Modularization and Software Architectures

UNIVERSITÄT SALZBURG

Overview of SEI architectural styles

Architectural style	**Characteristics**
Data-centered	Repository Architecture Blackboard Architecture
Data-flow	Batch/Sequential Architecture Pipes&Filters Architecture
Call & Return	Top-Down Architecture Network Architecture (Object oriented) Layered Architecture
Virtual Machine	Interpreter Architecture Rule-based Architecture
Independent Components	Event-driven Architecture

UNIVERSITÄT SALZBURG

Call & Return (I)

- The Call&Return style is used for describing typical control flow for imperative programming: Procedures, functions and methods of a module are called from other modules and upon finishing their execution the control jumps back to immediately after the calling place.

- **Top-down form of the Call&Return architectural style:** In conventional, not object-oriented implementations, the Call&Return leads to a top-down-oriented architecture, i.e. a (main or root) procedure/ function/method calls further procedures/functions/ methods, etc.

UNIVERSITÄT SALZBURG

Call & Return (II)

- **In network and/or object-oriented implementations of the Call&Return architectural style:**

  The constructs available in object-oriented languages allow the formation of network-oriented architectures alongside with the hierarchical structuring of top-down-oriented architectures. Method calls take place in a network of objects.

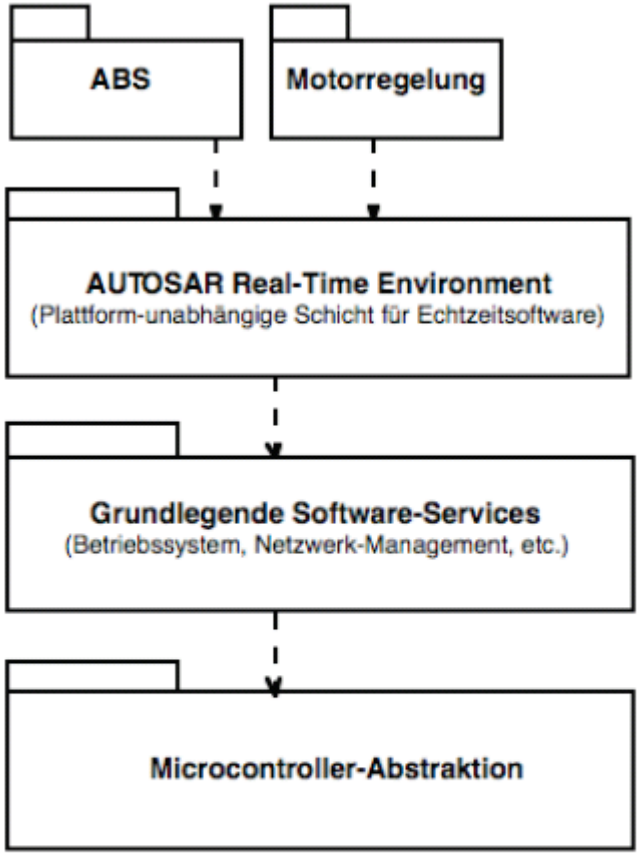
UNIVERSITÄT SALZBURG

Call & Return (III)

- A further form of the Call&Return architectural style is the so-called **layered architecture**, which is used in order to introduce abstractions.
- A layer corresponds to a module which offers a certain functionality and which can contain a number of other modules.
- Each layer exposes an interface to be used by the above layer and uses an interface implemented by the layer below.

UNIVERSITÄT SALZBURG

Example: Layer architecture of AUTOSAR (Automotive Open Systems Architecture)



Legende:

Schicht (UML Package)

Richtung, in die der Zugriff erlaubt ist

more information at
www.autosar.org

UNIVERSITÄT SALZBURG

Advantages of the layered architecture

- Abstractization
- Ease of change: a layer affects at most two adjacent layers
- Information hiding
- Standardized layer interfaces for libraries and frameworks

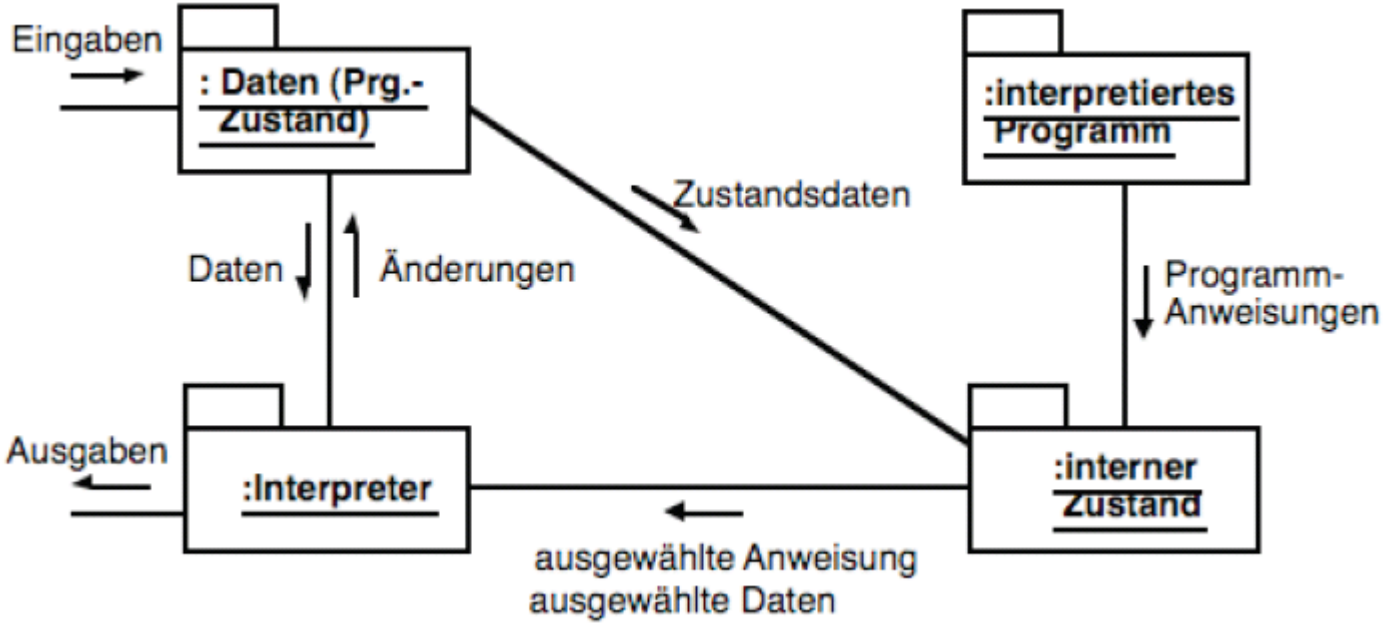
UNIVERSITÄT SALZBURG

Issues to be addressed in layered architectures

- Determining the right abstractization level
- Avoiding multi-layer crossing
- Performance

UNIVERSITÄT SALZBURG

Virtual Machine (I)



- A virtual machine serves for providing functionality which is needed for the execution of an application, without making available details specific to the hardware and/or system software on which the application may run.
- **Portability** is improved by using this architectural style.

UNIVERSITÄT SALZBURG

Virtual Machine (II)

- **Interpreter architecture:** The concept of the virtual machine has strengthened since the introduction of Java, becoming more commonly used. The fact that software portability can be improved by using virtual machines was already shown at the beginning of the 70s by the definition of the P-Code (Pascal code). Thus Pascal compilers, which produced P-Code instead of machine code, became portable. A virtual machine (which interpreted the P-Code), had to be made available for each hardware platform on which the compiled code had to be executed.
- **Rule-based architecture:** Expert systems, which work by interpreting rules (mostly of logical inference).

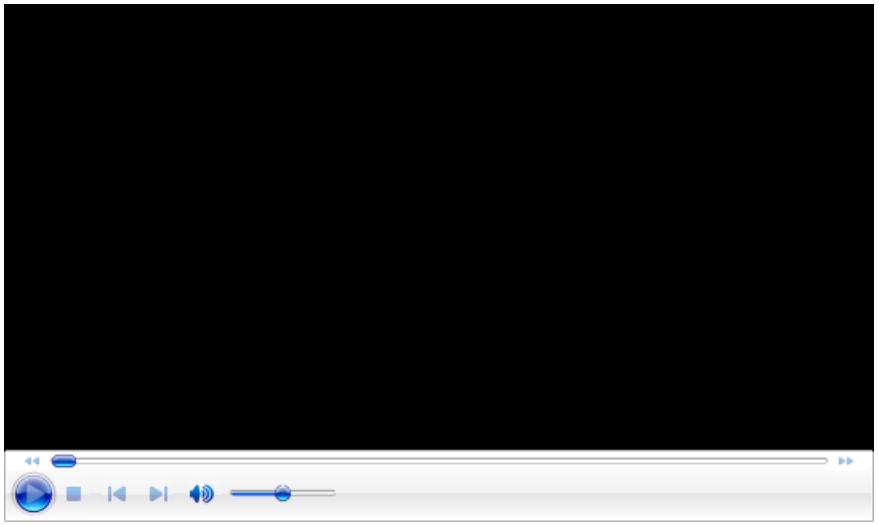
UNIVERSITÄT SALZBURG

Independent Components (I)

- This architectural style postulates loose coupling between independent components. We call components independent, if the components do not directly call functions/procedures/methods of other components.
- A usual way to loosely connect independent components is by so-called **event-oriented linkages**: A component registers itself with another component, from which it wants to be informed about changes. When a change (event) occurs, a component informs all its registered components. The coupling is loose, because the component only informs other components about a change, without specifying what they have to do. They may or may not react to the change.

UNIVERSITÄT SALZBURG

Independent Components (II)



The video-clip component registers itself with the buttons of the controller component, to be informed about occurring events. When pressing the “Play” button, for example, the video-clip component is informed by the button with the message that the event „pressed“ occurred. The video-clip component can react now to this event by playing the clip.

UNIVERSITÄT SALZBURG