Campus Sontheim

HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

› C4 MODEL

Software Architecture | MSEM | Winter Semester 2019/20

Prof. Dr.-Ing. Andreas Heil

HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

LEANING OBJECTIVES

Understanding problems when **documenting** software architectures

Being familiar with the **four levels** of the **C4 model**

Being able to **apply** the C4 model to **real problems**

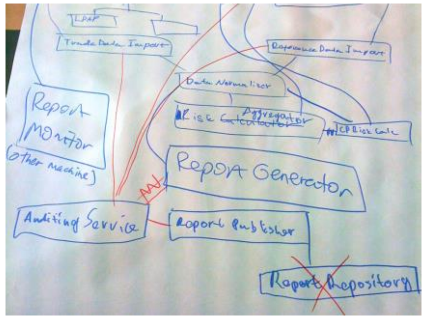
Being able to **document** software architectures using the C4 model

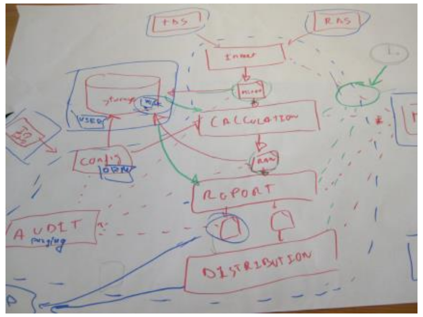
HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

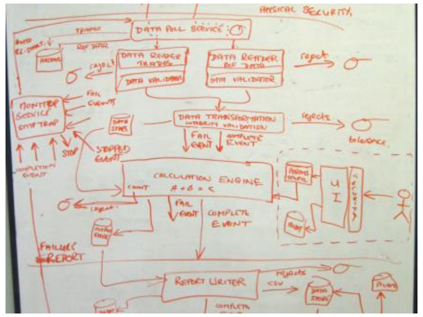
› VISUALLY COMMUNICATING SOFTWARE ARCHITECTURES

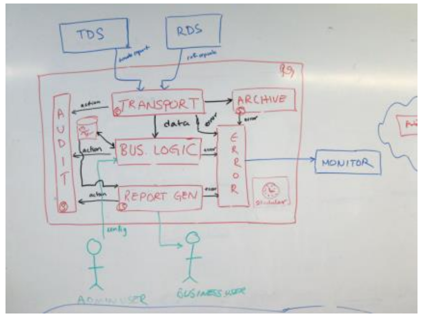
HEILBRONN UNIVERSITY OF APPLIED SCIENCES

THE PROBLEM OF BOXES AND LINES

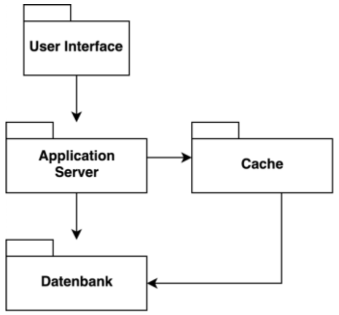


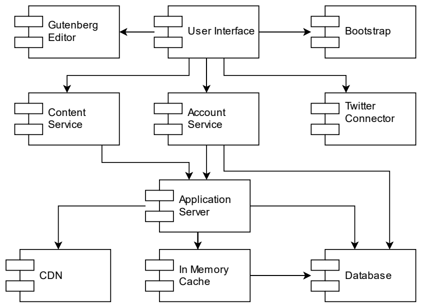


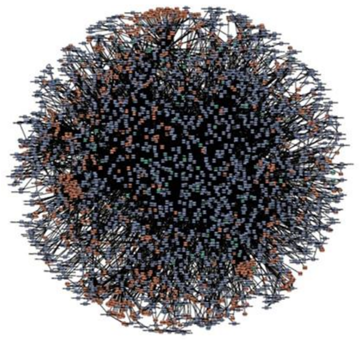


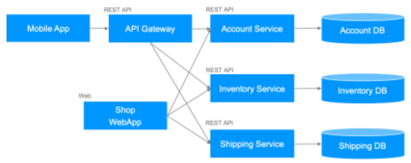


Like this? Or this? Maybe this? Or like this?







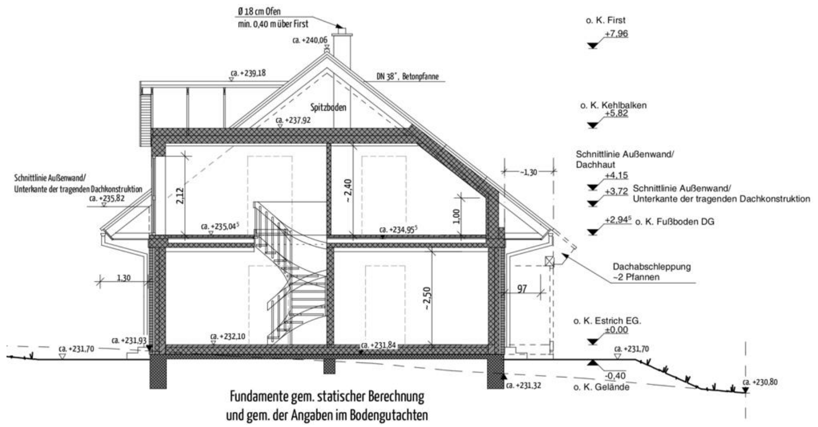


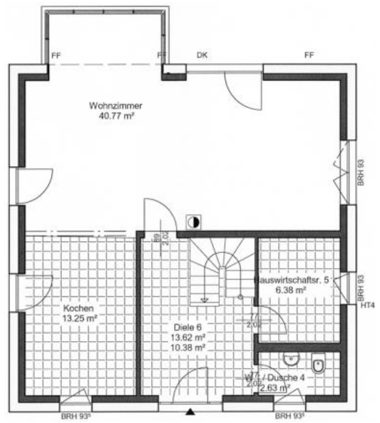
UML Packages? UML Components? Deatchstar Diagrams? Nicely drawn?

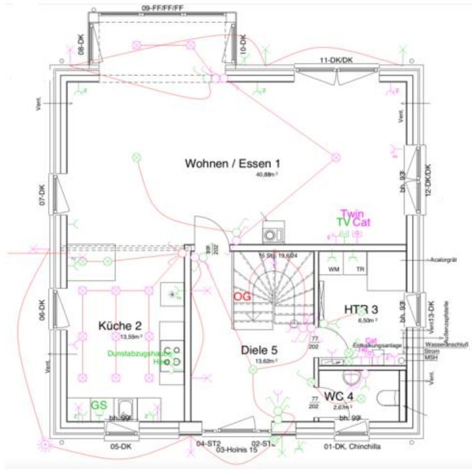
HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

VISUAL COMMUNICATING IN THE CONSTRUCTION INDUSTRY

- Site Plans
- Floor Plans
- Elevation Views
- Cross-Section Views
- Detailed Drawings
- Technical Drawings



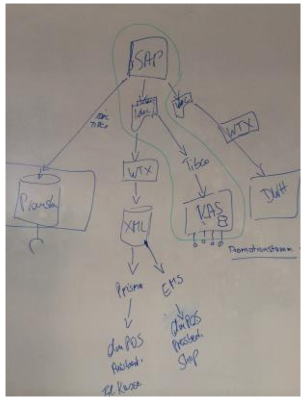
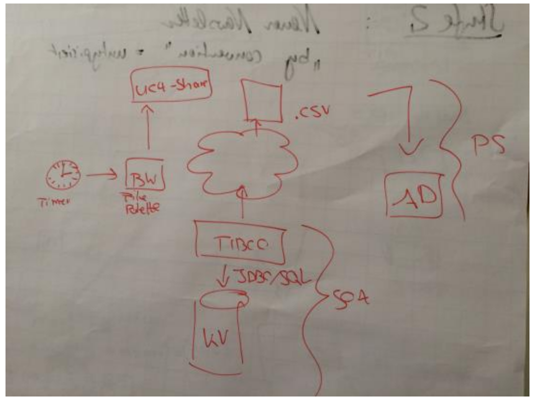
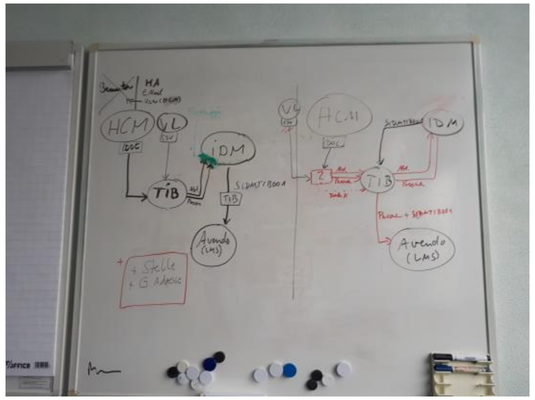
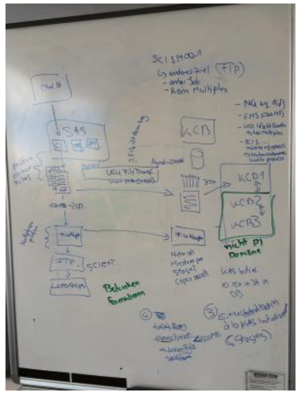




HEILBRONN UNIVERSITY OF APPLIED SCIENCES

VISUAL COMMUNICATING IN THE SOFTWARE INDUSTRY

- Confused mess of boxes and lines
- Inconsistent notations, colours, shapes and line styles
- Ambiguous naming
- Unlabelled Relationships
- Missing technology choices and details
- Mixed levels of abstraction

REASONS

> Side effects due to the shift towards agile development
- Less up-front design
- Less software diagrams in general

> If diagrams are used the are often
- vague
- ambiguous
- confusing
- provide mixed levels of details / abstractions

› C4 MODEL

CODE MAPS

- C4 model is used to **describe** and **communicate** software architectures
- **Abstraction first** approach
- Reflecting how **developers and architects think** about software
- **Up-front design** AND **retrospectively documenting**
- Provides **code maps** on **various levels of detail**

Hochschule Heilbronn – Campus Sontheim
Max-Planck-Straße

Heilbronn
Hochschule Heilbronn – Campus Sontheim

Germany
Hochschule Heilbronn – Campus Sontheim

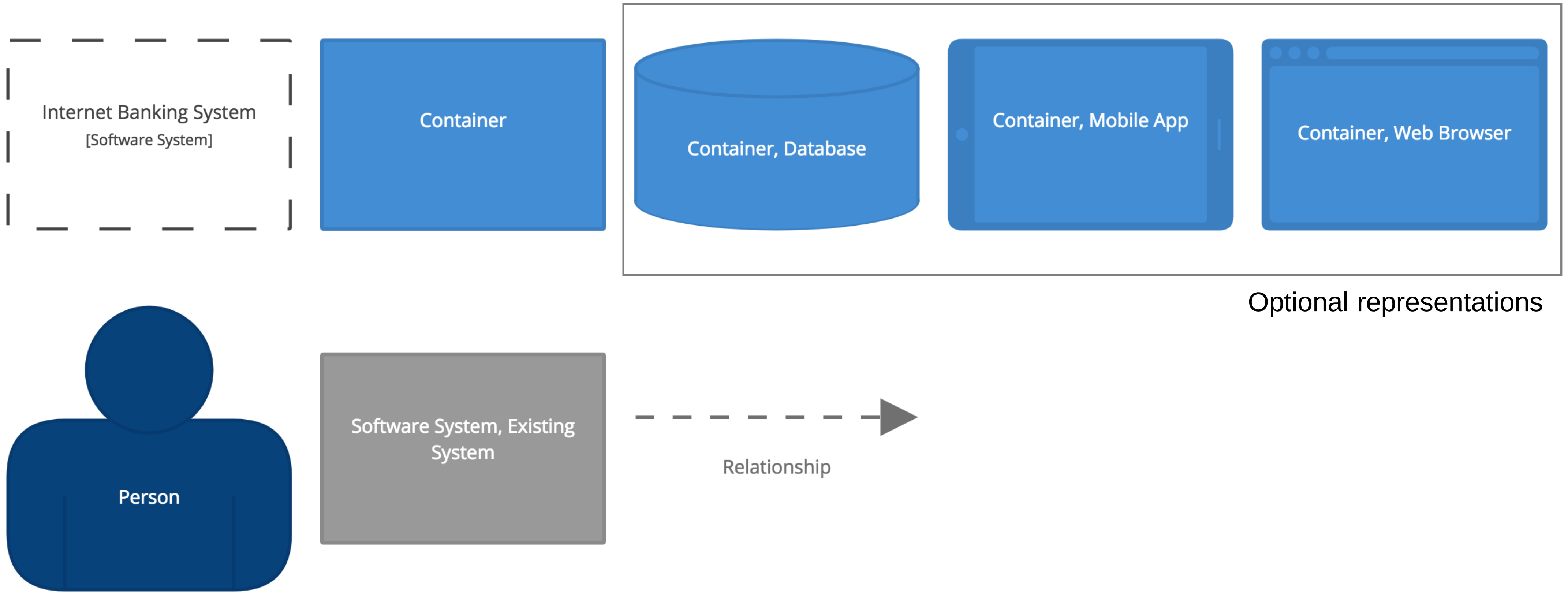
Source Peter Schmelzle - Self-photographed, CC BY-SA 3.0
Source https://maps.google.com

STATIC STRUCTURES OF SOFTWARE SYSTEMS

- Context
  - Big Picture
  - How does it fit
- Containers
  - Something that hosts data or code
- Components
  - Group of related functionality encapsulated behind well-defined interfaces
- Code
  - Classes, interfaces, objects, functions, tables etc. within a component in scope

HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

DIAGRAM ELEMENTS

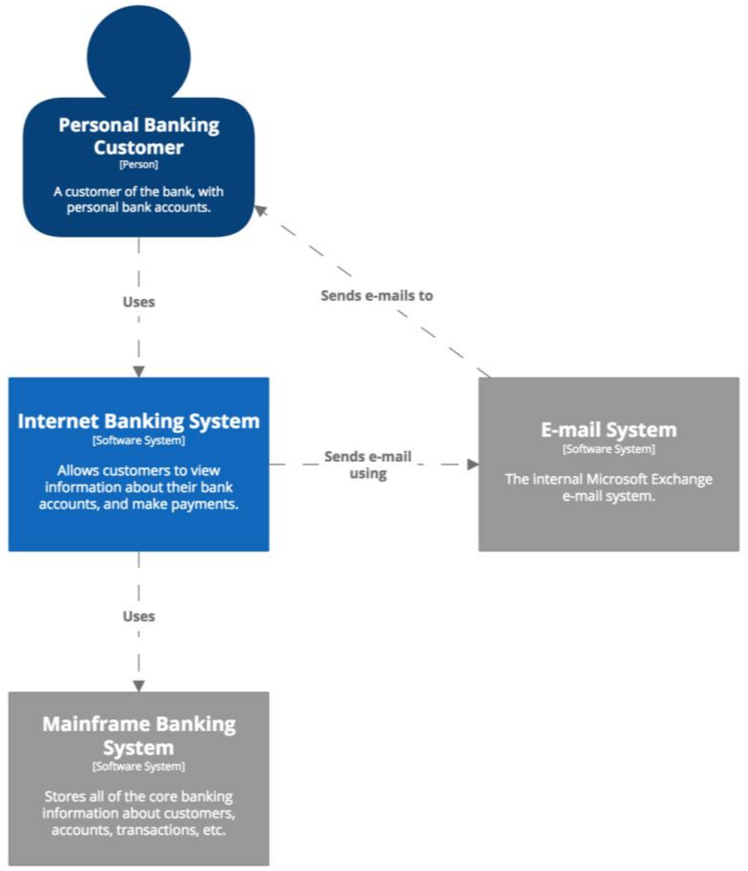


HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

› ABSTRACTION LEVELS

LEVEL 1: SYSTEM CONTEXT DIAGRAM

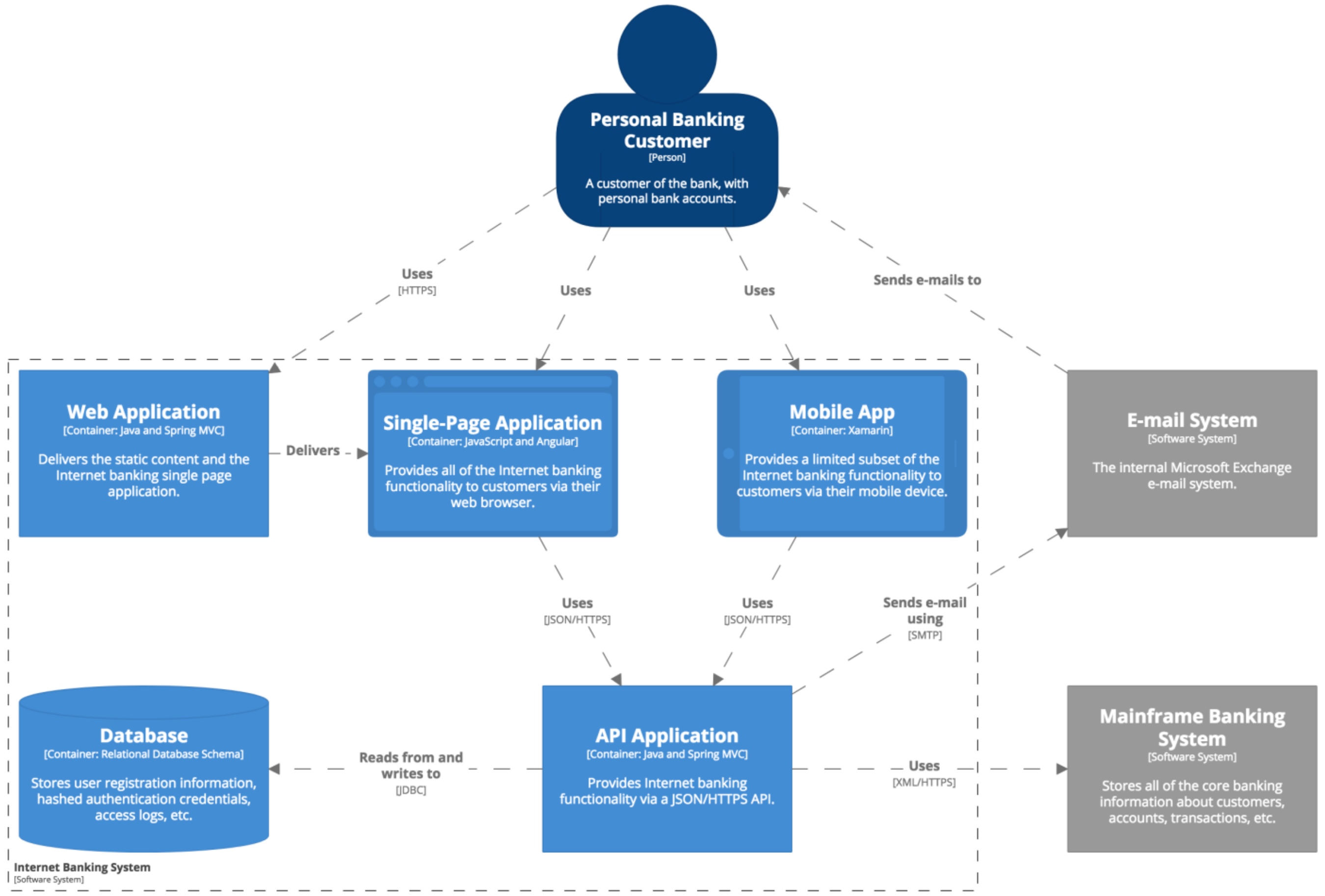
- **Scope**: A single software system
- **Primary elements**: The software system in scope
- **Supporting elements**: People and software systems directly connected to the software system in scope
- **Intended audience**: Everybody, both technical and non-technical people, inside and outside of the software development team


Personal Banking Customer
[Person]
A customer of the bank, with personal bank accounts.
Uses
Sends e-mails to
Internet Banking System
[Software System]
Allows customers to view information about their bank accounts, and make payments.
Sends e-mail using
E-mail System
[Software System]
The internal Microsoft Exchange e-mail system.
Uses
Mainframe Banking System
[Software System]
Stores all of the core banking information about customers, accounts, transactions, etc.

LEVEL 2: CONTAINER DIAGRAM

- **Scope**: A single software system
- **Primary elements**: Containers
  within the software system in scope
- **Supporting elements**: People and
  software systems directly connected to the containers
- **Intended audience**: Technical people inside and outside of the software development team; including software architects, developers and operations/support staff
- **Notes**: This diagram says nothing about deployment scenarios, clustering, replication, failover, etc.

LEVEL 2: CONTAINER DIAGRAM


Personal Banking Customer
[Person]
A customer of the bank, with personal bank accounts.
Uses
[HTTPS]
Uses
Uses
Sends e-mails to
Web Application
[Container: Java and Spring MVC]
Delivers the static content and the Internet banking single page application.
Delivers
Single-Page Application
[Container: JavaScript and Angular]
Provides all of the Internet banking functionality to customers via their web browser.
Mobile App
[Container: Xamarin]
Provides a limited subset of the Internet banking functionality to customers via their mobile device.
E-mail System
[Software System]
The internal Microsoft Exchange e-mail system.
Uses
[JSON/HTTPS]
Uses
[JSON/HTTPS]
Sends e-mail using
[SMTP]
Database
[Container: Relational Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.
Reads from and writes to
[JDBC]
API Application
[Container: Java and Spring MVC]
Provides Internet banking functionality via a JSON/HTTPS API.
Uses
[XML/HTTPS]
Mainframe Banking System
[Software System]
Stores all of the core banking information about customers, accounts, transactions, etc.
Internet Banking System
[Software System]

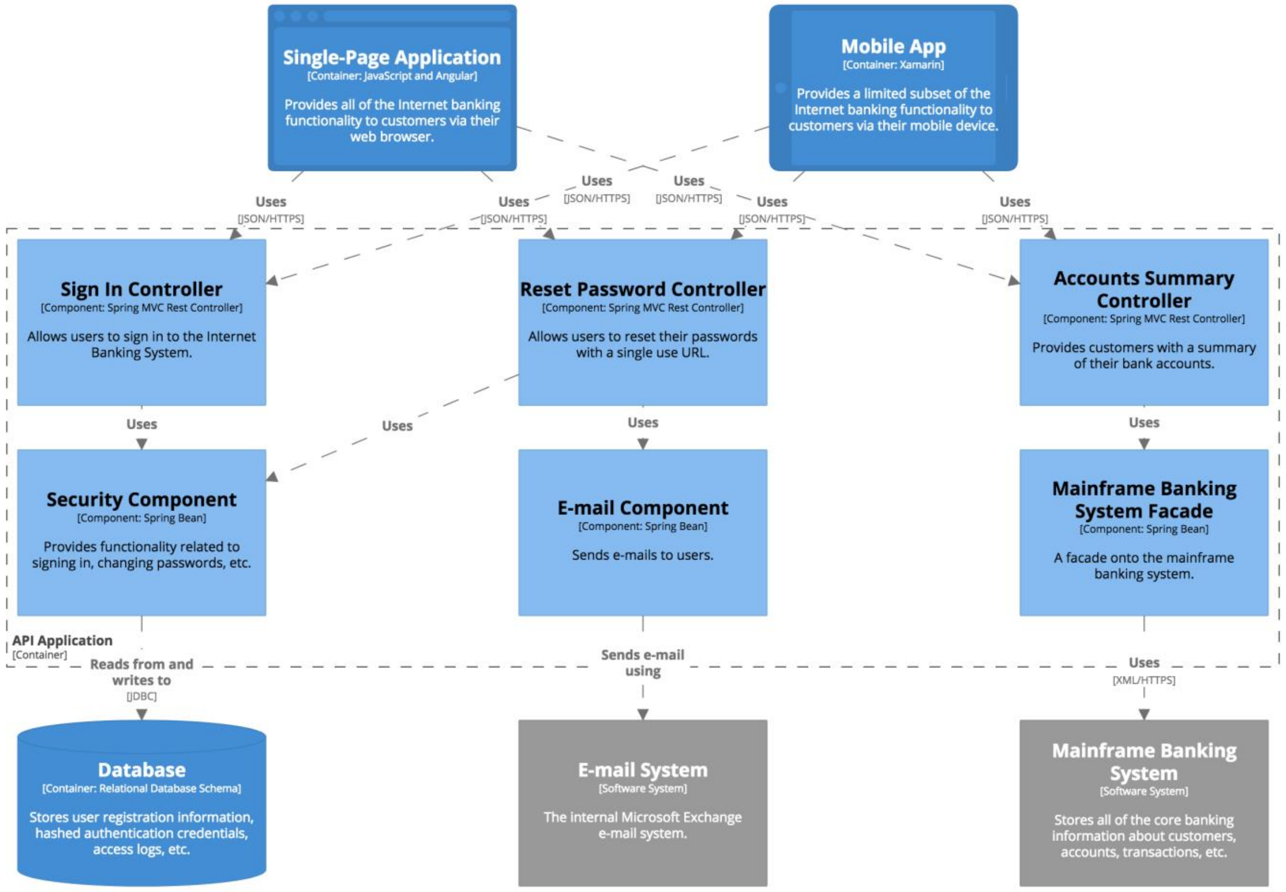
HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

LEVEL 3: COMPONENT DIAGRAM

- **Scope**: A single container
- **Primary elements**: Components within the container in scope
- **Supporting elements**: Containers (within the software system in scope) plus people and software systems directly connected to the components
- **Intended audience**: Software architects and developers

HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

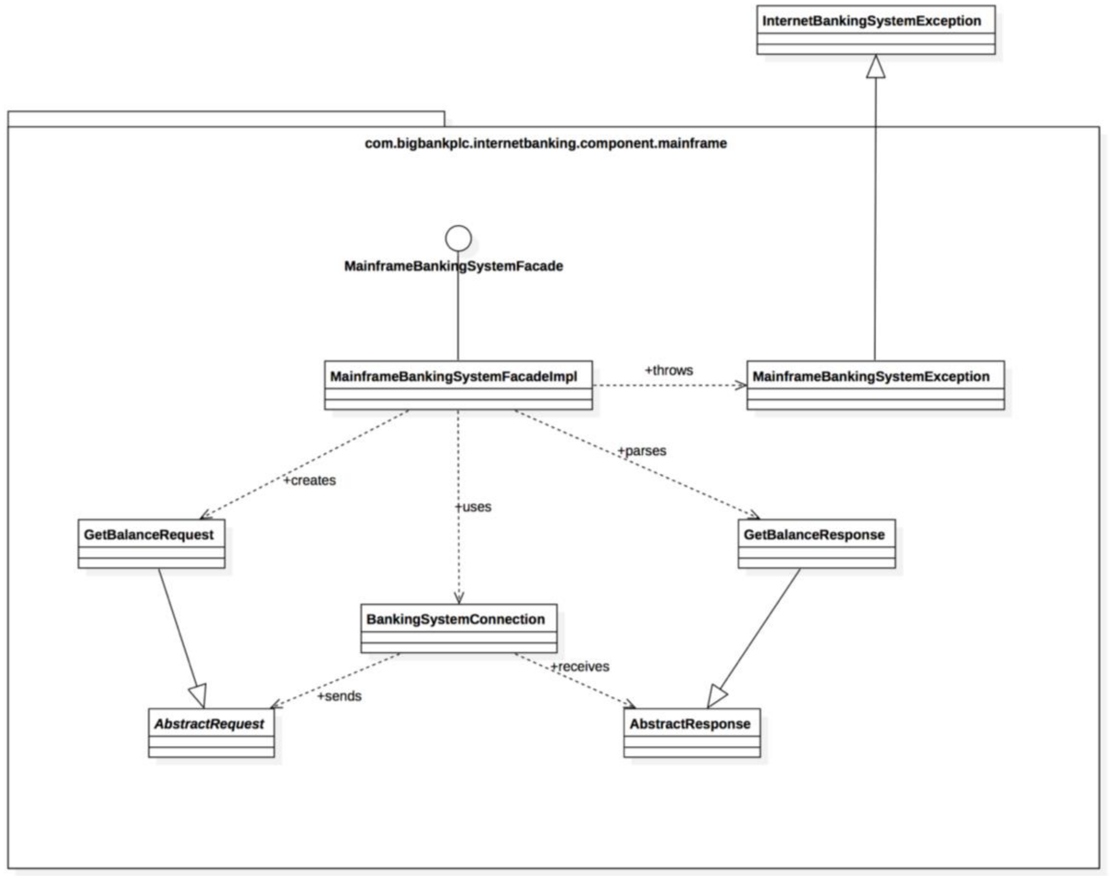
LEVEL 3: COMPONENT DIAGRAM


Single-Page Application
[Container: JavaScript and Angular]
Provides all of the Internet banking functionality to customers via their web browser.
Mobile App
[Container: Xamarin]
Provides a limited subset of the Internet banking functionality to customers via their mobile device.
Uses [JSON/HTTPS]
Uses [JSON/HTTPS]
Uses [JSON/HTTPS]
Uses [JSON/HTTPS]
Uses [JSON/HTTPS]
Uses [JSON/HTTPS]
Sign In Controller
[Component: Spring MVC Rest Controller]
Allows users to sign in to the Internet Banking System.
Reset Password Controller
[Component: Spring MVC Rest Controller]
Allows users to reset their passwords with a single use URL.
Accounts Summary Controller
[Component: Spring MVC Rest Controller]
Provides customers with a summary of their bank accounts.
Uses
Uses
Uses
Uses
Security Component
[Component: Spring Bean]
Provides functionality related to signing in, changing passwords, etc.
E-mail Component
[Component: Spring Bean]
Sends e-mails to users.
Mainframe Banking System Facade
[Component: Spring Bean]
A facade onto the mainframe banking system.
API Application
[Container]
Reads from and writes to [JDBC]
Sends e-mail using
Uses [XML/HTTPS]
Database
[Container: Relational Database Schema]
Stores user registration information, hashed authentication credentials, access logs, etc.
E-mail System
[Software System]
The internal Microsoft Exchange e-mail system.
Mainframe Banking System
[Software System]
Stores all of the core banking information about customers, accounts, transactions, etc.

HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

LEVEL 4: CODE DIAGRAM

- **Scope**: A single component
- **Primary elements**: Code elements (e.g. classes, interfaces, objects, functions, database tables, etc.) within the component in scope.
- **Intended audience**: Software architects and developers


InternetBankingSystemException
com.bigbankplc.internetbanking.component.mainframe
MainframeBankingSystemFacade
MainframeBankingSystemFacadeImpl
+throws
MainframeBankingSystemException
+creates
+uses
+parses
GetBalanceRequest
GetBalanceResponse
BankingSystemConnection
+receives
+sends
AbstractRequest
AbstractResponse

ADDITIONAL DIAGRAMS

> System landscape diagram
  - In the real-world, software systems never live in isolation
  - Shows how software systems work together

> Dynamic diagram
  - UML collaboration diagram
  - UML sequence diagram
  - UML deployment diagram

> UML deployment diagram
  - Incl. containers and deployment nodes

SUMMARY META MODEL – 4 CS

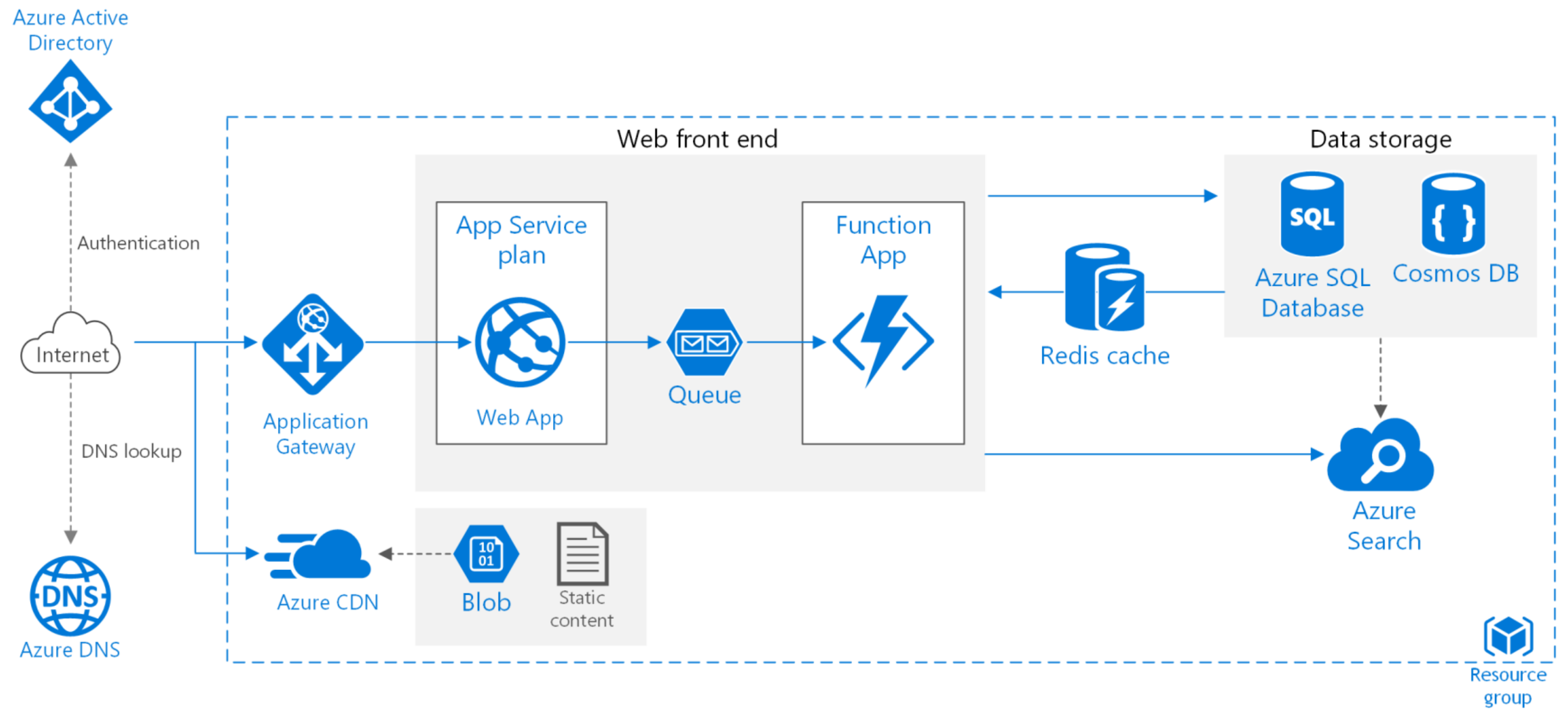
View	Scope	Property
System Context	A single software system	**Primary elements**: The software system in scope. Supporting elements: People (e.g. users, actors, roles, or personas) and software systems (external dependencies) that are directly connected to the software system in scope. Typically these other software systems sit outside the scope or boundary of your own software system, and you don't have responsibility or ownership of them.
Container	A single software system	**Primary elements**: Containers within the software system in scope. **Supporting elements**: People and software systems directly connected to the containers.
Component	A single container	**Primary elements**: Components within the container in scope. **Supporting elements**: Containers (within the software system in scope) plus people and software systems directly connected to the components.
Code	Component	**Primary elements**: Code elements (e.g. classes, interfaces, objects, functions, database tables, etc) within the component in scope.

META MODEL: ELEMENTS AND RELATIONS

Element	Superior Element	Properties
Person	None	• Name* • Description • Location (Internal or External)
Software System	None	• Name* • Description • Location (Internal or External) • The set of containers that make up the software system
Container	A Software System	• Name* • Description • Technology • The set of components within the container
Component	A container	• Name* • Description • Technology • The set of code elements (e.g. classes, interfaces, etc) that the component is implemented by
Code Element	A component	• Name* • Description • Fully qualified type
Relationship**		• Description • Technology

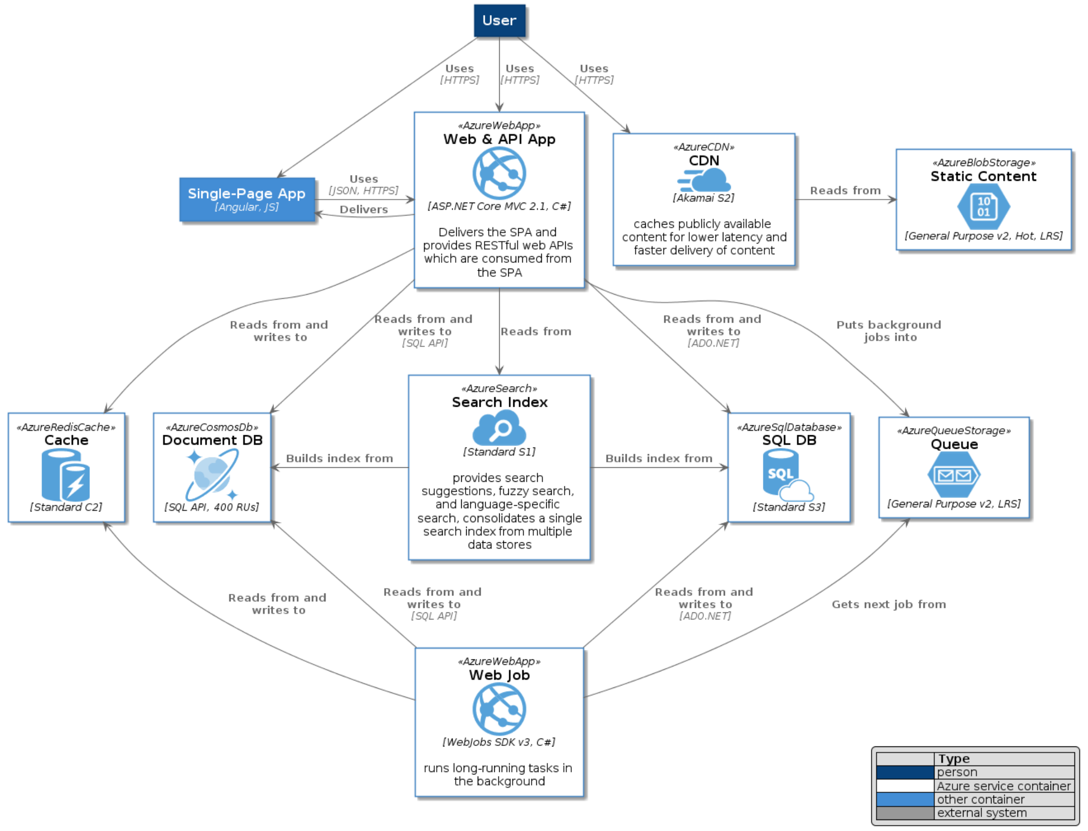
* All elements in the model must have a name, and that name must be unique within the parent context.
** Relationships are permitted between any elements in the model, in either direction.

REAL WORLD EXAMPLE: REFERENCE ARCHITECTURE FOR MICROSOFT AZURE APP SERVICE



Source: https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/app-service-web-app/scalable-web-app

REAL WORDL EXAMPLE: C4 MODEL APPLIED



	Type
■	person
□	Azure service container
■	other container
▨	external system

Source: https://azure-development.com/1918/09/11/save-the-world-from-powerpoint-cloud-solution-architects/

HEILBRONN UNIVERSITY
OF APPLIED SCIENCES

KEY TAKEAWAYS

- The creation of software diagrams has been scaled back as a result of the shift to agile methodologies
- When diagrams are created, they are often confusing and unclear
- The C4 model consists of a hierarchical set of software architecture diagrams for context, containers, components and code
- The hierarchy of the C4 diagrams provides different levels of abstraction, each of which is relevant to a different audience
- Avoid ambiguity in your diagrams by including a sufficient amount of text as well as a key/legend for the notation you use

LITERATURE

Simon Brown

Software Architecture for Developers Volume 2

Leanpub Books, 2018

https://leanpub.com/visualising-software-architecture

Free eBooks are provided to the students of this course

QUESTIONS SO FAR?