Hogeschool van Arnhem en Nijmegen
HAN University of Applied Sciences

Software Architecture Course

Simon Brown’s C4 Framework

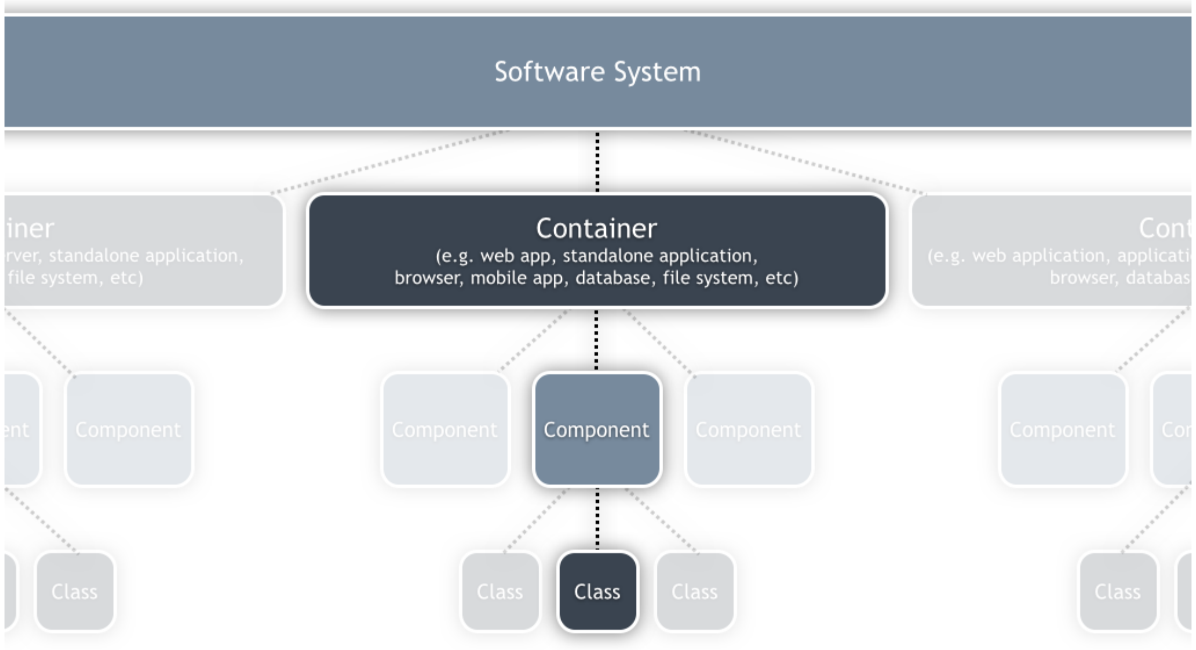
2016 Uwe van Heesch

Motivation

- With the rise of agile software development, people lost their ability to communicate and document software design
- Development teams **don't use UML** any more
- Instead they draw **informal sketches** using mainly boxes and lines
- Such diagrams do not capture the design itself, they only support the story of someone explaining the design
- This is problematic, because **agility requires a solid understanding of the architectural design** of the system
- "Software architecture enables agility"

The C4 Framework

- Document the architectural design of a system using a number of **diagrams at varying levels of abstraction**
- Make sure you don't create a single "uber-diagram"
- Use a **common set of abstractions** to describe the major building blocks of a software system: containers, components, and classes


Software System
iner
rver, standalone application,
file system, etc)
Container
(e.g. web app, standalone application,
browser, mobile app, database, file system, etc)
Cont
(e.g. web application, applicati
browser, databas
ent
Component
Component
Component
Component
Component
Co
Class
Class
Class
Class
Class

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**.

The C4 Framework

- **System**: a system is the highest level of abstraction and represents **something that delivers value to somebody**. A system is **made up of a number of separate containers**. Examples include a financial risk management system, an internet banking system, a website and so on.
- **Container**: a container represents **something in which components are executed or where data resides**. This could be anything from a web or application server through to a rich client application or database. Containers are typically **executables** that are started as a part of the overall system. For example, a Java EE application or .NET website. Any inter-container communication is likely to require a remote interface such as a SOAP web service, RESTful interface, Java RMI, or the like.

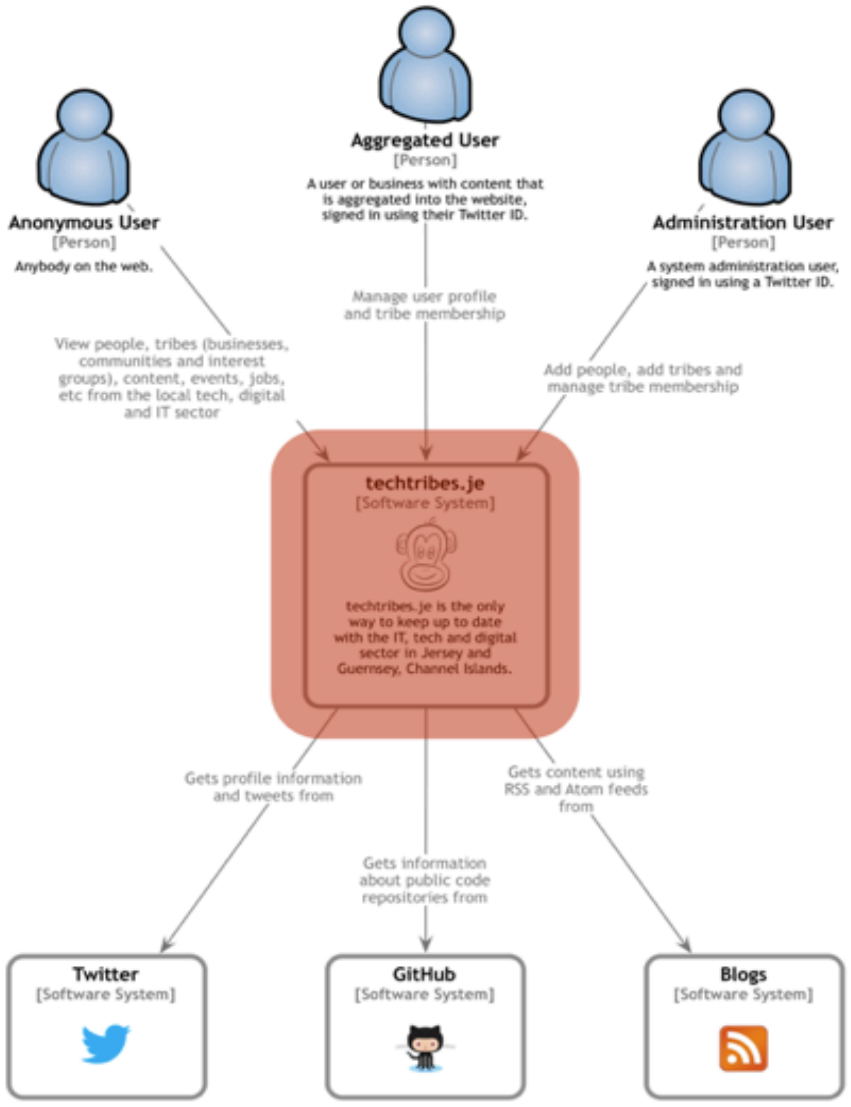
The C4 Framework

- **Components**: a component can be thought of as **a logical grouping of one or more classes**. For example, an audit component or an authentication service that is used by other components to determine whether access is permitted to a specific resource. Components are typically made up **of a number of collaborating classes**, all sitting **behind a higher level contract**, which is typically expressed using interfaces.
- **Classes**: in an OO world, classes are the **smallest building blocks** of our software systems. Think of a Java class as one example.

The C4 Framework

- To document a software architecture, you create a collection of diagrams of four different types (therefore the name "C4")
  - **Context** diagram: A high-level diagram that sets the scene; including key system dependencies and actors.
  - **Container** diagram: A container diagram shows the high-level technology choices, how responsibilities are distributed across them and how the containers communicate.
  - **Component** diagrams: For each container, a component diagram lets you see the key components and their relationships.
  - **Class** diagrams (optional): For key components, draw UML class diagrams to explain how a particular pattern or component will be (or has been) implemented. The classes may be incomplete and you may omit details, but they need to be consistent and syntactically correct.

Example Context Diagram



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

Anonymous User
[Person]
Anybody on the web.

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

Manage user profile and tribe membership

View people, tribes (businesses, communities and interest groups), content, events, jobs, etc from the local tech, digital and IT sector

Add people, add tribes and manage tribe membership

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.

Gets profile information and tweets from

Gets information about public code repositories from

Gets content using RSS and Atom feeds from

Twitter
[Software System]

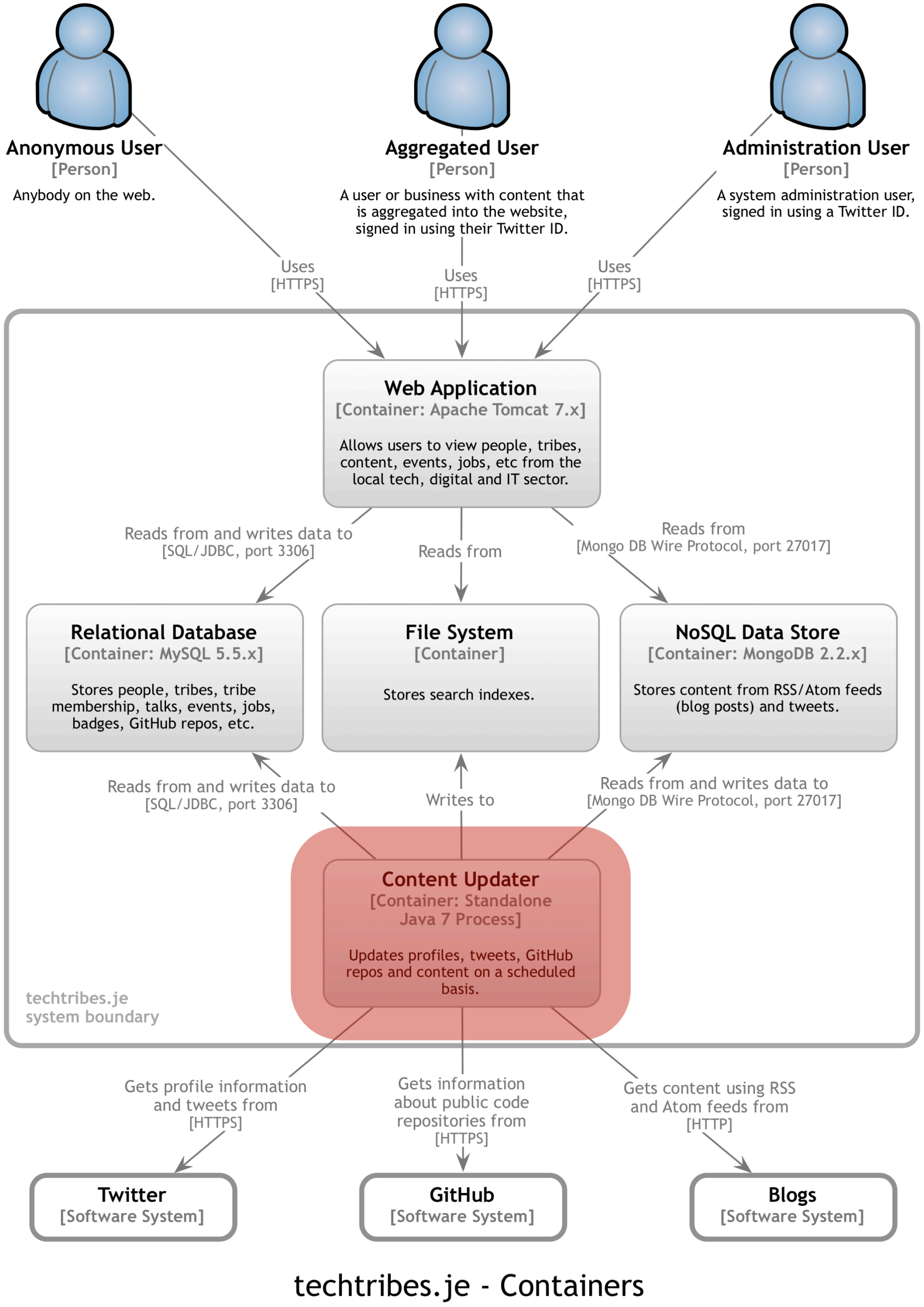
GitHub
[Software System]

Blogs
[Software System]

techtribes.je - Context

http://www.codingthearchitecture.com/presentations/wicsacomparch2016-the-art-of-visualising-software-architecture

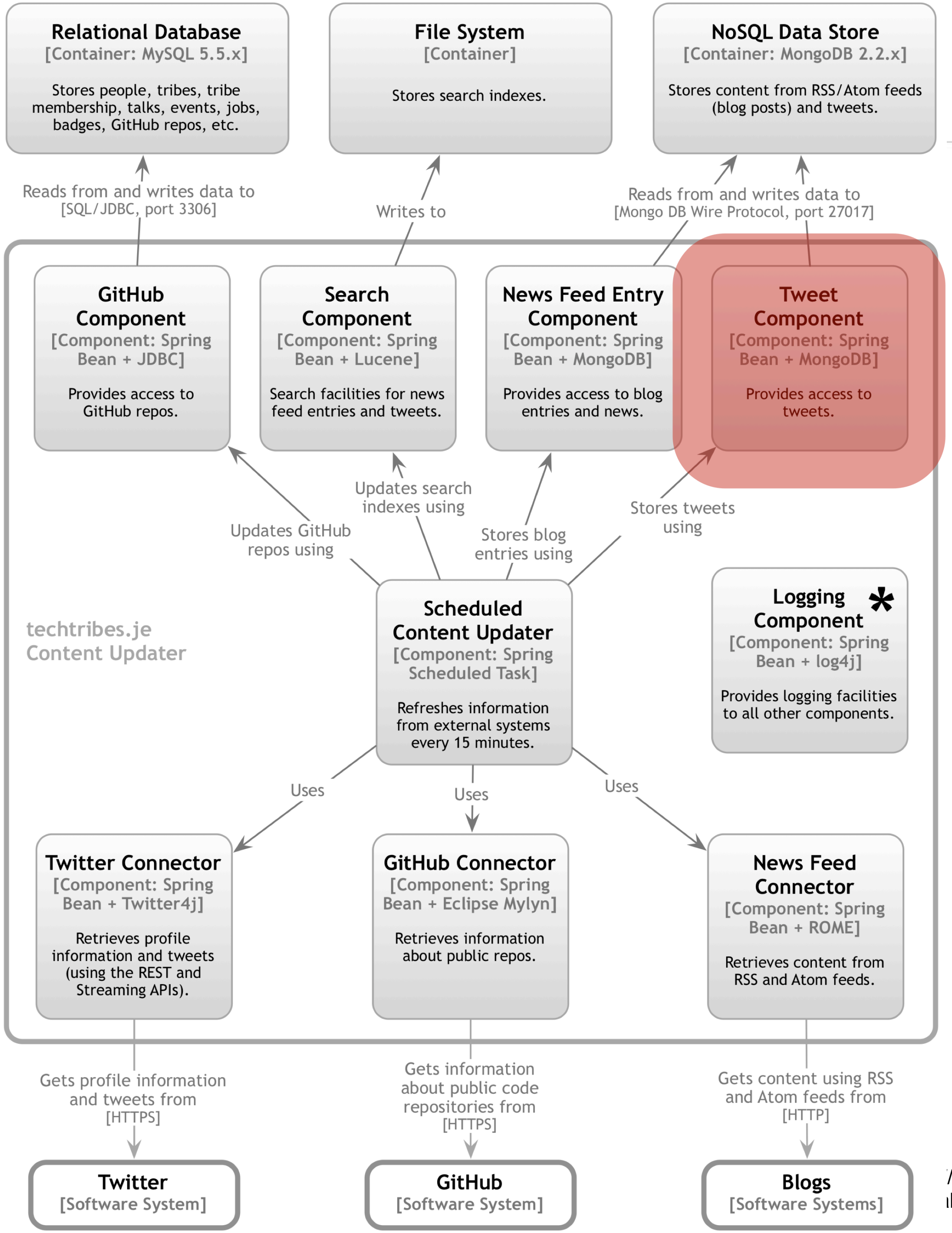
Example Container Diagram


Anonymous User
[Person]
Anybody on the web.
Aggregated User
[Person]
A user or business with content that is aggregated into the website, signed in using their Twitter ID.
Administration User
[Person]
A system administration user, signed in using a Twitter ID.
Uses
[HTTPS]
Uses
[HTTPS]
Uses
[HTTPS]
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.
Reads from and writes data to
[SQL/JDBC, port 3306]
Reads from
Reads from
[Mongo DB Wire Protocol, port 27017]
Relational Database
[Container: MySQL 5.5.x]
Stores people, tribes, tribe membership, talks, events, jobs, badges, GitHub repos, etc.
File System
[Container]
Stores search indexes.
NoSQL Data Store
[Container: MongoDB 2.2.x]
Stores content from RSS/Atom feeds (blog posts) and tweets.
Reads from and writes data to
[SQL/JDBC, port 3306]
Writes to
Reads from and writes data to
[Mongo DB Wire Protocol, port 27017]
Content Updater
[Container: Standalone Java 7 Process]
Updates profiles, tweets, GitHub repos and content on a scheduled basis.
techtribes.je
system boundary
Gets profile information and tweets from
[HTTPS]
Gets information about public code repositories from
[HTTPS]
Gets content using RSS and Atom feeds from
[HTTP]
Twitter
[Software System]
GitHub
[Software System]
Blogs
[Software System]

techtribes.je - Containers

http://www.codingthearchitecture.com/presentations/wicsacomparch2016-the-art-of visualising-software-architecture

Example Component Diagram



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

File System
[Container]
Stores search indexes.

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

Reads from and writes data to
[SQL/JDBC, port 3306]

Writes to

Reads from and writes data to
[Mongo DB Wire Protocol, port 27017]

GitHub Component
[Component: Spring Bean + JDBC]
Provides access to GitHub repos.

Search Component
[Component: Spring Bean + Lucene]
Search facilities for news feed entries and tweets.

News Feed Entry Component
[Component: Spring Bean + MongoDB]
Provides access to blog entries and news.

Tweet Component
[Component: Spring Bean + MongoDB]
Provides access to tweets.

Updates GitHub repos using

Updates search indexes using

Stores blog entries using

Stores tweets using

techtribes.je
Content Updater

Scheduled Content Updater
[Component: Spring Scheduled Task]
Refreshes information from external systems every 15 minutes.

Logging Component *
[Component: Spring Bean + log4j]
Provides logging facilities to all other components.

Uses

Uses

Uses

Twitter Connector
[Component: Spring Bean + Twitter4j]
Retrieves profile information and tweets (using the REST and Streaming APIs).

GitHub Connector
[Component: Spring Bean + Eclipse Mylyn]
Retrieves information about public repos.

News Feed Connector
[Component: Spring Bean + ROME]
Retrieves content from RSS and Atom feeds.

Gets profile information and tweets from
[HTTPS]

Gets information about public code repositories from
[HTTPS]

Gets content using RSS and Atom feeds from
[HTTP]

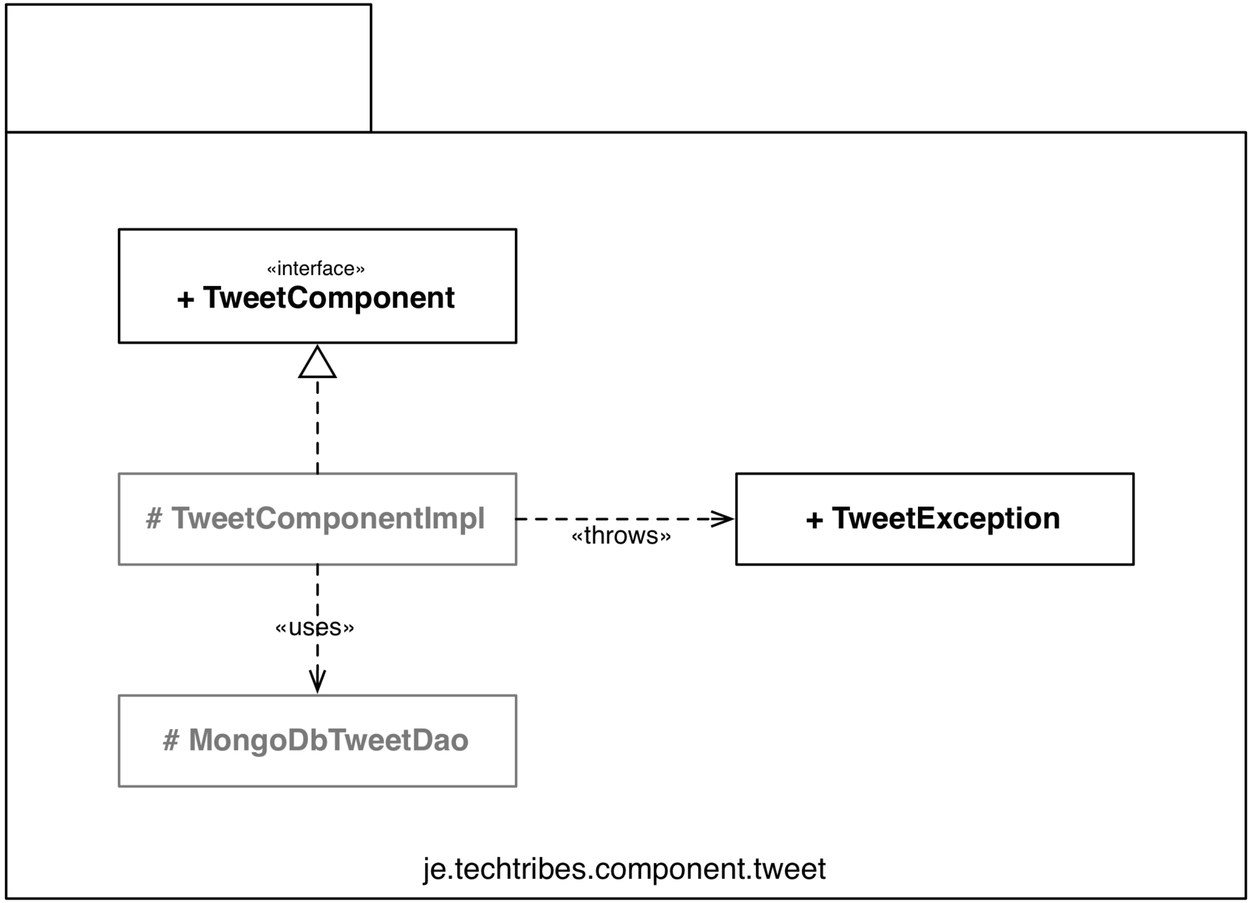
Twitter
[Software System]

GitHub
[Software System]

Blogs
[Software Systems]

/www.codingthearchitecture.com/presentations/wicsacomparch2016-the-art-of
ılising-software-architecture

Example Class Diagram



http://www.codingthearchitecture.com/presentations/wicsacomparch2016-the-art-of visualising-software-architecture

Discussion

- C4 embraces the concepts of ISO/IEC/IEEE 42010: **architecture is documented in multiple views**
- Each view (here diagram) documents specific aspects of the system only, i.e. it **satisfies different concerns**
- A limitation is that views in C4 are hierarchical, i.e. apart from context and class diagrams, each diagram is a sub-diagram of another diagram
- C4 has its own graphical notation, UML is only used in the class diagram (not mandatory though)
- In principle, the same views could be created using UML diagrams
- How do the diagrams map to Kruchten's views?

Next lesson

For the next lesson

Have a look at:

https://www.structurizr.com/

Skim the articles

- Tyree, J., & Akerman, A. (2005). Architecture Decisions: Demystifying Architecture. IEEE Software, 22(2), 19-27.
- van Heesch, U., Avgeriou, P., & Hilliard, R. (2012). A documentation framework for architecture decisions. *Journal of Systems and Software*, *85*(4), 795-820.

Prepare a 5 minute lightning talk* on the importance of architecture decision documentation

*http://en.wikipedia.org/wiki/Lightning_talk

References

- ISO/IEC/IEEE Systems and software engineering -- Architecture description," ISO/IEC/IEEE 42010:2011(E) , Dec. 1 2011
- Kruchten, P. (1995). The 4+ 1 View Model of Architecture. IEEE Software,12(6), 42-50.
- Brown, S. (2013). Software Architecture for Developers. Leanpub