A Model-driven development framework for highly Parallel and EneRgy-Efficient computation supporting multi-criteria optimisation

D1.6 Use case evaluation

Version 1.0

Documentation Information

Contract Number	871669
Project Website	www.ampere-euproject.eu
Contratual Deadline	30.06.2023
Dissemination Level	PU
Nature	R
Author	GTSI
Contributors	BOS, BSC
Reviewer	ETHZ
Keywords	Use Case, Railway domain, Automotive Domain

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 871669.

Change Log

Version	**Description Change**
V0.1	Draft of template D1.6
V0.2	GTSI draft version
V0.3	Integrated version
V0.4	Version ready for submission to consortium reviewer
V0.5	Reviewed version
V0.6	Reviewed version refinement
V1.0	Version ready for submission

Table of Contents

1 Executive Summary

Deliverable D1.6 was released as part of the activities of T1.5 of WP1 at M42 and evaluates the AMPERE ecosystem which was developed in WP2, WP3, WP4 and WP5 and integrated in WP6 with activities spanning from M28 to M42. Hence evaluation activities were also distributed in the work packages just mentioned and are reported in details the relevant final deliverables: D2.5 [9] , D3,4 [10], D4.4 [11], D5.4 [12].

The evaluation methodology applied to the AMPERE ecosystem in this document reflects the features of the two use cases, railway use case and automotive use case, which were not symmetric.

In fact, while the railway use case does not initially rely on a model-based system and its modelling was made from scratch, *a posteriori,* using a THALES tool, Capella, which does is mainly tailored to systems of systems and is neither specifically dedicated to software nor to low level design. Nonetheless, the application of the modelling techniques envisaged by AMPERE project has been extremely useful to the current ODAS development as will be described later.

This document outlines:

1. A description of use-cases
2. Evaluation of the AMPERE ecosystem application to railway use-case
3. Evaluation of the AMPERE ecosystem application to automotive use-case

2 Railway use-case

2.1 Florence tramway

In the Florence metropolitan area there are about 720 cars per 1.000 inhabitants (increasing trend) – far above the EU average- and within the city of Florence there are about 510 cars per 1.000 inhabitants.

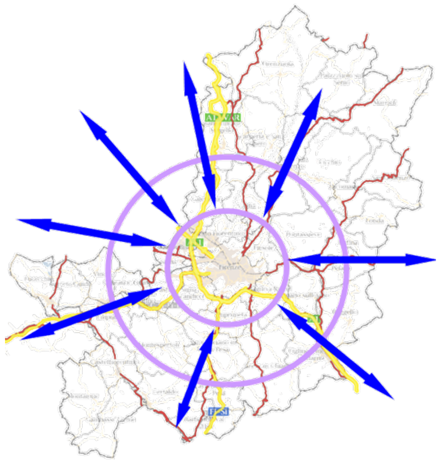


Figure 2-1: The city of Florence has a central position in the metropolitan area, thus playing a critical role in the mobility of the area

Due to geographical, infrastructural and socio-economic factors, the city of Florence plays a central role in the mobility scenario of the area, with about 80.000 and 70.000 cars moving respectively to and from Florence every day, accounting for severe traffic congestions and affecting quality of life.

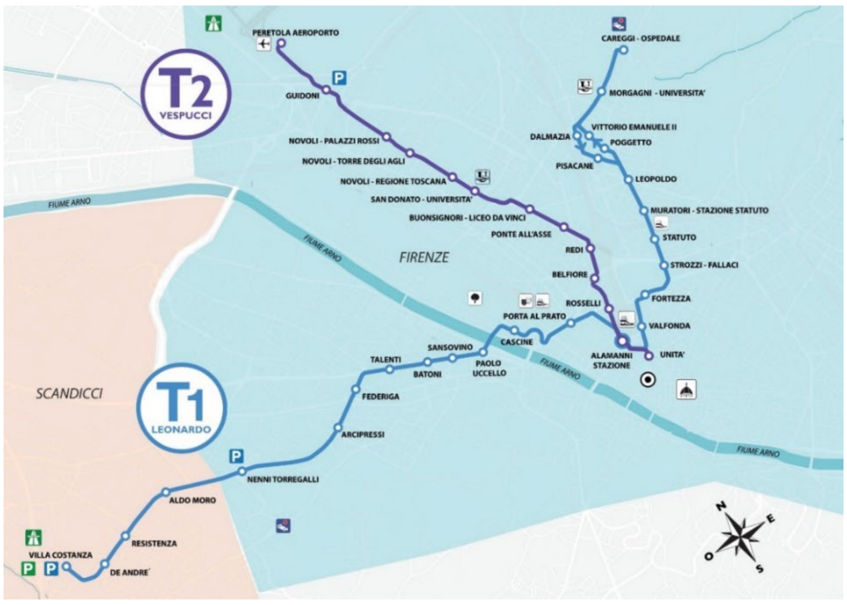


Figure 2-2: Active lines of the Tramway network in Florence

The tramway receives 30 million passengers each year (2022). The reasons that lead to the choice of the tramway are comfort, safety, economic convenience, and certainty of travel times. 99.6% of passengers are happy with it. In particular, a survey shows that the strengths of the Florence tramway are punctuality and regularity (with 99.7% approval), as well as the frequency and number of rides (98.9%).
The obstacle detection application (as well as other related technologies such as precise autonomous positioning) has the aim of improving the technology for newly conceived tramway solutions. The obstacle

detection and avoidance system (ODAS) will enhance the security and safety of people in the urban environment.
ODAS development is currently under finalization and under test on two streetcars in the Florence tramway system.
This new innovative approach will provide:

- A Multi-sensor Integrated solution
    - Cameras
    - Doppler RADAR
    - LiDAR
- Rail Track mathematical model
- Sensor Fusion Algorithm (SFA)
    - Kalman filter
    - Data fusion

The relevant on-board set of sensors and relevant control processing units which are installed in a set of streetcars must be strictly compliant with relevant regulation of Public Transportation and vehicles homologation because these vehicles are on-service.

2.1.1 **ODAS system**

The goal of the ODAS system is detecting objects in the Field of View of sensors, identifying potential obstacles thanks to data collected by sensors, AI algorithms and the implementation of the Sensor Fusion Algorithm (SFA), and rising warnings to the driver.

Hence, the missions of the ODAS system are:

- Fuse different sensors output information;
- Detect objects;
- Associate objects detect by different sensors
- Track objects;
- Warn the driver, in order to prevent the collision with the target, reducing his reaction time.

When an object is found in the sensors Field of View (FoV), ODAS starts tracking this object in order to determine if it is an obstacle or predict if it could move to the dangerous area. The general architecture block diagram is represented in the following figure:

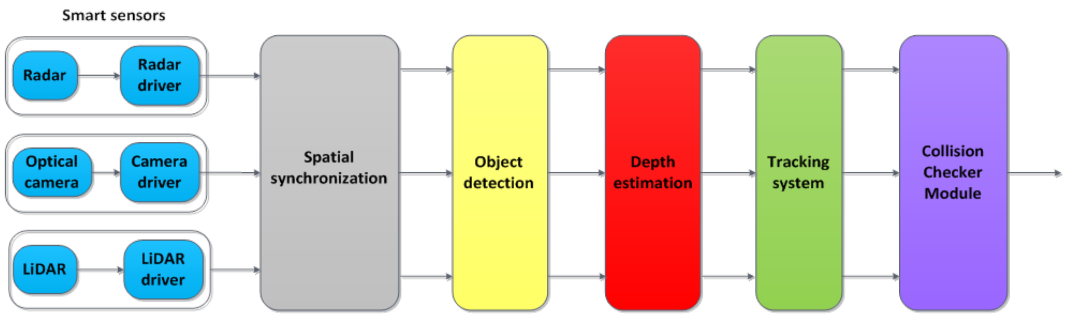


Figure 2-3 - ODAS architecture

2.1.2 **ODAS Installation**

The ODAS systems prototypes are installed for testing on two streetcars in service in Florence tramway system.

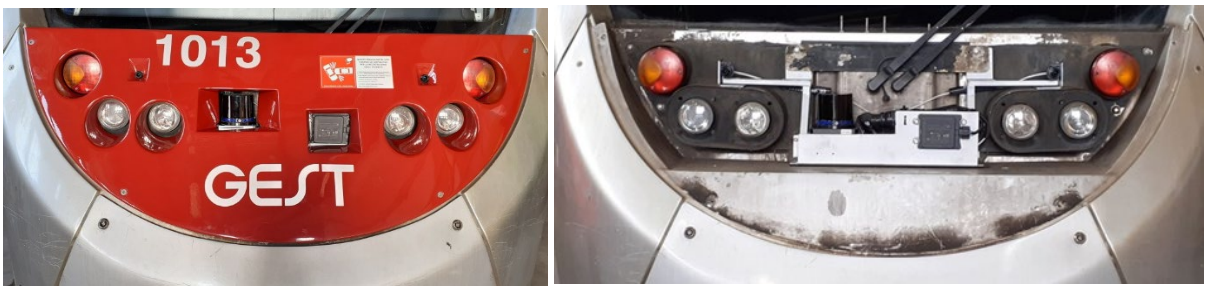


Figure 2-4: Sensors installation on board of streetcar 1013. The radar, LiDAR, and stereo camera are visible in the left figure where the tram nose skin is shown and in the right picture where it was removed and shows the relevant brackets.

2.1.2.1 Sensors

Sensors are installed in the front part of the vehicle nose, under the skin properly shaped to host them.

The RADAR system scan the area in front of the tram. Radars are widely used in automotive to implement various ADAS functions such as adaptive cruise control, traffic jam assist, or forward collision warning. As ODAS works in a similar environment and with similar constraints, automotive RADAR is currently used.

LiDAR scans the area in front of the tram and provides a cloud of points. In the current application a seven blades rotary LiDAR is used but solid-state device with a much larger number of blades will

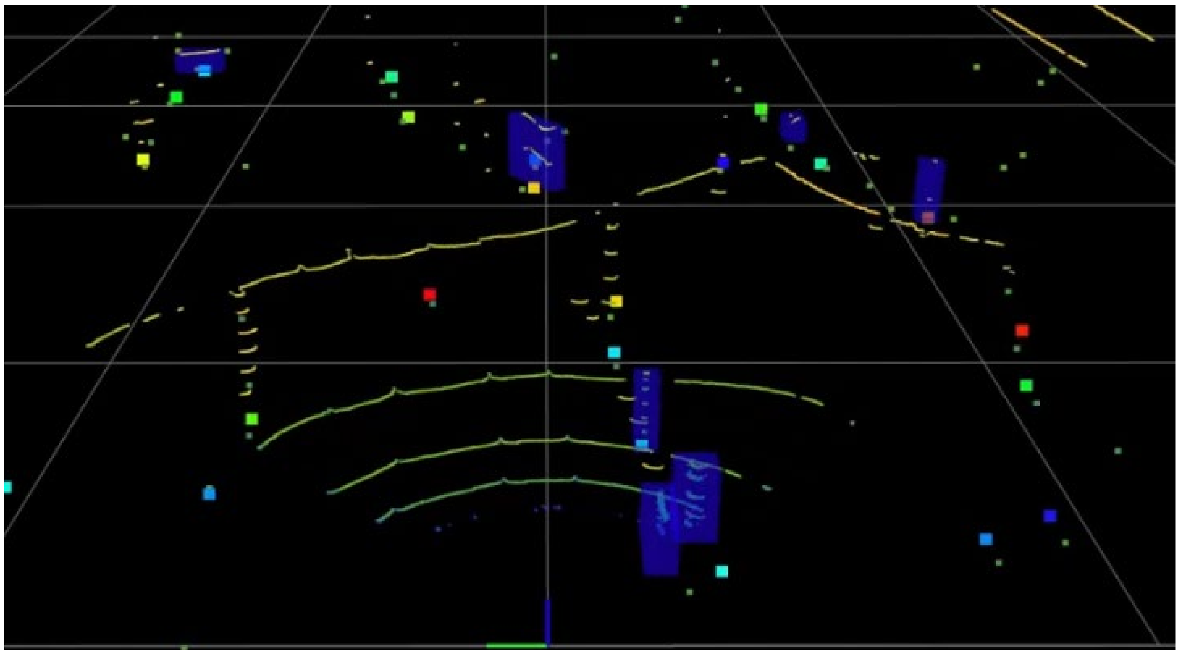


Figure 2-5: Cloud of point of a LiDAR sensor are clearly visible as lines "cut" by the LiDAR blades. Blue boxes are bounding boxes detected by LiDAR. Small green dots represent objects detected by RADAR. Coloured squares are objects tracked after sensor data fusion.

A stereo camera is a sensor that exploits the difference between two bidimensional views of the same scene to infer the third dimension (depth). The mathematics behind stereo vision will not be discussed here. What is important to note here is that the capability of a stereo camera to reconstruct a 3D space is directly related to a few parameters:

1. The orientation of each camera
2. The resolution of the two images
   The baseline between the cameras, that is, the distance between left and right cameras

Resolution of the left and right camera images has an important effect on the resolution of the reconstructed 3D volume, and consequently on the accuracy of localization in the 3D space. Without going into technical details, the 3D reconstruction function basically maps pairs of pixels (one from the left image, the other from the right image) to 3D points, by means of a triangulation method.

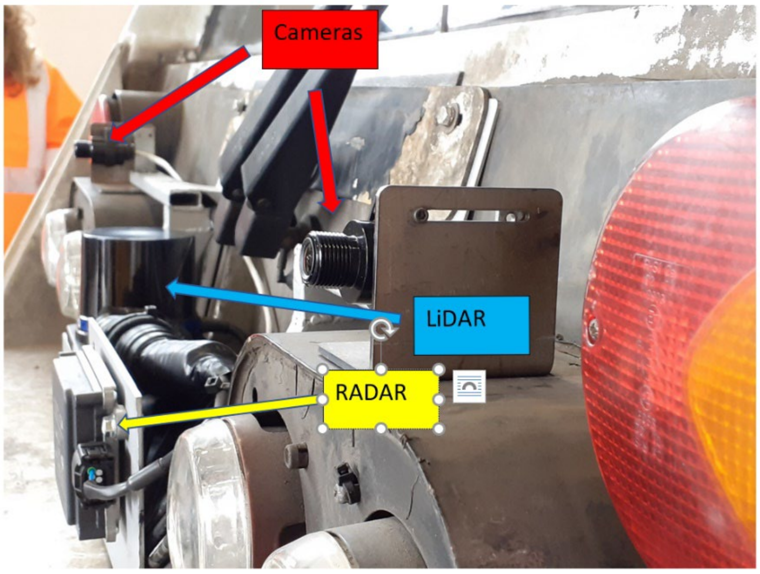
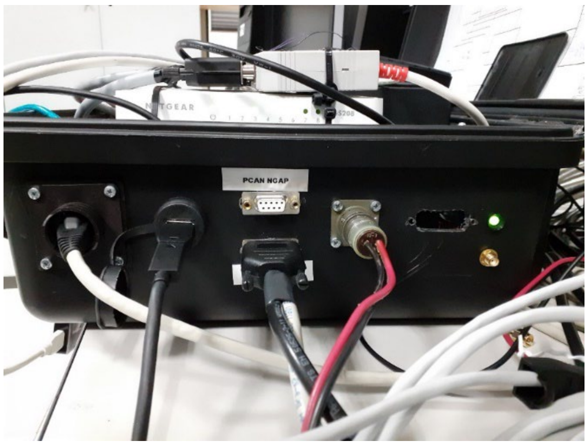


Figure 2-6: RADAR, LiDAR and cameras position in the front part of streetcar

Control system unit (NVIDIA XAVIER) and other device for concentrating sensors output are hosted in a suitcase fastened in the cabin near the driver seat.



2.1.2.2 Power management consideration

Energy and power management applied to use-cases is described in deliverable D3.3 [7] and in particular in chapter 3.2.

From the specific perspective of an ODAS system installed in streetcars, energy absorbed by the ODAS is not a practical issue. In fact, the system works on a tram vehicle (SIRIO manufactured by Hitachi Rail Italia) which is equipped by four electric motors whose absorption is 118 kW for each one . The rated power requested by NVIDIA board is 60 W that is negligible if compared to the power absorbed by engines and similar to power taken by the sensors. The same consideration is applicable to battery-equipped vehicles where onboard energy storage system stores more than 50 kWh and enables streetcars to travel across

sections of their route that lack catenaries [14]. Nevertheless, AMPERE ecosystem multi-criteria optimizations and analysis of use-cases is reported in D3.4 [10].
An important aspect to consider and which is associated to energy consumption was indirectly obtained thanks to modelling techniques derived by project AMPERE which led the current version of ODAS system to reduce the false alarm to an alarm rate of "one false alarm in three months of system exercise". That is one of the results which GTSI could demonstrate in the GTSI internal review (named Main Gate[1]), positively passed on mid-June and reported in [20]. In fact, the availability of accurate and precise warnings to the tram driver, has the effect of reducing the number of braking and acceleration events so that the energy general waste was heavily reduced. In fact, it must be considered that even if the regeneration system is able to collect large part of the energy paid during deceleration, a much larger quantity of energy is spent in the subsequent acceleration.

2.1.3 **Objectives**

The main objective of the Obstacle Detection System is to be the protection system (towards passengers, the Tram itself and the environment) during the automatic movement of the Tram. By autonomously detecting of the imponderable that can occur during an automatic movement, it will allow the Tram to autonomously adjust itself and carry out further automatic actions in order to avoid the effects of such possible conflicts/collisions.
It is therefore clear that, in the context of this technologically complex challenge, the fact that the Obstacle Detection System will be an integral part of the autonomous movements of the Tram means that it will have to achieve the same levels of integrity as the standard railway protection systems (SIL), in accuracy, reliability and availability.
Anyway, the Obstacle Detection Systems introduces benefits even if not applied in close convergence with autonomy. Indeed, it helps in any cases to reduce the average number of incidents due to incorrect behaviour of both the Driver and the Public. Possibly together with dissuasion system.

2.1.4 **State of the art**

The input information of the ODAS is sensors raw data like:

- RADAR messages. The radar sends messages including clusters information on: object ID, lateral and longitudinal coordinates of the target, relative velocity and Radar Cross Section (RCS).
- Cameras Video frames.
- LiDAR points clouds in its internal spherical coordinates reference system.

After the elaboration of sensors raw data in relevant drivers, the information is represented in Vehicle Body Frame common reference system implementing geometric transformations in the spatial synchronization sub-module.

Objects are detected as following:

- Detected RADAR objects: implementing clustering algorithm to aggregate radar clusters.
- Detected camera objects: a deep learning algorithm runs on the stream video, in order to identify the targets position, class and dimensions through Bounding Boxes (BBs).
- Detected LiDAR objects: implementing clustering algorithm to aggregate LiDAR points cloud.

[1] Main Gate is the last internal severe review step that new systems must undergo in GTSI before activities to release it as a product are allowed and consequently financed.

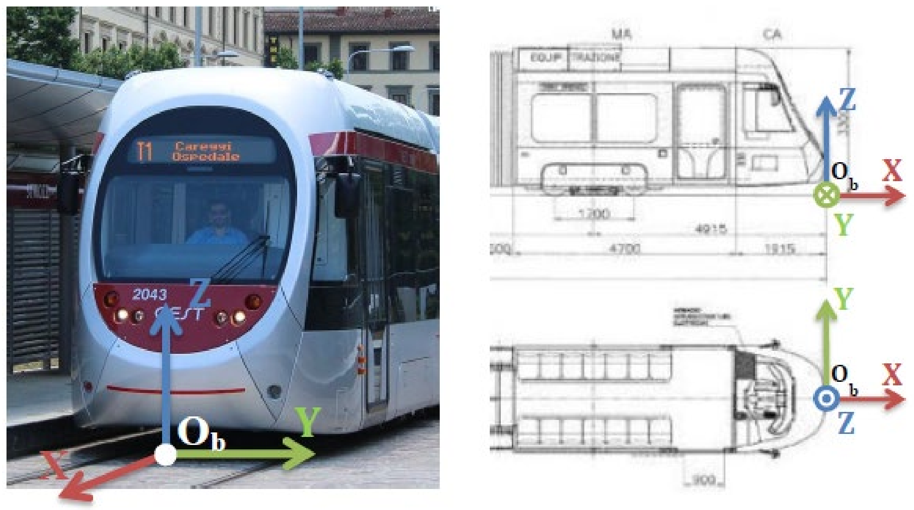


Figure 2-7 – Vehicle Body Frame (VBF)

The Tracking and Fusion Module includes the:

- Data association made by Global Nearest Neighbour algorithm;
- Tracking using Kalman filters. A limited number of associated objects is tracked in consecutive temporal instants. For each tracked object, a non-linear Kalman filter is instantiated.

Therefore, the Sensor Fusion Algorithm combines objects detected by RADAR, LiDAR and cameras. The Collision Checker Module is responsible to evaluate if a tracked object could be a "hazardous" one. When the tracked object is recognized as a "dangerous" target by the Collision Checker Module due to its proximity to the tram, an alarm is sent through the tower LED providing a warning alert to the driver, whom can activate the braking system.

The output information of the ODAS is:

- Time stamp;
- Objects class: tram, car, pedestrian, motorbike, bicycle;
- Objects dimensions
- Objects (x,y) position;
- Objects (vx,vy) speed;
- Uncertainty values of given data.

2.1.5 **The future**

The Autonomous Tram Roadmap, that has the final objective to reach automatic movements decided and made by the system, autonomously and safely, at the end is fully based on these three fundamental technological pillars:

- Autonomous Positioning System (NGAP)
- Obstacle detection System (ODAS) and Collision Avoidance System (CAS)
- Unified Tram to Ground Communication System (Unified T2G)

On top of that, a number of features based on these three pillars, will bring step by step (phase by phase) the tram to reach fully autonomous operations.
It must be noted that the AMPERE project does not rely on functional performances of use cases because its technology must be applicable in general and not to the specific functions. The description that has been provided aims at describing the technologic environment of the ODAS which represents the railway use case. Hence, no improvement of the functional performances of the ODAS was in the scope of the project.

The evaluation of the reduction of development and integration time (costs) of the ODAS system by the application of AMPERE technologies is valuable as a Prove of Concept because the regulations on certification of software associated to safety functions clearly requests that the related software tools have the same SIL certification. None of the software tools was developed in order to have these SIL levels because it was out of the scope of the project. Nevertheless, the code optimization implemented by AMPERE have been evaluated as a possible starting point for ODAS implementation which, in the version implemented to create a *product,* relies on human written code and does not use code generated from models.

2.1.6 Patents

For the moment no patents have been generated by GTSI during this Research and Technology action. In the following months GTSI reserves the right to analyze the feasibility of patenting some parts of the ODAS/CAS solution. Regarding the software modules developed by GTSI during the research and development activities, the collected sensors data and the achieved results, the Intellectual Property Rights (IPRs) are owned by GTS Italy.

2.2 Railway Use Cases modelling

2.2.1 **Overview**

GTSI (and formerly THALIT) contribution to the AMPERE project is based on a use-case taken from a real-world scenario.

The goal of this use-case is "stressing" the AMPERE model by injecting non-functional constraints coming from a real-world system. This real-world example consists of a system which detects objects that could cause a risk to the streetcar movement (Obstacle Detection and Avoidance System, ODAS) supporting the driver's actions.

A description updates of ODAS system has been given in previous paragraphs to refresh the one provided with D1.1 in the project very beginning time.

This chapter describes the system model and the relevant tools which were developed and involved.

The detailed description of the relevant toolchain is included in "D1.5 Meta model-driven abstraction and model-driven extensions and use case enhancements" [3].

2.2.1.1 Capella

Capella is an open-source tool for model-based systems engineering derived by a proprietary tool developed by Thales.

Capella implements the principles and modelling methodologies defined by Arcadia, the Thales standard systems engineering method based on the use of models. Initially, before releasing it as an opensource tool, THALES developed it as a proprietary tool called Melody Advance.

Capella is now included among the Eclipse Foundation tools. It is described in D6.4 [16] but plenty of documents are also publicly available at [4], [15].

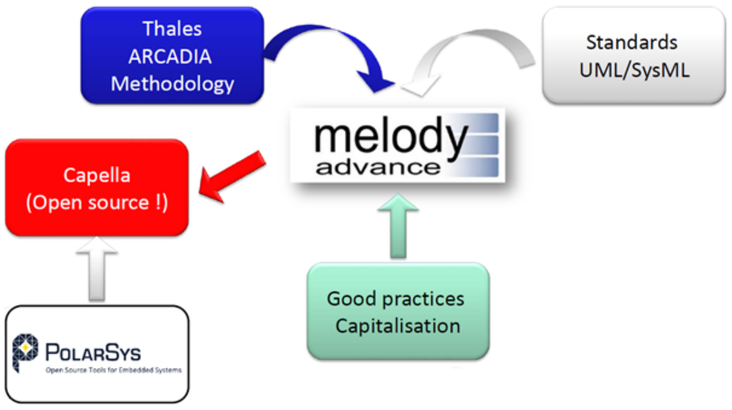

Thales ARCADIA Methodology
Standards UML/SysML
melody advance
Capella (Open source !)
Good practices Capitalisation
PolarSys
Open Source Tools for Embedded Systems

Figure 3-8 – Capella, Arcadia and Melody Advance

The Arcadia method enforces an approach structured on different engineering perspectives. These perspectives are clearly separated between:

- system contexts;
- need modelling;
    - operational need analysis;
    - system need analysis;

- solution modelling;
  - logical architectures;
  - physical architectures.

Documentation on Arcadia are available at [5], [17]and [18]

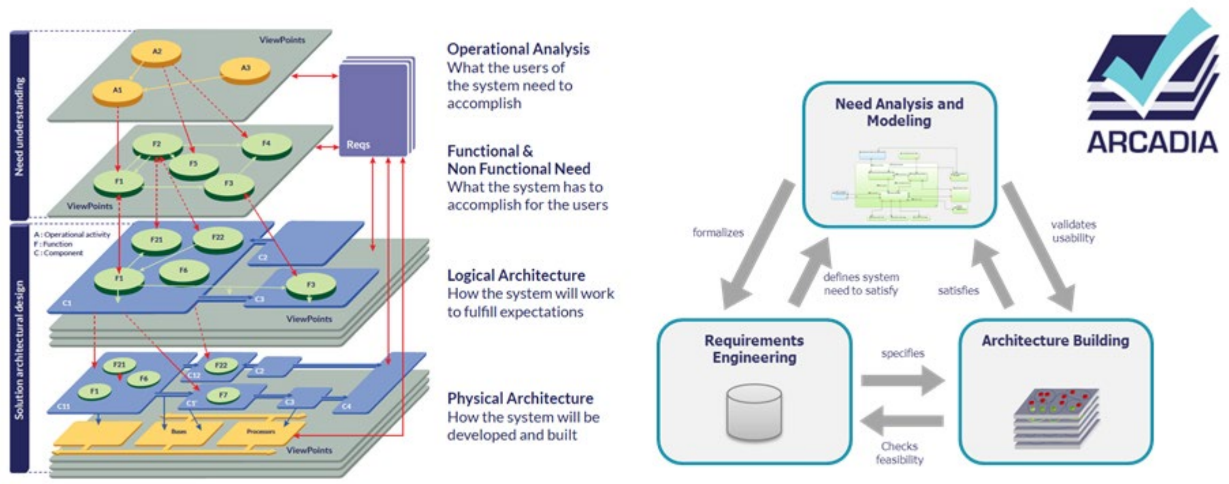

Need understanding
Solution architectural design
ViewPoints
Reqs
A : Operational activity
F : Function
C : Component
Operational Analysis
What the users of the system need to accomplish
Functional & Non Functional Need
What the system has to accomplish for the users
Logical Architecture
How the system will work to fulfill expectations
Physical Architecture
How the system will be developed and built
ARCADIA
Need Analysis and Modeling
formalizes
defines system need to satisfy
satisfies
validates usability
Requirements Engineering
specifies
Checks feasibility
Architecture Building

Figure 3-9 – Arcadia

In Capella model, real world non-functional constraints to be injected in AMPERE ecosystem have been represented by safety requirements. For an in-depth description of identified safety requirements see “D1.4 Analysis of functional safety aspects on multi-criteria optimization and final release of the test bench suite” [1].

2.2.1.2 Bridge, a tool from Capella to Amalthea

The Bridge is a software tool developed by THALES TRT to transform automatically Capella models to Amalthea models.

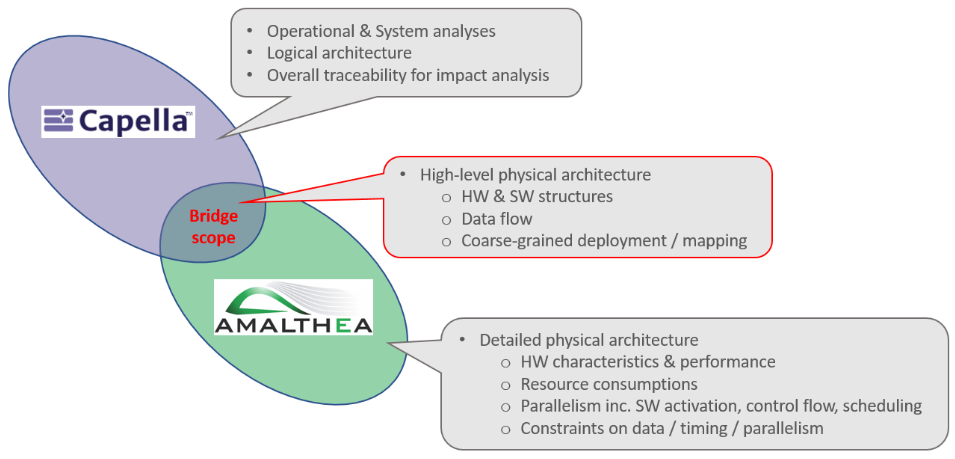

Operational & System analyses
Logical architecture
Overall traceability for impact analysis
Capella
Bridge scope
High-level physical architecture
HW & SW structures
Data flow
Coarse-grained deployment / mapping
AMALTHEA
Detailed physical architecture
HW characteristics & performance
Resource consumptions
Parallelism inc. SW activation, control flow, scheduling
Constraints on data / timing / parallelism

Figure 3-10 – From Capella to Amalthea through the Bridge

Capella and Amalthea provide system description at different levels of abstraction: higher level Capella, lower Amalthea. In fact, Capella is a MBSE tool that is to say a Model Based Systems Engineering tool developed to cover each engineering phase. It has no specific embedded tool to manage to create software code.

Hence a tool has been developed as an extension to translate a Capella model to an Amalthea model while keeping consistent that part of the model containing information shared by both tools.

More information about this tool can be found in “D6.4 First release of the AMPERE ecosystem” [2].

From the railway use-case perspective this was a huge step because could link Capella models to Amalthea models and, more, to the dedicated tools of the AMPERE toolchain.

2.2.1.3 APP4MC

APP4MC is an open-source platform for engineering embedded multi- and many-core software systems based on the Amalthea modelling language. APP4MC is a project of the Eclipse Foundation under the Eclipse Public License (EPL) v2.0 [19].

In AMPERE, APP4MC is used to model the detailed software and hardware structures as well as the behaviour of the system.

The Amalthea platform allows users to distribute data and tasks to the target hardware platforms, with the focus on optimization of timing and scheduling. It addresses the need for new tools and techniques to make effective use of the level of parallelism in this environment.

Real world non-functional constraints injected in AMPERE ecosystem at the Capella level (as written above, in the AMPERE scope of work these are represented by safety related requirements) have been transferred to the Amalthea model using the Bridge tool. The Amalthea model has then been manually finalised by engineers to add more details that were required at this level of abstraction.

2.2.1.4 ODAS Modelling Principles

The ODAS model was initially developed using Capella and then translated to AMALTHEA language with the Bridge. It finally consists of seven Amalthea tasks representing main computational units in the model. Each Amalthea task is then split into Amalthea runnable implementing functions modelling basic operations:

- Task: LiDAR processing
    - Runnable 1: LiDAR messages management
    - Runnable 2: LiDAR spatial synchronization
    - Runnable 3: LiDAR object detection
- Task: RADAR Processing
    - Runnable 1: RADAR messages management
    - Runnable 2: RADAR spatial synchronization
    - Runnable 3: RADAR object detection
- Task: CAMERA Processing
    - Runnable 1: CAMERA raw data management
- Task: Offloaded CAMERA Processing
    - Runnable 1: CAMERA object detection
    - Runnable 2: Homography transformation
- Task: Sensor Fusion
    - Runnable 1: Time synchronization
    - Runnable 2: Data association
    - Runnable 3: Track management
- Task: KFs (Kalman Filters)

    - Runnable 1: KF management
    - Runnable 2: KF 0
    - Runnable 3: KF 1
    - Runnable 4: KF 2
- Task: CCM
    - Runnable 1: Collision checking

According to the Amalthea modelling language, runnables are executed in the same order they appear in tasks, the first one being triggered by some sort of input received (a periodic stimulus or a label/channel being written). Runnables execute instructions similar to those that can be found in modern programming languages.

The following figure represents the whole Capella model and shows the modules relating to runnables just described.

Data from sensors are collected independently one from each other, processed for spatial and time synchronization, detection of obstacles, data association do establish when different detection from different sensors refer to the same objects, tracking of objects and collision check in order to rise alarms to driver. A Kalman filters tracks each detected object up to the maximum number (now heuristically set to 60). In case the number of detected objects exceeds the maximum number, the farthest one is not tracked.

2.2.1.4.1 ODAS Modelling results

The ODAS system was initially developed without using a modelling approach. Nevertheless, the participation to AMPERE project led THALES/GTSI to develop a model that started with an accurate system analysis which allowed the improvement of design and had benefits on the real implementation of ODAS

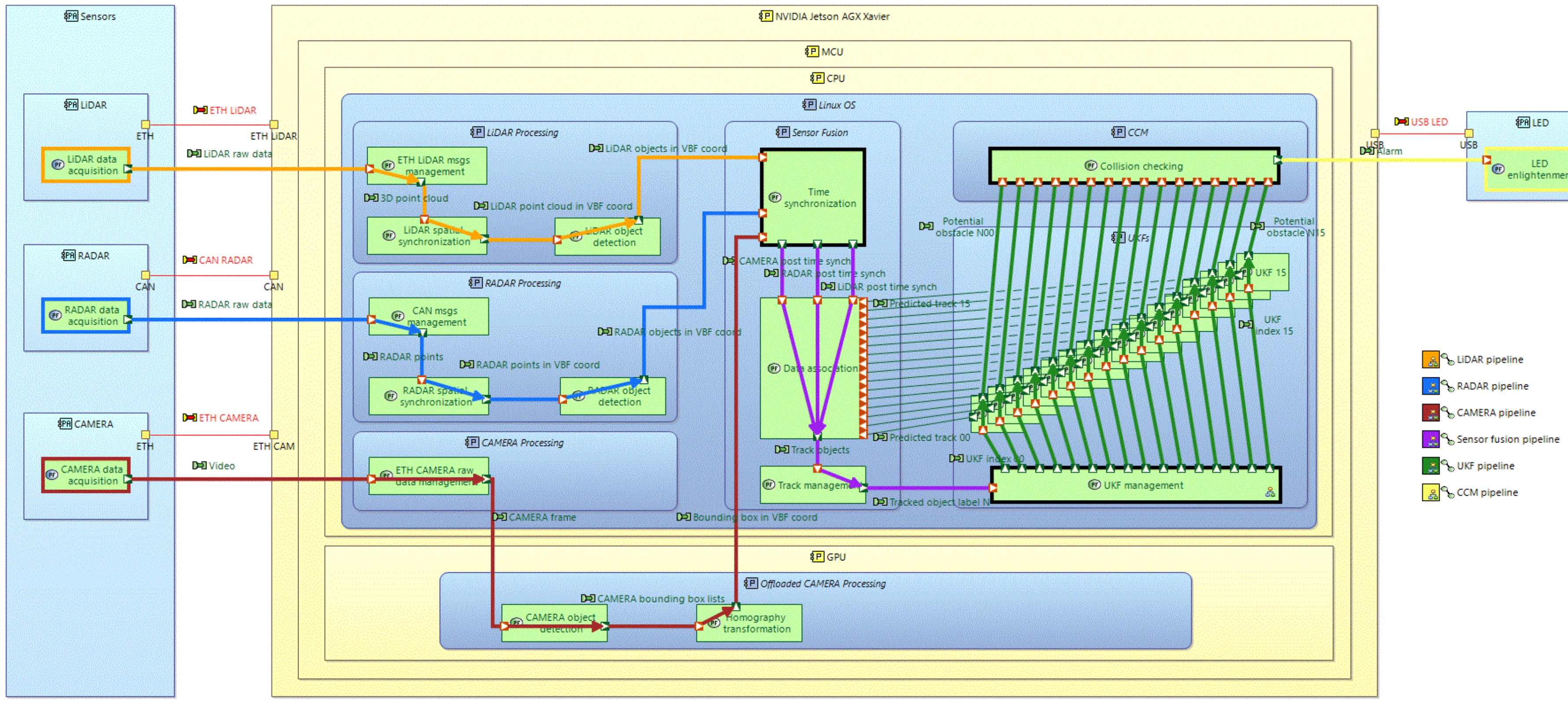


Figure 3-11 – ODAS model in Capella

2.2.1.5 Timing parameters

Execution time is represented in Amalthea through a model item named “Ticks”. When such an instruction is found, the runnable starts sleeping for the time specified (in CPU ticks).

Runnables are used to reproduce the classical algorithm execution. Basically, a runnable:

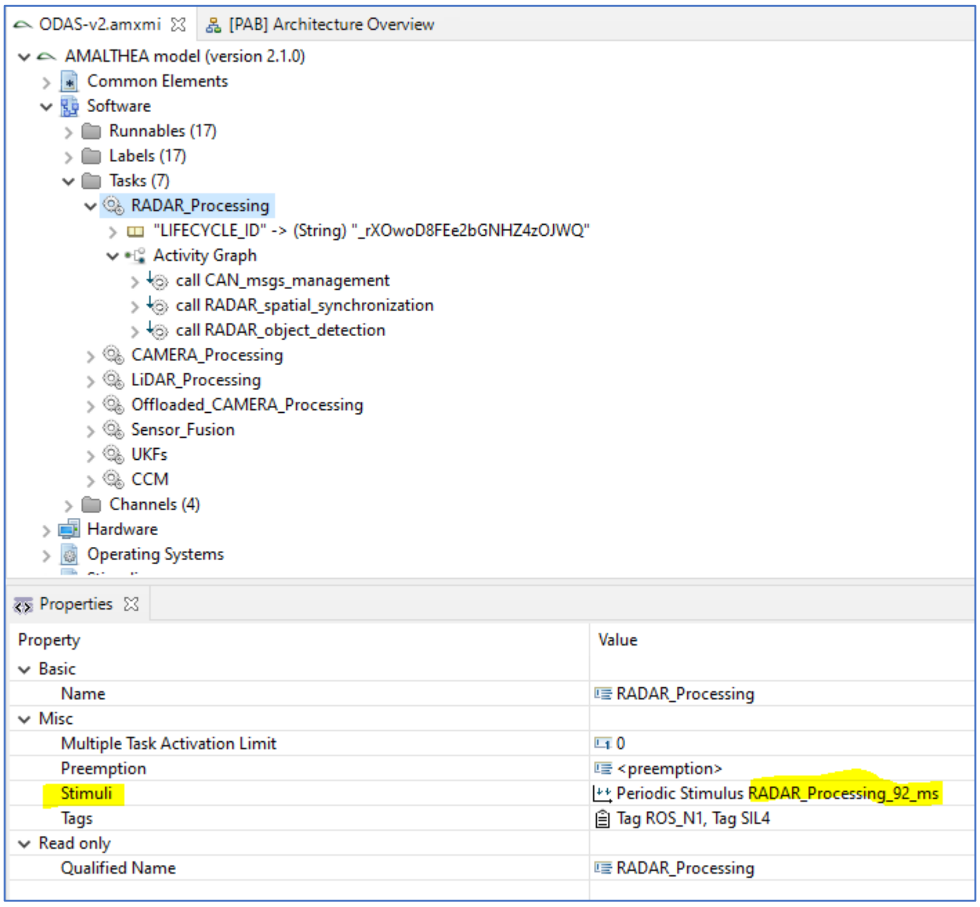
- starts by reading a value on its inputs;
- performs some operations involving input data (in our use case no real operations are performed so basically a “Ticks” function is called for a given number of CPU ticks);
- performs a write on an output (a label or a channel);
- exits.

Another time parameter used in the Amalthea model is the recurrence interval of a Periodic Stimuli. Periodic Stimuli are used to simulate raw inputs from sensors (LiDAR, RADAR and CAMERA). Every time a Periodic Stimuli is triggered a task performing messages management is activated together with its corresponding runnables.

The Periodic Stimuli scheduling is as follows:

Every 100 ms: LiDAR raw data	→	activation of task “LiDAR processing”
Every 92 ms: RADAR raw data	→	activation of task “RADAR processing”
Every 40 ms: Video raw data	→	activation of task “CAMERA processing”

A Periodic Stimuli is associated to the Task it triggers as shown in Figure 3-5.


ODAS-v2.amxmi [PAB] Architecture Overview

```
AMALTHEA model (version 2.1.0)
  Common Elements
  Software
    Runnables (17)
    Labels (17)
    Tasks (7)
      RADAR_Processing
        "LIFECYCLE_ID" -> (String) "_rXOwoD8FEe2bGNHZ4zOJWQ"
        Activity Graph
          call CAN_msgs_management
          call RADAR_spatial_synchronization
          call RADAR_object_detection
      CAMERA_Processing
      LiDAR_Processing
      Offloaded_CAMERA_Processing
      Sensor_Fusion
      UKFs
      CCM
    Channels (4)
  Hardware
  Operating Systems
```

Properties

Property	Value
Basic	
Name	RADAR_Processing
Misc	
Multiple Task Activation Limit	0
Preemption	<preemption>
Stimuli	Periodic Stimulus RADAR_Processing_92_ms
Tags	Tag ROS_N1, Tag SIL4
Read only	
Qualified Name	RADAR_Processing

Figure 3-12: Stimuli to Task association

2.2.1.6 Safety modelling

The main goal of the railway use case is giving AMPERE ecosystem the possibility to be tested on real world non-functional requirements.

As most railway projects/products, ODAS also has strong safety requirements to be implemented and a Safety Integrity Level (SIL) to be assured.

Safety in railway domain is regulated through international standards such as EN50126, EN50128 and EN50129.

EN50126 describes a process framework to consistently manage safety concerns using methods and tools that do not depend on the system/subsystem technology. Safety aspects are identified and managed through risk assessment and hazard control. EN50126 standard also provides methods to derive the Safety Integrity Levels (SIL) for safety related functions. Besides risks assessment and hazard control, the standard also sets constraint on company organization. One of the goals of these organizational requirements is enforcing independence among roles. Independence is needed to reduce the probability of systematic failures, i.e. failures depending on poor design or poor Verification & Validation activities.

EN50128 provides a process (with organisational structure and division of responsibility) and methodologies for development, deployment, verification, validation and maintenance of any safety-related software for the railway industries having been allocated a given software integrity level. This standard has been built around the key concept of Safety Integrity Level (SIL) and gives guidance on how software code must be designed, implemented, validated and maintained to be compliant to a given SIL level.

EN50129 is focused on the evidence to be presented for safety-related systems acceptance. The concept of a Safety Case is used as a framework for a systematic documented approach to quality management, safety management, functional and technical safety, safety acceptance and approval. This standard also contains details on SIL use in safety-related systems for railway applications.

ODAS safety requirements have been explored in "D1.4 Analysis of functional safety aspects on multi-criteria optimization and final release of the test bench suite" [1].

As Capella and Amalthea works at two different levels of abstraction, a different description of these non-functional requirements has been used.

In Capella, a tool that works at a higher abstraction level than Amalthea, safety requirements are expressed in a qualitative way. A property named "Safety-critical" has been defined at component level, this property is available to the engineer that wants to label a component to specify it implements safety functions.

When the Bridge tool is applied to the Capella model, all components being labelled "Safety-critical" are transformed into Amalthea Tasks labelled with a tag "SILx". This tag must be replaced by the engineer with the proper SIL required for the implementation of that safety function (SIL0, SIL1, SIL2, SIL3 or SIL4, in ascending order of safety required).

Result concerning Safety aspects and resilience were analysed and evaluated in deliverable D3.3 [7] (chapter 2) and in deliverable D4.2 [8] (chapter 6).

2.2.1.7 DSML extension

Some extension to the DSML tools used to model the railway use case have been required and they are summarised in the following subsections.

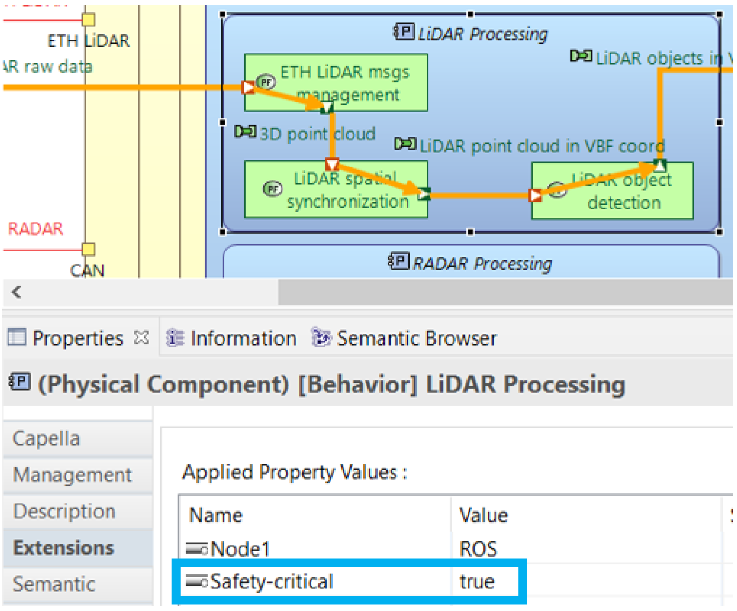
More information on this topic can be found in deliverable “D1.5 Meta model-driven abstraction and model-driven extensions and use case enhancements” [3].

2.2.1.7.1 Safety-critical tag (Capella)

Capella does not have a standard way to identify model components as safety related. For this reason, a Boolean property has been defined named “Safety-critical”.

Property “Safety-critical” can be attached to any model component to keep track of modules that will (or could) be used to implement safety functions, as shown in Figure 3-6.

More strategy to implementing resilient solutions were applied to Kalman filter algorithm and evaluated in chapter 3 of deliverable D3.2 [6].



(Physical Component) [Behavior] LiDAR Processing

Applied Property Values :

Name	Value
Node1	ROS
Safety-critical	true

Figure 3-13: Capella “Safety-critical” property

Using this property, safety related modules can be identified at the earliest phases of the project. Moreover, safety information can be recorded in the model to avoid missing important requirements and propagated to other tools (for instance Amalthea) where safety related analysis can be continued.

2.2.1.7.2 Model transformation (Bridge)

The Bridge software tool has also the function of transforming the “Safety-critical” property used in Capella in “SILx” tags applied to Amalthea runnable so that safety requirements information can be propagated to lower level (and more specialized) description of the system.

“SILx” is a general notation that is used as a place holder: the designer is in charge of allocating the right SIL to each runnable where the “SILx” tag appears, in the range SIL0 (no particular safety needs) to SIL4 (maximum safety integrity level).

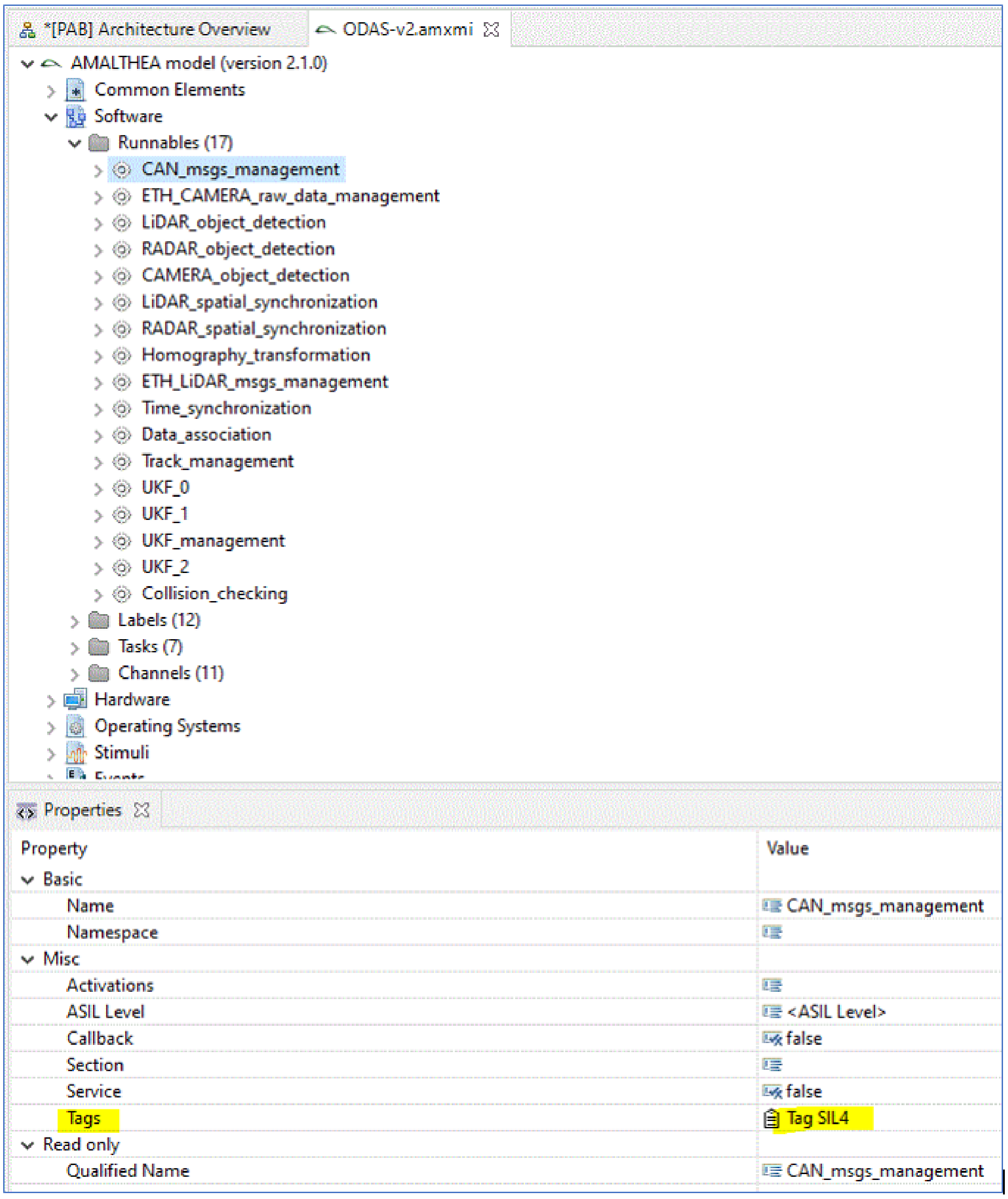
2.2.1.7.3 SILx tag (Amalthea)

Amalthea also does not provide the designer with any way of showing a safety requirement, for this reason a new tag has been defined.

The "SILx" tag is a place holder the Bridge appends to Amalthea runnable involved in implementation of Capella component that had been previously labelled (in Capella) using the property "Safety-critical".

It acts as a reminder to the designer that a given task implements a safety function and that it must be allocated the proper SIL level.

Figure 3-7 shows a "SILx" tag associated to an Amalthea runnable.


*[PAB] Architecture Overview | ODAS-v2.amxmi

- AMALTHEA model (version 2.1.0)
  - Common Elements
  - Software
    - Runnables (17)
      - CAN_msgs_management
      - ETH_CAMERA_raw_data_management
      - LiDAR_object_detection
      - RADAR_object_detection
      - CAMERA_object_detection
      - LiDAR_spatial_synchronization
      - RADAR_spatial_synchronization
      - Homography_transformation
      - ETH_LiDAR_msgs_management
      - Time_synchronization
      - Data_association
      - Track_management
      - UKF_0
      - UKF_1
      - UKF_management
      - UKF_2
      - Collision_checking
    - Labels (12)
    - Tasks (7)
    - Channels (11)
  - Hardware
  - Operating Systems
  - Stimuli

Properties

Property	Value
Basic	
Name	CAN_msgs_management
Namespace	
Misc	
Activations	
ASIL Level	<ASIL Level>
Callback	false
Section	
Service	false
Tags	Tag SIL4
Read only	
Qualified Name	CAN_msgs_management

Figure 3-14: Amalthea "SILx" tag

The designer must specialise the "SILx" tag to capture particular safety requirements in each runnable labelled with this tag.

The information contained in the SIL tag can also be exploited but subsequent tools in the AMPERE tool chain to implement required safety management techniques such as hardware redundancy, software coding rules, FPGA offloading, etc.

2.2.1.8 Safety regulation extension

No extensions to safety standards have been deemed necessary to address safety concerns in AMPERE ecosystem.

2.2.1.9 Kernel implementation

The ODAS software system is made of several modules and one of the most important one is the estimation and prediction of the status associated to the objects' positions. It is implemented by Kalman filters and the software modules that were chosen to provide the kernel resembling a functionality of the railway use-case.

A further development which was developed as part of AMPERE project as part of work package 3 is the Proactive orchestration.

It aims at building fault tolerance system from the application layer. The programmer is responsible for deciding which aspects of the algorithm need to be protected, making a distinction between programmer responsibility and automatic fault tolerance. It allows the programmer to implement a code observer of some state variables of the algorithm by focusing on early detection of symptoms that may lead to system faults. Hence it can monitor the behavior of a specific software component without interfering to the possible extent to the observed object. Observer monitor whether the observed variables keep satisfying the predicate which indicates a correct behaviour.

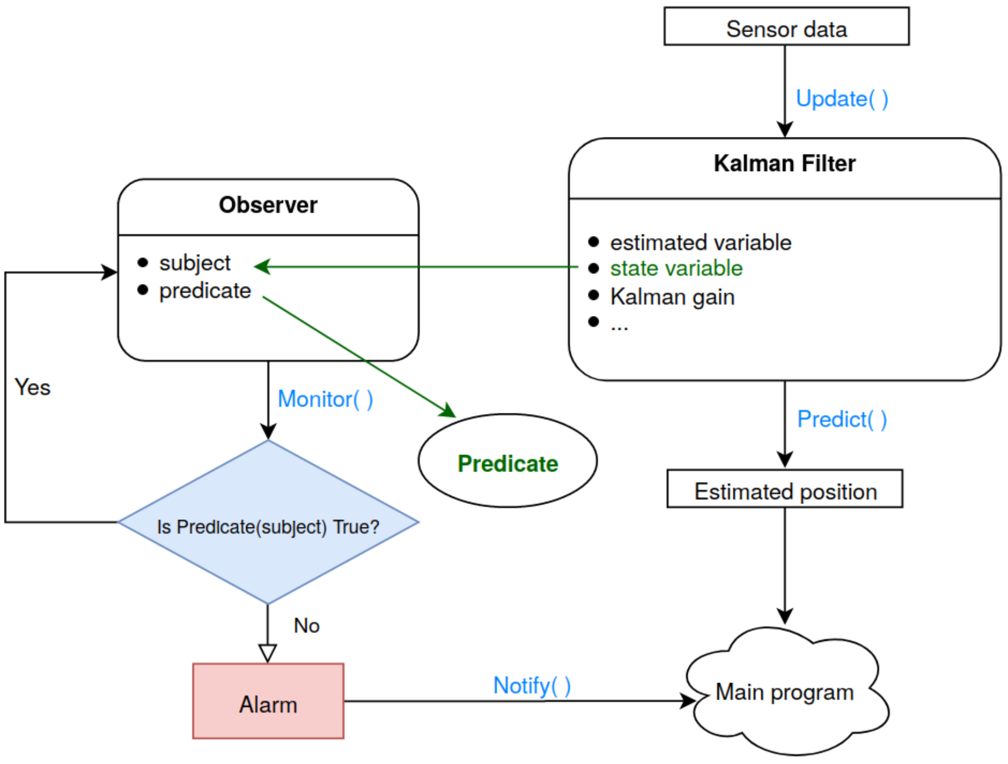


Figure 3-15: Observer

This technique is very interesting and will be applied to the ODAS product.

Finally, it is interesting for the AMPERE project reporting that a paper based on this study received the ***The BEST Student Paper Award by the*** **MECO'2023 Rewards Committee**[2]**.**

[2] The Award was communicated on June 15th, 2023 to Alessio Medaglioni of UNISI which collaborated to the project: "In front of the MECO&CPSIoT'23 Conference Organizers, it is our great pleasure to inform you that the **MECO'2023 Rewards Committee,** consisting of dr **Vicent J. Mooney** (Georgia Tech University, USA), dr **Rajeev Kanth** (Savonia University of Applied Sciences, Finland) and dr **Gehrard Rath** (University of Leoben, Austria) has, on it's session of 8.6.2023, after reviewing 128 Conference papers, made the decision to award you: ***The BEST Student Paper Award "***.

GTSI and BSC contributed to the paper.

3 Evaluation of the AMPERE ecosystem for the Railway Use-Case

3.1 General evaluation

The AMPERE ecosystem evaluation has been performed by moving the ODAS use case through the AMPERE tool chain (see Figure 4-1), from the Capella model generation to the executable on the Xavier board.

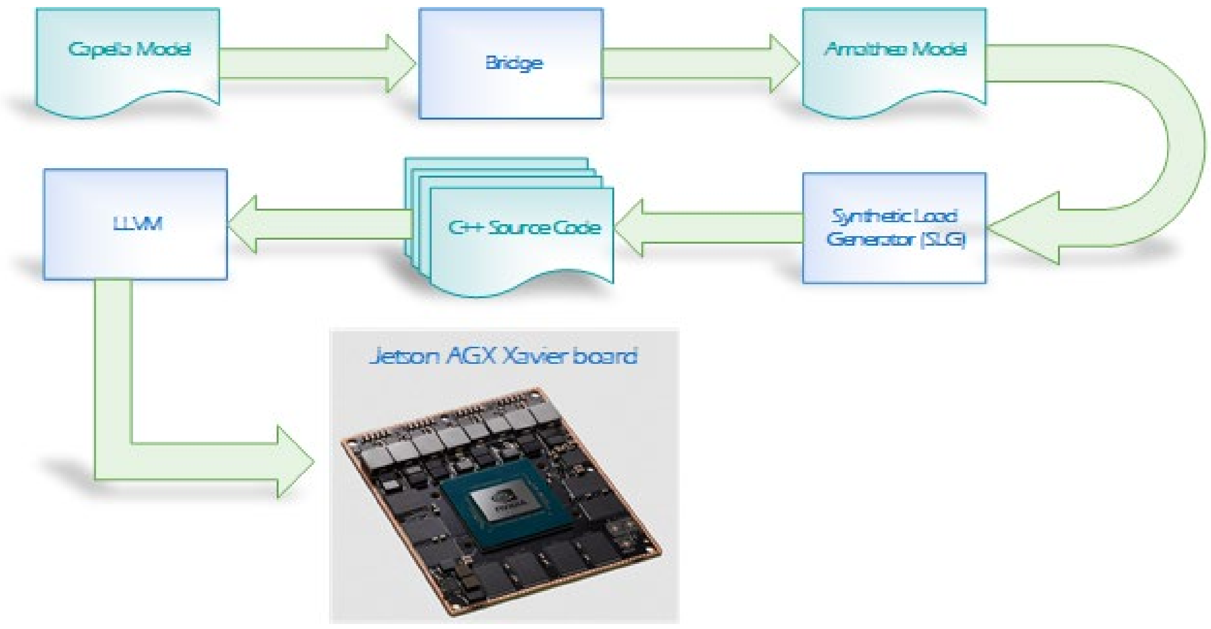


Figure 4-1. AMPERE toolchain for ODAS use case evaluation

Every step "n" in this chain has been fed with output from step "n-1" and used to generate output for step "n+1". This way every tool has been used as a validator for tool n-1 and has been validated by tool n+1.

The first input (n=1) comes from the Capella model and the last output goes to the Xavier board that validates it through execution.

To validate the Capella use case an Amalthea model has been generated through the Bridge. The output of the Bridge has been fed to the BSC version of the SLG tool that generated the software code. This code has subsequently been built by the BSC version of LLVM and, finally, executed on the Xavier board selected as the ODAS system hardware platform.

As this test case has been executed without any error, or other obstacles, the AMPERE ecosystem evaluation can be considered being passed successfully.

The final evaluation that THALIT/GTSI of the application of AMPERE ecosystem to the railway use-case is extremely positive because showed the advantages of the application of modelling techniques even to project which did not originally used it. Besides the optimization techniques are also very interesting and can be useful to the industrial projects where the optimization of hardware resources has a direct link with the kind of equipment necessary to processing and finally to their cost. Nonetheless, it must be said that the application of this very effective optimizations techniques is not immediately possible in the railway domain because all the railway products must be certified (EN50126, EN128, EN129). This aspect has a heavy impact on the working environment and the relevant software "usable" tools because EN50128 regulations impose the use of software tools developed and certified at the same SIL level.

3.2 Requirements evaluation

Railway requirements were listed in D1.1 v2.3.

3.2.1 Timing

- **[SYS-ODAS-REQ-101]:** The AMPERE ecosystem shall guarantee the completion of relevant processing from sensors data acquisition to issuing output data in less than 2.5 s.
  Test method Expected: Test
  Evaluation: Tested on NVIDIA Xavier board that relevant processing time is 100ms for each sensors' new data input. Cumulated time with statistical verification does not exceeds 800ms.

- **[SYS-ODAS-REQ-102]:** The AMPERE ecosystem shall not reach the deadlock status, in case of concurrent processes.
  Test method Expected: Test
  Evaluation: Requirement met by design and tested by Observer. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-103]:** The AMPERE ecosystem scheduler shall prevent a process resource starvation, in case of concurrent processes.
  Test method Expected: Test
  Evaluation: Requirement met by design and tested by Observer. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22]. Observer tool has also been developed to keep relevant variables under control.

- **[SYS-ODAS-REQ-104]:** The run-time shall support proper shutdown of replicated applications in case of failure.
  Test method Expected: Demonstration
  Evaluation: Requirement met by the application of a 2oo2 architecture and by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-105]:** The run-time shall support proper task application recovery after failure fix.
  Test method Expected: Demonstration
  Evaluation: Verified by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

3.2.2 HW/SW Platform

- **[SYS-ODAS-REQ-106]:** The HW/SW platform shall support ROS framework to transfer ROS messages between components.
  Test method Expected: Inspection
  Evaluation: Requirement met by design and code inspection. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-107]:** The AMPERE ecosystem shall guarantee the parallel execution of N Unscented Kalman Filters instances.
  Test method Expected: Demonstration
  Evaluation: Requirement met by design. N was set to 60. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22]. Observer tool has also been developed to keep relevant variables under control.

- **[SYS-ODAS-REQ-108]:** The run-time shall allow the contemporary execution of safe and non-safe applications over the same OS/safety layer software architecture.
  Test method Expected: Demonstration
  Evaluation: Requirement met by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-109]:** The run-time shall guarantee application replica determinism through synchronization mechanisms (SW-lockstep or similar) and message/data exchange methods.
  Test method Expected: Test
  Evaluation: Synchronization mechanisms is met by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-110]:** The Sensor Fusion Algorithm shall be executed in the CPU of the AGX Xavier board.
  Test method Expected: Demonstration
  Evaluation: Requirement implemented in model and met by design. For details, see also D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-111]:** The object detection SW module on camera frames shall run in the GPU of the AGX Xavier board.
  Test method Expected: Demonstration
  Evaluation: Requirement implemented in model and met by design. For details, see also D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-112]:** The AMPERE ecosystem shall be able to dynamically allocate applications tasks to CPU cores according to the SCHED_FIFO or SCHED_DEADLINE of the Linux kernel.
  Test method Expected: Test
  Evaluation: Verified by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-113]:** The AMPERE ecosystem shall support POSIX queue mechanisms for voted and non-voted message passing paradigms.
  Test method Expected: Test
  Evaluation: Verified by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-114]:** The HW/SW platform shall include the following libraries: ROS, OpenCV, GStreamer, Qt, Python3, PyTorch/Tensorflow, CUDA, DeepStream, Spdlog.
  Test method Expected: Inspection
  Evaluation: Verified by design and code inspection. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

3.2.3 Energy

- **[SYS-ODAS-REQ-115]:** The HW/SW platform power consumption shall be lesser than 200 W.
  Test method Expected: Analysis
  Evaluation: XAVIER board power consumption is 60W while the sensors power consumption is

lower than 80 W.

- **[SYS-ODAS-REQ-116]:** The HW/SW platform should be able to perform dynamic power management both in the active and passive states of the system.
  Test method Expected: Test
  Evaluation: Requirement met by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

3.2.4 Safety

- **[SYS-ODAS-REQ-117]:** The software ecosystem shall guarantee the support of fault tolerant architectures like 2oo2 (2-out-of-2) or 2oo3.
  Test method Expected: Test
  Evaluation: Requirement met by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-118]:** The AMPERE ecosystem should fulfil the requirements of the safety standard IEC 61508.
  Test method Expected: Inspection
  Evaluation: IEC 61508 is the standard governing functional safety of programmable electronic systems along the whole system life-cycle. In this sense the requirement is not completely applicable to the project’s objectives.

- **[SYS-ODAS-REQ-119]:** The AMPERE ecosystem should guarantee the contemporary execution of Safe and non-Safe applications on the same HW/SW platform.
  Test method Expected: Test
  Evaluation: Requirement met by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-120]:** The AMPERE ecosystem should implement memory protection mechanisms in order to avoid any application could-intentionally or unintentionally-corrupt the code, data or stack of another application.
  Test method Expected: Test
  Evaluation: Requirement met by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22]. Observer tool has also been developed to keep relevant variables under control.

- **[SYS-ODAS-REQ-121]:** The AMPERE software architecture should implement a fault supervisor to assure that when an application faults (for example, due to a stack overflow), the kernel should provide some mechanism whereby notification can be sent to the supervisor.
  Test method Expected: Inspection
  Evaluation: Requirement met by the application of a 2oo2 architecture and by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-122]:** The AMPERE ecosystem should provide the capability to configure mandatory access control (vs discretionary access control) of critical system components and applications. After selecting the configuration option, the system should guarantee do not change it at runtime.
  Test method Expected: Inspection
  Evaluation: Critical components are tagged and relevant implementation must be done accordingly. Observer tool has also been developed to keep relevant variables under control.

- **[SYS-ODAS-REQ-123]:** In the AMPERE safety-critical system, dynamic process creation should not be tolerated and the RTOS should be configurable such that this capability can be removed from the system.

Test method Expected: Inspection
Evaluation: Requirement met by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-124]:** The AMPERE ecosystem should provide a process-level scheduling function to guarantee resource availability to applications with different levels of integrity in the time domain.
  Test method Expected: Inspection
  Evaluation: Requirement met by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

- **[SYS-ODAS-REQ-125]:** In order to allow computation of the overhead of system calls that an application will execute while doing its work, the AMPERE ecosystem should provide bounded execution times for all such calls.
  Test method Expected: Inspection
  Evaluation: Overhead is monitored. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22].

3.2.5 Integrity

- **[SYS-ODAS-REQ-126]:** The AMPERE ecosystem shall be robust to maintain the accuracy of data.
  Test method Expected: Demonstration
  Evaluation: Requirement met by design. For details, see D4.3 [21], D2.4 [9], D3.3 [7], D5.3 [22]. Observer tool has also been developed to keep relevant variables under control.

- **[SYS-ODAS-REQ-127]:** The AMPERE ecosystem shall prevent the input of fake data.
  Test method Expected: Inspection
  Evaluation: This requirement is out of the scope of the project.

4 Recap Use-Case and application of AMPERE ecosystem

The Automotive use-case and the use-case evaluation was described in D1.5 [3] in Chapter 4. For easier readability we provide a small recap of the use-case and the application of the AMPERE ecosystem to it. As depicted in the use-case consists of four main applications in a system-of-systems use-case. An existing Cyber Physical System (CPS) is enhanced with a new CPS implementing a new advanced functionality.

The use case consists of four components: the already existing *Adaptive Cruise Control* (ACC) and the *powertrain control* subsystem (ECM), and the new advanced functionalities PCC and *Traffic Sign Recognition* (TSR) subsystem.

The power-train control and the adaptive cruise control are representing two classical CPS or AUTOSAR systems while the two new CPS systems are a traffic sign recognition and the predictive cruise control. For composability reasons the application exchanges data via a publish/subscribe middleware. In the AMPERE scope we use ROS2 as a representative for the new AUTOSAR Adaptive standard.

The evaluation of the automotive use case does not consider functional aspects of the implementation but focuses on adherence to non-functional requirements, composability, and the efficient and easy usage of heterogeneous hardware platforms with accelerators.

The provided use-case was evaluated on two different platforms, the NVIDA Jetson AGX and the Xilinx Ultrascale Plus.

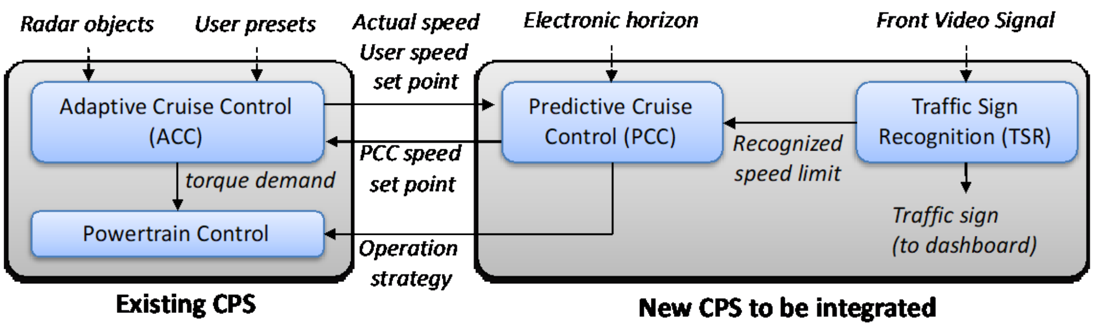


Figure 5-1: Automotive Use-Case

5 Evaluation of the AMPERE ecosystem for the Automotive Use-Case

5.1 Requirements

The evaluation of the toolchain follows the requirements stated in the Deliverables D1.1 [9] and D1.2 [10]. The AMPERE project does not explicitly aim at guaranteeing functional correctness and the PCC use-case therefore consists of representable code snippets mimicking the impact of computation and access to shared resources on the system performance. Therefore, the evaluation focuses on how the methods and tools of AMPERE adhere to the requirements stated for the Automotive domain with the focus on prevention of timing and spatial interference, which is a prerequisite for certifying the system at a reasonable time and cost.

5.1.1 Timing

- **[SYS-PCC-REQ-101]**: The implementation shall make efficient use of resources in order to be accepted in the cost-sensitive high-volume automotive market.
  Test method Expected: Analysis.
  Evaluation: The AMPERE Toolchain automatically generates the task dependency graphs for the use-case provided as an Amalthea model. It provides an automatic instrumentation, profiling and optimization design flow that explores and exploits the parallel capabilities of the hardware platform and its accelerators. The toolchain allows for an automatic detection of even fine-grain parallel execution capabilities in the software which enables a fast performance analysis and exploration of the system. The automatic toolchain allows an extensive evaluation of deployment and implementation alternatives enabling an optimization for resource efficiency. Even functionality that is not yet implemented can be considered via the synthetic code generator starting from abstract descriptions in the model.
- **[SYS-PCC-REQ-102]**: The HW/SW platform shall support the dynamic addition of applications (also during operation time) without jeopardizing the correct functioning of the existing applications deployed on the platform (performance compositionality).
  Test method Expected: Demonstration.
  Evaluation: The AMPERE toolchain uses mechanisms to spatially and temporally sperate applications from each other if needed. In the use-case on the Xilinx board the PikeOS hypervisor was used as a virtualization technique to separate the ASIL-B ECM running on a safety-certified ErikaOS from the other applications in the system running on a Linux partition. The concepts were described in D5.3 [11] and D5.4 [12].
  In the Linux partition both use-cases uses SCHED_DEADLINE. The options are explained in D3.3 [7] and in the run-time architecture overview in D4.4 [13]. With EDF a high utilization is possible in the system, with the constant bandwidth server (CBS) it is possible to also bound the computation time of tasks and therefore the interference to other applications.
  The controlled bounding of interference on shared resources like the DRAM memory was highlighted in D3.2 [6] and D3.3 [7] and adds another layer towards predictable performance. The final paper submission to ECRTS 2023 [14] on this topic received the outstanding paper award. Dynamically adding applications during operation time was out of the scope of AMPERE.

- **[SYS-PCC-REQ-103]**: The AMPERE ecosystem shall support the specification of timing constraints (most notably end-to-end latency) as well as the assurance of their adherence.
  Test method Expected: Demonstration.
  Evaluation: Timing constraints are modelled in the use-case model provided by an Amalthea model. They include deadlines on the response times of individual tasks as well as end-to-end latency requirements including multiple tasks in the chain. They are propagated through the toolchain to the multi-criteria analysis which assures the adherence to the timing constraints. The constraints can be found in the model, partially described in D1.5 [3], Chapter 4.1.2. The traceability and the evaluation of adhering to the end-2-end latency requirements are described in D3.4 [15].

- **[SYS-PCC-REQ-104]**: The AMPERE ecosystem shall support applications that are resilient to sporadic timing errors by appropriate mechanisms such as error counters and mitigation actions.
  Test method Expected: Demonstration.
  Evaluation: In addition to the SCHED_DEADLINE details explained in **[SYS-PCC-REQ-102]** and D3.3 [7], the Greedy Reclamation of Unused Bandwidth (GRUB) allows tasks to reclaim bandwidth unused by other real-time threads in order to cope with sporadic timing overruns.

5.1.2 **Energy**

- [**SYS-PCC-REQ-105]**: The AMPERE ecosystem should include development tools and run-time mechanisms enabling an energy-aware design in order to guarantee an energy-efficient execution of the applications.
  Test method Expected: Demonstration
  Evaluation: The multi-criteria analysis provides an optimization to minimize the energy consumption while still adhering to the timing constraints provided by the use-case model. The results are explained in D3.4 [15].

5.1.3 **Safety**

- **[SYS-PCC-REQ-106]**: The AMPERE ecosystem should fulfil the requirements of the functional safety standard ISO26262.
  Test method Expected: Inspection
  Evaluation: In the scope of AMPERE, we focused on the ISO 26262 requirements which are relevant for the optimized mapping and parallel execution of SW elements on heterogeneous HW platforms. Other ISO 26262 requirements are dealt with in the ITEA3 project PANORAMA [20] which is also based on APP4MC and the common meta model AMALTHEA. Thus, PANORAMA results such as the structured modeling of assurance cases via the Open Dependability Exchange Meta model (ODE) or the traceability between (safety) requirements and design elements via Eclipse Capra will also be compatible with AMPERE results. The PCC consists of different subfunctionalities or applications with different ASIL classifications which will have to be simultaneously executed on the same HW/SW platform (see [**SYS-PCC-REQ-107**]). These applications (even the ones with the same ASIL) have to be spatially and temporally isolated from each other in order to prevent interference and fault propagation between them (see [**SYS-PCC-REQ-108**]). Additionally, high ASIL applications need to be protected against random HW faults by being executed in lockstep which can be realized via dedicated lockstep HW cores or by redundant execution of the respective SW elements of the application (see [**SYS-PCC-REQ-109**]).

- **[SYS-PCC-REQ-107]**: The HW/SW platform shall support the simultaneous execution of applications with different safety-criticality levels (ASIL).
  Test method Expected: Demonstration.
  Evaluation: The automotive use-case includes multiple applications with different ASIL levels. The deployment was demonstrated on the evaluation boards, e.g., the Xilinx board hosted the ASILB engine control management on a separate ErikaOS partition while the e.g., the replicated ACC ASIL A function was running on the Linux partition. On top of that QM software was also running on the Linux machine as demonstrated in D5.4 [12]. **The evaluation of the replication method was presented in D3.4[15].**

- **[SYS-PCC-REQ-108]**: The HW/SW platform shall separate applications from each other spatially and temporally such that they do not interfere with each other, and such that cascading failures are prevented.
  Test method Expected: Demonstration.
  Evaluation: See evaluation of **[SYS-PCC-REQ-102]**

- **[SYS-PCC-REQ-109]**: The HW/SW platform shall support the implementation of high ASIL applications on HW execution units without HW lockstep by providing adequate means to allow SW lock-step implementations.
  Test method Expected: Demonstration.
  Evaluation: The AMPERE Toolchain allows the generation of automatic replicas for safety critical software. The automatic replication method was evaluated for the automotive use case in D3.4 [15], Chapter 2.1.

- **[SYS-PCC-REQ-110]**: The HW/SW platform shall warrant deployment-independent behaviour such that a distributed application exhibits the same deterministic input-output behaviour regardless of how it is deployed onto the platform.
  Test method Expected: Test.
  Evaluation: We can adapt the ROS2 communication to support LET execution scheme which warrants deployment independent behaviour (D3.1). Replacing the ROS2 communication primitives with AUTOSAR communication primitives, logical execution time (LET) communication semantics are provided by the AUTOSAR software stack. Since the AMPERE multi-criteria analysis ensures the adherence of task deadlines a deployment-independent input-output behaviour is provided. Additionally, when exploiting the intra-task, inter-runnable parallelism of the automotive use-case, the data-flow analysis identifies data dependencies and ensures a deterministic input-output behaviour between parallel worker threads. This work was described in D2.2 [16], D2.3 [17] and D2.4 [18].

- **[SYS-PCC-REQ-111]**: The AMPERE ecosystem shall support the execution of AUTOSAR components (AUTOSAR Classic and ROS2 as substitute for AUTOSAR Adaptive).
  Test method Expected: Demonstration.
  Evaluation: Demonstrated on the Xilinx board, running the ECM on the AUTOSAR compliant ErikaOS in parallel to a Linux system hosting ROS2 components D5.3 [11], D5.4 [12].

- **[SYS-PCC-REQ-112]**: The AMPERE ecosystem shall support AMALTHEA as an exchange format to existing automotive toolchains and a base for code generation.
  Test method Expected: Demonstration
  Evaluation: Amalthea is the starting point for the AMPERE toolchain for the automotive use-case, see the tool flow in D2.5 [19].

- **[SYS-PCC-REQ-113]**: The AMALTHEA model shall be extended to cover the key characteristics of the CPSoS (software parallelism, accelerator offloading, and publish-subscribe middleware communication paradigms).
  Test method Expected: Demonstration.
  Evaluation: In AMPERE we added a mechanism in AMALTHEA to specify specialization of runnables to express the different memory/timing behaviour of runnables within one model. Callers of the runnables can specify the implementation to be triggered while the AMPERE Ecosystem enables the automatic offloading to hardware accelerators. Additionally, modelling mechanisms were added to specify typical activation patterns in publish-subscribe middleware's like callbacks and additional modelling elements were added to specify the communication queue accesses. This allows a detailed modelling and the corresponding code generation of the execution model dealing with publish/subscribe middleware's explained in D1.5 [3].

- **[SYS-PCC-REQ-114]**: The AMPERE ecosystem shall allow incremental (re-)design with localizable changes (in contrast to global optimization only where local changes potentially change everything globally).
Test method Expected: Demonstration
Evaluation: The platform-runtime supports freedom-of-interference mechanisms as evaluated in **[SYS-PCC-REQ-102]**. In general, the multi-criteria analysis could take an existing system deployment and configuration as a constraint set to make incremental optimizations.

- **[SYS-PCC-REQ-200]**: The AMPERE ecosystem shall support to express the membership of a SW element to an application. (This is required for the correct configuration of the temporal and spatial isolation mechanisms of the HW/SW platform to prevent fault propagation between SW elements of different applications.)
Test method: Inspection.
Evaluation: We used the tag system in AMALTHEA to indicate the membership of a runnable to an application as explained in D1.4 [1] and D1.5 [3].

- **[SYS-PCC-REQ-201]**: The AMPERE ecosystem shall support the assignment of ASIL to a SW element.
Test method: Inspection.
Evaluation: We used the tag system in AMALTHEA to assign the ASIL level of a runnable/application D1.4 [1] and D1.5 [3].

- **[SYS-PCC-REQ-202]**: The AMPERE ecosystem shall support the assignment of ASIL to a HW element.
Test method: Inspection
- **Evaluation:** We used the tag system in AMALTHEA to assign the ASIL level of a HW element D1.4 [1] and D1.5 [3].

- **[SYS-PCC-REQ-203]**: The AMPERE ecosystem shall support to express mapping separation constraints between (groups of) SW elements denoting that these (groups of) SW elements may not be mapped to the same HW element.
Test method: Inspection.
Evaluation: Mapping separation constraints are modelled in Amalthea and respected in the multi-criteria optimization.

5.1.4 AMPERE Use Case Key Performance Indicators (KPIs)

- Satisfy the high computation demands of PCC algorithms while guaranteeing the safety properties of the powertrain control and ACC functionalities.
Measure: High system utilization (> 90%) with provable safety properties
Evaluation: The evaluation in D2.5 [18], D3.4 [15], D4.4 [13] and D5.4 [12] shows different deployment options of the automotive use-case. E.g., the powertrain control run on a dedicated core on the AUTOSAR ErikaOS. Real products of the powertrain and an AUTOSAR-compliant operating system are running in real deployed products. The Linux partition supports SCHED_DEADLINE with CBS or EDF to either guaranteed fixed execution budgets or guarantee a system utilization of up to 100% for the software components with software-lockstep replicas for safety-critical software.

- Maintain the functional properties of the PCC when integrating further synthetic applications, to demonstrate the compositional integration capabilities of the AMPERE ecosystem.
  Measure: Maintain exactly the same functional properties
  Evaluation: See evaluation of **[SYS-PCC-REQ-102]**

- Providing a reduced development effort for integrating new functionalities in an existing system, by coupling the AMPERE ecosystem with existing automotive standards and tools.
  Measure: 30% reduction of development efforts
  Evaluation: Programming highly parallel platforms with different heterogeneous computing units is a very complex, error-prone, and time-consuming endeavour. Estimating the impact of decisions with respect to several criteria such as time and energy is practically impossible to be handled manually by a developer. Having developers and system integrators, understanding the impact of concurrent execution, and exploiting the provided parallelism by the hardware platform including mastering the different technologies to use hardware accelerators efficiently and safely is rare. Hiding the complexity of the technology and optimizations in tools like in AMPERE should close the productivity gap we currently see in the engineering of modern automotive applications. We did not systematically evaluate the development effort reduction, but a 30% reduction seems a reasonable assumption in a compositional use-case where functionality with different ASIL levels need to be integrated and efficiently deployed.

6 Acronyms and abbreviations

Acronym	Description
ACC	Adaptive Cruise Control
ADAS	Advanced Driver Assistance Systems
ASIL	Automotive Safety Integrity Level
CPU	Central Processing Unit
DSML	Domain-Specific *Modeling* Languages
FoV	Field of View
FPGA	Field Programmable Gate Arrays
GPU	Graphic Processing Unit
GRUB	Greedy Reclamation of Unused Bandwidth
HW	Hardware
IDE	Integrated Development Environment
KF	Kalman Filter
KPI	Key Performance Indicator
LET	Logical Execution Time
LiDAR	Light Detection and Ranging
LRT	Light Rail Transit
NGAP	Autonomous Positioning System
ODAS	Obstacle Detection and Avoidance System
PCC	Intelligent Predictive Cruise Control
ROS	Robot Operating System
RTOS	Real-Time Operating System
SFA	Sensor Fusion Algorithm
SIL	Safety Integrity Level
SW	Software
ToF	Time of Flight
TSR	Traffic Sign Recognition

Table 1 Acronyms and descriptions

7 References

[1] "D1.4 Analysis of functional safety aspects on multi-criteria optimization and final release of the test bench suite"; AMPERE Deliverable; M33

[2] "D6.4 First release of the AMPERE ecosystem"; AMPERE Deliverable ; M33

[3] "D1.5 Meta model-driven abstraction and model-driven extensions and use case enhancements"; AMPERE Deliverable; M33

[4] Capella; Eclipse Foundation; https://www.eclipse.org/capella/

[5] ARCADIA; Eclipse Foundation; https://www.eclipse.org/capella/arcadia.html

[6] "D3.2 Single-Criterion Energy-Optimization Framework, Predictable Execution Models, and Software Resilient Techniques"; AMPERE Deliverable; M15

[7] "D3.3 Energy optimisation framework, predictable execution models and analysis, and

Software resilient techniques"; AMPERE Deliverable; M33

[8] "D4.2 Independent run-time energy support, and predictability, segregation and resilience mechanisms"; AMPERE Deliverable; M15

[9] "D2.4 Multi-criteria optimization model transformation"; AMPERE Deliverable; M42

[10] "D3.4 Evaluation of multi-criteria optimizations"; AMPERE Deliverable; M33

[11] "D4.4 Evaluation of run-times" AMPERE Deliverable; M42

[12] "D5.4 Evaluation of the operating systems and hypervisors" AMPERE DeliverableM42

[13] https://www.hitachi.com/rev/archive/2020/r2020_06/06a04/index.html

[14] "D1.1 System models requirement and use case selection; AMPERE Deliverable; M6

[15] https://github.com/eclipse/capella/commit/7edd98d0ebecbe2073842fa562de54207674ecef#diff-9537e444c48cea71d2a8ea09f96410c82c56b0045cd99feb4a937ed92c630e1e

[16] "D6.4 First release of the AMPERE ecosystem" ; AMPERE Deliverable; M33

[17] https://www.eclipse.org/capella/arcadia-reference.html

[18] Model-based System and Architecture Engineering with the Arcadia Method; Jean-Luc Voirin; Hardback ISBN: 9781785481697; eBook ISBN: 9780081017944

[19] https://www.eclipse.org/app4mc/

[20] OBSTACLE DETECTION AND TRACKING (ODT) SYSTEM FOR TRAMS; ODT 1.0 MAIN GATE REPORT; GTSI Internal use only.

[21] "D4.3 Integrated run-time energy support, and predictability, segregation and resilience mechanisms"; AMPERE Deliverable; M33

[22] "D5.3 Final (multi-criteria) operating systems and hypervisor"; AMPERE Deliverable; M33