

Architecture, Issues and Challenges for Safety Related Autonomous Systems

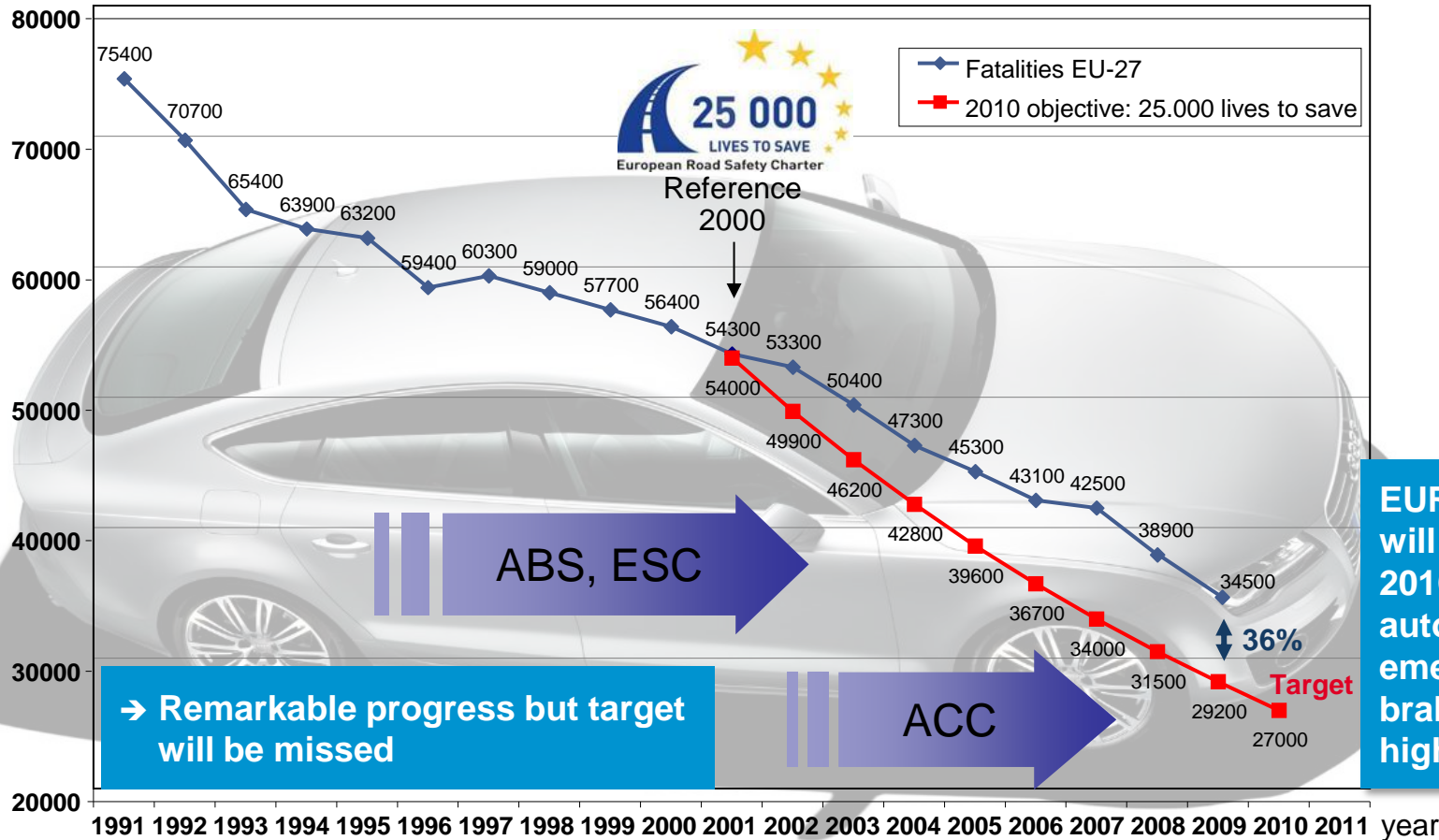
IFIP 10.4 Workshop Viseqrád,

Stefan Poledna
June, 2013

Key Drivers: Safety

Evolution of European Road Fatalities (EU-27)

Fatalities



Source: CARE or national publications; EC Directorate General Energy and Transport December 2009

According to WHO: 50 million injuries in 2010, 1.2 million fatal injuries

Key Drivers: Convenience

„When I don't feel like driving,
I let my car do it for me...”



traffic jams



parking

Quelle: Audi

Goals for Piloted Driving

Convenience

- ▶ **Use time won**
 - work
 - entertainment
 - relax



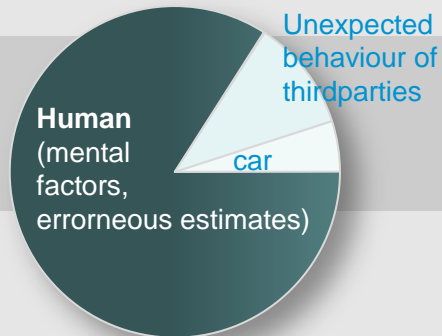
Arriving Relaxed

- ▶ **Arrive stressfree**
- ▶ **Considerable higher productivity upon arrival**



Increased Safety

- ▶ **84% of all accidents are driver induced**
- ▶ **Sensors can monitor 360°**
- ▶ **no driver fatigue**



Environmental Friendly

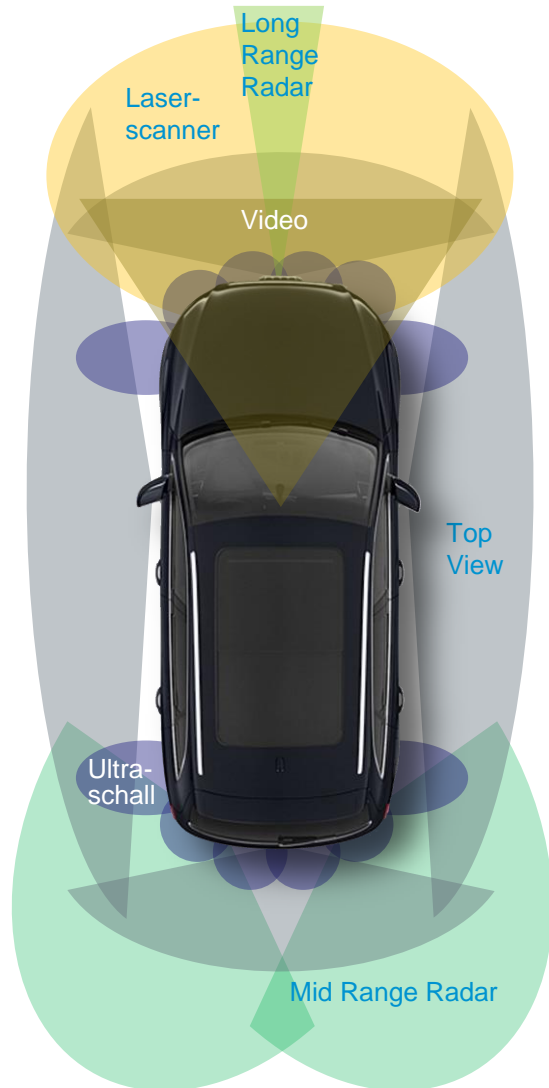
- ▶ **CO2 reduction**
- ▶ **Optimized setup based on route data**


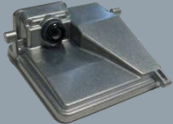


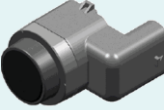
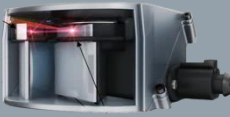


Examples



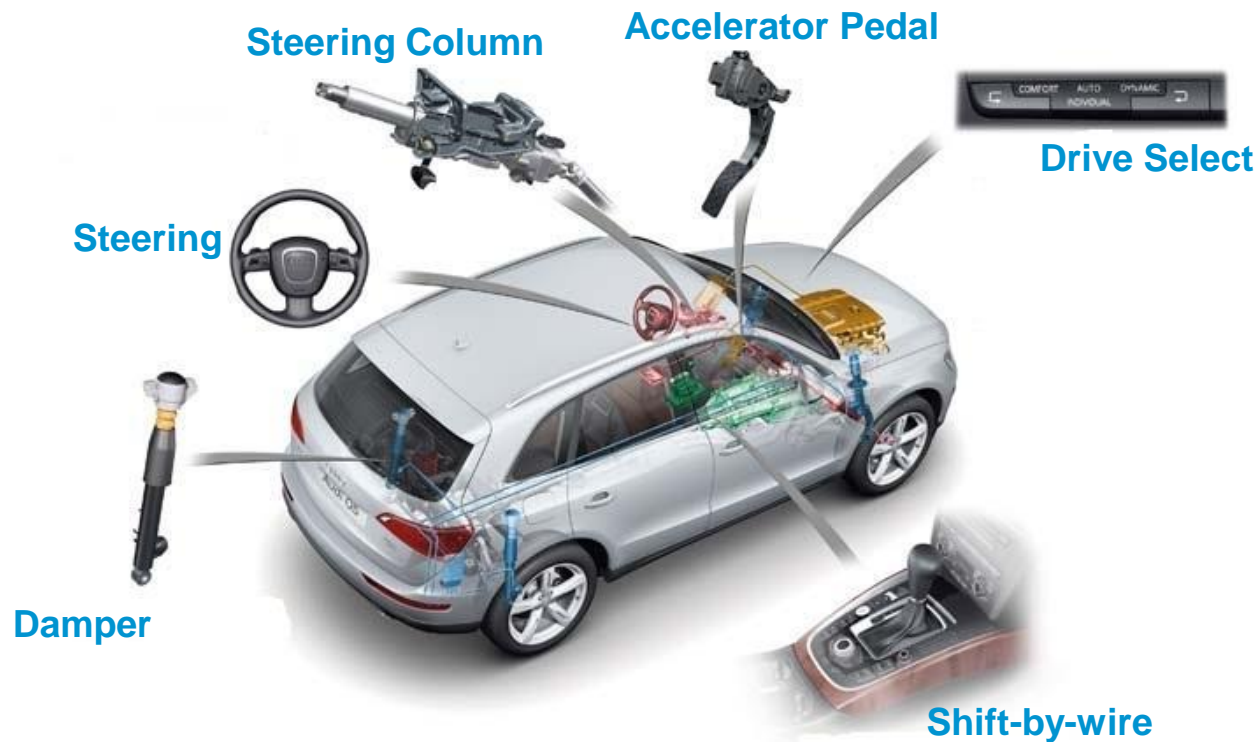
Sensors



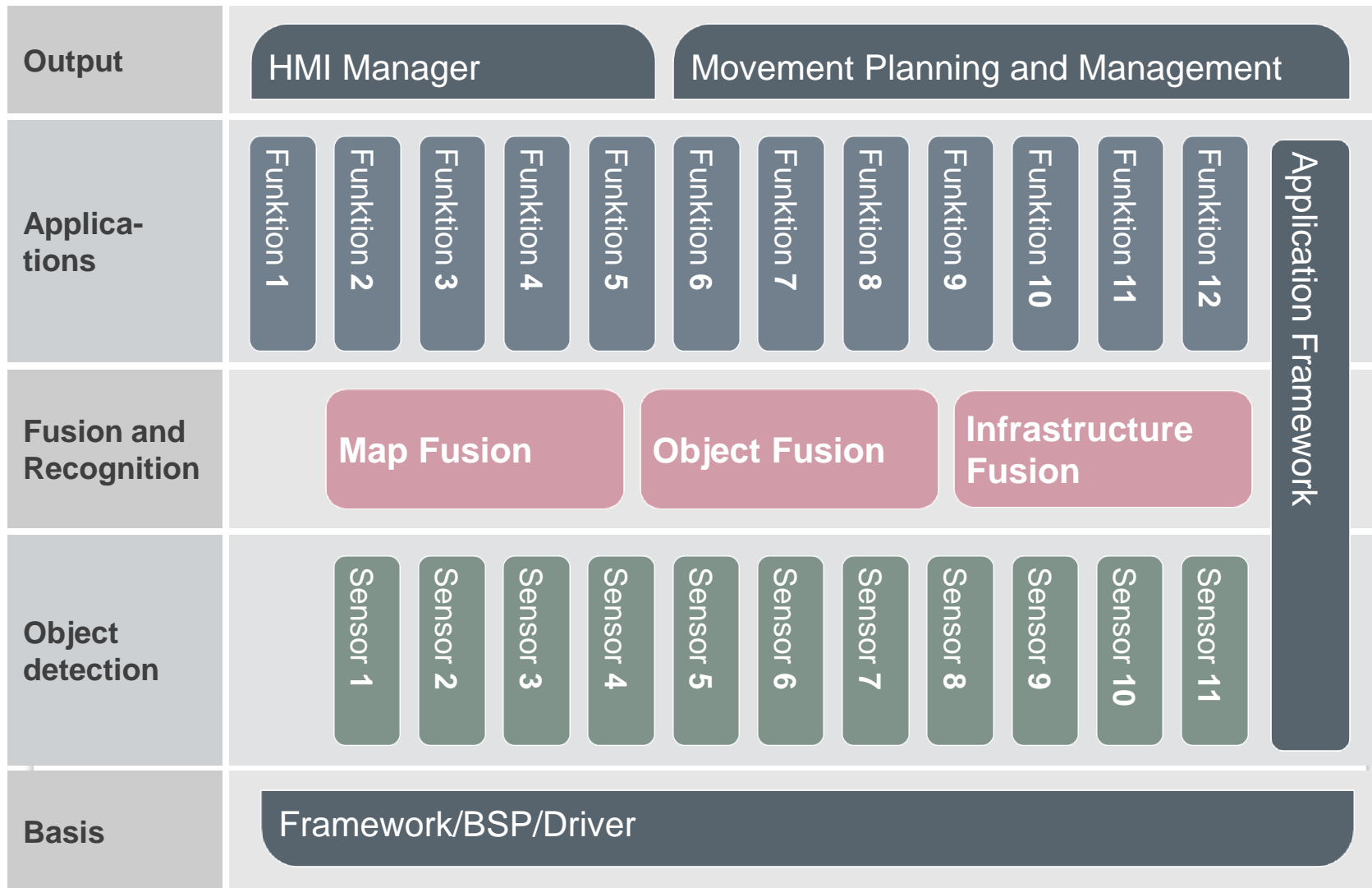
	Long-Range-Radar (LRR 4)
	Video Camera
	Top view Camera
	Middle-Range-Radar (MRR)
	Ultra Sonic
	Laser Scanner
	Predictive Map Data Car2x Connectivity

Necessary Actuators for Automated Driving

- | | |
|---------------------------------------|----------------------------------|
| ▶ Electronic Stability Control | ▶ Powertrain Coordination |
| ▶ Hold management system | ▶ Shift-by-Wire |
| ▶ Deceleration management | ▶ Electric Power Steering |



Layered Function Architecture Decoupling from Sensors



Some Safety Considerations

Driver takes over control

Driver needs some time to be prepared for take-over

- ▶ System is no longer fail-safe
- ▶ Fail-operational behavior for limited time required

Safe State



Reaching a safe state is limiting functions that can be automated

Issues: Complexity Management and Integration

Architecture Level

1

Observability at the functional Level

- Essential for integration and diagnostics

2

Stabile (deterministic) Integration Behaviour

- Seamless flow from simulation, prototyping, validation, integration, to release
- Re-usable and en-richable testcases throughout all pahses

3

Robustes behaviour in case of errors → recovery

- Faults are normal

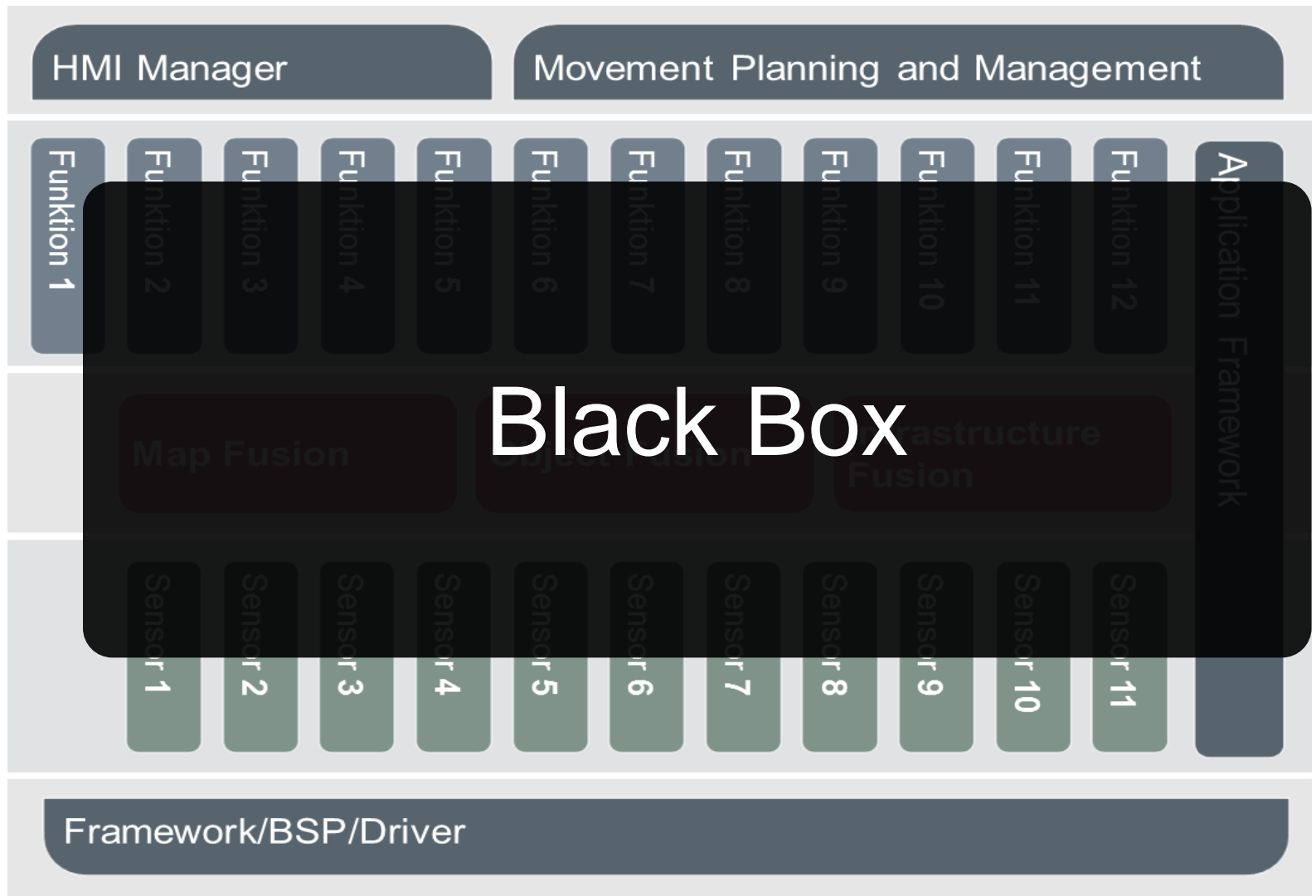
Process Level

4

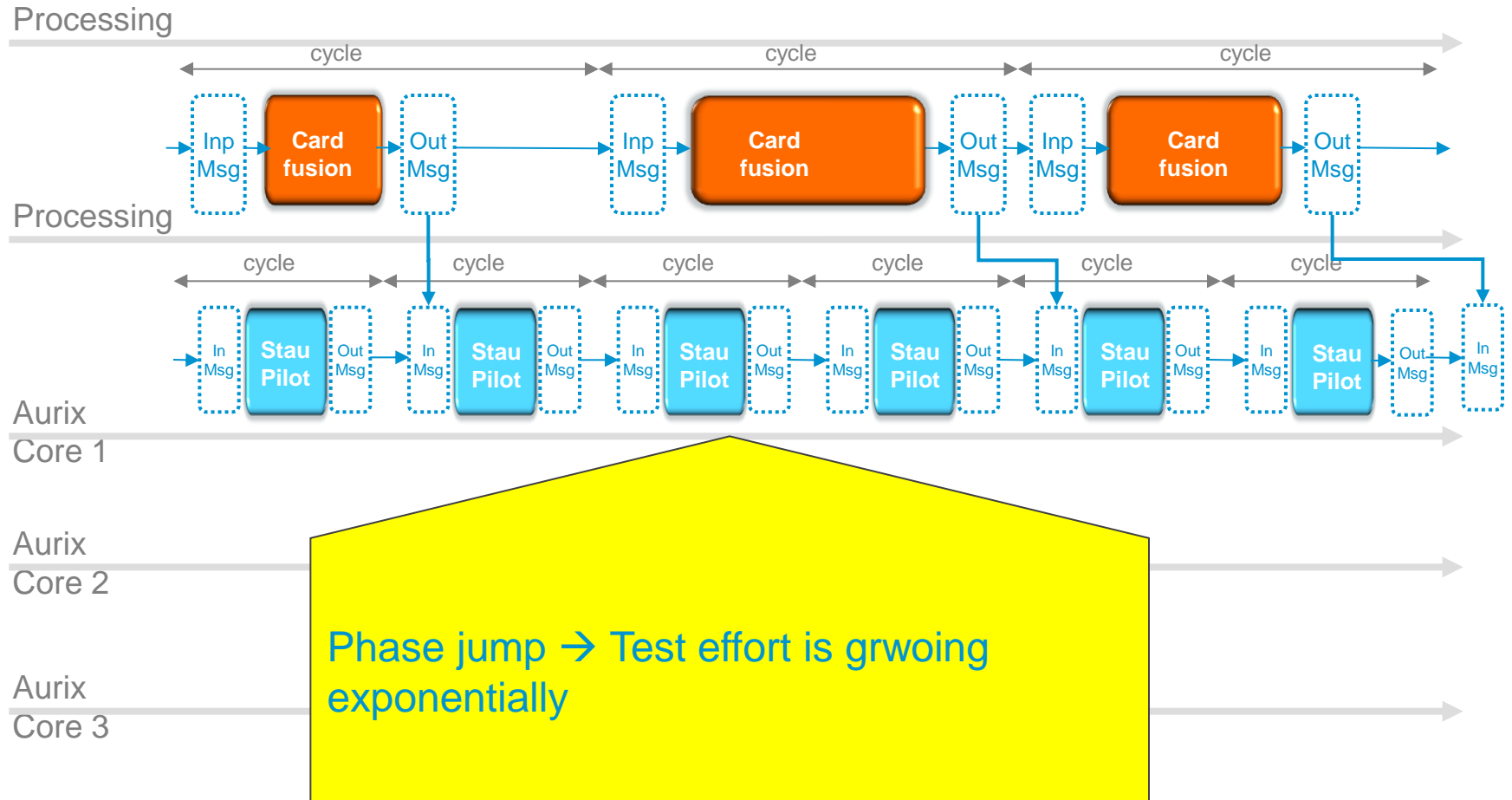
Seamless Toolchain

- RM, CM, Test Cases, Issue Tracking,

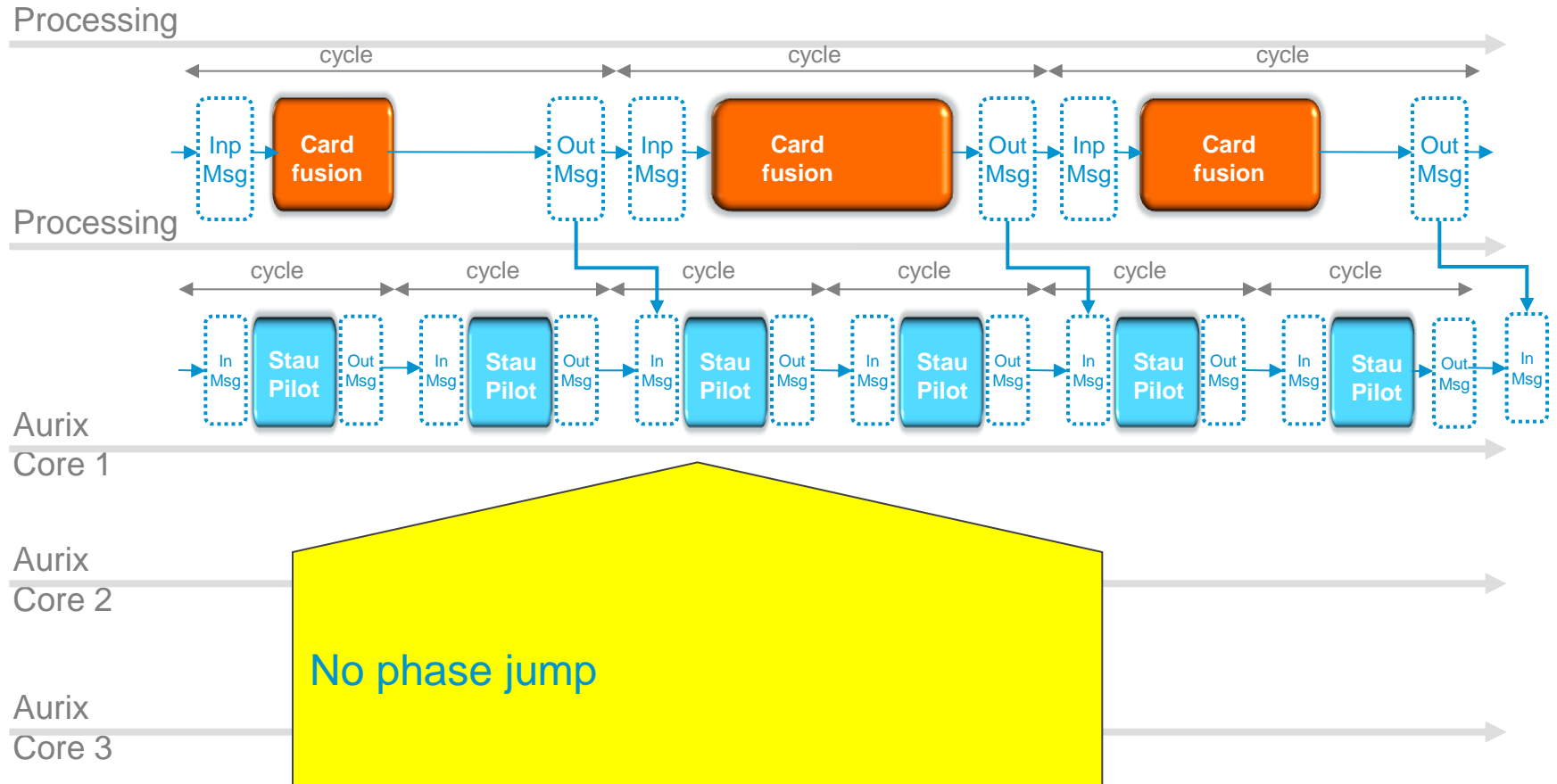
Observability is key for Integration, Test and Validation



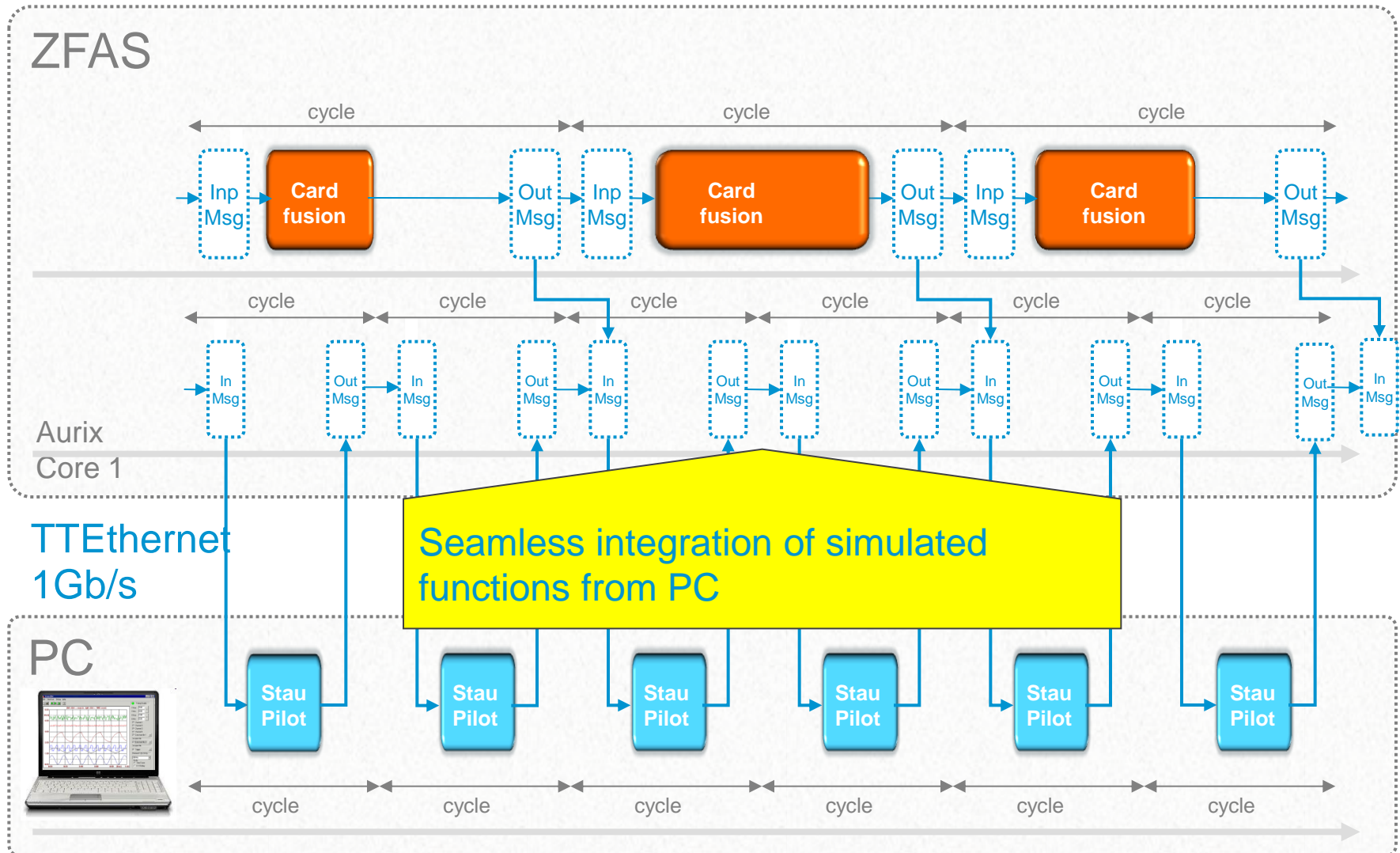
Integration Issue example



Composability or Stable Integration



Transparent Bypassing for efficient development cycles

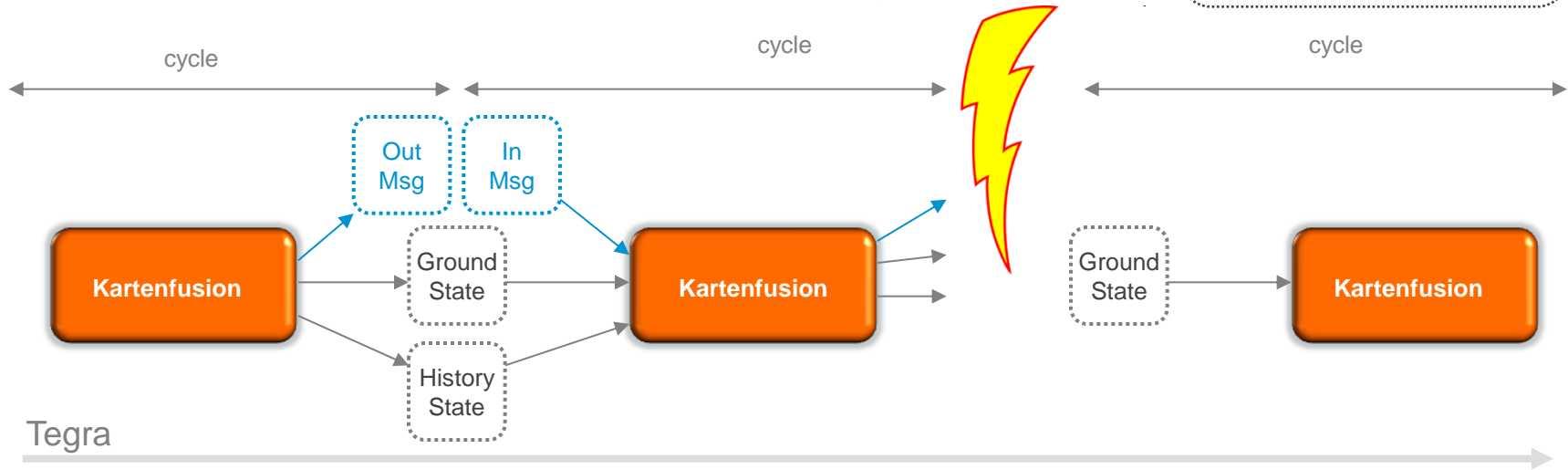


Faults are Normal: Recovery at Function level necessary

Fault → Reset



- Count and Time Recoveries
- Log Error



Transient Hardware failures and „Heisenbugs“ can be tolerated
→ higher availability

- Robustness of “bleeding edge” chips
 - Non-automotive qualified chips → Robustness validation
 - Safety functions on chips without dedicated safety support → Fault-detection based on application knowledge
- SoS interaction Man / Machine
 - At up to about 50 km/h under typical conditions a car can be stopped by an emergency braking assistant before the driver has started to react
 - Brake or accelerate when overtaking a car: Who is charge man or machine → must not oscillate

Additional Issue: “Safe Object Recognition”

- Pattern recognition based on probabilistics
- Safety Standards (e.g. IEC 61508, ISO 26262) assume
 - Component failures of HW (FIT rates)
 - Design faults (SW + HW)
 - But SW is assumed to be deterministic
- What need to be done to get a “Safe Object Recognition”?
 - Testing?
 - Diverse implementation?
 - (Semi-)Formal Methods?

TTTech

Ensuring Reliable Networks

Thank you!