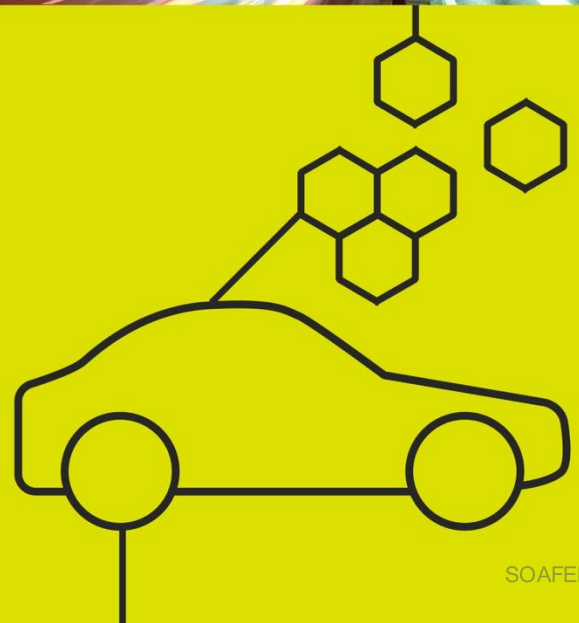
SOAFEE

Deterministic Middleware for Cloud Native Development

Ravi Akella

Sr. Manager

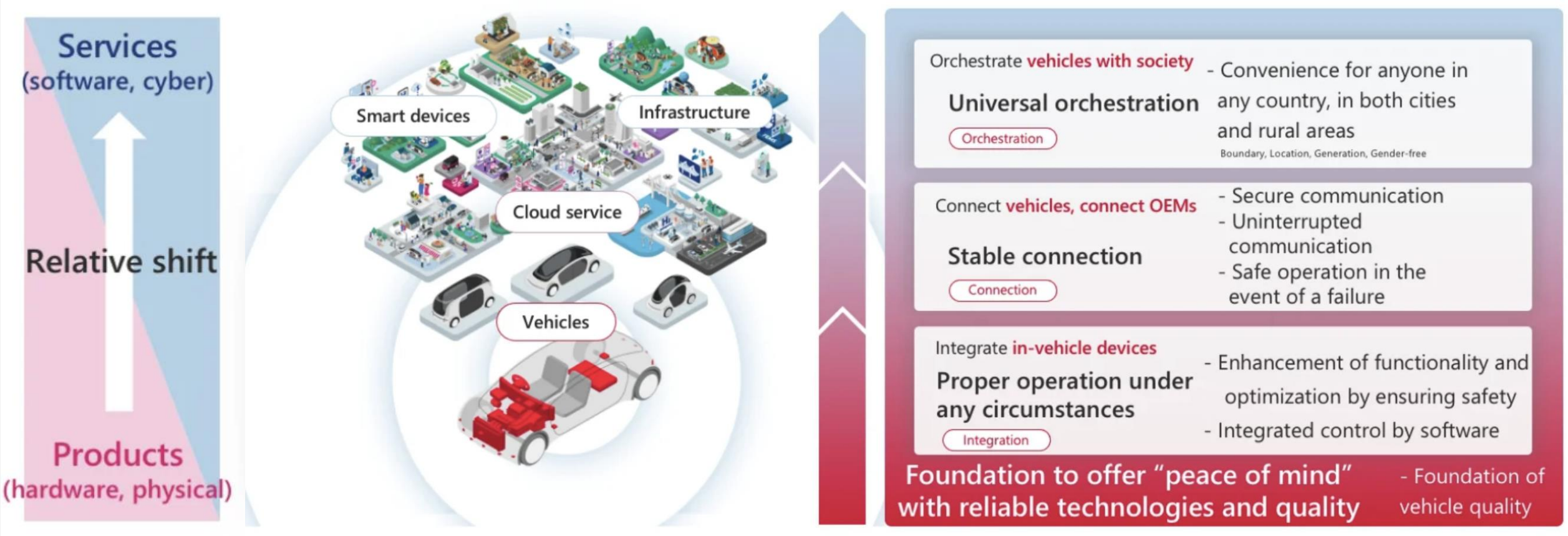
DENSO, Palo Alto, CA, USA



APAC SOAFEE Seminar, Tokyo

May 15, 2025

DENSO’s shift to Software-First



Building social infrastructure based on reliable technologies and a commitment to quality

SDV as a social infrastructure

- SOAFEE is addressing key challenges making the journey exciting!
  - Environmental parity
  - Mixed Criticality
  - Now, Determinism

Overview of determinism

Given the same set of inputs repeatedly, get same outputs

Guaranteeing that critical tasks and processes meet their deadlines consistently

Prioritizing time-critical tasks through real-time scheduling policies

Future state is entirely predictable based on the current state and inputs

End-end considerations throughout the SW lifecycle, leveraging Cloud native development

How do we want our application to behave: Specification to model?

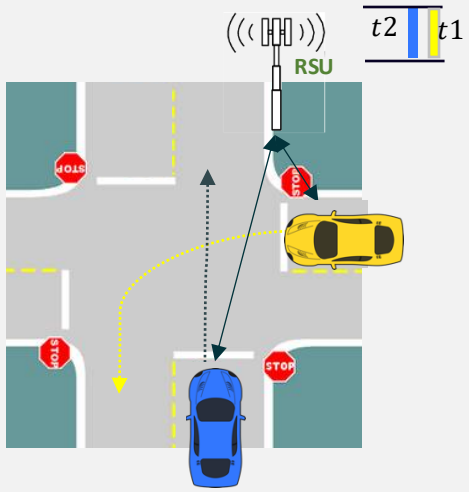
Can we preserve the intended behavior with version changes and new features

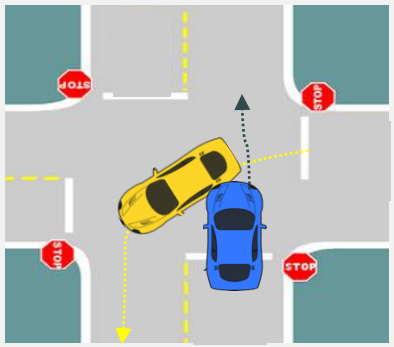
Runtime environment that guarantees application behavior

What tools do we have to ensure behavior is preserved?

Environmental Parity: Can we detect divergence from spec and tune for optimality?

Need for Determinism in future SDV applications





Cooperative maneuvers using an Road Side Unit

Expected behavior:

- Safety (no collisions)
- Fairness (First in First out)
- Utility (Optimal traffic flow at intersection)
- Reliability (Enforce specific policies)

Non-deterministic behavior:

- Collisions
- Intersection blocking
- Unsafe maneuvers
- Lower intersection throughput with more vehicles

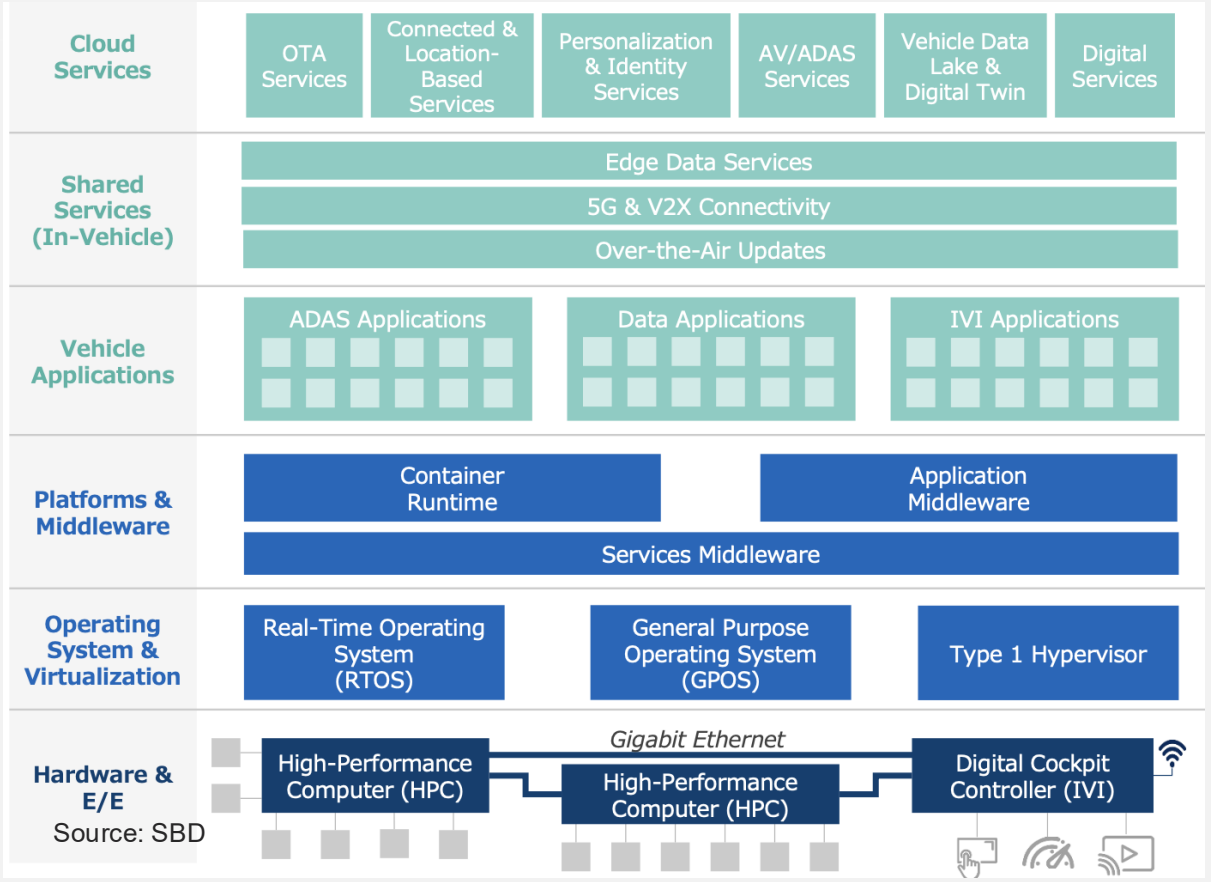
Possible sources of observed non-deterministic behavior

- Network latencies (Usual and injected)
- Clock synchronization error between vehicles
- Simulation server

Determinism: An end-end requirement for SDV Applications

Why is it hard

- Growing software complexity
- Target Environment dependencies
  - WCETS
  - Task Priorities
  - OS specific scheduling policies
  - Interrupt
  - Context switching
  - Networking
- Integration
  - Cross domain
  - Legacy assets part of SDV transition
  - OSS
  - Communication middleware

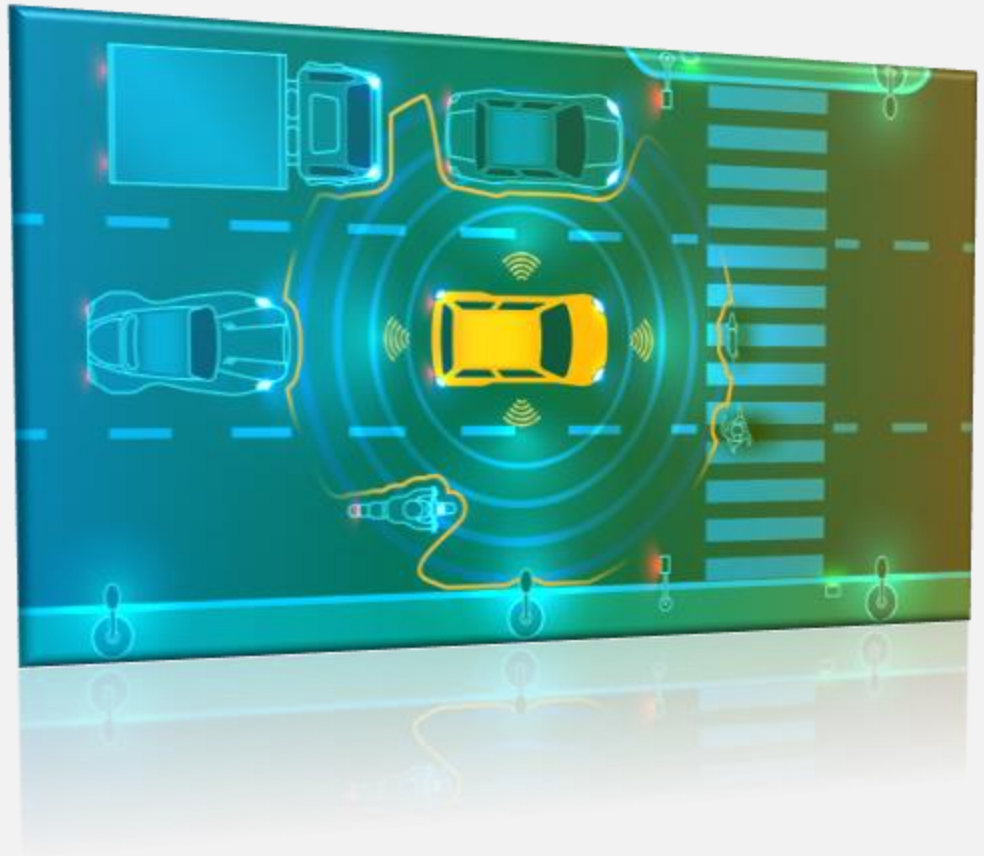


Source: SBD

Key benefits of deterministic SW

- Reduces code complexity
- Efficient HW utilization
- Performance through parallelism
- Helps meet security and privacy analysis

Determinism in the context of SDV



How deterministic can you get

- No bargain for hard safety critical applications
  - Workload attached to a core should always be prioritized
  - Workload execution schedules set per core
  - WCET calculated is based on such assumptions
- Lesser determinism is acceptable
  - Set guard rails: micro behaviors not important
  - Let virtualization/OS deal with it. Just account for the delays.

Solution classes

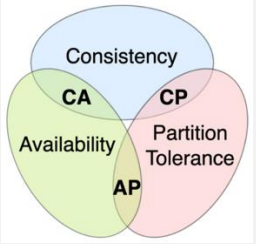
- Reactive: Detect when bad things happen, Trigger fault response
- Proactive: Look ahead, prevent from happening

Expressing determinism: an approach

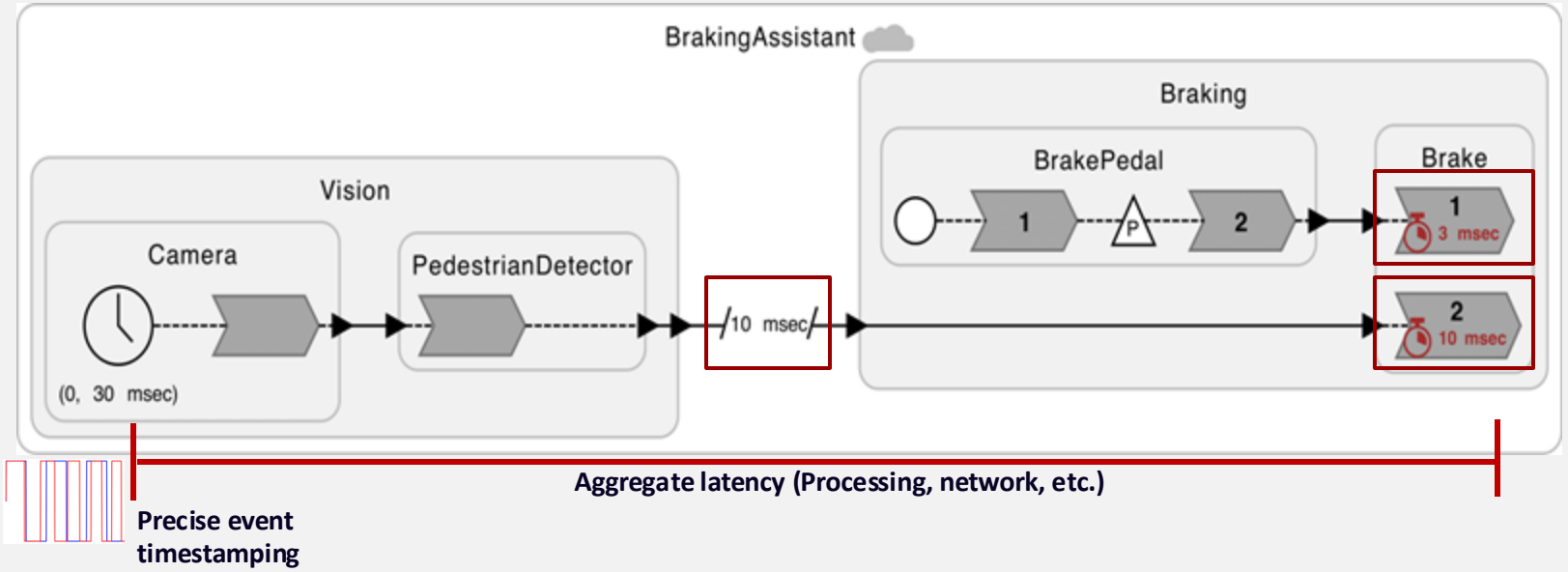
Consistency requirement, expressed as logical delays: the Braking system's camera data can be 10 milliseconds more outdated than the Vision system's camera data.

Availability requirements, expressed as deadlines: i) braking triggered by the brake pedal must respond within 3 milliseconds, ii) braking triggered by the Vision system must respond within 10 milliseconds.

Source: Prof. Lee, UC Berkeley



CAP Theorem

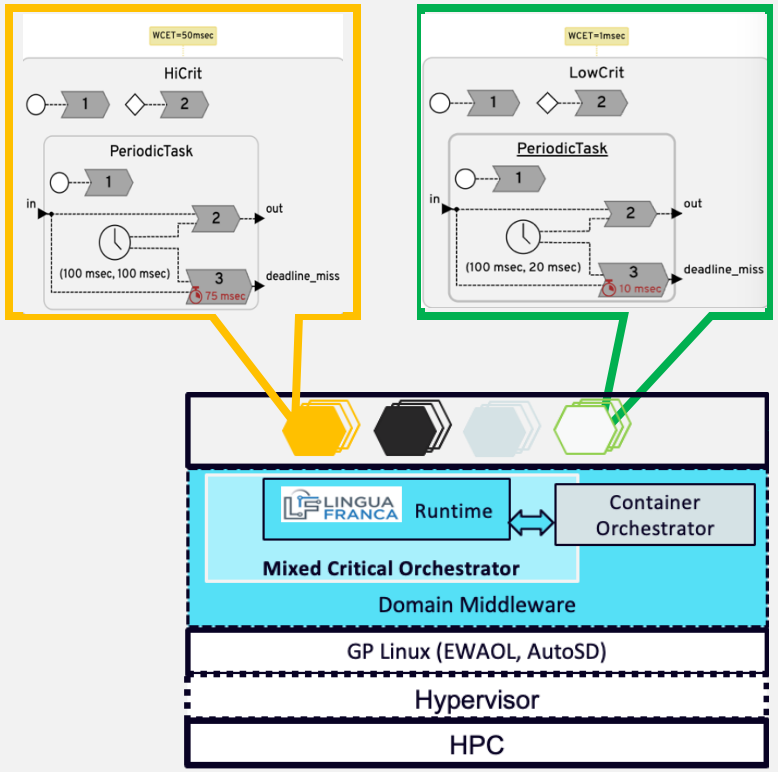


lf-lang.org

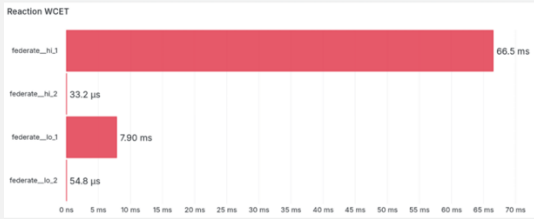
Ability to express consistency and availability requirements, allowing application redesign and performance tuning

Determinism in the context of mixed criticality

- High-criticality components prioritize availability over consistency, lower-criticality components may prefer consistency and can tolerate delays.

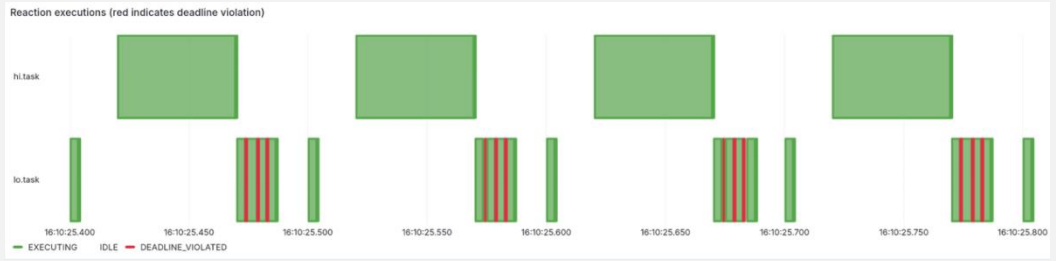


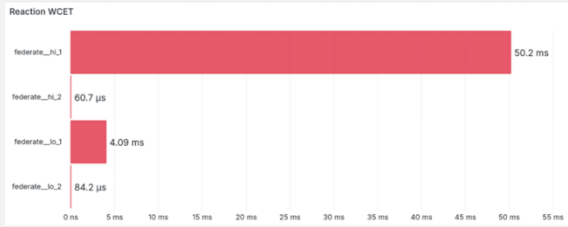
No prioritization



- Intermittent deadline misses at the low-criticality task

Priority based on criticality



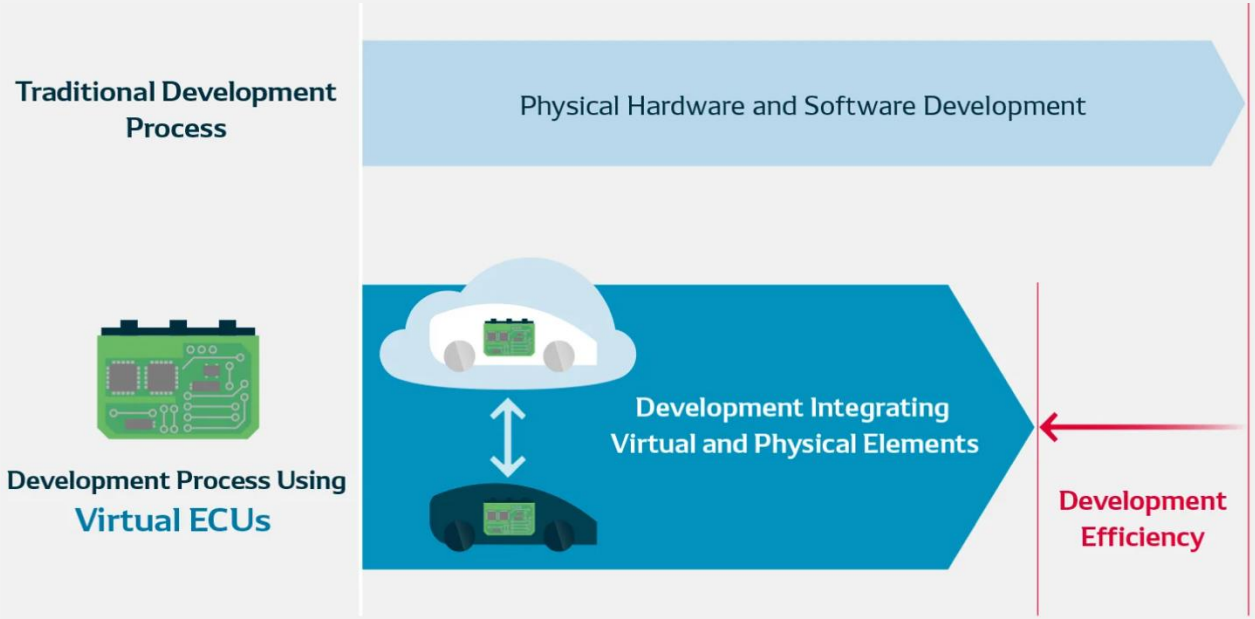


- The high-criticality task takes priority and executes to completion before the low-criticality tasks start executing

Static priority assignment based on criticality is not sufficient to handle a mixed-criticality workloads

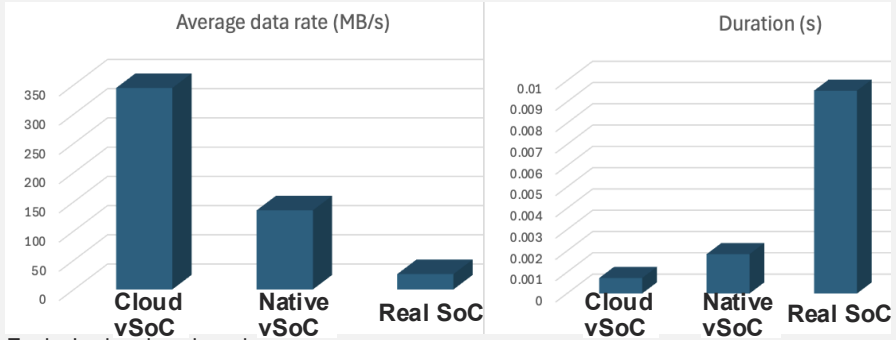
Determinism in the context of Environmental Parity

- Development environment that enables hardware-independent software development



- Performance may not match real hardware
  - Fixed timings for CPU, Memory, and I/O operations
  - Real hardware may perform better or worse than the model

- Deterministic system design is required to achieve environmental parity while reaping the benefits of virtual SoCs



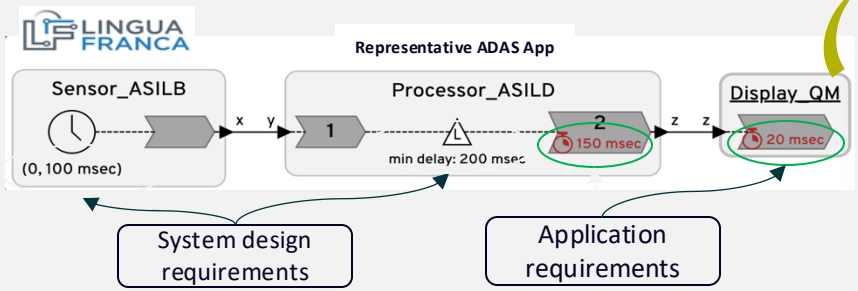
Zephyr's zbus benchmark

Prepare for these differences in your application design

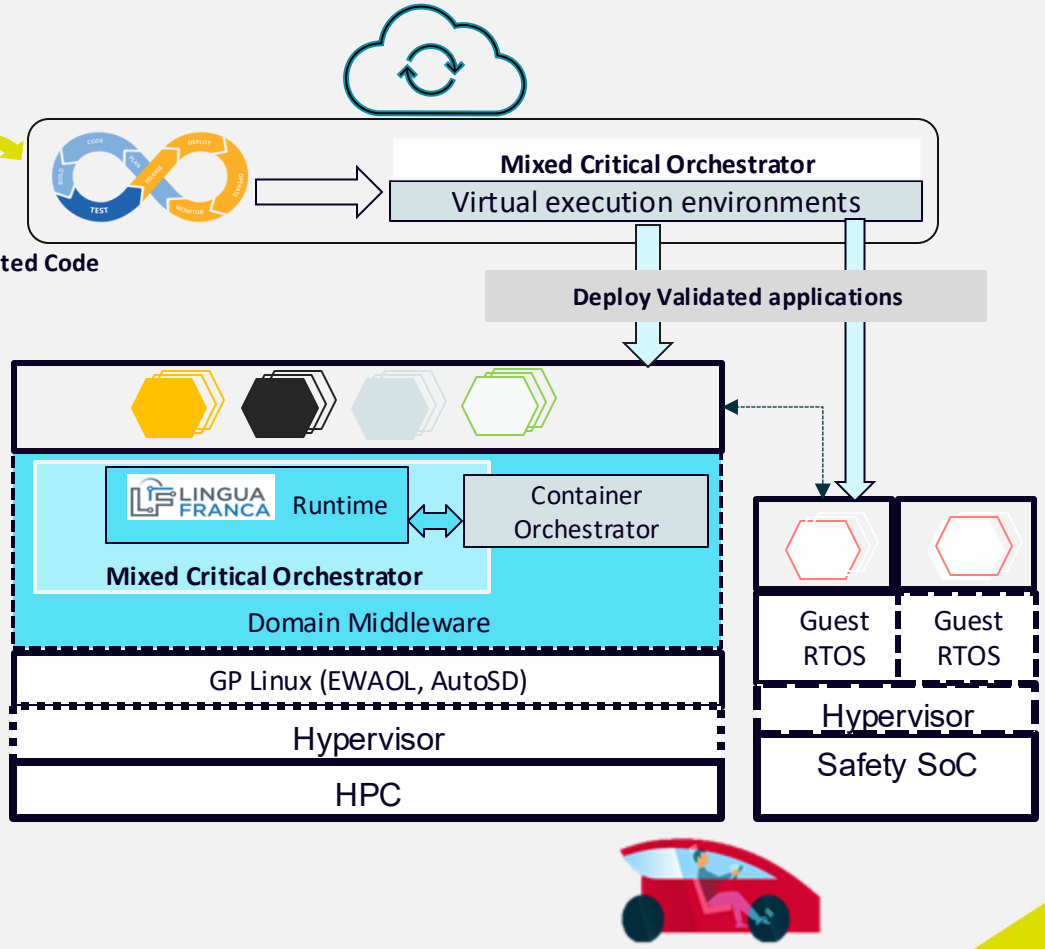
Deterministic SDV Middleware: Architecture

A time centric solution for design and cloud native development of SDV applications

1. System modeling: Lingua Franca model specification and code generation for mixed critical SDV applications



Generated Code



2. Mixed Critical Orchestration (MCO): Runtime enforcement of time properties

- Detection of deadline violations and novel handling mechanisms
- Real-time application monitoring with insights

3. Cloud Native Engineering: Life cycle management of application

- SDV Software Development Kit (SDK): Virtual SoC, Embedded mixed critical deployment support

Features of deterministic middleware

Deterministic app development

- Data flow modeling of applications
- Distributed event scheduling and real-time processing
- Repeatable and testable software behavior

Runtime Observability

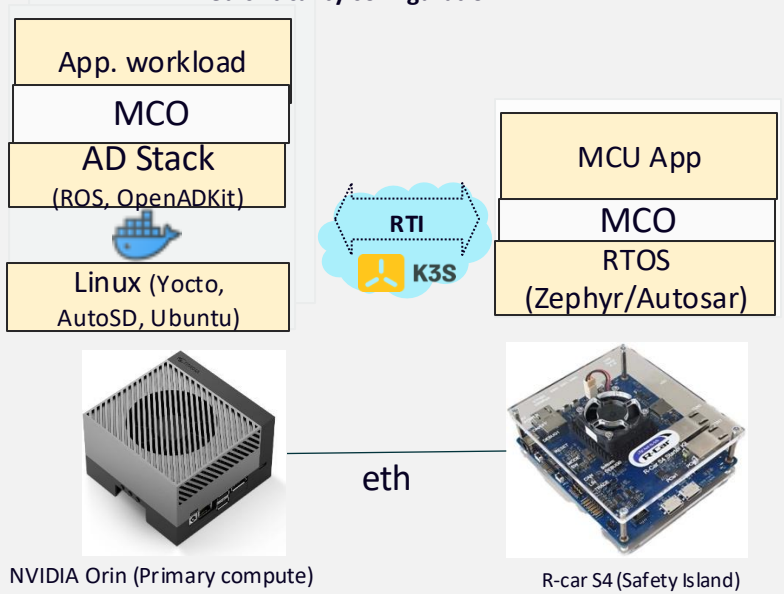
- Detection of application bottlenecks using live monitoring
- Scheduling analysis for performance tuning

Cloud provisioning playbooks

ANSIBLE HashiCorp Terraform

- Automated, turnkey provisioning and deployment of applications
- Support for edge deployment

Mixed criticality configuration



- Workload orchestration with safety island

Container orchestration

K3S

- Triggering pod management based on deadline violations

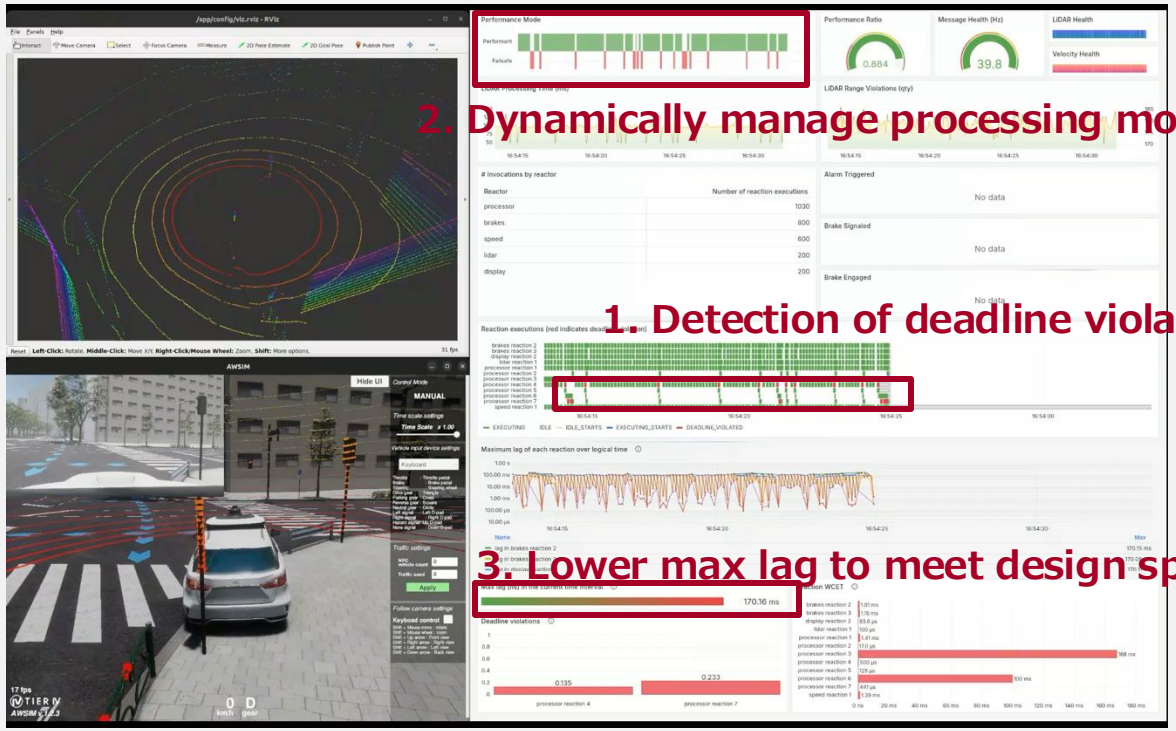
How to achieve determinism: using an example

- Familiar safety critical application leveraging SDV workflow
- Integration of system modeling approach with cloud native development

Nondeterministic case

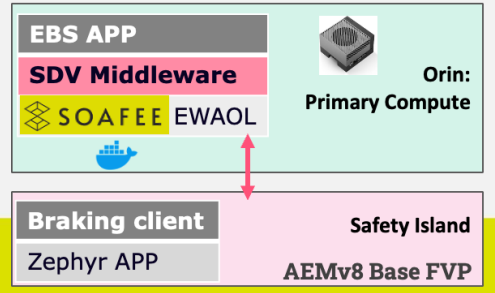
As the event queue size grows (Max. Lag), the vehicle becomes less capable of stopping on time

Deterministic behavior with our middleware



Vehicle can consistently trigger the brakes in time not to collide.

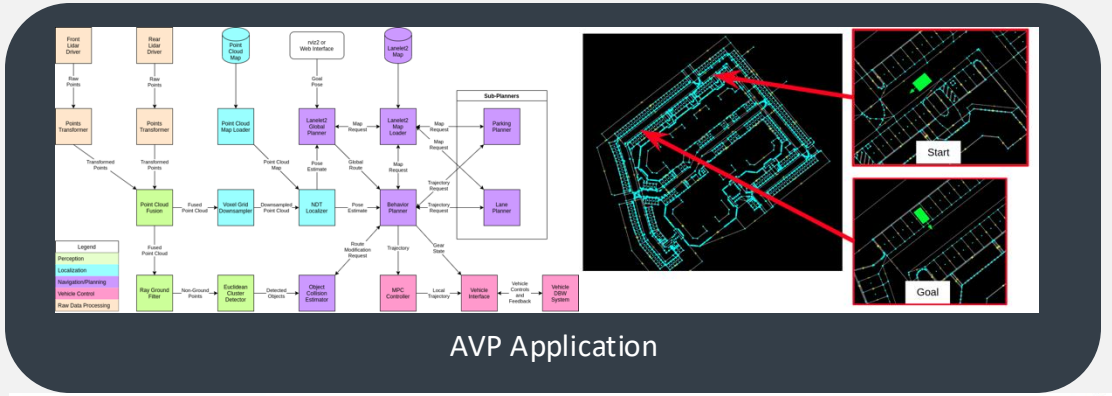
Visit our desk for live demo of this app



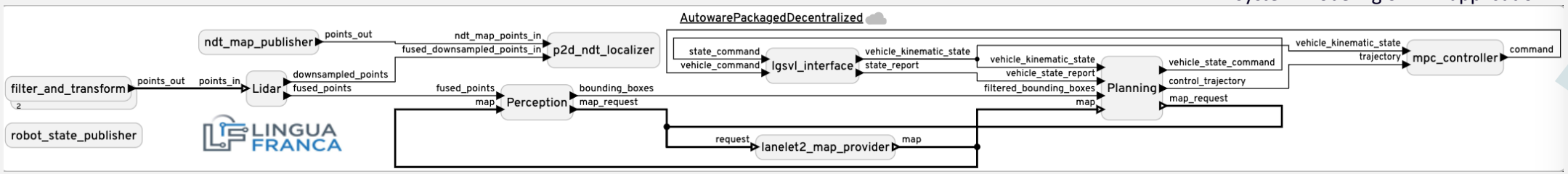
Open AD Kit application on Deterministic SDV Middleware

Automated Valet Parking (AVP)

- Autonomously park and return to a pick-up/drop-off area in a parking lot
- Autoware Foundation's original demo* ported to Open AD Kit v1.0
- AVP exhibits non-deterministic behavior (Eg: unresponsiveness, jitteriness, etc.) on SDV platform
- This problem highlights the importance of deterministically scheduling various interacting subcomponents



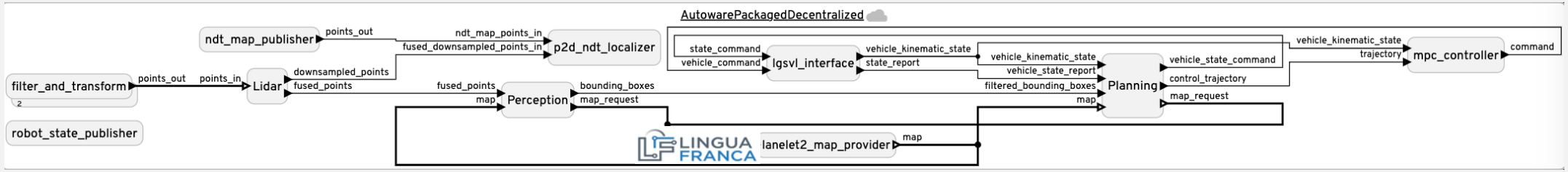
LF system modeling of AVP application



Lingua Franca enforced deterministic scheduling to suppress observed issues in original demo

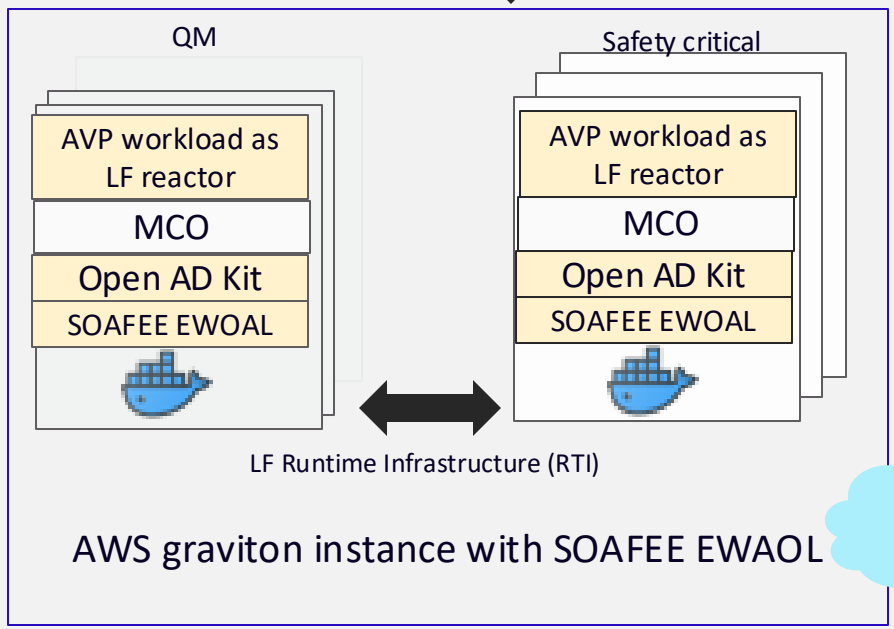
*URL:https://autowarefoundation.gitlab.io/autoware.auto/AutowareAuto/avpdemo.html

Integrated LF and Open AD Kit application



Deploy generated code as containers on cloud

- Coordinated federated execution among SDV “microservices”
- Mixed Critical Orchestrator (MCO) manages the scheduling across workloads
- Porting ROS2 nodes to our midleware
- In current configuration, safety critical and QM containers run on virtual High Performance Compute (HPC)



K3S

DENSO's (first) SOAFEE blueprint

- Open-sourced Automated Valet Parking application

SOAFEE About Architecture Information Calendar Join Our Community

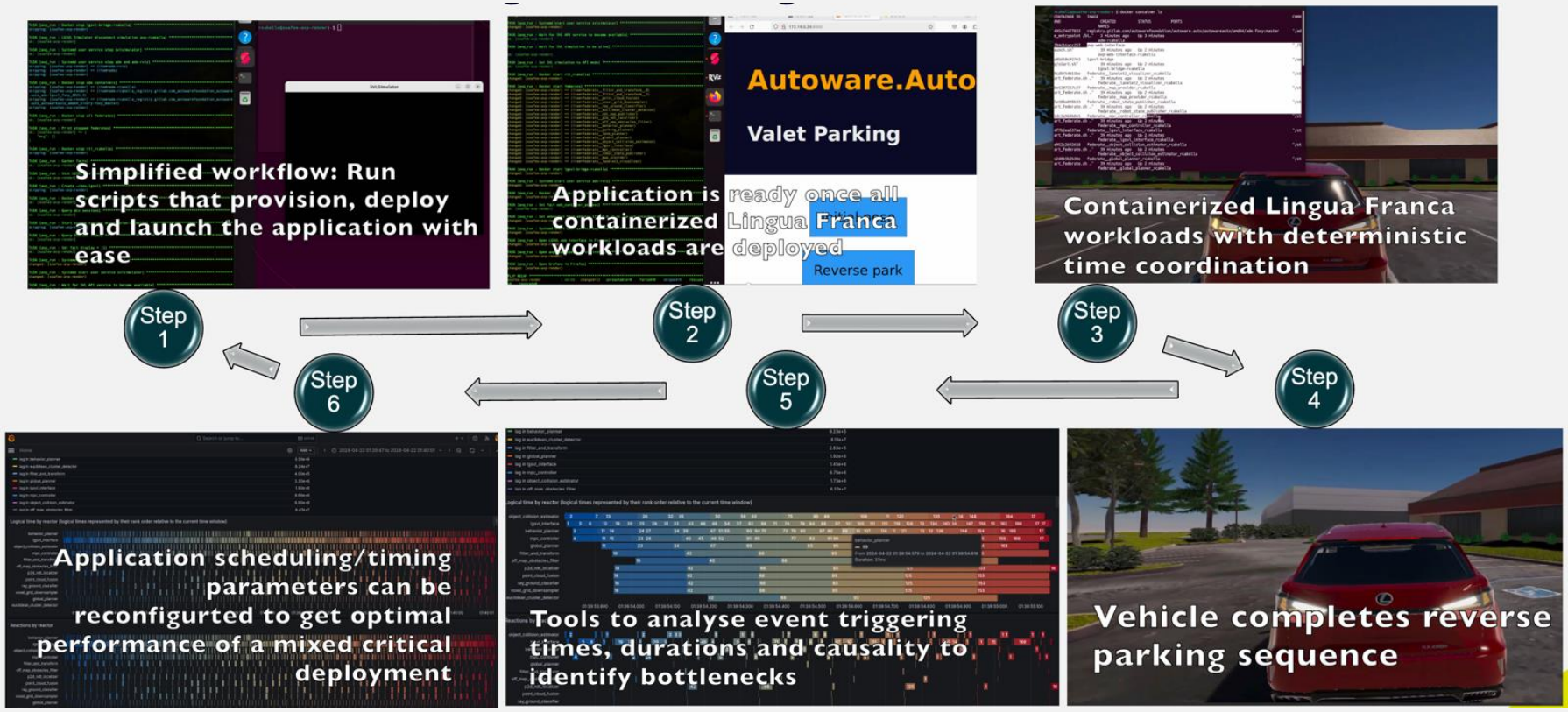
Deterministic middleware for SDV application development

By Shinichiro Nakamura, Ravi Akella

Blog soon to be published on soafee.io

DENSO Blueprint to SOAFEE

Available for trial



Original AVP application exhibits unreliable behavior (Eg: unresponsiveness, jitteriness, etc.)

Deterministic middleware provided end-end methodology and tooling to address such issues

SOAFEE

Thank you!