(ITEA 3 – 17003)

PANORAMA
Boosting Design Efficiency for Heterogeneous^3 Systems

Deliverable: D 7.2
Requirements and evaluation criteria

Work Package: 7
Demonstration

Task: 7.2
Specification of requirements for technological and methodological solutions, and definition of evaluation criteria

Document Type:	Deliverable	**Classification:**	Public
Document Version:	Improvement phase	**Contract Start Date:**	2019-04-01
Document Preparation Date:	2021-01-31	**Duration:**	2022-03-31

History

Rev.	Content	Resp. Partner	Date
1	Initial version	Siemens AG and others	2020-06-30
2	Baseline phase	Siemens AG and others	2020-07-21
3	Improvement phase	Siemens AG and others	2021-01-31

Contents

List of Figures

List of Tables

Summary

This deliverable is the third document of the PANORAMA Work Package 7 ”Demonstration”. It contains the results of improvement phase in task T7.2 ”Specification of requirements for technological and methodological solutions, and definition of evaluation criteria”.

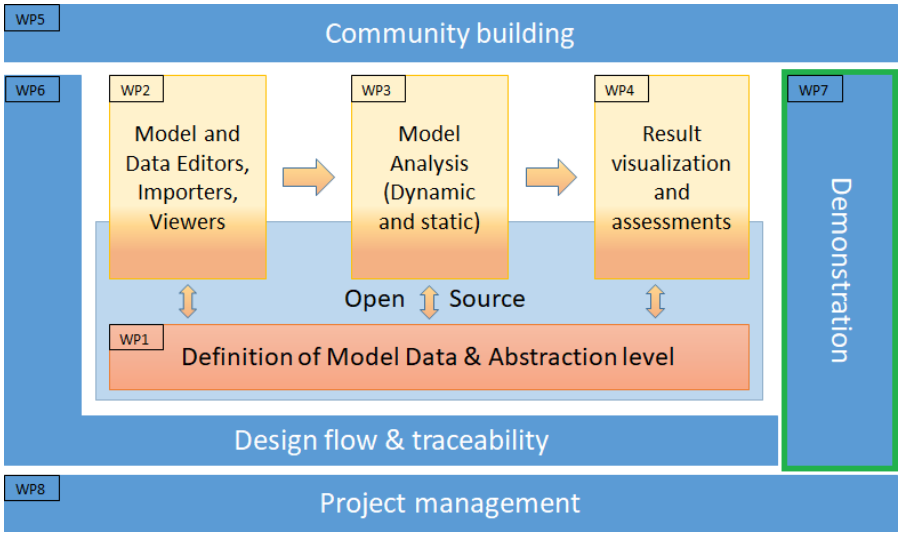


Figure 1: PANORAMA Work Package Structure including their relation to WP7

In order to put this document into context, the relation between WP7 (green box) and the other work packages is illustrated in Figure 1. The project is organized into two groups of work packages: The technical solutions to the project objectives are developed by WP1-4 (colored in red / orange), whereas WP5-8 cover management, packaging, and dissemination related topics (colored in blue).

1 Introduction

1.1 Purpose and objectives

This deliverable provides an update of initial requirements and evaluation criteria for individual planned demonstrators described in deliverable 7.2 (baseline phase). Based on the specification of the demonstrators described in deliverable 7.1 and upon completion of corresponding initial implementations in task T7.3 "Implementation of the demonstrators, baseline phase", the document mainly focuses on illustrating how the demonstrators should work in the contexts of technological and methodological work packages 1, 2, 3, 4, and 6 and how individual technical achievements can be addressed by evaluation criteria of planned demonstrators. The specifications of the baseline version were analyzed and refined by the partners in order to determine a set of important aspects that are potentially not covered by a demonstrator but would be essential to elaborate the complete set of evaluation criteria for technical achievements. To some extent, the document also provides an update of planned demonstrator use cases with refined / extended features of respective proof-of-concept platforms.

1.2 Structure of the deliverable

Industrial and academic demonstrators List of requirements and evaluation criteria

Outlook Summary with brief analysis of described requirements and evaluation criteria and description of the next steps and contributions.

2 Industrial and academic demonstrators

In the following sections we provide an update of individual descriptions to give an up-to-date understanding on how the demonstrators specified in deliverable D7.1 should work in the contexts of technical achievements in technological and methodological work packages 1, 2, 3, 4, and 6. The descriptions mainly focus on requirements and evaluation criteria (if applicable, including KPIs). Furthermore, the demonstrator specifications are refined and/or extended by further relevant demonstration scenarios and challenges.

2.1 AVL ADAS Feature Simulator

2.1.1 Introduction

Advanced drive assistance systems (ADAS) are the initial move toward a completely automated future. Features like Adaptive Cruise Controller (ACC), Lane Changing Assistant (LCA), Lane Keeping System (LKS) help drivers and diminish the hazard factor. But, before releasing these ADAS products, they need to go through diverse testing procedures to prove their safety and reliability. However, with the growing complexity of the system, the testing and assessment criteria becomes complex. The safety and reliability of ADAS feature needs to be approved in complex traffic situations, including traffic participants. This kind of higher-level system is tested by a distance-level approach which means holding test drives. Be that as it may, real-world testing won't cover numerous unforeseen scenarios that a vehicle can get into. As, a result of that, assessing a car's on-road performance isn't the most excellent thought for ADAS feature testing. Additionally, real-world tests are costly, and time expending. It is additionally specified here, test drivers are perilous for everybody, particularly for drivers. All in all, road testing alone won't guarantee an ADAS feature's safety. However, creating a virtual environment for ADAS testing means modeling a complete driving scenario which includes all the traffic participants, using software is a promising alternative. In contrast with real-world testing, the virtual environment is safe. Further, it permits testing in different scenarios.Moreover, virtual environments for ADAS testing help to prototype and develop new system features. The key element of virtual environment is the scenario file. Scenario generation is a significant step to evaluate the safety and reliability before releasing a product. So, the scenarios have to be derived and documented methodically. In addition, traceability along the development process is also a crucial evaluation criterion. There are some methodologies in literature for scenario generation, but A three level of abstraction which are functional scenarios, logical scenarios and concrete scenarios is mostly preferred. Functional scenarios are described in a linguistic way by the help of the experts scenarios in the beginning of the development process. Then, logical scenarios which specify parameters for the scenarios and define parameter ranges is derived by using functional scenarios. Finally, concrete scenarios are produced to specify a concrete value for each parameter and, thus, a various of test cases are created. Basically, each scenario includes a start scene and an end scene. So, the transformation from functional to logical scenarios, the keyword-based representation requires to be detailed in a parameter space

representation and after that changed over into the formats for the simulation environment. This phase also involves the definition of the traffic participants interactions between the start and the end scene within the scope of a sequence control for simulation. It is important to consider relationships and dependencies between parameters and parameter values in order to accurately model the road and the interactions of the traffic participants according to the data formats. The scenario description and scenario conversion steps currently require enormous manual effort. In addition, a structured interpretation of linguistic scenarios and a coherent representation compatible with the data formats for the simulation setting are not guaranteed, in particular for multiple authors. In this project, a tool for an automated transformation from a keyword-based scenario description to a parameter space representation and a conversion into the formats for a simulation environment will be provided by using the example of ADAS Adaptive Cruise Controller feature scenarios on highways. As a simulator, an unreal engine-based simulation environment will be used. The relationships and dependencies of the elements and the parameters of the scenario will be modelled within the space representation parameter. It is important to ensure that the interactions between the traffic participants are resolved in compliance with the specification of the respective functional scenario and to ensure consistent implementation of the data formats compliant scenario. The data format OpenDRIVE is used for the representation of the road network as well as the data format OpenSCENARIO is used for the representation of the traffic participants and the environment. The main goal of this demonstration is to create an automated way for scenario generation, increase the design efficiency and reduce the time-consuming. By doing so, the heterogeneous structure between developing software and data formats will be eliminated.

Mantis is responsible for proposing a novel scenario definition language, which can be used for specifying visual scenario models in the context of Adaptive Cruise Control (ACC) systems. The visual scenario models specified with our scenario definition language consider two perspectives, the requirements and behaviour. The requirements perspective is for specifying the constraints on the environment. Herein, the environment constraints are concerned with the road information (e.g., road surface, road type, lane marking type, road curvature, and road width) and other information such as weather condition, time of day. With the behaviour perspective, firstly, the initial state of the vehicles that contribute to the scenario are specified in terms of their speed, positions, and lanes. Then, the behaviour perspective model may describe how the target vehicles change their state dynamically (i.e., change lane, speed, or position) given some conditions satisfied.

To design the scenario definition language, Mantis performed an empirical study and conducted a series of interviews with the AVL engineers and intended to understand their challenges on creating scenarios using their current settings and their expectations from the new language. The interviews were so useful in understanding the abstract and concrete notation set required for the new language and some other requirements such as editor support, code generation, model analysis, etc.

So, Mantis focussed on developing a modeling editor for the scenario definition language using the Eclipse Modeling Framework (EMF) in order to integrate the editor into any platform compatible with EMF (e.g., Eclipse Capra). To this end, Mantis used the Eclipse Sirus metamodeling tool and developed the first prototype of the modeling editor through which the scenario models from the requirements and behaviour perspectives can be specified. Having got feedback from the AVL engineers, the modeling editor has been revised and corrected to better meet the needs of practitioners. Currently, Mantis has been using the constraint definition technology of Eclipse Sirius so as to introduce the pre-defined well-formedness rules

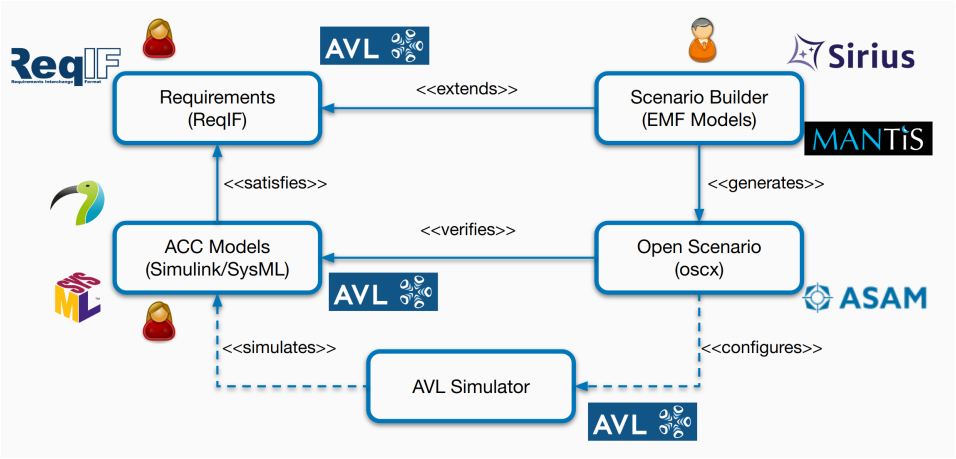


Figure 2.1: Diagram of the Archictecure

that can be checked automatically at modeling time and need to be satisfied for semantically correct scenario models as it can be seen in Figure 2.1.

Mantis will also develop and integrate a code generator for our modeling editor, which is supposed to produce OpenScenario models from any scenario models created with our language so as to make the scenario models processed by the simulators that support the OpenScenario standard. Mantis will use Eclipse's Acceleo technology for developing the code generator and integrating that into our modeling editor developed with Sirius.

Further to studies discussed above, Mantis will consider checking the requirements perspective of the scenario models with regard to the initial requirements of the scenarios specified in the ReqIF format. By doing so, any inconsistency and incompleteness issues between the initial set of requirements and the scenario model requirements can be determined and the necessary actions can be taken early on.

2.1.2 Related Work

[UMR+15] propose a basic definition of terms scene, situation, and scenario for the autonomous vehicle domain. These definitions are used by [MBM18] to define various scenario representation notations during the development process. Therefore, for scenarios, they suggest three levels of abstraction: functional scenarios, logical scenarios, and specific scenarios described below. [SSL+13] propose a scenario-based test process and a 4- layer-model for structuring scenarios. Various approaches are proposed for the generation of scenarios along the automated vehicle development process. As indicated by the ISO 26262 standard, [BRSM16] establish a method for creating potentially hazardous scenarios within the process phase of a hazard analysis and risk assessment. [BOS16] suggest, as a general scenario notation along the development phase of the ISO 26262 standard, a model-based scenario representation with spatial and temporal relations. It should be noted that complete specifications for ADAS systems can only be achieved through a reliable, traceable and verifiable requirements engineering process in compliance with the V-model [BJS+15]. Two methods that are data-driven approach and knowledge-driven approach are being used in today's scenario generation. The key idea of a data-driven approach is to collect and identify the measurement data and classify occurring scenarios as well. There are some difficulties for most data-driven strategies. First,

measurement data does not define every aspect of a scenario. Additionally, when summarizing scenario-based representations as logical scenarios, consideration needs to be given to further knowledge in order to specify parameter dependence. Data-driven approaches do not provide information about what has been encountered from the operational design domain so far. Data-driven approaches can be combined with a knowledge-driven approach to mitigate the mentioned problems. The key concept is to create scenarios from existing knowledge, such as regulatory guidelines, and then enhance these synthetic scenarios with information gathered from measuring results which is used as a concept for this project. The simulation environments utilize different data formats to specify scenarios. The tool presented in this project transforms functional scenarios into the data formats for the unreal engine-based simulation environment. For the description of the road network, the simulator requires the data format OpenDRIVE. For the description of the traffic participants and the environment, the simulator utilizes the event-based data format OpenSCENARIO which is the key element of the project. Some publications define an event-based representation of the scenarios. [BOS16] define a framework and a prototypical application for the abstract representation of concrete scenarios. The definition is based on splitting a situation into separate actions and events. The definition then realizes a similar framework for traffic participant interactions as the OpenSCENARIO data format. [Xio13] presents a method of the scenarios in a knowledge-based simulation environment. The ontology describes in detail the elements of the road network and the traffic participants interactions as well as the models to be used. The simulations are performed by a sequence control system in a simulated environment, and if necessary tailored to the actions of the driving function. They also provide event-based assessments for the respective test cases in concrete scenarios. However, in both papers, they do not define an method for automatic details of a keyword-based scenario description to an event-based scenario description as the basis for the production of a test case either. The concepts and implementations presented in this project use practical scenarios developed as the starting point for detailing the scenarios and translating them into OpenSCENARIO data format.

2.1.3 Use Case Definition

In this project, it is basically realized that, the transformation from a keyword-based scenario description into the formats for execution in a simulation environment. As a visualization instance, an Adaptive Cruise Control System Scenarios for Highways will be generated by using the proposed methods and developing tools. The main architecture of the concept is given in Figure 2.2.

As starting point for the process, functional scenarios are created for ACC system. These functional scenarios can be created by the help of expert and also the results of functional safety analysis. Each functional scenario is a semantic, keyword-based description. An example of functional scenario is given in Figure 2.3.

Then, the functional scenarios are imported and transformed into a representation of space parameters. For this, every term of the respective functional scenario is defined by parameters in detail. The parameters are deduced by testing which parameters to represent the corresponding element in the data formats are to be specified for simulation. The modelling tool is modelled the scenario by using the parameter space. The output of modelling tool creates the input of connector tool. The role of this tool creates an OpenSCENARIO data format file for virtual simulation as well as the input parameters for controller code run on Matlab/Simulink environment. A basic flow diagram of the planned system is shown as Figure 2.4. A semantic

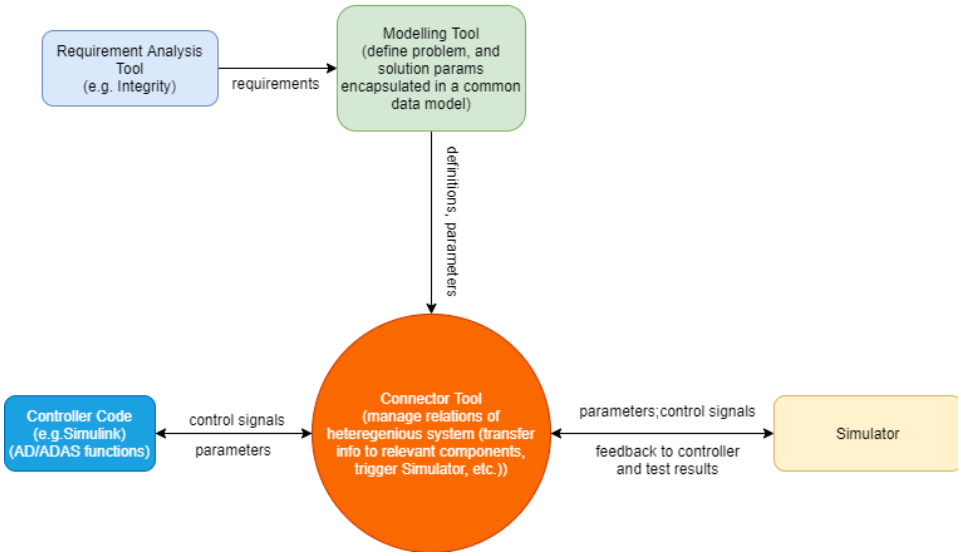
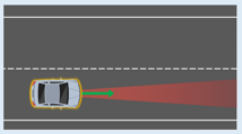


Figure 2.2: High level architecture for demonstrator use case

scenario representation, which is modeled with the model-based language data format, is used as the basis for scenario description. Such functional scenarios use a pre-defined vocabulary as well as semantic relationships between vocabulary words to describe the road level, traffic infrastructure, moving objects, and environmental conditions. The terminology used for the definition of the scenario are arranged according to the proposed model. Each term is then augmented with parameters that detail the respective feature. The parameters needed for the definition of each feature depend on the simulation environment's data formats. Therefore, the parameters extract from the specifications of OpenSCENARIO data formats which are used by real-time the simulation environment. The data format OpenSCENARIO describes the interactions of traffic participants as well as the environmental conditions which will be our main outcome of the tool chain. The data format OpenDRIVE describes the road network and a predefined OpenDRIVE file will be used for the simulation environment. As an example of OpenSCENARIO file can be defined as follows; the trajectories of traffic participants are represented via actions and events. For example, actions can be maneuvers and other simulation control commands as well as event cause these actions. In functional scenarios, the movement of each vehicle is defined in the start and end scene, as well as the maneuver to be performed, by its relative location with other traffic participants. To depict the maneuvers defined in the functional scenarios with the aid of OpenSCENARIO parameters, the traffic participants' interactions should be converted into an event-based representation. The relationships and constraints between the elements and parameters of the elements are modelled in the second stage of modelling process. Three relationship types have been defined and are used for definition of the scenario: arrangement relationships, entity dependencies, and parameter dependencies. All three types of relationships are mutually exclusive and focus on various aspects of the scenario generation [MBI+19]. These relationships will be detailed in the following parts of the projects. In the final step, the representation of the traffic participants and environmental conditions in OpenSCENARIO is generated. In addition, the set of rules used to determine default values for each parameter is recorded in a separate file. By doing so,

Scenario Description	Rationale	Figure	Pre Condition	User Action	Acceptance Criteria
The Ego vehicle moves a straight road and the ACC Feature is enabled	To test feature is enabled and inactive when requested by the driver when inhibiting conditions are not present.		ACC Feature disabled. Ego vehicle on, either standstill or moving.	Tester requestes ACC Feature activation through the user interface.	**Passed:** ACC is enabled and no longitudinal control is provided by ACC. If inhibiting conditions not present: ACC available for activation. **Failed:** ACC does not changes its mode.

Expected Behaviour on HMI	Post Conditions	Comments	Test Environment	Target Object(s)	Road Type	Road Surface	Lane Marking Type	Road Curvature (in terms of radius)	Road Width
Information of feature in standby 1. Information of feature available for activation 2.Request driver to set target speed			Simulation Test	No Target	Highway or Highway-tunnel	Dry, Wet	Solid, Dashed	Any	standard highway, narrow lane (2.70m < w < 3.40m)

Ego Speed	Target Speed	Target Position	Time of day	Weather conditions	Road Coondition	Action time	Action distance	Traffic density	Ego lane position
Any	No Target	No Target	Time Date	Sun Fog Precipitation				standard	middle driving lane, right driving lane, left driving lane

Figure 2.3: Functional scenario example

Figure 2.4: Flow diagram of the system

the collection of rules can be used to produce appropriate combinations of parameter values in the following process steps, for example as part of the generation of test cases.

2.1.4 Evaluation Criteria

The OpenSCENARIO files can be analyzed in two levels. In the first step, the created OpenSCENARIO files are checked against a schema file during a static test. In this sense the syntax for each file of OpenSCENARIO is contrasted with the specification of the formal OpenSCENARIO data format. This way, missing elements and attributes can be identified. In the second stage, in a simulation environment, the OpenSCENARIO files are loaded and executed. In this process, it is verified whether each created OpenSCENARIO file is executed in simulation according to the respective functional scenario. Some key questions during this test can be given as: Do all vehicles start at the given starting position? Does each vehicle's conduct conform to the maneuver described in the start scene? Do all maneuvers start and finish at the right time, so that there are no crashes?

2.1.5 Requirements

In this section, the demonstrator's high-level requirements correspond to the above information are described;

- The proposed toolchain framework will allow implementing ADAS features a virtual simulation environment based on an OpenScenario data format.
- Each keyword-based scenario is transformed into a parameter space representation and afterwards converted into the data formats and OpenSCENARIO.
- All the functional requirements will be traced along the all process by the help of Eclipse Capra platform.

2.2 Automated Traceability Analysis and Configuration for Eclipse Capra

2.2.1 Contribution

UNIT has been working on integration of Automated Traceability Analysis and Configuration modules of Tarski platform (`https://modelwriter.github.io/Tarski/`) initiated in the ITEA-ModelWriter project (`https://itea3.org/project/modelwriter.html`) with the Eclipse Capra Platform (`https://eclipse.org/capra`) in coordination with Panorama project member, Software Engineering Division of Chalmers | University of Gothenburg. The main blocks under current development are as follows:

- The Configuration Language for Traceability Semantics: the automated analysis module provides the capability of consistency and completeness checking of traceability links according to the given semantics provided by the user [EGTK17].
- The Automated Reasoning Support for Traceability: a domain specific language to configure the semantics of traceability types [EGG+17].

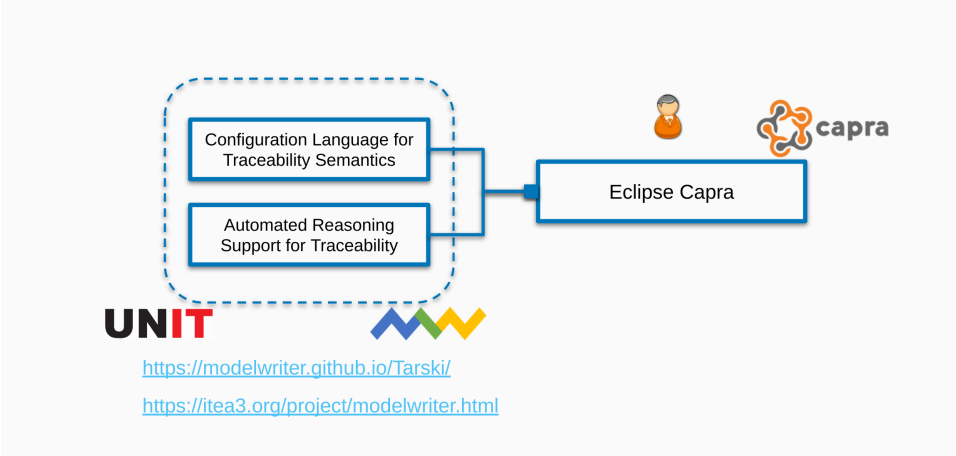


Figure 2.5: Eclipse Capra Demonstration

2.2.2 Demonstration

UNIT will demonstrate the analysis on two use cases, one of which is the project's **WATERS** challenge use-case scenario and the other is the development and simulation of **AVL**'s Adaptive Cruise Control (ACC) function. In the context of the latter, the traceability links will be maintained by means of Eclipse **Capra** tool and established mainly among the requirements, simulation scenarios provided by **Mantis**' EMF-based Scenario builder tool, and SysML models of the ACC's controller. Based on the configuration semantics, thanks to the Eclipse Capra tool and the aforementioned modules, we will be able to demonstrate the detection of inconsistencies among the development artifacts and confirm whether simulation scenarios generated by the Scenario Builder would be sufficient enough to cover the timing requirements and system models.

2.3 Cloud-based performance simulation service

A new cloud-based approach for a joint development provides a common collaboration environment with standardized services and workflows. Instead of individual IDEs on local machines the cloud environment can be used to deploy a commonly defined workflow and tool setup. With that definition the collaborative parties can use the same dataset, infrastructure and assessment results to optimize the system output. New ways of collaboration can be developed in this way to overcome misunderstanding and data transfer in all directions.

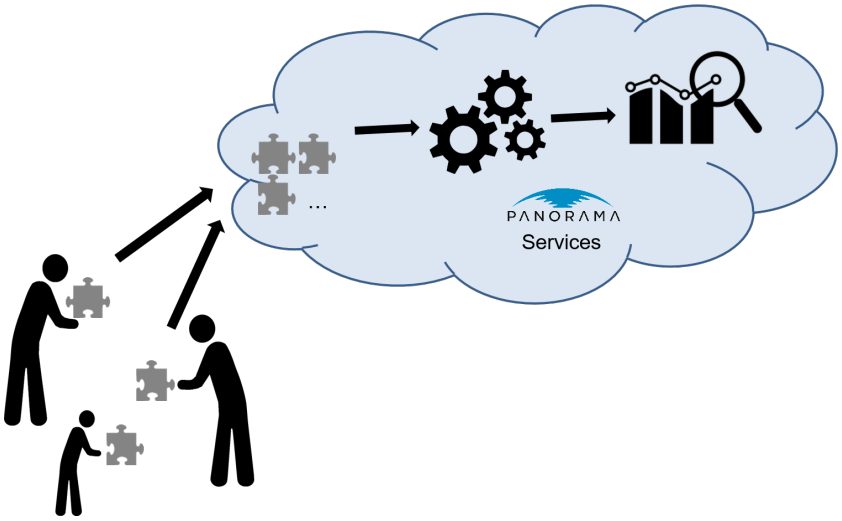


Figure 2.6: Collaboration based on Panorama Cloud Services

The cloud infrastructure will be hosted under the Eclipse Cloud System. Users of this Open Source service get an impression of how the overall framework enables new and easy possibilities to analyze the given system. There is no individual installation necessary, but if required adaptations and combinations with commercial tools are possible. In some cases this is essential to achieve highly sophisticated simulation and optimization.

2.3.1 Architecture

The main objective of the demonstrator is to provide an collaborative environment to demonstrate our Open Source possibilities. They enable the developers to evaluate, simulate and assess the provided data according to their performance analysis needs.

Our Open Source tool environment Eclipse APP4MC expands into a tool infrastructure hosted in an cloud environment. This enables companies to join their efforts in a common collaborative environment. We showcase the possibility using our data-model and parts of our tool landscape enhanced by self-contained services like analysis, simulations or result visualizations.

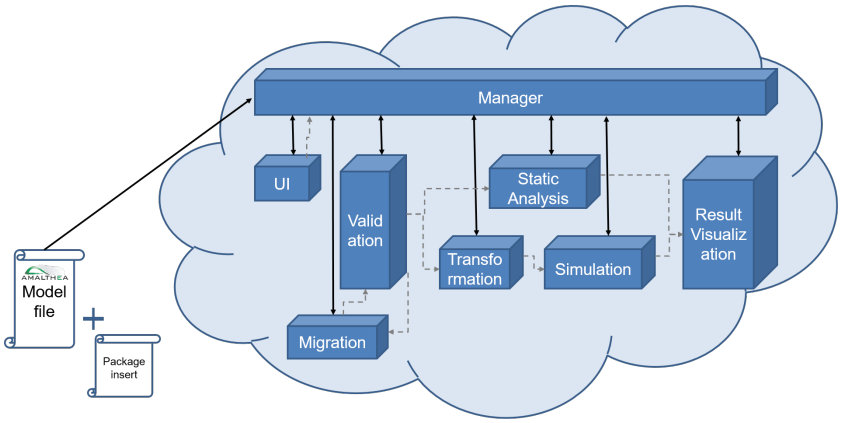


Figure 2.7: basic architecture of the APP4MC-cloud environment

To realize the environment we will provide services like simulation, migration, analysis and report visualization as an example for the possibilities we can achieve with this kind of environment. The considered services are containerized and can be combined via API to an entire workflow. If additional services are needed the user can enhance the system accordingly and enhance its possibilities. This is important as there might be specialized solutions for company specific needs.

The cloud solution therefore is capable to bring project specific parties together to work in the same structure. The input data is always based on the AMALTHEA data format to ensure a common data structure and definition as input. This is very important as the modeling infrastructure at company side is highly diverse.

2.3.2 Outcome

This approach can reduce individual tool-chain development, installation and maintenance cost. Also the collaboration between the parties is easily enabled and individual data transfer between the parties can be minimized. Results will be comparable and assessment can be made on common data. The data will be used only during processing and will be deleted afterwards. As the system has a services oriented structure individual services and tools can be connected according to needs. Also the collaboration during framework development is granted as individual parties can contribute their development in a containerized structure.

Boundaries on frameworks, programming languages are minimized due to the containerized service oriented structure.

2.3.3 Evaluation Criteria

The main evaluation criteria envisaged for this demonstrator, targets several aspects related to collaborative engineering of heterogeneous systems:

- A common framework for System Engineers to collaborate on complex and distributed projects
- Product design decisions based on performance needs
- Sharing of common taxonomies between teams, minimizing ambiguity
- Flexibility to enhance framework according to analyzes needs
- Joint data for system evaluation and assessment

2.4 Traceability Management Using Eclipse Capra for Safety and Security Assessments

Traceability between different development artifacts is an essential requirement in safety and security standards such as ISO 26262, ARP 4754, IEC 62304, IEC 62443, and the upcoming ISO/SAE 21434 standard. It ensures that safety requirements can be traced throughout the entire development lifecycle and that analysis results, e.g., generated with the tools provided by Eclipse APP4MC can be traced back to the relevant system properties. This is an essential step in the creation of safety assurance cases.

In the following, we provide background information about safety and security assessment before describing a concrete use case and how we address it in the context of PANORAMA. Three partners are involved in this use case: The University of Gothenburg, OFFIS, and Siemens.

2.4.1 Safety- and Security-critical Systems Are Complex, Multi-vendor Development Projects

Standards, such as those listed above, imply that:

- Requirements must be assigned to components from the functional architecture.
- Dependencies between requirements must be tracked.
- Safety- and security-relevant aspects must be tracked from hazard or threat identification through to implementation and verification of the safety or security mechanisms used.

To create the final product in safety- and security-critical domains, such as automotive, avionics, and medical devices, heterogeneous embedded systems are built by different suppliers and integrated by *original equipment manufacturers* (OEMs). The collaborative systems engineering processes necessary for such products create new traceability challenges because each participant in this complex value chain uses different processes, methods, and tools.

One recurring challenge is the need to connect artifacts from different tools, including commercial tools used for modeling, requirements management, and analysis, and the open source tools that complement these commercial tools. Current approaches are often based on manually created documents that are time-consuming and error-prone to maintain. In contrast, Eclipse Capra offers customizable and extensible traceability management right in the IDE.

2.4.2 All Safety-related Claims Must Be Proven with Evidence

When developing safety-critical systems, engineers must be able to demonstrate that all safety requirements have been fulfilled. That means a mitigation strategy must be defined for each identified hazard, or rather, for the safety requirements derived from it. A safety or assurance case demonstrates there is evidence for every safety-related claim and that the mitigation strategies are effective in addressing the hazard. The evidence can be process-related, demonstrating, for example, that code reviews ensured code quality. Alternatively, the evidence can be artifact-related, demonstrating that all safety requirements have been tested. Artifact-related evidence can be collected using trace links between development artifacts.

Similar considerations are necessary in systems that are vulnerable to cybersecurity attacks. For example, the IEC 62443 standard requires security design and implementation reviews, among other requirements. Upcoming standards, such as ISO/SAE 21434 for cybersecurity engineering in road vehicles, demand the creation of cybersecurity cases, which are similar in structure to safety assurance cases.

2.4.3 Trace Links Support Safety and Security Claims

Trace links strongly support the argumentation of safety and security claims during engineering reviews and assessments. A trace link connects two artifacts and denotes the relationship between them. Typical examples of trace links that are relevant for safety assurance cases are those between requirements and test cases. For example, developers might leave issue identifiers in the test code, or they might create trace links between code and the corresponding tests using a specialized tool for storage in a model or database.

In both cases, trace links should be accessible for analysis, or for visualization to show that all requirements are tested in a traceability matrix. Such matrices can be used in assurance cases as evidence that safety or security requirements have been validated. In addition, when analyzing failures or security incidents during operation, trace links support impact analysis, such as identification of affected system components.

2.4.4 An Automotive Driver Assistance System

To illustrate how the technology developed in PANORAMA can support the creation of safety and security assurance cases, we have developed an extensive dataset that serves as the basis of a use case. This dataset is based on an AMALTHEA model and its accompanying description that has been created for the WATERS FMTV Challenge 2019[1]. The AMALTHEA model describes a driver assistance system that provides lane following and waypoint navigation. It is based on a heterogeneous architecture, including CPUs and GPUs. The model defines several components and how they interact with each other. It is intended to calculate proper throttle,

[1] `https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/waters-industrial-challenge/index.html`

steering, and brake signals to follow a map of predetermined waypoints. In addition, a high-priority task to brake and perform emergency maneuvers to avoid obstacles is included. Details as well as a high-level overview of the system are described in the accompanying document.

Based on the assumptions of operation in urban areas and implementation of level 3 automation, we performed a *hazard analysis and risk assessment* (HARA) according to ISO 26262 to determine the corresponding automotive safety integrity levels. In addition, we performed an initial safety analysis that includes a *failure modes and effects analysis* (FMEA) based on the component model and the derived safety requirements. Overall, we supplemented the AMALTHEA model with the following additional artifacts to allow us to showcase safety and security assurance:

- We added functional and non-functional requirements with a focus on safety requirements. These requirements are stored in Microsoft Excel files.
- We added architectural models that refine the components from the AMALTHEA model. The architecture is defined using the *Unified Modeling Language* (UML) in Eclipse Papyrus.
- The outcome of the HARA is available as a Microsoft Excel file.
- The results of the FMEA are available both as a Microsoft Excel file and as an *Open Dependability Exchange* (ODE)[2] model.
- Based on an analysis of the involved artifacts, we also created a *traceability information model* (TIM), available as an Xcore file.

With this set of artifacts, it will be possible to construct a safety assurance case for the system and to perform model-based safety assessments. The TIM defines trace link types that allow connection claims about safety (i.e., safety requirements) to evidence and mitigation strategies defined in the AMALTHEA model and elsewhere. Construction of this safety case will be perfomed as the next step, using technology developed within PANORAMA. In particular, Eclipse Capra will be used to manage trace links between these different artifacts.

2.5 SAAB demonstrator ConCept

New methodology is needed for development of avionics systems to meet today's software explosion in complexity and related cost due to increased functionality in the aircraft. To meet future demands it is essential to find new methodology to be used at early concept evaluation stage for functional and component abstraction levels, based on rough estimation of overall needed performance in terms of desired aircraft functions and a rough estimation of available characteristics of the foreseen computation resources and network. This demonstrator intends to demonstrate use of new top down Correct by Construction (CbC) flow with focus on function mapping, addressing real-time performance, energy and safety requirements.

The main challenge is to find top level necessary and sufficient safety requirements that are useable design constraint inputs for the Design Space Exploration (DSE) method. Another challenge is to design methods that allows scalability of the problem over a larger network.

[2] `https://github.com/Digital-Dependability-Identities`

2.5.1 Characteristics

The demonstrator will provide a scenario for applying the methods and tools of PANORAMA, in accordance with the big picture in document D1.1 chapter 2.4, describing connection to ARP-4754 and ARP-4761 together with defined abstraction levels, see Figure 2.8 and abstraction levels description.

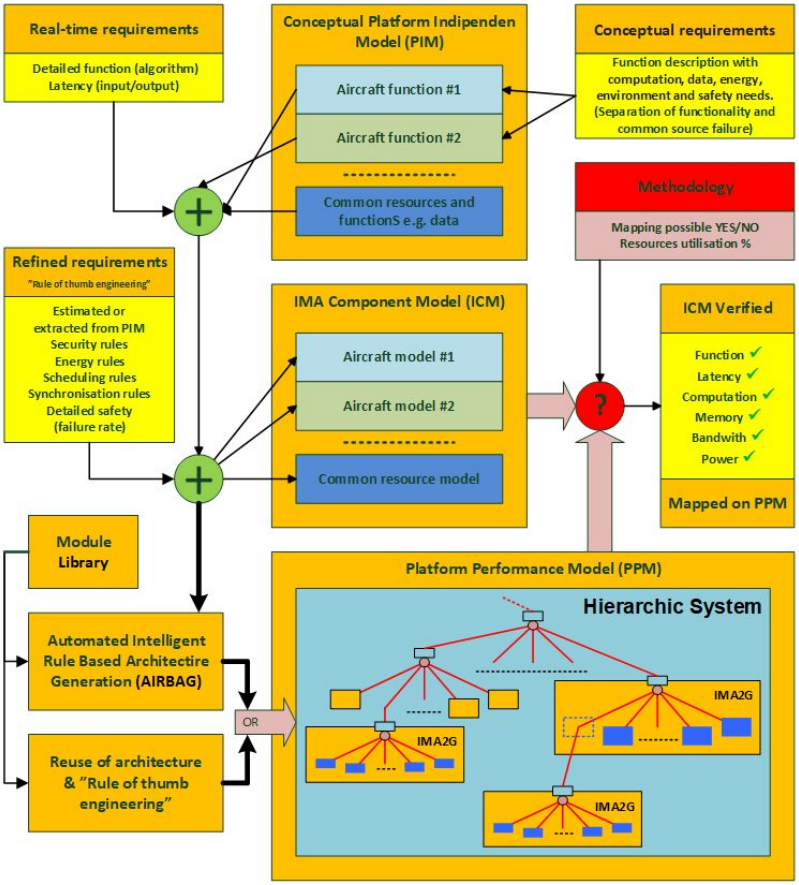


Figure 2.8: Early Concept Flow

Modelling views and abstraction levels

In order to facilitate the early system development at least three levels of abstraction with several modelling views are required. The levels of abstraction are the functional level, the component level and the resource allocation level.

To further clarify the abstraction levels a model with five abstraction levels is chosen which corresponds well with IMA terminology defined in RTCA/DO-297. The levels are functional level (as above), Function de-composed to Components level (part of component level above) and IMA system level (the resource allocation level as above). The additional two levels IMA platform level and Platform module level mirrors the functional view on top but for a platform perspective from bottom. The levels are further described below and visualized in Figure 2.9.

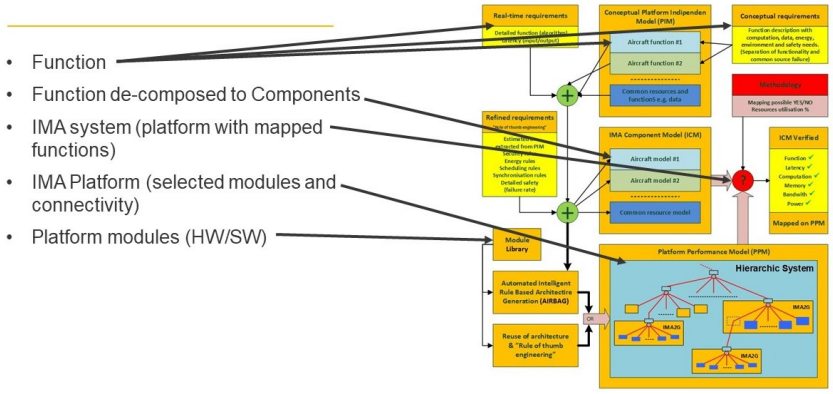


Figure 2.9: Refined abstraction levels

The abstraction layers gives an isolation between the functions models and platform modules. This facilitates the replacement of modules, e.g. due to obsolescence, without affecting the functions. It also means that function models can continuously be refined and mapped to a potential platform. The methodology support concept level work on function models, foreseen platform modules and continues refinement of the function modules and enhanced descriptions on platform modules.

The functions de-composed to components is given from top as input to DSE. At the same time the IMA platform is given from bottom as input to DSE. DSE shall find a mapping (the red dot) between these two layers as indicated in Figure 2.10.

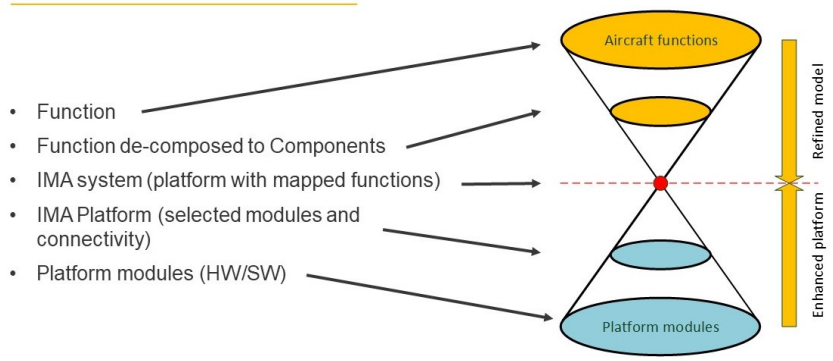


Figure 2.10: Hour glass illustration of aircraft functions mapped on Platform modules

The design space between the two layers means that DSE can find several possible solutions that fulfills functional and non-functional requirements as visualized in Figure 2.11. An optimal solution is not required from the DSE.

The functional abstraction level (Platform Independent Model (PIM))

Within the functional abstraction level, a functional view is required in order to describe all the Aircraft Functions, within this abstraction level a refinement from a textual description of the function to a service-oriented breakdown of the function can be described. Given the set of high-level and support functions to be realized in the system, they can be described in

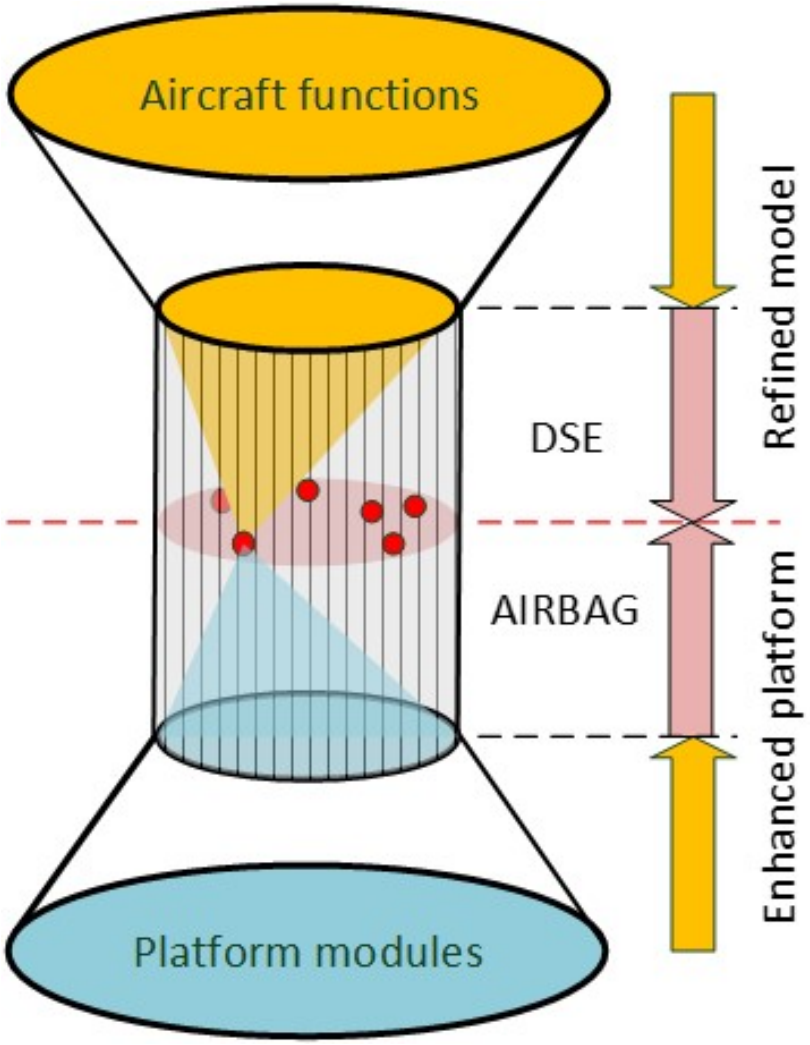


Figure 2.11: DSE available solution space

a connected dependency graph by provided and utilized services. In this level a set of high-level requirements, either functional, safety or security oriented, are associated with the set of functions. Functional requirements may be on performance, energy usage, response time, delay etc. Safety requirements may be on separation or absence of single-source failures. Security requirements may be identified at this level based on the security targets.

The component abstraction level (IMA Component Model (ICM))

The individual functions are further de-composed into functional components in a component view. The component view provides the capability of describing behavior using various models of computation. Within this view, the components can be refined, providing several potential different breakdowns of intended functions. The refinement into the component abstraction level will introduce additional requirements resulting from design decisions that are not directly traceable to higher-level requirements. These refined requirements that traditionally where decisions taken by an engineer is now expressed by rule based algorithm (describing acceptable

decisions), suitable as input to DSE to elaborate. E.g. are synchronization and energy rules.

The resource allocation abstraction level (red question mark, ICM mapped on Platform Performance Model (PPM))

This level has the capability to describe the applications (according to the [RTCA-DO-297] terminology), which means that they are described considering a specific target (CPU Core, programmable logic, DSP, GPU, or similar). The functional mapping on the Platform is fully described on this level with the needed set of module resources including the core software. In a resource allocation view, the IMA System is described with the instantiated functional components allocated to platform resources (allocated processing units, memory, time, communication slots and bandwidth, energy, etc.).

IMA Platform (selected modules and connectivity, Platform Performance Model (PPM))

This level of abstraction is the IMA Platform view (PPM). This view describes a network of selected modules and their connectivity through a communication network. This structure is given as input to DSE or might be, as an option, be enhanced by the AIRBAG.

Platform Modules (HW/SW components)

This level of abstraction defines a number of execution resource modules. The modules performance characteristics are given and stored in a module library from which the DSE can request characteristics, e.g. execution time for processing loads. As a minimum one module type must be characterized and stored in the library.

2.5.2 Early concept flow in relation to DSE

Now is focus on the challenge to find useable interfaces to DSE that maintain abstraction levels above without limiting the design space and support scalability. The approach is to explore the entire design space, guided by the algorithmic rules and bounded by top level requirements, both functional and non-functional (such as safety and energy). Figure 2.12 illustrates this from the DSE perspective, mainly at the component abstraction level above. The main objective for DSE is to find an instance of hierarchic system design that fulfills all requirements and rules. In future, it is possible to optimize between several instances that fulfill requirements by refinement of the rules.

The mapping between the Figure 2.8 and Figure 2.12 are as follows in Figure 2.13 (combined view of Early concept flow and KTH DSE).

This mapping has been further elaborated as shown in Figure 2.14.

2.5.3 Aircraft functions

The purpose of the aircraft functions in the demonstrator is to exemplify how safety requirements in term of separation and redundancy are handled by the methodology and development tools together with the functional requirements and performance requirements. The initial functional break-down is shown in Figure 2.15

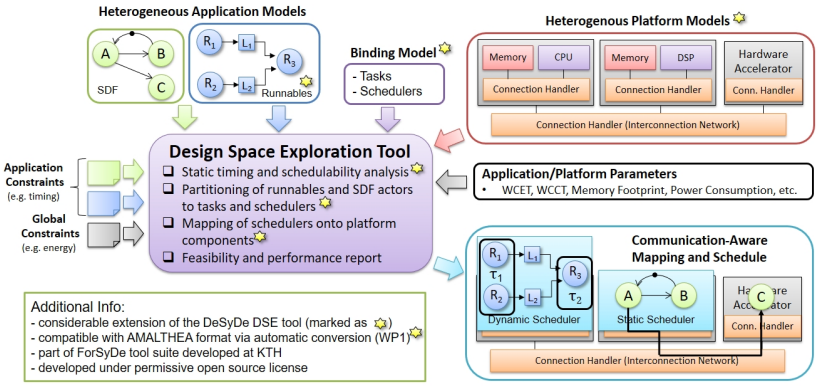


Figure 2.12: KTH view of DSE for heteroegeneous systems

Flight Information Function

The Flight Information Function is an important function that process flight data, e.g. speed, heading, altitude etc., and relay it to a display function. This function may have its own partition and be periodically scheduled on a core(s). The DSE tool will from the requirements and given rules realize the functions on the platform.

AESA Radar Function

This function represents an AESA radar sensor. It will have the capability of simultaneously handle multiple beams and multiple targets. The data to be processed comes from simulated AESA radar data to mimic a real AESA radar input signal. The radar function comprises of some or all of the following components:

- Digital Beamforming (DBF)
- Pulse Compression (PC)
- Doppler Filter Bank (DFB)
- Constant False Alarm Rate (CFAR)
- Integration

DBF filter takes the incoming radar pulse and maximize the signal to noise ratio. For a rectangular pulse, the DBF filter is just a simple pass band filter.

The PC function is to improve range resolution and increase signal to noise ratio. It picks out the part of the pulse that contains a potential target, i.e., it refines the DBF result by using binary phase code compression or linear frequency modulated compression.

The DFB filter function is used when targeting moving targets with a monopulse radar. Here it will be implemented as a fast fourier transformation (FFT) filter. This will reduce the sidelobe effects and discriminate against fixed and moving clutter.

The CFAR function is a technique used when the background is not constant, i.e. the background noise is varying, e.g. between a blue sky and rain cloud situation. Thus, the CFAR is needed in a real environment. CFAR uses adaptive threshold strategy to cope with

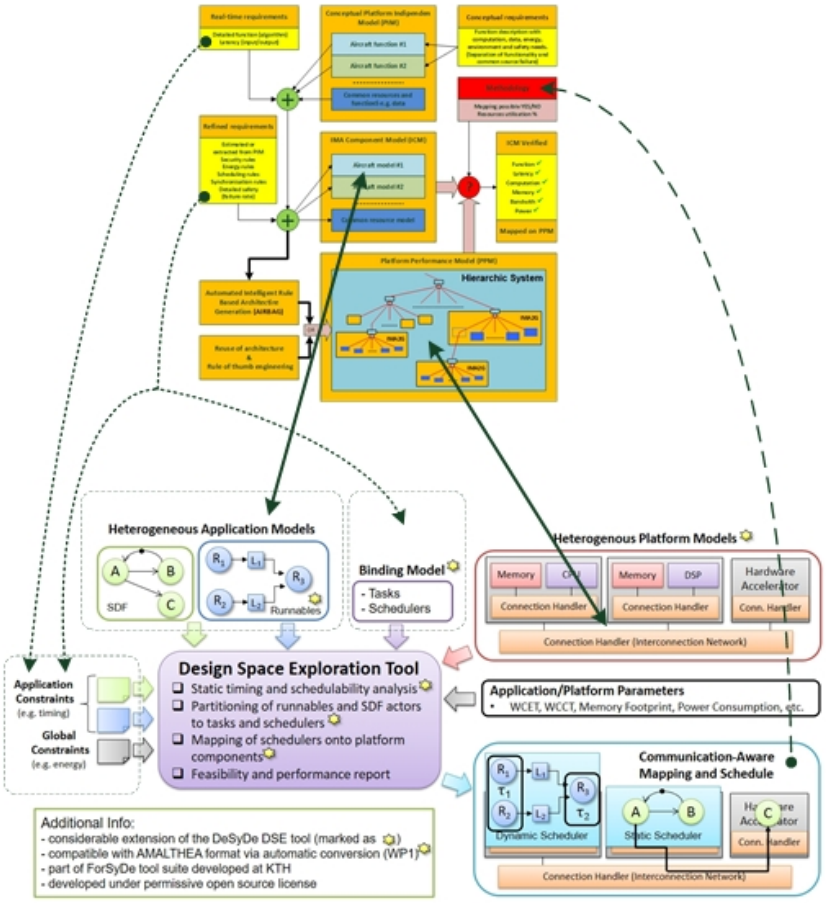


Figure 2.13: Relation between Early Concept Flow and DSE

these realistic situations. This further refines the target acquisition. It is a function of both the probability of detection and the probability of false alarm. The Integration component is used for increasing the noise to signal ratio by integrating two or more radar beam signals.

In Figure 2.16 the components are arranged in the order of execution. However, the CFAR and DFB is performed in parallel.

2.5.4 Hardware Modules, Processing module

One example of hardware module that will be used to demonstrate the design flow is the Zynq board XCZU5EV presented below.

Its characteristics will be stored in the Module library. Other types of modules may also be defined and stored in the Module library.

These hardware modules can be used as either processing modules or switching modules (see connectivity module further below). Each module consists of two PCB cards: a Motherboard and a Power board. The focus here is on the Motherboard. The Motherboard holds all the resources: process cores, memories and communication.

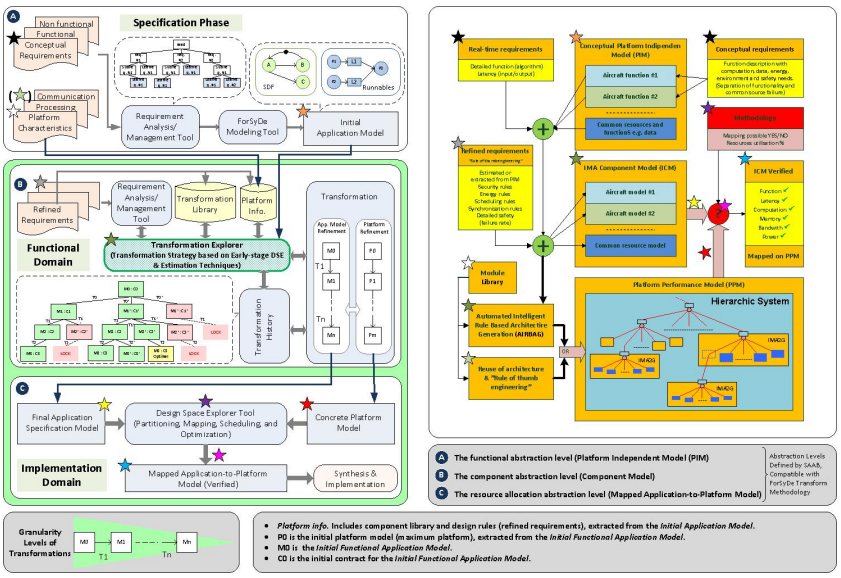


Figure 2.14: Enhanced relation between Early Concept Flow and DSE

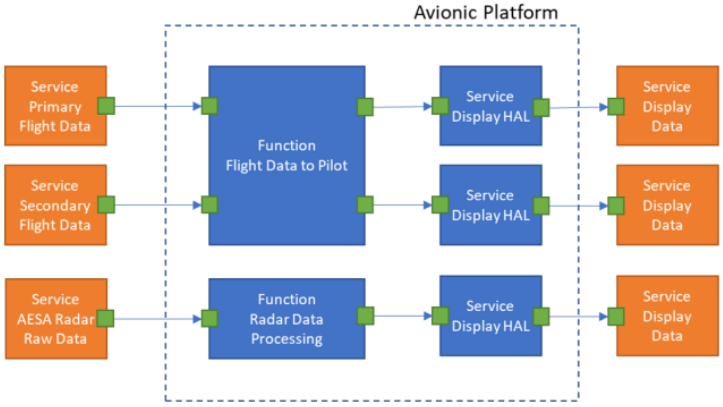


Figure 2.15: Functional break-down at functional abstraction level

Processor cores and their memories

The main component on the Mother board is a Xilinx Zynq Ultrascale+ MPSoC (XCZU5EV). It holds three CPUs, some memories and a programmable logic area. The processor cores available are a 64-bit quad-core ARM Cortex –A53, a dual-core ARM Cortex-R5 processor and a GPU ARM Mali-400 processor. The GPU is not to be used in the demonstrator. Thus, in total 6 cores, where each core has its own L1-cache for data and instructions of 32kbyte. In addition, the caches comes with ECC as well. The dual-core ARM cores also has an extra memory shared between them of 128kbyte, also with ECC.

ARM R5 Cortex performance

On these two cores, up to 16 ARINC653 partitions can be scheduled in which each partition can host a number of processes that are rate monotonic scheduled. The ARM Cortex-R family

Figure 2.16: The data flow chain

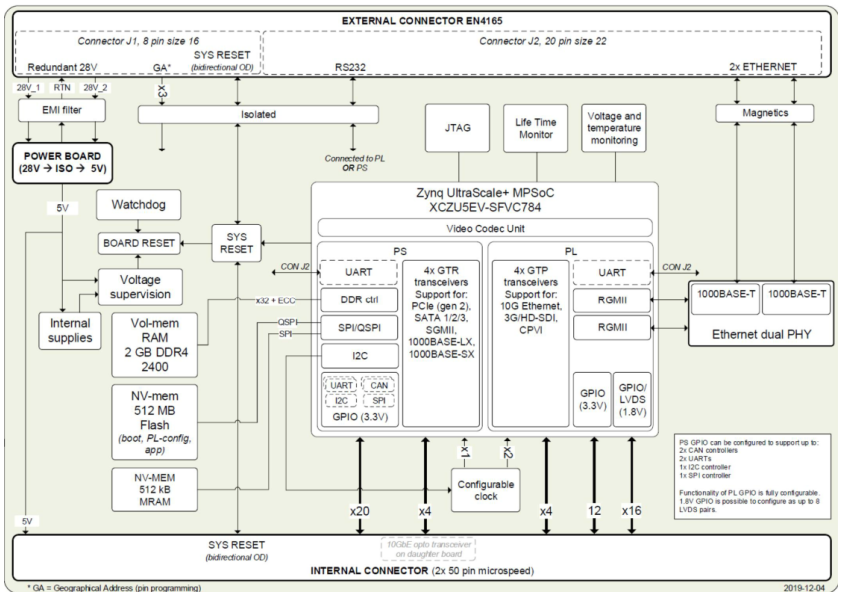


Figure 2.17: Block diagram of the Motherboard

is optimized for hard real-time and safety critical applications.

ARM A53 Cortex performance

On these four cores, up to 16 partitions can be scheduled in which each partition can host a number of processes that are rate monotonic scheduled. Alternatively, event triggered schedule where each hosted process is assigned a trigger. When multiple event triggered processes are available, they are executed in a round robin queue. The ARM Cortex-A family is designed for handling complex computational tasks, such as supporting multiple software components.

Available memories

The memories on the Mother board resides both on-chip of the MPSoC and outside the MPSoC.

On-Chip MPSoC memories

Besides the memories devoted for the cores four more memories exists within the MPSoC:

- 256KByte on-chip RAM (OCM) in PS with ECC
- 18Mbit on-chip RAM (UltraRAM) with ECC in PL
- 5.1Mbit on-chip RAM (block RAM) with ECC in PL
- 3.5Mbit on-chip RAM (distributed RAM) in PL

Memories available on Motherboard

Besides the memories on the MPSoC three extra memories exists:

- 2Gbyte (32-bit wide, data rate 2400MHz) Volatile DDR4 SRAM with ECC (for the PS-side).
- 512Mbyte Flash memory (for booting, applications and configurations for the PL-side)
- 512kbyte non-volatile MRAM (connected to the MPSoC (for the PS-side) via SPI)

Programmable Logic area

A large part of the MPSoC is an FPGA. Its configuration area may be used for the switching nodes primarily and very likely for the radar application as well. This resource can be used to accelerate specific functionality, as it normally outperforms processor-based implementations. It is therefore commonly used in radar applications to make real time data processing. This area consists of:

- 256200 logic cells
- 117120 Look-Up tables
- 234240 flip flops

System clocks

It is a fixed frequency running at 200MHz.

External communication interfaces

Two communication interface are built in for the user: Ethernet and RS-232. The Ethernet accesses the PL side of the Zynq MPSoC and the RS-232 may be connected to either the PL-side or the PS-side of the same. In order to provide deterministic traffic on the Ethernet network one may use:

- Bandwidth limiting solutions, e.g. AFDX
- TDMA (Time Division Multiple Access) base solutions, e.g. Time Triggered approaches

Built-in Monitoring

The Zynq MPSoC has monitoring feature, called SYSMON, infused in its hardware to monitor both temperature and supply voltage. This monitors both the PS-die and PL-die separately, which in addition enables the possibility to exercise power management.

Programming and Debugging

Programming and debugging is done via the JTAG ports. One of the two JTAG ports is for programming and debugging the MPSoC. The other JTAG port is for the reset function; reset PLD.

2.5.5 Hardware Modules, Connectivity module

The external communication uses any combination of accepted industrial standards to get the information across the hardware nodes.

The trend for Ethernet real time network is to move from proprietary standards to accepted industry standards, e.g. CAN, AFDX, 1553B, and ultimately into cross-domain standards. Thus, in the same network time triggered hard real time traffic, rate constrained traffic and best effort traffic may be intermingled.

In the demonstrator, e.g. a switched Ethernet network with one or several switches as connectivity modules, will be used.

The connectivity module detailed characteristics are defined and stored in the module library. Examples of such characteristics are:

- Its number of ports where other units (processing modules as above or other connectivity modules) can connect
- Max communication capacity for each port (bits/second)
- Data transfer delay time

2.5.6 Principle of Rules for DSE guidance

The justification of rules is to try to guide DSE in some direction to find solution. The rules shall in some way replace the design engineers manual work and judgement. Rules can be organized in different categories as:

- Safety rules, e.g. separation of functions in modules (space, time), on network etc
- Connectivity rules, e.g. each processing module must connect to one switch module
- Resource Utilization rules, e.g. max allowed execution and communication load
- Energy rules, e.g. max allowed energy usage

2.5.7 Characteristics to be addressed by DSE

The desired applications as output from DSE have both functional and non-functional characteristics, and focus varies from application to application. I.e. the final requirement for an application bound the characteristic to a specific value or allowed interval. These characteristics are often referred to as safety critical or mission critical functionality. All functions mapped to the platform gives the final IMA system. These together constitute a mix of safety and mission critical functions with concurrent execution. From these, there is both functional requirements such as response time, latency and processing capability as well as non-functional requirements safety requirements such as that no single failure shall lead to catastrophic failure conditions. The safety assessment have been addressed by the ARP-4761 and the outcome is expressed in the safety related refined requirements below.

Figure 2.18 illustrates a principal IMA platform with connected modules, Switch module, Core Processing Modules (CPM) and Remote Data Concentrators (RDC). Their characteristics are stored in the Module library. CPM and RDC are processing modules where the RDC has connection to the outer world (external IO). End-to-end means when the RDC to the left has received data from sensors to when RDC to the right has delivered data to actuators.

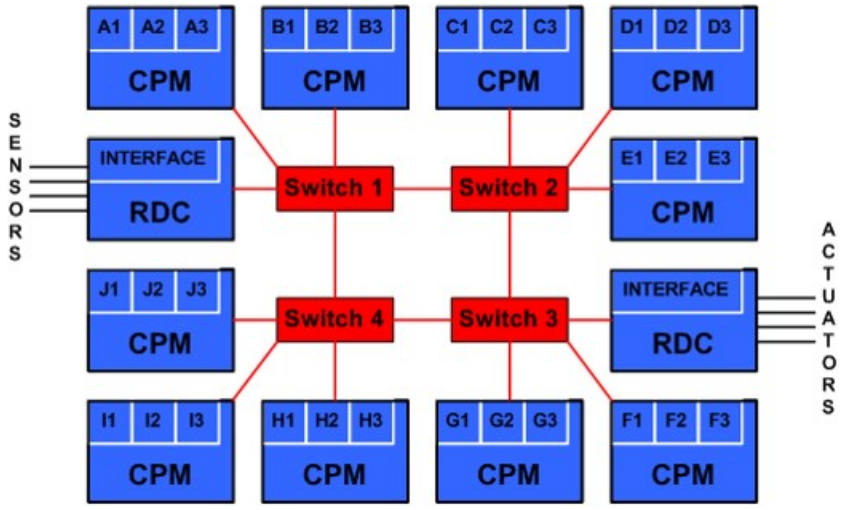


Figure 2.18: Data input and output to/from IMA Platform

Conceptual requirements

For each function:

- Algorithm size, Number of Operations
- In data size to algorithm, Bytes
- Out data size produced by algorithm, Bytes
- Energy budget for the function
- Safety requirements
- …

Real-time requirement

For each function:

- Periodicity
  - periodic need of algorithm computation with frequency
  - Aperiodic Event-trigged algorithm computation with intensity
- Latency (time from in data arrival (input from sensors into the receiving RDC) to needed out data delivery (output to actuators from the sending RDC))
- Jitter (accepted time windows for data arrival and data delivery)
- …

Refined requirements and application rules

For each Application model:

- Execution frequency, Hz (cyclic scheduling rule) given from abstraction level above
- Data flow or state machine oriented processing.
- Safety related requirement, separation (required separation of applications, degree of allowed disturbance between applications)
- Safety related requirement, redundancy (required redundancy between applications, degree of allowed sensitivity for single failures)
- Power peak restrictions (distribution of energy over time)
- Estimated number of operations needed for the function
- Estimated Program memory need , Bytes
- Estimated Internal data storage need, Bytes
- …

Platform Characteristics/Performance

For the platform, i.e. collection of Modules:

- Max performance per Core, operations per second
- Max communication volume per port, bytes per second
- Equation based execution time prediction based on number of operations needed
- Core Connectivity scheme (i.e. network architecture, possible data channels)
- Energy per operation
- …

2.5.8 Evaluation Criteria

The main evaluation criteria envisaged are:

- Dependability, functional requirements. The DSE tools ability to propose an explainable solution together with resource utilization within modules and architecture for functional correctness
- Dependability, non-functional requirements. The DSE tools ability to maintain and show non-functional correctness from conceptual level down to proposed platform mapping
- Evaluate rules as guidance mean for DSE to maintain non-functional requirements
- Scalability. Possibility for DSE to find solutions when size and number of function (conceptual requirements) are increasing by e.g. a factor 10

2.6 Enhanced Project Development Life-Cycle

As presented in Deliverable 7.1, Critical Software's goal with the PANORAMA project is to develop the capacity to employ Model Based System Engineering techniques in its diverse project portfolio.

Using a model-based approach, Critical Software expects to tackle one of the nowadays challenges of having multidisciplinary teams spread over different geolocations working on different components of, highly complex, safety critical projects.

This demonstrator main objective is to provide a reliable way of benchmarking and comparing the results of safety related activities with and without PANORAMA tools and methodologies. To achieve this, Safety Engineers will apply typical hazard analysis techniques to the two-case study projects, *UC-1-FPM-1* defined in work package 1, and the produced artefacts used to assess the safety relevance of the results obtained from the Tools and techniques developed in the PANORAMA project.

This demonstrator has a secondary objective of providing a way of validating the fault propagation path heuristics used to highlight how the relation between system sub-components (tasks, runnable's, etc.) can have an impact in the overall system safety and how the introduction of localized barriers at the model level, can improve the system safety.

2.6.1 Approach

To prevent a highly complex and difficult to assess case study, this demonstrator will follow an iterative approach, where, using the toolset developed in work package 2, the AMALTHEA system model complexity will be increased after each iteration, and the output artefacts compared with the ones obtained during the Safety Engineer activities.

This type of approach will provide a dynamic that is, not only, visually appealing but also easy to understand, as the complexity associated to simpler models, with less safety ramifications, is easier to grasp, even by non-technical personnel. After each iteration, safety analysis results, supported by the methods defined in work package 3, will be obtained, where both static and the dynamic analysis will be mixed to provide a reliable characterization of the system faults propagation and the overall system safety.

The output of each iteration can also be used to define and introduce barriers in the system model, whose ramification impacts can be assessed using the visualization tools developed in work package 4. Supported by these visualization tools, it will be possible to visually see the ramification impact a barrier has in the existing safety related issues, which will be represented by a decrease in the number of propagation paths, i.e. tree branches, associated to a specific failure. Nevertheless, this approach will prevent the complete characterization of the impact associated to the introduction of barriers in the system model, since a barrier in a less complex system model could have an exponentially higher impact on more complex system models. The introduction of barriers after each iteration, will be removed before the next iteration or order to prevent an unfair comparison between the artefacts produced by the tools and techniques developed during the PANORAMA project and the ones obtained by the safety engineers.

Figure 2.19 Illustrates how this demonstrator will use the tools and techniques developed in the PANORAMA project, and how this iterative approach follows Critical Software envisioned project development lifecycle based in MBSE, *D7.1, Safety Critical MBSE Project life cycle.*

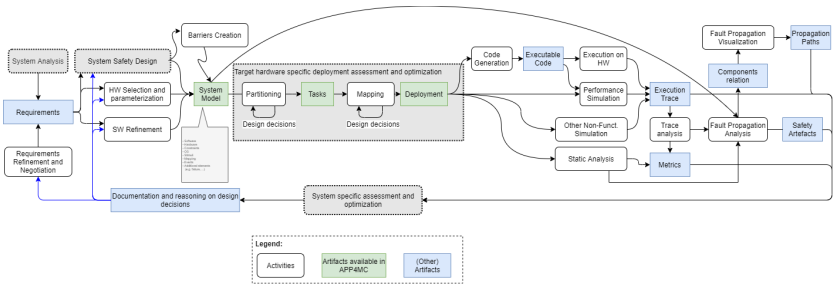


Figure 2.19: Demonstrator flow

2.6.2 Evaluation Criteria

The main evaluation criteria envisaged for this demonstrator, targets several aspects related to the engineering of heterogeneous systems supported by MBSE, namely:

- Methodology ability to impact project design decisions on the system's safety in all phases of the project development life cycle, by providing relevant system safety related information.
- Methodology allow for the re-use of previously designed models, and their safety analysis, in a modular way.
- Provide a common framework for System Engineers and Safety Engineers to collaborate on complex and distributed projects.
- Sharing of common taxonomies between teams, minimizing ambiguity.

2.7 Rover: Autonomous Driving System Development Platform

The main objective of the planned demonstrator as described in Deliverable D7.1 is to provide real-world application cases for the assessment, optimization and final validation of the model-based methodology developed in PANORAMA project in the context of Siemens' related modeling and design software tools. Since the majority of detailed requirements specified in Deliverable D7.1 has not changed, in this document we will mainly focus only on evaluation criteria and on relation of these with the defined use cases in the technological and methodological work packages 1, 2, 3, 4, and 6. The minor architectural changes associated with the need to better address AUTOSAR aspects as well as communication via ethernet are briefly described in subsection 2.7.1.

The high-level requirements can be briefly summarized again for the sake of clarity and ease of comprehension as follows:

- In order to be able to address the main engineering challenges related to the timing behavior in complex heterogeneous hardware/software systems, an exemplary development of a self-driving toy vehicle (a rover) shall be shown.
- As the development of modern complex systems is usually undertaken in the context of an embedded development process with multiple levels of modeling abstraction and various development stages, the demonstrator shall also cover the integration chain of

various design artifacts such as requirements, safety models, various types of architecture models, analysis results, and implementations.

- The demonstrated platform, on the one hand, must be challenging enough to be able to highlight main engineering problems associated with the complex timing behavior in heterogeneous hardware and heterogeneous functions, and on the other hand, it must comprehensively represent real-world application scenarios in a tangible way using the limited available resources in the project.

2.7.1 Heterogeneous HW/SW Platform

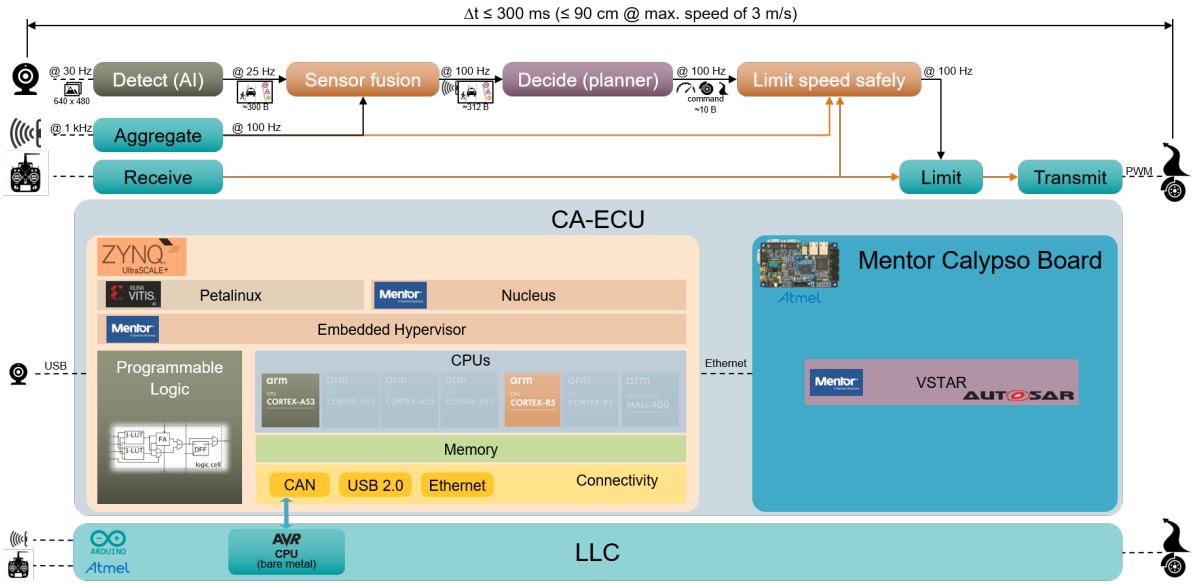


Figure 2.20: Changed heterogeneous hardware architecture now additionally addressing AUTOSAR and communication link via ethernet

The planned realization of the emulated collision avoidance ECU (CA-ECU) subsystem will provide multiple sources of sensed information sampled with various periodicities, that shall be merged by means of sensor fusion functionality to obtain a more confident representation model of the environment. This will also result in a wide variety of possible complex timing scenarios like overruns/underruns of real-time processes with multi rate sampling behavior that can be thoroughly investigated using the PANORAMA model-based timing analysis methodology. Furthermore, the hardware/software platform has been extended in order to be able to address important aspects of AUTOSAR software stack configuration as well as communication via ethernet. In addition to the hardware architecture version described in Deliverable D7.1, a Mentor Calypso board has been introduced for realization of decider (planner) functionality using Mentor VSTAR AUTOSAR compatible basic software stack as shown in Figure 2.20. This board is connected to the Xilinx board via ethernet. Aggregation of both boards now represents the CA-ECU subsystem of the rover. The hardware implementation of the CA-ECU implies integration and instrumentation of a mixed-criticality HW platform with different

levels of heterogeneity for embedded vision use cases (i.e., Xilinx Zynq UltraScale+ ZCU 104 Evaluation Kit board):

- a complex multi-clustered CPU architecture based on a quad-core ARM Cortex-A53 applications processor and a dual-core Cortex-R5 real-time processor,
- a tightly coupled programmable logic accelerator (FPGA),
- different memory architectures and cache structures realizing different partitioning/locking strategies, and
- virtualization features for multi-OS hypervisor-based cross-domain architectures.

This high degree of HW configurability allows us to realize different scenarios of software function allocation (distribution) and memory mapping. Currently we consider two such allocation and memory mapping scenarios that shall be evaluated using the demonstrator platform: The object detection functionality of CA-ECU can be realized either by means of a software using an ARM Cortex processor or as a dedicated hardware accelerator implemented in the FPGA. These allocation scenarios shall be accompanied by corresponding PANORAMA/AMALTHEA system design models in order to identify possible conceptual gaps in the underlying data model. The following aspects are planned to be evaluated on the heterogeneous HW/SW platform:

- End-to-end timing analysis based on corresponding PANORAMA/AMALTHEA models for different software allocation scenarios shall be compared to the generated trace data of the real implementation. In this case the timing analysis first will be done manually for a manageable functional subset of the CA-ECU subsystem and later on using behavioral simulation models for a larger functional scope. This evaluation is directly related to the use case *UC ID 1-03: Clear interpretation of semantics* in work package 1.
- On the hardware side, the scenario with involvement of FPGA for implementation of object detection functionality will consider a mapping of high-level functional and timing-related non-functional requirements onto a HW specification model using existing PANORAMA/AMALTHEA hardware modeling concepts. The evaluation should indicate, how well all of the required HW features, that are essential to perform end-to-end timing analysis, can be captured using the PANORAMA data model. This evaluation directly addresses the generic use cases *UC ID 1-2: Representation of HW/SW system design* and to some extent the use case *UC ID 1-1: Data-model has to be capable to represent legacy and new data* in work package 1. Furthermore, the following use cases of work package 4 are related to this experiment: *UC-4-ATB: Application Temporal Behavior*, *UC-4-ME: Mapping Evaluation*, and *UC-4-DA: Deployment Alternatives*.
- Implementation of the software first will be done manually and later on using a model-based approach, orchestrated by a software architecture model which will be imported/derived from a predefined validated PANORAMA/AMALTHEA system model. Both approaches can be compared in order to demonstrate correctness/completeness of both implementations and improvements in efficiency of software development. This evaluation is directly tangent to the use cases *UC ID 2-4: Data Export* in work package 2, to the use case *UC ID 1-04: Efficient collaboration* in work package 1, and to some extent to the use case *UC ID 1-03: Clear interpretation of semantics* in work package 1.

2.7.2 Collaborative Development with Traceability Workflow

Development of the demonstrator PANORover will be accompanied with requirements definition in the Polarion tool. The process would allow to present the possibilities of the Polarion system to specify the system and to identify the evaluation criteria for the developed demonstrator by creating a verification test document with corresponding traceability links. During the development process it is expected to look at the requirements on future collaborative processes (see "Collaboration Requirements" section in Deliverable 6.1) with the purpose to illustrate how Polarion tool together with the other Siemens tools would allow to achieve the specified requirements. Due to the ability of the Polarion tool to provide a full traceability among requirements, tests and external modelling tool elements, it is planned to dedicate a special attention to specifying timing requirements with their later evaluation.

For the efficient collaboration use-case *UC ID 1-04* it will be shown how AMALTHEA model can be imported into Capital Software Designer Tool. Requirements specified in REQ format or in Word Document can be imported into Polarion Tool. Since the Polarion tool allows to define traceability among system model elements defined within Siemens toolchain and requirement items, it is planned to specify evaluation criteria in Polarion and trace it to the implementation (as depicted on Figure 2.21).

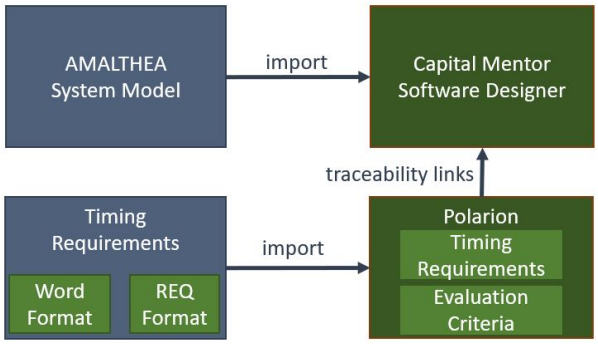


Figure 2.21: Evaluation aspect in collaborative development with traceability workflow

2.7.3 Safety Engineering

In order to perform the safety assurance in the PANORover demonstrator scenario, an initial setup of Hazards, safety goals, and safety requirements is defined in the tool Polarion. Based on the system and safety requirements defined in Polarion, a Capella model of the system architecture of the PANORover demonstrator is created. In order to analyze the system architecture in terms of safety and to check, if safety requirements are fulfilled by the proposed design, we enhance the Capella model with failure propagation in form of a Component Fault Trees (CFTs). Hence, qualitative and quantitative fault tree analyses can be performed.

As a next steps, safety analysis will be conducted on the refined of the system design in a detailed design, which is composed of detailed models for hardware and software sub-systems. Thereby, we plan to evaluated how the safety assessment on different levels of abstracts supports the evaluation of different designs alternatives and how a traceability of the various artifacts related to the safety engineering life-cycle can be established using the tools Polarion, Capella, etc. Moreover, we will compare the results form the PANORover demonstrator with the gap

analysis provided in Deliverable D6.1 to check for the use case which of the identified can be solved with the results of PANORAMA. Moreover, this use case is related to *UC-4-SA: Safety Analysis* described in work package 4.

2.8 RTFParallella

2.8.1 Requirements

In this section, we highlight the high level requirements of the demonstrator corresponding to the description provided in the previous deliverable (D7.1).

The RTFParallella framework will allow for the implementation of amalthea models on the Adapteva Parallella hardware platform. Additionally, this demonstrator will produce a BTF trace of the execution of the implemented model.

The framework will demonstrate Inter-process communication according to the *implicit communication* and *Logical Execution Time* (LET) semantics.

The framework will be designed based on the OSEK task model. Additionally, tasks will be scheduled according to a fixed priority scheduling strategy. Subsequently, the framework will be based on FreeRTOS.

Hardware related delays such as the effect of contention in shared and distributed-shared memory on event chain latencies will be demonstrated.

To allow the implementation of inter-process behavior discussed above on the underlying hardware, the shared labels used by tasks will be assigned to the shared or distributed-shared memory as *sections*. Each section will contain shared labels of similar size.

The execution of the implemented model in RTFParallella will be traced in BTF format. Task, runnable, and core events will be initially traced.

2.8.2 Evaluation

RTFParallella will be evaluated based on the ability to implement complicated models, its tracing capability, and the tracing overhead incurred and its effect on the tracing results in contrast to the real execution of the implemented model.

These criteria will be evaluated during the implementation time of the demonstrator.

2.8.3 Relationship to other work packages

This demonstrator will be used to support efforts in Work packages 3 and 4, where static analysis results could be compared with the execution the corresponding model and the hardware, which will allow for an estimate of the pessimism introduced by static analysis. Furthermore, the demonstrator could be used in combination with Eclipse Trace Compass to verify results obtained by visualization tools which are part of work package 4.

Furthermore, efforts in work package 2 aim at generating deployment-ready implementations of Amalthea models to run on the Adapteva Parallella hardware platform based on this framework.

2.8.4 Main challenges

The main challenge associated with building and maintaining this demonstrator is the lack of official support, since this hardware platform is not in production anymore. Tool support for

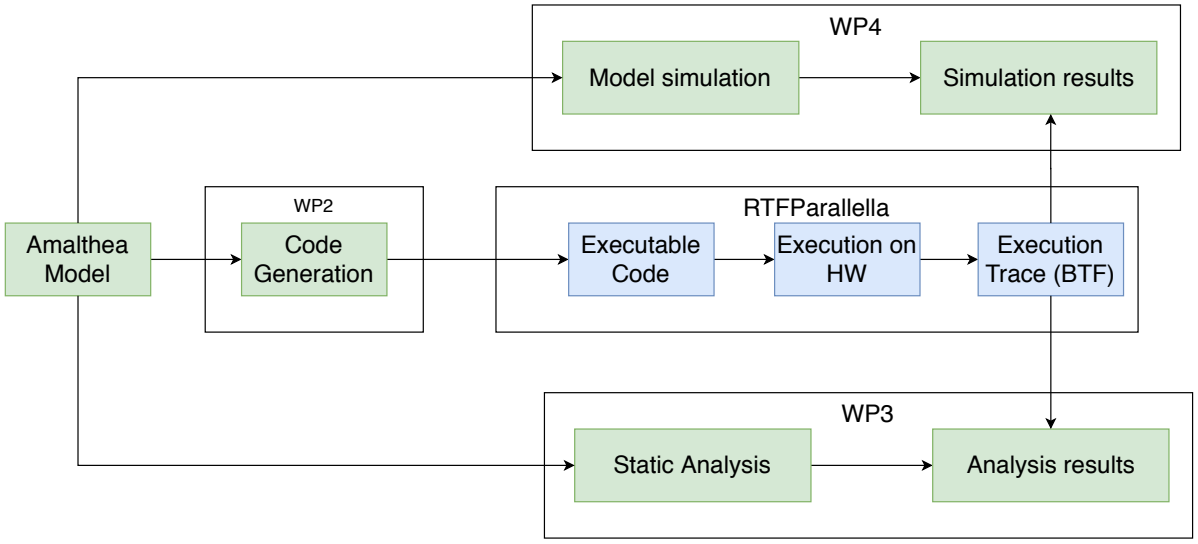


Figure 2.22: RTFParallella in connection to other work packages in Panorama

this platform is only available on community forums and other unofficial sources. The tooling contains bugs that slow down the development process with respect to certain aspects such as memory management, workarounds specific for this demonstrator have been created to allow for the simple implementation of Amalthea models. The platform is not officially supported by any real-time systems vendor. Although it is possible to use it for the purpose of demonstrating certain timing behavior aspects of a system, it is not possible to use this platform for realistic implementation of a commercial embedded system.

2.9 RaceCar

Advanced Driver-Assistance System (ADAS) applications deployed on vehicles are becoming increasingly complex. These applications are computationally intensive and require high performance hardware. The variation of computational demand by these applications requires a variety of hardware such as multi-core processors and hardware accelerators. Most state-of-the-art ADAS applications consist of computationally heavy image processing algorithms. Therefore, finding the most suitable heterogeneous System on Chip (SoC) for a given application is not trivial. Due to the complexity of heterogeneous systems, it is difficult to predict the execution time of an algorithm in a real time system.

In order to address these challenges, a demonstrator is designed based on a 1/10 scale RC car. This demonstrator will provide a platform for applying the methods and tools developed in PANORAMA to measure the end-to-end latency of an ADAS application on a heterogeneous system in a real life environment.

2.9.1 Requirements

The RaceCar demonstrator is comprised of a 1/10 scale RC car that is controlled by an ADAS application. The application is designed for the use case of autonomous racing. To address this use case, the demonstrator is designed using tools and methodologies developed in the context of PANORAMA. In addition, RaceCar will help verify the results of the analysis of a given real-time ADAS application by running allowing this application to run on a real vehicle in a high performance environment such as racing. In particular, this demonstrator will allow for the following:

- Measuring the end-to-end response time of the implemented ADAS application.
- Generating a BTF trace of the implemented ADAS application. The generated trace will be analyzed to determine whether any deadline misses exist and measure the response time of each task.
- Measuring memory access latency, memory contention, and communication interference.
- Tracing the offloading of computational loads from processing units to hardware accelerators.

The ADAS application implemented in this demonstrator should be able to identify objects, obstacles and keep the RC car within lane boundaries.

2.9.2 Functional Model

The RaceCar's functional model is based on the application introduced in the WATERS Challenge 2019 [HDW19]. The application consists of multiple dependent tasks being executed on different processor cores. The tasks co-ordinate with each other via shared memory and external communication interfaces.

The data flow for the ADAS application begins with grabbing the LIDAR and Camera image data which are processed by subsequent tasks deployed to different cores and accelerators. The Localization algorithm determines the position of the demonstrator which is fed to the planner to define and follow a trajectory based on depth estimation and lane boundaries. The CAN bus polling is used to provide the current vehicle status which include speed, steering angle, acceleration. The DASM task computes and establishes the speed and steering angle to keep the trajectory based on the information provided by the Planner task.

2.9.3 Hardware Modules

The emphasis is to develop a demonstrator on a heterogeneous platform. Making it fully autonomous requires multiple sensor technologies and different sets of processing units. The most important parts of the demonstrator are the processing cores which run the operating system which will in turn be running all the software necessary for the end-to-end ADAS application. The remainder of this section briefly describes the major components used in this demonstrator.

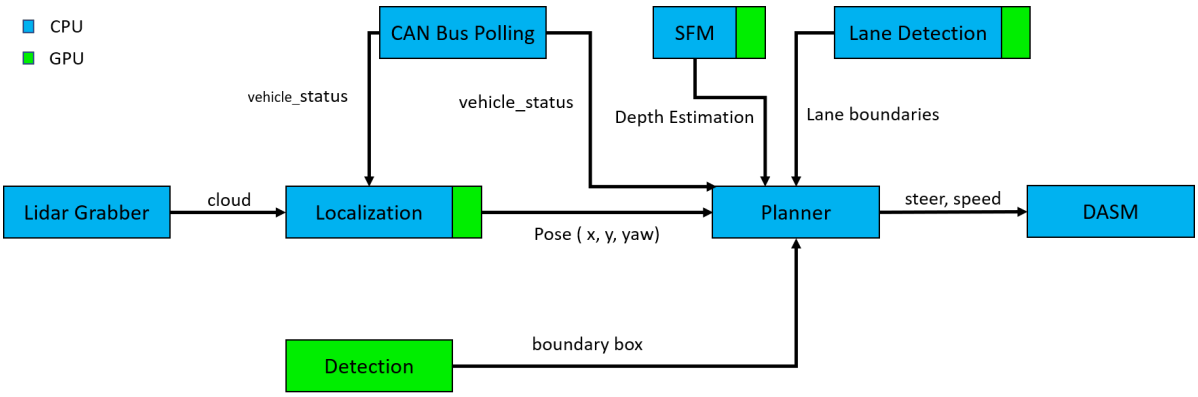


Figure 2.23: The RaceCar Functional Model

Lidar Sensor

The RPLIDAR A3M1 [Sla19] is a 360 degree 2D laser scanner which can take up to 16000 samples of laser range data per second with high rotation speed. The lidar's maximum range is 25 meters. The generated 2D point cloud data can be used to perform mapping, localization and object modeling.

Camera

The ZED2 [Inc19] stereo camera provides high definition 3D videos and 2D images that can be used for neural depth perception of the environment. The camera image will be used for depth estimation, lane boundary detection and object detection and classification. It also features an integrated 6 axis IMU sensor which will help in accurate estimation of acceleration and orientation of the demonstrator.

Primary Processing Unit

The NVIDIA Jetson AGX Xavier [Fra18] is a high performance System-on-Module featuring Carmel CPU based on ARMv8.2 (64 bit) architecture. The CPU complex (CCPLEX) is comprised of Carmel dual-core CPU clusters in a coherent multi-processor configuration. Each CPU cluster contains two identical Carmel processors; each core includes 128 KB Instruction (I-cache) and 64 KB Data (Dcache)Level 1 caches, a 2 MB L2 cache is shared by both cores. The accelerator is comprised of 512-core Volta GPU with 64 Tensor cores and dual Deep Learning Accelerator(DLA) Engines.

Secondary Processing Unit

Beaglebone AI [Kri20] will be used as a secondary processing unit. It features a dual ARM cortex A15 microprocessor and 2 dual ARM Cortex M4 co-processors. Additionally, the computation power is enhanced with the help of a dual core Programmable Real-Time Unit, graphics accelerator and dual core PowerVR 3D GPU.

Electronic Speed Controller

The Vedder Electronic Speed Controller [Boa20] is an advanced speed controller which provides an interface to control servo and DC motors. A Single Board Computer (SBC) such as Raspberry Pi, Beaglebone AI can be easily integrated to control the movement of the demonstrator. It also supports multiple interfaces such as CAN, I2C, SPI, UART and USB. In terms of memory, it includes 1GB RAM and 2.5MB of on-chip L3 Cache.

2.9.4 System Architecture

This section describes the preliminary architecture of the demonstrator. The application consists of multiple dependent tasks hosted on the processor cores with fixed priority. Tasks with high computational demand will be implemented such that it could be accelerated using the GPU.

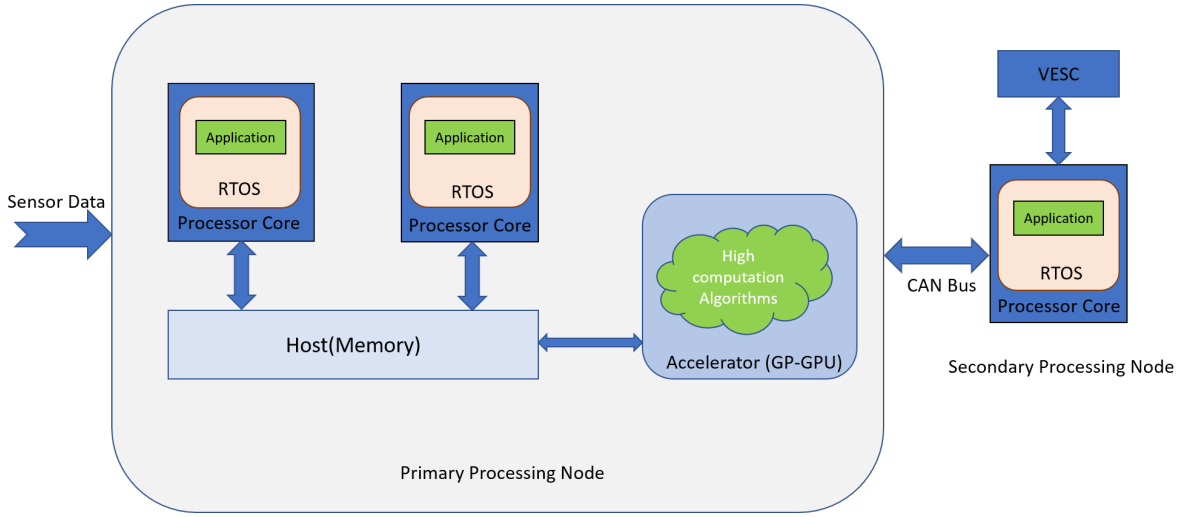


Figure 2.24: RaceCar System Architecture Design

The overall architecture consists of a primary and secondary processing units. The former is responsible for gathering the data from the sensors and processing them using the hardware accelerator, and therefore determining the control action required. The required control action is then sent to the secondary ECU which controls the RaceCar through the VESC.

2.9.5 Relationship with other work packages

The relationship of this demonstrator with other work packages is similar to that of RTFParallella as depicted in Figure 2.25. The demonstrator will aim to measure the end-to-end latency of the overall application as well as individual tasks.

The Amalthea model will be used to generate the code structure that could be used to verify the applications deployment with respect to timing requirements for the main processing unit as part of work package 2. At the same time, visualizations of the application's execution developed in work package 4 can be used to compare the simulation results with the generated

trace of the application's execution on the demonstrator. The results of this demonstrator can also be combined with the static analysis results of work package 3 for comparison if measured timing properties to the analysis results. The generated trace can also be used to validate against the requirements developed in work package 6.

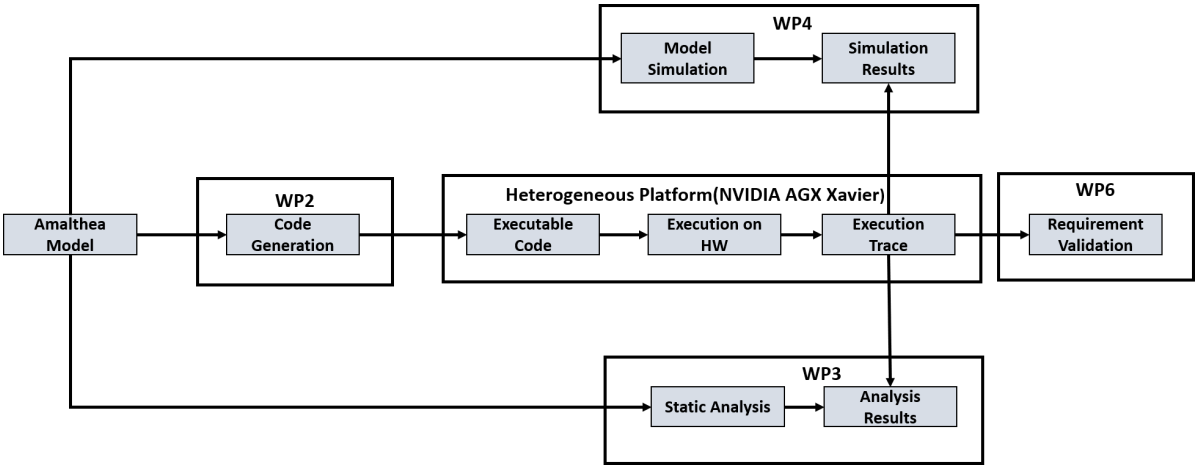


Figure 2.25: RaceCar in connection to other work packages in Panorama

2.9.6 Evaluation Criteria

The evaluation criteria for this demonstrator is based on the the implementation of an ADAS application modeled with Amalthea on a heterogeneous hardware platform. The demonstrator will be evaluated based on the ability to meet the functional and non-functional requirements of the ADAS application while validating the usability and correctness of the methodologies and tools developed in panorama. In particular, the evaluation of this demonstrator will mainly depend on the following factors,

- The demonstrator's validation of functional requirements such as being able to drive autonomously along a pre-defined course while avoiding obstacles and reacting to its physical environment in real time.
- The demonstrator validation of non-functional requirements. In particular, the adherence to timing requirements such as the end-to-end reaction latency of the ADAS application implemented in this demonstrator.
- The conformance to design and deployment decisions embedded in the Amalthea model and then used by the tools and methodologies developed in Panorama to implement the system.

2.9.7 Main Challenges

The main challenges in developing this demonstrator is related to lack of official support for sensors used. Since it is a heterogeneous platform with multiple sensor technologies and development boards from different vendors, smooth integration of each component of the demonstrator

is not trivial. The demonstrator is required to react the its physical environment in real time and therefore it is a *real-time application* which requires RTOS (Real Time Operating System) installed on each processor cores. The community support for a RTOS on both the primary and secondary processing units is limited.

2.10 FiSimSo

In the real-time embedded systems, task scheduling perched on a high place in the hierarchy of design and analysis. Various analysis tools has been developed for the purpose of assessing the competence of a system to fulfill the tasks timing constraints. The supreme goal of real-time scheduling analyses tools is compute if the system is schedulable or not. One method of checking the compliance of a systems to meet the scheduling constraints is simulating the run-time execution of the various tasks. The advantage of simulation is providing detailed information about the execution of tasks beside the original goal of examining the schedulability of a system. The various simulation tools does not take into account the faulty software/hardware of a system. These faults can not only be due a blemish in the design/realization of software/hardware, we can also result of unexpected occurrence of a high priority event, or due to bit(s) flip in memory and registers.

In D7.1, the description and specifications of *PyFI* fault injection tools was introduced. *FiSimSo*, which stands for *Robust SimSo* is a planned extension of the open source tools *SimSo* [CHD14]. *SimSo* is a scheduling simulator for real-time multiprocessor architectures that takes into account some scheduling overheads (scheduling decisions, context switches) and the impact of caches through statistical models. Based on a Discrete-Event Simulator (SimPy), it allows quick simulations and a fast prototyping of scheduling policies using Python [CHD14]. (see Figure 2.26)

FiSimSo is a robustness analysis tool that combines the aptitude of *PyFI* fault injection with *SimSo* tasks-scheduling simulation tool. This can be achieved by introducing *deliberated* randomness in tasks parameters, then a deeper insight can be gained bout the behavior of real-time system under test. FiSimSo is a probabilistic analyses of real-time systems the provides levels of confidence on the response time distributions, helping to evaluate the robustness of the real-time system and as the result comprehensive understanding of the safety of using the system.

2.10.1 Evaluation

The evaluation of *FiSimSo* will depend on the ability of utilizing the tool for scheduling lengthy and complex tasks. Beside examining the scalability of the simulation tools it is vital to test the added overhead for the probabilistic analysis and the inference of robustness of the targeted system.

Another aspect of assessment, is the impeccable robustness analysis for various types of injected faults. The intended space of examination would include testing the behavior of the system for different faults manifestations. These include:

- Faults **persistence**

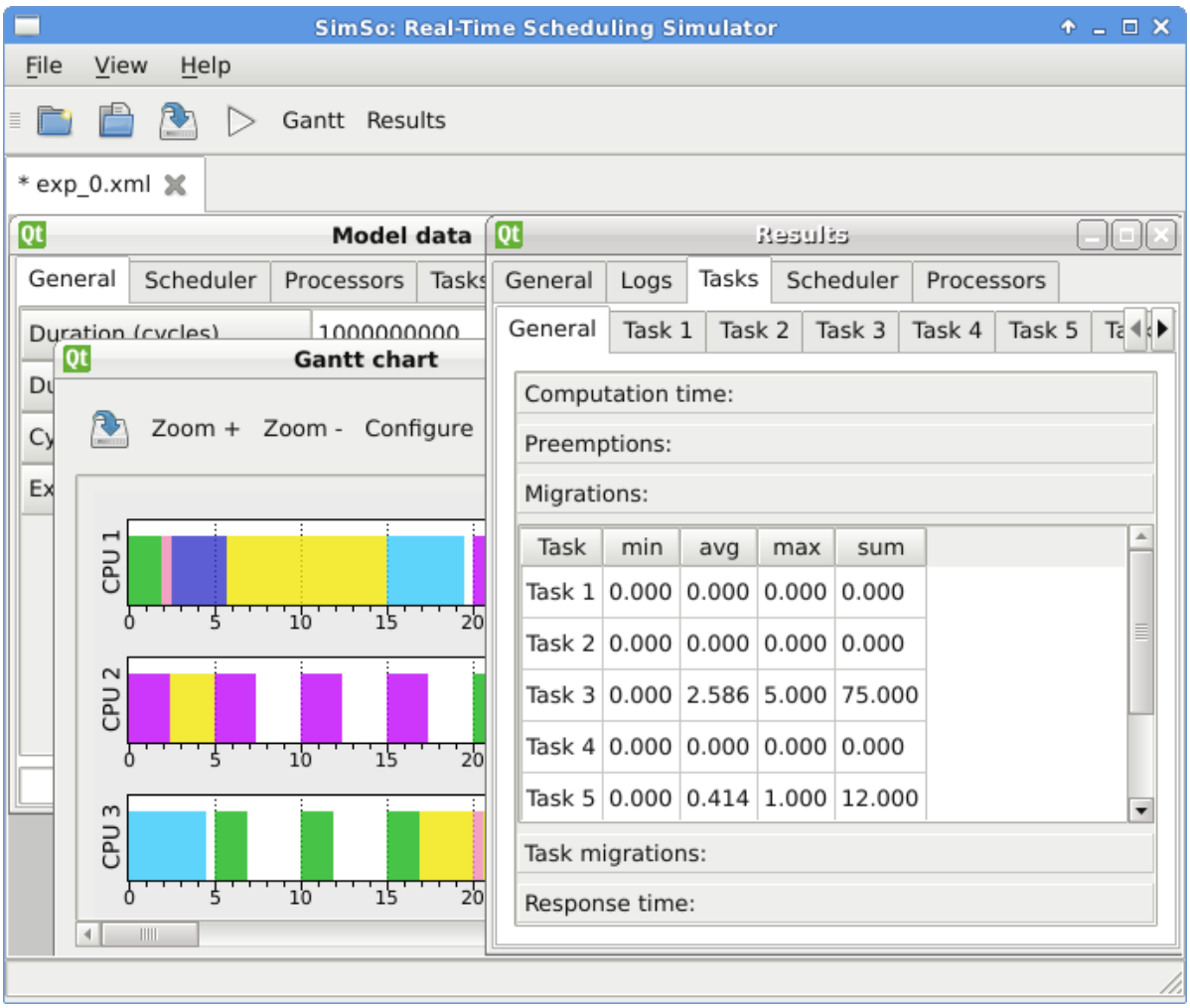


Figure 2.26: *SimSo* Simulation GUI [Ché14]

  - Permanent faults
  - Transient faults

- Faults **plurality**
  - Single faults
  - Multiple faults

- Faults **correlation**
  - Independent faults
  - Related faults

2.10.2 Challenges

The main challenges associated with *FiSimSo* development is the terminated support of *SimSo* tools and as a result the outdated python libraries, such as *SimPy* and *PyQt*. This enforce not only the extension of probabilistic analysis as and inference tools, but also the update of outdated libraries and python-language syntax (i.e. updating from python 2.x to 3.x).

3 Conclusion and outlook

Within this document we described the general purpose of the deliverable and provided a set of updated requirements and evaluation criteria. We also provided an update of planned demonstrator use cases with refined / extended features of respective proof-of-concept platforms.

Generally, the allocation of described evaluation criteria to individual work packages can be summarized as follows:

- **Work packages 1 and 2 (model data, abstraction levels, editors, importers, viewers):**
  Dortmund/Rostock/INCHRON/Bosch/Vector/KTH Collaborative Cloud-based Service Environment (common data structure for system evaluation and assessment, migration service, trace specifications in BTF), CSW Enhanced Project Development Life-Cycle (UC-1-FPM-1, failure propagation modeling), Siemens Rover: Autonomous Driving System Development Platform (UC ID 1-1, UC ID 1-2, UC ID 1-03, UC ID 1-04, UC ID 2-4), Dortmund RTFParallella and RaceCar (generating deployment-ready implementations of Amalthea models),

- **Work package 3 (model analysis):**
  Dortmund/Rostock/INCHRON/Bosch/Vector/KTH Collaborative Cloud-based Service Environment (simulation and DSE service supporting product design decisions based on performance needs), SAAB demonstrator ConCept (ForSyDe DSE), CSW Enhanced Project Development Life-Cycle (static fault propagation analysis), Siemens Rover: Autonomous Driving System Development Platform (static fault tree analysis), Dortmund RTFParallella and RaceCar (comparison of measured timing properties to the analysis results, evaluation of static analysis results and estimation of the introduced pessimism by them), OTH-R FiSimSo (schedulability/robustness analysis based on DES simulation, safety patterns and fault injection),

- **Work package 4 (result visualization and assessment):**
  Dortmund/Rostock/INCHRON/Bosch/Vector/KTH Collaborative Cloud-based Service Environment (analysis and report visualization, flexibility for extension), CSW Enhanced Project Development Life-Cycle (visualization of fault propagation analysis), Siemens Rover: Autonomous Driving System Development Platform (UC-4-SA, safety analysis), Dortmund RTFParallella (Amalthea model simulation and visualization of simulation results, Eclipse Trace Compass, verification of results obtained by visualization tools), and

- **Work package 6 (design flow and traceability):**
  AVL/Mantis/UNIT ADAS Feature Simulator (requirements traceability aspects, Eclipse Capra), Dortmund/Rostock/INCHRON/Bosch/Vector/KTH Collaborative Cloud-based Service Environment (a common cloud-based framework for collaborative systems engineering, sharing of common taxonomies between teams), GU/OFFIS/Siemens Traceability with Eclipse Capra (change impact and safety analyses), SAAB demonstrator

ConCept (correct by construction flow with focus on function mapping, real-time performance, energy and safety requirements as inputs for DSE), CSW Enhanced Project Development Life-Cycle (MBSE, safety analysis), Siemens Rover: Autonomous Driving System Development Platform (collaborative development with traceability workflow, safety engineering/assessments), Dortmund RaceCar (validation of requirements by generated traces).

Future work will mainly focus on implementation of demonstrators and generating a direct feedback to methodological/technological work packages based on the demonstrator results and their assessments. Further detailed analysis of the requirements and evaluation criteria will be conducted. These aspects should then be addressed in the baseline version of deliverable D7.3 "Detailed reports about demonstrators and assessment results" as well as in the improvement phase of demonstrator implementation.

Bibliography

[BJS+15] C. Bergenhem, R. Johansson, A. Söderberg, J. Nilsson, J. Tryggvesson, M. Törngren, and S. Ursing, "How to reach complete safety requirement refinement for autonomous vehicles," 2015.

[Boa20] T. Boards, *VESC 6 MKIV Speed Controller*, 2020. [Online]. Available: `https://trampaboards.com/1-x-vesc-6-mkv--170-tax-each-p-27529.html` (visited on 07/10/2020).

[BOS16] J. Bach, S. Otten, and E. Sax, "Model based scenario specification for development and test of automated driving functions," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2016, pp. 1149–1155.

[BRSM16] G. Bagschik, A. Reschka, T. Stolte, and M. Maurer, "Identification of potential hazardous events for an unmanned protective vehicle," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2016, pp. 691–697.

[CHD14] M. Chéramy, P.-E. Hladik, and A.-M. Déplanche, "Simso: A simulation tool to evaluate real-time multiprocessor scheduling algorithms," 2014.

[Ché14] M. Chéramy, *SimSo - Simulation of Multiprocessor Scheduling with Overheads*, `http://projects.laas.fr/simso/`, 2014.

[EGG+17] F. Erata, C. Gardent, B. Gyawali, A. Shimorina, Y. Lussaud, B. Tekinerdogan, G. Kardas, and A. Monceaux, "Modelwriter: Text and model-synchronized document engineering platform," in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2017, pp. 907–912. DOI: `10.1109/ASE.2017.8115703`.

[EGTK17] F. Erata, A. Goknil, B. Tekinerdogan, and G. Kardas, "A tool for automated reasoning about traces based on configurable formal semantics," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2017, Paderborn, Germany: Association for Computing Machinery, 2017, pp. 959–963, ISBN: 9781450351058. DOI: `10.1145/3106237.3122825`. [Online]. Available: `https://doi.org/10.1145/3106237.3122825`.

[Fra18] D. Franklin, *NVIDIA Jetson AGX Xavier Delivers 32 TeraOps for New Era of AI in Robotics*, 2018. [Online]. Available: `https://developer.nvidia.com/blog/nvidia-jetson-agx-xavier-32-teraops-ai-robotics/?ncid=so-fac-mdjngxxrmllhml-69163`.

[HDW19] A. Hamann, D. Dasari, and F. Wurst. (2019). Waters industrial challenge 2019. Accessed 2020-07-30, [Online]. Available: `https://www.ecrts.org/forum/viewtopic.php?t=124` (visited on 07/30/2020).

[Inc19] S. Inc, *ZED 2 Camera*, 2019. [Online]. Available: `https://www.stereolabs.com/zed-2/%20https://cdn.stereolabs.com/assets/datasheets/zed2-camera-datasheet.pdf`.

[Kri20] J. Kridner, *BeagleBone AI System Reference Manual*, 2020. [Online]. Available: `https://github.com/beagleboard/beaglebone-ai/wiki/System-Reference-Manual` (visited on 07/21/2020).

[MBI+19] T. Menzel, G. Bagschik, L. Isensee, A. Schomburg, and M. Maurer, "From functional to logical scenarios: Detailing a keyword-based scenario description for execution in a simulation environment," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2019, pp. 2383–2390.

[MBM18] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1821–1827.

[Sla19] Slamtec, *RPLIDAR A3M1 Datasheet*, 2019. [Online]. Available: `http://bucket.download.slamtec.com/e6f98338018505f6c510e83f78f03b6abb35454b/LD310_SLAMTEC_rplidar_datasheet_A3M1_v1.7_en.pdf`.

[SSL+13] F. Schuldt, F. Saust, B. Lichte, M. Maurer, and S. Scholz, "Effiziente systematische testgenerierung für fahrerassistenzsysteme in virtuellen umgebungen," *Automatisierungssysteme, Assistenzsysteme und Eingebettete Systeme Für Transportmittel*, 2013.

[UMR+15] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, IEEE, 2015, pp. 982–988.

[Xio13] Z. Xiong, *Creating a computing environment in a driving simulator to orchestrate scenarios with autonomous vehicles*. University of Leeds, 2013.