Simulation Framework for Executing Component and Connector Models of Self-Driving Vehicles

Talk at EXE 2017 (Austin)

18.09.2017

Filippo Grazioli, Evgeny Kusmenko, Alexander Roth,
Bernhard Rumpe, Michael von Wenckstern

Software Engineering
RWTH Aachen
http://www.se-rwth.de/

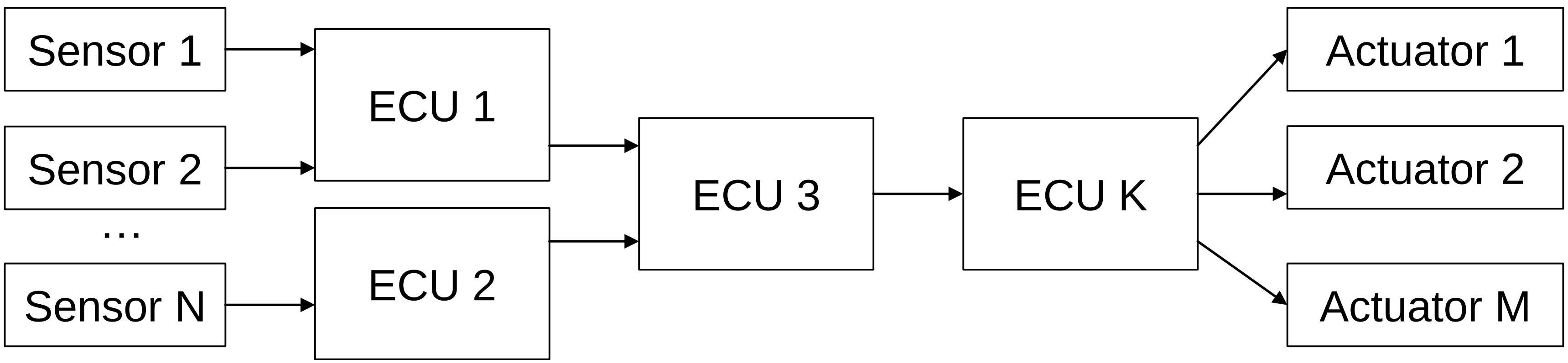
Evgeny Kusmenko

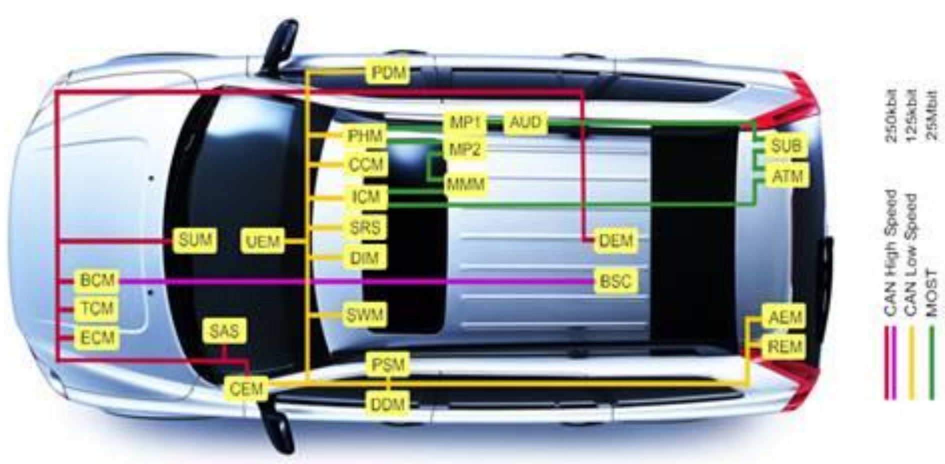
Software Engineering
Chair (Bernhard Rumpe)

RWTH Aachen University

Our Mission

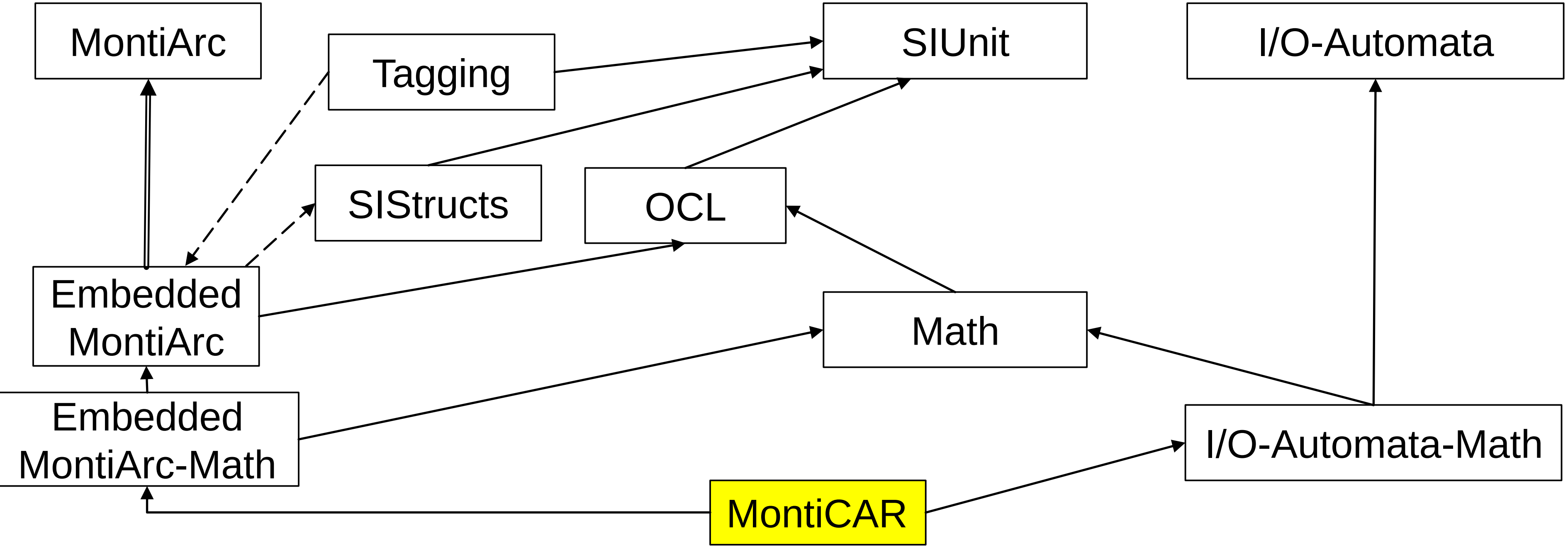
Provide better component & connector methodologies for model-based design of cyber-physical systems





MontiCAR – Main Characteristics

- Textual C&C ADL
- MontiArc + domain specific concepts
- Simulink-compatible semantics and timing (weakly causal)
- Strong Type-System
- Generation to C++-Code
- Compile-time checks and verification
- Optimization based on type information
- Extensibility



Evgeny Kusmenko
Software Engineering
Chair (Bernhard Rumpe)
RWTH Aachen University

MontiCAR Domain Specific Concepts

The following Requirements for Cyber Physical Systems are satisfied:

- No support of datatypes which may overflow (e.g. `String, List`)
- Complete Unit Support
- Supporting (even multiple) Ranges for Sensors, Actuators
- Each Range can has its own Accuracy
- Support of Component and Port Arrays
  - Length of Port Array can be even a generic (higher reuse of components)

EMA

```
component AutomatedVehicle {
  ports in GPS posCar,
                                                port name      port array size
                   Port type
           Q(0.01m:0.01m:4.2m) distance[10],...
        out Z(0N:1N:200kN) brakeForce[4];}
```

MontiCAR – Math for Behavior

Of great interest for cyber-physical systems:

- MATLAB like language for behavior specification
- Advanced Matrix-Vector Support with very powerful Matrix-Type-System
  - Errors can be found at compile/generate time (not at runtime as it is the case in Matlab if matrix dimensions do not fit)
  - Strong Type System allows efficient computations based on Matrix properties (e.g. sparse or full matrix)

- `Q(0m:10m)^{1,10} distance;` (row vector definition)
- `diag inv Q(0:1)^{10, 10} facMatrix = …;` (type is a diagonal invertible 10x10 rational matrix which elements are between 0 and 1)
- `distVector = distance*facMatrix;` (Matrix-Matrix-Multiplication)
- `min(distVector);` (returning the smallest element of the vector)

Evgeny Kusmenko

Software Engineering
Chair (Bernhard Rumpe)

RWTH Aachen University

Simulators / Case Studies

- TORCS
  - Deep Learning Direct Perception Control
  - Evolutional Controller Tuning

- Gazebo / ROS
  - PID based controllers
  - Distributed vs Centralized Control

- SUMO + Veins
  - Scenarios of Cooperative Driving

- VDrift / OpenDaVinci
  - End2End Learning

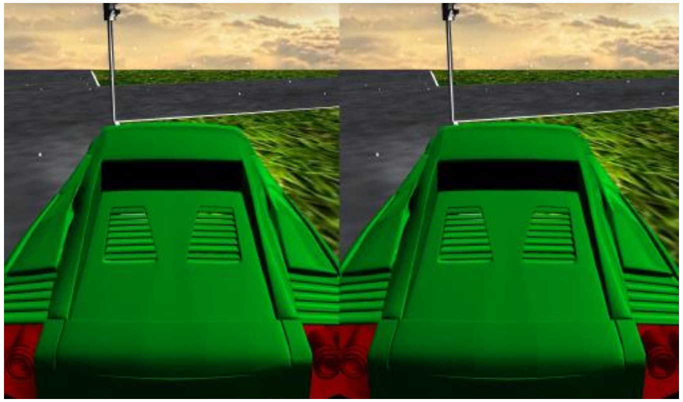
- Many others

Requirements

- (R1) Import and reuse of existing real world environment data.
- (R2) Capability to simulate large-scale everyday scenarios, e.g., different traffic densities, light and weather conditions.
- (R3) Support for realistic and extensible car models with sensors, controllers and actuators.
- (R4) Multi-platform and portable devices support.
- (R5) Automated support for continuous integration and regression testing.
- (R6) Simulator should contain a physics engine.
- (R7) 3D visualization for demonstration purposes.

MontiSim – Main Features

- Browser based 3D visualization
  - Simulator: Java
  - Visualization: JavaScript / ThreeJS
  - Enables CV + ML capabilities

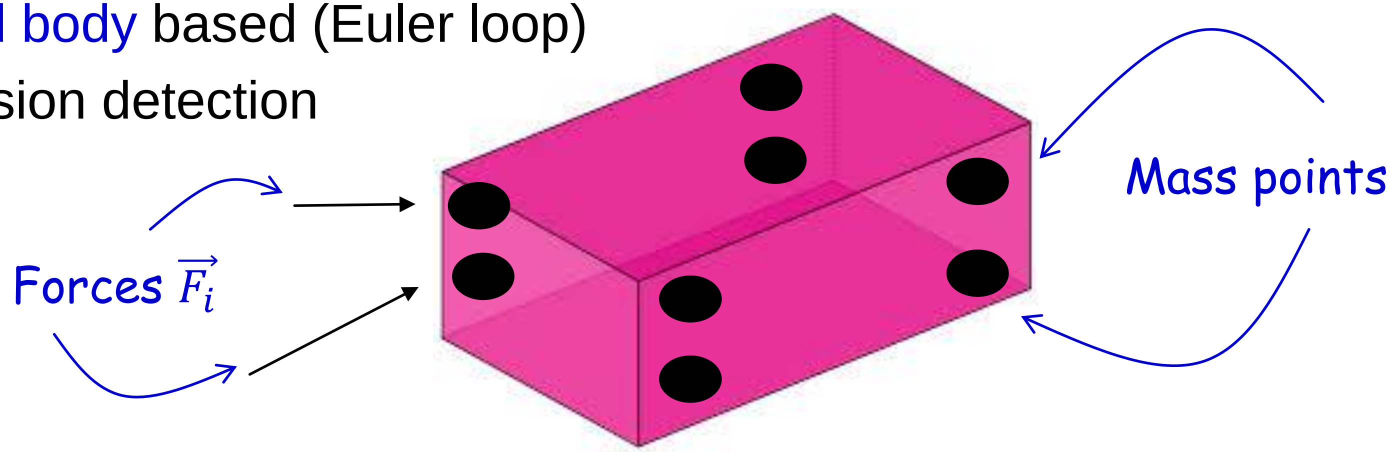


- Environment model
  - OpenStreetMap
  - Probablistic models for pedestrian behavior
  - Weather effects (e.g. changing the friction coefficient)

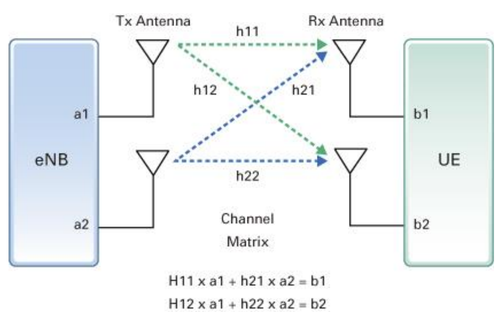
Evgeny Kusmenko
Software Engineering
Chair (Bernhard Rumpe)
RWTH Aachen University

MontiSim – Main Features (2)

- Physics engine
  - Discrete time
  - Rigid body based (Euler loop)
  - Collision detection

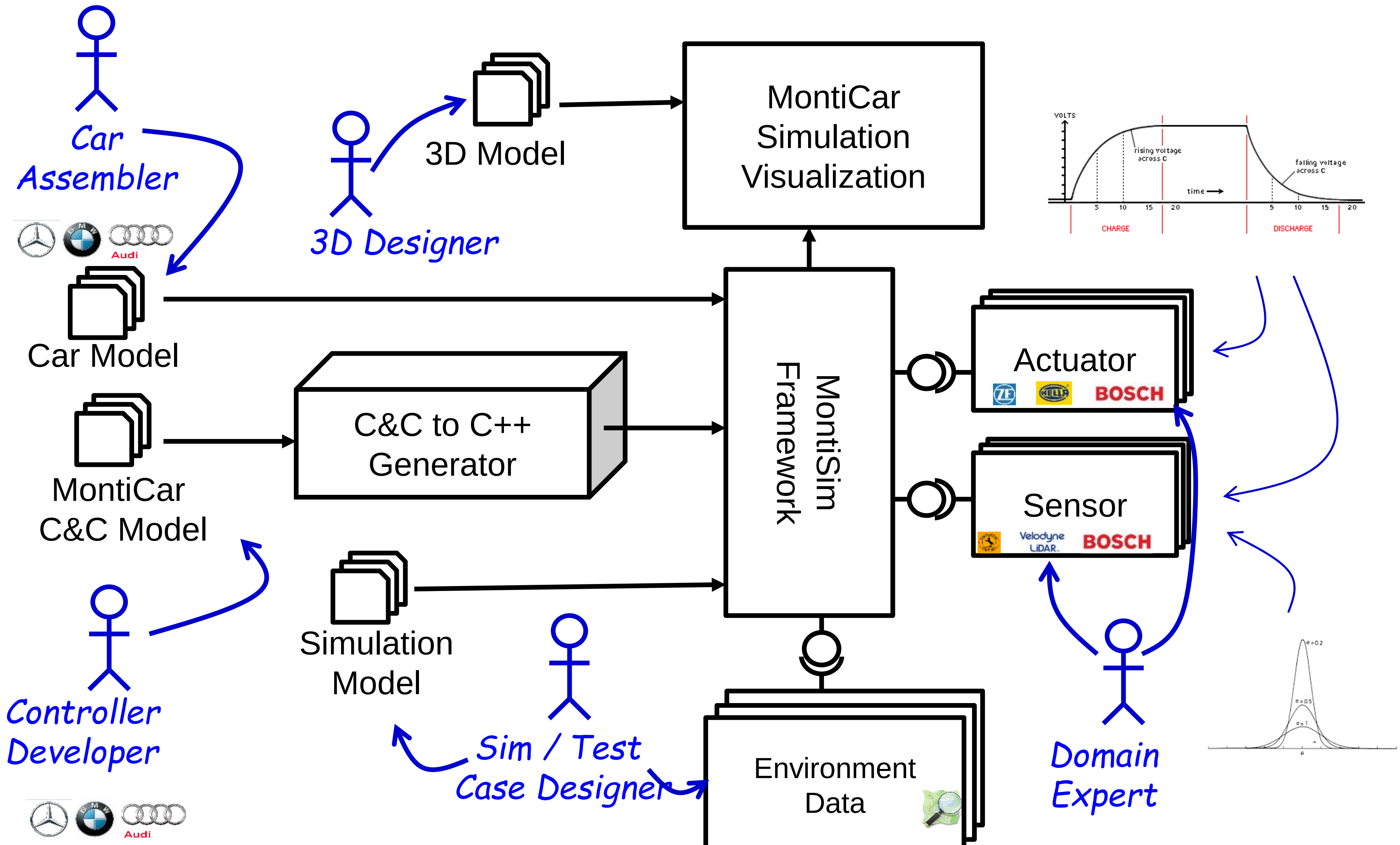


- Simulator coupling
  - e.g., for Vehicle-to-Vehicle (V2V) communication



Evgeny Kusmenko
Software Engineering
Chair (Bernhard Rumpe)
RWTH Aachen University

MontiSim - Framework Overview



MontiSim - Simulation Models

- What do we want to simulate?
  - Number of vehicles
  - Goals
  - Map
  - Duration
  - Resolution
  - …

Sim

```
simulation Example1 {
  map = AachenCity.osm;
  startTime = 22.06.2017 13:30;
  deltaT = 1ms
  wheather = noRain, noSnow;
  duration = 30s;
  cars {
    AC-SE001:50°46'43.7"N 6°03'38.6"E ->
             50°46'49.7"N 6°04'32.5"E,
    M-SE003: … -> … }}
```

Start Position

Target Position

- Is the model consistent?
- Are there type errors?

Evgeny Kusmenko
Software Engineering
Chair (Bernhard Rumpe)
RWTH Aachen University

MontiSim – Car Models

- Car models need to be specified precisely
  - Shape
  - Sensors
  - Actuators
  - …
- Needs to be consistent with the simulation model

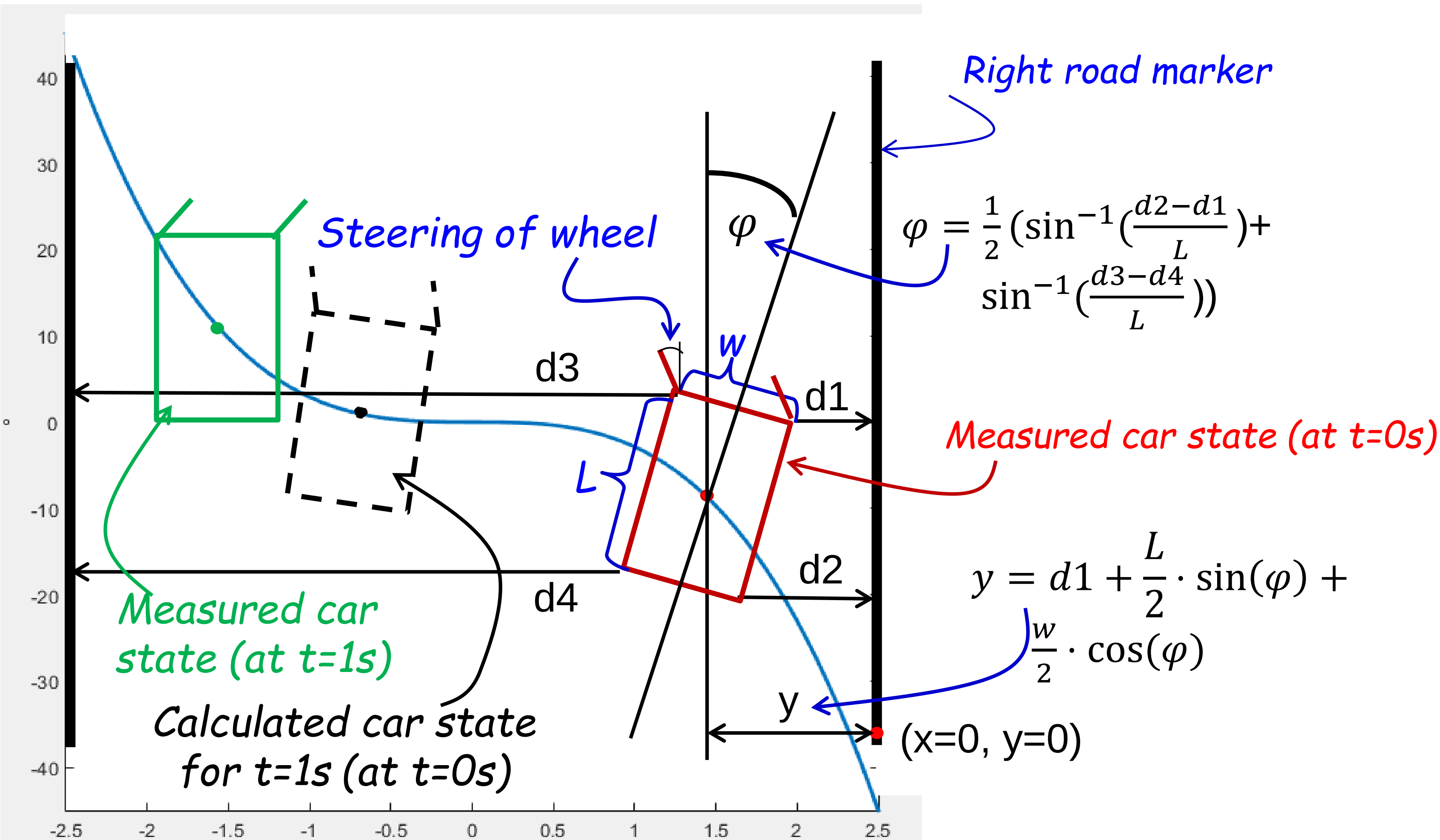
Car

```
car AC-SE001 {
  dimension = 4.43m,1.93m,1.25m;
  visualModel = R8Red.json;
  weight = 1`655 kg;
  controller =
     LaneKeepingController;
  sensors {
    SpeedSensor => velocity;
    TiHighAccGPS => position;
    Compass => direction;     }
  actuators {
    steering => SteeringFIR4; }}
```

Controller's input port

Controller's output port

Model Predictive Lane Keeping Controller



$$\varphi = \frac{1}{2}\left(\sin^{-1}\left(\frac{d2-d1}{L}\right) + \sin^{-1}\left(\frac{d3-d4}{L}\right)\right)$$

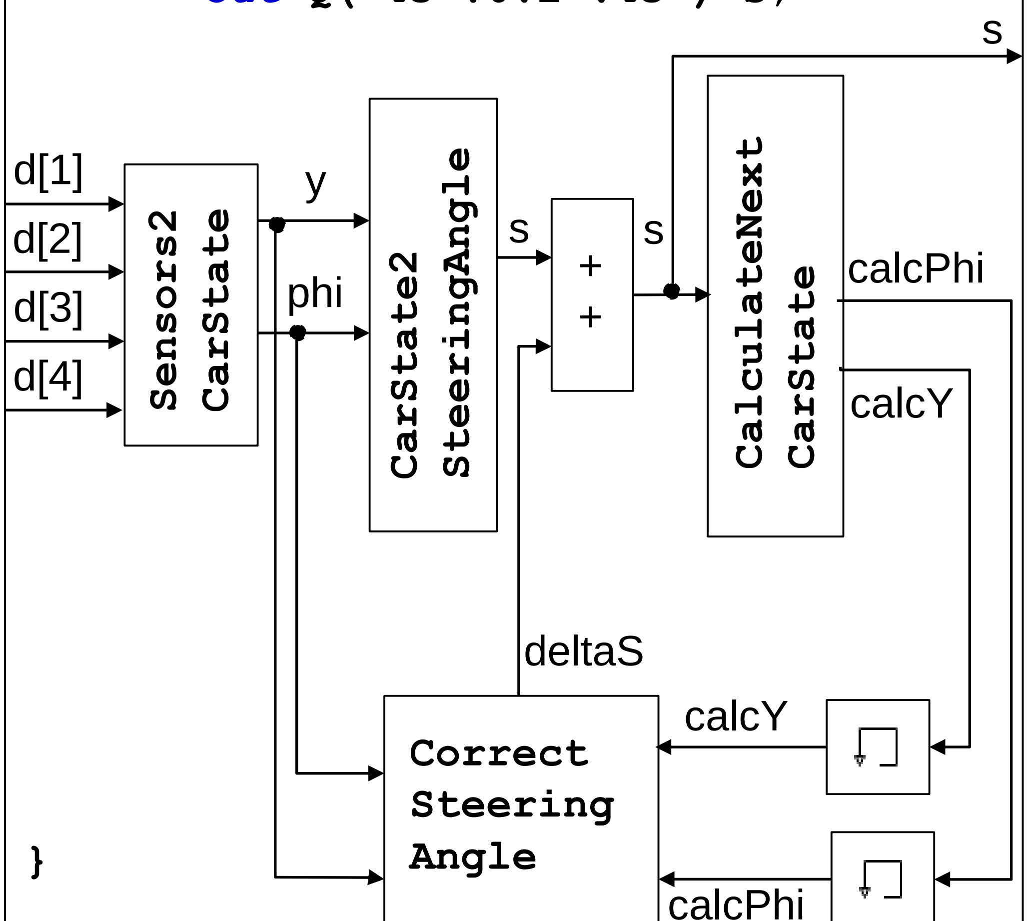
$$y = d1 + \frac{L}{2}\cdot\sin(\varphi) + \frac{w}{2}\cdot\cos(\varphi)$$

Evgeny Kusmenko
Software Engineering
Chair (Bernhard Rumpe)
RWTH Aachen University

Lane Keeping Controller - MontiCAR

EMA

```
component LaneKeepingController {
  ports in Q(0m:0.1m:5m) d[4],
        out Q(-45°:0.2°:45°) s;
```




```
}
```

EMA

```
component Sensors2CarState
 (Q(0m:20m) L, Q(0m:2.4m) w) {
  ports in Q(0m:0.1m:5m) d[4],
      out Q(-2.5m:2.5m) y,
          Q(0:360°) phi;
  implementation Math {
    phi=0.5(asind(((d(2)-d(1))/L)+
             asind(((d(3)-d(4))/L));
    y=saturate(
          d1+ L/2*sin(phi)+
              w/2*cos(phi),
          -2.5m, 2.5m);    }    }
```

EMA

```
component CarState2SteeringAngle {
ports in Q(-2.5m:0.1m:2.5m) y,
            Q(0:0.1:360°) phi,
        out Q(-45°:0.02°:45°) s;
  implementation Math {
    const a = -2.88°/m^3;
    s=saturate(phi + a*y^3,
      -45°, 45°);        }     }
```

Evgeny Kusmenko
Software Engineering
Chair (Bernhard Rumpe)
RWTH Aachen University

Automated Testing

- Model based testing using MontiCAR Stream language
- Each component can be tagged with stream tests
- No need to know anything about the generated code
  - Saves a lot of boiler plate code

Stream

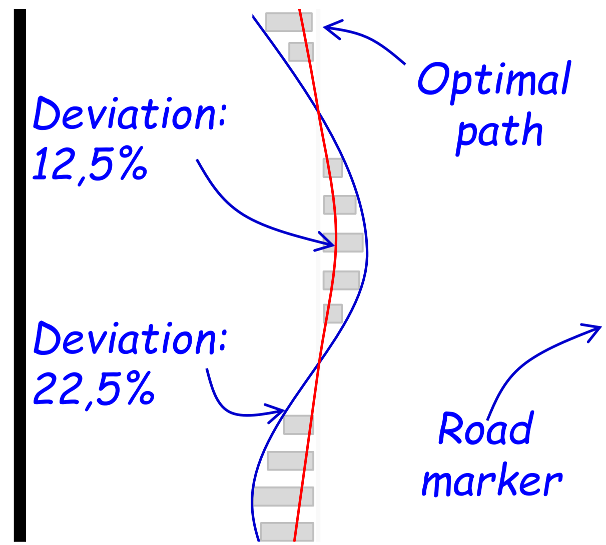
```
stream StearingAngleTest
  for Sensors2CarState
      (L=3m, w=2m) {
  d = [50cm 50cm 2.5m 2.5m]
   tick [50cm 1.3m 2.6m 1.8m];
  phi = 0° tick 15.45°+/-0.05°;
  y = - tick -;
}
```

Values for input port d

time step

Expected Values for output ports

15.4° ≤ exp. value ≤ 15.5°



The end

Thank you for your attention!