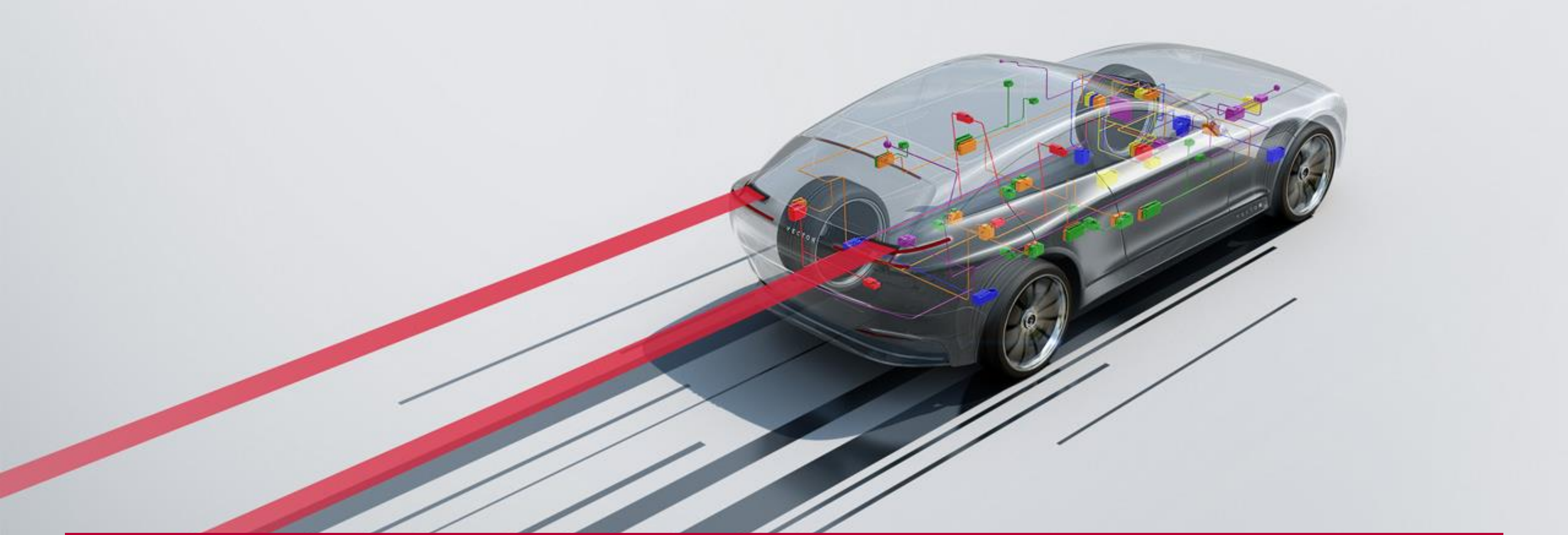
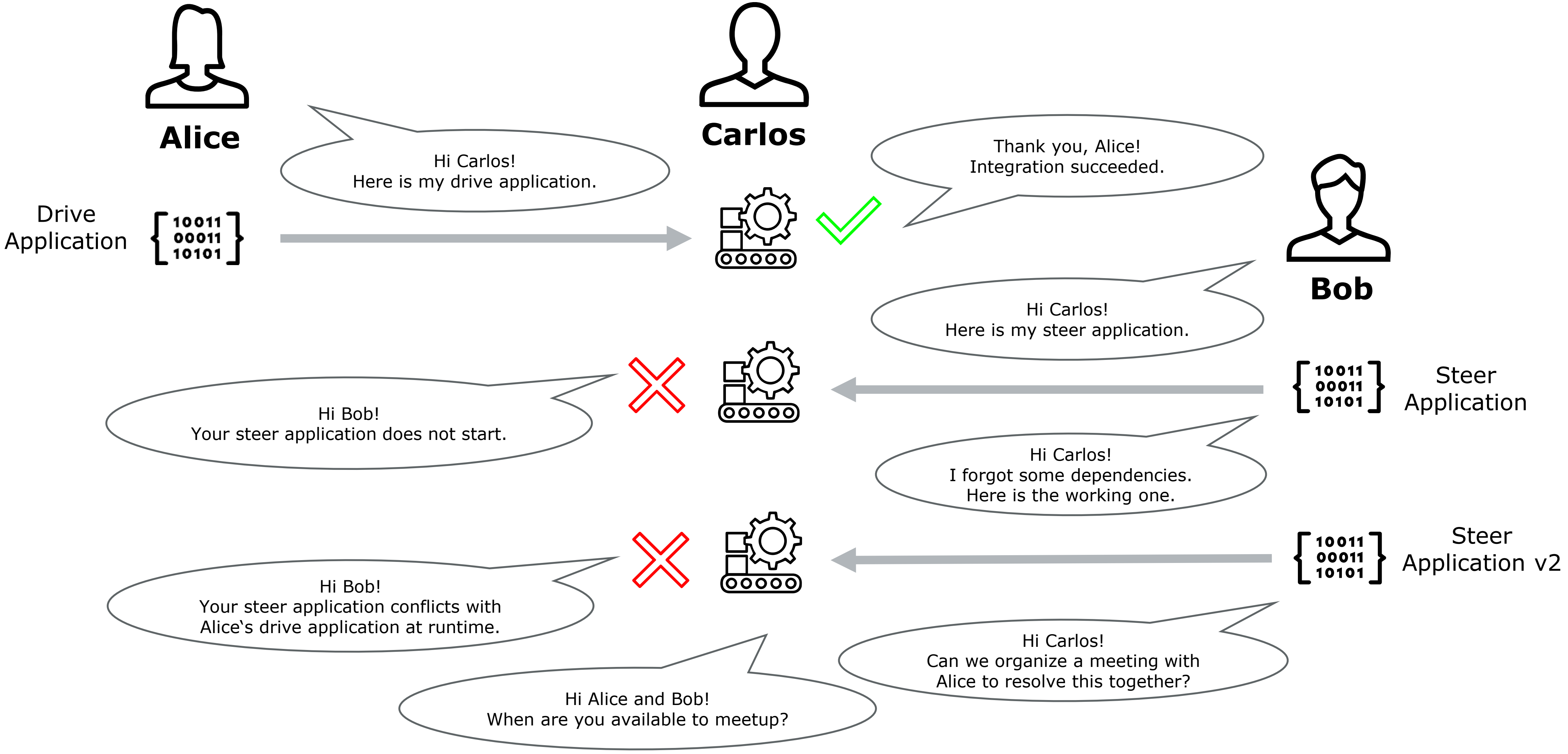


Containerization of AUTOSAR Adaptive - Challenges & Solutions

V1.0 | 2022-05-03

Some day in AUTOSAR Adaptive Application development department…



Hi Carlos!
Here is my drive application.

Thank you, Alice!
Integration succeeded.

Hi Carlos!
Here is my steer application.

Hi Bob!
Your steer application does not start.

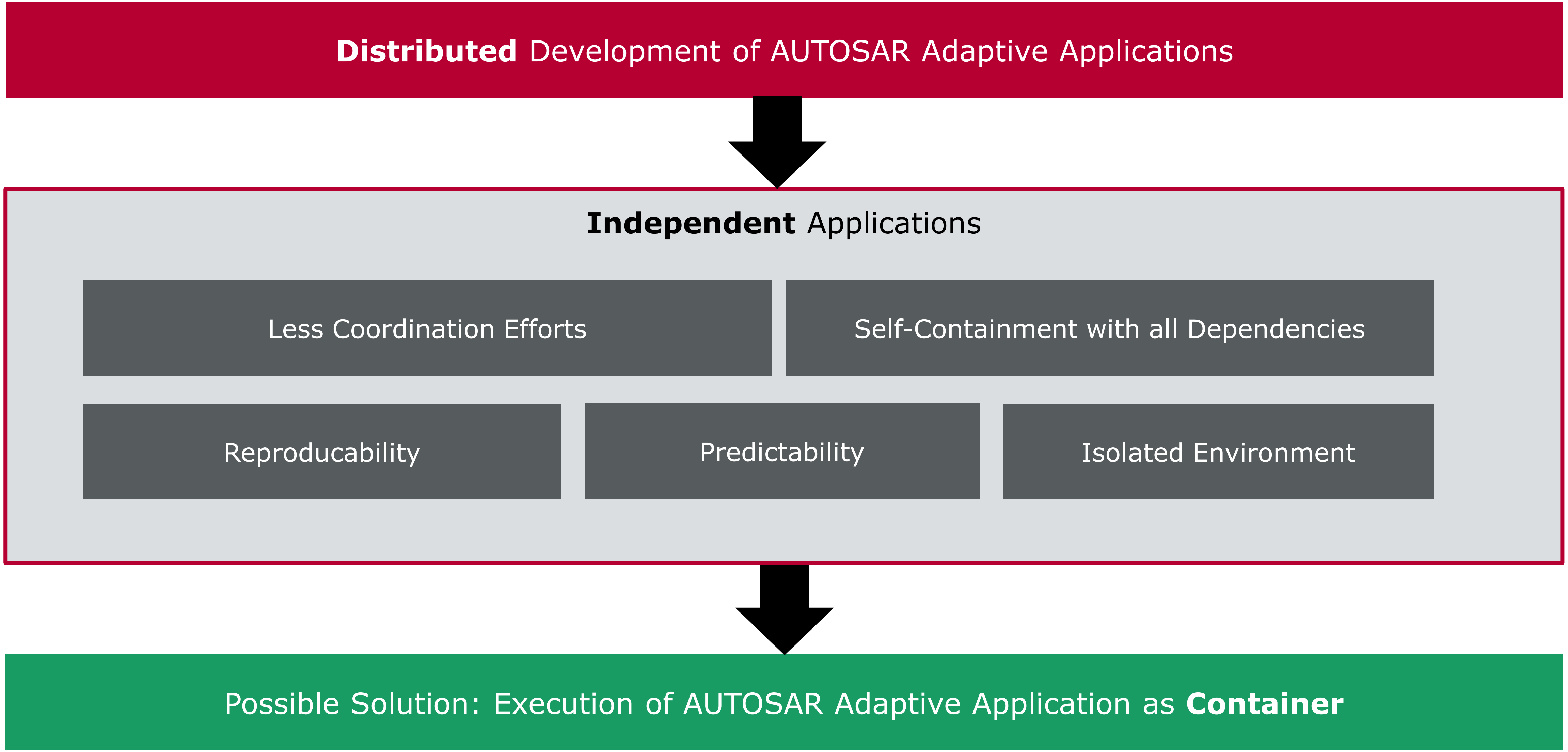
Hi Carlos!
I forgot some dependencies.
Here is the working one.

Hi Bob!
Your steer application conflicts with
Alice‘s drive application at runtime.

Hi Carlos!
Can we organize a meeting with
Alice to resolve this together?

Hi Alice and Bob!
When are you available to meetup?

What we want



Distributed Development of AUTOSAR Adaptive Applications

Independent Applications

Less Coordination Efforts

Self-Containment with all Dependencies

Reproducability

Predictability

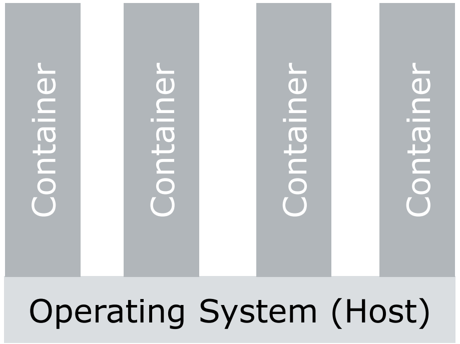
Isolated Environment

Possible Solution: Execution of AUTOSAR Adaptive Application as **Container**

What are Containers?

- Containerization enables to create multiple instances of the „user space“ of an operating system
  - ➜ Operating System-Level Virtualization

- Each instance of the „user space“ is called Container
  - Own Processes
  - Own Filesystem
  - Own Users
  - Own Network
  - Own Resource Assignments



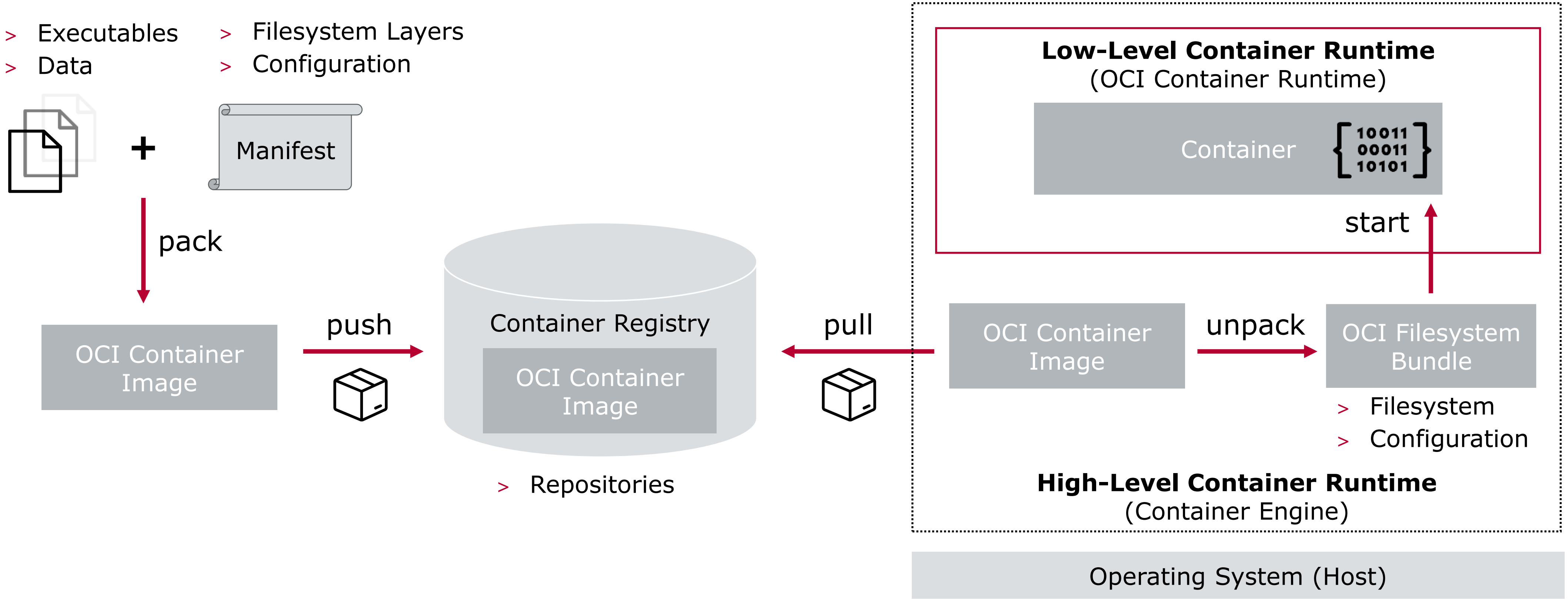
- The „kernel space“ of the operating system (host) is shared among all Containers
  - Kernel Code
  - Resources (CPU, Memory, Network, Storage, Devices)

- Container may access host environment if granted to access global files for instance

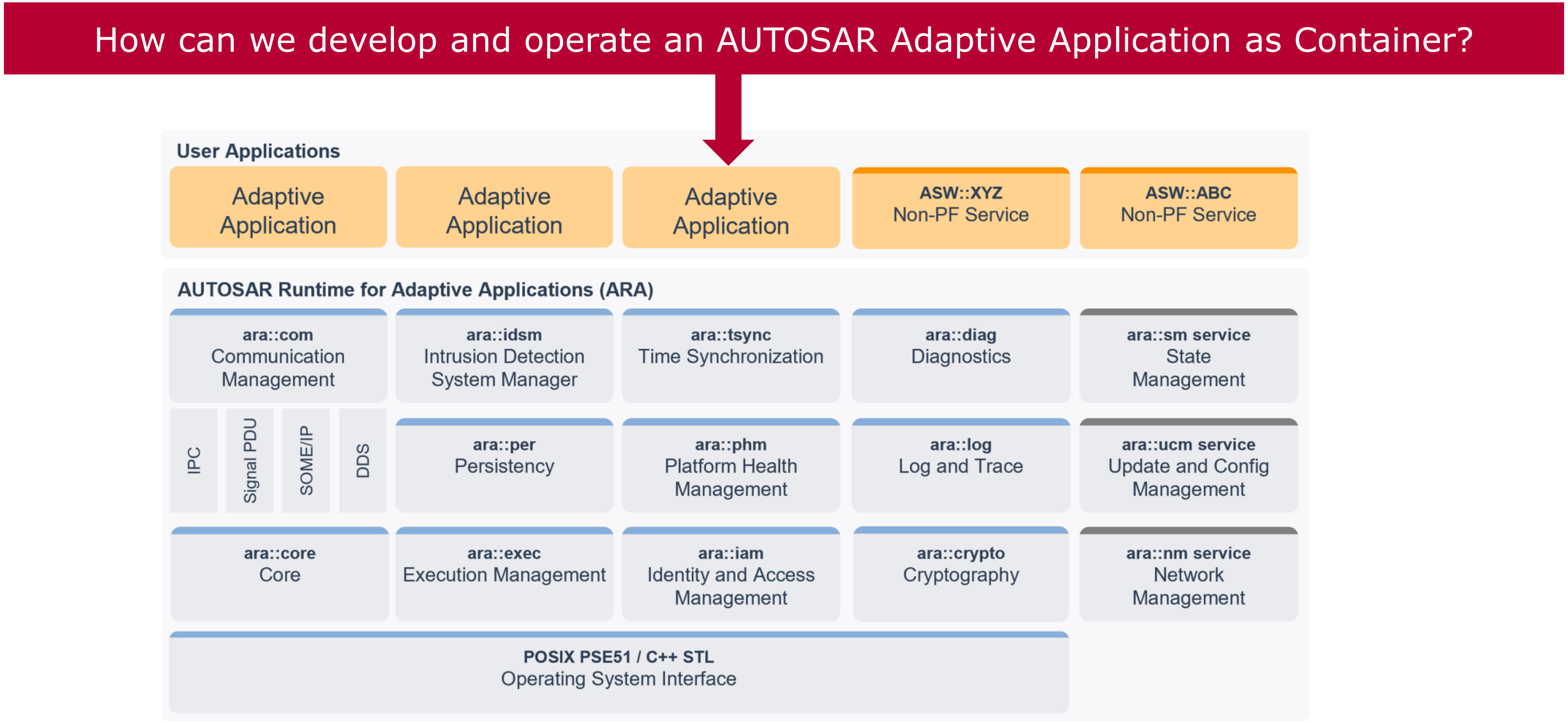
Goal: **Simplify** to develop, test, package, release, deploy and operate software units in a **self-contained** fashion

Container Workflow

- The Open Container Initiative (OCI) maintains open standards around the Container Technology
  - OCI Image Format Specification → Container Image Format (Manifest, Filesystem Layers and Config)
  - OCI Distribution Specification → HTTP-based Distribution Protocol for Container Images; Registry
  - OCI Runtime Specification → Filesystem Bundle, Execution Environment and Lifecycle for Container



Setting the Scene



How can we develop and operate an AUTOSAR Adaptive Application as Container?

User Applications

Adaptive Application

Adaptive Application

Adaptive Application

ASW::XYZ Non-PF Service

ASW::ABC Non-PF Service

AUTOSAR Runtime for Adaptive Applications (ARA)

ara::com Communication Management

ara::idsm Intrusion Detection System Manager

ara::tsync Time Synchronization

ara::diag Diagnostics

ara::sm service State Management

IPC

Signal PDU

SOME/IP

DDS

ara::per Persistency

ara::phm Platform Health Management

ara::log Log and Trace

ara::ucm service Update and Config Management

ara::core Core

ara::exec Execution Management

ara::iam Identity and Access Management

ara::crypto Cryptography

ara::nm service Network Management

POSIX PSE51 / C++ STL Operating System Interface

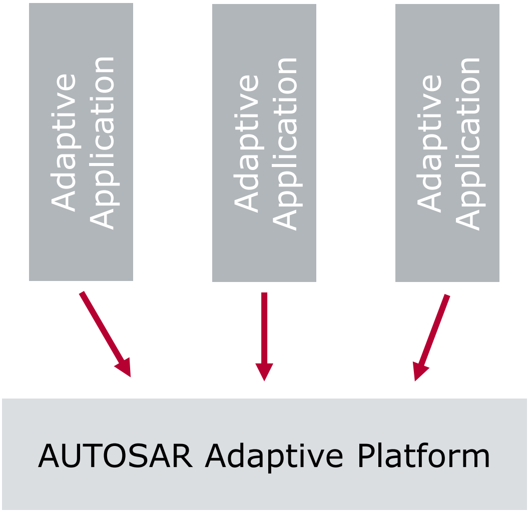
Image Source: Explanation of Adaptive Platform Design AUTOSAR AP R21-11

Challenges

Development

1. Application Design must respect Container boundaries
   - Applications and Platform are modeled as Software Clusters
   - Global manifest is split to enable independent Sub-System Design
   - Service interfaces are used to externalize the functionality of Applications

2. Application Deployment must enable Container independence
   - Executables ship with all their dependencies like libraries
   - Configuration becomes partial and an additive fragment for integration
   - Persistent data is only local

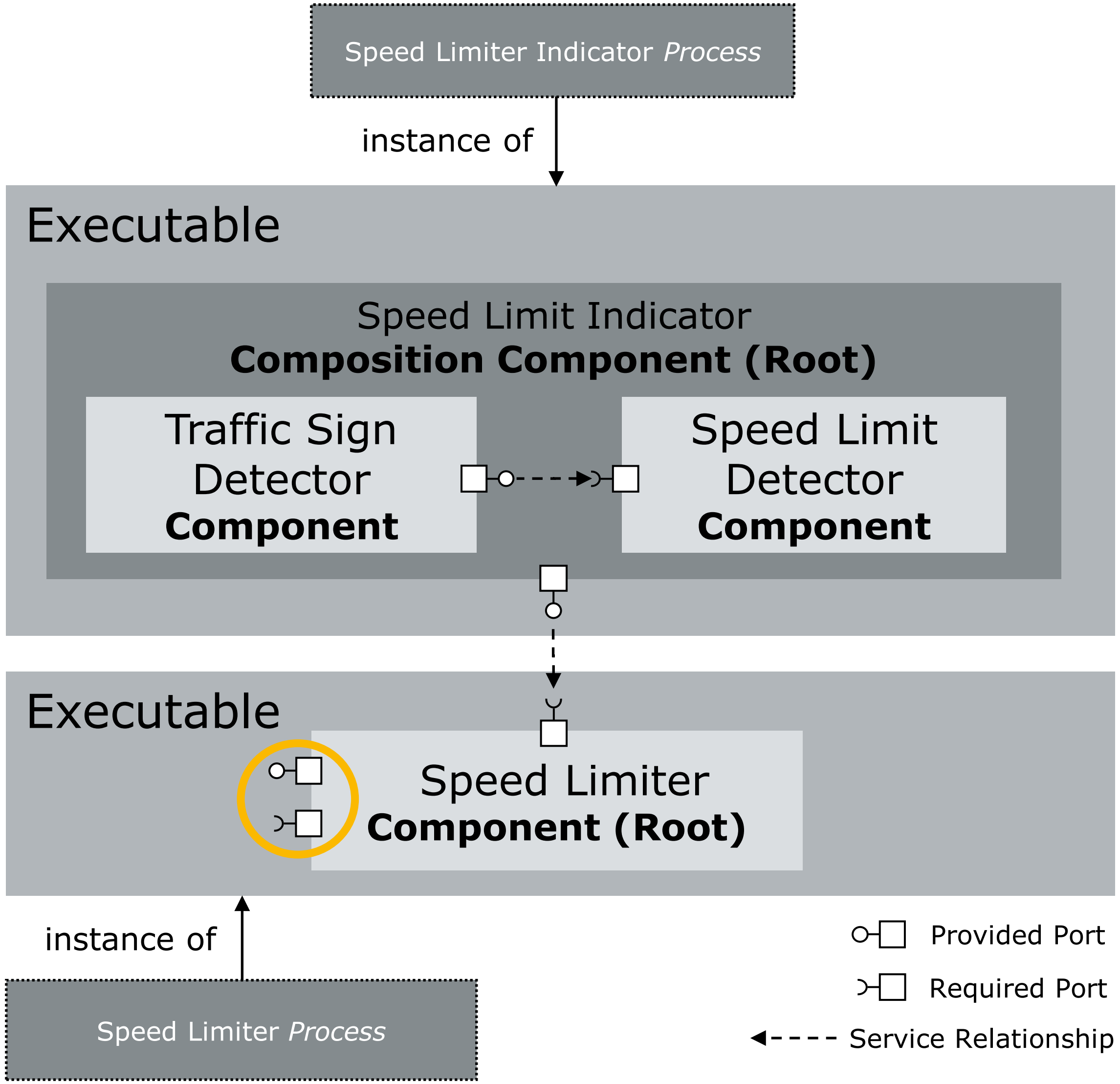


Operations

1. Lifecycle of Container must be managed
   - Container Orchestrator needed to manage lifecycle of multiple Containers
   - Execution, State and Platform Health Management become distributed
   - Update Management must check compatibility of Container during deployment

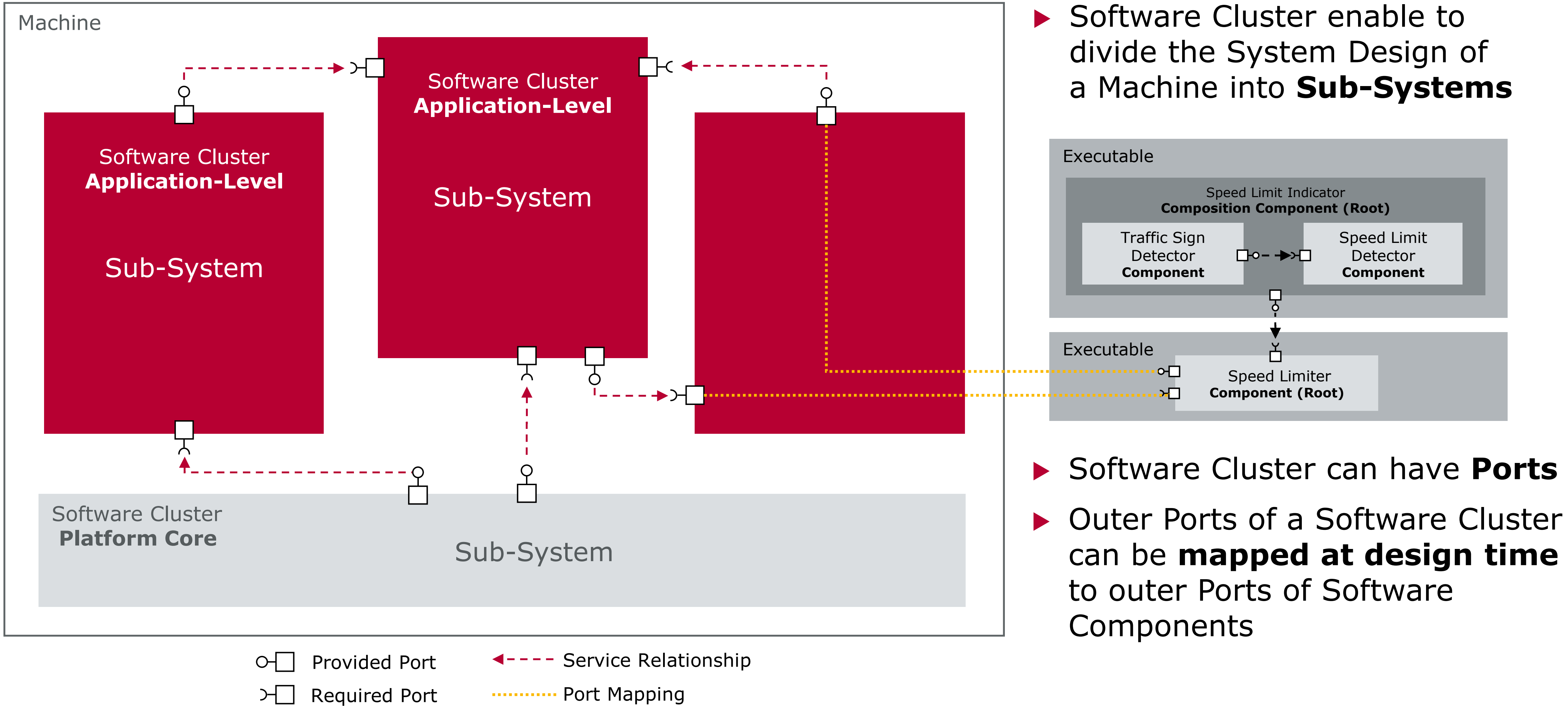
2. All Functional Clusters of Platform must operate cross Container boundaries
   - Configuration Management must access Configuration fragments inside Container
   - Data exchange especially for Communication Management must work cross Container

Application Design with Software Components, Executables and Processes


Speed Limiter Indicator *Process*
instance of
Executable
Speed Limit Indicator
Composition Component (Root)
Traffic Sign Detector **Component**
Speed Limit Detector **Component**
Executable
Speed Limiter **Component (Root)**
instance of
Speed Limiter *Process*
Provided Port
Required Port
Service Relationship

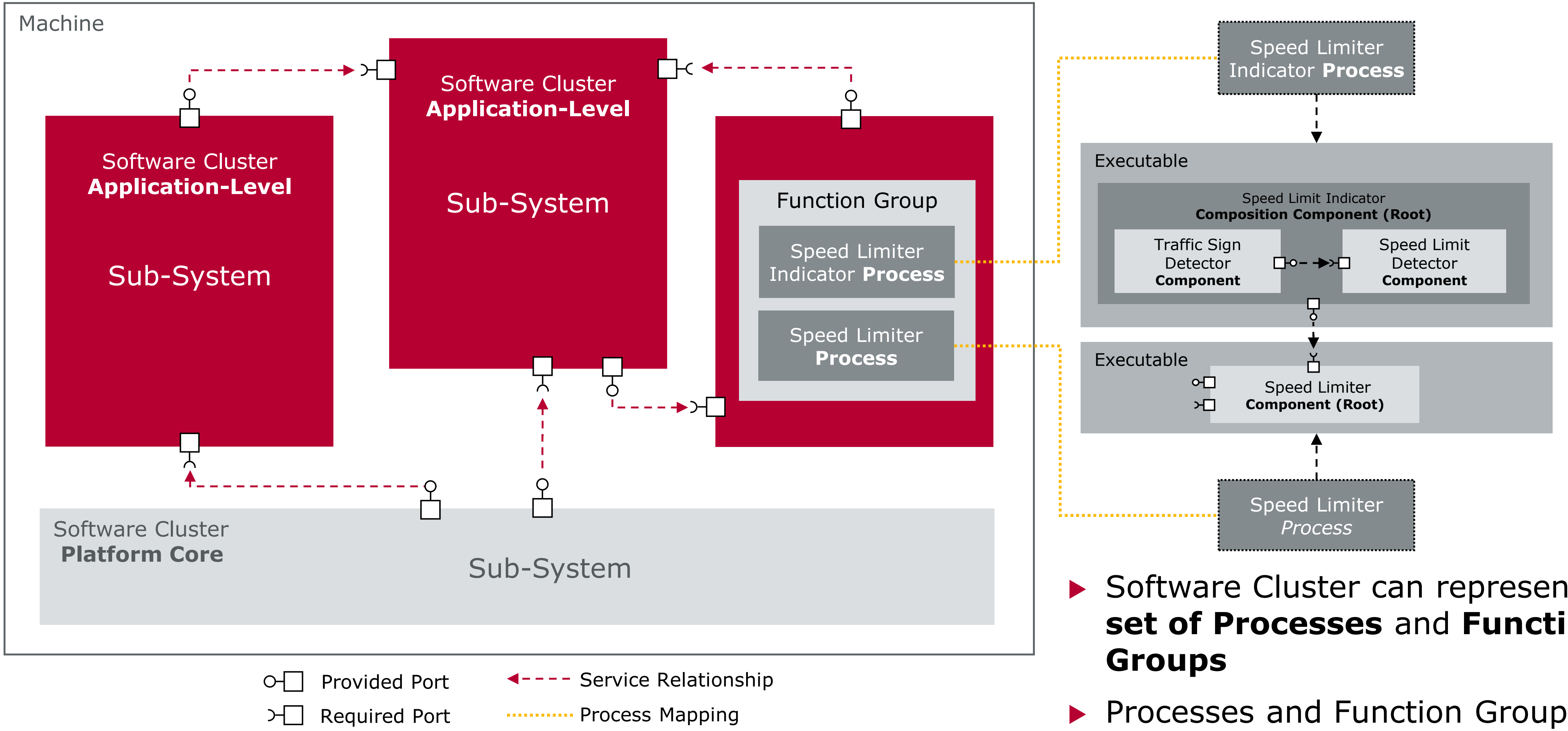
- Executable is a software unit that consist of **one Root Software Component**
- Software components itself can be **composed** of multiple sub-components
- Software components have **Ports**
- Ports enable to **provide** and **require** Services
- Communication between ports is realized **at runtime** with ara::com for instance
- Each Executable can be instantiated as **Processes**

Sub-System Design with Software Cluster



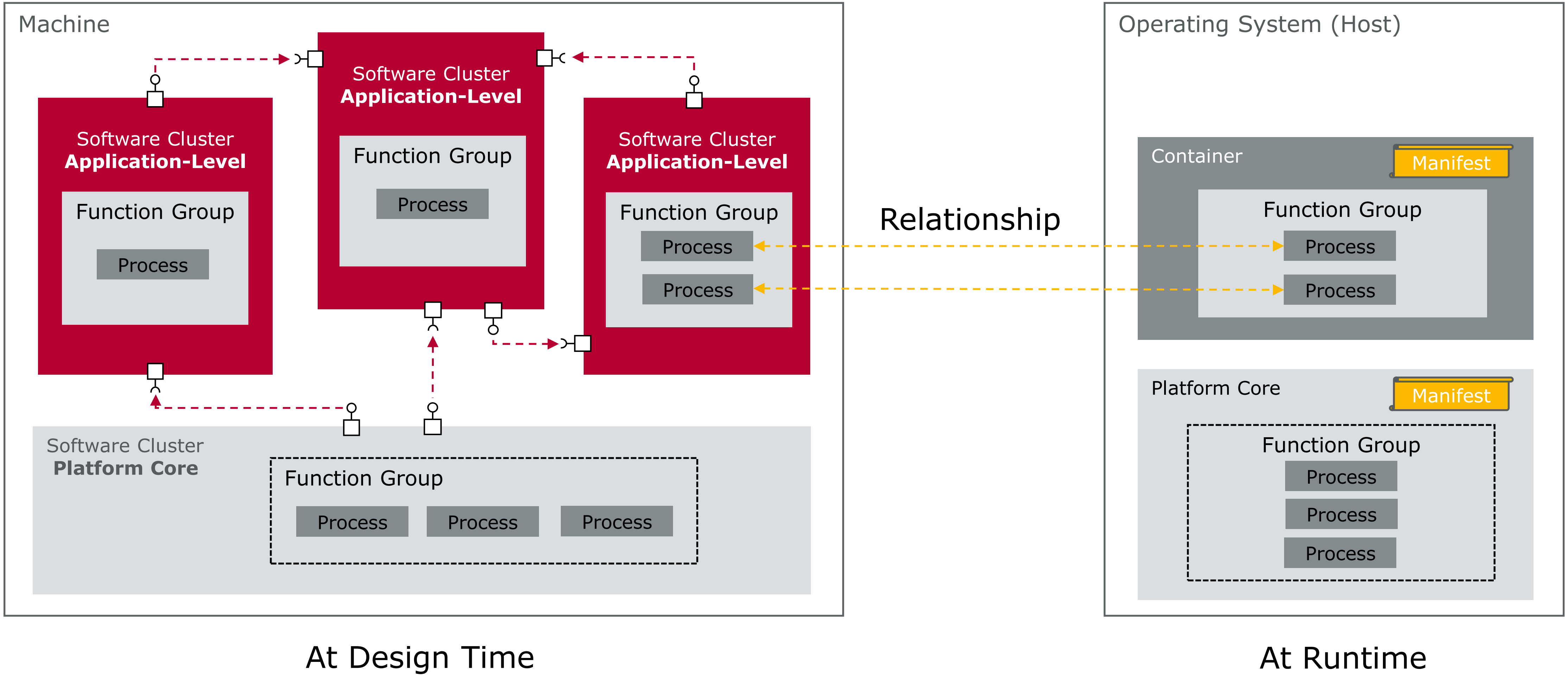
- Software Cluster enable to divide the System Design of a Machine into **Sub-Systems**
- Software Cluster can have **Ports**
- Outer Ports of a Software Cluster can be **mapped at design time** to outer Ports of Software Components

Deployment of Processes as Software Cluster



- Software Cluster can represent a **set of Processes** and **Function Groups**
- Processes and Function Groups are **mapped at design-time**

Concept Proposal

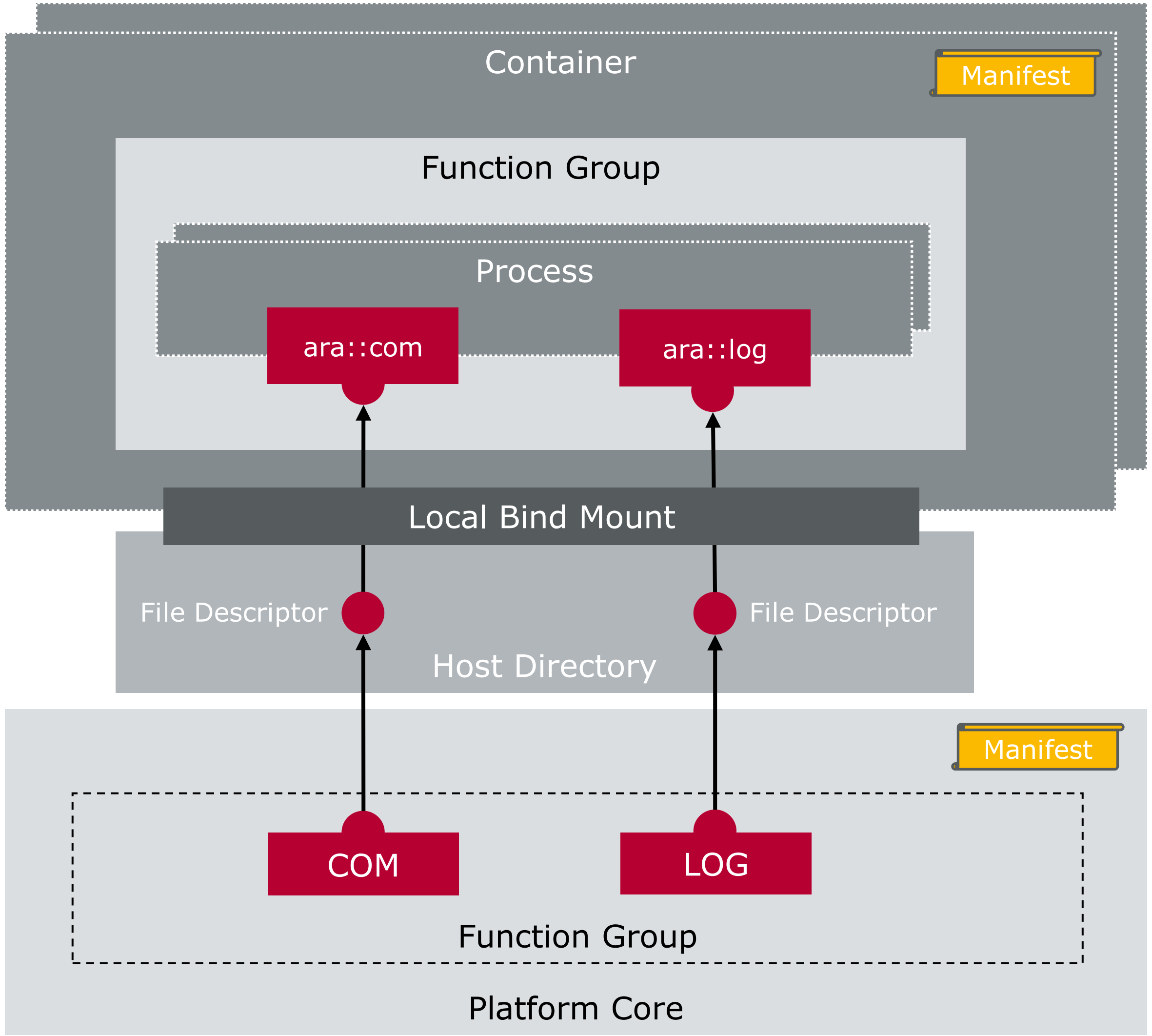


Idea: Execute the Processes of each Application-Level Software Cluster in a **Container**

Concept Proposal: Data Exchange beyond Container Boundaries

- Container and Platform Core need to exchange data for example
  - Service-oriented Communication
  - Log and Trace
- Inter-Process Communication
  - Unix Domain Socket, Shared Memory
  - Accessible typically through Files
- File descriptors are accessible from Containers and Platform Core by using **shared** Host Directories

➜ **Bind mount Host Directory** into Container



Concept Proposal: Distributed Execution and Platform Health Management

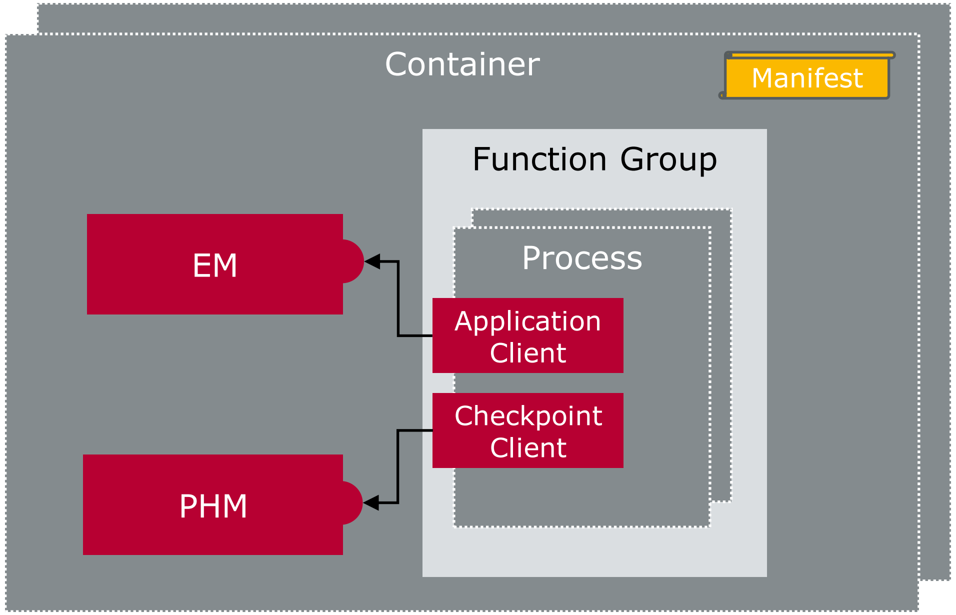
- Each Container has own
  - Execution Management
  - Platform Health Management

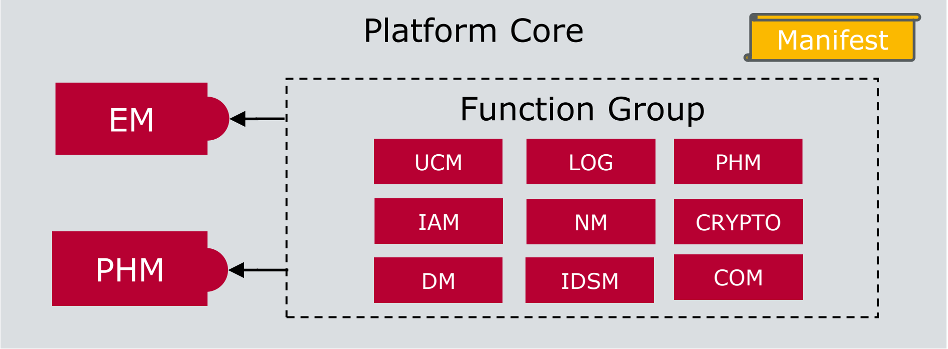
  since those operate on **Process-level**

➜ Execution Management is responsible for lifecycle of Processes **inside** Container

➜ Platform Health Management supervises only Processes **inside** Container

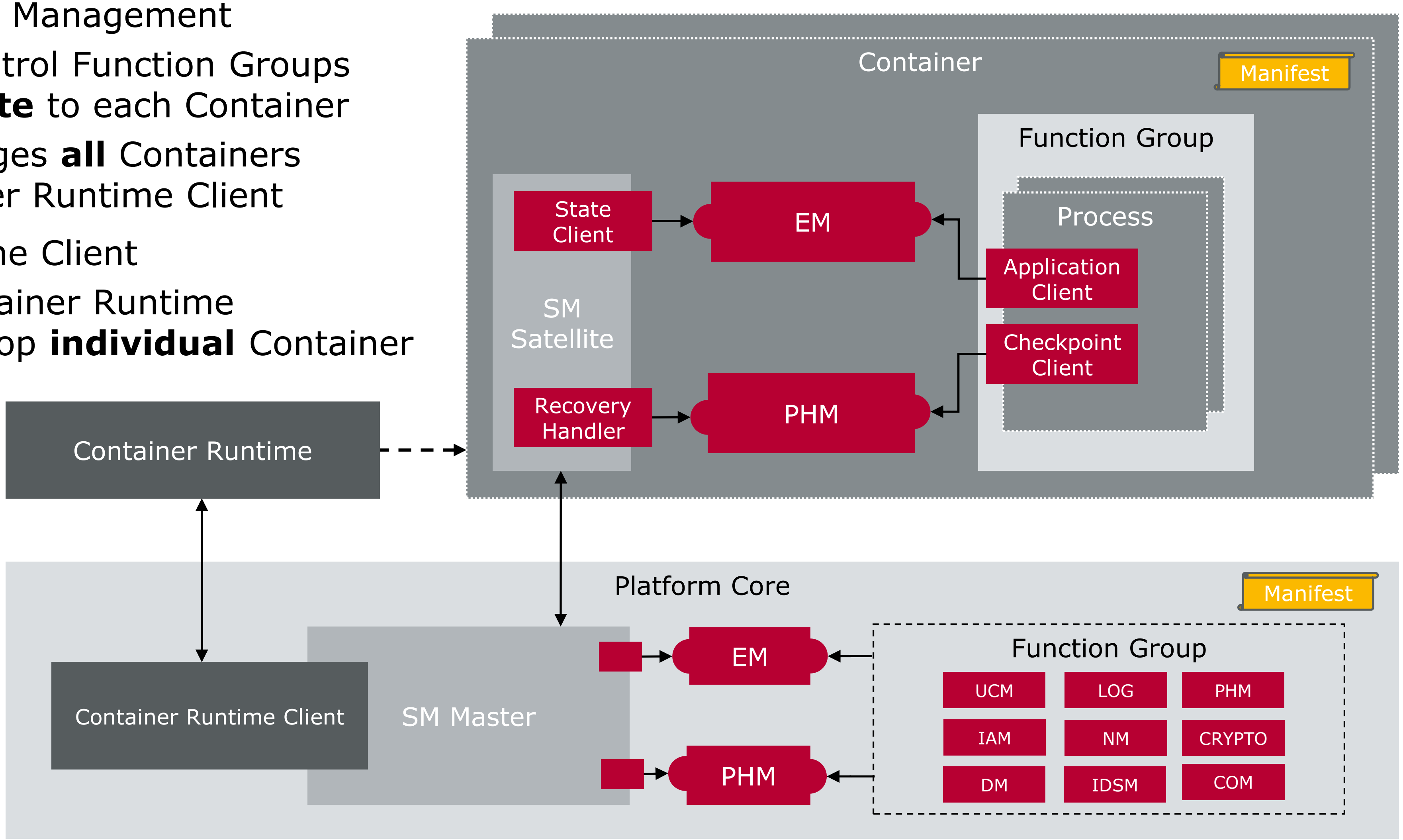
➜ External **Container Orchestration** needed to manage dependencies and lifecycle of **multiple Container**





Concept Proposal: Container Orchestration and Distributed State Management

- Distributed State Management
  - ➜ **Satellites** control Function Groups that are **private** to each Container
  - ➜ **Master** manages **all** Containers using Container Runtime Client
- Container Runtime Client
  - ➜ Instructs Container Runtime to start and stop **individual** Container



Some of the Open Questions towards a Final Solution

Final Solution must also ensure Functional Safety

Execution Management

- Are multiple Processes (Co-Location) per Container desired?
- How does the Container Lifecycle and Function Groups interplay?

State, Platform Health and Network Management

- Do we want to control the Lifecycle of Container depending on Partial Network Requests?
- How does the Liveness Supervision of Container and their Processes work?
- What are suitable Recovery Strategies if Container and their Processes do not react?

Diagnostic, Update and Configuration Management

- How is the interplay between Container Runtime and Update Management?
- How to integrate Update Workflow of Container with Container Registries?
- How do we access and apply the Configuration Fragments (Manifest) of each Container?
- How do we handle Diagnostic services and addressing?

Security

- How do we manage Access Policies for shared resources between Container?

Towards loosely coupled Systems

Distributed Development of AUTOSAR Adaptive Applications

- Software Cluster enable independent development of AUTOSAR Adaptive Applications
- Container are a possible execution environment for the Processes of Software Cluster
- Container Orchestration needed to manage lifecycle of multiple Container
- Distributed Execution, State and Platform Health Management is a consequence
- Affects Update, Configuration and Diagnostic Management and Security
- Future Work: Consider execution of Functional Cluster Processes of Platform in Container

Contact us if you are interested! We are open for projects to enable this vision.

For more information about Vector
and our products please visit

www.vector.com

Author:
Suffel, Max-Ferdinand
Vector Germany