Technische Universität Braunschweig

Institut für Regelungstechnik

IDA INSTITUTE OF COMPUTER AND NETWORK ENGINEERING

Towards model-based integration of component-based automotive software systems

Johannes Schlatow, Mischa Möstl and Rolf Ernst,

Marcus Nolte, Inga Jatzkowski and Markus Maurer

Oct. 30, 2017

Motivation

Driving factors

- ADAS / automated driving
- complex high-performance architectures (Audi zFAS, NVIDIA Drive PX)
- complex software systems
  → **component-based/service-oriented design**
- in-field **deployment** of changes (features, updates)

[Source: NVIDIA]

Challenges

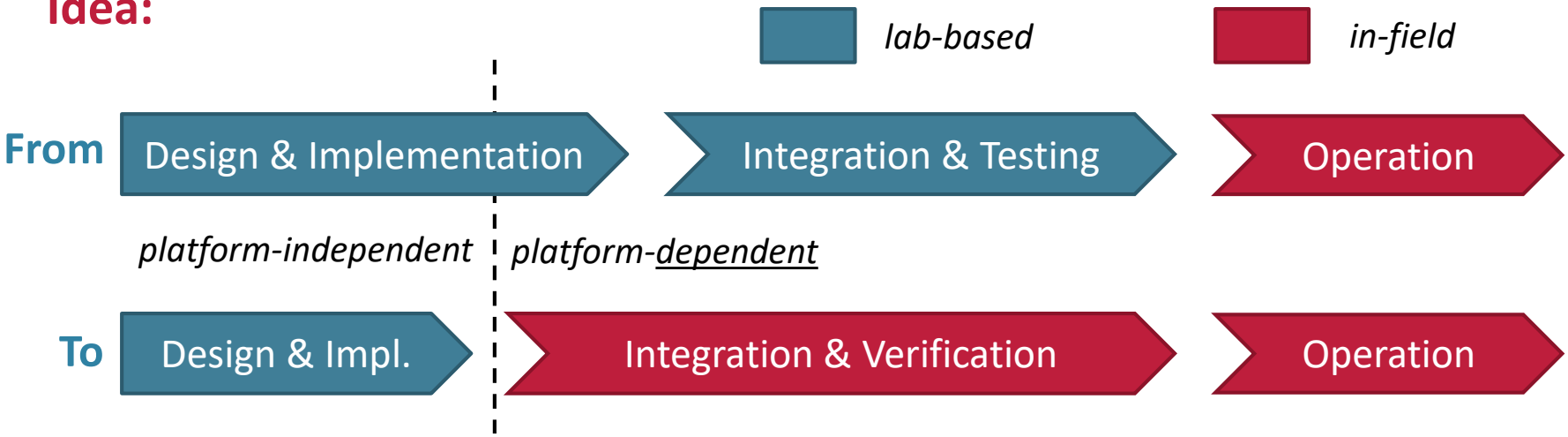
- non-functional requirements (e.g. safety) cannot be verified on interface basis alone (**non-composability**)
- in-depth understanding of different system layers
- automation/tooling → **model-based integration**

Background: Controlling Concurrent Change (CCC)

- research project at TU Braunschweig funded by DFG (Germany Research Foundation), 6yrs
- application scenarios: **ADAS** and **space robots**

Controlling Concurrent Change
http://ccc-project.org

Idea:



This talk…

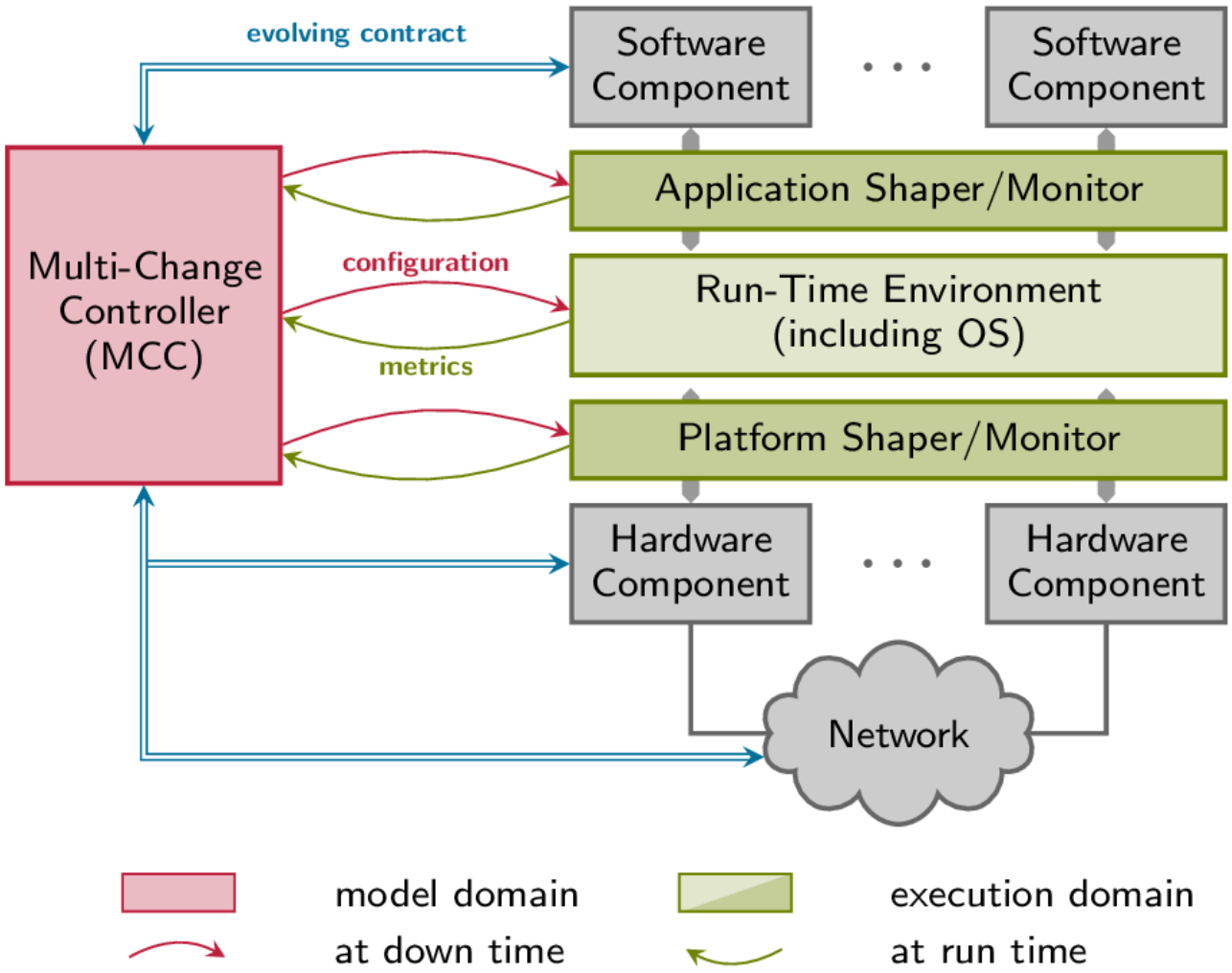
Update problem:

“Here, I have a new functionality for you.

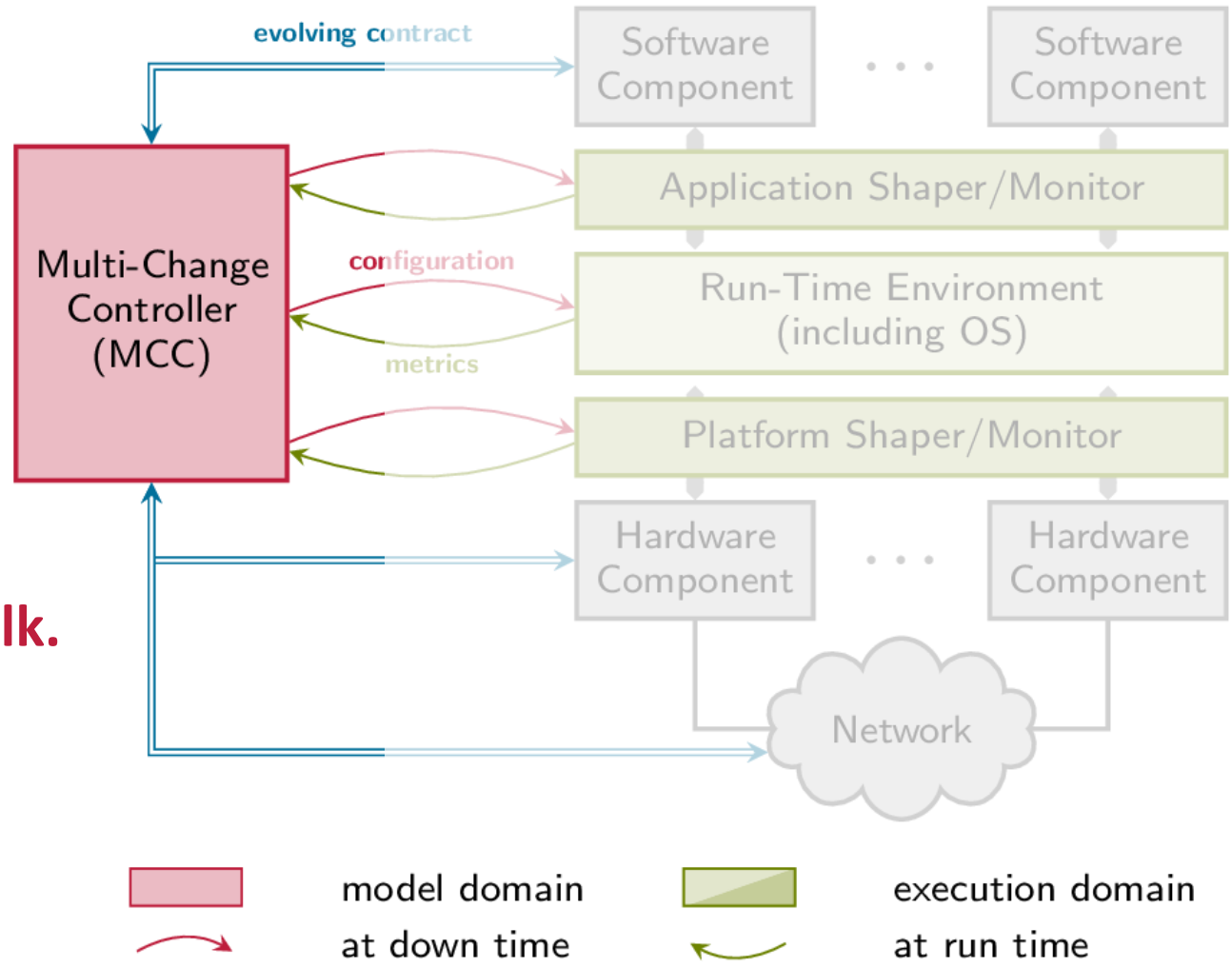
Can you please install it on your platform?”

Photo: Daimler & Benz Foundation, Oestergaard

CCC architectural approach



CCC architectural approach



This talk.

Model domain and Multi-Change Controller

Given:

- component repository (incl. models, requirements, dependencies, etc.)
- update query

Wanted:

- a corresponding system configuration
- s.t. requirements and constraints (safety, real-time, etc.)

Modelling challenges:

- What components can I use (are used) to implement this functionality?
- In what functionality is this component involved?
- What is the sensing-to-actuation delay for XY?
- Are functions A and B sufficiently independent?
- …

➔ **multi-layered and graph-based modelling approach**

Cross-Layer Integration

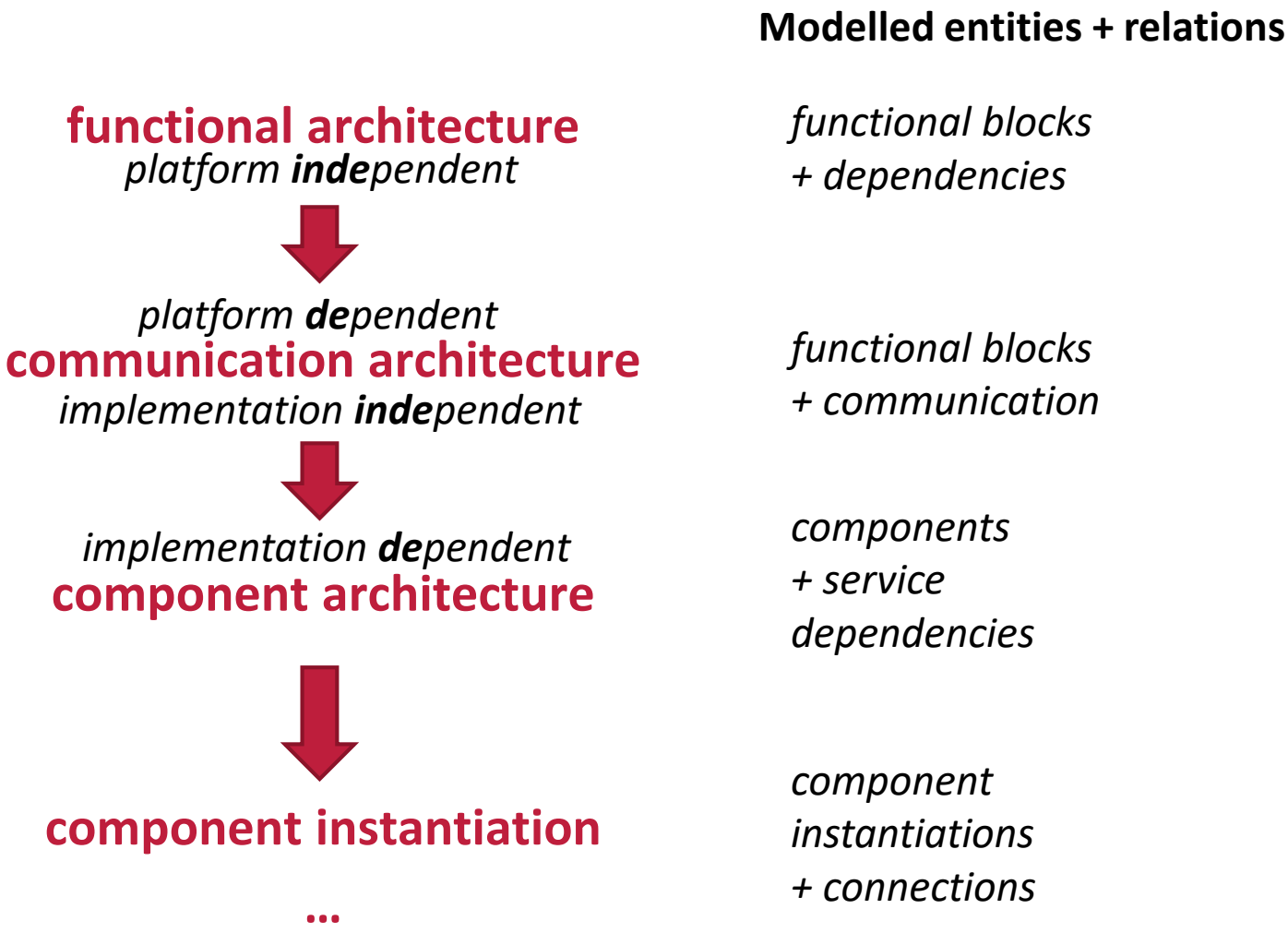
- layer = directed graph
- integration = graph transformations

Basic transformations

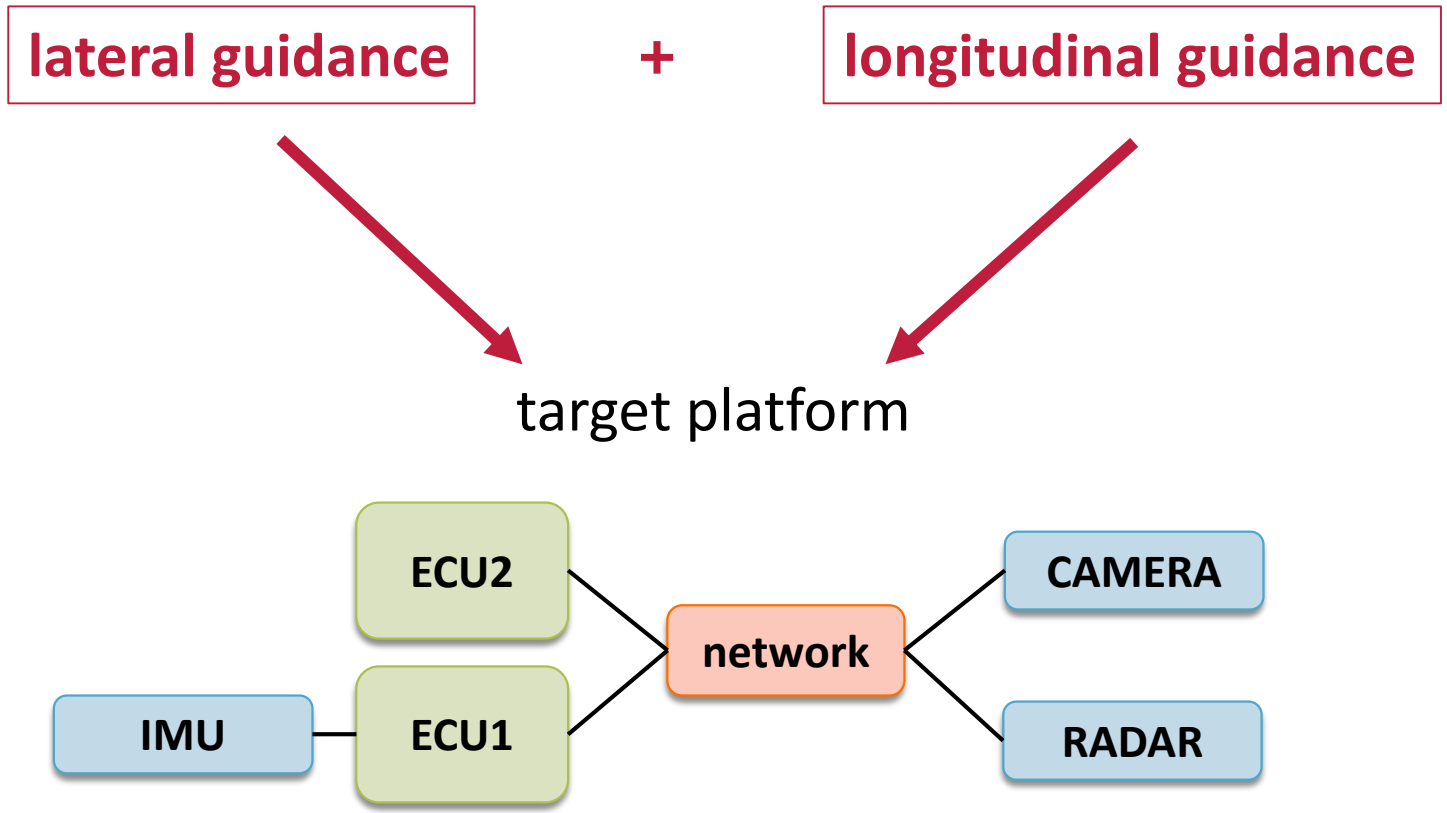
- Arc splitting (Def. in paper)

- Pattern-based transformation (Def. in paper)

Layer overview



Example: inertial navigation system

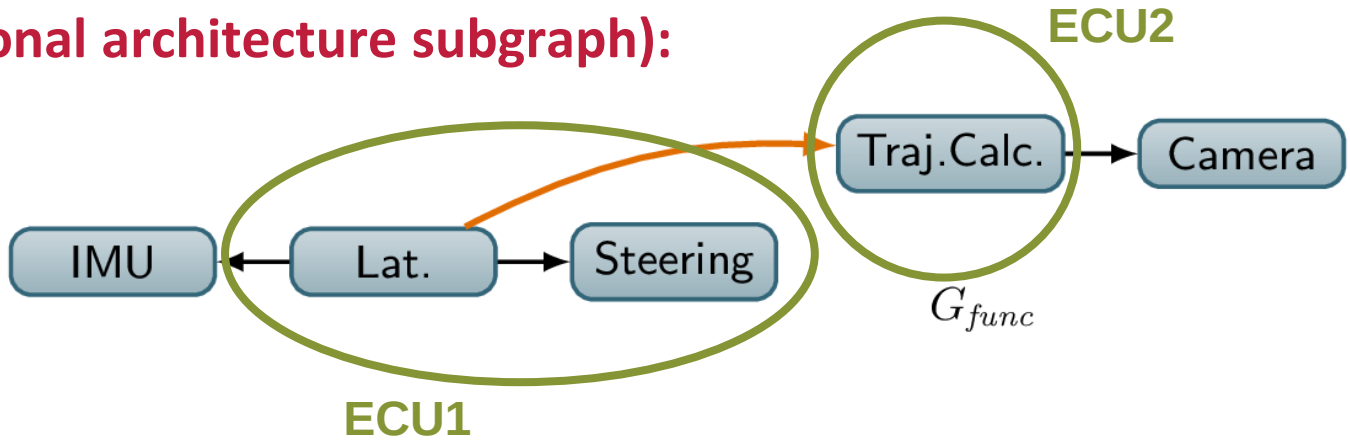


IMU: inertial measurement unit

Function architecture → communication architecture

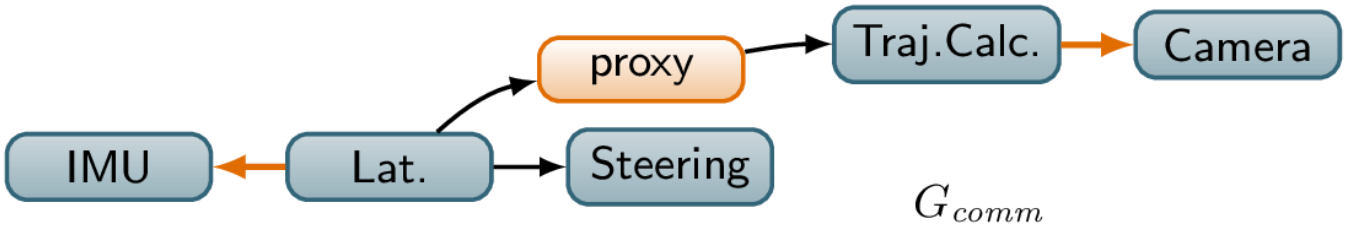
Query (= functional architecture subgraph):

functional blocks
*+ **dependencies***



arc splitting:
mapping + reachability

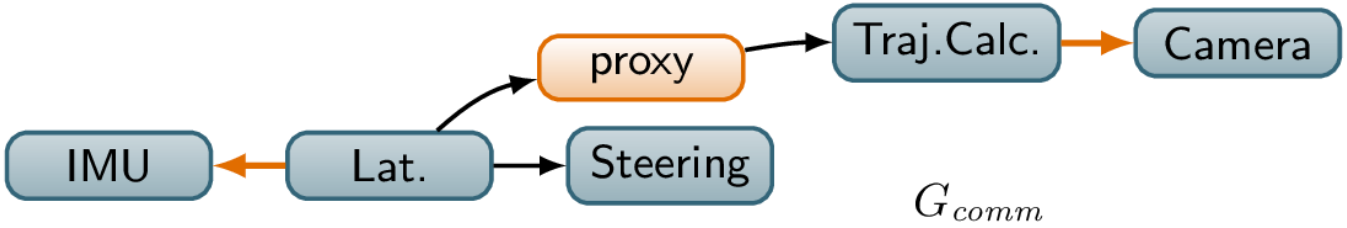
functional blocks
*+ **communication***



Lat.: Lateral Guidance

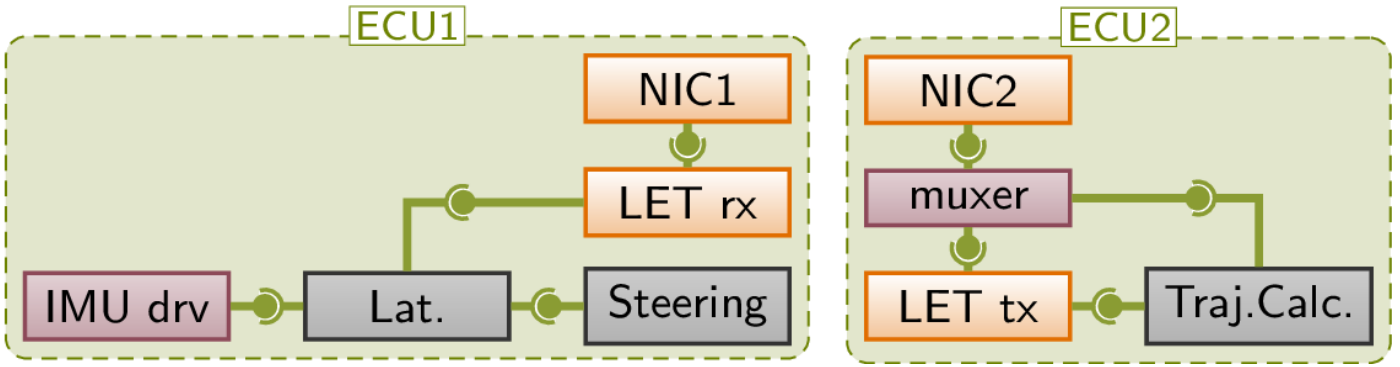
Component architecture → component architecture

functional blocks + communication



- ***pattern-based transformation***
- ***arc splitting:***
  compatibility + cardinality

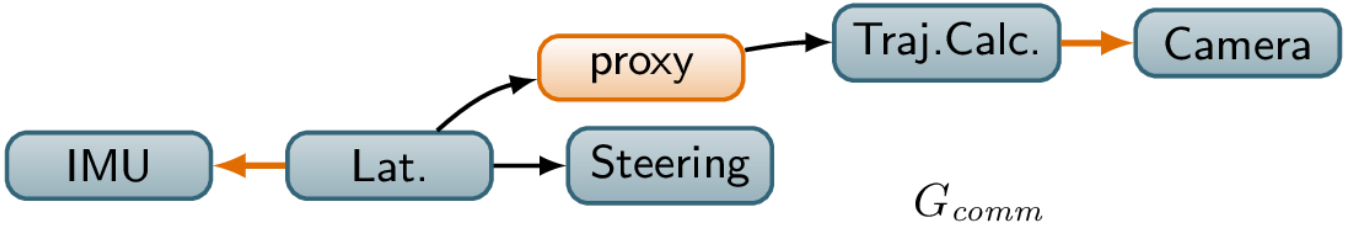
components + service dependencies



drv: driver component LET: logical execution time NIC: network interface controller

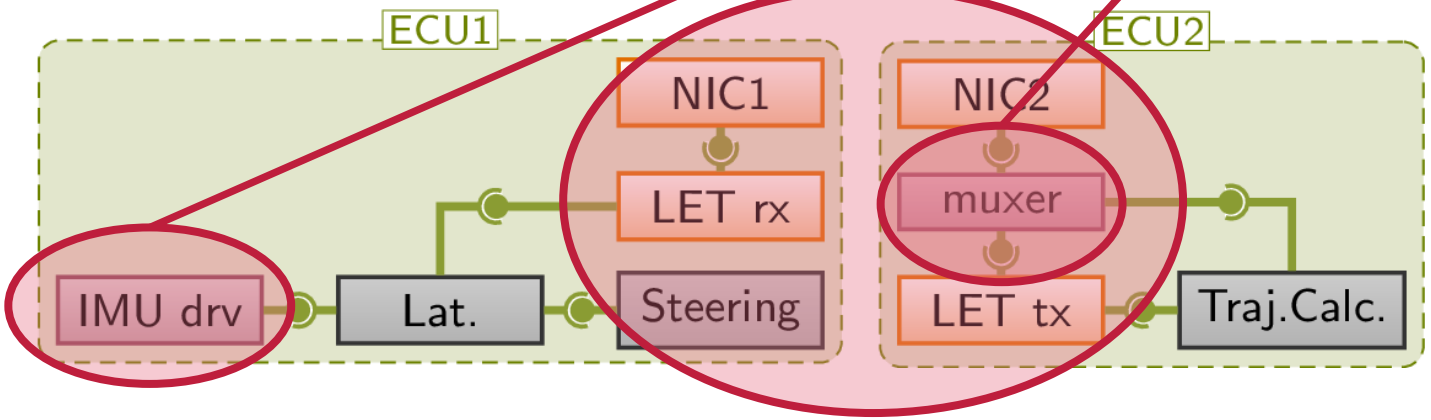
Component architecture → component architecture

functional blocks + communication



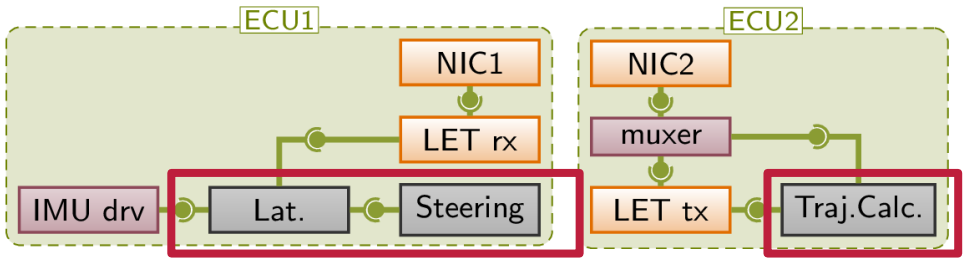
- ***pattern-based transformation***
- ***arc splitting:***
  compatibility + cardinality

components + service dependencies



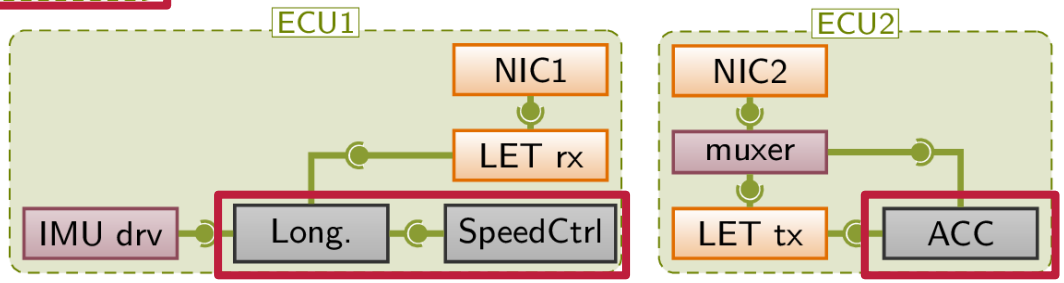
drv: driver component LET: logical execution time NIC: network interface controller

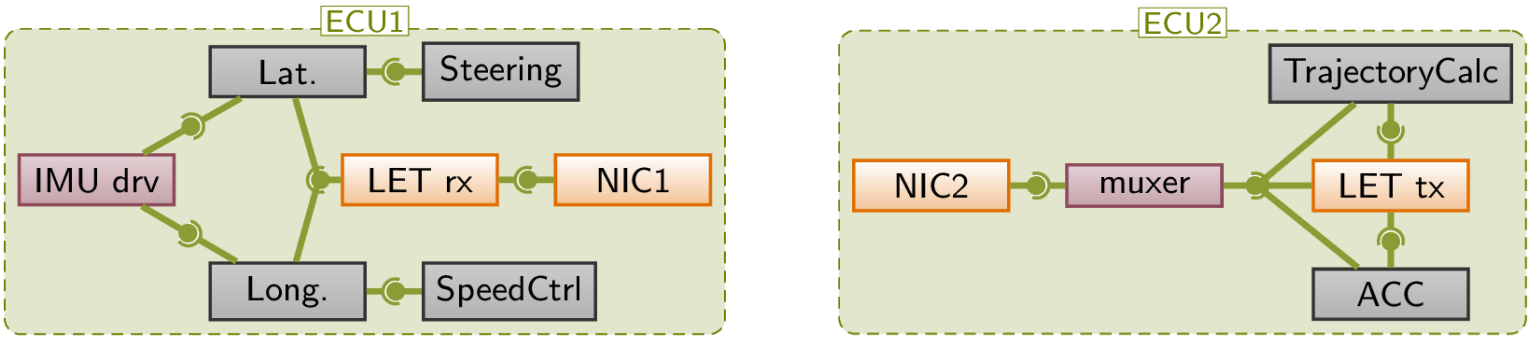
Component instantiation



Component architecture lateral guidance

Component architecture longitudinal guidance (similar process)





ACC: adaptive cruise control Long.: Longitudinal Guidance

Summary

- **in-field** automated integration of software (updates) for vehicles
- **multi-layer** modelling for component-based systems
- integration = graph transformation + **synthesis**
- enables **tracking** of relations and dependencies across layers
  → essential for verification
- still requires methods and mechanisms for **automating design decisions**
- subject to **non-functional requirements** (e.g. latency)

Questions?