Software-Defined Vehicle Support and Coordination Project

D3.1 First prioritized backlog report

Author: Daniel Krippner & Detlef Zerfowski

October 2024

Deliverable	D3.1 First prioritized backlog report
Work Package(s)	WP2 - Technology and high-level requirements solicitation WP3 - Common Understanding
Dissemination Level	public
Due Date	30-09-2024
Actual Submission Date	22-10-2024
WP Leader	Detlef Zerfowski, ETAS GmbH
Deliverable Leader	Detlef Zerfowski, ETAS GmbH
Contact Person	Detlef Zerfowski, Daniel Krippner ETAS GmbH
Email	

Document History			
Revision No.	**Date of the review**	**Name of the reviewer**	**Status of the document** ***(in progress, ready for review, released)***
V0.1	29-08-2024	D. Zerfowski	In progress
V0.1	29-08-2024	M. Driussi	In progress
V1.0	10-09-2024	M. Driussi	Done
V1.1	15-10-2024	D. Krippner	Done
V1.2	21-10-2024	P. Priller	Finalized content

The project has been accepted for funding within the Chips Joint Undertaking (CHIPS JU), a public-private partnership in collaboration with the Horizon Europe (HORIZON) Framework Programme under Grant Agreement No. **101139749**

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the granting authority can be held responsible for them.

Table of Contents

Tables

Figures

1. Executive Summary

The FEDERATE project, aimed at advancing the development of software-defined vehicle (SDV) components, has made significant strides in its early stages, focusing on the collection, organization, and alignment of use cases and requirements inputs. WP2 and WP3 played an important role in several key areas, contributing to the project's success, and ensuring that foundational elements are effectively established.

A primary deliverable of the project was the creation of a comprehensive GitHub repository, which houses an elaborate folder structure as organizing principle for SDV software building blocks. GitHub was determined to be the most suitable option due to its superior user management capabilities compared to GitLab.

This document describes methodology and first results for the prioritized backlog. Main focus is on how building blocks (BB) are described and prioritized. It is strongly connected with D3.4 (First orchestrated backlog report)

An efficient workflow for creating and reviewing issues on the platform was established, ensuring smooth collaboration among project participants. WP2 and WP3 hosted discussions around the organizational structure of the SDV building blocks, providing feedback and insights that shaped the final framework. Furthermore, a first set of use case scenarios was collected, which were reviewed, adapted, and accepted into the project repository.

2. Introduction

The purpose of this report document is to provide an account of the contributions generated by of WP2 and WP3 within the scope of the FEDERATE project. It aims to outline the key contributions, methodologies, and outcomes, particularly focusing on the creation and organization of the SDV software building blocks repository. By providing a comprehensive overview of these activities, the report ensures transparency, facilitates knowledge sharing among project stakeholders, and supports ongoing and future work within the FEDERATE project by offering a clear reference point for the project's progress and achievements thus far.

2.1. Aims and Objectives of the Deliverable

The primary aim of the deliverable is to describe the comprehensive and well-organized repository of SDV software building blocks, which will serve as a foundational resource for the development of software-defined vehicle components that can be supported and collaboration-aligned by the FEDERATE project. This repository is designed to collect and organize use cases and requirements inputs, enabling the project team to systematically approach the development of SDV software components. The objective is to provide a robust framework that facilitates collaboration among project participants, supports the efficient creation and review of building blocks, and ensures that the SDV components developed can be in line with industry standards and best practices.
This document describes methodology and first results for the prioritized backlog. Main focus is on how building blocks (BB) are described and prioritized. It is strongly connected with D3.4 (First orchestrated backlog report)

2.2. Context and Fit within the Project

This deliverable fits into the broader context of the FEDERATE project by addressing one of its core goals: the systematic development of SDV software components that can be utilized in the automotive industry. The repository serves as a central hub for organizing the building blocks that will form the basis of these components, ensuring that all relevant use cases, scenarios, and requirements are documented and accessible to the entire project team, as well as to organizations that want to contribute to building block implementation. By providing a clear and structured framework, this deliverable ensures that subsequent development activities are grounded in well-defined and widely reviewed inputs.

2.3. Interdependencies and Integration with Other Tasks

The development of this deliverable is interdependent with earlier tasks and deliverables. For instance, the selection of GitHub as the project hosting platform and the establishment of a workflow for creating and reviewing issues were critical prerequisites for the successful creation of the repository. These earlier tasks provided the technical and procedural foundation necessary for organizing and managing the SDV building blocks.

Furthermore, the scenarios and use cases included in this repository are derived from initial discussions and inputs from various project stakeholders, ensuring that the repository reflects a wide range of perspectives and requirements. As such, this deliverable is built on the collaborative efforts and contributions of the entire group of participants.

Going forward, this repository will serve as a key input for subsequent tasks within the FEDERATE project. It can be used by teams responsible for developing specific SDV software components, ensuring that their work is informed by the comprehensive use cases and scenarios documented in the repository. Additionally, the structure and organization principles established in this deliverable will guide future updates and expansions of the repository, as new requirements and building blocks are identified and integrated into the project.

3. Contribution and task description

In this section, we describe the tasks which were essential to laying the groundwork for the development of software-defined vehicle (SDV) component descriptions and ensuring that the project's deliverables were aligned with its overarching goals.

3.1. Evaluation of Project Hosting Platform (GitHub vs. GitLab)

One of the first tasks was supporting the evaluation of potential project hosting platforms. The choice between GitHub and GitLab was critical as it would impact the efficiency of collaboration, user management, and overall project organization. After a comparison of the two platforms, considering factors such as user access control, integration capabilities, community support, and scalability, GitHub was chosen as the most appropriate platform for FEDERATE artifacts. The decision was based primarily on GitHub's superior user management features, which allow for more granular control of permissions and roles, making it easier to manage a diverse set of contributors across various project stages. This decision has provided a stable foundation for the project, enabling seamless collaboration and more effective management of project activities.

It is worth noting that the project hosting decision initially leant towards benefitting from close collaboration with the Eclipse Foundation, one of the major Open Source Foundations that already driving a very active SDV software community and have expressed interest to support and align with FEDERATE and related projects. Eclipse Foundation personnel supported the setup of a project space and demonstration/onboarding of FEDERATE members, up to the point where project work could begin. However, after one project member expressed reservations about using Eclipse Foundation infrastructure, the FEDERATE team pivoted to evaluating and setting up its own and therefore independent artifact hosting space, as described above.

3.2. Proposing and Testing an Appropriate Workflow for Creating and Reviewing Issues

To streamline collaboration and ensure that all project participants could effectively contribute, we proposed and tested a workflow for creating and reviewing issues on the chosen GitHub platform. This workflow was designed to facilitate clear communication, accountability, and traceability throughout the project's lifecycle. It involved reviewing standardized templates for issue creation, aligning guidelines for assigning issues, and establishing review and approval processes. We also conducted test runs of the workflow to ensure its practicality and efficiency. This task was crucial in ensuring that the project could handle the complex and iterative nature of SDV component development, with all contributions being properly tracked and reviewed.

The workflow suggested and evaluated by the project team is based on best practices from the open source community: A standard PR (Pull Request, submission of new content or changes to existing content) review-accept-merge workflow on GitHub for markdown content typically begins with a contributor creating a new branch from the main branch, where they make their changes or additions to the markdown files. Once the changes are complete, the contributor opens a Pull Request, which prompts a review process.

During the review, team members are notified and can review the proposed changes directly on GitHub. Reviewers can leave comments, request modifications, or approve the changes. If modifications are requested, the contributor makes the necessary adjustments on the same branch, and the Pull Request is updated automatically. Once the reviewers are satisfied with the changes, they approve the Pull Request.

Following approval, the Pull Request is merged into the main branch, typically using a "Merge" button on GitHub. The branch used for the changes can then be deleted, keeping the repository clean and organized. This workflow ensures that changes are thoroughly reviewed and properly integrated into the project.

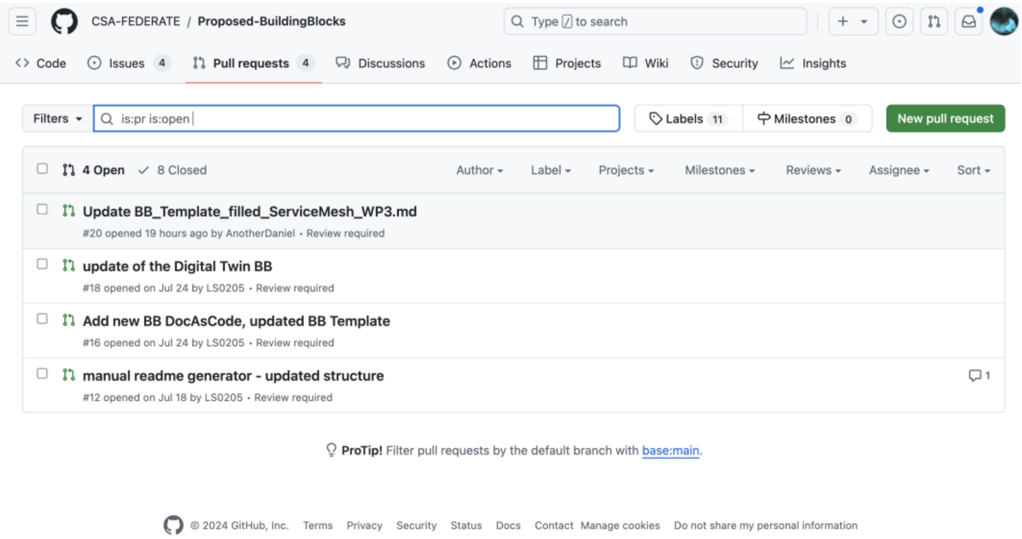

Figure 1 - Snapshot of backlog items (PRs)

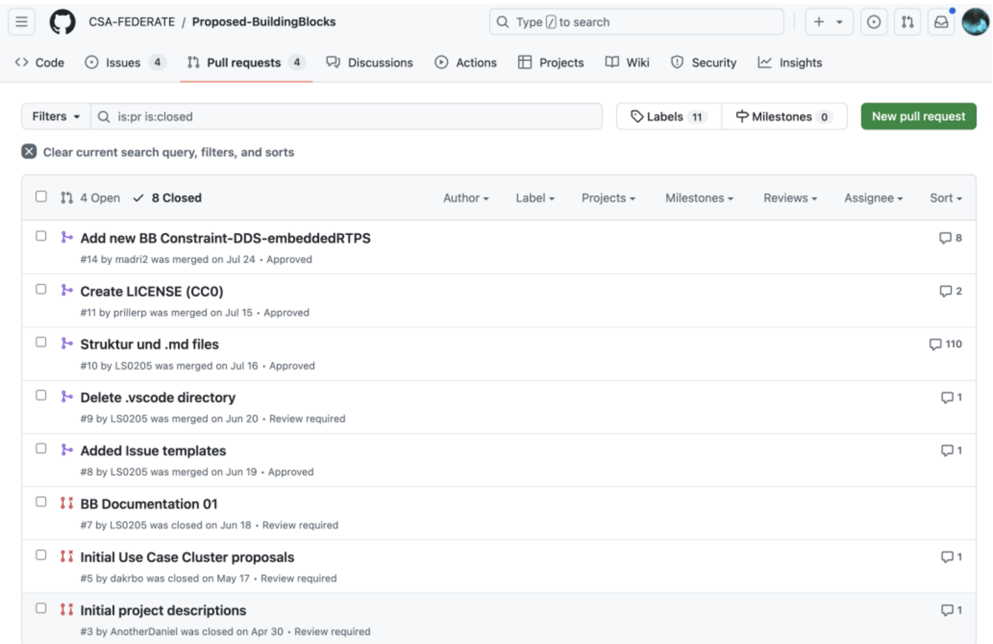

Figure 2 - Snapshot of closed work items (PRs)

3.3. Discussing and Reviewing the Organizational Structure for SDV Building Blocks

The organization of SDV building blocks within the project repository is a critical aspect of the FEDERATE project. This includes shaping the structure of these building blocks, engaging in discussions with project stakeholders to determine the most logical and efficient way to categorize and store them. We reviewed various organizational models, considering factors such as modularity, scalability, and ease of access. The final organizational structure not only supports the current project needs but is also adaptable to future expansions. This structure has become the backbone of the repository, ensuring that all building blocks are systematically organized and easily accessible for future development efforts.

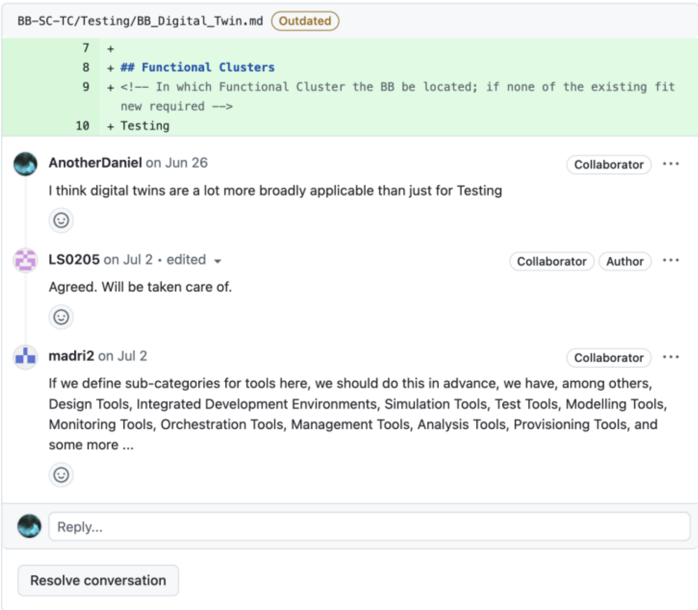

Figure 3 - Example outtake of review discussion

3.4. Building Block template and properties

The project team has defined and agreed on a template structure for building blocks (refer to Appendix 5.b), which contains a number of ordering and structuring elements that help allocating each building block into the larger SDV architecture structure for easier generalization and discoverability of building blocks. Alongside the template, the project team also provides an implementation guideline document [7] to aid authors with documenting new building blocks.

It is worth pointing out that at this time, the project is primarily in the Building Block collection phase, and while some fields are provided to allow Building Block authors to indicate their assessment of Building Block relevance etc., the current set of documents does not represent a uniformly reviewed state where all project partners have agreed on all such assessments.
For instance, Building Block documents can define a “Priority” which is suggested to be one of ‘low’, ‘medium’ or ‘high’. However, this is not an absolute valuation and should rather be considered indicative of the authors sense of urgency about the documented Building Block.

Going forward, if the project team decides to perform an absolute priorization of the building block catalogue that the entire group can agree on, the Priority field might become more relevant as a decision input for companies that are looking where to assign development resources.
This kind of globally comparative evaluation requires dedicated statements from automotive product companies as to which functionalities are needed most urgently; it is unclear at the moment if and when such statements may be forthcoming.

3.5. Creating a First Set of Use Case Scenarios for Building-Block Descriptions

To jumpstart the documentation of SDV building blocks, we created an initial set of use case scenarios. These scenarios were developed based on research and industry experience, ensuring that they accurately reflected the potential applications and requirements of the SDV components. Each scenario was crafted to provide clear, actionable insights into how the building blocks could be used in real-world contexts. This initial set of use cases provide a foundation for subsequent building-block development, ensuring that they are grounded in practical, real-world applications.

Vehicle Fleet Data

BB Tags(s)

- SC

Functional Clusters

- Communication
- Storage
- Security

Layer

- Middleware
- Application

Known Implementation

- Eclipse Fleet Management blueprint: https://github.com/eclipse-sdv-blueprints/fleet-management

ID (unique name)

Description

The Goal of this use case cluster is systems, abstractions and capabilities related to the extraction of data from the in-vehicle mechatronics domains. This includes both higher layers of abstraction and processing in in-vehicle SDV Software stacks, as well as routing to off-board systems (backend, mobile, etc). More specific use cases in this cluster might initially address commercial fleet management (FMS) functionality (where bespoke use cases and related data needs are generally well known, and joint definition work is already ongoing e.g. in COVESA). The following key aspects are written from the point of view of an in-vehicle SDV Software stack. This use case cluster is focusing on data and data flows that are not hard real-time critical, and do not have safety implications beyond basic ASIL QM.

Figure 4 - Example use case scenario

SOA

BB Tags(s)

S-BB

Functional Clusters

Layer

MWLayer

Known Implementation

Eclipse uProtocol

ID (unique name)

Description

Service oriented communication allows services to communicate between ECUs with each other in the vehicle typically by publish-subscribe mechanism. Typically this is enabled by middleware framework implementing a service protocol (SOME/IP, DDS,..) and providing necessary interface connectors in various programming languages.

Rationale

Standardization of SOA communication services across different vehicle manufacturer

Figure 5 - Example SW Building Block: SOA Middleware

4. Conclusions

The work of WP2 and WP3 has been instrumental in laying the groundwork for the successful development of software-defined vehicle (SDV) components. This includes evaluating the project hosting platform, proposing and testing workflows, and shaping the organizational structure for SDV building blocks. This has established a solid foundation that supports collaborative and systematic development within the project. The next step was creating and refining use case scenarios. All together is the base for the ongoing work in the next months.

5. Appendices

a. Building Blocks

Table 1 - Snapshot of Building Blocks

Building Block type	Building Block name	Priorisation (exemplary)
BB-CEST/_Not_Clustered	BB_Car_Simulator	
BB-SC-TC/Testing	BB_Shadowing	
BB-SC-TC/Virtualization	BB_Digital_Twin	High
BB-SC/AppLayer/Communication	BB_AOSP_Push_Notification_Service	until 30.6.2025
BB-SC/MWLayer/Communication	BB_Communication_Server_S2S	Low
BB-SC/MWLayer/Communication	BB_Constraint_DDS_embeddedRTPS	
BB-SC/MWLayer/Communication	BB_Gateway_Mirroring	Low
BB-SC/MWLayer/Communication	BB_Network_Management	High
BB-SC/MWLayer/Communication	BB_SecOS	Low
BB-SC/MWLayer/Communication	BB_Smart_Charging_Communication	Low
BB-SC/MWLayer/Communication	BB_Standard_Android_VHAL	
BB-SC/MWLayer/Configuration	BB_Local_Update_Manager	High
BB-SC/MWLayer/Configuration	BB_OTA_Master	High
BB-SC/MWLayer/Diagnostics	BB_Policy_Manager	Low
BB-SC/MWLayer/Platform-Health-Management	BB_Distributed_Health_Management	Low
BB-SC/MWLayer/Platfor	BB_Watchdog	Low

m-Health-Management		
BB-SC/MWLayer/Power-Management	BB_Power_Management_Coordination	High
BB-SC/MWLayer/Runtime	BB_Diagnostic_Services_Applications	Medium
BB-SC/MWLayer/Runtime	BB_State_Management	High
BB-SC/MWLayer/Security	BB_Crypto_Service_Manager	High
BB-SC/MWLayer/Security	BB_Internet_Protocol_Security	Low
BB-SC/MWLayer/Security	BB_Intrusion_Detection	Low
BB-SC/MWLayer/Security	BB_Secure_Onboard_Communication	High
BB-SC/MWLayer/Security	BB_Security_Event_Manager	High
BB-SC/MWLayer/Security	BB_Security_Transport_Layer	Low
BB-SC/MWLayer/Storage	BB_Vehicle_Data_Collector	High
BB-SC/MWLayer/Storage	BB_Vehicle_Data_Persistency	Medium
BB-SC/MWLayer/Storage	BB_Vehicle_Logging_and_Recording	High
BB-SC/MWLayer/Time	BB_Time_Service	High
BB-SC/MWLayer/Tools-and-Methods	BB_Key_Management_System	Medium
BB-SC/OSLayer/Time	BB_Automotive_Edge_Runtime	High
S-BB/AppLayer	BB_Standardization_of_Vehicle_API	
S-BB/AppLayer	BB_Standardized_Architectural_Patterns_for_Cross_Platform	
S-BB/AppLayer	BB_Standardized_Description_of_Data_from_Related_Domains	
S-BB/AppLayer	BB_Standardized_Procedure_and_Tooling_for_Combining_Data_from_Different_Domains	
S-BB/AppLayer	BB_Standardized_Procedure_and_Tooling_for_Modelling_Data	
S-BB/MWLayer	BB_SOA	High
S-BB/MWLayer	BB_Standardized_Data_Conversion_Tools_for_Info_Knowledge_Layers	
S-BB/MWLayer	BB_Standardized_Data_Description_for_Vehicle_Sensors_Attributes_Actuators	
S-BB/MWLayer	BB_Standardized_way_for_Reasoning_on_Data_Streams	
S-BB/MWLayer	BB_sSOA	Medium

b. Building Block template

 # (insert name of BB)

BB Tags(s)

<!-- Tag(s) define in which area(s) (cloud, in-vehicle) the BB is executed, and what type of BB it is (tool, process, microservice) -->

Functional Clusters

<!-- In which Functional Cluster the BB be located; if none of the existing fit new required -->

Layer

<!-- AppLayer, MWLayer, OSLayer, HWLayer -->

Known Implementation

ID (unique name)

Description

<!-- General Description of the BB -->

Rationale

<!-- Explanation why we need the BB; what problem want to be solved -->

Governance Applicable S-BB(s)

<!-- Reference to e.g. UN/EU CRA Cyber Resilience Act; UNECE 156 - Software update and software update management system

Reference to defined S-BB(s)

Reference to e.g. IS026262, AUTOSAR Spec. X -->

Compose BB(s)

<!-- Link to required BB(s)

E.g. BB-SC StateManagement

BB is a composition of other BBs -->

What is needed to Design and Implement

<!-- e.g. we expect to have a certain HW capability and or SW environment or Tool support, or a documentation, or an extra audit, or Test, or Compiler, or Prog. Language, … -->

What is needed to build and run

<!-- e.g. we expect to have a certain HW capability, or Runtime Environment, or Pre-configuration, or Code-signing, or Test, … -->

Non-Functional Requirements

<!-- With respect to Safety, Security, Realtime, … -->

Dependencies to other Clusters

<!-- Other clusters are needed. FC Security, FC Storage, …

e.g. If FC Security : Security BBs are needed but you can choose for example crypto BB-SC from company A or crypto BB-SC from company B; several compositions may work -->

Vehicle API Relevant

<!-- If "Yes exists" – where – e.g. COVESA VSS

If "No" – nothing more to do

If "Yes, proposal for additional Signals/Information – what should be made available, and where e.g. via (COVESA) VSS/VISS -->

Author/Company

Priority

<!-- High, Medium, Low -->

Related Project(s)

<!-- If Yes – e.g. The BB should be used/added in the Eclipse Blueprint A – for demo purposes, show added value,

If No – Project Proposal (e.g. WP4 in FEDERATE, or in the SDV EcoSystem Community Framework -->

Availability of Source Code

<!-- Yes / License (e.g. Yes/MIT)

No – Commercial Closed Source -->

Availability of API

<!-- Yes / License (e.g. Yes/Apache 2.0)

No - Commercial -->

Potential obstacles

6. References

[1] Open Source – Eclipse SDV Community Event, Presentation on March 20: "Cross Organization Activities: EU Updates FEDERATE & HAL4SDV, SDV Alliance" - *https://sdv.eclipse.org/sdv-community-day-graz-2024/*

[2] FEDERATE project repository: *https://github.com/CSA-FEDERATE/Proposed-BuildingBlocks*

[3] example GitHub review process: https://github.com/CSA-FEDERATE/Proposed-BuildingBlocks/pull/10

[4] use case scenario fleet data: https://github.com/CSA-FEDERATE/Proposed-BuildingBlocks/blob/main/unsorted_BB/Daniel/BB_Template_filled_FleetData_WP3.md

[5] use case scenario remote vehicle interaction: https://github.com/CSA-FEDERATE/Proposed-BuildingBlocks/blob/main/unsorted_BB/Daniel/BB_Template_filled_RemoteVehicleInteraction_WP3.md

[6] use case scenario universal service mesh: https://github.com/CSA-FEDERATE/Proposed-BuildingBlocks/blob/main/unsorted_BB/Daniel/BB_Template_filled_ServiceMesh_WP3.md

[7] Building Block Implementation Guideline: https://github.com/CSA-FEDERATE/Proposed-BuildingBlocks/blob/main/other/utils/BB_Implementation_guideline.md