

Software Architecture

as Code



Simon Brown
@simonbrown



Kristijan | Криштиџн

@Krishtidzn

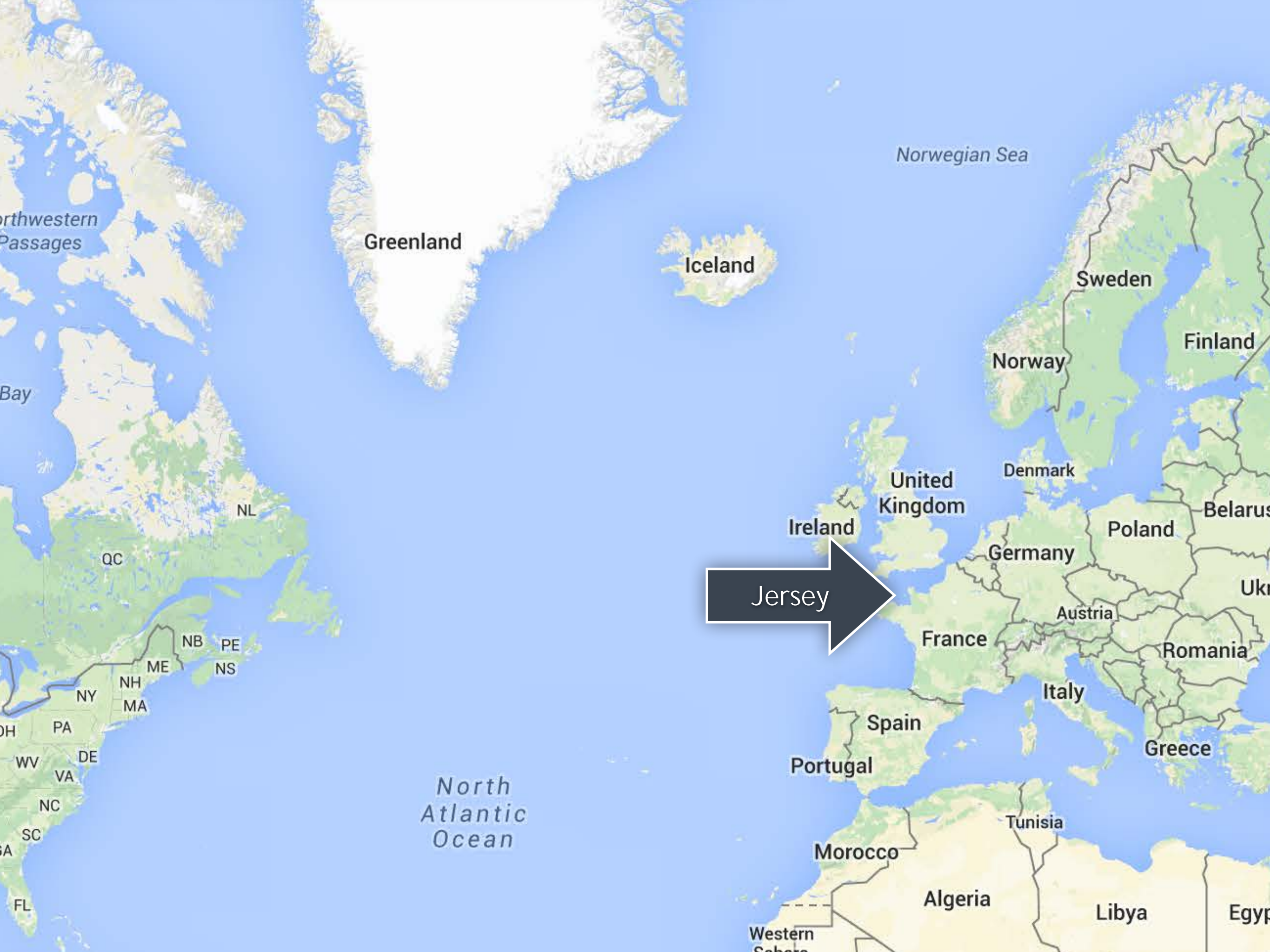


Follow

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



11:11 AM - 16 Apr 2015



Greenland

Iceland

Norwegian Sea

Sweden

Norway

Finland

Denmark

United Kingdom

Ireland

Poland

Belarus

Germany

Austria

Ukraine

France

Romania

Italy

Greece

Spain

Portugal

Tunisia

Morocco

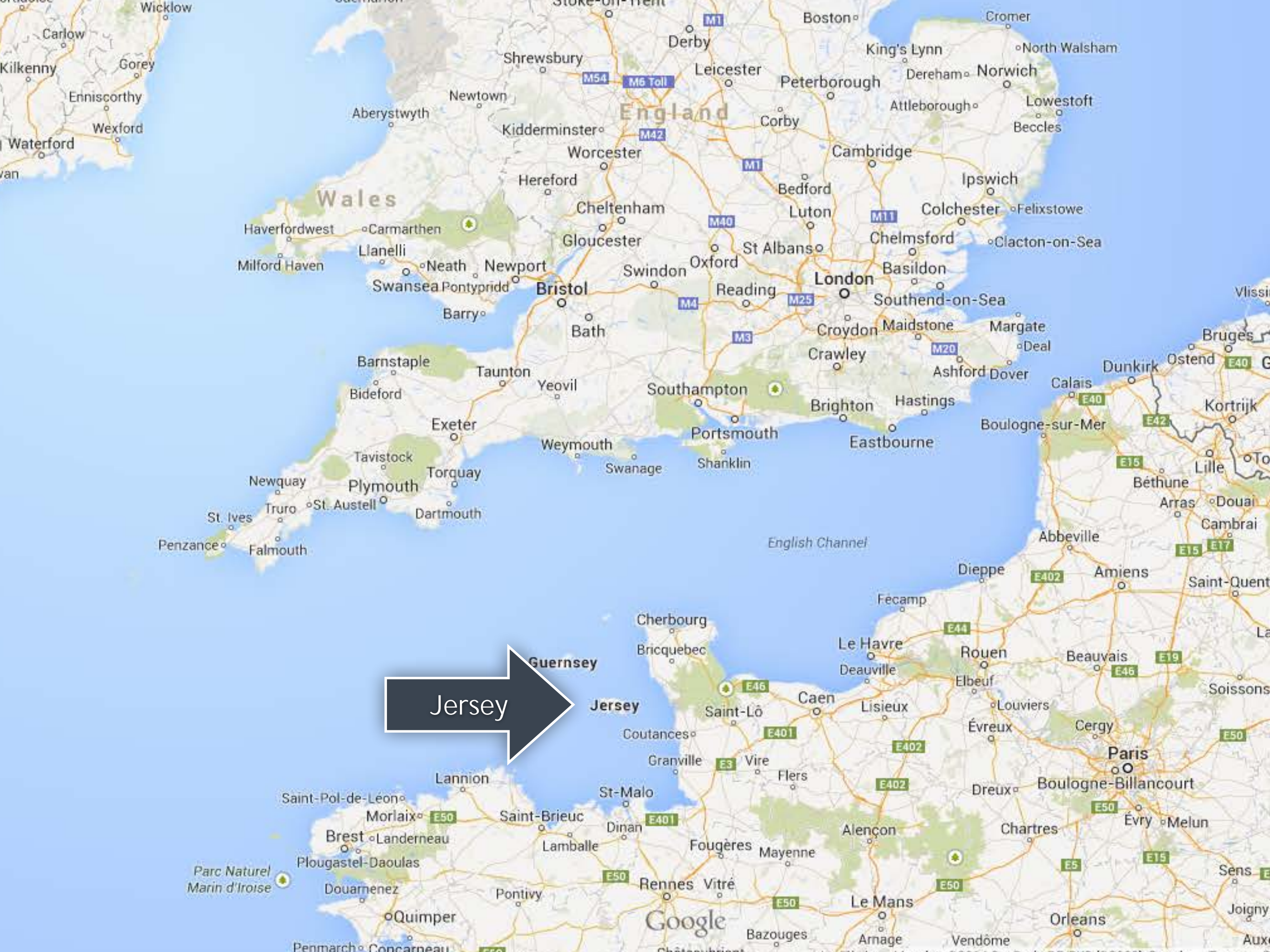
Algeria

Libya

Egypt

North Atlantic Ocean

Jersey





Guernsey



Jersey



St Ouen

Trinity

Jersey

St Martin

St Brelade

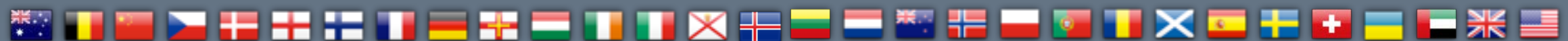
St Helier

A8

Flaman



I help software teams understand
software architecture,
technical leadership and
the balance with agility



Software architecture
needs to be more

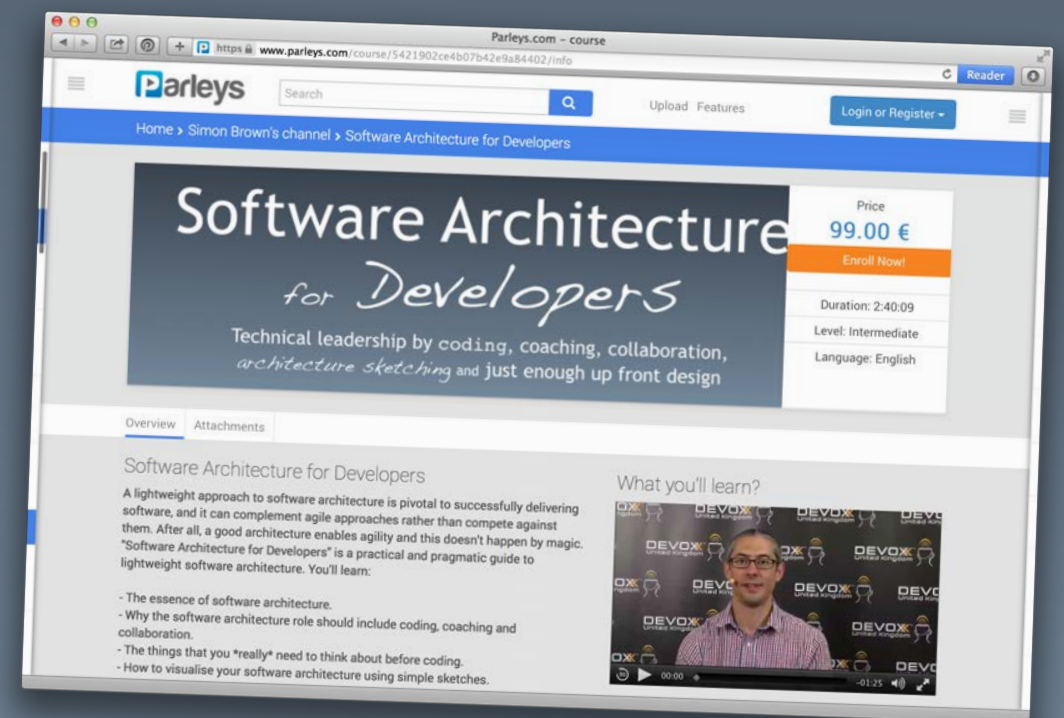
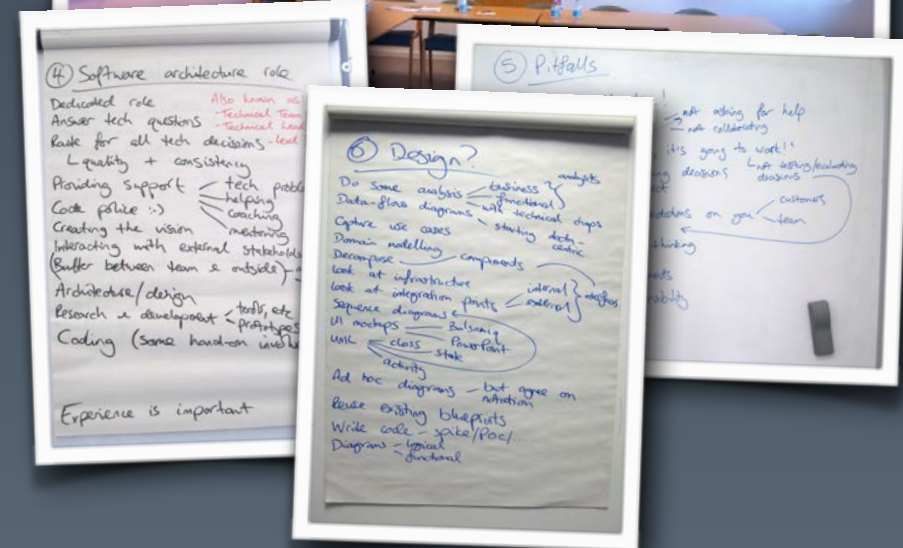
accessible

Software Architecture for Developers

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

Simon Brown

coding
(the)
architecture

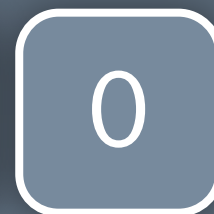
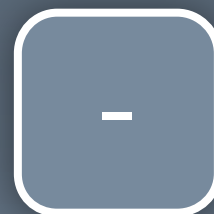


A developer-friendly guide to software architecture,
technical leadership and the balance with agility


Leanpub



I code too



The intersection between

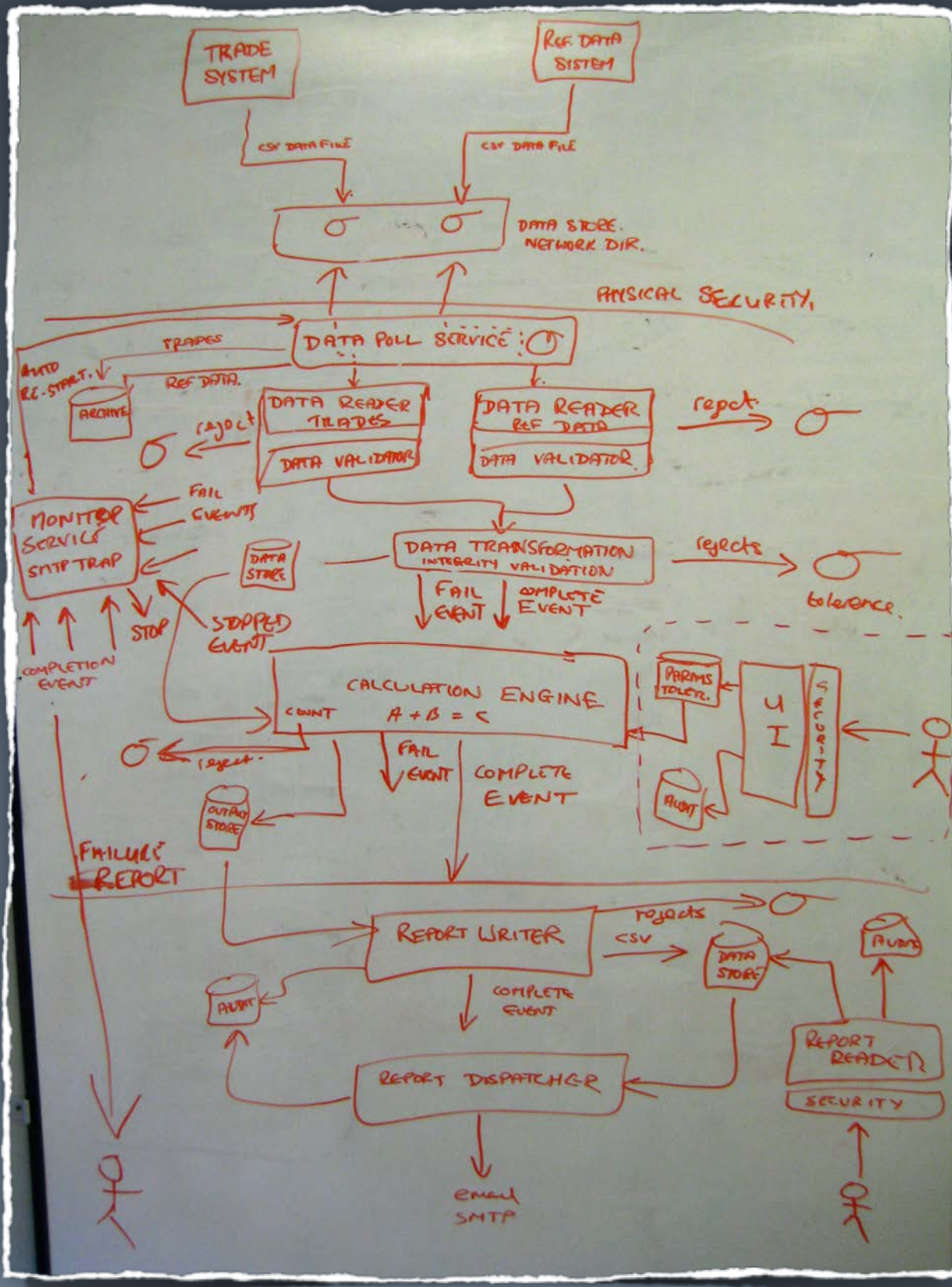
software architecture and code

How do we
communicate
software architecture?

The tension between software architecture and code

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

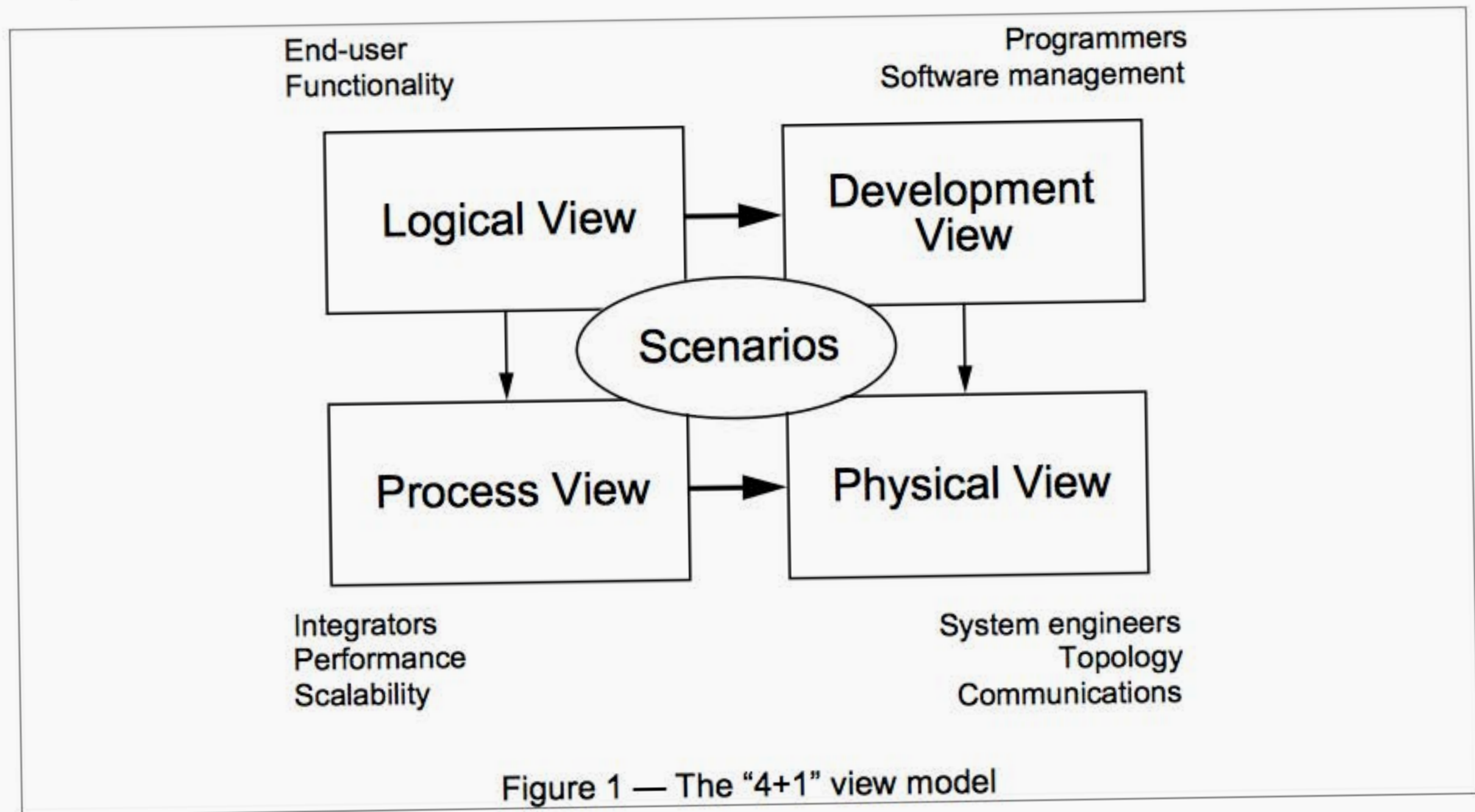


Software architecture deals with abstraction, with decomposition and composition, with style and esthetics.

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.



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.

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

Logical and
development

views are often

separated

BRAIN

FREEZE!

Cookies help us deliver our services. By using our services, you agree to our use of cookies.

[Learn more](#) [Got it](#)

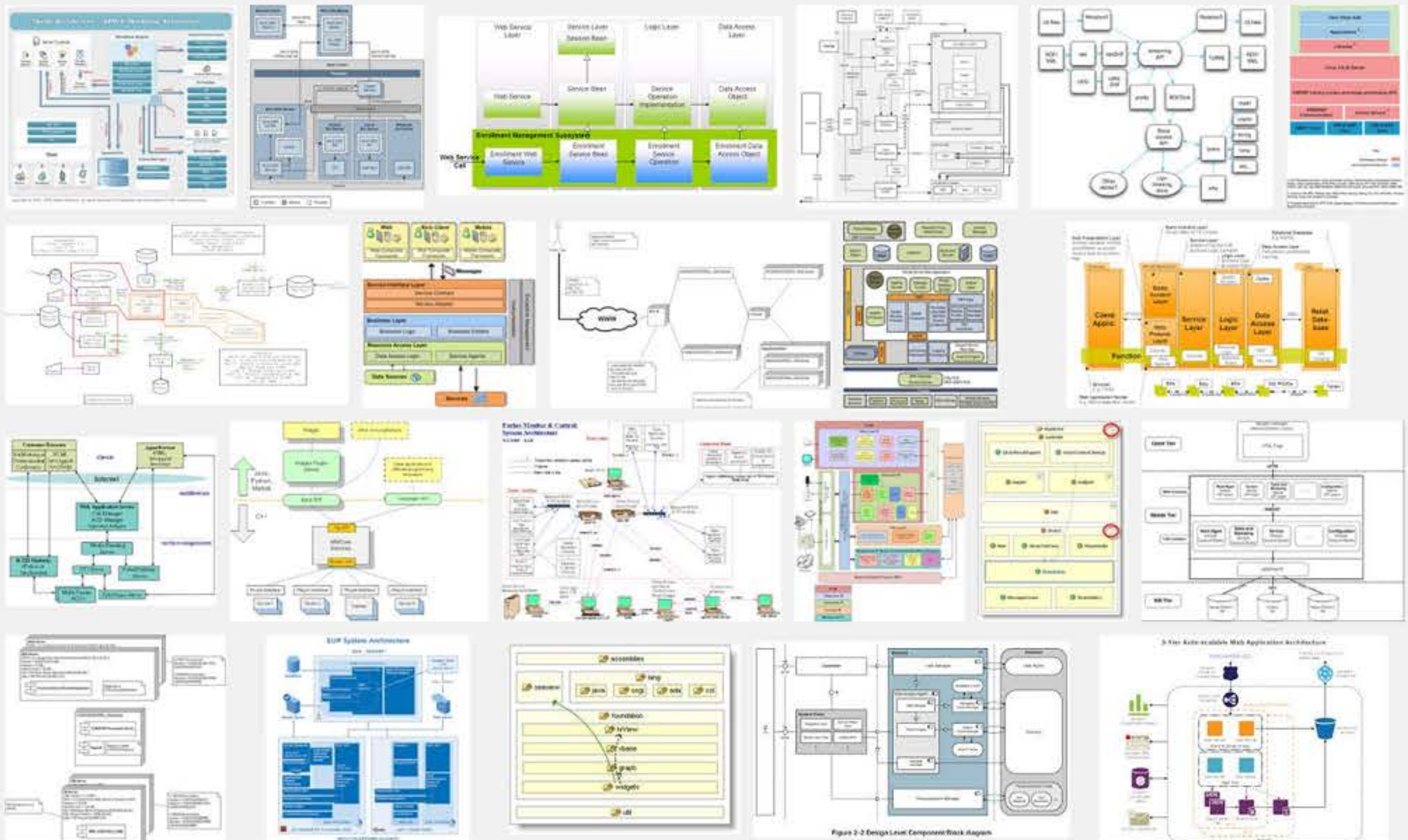
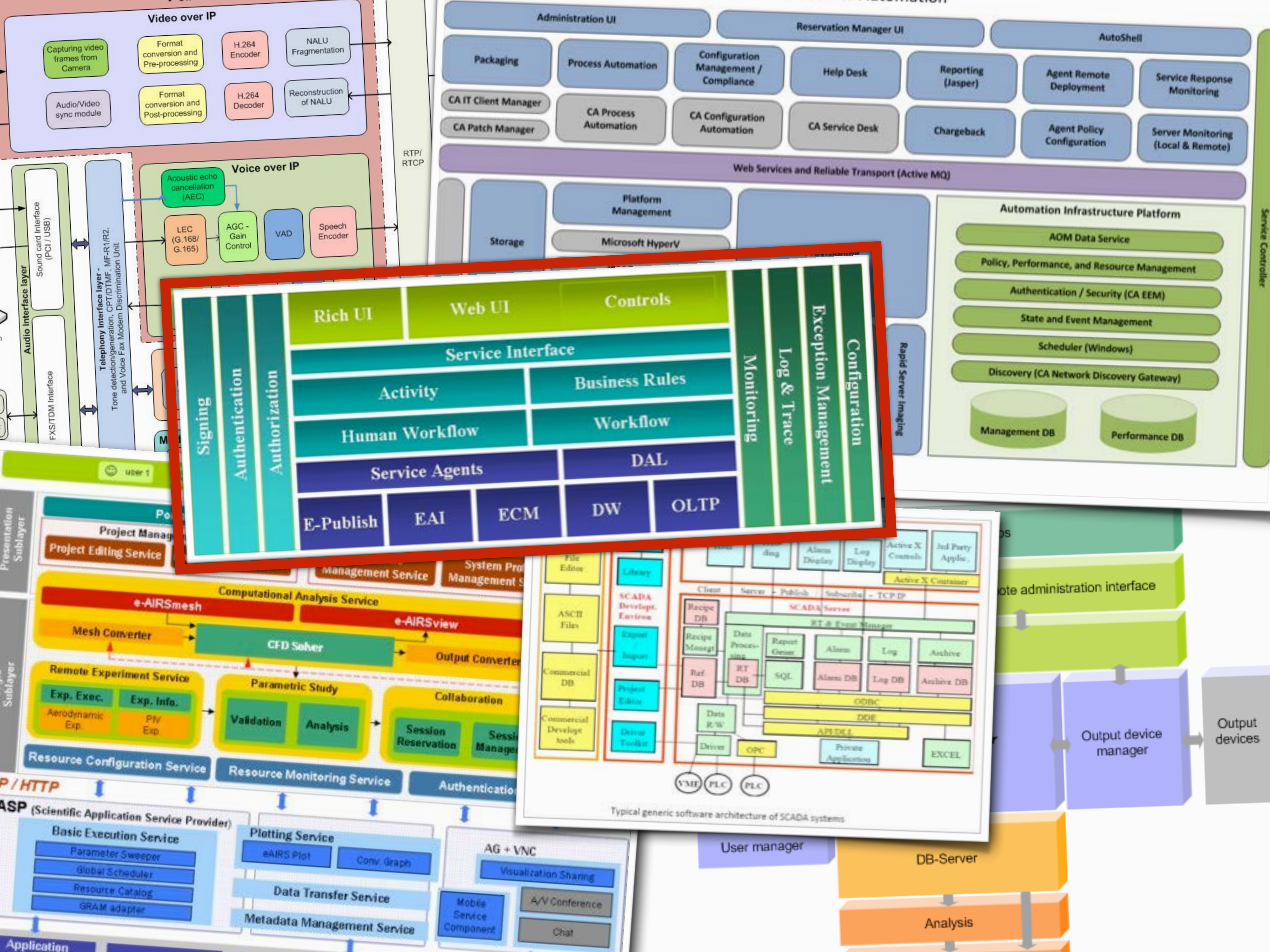


Figure 2-2 Design Level Component Work Diagram

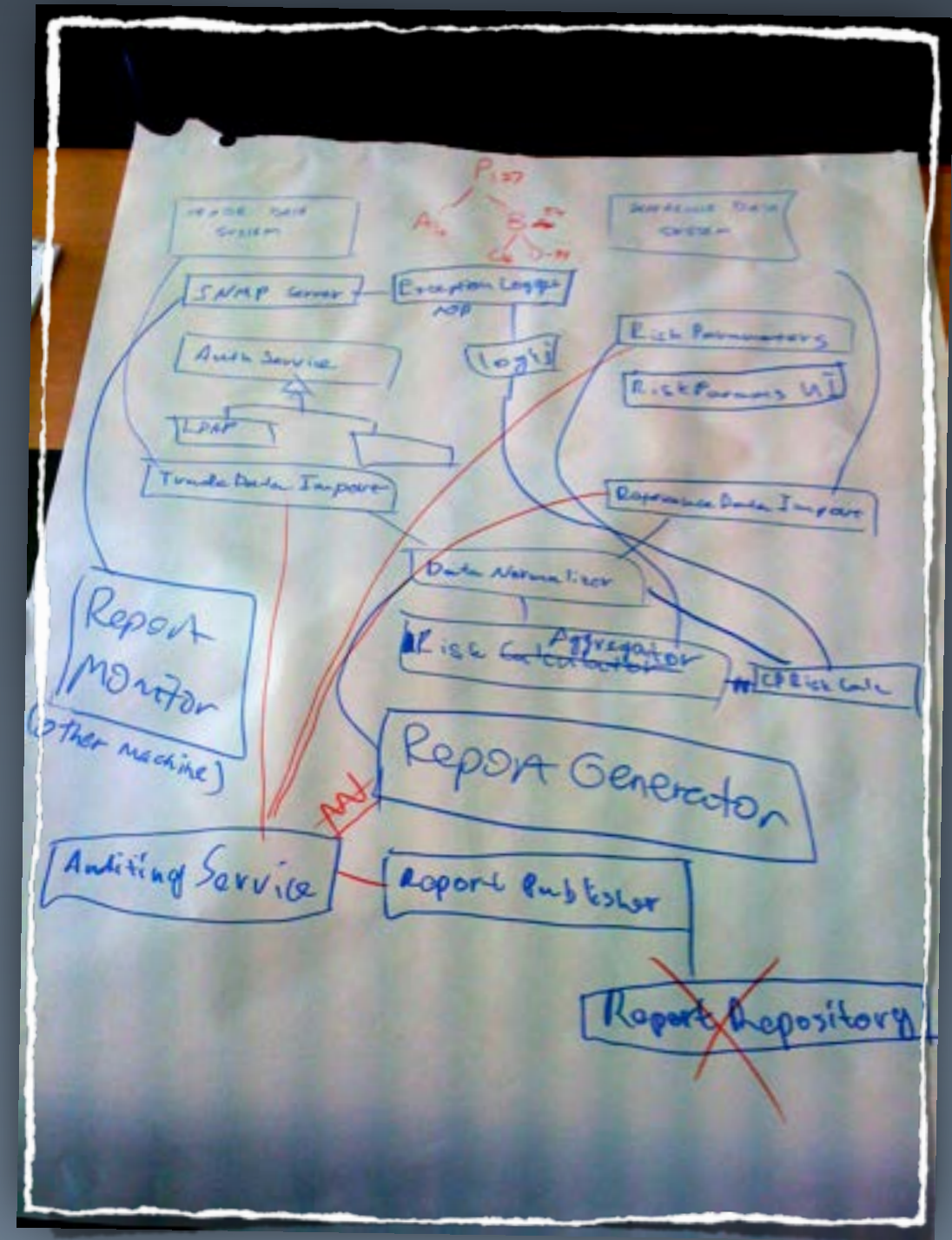


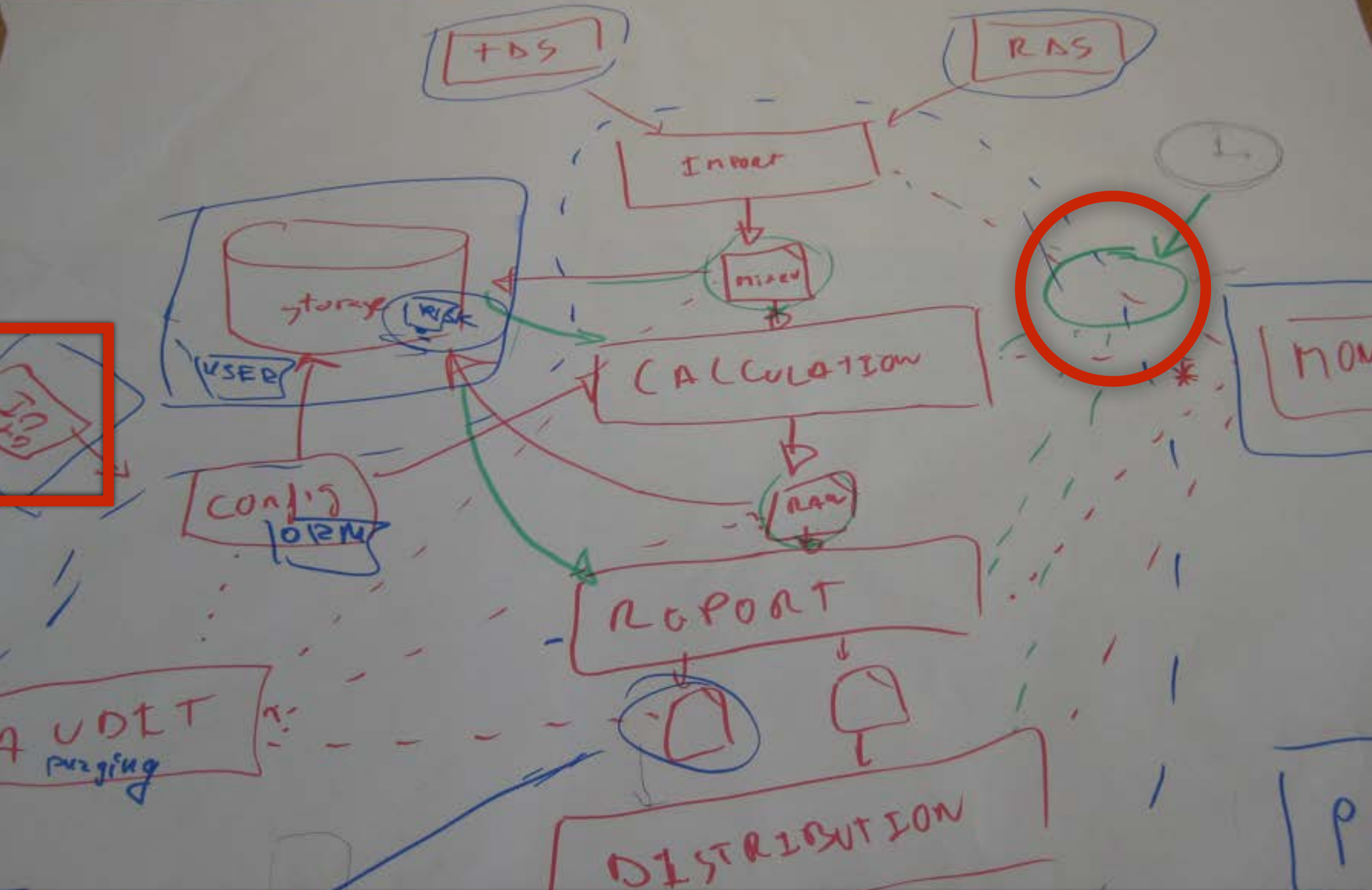
9 out of 10 people
don't use UML

(in my experience)

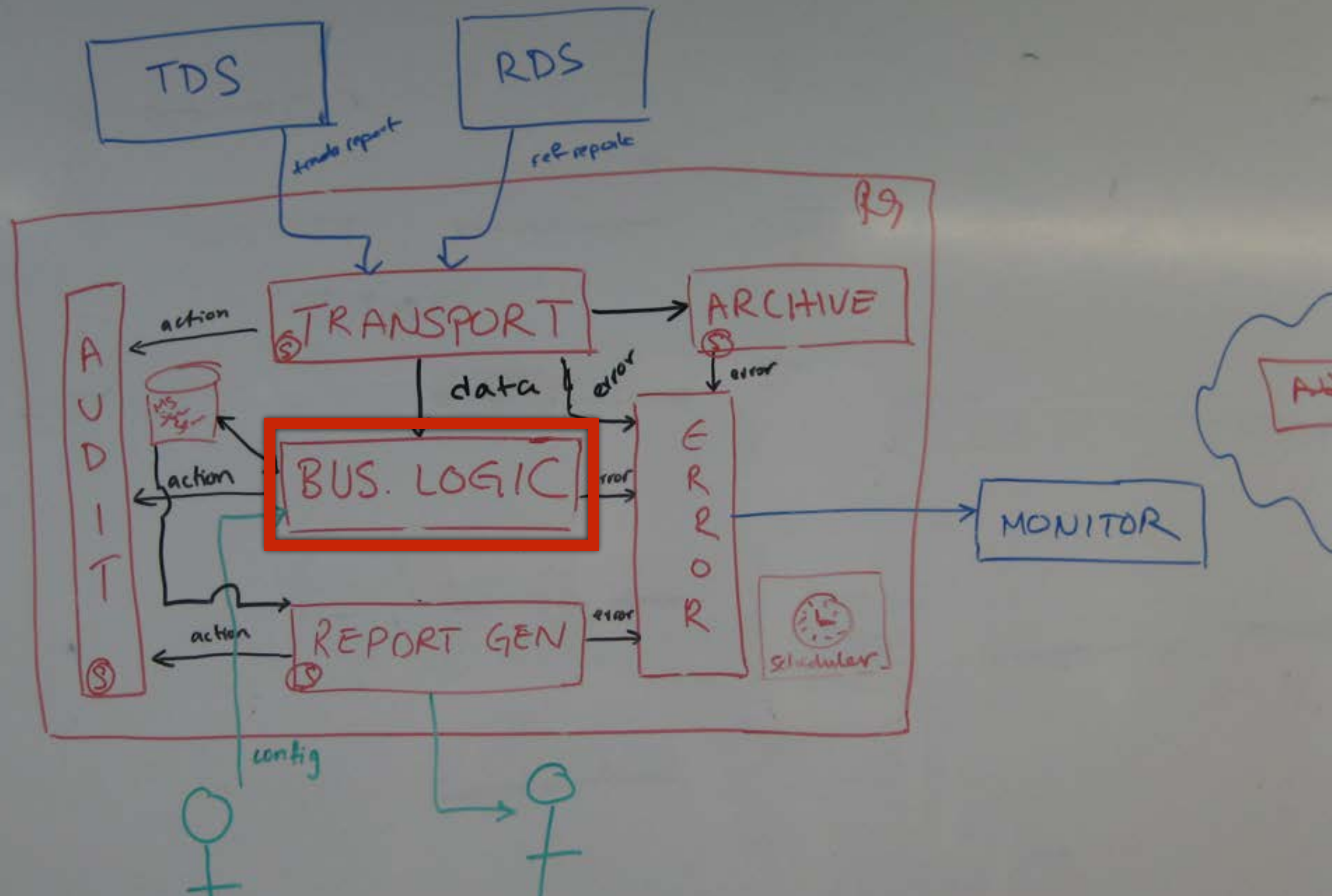


In my experience,
software teams
aren't able to
effectively
communicate
the software
architecture
of their systems

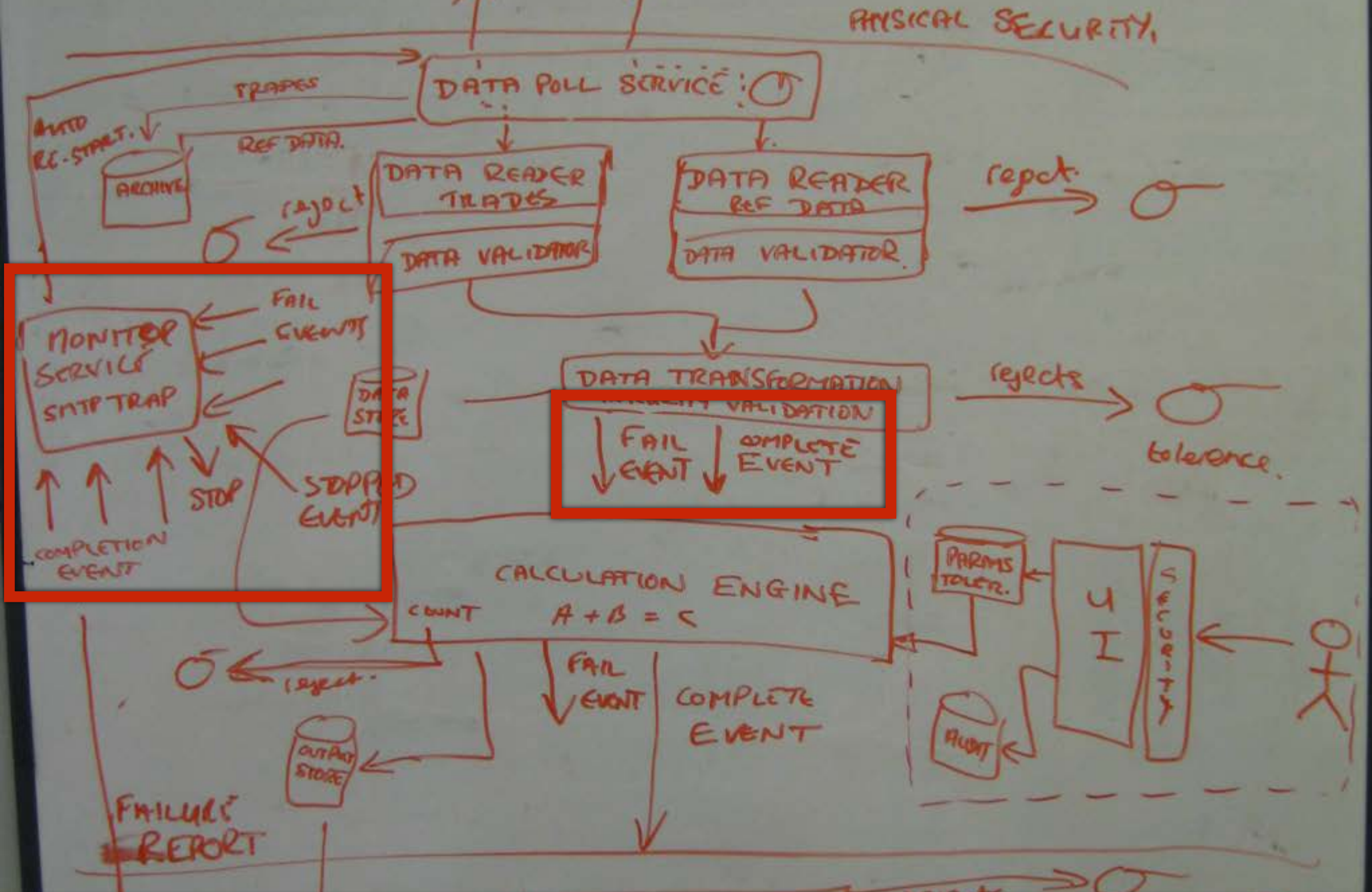




The Airline Route Map



Generically True



Choose your own adventure

Would we

code

it that way?

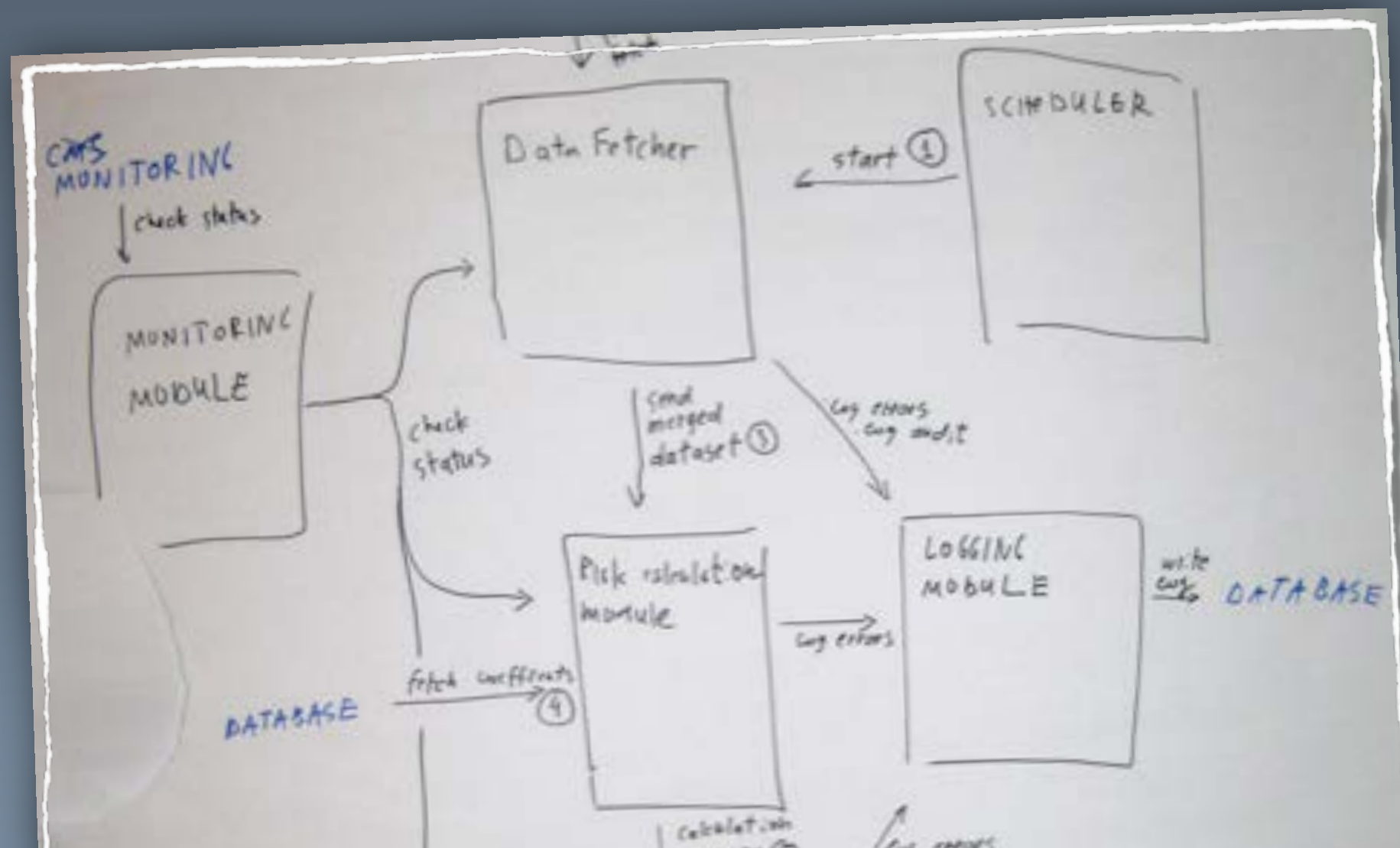
Did we

code

it that way?

Abstraction

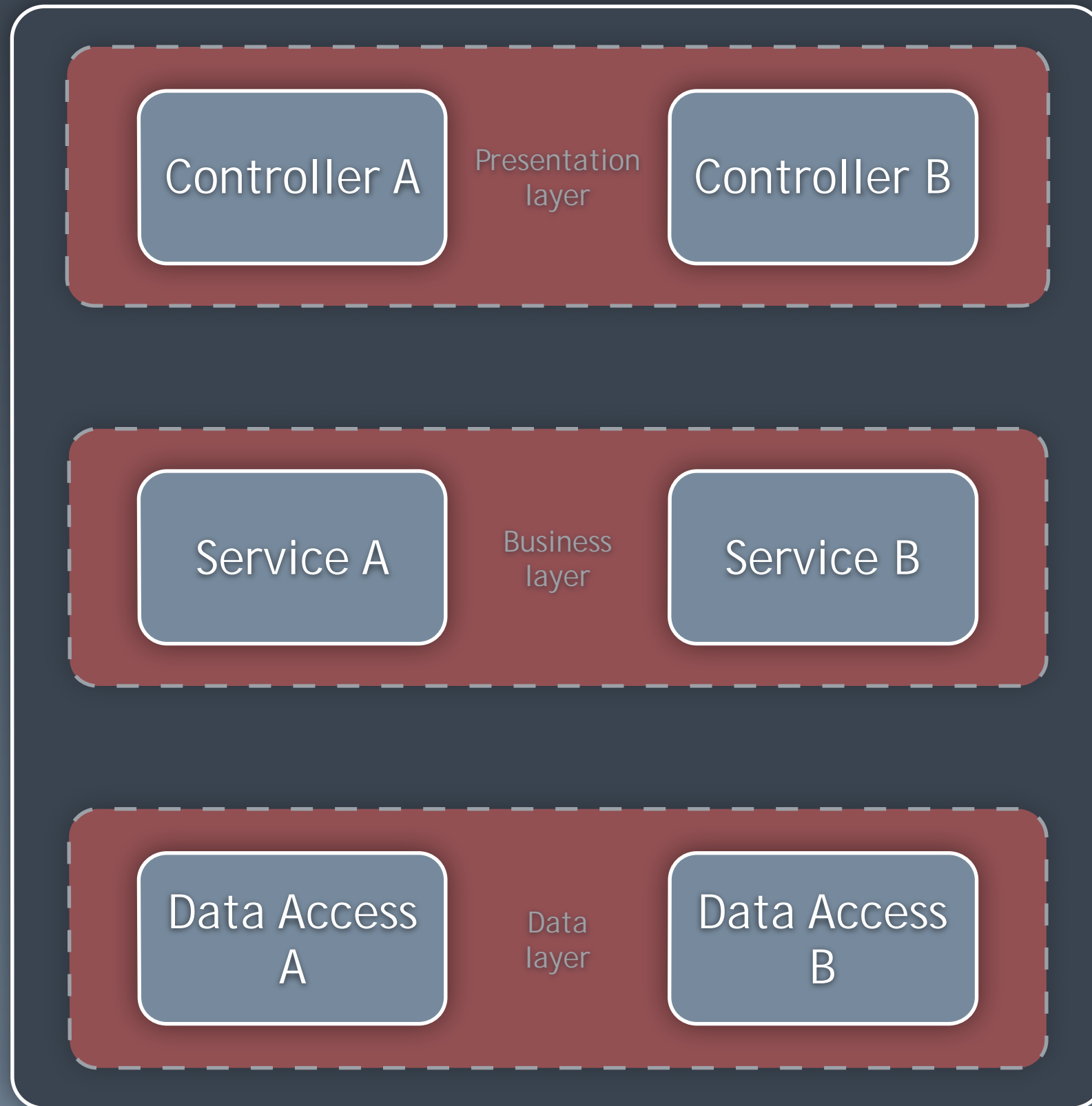
is about reducing detail
rather than creating a different representation



Abstractions help us
reason about
a big and/or complex
software system

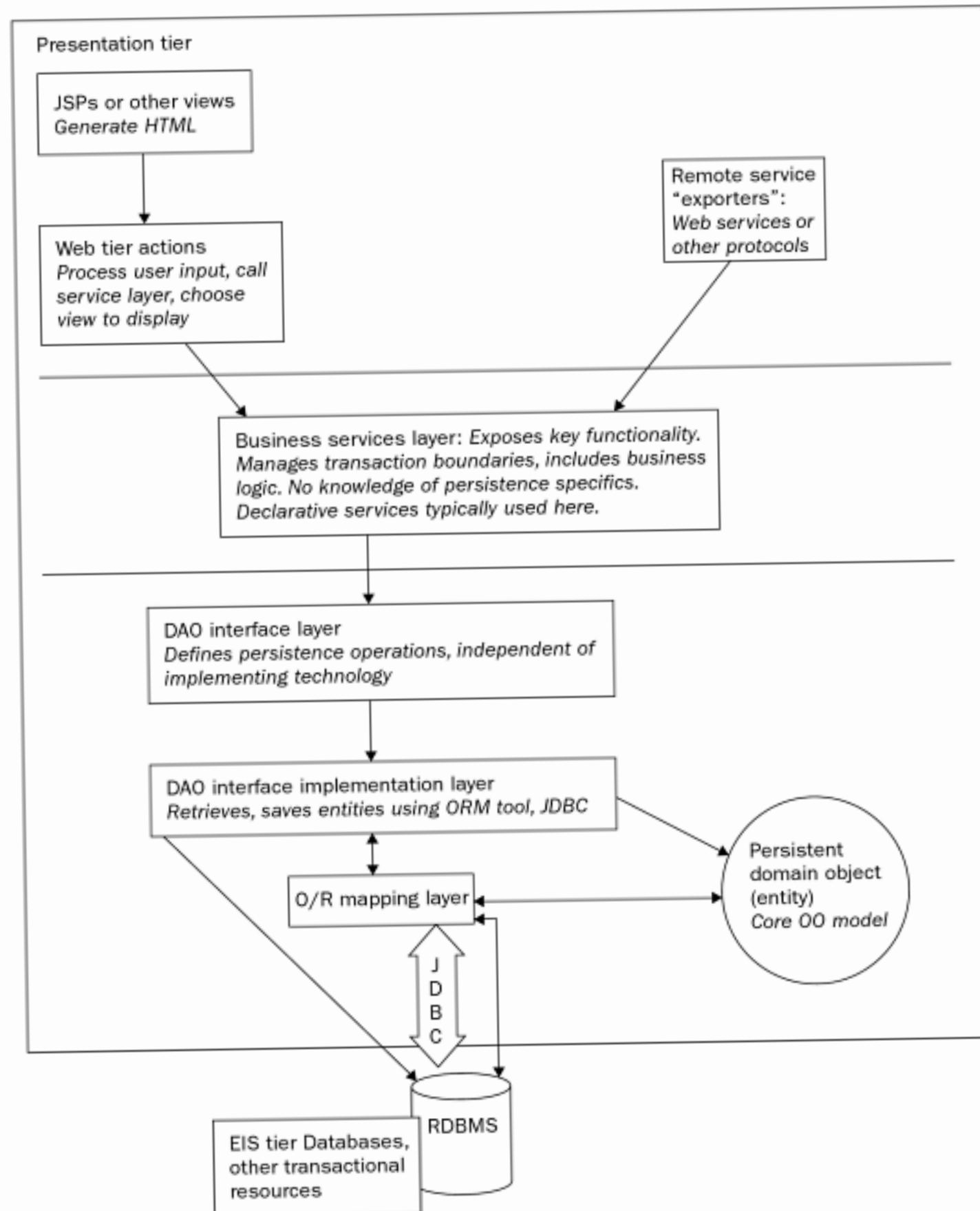
Does your code reflect the
abstractions
that you think about?

We often think
in components
but write classes
(usually in layers)



Package by layer (horizontal slicing)

Chapter 1



Let's summarize each layer and its responsibilities, beginning closest to the database or other enterprise resources:

- ❑ **Presentation layer:** This is most likely to be a web tier. This layer should be as thin as possible. It should be possible to have alternative presentation layers — such as a web tier or remote web services facade — on a single, well-designed middle tier.
- ❑ **Business services layer:** This is responsible for transactional boundaries and providing an entry point for operations on the system as a whole. This layer should have no knowledge of presentation concerns, and should be reusable.
- ❑ **DAO interface layer:** This is a layer of interfaces *independent of any data access technology* that is used to find and persist persistent objects. This layer effectively consists of *Strategy* interfaces for the Business services layer. This layer should not contain business logic. Implementations of these interfaces will normally use an O/R mapping technology or Spring's JDBC abstraction.
- ❑ **Persistent domain objects:** These model real objects or concepts such as a bank account.
- ❑ **Databases and legacy systems:** By far the most common case is a single RDBMS. However, there may be multiple databases, or a mix of databases and other transactional or non-transactional legacy systems or other enterprise resources. The same fundamental architecture is applicable in either case. This is often referred to as the *EIS (Enterprise Information System)* tier.

In a J2EE application, all layers except the EIS tier will run in the application server or web container. Domain objects will typically be passed up to the presentation layer, which will display data they contain, *but not modify them*, which will occur only within the transactional boundaries defined by the business services layer. Thus there is no need for distinct Transfer Objects, as used in traditional J2EE architecture.

In the following sections we'll discuss each of these layers in turn, beginning closest to the database.

Spring aims to decouple architectural layers, so that each layer can be modified as

Spring aims to decouple architectural layers, so that each layer can be modified as far as possible without impacting other layers. No layer is aware of the concerns of the layer above; as far as possible, dependency is purely on the layer immediately below. Dependency between layers is normally in the form of interfaces, ensuring that coupling is as loose as possible.



Package Explorer

JUnit

- ▼ springapp
 - ▼ src
 - ▼ springapp.domain
 - ▶ Product.java
 - ▼ springapp.repository
 - ▶ JdbcProductDao.java
 - ▶ ProductDao.java
 - ▼ springapp.service
 - ▶ PriceIncrease.java
 - ▶ PriceIncreaseValidator.java
 - ▶ ProductManager.java
 - ▶ SimpleProductManager.java
 - ▶ springapp.web
 - ▶ test
 - ▶ JRE System Library [JVM 1.5.0 (MacOS X Default)]
 - ▶ Referenced Libraries
 - ▶ db
 - ▼ war
 - ▼ WEB-INF
 - ▼ classes
 - ▶ springapp
 - ▶ jdbc.properties
 - ▶ messages.properties
 - ▼ jsp
 - ▶ hello.jsp
 - ▶ include.jsp

1 Answer

active

oldest

votes

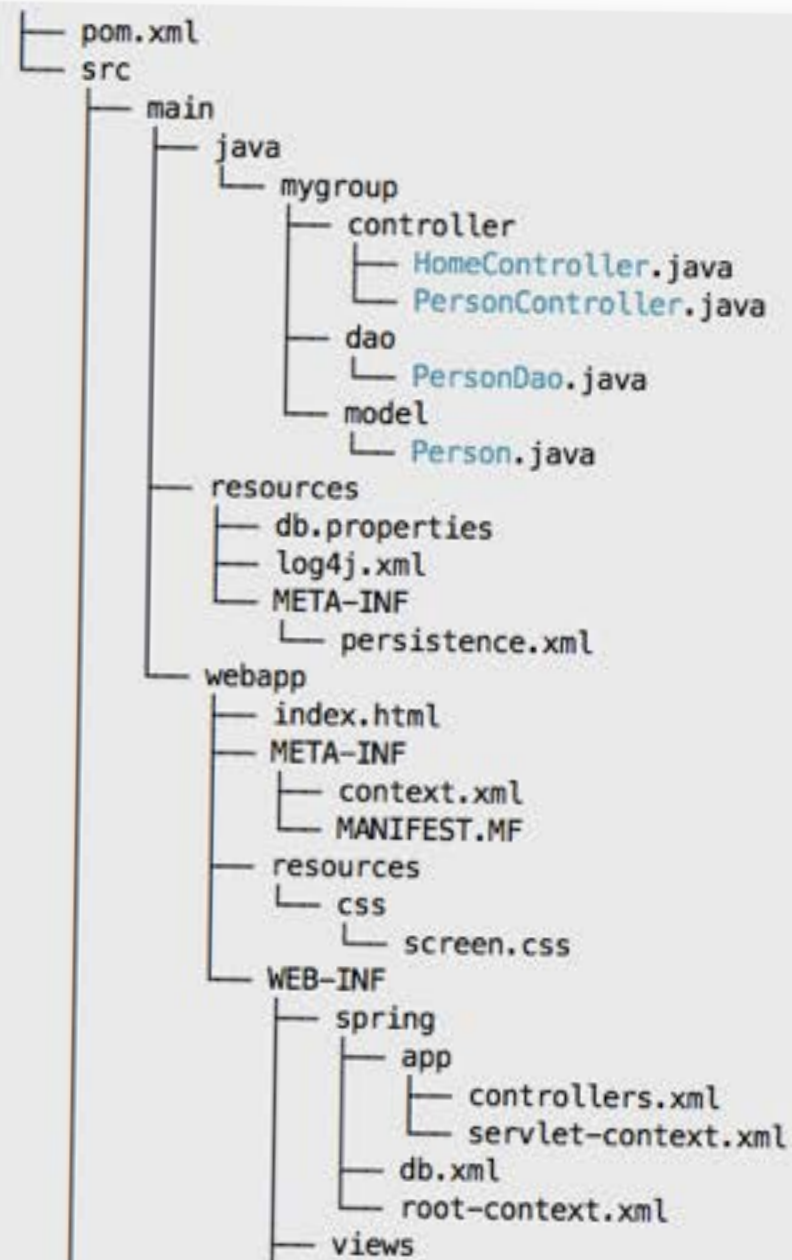
8

If you are using maven, it's best to follow the standard maven project layout. You can get maven to generate this structure for you by doing,

```
mvn archetype:generate
```

and select spring-mvc-jpa-archetype from the list of choices

This will give you a package structure like,



Technical Operations Engineer,
Devops, Cloud Systems and AWS
Voyanta

London, United Kingdom

Linux Service Engineers
Shazam

London, United Kingdom

3 x Senior Engineer (Ruby) 3
month+ contract

We are Friday Limited
London, United Kingdom

[More jobs near London...](#)

Related

- 2 How do I use a custom authentication mechanism for a Java web application with Spring Security?
- 0 Is there a sample web project of integrating Spring-MVC and Spring-Data-JPA?
- 1 Reusing DAOs in another web application
- 0 Java Web Application Warming
- 3 Is a parallel Spring-MVC application possible with a non-spring web app?
- 2 Adding a web interface (Spring MVC) to existing Java application
- 0 package structure for Spring MVC project with multiple sub projects using maven
- 0 How to add a setup



FILE

EDIT

VIEW

PROJECT

BUILD

DEBUG

TEAM

TOOLS

TEST

WINDOW

HELP



HomeController.cs

MvcMovie.Controllers.HomeCont

Index()

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MvcMovie.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.Message = "Modify this template to create your app's home page."

            return View();
        }

        public ActionResult About()
        {
            ViewBag.Message = "Your app description page."

            return View();
        }
    }
}
```

100 %

Ready

Ln 1

Col 1

Ch 1

INS

Solution Explorer

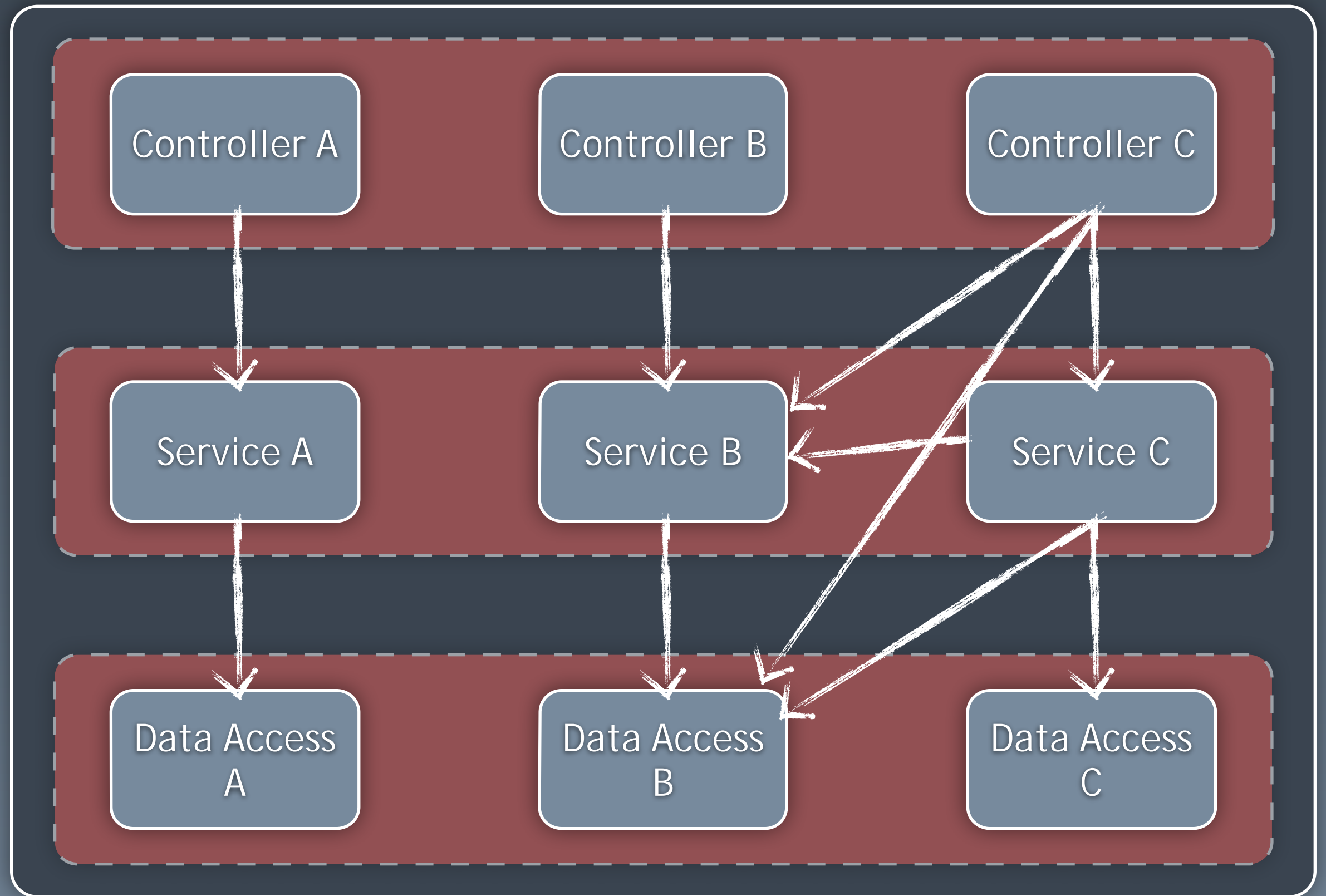


Search Solution Explorer (Ctrl+;)

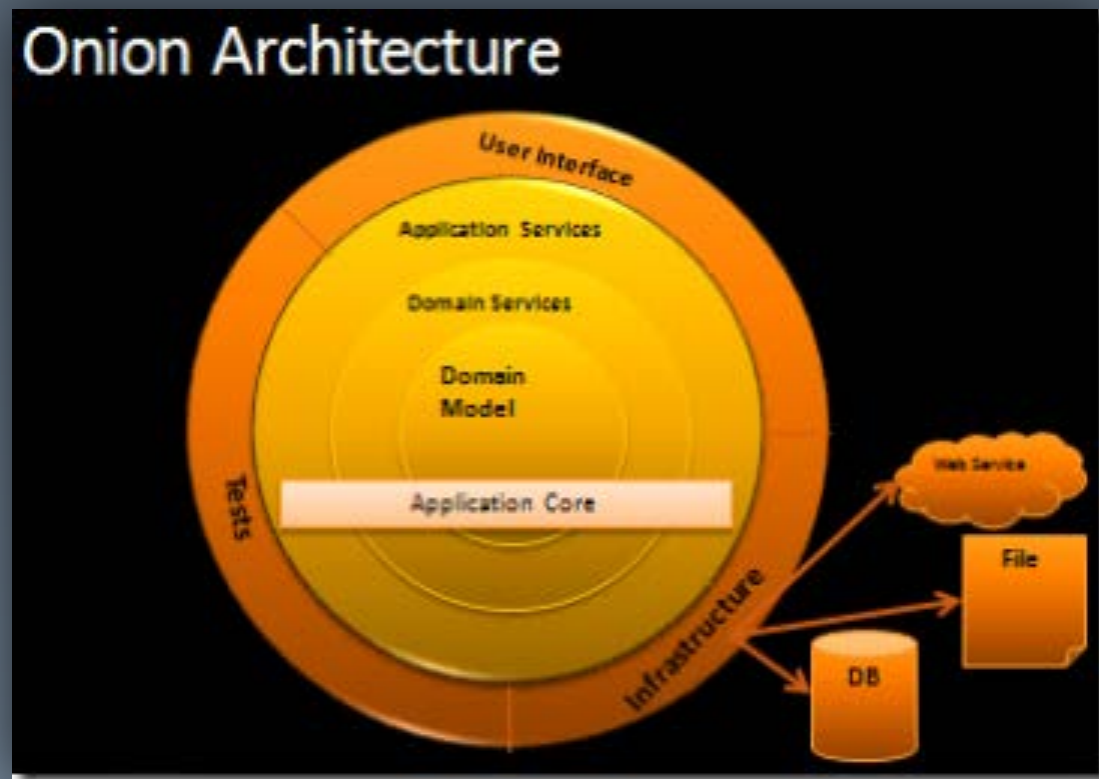
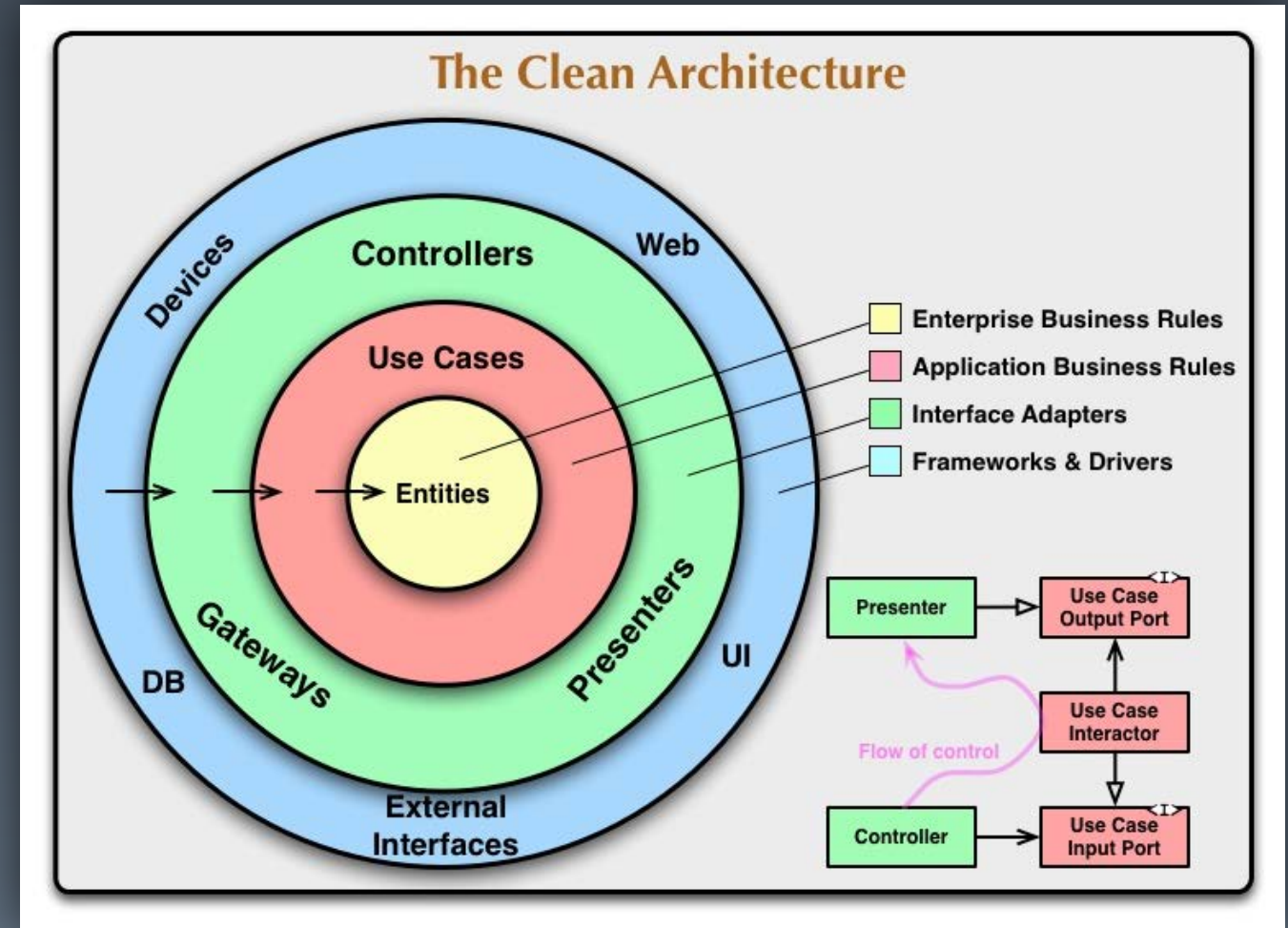
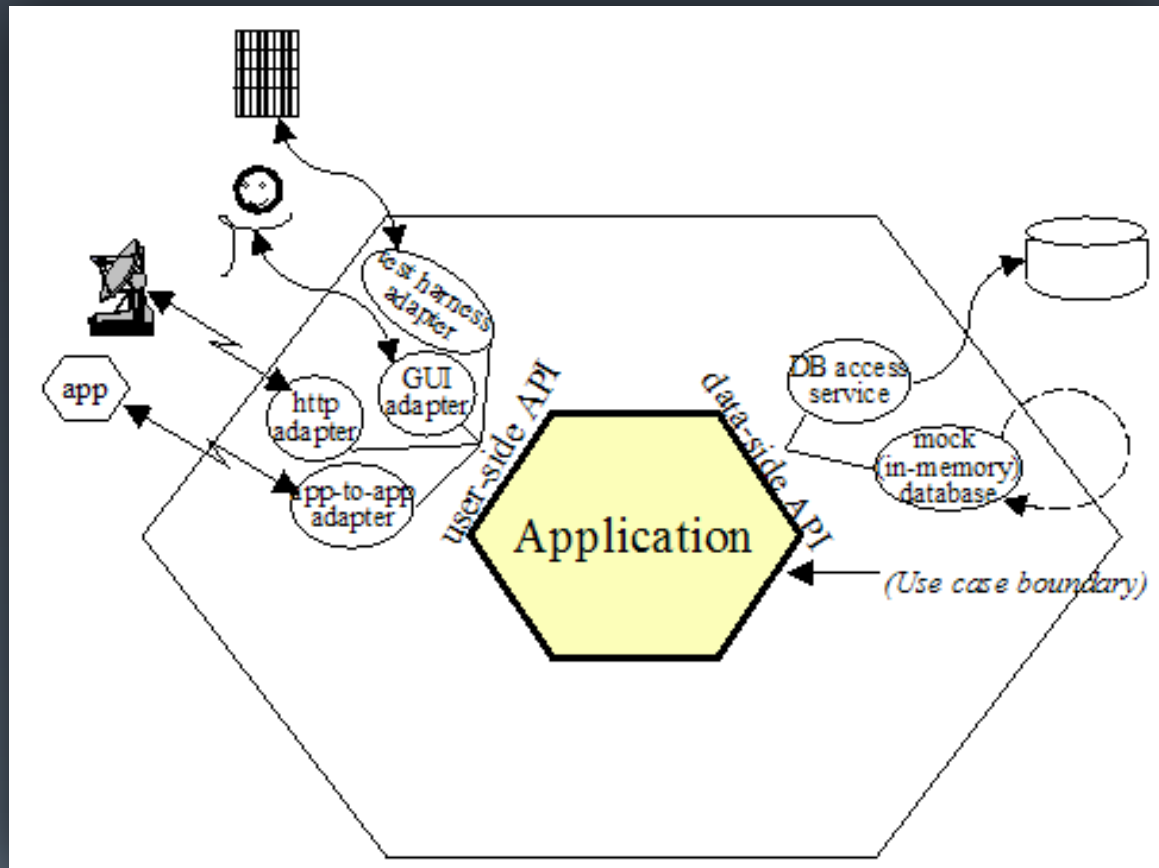
Solution 'MvcMovie' (1 project)

MvcMovie

- Properties
- References
- App_Data
- App_Start
- Content
- Controllers
- Filters
- Images
- Models
- Scripts
- Views
- favicon.ico
- Global.asax
- packages.config
- Web.config



Package by layer (horizontal slicing)



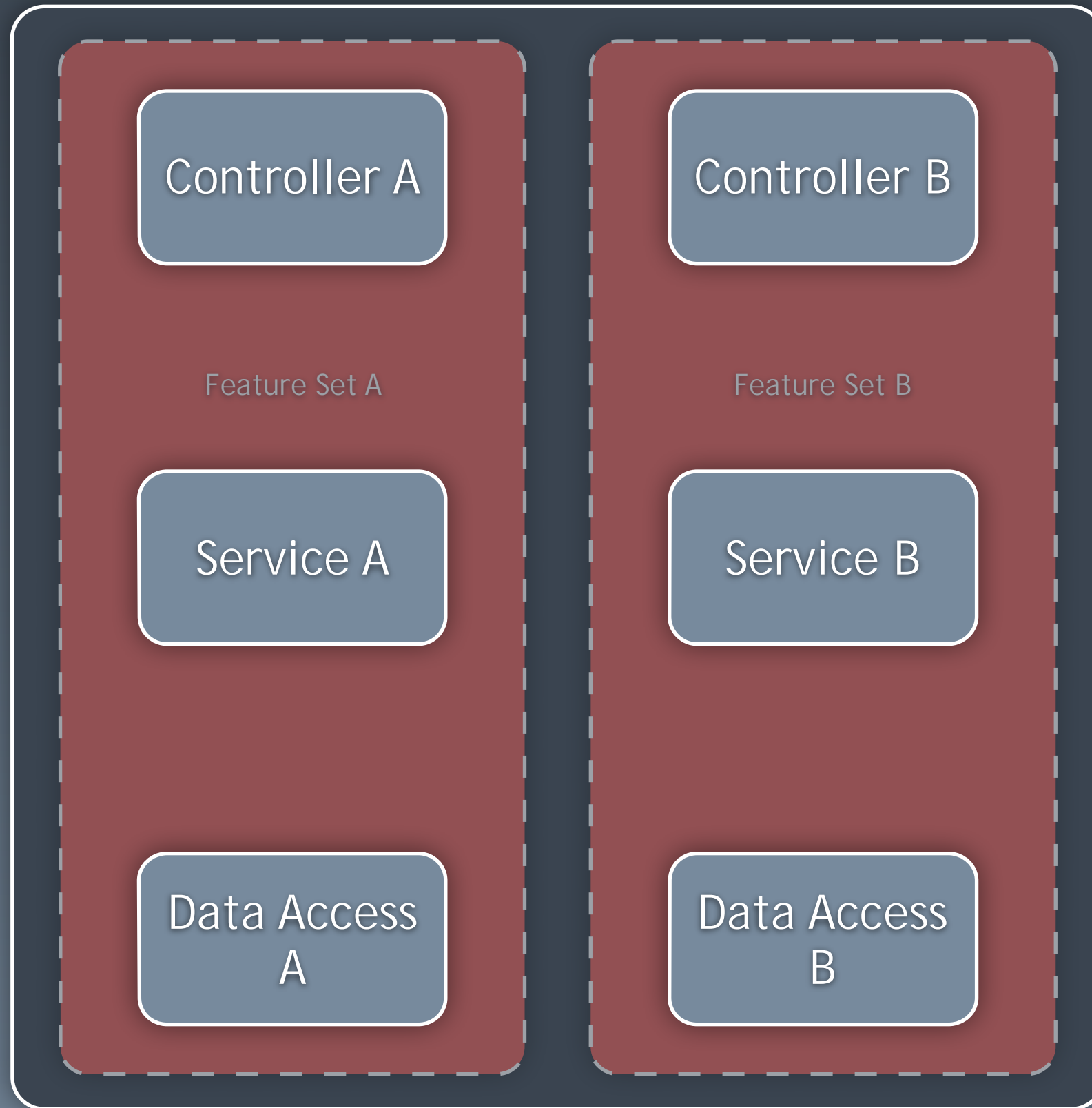
Hexagons
and onions

Should layers be
considered
harmful?

Are layers significant
structural
elements

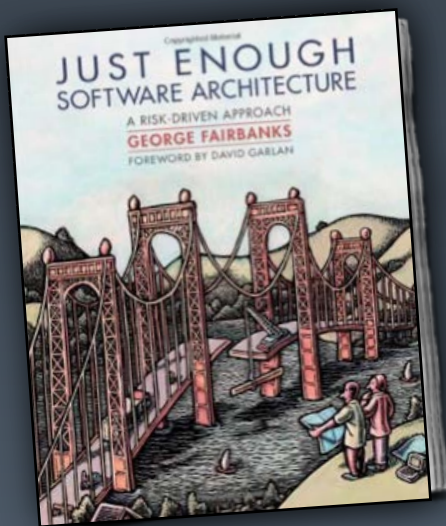
or just an

implementation
detail?



Package by feature (vertical slicing)

Organisation of code vs the architectural views



“the model-code gap”

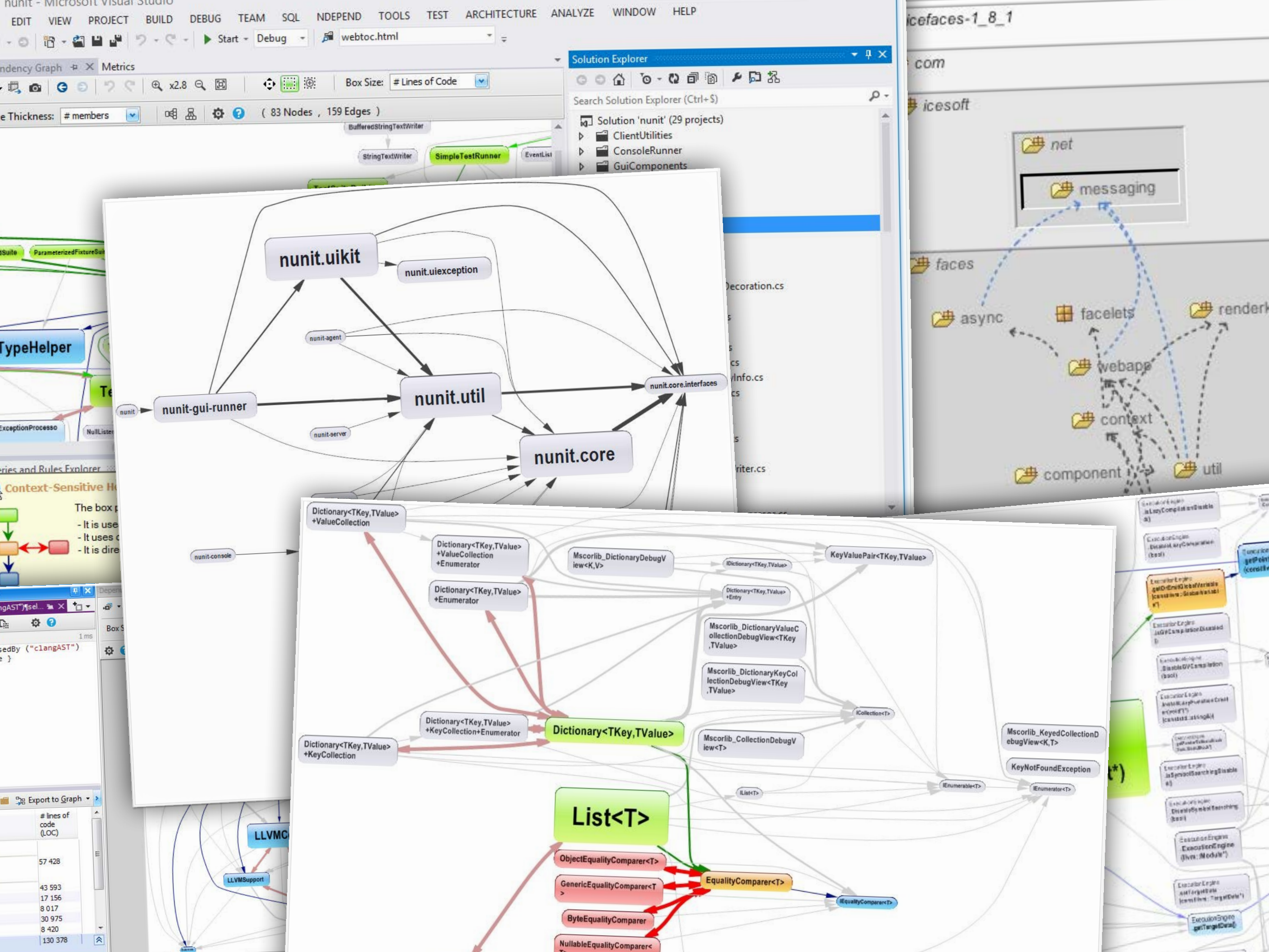
Model-code gap. Your architecture models and your source code will not show the same things. The difference between them is the *model-code gap*. Your architecture models include some abstract concepts, like components, that your programming language does not, but could. Beyond that, architecture models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

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

Diagramming tools see
packages
and **classes**
rather than components

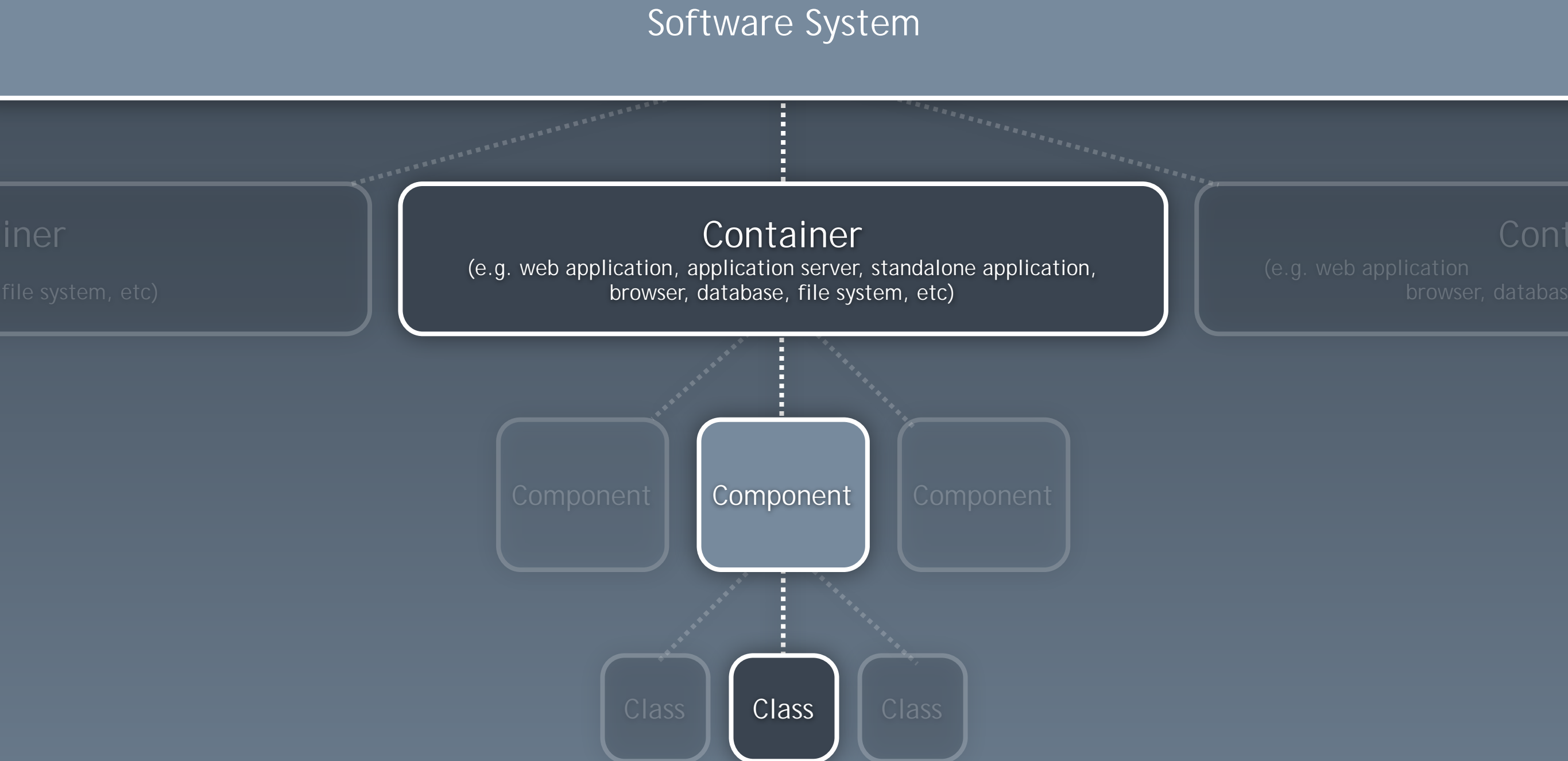


Describing software architecture

*“Modules, components
and connectors”*

Is this vocabulary
in common use?

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



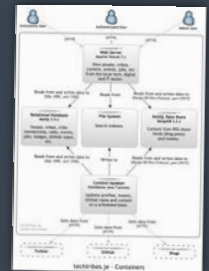
Agree on a simple set of **abstractions**
that the whole team can use to communicate

The C4 model



System Context

The system plus users and system dependencies



Containers

The overall shape of the architecture and technology choices



Components

Logical components and their interactions within a container

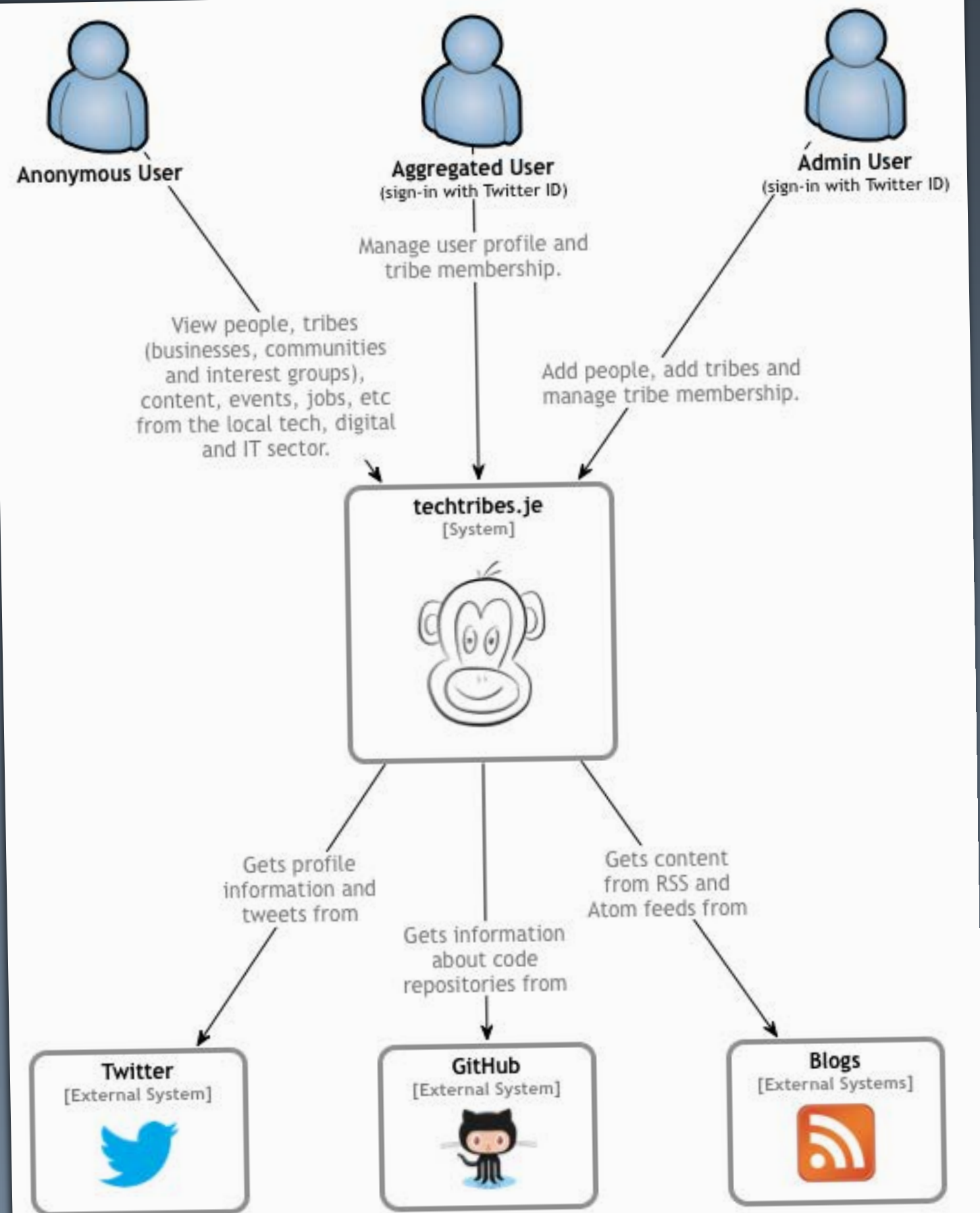


Classes

Component or pattern implementation details

Context

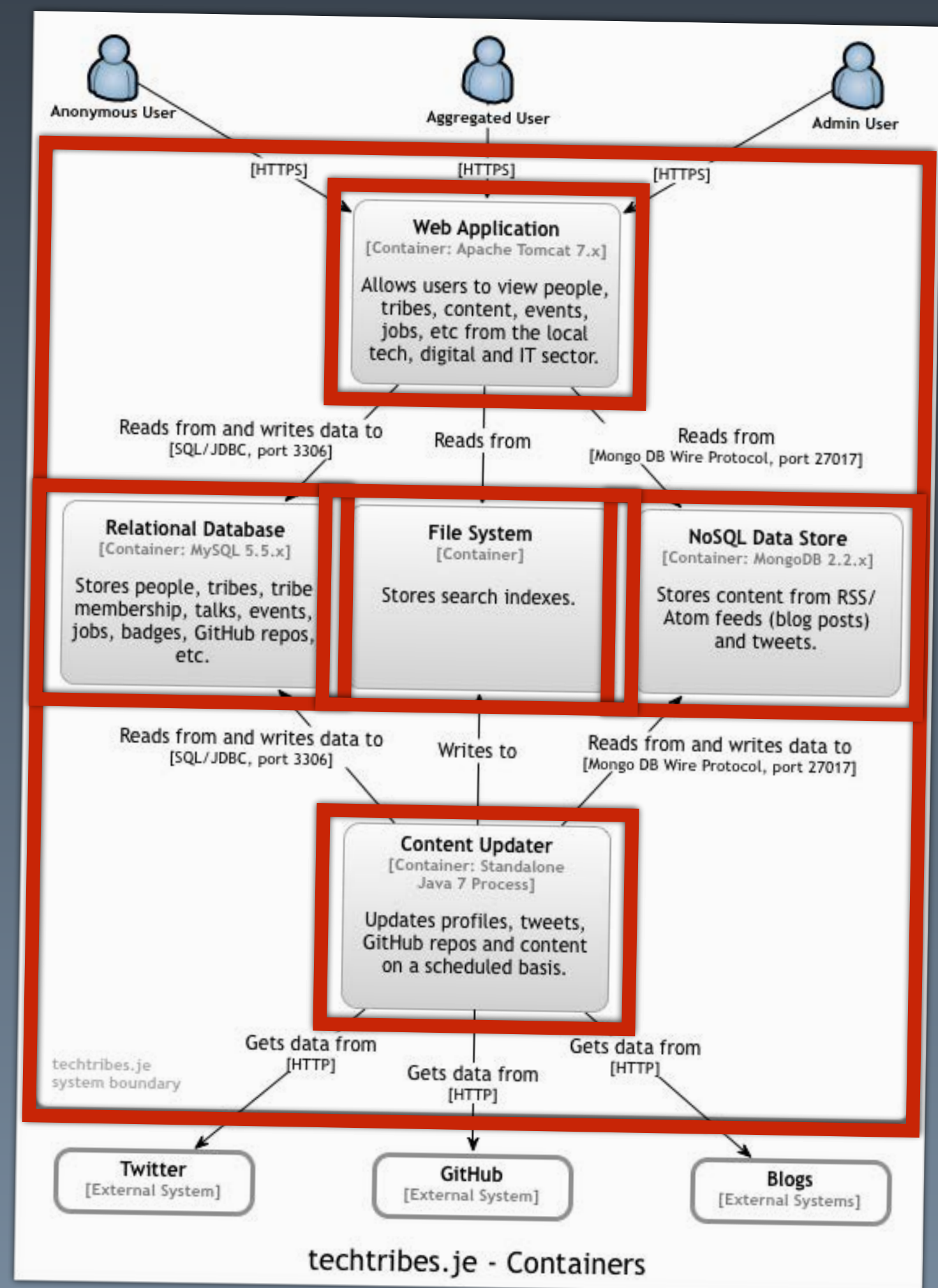
- What are we building?
- Who is using it?
(users, actors, roles, personas, etc)
- How does it fit into the existing IT environment?
(systems, services, etc)



techtribes.je - Context

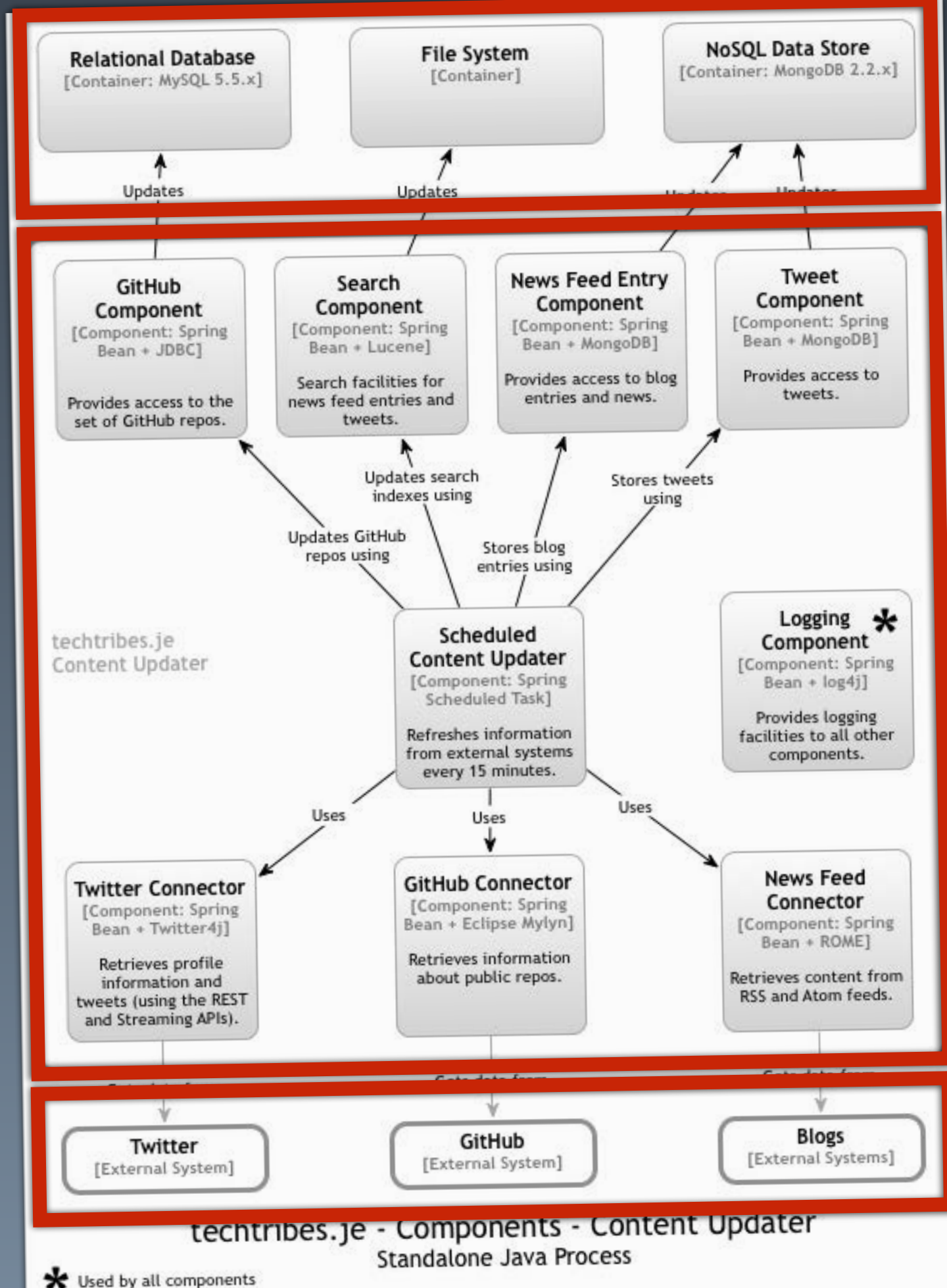
Containers

- What are the high-level technology decisions? (including responsibilities)
- How do containers communicate with one another?
- As a developer, where do I need to write code?



Components

- What components/services is the container made up of?
- Are the technology choices and responsibilities clear?



A notationless notation

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

My Web Application

[Container: Apache Tomcat 7.x]

Here is a list of the key
responsibilities for my
web application.

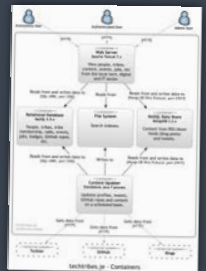


System Context

The system plus users
and system dependencies

Shneiderman's mantra

Overview
first



Containers

The overall shape of the architecture
and technology choices



Components

Logical components and their
interactions within a container

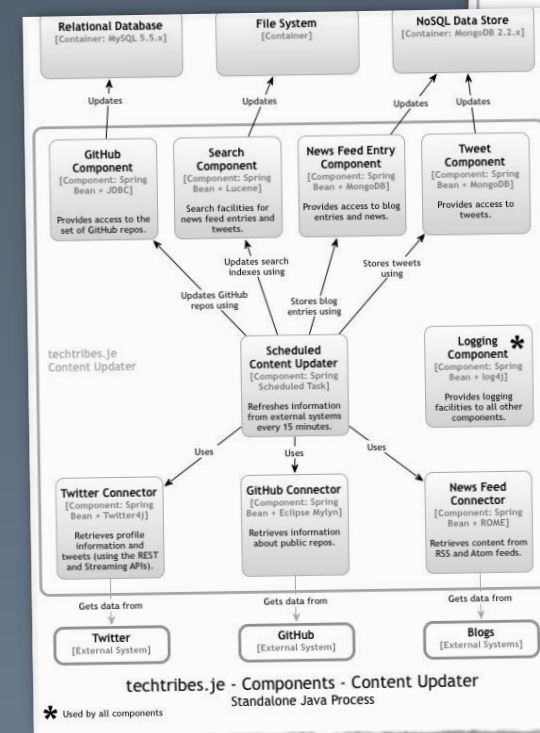
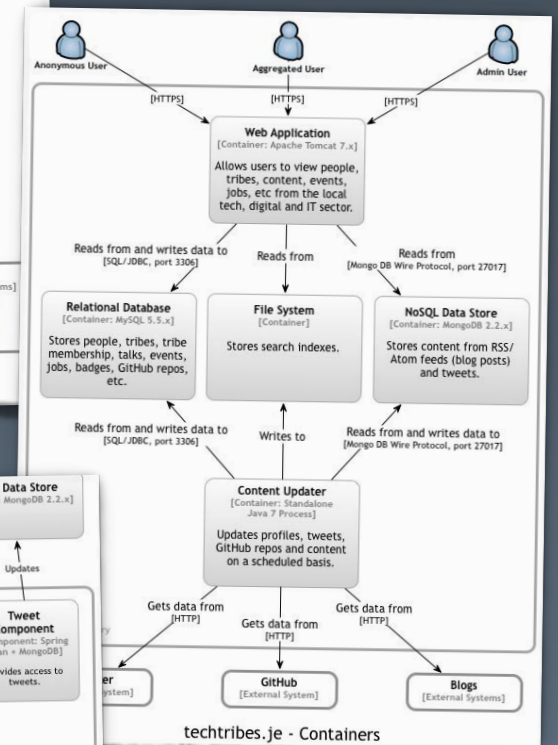
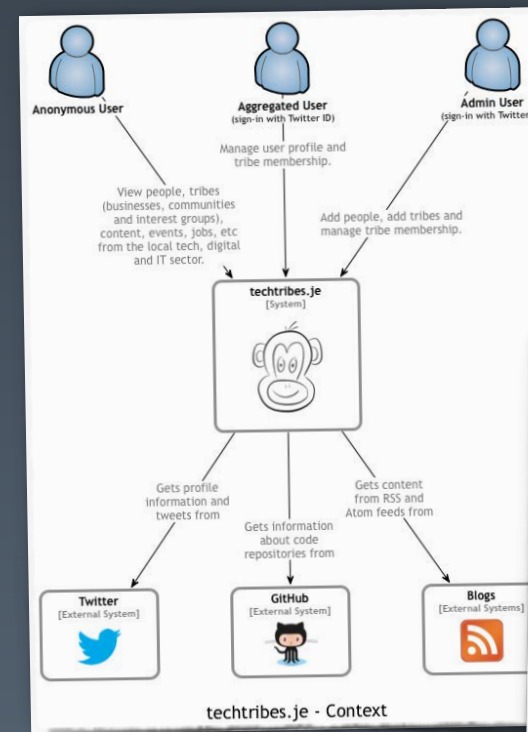
Zoom and
filter



Classes

Component or pattern
implementation details

Details
on demand

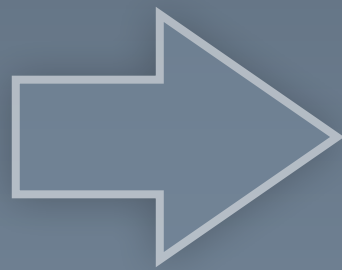


Diagrams are maps
that help a team navigate a complex codebase

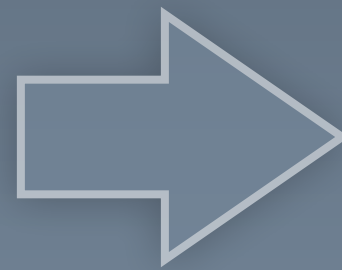
Think about the target audience



Non-technical



Semi-technical



Very technical

C4++

Enterprise context

User interface mockups and wireframes

Domain model

Sequence and collaboration diagrams

Business process and workflow models

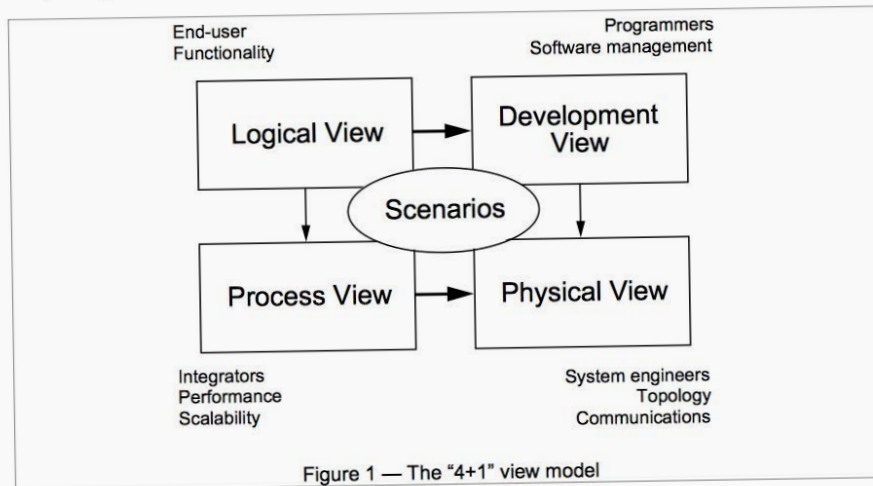
Infrastructure model

Deployment model

...

4+1 architectural view 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.

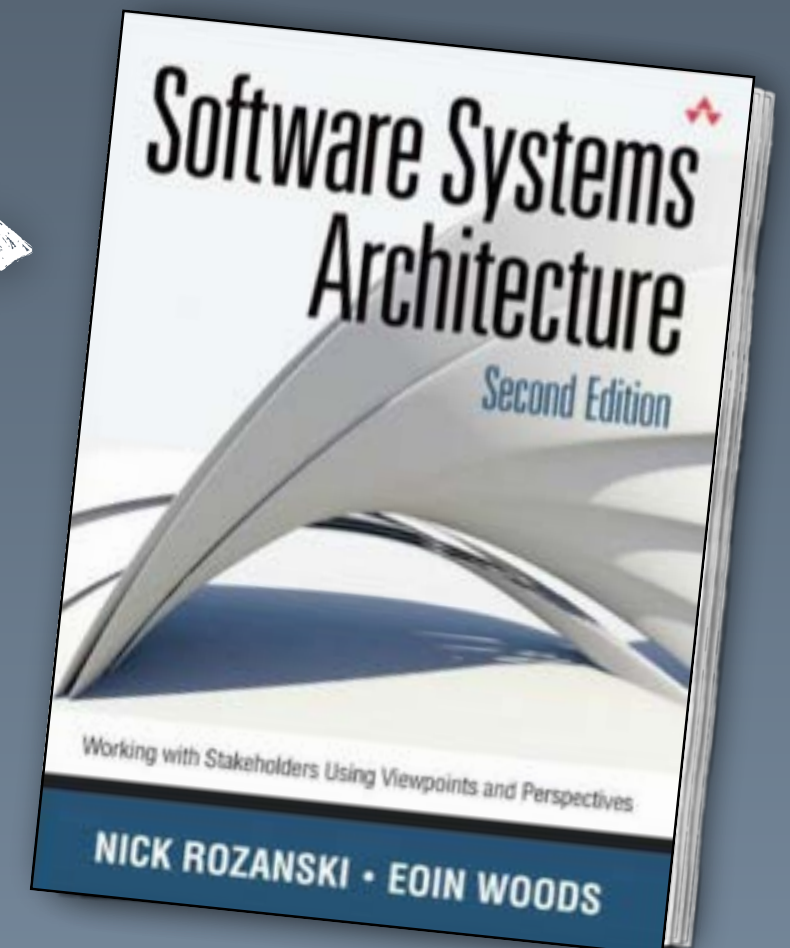


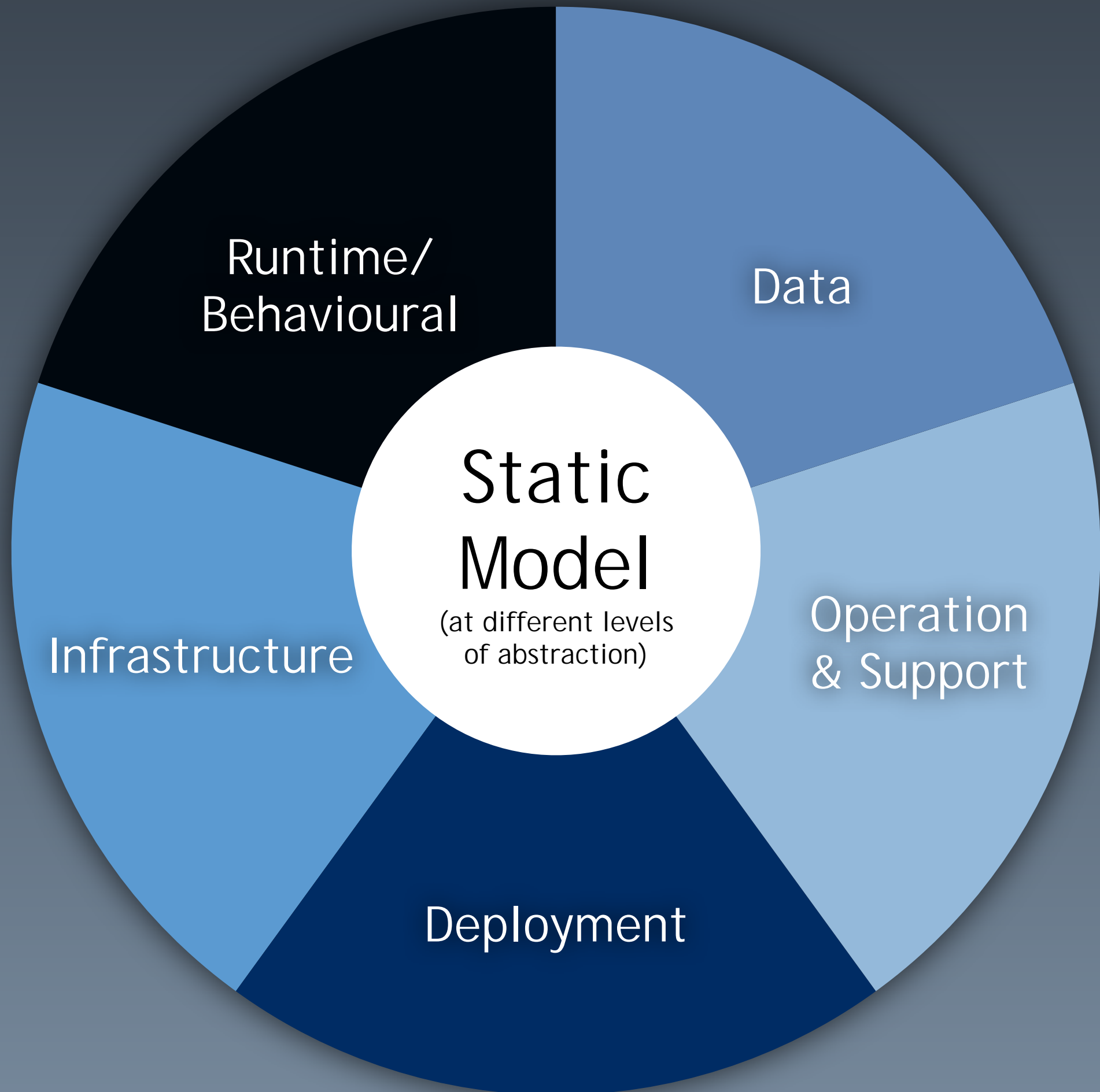
We apply Perry & Wolf's equation independently on each view, i.e., for each view we define the set of elements to use (components, containers, and connectors), we capture the forms and patterns that work, and we capture the rationale and constraints, connecting the architecture to some of the requirements.

Philippe Kruchten

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

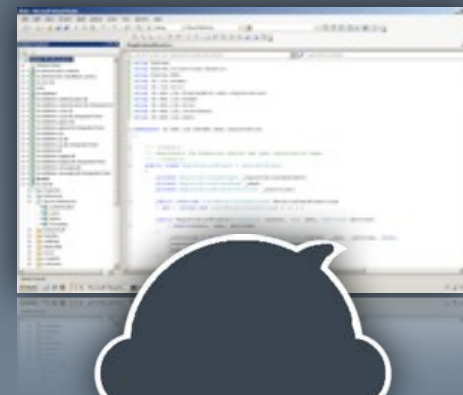
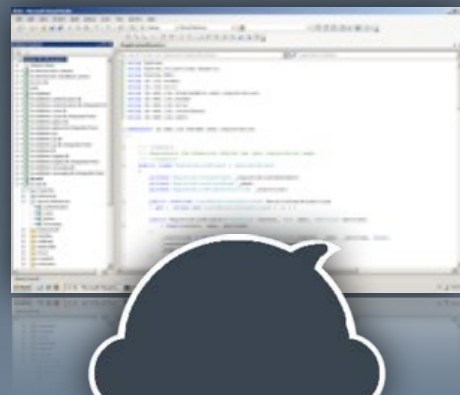
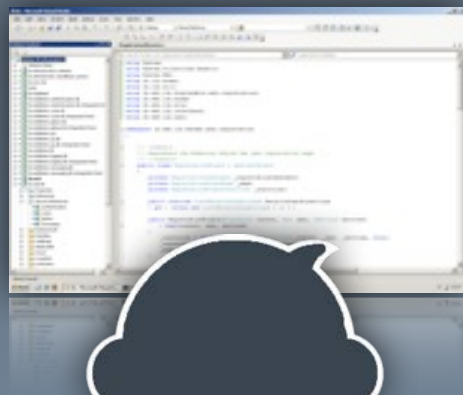
Nick Rozanski and Eoin Woods





C4 is about the
static structure
of software, which is ultimately about
code

Software developers are
the most important
stakeholders
of software architecture

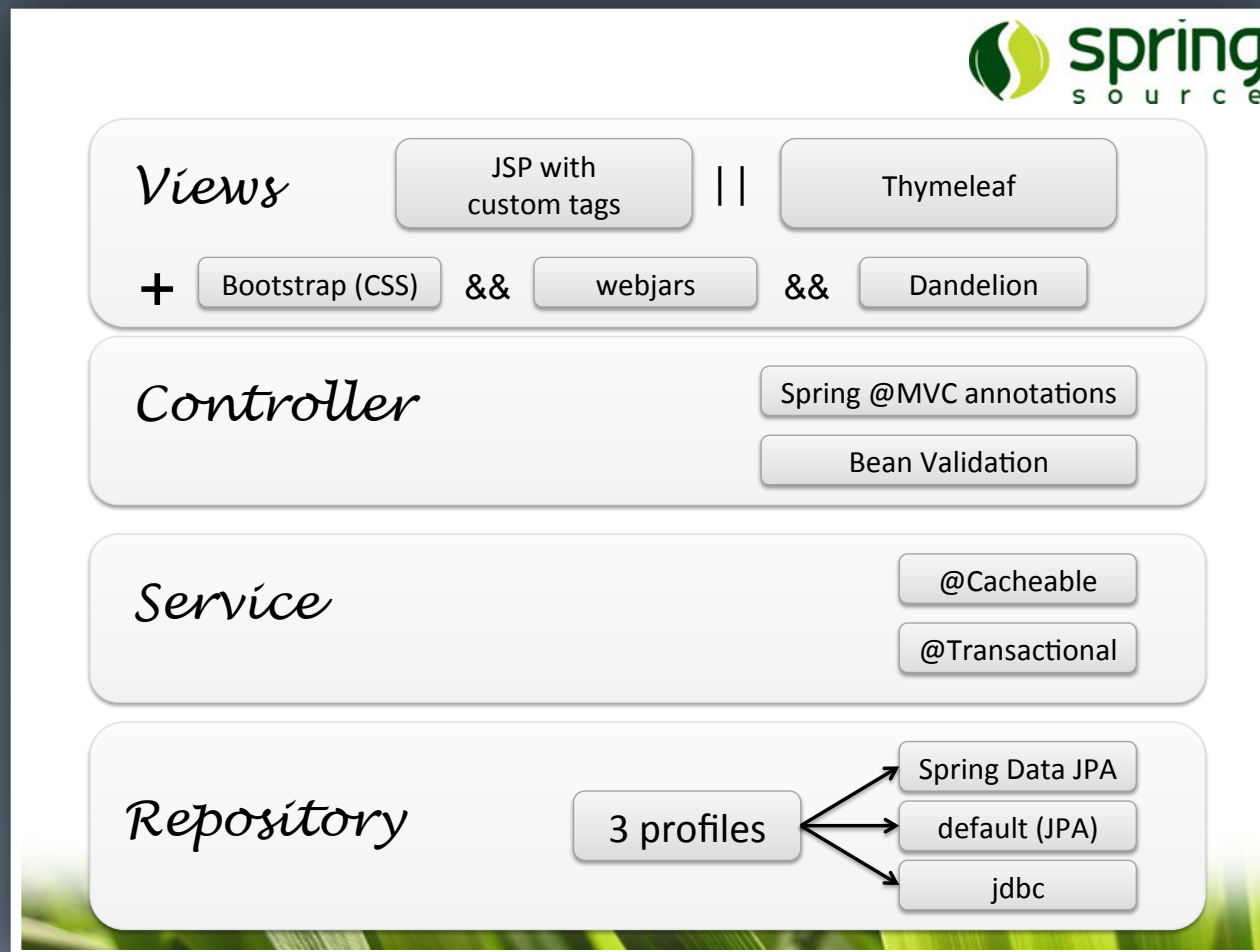


Components?

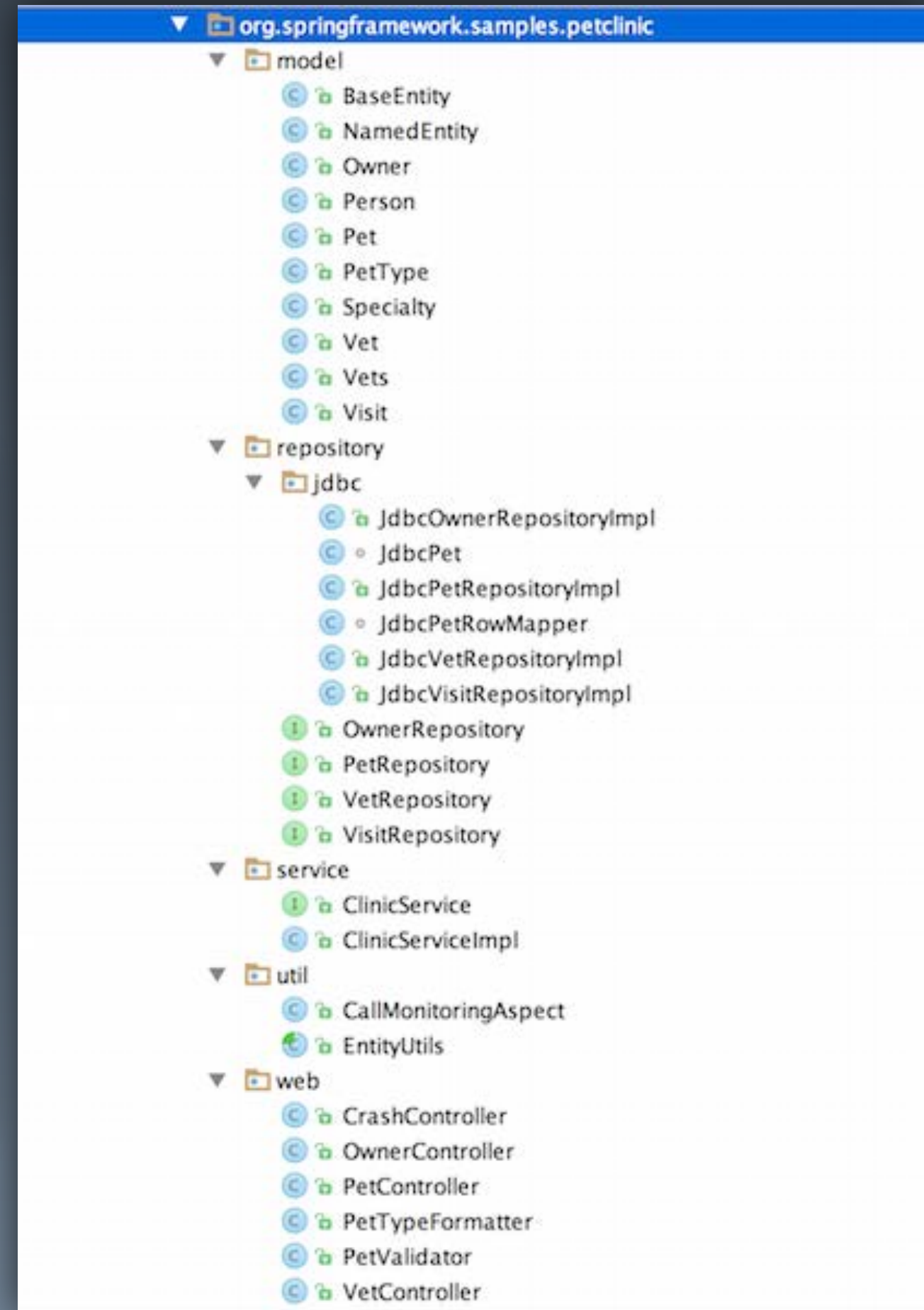
What is a
“component”?

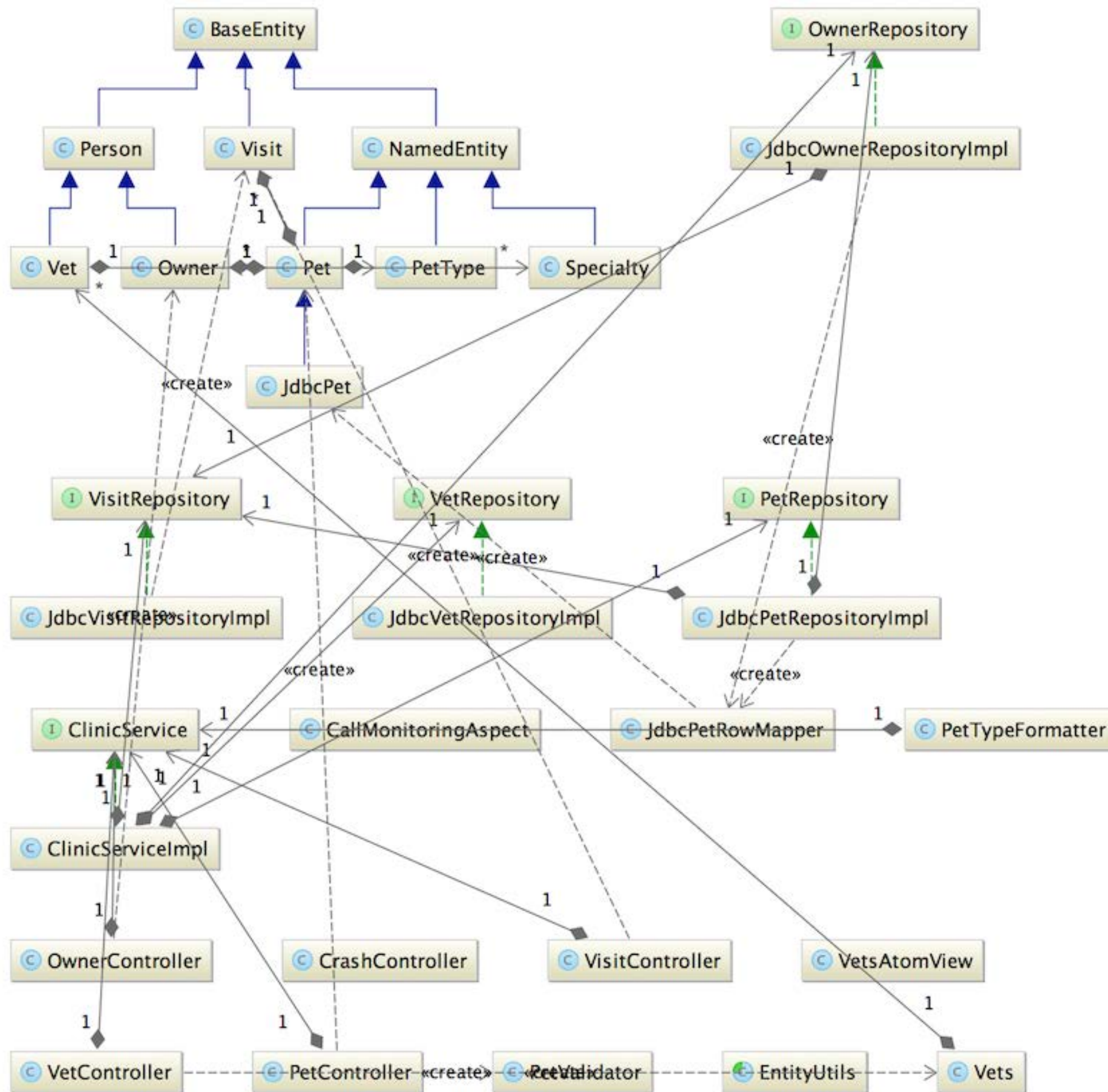
Spring PetClinic

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



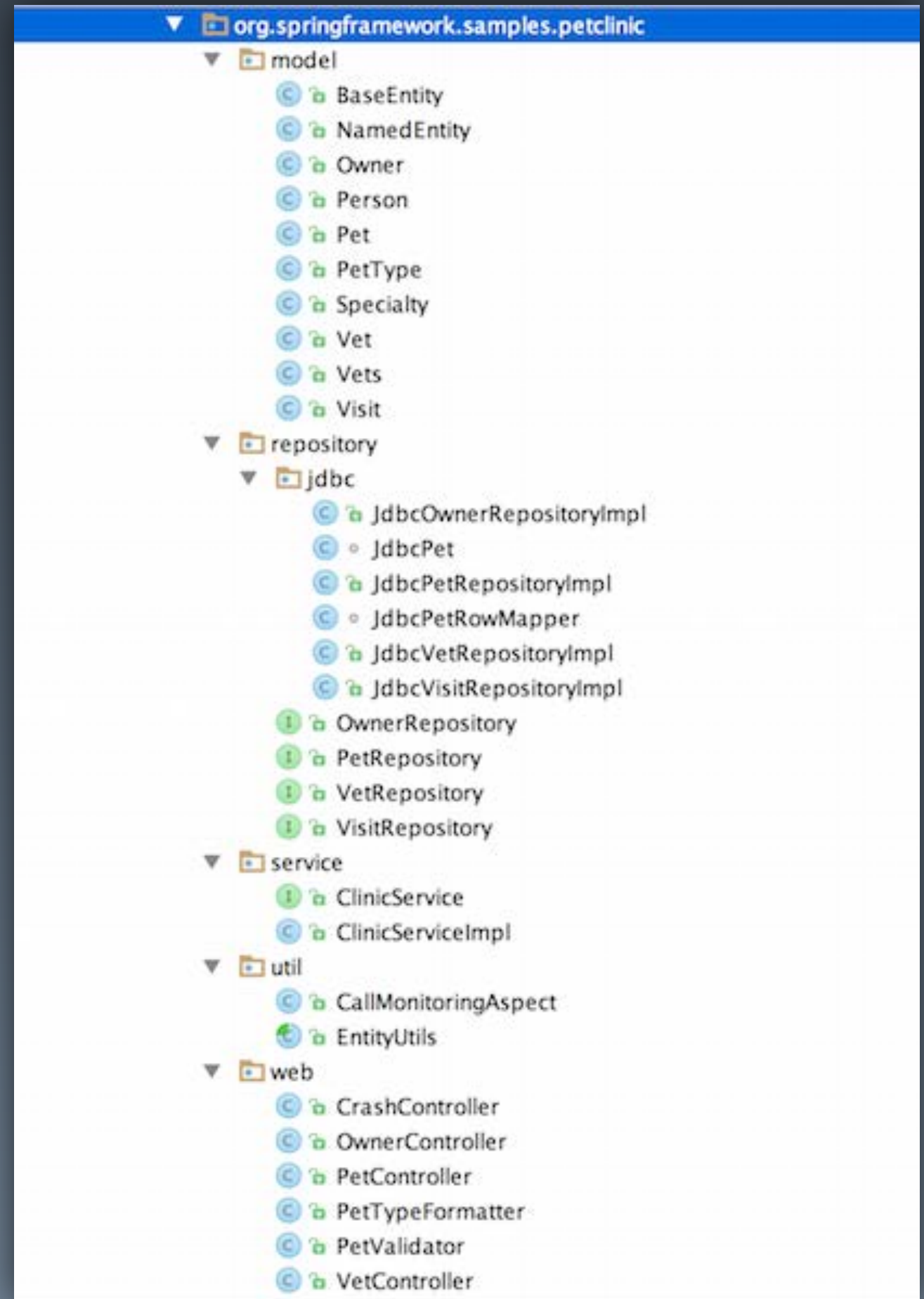
<https://speakerdeck.com/michaelisvy/spring-petclinic-sample-application>

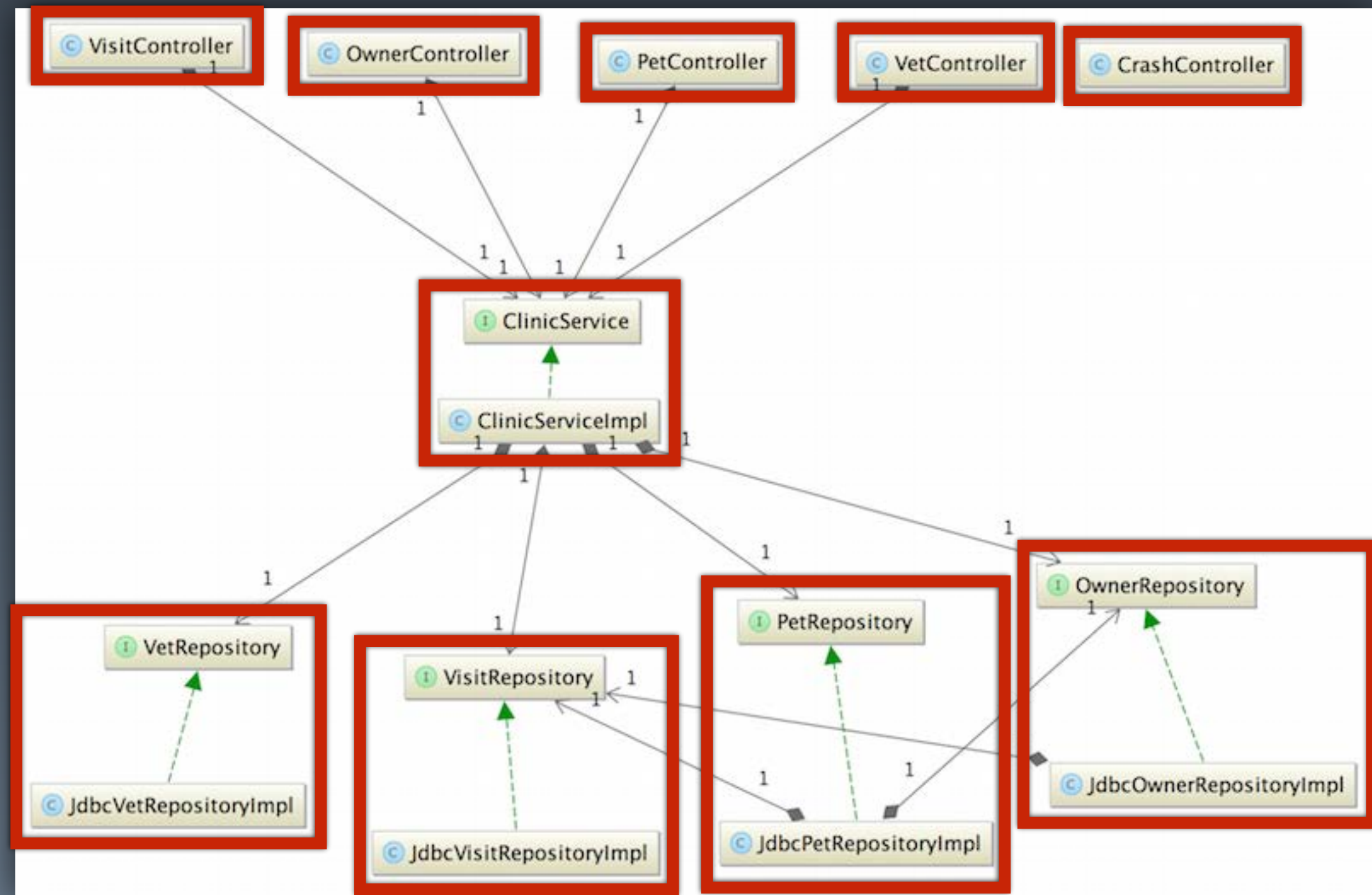




An auto-generated UML class diagram

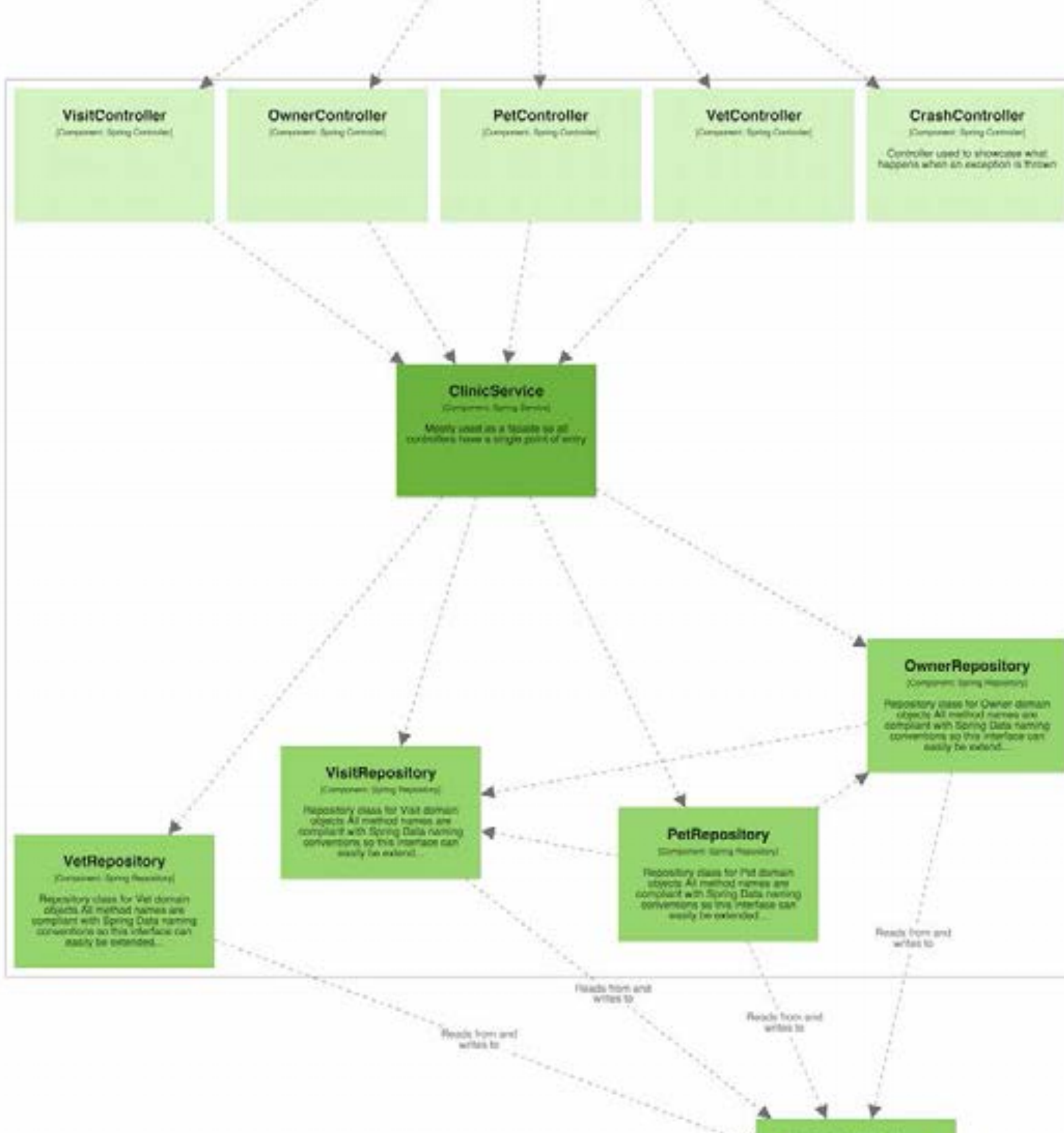
What are the
architecturally
significant
elements?





A UML class diagram showing architecturally significant elements

A component diagram,
based upon
the code



The intersection of software architecture and code

Merge

the code

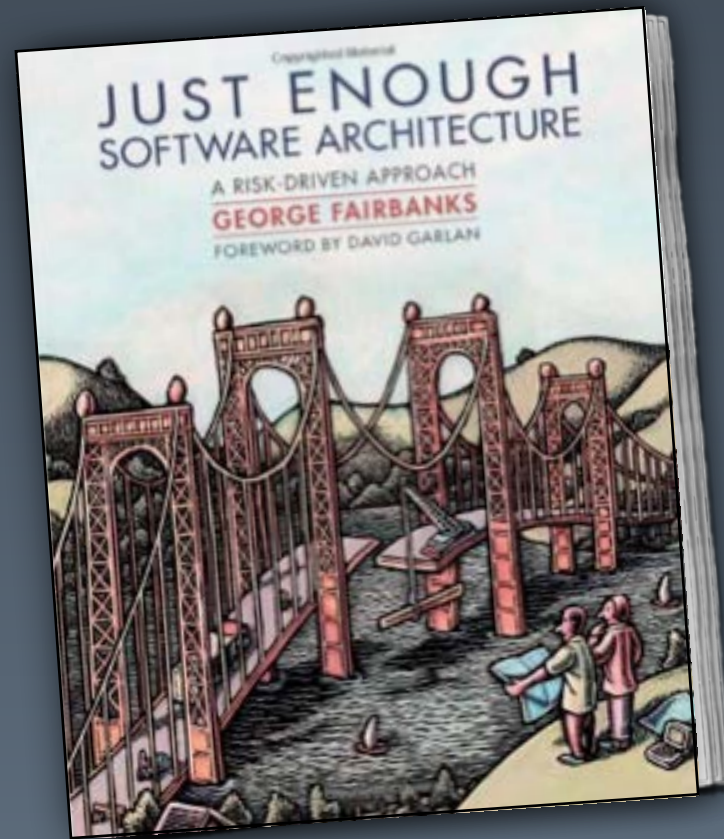
and the model?

Abstractions

on diagrams

should reflect the

code



“architecturally-evident coding style”

(subclassing, naming conventions, module dependencies, package structure, ...)

Project: techtribesje [techtribes] (~/.sandbox/techtribesje)

- docs
- lib
- production
- runtime
- test
- structurizr
- techtribes-core
 - sql
 - src
 - je
 - techtribes
 - component
 - activity
 - badge
 - book
 - contentsource
 - creation
 - event
 - github
 - log
 - newsfeedentry
 - search
 - talk
 - tweet**
 - component.xml
 - MongoDbTweetDao
 - TweetComponent**
 - TweetComponentImpl
 - TweetException
 - domain
 - util
 - config.xml
- test
- techtribes-core.iml

```

package je.techtribes.component.tweet;

import ...

/**
 * Provides access
 */
@Component
public interface

/**
 * Gets the m
 */
List<Tweet> g

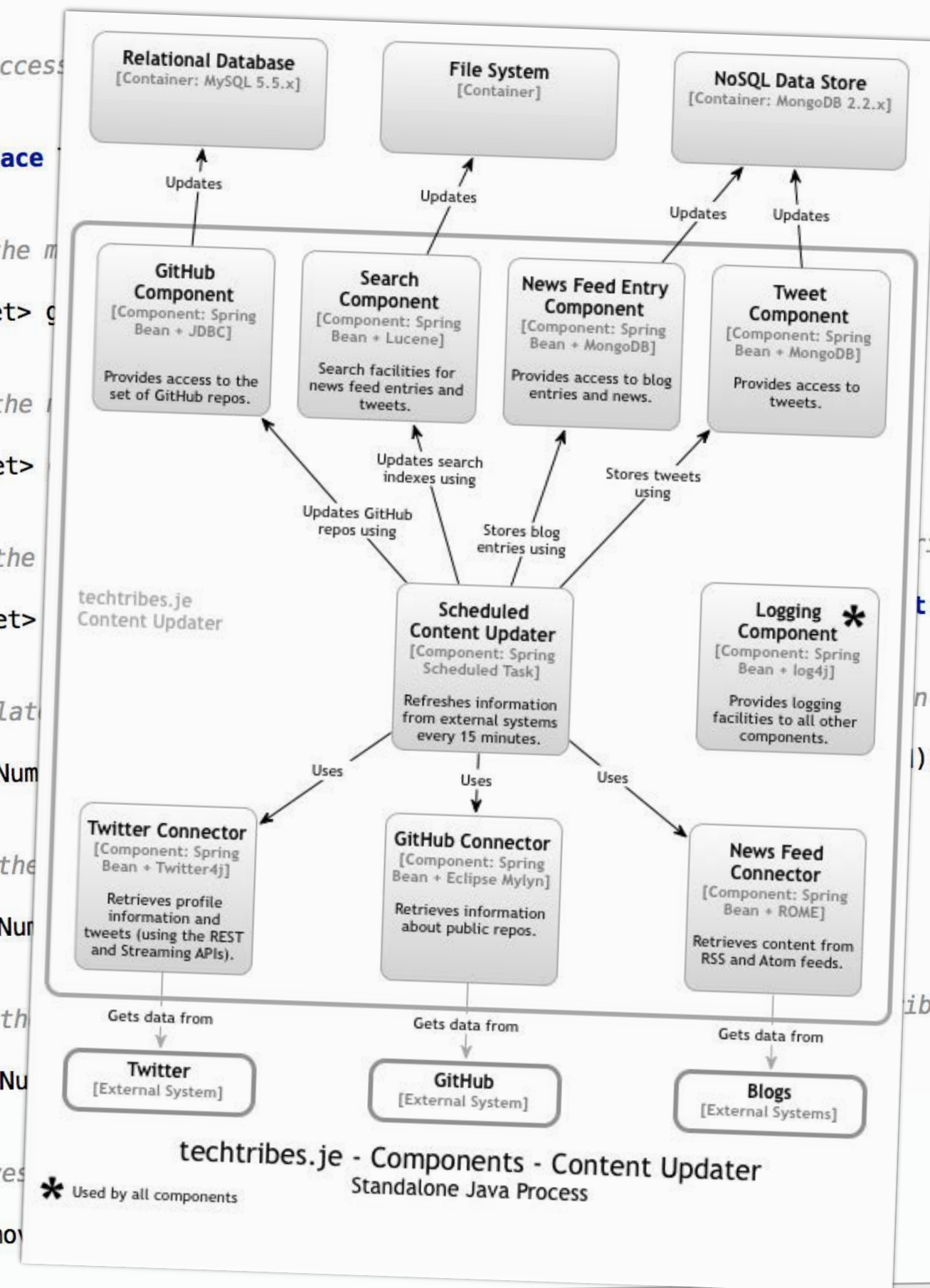
/**
 * Gets the
 */
List<Tweet>

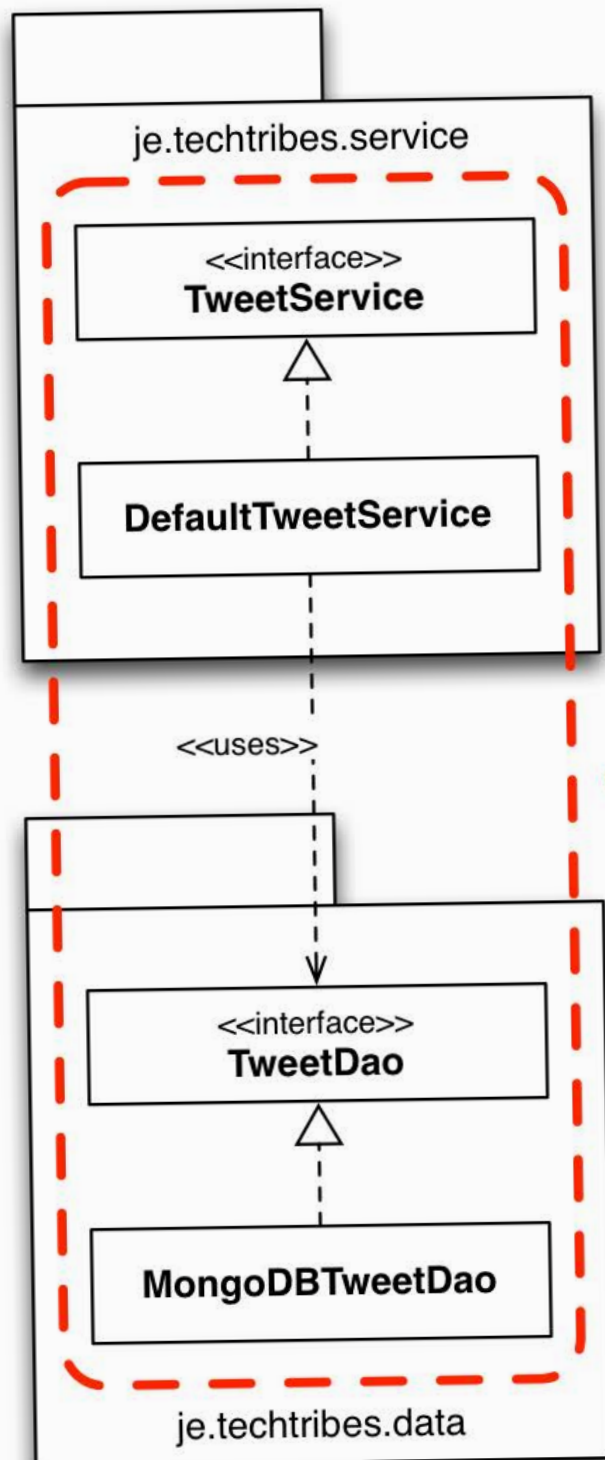
/**
 * Calculat
 */
long getNum

/**
 * Gets the
 */
long getNum

/**
 * Gets th
 */
long getNu

/**
 * Removes
 */
void remov
  
```

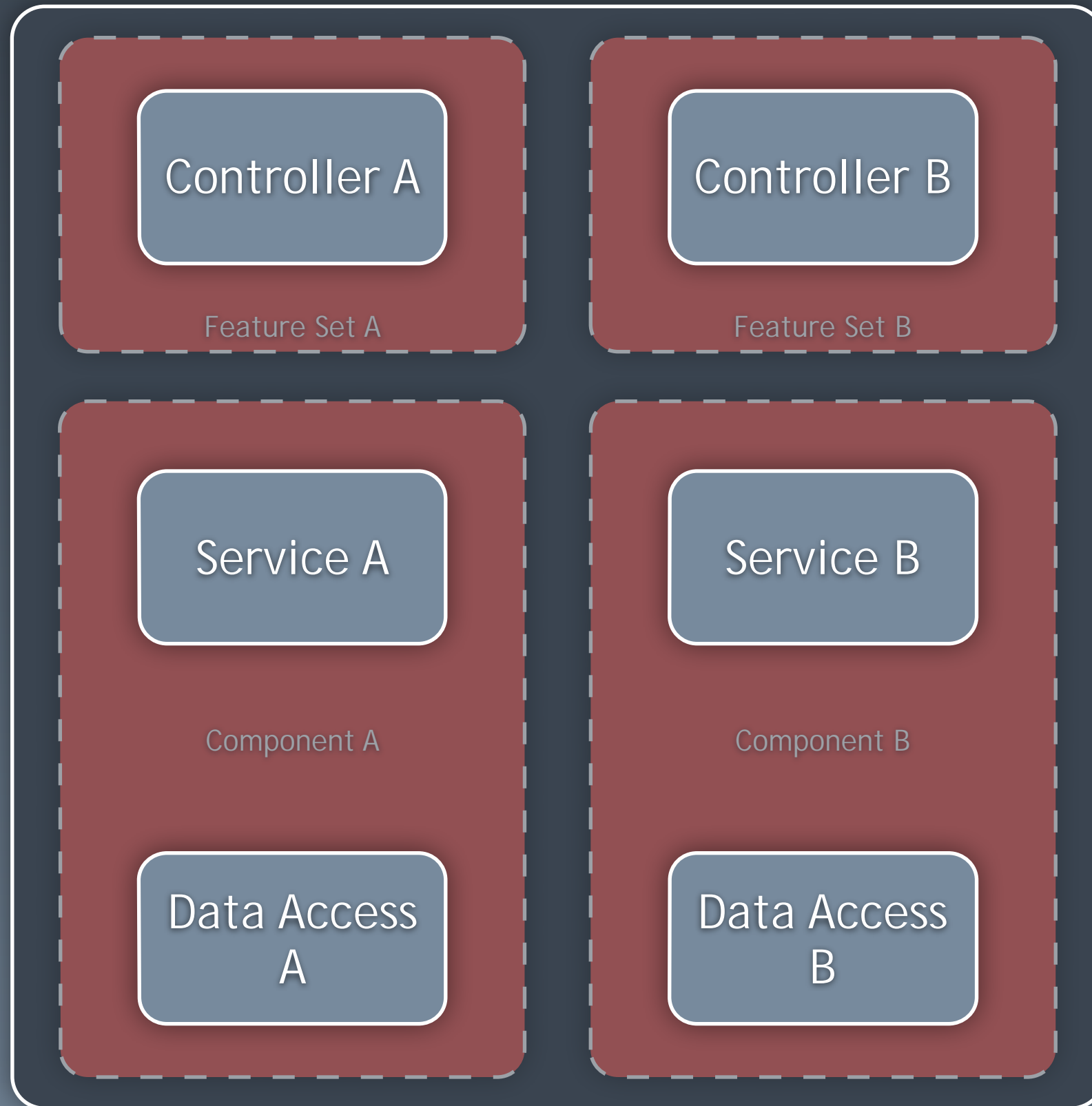




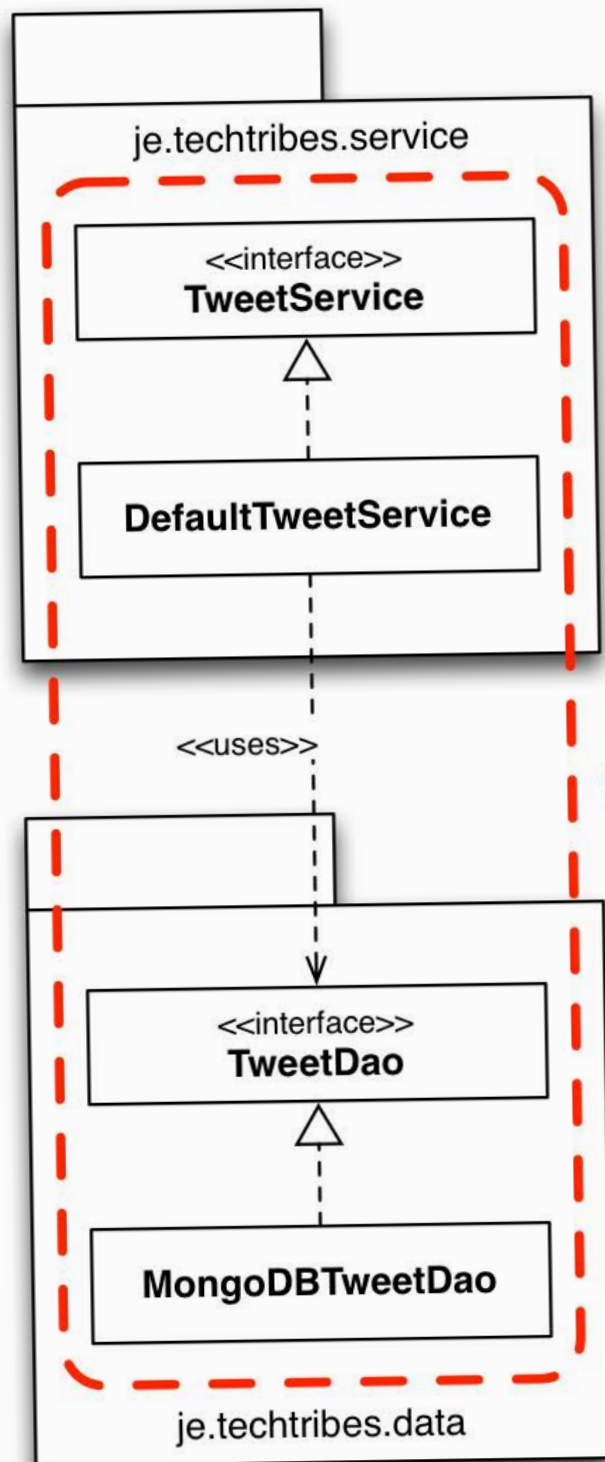
A component is often a combination of a number of classes in different layers.

Package by layer

What's a "component"?

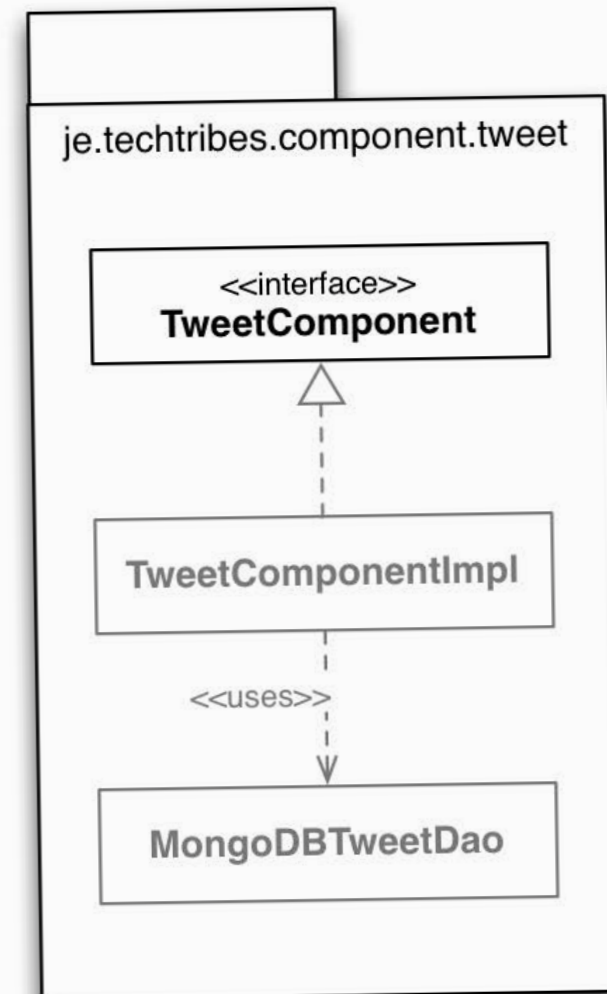


Package by component



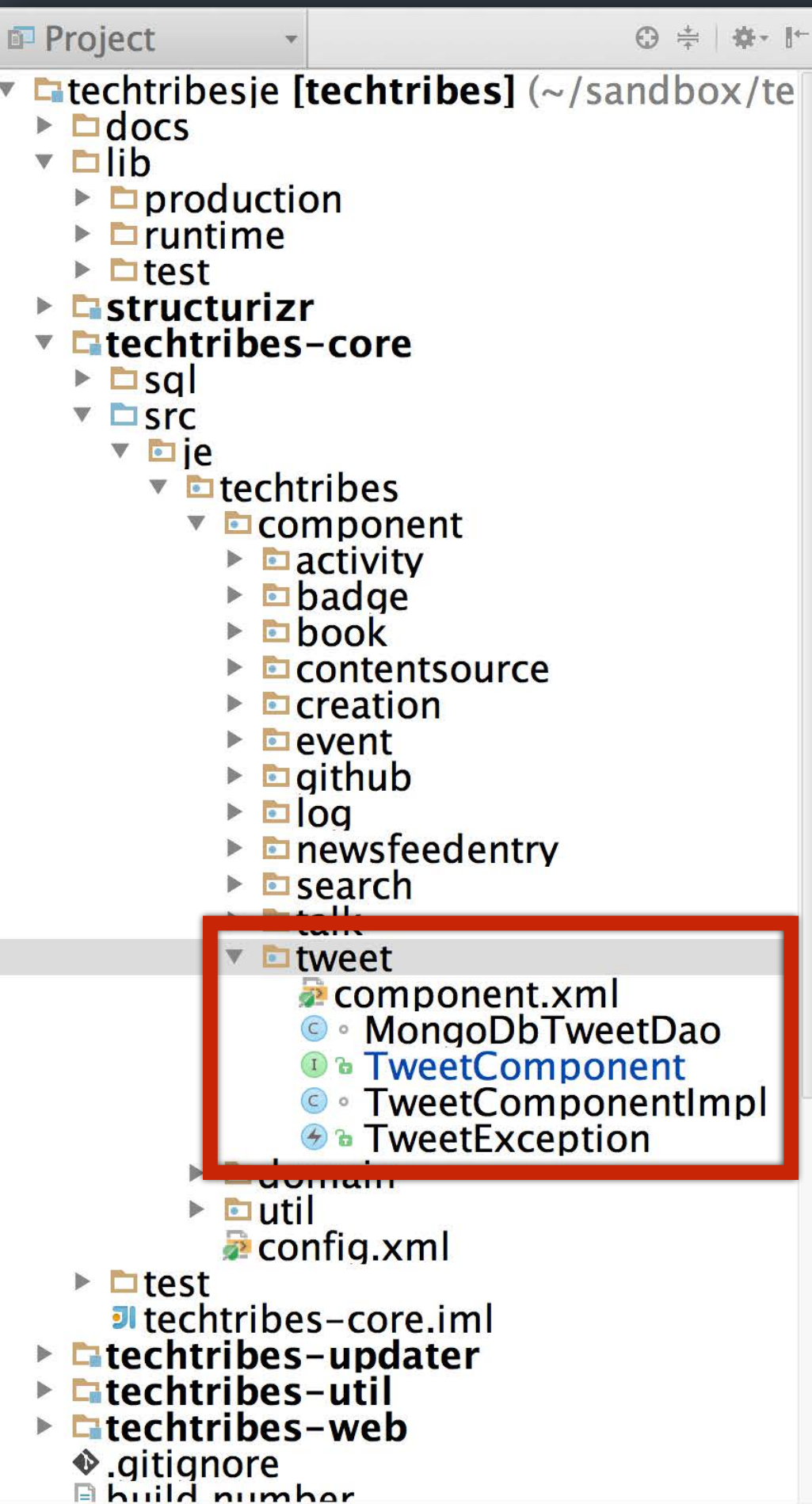
Package by layer

A component is often a combination of a number of classes in different layers.



Package by component

What's a "component"?



```
package je.techtribes.component.tweet;
```

```
import ...
```

```
/**  
 * Provides access to tweets.  
 */  
@Component  
public interface TweetComponent {
```

```
/**  
 * Gets the most recent tweets by page number.  
 */
```

```
List<Tweet> getRecentTweets(int page, int count);
```

```
/**  
 * Gets the most recent tweets for a given content source.  
 */
```

```
List<Tweet> getRecentTweets(ContentSource source);
```

```
/**  
 * Gets the most recent tweets for a given user.  
 */
```

```
List<Tweet> getRecentTweets(Collection<ContentSource> sources);
```

```
/**  
 * Calculates how many tweets a given person has.  
 */
```

Architecturally-evident coding styles include:

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

Naming conventions (`*Service`)

Namespacing/packaging

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

Maven modules, OSGi modules,
microservices, etc

Modularity

as a principle

Software architecture vs testing (a quick aside)

“In the early days of computing when computers were slow, unit tests gave the developer more immediate feedback about whether a change broke the code instead of waiting for system tests to run. Today, with cheaper and more powerful computers, that argument is less persuasive.”



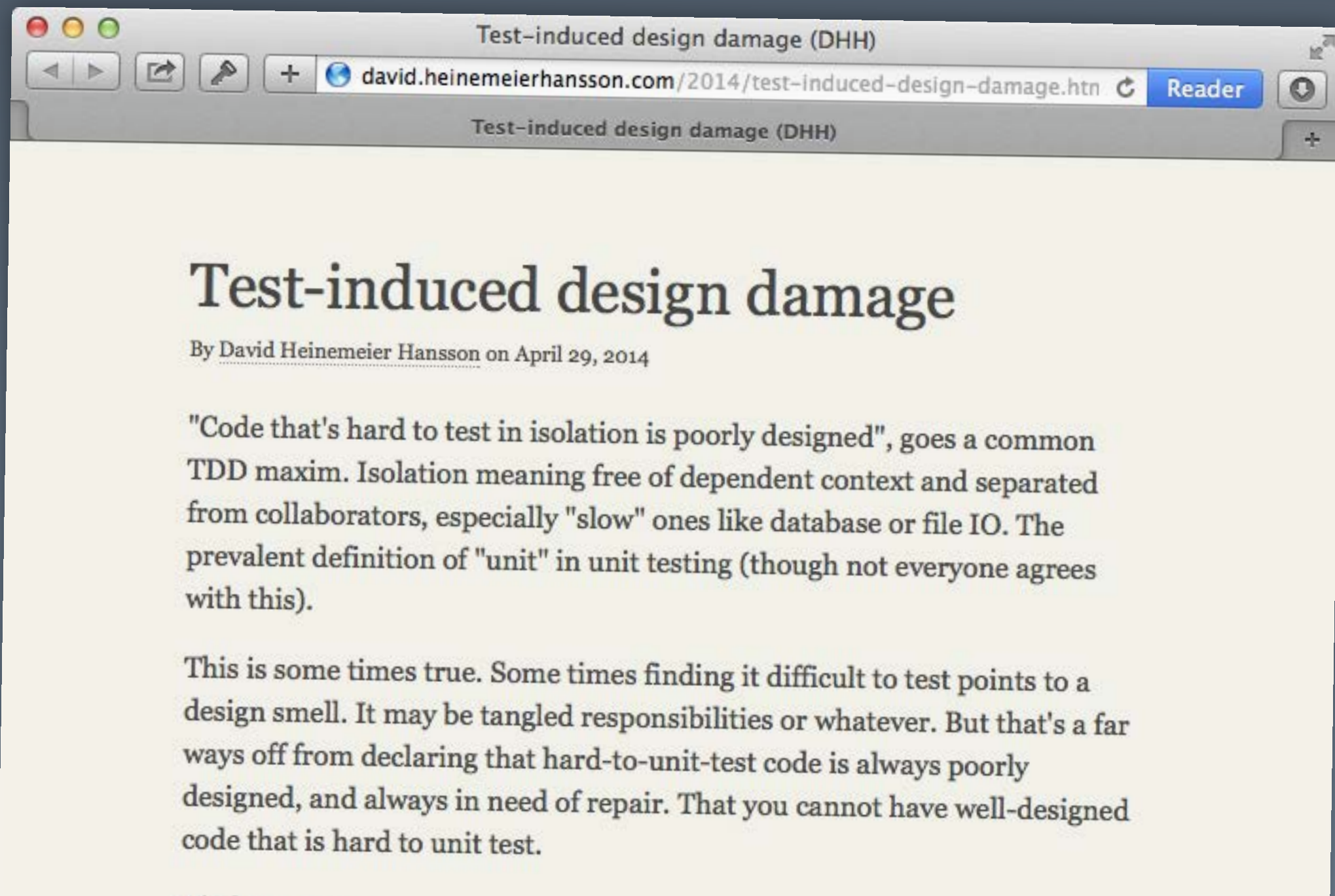
Why Most Unit Testing is Waste

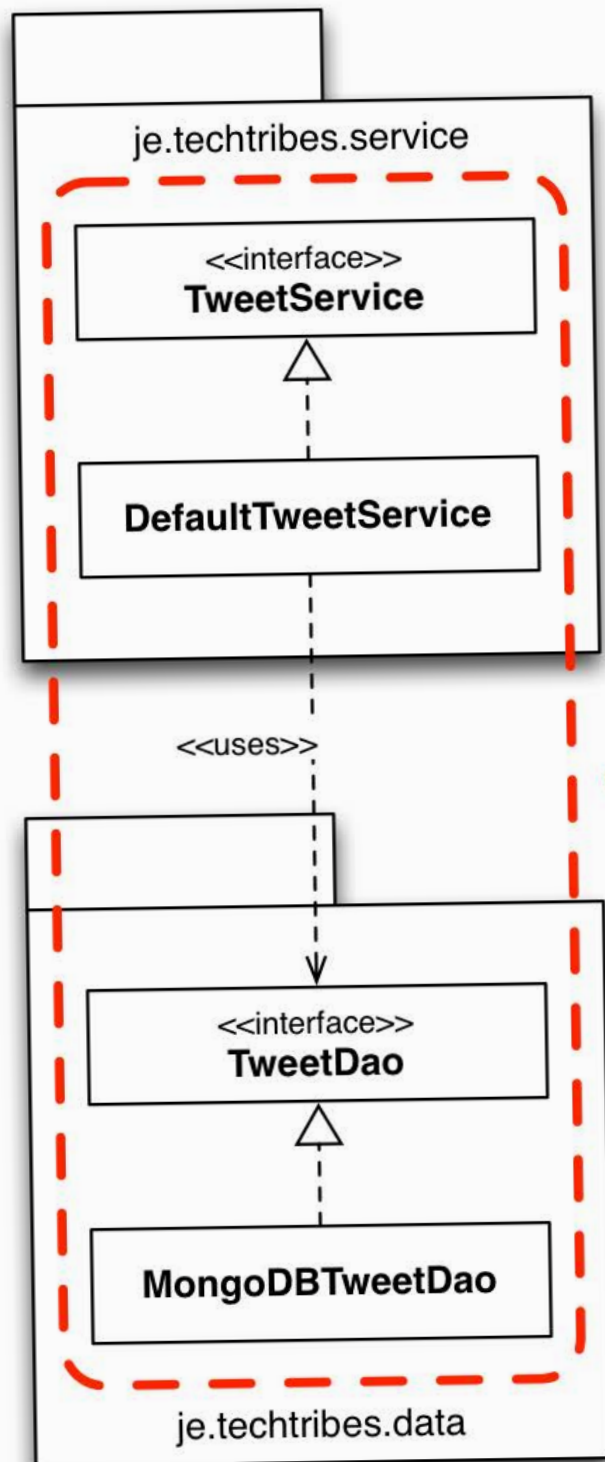
By James O Coplien

1.1 Into Modern Times

Unit testing was a staple of the FORTRAN days, when a function was a function and was sometimes worthy of functional testing. Computers computed, and functions and procedures represented units of computation. In those days the dominant

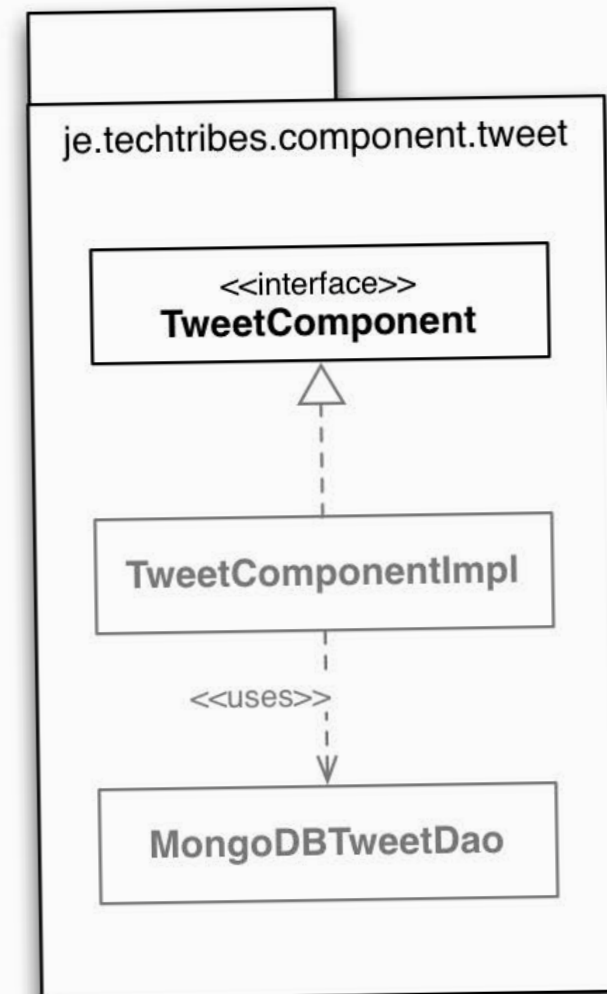
“do not let your tests
drive your design”





Package by layer

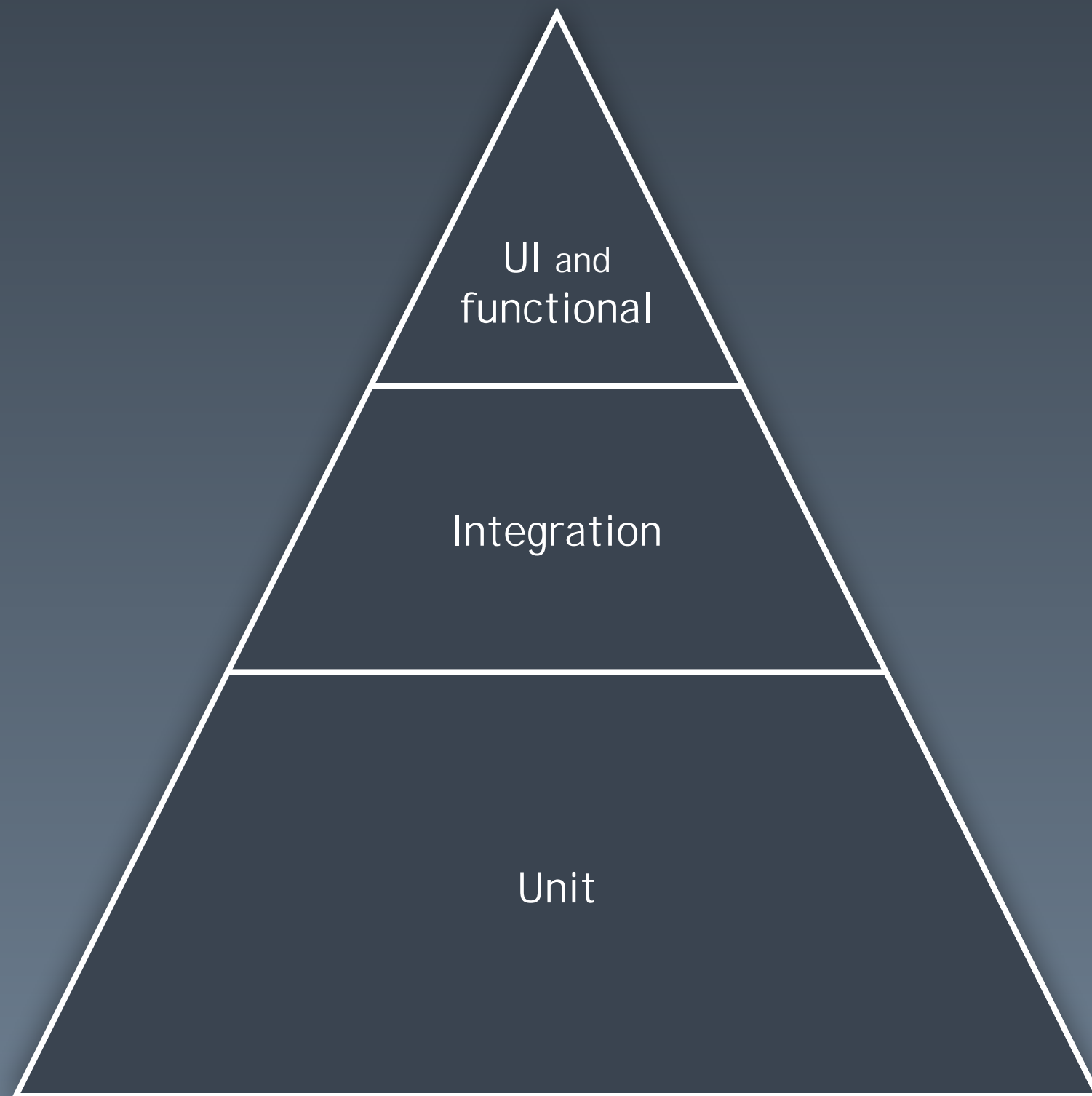
A component is often a combination of a number of classes in different layers.



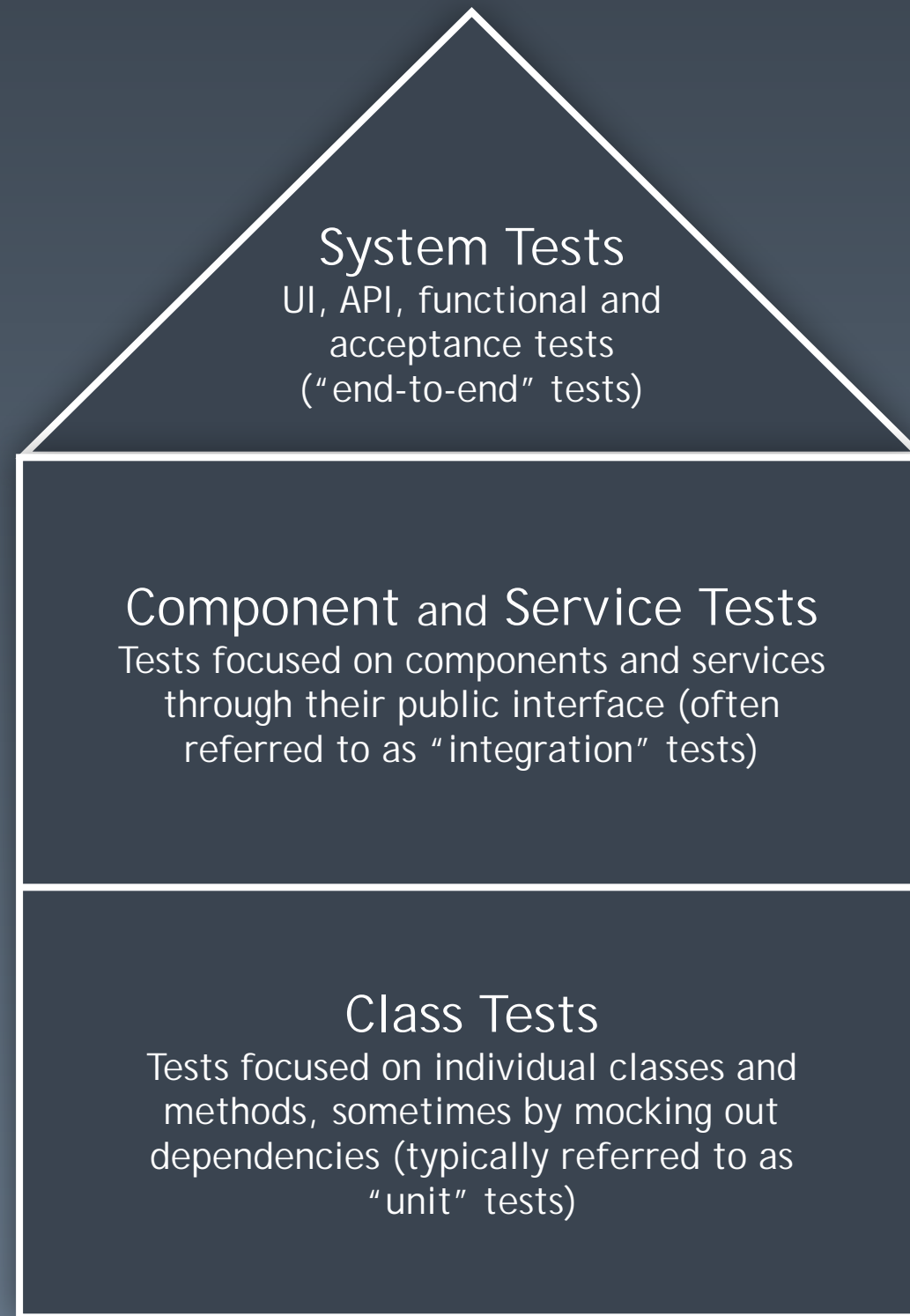
Package by component

What's a "component"?

Instead of blindly unit testing
everything, what about
testing your significant
structural elements
as black boxes?



Do we need to rethink the testing pyramid?



Architecturally-aligned testing
(and a reshaped testing pyramid)

Software architecture as code

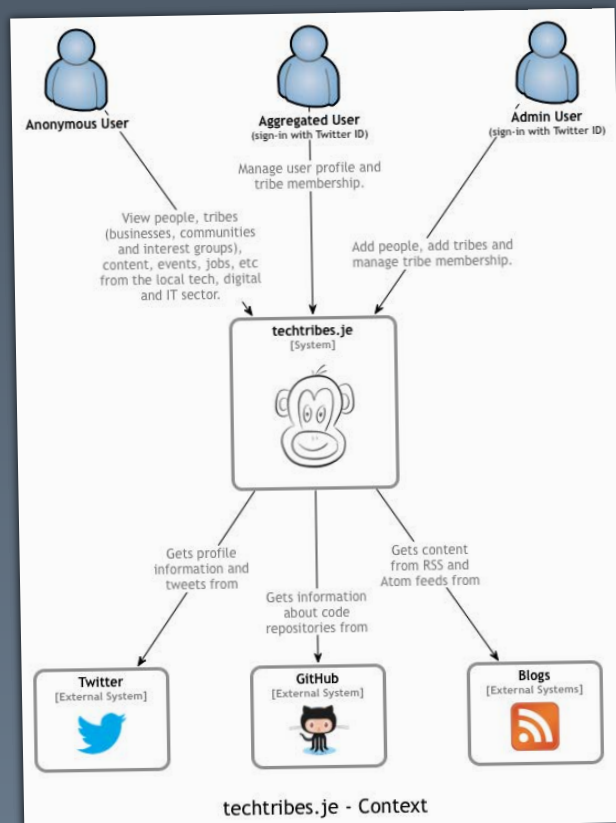
The code is the
embodiment
of the architecture

Is the architecture
in the code?

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

Context



People

Security groups/roles in configuration files, etc.

Software Systems

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

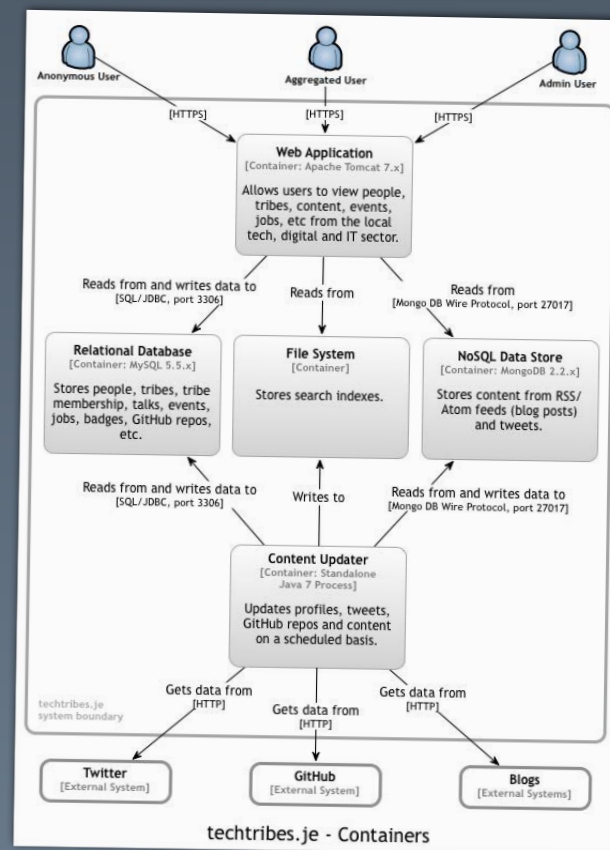
Containers

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

Components

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

Containers



People

Security groups/roles in configuration files, etc.

Software Systems

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

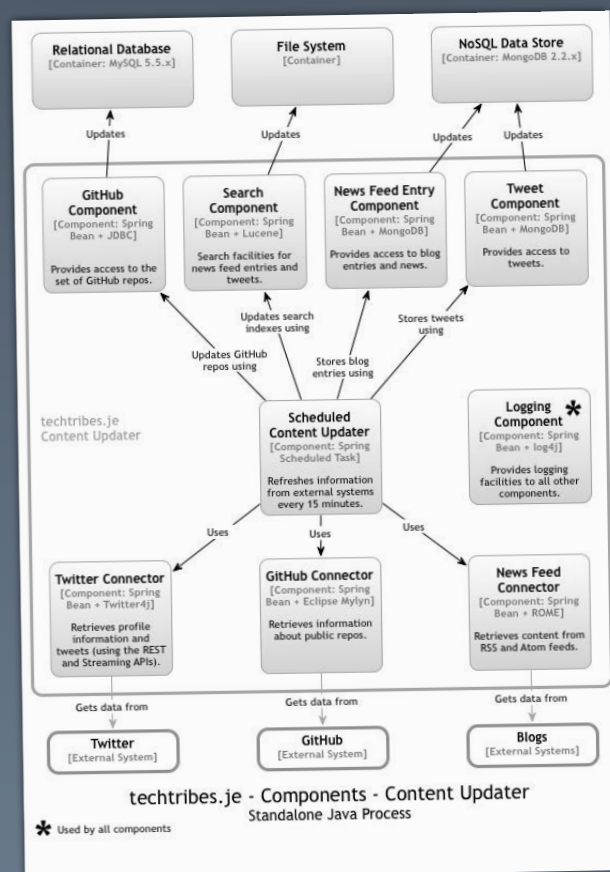
Containers

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

Components

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

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.

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

Create an architecture
description language
using code

```

/**
 * 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-projects/spring-petclinic/)");
        Model model = workspace.getModel();

        // create the basic model (the stuff we can't get from the code)
        SoftwareSystem springPetClinic = model.addSoftwareSystem("Spring PetClinic", "");
        Person user = model.addPerson("User", "");
        user.uses(springPetClinic, "Uses");

        Container webApplication = springPetClinic.addContainer(
            "Web Application", "", "Apache Tomcat 7.x");
        Container relationalDatabase = springPetClinic.addContainer(
            "Relational Database", "", "HSQLDB");
        user.uses(webApplication, "Uses");
        webApplication.uses(relationalDatabase, "Reads from and writes to");

        // and now automatically find all Spring @Controller, @Component, @Service and @Repository
        ComponentFinder componentFinder = new ComponentFinder(
            webApplication, "org.springframework.samples.petclinic",

```



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

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

```
for (Component component : webApplication.getComponents()) {  
    if (component.getSourcePath() != null) {  
        component.setSourcePath(component.getSourcePath().replace(  
            "/Users/simon/Documents/sandbox/spring/spring-petclinic/",  
            "https://github.com/spring-projects/spring-petclinic/tree/master/"));  
    }  
}
```

```
// finally create some views
```

```
ViewSet viewSet = workspace.getViews();
```

```
SystemContextView contextView = viewSet.createContextView(springPetClinic);
```



```
// 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);
```

```
// tag and style some elements
```

```
springPetClinic.addTags("Spring PetClinic");  
webApplication.getComponents().stream().filter(c -> c.getTechnology().equals("Spring C  
webApplication.getComponents().stream().filter(c -> c.getTechnology().equals("Spring S  
webApplication.getComponents().stream().filter(c -> c.getTechnology().equals("Spring R
```

```
viewSet.getStyles().add(new ElementStyle("Spring PetClinic", null, null, "#6CB33E", "w  
viewSet.getStyles().add(new ElementStyle(Tags.PERSON, null, null, "#519823", "white",  
viewSet.getStyles().add(new ElementStyle(Tags.CONTAINER, null, null, "#91D366", "white  
viewSet.getStyles().add(new ElementStyle("Spring Controller", null, null, "#D4F3C0", "  
viewSet.getStyles().add(new ElementStyle("Spring Service", null, null, "#6CB33E", "bla  
viewSet.getStyles().add(new ElementStyle("Spring Repository", null, null, "#95D46C", "
```

Structurizr for Java

(open source on GitHub)



GitHub

This repository Search

ExploreFeaturesEnterpriseBlog

Sign upSign in

structurizr / java

Watch24Star59Fork30

Java tools

131 commits1 branch0 releases3 contributors

branch: master java / +

Added a software architecture model for the Java EE Hands on Lab "Mov...
simonbrowndotje authored 14 days ago latest commit 90dea447f9

gradle

Gradle Support: cleanup workspace, commit gradle wrapper, remove ant/ivy

2 months ago

structurizr-annotations/src/co...

Some refactoring and an initial version of a Javadoc component finder...

25 days ago

structurizr-client

This saves a little bandwidth. :-)

14 days ago

structurizr-core

Fixed a bug where layout information wasn't getting copied when inter...

14 days ago

structurizr-examples

Added a software architecture model for the Java EE Hands on Lab "Mov...

14 days ago

structurizr-spring

Added a sourcePath property to components ... which is automatically ...

23 days ago

.gitignore

Added some comments.

2 months ago

LICENSE

Initial commit

10 months ago

README.md

Added build instructions.

2 months ago

build.gradle

Changed styles -> configuration.styles

27 days ago

copyJars.sh

Added fixed some bugs and added an initial implementation to copy lay...

2 months ago

gradle.properties

Renamed components and messing with Maven repo publication.

2 months ago

gradlew

Trimmed down the Gradle config and stopped tracking the IDEA project ...

2 months ago

gradlew.bat

Gradle support

2 months ago

settings.gradle

Renamed components and messing with Maven repo publication.

2 months ago

HTTPS clone URL
https://github.com
You can clone with HTTPS or Subversion.
Clone in Desktop
Download ZIP

README.md

Structurizr for Java

Structurizr is an implementation of the C4 model as described in Simon Brown's [Software Architecture for Developers](#) book, which provides a way to easily and effectively communicate the software architecture of a software system. Structurizr allows you to create **software architecture models and diagrams as code**. This project contains the Java implementation and tooling.

Everything you see here is a work in progress. See www.structurizr.com for more information.

Building

To build Struturizr for Java from the sources (you'll need Java 8)...

```
git clone https://github.com/structurizr/java.git  
  
./gradlew build
```

© 2015 GitHub, Inc. Terms Privacy Security Contact

Status API Training Shop Blog About

Structurizr

No more messing with drawing tools. Create software architecture models and diagrams as code based upon the [C4 software architecture model](#).



Simple

No more endless hours spent messing with drawing tools figuring out what to include and then manually drawing boxes and lines.

Versionable

The software architecture model can be versioned as code.

Up-to-date

Integration with...

Scalable

From code

You create a software architecture model by extracting them automatically from your code.

```
Workspace workspace = new Workspace("My model");
Model model = workspace.getModel();

SoftwareSystem mySystem = model.addSoftwareSystem("mySystem");

Person user = model.addPerson("Anonymous User");
user.uses(mySystem, "Uses");

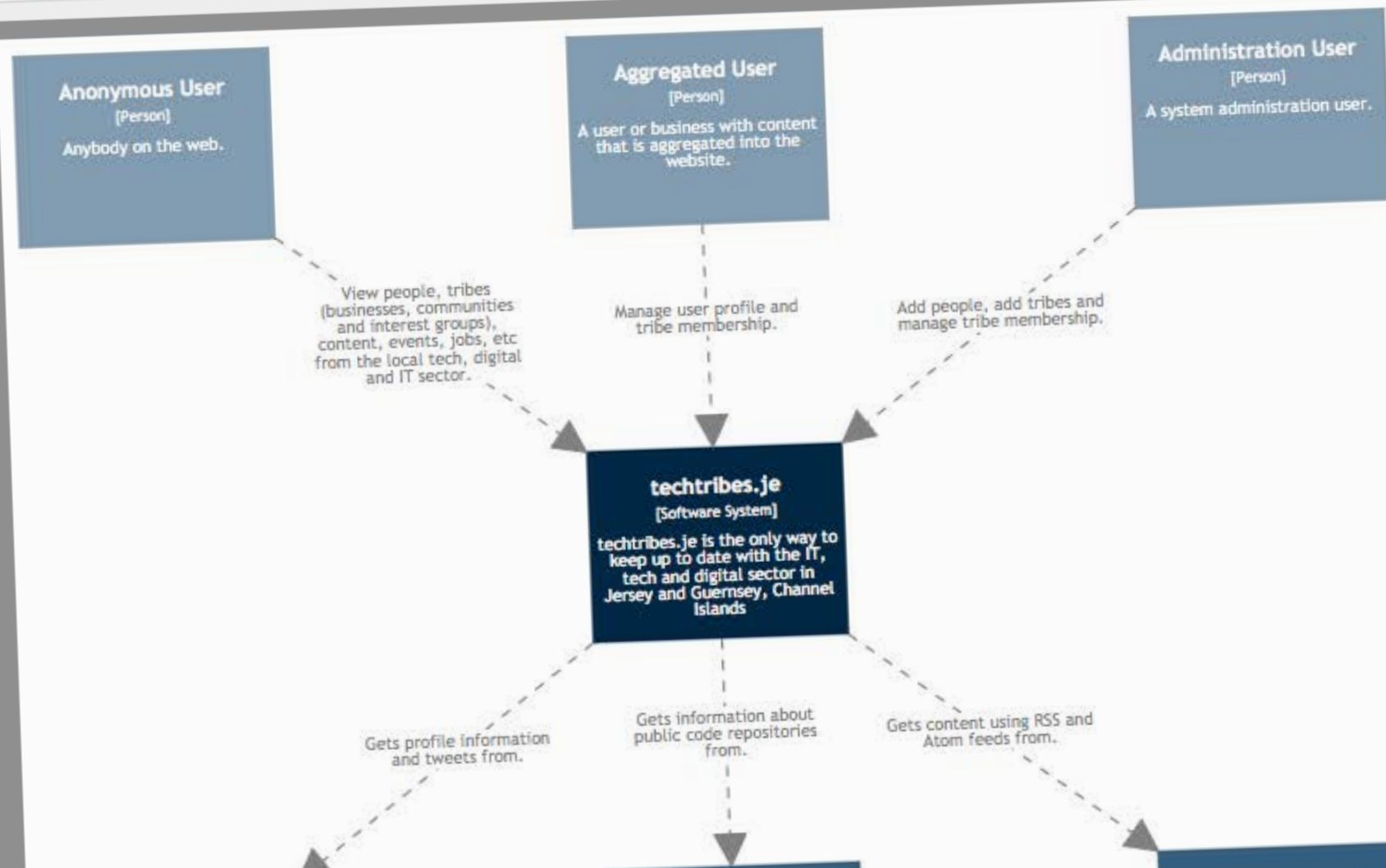
SoftwareSystem systemA = model.addSoftwareSystem("systemA");
mySystem.uses(systemA, "Gets some information");

ViewSet viewSet = workspace.getViews();
SystemContextView contextView = viewSet.createContextView();
contextView.addAllSoftwareSystems();
contextView.addAllPeople();

StructurizrClient structurizrClient = new StructurizrClient(workspace);
workspace.setId(1234);
structurizrClient.putWorkspace(workspace);
```

techtribes.je - System Context

Slide 4:3 - Landscape

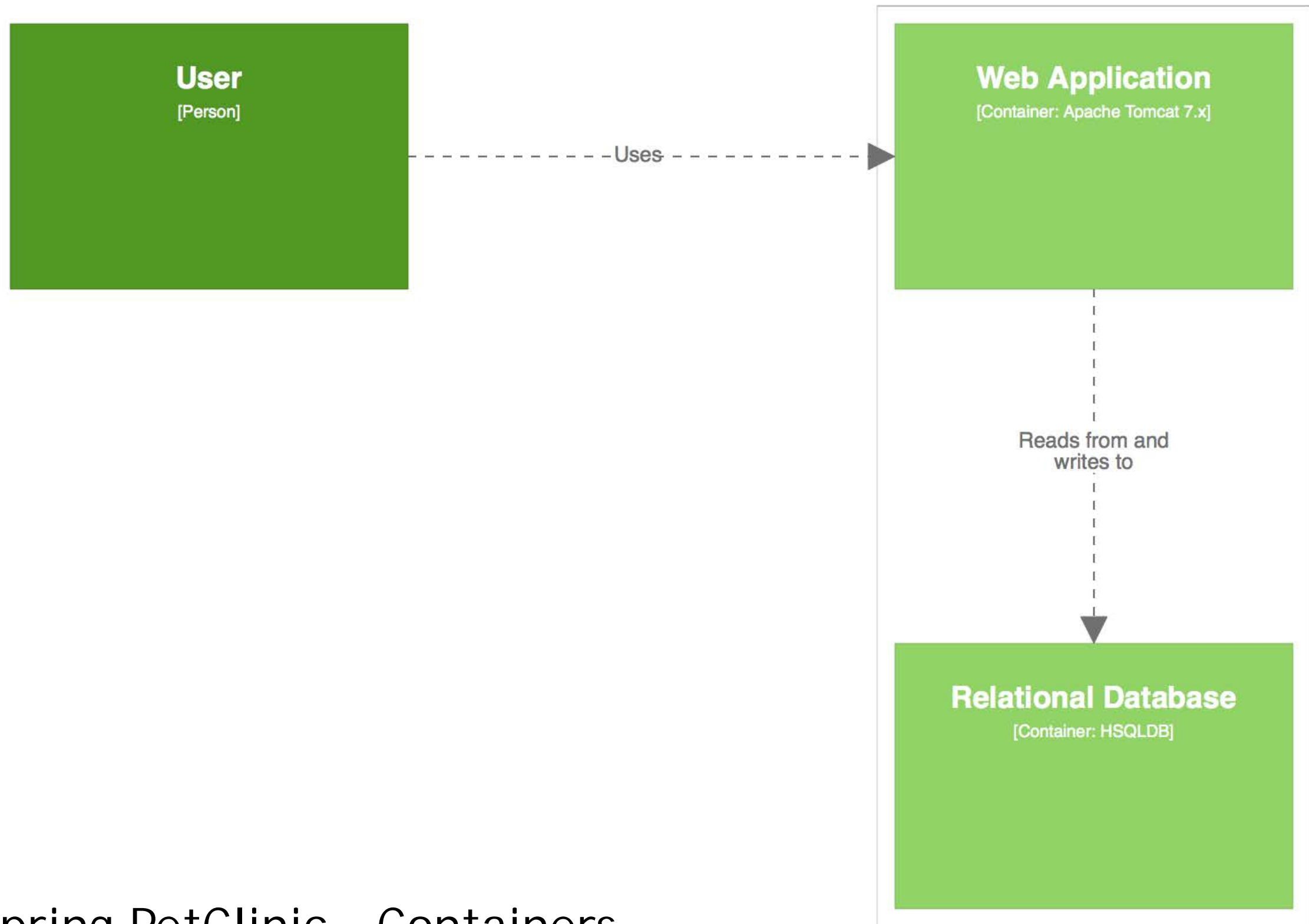


structurizr.com

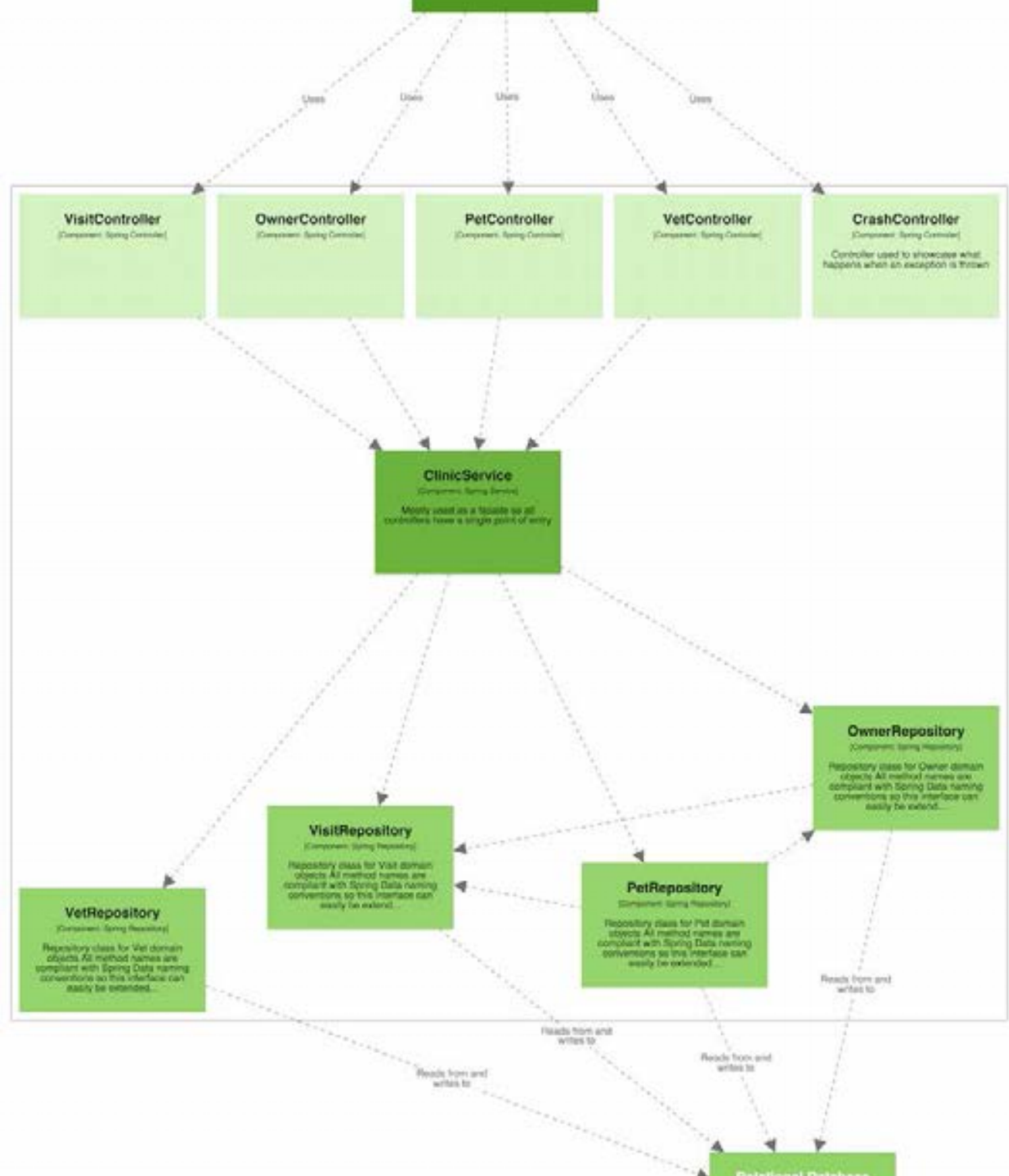
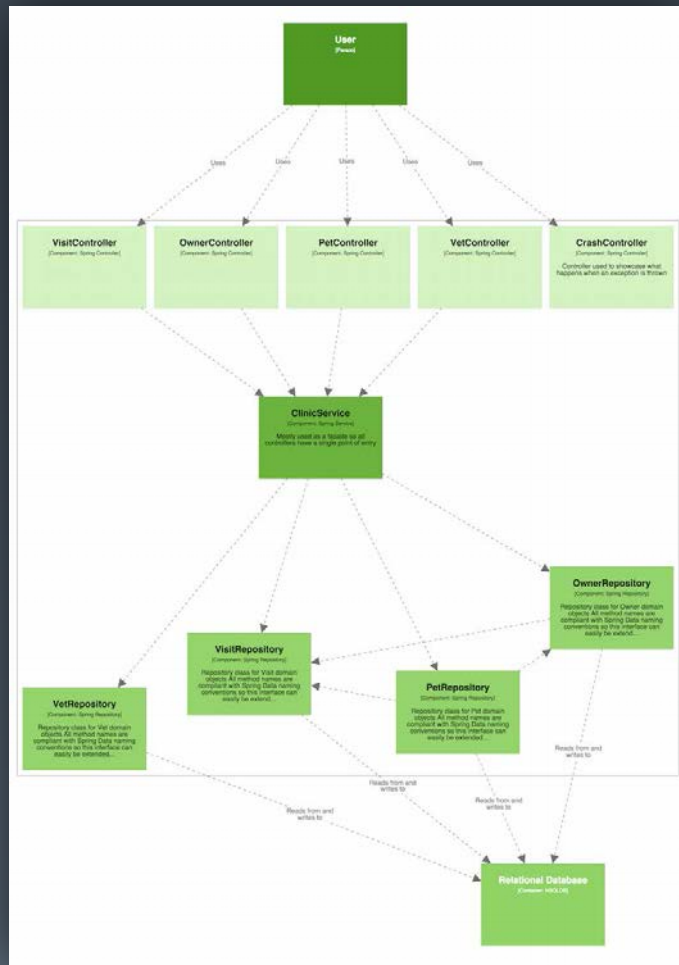
(not open source, but there is a fully functional free tier)



Spring PetClinic - System Context



Spring PetClinic - Containers

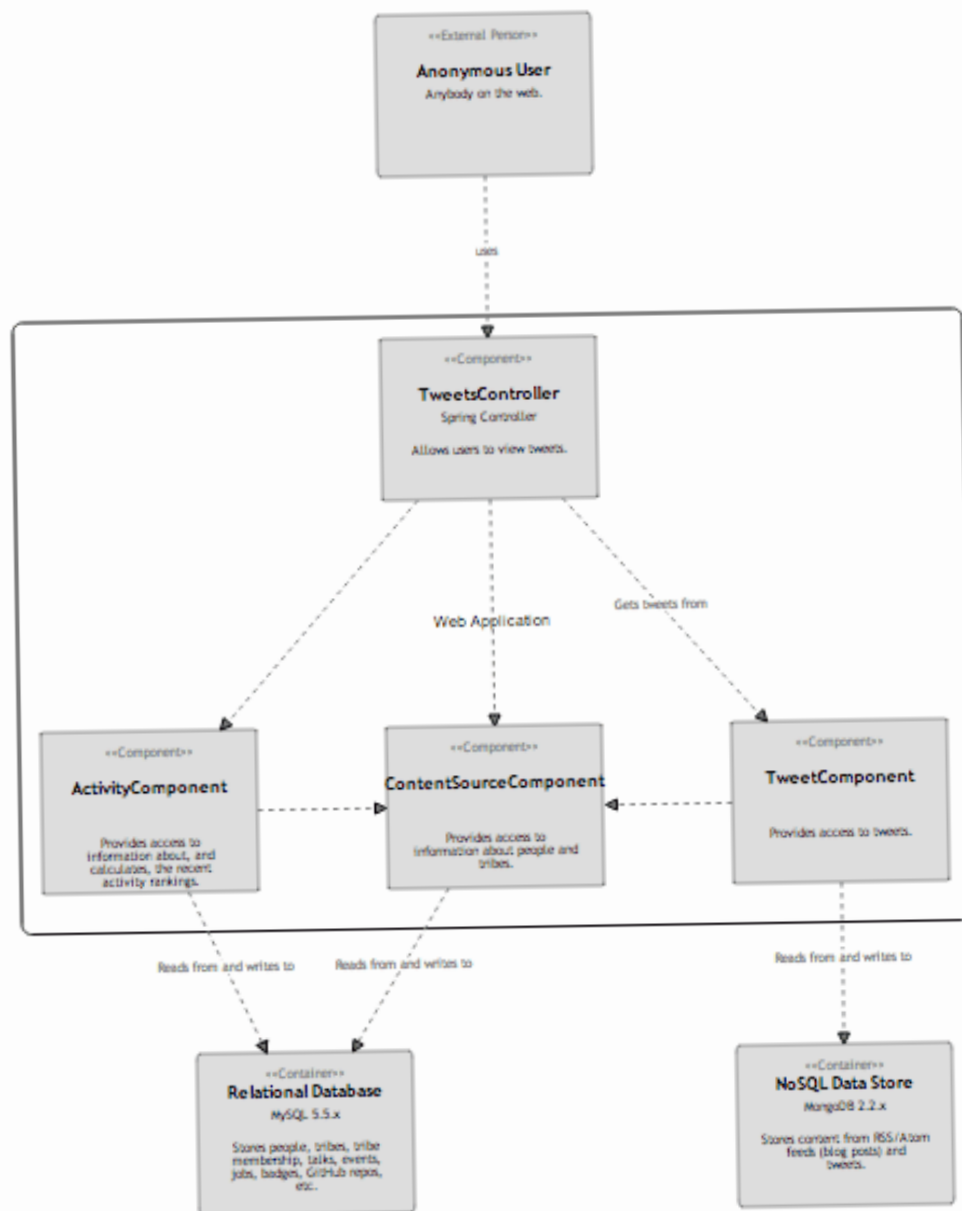


Spring PetClinic

- Web Application
- Components


```
private static void createComponentViewsForWebApplication(Model model) {
    SoftwareSystem techTribes = model.getSoftwareSystemWithName("techtribes.je");
    Container contentUpdater = techTribes.getContainerWithName("Content Updater");
    Container webApplication = techTribes.getContainerWithName("Web Application");

    // create one component view per Spring controller
    Set<Component> controllers = webApplication.getComponents().stream()
        .filter(c -> c.getTechnology().equals("Spring Controller")).collect(Collectors.toSet());
    for (Component controller : controllers) {
        ComponentView view = model.createComponentView(techTribes, webApplication);
    }
}
```



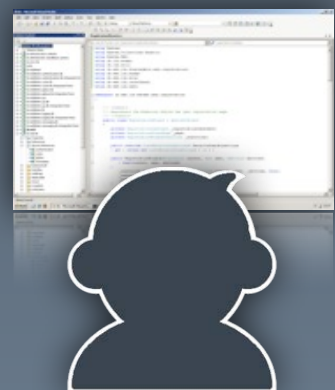
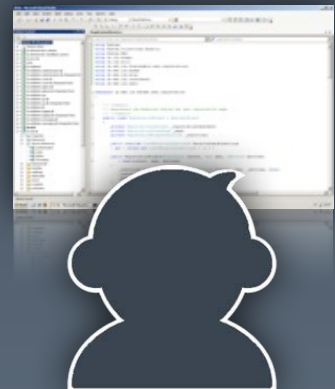
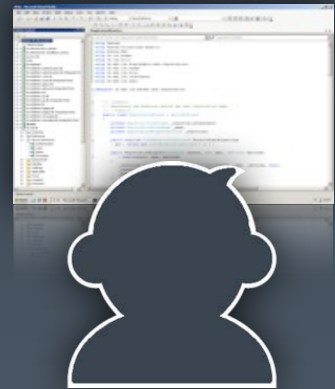
3,"name":"Aggregated User","description":"A user or business with content that is aggregated into the website.,"location

✓ techtribes.je – System Context

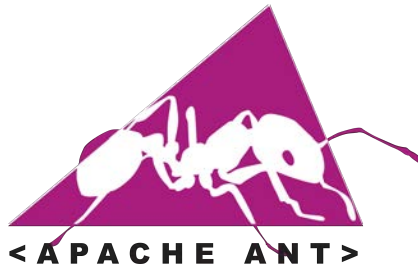
techtribes.je – Containers

techtribes.je – Content Updater – Components – Awarding badges
 techtribes.je – Content Updater – Components – Updating information from external systems
 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 – BadgesController
 techtribes.je – Web Application – Components – CodeController
 techtribes.je – Web Application – Components – ContentController
 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 – SecurityController
 techtribes.je – Web Application – Components – TalksController
 techtribes.je – Web Application – Components – TribesController
 techtribes.je – Web Application – Components – TweetsController
 techtribes.je – Web Application – Components – UserProfileController

A model as code provides opportunities...



TC TeamCity

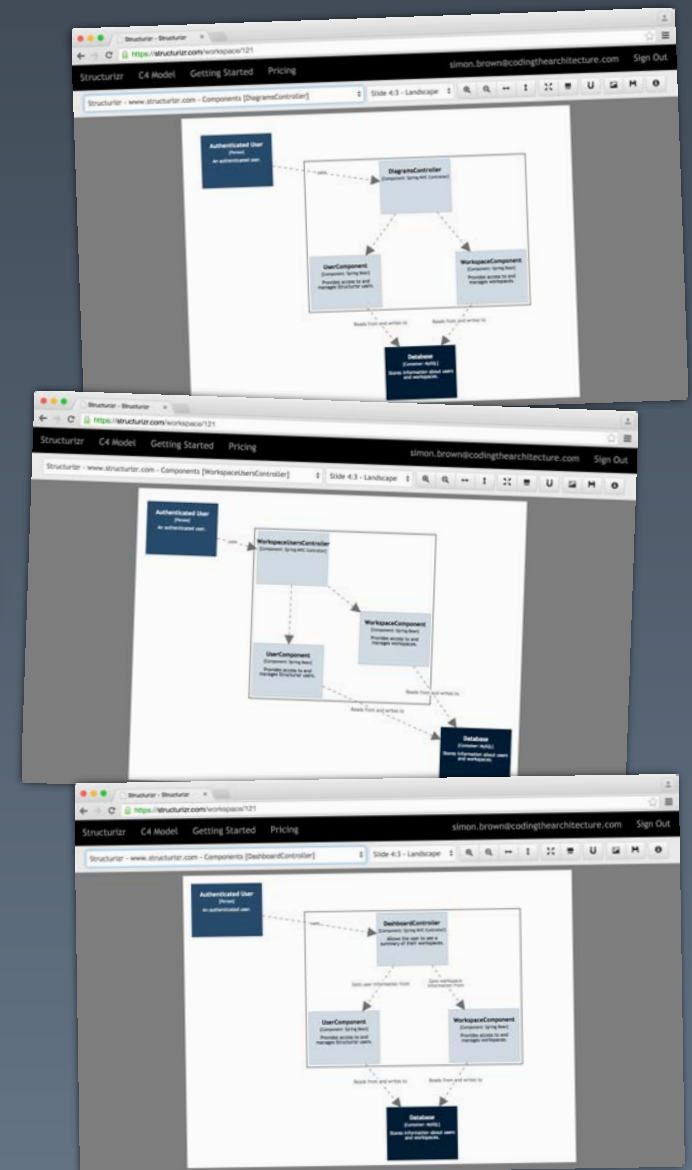


gradle

maven



Jenkins



...and build pipeline integration keeps software architecture models up-to-date

Software architecture

really is

for developers

: -)

The point of this?

Maintainability is
inversely proportional
to the number of

[public classes
dependencies
microservices]

A good
architecture
enables

agility

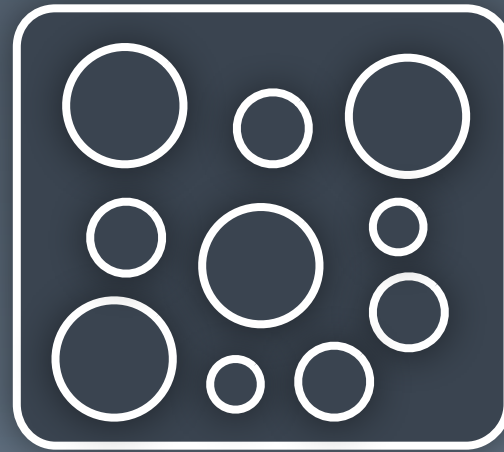
Agility

is a

quality attribute

*Monolithic
architecture*

*Service-based
architecture*
(SOA, micro-services, etc)



Something in between
(components)

If you can't build a
**structured
monolith,**

what makes you think
microservices is the answer!?



Simon Brown

@simonbrown

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



RETWEETS

223

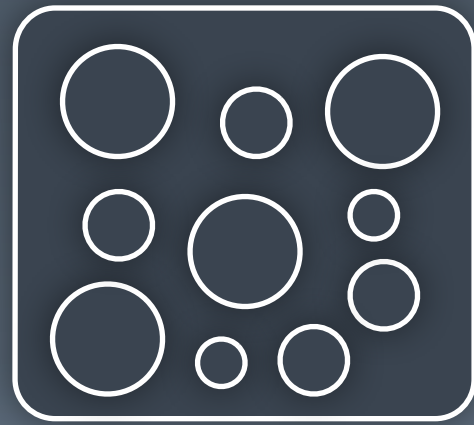
FAVORITES

86



10:49 AM - 4 Mar 2015

Well-defined, in-process components is a stepping stone to out-of-process components (i.e. microservices)



High cohesion

Low coupling

Focussed on a business capability

Bounded context or aggregate

Encapsulated data

Substitutable

Composable

From components
to microservices



<- All of that plus

Individually deployable

Individually upgradeable

Individually replaceable

Individually scalable

Heterogeneous technology stacks

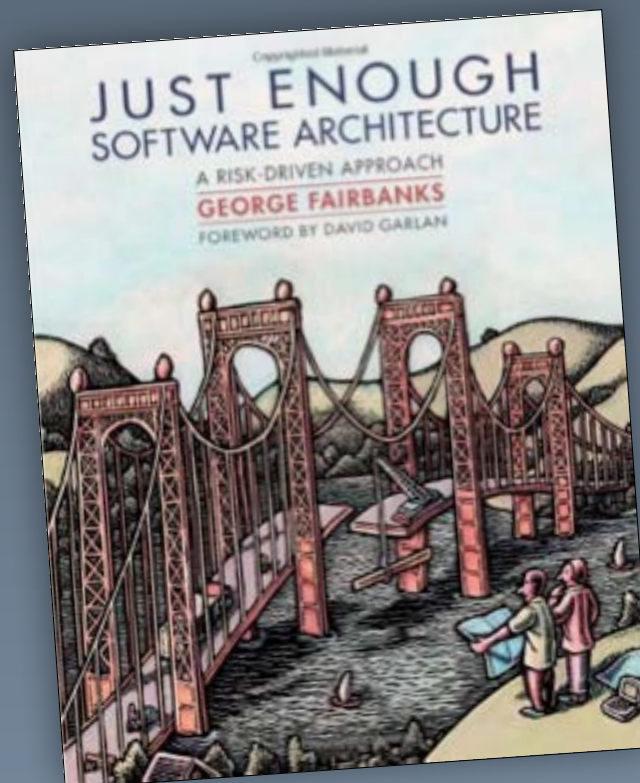
Choose
microservices for
the benefits,
not because your monolithic
codebase is a mess :-)

If your software system
is hard to work with,

change it!

Think about how to align the
software
architecture
and the
code

Be conscious of the software
architecture model ... adopt an
architecturally-evident
coding style



Stop making every class

public

\$1 charity donation
every time you type

public class

without thinking :-)

If the software architecture
model is *in* the code,
it can be extracted
from the code



simon.brown@codingthearchitecture.com

@simonbrown on Twitter



\$10

until
May 7th
with this code



<https://leanpub.com/software-architecture-for-developers>

/c/saturn15