MODEL BASED SYSTEMS ENGINEERING
CONTINUOUS DEVELOPMENT OF AUTOMATED DRIVING STACKS

Dr.-Ing. Bernd Gassmann
Frederik Pasch

Autonomous Mobile Systems Research Lab
Intel Labs Europe
DISCLAIMER

Intel, the Intel logo, Mobileye, are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

© Intel Corporation
Motivation

• Automated Driving cars will become more complex than any other vehicle before.

• Experience shows that overall system integration and traceability of all requirements is a very difficult task.

• How does such a complex system remain controllable?

• How can the overall development process be simplified?

• How to remain flexible for significant changes in an existing solution?
Our Vision on System Development for Automated Driving

• Use a Model Based Systems Engineering (MBSE) approach
• Stay agnostic to IPC frameworks, operating systems and hardware
• Make heavy use of code generators
• Significantly simplify and reduce efforts in system integration, test, and validation → Automation
• Use Simulation for testing and verification
MODEL BASED SYSTEMS ENGINEERING
Flexibility by Model-Driven Development

• All system components & interfaces are specified in a SysML model
• Automatically build different configurations via code generators
• High flexibility & easy adaptations of the system
From Models to AD Systems

Model
- Functional
  - PE A
  - PE B
  - PE C
  - PE D

Deployment
- PE A
- PE B
- PE C
- PE D

Code Generators
- PIPE
- Middleware Binding
- Build system
- System Startup

AD SW stack
- Application 1
  - PE A
  - PE B
- Application 2
  - PE C
  - PE D

PE: Processing Entity
PIPE: Programming Interface for Processing Entity
Model Complete Automated Driving Software

- **Model**
  - Functional
    - PE A
    - PE B
    - PE C
    - PE D
  - Deployment
    - PE A
    - PE B
    - PE C
    - PE D
- **Code Generators**
- **Sensors**
  - Sensor protocols
  - Camera
  - Radar
  - Lidar
  - USS
  - GPS
  - IMU
- **Perception**
  - Object detection & tracking
  - Lane Detection & tracking
- **Environment modeling**
  - Map server & road model
  - Localization
  - Object fusion
- **Driving behavior**
  - Scene interpretation
  - Behavior prediction
  - Maneuver planning
  - Path planning
  - Trajectory planning
- **Actuators**
  - Actuator access
  - Low level control

**PIPEs**
- Binding (e.g. ROS)

**Framework**
- Operating System
- Execution Unit
- Accelerator

**Platform**
The Processing Entity (PE)

- Key element in the Modeling Language
- Provides a simplistic Programming Interface (PIPEs) which abstract from the underlying IPC middleware without any typecast or copy overhead.
Demonstration

1. Simple Example
   a. Modelling
   b. Code generation
   c. Implementation

2. Browse more extensive model
Outlook

- ROS2
- Improve User Experience
- Generate Documentation from Model