



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

Fabian Oboril, Apply & Innovate

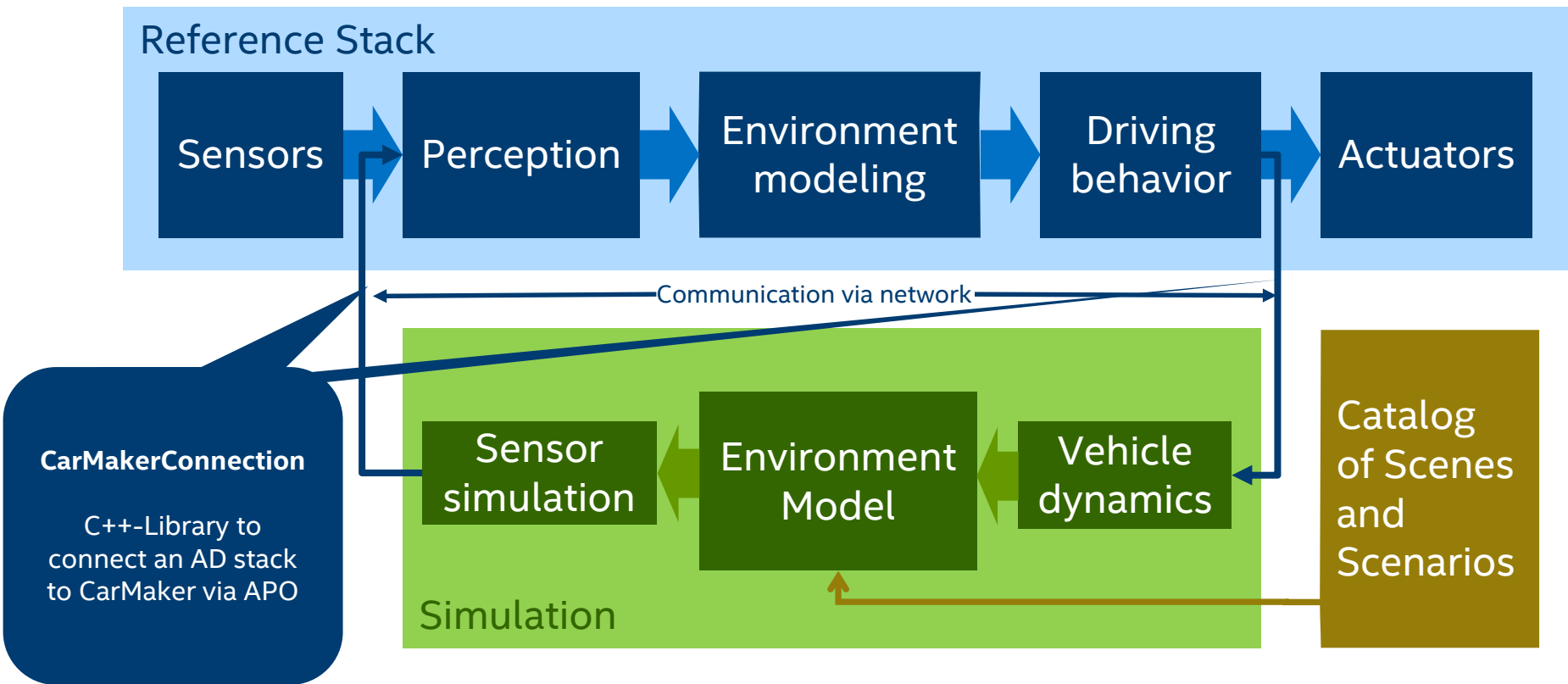
# Our Vision: AD - Automated Driving

- 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

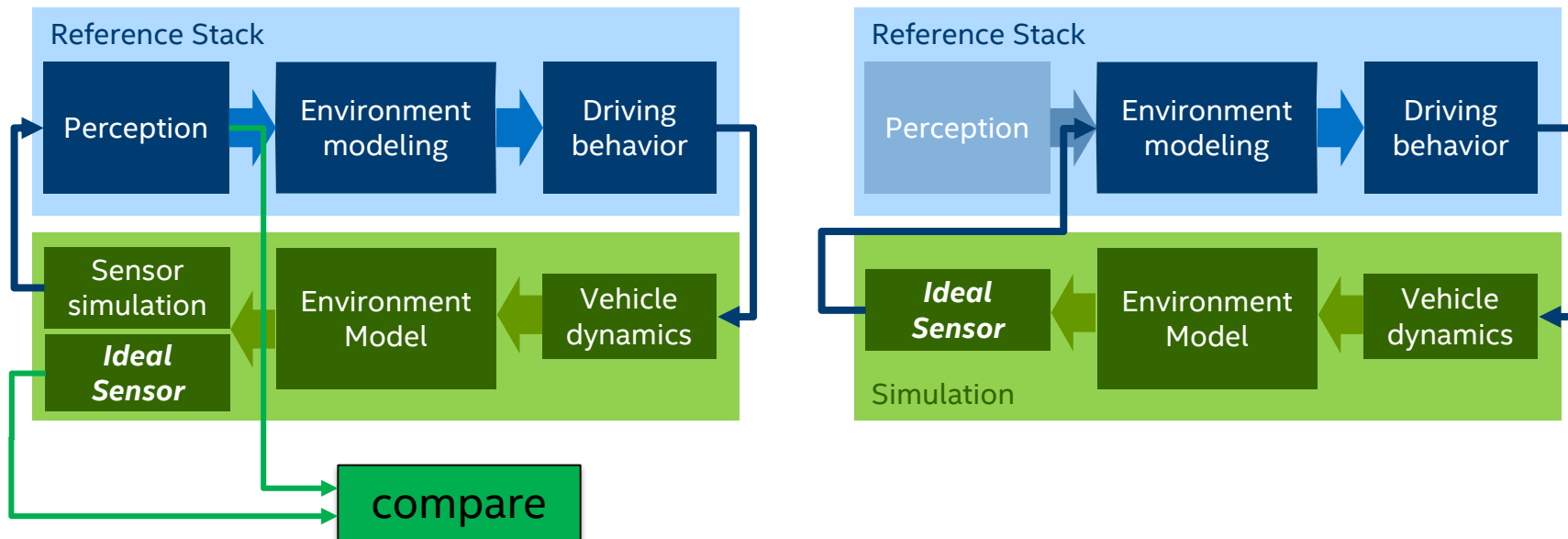
How to use simulation best?

# USING CARMAKER WITH AD SOFTWARE

# Using CarMaker with an AD stack



# Using CarMaker with an AD stack: Variants

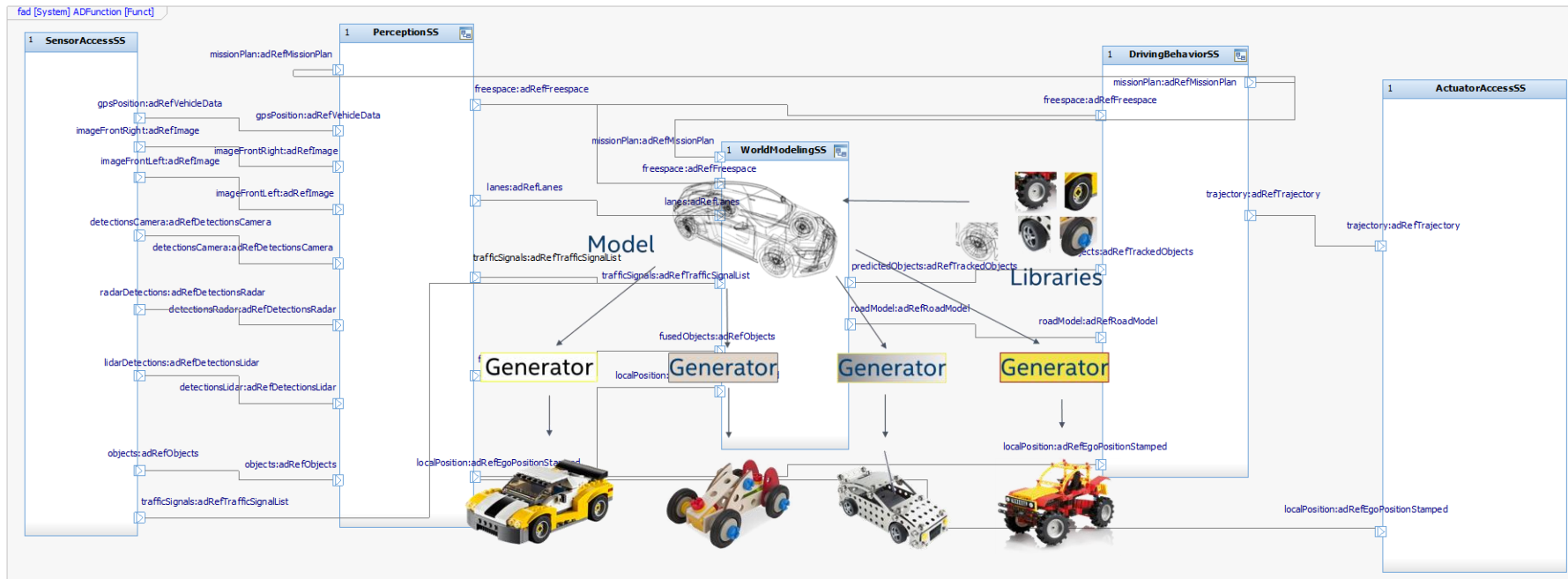


Use flexibility of CarMaker to facilitate development and testing

# FLEXIBLE AD SOFTWARE DEVELOPMENT

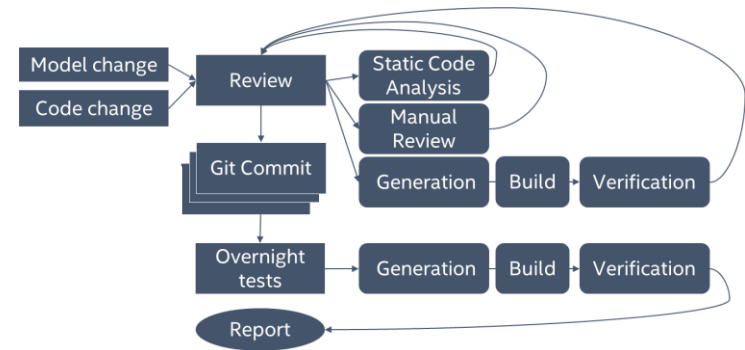
# Flexibility by Model-Driven Development

- All system components & interfaces are specified in a SysML model



# 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

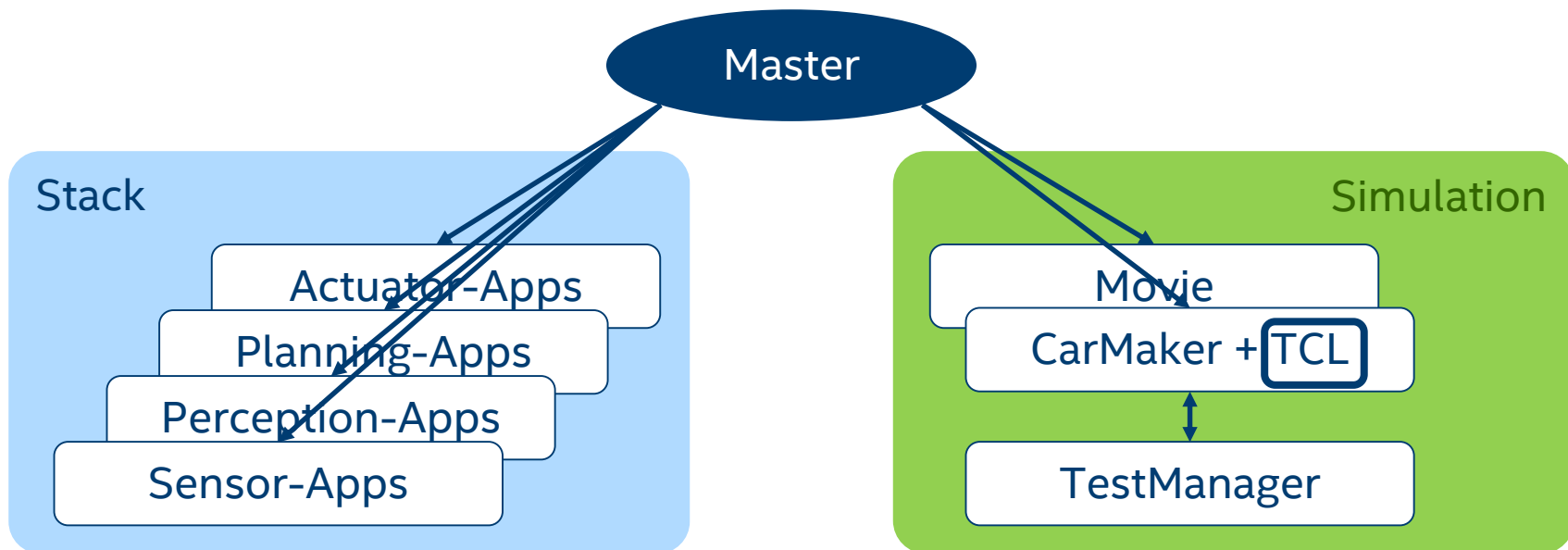




# Continuous Testing with CarMaker: Requirements

- 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

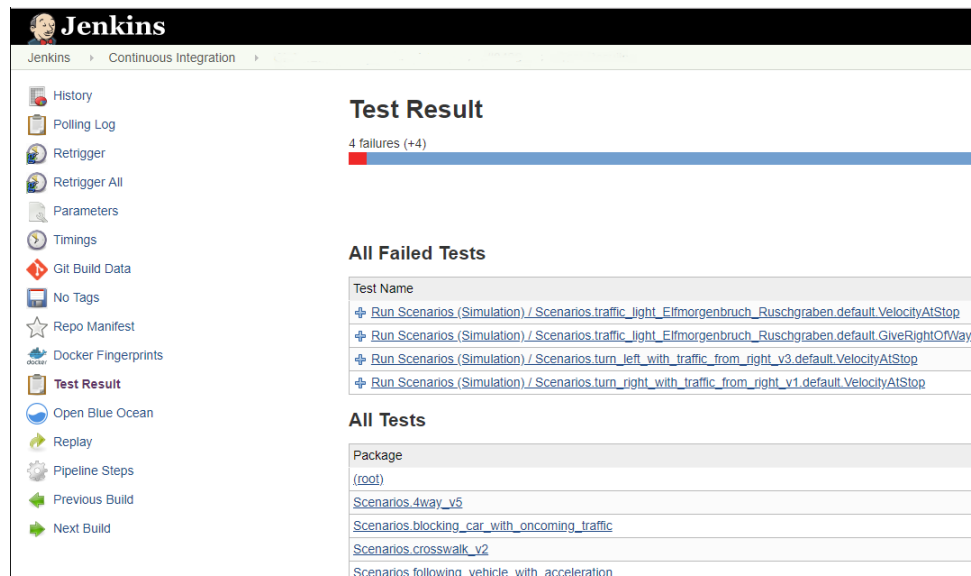
# Continuous Testing with CarMaker: Framework



- 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

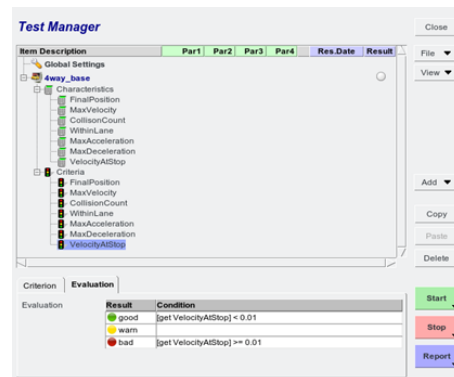


The screenshot displays the Jenkins web interface for a 'Test Result' page. The breadcrumb navigation shows 'Jenkins > Continuous Integration > Test Result'. The left sidebar contains various navigation options: History, Polling Log, Retrigger, Retrigger All, Parameters, Timings, Git Build Data, No Tags, Repo Manifest, Docker Fingerprints, **Test Result** (selected), Open Blue Ocean, Replay, Pipeline Steps, Previous Build, and Next Build. The main content area is titled 'Test Result' and shows '4 failures (+4)' with a red progress bar. Below this, the 'All Failed Tests' section lists four failed test names, each with a plus icon for expansion. The 'All Tests' section lists several test packages, including '(root)', 'Scenarios.4way\_v5', 'Scenarios.blocking\_car\_with\_oncoming\_traffic', 'Scenarios.crosswalk\_v2', and 'Scenarios.following\_vehicle\_with\_acceleration'.

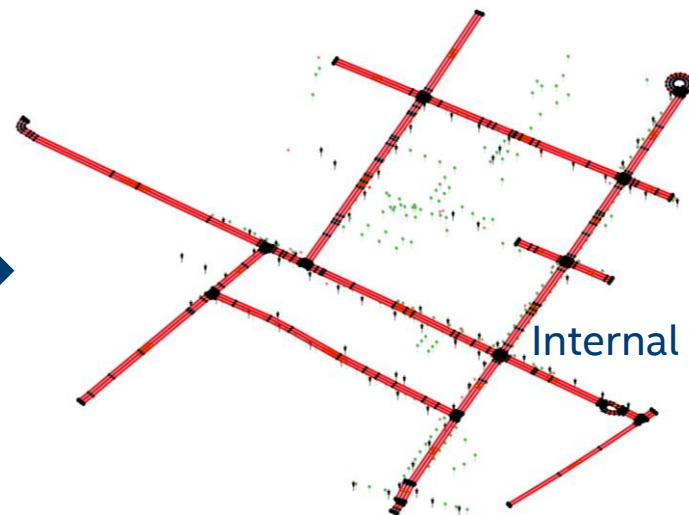
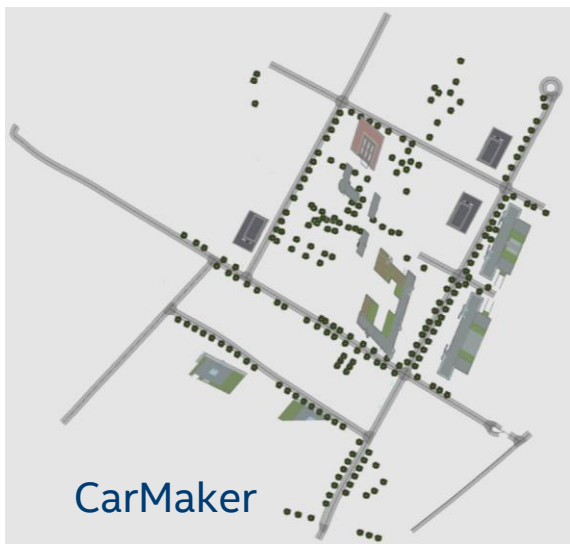
# Continuous Testing with CarMaker: Scenarios

- 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

