Ref. Ares(2021)4337959 - 03/07/2021

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

D1.2 Analysis of functional safety aspects on single-criterion optimization and first release of the test bench suite

Version 1.0

Documentation Information

Contract Number	871669
Project Website	www.ampere-euproject.eu
Contratual Deadline	30.06.2021
Dissemination Level	[PU]
Nature	R
Author	Dirk Ziegenbein, Arne Hamann, Michael Pressler (BOS)
Contributors	Viola Sorrentino, Francesca Nizzi, Massimiliano Polito (THALIT) Delphine Longuet (TRT) Enkhtuvshin Janchivnyambuu (SYS)
Reviewer	Delphine Longuet (TRT)
Keywords	safety, security, AMALTHEA, Capella

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

Change Log

Version	Description Change
V0.1	Initial version
V0.2	Added ODAS use case related info
V0.3	Added security information for automotive use case
V0.4	Added chapter for SLG
V0.5	finalized for review
V0.6	incorporated review comments and migrated to new template
V1.0	Final version for submission

Table of Contents

1 Introduction

This document describes Deliverable 1.2, related to WP1 *System Model Description and Use-cases* and targeting milestone MS2 (i.e., from M7 until M15, with a 3-months extension). This deliverable corresponds to the first release of the test bench suite, upon which single-criterion optimization model transformations are applied. The tasks *Task 1.2: Generation of AMPERE test bench suite and use case preparation* as well as *Task 1.3: Functional safety and security* contribute to this deliverable.

Task 1.2, involving partners THALIT and BOS, develops a test bench suite, composed of the use cases as well as smaller and manageable kernels resembling a limited functionality of each use case which shall enable rapid evaluations of the AMPERE optimization framework.

Task 1.3, involving partners TRT, THALIT, BOS, EVI and SYS, evaluates the functional safety and security aspects required at each integrity level as defined in the relevant standards. The purpose is to ensure that the AMPERE ecosystem as well as the HW/SW platform complies with these requirements.

This deliverable first analyzes the relevant functional safety aspects in Chapter 2, the security aspects in Chapter 3 and discusses the first release of the AMPERE Test Bench Suite in Chapter 4.

2 Functional Safety Aspects

In this chapter, the AMPERE use cases from the railway and automotive domains are analyzed with respect to the relevant safety standards and the requirements they pose on the use cases. However, we will focus only on the aspects and requirements which are relevant to the AMPERE project, namely the optimized mapping and parallel execution of applications on heterogeneous HW platforms. In addition, it is also shown how to reflect the safety aspects in the system model as well as in the SW platform.

2.1 Railway Use Case

2.1.1 Relevant standards

Most of railway safety rules are addressed in the following CENELEC standards:

- EN50126;
- EN50128;
- EN50129.

According to EN50126 [1], the standard *"provides railway duty holders and railway suppliers [...] with a process which will enable the implementation of a consistent approach to the management of reliability, availability, maintainability and safety"*. The standard introduces events influencing railway safety, such as HW failures and human factors. A risk-based approach is presented to mitigate hazards that takes into consideration both their frequency and severity. A risk matrix (that in real situations is defined by the Railway Safety Authority) is used to associate a risk rating to each frequency/severity pair. Objectives, requirements and deliverables for safety activities through all life cycle phases are also presented.

EN50126 standard also provides methods to derive the Safety Integrity Levels (SIL) for safety related functions. Four SIL are defined starting from SIL1 (lowest level of safety integrity) to SIL4 (highest level of safety integrity).

Standard EN50128 *"concentrates on the methods which need to be used in order to provide software which meets the demands for safety integrity which are placed upon it"* (from EN50128 "Introduction") [2]. The Standard provides a set of requirements on development, deployment, verification, validation and maintenance of any safety-related software for the railway industries. It also defines requirements concerning organisational structure and the division of responsibility involved in the development, deployment and maintenance activities. A SIL0 is also defined to identify systems that are not safety related.

EN50129 *"is concerned with the evidence to be presented for the acceptance of safety-related systems, it specifies those life-cycle activities which shall be completed before the acceptance stage"* (from EN50129 "Introduction") [3]. The concept of a Safety Case is used as a framework for a systematic documented approach to quality management, safety management, functional and technical safety, safety acceptance and approval. This document also contains details on SIL use in safety-related systems for railway applications.

Last but not least, EN50159 [4] has also been taken into consideration in our analysis as it deals with safety-related communication between safety-related equipment using a transmission system in railway applications.

2.1.2 Safety requirements

A Preliminary Hazard Analysis (PHA) has been performed starting from the safety function that characterises the system, i.e. detection and avoidance of track obstacles.

During the PHA some hazards have been identified and analysed (e.g., "the obstacle is not detected", "the obstacle is detected in a more distant position than the actual one", etc.). For each hazard, causes and potential

effects have also been identified. Hazards causes have been subsequently analysed more in depth to define lower level causes (e.g., “sensors faults”, “track layout constraints”, “environmental conditions”, etc.).

Risk mitigations have been identified for each low-level cause with the aim of reducing the hazard residual risk as low as reasonably possible, as required by EN50126 standard (e.g., “a sensor periodical vitality test should be made ensuring its integrity”).

Finally, a bottom up approach was followed to map railway level mitigations to a set of higher-level mitigations to be used as safety requirements for AMPERE system design ecosystem.

As a result of this analysis the following system safety requirements have been identified.

[SYS-ODAS-REQ-200]: AMPERE items managing safety critical aspects must be managed (e.g., designed, implemented, tested, etc.) according to EN50126.

Test method: Analysis.

[SYS-ODAS-REQ-201]: Code generation tools must consider EN50128 constraints when generating code for safety related design items.

Test method: Analysis.

[SYS-ODAS-REQ-202]: If components are executed in an environment in which computing platforms are connected through a local interconnection network, the communication system must comply to EN50159.

Test method: Analysis.

[SYS-ODAS-REQ-203]: Safety related SW modules and their corresponding data-set can only be offloaded to hardware acceleration devices having the required Safety Integrity Level (SIL).

Test method: Analysis.

[SYS-ODAS-REQ-204]: A design item attribute “safety related” must be available to designers to identify design items involved in implementing safety functions (safety related design items).

Test method: Demonstration.

2.2 Automotive Use Case

AMPERE Deliverable D1.1 [5] already listed the relevant top level safety requirements[1] for the automotive use case Predictive Cruise Control (PCC). In this section, these requirements are further detailed and in some cases additional requirements are derived.

The PCC system is subject to the international standard ISO 26262 [6] such that the AMPERE ecosystem should fulfill the requirements of ISO 26262 as stated in *[SYS-PCC-REQ-106]*. ISO 26262 defines a safety lifecycle for automotive applications, i.e., a process to be performed during development of safety-critical automotive applications, as well as requirements on activities to be performed during product development on system, HW and SW levels.

In the ISO 26262 safety lifecycle, a key concept is to derive an Automotive Safety Integrity Level (ASIL) for each safety goal as well as each architectural element (e.g., software and hardware elements). The ASIL ranges from A to D, with A being the lowest and D being the highest level. A special value QM is assigned to elements that are not safety critical and thus outside the scope of ISO 26262. The higher the ASIL, the more stringent are the requirements of ISO 26262 on the development of the element. For the safety goals, the assigned ASIL depends on the probability, severity and controllability of the consequences of the hazards that are prevented by that safety goal.

In general, new architecture elements and requirements inherit the ASIL of their parent, i.e., the safety goal they belong to. However, ISO 26262 allows to decompose elements into elements with lower ASIL, provided they are sufficiently independent (e.g., through diverse implementations) and together implement some kind of redundancy (i.e., the parent component fails only if multiple of the children fail). This is named ASIL tailoring

[1] These requirements from D1.1 are numbered *[SYS-PCC-REQ-1XX]*.

in ISO 26262. In case ASIL tailoring is applied, decomposed elements have extended ASIL assignments in the form $X(Y)$ (with $X, Y \in \{QM, A, B, C, D\}$ where X is the elements' reduced ASIL and Y the initial ASIL of the safety goal where the elements have been derived from). An element with ASIL $X(Y)$ may thus be decomposed into two elements with ASIL $X1(Y)$ and $X2(Y)$ such that $f(X1) + f(X2) \geq f(X)$ with $f = \{QM \mapsto 0, A \mapsto 1, B \mapsto 2, C \mapsto 3, D \mapsto 4\}$. The reference to the initial ASIL is important because some properties of the elements (e.g., the ability to cope with random HW faults) are not subject to ASIL tailoring and have to still be evaluated according to the initial ASIL of the safety goal.

In the scope of AMPERE, we focus on the ISO 26262 requirements which are relevant for the optimized mapping and parallel execution of SW elements on heterogeneous HW platforms. Other ISO 26262 requirements are dealt with in the ITEA3 project PANORAMA [7] which is also based on APP4MC and the common meta model AMALTHEA. Thus, PANORAMA results such as the structured modeling of assurance cases via the Open Dependability Exchange Meta model (ODE) or the traceability between (safety) requirements and design elements via Eclipse Capra will also be compatible with AMPERE results.

The PCC consists of different subfunctionalities or applications with different ASIL classifications which will have to be simultaneously executed on the same HW/SW platform (see *[SYS-PCC-REQ-107]*). These applications (even the ones with the same ASIL) have to be spatially and temporally isolated from each other in order to prevent interference and fault propagation between them (see *[SYS-PCC-REQ-108]*). Additionally, high ASIL applications need to be protected against random HW faults by being executed in lockstep which can be realized via dedicated lockstep HW cores or by redundant execution of the respective SW elements of the application (see *[SYS-PCC-REQ-109]*).

2.2.1 Derived Safety Requirements

From these requirements, we can derive detailed requirements on the modeling of SW elements which are needed to expose their safety-motivated needs to the AMPERE optimization framework such that the optimized system is able to meet its safety requirements.

[SYS-PCC-REQ-200]: The AMPERE ecosystem shall support to express the membership of a SW element to an application.

Test method: Inspection.

This is required for the correct configuration of the temporal and spatial isolation mechanisms of the HW/SW platform to prevent fault propagation between SW elements of different applications.

[SYS-PCC-REQ-201]: The AMPERE ecosystem shall support the assignment of ASIL to a SW element.

Test method: Inspection.

[SYS-PCC-REQ-202]: The AMPERE ecosystem shall support the assignment of ASIL to a HW element.

Test method: Inspection.

While it is obvious that the ASIL of a function needs to be assigned to the SW elements realizing the function, please note that there is a subtle difference in the assignment of an ASIL to a HW element executing these SW elements. HW elements can be certified to an ASIL as a safety element out of context (SEooC), which means that the execution of SW elements requiring this ASIL is supported if all contextual assumptions of the certification (typically listed in the safety manual) are fulfilled. Thus, a simple "deploy-and-forget" matching of ASIL X SW elements on ASIL $Y > X$ HW elements is not possible but needs to be accompanied by a careful analysis of the corresponding safety manual.

[SYS-PCC-REQ-203]: The AMPERE ecosystem shall support to express mapping separation constraints between (groups of) SW elements denoting that these (groups of) SW elements may not be mapped to the same HW element.

Test method: Inspection.

Mapping separation constraints are required for the elimination of common cause failures which can be mandated for diverse applications due to ASIL decomposition or for redundant applications which are introduced e.g. in order to protect against random HW faults. Please note that we see ASIL decomposition as a manual design step resulting in the designer specifying the separation constraint. However, we envision that the AMPERE optimization framework will be able to introduce redundancy on its own in order to realize a SW lockstep or to meet a resiliency or availability goal.

2.3 Modeling Safety Aspects

This section outlines how to implement the derived requirements in the system models AMALTHEA and Capella.

2.3.1 AMALTHEA

For specifying properties of model elements, there is the generic concept of tags in AMALTHEA which we will use to assign SIL and ASIL as well as the membership to an application. For the railway use case, SIL will be defined in AMALTHEA using tags as shown in Figure 1. Similarly, for the automotive use case, tags for the ASIL will be defined. As mandated by *[SYS-PCC-REQ-201]* and *[SYS-ODAS-REQ-203]*, every safety-related runnable in the SW model will be annotated with a tag denoting its (A)SIL. In the HW model, HW elements will be annoted with a tag denoting its (A)SIL as required by *[SYS-PCC-REQ-202]* and *[SYS-ODAS-REQ-203]*. Analogously, runnables are also annoted with a tag denoting their membership to an application.

In order to constrain the mapping of SW elements to HW elements, AMALTHEA has the concept of runnable affinity constraints (see Figure 2). We use in particular the RunnableSeparationConstraint to define separation constraints between (groups of) runnables realizing diverse implementations or being redundant instances.

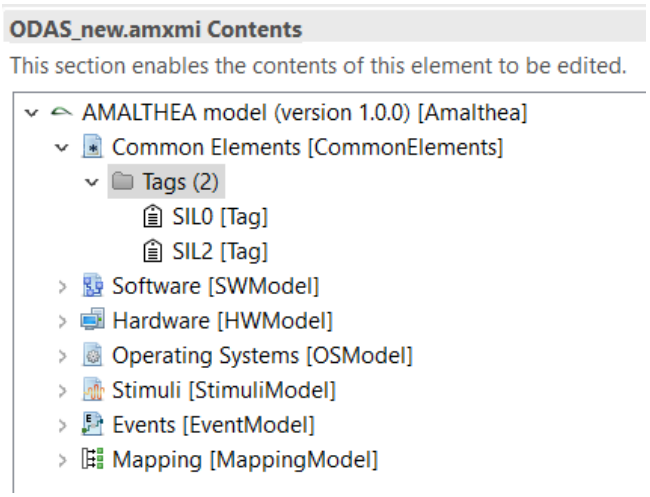

Figure 1: AMALTHEA Tags for SIL assignment

2.3.2 Capella

In the AMPERE project, we use Capella to describe the physical architecture of the use cases. It is the lowest level of abstraction of the Arcadia model-based system engineering method. In the AMPERE toolchain, this physical architecture is translated to an AMALTHEA model, via a synchronisation bridge developed by TRT (see D6.3 [8] for the details about the synchronisation). From the safety requirements of both use cases, we extracted safety-related elements that may be modeled in the physical architecture of the system in Capella.

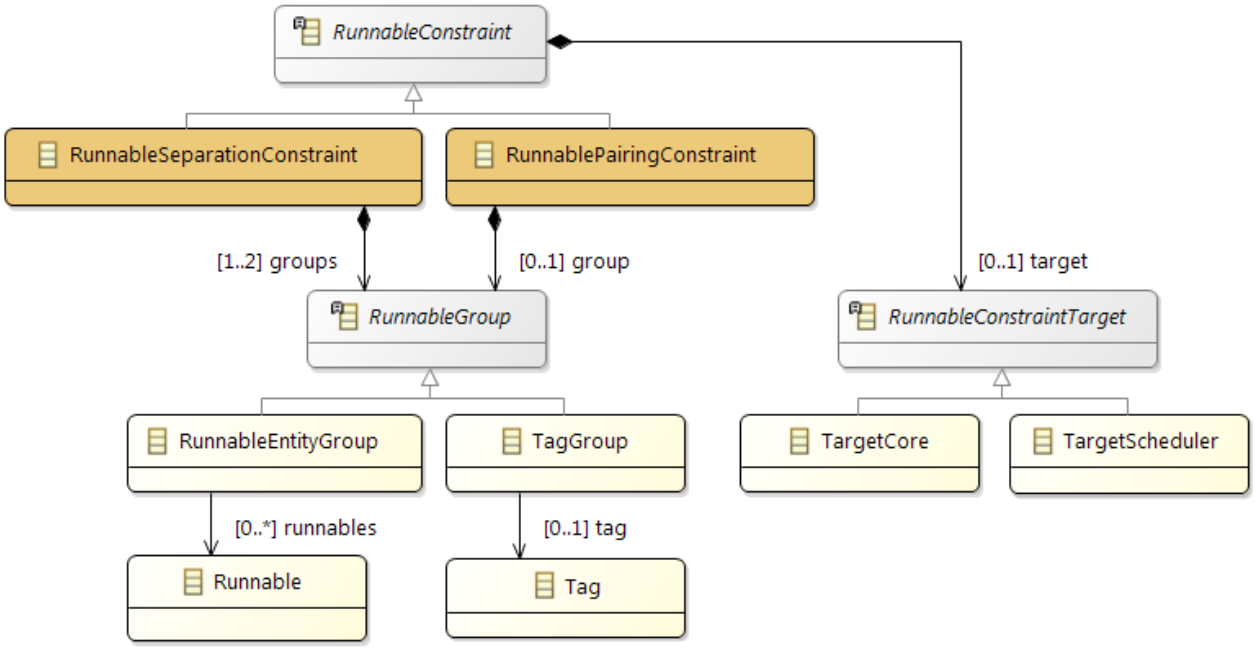

Figure 2: AMALTHEA runnable affinity constraints

(A)SIL. In both the automotive and the railway use cases, the safety requirements refer to the (automotive) safety integrity levels and the need to express constraints about the mapping between software and hardware elements based on these levels (see *[SYS-ODAS-REQ-204]*, *[SYS-ODAS-REQ-203]* and *[SYS-PCC-REQ-109]*). In Capella, a natural way to assign (A)SIL to software and hardware elements is to use property values. A property value is a kind of custom attribute any model element may carry. Figure 3 shows the definition of a SIL enumeration property type and the assignment of a value of this type to physical function *CAMERA object detection*, through a property value of type SIL.

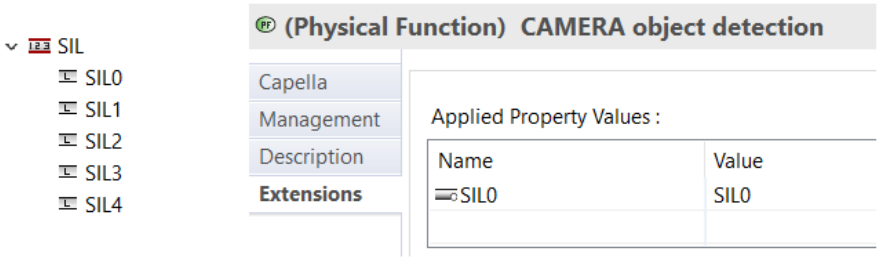

Figure 3: Capella property values for SIL assignment

Since a property value is a model element itself, all the elements assigned with the same value are linked to the same model element, therefore can be grouped by this property value. Since the concept of tag in AMALTHEA can be used in the same way, the property value denoting the SIL of a model element will be translated to an AMALTHEA tag by the Capella-AMALTHEA bridge.

Applications. Another need that was extracted from the safety requirements is that of grouping physical functions by applications, to be able to express separation constraints. We think the notion of application might find a natural counterpart in the notion of non-deployable behavioral component in Capella. Deployable and non-deployable behavioral components are used to physically or logically, respectively, group physical functions. If the grouping is only intended to be logical and does not constrain the allocation of physical functions to hardware elements, the behavioral component is non-deployable. If the physical functions grouped by a

behavioral component must be allocated to the same hardware element, then we use a deployable one. Since the notion of application here is lightweight and does not constrain the allocation of the physical functions it contains, non-deployable behavioral components seem to be a relevant solution to represent it in Capella. The left-hand side of Figure 4 shows a Capella physical architecture where physical functions (green boxes) are grouped by non-deployable behavioral components (blue boxes).

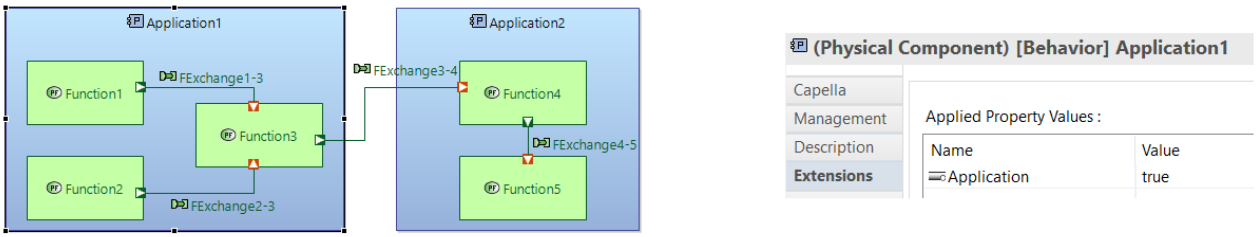

Figure 4: Capella representation of applications

However, these components may be used for different purposes in the Capella model, and perhaps not only to represent applications, therefore we plan to add a boolean property value *Application* to tag those that must be covered by the translation between Capella and AMALTHEA (see the right-hand side of Figure 4). Thus, the Capella-AMALTHEA bridge will be extended to translate the physical functions belonging to a behavioral component assigned with this property value to runnables tagged with the corresponding application.

2.4 Safety Aspects of the SW Platform

The PikeOS hypervisor already is a Multiple Independent Levels of Security/Safety (MILS) compliant architecture [9]. The PikeOS with MILS architecture enables developers to achieve safety and security by design using the MILS approach and, thus, to fulfil *[SYS-PCC-REQ-107]* and *[SYS-PCC-REQ-108]*.

PikeOS comes up with a modular and hardware independent certification approach according to the main industrial and functional safety standards including ISO 26262. PikeOS qualified tool chain and Safety CertKit include pre-certified components enabling fast and affordable system certification up to ASIL D. The resource and time partitioning capabilities of PikeOS allow to execute software components of different criticality levels on one module if all critical software components have the required design assurance level and all non-critical software components are executed in a normal partition. The certified PikeOS Microkernel is the only component, which runs with supervisor privileges. It provides:

- Hardware abstraction
- Resource and time partitioning
- Execution entities (threads)
- Separate address spaces (tasks)
- Communication primitives
- Timers
- Exception and interrupt handling
- Health Monitoring

For more information about the PikeOS architectural design, please refer to the AMPERE Deliverable D4.1 “Runtime architecture” [10].

3 Security Aspects

In this chapter we discuss security aspects related to the railway and automotive use cases addressed in AMPERE.

3.1 Railway Use Case

THALIT will follow the ISA/IEC 62443 Cybersecurity Standards [11] in the ODAS development.

The ISA/IEC 62443 Cybersecurity Standards is divided into different sections and describes both technical and process-related aspects of industrial cybersecurity.

Primary objective of cybersecurity is to detect, react and prevent:

- malicious disclosures of information (*Confidentiality*) that could be used to perform malicious acts;
- malicious modifications of functions/information (*Integrity*) that may compromise the delivery or integrity of the service;
- unavailability of the system that may lead to an accident or an unsafe situation (*Availability*).

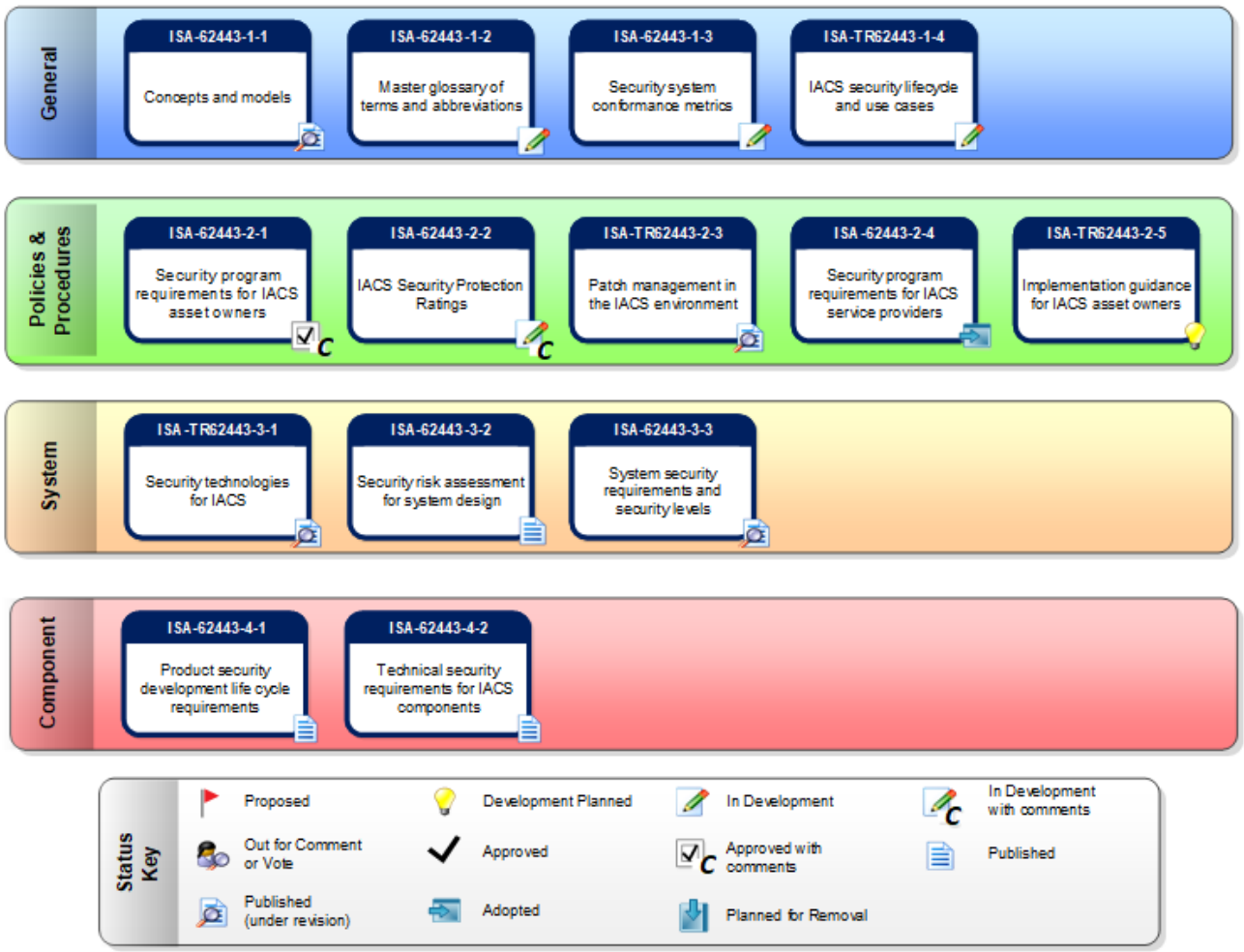

Figure 5: ISA/IEC 62443 Cybersecurity Standards.

The sections of ISA/IEC 62443 are four:

- General: in this section, publications that are common to the entire series are inserted. In particular, in this section fundamental information such as concepts and terminology are presented.
- Policies & Procedures: this section deals with policies and procedures associated. In particular the concept of Asset Owner is introduced and various aspects of creating and maintaining an effective security program is presented.
- System: the elements in this section address requirements at the system level. In particular, this section describes system design guidance and requirements for the secure integration of control systems.
- Component: the elements in this section address requirements at the component level. In particular, the elements describe the specific product development and technical requirements of control system products.

As shown in Figure 5, some of these publications are under development.

The ISA/IEC 62443 standards include the concept of security assurance levels (SL). The specification defines a series of requirements designed to bring system security to one of the four defined levels. These four levels are summarized in Table 1. Moreover a SL 0 level is defined: this class means that no specific requirements or security protection in necessary. Every solution must be designed to address attacks coming from one of these classes.

Security Level	Target	Skills	Motivation	Means	Resources
SL 1	Casual or Coincidental violations	-	No motivation	Non intentional	-
SL 2	Cybercrime, Hacker	Generic	Low	Simple	Low
SL 3	Hacktivist, Terrorist	System Specific	Moderate	Sophisticated	Moderate
SL 4	Nation	System Specific	High	Sophisticated	Extensive

Table 1: Security Level defined in the standard.

The ISA/IEC 62443-3-3 specification defines a list of requirements necessary to obtain compliance to the desired SL. In general, these mandatory requirements are divided into seven Fundamental Requirements (FR). Every FR is moreover divided into System Requirements (SR).

The ISA/IEC family standard addresses what is expected and required for secure components and system, including the processes used in their development. This family puts the *secure by design* principle as an essential element of an effective long-term response to the need for secure industrial systems.

Another concept introduced by ISA/IEC 62443 in the *compensating controls*. If a product does not have the required security functionality, the overall system can still meet the requirements if the required functionality is provided by a different component in the system.

3.2 Automotive Use Case

In March 2021, a new automotive cybersecurity standard *ISO/SAE 21434 "Road vehicles - Cybersecurity engineering"* [12] has reached the status *Final Draft International Standard (FDIS)*. This means that the standard is in principal final even if some minor details might still be changed. Due to the increasing risks from cyber attacks on vehicles and because the infrastructure for online vehicle updates (OTA), fleet management, communication between vehicles (Car2X/V2X) and other requirements provide vehicles with new attack surfaces, the standard proposes measures and best practices to be taken into account during development. The standardization is related to the currently developed EU regulation on cyber security.

The AMPERE toolchain is an ideal starting point to consider security requirements according to ISO/SAE 21434 (especially the V&V methods described in Chapter F). Even if security is not the focus of the AMPERE project,

it should be ensured that the AMPERE toolchain can later be extended to meet the described security requirements, e.g. by integrating state-of-the-art open source tools that cover relevant security aspects described in ISO/SAE 21434. The following aspects are the most important:

- Generate code such that side channel attacks (timing, energy) are prevented (e.g. by choosing low-level libraries in an adequate way)
- Partition SW such that security-critical functionality is executed in trusted execution environments (e.g. special cores)
- Automatically check for security vulnerabilities (e.g. using fuzzing or static code analysis) after synthesis
- Automatically check for known vulnerabilities using CVE (Common Vulnerability Enumeration)
- Check for “static secrets" embedded in the code (e.g. password, keys, user names)

3.3 Security Aspects in AMPERE

In contrast to safety, modeling security aspects using the modeling approaches AMALTHEA and Capella that are employed in AMPERE is not purposeful.

In summary, it can be noted that the AMPERE project does not need to actively work on security modeling and analysis, as there are already many mature open source tools that cover all relevant security aspects. The AMPERE project only has to ensure that the developed toolchain is compatible in principle, and does not block the way to link these tools in an appropriate way later.

4 AMPERE Test Bench Suite

In this chapter, we will discuss the current state of the AMPERE test bench suite. By developing and releasing a synthetic load generator (SLG) described in Section 4.1, the project partners are enabled to generate their own synthetic kernels of arbitrary complexity from AMALTHEA models in order to test their methods and tools. The current state of the models and code of the ODAS and PCC use cases is described in the following sections. The test bench suite is intended to grow in the subsequent project phases.

4.1 Synthetic Load Generator

Bosch decided to contribute a Synthetic Load Generator (SLG) that eases the exchange of application performance models without the need of code sharing. As already described in Deliverable 1.1 [5] the SLG has a standard behavior that generates code to mimic the timing behavior of real applications. The focus is to mimic executed computation cycles on the cores as well as memory accesses to represent the interferences on the memory hierarchy between applications executed concurrently on multi-core platforms. On top of that the synthetic load generator also generates the "glue code" allowing different sub-systems to exchange messages and data via publish-subscribe messaging patterns.

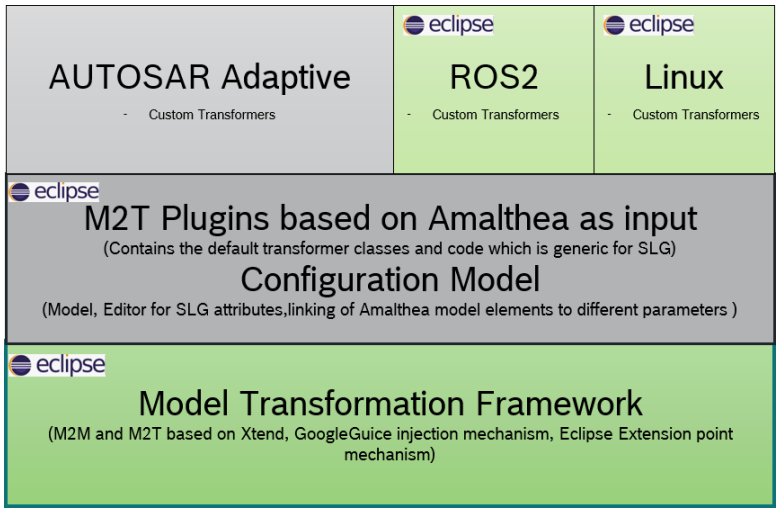

Figure 6: Synthetic Load Generator

Figure 6 depicts the structure of the SLG hosted by the APP4MC project [13] in a sub-repository [14]. The base technology is a generic transformation framework that was described in Deliverable 2.1 [15] which allows model-to-model (M2M) and model-to-text (M2T) transformations. The framework provides a simple extension mechanism to inject custom code extensions to alter or add transformation logic.

On top of the generic SLG generator we added two adapters to generate code for a Linux based system where *pthreads* are generated for every *task* modelled in APP4MC. To activate the tasks the user can either model a periodic stimulus in APP4MC or use the *InterProcessTrigger* to activate another task. The SLG will generate the logic to handle the *pthread* activation in a Linux environment. Figure 7 illustrates the relationship between the model and the generated code structure. Additional information and a developer guide can be found in the path `load_generator\linux\docu` of the SLG repository. The input folder of the product contains examples for users.

Another plugin developed for Ampere is the generator for ROS2 middleware communication. In this case the SLG generator is using ROS2 primitives to handle communication between processes via a publish/subscribe mechanism. The general mechanism how the execution semantics of ROS2 can be mapped to APP4MC is

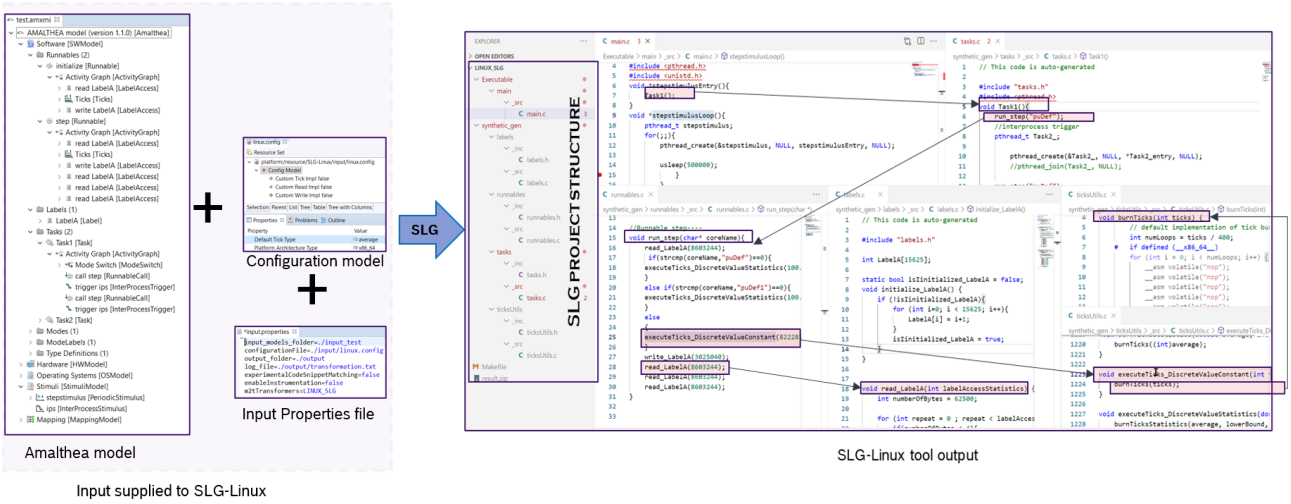

Figure 7: Synthetic Load Generator Linux

described in Deliverable 1.3 [16].

To use the SLG for ROS2 we need to model nodes. A node is a crucial part of ROS2. A node in APP4MC is modelled by defining a *Tag* with the respective name of the node. This tag is assigned to multiple tasks in APP4MC since a ROS2 node can implement multiple callbacks. Every callback is modelled as a separate task in APP4MC. Model extensions to cover the execution semantics of multiple callbacks within a task are currently discussed with the APP4MC team and are described in Deliverable 1.3 [16].

A ROS2 node can subscribe to topics and publish data via topics. This is modelled via channels in APP4MC. A node can either be triggered via a periodic stimulus or via an event because of data that was published to a subscribed topic. A timer callback is triggered via a periodic stimulus in APP4MC. A data driven activation is modelled via an *EventStimulus*. An example model is publicly available in the repository [14] in the path `load_generator\ros2\releng\org.eclipse.app4mc.slg.ros2.product\input`.

ROS2 is a publicly available publish/subscribe middleware that is used in Ampere. It can also be seen as a substitute for AUTOSAR Adaptive. While the common SLG with the extensions towards ROS2 and Linux is publicly available we also implemented a SLG generator for the ETAS RTE-VRTE [17] implementation of AUTOSAR Adaptive. This generator is only available within Bosch.

4.1.1 Code Extensions

Since the SLG focuses on generating synthetic loads to mimic the executed computation cycles on a core, the interference impact on the memory and the overhead introduced by the communication middleware of the applications generated code is rather simple. To include representative code for other use cases within Ampere an extension to include external code and even complete applications is currently in the final stages of implementation. The user will be able to specify entry functions for external code in the APP4MC model via custom properties. This is shown in Figure 8. In this case the SLG will not generate the synthetic code for a task or runnable but call a void void function specified in the custom property.

There exists multiple option to provide external code or applications to the SLG. In the configuration model new extensions are introduced to specify e.g. the location of *include* files to find the entry function for the external code, the location of provided *make* files that will generate a library out of the provided code, locations of external libraries, and also a possibility to provide keywords (e.g: -pthread, -fopenmp) to the linker.

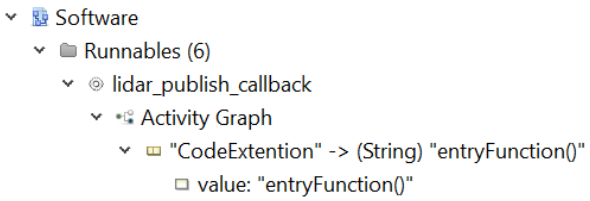

Figure 8: Synthetic Load Generator Code Extensions

4.2 Railway Use Case

The Test Bench Suite of the railway use case starts with the ODAS model in Capella and proceeds with the ODAS model in AMALTHEA applying the bridge implementation, which allows the interaction between these two different tools. Finally, using the Synthetic Load Generator, synthetic code for the ODAS application is obtained for the tasks and runnables in the ODAS model in AMALTHEA.

The ODAS models in Capella, in AMALTHEA and the resulting code generated with the SLG are available in the Software directory of the AMPERE GitLab.

4.3 Automotive Use Case

At project start, Bosch already provided APP4MC models of the engine control management system in the Waters Industrial Challenges 2016 [18] and 2017 [18] as well as a representative model of an automated driving application in 2019 [19]. Currently Bosch is working on representative performance models for the Adaptive Cruise Control, a Traffic Sign Recognition, and the Predictive Cruise Control. The Predictive Cruise Control uses existing data from the other applications and/or provides input to the other application via ROS2 communication mechanisms. Due to support dependencies to our business units and their Covid19 related delays (work reduction) this work is delayed but will be finished in the early second half of 2021. In the meantime the code extension mechanisms provides an excellent replacement to hook external representative benchmarks to the SLG like the San Diego [20], DAPHNE [21] and synthetic IsolBench [22] suite.

5 References

[1] CEI, "CEI EN 50126 Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)." [Online]. Available: https://standards.globalspec.com/std/1507670/cei-en-50126

[2] CENELEC, "CENELEC - EN 50128 Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems," 2020. [Online]. Available: https://standards.globalspec.com/std/14317747/EN2050128

[3] ——, "CENELEC - EN 50129 Railway applications - Communication, signalling and processing systems - Safety related electronic systems for signalling," 2018. [Online]. Available: https://standards.globalspec.com/std/13113133/EN2050129

[4] ——, "CENELEC - EN 50159 Railway applications - Communication, signalling and processing systems - Safety-related communication in transmission systems," 2010. [Online]. Available: https://standards.globalspec.com/std/14256321/EN2050159

[5] AMPERE, "D1.1 System models requirement and use case selection," 2021.

[6] International Organization for Standardization (ISO), "ISO/IEC 26262:2018, Road Vehicles - Functional Safety," 2018.

[7] ITEA3 Project PANORAMA, "PANORAMA Project Website," May 2021, https://www.panorama-research.org/.

[8] AMPERE, "D6.3 Single criterion AMPERE ecosystem," 2021.

[9] J. Alves-Foss, P. Oman, C. S. Taylor, and S. Harrison, "The MILS architecture for high-assurance embedded systems," *International Journal on Embedded Systems*, vol. 2, pp. 239–247, 2006.

[10] AMPERE, "D4.1 Run-Time Architecture," 2020.

[11] International Society of Automation (ISA) 99 committee, "ISA/IEC 62443," 2010. [Online]. Available: https://www.iec.ch/blog/understanding-iec-62443

[12] International Organization for Standardization (ISO), "ISO/SAE FDIS 21434, Road Vehicles - Cybersecurity Engineering," 2021. [Online]. Available: https://www.iso.org/standard/70918.html

[13] "Eclipse APP4MC." [Online]. Available: https://www.eclipse.org/app4mc/

[14] "APP4MC Transformation Repository." [Online]. Available: https://git.eclipse.org/r/plugins/gitiles/app4mc/org.eclipse.app4mc.addon.transformation

[15] AMPERE, "D2.1 Model Transformation Requirements," 2020.

[16] ——, "D1.3 First release of the meta model-driven abstraction release," 2021.

[17] "ETAS RTA-VRTE." [Online]. Available: https://www.etas.com/de/portfolio/rta-vrte.php

[18] "Waters industrial challenge 2016." [Online]. Available: https://waters2016.inria.fr/challenge/

[19] "Waters industrial challenge 2019." [Online]. Available: https://www.ecrts.org/archives/fileadmin/WebsitesArchiv/ecrts2019/waters/waters-industrial-challenge/index.html

[20] S. K. Venkata, I. Ahn, D. Jeon, A. Gupta, C. Louie, S. Garcia, S. Belongie, and M. B. Taylor, "Sd-vbs: The san diego vision benchmark suite," in *IEEE International Symposium on Workload Characterization (IISWC)*, 2009.

[21] L. Sommer, F. Stock, L. Solis-Vasquez, and A. Koch, "DAPHNE - An automotive benchmark suite for parallel programming models on embedded heterogeneous platforms: work-in-progress," in *International Conference on Embedded Software Companion (EMSOFT)*, 2019.

[22] P. K. Valsan, H. Yun, and F. Farshchi, "Taming Non-Blocking Caches to Improve Isolation in Multicore Real-Time Systems," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2016.