***Lecture from 18.10.2018***

***Introduction to software architecture***

Here we already have a prototype of a software product and we are thinking how to implement and design the whole application.

***Roadmap***

* Definition of SA
  + Definition
  + Perspectives, View
  + Topology

***The functionality doesn’t define the software architecture.***

***Software architecture =! Software functionality.***

An example is the situation with skype and the telephone. Same functionality but different implementation and architecture.

***Software architecture (SA)*** *– is defined as the “set of structures” needed to reason about a software system, which comprise software elements, relationships among them and properties of both.*

Why we use SA

* Eases communications among stakeholders
* Defines constraints on implementations
* Helps to satisfy the system quality attributes
* Makes it easier to reason and manage change
* Enables more accurate cost and schedule

***Perspectives and views***

***Static*** – how is it structured as a set of code units and module views (UML diagram) – What

***Dynamic*** - happens during an execution

***Allocation, Deployment perspective -*** how does it relate to other than software structures; like what servers would we use

***Components and Connectors view***

Components: run-time computational and data store elements

Connectors: interaction mechanism that represents the communication between components

Components use their ports to communication between components.

***Topology***

The architecture elements, their interconnections and the constraints concerning them compose the topology of a system.

The topology of the system can be represented as a graph of interconnected components and connectors.

When this graph changes during runtime the architecture is referred to as ***dynamic***.

***Quality attributes***

* Availability
* Reliability
* Usability
* Performance
* Scalability
* Safety
* Modifiability
* Compatibility

***Effects of a quality attributes***

Results/Conclusion: you should decide which quality attribute is important for you. When you decide to use m

***Quality attribute – Taxonomy***

Design Qualities

* Maintainability
* Reusability

Run-time qualities;

* Performance

…

Quality Attributes Scenarios

* The definitions provided for an attribute are not operational

1. A system will be modifiable.

* Which quality a particular aspect belongs to

1. Is a system failure an aspect of availability?

* Each attribute community has developed its own vocabulary

***Quality attributes and scenarios***

1. Source of stimulus
2. Stimulus
3. Environment
4. Artifact
5. Response
6. Response measure

*Scenarios are made to test how much our system resolves the different quality attributes. Scenarios are tests for quality attributes.*

You will always have to make trade-offs for the different quality attributes.

Architecture Drivers

* Functional requirements.
* Business constraints.

***Domain Specific Languages (DSLs)***

Other way to represent software architectures are DSLs.

***DSLs are computer programming languages of limited expressiveness focused on a particular domain.***

SQL, jQuery, …

There are also graphical and textual languages or description languages. But are not used so much in practice... because every single graphical language comes with its own set of tools how to parse text or visualize something, and their dependencies are not compatible with the other tools.

***ACME***

ACME is a graphical language for software architecture and consists (basically) of:

* Components (computational elements)
* Ports
* Connectors
* Roles
* Systems