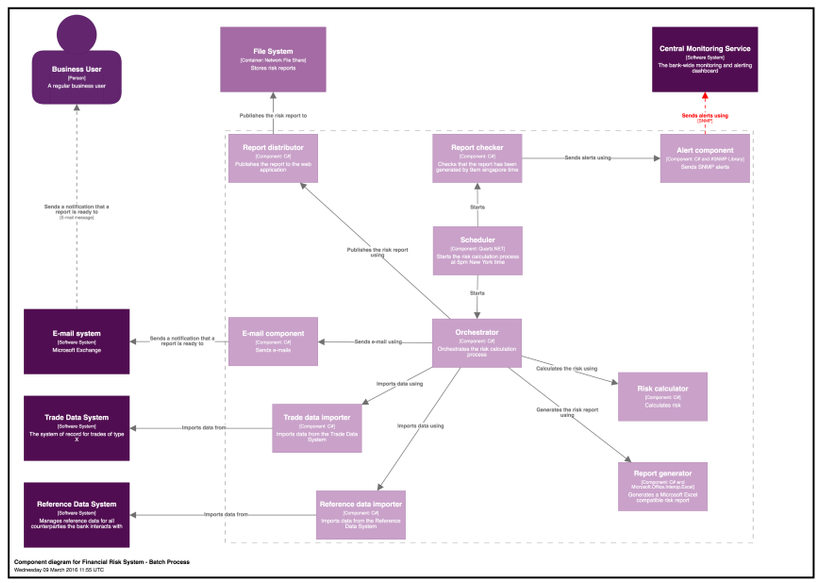


```
Workspace workspace = new Workspace("My model", "This is a model of my software system.");
Model model = workspace.getModel();

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.getViews();
SystemContextView contextView = viewSet.createSystemContextView(softwareSystem, "context", "A simple example...");
contextView.addAllSoftwareSystems();
contextView.addAllPeople();

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

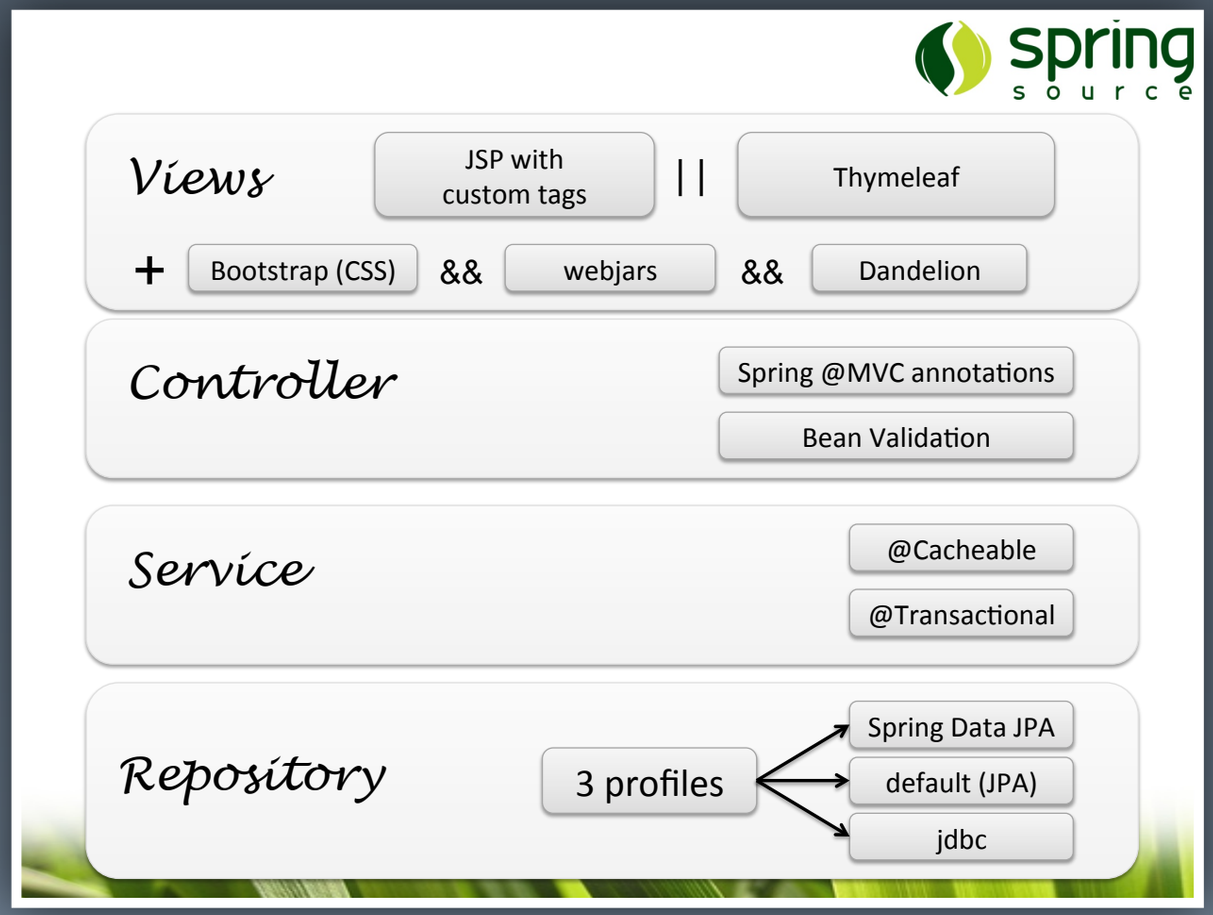




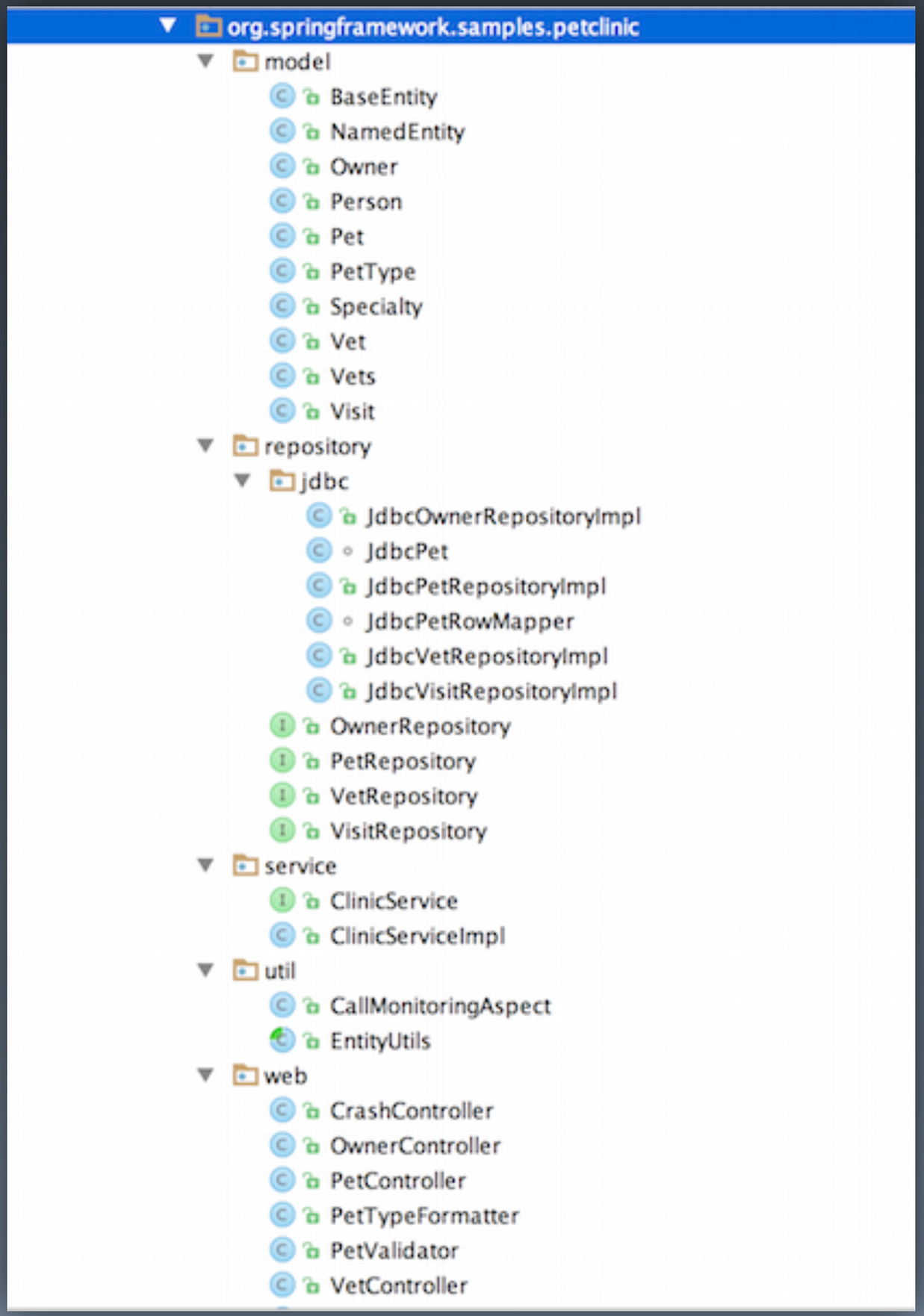


Spring PetClinic

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



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



```
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-projects/spring-petclinic/)");

    Model model = workspace.getModel();




}
```

```
// software systems and people
SoftwareSystem springPetClinic = model.addSoftwareSystem(
    "Spring PetClinic",
    "Allows employees to view and manage information regarding the
     veterinarians, the clients, and their pets.”);

Person clinicEmployee = model.addPerson(
    "Clinic Employee", "An employee of the clinic");

clinicEmployee.uses(springPetClinic, “Uses");
```

```
// containers
Container webApplication = springPetClinic.addContainer(
    "Web Application",
    "Allows employees to view and manage information regarding the
     veterinarians, the clients, and their pets.",
    "Apache Tomcat 7.x”);

Container relationalDatabase = springPetClinic.addContainer(
    "Relational Database",
    "Stores information regarding the veterinarians, the clients,
     and their pets.”, "HSQLDB");
```

```
clinicEmployee.uses(webApplication,
    "Uses", “HTTP");

webApplication.uses(relationalDatabase,
    "Reads from and writes to", "JDBC, port 9001”);
```

“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)

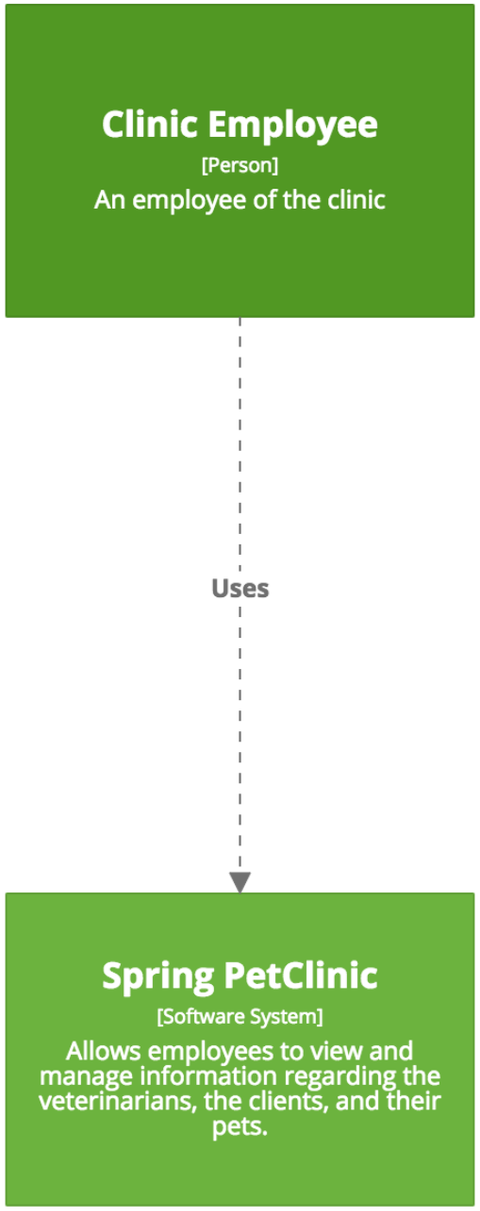
```
// components
ComponentFinder componentFinder = new ComponentFinder(
    webApplication,
    "org.springframework.samples.petclinic",
    new SpringComponentFinderStrategy(
        new ReferencedTypesSupportingTypesStrategy()
    ),
    new SourceCodeComponentFinderStrategy(
        new File(sourceRoot, "/src/main/java/"), 150));

componentFinder.findComponents();
```

```
// connect components with other model elements
webApplication.getComponents().stream()
    .filter(c -> c.getTechnology().equals(SpringComponentFinderStrategy.SPRING_MVC_CONTROLLER))
    .forEach(c -> clinicEmployee.uses(c, "Uses", "HTTP"));

webApplication.getComponents().stream()
    .filter(c -> c.getTechnology().equals(SpringComponentFinderStrategy.SPRING_REPOSITORY))
    .forEach(c -> c.uses(relationalDatabase, "Reads from and writes to", "JDBC"));
```

```
// system context, container and component views
ViewSet viewSet = workspace.getViews();

SystemContextView contextView = viewSet.createContextView(
    springPetClinic, “context",
    "Context view for Spring PetClinic");
contextView.addAllSoftwareSystems();
contextView.addAllPeople();

ContainerView containerView = viewSet.createContainerView(
    springPetClinic, “containers",
    "Container view for Spring PetClinic");
containerView.addAllPeople();
containerView.addAllSoftwareSystems();
containerView.addAllContainers();

ComponentView componentView = viewSet.createComponentView(
    webApplication, “components",
    "The Components diagram for the Spring PetClinic web application.");
componentView.addAllComponents();
componentView.addAllPeople();
componentView.add(relationalDatabase);
```

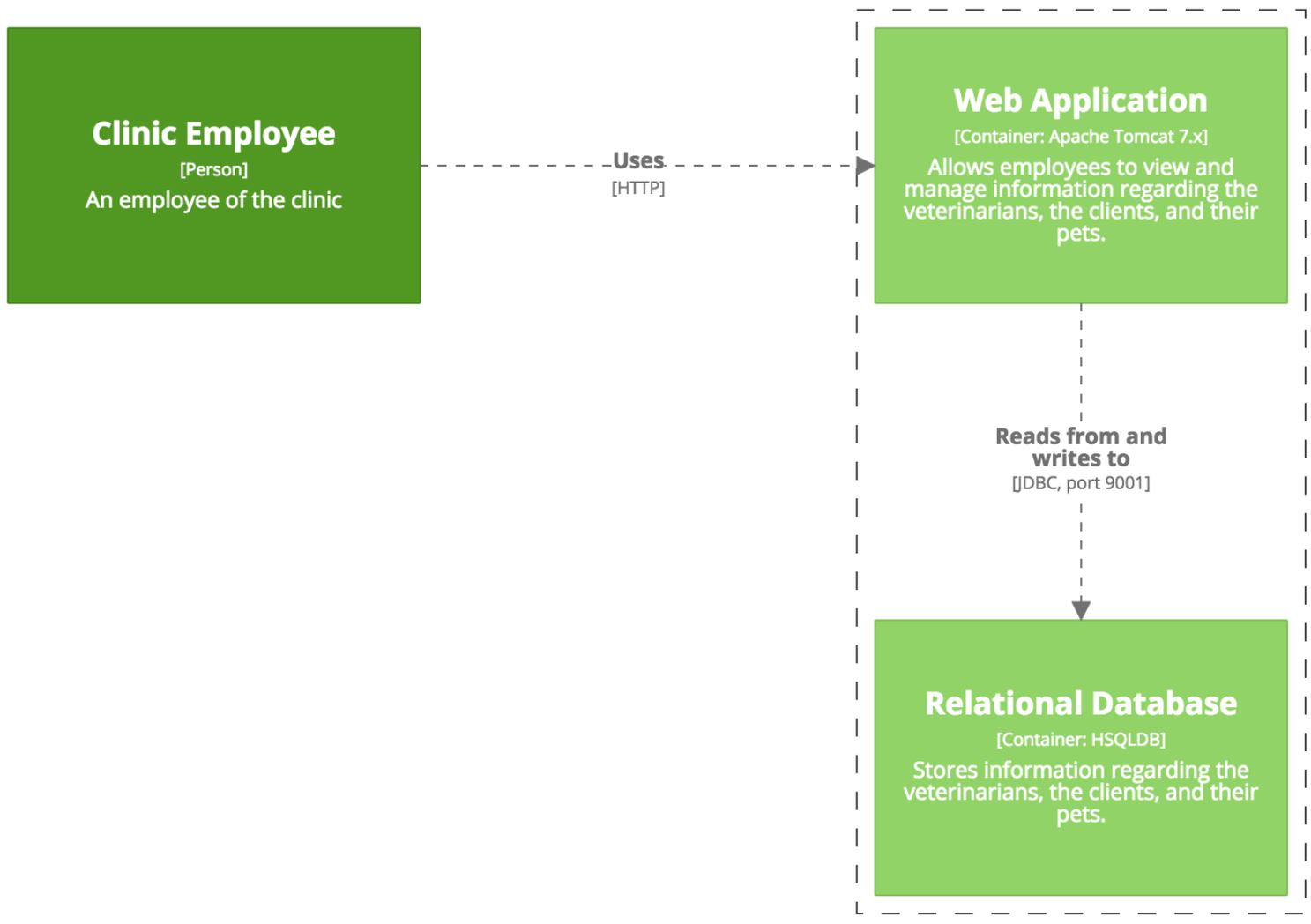
```
// upload the software architecture model to structurizr.com
StructurizrClient client = new StructurizrClient("key", "secret");
client.mergeWorkspace(1234, workspace);
```


```
{
 "id" : 0,
 "name" : "Spring PetClinic",
 "description" : "This is a C4 representation of the Spring PetClinic sample app (https://github.com/spring-projects/spring-petclinic/)",
 "model" : {
   "people" : [ {
     "tags" : "Element,Person",
     "id" : "2",
     "name" : "Clinic Employee",
     "description" : "An employee of the clinic",
     "relationships" : [ {
       "tags" : "Relationship,Synchronous",
       "id" : "3",
       "sourceId" : "2",
       "destinationId" : "1",
       "description" : "Uses",
       "interactionStyle" : "Synchronous"
     }, {
       "tags" : "Relationship,Synchronous",
       "id" : "6",
       "sourceId" : "2",
       "destinationId" : "4",
       "description" : "Uses",
       "technology" : "HTTP",
       "interactionStyle" : "Synchronous"
     }, {
       "tags" : "Relationship,Synchronous",
       "id" : "28",
       "sourceId" : "2",
       "destinationId" : "8",
       "description" : "Uses",
       "technology" : "HTTP",
       "interactionStyle" : "Synchronous"
     }, {
       "tags" : "Relationship,Synchronous",
       "id" : "29",
       "sourceId" : "2",
       "destinationId" : "9",
       "description" : "Uses",
       "technology" : "HTTP",
       "interactionStyle" : "Synchronous"
     }, {
       "tags" : "Relationship,Synchronous",
       "id" : "30",
       "sourceId" : "2",
       "destinationId" : "10",
```



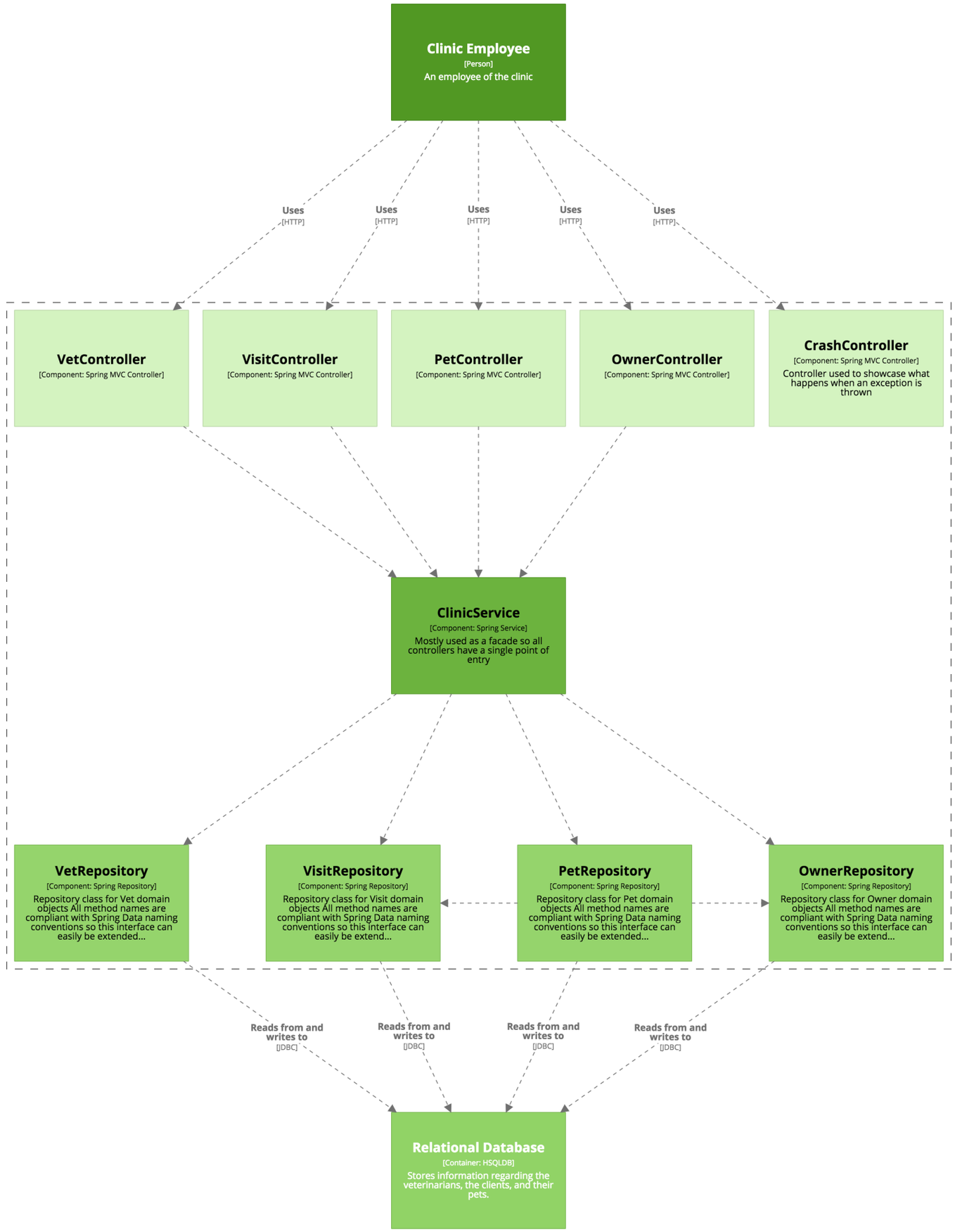
System Context diagram for Spring PetClinic

The System Context diagram for the Spring PetClinic system.
Wednesday 19 October 2016 12:52 EDT



Container diagram for Spring PetClinic

The Containers diagram for the Spring PetClinic system.
Wednesday 19 October 2016 12:52 EDT

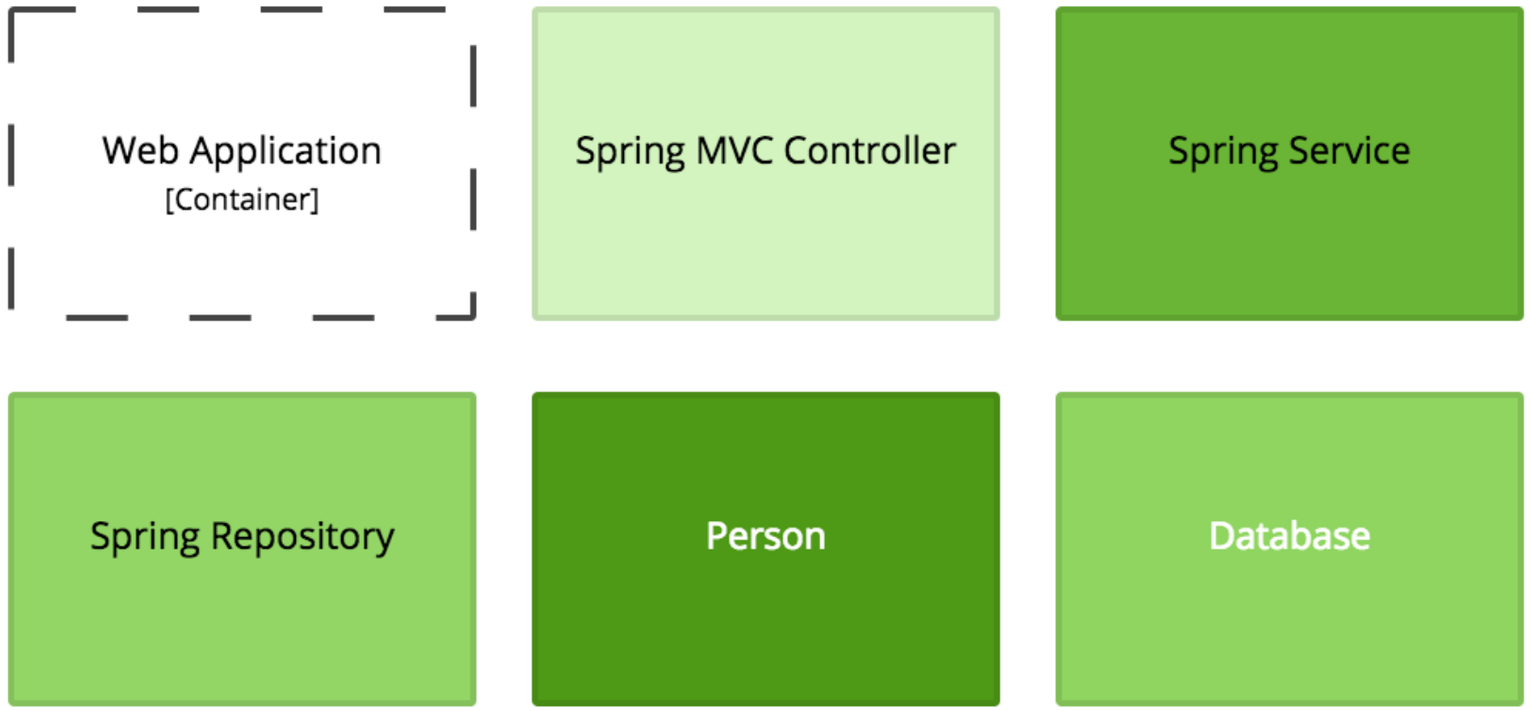


Component diagram for Spring PetClinic - Web Application

The Components diagram for the Spring PetClinic web application.
Wednesday 19 October 2016 12:52 EDT

Diagram key

Here are the styles that have been used on this diagram.



Relationship

Close

Uses [HTTP]

Uses [HTTP]

Uses [HTTP]

Uses [HTTP]

Uses [HTTP]

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

Navigate from diagram to source code

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...

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...

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]

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**

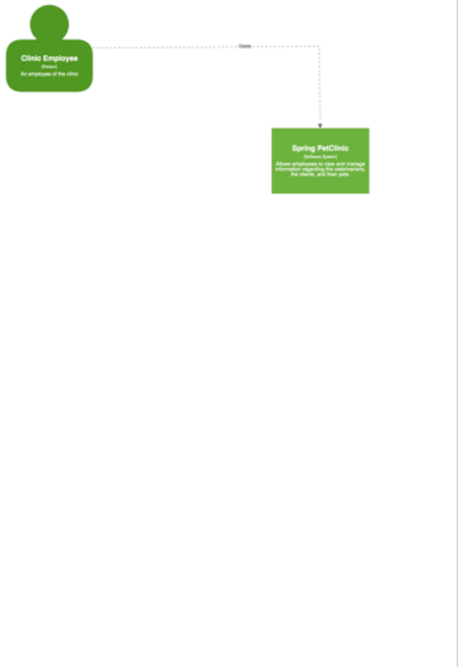
michaelisvy test methods:used should/shouldNot 5c9ab6 on 16 Jan

2 contributors

52 lines (38 sloc) 1.67 KB Raw Blame History

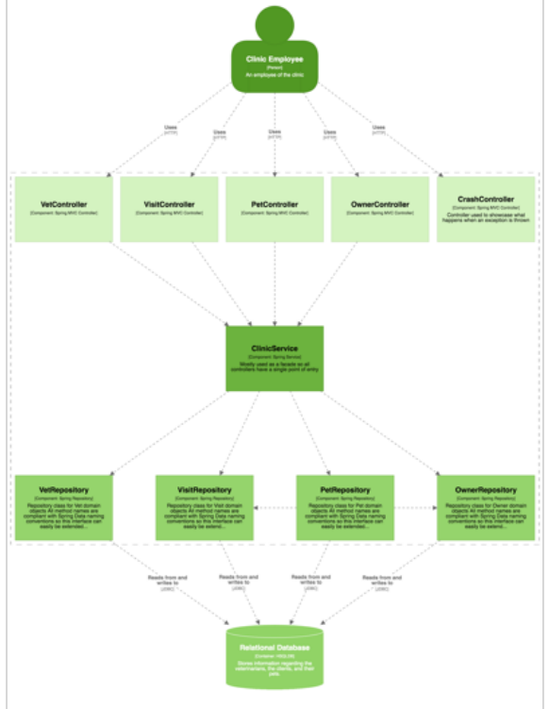
```
/*
 * Copyright 2002-2013 the original author or authors.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *      http://www.apache.org/licenses/LICENSE-2.0
 *
 * 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



System Context diagram for Spring PetClinic
Wednesday 23 March 2016 11:25 UTC

Container diagram for Spring PetClinic
Wednesday 23 March 2016 11:25 UTC



Component diagram for Spring PetClinic - Web Application
Wednesday 23 March 2016 11:46 UTC

System Context diagram → Container diagram → Component diagram → Source code

Double-click a software system

Double-click a container

Double-click a component

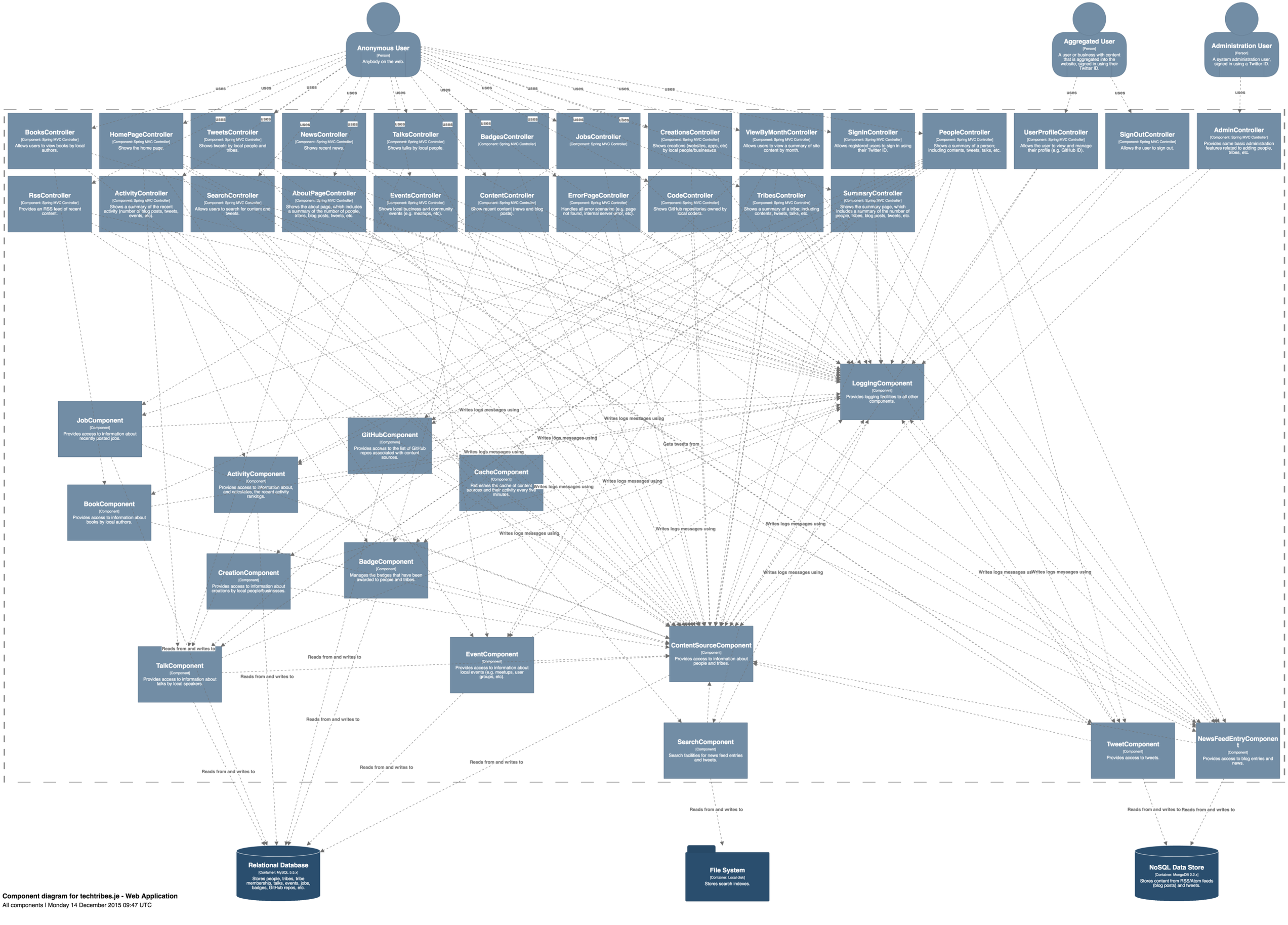
Diagrams are maps

Alec Nikolas Reiter
@just_anr

My computer is currently struggling to open a generated UML diagram of Django. It's 388,500px by 11,000px. And 114mb.

LIKE
1

1:19 AM - 27 Feb 2016

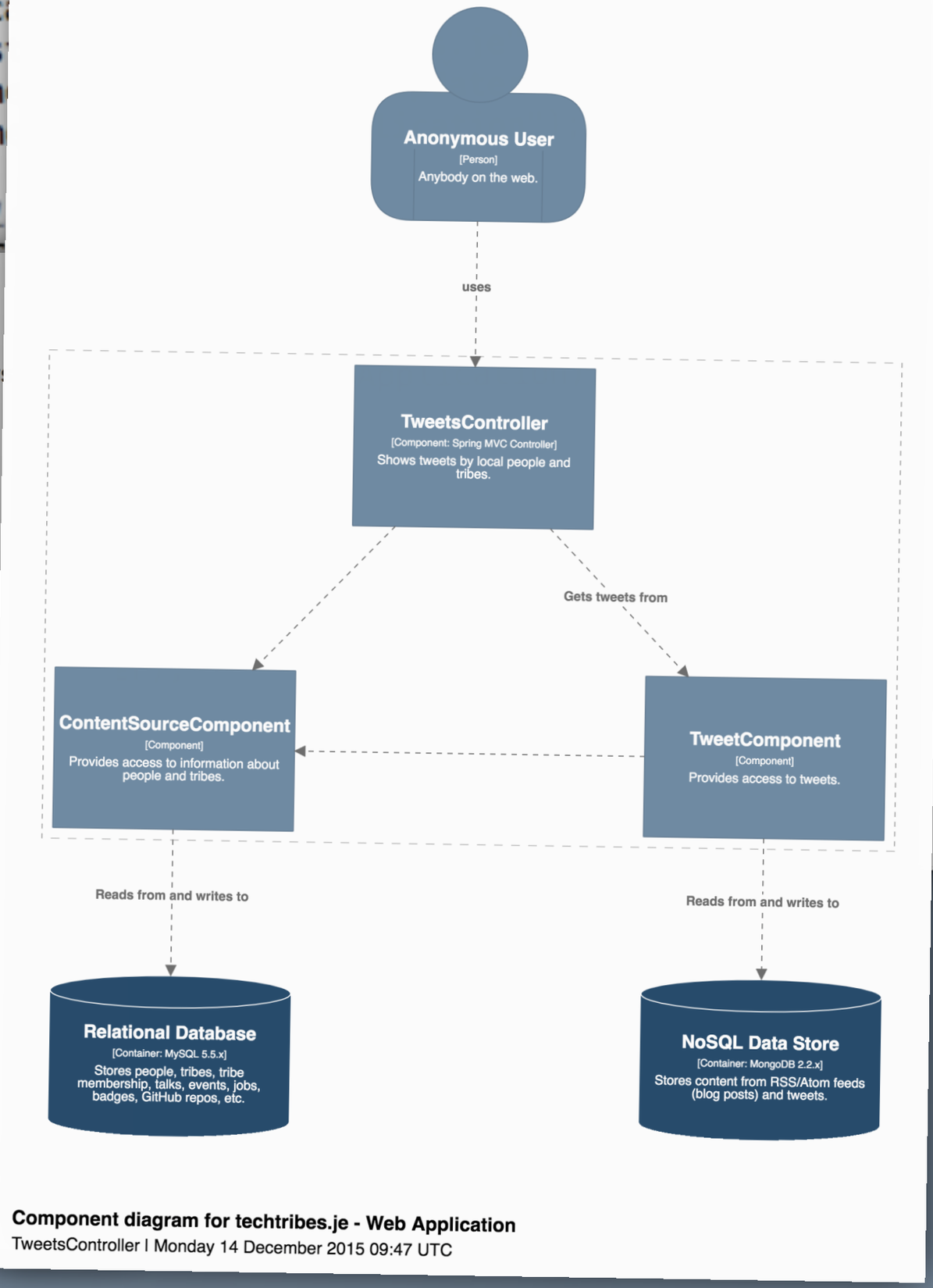


Component diagram for techtribes.je - Web Application

All components | Monday 14 December 2015 09:47 UTC

```
private static void createComponentViewsForWebApplica
    SoftwareSystem techTribes = model.getSoftwareSys
    Container contentUpdater = techTribes.getContain
    Container webApplication = techTribes.getContain

    // create one component view per Spring controll
    Set<Component> controllers = webApplication.getC
```

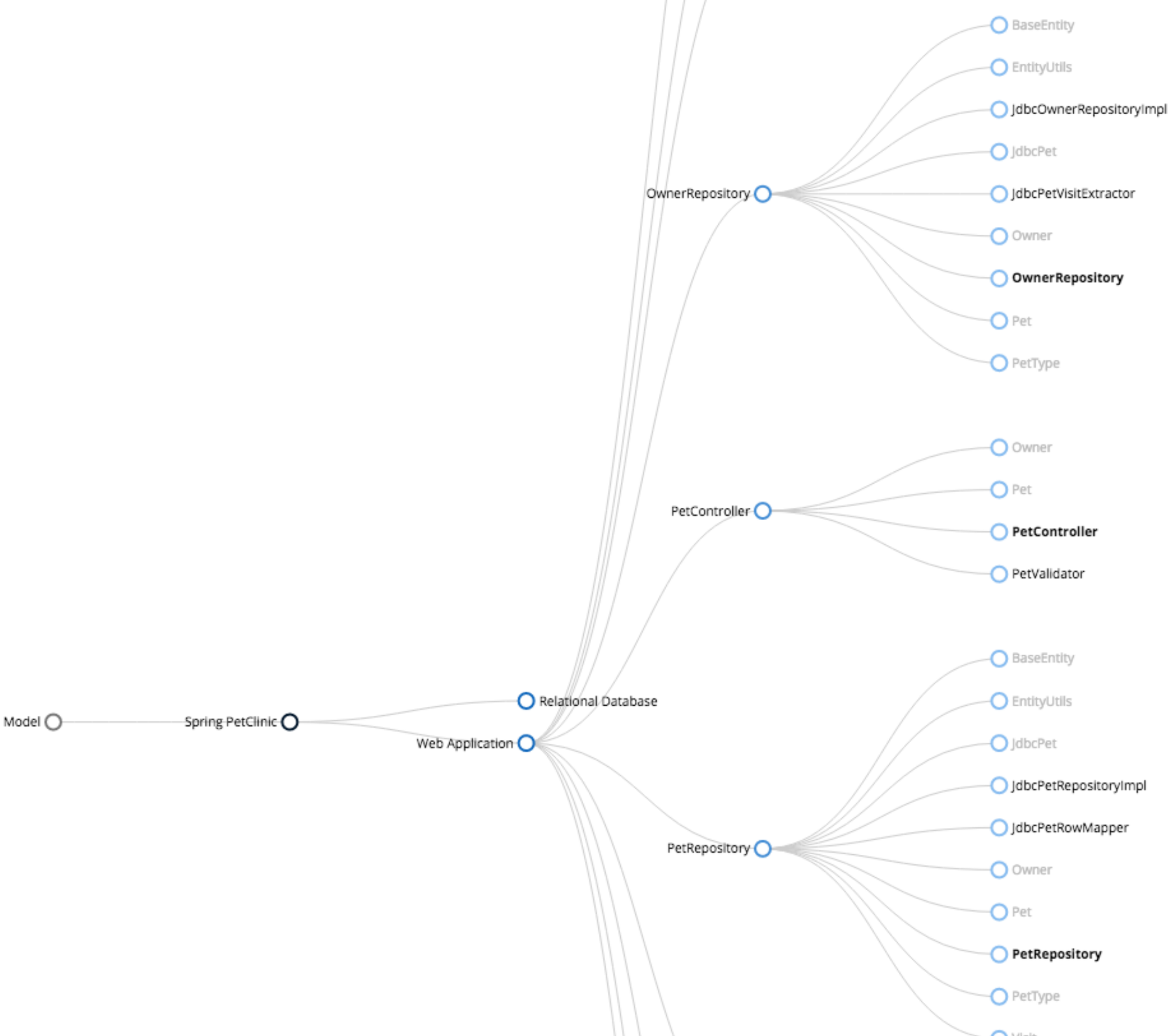
techtribes.je - System Context
techtribes.je - Containers
techtribes.je - Content Updater - Components - Awarding badges
techtribes.je - Content Updater - Components - Updating information from external sy
techtribes.je - Content Updater - Components - Updating recent activity
techtribes.je - Web Application - 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

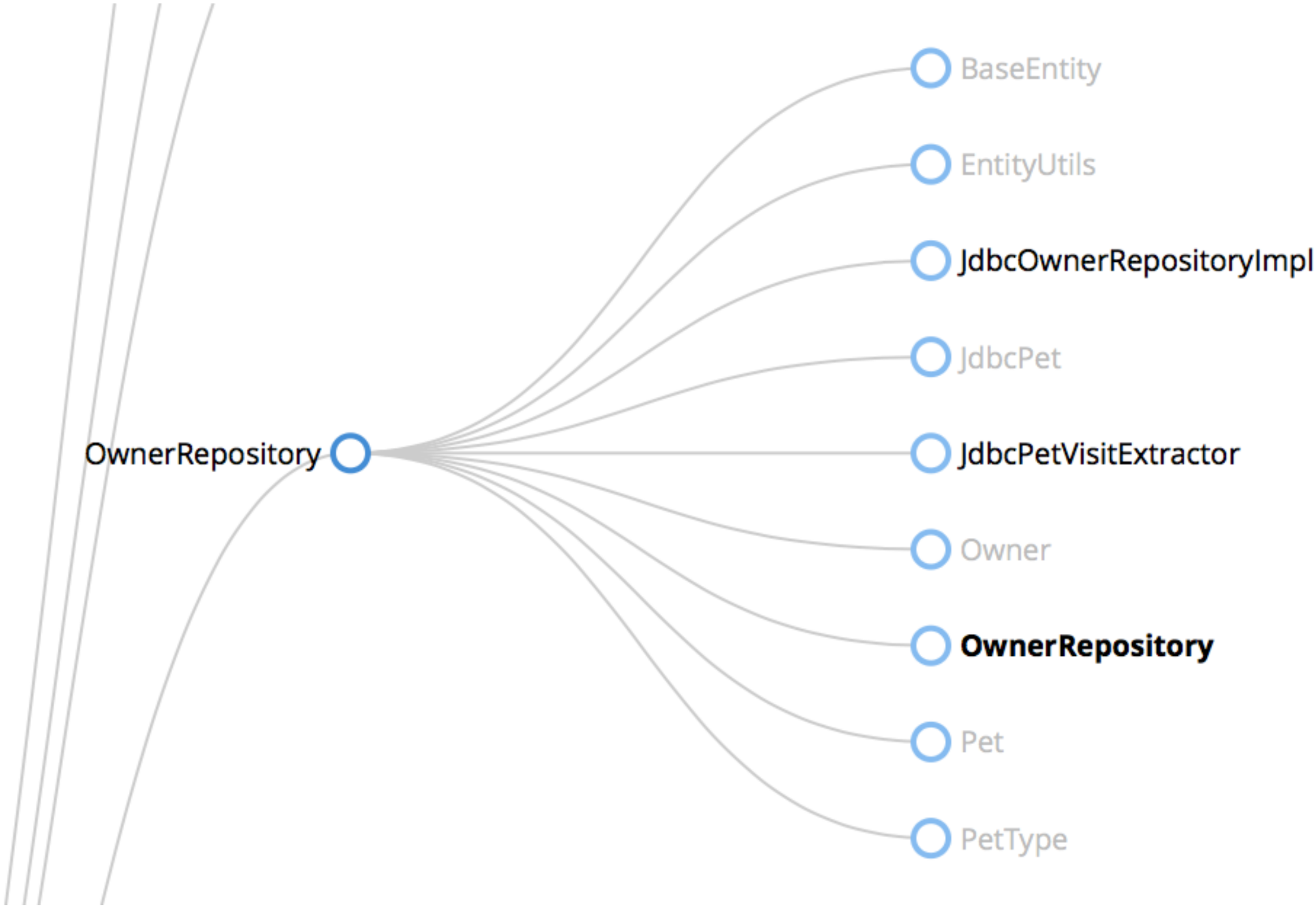


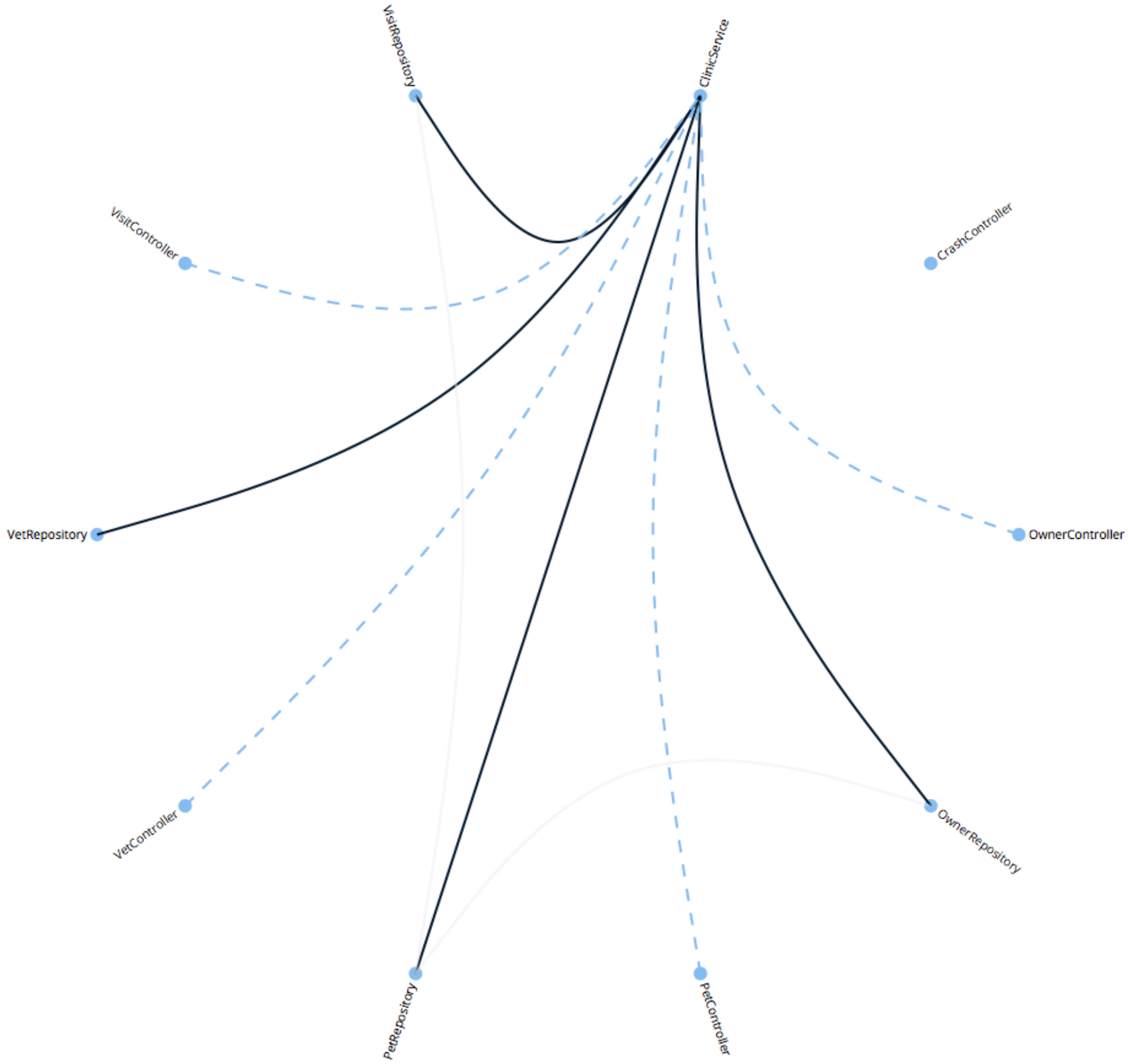
Component diagram for techtribes.je - Web Application
TweetsController | Monday 14 December 2015 09:47 UTC

Creating the model as code provides opportunities…

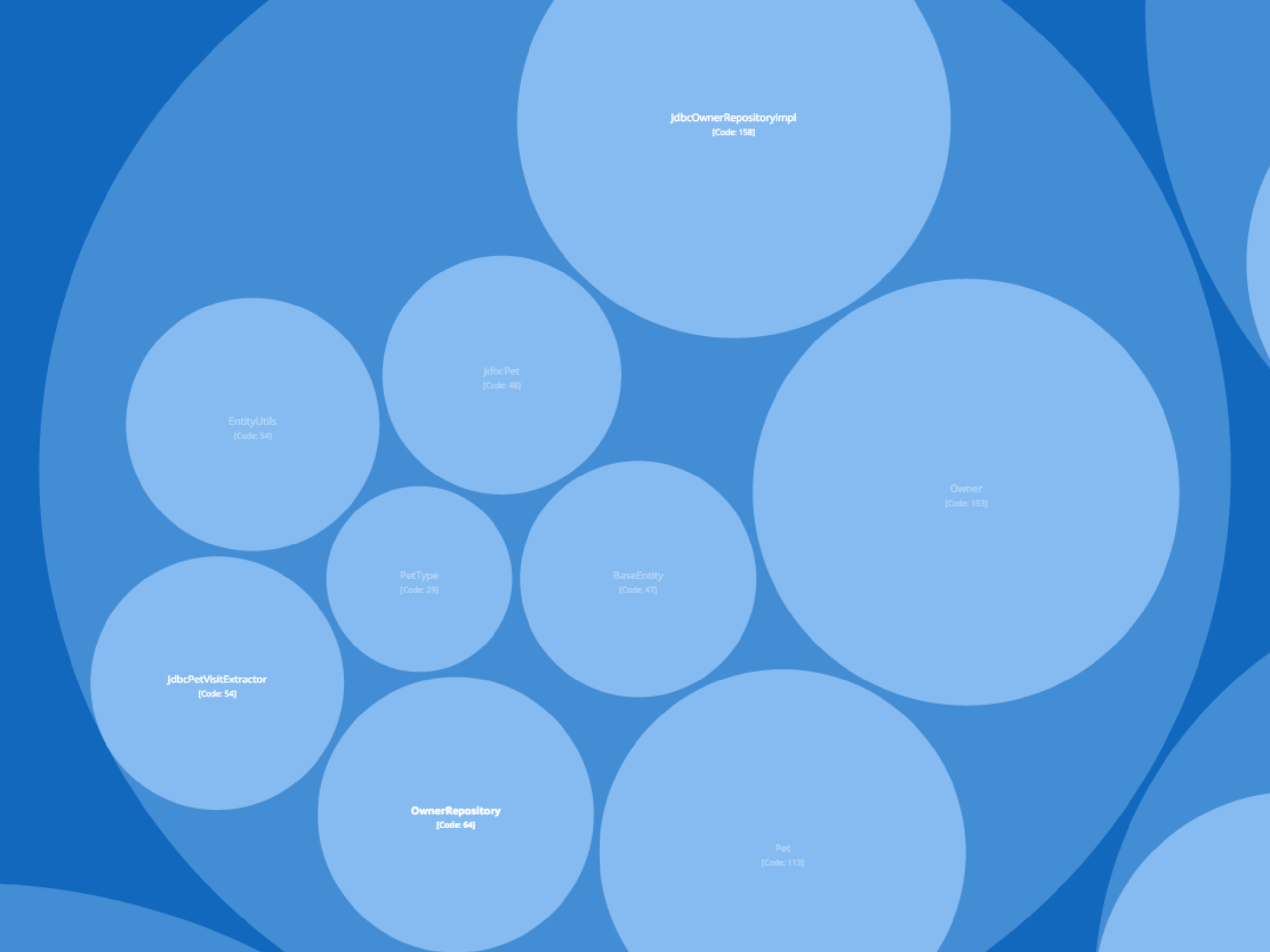
Once you have a model, you can export that model and visualise it however you like…

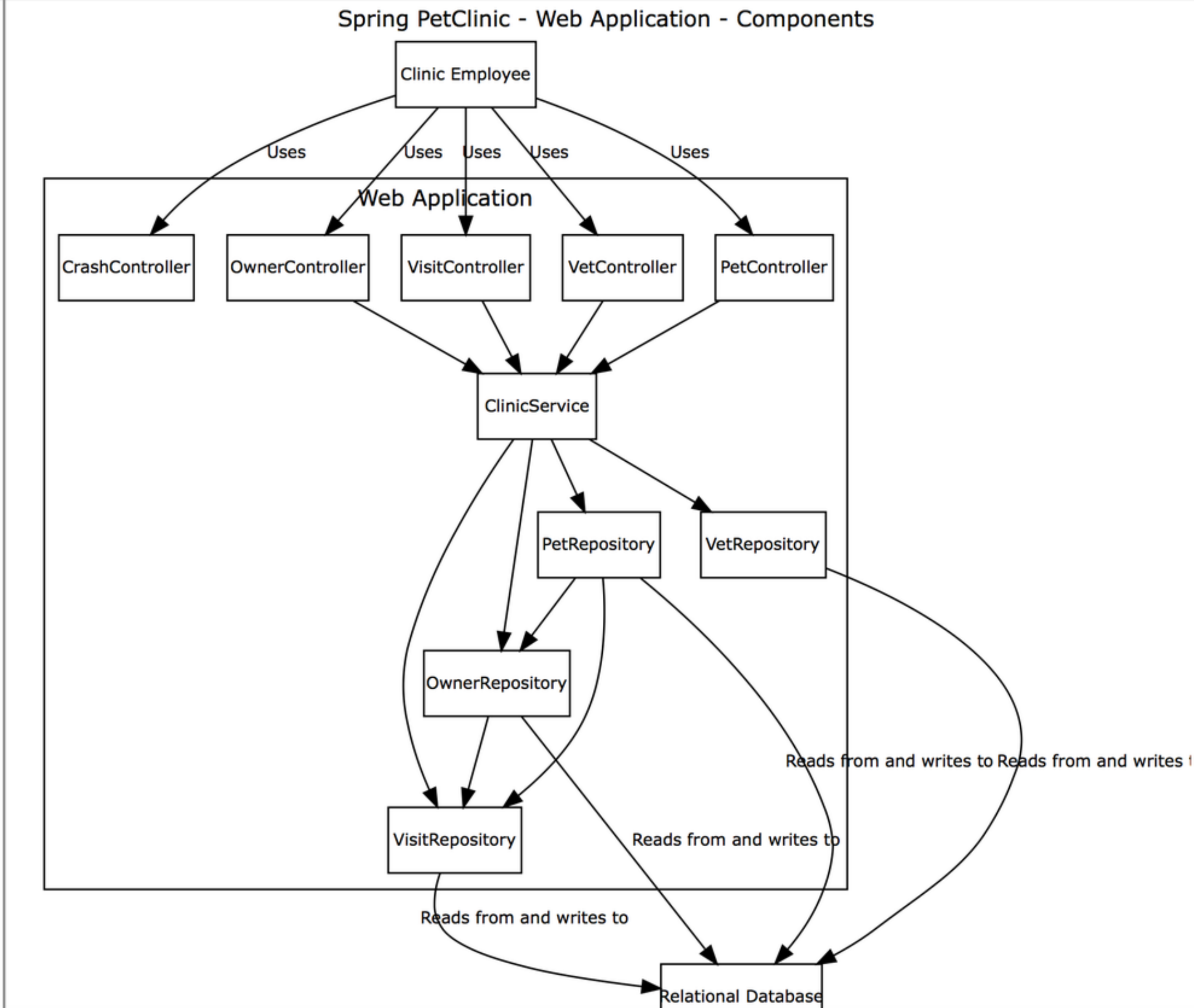

BaseEntity
EntityUtils
JdbcOwnerRepositoryImpl
JdbcPet
JdbcPetVisitExtractor
OwnerRepository
Owner
OwnerRepository
Pet
PetType
Owner
Pet
PetController
PetValidator
BaseEntity
EntityUtils
JdbcPet
JdbcPetRepositoryImpl
JdbcPetRowMapper
Owner
Pet
PetRepository
PetType
Model
Spring PetClinic
Relational Database
Web Application
PetRepository
PetController


BaseEntity
EntityUtils
JdbcOwnerRepositoryImpl
JdbcPet
OwnerRepository
JdbcPetVisitExtractor
Owner
OwnerRepository
Pet
PetType


VisitRepository
ClinicService
VisitController
CrashController
VetRepository
OwnerController
VetController
OwnerRepository
PetRepository
PetController


ClinicService
[Component: 600]
OwnerRepository
[Component: 784]
OwnerController
[Component: 427]
PetController
[Component: 553]
PetRepository
[Component: 863]
VisitController
[Component: 432]
CrashControlle
[Component: 80]
VetController
[Component: 175]
VetRepository
[Component: 382]
VisitRepository
[Component: 456]


JdbcOwnerRepositoryImpl
[Code: 158]
EntityUtils
[Code: 54]
JdbcPet
[Code: 48]
Owner
[Code: 153]
PetType
[Code: 29]
BaseEntity
[Code: 47]
JdbcPetVisitExtractor
[Code: 54]
OwnerRepository
[Code: 64]
Pet
[Code: 113]

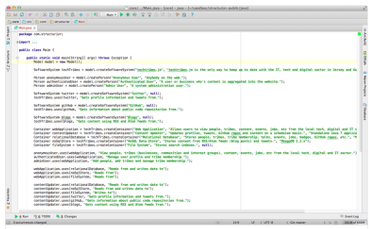

Spring PetClinic - Web Application - Components
Clinic Employee
Uses
Uses
Uses
Uses
Uses
Web Application
CrashController
OwnerController
VisitController
VetController
PetController
ClinicService
PetRepository
VetRepository
OwnerRepository
Reads from and writes to
Reads from and writes to
VisitRepository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

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.

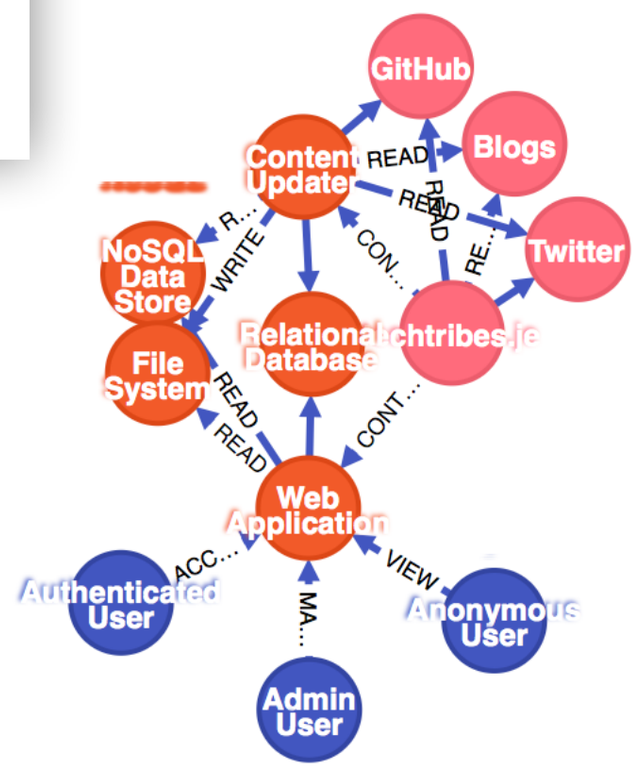
This interactive graph document describes its system architecture, and demonstrates some use-cases.

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

What do you see as the future of software architecture documentation?

Eoin: I hope that in the future we'll need very little software architecture documentation because we'll be able to see the architecture in the code and the running system! One of the reasons we need much of our architecture documentation today is because there's no way of representing architectural structures directly using the technologies we have at our disposal. I'd love to see our architectural constructs as first class implementation structures and then architecture documentation can evolve to capture decisions, rationale and analysis, rather than just capturing structures. On the way to this nirvana, I hope that work going on in the areas of DSLs and ADLs (architecture description languages) point the more immediate way forward, as we improve our description languages, on the way to working out how to embed the information right in the running system.

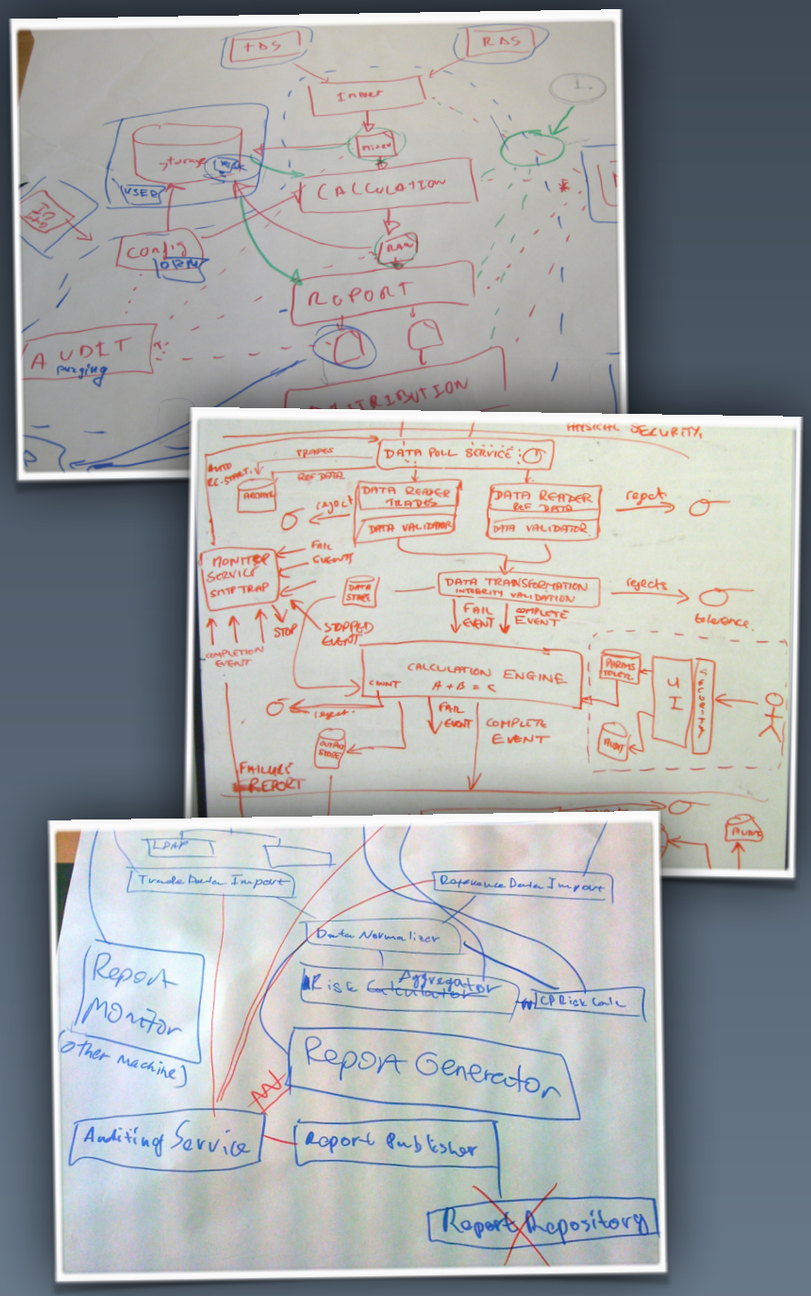
Paulo: The software architecture discipline is fairly new. There is a long path ahead until we get to a point where an architect creates architecture documentation that is readily understood by a developer who has never worked with that architect. The way to get there is to let new architects learn software architecture at school rather than try-and-error in the battlefield. This education includes proper ways to represent the software architecture for other people's consumption. Important initiatives in the direction of good software architecture education are: the work of Grady Booch on the handbook of software architecture and the publications and curriculum developed at the SEI.

Grady: There is a lot of energy being applied today with regard to architectural frameworks and methods: TOGAF, NEA, DoDAF, MoDAF, FSAM, Zachman, and so on. The good news is that there is a vibrant dialog going on with regard to these frameworks and methods - but I expect there will be a shakeout/simplification over time.

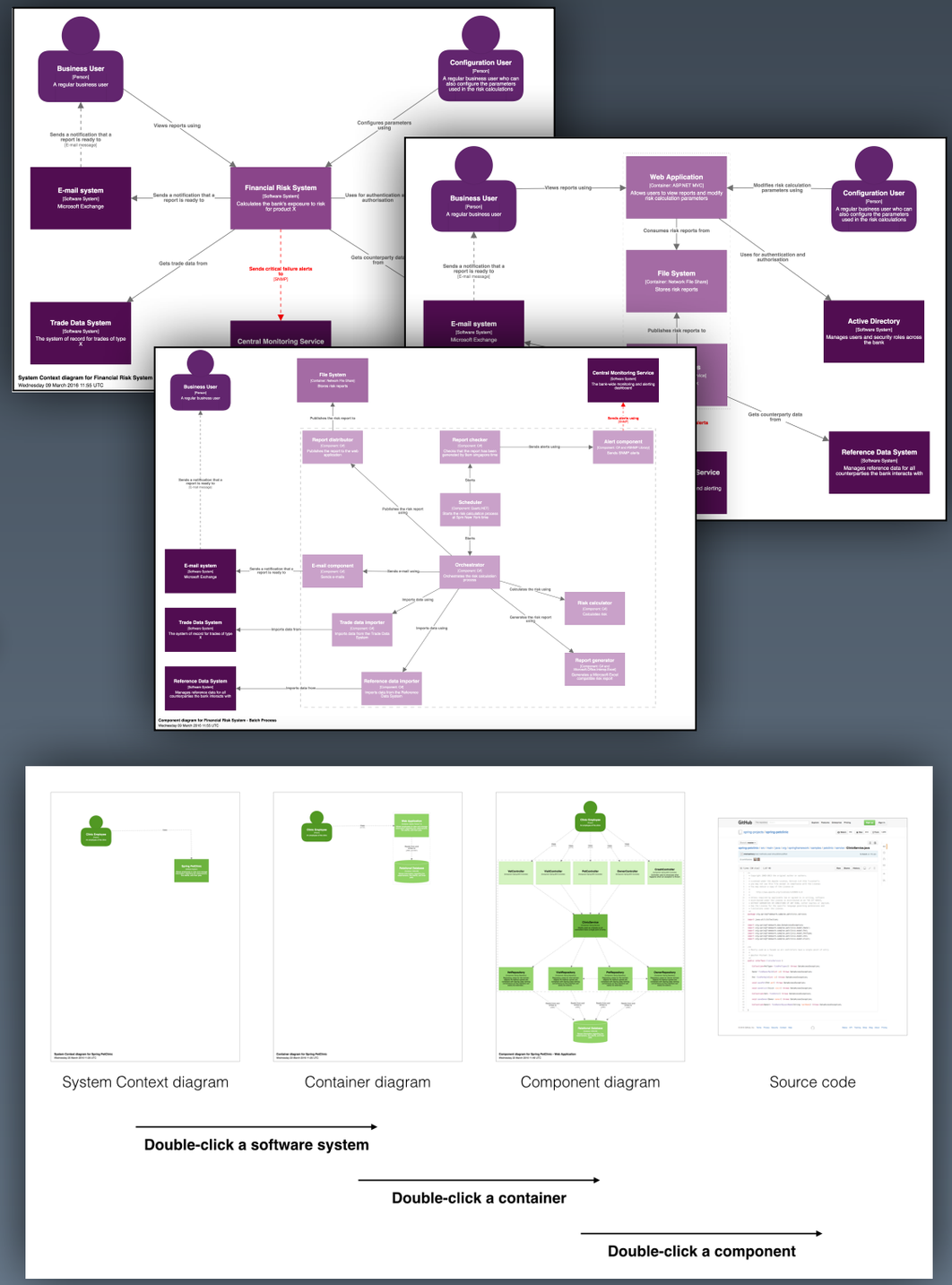
Len: The ideal development environment is one for which the documentation is available for essentially free with the push of a button. This will require an integrated development, requirements management, and project management environment. Although this will be a long time coming, it provides a worthy goal to strive for.

Virtual Panel on Software Architecture Documentation (2009)

http://www.infoq.com/articles/virtual-panel-arch-documentation



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!

Do you have a ubiquitous language to describe your software?

simon.brown@codingthearchitecture.com
@simonbrown on Twitter