ACADEMIE IT EN MEDIADESIGN – MINOR DEVOPS

THE C4 MODEL

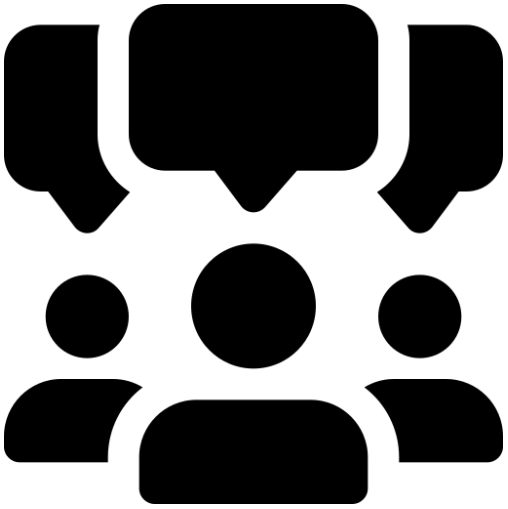
FOR VISUALISING SOFTWARE ARCHITECTURE

HAN_UNIVERSITY
OF APPLIED SCIENCES

DESIGN HAPPENS IN YOUR BRAIN, NOT ON PAPER

- (Software-) design basically happens in your brain
  - You reason about problems
  - You consider (partial) solutions from the past
  - You think of new solution candidates
  - You make a choice
- When collaborating with others, we need to communicate design
- Among others, we use different forms of written documentation for explaining design ideas or design outcome
- Each form of documentation has pros and cons
- Each form of documentation addresses different stakeholder concerns

HANDS ON – 30 MINUTES



- Form groups of 3-5 students
- Discuss the following questions and write down your ideas

- What forms of documentation exist?
  - E.g. whiteboard sketches, class diagrams, …

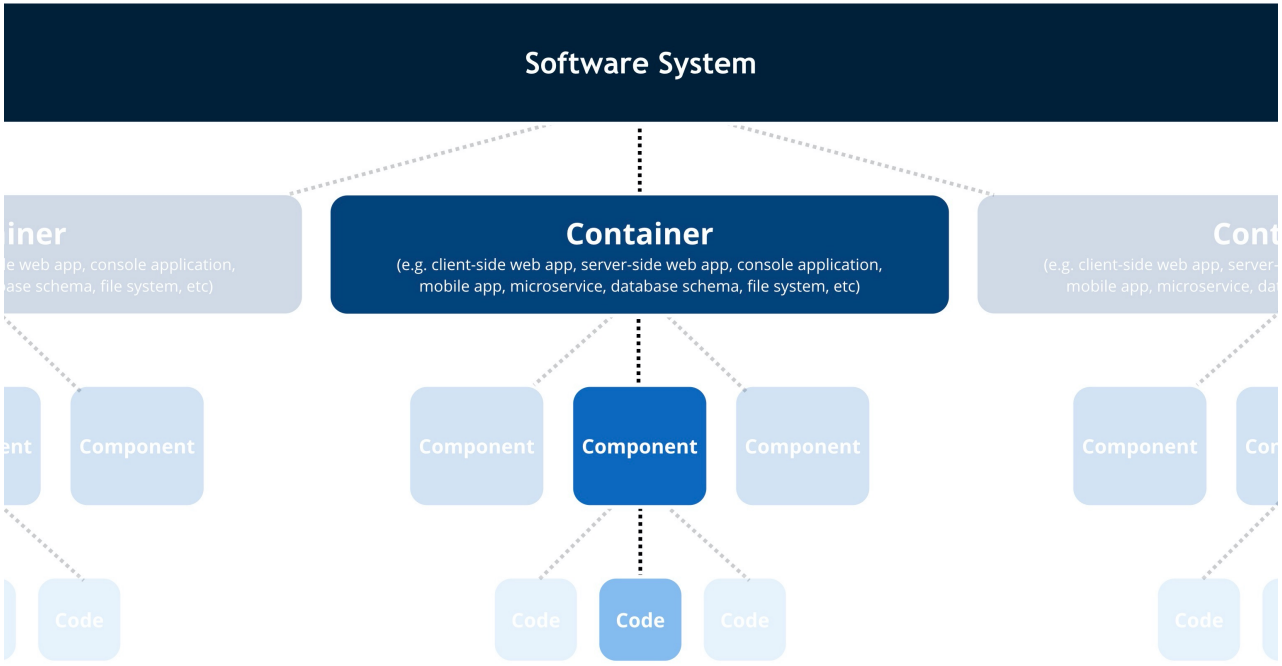
- What are typical consumers (readers) of the documentation?
- What are the information-needs of these consumers?
- What forms of documentation are best suitable for which information needs?

MOTIVATION

- In agile software development, people put less emphasis on the communication and documentation of software design
- Agile teams usually **don’t use UML**
- Instead they draw informal sketches on a whiteboard
- Such sketches do not capture the design itself, they only support the story of someone explaining the design
- A solid shared understanding of the envisioned architecture is vital for agile working

THE C4 FRAMEWORK

- Software documentation **for developers**
- Document architectural design using **hierarchical diagrams**
- Make sure you don’t create a single “uber-diagram”
- Common set of abstractions for major building blocks of a system
- Subsequent **zooming-levels** into the system



C4 BUILDING BLOCKS

- **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 webshop, and so on.

- **Container**: A container is essentially a context or boundary inside which some code is executed or some data is stored; something like a web application, mobile app, desktop application, database, file system, etc. Usually, a container is a separately deployable unit that executes code or stores data. The Container diagram shows the high-level shape of the software architecture and how responsibilities are distributed across it. It also shows the major technology choices and how the containers communicate with one another

C4 BUILDING BLOCKS

- **Component**: 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. These components should map to real abstractions (e.g., a grouping of code) in your codebase.

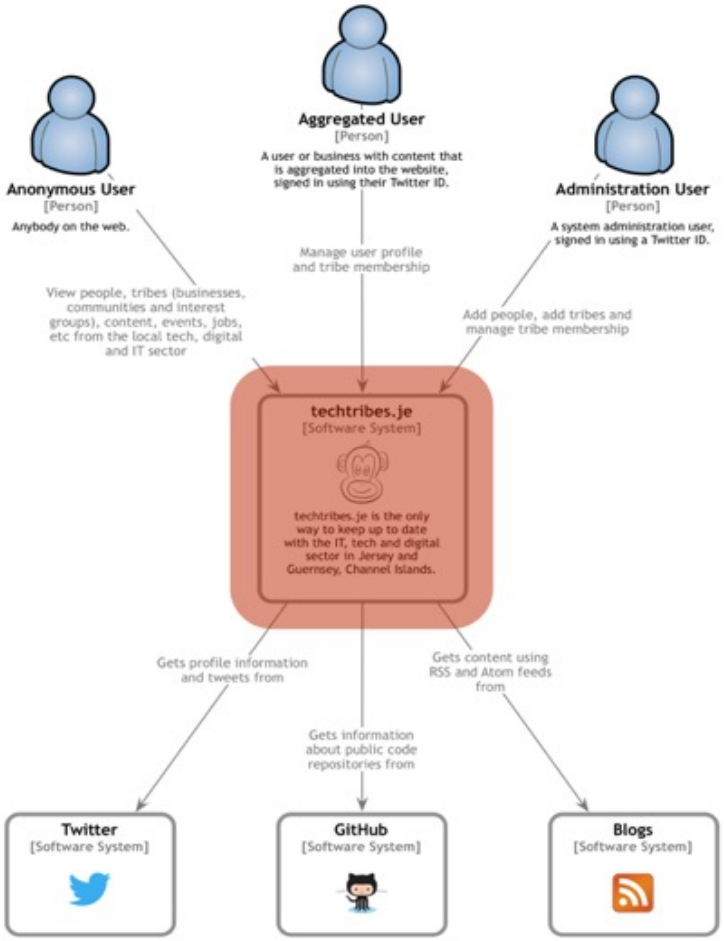
- **Code**: For most projects, components should suffice as most low-level modeling blocks. However, if you really want or need to, you can zoom into an individual component to show how that component is implemented. You can do this, for instance, using a UML class diagram.

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.

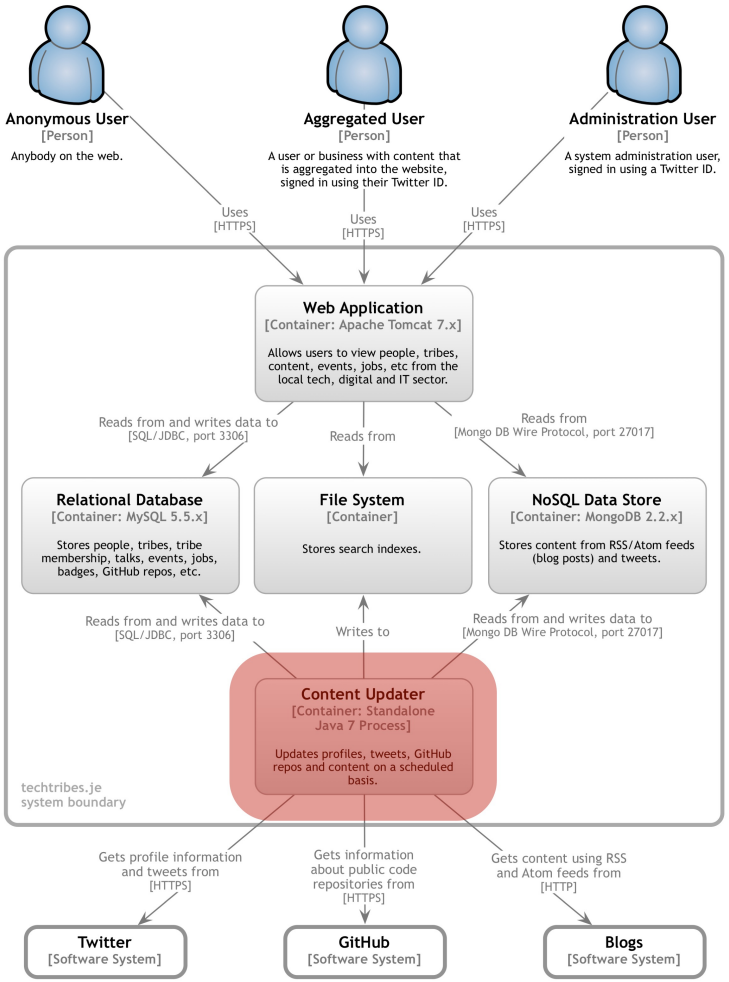
- Supplementary diagrams
  - https://c4model.com/#Notation

CONTEXT DIAGRAM (EXAMPLE 1)



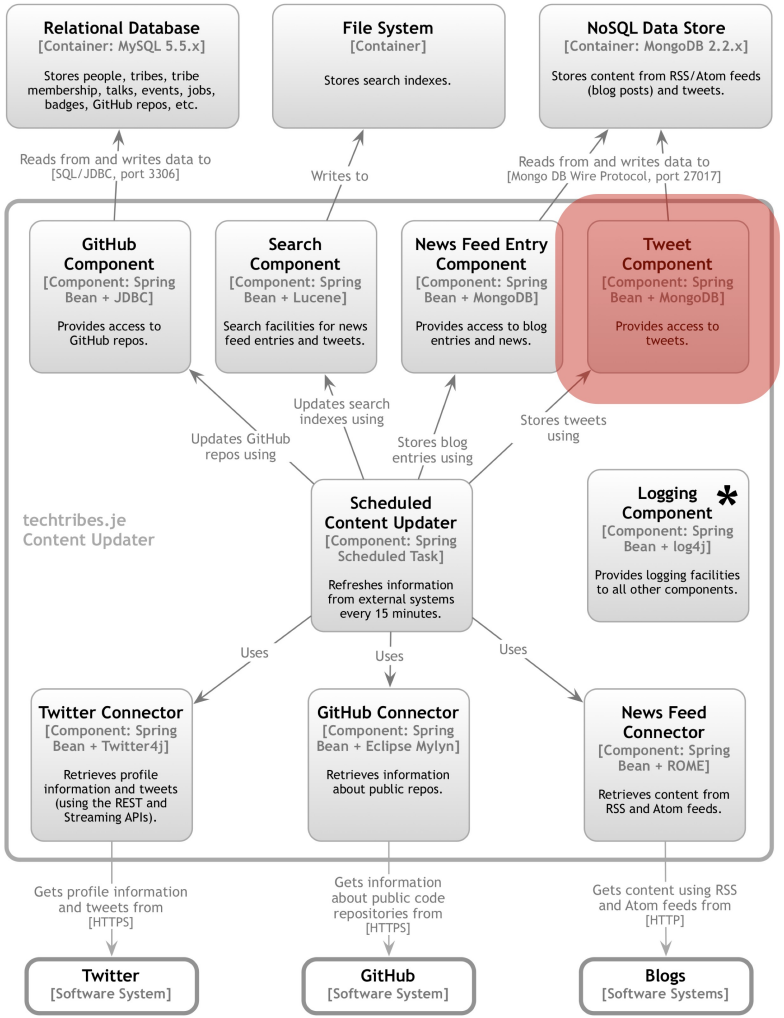
techtribes.je - Context

CONTAINER DIAGRAM (EXAMPLE 1)



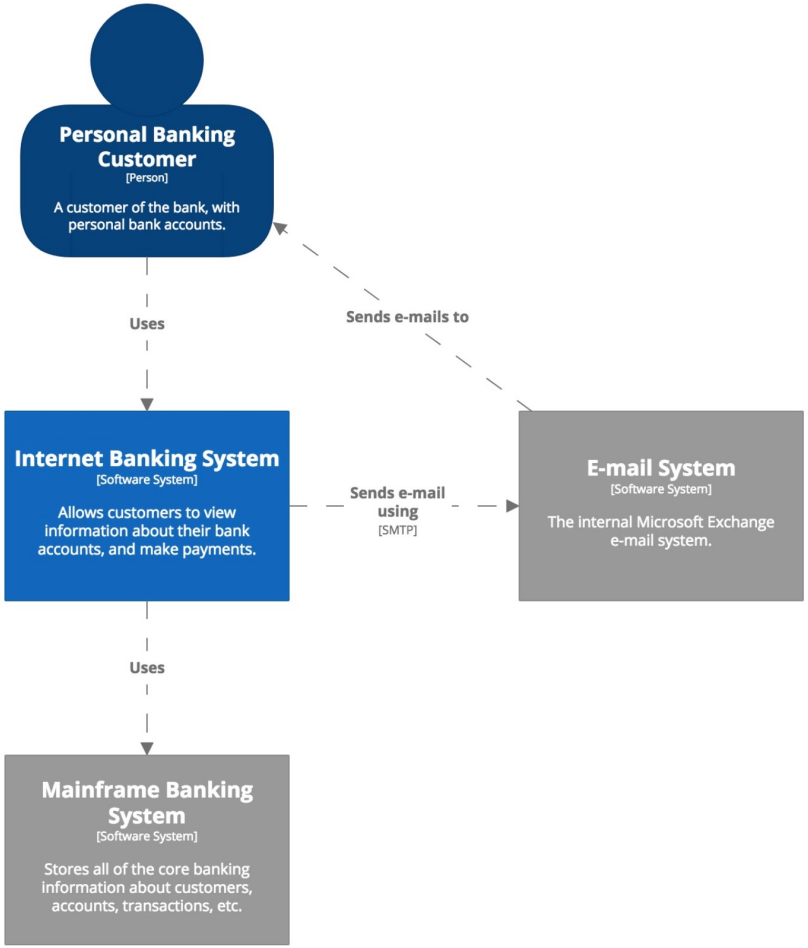
techtribes.je - Containers

COMPONENT DIAGRAM (EXAMPLE 1)

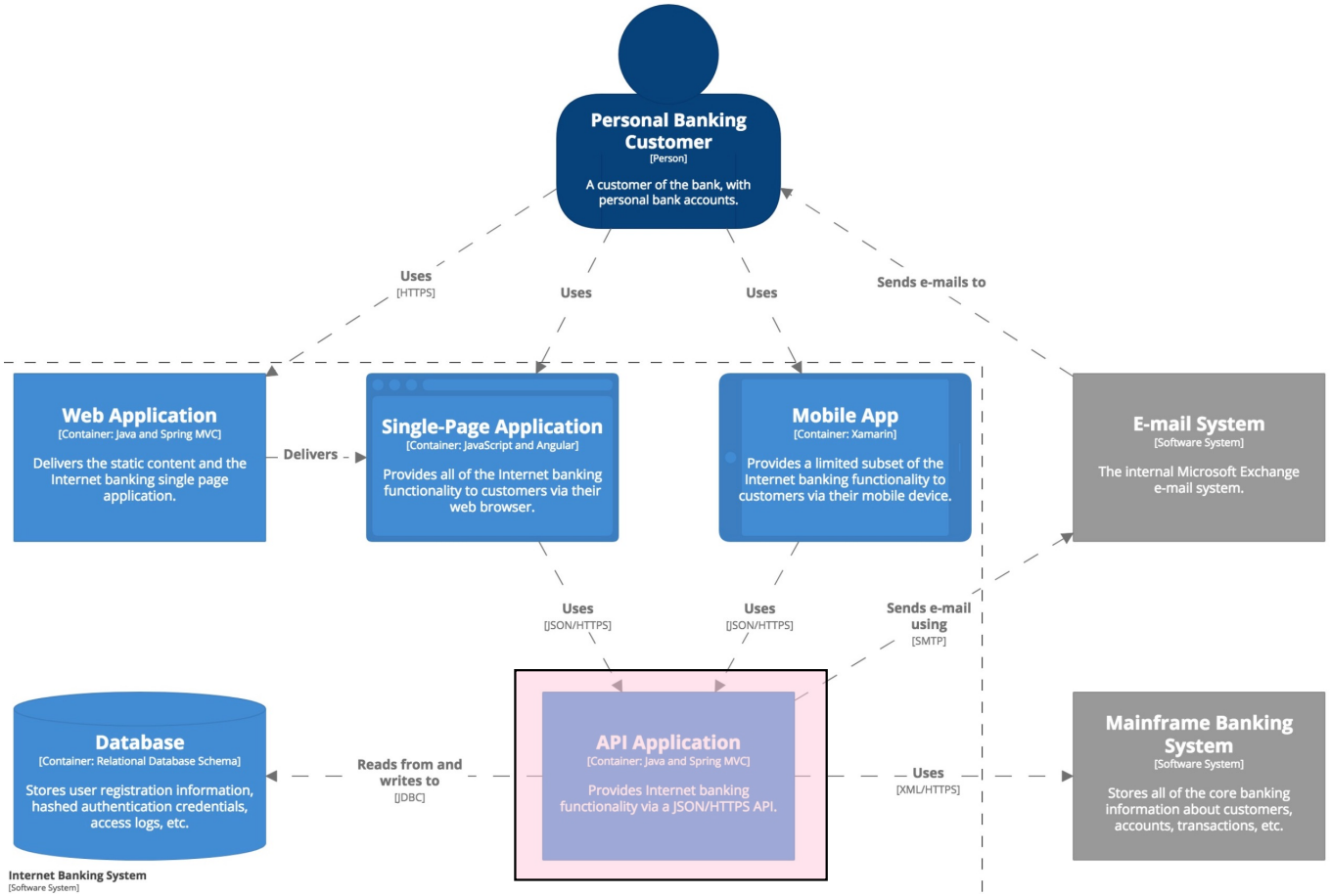


LET'S GO FOR ANOTHER EXAMPLE

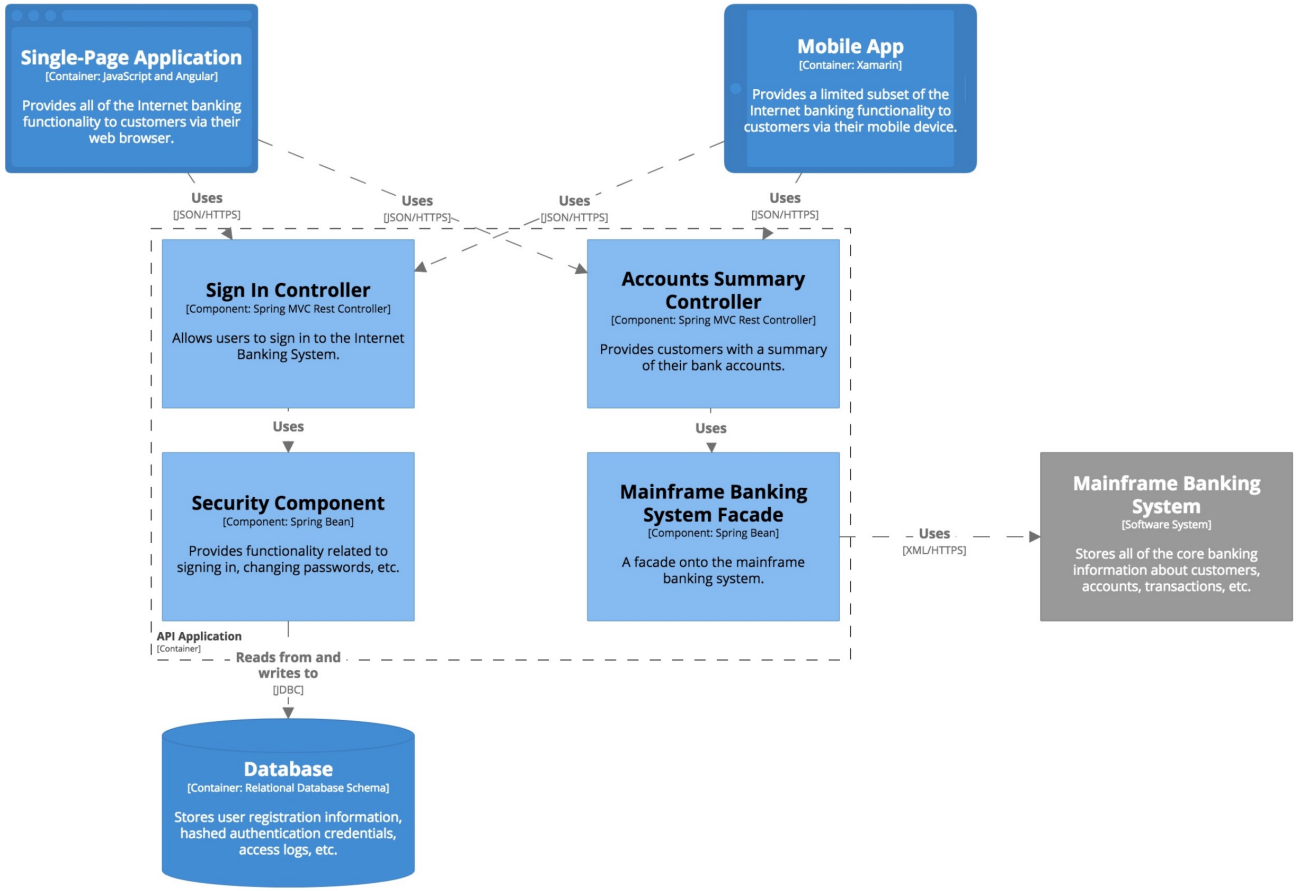
CONTEXT DIAGRAM (EXAMPLE 2)



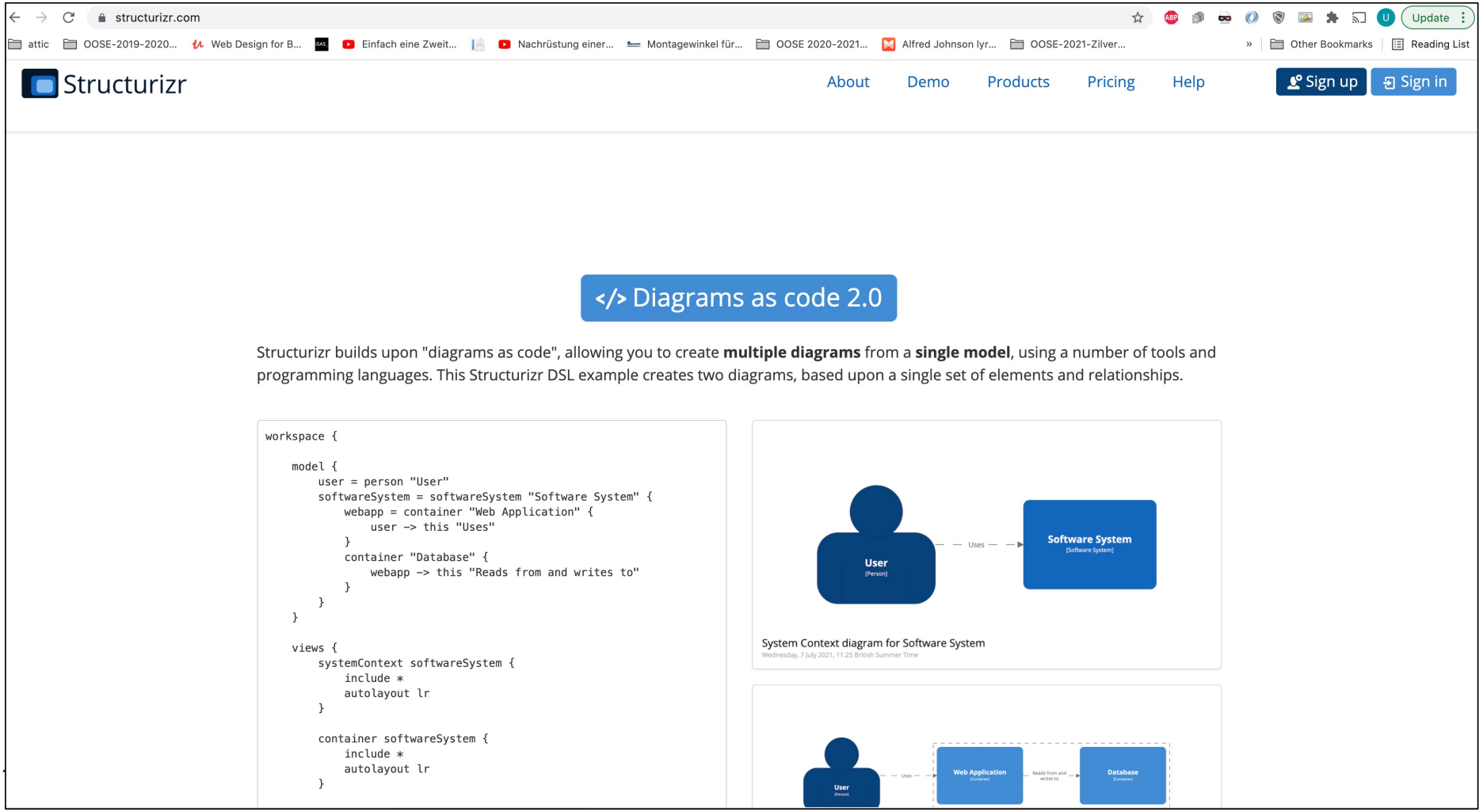
CONTAINER DIAGRAM (EXAMPLE 2)



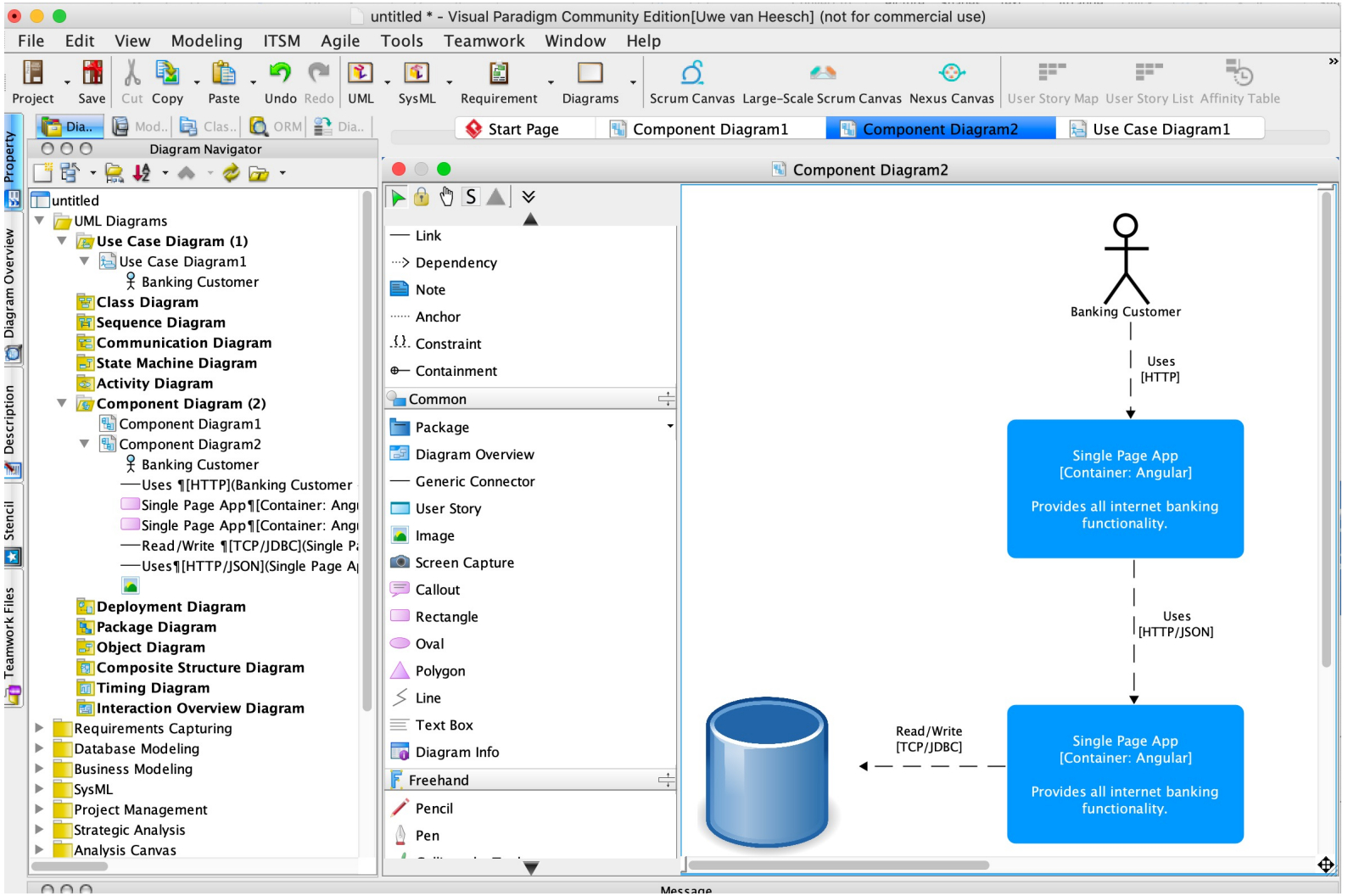
COMPONENT DIAGRAM (EXAMPLE 2)



CREATING DIAGRAMS USING STRUCTURIZR



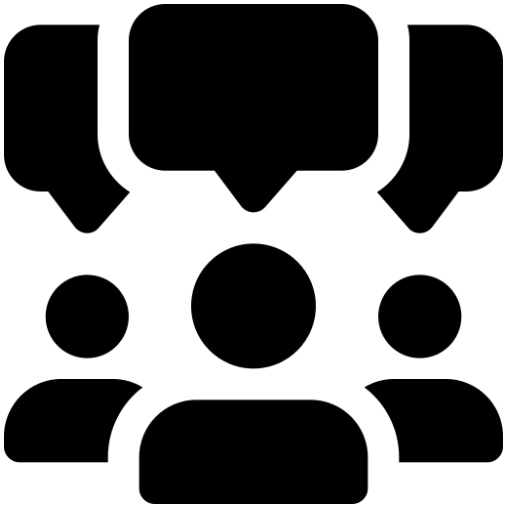
CREATING DIAGRAMS USING VISUAL PARADIGM



CREATING DIAGRAMS USING ANYTHING YOU LIKE

- Just a whiteboard
- Sticky notes on a white board
- Your favorite UML Editor
- A drawing tool
- …

HANDS ON – 60 MINUTES



- Form groups of 3-5 students
- Document a software system you know using C4
- If you are in doubt how to model specific elements of your (envisioned) system
  - Write down the uncertainty
  - Consult https://c4model.com/#Notation
- What information needs from the first exercise are satisfied by the models? What needs are not?
- Some aspects of the systems cannot be modelled using the described C4 models; which aspects and why?