Safe Open Vehicle Core Architecture Workshop

December 3rd – 4th 2024

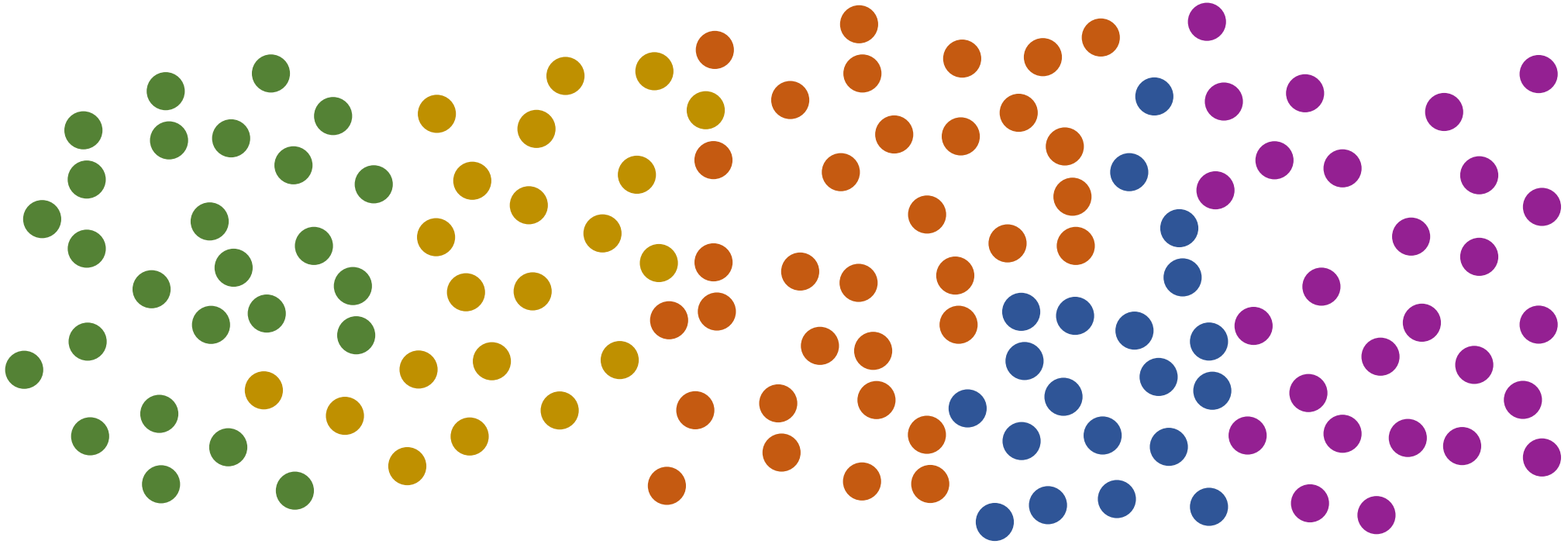
QORIX

Agenda

- **Day 1**
  - Agreement on agenda
  - Introduction (1000-1130)
    - Setting the Scene
    - Requirements
    - Targets
  - Core Architecture (1230-1800)
    - Principles & Paradigms
    - Building Blocks
    - Scalability
  - Agenda Crosscheck
- **Day 2**
  - Architecture Details Core (0830-1130)
    - Core
    - Memory
    - Data
    - Communication
    - Runtime
  - ADAS application framework
    - Requirements and architecture
    - Breakdown of requirements towards the platform
  - Architecture Details Core Services (1230-1800)
    - Log & Trace
    - Persistency
    - State & Execution
    - Diagnostics
    - Health
    - ...
- **Day 3**
  - Implementation (0830-1130)
    - Guidelines
    - Programming Languages
    - Existing Contributions
  - Roadmap & Resources (1230-1500)
  - Wrap-Up & Next Steps

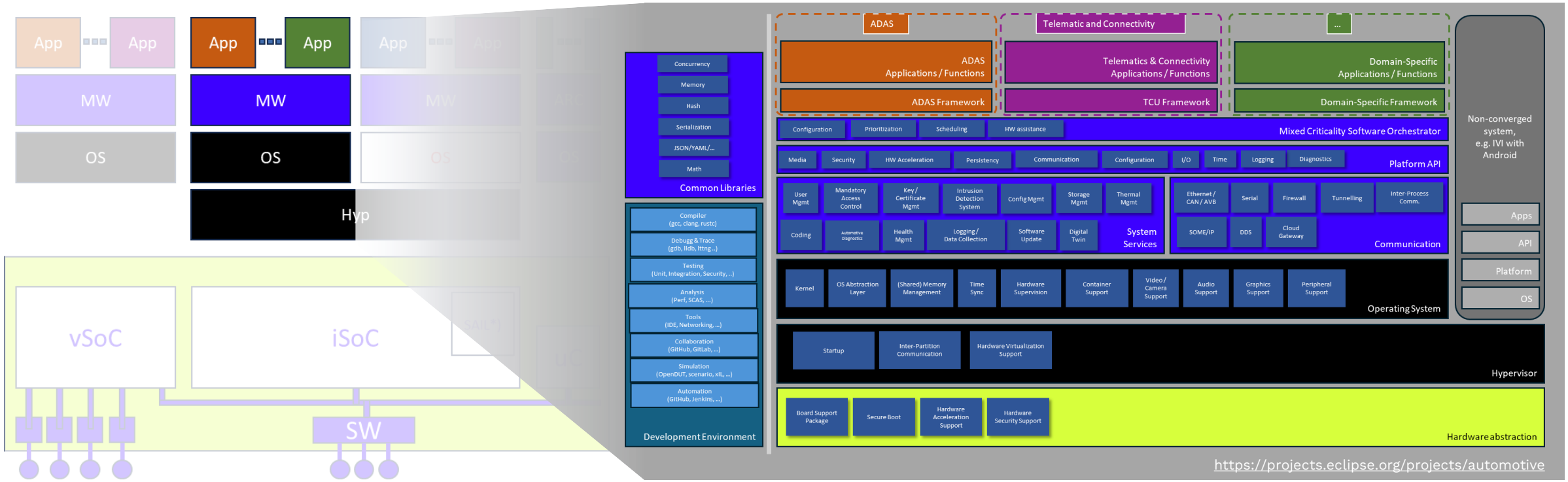
Introduction

What we want



The Stack

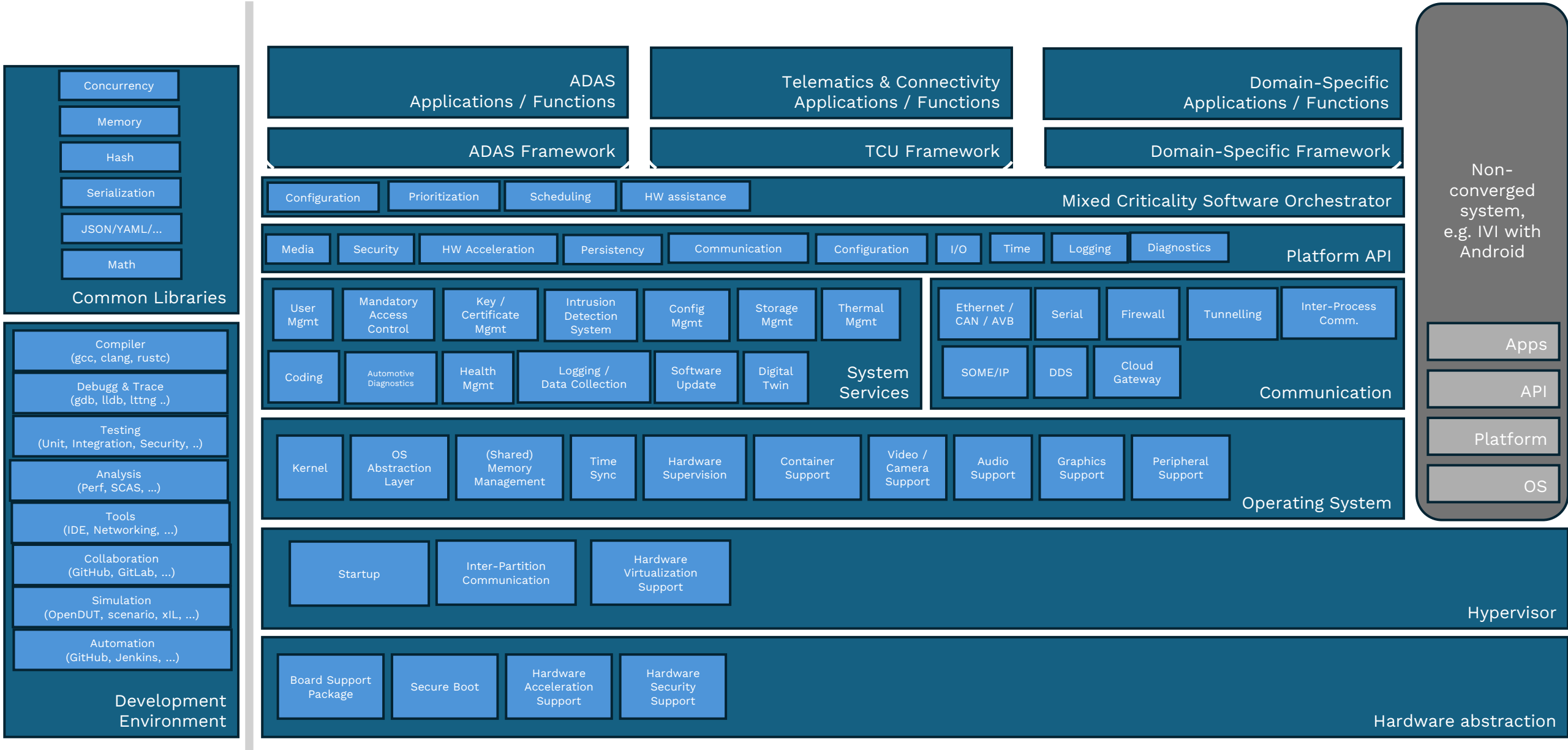
What we want



Applications	Complexity	Integration Effort
x 10	x 100	x 100?

QORIX

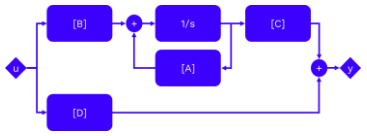
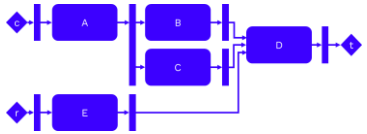
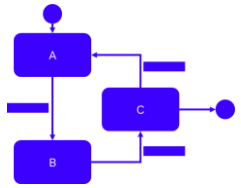
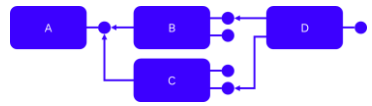
High Level View



Requirements & Targets

- Multiple OS: (Safe-)Linux, QNX(SDP 8) (Reference): ✅
  - CI/CD/CT exists for both OS configs, builds and passes validation
  - Hypervisor supported
- Multi Language: C++(17), Rust: ✅(auf API Level)
- Performance Communication: Zero Copy IPC ✅→ Deep Dive
  - Which domains (hypervisor, OS, languages, compiler), security, safety
- Assumptions & KPI on OS → tbd while designing
  - KPI: Boot Time; Suspension Mechanisms
- HW Accelerators: NPU, TPU, GPU ✅(data and execution interaction)
- HW Peripherals:
  - Abstraction on Data/Information.
  - Direct HW access out of scope.
- App Deployment & Integration → Deep Dive, Allocation of capabilities for Apps
- Safety & Security ✅
  - Target: ASIL-D (correctness); An ASIL-D application can execute as ASIL-D
  - Seooc
  - Reference Platform Certification(?)
- Validation: Record & Replay

Analysis and mapping of domains

	Realization	“Service Oriented Communication kinda does the job”		
	Powertrain Chassis	ADAS	Body	Infotainment Connectivity
				
S System	Control System	Data Flow	State Machine	Service
P Process	State Space	Activity	Activity	Activity
D Data	Signal	Value	State	Parameter
T Trigger	Timer	Update	Event	Invocation Notification

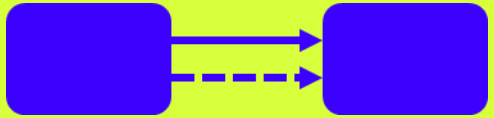
QORIX

Target setting

Target	The core stack shall support SOA Topics, RMI and Notifications
Target	The core stack shall support Runtime Orchestration

Information Flow

- We need Information Items for Data *and* Control
- We have scenarios where Data Flow and Control Flow are separated
- We need a stack platform that can handle both

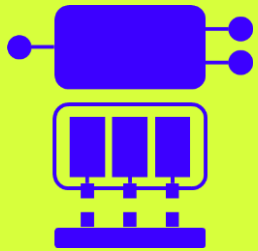


Scalability

- We need a stack that can grow over time with it's components
- We need stability of the stack component API
- We need long-term maintainability of the stack component API

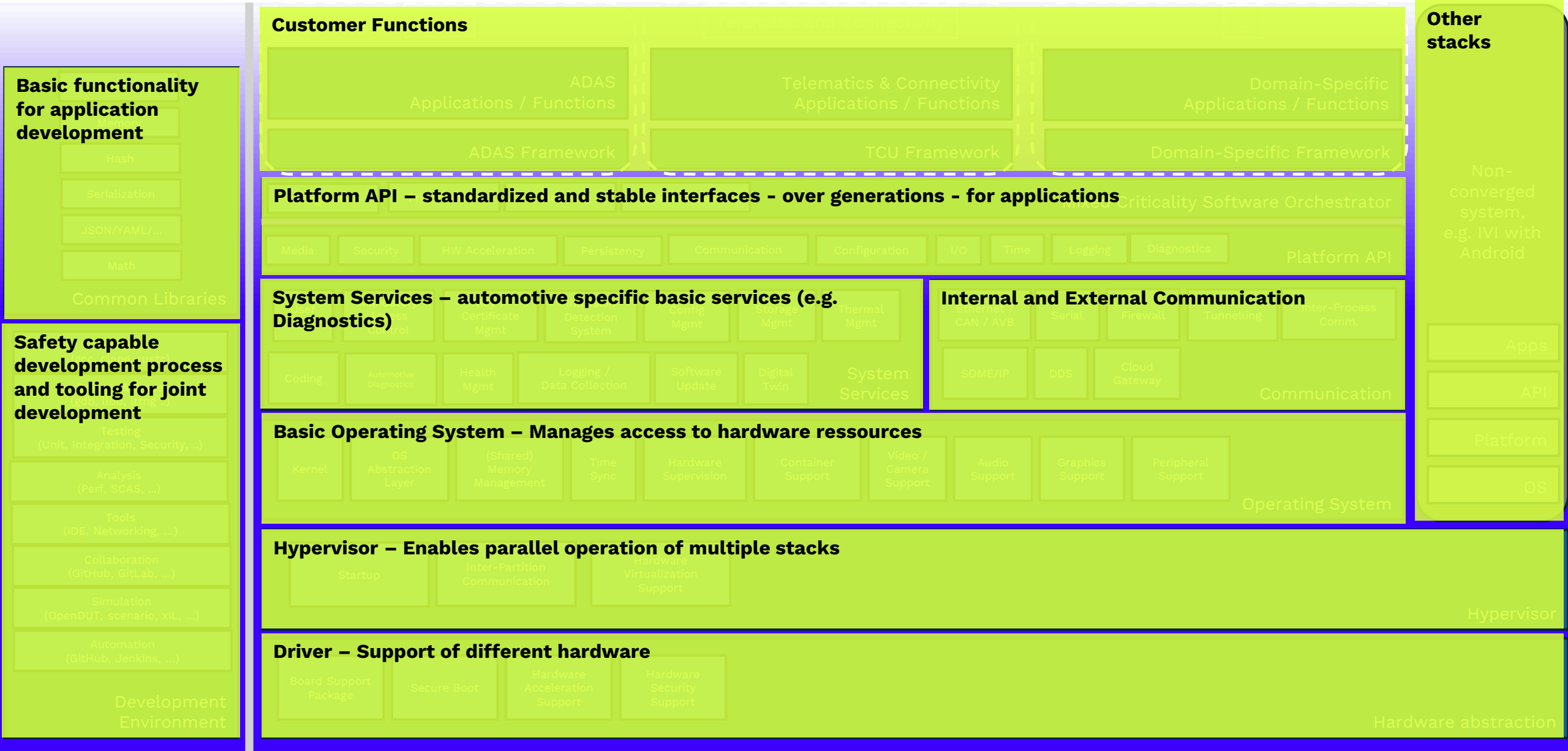
Integration

- We need clear definition of data and interfaces
- We need portable implementations
- We need a profound deployment model

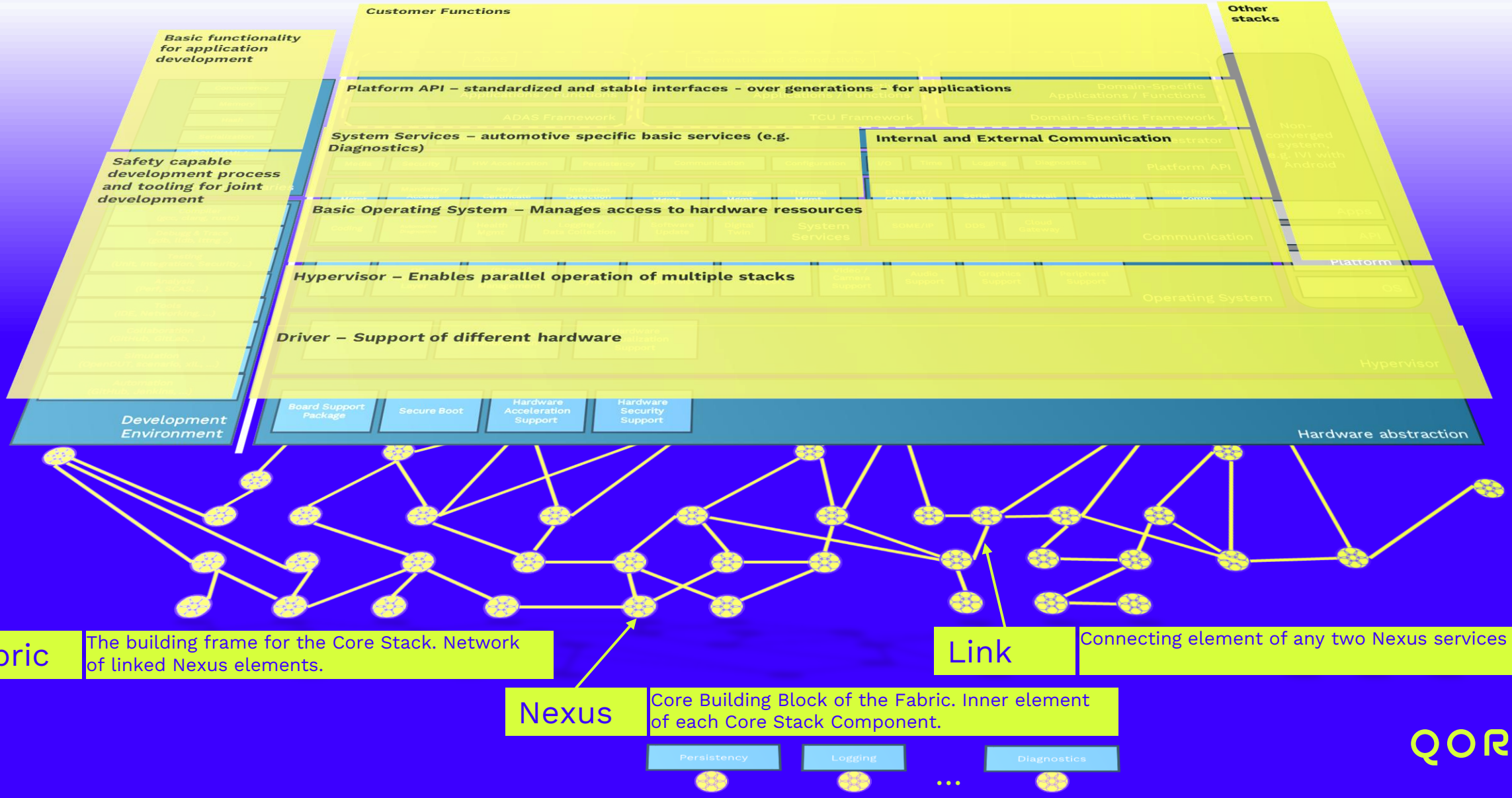


Scalable Core Architecture

High Level View



Under the hood: The Nexus Fabric



Fabric The building frame for the Core Stack. Network of linked Nexus elements.

Link Connecting element of any two Nexus services

Nexus Core Building Block of the Fabric. Inner element of each Core Stack Component.

QORIX

Results

Glossary - General

- Container
- Process
- Component
- Module
- Package
- Resource
- Executable
- Application

Glossary – Runtime & Orchestration

- Process
- Thread
- Thread Pool
- Executable
- Activity
- Application
- Function
- Executor
- Scheduler
- Runtime Configuration (Program)
- Task
- Task Chain
- Monitors & Watchdogs
- Exception Model
- Event
- Engine

Glossary – Communication & Data

- Data Types
- Memory Layout (Compiler/Lang InterOp)
- Queues
- Locking
- Atomic Operations & Consistency
- Memory Allocation / Deallocation & Coalescing
- Zero-Copy
- Topic
- RMI
- Event / Notification
- OS & Memory
- State
- Port

API Modules

Initial

- Runtime
- Com (Data & Runtime)
- Time, Clock, Timepoint, Duration
- Persistency: KV (Config), NVM, Blob: Safe application data consistently and have an error strategy if that is not the case
  - API Level as initial concept
- Log:
  - Logging Guidelines, Logging API, eventually Logging contribs

Postponed

- Diag
- Trace & Profiling
- Crypto

Languages

- C++ 17
- Rust

Repos

Cross-Functional Teams:

- Eclipse-Score-FEO (Fixed Execution Order Framework)
- Eclipse-Score-IPC
- Eclipse-Score-Runtime
- Eclipse-Score-Log
- Eclipse-Score-Persistency

Eclipse-Score-Architecture Community