Software Orchestration & Resource Scheduling Summary Report

Eclipse SDV Community Days,
March 29 and 30

Lisbon

SW Orchestration & Resource Scheduling
The Team

Aymeric Rateau (Toyota),
Badii Ennouri (badii.ennouri@gmail.com)
Daniel Krippner (ETAS GmbH)
Fergus Duncan (CapGemini)
Filipe Prezado (Microsoft),
Holger Dormann (Elektrobit),
Kai Hudalla (Robert Bosch GmbH),
Liviu Tiganus (Microsoft),
Sergey Markelov (Microsoft),
Sven Erik Jeroschewski (Robert Bosch GmbH),
Thilo Schmitt (Mercedes-Benz Tech Innovation),

Please add yourself to the list.

SW Orchestration & Resource Scheduling
Our Goal

Define ~~one~~ a concrete software architecture that can be implemented on an in-vehicle computer which supports the use cases described below.

SW Orchestration & Resource Scheduling
Some Definitions

Software Orchestration and Resource Scheduling refers to a set of software components that work together to support deployment and management of In-Vehicle Resources.

In-Vehicle Resources consists of one or more software components that need to be installed and managed in the target in-vehicle runtime environment.

Software components can be applications, containers, firmware. Software components often also include (or might require) configuration data to adapt the software components to the target in-vehicle runtime environment.

Containers are a form of operating system virtualization. A single container is a way of encapsulating a set of software components. The set of software components is installed in a Container and is removed with the Container. This does not "pollute" the host OS. Container A can have a software component of a conflicting version or nature that is in Container B, but since all data and software components are encapsulated in Container A and Container B respectively, that does not manifest itself as a problem on the host OS.

Control Plane - Provides a management layer that enables deployment, update and deletion of in-vehicle resources. It ensures that every in-vehicle resource is kept in the desired state. Makes global decisions as well responds to events.

SW Orchestration & Resource Scheduling
Some Principles (and constraints just to start)

Platform and Protocol Neutral
Focus on the software architecture and try to be agnostic of concrete operating system and hardware platform. In particular, we should not assume to run on bare metal or a virtual machine managed on top of a hypervisor as, for example, proposed in the SOAFEE project.

Extensible architecture supporting a broad range of devices, hardware, and software components
Start with in-vehicle computer is an x86_64 or arm64 based system with at least 8GB of RAM running a Linux kernel based operating system (non-RT), e.g. Automotive Grade Linux (AGL), Debian, custom Yocto-based etc.

1.

Open Source
The software components being used are required to be developed within the context of (existing?) open source projects.

1.

Do not re-invent the wheel.
We should strive for (re-)using, aligning and extending existing software wherever reasonably possible.
Highlight EB, Kanto and Muto projects

Should be consumable (quality requirements) by the OEM's, Tier1,s
And potential reference testing & validation principles from Johannes workstream
(make this codebase an example of "automotive-grade" code quality to be adopted and consumed)

SW Orchestration & Resource Scheduling Use Cases

UC1 - Deploy and Manage non-safety critical application

Automate deployment of a (multi-container/-artifact) application into the in-vehicle computer, schedule, manage and run it.

•

UC2 - Deploy and Manage mixed-criticality application components to multiple targets

Automate deployment of an application that consists of artifacts of mixed criticality (QM & ASIL-B...) that need to be installed, managed and run in (a containerization platform) dedicated platforms/controllers.

•

UC3 - Dynamically deploy application components to multiple targets (wishlist)

Dynamic deployment of application containers into the containerization platform across the different controllers/partitions based on resource availability.

•

UC4 Download and Install application components to multiple targets (wishlist)

Download and manage the installation of the application components (inc. configuration and dependencies) across the multiple targets within the vehicle E/E platform - the automotive application components are generally distributed over multiple targets within the vehicle. For example, an embedded control unit (ECU) that provides safety-critical functionality needs to have its firmware updated and/or an AUTOSAR application needs to be deployed using AUTOSAR Update and Configuration Management (UCM).

SW Orchestration & Resource Scheduling
Other related Projects

- COVESA - in particular, the Vehicle Signal Specification might be a candidate for vehicle abstraction.
- Aos - Edge Orchestration Platform
- Rancher/k3s - Kubernetes implementation for small devices
- Flatpak - Packaging and distributing applications for Linux
- CNAB: Cloud Native Application Bundles -Cloud Native Application Bundle
- Autoware Open AD Kit promises to accelerate the development of optimized hardware and software solutions for autonomous driving
- SOAFEE
  - ~~Promote overall vision (as laid out in this presentation)~~
  - Push subgroups to address some implementa
  - Mixed Criticality portability
  - Secure standard device assignment for type 1
  - Freedom of interference from secure firmware
- Linaro
  - Trusted sensors (sensor signing component from different entity than the sensor business logic) => link to confidential compute
- For Use Case 4:
  - AUTOSAR (UCM)
  - Uptane - An open and secure software update framework design which protects software delivered over-the-air to automobile electronic control units (ECUs)
  - TOSCA - Topology and Orchestration Specification for Cloud Applications

@Daniel, @Holder - Highlight also the identified challenges related to run k8s in existing in-vehicle compute capacity

SDV
Eclipse Software Defined Vehicle

SW Orchestration & Resource Scheduling Requirements identified (not exhaustive)

MUST HAVE support for managing in-vehicle compute as Pets (and not Cattle) (refer to Cattles vs Pets terminology/history)
SHOULD HAVE support for software component deployment ordering across in-vehicle target runtimes
MUST HAVE support for software component rollback capabilities/features
COULD HAVE support for deterministic application start-up ordering
COULD HAVE support for failover strategies
WON'T HAVE support for horizontal scaleout (as is not an immediate concern)
MUST/SHOULD/COULD/WON'T support updating operating system and/or firmware of ECUs.
MUST/SHOULD/COULD/WON'T support updating autosar classic applications
MUST/SHOULD/COULD/WON'T support for Open Container Initiative
MUST/SHOULD/COULD/WON'T support for Certified Kubernetes Software Conformance
MUST/SHOULD/COULD/WON'T support for integration with OTA/FOTA software
MUST/SHOULD/COULD/WON'T be a replacement for OTA/FOTA software
MUST/SHOULD/COULD/WON'T support for end-to-end Observability using OpenTelemetry
MUST/SHOULD/COULD/WON'T support for computational nodes like ECU's/MCU's
MUST HAVE support for ordering of applications at startup
COULD HAVE support for rollback
COULD HAVE self healing capabilities
COULD HAVE self governing capabilities
MUST HAVE support for offline/online network connectivity
MUST HAVE support for offline/online hardware internal state

To be Completed

SW Orchestration & Resource Scheduling Community Call for Action

- Contribute
  - review requirements and provide feedback
  - Use the Assessment as a guiding tool
- Can we align implementation of these requirements across projects? Eg: Kanto, Muto, … => unify developer base
- Outlook
  - Rust/compiler certification => make a ready-for-cert showcase out of Orchestrator project?
  - Collaboration potential with eg Ferrous, Exida, Codethink, etc


Assessment Questionnaire (Draft) for Eclipse SDV Projects

Control Plane

- Which aspects are most important to create an experience of easy deployments?
- Do you require integration of in-vehicle control plane with a cloud control plane for management?
- What are target memory and CPU footprint / target specs for in-car hardware?
- Do you require the control plane to be aware of existing/"legacy" network infrastructure (ie, CAN, LIN Bus)?
- Do you require the control plane sw to be a high availability component (even in single node setup)?

grams?

- Do you require dynamic prediction of application resource requirements ?
- Do you require application load balancing and service/application discovery?
- Do you require dynamic allocation of applications on single node? and multi node?
- Do you require application auto-scaling?

Package/Configuration formats

- What application package and formats do you support or would like to be supported?

@Daniel, @Kai, @Holder, we should reach out to Kanto team and Naci to confirm this statement proposal.

[Assessment Questionnaire (Draft) for Eclipse SDV Projects · Wiki · Eclipse Working Groups / SDV WG / sdv-technical-alignment / sdv-technical-topics / vehicle-SW-orchestration-sdv-topic · GitLab]

SDV Eclipse Software Defined Vehicle