München, 2019-04-17

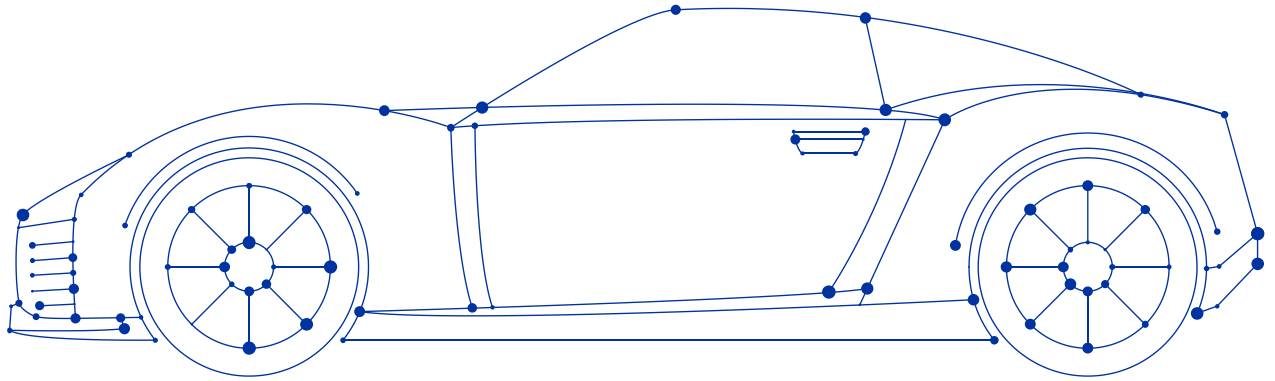
From Sensors to Driving Functions – Develop Your Own Car

Kick-off

Praktikum, summer semester 2019

Hernan Ponce de Leon, Sudeep Kanav, Marco Volpe
Thomas Böhm, Martin Eisenmann

fortiss GmbH
An-Institut Technische Universität München



Agenda

Morning

10:00 – 10:45 Welcome

- **Personal Introduction**
- Lets order some Pizza!!!
- Motivation / Introduction MbSE

10:45 – 13:00 AF3

- Installation

-- Pause 15min --

- Introduction to AF3 and its models
- Hands-on: Modeling and Simulation
- Introduction to Technical architecture and deployment
- Hands-On: Deployment, code generation and execution

Lecturers

Sudeep

Marco

Hernan

Thomas

Martin

Participants

Give us a short introduction about yourself

Agenda

Morning

10:00 – 10:45 Welcome

- Personal Introduction
- **Lets order some Pizza!!!**
- Motivation / Introduction MbSE

10:45 – 13:00 AF3

- Installation

-- Pause 15min --

- Introduction to AF3 and its models
- Hands-on: Modeling and Simulation
- Introduction to Technical architecture and deployment
- Hands-On: Deployment, code generation and execution

Lunch: Pepenero

	Kind
Pizza 1	
Pizza 2	
Pizza 3	
Pizza 4	
Pizza 5	
Pizza 6	
Pizza 7	
Pizza 8	
Pizza 9	
Pizza 10	
Pizza 11	
Pizza 12	

	Kind
Pizza 13	
Pizza 14	
Pizza 15	
Pizza 16	
Pizza 17	
Pizza 18	
Pizza 19	

Agenda

Morning

10:00 – 10:45 Welcome

- Personal Introduction
- Lets order some Pizza!!!
- **Motivation / Introduction MbSE**

10:45 – 13:00 AF3

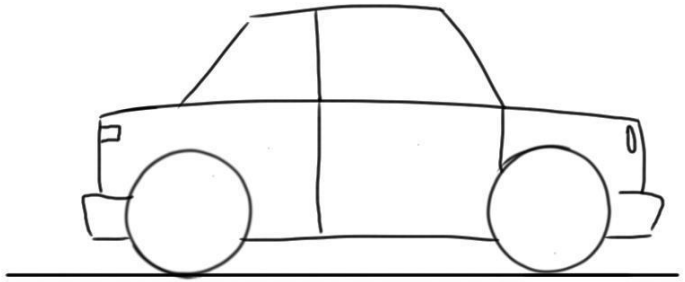
- Installation

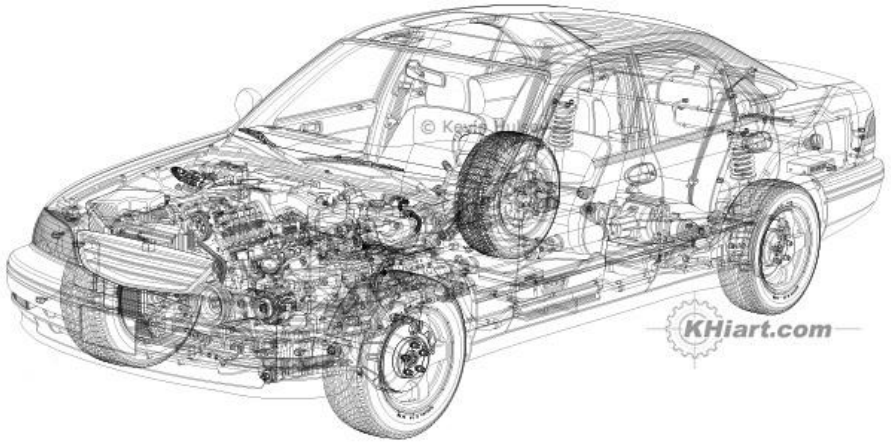
-- Pause 15min --

- Introduction to AF3 and its models
- Hands-on: Modeling and Simulation
- Introduction to Technical architecture and deployment
- Hands-On: Deployment, code generation and execution

What is a model?

A representation of something for a certain purpose





Abstract

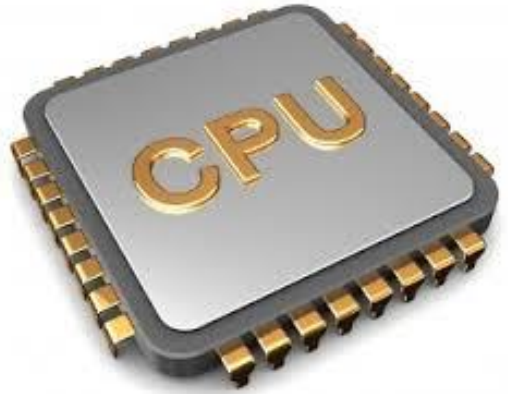
Concrete

Can we model software?

```cpp
// Quellcodebeispiel in C++

#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int alter; // Variable vom Typ Integer

    cout << "Wie alt bist du?";
    cin >> alter;
    cout << "Du bist " << alter << " Jahre alt" << endl;
    getc();
    return 0;
}
```

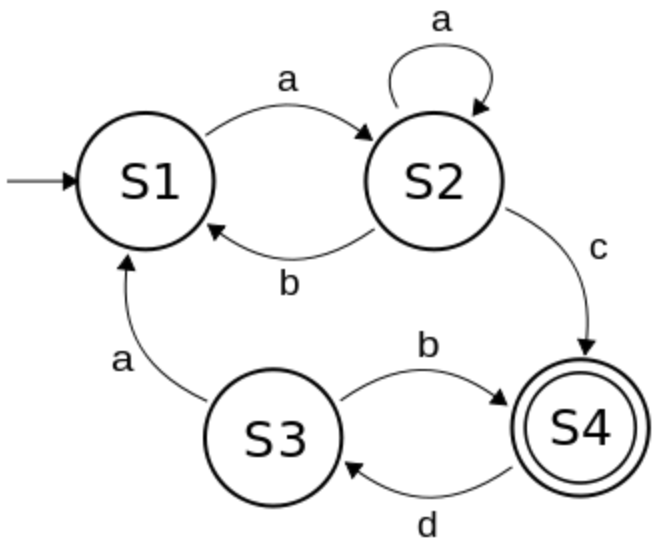
This is a model …

```
LDR r0,[p_a]    ; load a into r0 using pointer to a (p_a)
LDR r1,[p_b]    ; load b into r1
ADD r3,r0,r1    ; compute a + b
STR r3,[p_w]    ; w = a + b
LDR r2,[p_c]    ; load c into r2
ADD r0,r2,r3
STR r0,[p_x]
LDR r0,[p_d]
ADD r3,r2,r0
STR r3,[p_y]
```



… of this!

Can we model software?




```cpp
// Quellcodebeispiel in C++

#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int alter; // Variable vom Typ Integer

    cout << "Wie alt bist du?";
    cin >> alter;
    cout << "Du bist " << alter << " Jahre alt" << endl;
    getc();
    return 0;
}
```

But this, too, is a model …

… of this!

Why use models?

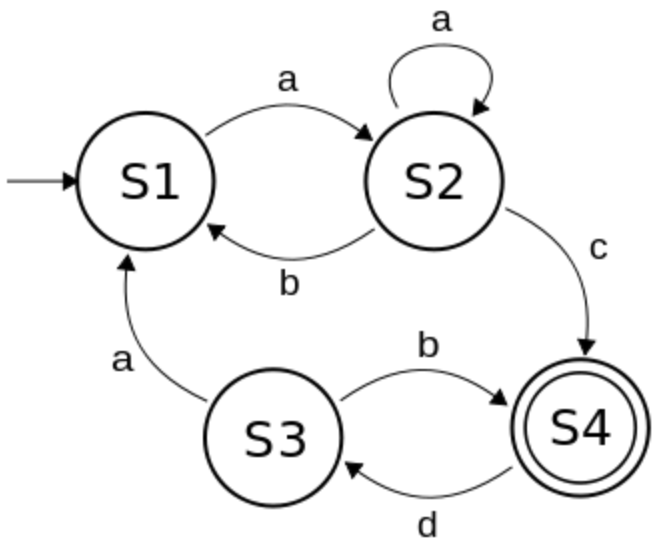
```cpp
// Quellcodebeispiel in C++

#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int alter; // Variable vom Typ Integer

    cout << "Wie alt bist du?";
    cin >> alter;
    cout << "Du bist " << alter << " Jahre alt" << endl;
    getc();
    return 0;
}
```

```
LDR r0,[p_a]    ; load a into r0 using pointer to a (p_a)
LDR r1,[p_b]    ; load b into r1
ADD r3,r0,r1    ; compute a + b
STR r3,[p_w]    ; w = a + b
LDR r2,[p_c]    ; load c into r2
ADD r0,r2,r3    ; compute c + w, reusing r0 for x
STR r0,[p_x]    ; x = c + w
LDR r0,[p_d]    ; load d into r0
ADD r3,r2,r0    ; compute c + d, reusing r3 for y
STR r3,[p_y]    ; y = c + d
```

Would you prefer to write this …

… or this?

Why use models?




```
// Quellcodebeispiel in C++

#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int alter; // Variable vom Typ Integer

    cout << "Wie alt bist du?";
    cin >> alter;
    cout << "Du bist " << alter << " Jahre alt" << endl;
    getc();
    return 0;
}
```

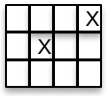
So, why not this …

… instead of this?

Code-based vs. Model-based Development

Textual Requirements (.doc, .xls, ...)

Traceability-matrix

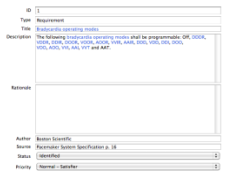


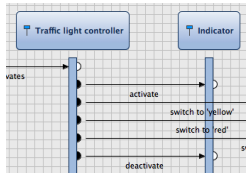
Code (z.B. C, C++)

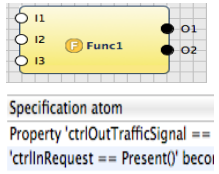
Config.

Hardware

Requirement Models



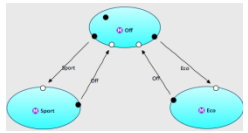


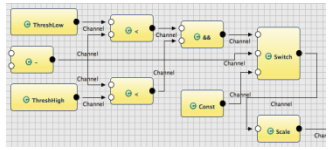


Use-Cases,
MSCs,
Interfaces
Data-Types
Contracts, ...

Consistency Checks

Application Models

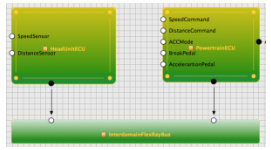




Components,
State-machine,
Code,
Tables, ...

Deployment

Platform Models



ECUs,
BUS,
I/O,
Cores,
Memory

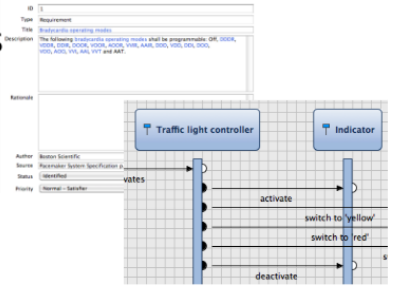
Hardware

fortiss

Advantages of model based development

- Using abstraction to master complexity
- Simplifies re-use
- Enables code generation
- Allows analysis, testing, design-state-exploration at early stages of development

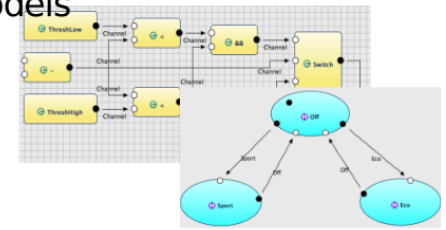
Requirements



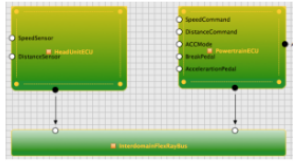
Functions

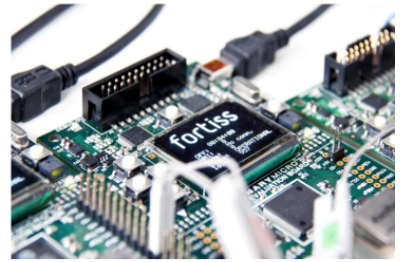
Specification atom
Property 'ctrlOutTrafficSignal ==
'ctrlInRequest == Present()' beco

Logical structure models



Technical HW models





Agenda

Morning

10:00 – 10:45 Welcome

- Personal Introduction
- Lets order some Pizza!!!
- Motivation / Introduction MbSE

10:45 – 13:00 AF3

- **Installation**

-- Pause 15min --

- Introduction to AF3 and its models
- Hands-on: Modeling and Simulation
- Introduction to Technical architecture and deployment
- Hands-On: Deployment, code generation and execution

Installation AF3

- Follow the installation instructions:
  https://af3-developer.fortiss.org/projects/autofocus3/wiki/AF3_Developer_Installation

- Use the setup file from gitlab:
  https://git.fortiss.org/mbse-praktikum-19/dev/blob/master/af3.setup

- Start AF3
  - You might need to “Add required plugins” in your runtime configuration

- Use the model from gitlab:
  https://git.fortiss.org/mbse-praktikum-19/dev/blob/master/models/KickOff-19.af3_23

Agenda

Morning

10:00 – 10:45 Welcome

- Personal Introduction
- Lets order some Pizza!!!
- Motivation / Introduction MbSE

10:45 – 13:00 AF3

- Installation

-- Pause 15min --

- Introduction to AF3 and its models
- Hands-on: Modeling and Simulation
- Introduction to Technical architecture and deployment
- Hands-On: Deployment, code generation and execution

Agenda

Morning

10:00 – 10:45 Welcome

- Personal Introduction
- Lets order some Pizza!!!
- Motivation / Introduction MbSE

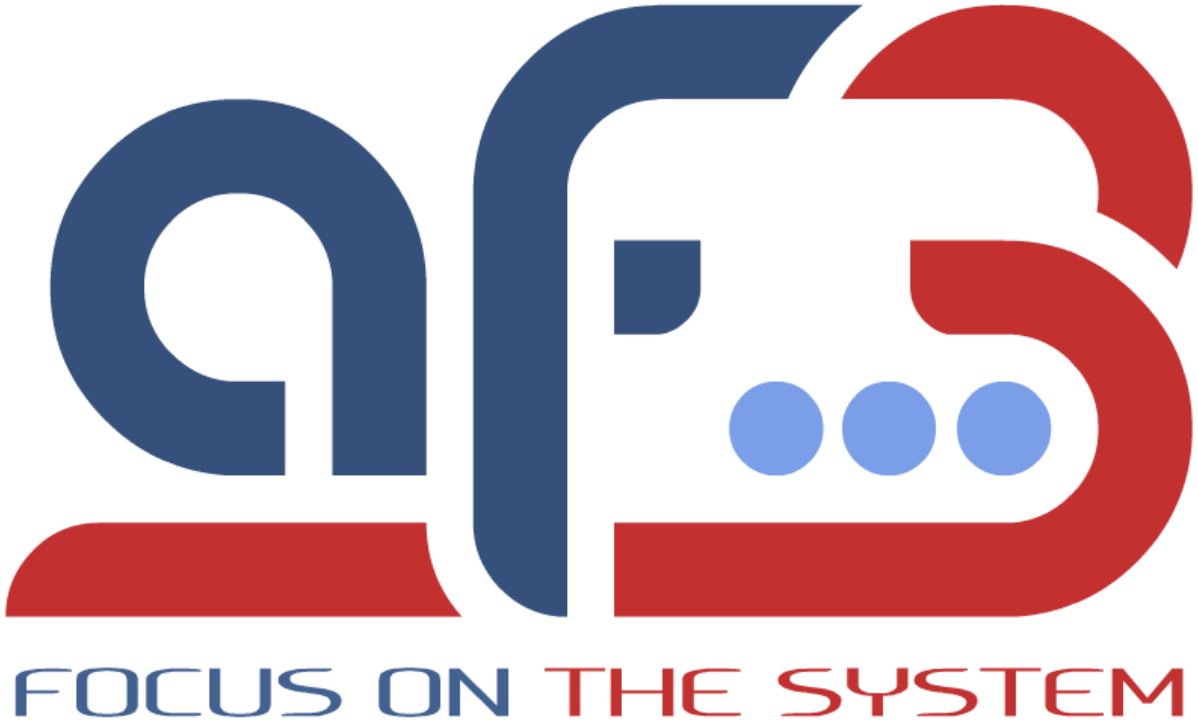
10:45 – 13:00 AF3

- Installation

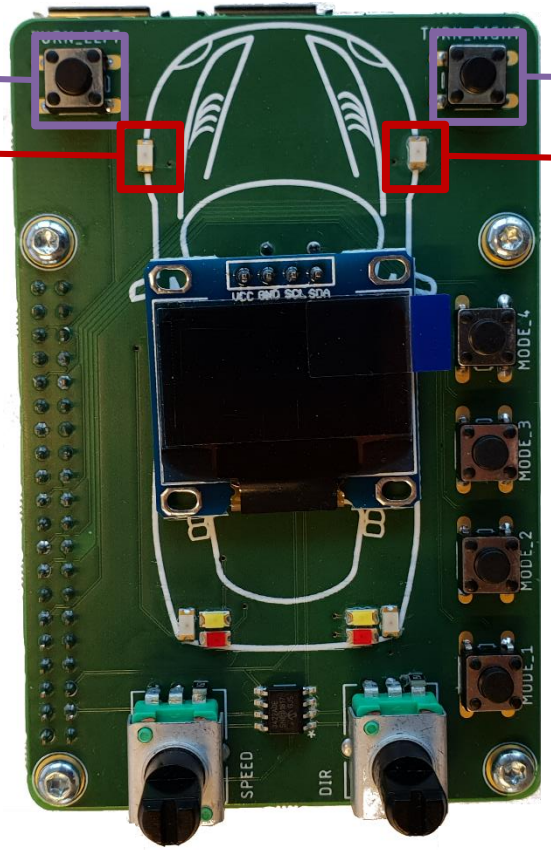
-- Pause 15min --

- **Introduction to AF3 and its models**
- **Hands-on: Modeling and Simulation**
- **Introduction to Technical architecture and deployment**
- **Hands-On: Deployment, code generation and execution**

Introduction to AF3



Requirements for a Blinker



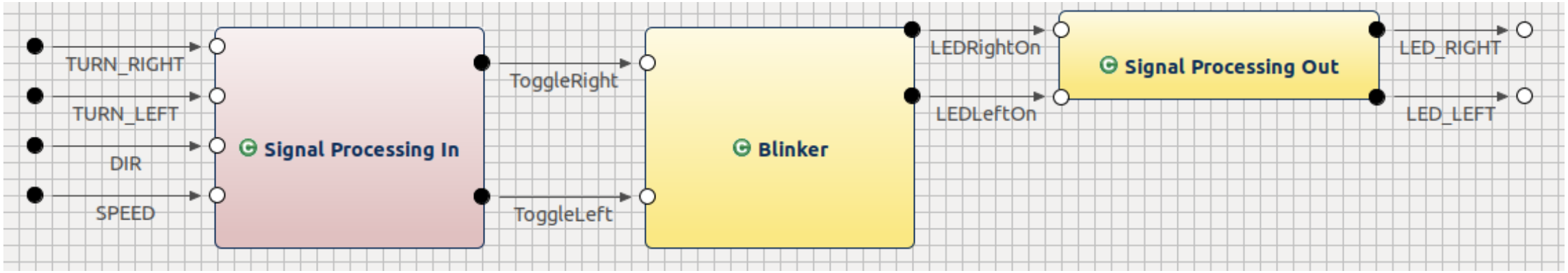
- If the left blinker is not activated, pressing and releasing the TURN_LEFT button shall activate it.
- As long as the left blinker is activated, the LED_LEFT should be on intermittently.
- The same applies for turning right.
- Both blinkers shall not be activated simultaneously.

Modeling the logical Architecture

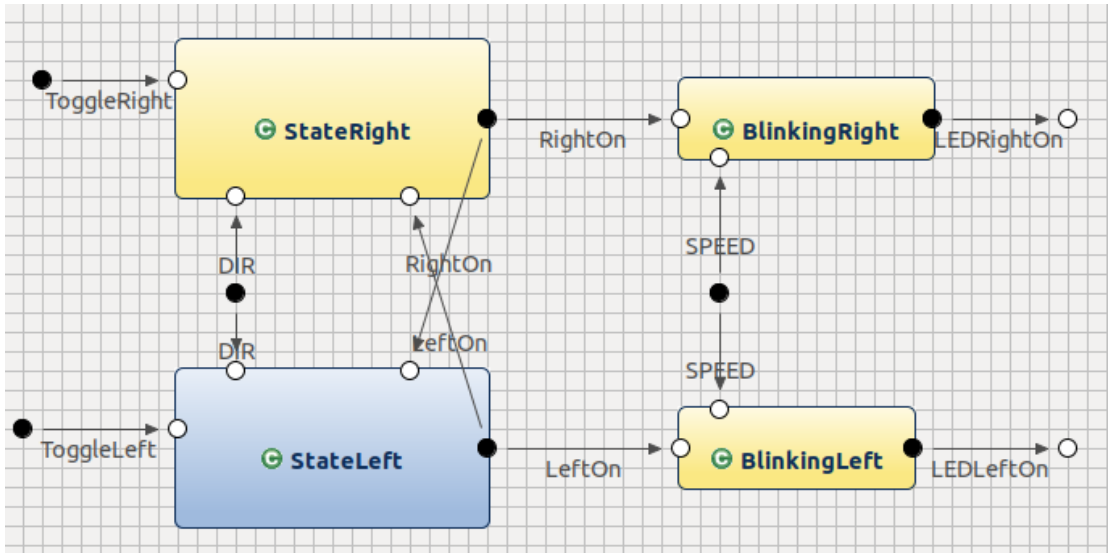
- We focus on a sub-component

▶ - Input signals: ToggleRight, ToggleLeft of type bool

▶ - Output signals: LEDRightOn, LEDLeftOn of type bool



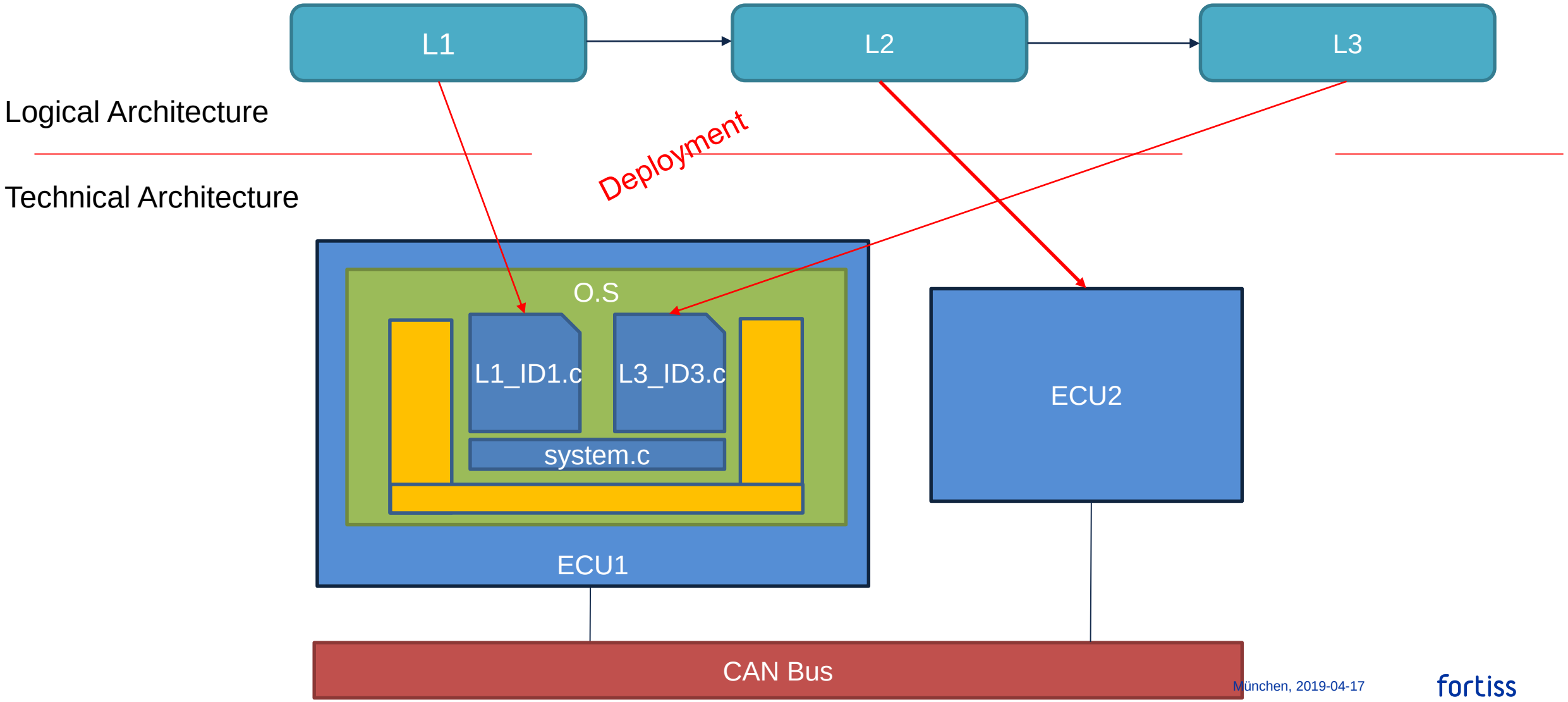
Model Quality



Use the right model for the right purpose: there is a notion of state … use state machines!

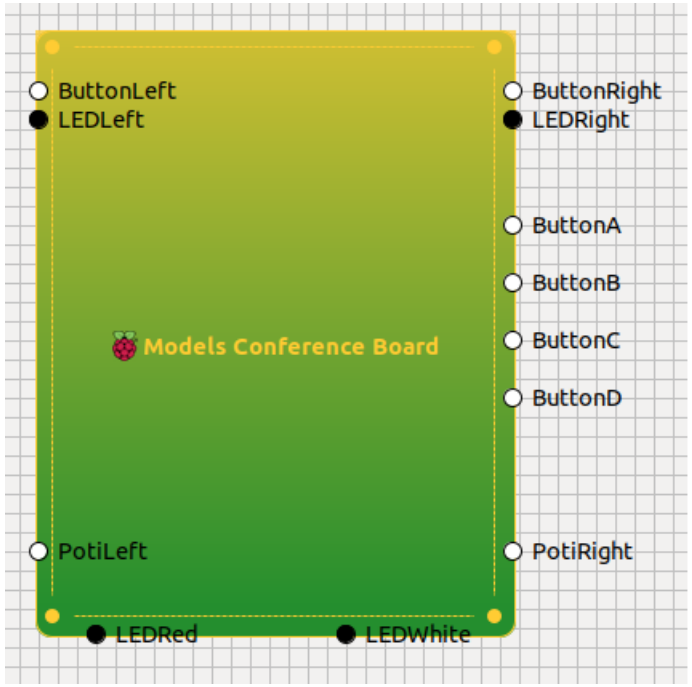
Is your behavior stateless? Maybe is better to use (functional) code

Introduction to Technical Architecture and Deployment



fortiss

Models for Technical Architecture and Deployment



Resulting Mappings

Remove

Logical Input	Hardware Receiver
TURN_LEFT : InputPort	ButtonLeft : Button
TURN_RIGHT : InputPort	ButtonRight : Button
ButtonA : InputPort	ButtonA : Button
ButtonB : InputPort	ButtonB : Button
ButtonC : InputPort	ButtonC : Button
ButtonD : InputPort	ButtonD : Button
SPEED : InputPort	PotiLeft : Poti
DIR : InputPort	PotiRight : Poti

Logical Output	Hardware Transmitter
LED_LEFT : OutputPort	LEDLeft : LED
LED_RIGHT : OutputPort	LEDRight : LED
LEDRed : OutputPort	LEDRed : LED
LEDWhite : OutputPort	LEDWhite : LED

Lunch

Agenda

Afternoon

14:00 – 15:00 Team organization + Team building

15:00 – 15:30 Project planning

- Gitlab
- Team tasks

15:30 – 16:00 Organizational matters and Wrap-Up

- Results
- Reviews
- Events
- Rooms

Grouping

Platooning	Autonomous Parking
1) Runtao Duan 2) Daniel Erler 3) Batuhan Canlıtürk 4) Zhuoling Li	1) Marcel Wagenländer 2) Daniel Svendsen 3) Jannik Peters 4) Julia Pühl 5) Zhenrui Yue 6) Hongtao Zhang

Agenda

Afternoon

14:00 – 15:00 Team organization + Team building

15:00 – 15:30 Project planning

- Gitlab
- Team tasks

15:30 – 16:00 Organizational matters and Wrap-Up

- Results
- Reviews
- Events
- Rooms

Task for each Team

Platooning Team	Autonomous Parking Team
• **ACC** – Integrate ACC into platooning: if the front vehicle breaks, emergency break should be triggered • **Platooning** – Management • Vehicle detection • Direction recognition • Common Interface – Control • Breaks and acceleration	• Enable rover's **autonomous parking**: **1. Sense** in the environment: ➢ an appropriate parking location ➢ possible obstacles **2. Plan** a path: ➢ reaching the parking location ➢ avoiding possible obstacles **3. Act** in the environment: ➢ by moving according to the path planned

Agenda

Afternoon

14:00 – 15:00 Team organization + Team building

15:00 – 15:30 Project planning

- Gitlab
- Team tasks

15:30 – 16:00 Organizational matters and Wrap-Up

- Results
- Reviews
- Events
- Rooms

Results

Joint final report (all participants)

- Documentation of the results
- Challenges / problems / solutions
- Approx. 20 pages

Personal report

- Performed tasks by the participant
- Feedback
- Approx. 2/3 pages

Reviews

Valuation basis

What does **not** count:

- LoC
- Logged time

What counts:

- **Quality** of the generated models / code / documentation
- **Use** of the management tools (gitlab, issues, logged time, ...)
- (Team) **Organisation**
- **Reports** (presentations / final reports)

Weekly Meetings

- Mandatory presence
- Short status / update of the teams
- Discussion of problems / questions during the week
- Coordination between the teams and lecturers
- Short presentation on current topic
- Every Thursday 10:00-13:00
- **Exceptions**
  - 30.05.19 (Ascension Day)
  - 20.06.19 (Corpus Christi)

Short Presentations

- At the beginning of every meeting
- 10 Minutes
- Max. 5 slides
- Mode:
  - Presenter will be informed one week in advance
  - Alternation between teams
  - First presentations 02.05.19

Integration sprints

- Development in sprints according to agile development
- Sprint length: ~3 weeks
- After every sprint, an executable product should exist
- Scheduling sprint goals by participants in weekly meeting after sprint end

- Sprint:
  - **25.04.19 to 16.05.19**
  - **23.05.19 to 13.06.19**
  - **27.06.19 to 18.07.19**

Intermediate presentation at the Chair

- Tuesday **04.06.19** from 15:30 to 17:30

- Mode:
  - Mandatory attendance
  - Presentation of current status
  - Unresolved issues / challenges
  - Planning
  - Demo from the last sprint
  - One presentation per team (lecturer selected by the team)
  - Max 20 min + 10 min questions
  - Max 12 slides

Final presentation at the Chair

- Thursday **25.07.19** from 9:30 to 11:30

- Mode:
  - Mandatory attendance
  - Presentation of the results
  - 5 min per participant + 2 min for question
  - 20 min final demo

Communication

- Mattermost (select login with gitlab)
- https://mattermost.fortiss.org

- Channel
- https://mattermost.fortiss.org/fortiss/channels/mbse-praktikum-19

- Email
- mbse.praktikum@fortiss.org