Project number: 826276

CPS4EU

Cyber Physical Systems for Europe

D5.4 – CPS Tool Best Practice Guide

Reviewer: GOUGEON Philippe (Valeo), HAMELIN Etienne (CEA)

Dissemination level: Public

Funded by the European Union's H2020 GA - 826276

Document Manager:	**Luis PALACIOS**		
Project Title:	Cyber Physical Systems for Europe		
Project Acronym:	CPS4EU		
Contract Number:	826276		
Project Coordinator:	VALEO		
WP Leader:	CEA		
Task:	T5.1 T5.2 T5.3 T5.4	**Task Leader:**	CEA
Document ID:	D5.4	**Version:**	Rev1.0
Deliverable Title:	CPS Tools – Best Practice Guide	**Date:**	July 01, 2021
		Approved:	
Document Classification:	Public		

Approval Status

Prepared by:	Luis Palacios, Rèda Nouacer, Morayo Adedjouma
Approved by (WP Leader):	
Approved by (Coordinator):	
Approved by (TPM)	

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

Contributors

Name	Partner
Réda Nouacer, Morayo Adedjouma, Yves Lhuillier, Zakaria Chihani, Palacios Luis, François Terrier, Chokri Mraidha, François Bobot, Marwa Zeroual, Gilles Mouchard	**CEA**
Valéry Morgenthaler	ANSYS
	INRIA
	SHERPA-Engineering
Paolo Azzoni	EUROTECH
Srdjan Krivokapic, Oliver Oey, Timo Stripf	EMX
	TUC
	TRUMPF
Noël Hagemann, Julia Rauscher, Bernhard Bauer	UnA
Antonio Ruiz-Alba, Miguel García Gordillo, Javier Coronel	ITI
	LEONARDO
	UGA

Version History

Version#	Date	Reason for change	Released by
0.1	May 10, 2021	Template First Release	L. Palacios (CEA) R. Nouacer (CEA) M. Adedjouma (CEA)
0.2	July 7, 2021	Review Version	All WP5 Participants
1.0	July 22, 2021	Final Version	R. NOUACER

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

Distribution List

Name	Company/Organization	Role / Title
Consortium	CPS4EU Consortium	n/a

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

Table of Contents

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

Tables

Figures

EXECUTIVE SUMMARY

The deliverable D5.4 comprises the guidelines, potential, limits and risks of the proposed toolchains and *clusters of tools* identified in previous deliverable D5.3. The main scope of the deliverable is to further specify the interactions proposed in deliverable D5.3, enriched by the experience and expertise of the partners, and guided by specific application scenarios. The specification is not only a necessary cornerstone for the POCs and demonstrators in the next stage (integration deliverable) but serves as the main reference for the generalization of the presented approaches into other CPSs and future projects.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

1 INTRODUCTION

1.1 Purpose

In order to facilitate the integration of CPS components into the overall system of systems architecture, we have provided in previous deliverables of WP5 engineering scenarios and *clusters of enabling tools* (deliverable D5.3) that describe the applicability and envisioned synergy of the project partners' capabilities for design, test, deploy and verification of CPSs. This document (deliverable D5.4) further specifies these scenarios and clusters, by providing guidelines and best practices to enable the required interaction between tools. These guidelines are provided from the point of view of the technology providers and industrial partners, and built atop the expertise and experience of each contributor in their respective fields.

Some of the main challenges for integration of tools and architectures in CPS systems, rely on the complexity of the control/communication systems, on the variety and heterogeneity of the components (sensors, actuators, sub-systems) and their composition to provide higher-level services and functions, in an aggregated and complex structure. The main contributions to the reuse, integration and deployment of CPSs are two folded: 1) the toolchains are enablers to support deployment of components into pre-integrated architectures (PiArchs specified in WP6), and 2) the toolchains are enablers for the integration of PiArchs into cyber physical systems of systems (CPSs). Hence, there is the need for guidelines, recommendations and best practices on how the proposed tool-chains can be integrated and applied, and on how the clusters of tools can interact.

Part of the challenges of the integration of heterogeneous services and tools, like distributed co-simulation, iterative code generation, scenario based simulation and AI-Based CPS system toolchain, rely on the proper delimitation and scope of each tool and the tool's requirements and mechanism for interoperability and analysis. The specifications in this document aim to provide the next generation of industrial projects with basis on how to integrate the available services, in the best manner in the context of the partner's tools and capabilities. The described resources are not limited to CPS4EU application, but are also extendable/adaptable to other CPS application domains. Ongoing development of demos, proofs of concepts and approaches in close interaction with these guidelines are in progress, and thus demonstrators of the integration and interaction approaches that exemplify and fine-tune these guidelines, will be provided at the end of the project.

The deliverable D5.4 specifically focuses on: a) the integration of simulation tools with approaches such as system design, code generation and optimization tools, and on b) knowledge-based engineering to cope with standardization, interoperability, exchangeability and reasoning services that enhance the design, test, deployment, maintenance and evolution of CPSs. Even though conceptually the approaches are tool/technology independent, it must be pointed out that there is an important effort required to specify the interaction and combination between the tools and partners. The deliverable presents the results of those efforts, and provides a good basis for dimensioning it.

1.2 Scope

This document covers tasks:

- T5.1: AI integration in CPS
- T5.2: Simulation for CPS
- T5.3: Trustworthy system engineering
- T5.4: Tool chain development and integration

1.3 Link to other documents/TASKS

ID	Description

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

WP6	CPS Pre-integration
WP7	CPS Automotive
WP8	CPS Industry automation
WP9	CPS for other industrial sectors

1.4 Definitions, acronyms, and abbreviations

Definition / acronym / abbreviation	Description
CPS	Cyber-Physical System
AI	Artificial Intelligence
DL	Deep Learning
NN	Neural Network
DT	Digital Twin
KB	Knowledge Base
PDDL	Planning Domain Definition Language

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

2 INTEGRATION SCENARIOS

In this section, we present the scenarios that portray the interaction and integration of tools, as described in deliverable 5.3. The following subsections aim to further specify the envisaged integration of tools in the context of CPSs and pre-integrated components, and provide guidelines and best practices to achieve the intended synergy.

2.1 HETEROGENEOUS CO-SIMULATION

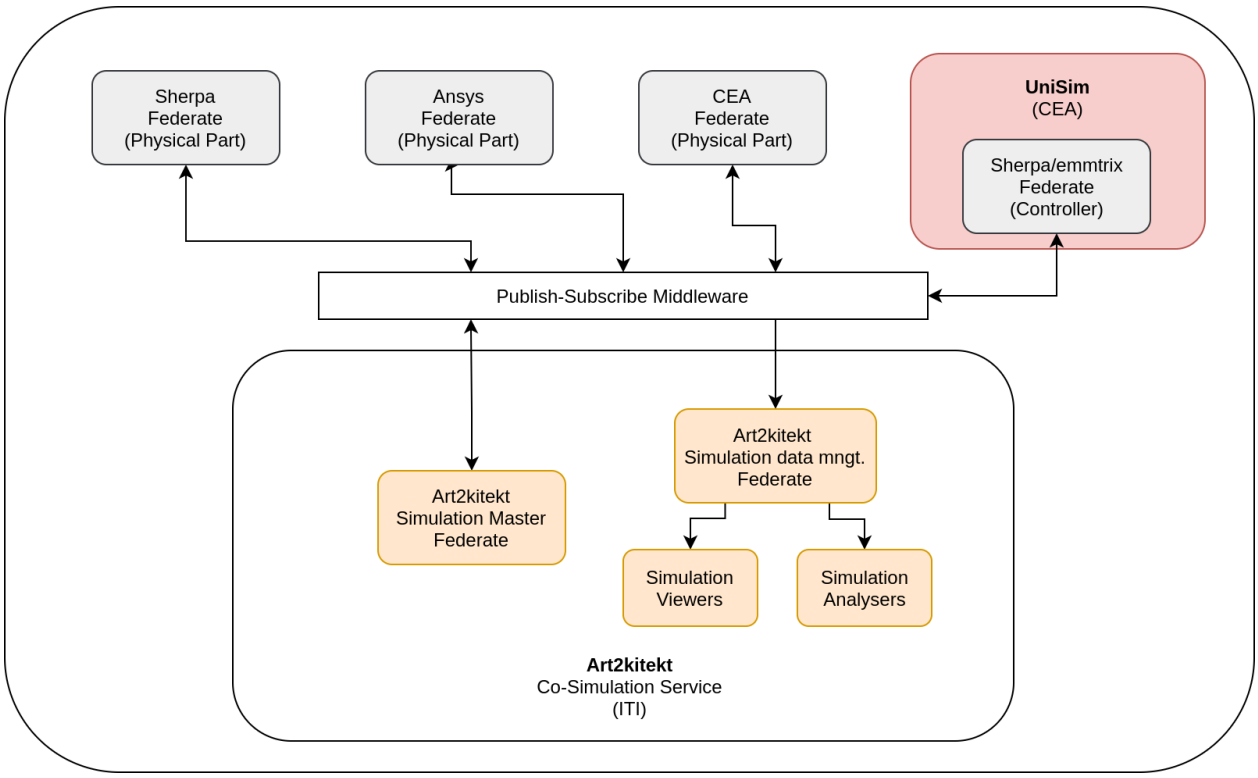

Figure 1 - Distributed Co-Simulation Execution

2.1.1 Introduction and purpose

Cyber-physical systems are often part of multi-domain heterogeneous complex systems. Simulating the complete system, or parts of it, and analyzing the component's behavior to be integrated is a widespread practice to ensure its correct operation before deployment in the real system. Due to the different subsystems' heterogeneity, incorporating them within the same simulation becomes a complex process.

The Heterogeneous Co-Simulation cluster (see Figure 1) focuses on the integration of simulation components in distributed simulations, when they are generated from different tool eco-systems. Each of these components is responsible for simulating a specific characteristic and all together they will form a complex simulation (*interoperability*). Additionally, once a simulation component is developed, it can be reused for other simulations, reducing development time (*reusability*).

This cluster aims to reduce the integration's effort in a heterogeneous co-simulation, improving both interoperability and reusability of the simulation components. To achieve this, we study the possible interactions between the different tools involved and the methodology to develop, adapt, and integrate the simulation components.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

2.1.2 How is this going to be achieved

The heterogeneous co-simulation cluster is divided into two steps: The first one is the generation of the simulation components, which must follow some guidelines to be executed in the same co-simulation. The second step is the execution of the co-simulation, in which the different components, created by different tools, are coordinated in a common scenario.

The following sections explain these steps. Section **Erreur ! Source du renvoi introuvable.** describes the context of the distributed co-simulation and how the execution is coordinated, and section **Erreur ! Source du renvoi introuvable.** introduces the different methods to generate simulation components.

2.1.2.1 Execution of the co-simulation

2.1.2.1.1 Coordination of the distributed co-simulation: HLA (RTI)

The distributed simulations require a complex architecture. Therefore, it is desirable to use middleware that can help in the integration and development of new simulation implementations by reusing existing modules. In this cluster, we use a middleware based on the High Level Architecture (HLA).

HLA is an IEEE standard **IEEE 1516-2010** for distributed computer simulation systems. In the HLA standard, a distributed simulation is called a federation. A federation comprises several HLA simulation entities, called federate, which can interact with them using the Run-Time Infrastructure (RTI). The RTI represents a federation execution backbone and provides services to manage communications and data exchanges.

A classical HLA federate consists of a simulation model and local RTI component (RTI ambassador). The simulation model is a physical, mathematical, or logical representation of processes and systems. These entities can communicate with each other through the RTI, that manages the federation, authorizes federates to contact others, and provides various services such as time management. HLA proposes multiple time management systems to ensure that messages are sent correctly and do not violate causal constraints.

Figure 2 shows a distributed simulation example showing the benefits of this approach. In the example there are four simulation units. Different technologies (Simulink, Modelica, etc.) can implement each one, but all must contain an RTI ambassador to interact with the RTI. This ambassador is encapsulated in the *Communication Federate Library* (Figure 4), that includes the procedures to initialise the federate and exchange data with the federation.

The RTI keeps track of which federates subscribe to each type of object, attribute, or interaction, which means that the federates want to receive that type of data. It also keeps track of which federates publish them.

The example consists of an inverted pendulum mounted on a cart that can move horizontally with a motor that generate thrust. It can be suspended stably in this inverted position by using a control system to monitor the pole's angle and keeping it balanced. To simulate the system, four federates were implemented; Pendulum, Controller, IMU, and Viewer federates. The pendulum has the physical dynamic of the system, the controller implements the algorithms to balance the pendulum, the IMU simulates the sensors, and the viewer shows a graphical representation.

The pendulum publishes its angle position, and the IMU and Viewer subscribe to it. The IMU publish its position output, and the controller subscribes to it, and finally, the controller publishes its thrust output, and the pendulum subscribes to it.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

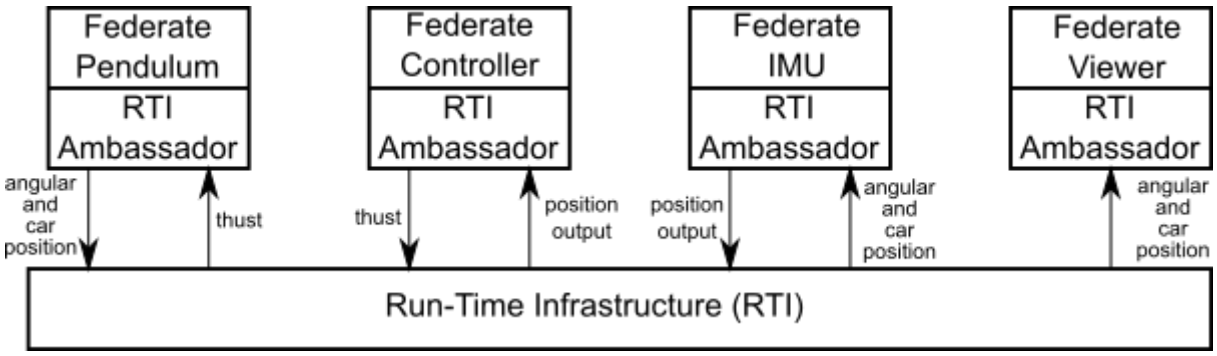

Figure 2 - HLA Federation

2.1.2.1.2 Explanatory diagram of the simulation execution flow

An entire life cycle from a federate perspective is depicted in Figure 3. Usually, this execution flow is divided into three stages: initialization, operation and termination.

The Initialization stage includes connecting the federate to the RTI and joining the federation execution. After that, the federate must specify the data exchange configuration.

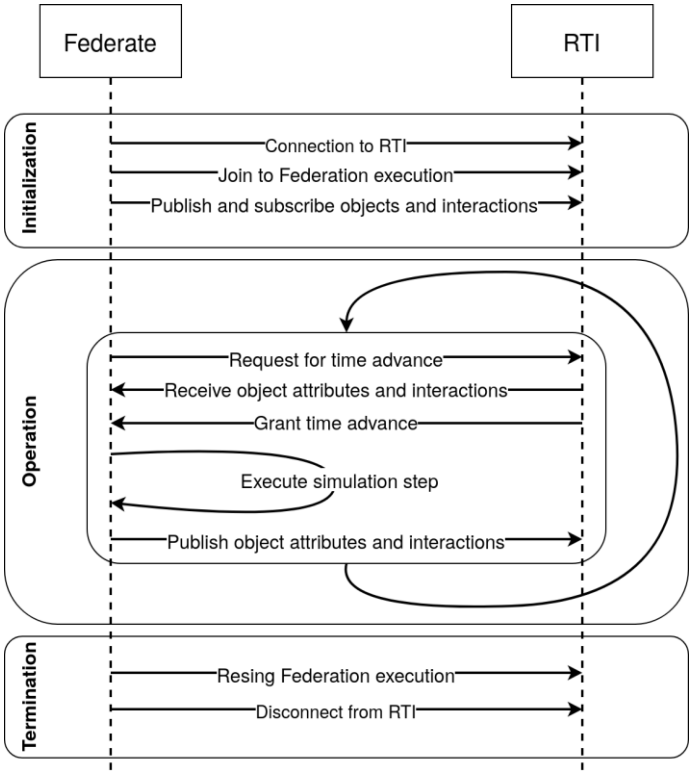

Figure 3 - HLA Federate Life Cycle

In this case, the operation phase is responsible for executing the simulation step in a loop process. A logical time is assigned to each federate, which will increase in each step of the operation stage. The federation must explicitly request the time advance to the RTI, which will only be possible when the rest of the federation also advances.

The termination of the simulation is carried out according to the strategy used. In a centralized approach, one federate is in charge of notifying the end of the simulation of the federation. Otherwise, each federation is responsible for defining when to end its execution.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

2.1.2.1.3 Emulation of execution environments

To run software designed for a specific platform in the distributed simulation, we use the UNISIM-VP machine emulator. UNISIM-VP[UNISIM Virtal Platforms] environment provides emulation capabilities at executable binary level which allows running and instrumenting (using DWARF standard debugging information format [Tool Interface Standard (TIS)] and symbol table of ELF file format) an unmodified software stack in binary form, the same that will run on the real target. UNISIM-VP environment supports several instruction sets: PowerPC, ARM, Intel, Mips, Sparc. In order to integrate with the RTI, the environment will non-invasively instrument execution (program and state) and use the *Communication Federate Library* to interface with the RTI.

UNISIM-VP emulation capabilities can roughly estimate execution time and provide the same floating-point precision as the real execution of the controller on the target processor. Therefore an emulation technique over a straightforward execution is more profitable than running the controller using the host compiler and floating-point precision.

2.1.2.1.4 Visualization and analysis of the evolution of the simulation and results using art2kitekt

art2kitekt provides a HLA-based simulation service where a series of components classified into viewers and checkers are included. Viewers to observe the evolution of the simulation and checkers to check the correct operation of the simulation elements. To do this, the art2kitekt simulation master federate (Figure 1) subscribes to objects and attributes of the different federations participating in the simulation during the configuration stage. During the execution stage, the master will publish the data generated by the different federations and the art2kitekt simulation service will be in charge of sending them to the configured viewers and checkers. The data received will be analyzed and will determine what the result of the simulation has been.

2.1.2.2 Generation of the simulation components

2.1.2.2.1 Generation of simulation models using FMU

The Functional Mock-up Interface (FMI) is a free standard that defines an interface to exchange dynamic models using a combination of XML files, binaries and C code zipped into a single file.

The implementation of this standard in tools, particularly in Matlab/Simulink, offers the capability to export (and import) simulation models as FMUs that support co-simulation in FMI version 1.0 and 2.0.

Papyrus-Moka allows exporting the executable model as an FMU. The executable model must be designed within the Papyrus UML editor based on the Foundational Subset for Executable UML Models (fUML), the PSCS (Precise Semantics of UML Composite Structures) and the Action Language for fUML (Alf). fUML enables the definition of the structural and behavioural semantics of systems using class and activity diagrams. Alf enables to detail the model elements with textual notations, e.g. for the implementation of operations, for the model to become executable. Specific behaviours can also be described according to UML profile or alternative execution semantics, which will be considered through Moka in the executable model. In addition, Moka allows defining input data required to execute the fUML. Moka enables the export of such defined executable model as a black box standalone FMU unit compliant to FM standard API and provides a standard binary interface as a shared library. The FMU contains the structure of the UML model and implements the mandatory functions of the FMI API. This export requires first encoding the execution semantics of UML models to FMI API by generating corresponding C code and then the model-to-model transformation from UML model to XML file.

TwinBuilder is a tool dedicated to the system simulation processes setup and deployment. It contains several methods and tools made specially to generate Reduced Order Models (ROM). All of these models can be generated a Model Exchange FMUs and can be exported as co-simulation FMUs. The generation process is a machine learning process based on simulation results. Static ROM Builders and Dynamic ROM Builders namely are generated based on data coming from results of simulation software. The simulation softwares can be any softwares from Ansys or not. It rely on very simple file

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

formats described in D5.3. All those models and system schematics engulfing them can be exported from TwinBuilder as a single FMU and can be tested and deployed in a Windows or Linux environment using Twin Deployer.

2.1.2.2.2 Generation of simulation models using Simulink

Sherpa engineering proposes C code generation for simulation models (controller part) using the Simulink coder toolbox. Then, the generated code can be used for deployment in various applications such as simulation acceleration, rapid prototyping, and hardware-in-the-loop (HIL) simulations. Moreover, the output generated code can be given as an input to the eCG tool for code optimisation.

2.1.2.3 Code generation and optimization of simulation models

emmtrix Code Generator (eCG) can be used to generate C code out of the Simulink models as an alternative to Simulink Coder from Mathworks. The code generated by eCG offers some advantages when used for CPS:

- The advanced data type inference system ensures that the smallest, most memory-efficient data types for variables are used without affecting the accuracy of the results. This feature is primarily helpful for the execution of embedded systems with a low amount of memory.
- Static memory management to avoid dynamic memory allocation.
- The code is optimised for further automated processing by preserving most information inherent in the model directly in C code. This optimisation means that arrays are represented as multi-dimensional arrays. Data processing is preferably handled in for-loops with constant boundaries, and that functions are only kept when necessary.
- It offers an integrated way to support dynamically sized arrays without the use of dynamic memory management.

emmtrix Parallel Studio (ePS) can be used to take the C code from Simulink Code or eCG and further optimise it to execute the pre-integrated architectures of CPS4EU. The main focus here is the distribution of the individual parts of the model/application onto the available processing elements of the target architecture. Additionally, ePS offers different ways to customise the generated code for specific purposes like its execution within a simulator. The code can therefore be adapted to the requirements of the simulator and provide additional instructions or information.

2.1.2.4 Adaptation of the models to common simulation architecture (HLA)

The simulation components (federates) must be executed as part of the HLA federation, regardless of the tools or standards used to define the models.

In the specific case of simulation models generated to comply with the FMU/FMI standard, most code generation tools provide a compressed FMU file containing the XML description model and the compiled library. The "FMU Adapter" can load the compressed FMU file at run-time, integrating the resulting model in a typical simulation architecture and avoiding linking it when the federate compiles.

When a different standard generates the model, the generated code must provide an interface that updates its inputs, obtains its outputs and executes its simulation step. For this, all federates shall use the *Communication Federate Library* (Figure 4) to communicate with the RTI, allowing them to publish and subscribe to the simulation data and synchronize in time.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

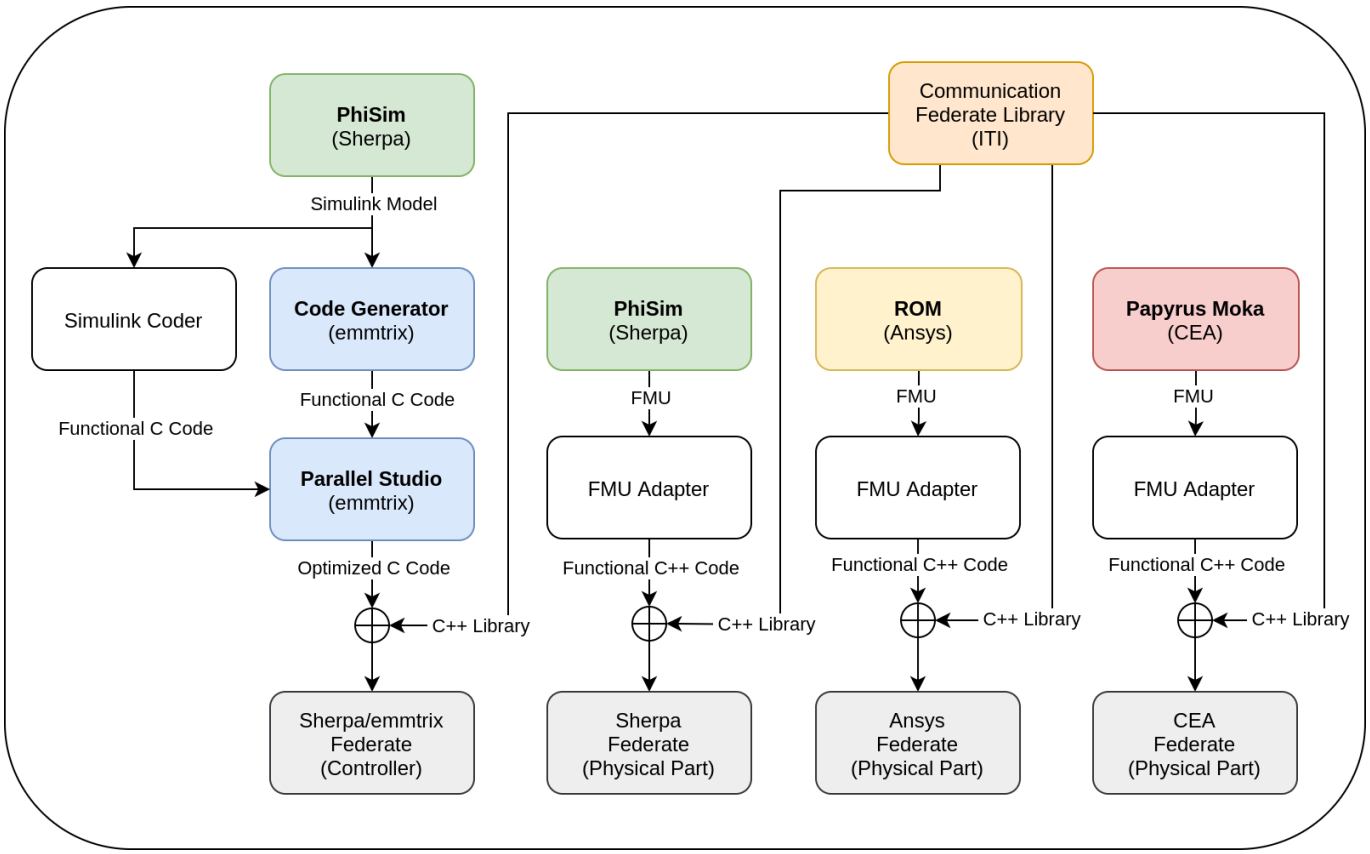

Figure 4 - Simulation Components Generation

2.1.3 Recommendations, guidelines & best practices

2.1.3.1 Guidelines about modelling a simulation component using FMU tools

The modelling of executable models with Papyrus-Moka is based on graphical fUML models and textual Alf code, enabling the model detailed semantics of the specifications. The fUML model includes structural and behavioural viewpoints. The structural model is defined using class diagrams, composite structure diagrams with the classes' specifications, attributes, and operations. The behavioural viewpoint is modelled via state machines and activity diagrams, which describes step-by-step actions to be performed to model the operation's functionality. A more detailed specification can be defined using Alf textual notations. Specific behaviours, parameters simulation, e.g. preconditions for tasks and resources, can also be described in the UML profile model. An Alf test script may encode the test inputs and add them to the original model. The Alf script can be generated by accessing the context menu of the behavioural model in the model explorer view of Papyrus. The execution of a model usually starts by executing a kind of "main" activity responsible for instantiating objects and stimulating them if needed (through signals or operation calls). Moka provides some facilities to generate these kinds of activities. To do so, right-click on an element of the model, then go to Moka / Modeling Utils / Generate Factory. By nature, a fUML model is agnostic about the semantics of time. However, extending the execution model with a designed scheduler to reroute its usual fUML execution flow is possible.

The functionality of exporting simulation models as standalone FMUs depends on the version of Matlab/Simulink. It is possible for Matlab R2020a and later versions to export Simulink models as a standalone FMU having a Simulink Compiler toolbox.

The following link refers to the procedure to follow for the creation of FMU according to the recommendations of MathWorks: Export Simulink Model to Standalone FMU (https://fr.mathworks.com/help/slcompiler/ug/simulinkfmuexample.html).

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

Sherpa experienced the generation of FMUs from PhiSim models using the Simulink compiler. These are some recommendations:

- Remove the top-level model and subsystem callbacks if they exist.
- Declaration of the source files in the "Solver / Code Generation / Custom Code" part of the model properties (see Figure 5).

Figure 5 - Simulink Code Generation configuration

For R2018a and prior, third-party products export Matlab/Simulink models to FMUs for co-simulation. Modelon, for example, provides an FMI Toolbox that offers the capability to export simulation models as FMUs that support co-simulation in FMI version 1.0 and 2.0 from MATLAB/Simulink environments, provided a Simulink Coder license is available.

The generated FMU contains the model implementation, as well as the metadata provided during export. Its version is independent of the version of the tool used. It is defined by the version of FMI standard chosen for the export.

The model must satisfy these conditions for exporting:

- The model must be in Normal or Accelerator simulation mode.
- Root input and output ports must be of numerical data type.

Suppose the co-simulation component is an FMU exported from Simulink. The local sample time for that FMU is the sample time of the original model. Only fixed-step solvers are supported.

2.1.3.2 Guidelines about generation of simulation models from Simulink

For the generation of C code from Simulink models, we need the following tools/toolboxes:

- Matlab / Simulink / Stateflow (Optional),
- Simulink Coder / Matlab Coder toolboxes,
- Embedded Coder.

To avoid issues with the code generation, the simulation models need some features:

- Have a fixed-step solver, preferably discrete,
- Respect a set of modelling rules concerning:
  - The used Simulink blocs,
  - The configuration of Simulink blocs (for example, sample time should be -1),
  - The architecture of the simulation model, with function-calls (recommended) or not,
  - The implementation of the Simulink blocs,

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

- Have a data dictionary that defines the parameters and variables of the model.

Given a Simulink model, we only need to right-click on the model to generate an s-function and build it.

2.1.3.3 Guidelines about code generation and optimization of simulation models

As mentioned before, the code optimisation is based on the emmtrix Code Generator (eCG) and Simulink tools. Then, the following points must be taken into account:

- *Input via Simulink model:* Simulink version 2016a or newer shall be used. If ordinary differential equations are needed, one of the following fixed-step solvers shall be used during simulation or code generation: ode1, ode2, ode4, ode5 or ode8.
- *Recommendations:* Parts of the model without solver can be independently optimised for code generation. Subsystems can be used to structure models (subsystem structure is preserved whenever feasible) MATLAB® scripts or C code within S-Functions can be used.
- *Validation:* Functional tests can be made use of to ensure the correctness of the generated code (automatable back-to-back tests for functional validation are supported).

2.1.3.4 Guidelines about adaptation of simulation models (FMU/FMI and Simulink) to be executed in the common HLA federation

The HLA distributed simulations require a complex architecture. Therefore, it is desirable to use middleware that can help speed up the integration and development of new simulation implementations by reusing existing modules. There are different approaches to reach it, in this case, the open source library "*OpenRTI*" have been selected as middleware. OpenRTI is a RTI library implementation based on C++ programming language.

The federate needs some features to run, such as the step period, the inputs and outputs and the step functional code. These features can be added to the federate using software design patterns based on HLA, where the user can add the characteristics required for simulation.

When the simulation unit is very complex or is developed in numerical computing environments such as Simulink or OpenModelica, it is convenient to package its functionality in simulation units that can be loaded dynamically. The FMI standard allows these actions, providing an interface to access simulation services. These functionalities can be solver engines, local variables, simulation step functions, etc. To adapt the FMU to the HLA environment, the user can read the model structure file that the FMU provides and generates the federate code to publish and subscribe to the data. A"FMU adapter" is provided in order to facilitate this integration FMU/HLA. It is also possible to use a federate template code that guides the user through the FMU and the federate interface implementation. The user can change some parameters in the template to adapt his FMU to the federate.

2.1.3.5 Guidelines about functional emulation using UNISIM

UNISIM-VP environment functionally emulates the controller in an executable binary form that runs on the real target. The executable binary is obtained from a source code compiled with a cross tool-chain for the target processor architecture and run-time. UNISIM-VP bridges the executable binary simulation and the Communication Federate Library using the non-invasive instrumentation capabilities of the UNISIM-VP environment. To seamlessly integrate into the Co-simulation environment, UNISIM-VP provides a configuration mechanism or uses an existing one to select which controller inputs and outputs to expose to the Publish-Subscribe Middleware. Indeed, at run-time, the UNISIM-VP environment samples controller inputs making these inputs available to the controller (e.g. in global variables) using the Federate Communication Library. Then, it runs the computing functions

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

of the controller executable binary and finally, sample and forward the controller outputs to the Communication Federate Library, following the execution flow of Figure 3.

2.1.3.6 Guidelines about ontologies.

The simulated entities of interest in the federates, can be semantically annotated (or come already with a semantic *id* if the origin design environment allows it) that identifies the type of entity being simulated and its properties. The application domain (e.g. Robot Arm, Rovers, Autonomous Vehicle, etc.) and the viewpoint (mechanical, logical, safety, etc.) define the ontological resources needed to integrate the simulated entities in a KB ecosystem. Part of the heterogeneity of the system can be lifted thanks to the abstract models ontologies provide, that is, even though there might be effort associated to properly adapt the content of the federates into the model, the model itself is independent from the implementation of the system. The co-simulation process and results can then be matched against constraints, coming from design, safety and/or planning. This enables property verification, consistency and complex querying of simulations of large systems of heterogeneous components.

2.1.4 Risks and Considerations

The federation models have to be defined to participate in the simulation. For this reason, the models of each federate, called SOM (simulation object model) **IEEE 1516-2010** , are described in a XML file. These must be compatible with the general specifications of the simulation, which are defined in the FOM (federation object model) **IEEE 1516-2010** , which is another XML file. Defining the FOM is an essential step for the RTI to manage communications. Once the simulation is configured through the FOM, it can be started, and federates can interact by publishing and subscribing to the necessary information.

The FMU models contains mainly a description of the inputs and outputs via an XML file. To further extend and ease the introduction inside an HLA-based environment more meta data may need to be added in the future. This addition can be defined freely using Vendor Annotations.

HLA-based simulation has similar problems as distributed computing. Often, the simulation nodes do not have the same performance and the workload could be not balanced, i.e. some compute nodes are overload while other compute nodes are left idle. In this case, the node with the worst response time will limits the simulation performance.

Other risks are related to communications where some nodes can be in a network with low bandwidth or many losses of messages. A federate can be waiting an indeterminate time for some messages, blocking the simulation. Furthermore, some companies could have a firewall that prevents communications outside the local network. In these scenarios, running a distributed simulation can be a complex task involving the company's systems administration team. Computer security must also be taken into account. An unauthorized person can exploit the communications between nodes in the distributed simulation to break into computer systems and access sensitive data. The system is addressing the time wise-collaboration of the different part of the whole simulation but does not address the post-treatment data location especially when we talk about distributed computing.

2.1.5 Additional resources

UNISIM-VP virtual platforms examples (simulator source code), Benchmarks (with source code and pre-built executable binaries), Tutorials (with videos) are available at http://unisim-vp.org/site/index.html

2.1.5.1 Documentation for the High Level Architecture (HLA) standard

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

The IEEE 1516-2010 standard was published in August 2010 by IEEE and is commonly known as HLA Evolved. It consists of:

- Framework and Rules:
  - https://standards.ieee.org/content/ieee-standards/en/standard/1516-2010.html
- Federate Interface Specification:
  - https://standards.ieee.org/content/ieee-standards/en/standard/1516_1-2010.html
- Object Model Template:
  - https://standards.ieee.org/content/ieee-standards/en/standard/1516_2-2010.html

Some books guide the construction of distributed simulation systems, with a particular focus on High Level Architecture, one of these is:

- Okan Topçu, Halit Oğuztüzün. "*Guide to Distributed Simulation with HLA (Simulation Foundations, Methods and Applications)*"

HLA-RTI implementations:

- **Pitch pRTI**. Pitch Technologies. Commercial license. https://pitchtechnologies.com/prti/
- **OpenRTI**. FligthGear Project. Non-commencial LGPL license. https://sourceforge.net/projects/openrti/

2.1.5.2 Documentation for Functional Mock-up Interface (FMI)

Releases and the latest development version of the specification are available on the FMI website:

- https://fmi-standard.org/

FMU-FMI integration in a C++ implementation

- **FMU SDK**. Free software development kit provided by Synopsys. It demonstrates basic use of Functional Mockup Units (FMUs) as defined by the Functional Mock-up Interface specifications. https://github.com/qtronic/fmusdk

2.1.5.3 Documentation for hybrid distributed simulation based on HLA and FMI

- Youssef Bouanan, Simon Gorecki, Judicael Ribault, Gregory Zacharewicz, Nicolas Perry. "*Including in HLA Federation Functional Mockup Units for Supporting Interoperability and Reusability in Distributed Simulation*". Summer Simulation Conference, Jul 2018, Bordeaux, France.
- Awais, Muhammad Usman, Peter Palensky, Atiyah Elsheikh, Edmund Widl, and Stifter Matthias. 2013. **"***The High Level Architecture RTI as a Master to the Functional Mock-up Interface Components***."** In Computing, Networking and Communications (ICNC), 2013 International Conference On, 315-320. IEEE

2.1.5.4 Modeling simulation components

A detailed tutorial about how to develop an executable model in Papyrus can be found following these links:

- Papyrus Documentation:
  - https://wiki.eclipse.org/Papyrus/UserGuide/ModelExecution#Designing_FMUs_with_Papyrus
- Papyrus Eclipse YouTube Channel: the series of video tutorial on Designing FMUS:
  - https://www.youtube.com/channel/UCxyPoBlZc_rKLS7_K2dtwYA
  - https://wiki.eclipse.org/images/1/1f/MOKA-FMI-ModellingDays-13-09-2016.pdf
- S. Guermazi, J. Tatibouet, A. Cuccuru, S. Dhouib, S. Gérard, et al. *Executable Modeling with fUML and Alf in papyrus: Tooling and experiments***.** 1st International Workshop on Executable

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

Modeling, EXE 2015, Sep 2015, Ottawa, Canada. pp.3-8.

2.1.5.5 Documentation for the generation of FMUs with Simulink

- User guide and example about how to generate FMUs from Simulink using Simulink compiler are available here:
  - https://fr.mathworks.com/help/slcompiler/gs/export-simulink-models-to-functional-mock-up-units.html
  - https://fr.mathworks.com/help/slcompiler/ug/simulinkfmuexample.html
- A detailed user guide for the use of the FMI toolbox is available here:
  - https://3pn0itd4zke1w5rdih3i1jfi-wpengine.netdna-ssl.com/wp-content/uploads/2018/08/UsersGuide-FMI-Toolbox-2.6.4.pdf
- A quick start with c code generation from Simulink models is given here
  - https://fr.mathworks.com/help/dsp/ug/generate-c-code-from-simulink-model.html

Details tutorial and examples can be found in Ansys TwinBuilder documentation and on internet:

- Resource Web Page: series of case studies
  - https://www.ansys.com/products/digital-twin/ansys-twin-builder
- Multiple Webinars organized either on-demand or on a regular basis
  - https://www.ansys.com/events/digital-twin-webinar-series
- Documentation for TwinBuilder
  - https://ansyshelp.ansys.com/account/secured?returnurl=/Views/Secured/Electronics/v211/en/Subsystems/TwinBuilder/TwinBuilder.htm

2.1.6 Future work

The Heterogeneous Co-Simulation cluster has components that are FMI compliant. From that, at least one or more federate will be required to adapt a FMI to HLA context, and FOM files must describe its interactions. According to this, we could automatically generate FOM files depending on the output and input of each FMU involved in the federation. It could also be possible to automatically generate a hybrid federate for each FMU needed in the distributed simulation.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

2.2 ITERATIVE CODE OPTIMIZATION

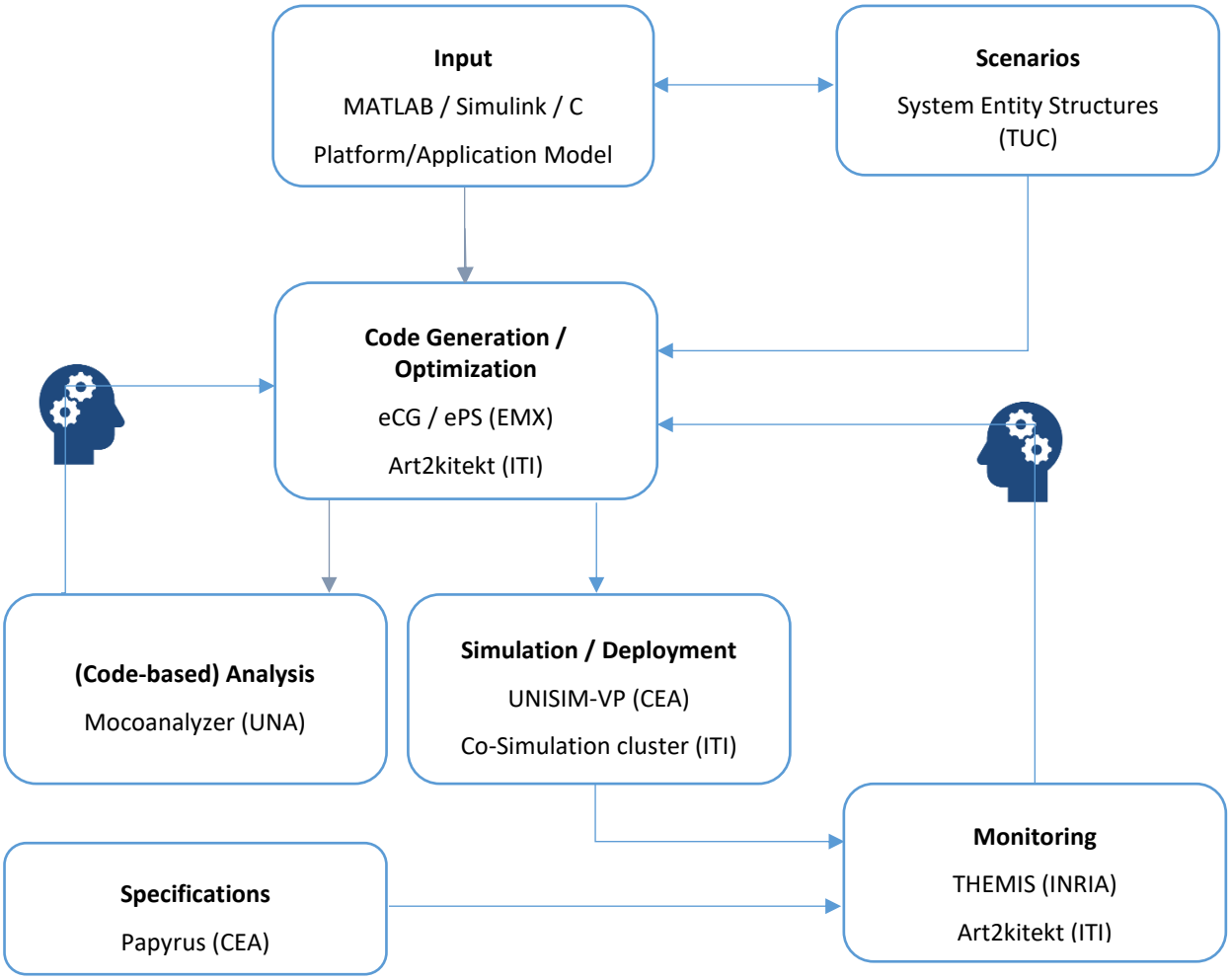

Figure 6: Iterative Code Optimization Toolchain

2.2.1 Purpose

Main goal of the cluster is an optimization of an input source code written in Simulink, MATLAB or C. This shall be achieved through an iterative execution of the optimized code on a simulator (see Figure 6). Static and dynamic code analysis shall be performed. They are responsible for the iterative nature of the algorithm. Additionally, scenarios complement the purpose by ensuring the code coverage and the use of all relevant application parts and by providing various input signals.

2.2.2 How is this going to be achieved

Tool interactions can be divided into three groups:

1. Static optimization: Code Generation / Optimization & (Code-based) Analysis
   Generated C-code shall be put into a loop of static code analysis where after each iteration improved C-code shall be obtained and used for either another loop iteration or the input into the dynamic optimization part.

2. Dynamic optimization: Code Generation / Optimization & Simulation / Deployment & Monitoring

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

Generated C-code shall be put into a loop of dynamic code analysis where after each iteration monitoring shall be used for validation. Monitoring shall decide whether the C-code is sufficiently improved and consequently whether there is a need for another loop iteration. In this sense monitoring shall be used as a validation step.

3. Definition of scenarios: Code Generation / Optimization & Scenarios
   Scenarios shall give meaning to the cluster by providing predefined input and ensuring that the correct C-code is executed

In both static and dynamic optimizations, the improvement of the C-code is reflected through:

- Smaller runtime
- Less memory requirements
- Higher functional safety

Every tool interaction (arrow in diagram) within the workflow is described separately.

2.2.2.1 Code Generation / Optimization => (Code-based) Analysis

The emmtrix tools output C code, potentially with optimizations from previous iterations. The C code is generated according to the C standard and can by further processed afterwards.

Art2kitekt generates the structure of the application in C code, based on the platform (hardware) and application (software) models, and by analysing the time constraints set for the system. The generator creates the code taking into account the different process that the system contains. It implements the application skeleton with threads of execution with their corresponding priorities. Different tools, such as eCG from emmtrix or Simulink, can merge their functional code with the generated thread structure. Afterwards, C analysers can inspect the created code and optimises its structure.

2.2.2.2 (Code-based) Analysis => Code Generation / Optimization

Currently, the EMX tools can process additional constraints in specific XML and Json files, but with the help of the graphical user interface, more fine granular optimizations are possible. The supported optimizations can be placed in two distinct categories: parallelization (assignment to specific processing elements of the target architecture) and code transformations that can change memory layouts, loops or how expressions should look like.

The results of the static code analysis allow the engineer to know if the code generated by art2kitekt meets the necessary quality and corrects potential vulnerabilities.

Currently, UnA's tool MoCoAnalyzer provides an iterative analysis workflow that starts during design phase by analysing model components with code artifacts. Therefore, model-based analyses approaches are used with input from diverse scenarios which are based on the CPS4EU meta-model. The resulting outcomes are used continuously during development phase to allow a final C code optimization. MoCoAnalyzer output depends on the used input (scenarios and data) and starting point in the analysis workflow.

CEA's tool UNISIM-VP provides profiling analysis of executable binaries against any metric (execution time estimation, number of executed instructions, memory and cache usage statistics) at instruction level, and at source code statement and function levels when these executable binaries comes with embedded DWARF debugging information. UNISIM-VP will export these profiling data as either JSON or XML data formats. CEA's tool Frama-C together with UNISIM-VP can provide assertion-based verification of executable binaries against security issues provided that these executable binaries comes with embedded DWARF debugging information. Using this combination of formal and concrete analysis tool, the security expert can verify, debug and visualize security properties on executable codes.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

2.2.2.3 Code Generation / Optimization => Simulation / Deployment

The generated C code is optimized for the selected target platform and can be compiled by any compatible compiler in order to be executed in the simulation. The EMX tools allow the generation of special commands like pragmas or extra function calls to mark code for profiling or other things.

Art2kitekt code generation service can be configured to generate POSIX compliant code, as well as RTEMS compliant. These applications can be deployed for different platforms such as x86, ARM or Leon3.

About co-simulation, verifying the functionality in a simulated environment should be a previous step before deploying the code on the final platform. Section 2.1 defines a Heterogeneous Co-Simulation cluster based on the High Level Architecture (HLA) and the guidelines to integrate generated code from models into a distributed simulation.

2.2.2.4 Simulation / Deployment => Monitoring

To simulate the generated code is a very appropriate method to check the functionality of the application. This code must contain the monitors to generate the traces, that provide evidence of compliance with the functional requirements. Section 2.1 describes how the distributed co-simulation is coordinated and how the different components in the simulation are executed.

The code generated by art2kitekt contains the monitors, which provides the monitoring traces that are stored during the execution or simulation of the application. They are lists of timestamped actions, such as the beginning of an activity or the change of context between two threads in a processor. Once the system simulation has finished, the information can be provided to the monitoring listener service at runtime or, depending on the characteristics of the system and time constraints, stored to send it.

THEMIS uses a black-box approach to monitoring and finding bugs. THEMIS can be applied here probably with Papyrus PSCS, receiving traces that contain sufficient information to evaluate the specifications. The specifications should be high-level specifications spanning over multiple components of the entire system. The traces need to refer to the atomic propositions describing the state of the system. Indeed, as THEMIS uses time-stamped black-box approach where the actual system states and events are abstracted into the trace, instrumentation should be done collaboratively with system designers in order to ensure that sufficient information will be available during simulation the deployment. Moreover, we should be able to map events in traces to components in the architecture of the monitored system.

CEA's tool UNISIM-VP provides non-invasive instrumentation of executable binaries, in a way that it is possible to observe the binary code execution without modifying the code and the timing, and trigger programmed actions (online) such as recording the software activity for monitoring purposes, e.g. state of symbolic variables at breakpoints provided DWARF debugging information is available.

2.2.2.5 Monitoring => Code Generation / Optimization

Besides the optimizations already mentioned, additional timing and memory information could be used to improve the parallelization of the application onto the available cores.

The art2kitekt monitoring service can obtain the worst execution times for the application activities after processing the traces captured in the execution. Art2kitekt can use these times to reallocate the threads' execution and priorities before the generation of code. Similarly, emmtrix tools can use this timing information to configure the code optimisation.

Themis will generate verdicts for each monitored specification that can be formalized in LTL. These verdicts indicate whether the execution trace satisfies the specification or not, or whether it is inconclusive.

2.2.2.6 Scenarios => Code Generation / Optimization

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

The EMX tools will generate the code as specified by the scenarios from TUC. This should improve the code coverage by providing important input data as well as allows the verification of the results of the execution of the applications.

TUC responsibility is two-fold in this cluster. TUC's Scenario Definition Language could be used to model all possible combinations of test cases for EMX tool. Its main goal in this cluster is to act as an assistance to set up a functional test bench. This includes not only the tool's configuration but also setting up external factors like its interaction with other tools. This can ensure the correctness of the generated code. TUC's SDL is also expected to be used for code coverage to ensure the important combinations of test cases are determined and executed. TUC will also provide input code for this cluster from "collaborative lifting" use case in WP8. The code will be generated C code from MATLAB/ Simulink model of the crane and the detection of the object to be lifted and its position estimation by a drone.

2.2.3 Recommendations, guidelines & best practices

2.2.3.1 CEA Guidelines

- Input:
  - Executable binaries obtained from a source code compiled with a cross tool-chain for the target processor architecture and runtime, preferably crafted with embedded DWARF debugging information
- Code Generation / Optimization:
  - For Code-Analysis:
    - Simulation variables names to sample must be provided for program profiling, e.g. execution time estimation, number of executed instructions, memory and cache usage statistics
    - An annotated C source code with ASCL (Abstract C specification language) assertions must be provided to apply safety/security checks with UNISIM-VP used in conjunction with Frama-C.
  - For Simulation / Deployment:
    - The target machine must be defined: e.g. processor instruction set architecture
    - The software architecture of run-time and operating system should be documented to leverage on hardware/software interface (e.g. run-time APIs, OS ABI, Hardware abstraction layer interfaces) to cut the simulation budget, speed-up simulation and ease deployment
- Monitoring:
  - For Simulation / Deployment: the locations of points of interest (breakpoints) where to install hooks that observe the state of symbolic variables under monitoring

2.2.3.2 EMX Guidelines

- Input:
  - MATLAB and C input files shall be written according to the standards;
  - Code shall be statically analyzable (dynamic sizes, memory management etc. is difficult to analyze).
- Scenarios:
  - Proper scenario coverage shall ensure that all parts of the code become executed;
  - Scenarios shall be compatible with the input.
- Input from analysis:
  - Code-based analysis shall provide information on parts of the code that are to be changed, considering:
    - Security
    - Memory usage

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

- Monitoring:
  - Instrumentation of the source code shall be automated;
  - Information about the coverage of the specification shall show, where the code can be improved. In order to achieve this, the monitoring results shall have a unique reference to the appropriate source code.

2.2.3.3 INRIA Guidelines

- Input:
  - Traces: THEMIS expects to receive the components behavior abstracted as observations. For each component of the system, it expects a file/stream where each line represents a timestamp containing atomic propositions and their Boolean values. For example, for component 1 the file may contain 2 lines: task1_entered:t, task1_ended:t. The traces can be in text files or streams over network sockets. The instrumentation of the code to emit the observations needs to be handled by each tool/component.
  - Specification
    - The specification can be define using automata, LTL, or other formalisms. Themis expects multiple specification files written is a specific XML format.
- Output:
  - Verdicts: emit a verdict in a truth domain that indicate the compliance of the system to the specification(s).

2.2.3.4 ITI Guidelines

- Input:
  - Platform and application models (hardware and software) shall be defined using art2kitekt editor to generate the thread structure code through the art2kitekt code generator service.
- Code Generation:
  - Functional code can be merged with the thread structure code in order to obtain a functional time-constrained application.
  - The temporal constrains shall be defined in the deployment and execution model in art2kitekt.
  - Generating, on the one hand, the code with the thread structure and on the other, the functional code is a methodology that helps to isolate future changes in the temporal configuration of the application, without the need to modify the functionality.
  - The art2kitekt code generator service automatically includes the software monitors in the generated code, aiming to observe the temporal behavior of the application during its execution.
- Monitoring:
  - Software monitors shall generate the monitoring traces when the application execution passes through them. These monitoring traces includes the timestamp of the execution and the information to locate the execution point.
  - Two methods are usually performed to collect the monitoring traces on runtime:
    - Continuous monitoring: Temporal traces captured by the software monitors are provided using an external interface during the execution of the application. The necessary time to read and send these traces shall be minor than the available time reserved for this task.
    - Snapshot monitoring: Temporal traces are captured during the snapshot duration and stored in a buffer. After the snapshot, the buffer is sent step by step during the execution of an idle task through an external interface.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

    - The monitoring traces shall be processed to obtain the temporal behaviour of the application, which is defined by:
        - The execution time and response time of the different tasks and threads of the application
        - The occupation of each processor and what task it has at all times.
        - The periodicity and jitter of these tasks.
        - The sporadic execution of asynchronous tasks, such as interrupts, and how they affect the execution of periodic tasks.
- Input from monitoring:
    - The observed behaviour in monitoring shall provide the necessary information to analyse if the application is not feasible, and verify if one or more of the temporal requirements are not fulfilled. Note that the observed behaviour cannot be used to ensure that the application is feasible in all cases. Not all possible cases can be observed during execution.
    - The observed behaviour can be used to understand the application better and redesign the temporal configuration of the initial models.

2.2.3.5 UnA Guidelines

- Input:
    - Required are XMI files for scenario description and according C code files
    - XMI input files shall be written according to syntax standards
    - C Code shall be statically analyzable
    - Scenario safety and security requirements shall be provided
- Scenarios:
    - Scenario models need to be conformed to CPS4EU meta model (WP1)
    - Scenarios must be depicted with MoCoAnalyzer Modeling Editor
- Workflow:
    - First, scenarios shall be analyzed on model level to prepare input information for code-based analyses
    - Code-based analyses shall provide information about security vulnerabilities

2.2.4 Risks and Considerations

- Bad communication / interaction between the tools
    - Content of one tool's output not compatible as the input of another tool
    - Not generic usage of all optimization steps
    - Have to provide diverse interfaces
- Not improving upon the current solution
- Making code worse
    - Increasing complexity of the code
    - Solution within dynamic optimization: monitoring during simulation
- Term iteration not representative for static code analysis
    - Iteration might end up in one cycle
- Feasibility of formalizing the specification into LTL formulae.
- Possibility of instrumenting the target simulation of the system to retrieve sufficient information.
- Availability of traces.

2.2.5 Future work

There are several major points that shall be considered for the future work:

- Automating the cluster

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

    - Decreasing manual steps needed
    - Using AI optimization to improve the steps
    - Making the cluster user-friendly
- Making requirements more detailed, specific and clear
- Optimization viewpoints can be added or updated
- Introducing Simulation as a service
    - at least for the dynamic optimization loop

2.3 SCENARIO BASED SIMULATION

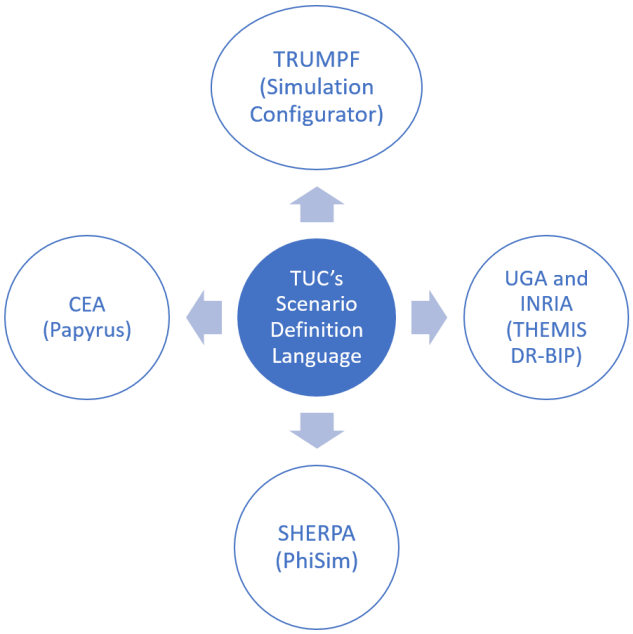

Figure 7: Scenario-based simulation

2.3.1 Purpose

Scenarios are an essential part of the whole simulation engineering process. Due to the complex nature of Cyber-Physical Systems (CPS), scenarios are being used for simulation-based verification as a cost-effective method. Scenario-based testing is already being used extensively in automated vehicles, especially in validating Automated Driving Systems (ADS). Tools and standards such as OpenScenario, OpenDrive and OpenCRG illustrate the effort in this direction. In the last few years, aviation has started using scenarios, and a few working groups are developing a standard scenario definition language for aviation. The scenario definition language (SDL) used by TUC is one such example.

The scenario-based simulation (see Figure 7) cluster aims to demonstrate the efficiency of using scenarios in multiple domains of cyber-physical systems ranging from shop floor simulation to crane simulation. Scenario Definition Languages can also be used in conjunction with Domain-Specific Languages (DSL) and monitoring tools. The process of scenario development can be complex and time-consuming. Using simple constructs to build and express ontologies, TUC aims that its SDL will help simplify the process of scenario development and make it accessible for different CPS to use scenarios for its safety assessment. The standard XML format supported by the tool allows sharing scenarios among the stakeholders, and with parsing, it can be used with other applications like simulator configuration.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

2.3.2 How is this achieved

At the heart of TUC's Scenario Definition Language (SDL) lies ontology-based domain modelling. An ontology describes the concepts and relationships important in a particular domain in an easily understood manner while maintaining the ability to be machine interpretable. This bridges the gap between people and systems and can be used as a starting point for further development as a domain expands, or the ontology embraces new or additional concepts.

TUC's SDL uses a meta-model called System Entity Structures (SES). The SES is a formal ontology framework, axiomatically defined, to represent the elements of a system (or world) and their relationships hierarchically. SES uses four main elements and six axioms to model complex CPS in a simplified manner. The main idea when defining scenarios is to identify the environment (static and dynamic), events and simulation termination. Once all possible scenarios are modelled in the SES domain model, selecting a particular scenario is defined by choosing a specific configuration through a process call pruning. This resultant structure is in the form of a decision free tree called a Pruned Entity Structure (PES). A PES represents a particular scenario in XML format that can configure a simulation and assess the system for safety. The following diagram in Figure 8 illustrates the workflow.

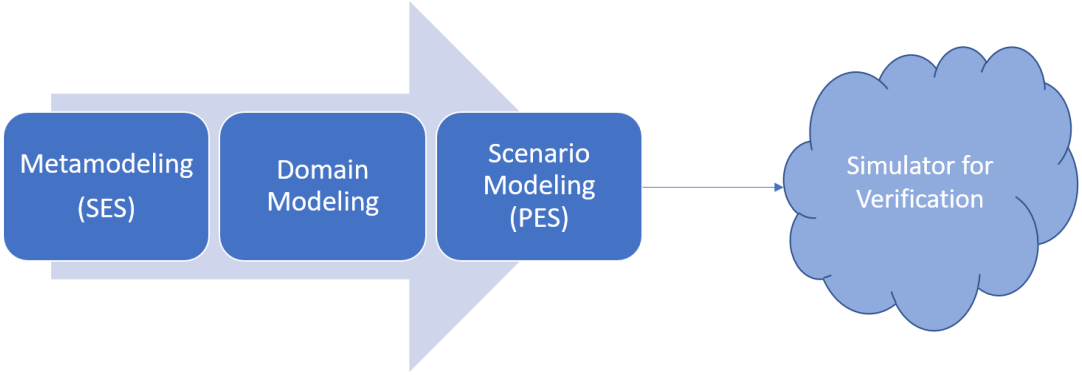

Figure 8: Model verification workflow

2.3.2.1 Interactions with Partner's tools: TRUMPF ⇔ TUC

TRUMPF uses a tool called simulation configurator, which will be executed by the machines during the simulation run. These xml files contain the number and routing of each single part through the production. Therefore, the products and all contained parts are configured within the configurator tool. Then the required number of parts is ordered and the machines for the single production steps are defined. The simulation software used is AnyLogic. TUC's SDL will model the necessary configuration of the welding, bending and cutting machines as well as storage units and transportation options to be used in the simulation as well external factors such its interaction with other static components, and events that take place on the machines. THE SDL will load the configuration such as initial state of the machines and the static objects and will configure the timelines of different events affecting the state of the machines.

2.3.2.2 Interactions with Partner's tools: TUC ⇔ CEA

There are two possible use cases where TUC and CEA will look to collaborate.

Use case 1: Safe Drone Navigation

The use case is about ensuring the safe navigation of a drone in a 3D space by tracking trajectories between waypoints computed by a motion planner (see Figure 9). A controller provides a sequence of control actions that regulate the state of the drone as a function of time. Given the next waypoint, the appropriate controller is used to track the reference trajectory.

In the use case, we would like the drone to avoid deviating too much from the reference trajectory provided by the planner so that the drone is in a safe operating state. The added value is to provide

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

extra safety for drones operating in critical environments (like a hospital, a shared workspace with human). In that sense, we want to assess the safety and the performance of an advanced controller (ML-based) by measuring the cross-track error (CTE), i.e. the distance between the drone and the closest point in the path (between two known WPs). Figure 9, illustrates examples of drone navigation with advanced controllers, where the reference trajectory between waypoints (red) and the effect of an advanced controller in the drone's position (blue) is plotted to have a graphical representation of the controller performance.

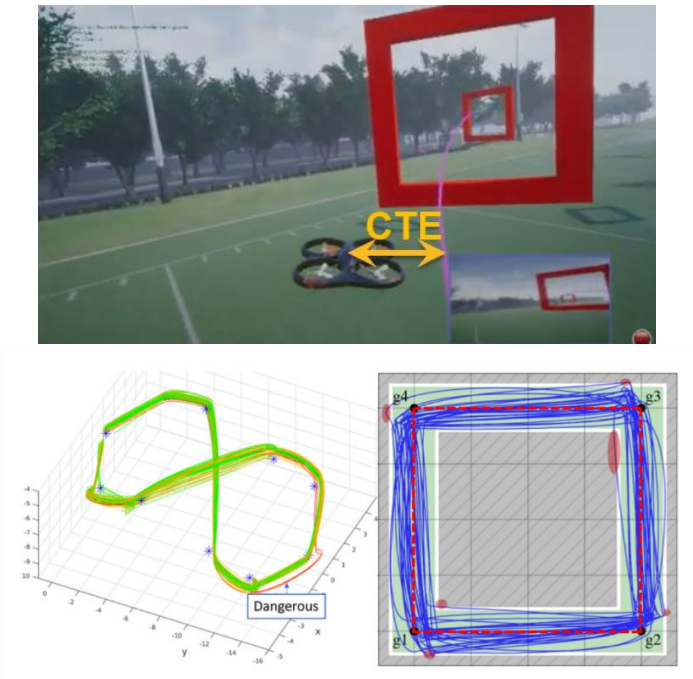

Figure 9: Examples of Navigation with Advanced Controllers and their Deviation from the Reference Trajectory

To assess the safety and performance of the drone, we would like to define a list of critical situations that cover some different drone missions (go home, go to next waypoint, land, go back to last good, hovering) in different critical conditions (battery checking, loss of signal from remote control, loss of GPS, object avoidance, payload checking, etc.). Table 1 shows a list of critical situations and possible critical conditions (not exhaustive) that we would like to describe in a standardized format. TUC's SDL can help model all the situations involving different kinds of missions in form of scenarios. This will help in identifying critical scenarios using the combination of these drone missions.

Table 1: Example of critical scenarios for drone safe navigation

Mission	Detailed description	Critical conditions
Go home, go to first waypoint	Going to recorded position (mostly if GPS available)	Battery Condition Remote controller lost GPS lost Object avoidance Payload checking (+ Camera function check, image quality check)
Hovering	Specify conditions to hovering for autonomous actions (only while trying to avoid critical condition) then continue	Battery condition, wind, …
Land	Immediate landing (in some systems with down camera) with exploring landing place. If lands, inform pilot by send email/SMS or if possibly make	Camera lost, bad environmental conditions, GPS lost, …

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

	journey to home position	
Land in designated places	Defined places for landing, in critical territories to avoid safety	Camera lost, Battery condition, object avoidance, GPS lost…

Use case 2: Cooperative drone in a warehousing space

Collaboration and mission planning in a heterogeneous environment involving multiple agents (robots or humans) present multiple challenges. They are mainly due to the complexity of the system and the high heterogeneity introduced by different vendors and systems involved in a distributed manner. Warehousing deals with the arrangement and transportation of physical assets. When a business process involves multiple warehouses and multiple agents performing this mission, it requires that the agents communicate in the same and unambiguous terms, and the systems involved in the operations analysis also need to integrate this knowledge. Aided by ontologies and knowledge-based systems, we link system design to the mission and the ongoing operation in a multi-agent, multi-warehouse environment. The scenario consists of (at least 2) warehouses, robot arms picking the packages from the shelves, UAVs transporting them to a lift-up point, and drones transporting them to delivery points in the second warehouse. If a change in the mission conditions occurs (e.g. volume, weight or number of packages to be transported), we need to evaluate if the current set of agents is capable of accomplishing the mission, if new agents can be deployed compliant with the mission and whether the capabilities of the components and agents conform with the requirements of the mission. The above analysis permits to early determine possible conflicts of a system (in this case a Drone) being design, when deployed in the working ecosystem. Knowledge-based system design enforces correct-by-design models by analyzing the realized model and establishing consistency with the design constraints. These constraints can be extended by mission-specific conditions like the interaction with other agents, the abilities required, and safety requirements.

Specifically, systems designed with UML / SysML are translated and mapped to standardized and recommended domain-specific ontologies to enable interaction and analysis. The intended analysis about consistency and capabilities are done using reasoners with different scopes: PDS Based (SPARQL, pattern latching) and OWL Based (DL-reasoners and SWRL). This analysis is returned as a feedback to the system designer so that: the compliance with the (selected) mission constraints is established, the conflicts are identified and explained, and alternatives (either to the design or the mission) are proposed so the designed system and the mission are consistent. This use case and tooling involves AI in two levels: at the level of the components of the systems and their correct integration into the systems via the capabilities and properties of the component, and the level of the analysis and reasoning about the successful integration of the system in the larger ecosystem.

TUC's scenario definition language as depicted in the following Figure 10 will bring together all the different models, whether mapped through system design languages like UML/SysML or official ontologies, under a common definition language. The SES in TUC's SDL will define all the model's characteristics, their interactions (with each other or other entities), and their associated constraints. Using the constraints and specific parameters, models can be configured and initialized using other tools or executed in a simulator like Papyrus.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

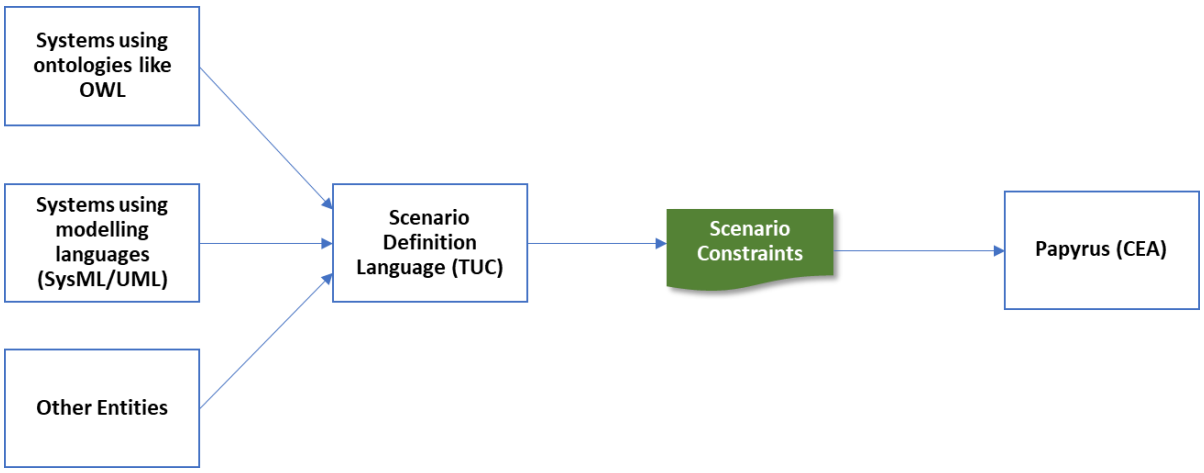

Figure 10: Integration of System model into Scenario Definition Language and Scenario constraints.

2.3.2.3 Interactions with Partner's tools: TUC ⇔ Sherpa

Sherpa Engineering provides a tool-based methodology for the design, the evaluation and the validation of cyber-physical systems (CPS). Our engineering offer is based on a software suite with:

- *PhiSystem*: A UML/SysML based MBSE tool for CPS description (Requirement organization, System definition, Test definition and traceability). Scenarios are specified using UML sequence diagrams involving the system and its environment.
- *PhiSim*: A Simulink based platform for CPS evaluation. *PhiSim* is a set of libraries of system level models which are specialized by applicative domain. The scenarios are defined giving a configuration to the system environments.

Sherpa works with these tools in different domains. For automotive domain for example, a set of simulation models for autonomous vehicle, ADAS, powertrain, energy management for electric and hybrid vehicle are tested and can be used for the validation of the TUC and Sherpa collaboration workflow.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

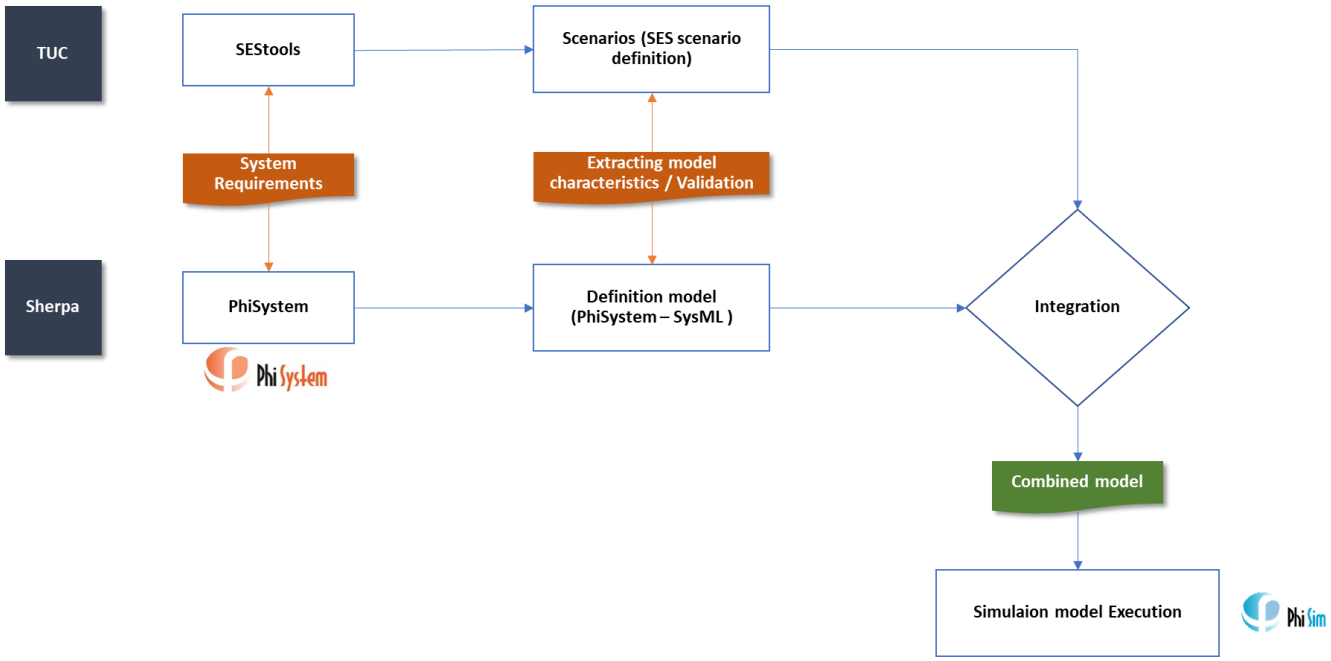

Figure 11: Integration and interaction of SEStools and PhiSystem.

The collaboration between Sherpa and TUC is scenario based where TUC offers a formalized way for scenarios definition using SES (see Figure 11). Sherpa and TUC have two potential interactions:

- At design level with an interaction between SysML and the SES meta-model/domain modelling. In this case we can extract specific system/domain characteristics from the description model and use then for the definition of high-level SES scenarios.
- At simulation level where the scenarios defined with the SES tool can be used to configure the simulator. Once the SES scenario is integrated with the simulator, the simulation can be launched, results can be monitored and verified.

In case SES is used at system level (SES/SysML interaction) and at simulation level (SES/Simulink interaction), the workflow permits also to validate the simulation model against the description model and then ensure their coherence, something very important in the design of cyber physical system.

2.3.2.4 Interactions with Partner's tools: TUC ⇔UGA

UGA and INRIA propose to integrate the BIP toolset and Themis tool for modeling and monitoring a given system or protocol starting from scenarios generated by SEStools from TUC in order to ensure the correctness.

The integration strategy proposed between DR-BIP and THEMIS is shown in Figure 12. Grey components designate input for the corresponding tools while orange components designate output usable by other tools. Purple components are library/code extensions to existing components for extending functionality to meet the integration goals. Two integration modes are possible:

- Offline integration allows to use the output from one tool then transforms it to compatible input for another tool, tools are then run sequentially in a chain. UGA and INRIA already implemented trace converters that converts a DR-BIP trace output by a simulated system into a THEMIS trace that can be used for monitoring.
- Online integration allows the program to interact directly with the running system to read or modify its state. Planned integration between UGA and INRIA allows monitors to observe the DR-BIP system state during simulation and act upon it to possibly correct its behavior.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

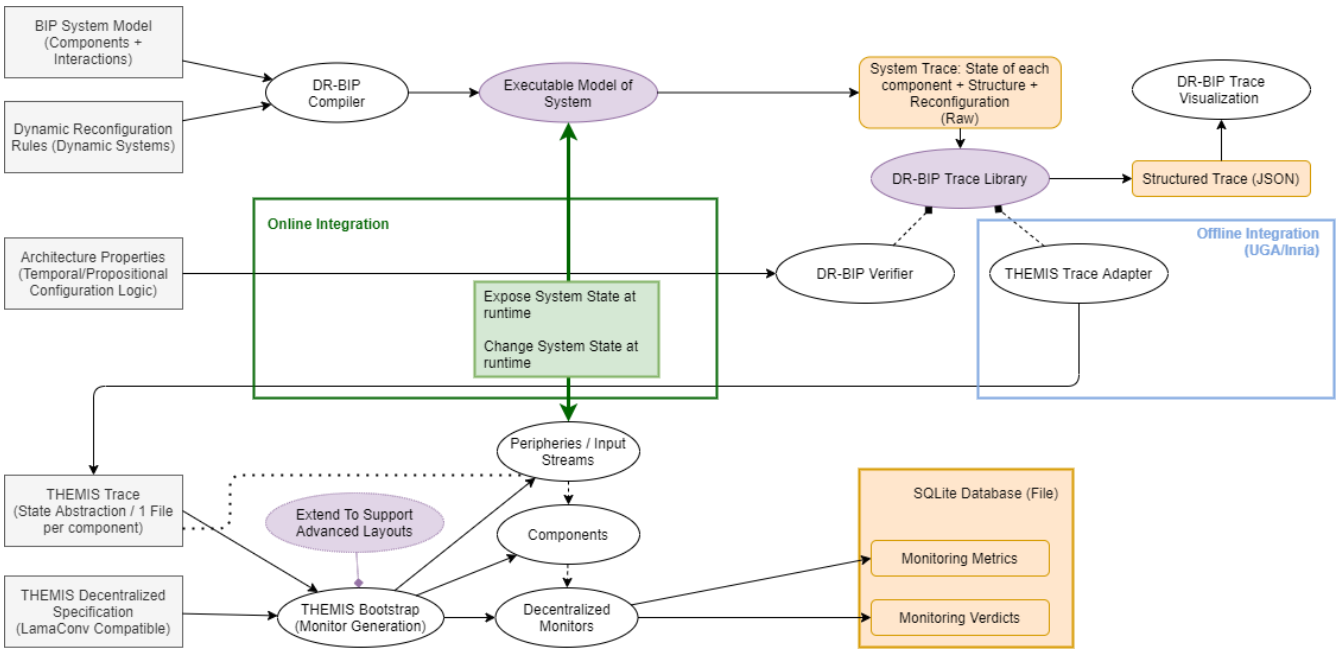

Figure 12: Offline and online integration between DR-BIP and THEMIS tools

In the case of WIKA use case, as shown in Figure 13, we integrate with SEStools from TUC to feed DR-BIP specific scenarios. Starting from the scenarios generated by SEStools, DR-BIP can be used to specify high-level model scenarios for a given protocol and output useful traces that can be monitored and verified by THEMIS. Furthermore, since we are given multiple sensor data from WIKA and TUC drone(s), combined with generated trajectories from WIKA, there needs to be agreement between sensors which is not trivial in addition to computation to match trajectories, the protocol for agreement along with the threshold for errors can be modeled in DR-BIP after which it generates a trace that can be monitored by THEMIS. Optionally, partners can provide the traces immediately to THEMIS for monitoring in the case the traces are simple. THEMIS utilizes state-based decentralized information with discrete time, for each component a file must be provided where each line represents a timestamp, the line contains atomic propositions and their Boolean values at that time (e.g.: crane1_moving:t, crane2_moving:f, sensors_correlation_safe:t). Additional input to THEMIS can be provided in terms of sensors and thresholds when needed, as THEMIS has already been used to monitor smart apartments with various sensors. We note that possibly, input can be directly fed to THEMIS monitors by writing a custom “THEMIS Bootstrap” component and “Peripheries” which are input streams for monitors.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

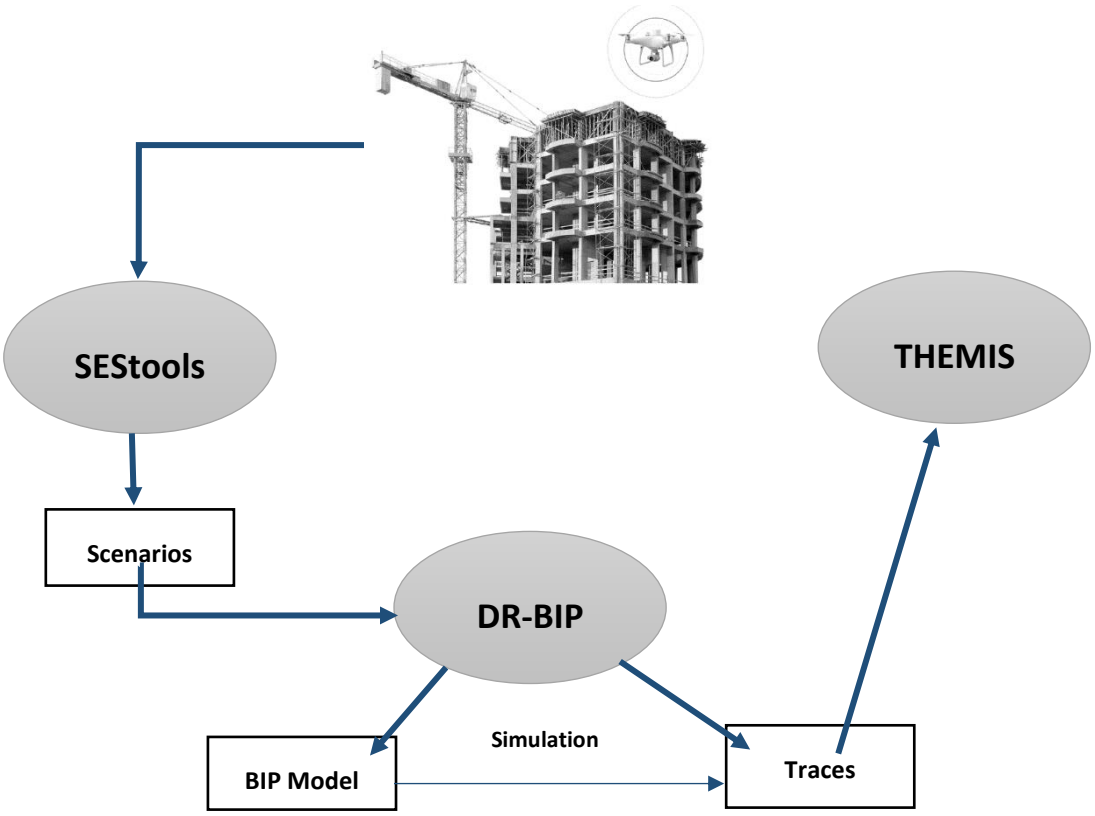

Figure 13: SEStools, DR-BIP and THEMIS tools for modeling, simulating and monitoring WIKA protocols.

2.3.3 Recommendations, guidelines & best practices

A strong collaboration is needed between the stakeholders during the initial stages. The partners must define possible scenarios for their particular use case, which must be decomposed and modelled using the SDL by TUC. The modelled domain of the scenarios needs to be parameterized, and appropriate configurations need to be discussed with the simulation expert. The pruned scenario has to be mapped with the simulation configuration.

2.3.4 Risks and considerations

- Needs strong time commitment upfront and exchange of information to model scenarios and their configurations
- The scenario elicitation method is primarily knowledge-driven which relies on experts from a particular domain to provide input to the ontology. No data or subsequent analysis is used to assist this process. This might lead to gaps in knowledge. However this effect is not anticipated to be big for small sue cases.
- Scenarios may not be directly mappable as configurations to simulators or other similar tools. Interface scripts might be needed to assist this process.
- The tool does not have a big support community. TUC with other research groups need to handle possible bugs.

2.3.5 Additionnal resources

- Papers on SES [6] [7] [17] [18] [19]
- Applications of TUC's SDL in AVES flight simulator [8] and Aerial refueling [9]

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

2.3.6 Future work

- Work on scalability by building complex scenarios with partners
- TUC's SDL needs to be a part of a complete framework with support for multiple simulators and monitoring feedback. Can use other partners' tool to complete this framework. Monitoring and feedback can lead to interesting applications later.
- Automated integration of various simulators (Simulink, Phisim, Anylogic, Gazebo) with the tool. The tool should launch the simulator directly.

2.4 MODELLING AND ANALYSIS OF AI-BASED SYSTEMS

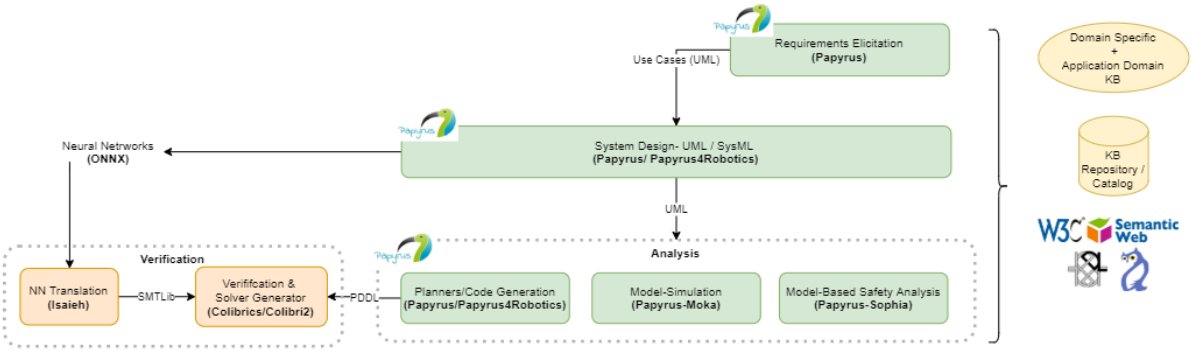

Figure 14: AI-Based CPS-Systems Toolchain

2.4.1 Purpose

This cluster focuses on the development of a **toolchain to develop trustworthy AI-based systems**. This is achieved thanks to the seamless integration of the tool's services during the requirements elicitation, design, analysis and validation of the CPS, which considers AI-based pre-integrated components. AI technologies are also present in the integration and exchange process itself, by attaching unambiguous and well-established semantics to the data and services exchanged. The knowledge behind the integration is encoded into a machine-readable formalism (OWL) that permits its analysis via reasoning services, for example to check the consistency of the system.

The AI-Based CPS-Systems Toolchain cluster mainly considers interaction between intra-CEA tools (Figure 14). Nevertheless, this interaction is tool agnostic (the elements, services and models exchanged follow high level specifications that the CEA tools implement) and can be extended to other partners tools since there exist compatibility and coherence between services/tools inside the CEA ecosystem and their counterparts in external partners: UML/SysML for system modeling, OWL/RDF for knowledge representation, MLX files for simulation, etc.

Our objective is to define a methodology to support requirements elicitation, system design, analysis, and formal verification of AI-based systems, supported by an AI-driven approach.

The intuition behind the approach is that each tool focuses on its intended task while enabling the results to be understood by other tools and actors. The purpose of the semantic integration using ontologies is to ensure coherence, propose relevant alternative models and reduce design and implementation time & costs.

At the same time, we aim to expose all the above information to high level analysis via complex queries and reasoning. A more complete view and details on the cluster structure can be found in deliverable 5.3.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

2.4.2 How is this achieved

In this cluster System Engineering, Safety Analysis, Formal Verification and Planning interact to deliver trustworthy AI-based systems.

- **Knowledge Driven Engineering** provides the building blocks and the base functionality/services that a system possesses. The components in the system (included those carrying AI-based functions) define the system's capabilities. In order to deliver trustworthy AI-systems, these capabilities need to be compliant with the constraints imposed by their intended usage. A system can be deployed in different environments and integrated in unforeseen ways with new pieces of technology. To ensure the integration is compatible and that the provided services and behavior of the system remain within acceptable parameters, it is required to automatically understand these capabilities and their requirements. By integrating *knowledge* in the engineering process in a standardized manner (knowledge based engineering), we expose the relevant capabilities, properties and parameters, so that other tools can provide their evaluation and analysis to assist and enhance the design process. This is achieved by the annotation of the entities in the UML design with the corresponding recommended domain specific vocabularies (ontologies).
- **Safety Analysis** helps define safety objectives and requirements to design the AI based system. The safety objectives also serve to derive safety rules and constraints for run-time monitoring of the system in order to detect during the operational phase any deviation from normal behavior and apply necessary corrective actions.
  Planning is essential for CPS to fulfill their goals in open environments autonomously. Planning is the activity of producing a plan of actions to reach a goal from an initial state using a given declarative domain model, which describes the system, the objects it can interact with, the system's possible actions and the associated constraints. The model-based representations make planning systems transparent, i.e., the process by which the decisions are made can be understood by human designers. Hence planners have the potential to contribute to achieve explainable and trusted decision-making for autonomous CPS.
- **Verification of NN properties** covers global verification (so called formal verification of properties) as well as local property verification, otherwise known as property-based testing, where tests are generated automatically according to properties about the domain of applications. This two-pronged approach, which has been demonstrating its effectiveness in "traditional" (*i.e.,* non-AI) software for decades, remains relevant for the paradigm shift brought by NN.

2.4.3 Recommendations, guidelines & best practices

2.4.3.1 Knowledge driven engineering

In order to seamlessly integrate the tools, it is required that they speak the same language and in the same terms, i.e. shared semantics. To unambiguously identify a component in a knowledge base and elicitate its properties, first it is necessary to have clearly defined the **context, a viewpoint and the objectives** of the system being designed. The final properties of an AI-pre-integrated component , for example a face recognition device, will depend on the input sensors (camera resolution, frame rate, NN accuracy, nominal conditions, etc.) which are logical properties of the camera, as opposed to the weight, volume, and current necessary, which are at the mechanical and electrical level. Likewise the considerations for an acceptable & optimal device integration and deployment environment, depend on **the mission** of the system and **the environment** where it is being deployed. Once the components required to interact within the system, their properties and the viewpoint of interest are defined, we can proceed to apply the proposed toolchain.

The context, viewpoint along with the *systems purpose*, which determine the *necessary* and *sufficient* concepts to consider, i.e. the ontologies, which in turn will define the underlying vocabulary. The reasoning based analysis (i.e. consistency among the tools impact on the system) is made in the form

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

of rules and queries that depend on this terminology. Because of the central role of this vocabulary, it is highly recommended that the core vocabulary comes from already established sources, and extended on top of them.

As best practices and recommendations, tools that integrate knowledge based functions, should provide base languages and/or mechanisms to integrate external selected languages. In the case of CPS, **we aim to provide the basis and recommended terminology for autonomous and semi-autonomous systems that compose** CPSs, as well as the shortcomings and needs of the available ones. To support the analysis and selection of the ontological resources, a set of vocabularies, catalogues, ontologies and data-models relevant to CPS and (semi) autonomous systems is provided in the *resources* section.

A **demonstrator** on how to integrate domain specific knowledge in the cluster toolchain, is currently on progress guided by the following methodology:

- Provide the system model in UML (other modeling formalisms are subject to the same process).
- Translate/annotate the model (a view of the model) mapping UML concepts into a compatible and relevant ontology in OWL. By compatible, we refer to an ontology that refers to the type of system we model (e.g. drone) and by relevant we mean the ontology should consider the viewpoint we want to analyze (e.g. mechanical viewpoint, electrical viewpoint, logical viewpoint, etc.)
- Expose the translated/annotated model to a common ecosystem, in the form of OWL ontologies, where the other tools can get access to it.
- Analyze the results by importing the translated/annotated model into the required tools format.
- Recover/extract the results made by the analysis tools into the common ecosystem.
- Provide feedback to the designer.

2.4.3.2 Safety Analysis

Several AI related characteristics may impact the trustworthiness of an autonomous systems with the introduction of new risks into the system - with both, negative or positive outcomes – or increase their likelihood due to specific characteristics of the technology. These new risks require specific focus and additional guidance through a novel risk management framework and process. We are developing a risk based approach in order to identify and to assess risks to which such systems may be exposed and derive appropriate safety principles and mitigation measures. The approach focus on the design-stage of the CPS development through the following activities:

- Operational Design Domain (ODD) definition to serve for Risk Assessment in order to address the lack of specification of AI application.
- Identification of critical scenarios to address the SOTIF related risks and cyberattacks.
- Risk quantification and criticality analysis enable to deal with the evaluation metrics for autonomous functions and uncertainty that may entail the operational contexts and the AI models.

The purpose of the ODD definition is to serve as input for building the catalog of situation in which the system can operate for the hazard analysis. We aim to use the knowledge of experts through ontologies to ensure that we cover all scenarios for which a system is designed to operate properly. For the representation of the scenario-space of a specific autonomous system, i.e. the ODD, we must select in the ontology the parameters relevant to the system functionality and envisaged context and apply boundaries to them to take into account the operating limitations. The ODD will be used to define a *situation catalog* for the system. The ODD will also serve to derive *1/ safety requirements* on the architecture model of the system, based on the operating limitations identified; and *2/ safety constraints* (including safety variables) needed to monitor the system at runtime - so that to avoid the system to exit the ODD.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

In addition to the ODD-based situation catalog, the hazard analysis requires to identify the conditions that will lead, if occurring in the operational situation, to an accident. For classical systems, such conditions are predefined in some hazard list and that are essentially based on known HW/SW failure. For the AI-based systems, we must consider also as initiating condition, any functional insufficiency, or human misuse to cope with the potential autonomous functions weaknesses. To do so, we are developing a systematic approach for deriving critical scenarios based on adapted FMEA, STPA, HAZOP guidewords, extended with some specific guidewords propose by the SOTIF to handle the potential human misuses. The guidewords must be applied at vehicle level (vehicle level functionality) but also at functional component level (e.g. perception, navigation, etc.) to derive as complete as possible the critical scenarios.

For classical systems, the risk is quantified based on the *exposure* of the events in the accident scenario and the *severity* of the resulting consequences. In addition, the human involved in the system environment can low the risk based on their intervention. For some AI-based systems, there may not be a human-in-the loop, and we have to take into account other parameters that may impact the negative outcome occurrences and their severity. Some of these factors are time-dependent and context-dependent. To do so, we use accident/pre-crash prediction analysis. The prediction model is based on a causal analysis including common cause and common failure analysis represented as a directed acyclic graph (following a STPA/FTA like approach) to infer the future states of the system and its environment until accident happens. The nodes of the graph represent the events (identified as the initiating conditions in the critical scenarios) that can modify the state of the context and system.

The overall analysis will help classify and prioritize the critical scenarios and determine when one need to define mitigation measures. Those mitigation measures may be four-fold: design-based, fail-operational, fail-safe or informative (i.e. with no impact on the development process). Some measures are also the input to derive the safety constraints and safety variables to deal with at runtime.

Future work is to take into account the uncertainty in the approach.

2.4.3.3 Planning

Another contribution is the extension of the existing CEA technology for model-based engineering of robot software systems, Papyrus for Robotics (P4R), in order to enable modeling and (semi-)automated synthesis of AI planning in robotic architecture solutions.

The contribution is organized in two parts. First, we extend P4R (Papyrus for Robotics) to integrate PDDL-based planning and reactive and fault-tolerant task execution into deployed ROS2 solutions based on Plansys2[1], BehaviorTree.CPP[2] and other supporting technologies. We develop dedicated modules for code-generation from P4R models to enable *(i)* the automated creation and run-time update of behavior trees to implement PDDL plans, and *(ii)* the implementation of run-time monitors and recovery strategies to deal with adverse and/or unspecified situations in operation through observation of the system and its environment.

Second, we provide P4R with dedicated modules to support the specification and the verification of PDDL models by non-experts in AI-Planning. This work includes the definition of a DSL which hides the complexity of PDDL definitions and raises the abstraction of planning specifications, from which PDDL models (domain and problem models) can be generated. It also includes the exploration of new ways to integrate the formal specification of behavior requirements (safety, resource utilization, etc.) with PDDL (domain) models to enable their verification.

2.4.3.4 NN Verification

The ISAIEH tool (description provided in D5.3) was mainly aimed at bridging the gap between the NN world (through the handling of standard formats such as ONNX and NNnet) and the formal methods

[1] https://intelligentroboticslab.gsyc.urjc.es/ros2_planning_system.github.io/index.html
[2] https://www.behaviortree.dev/

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

world (by targeting formats such as SMT-Lib), hence the name (Inter-Standard AI Encoding Hub). The natural evolution of this line of work is to reinforce this connection with numerous plugins, covering different aspects of property verification, as well as clever heuristics to guide the various tools used for this analysis. This next step has been undertaken, partly in the context of CPS4EU, with the development of the CAISAR (Certifying AI Safety And Robustness) platform, for which ISAIEH served as the first step (offering the internal representation of AI, parsing features, *etc),* making it one of the main plugins of the CAISAR platform.

Other plugins are already added to this platform, but they represent only a part of the planned integrations to CAISAR, which will span over the upcoming years. These integrations aim at the long-term goal of CAISAR which is to cover verification aspects (both global and local) aided by clever guiding heuristics like the CEA developed DISCO (for Dividing Input Space into Convex polytops, which was the subject of a publication currently under review in FM2021), as well as other useful features (*e.g.,* explainability, feature extraction, visualisation). Some of the plugins already integrated in CAISAR are external to CEA (such as Marabou) but some of them include CEA tools such as PyRAT.

As its name implies, PyRAT (for Python Reachability Analysis Tool) applied to NN is focused on the robustness properties, both in the context of malicious attacks (*i.e.* adversarial) and in the context of nominal conditions (where the properties to verify would be, for example, resilience to perturbation in sensory inputs).

The integration of other plugins is currently taking place on the CAISAR platform (Eran, NNenum, *etc*). Here again, some of these plugins are developed at CEA, such as AIMOS, AI Metamorphism Observing Software, which applies metamorphic testing —a kind of property-based testing— to blackbox AI. The ColibriCS and Colibri2 solvers that are currently linked to ISAIEH (see Figure 1) are also projected to be included in the CAISAR platform. These tools can not only serve to prove properties about AI, but ColibriCS (for Colibri Certified Solver) can also provide a particular solution that is conspicuously lacking in the state-of-the-art regarding the safety of symbolic AI. Indeed, many critical systems use constraint programming, a type of symbolic AI, in their applications, and the need for safety insurance in these domains is no less paramount than for NN. ColibriCS answers this need by offering a set of individually verified solver bricks that can be combined together to form a solver that is tailor-made to the needs of a particular user.

The ultimate goal for CAISAR is to be released as an open-source platform (the administrative steps towards this goal are well underway) in order to encourage development and collaboration with exterior partners.

2.4.4 Risks and considerations

Special care has to be taken into the correspondence between a meta-model (UML) and a KB. The intended meaning (semantics) provided by the KB might not be fully aligned with the underlying meta-model. This specification has to be detailed and documented to avoid confusion and eliminate ambiguity. UML instances (e.g. of a class) might need to be classes in the OWL format, and associations might need to be specified as roles at the right level. Mismatches between these abstractions levels would risk the successful integration of the toolchain.

As already mentioned, the ***viewpoint*** is especially important. The integration assumes there is an entity (e.g. the system) for which several tools exist to handle specific aspects of its design, evaluation, deployment, etc. But the viewpoint can change the relations between these items, and this should be clear to other tools, services and actors outside the current tool. Take as an example a drone mechanical and logical viewpoint: in the mechanical perspective the Autopilot as well as the Motor as mounted in the drone's Body, and the Propeller is mounted in the Motor, whereas in the electrical view the Motor is connected to the Autopilot, and the Propeller is not in the view. Nevertheless, the energy that is output to the Motor, depends on the Propeller installed. To overcome this mismatch, all

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

concepts and relations must be properly described (with an *rdf:comment* tag or similar resource) and a complete catalogue must be accessible.

2.4.5 Additional resources

Regarding ontologies and knowledge based systems for autonomous systems, robotics and Industry 4.0, we have found especially relevant two recent surveys that provide guidance, approaches and resources to enable KR integration and its application in different disciplines and scenarios:

- 2019 survey regarding ontology based approaches to robot autonomy [1]: http://www.iri.upc.edu/files/scidoc/2257-A-review-and-comparison-of-ontology-based-approaches-to-robot-autonomy.pdf
- A 2021 survey on KR systems applied to robotics [2], with an emphasis in robot interaction with humans. The authors focus on applications and social roles of robots (in domestic, health and industrial activities). This survey complements the above mentioned survey, and is also a valuable resource and guide to KRs systems in industry, their implementation, scope and limits. https://www.mdpi.com/2076-3417/11/10/4324

2.4.6 Future work

In this section, we present some foreseen steps that came out of the analysis done in this section, but remain out of the scope of the current work:

- Integration with external KBs
- NLP based approaches can be built on top / NER (Named Entity Recognition).
- Annotation of object recognition can serve from the identified KBs/Ontologies.
- Machine learning algorithms to provide recommendations and patterns, that can be re-used in future projects.
- Other modeling environment can be used to implement the safety analysis approach.
- Publications of the approach to integrate knowledge into the design and the enabling of the interaction among the tools, is envisaged as the pieces are successfully coupled.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

3 CONCLUSION

In the previous deliverable (D5.3), the intended interaction of the tools are solely based on the targeted domains of the individual tools and the specified exchange formats and interfaces. In this deliverable, we have further specified this interaction by refining the toolchains, defining and identifying the functions and processes that are more compatible, and establishing, where possible, the risks, recommendations and limits to render this integration successful.

In the case of AI-Based CPS-Systems tool-chain, we have provided further details on the interaction of knowledge driven engineering, safety analysis, planning and NN verification tools. These descriptions provide insight on the process inside each tool, their capabilities, their outcomes and further development. The descriptions have been made considering the intended and potential interactions between the tools. We provided means (technology bricks and resources) to realize these interactions as well as the specific aspects of each tool's functions, which can be consumed by the other elements in the toolchain. As such, these specifications should be regarded as guidelines for the application of a toolchain integrating knowledge based engineering, safety analysis, planning and verification of NN properties.

In the same line of thought the heterogeneous co-simulation cluster, the generation of simulation components using different tools and methodologies has been described. This distributed co-simulation can be used to validate Cyber-Physical Systems, allowing a rapid prototyping, software-in-the-loop (SiL) and hardware-in-the-loop (HiL) techniques.

This cluster defines the use of the HLA standard to coordinate the simulation components in a distributed simulation. Each of the generated components will simulate a specific characteristic, which together form a complex simulation (interoperability). Additionally, once a simulation component is designed, it can be reused for other simulations, reducing development time (reusability).

Because these simulation components are very complex, the user guide presented in this document can be a reference for future implementations, including recommendations for the pre-integration of the generated models from different modelling tools and some considerations and risks.

In the next stages of the project, the development of demonstrators to showcase these tools chains, and that further refine these recommendations and materialize the specifications are envisaged.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

4 REFERENCES

[1] MANZOOR, Sumaira, et al., «Ontology-Based Knowledge Representation in Robotic Systems: A Survey Oriented toward Applications,» *Applied Sciences,* vol. 11, n° %110, p. 4324, 2021.

[2] OLIVARES-ALARCOS, Alberto, et al., «A review and comparison of ontology-based approaches to robot autonomy,» *The Knowledge Engineering Review,* vol. 34, 2019.

[3] N. Yakymets, S. Dhouib, H. Jaber, and A. Lanusse, «Model-Driven Safety Assessment of Robotic Systems,» chez *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'2013*, Tokyo, Japan, November 3-7, 2013.

[4] R. Nouacer, M. Djemal, and S. Niar, «Using Virtual Platform for Reliability and Robustness Analysis of HW/SW Embedded Systems,» chez *1st International ESWEEK Workshop on Resiliency in Embedded Electronic Systems, In conjunction with Esweek 2015*, Amsterdam, The Netherlands, 2015.

[5] Topcu, O., Durak, U., Oguztuzun, H., and Yilmaz, L., «Distributed Simulation – A Model Driven Engineering Approach,» chez *Springer*, Cham, 2016.

[6] Durak, Umut, et al., «Computational representation for a simulation scenario definition language,» chez *AIAA Modeling and Simulation Technologies Conference*, Kissimmee, Florida, 2018.

[7] Karmokar, Bikash, et al., «Towards an Open Simulation Scenario Infrastructure,» chez *ASIM 2018 - 24. Symposium Simulationstechnik*, HafenCity Universität Hamburg, 2018.

[8] Durak, Umut, et al., «Using System Entity Structures to model the elements of a scenario in a research flight simulator,» chez *AIAA Modeling and Simulation Technologies Conference.* , Grapevine, Texas, 2017.

[9] Ellis, Oliver, «Simulation Based Development and Verification of Drogue Detection Algorithms for Autonomous Air to Air Refuelling.,» chez *AIAA Scitech 2020 Forum*, Nashville, Tennessee, 2020.

[10] Simulation Interoperability Standards Organisation, «SISO-STD-008-01-2012 - Standard forCore ManufacturingSimulation Data – XML Representation,» 08 August 2012. [En ligne]. Available: https://www.sisostds.org/DesktopModules/Bring2mind/DMX/Download.aspx?Command=Core_Download&EntryId=36239&PortalId=0&TabId=105. [Accès le March 2020].

[11] Antoine El-Hokayem and Yliès Falcone, «THEMIS: A Tool for Decentralized Monitoring Algorithms,» chez *In proceedings of the 26th ACM SIGSOFT Symposium on Software Testing and Analysis (ISSTA). ACM,125-135*, New York, USA, 2017.

[12] Yliès Falcone, «THEMIS Artifact Repository for ISSTA 2017 paper: Monitoring Decentralized Specifications,» [En ligne]. Available: https://gitlab.inria.fr/monitoring/themis. [Accès le March 2020].

[13] C. U. Press, «Meaning of trust in English,» Cambridge University Press, 2014. [En ligne]. Available: https://dictionary.cambridge.org/dictionary/english/trust. [Accès le 04 06 2020].

[14] D. Artz et Y. Gil, «A survey of trust in computer science and the Semantic Web,» *Web Semantics: Science, Services and Agents,* pp. 58-71, 2007.

[15] F. Bobot, Z. Chihani et B. Marre, «Real Behavior of Floating Point,» 2017.

[16] J.-C. Filliâtre et A. Paskevich, «Why3 --- Where Programs Meet Provers,» *Proceedings of the 22nd European Symposium on Programming,* vol. 7792, pp. 125--128, 2013.

[17] Chandra Karmokar, Bikash, et al. "Tools for Scenario Development Using System Entity Structures." *AIAA Scitech 2019 Forum*. 2019.

[18] Jafer, Shafagh, and Umut Durak. "Tackling the complexity of simulation scenario development in aviation." *Proceedings of the Symposium on Modeling and Simulation of Complexity in Intelligent, Adaptive and Autonomous Systems*. 2017.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276

[19] Jafer, Shafagh, et al. "SES and Ecore for Ontology-based Scenario Modeling in Aviation Scenario Definition Language (ASDL)." *International Journal of Aviation, Aeronautics, and Aerospace* 5.5 (2018): 4.

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276