Fachhochschule Nordwestschweiz
Hochschule für Technik

Software Architecture

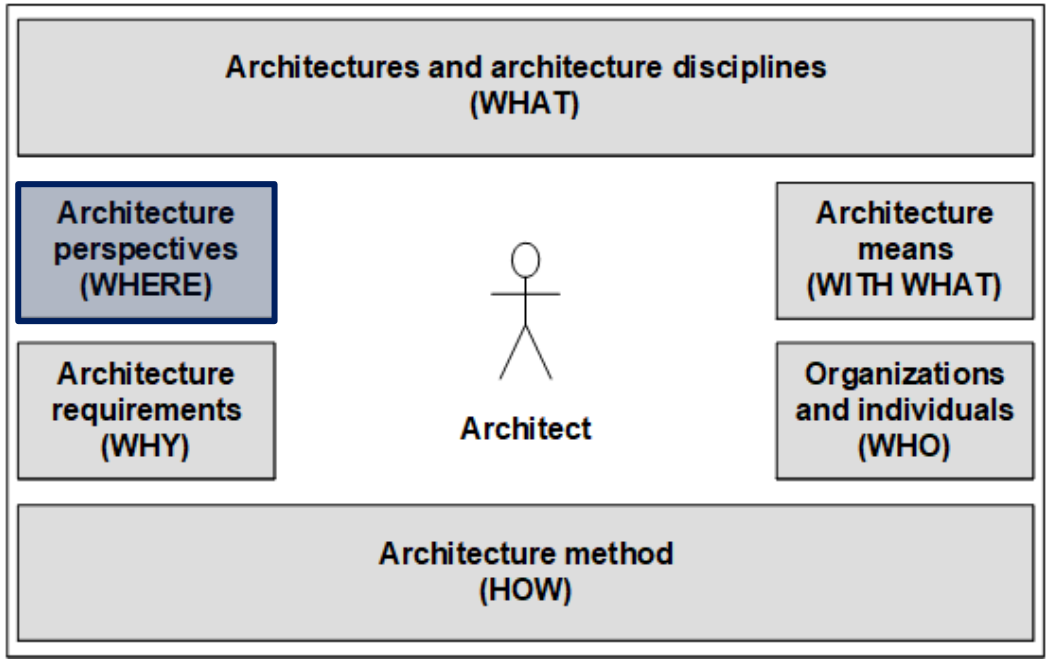
Development View (object-oriented design)

BSc

Ingo Arnold

Lecture Opening
Architecture Orientation Framework

Architecture WHERE deals with the **representation of architecture** in general as well as software architecture in particular [Vogel, Arnold et al 2011].



Lecture Opening
Motivation

A design oriented to the application domain in the paradigm of object orientation provides a blueprint in the **logical view** that respects the LRG principle.

The blueprint **contains objects and represents their cooperation** (i.e., mutual method calls of the objects) to realize the identified use cases. For this dynamic view, one usually uses sequence or communication diagrams.

Provided that a sufficient variety of use case realizations was elaborated, these **blueprints contain an architecture inherently in terms of interacting components** in the logical view.

The interactions between the objects in the form of operation calls are fixed which **limits the degrees of freedom in the realization of the individual objects**.

For the conversion into executable software one must now still produce the necessary programs (e.g., classes) for the individual objects – this is a job for Kruchten's development view.

Lecture Opening
Learning Objectives

You …

- can derive (e.g., from a communication diagram or a sequence diagram) which and how classes must be designed, and what their externally visible elements look like.
- can implement these classes in such a way that they jointly realize the scenarios on which their design is based.
- can take advantage of dividing the responsibility and necessary work on individual classes in order to delegate non-architecture design and to accelerate programming by working in parallel.

Lecture Agenda

- Development View

Development View
Overview

Model-, domain- and object-oriented design methods aim at a LRG between the well recognizable structures of the real world and those of the software.

Depending on the complexity of the task, you can proceed in one direct or two successive steps. In the latter case, you start with the modelling of the real world in the course of object-oriented analysis (OOA) as a basis on which an object-oriented design (OOD) is developed.

In contrast to analysis, objects are taken into account in the design, which have no direct counterpart in the real world (and are therefore missing in the analysis), while nevertheless their implementation is considered necessary. Examples are technical or administrative concerns.

A goal of object-oriented analysis and design is to understand which objects produce the desired effect and how these objects respectively cooperate (logical view).

Development View
Overview

Such design consists of **static structure models** which indicate the required objects (and their classes) and their relationships. Second, it consists of **behaviouristic models** that show how these objects work together to realize the required scenarios.

The **object-oriented paradigm connects logical and development view via the concept of classes**. The algorithmic program logic is described with objects.

However, the **artifacts to be programmed are the classes** based on which objects are instantiated.

Methods an object calls on another or messages an object sends to another, must be publicly available and implemented by classes.

Relationships between objects require stored references (e.g., in instance variables).

Development View
Overview

For all simple objects which present themselves identically to the outside, a common class is sufficient.

For more **complex objects, on the other hand, a whole series of further objects and thus classes may be required** behind the façade of a generating class. However, these are not part of the architecture-relevant design.

Common aspects of classes for different objects can possibly be **delegated to super classes**, which are then differently extended.

Design decisions in the development view often reflect what you anticipate as future changes to the solution. Also, you may introduce components to be able to utilize polymorphism. Usually in such cases you need to define a suitable type hierarchy.

Development View
Overview

For the systematic development of an **object-oriented design in the development view, the steps below are performed**.

1. You create a list of all classes to be programmed, so that ...
   - each object can be instantiated by exactly one class
   - identically behaving objects are instantiated by a common class
   - each class instantiates at least one of the required objects
2. Each arrow arriving at an object (e.g., in a sequence diagram) represents the activation of a method. For these you provide a callable (public) method in the corresponding class. The result of this step are complete method signatures including *return value* and *parameters* as well as additional *information about method behaviour*. This means at this stage you also consider which information is passed between objects.
3. Classes defined in such a way can be independently programmed, before they are combined to shape the algorithmic structures realizing identified use cases.

Development View
Exercise

*i.3-BSc_SWA_GamePlatform
i.4-BSc_SWA_GamePlatform

Questions