USING IPG CARMAKER IN THE CONTEXT OF CONTINUOUS DEVELOPMENT OF AUTOMATED DRIVING SOFTWARE STACKS

Fabian Oboril, Apply & Innovate
Our Vision: AD - Automated Driving

- A grand challenge towards AD: Verification and Testing
- “Virtual Tests” have to go hand in hand with practical "Field Tests"
- Advantages of development and testing with simulation
  - Low cost
  - Rapid development cycles
  - Closed loop testing on various levels down to Hardware-In-the-Loop
  - Any situation can be covered even dangerous ones

How to use simulation best?
USING CARMAKER WITH AD SOFTWARE
Using CarMaker with an AD stack

Reference Stack

Sensors → Perception → Environment modeling → Driving behavior → Actuators

Communication via network

CarMakerConnection
C++-Library to connect an AD stack to CarMaker via APO

Simulation

Sensor simulation ↔ Environment Model ↔ Vehicle dynamics

Catalog of Scenes and Scenarios
Using CarMaker with an AD stack: Variants

Use flexibility of CarMaker to facilitate development and testing
FLEXIBLE AD SOFTWARE DEVELOPMENT
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:
    - Testing of single components
    - Testing against recorded data
    - SiL or HiL tests to validate the complete AD stack
  - Use CarMaker’s RSI sensors to test perception
  - Use ideal sensors to test behavior planning
Quality by Continuous Software Development

- Code and model are modified gradually
- Every change has to be submitted for a code review
- A continuous integration platform invokes a build of the complete AD stack
  - The stack is tested in a SiL environment against a collection of scenarios
  - Save recordings and debug information
  - Failure → Reject code change & provide recordings
  - Pass → Code change can be accepted
Continuous Testing with CarMaker: Requirements

- Our requirements for continuous testing are:
  - Independence of the simulation tool
    - Support for data recordings & simulation required
  - No user interaction: complete automation
  - Integration in Jenkins
  - Usability in Docker containers for parallelization
    - Headless mode preferable

→ Test framework that uses CarMaker as a client node for simulation
Continuous Testing with CarMaker: Framework

- Every App + Simulation can run on a separate node (PC, server, etc.)
- TCL-Script: Start test, gather results, generate Junit-compatible XML file
Continuous Testing with CarMaker: Jenkins

- Jenkins result integration via Junit XML result files
- Docker containers with Ubuntu 16.04 used for parallelization
  - Evaluating nvidia-docker for Movie support
- Completely automated
- Challenges with remote test setup
  - CarMaker Popups
  - Reduced TestManager features
Continuous Testing with CarMaker: Scenarios

- Scenarios need to cover common sense behavior but also corner cases
- Created a set of scenarios to cover urban driving situations
  - For every scenario a test catalogue is defined
- Challenge: Two types of maps are required
  - AD stack requires map information
  - CarMaker requires a second map for scenario
→ We created a map generator that reads in OSM to ensure alignment
Continuous Testing with CarMaker: Maps
Putting all Together
Summary

- Model-driven software development with CarMaker provides high flexibility
- CI/CD with CarMaker ensures high code quality
- Test framework allows integration in Jenkins and usage of multiple nodes to accelerate execution
- Map generator simplifies scenario creation