Architecture Modeling
in
embedded systems

Ákos Horváth

Model Driven Software Development

Lecture 11

Budapest University of Technology and Economics
Department of Measurement and Information Systems

Abstract

- ”The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.”

- Software Architecture in Practice,
  Bass, Clements, and Kazman

General Concepts

Overview

- **First and foremost**: no universal agreement on what ADLs should represent
- Typically formal representation of architecture
- Human and machine readable
- Describes the system at a higher level
- Enables analysis on consistency, completeness, etc.

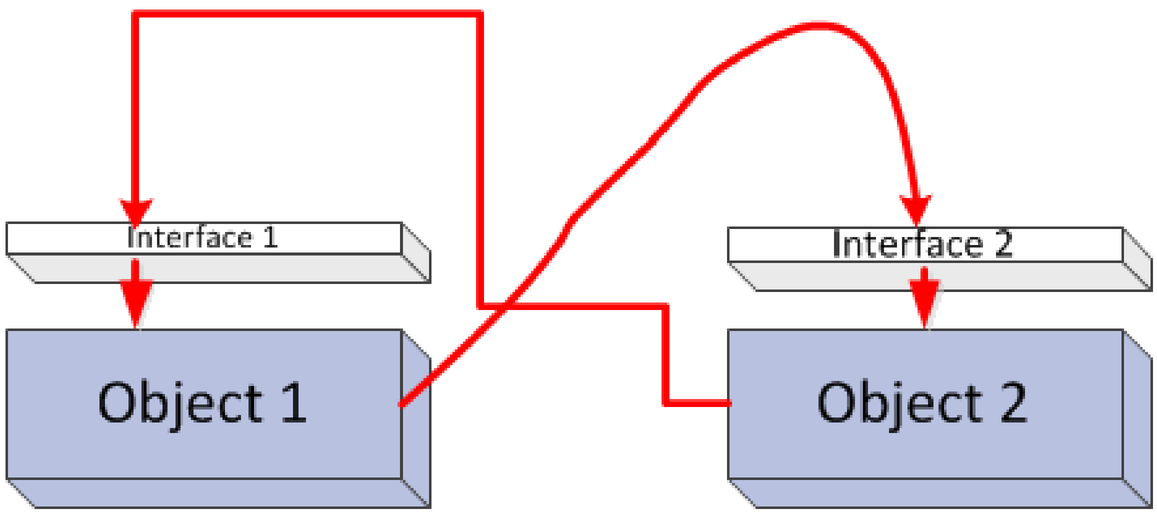
Design vs. Architecture

- Design
  - Functional requirements are addressed

- Architecture
  - Highest level of system description
  - Functional requirements are partitioned
  - Non-functional requirements are addressed
  - Typical Strategies
    - Layering
    - Diagnostics
    - Performance control and monitoring
    - COTS / reuse
    - GUI driven, API driven, etc.

Common Concept of Architecture (by Tw Cook)

- Object Connection Architecture
  - Configuration consists of
    - Interfaces: features that must be provided
    - Connections: object → interface (+ call graph)

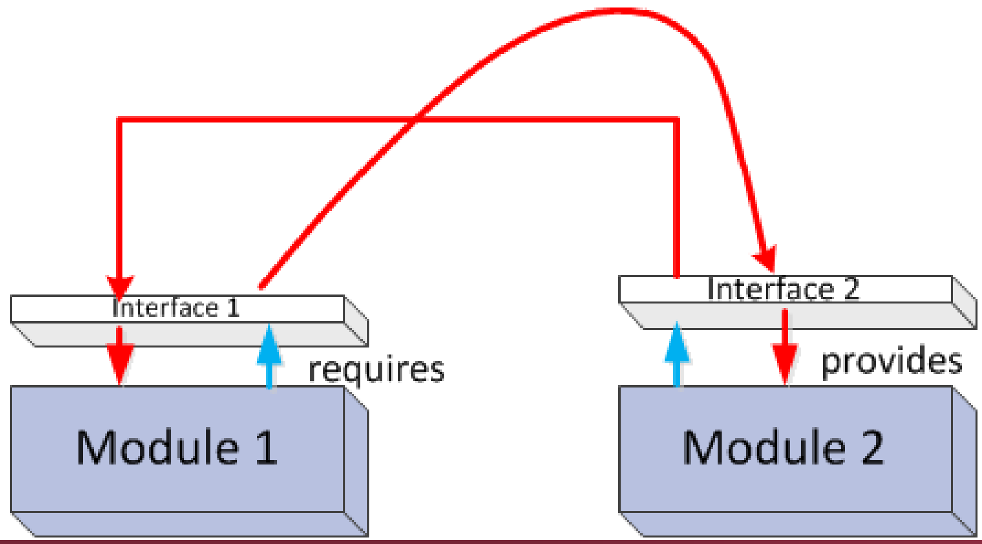


Common Concept of Architecture (by Tw Cook)

- Usually mature languages
  - C++, Java, Ada
- Module must be „built” before architecture is defined
- Conformance of a system to an architecture is low
- Architecture is sensitive to changes in the system

Common Concept of Architecture (by Tw Cook)

- Interface Connection Architecture
- Extends Interface and connection definition
  - Interface: both required and provided features
  - Connections: between required and provided interfaces
  - **Constraints** :
    - restricts behavior of connections and interfaces
    - Architecture constraints → system requirements

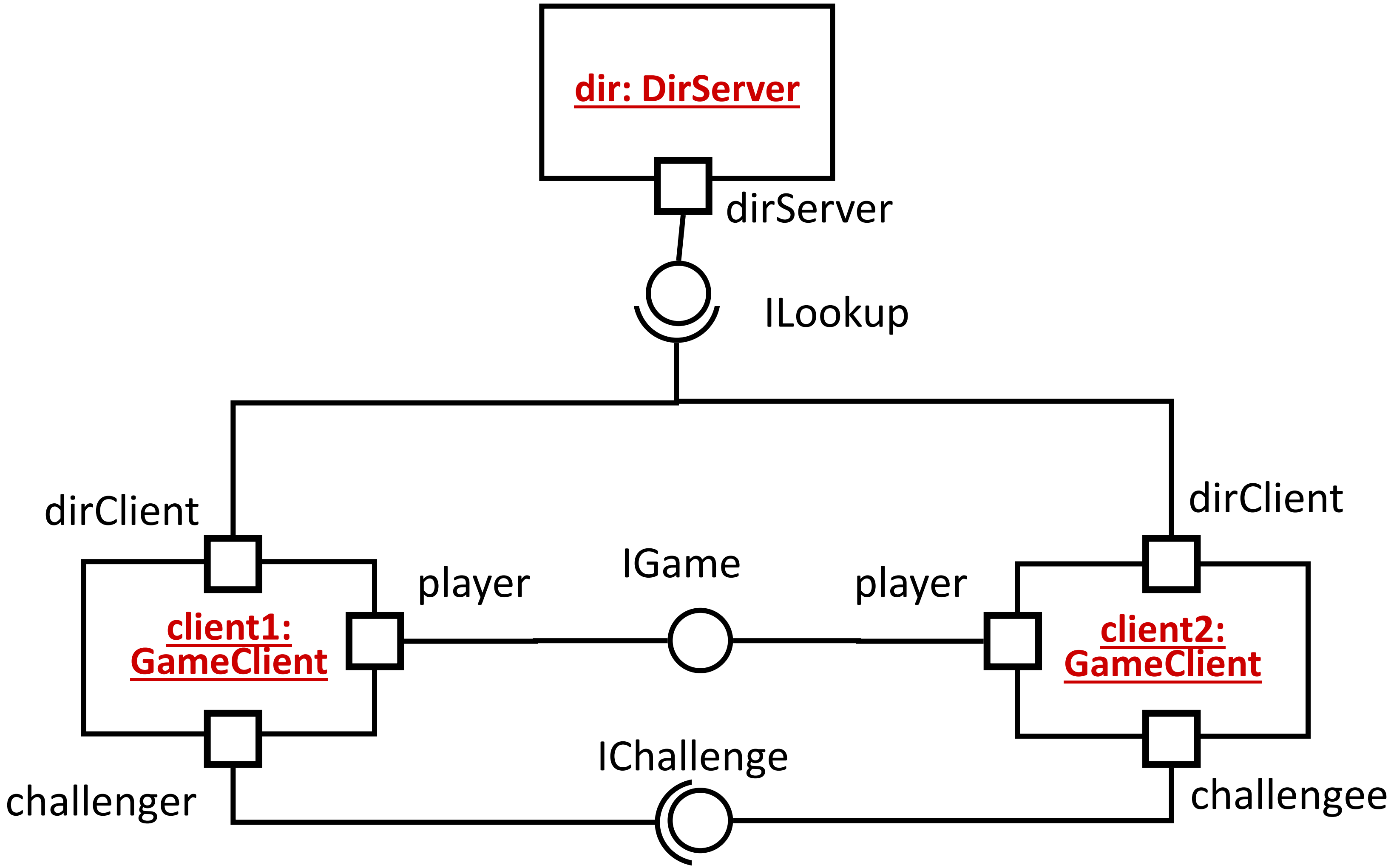


Common Concept of Architecture (by Tw Cook)

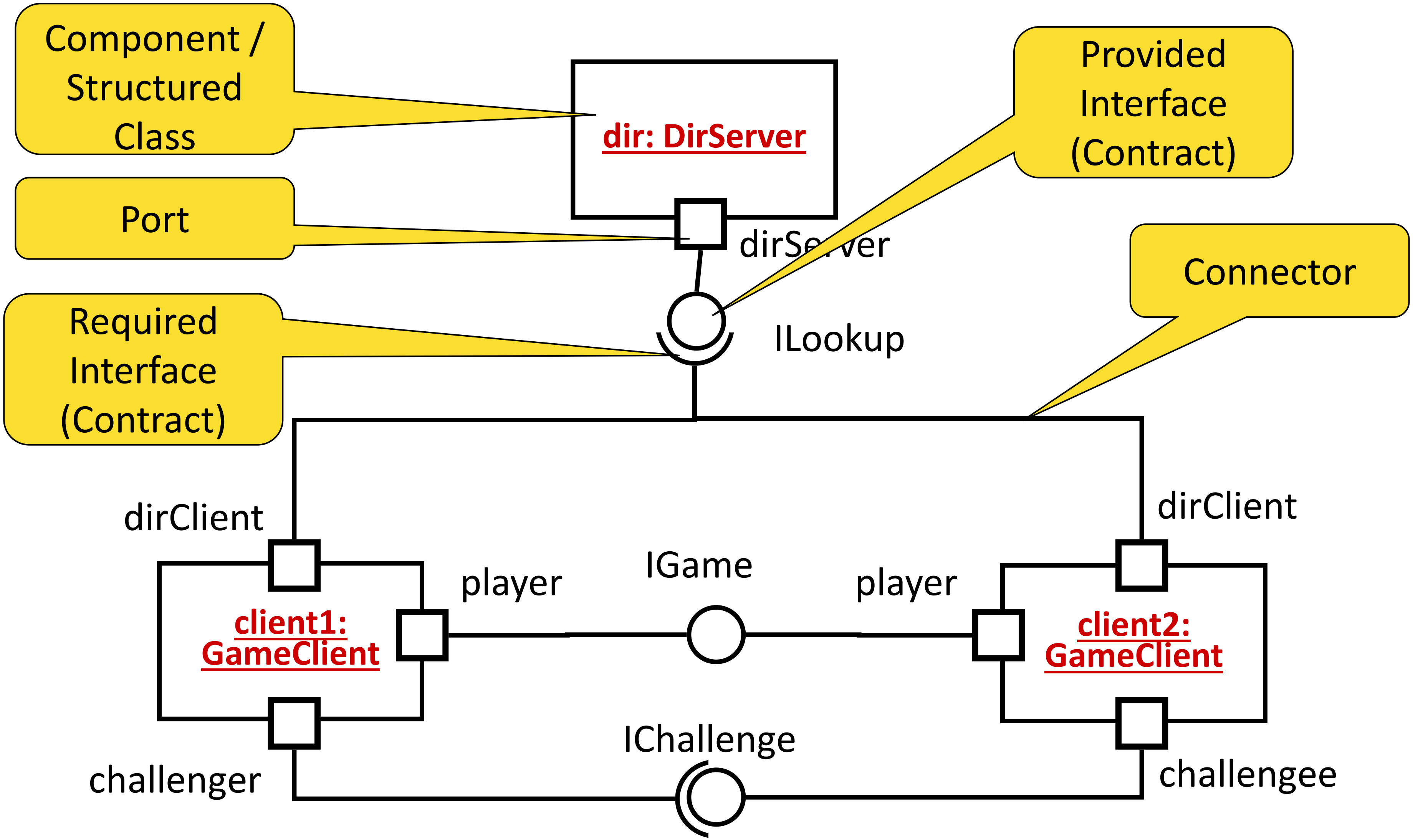
- Better conformance of a system to an architecture
- Architecture can be built before modules are „implemented”
- Most ADL approaches follows this concept

- Similar techniques widely used
  - Design-by-contract
  - Strong partitioning RTOS
  - Etc.

Example: Component Diagram in UML



Example: Component Diagram in UML



Architecture Analysis and Design Language (AADL)

AADL

- Architecture Analysis and Design Language (AADL) is a standard architecture modeling language
  - Avionics
  - Aerospace
  - Automotive
  - Robotics
- Component based notation
  - Task and communication architecture
- Designed for modeling and analysis in mind
- SAE standard (AS 5506A)

- First was called Avionics Architecture Description Language
  - Derived from MetaH created by Honeywell
- V1 version in 2004
- V2 version in 2009

AADL

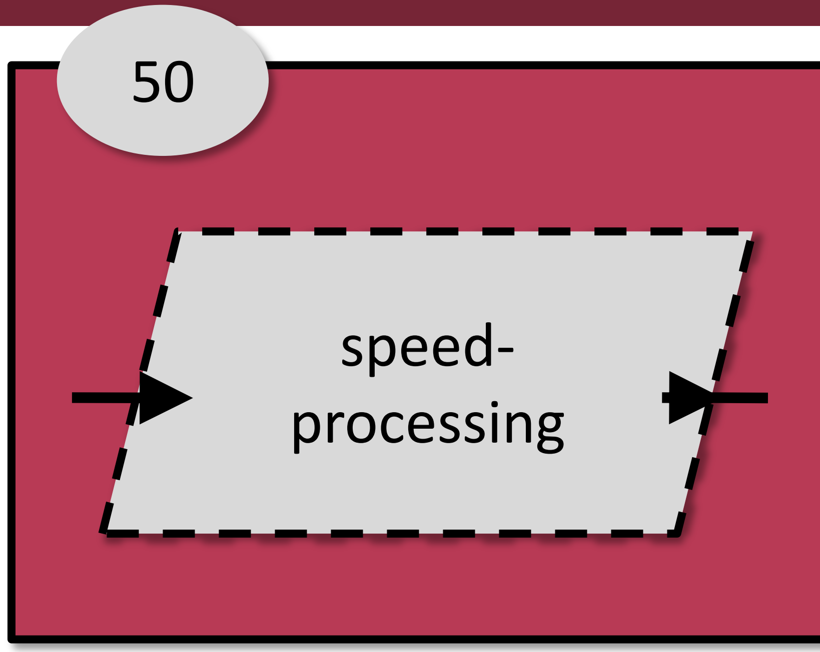
- Based on the component-connector paradigm
- Key Elements:
- Core AADL language standard (V2-Jan,2009, V1-Nov 2004)
  - Textual & graphical, precise semantics, extensile
- AADL Meta model & XMI/XML standard
  - Model interchange & tool interoperability
- Annexes Error Model Annex as standardized extension
  - Error Model Annex addresses fault/reliability modeling, hazard analysis
- UML 2.0 profile for AADL
  - Transition path for UML practitioner community via MARTE
- EMF representation also available (without EFeatureMap!)

AADL

- Precise execution semantics for components
  - Thread, process, data, subprogram, system, processor, memory, bus, device, virtual processor, virtual bus
- Continuous control & event response processing
  - Data and event flow, synchronous call/return, shared access
  - End-to-End flow specifications
- Operational modes & fault tolerant configurations
  - Modes & mode transition
- Modeling of large-scale systems
  - Component variants, layered system modeling, packaging, abstract, prototype, parameterized templates, arrays of components and connection patterns
- Accommodation of diverse analysis needs
  - Extension mechanism, standardized extensions

AADL Representation Forms

```
thread speed_processing
features
        raw_speed_in: in
data port;
        speed_out: out data
port;
properties
        Period => 50 ms;
end data_processing;
```




```
<ownedThreadType name=„speed_processing">
    <ownedDataPort name="raw_speed_in"/>
    <ownedDataPort name="speed_out" direction="out"/>
    <ownedPropertyAssociation property="Period"
        <ownedValue xsi:type="aadl2:IntegerLiteral"
        value="50" unit="ms"
        </ownedValue>
    </ownedPropertyAssociation>
</ownedThreadType>
```

AADL Language Elements

- Core modeling
  - Components
  - Interactions
  - Properties
- Engineering support
  - Abstractions
  - Organization
  - Extensions
- Infrastructure

- Strong modeling capabilities for embedded SW and Computer systems

AADL Components

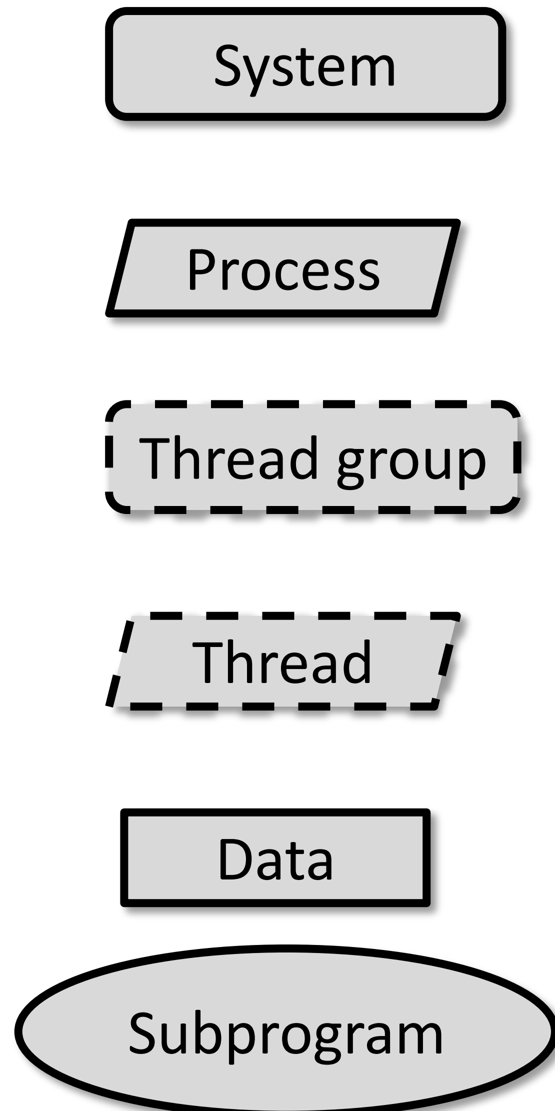
- Top element **system**

Example:

package F22Package
public
system F22System
end F22System;
system WeaponSystem
end WeaponSystem;
system implementation F22System.impl
subcomponents
weapon: **system** WeaponSystem;
end F22System.impl;
end F22Package;

AADL SW Components

- **System** – hierarchical organization of components
- **Process** – protected address space
- **Thread group** – logical organization of threads
- **Thread** – a schedulable unit of concurrent execution
- **Data** – potentially sharable data
- **Subprogram** – callable unit of sequential code



AADL SW Components

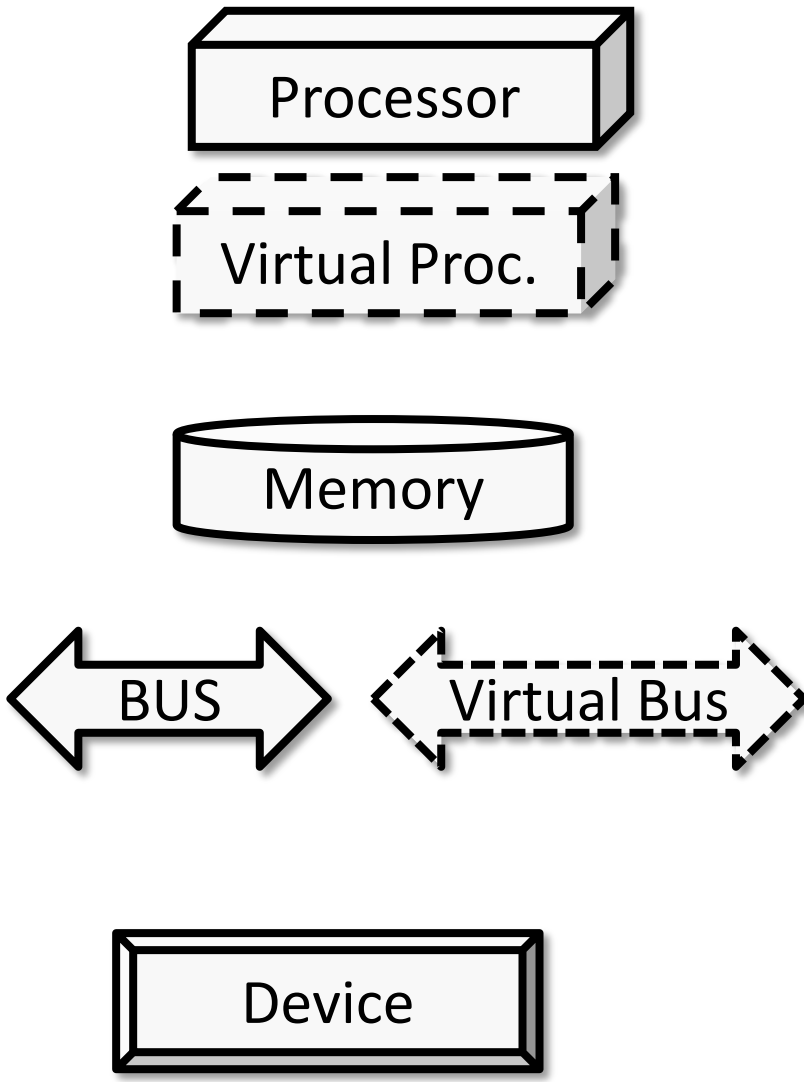
- **Process**
  - Protected virtual address space
  - Contains executable program and data
  - Must contain 1 thread
- **Thread**
  - Concurrent tasks
  - Periodic, aperiodic, sporadic ,background, etc.
  - Interaction through port connection, subprogram calls or shared data access
  - errors: recoverable, unrecoverable

AADL SW Components

- **Ports and Connections**
  - Data (non queued data), Event (queued signals) or Event data (queued messages)
  - Complex Connection hierarchies through components
  - Timing
  - Feature groups
- **Data**
  - Optional but makes the analysis more precise
- **Flows**
  - Logical flow of data and control

AADL Computer Components

- **Processor / Virtual Processor** – Provides thread scheduling and
- **Memory** – provides storage for data and source code
- **Bus / Virtual Bus** – provides physical/logical connectivity between
- **Device** – interface to external environment



AADL Computer Components

- ”Real” HW components
  - Bus transmission time, latency,
  - Processor timing, jitter
  - Memory capacity
  - Etc.
- Logical resources
  - Thread scheduling of a processor
  - Communication protocol overt network connection (modeled as bus)
  - Transactional memory (modeled as memory)

AADL Computer Components

- **Processor**
  - As HW
    - MIPS rating, size, weight, clock, memory manager
  - As Logical resource
    - Schedule threads → scheduling policies and interruption
    - Execute SW
- **Bus**
  - As HW
    - Physical connection inside/between HW components
  - As logical resource
    - Protocol, which are used for the communication
- **Memory**
  - Processes must be in memory
  - Processors need access to memory
- **Device Components**
  - Represents element that are not decomposed further
  - Sensors/Actuators
  - Device Driver

AADL Binding

- Binding
  - Bringing SW models and the execution platform together
  - Virtual processors → can be subcomponents of other virtual processors → ARINC653 partitioning
  - Hierarchical Scheduling
  - virtual buses to physical ones
    - One-to-one
    - Many-to-one

Summary

- After 15 years of mainly DoD research it is getting mature enough
- Many pilot project uses AADL
  - FAA
  - DoD
  - Lockheed Martin
  - Rockwell Collins (Steven P. Miller)
- Many research paper on formal analysis, simulation and code generation
- Ongoing harmonization with SysML and MARTE

AUTOSAR & EAST-ADL

History

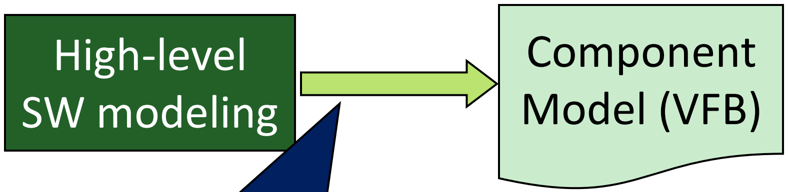
- AUTomotive Open System ARchitecture
- Started in 2002
- BMW, Bosch, Daimler, Conti, VW, + Siemens
- Industrial standardization group
  - Current standard version: 4.0 (end 2009)
  - Currently we use 3.1 (end 2008)
- Members: OEMs, Tool vendors, Semiconductor manufacturers Europe-dominated
- Scope
  - Modeling and implementation of automotive systems
  - Distributed
  - Real-time operating system
  - String interaction with HW and environment
- Out of scope
  - GUI, Java, internet connectivity, File systems, Entertainement systems, USB conncetivity etc.

Key Concepts of AutoSAR

- A standard runtime architecture
  - component-oriented
  - layered
  - extensible
    - New functionalities
    - New components (component implementations)
  - all major interfaces standardized
  - Standardized Run Time Environment (RTE)
- A standard modeling and model interchange approach
  - follows the principles of model-driven design
  - supports the interchange of designs
  - supports the collaborative development
    - Between different developers,
    - Teams,
    - And even companies
- Conformance test framework
  - assuring the conformance to the standard
  - Still evolving – new in version 4.0

High-level design flow

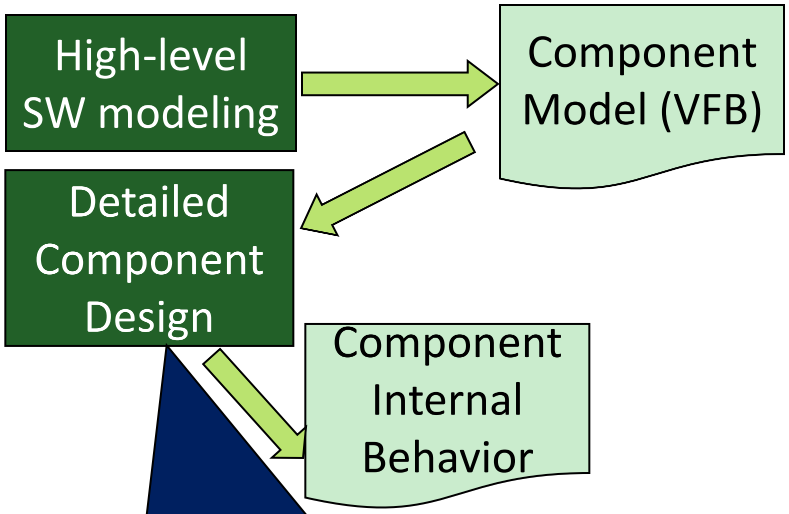
High-level design process



High-level software modeling

- Definition of
  - components
  - component ports
  - port interfaces
  - data types – logical
- Result
  - Virtual Functional Bus (VFB)-level software model

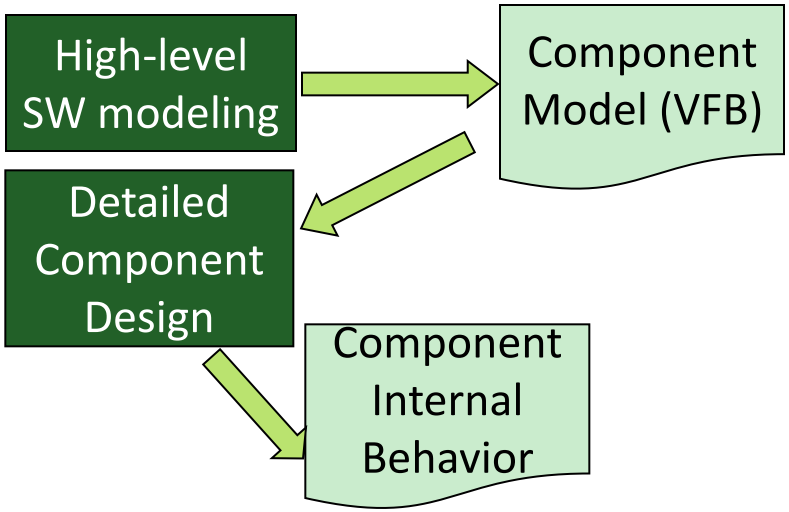
High-level design process

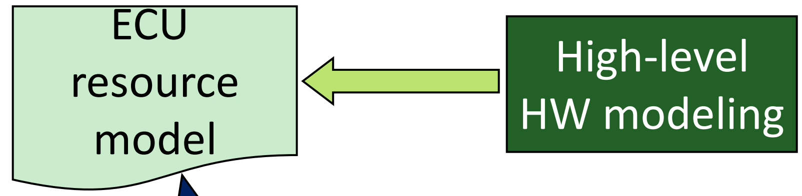


Detailed component design

- Specification of
  - component internal behavior
  - functional breakdown
  - implementation/use of ports
- Non-AutoSAR
  - specification of detailed behavior
  - any tool can be used
    - UML
    - Simulink
    - etc.
- Result
  - AutoSAR component internal behavior model
  - Non-AR: behavioral models/design

High-level design process

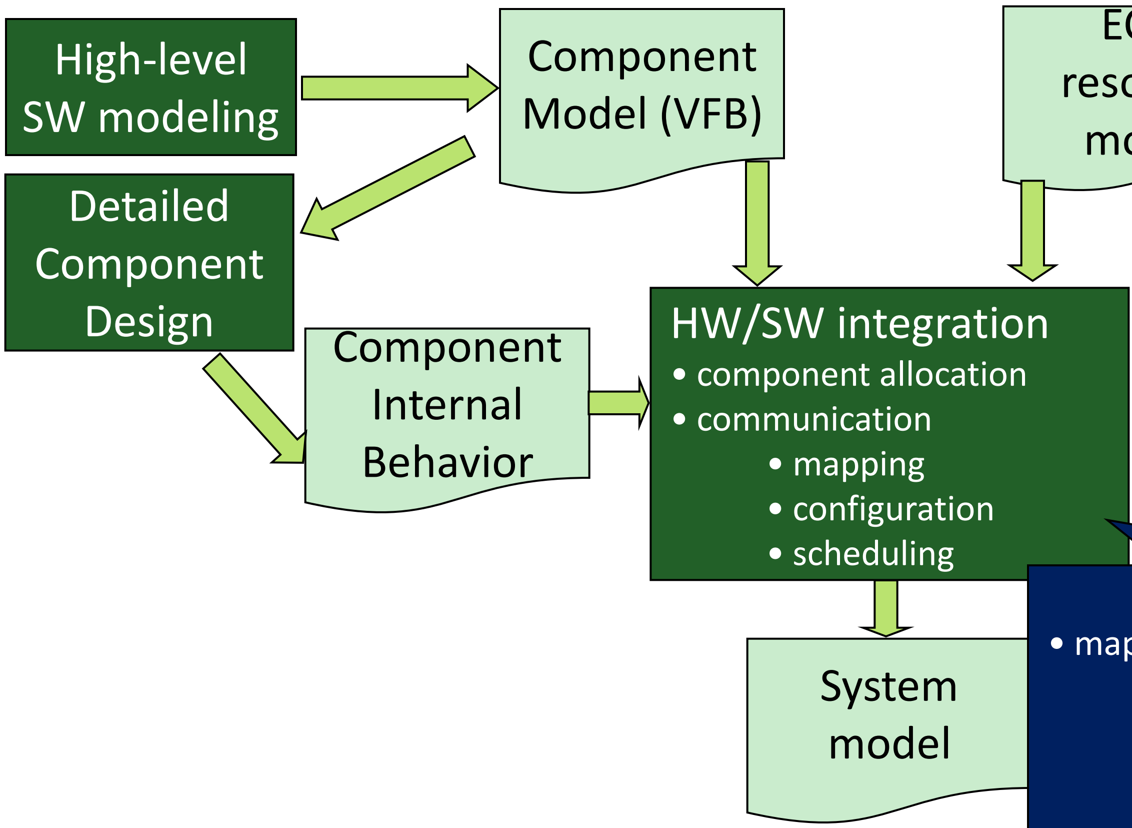




High-level hardware modeling

- Specification of
  - Electronic Control Unit (ECU) resources
    - CPU
    - memories
    - peripherals
    - communication hw
  - system topology
    - ECU instances
    - clusters
    - connections
- Result
  - ECU resource model – for all ECUs
  - System topology model

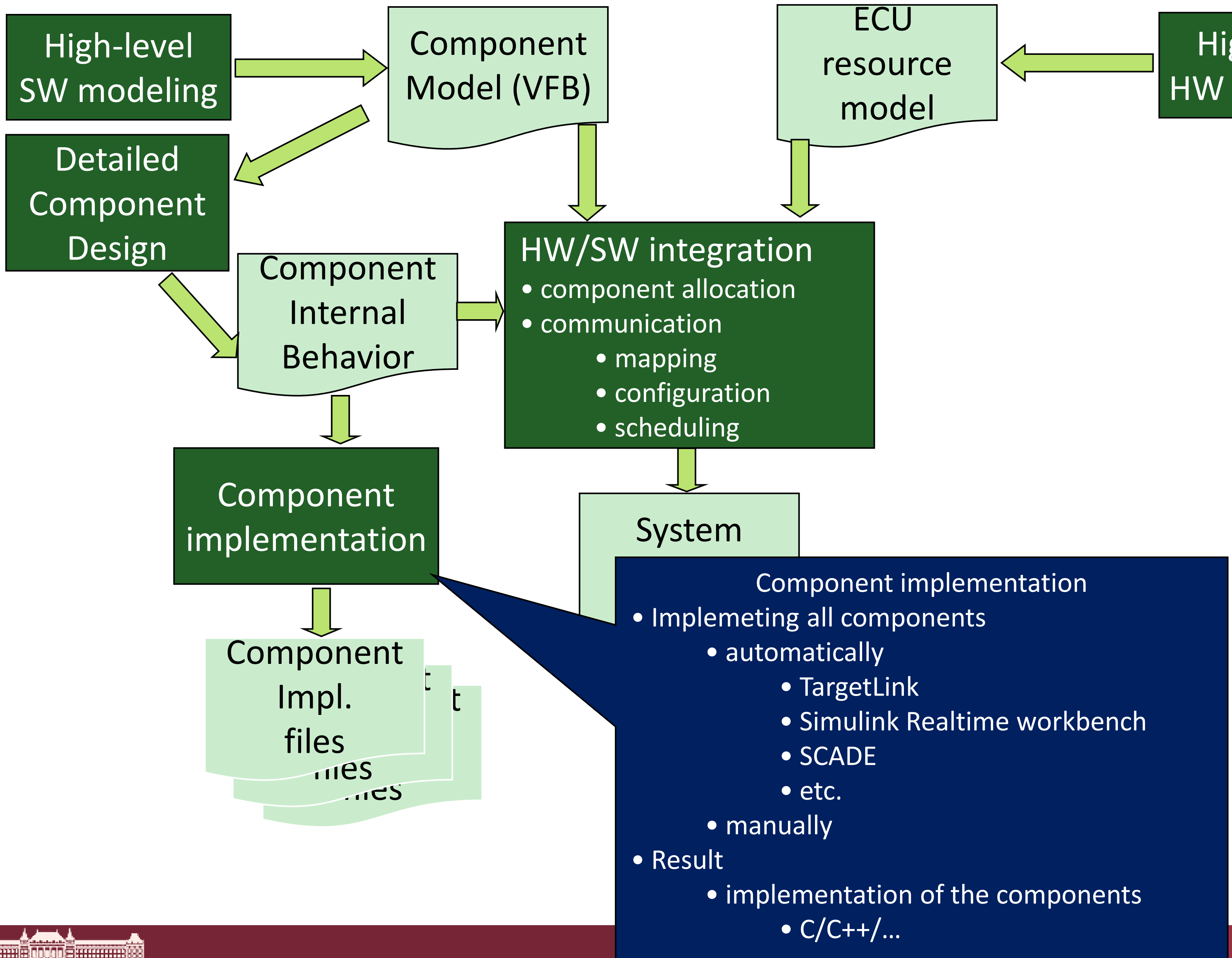
High-level design process



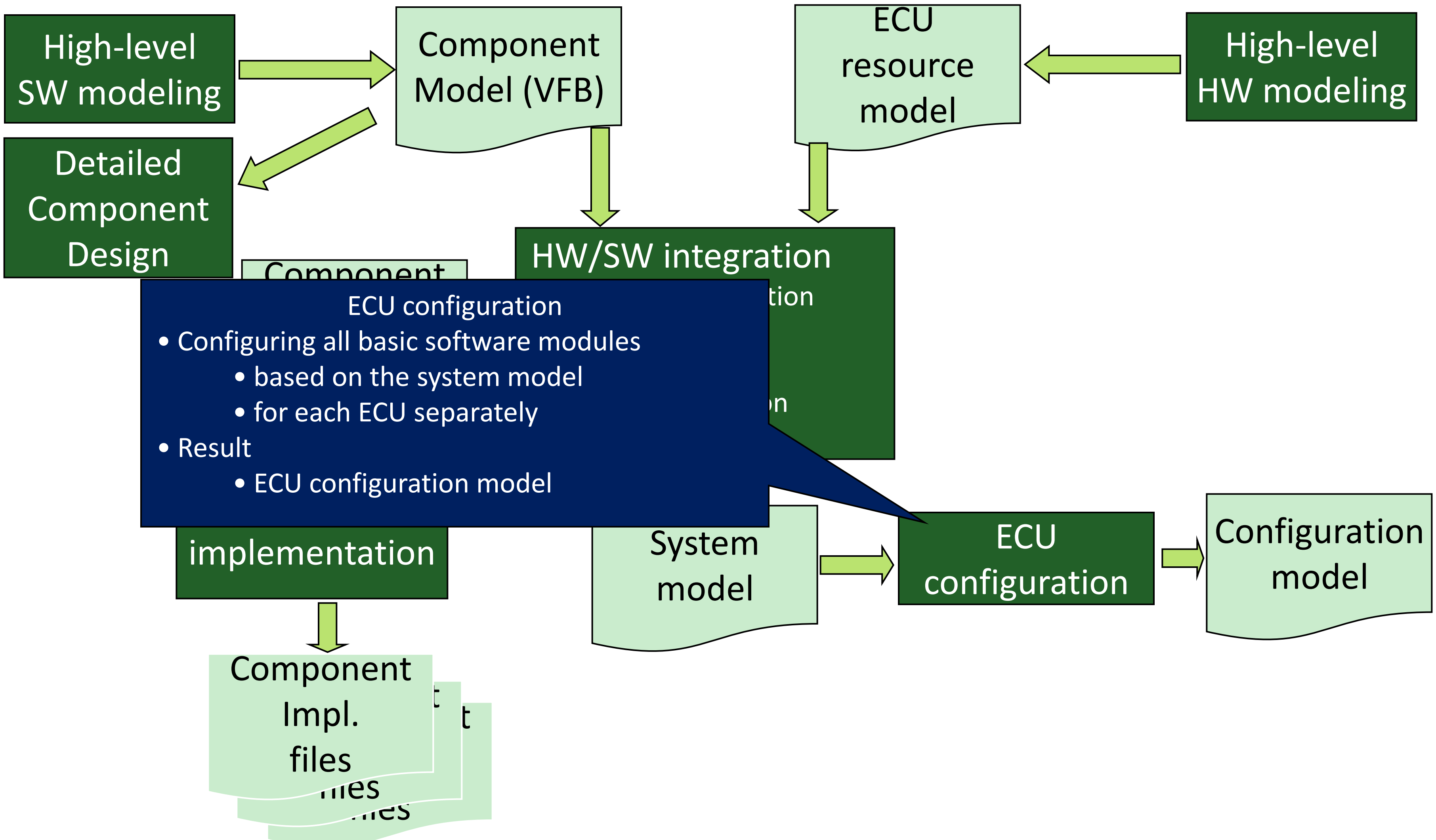
Hardware-software integration

- mapping
  - software component allocation
  - component implementation selection
  - data-element to signal mapping
    - inter-ECU communication
- communication configuration
  - signal to Protocol Data Unit (PDU) mapping
  - PDU to frame mapping
  - Signal, PDU, Frame triggering
  - Cluster and controller configuration
  - Frame scheduling (LIN, FlexRay)
- Result
  - System model describing the integrated HW/SW system

High-level design process



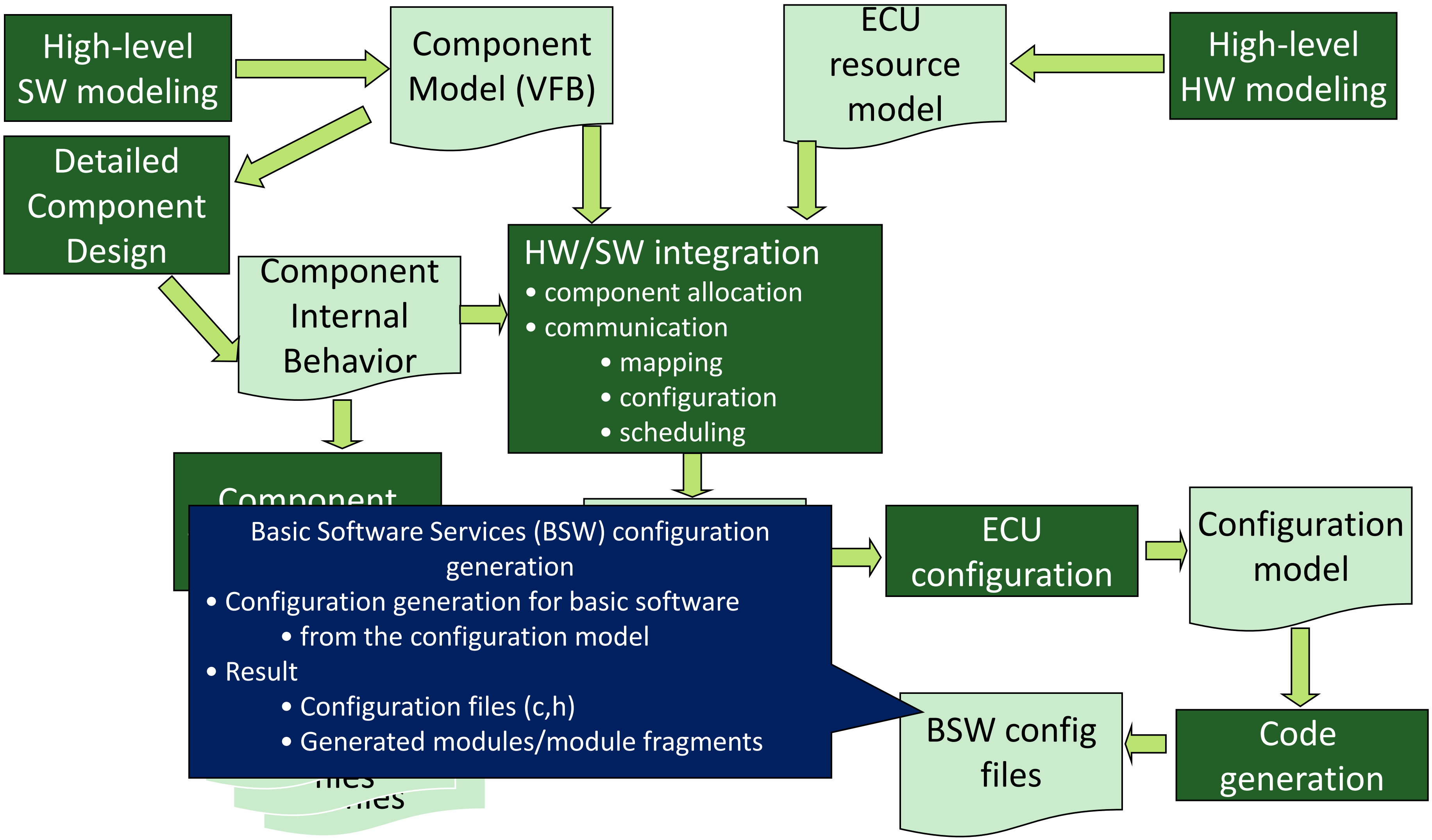
High-level design process



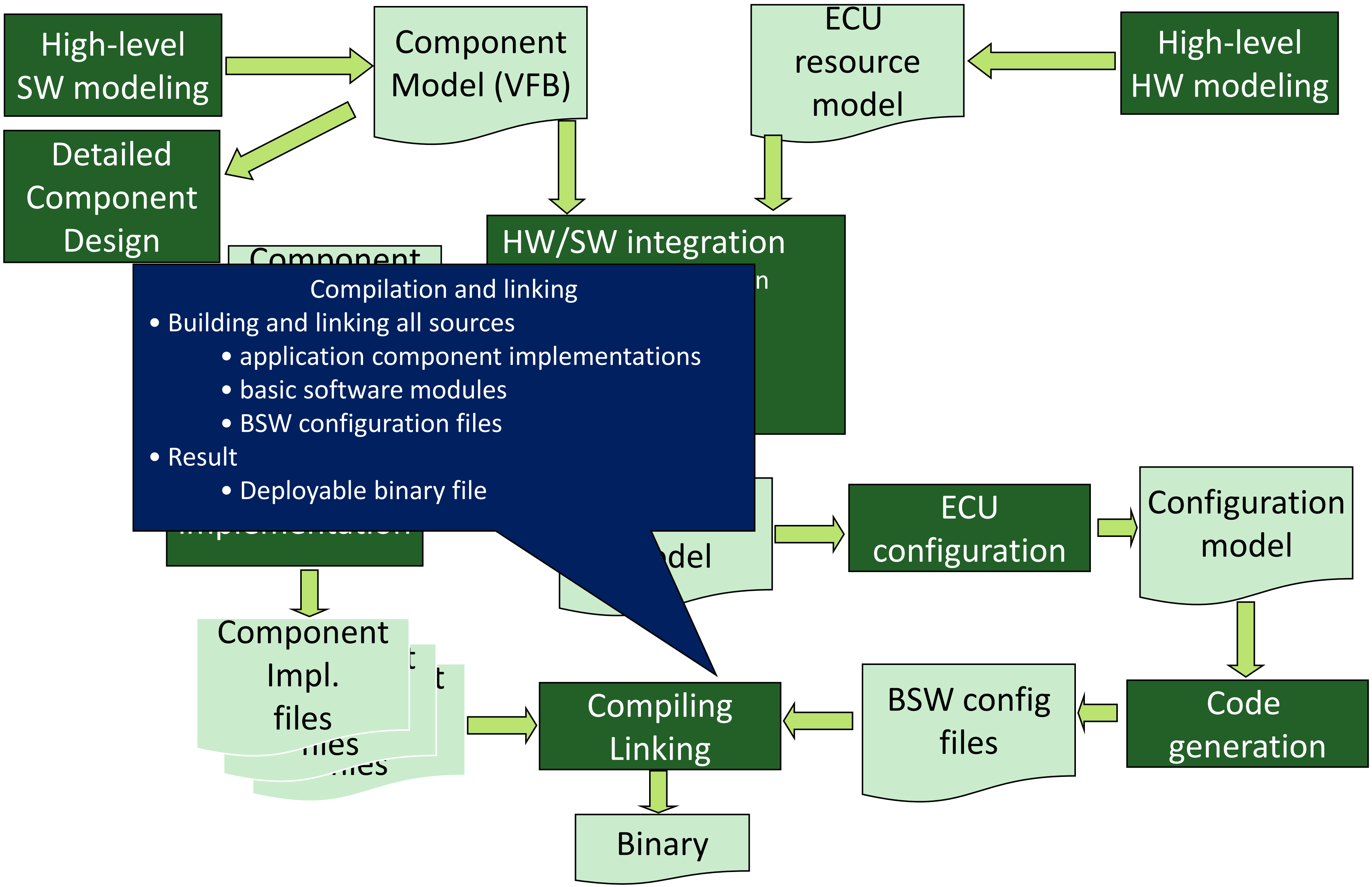
ECU configuration

- Configuring all basic software modules
  - based on the system model
  - for each ECU separately
- Result
  - ECU configuration model

High-level design process



High-level design process



Compilation and linking

- Building and linking all sources
  - application component implementations
  - basic software modules
  - BSW configuration files
- Result
  - Deployable binary file

Models in the design flow

- Software Component Template
  - Components, ports, interfaces
  - Internal behavior
  - Implementation (files, resource consumption, run time, etc.)
- ECU Resource Template
  - Hardware components, interconnections
- System Template
  - System topology, HW/SW mapping
  - Comm. matrix

Models in the design flow 2

- Basic Software Module Template
  - BSW modules
    - Services
    - Schedulable entities
    - Resource consumption
- ECU Configuration Parameter Definition Template
  - Configurable parameters of BSW modules
- ECU Configuration Description Template
  - Actual configurations of BSW modules
  - Based on the ECU Parameter Definition

AutoSAR vs. UML/SysML/... modeling

- AutoSAR defines models with
  - Domain Specific Constructs
  - *Precise* syntax
  - Synthesizable constructs
    - Direct model -> transformations
    - Direct model -> detailed model mappings
  - Different abstraction levels
    - From Virtual Function Bus to configuration
- Result
  - Models *are* primary design *and* implementation artifacts
    - More precise, consistent modeling should be done

AUTOSAR Components

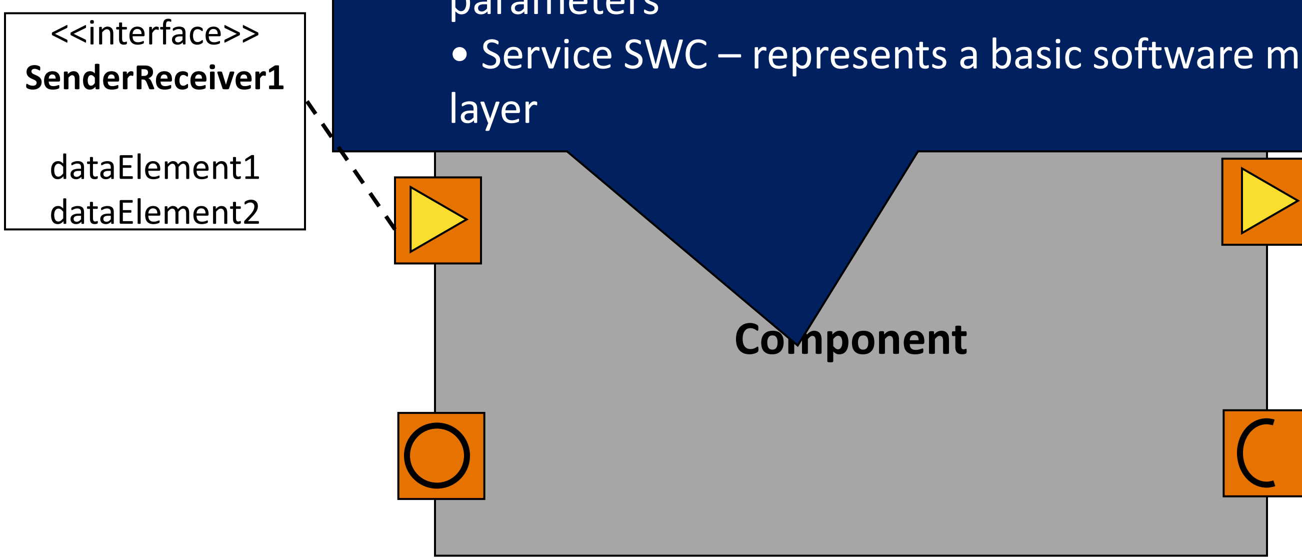
Component-oriented design

- What is a component?
  - *"A component is a self contained, reusable entity that encapsulates a specific functionality (and/or data), and communicates with other components via explicitly defined interfaces."*
- AutoSAR uses the term *component* for application-level components
  - Elements related to the high-level functionality of the system under design
- Basic software (middleware) components are called *module*s.
  - Standard elements of the AutoSAR architecture

Component-based approach

Component

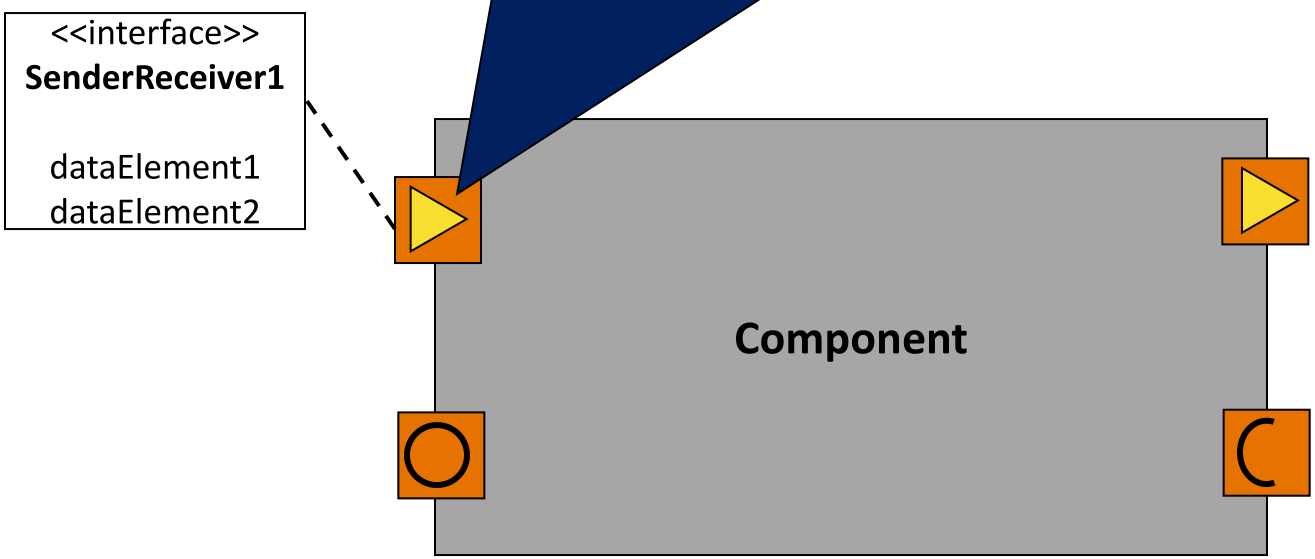
- Encapsulates a specific functionality
- Different kinds
  - Composite component – hierarchical refinement
  - Application SW component – generic, high level functionality
  - Sensor/actuator SW-C – handling sensor or actuator data
  - ECU HW abstraction – higher level device driver and abstraction
  - ComplexDeviceDriver – time-critical, low-level driver
  - Calibration parameter SWC – collects system calibration parameters
  - Service SWC – represents a basic software module from the service layer



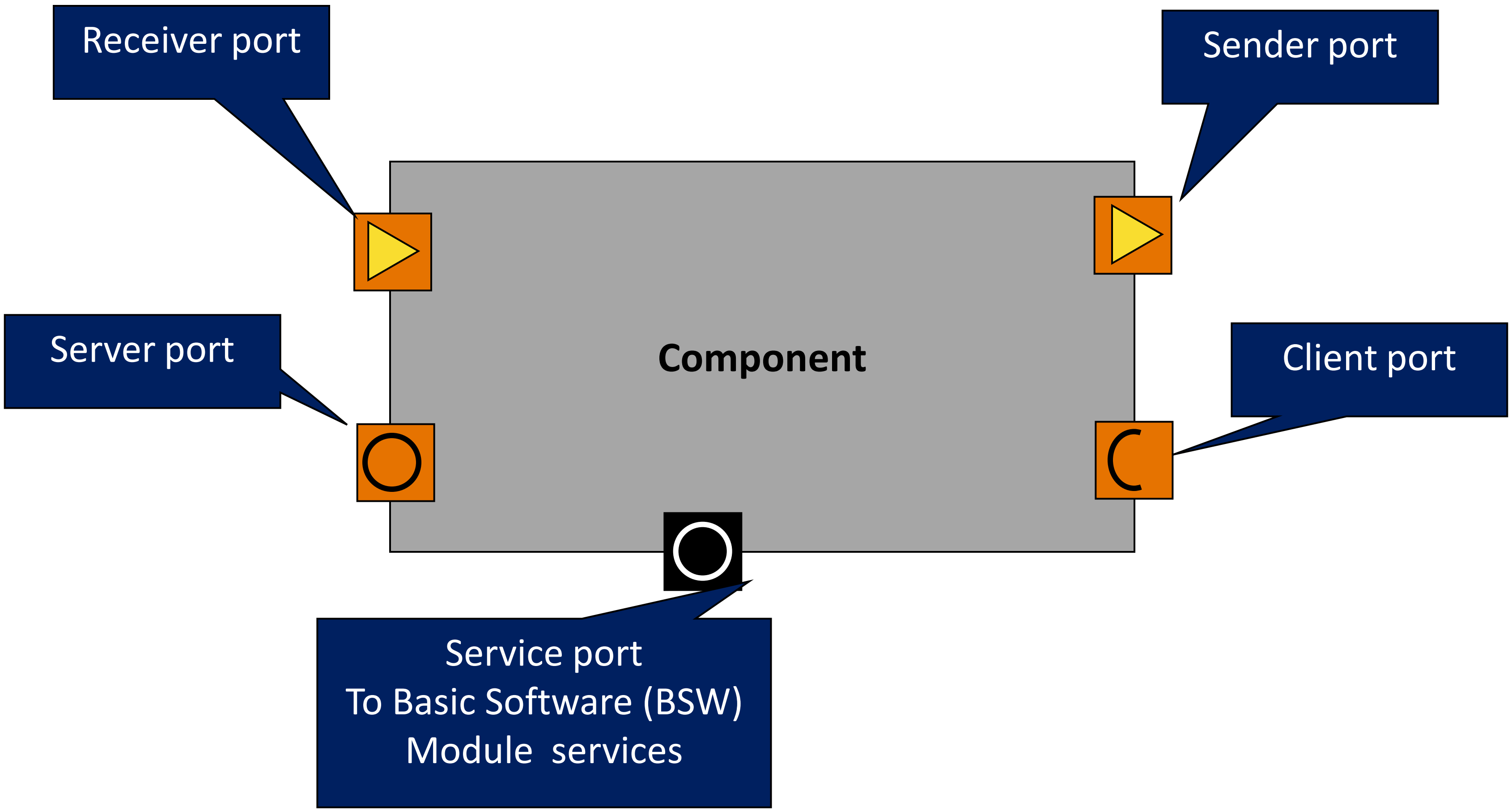
Component-based approach

Ports

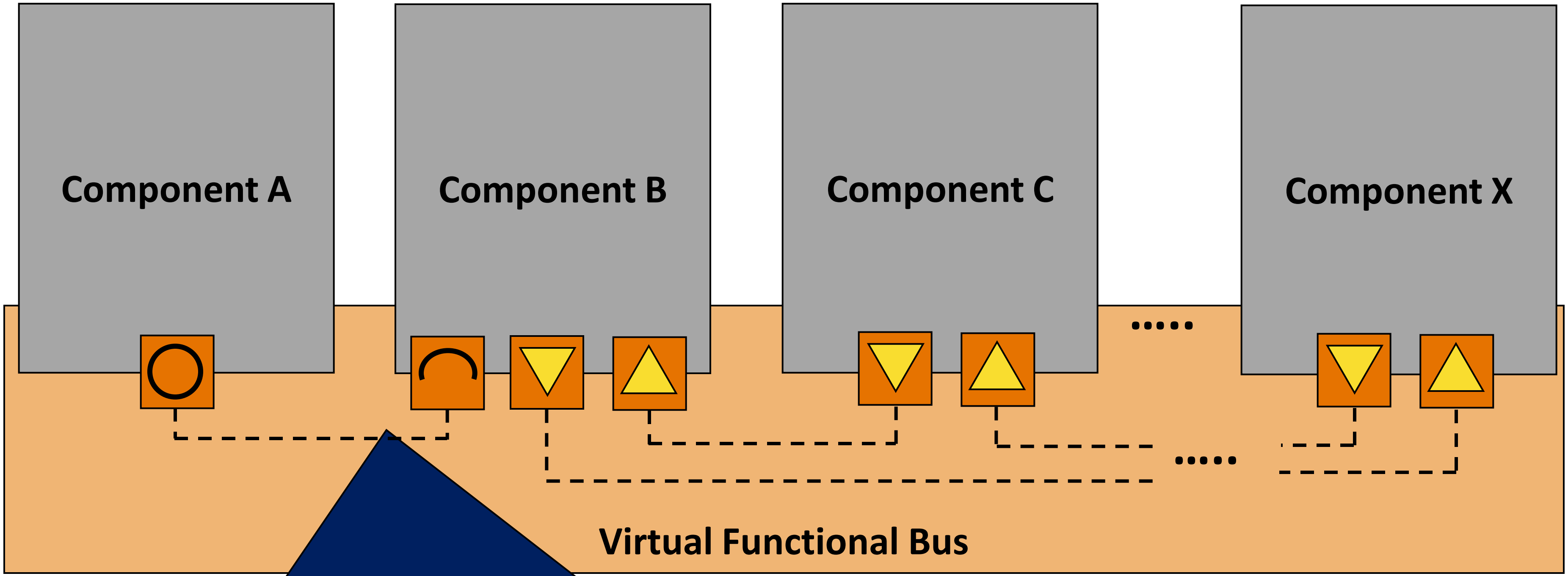
- The only interaction points between the component and its environment
- Are implementing *port interfaces*
  - sender receiver (message-based unidirectional communication)
  - client-server (remote procedure call)



Component-based approach – port notation



Component interconnection – the Virtual Functional Bus



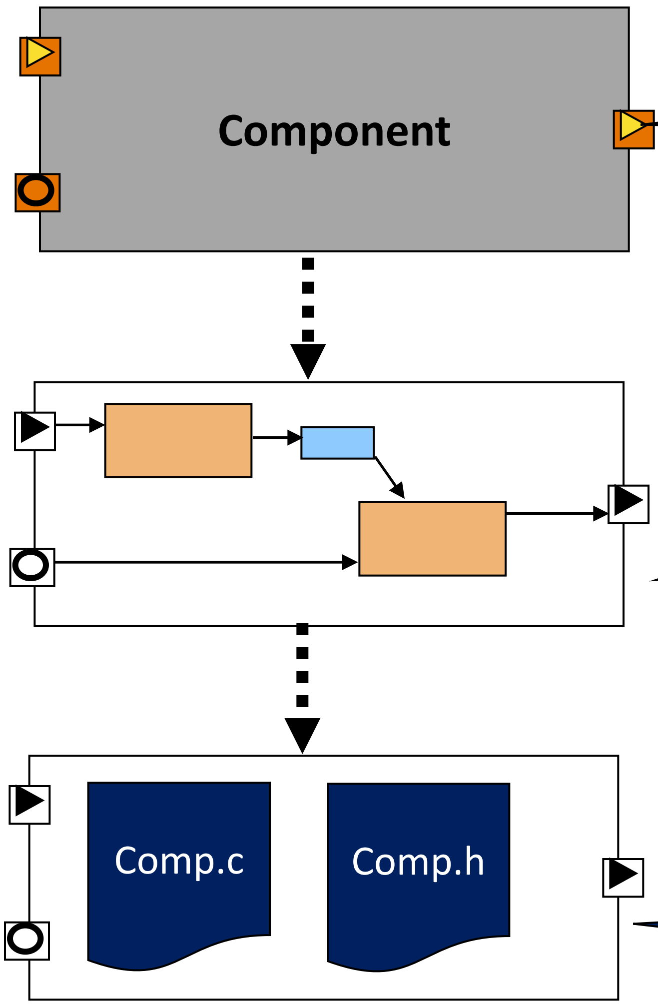
Virtual Functional Bus (VFB)

- Abstract interconnection layer
  - Implementation of data/control transport between components
  - No hardware/network dependency
  - Hides the details of the implementation
- Allows high-level integration *and simulation* of components
  - Before hardware architecture is chosen

Software Components

- On high-level, *atomic* components are black boxes
- Detailed design “looks into” these black boxes
- Main goals
  - Detail the behavior to get schedulable entities
  - Specify the semantics of port handling
  - Specify any service needs
  - Specify any RAM, nvRam needs

Refinement of a component



Black box definition of a component

Definition of component *internal behavior*
Schedulable entities, connections to the ports

Component implementation.
Specification of source and header files

Component internal behavior

- Specification of the internals of an atomic SWC
- Schedulable elements
  - Called: runnable entities
- Connection of ports
  - Port semantics
  - Port API options
- Inter-runnable communication
- Runnable activation and events

Summary

- AutoSAR defines
  - A component-oriented system design approach
    - Domain specific modeling language
    - A high level design process
    - Standard middleware (basic software) stack
      - Standard interfaces
      - Standard configuration descriptors
- AutoSAR compliant ECU software
  - Includes several BSW and application components
  - RTE provides the integration (glue) between these
  - Configuration and glue code is mostly auto-generated

EAST-ADL

EAST-ADL

- DSL for the vehicle electronics domain
- Complement/Embrace AUTOSAR
- Goal: handle all engineering information in an integrated way

- Development started in 2001
- Industry and Academic partners
- Acceptance (currently) is relatively low

Characteristics

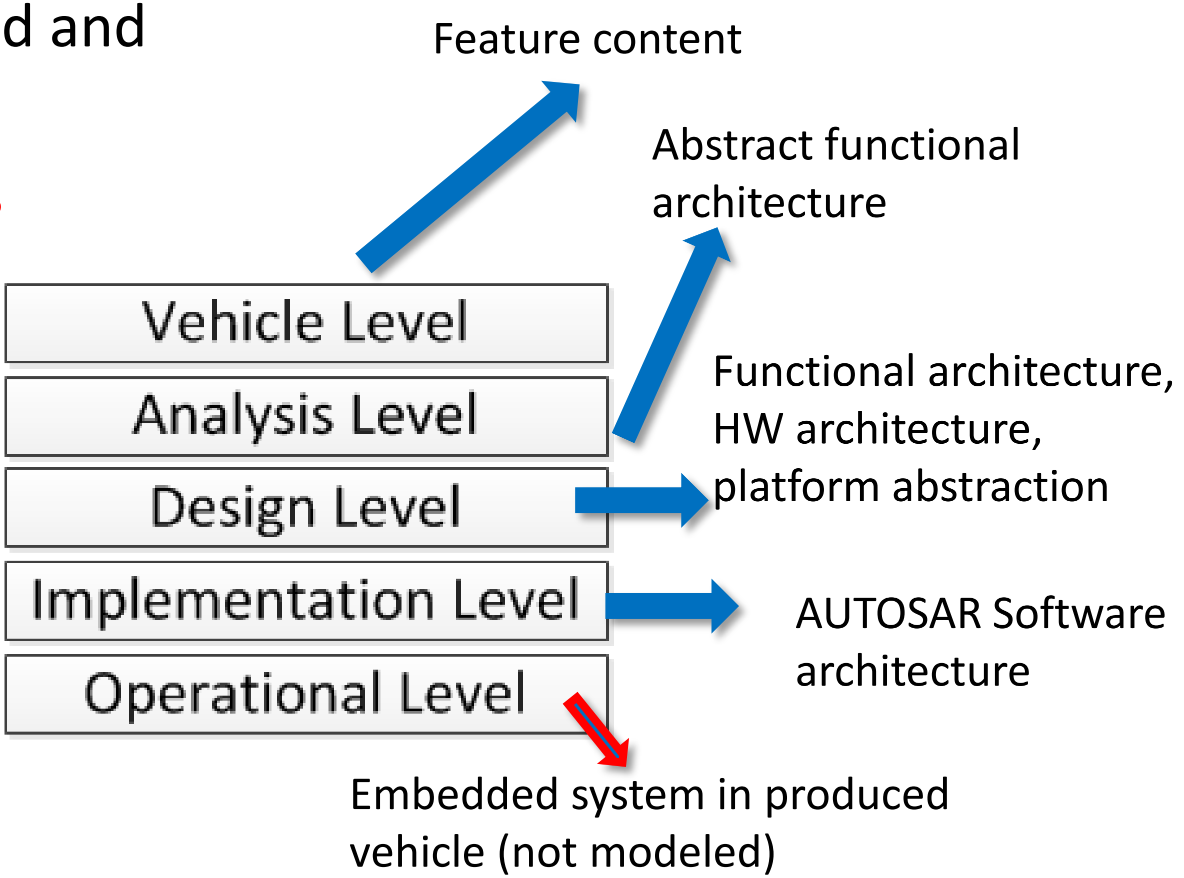
- Extends traditional ADL
  - Variability, requirements, safety, behavior environment modeling, design methodology
- Why not
  - UML : more vehicle specific
  - SysML: many concepts are similar but more vehicle specific
  - AUTOSAR: complements with respect to safety, functional structure, requirements, etc.
  - AADL: starts on a more abstract level
  - Proprietary (Matlab, Modelica, Statemate): provides an information structure of the engineering data and integrates external tools

EAST-ADL (by ATTESST)

- Typical vehicle engineering scenario
  - Vehicle manufacturer what to include in the next product
  - Chassis engineer analyses a novel algorithm
  - Application expert defines detailed design
  - SW engineer defines
    - SW architecture
    - Packing and allocation
    - Integration on ECU
  - Quality team does early phase validation and verification

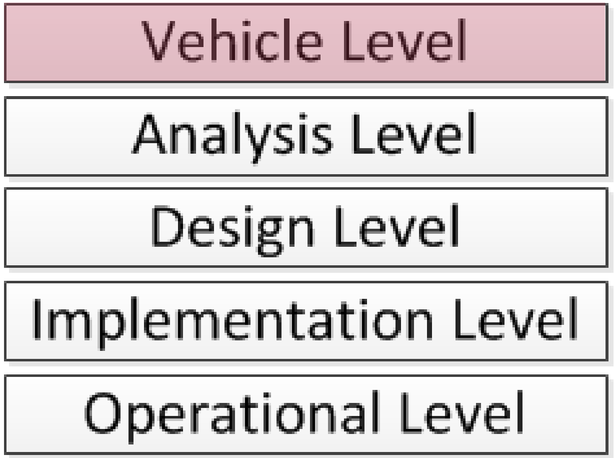
EAST-ADL

- System modeling Approach/framework
- Template how engineering information is organized and presented
- Separation of concerns
- Several abstraction layers
- Embrace the de-facto AUTOSAR SW representation



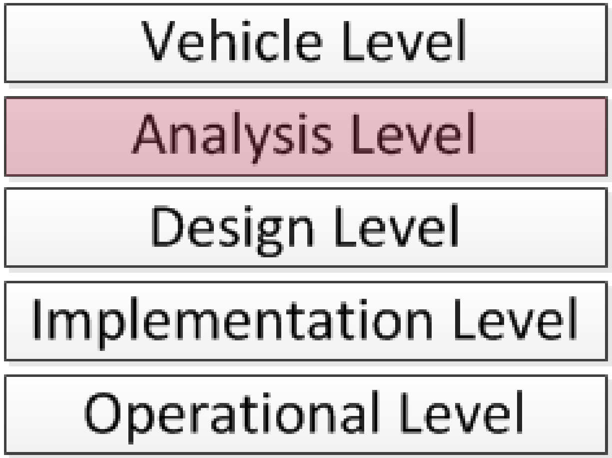
EAST-ADL

- Product Planner
  - Decide what to put in the next product
- Features represent
  - Properties/functionality/trait
  - Power window, Brake, steering, Collision Warning
- Vehicle Feature Model organize Features for the vehicle
- Variability mechanism supports → Product Line Architecture



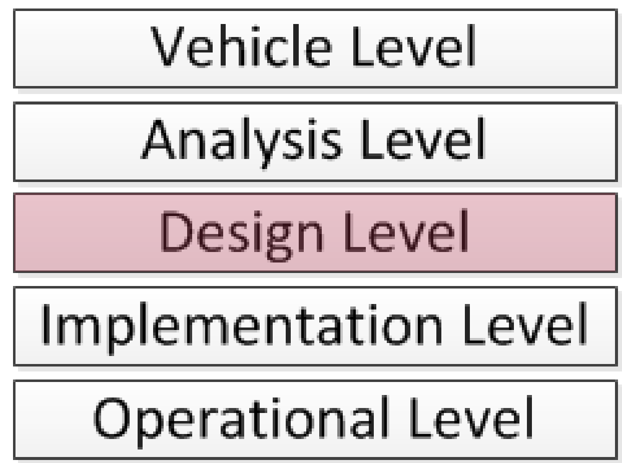
EAST-ADL

- Chassis Engineers
  - Analyses novel control algorithm
- Control algorithm is defined as a Function for the Environment model → OEM supplier agree on specification, model describes the requirements with traceability
- Focus on behavior and interaction functions
- EAST-ADL defines structure and allows legacy tools to be used for analysis, simulation, etc.
- Realization details are omitted
  - Mainly to understand key aspects



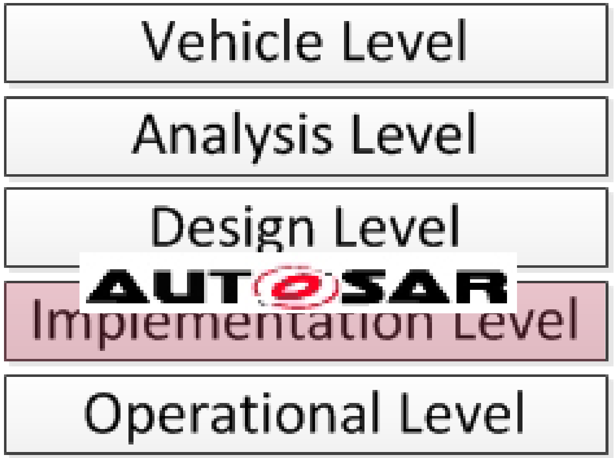
EAST-ADL

- Application expert defines detailed design
- Detailed functional architecture consist of
  - HW architecture
  - Allocation
  - Fault tolerance
  - Implementation concerns
  - Sensor, actuator constraints
- Focus on behavior and interaction of functions
- Abstract system architecture is defined and assesses



EAST-ADL

- Software engineer defines the SW Architecture
- AUTOSAR Application SW Components are defined
- Set of SW components realizes the Functional Architecture
- All SW related elements are defined in this level
  - Legacy code integration
  - Allocation (code level)
  - Performance tests and analysis
  - Verification of final product
  - Re-use
  - Mapping → which functions are realized by which SW component



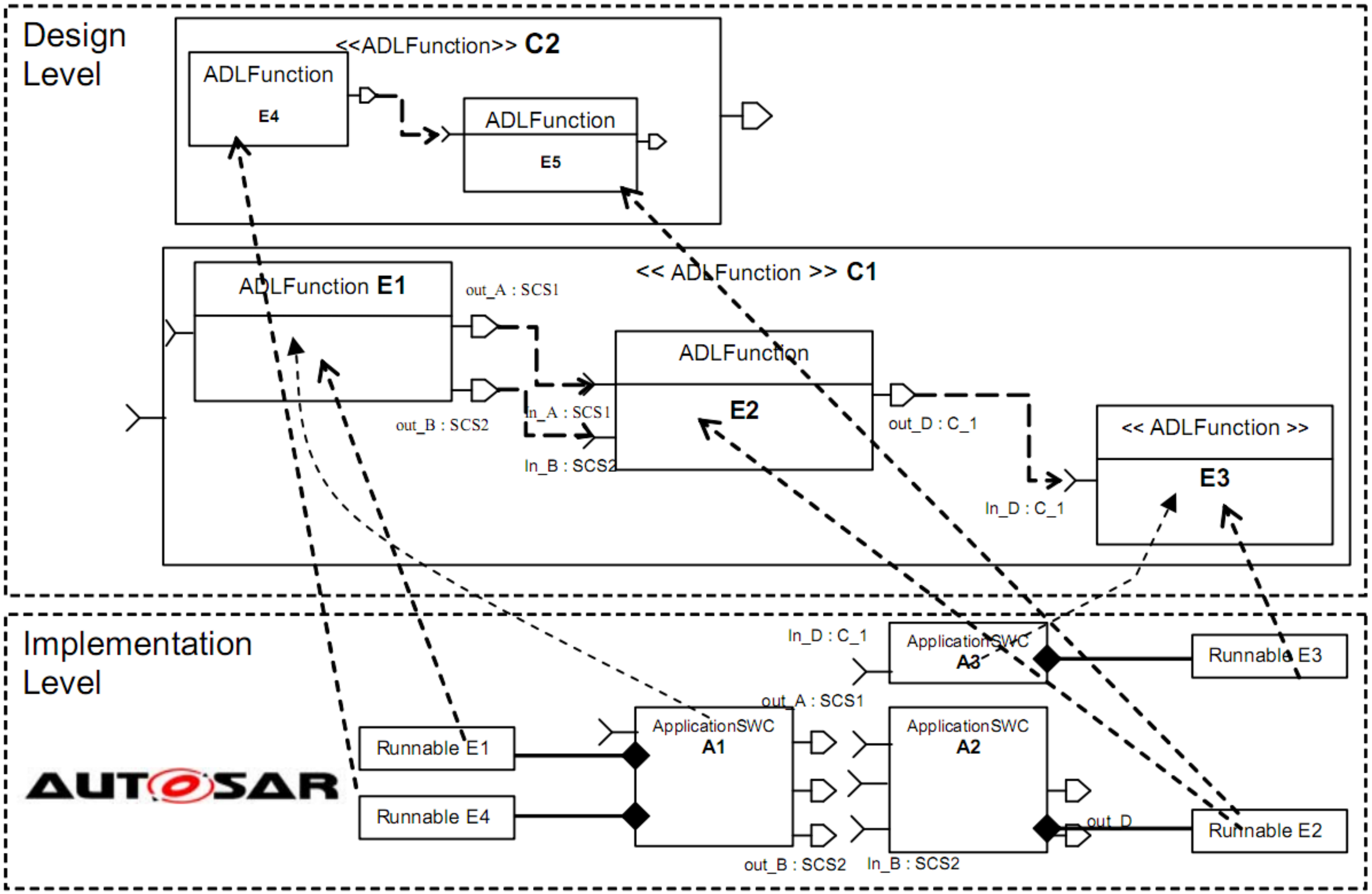
EAST-ADL

- Additional models
  - Environment model
    - In-vehicle, near and far environment
    - Different models for different scenarios
  - Traceability
    - Realization relation from top-to-down identify, which element is realized by which more concrete element
- EAST-ADL complements AUTOSAR
  - Aspects beyond SW architecture (variability, safety, etc.)
  - Provides means to define what the SW does
  - Provides means to model strategic properties
  - Error behavior modeling and safety related aspects

EAST-ADL

- Variability
  - Feature trees (mandatory/optional) → product line
- Error modeling and failure analysis
  - Modeling concepts of hazards and error propagation
  - Basis for Fault Tree, Fault Mode and Effect analysis
- Behavior
  - Definition of behavior semantics → allow legacy tool integration (Simulink)
- Timing
  - Formalization of timing requirements and properties in structural models (e.g., reaction, age, synchronization, etc.)

Sample EAST-ADL model and binding



EAST-ADL Summary

- EAST-ADL provide information structure for the design of vehicle embedded systems
- Uses multi layers of abstraction in a top-down manner
- Fully aligned with AUTOSAR