AIM

SIMON BROWN'S C4 FRAMEWORK

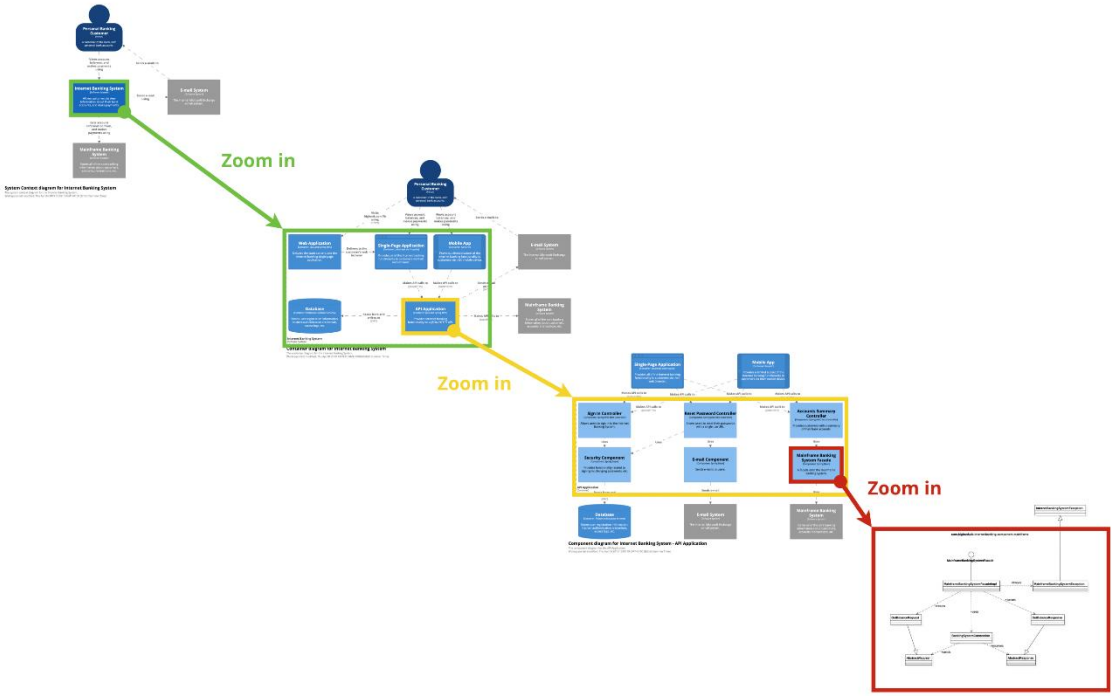
HAN_UNIVERSITY
OF APPLIED SCIENCES

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 for high level design
- 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



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.

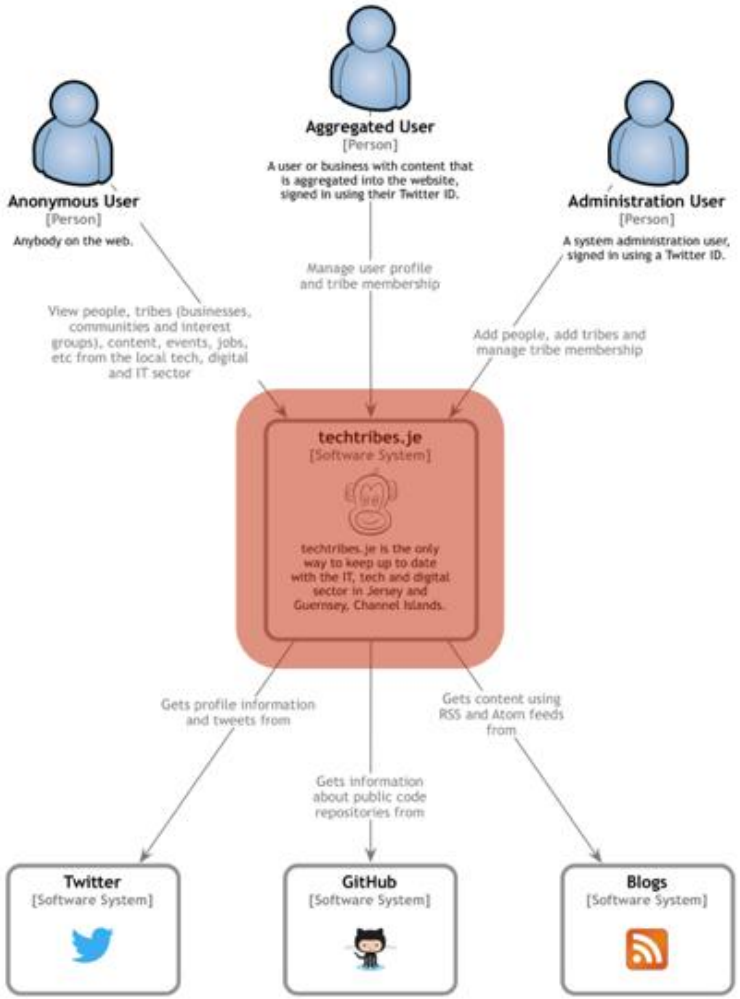
THE C4 FRAMEWORK

- 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.
- Code: 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

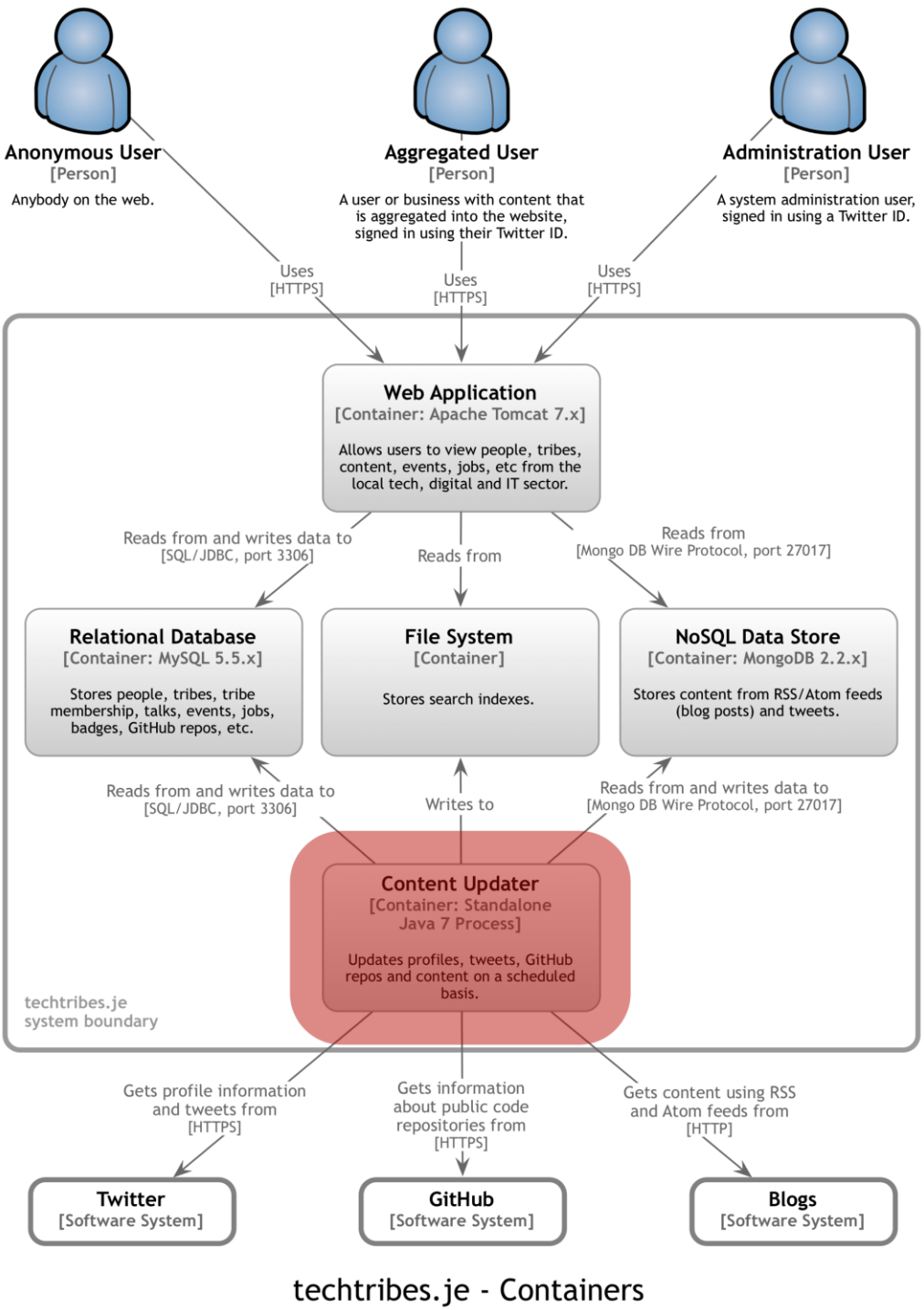
- To document a software architecture, you create a collection of diagrams of four different types (therefore the name “C4”)
  - **Context** diagram: A high-level diagram that sets the scene; including key system dependencies and actors.
  - **Container** diagram: A container diagram shows the high-level 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.
  - **Code** 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



techtribes.je - Context

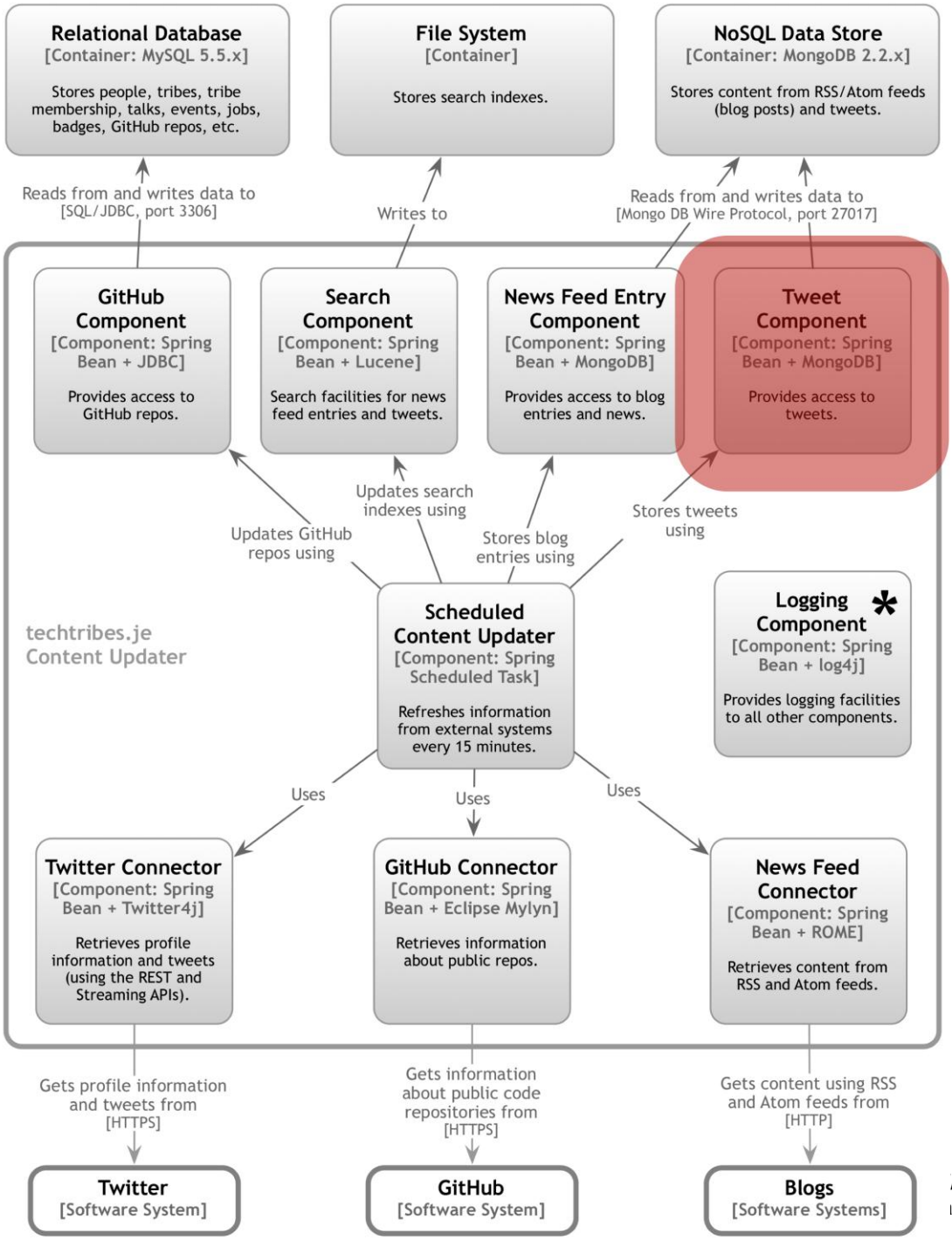
http://www.codingthearchitecture.com/presentations/wicsacompa
rch2016-the-art-of-visualising-software-architecture



techtribes.je - Containers

EXAMPLE CONTAINER DIAGRAM

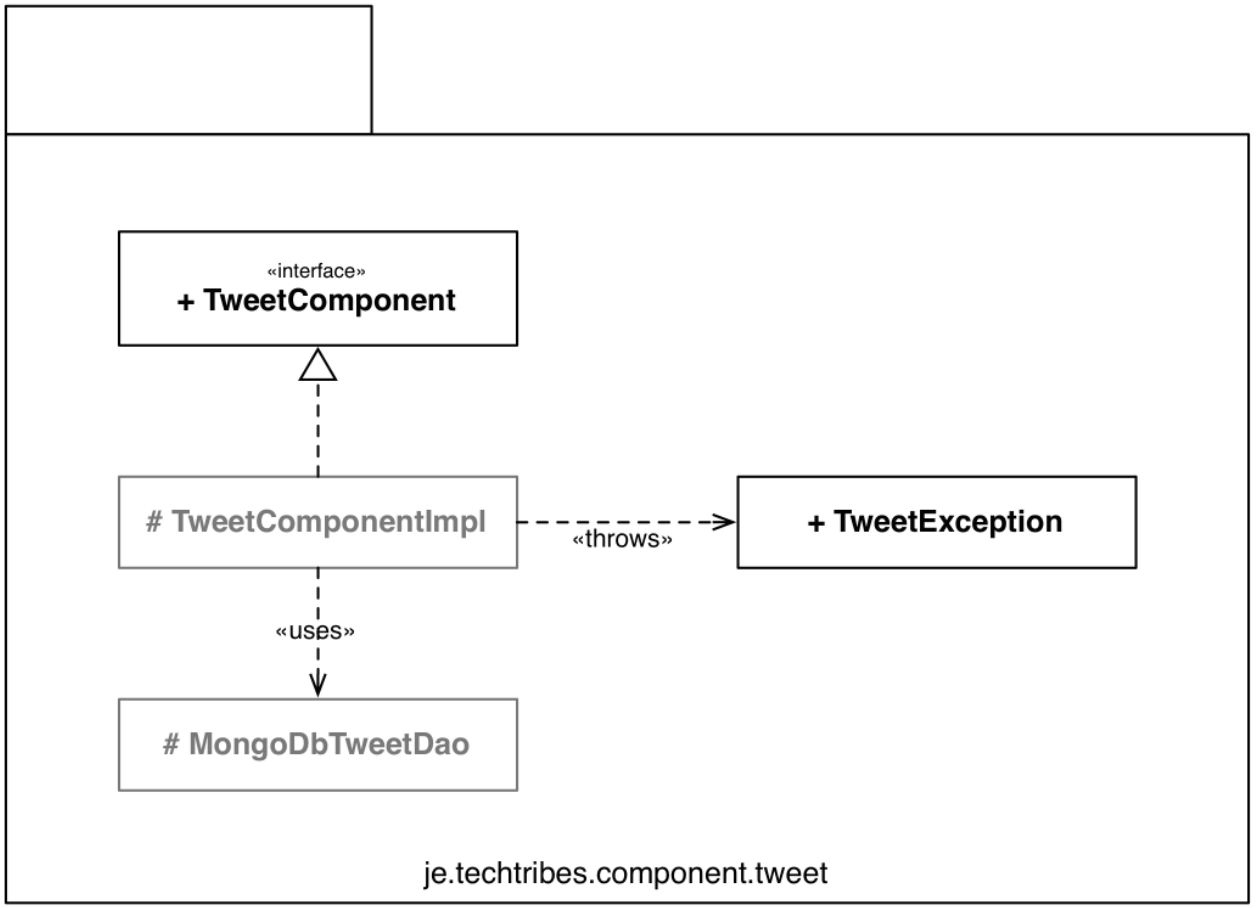
http://www.codingthearchitecture.com/presentations/wicsacomparch2016-the-art-of-visualising-software-architecture



EXAMPLE COMPONENT DIAGRAM

/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