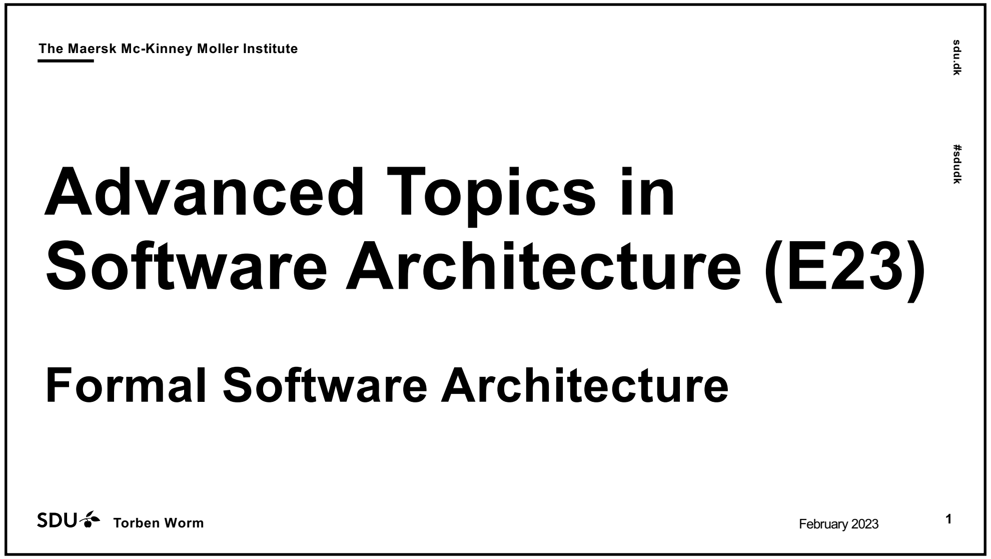


The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

Advanced Topics in Software Architecture (E23)

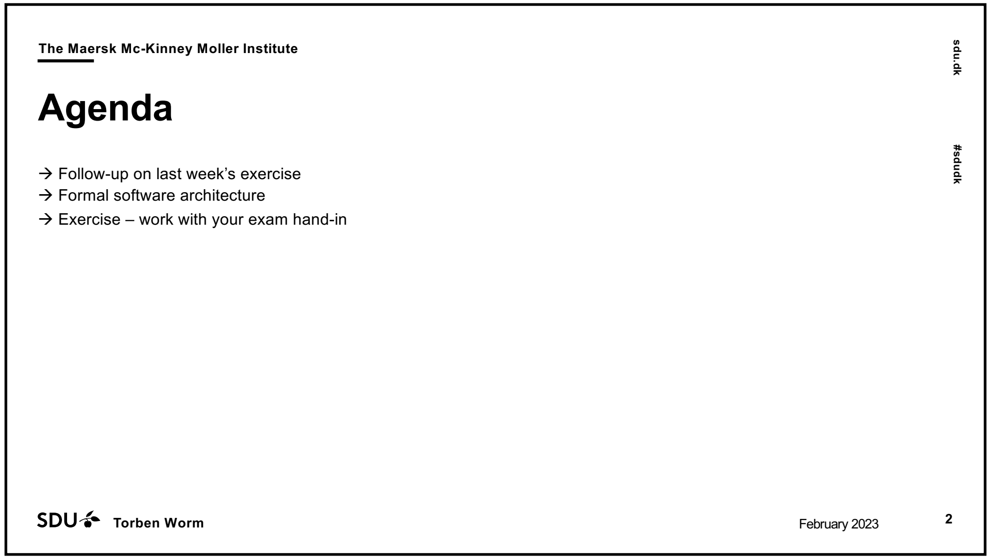
Formal Software Architecture

SDU Torben Worm

February 2023

1

1



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

Agenda

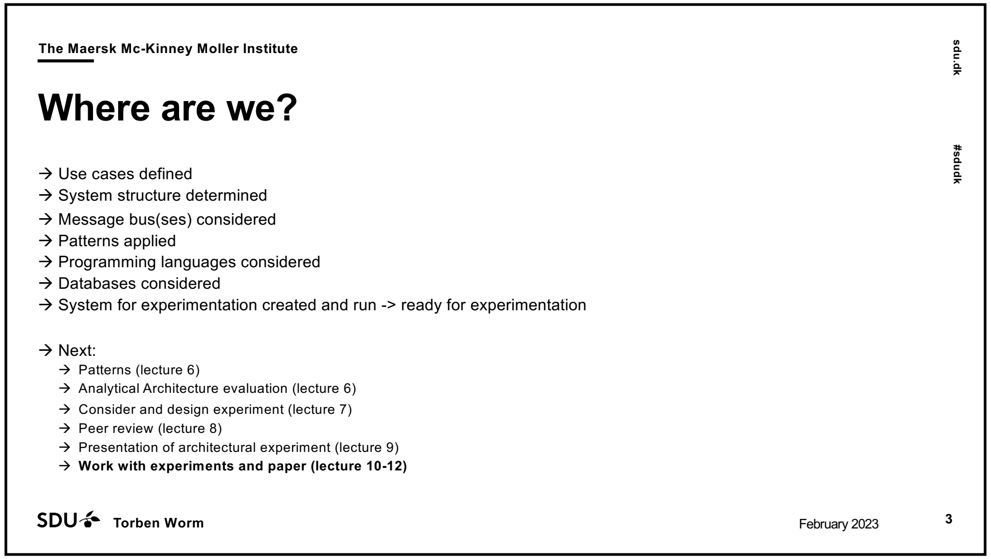
→ Follow-up on last week's exercise
→ Formal software architecture
→ Exercise – work with your exam hand-in

SDU Torben Worm

February 2023

2

2



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

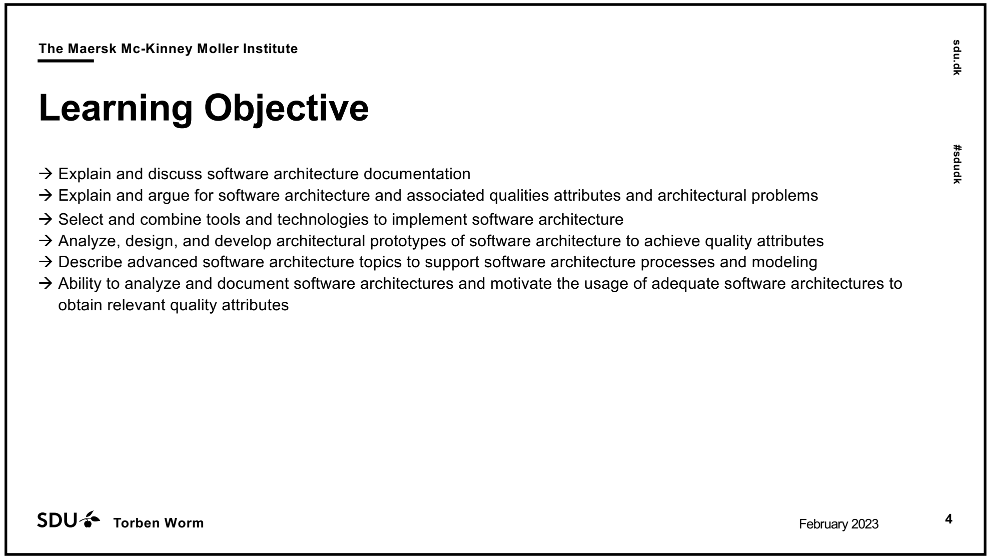
Where are we?

- → Use cases defined
- → System structure determined
- → Message bus(ses) considered
- → Patterns applied
- → Programming languages considered
- → Databases considered
- → System for experimentation created and run -> ready for experimentation

- → Next:
  - → Patterns (lecture 6)
  - → Analytical Architecture evaluation (lecture 6)
  - → Consider and design experiment (lecture 7)
  - → Peer review (lecture 8)
  - → Presentation of architectural experiment (lecture 9)
  - → **Work with experiments and paper (lecture 10-12)**

SDU Torben Worm February 2023 3

3



The Maersk Mc-Kinney Moller Institute

sdu.dk

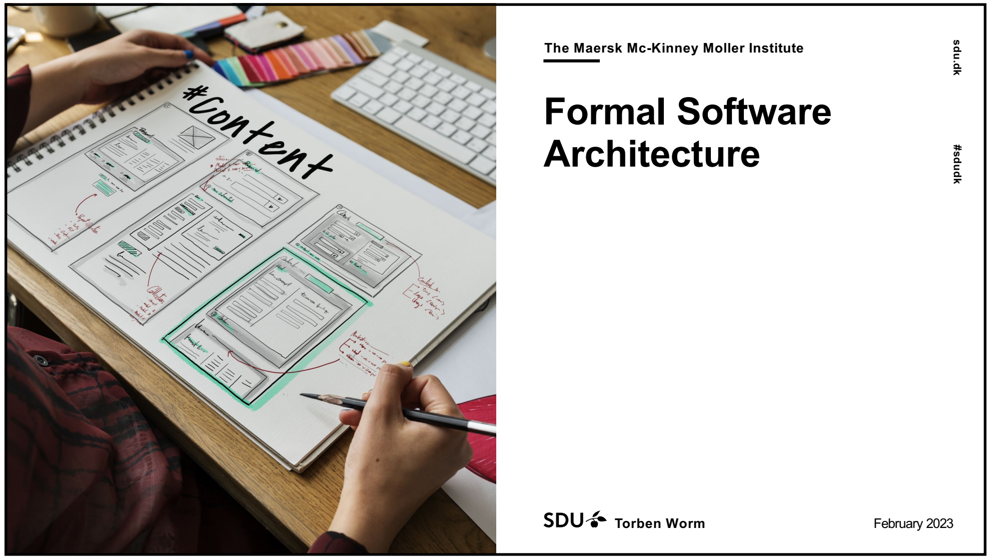
#sdudk

Learning Objective

- → Explain and discuss software architecture documentation
- → Explain and argue for software architecture and associated qualities attributes and architectural problems
- → Select and combine tools and technologies to implement software architecture
- → Analyze, design, and develop architectural prototypes of software architecture to achieve quality attributes
- → Describe advanced software architecture topics to support software architecture processes and modeling
- → Ability to analyze and document software architectures and motivate the usage of adequate software architectures to obtain relevant quality attributes

SDU Torben Worm February 2023 4

4



The Maersk Mc-Kinney Moller Institute

sdu.dk

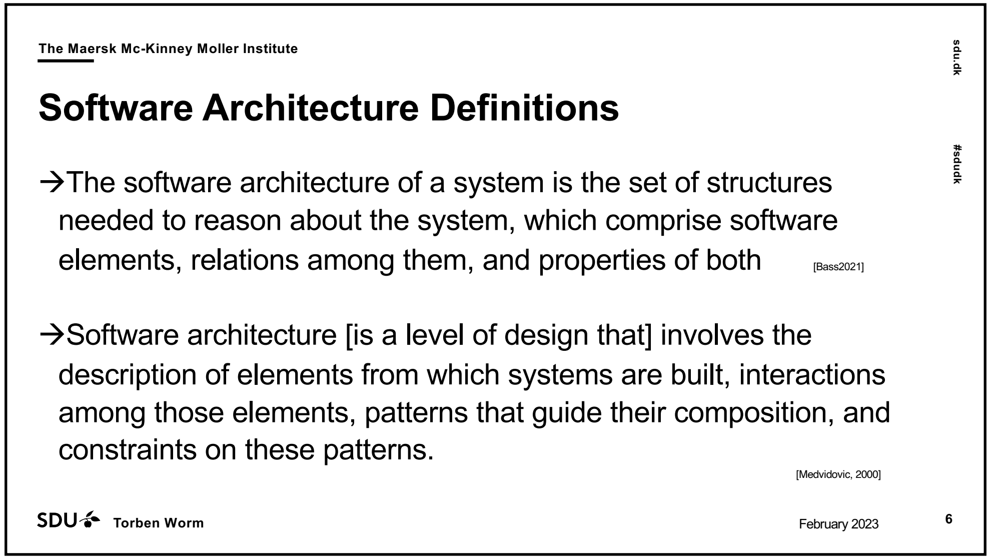
#sduk

Formal Software Architecture

SDU Torben Worm

February 2023

5



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sduk

Software Architecture Definitions

→The software architecture of a system is the set of structures needed to reason about the system, which comprise software elements, relations among them, and properties of both [Bass2021]

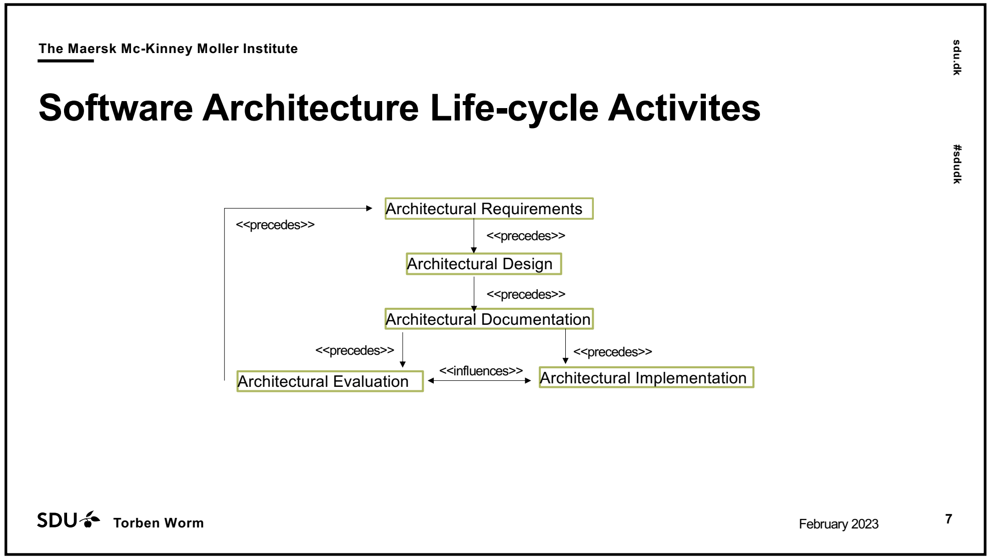
→Software architecture [is a level of design that] involves the description of elements from which systems are built, interactions among those elements, patterns that guide their composition, and constraints on these patterns.

[Medvidovic, 2000]

SDU Torben Worm

February 2023 6

6



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

Software Architecture Life-cycle Activites

Architectural Requirements

<<precedes>>

Architectural Design

<<precedes>>

Architectural Documentation

<<precedes>>

<<precedes>>

Architectural Evaluation

<<influences>>

Architectural Implementation

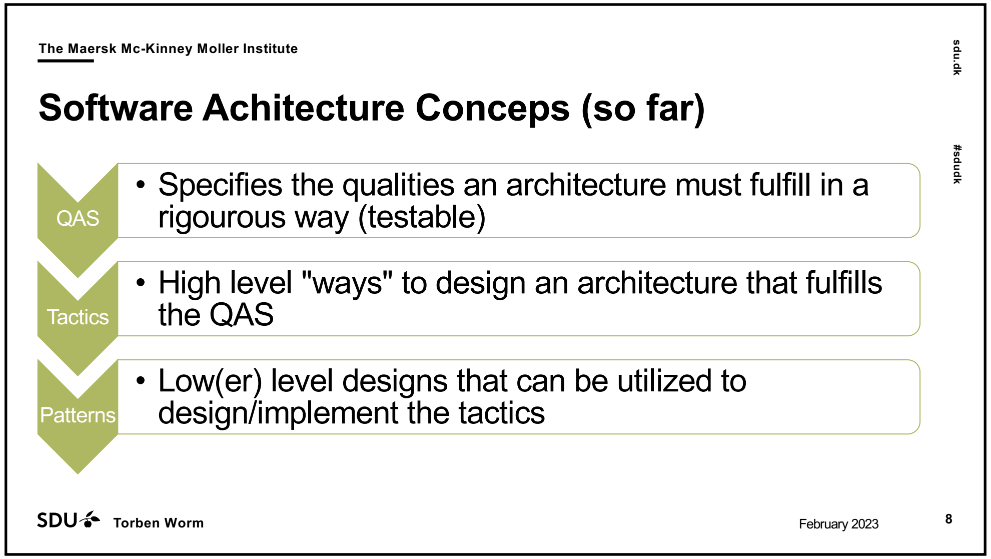
<<precedes>>

SDU Torben Worm

February 2023

7

7



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

Software Achitecture Conceps (so far)

QAS

- Specifies the qualities an architecture must fulfill in a rigourous way (testable)

Tactics

- High level "ways" to design an architecture that fulfills the QAS

Patterns

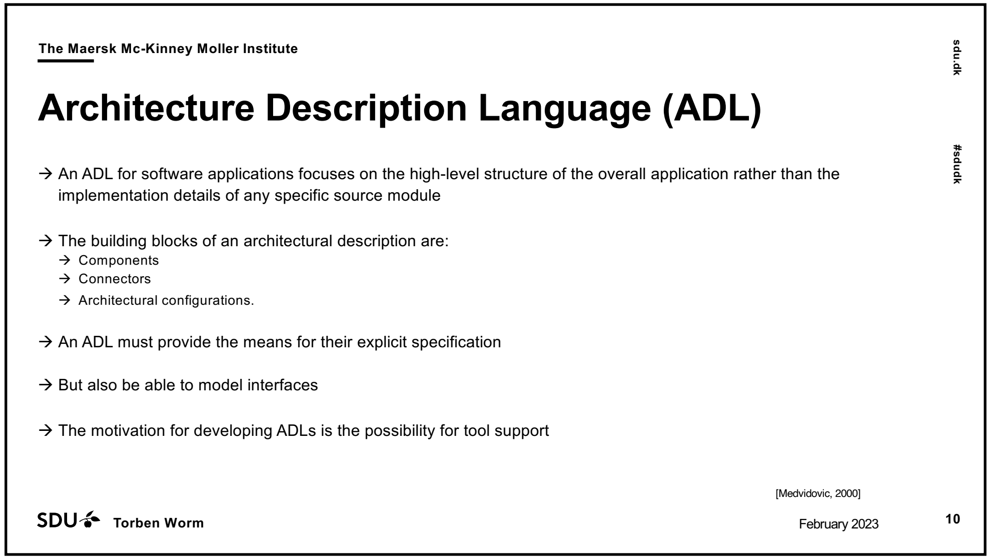
- Low(er) level designs that can be utilized to design/implement the tactics

SDU Torben Worm

February 2023

8

8



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sduk

Architecture Description Language (ADL)

→ An ADL for software applications focuses on the high-level structure of the overall application rather than the implementation details of any specific source module

→ The building blocks of an architectural description are:
- → Components
- → Connectors
- → Architectural configurations.

→ An ADL must provide the means for their explicit specification

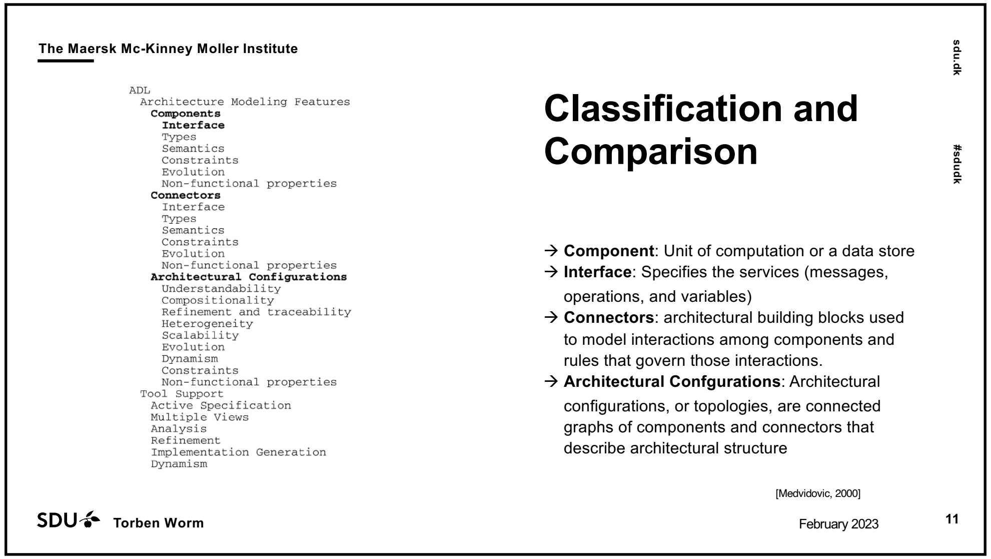
→ But also be able to model interfaces

→ The motivation for developing ADLs is the possibility for tool support

[Medvidovic, 2000]

SDU Torben Worm February 2023 10

10



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sduk

```
ADL
  Architecture Modeling Features
    Components
      Interface
      Types
      Semantics
      Constraints
      Evolution
      Non-functional properties
    Connectors
      Interface
      Types
      Semantics
      Constraints
      Evolution
      Non-functional properties
    Architectural Configurations
      Understandability
      Compositionality
      Refinement and traceability
      Heterogeneity
      Scalability
      Evolution
      Dynamism
      Constraints
      Non-functional properties
  Tool Support
    Active Specification
    Multiple Views
    Analysis
    Refinement
    Implementation Generation
    Dynamism
```

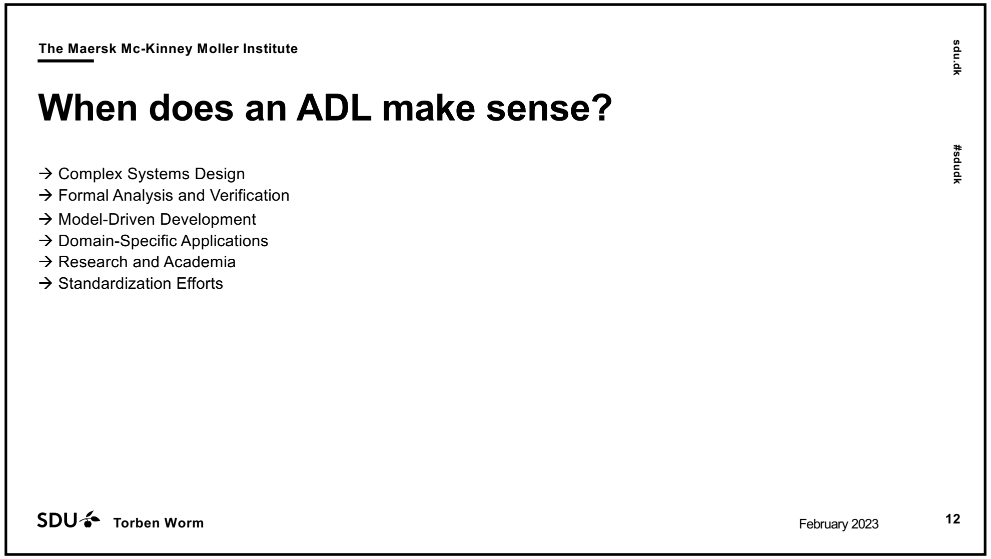
Classification and Comparison

→ **Component**: Unit of computation or a data store
→ **Interface**: Specifies the services (messages, operations, and variables)
→ **Connectors**: architectural building blocks used to model interactions among components and rules that govern those interactions.
→ **Architectural Confgurations**: Architectural configurations, or topologies, are connected graphs of components and connectors that describe architectural structure

[Medvidovic, 2000]

SDU Torben Worm February 2023 11

11



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

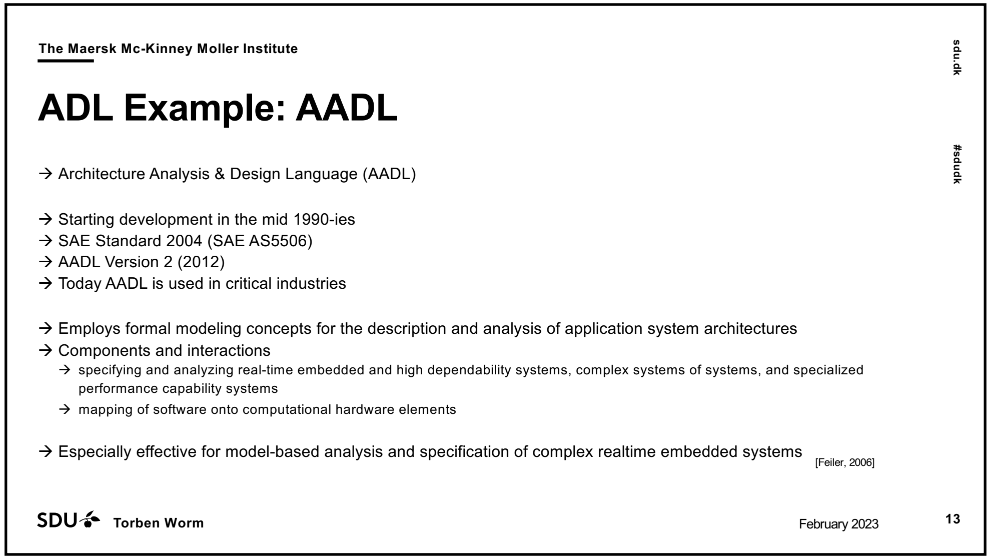
When does an ADL make sense?

→ Complex Systems Design
→ Formal Analysis and Verification
→ Model-Driven Development
→ Domain-Specific Applications
→ Research and Academia
→ Standardization Efforts

SDU Torben Worm

February 2023 12

12



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

ADL Example: AADL

→ Architecture Analysis & Design Language (AADL)

→ Starting development in the mid 1990-ies
→ SAE Standard 2004 (SAE AS5506)
→ AADL Version 2 (2012)
→ Today AADL is used in critical industries

→ Employs formal modeling concepts for the description and analysis of application system architectures
→ Components and interactions
  → specifying and analyzing real-time embedded and high dependability systems, complex systems of systems, and specialized performance capability systems
  → mapping of software onto computational hardware elements

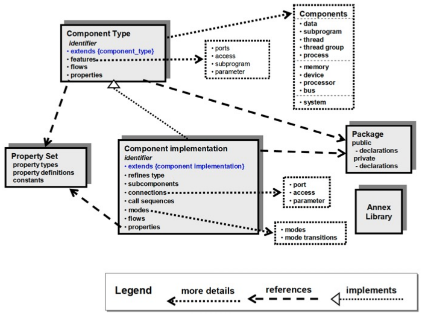
→ Especially effective for model-based analysis and specification of complex realtime embedded systems [Feiler, 2006]

SDU Torben Worm

February 2023 13

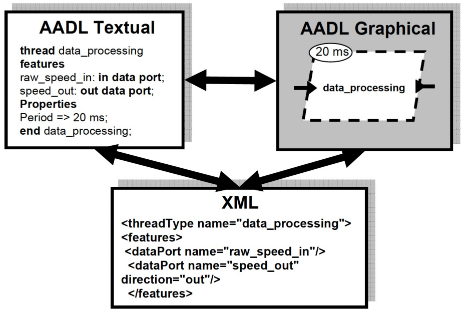
13

AADL Elements



[Feiler, 2006]

Representation

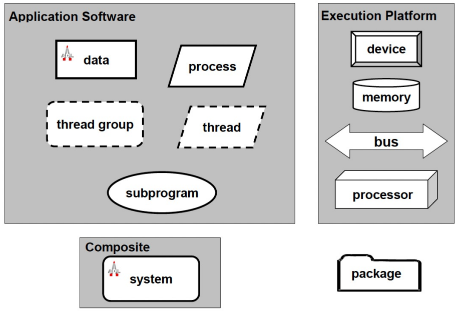


```
thread data_processing
features
raw_speed_in: in data port;
speed_out: out data port;
Properties
Period => 20 ms;
end data_processing;
```

```
<threadType name="data_processing">
<features>
 <dataPort name="raw_speed_in"/>
  <dataPort name="speed_out"
direction="out"/>
  </features>
```

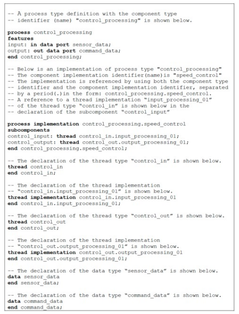
[Feiler, 2006]

Graphical Notation

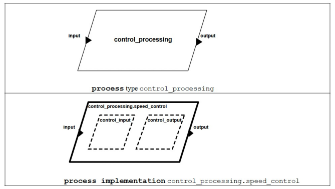


[Feiler, 2006]

Textual and Graphical Specification

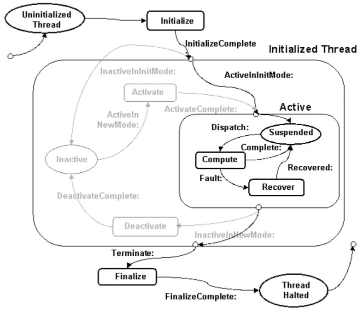


```
-- A process type definition with the component type
-- identifier (name) "control_processing" is shown below.

process control_processing
features
input: in data port sensor_data;
output: out data port command_data;
end control_processing;

-- Below is an implementation of process type "control_processing"
-- The component implementation identifier(name)is "speed_control"
-- The implementation is referenced by using both the component type
-- identifier and the component implementation identifier, separated
-- by a period(.)in the form: control_processing.speed_control.
-- A reference to a thread implementation "input_processing_01"
-- of the thread type "control_in" is shown below in the
-- declaration of the subcomponent "control_input"

process implementation control_processing.speed_control
subcomponents
control_input: thread control_in.input_processing_01;
control_output: thread control_out.output_processing_01;
end control_processing.speed_control;

-- The declaration of the thread type "control_in" is shown below.
thread control_in
end control_in;

-- The declaration of the thread implementation
-- "control_in.input_processing_01" is shown below.
thread implementation control_in.input_processing_01
end control_in.input_processing_01;

-- The declaration of the thread type "control_out" is shown below.
thread control_out
end control_out;

-- The declaration of the thread implementation
-- "control_out.output_processing_01" is shown below.
thread implementation control_out.output_processing_01
end control_out.output_processing_01;

-- The declaration of the data type "sensor_data" is shown below.
data sensor_data
end sensor_data;

-- The declaration of the data type "command_data" is shown below.
data command_data
end command_data;
```


input
control_processing
output
process type control_processing
control_processing.speed_control
control_input
control_output
input
output
process implementation control_processing.speed_control

[Feiler, 2006]

Thread Execution State Machine



[Feiler, 2006]

Thread Properties

```
thread control
properties
-- nominal execution properties
Compute_Entrypoint => "control_ep";
Compute_Execution_Time => 5 ms .. 10 ms;
Compute_Deadline => 20 ms;
Dispatch_Protocol => Periodic;
-- initialization execution properties
Initialize_Entrypoint => "init_control";
Initialize_Execution_Time => 2 ms .. 5 ms;
Initialize_Deadline => 10 ms;
end control;
```

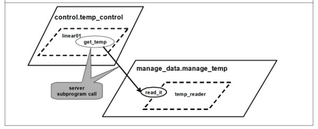
[Feiler, 2006]

The Maersk Mc-Kinney Moller Institute

sdu.dk #sdudk

Sub-routines

```
process manage_data
end manage_data;
--
process implementation manage_data.manage_temp
subcomponents
temp_reader: thread read.read_temp;
end manage_data.manage_temp;
--
thread read
features
read_it: server subprogram acquire.temp;
end read;
--
thread implementation read.read_temp
end read.read_temp;
--
subprogram acquire
end acquire;
--
subprogram implementation acquire.temp
end acquire.temp;
```



[Feiler, 2006]

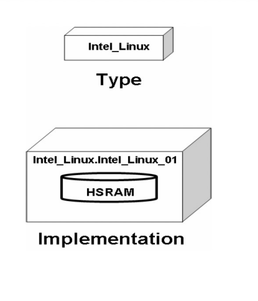
20

The Maersk Mc-Kinney Moller Institute

sdu.dk #sdudk

Processor

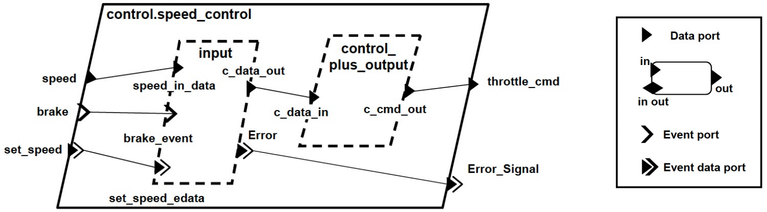
```
processor Intel_Linux
properties
Hardware_Source_Language=> VHDL;
Hardware_Description_Source_Text  =>
"intel_vhdl_1, intel_vhdl_2";
end Intel_Linux;
--
processor implementation
Intel_Linux.Intel_Linux_01
subcomponents
HSRAM: memory RAM.Intel_RAM;
end Intel_Linux.Intel_Linux_01;
--
memory RAM
end RAM;
--
memory implementation RAM.Intel_RAM
end RAM.Intel_RAM;
```



[Feiler, 2006]

21

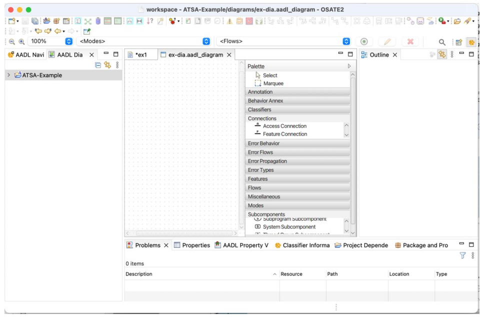
Ports



Legend

[Feiler, 2006]

Tool support - OSATE



The Maersk Mc-Kinney Moller Institute

sdu.dk

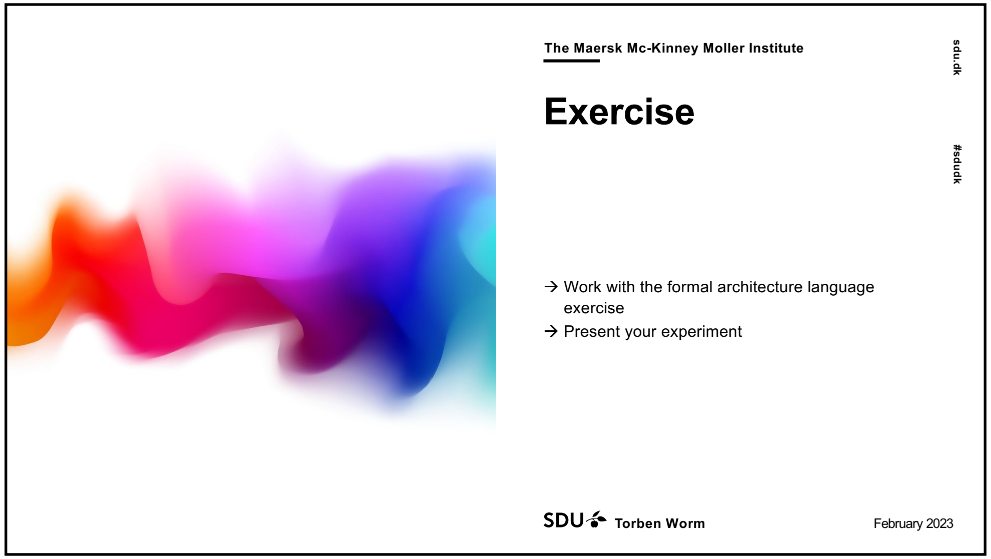
#sdudk

Next lecture

- → Wrap-up of the course – what have we been through?
- → Exam information
  - → Reiteration of the requirements for the exam
  - → Hand-ins
    - → Paper
    - → Reflection document
    - → Source code
- → Course evaluation
- → Q&A

SDU Torben Worm February 2023 24

24



The Maersk Mc-Kinney Moller Institute

sdu.dk

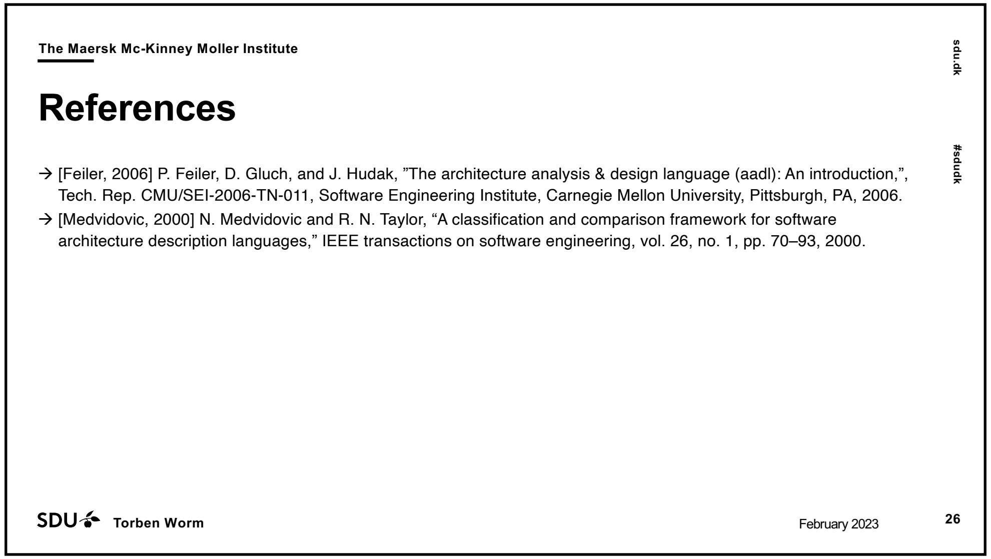
#sdudk

Exercise

- → Work with the formal architecture language exercise
- → Present your experiment

SDU Torben Worm February 2023

25



The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

References

- → [Feiler, 2006] P. Feiler, D. Gluch, and J. Hudak, ”The architecture analysis & design language (aadl): An introduction,”, Tech. Rep. CMU/SEI-2006-TN-011, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 2006.
- → [Medvidovic, 2000] N. Medvidovic and R. N. Taylor, “A classification and comparison framework for software architecture description languages,” IEEE transactions on software engineering, vol. 26, no. 1, pp. 70–93, 2000.

SDU Torben Worm February 2023 26

26

The Maersk Mc-Kinney Moller Institute

sdu.dk

#sdudk

End of Presentation

SDU Torben Worm February 2023 27

27