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

D6.5 Final release of the AMPERE ecosystem

Version 1.0

Documentation Information

Contract Number	871669
Project Webpage	https://www.ampere-euproject.eu/
Contractual Deadline	2023-06-30
Dissemination Level	Public (PU)
Nature	DEM
Authors	Thomas Vergnaud (TRT)
Contributors	Tommaso Cucinotta, Francesco Paladino, Gabriele Ara (SSSA) Sara Royuela (BSC) Sergio Mazzola (ETHZ) Luis Miguel Pinho (ISEP)
Reviewer	Sara Royuela (BSC)
Keywords	Integration, workflow, automation

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

Change Log

Version	Description Change
V0.1	Initial version
V0.2	Contributions from SSSA
V0.3	Comments and contributions from BSC
V0.4	Contributions from ISEP and ETHZ
V1.0	Final release of the document

Table of Contents

1 Executive Summary

This deliverable complements D6.4, *First release of the AMPERE ecosystem* [1] by describing the AMPERE ecosystem at the end of the project. It lists the tools and platforms involved in the ecosystem and explains how they are combined in the AMPERE engineering process. It also indicates the tools and platforms used by each use case of the project, i.e., ODAS and PCC.

The workflow of project AMPERE is built around three main artifacts, which are used by the AMPERE tools:

- the **Amalthea models**, described with the Amalthea domain specific modeling language (DSML) included in the APP4MC framework;
- the **C/C++ source code** and the associated **binaries**, generated by the synthetic load generator (SLG), included in APP4MC, and the LLVM compilation framework;
- the **Task Dependency Graph** (TDG), generated by LLVM and augmented by the different multi-criteria optimization tools.

Each AMPERE tool either produces or processes one or several of these artifacts. Thus, the AMPERE ecosystem is built by combining the AMPERE tools linked by these artifacts. Overall, the tools and techniques developed during the project are applied to the two use cases of the project.

2 Introduction

The AMPERE ecosystem consists of a collection of tools that allow for (1) system design, (2) model-to-code transformation, (3) analysis and optimization tools, and (4) execution platforms. Previous deliverable D6.4 [1] already described the modeling tools and the execution platform. This document, which corresponds to the description of demonstrator 6.5, as described in the AMPERE Grant Agreement [2], provides a global overview of how the tools are combined together. As a summary, Figure 1 illustrates the final software stack of AMPERE, with a minimal modification compared to that presented in D6.4, in particular, with regard to the multi-criteria optimization phase.

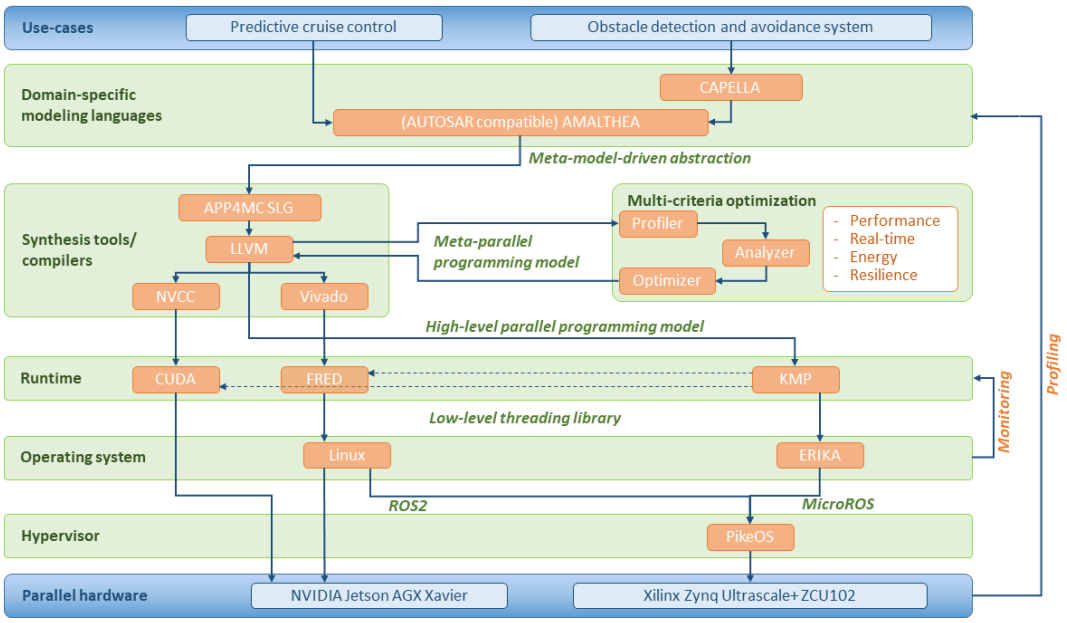

Figure 1: AMPERE software stack

The remainder of the document is organized as follows: Chapter 3 depicts the whole AMPERE software ecosystem, showing the different possible workflows and the tools conforming it, as well as the tools used by each of the use cases; Chapter 4 shows the continuous integration process; and finally, Chapter 5 presents the conclusions of this document.

3 The AMPERE software ecosystem

This chapter presents, in a nutshell, the complete AMPERE software ecosystem, listing the different tools and interfaces that conform each step of the workflow, and also identifying the tools used for each use case.

3.1 Workflow

The workflow of the AMPERE ecosystem is depicted in figure 2. Each phase is recognized by different colors in the arrows linking components. Hence, the AMPERE ecosystem is made up of three phases, as described in the following paragraphs.

System design and model-to-code transformations (blue). This step consists on two phases: (a) the generation of the models of the two use-cases addressed in the model representing the functional behavior and the non-functional restriction of the system and its components; and (b) the generation of synthetic source code that simulates the real code and includes both the functionalities and its requirements. The use of synthetic code allows to perform measurements and analysis before the real system is actually produced, hence enabling analyses detecting issues in the early stages of the system design.

Execution on the target platform and monitoring (orange). The simulation code is then executed on the target platform and monitored to produce metrics for different aspects which, in the scope of AMPERE, include power consumption and timing analysis, considering performance and resilience. At this stage, different system configurations (e.g., with and without replication for resilience, or different frequencies for parallelism and power consumption) are used in order to have enough data for the optimization.

Analysis of execution metrics and multi-criteria optimization (green). Finally, the metrics are analyzed to compare the performances of each configuration and decide which is the configuration that provides better results for all the non-functional requirements considered in the project.

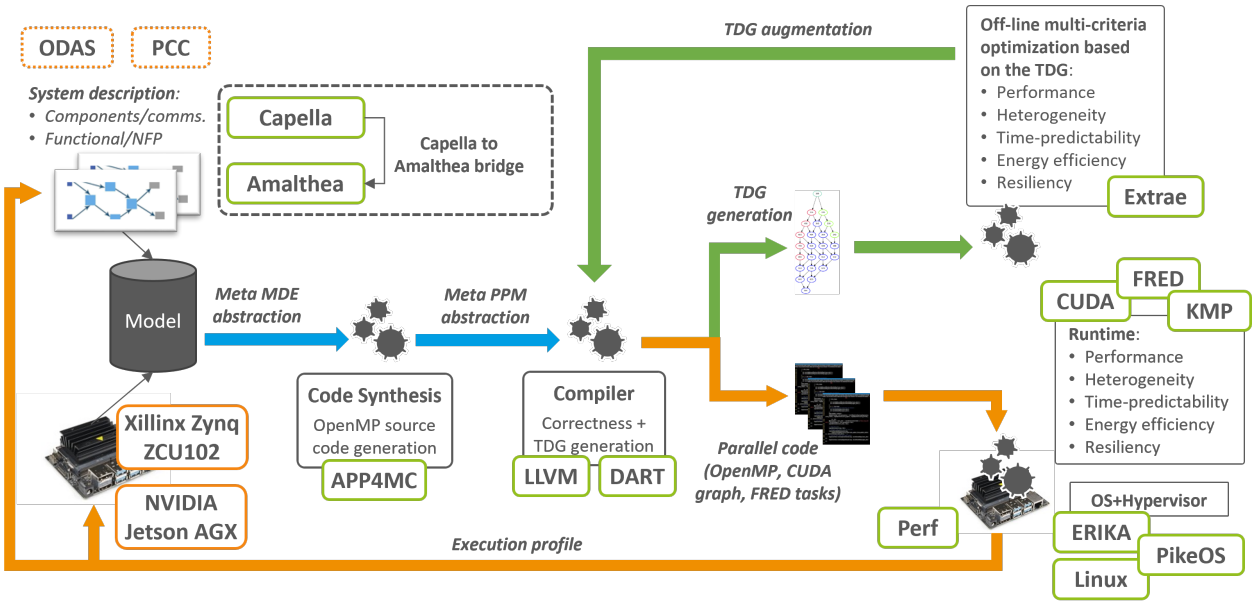

Figure 2: Overview of the AMPERE software ecosystem

The AMPERE workflow, considering the different development phases, relies on the following three main data representations:

- an *Amalthea model* of the system;
- executable *binary files*, i.e. the actual software system;
- a *task dependency graph* corresponding to the system.

Each of these data representation is a pivot shared by several tools, as represented in figure 3. The pivot for the *design* step is the Amalthea model of the system under construction. The pivot for the *execution & monitoring* step is the executable binary file of the system. The pivot for the *analysis* step is the task dependency graph.

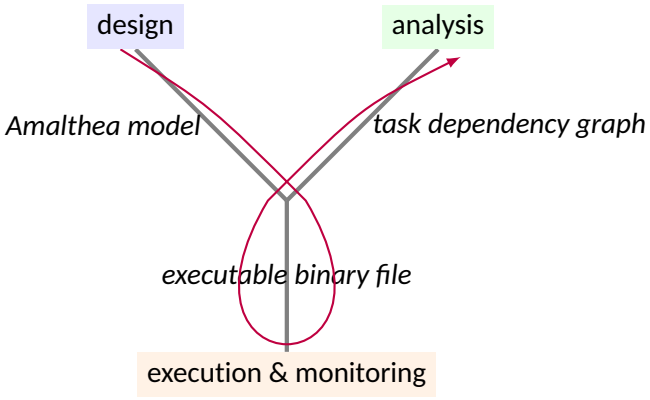

Figure 3: The AMPERE general workflow

3.2 AMPERE tools

This section presents the AMPERE ecosystem, which consists of several tools involved in the engineering process. Most of these tools are available from the AMPERE Gitlab repository hosted by BSC[1]. A few of them, however, are not developed specifically for AMPERE and they have their own release cycle.

3.2.1 System design and model-to-code transformation tools

This step comes first in the AMPERE workflow. It consists in designing the software and hardware systems, and producing a source code that behaves like the actual system. The tools associated with this phase are listed in Table 1, including the two modeling tools, i.e., Capella and Amalthea, and the bridge to transforms the former in the latter, and also the model-to-code transformation tools, including the different versions of the SLG and the LLVM compiler.

Most of the design tools have not been specifically developed for AMPERE. In particular, the modeling tools Capella[2] and Amalthea, as part of the APP4MC platform [3], which are industrial tools that exist besides AMPERE. Included in APP4MC, AMPERE has also produced two versions of the SLG: One that targets Linux/ROS2 and supports the execution in the NVIDIA Jetson (GPU platform), with support for ELinOS/PikeOS and the Erika OS; and another one that targets FRED and supports the execution in the Xilinx Ultrascale (FPGA platform).

[1]AMPERE public software repository: https://gitlab.bsc.es/ampere-sw
[2]Capella: https://www.eclipse.org/capella/
[3]App4MC: https://www.eclipse.org/app4mc/

Name	Work package	Deliverable	Input	Output
Capella	External, v5.2.0	D6.4	*Human input*	Capella model
Bridge	WP6	D6.4	Capella model	Amalthea model
Amalthea	External, v2.1.0 + WP1	D1.5	*Human input*	Amalthea model
SLG for ROS2	WP2	D2.3	Amalthea model	ROS2/OpenMP/C++ source code
Code gen. for FRED	WP1	D1.3	Amalthea model	FRED/C++ source code
LLVM	External 17.0 + WP2	D2.3	C++ source code	executable binary files and task dependency graph (TDG)

Table 1: System design tools with interfaces

3.2.2 Execution and monitoring tools

The execution and monitoring step of the AMPERE workflow consists of executing the binaries produced in the previous step, including different configurations of the system (e.g. CPU/GPU specializations, frequencies, etc.) in order to measure their performances, mainly in terms of power consumption and execution time.

The hardware boards used in AMPERE are the GPU-based NVidia Jetson Xavier AGX and the FPGA-based Xilinx Ultrascale+ ZCU102 rev1.1. Neither NVidia nor Xilinx are parts of project AMPERE, so both boards were purchased by the partners who had to use them. These platforms include parallel runtimes, operating systems, hypervisors, and communication frameworks based on ROS2[4] to execute the AMPERE use-cases. The operating systems and frameworks are listed in table 2.

OpenMP-LLVM allows for parallel orchestration in the CPU and the GPU, exploiting the resources as specified in the Task Dependency Graph generated during compilation, either using dynamic scheduling or the static scheduling obtained after the multi-criteria optimisation. FRED[5] allows for time-scheduling of hardware IPs over FPGA slots, for Xilinx devices capable of Dynamic Partial Reconfiguration (DPR). FRED can use at runtime hardware designs that have been compiled with the coupled off-line tool DART.

Several libraries and execution monitors are to be combined with the executable binaries to collect data related to the execution metrics of the binaries. These tools are listed in table 3. The metrics they produce are to be processed by the analysis tools listed in section 3.2.3.

In particular, PARTProf[6] can be used to measure the execution time and power consumption changes of a number of simple reference workloads on a board, trying out all frequency configurations available for the CPU (or a select subset of them, if preferred). Its output can be used to configure the coupled tool PARTSim[7], capable of simulating the timing, power consumption and thermal model behavior of a number of embedded boards.

The Voltmeter library collects power & performance counter datasets to train the power models. This tool is out of the multi-criteria optimization loop, and it is only part of the ETHZ workflow to build the power models.

[4] ROS2: https://www.ros.org
[5] FRED: https://github.com/fred-framework
[6] PARTProf: https://gitlab.retis.santannapisa.it/parts/partprof
[7] PARTSim: https://gitlab.retis.santannapisa.it/parts/partsim

Name	Version	Deliverable	Main function
FRED	Internal, open-source	D4.4	Run-time support for time-scheduling of FPGA accelerators
OpenMP-LLVM	LLVM 17.0 + WP2	D4.3	Run-time support for OpenMP parallelization and offloading
CUDA	External 11.4	–	Run-time libraries for GPU execution on NVIDIA platforms
µROS	Integrated within Erika	D5.4	Publish-subscribe communications middleware
ROS2	External, version *Galactic*	–	Publish-subscribe communications middleware
Erika	Internal development of Evidence	D5.4	Hard Real-Time, automotive-certified OS
PikeOS	External, v5.0	D5.4	Certified Real-Time Hypervisor
ELinOS	External, v7.0	D5.4	Linux distribution for PikeOS
Peta Linux	External, 2020.2	D5.4	Linux distribution for Xilinx bare-metal
Ubuntu Linux	External, 20.04.6 LTS with Jetson Linux 35.2.1	–	Linux distribution for NVIDIA Xavier Jetson AGX
Linux kernel	External, Xavier board: 5.10.104-tegra, Xilinx board: 5.10.107-ELinOS-5439-rt64	–	Linux OS kernel
Runmeter	v1.0	D4.3	Run-time energy estimator based on performance counters

Table 2: Execution tools and platforms

Name	Work package	Deliverable	Main function
PARTProf	WP2	D2.4	measure execution time and power consumption
Memory interference prediction and control	WP3	D3.3	count memory interference events
Voltmeter	WP3	D3.3	train the power models
Amperf	WP4	D4.3	measure CPU/GPU execution time and performance counters
Carmel PAPI component	WP4	D4.3	collect low-level metrics

Table 3: Monitoring libraries

3.2.3 Analysis and optimization tools

The tools used in the analysis and optimization step of the workflow are listed in table 4. They focuses on two concerns: time predictability and power consumption. The tools process the task dependency graph (TDG) produced by LLVM and update it with the measurements performed during the application execution.

The output of the multicriteria optimization allow the system designer to compare various configurations of the system to decide the most suitable depending on the requirement to optimize.

Name	Work package	Deliverable	Input	Output
Multicriteria optimization	WP2	D2.4	TDG	Analysis results
Time predictability	WP3	D3.3	TDG + Metrics	TDG
Power modeling	WP3	D3.4	TDG + Metrics	TDG
PARTsim	WP3	D3.4	TDG + Metrics	Simulation results
RT DAGs optimizer	WP3	D3.4	TDG + Metrics	TDG + Placement + DVFS config

Table 4: Execution analysis and configuration optimization tools

3.3 Use cases

Two use cases are considered in order to illustrate the AMPERE workflow: a Predictive Cruise Control (PCC) and an Obstacle Detection and Avoidance System (ODAS). The AMPERE ecosystem complete adapts to the execution of these use cases by customizing the tools and platforms used for their deployment, hence demonstrating the adaptability of the ecosystem.

Both use cases follow the same three steps of the AMPERE workflow: System design and code generation, execution and monitoring, analysis and optimization. Also both use cases use all the analysis tools. The PCC use clase, driven by Bosch, uses the APP4MC platform (part of the Bosch engineering process) as the starting point, and targets the two platforms of the ecosystem, i.e., the NVIDIA Jetson Xavier with Linux and the Xilinx Ultrascale+ Zynq ZCU102 with Erika/PikeOS. Accordingly, it uses the tools associated with FRED (PARTProf, PARTSim, DART). Instead, the ODAS use case, driven by GTSI, uses Capella (part of the GTSI ecosystem) as the starting point and targets only the Jetson platform.

Tool name	Used in PCC	Used in ODAS
Capella	–	✓
Bridge	–	✓
App4MC	✓	for debug only
SLG for ROS2	✓	✓
SLG for FRED	✓	–
LLVM	✓	✓
Erika, µROS	✓	–
PikeOS, ELinOS	✓	–
FRED & associated tools	✓	–
Jetson Linux kernel	✓	✓
ROS2	✓	✓
Monitoring libraries	✓	✓
Analysis tools	✓	✓

Table 5: The tools involved in the AMPERE use cases

4 Integration chain

The source codes of the tools developed in the scope of the project are hosted in a Gitlab by BSC[1]. Thus, project partners have a convenient access to a working version of the tools.

The files of the two use cases (PCC and ODAS) are also hosted in a public Gitlab by BSC[2]. The design step of the AMPERE workflow has been automated, as described in D6.4 [1]. Whenever new data is pushed into one of the use case repository, the source code is produced (using the AMPERE version of SLG fetched from the BSC Gitlab) then compiled (using the AMPERE version of LVVM fetched from the BSC Gitlab). Gitlab automatically sends an e-mail if the process could not succeed. This enables the use case providers to quickly detect problems in their input models.

The input model of use case PCC is an Amalthea model. The following operations are automatically performed upon any update in the PCC Amalthea model:

1. launch SLG to produce the source code of the application;
2. compile the source code against ROS2 with LLVM, using OpenMP instructions.

If no error occurs, the binary files are then available from the Gitlab.

The input model of use case ODAS is a Capella model. The following operations are automatically performed upon any update in the ODAS Capella model:

1. launch the Bridge to produce the Amalthea model;
2. launch the SLG to produce the source code;
3. launch LLVM to produce the binary files.

If no error occurs, the binary files are then available from the Gitlab.

Specific details for the configuration of the AMPERE ecosystem have been provided in previous deliverables, in particular D6.4 [1]. Regarding WP2, the steps for the configuration, installation and execution of the ROS2 SLG with extensions for OpenMP, and the LLVM compilation framework are provided in D2.3 [5].

[1] https://gitlab.bsc.es/ampere-sw
[2] https://gitlab.bsc.es/ampere

5 Conclusions

The AMPERE ecosystem consists of several tools and platforms. The different supported workflows enable the design of complex Cyber-Physical Systems (CPS), such as the use cases addressed in the project, and the production of software code that *simulates* the real-load of those systems. This simulation is automatically produced and it is thus available in the early stages of an industrial development process, before the real application is fully implemented. This allows the execution and monitoring of the system under various configurations with regard to parallelism, energy, resilience, time predictability, and heterogeneity to then select the most appropriate system configuration.

Several platforms are considered in project AMPERE: The NVidia Jetson Xavier AGX and the Xilinx Ultrascale+ ZCU102 FPGA, with Erika and PikeOS/ElinOS or Linux.

The main goals of the AMPERE project, as described in the Grant Agreement (GA) [2] include:

- G1. Providing a system design ecosystem for CPS.
- G2. Providing a computer software ecosystem capable of efficiently exploiting advanced energy-efficient and parallel heterogeneous platforms; and
- G3. Integrating AMPERE software solutions into two relevant use cases from the automotive and the railway domains.

In a nutshell, the design, production and analysis tools of AMPERE are combined to create an ecosystem supporting the design, the monitoring and the analysis of complex CPS, like those represented by the PCC and ODAS use cases, as shown in the analyses provided in D2.5 [6], D3.4 [7], D4.4 [8] and D5.4 [3].

Demonstration materials can be found at https://b2drop.bsc.es/index.php/s/65DCSTca6owaYRC.

6 Acronyms and Abbreviations

CPS	Cyber-Physical System
CPU	Central Processing Unit
D	Deliverable
DSML	Domain Specific Modeling Language
FPGA	Field-Programmable Gate Array
G	Goal
GPU	Graphics Processing Unit
HPC	High-Performance Computing
MDE	Model-Driven Engineering
ODAS	Obstacle Detection Avoidance System
PCC	Predictive Cruise Control
ROS	Robot Operating System
RR	Reconfigurable Region
SLG	Synthetic Load Generator
TDG	Task Dependency Graph
WP	Work Package

7 References

[1] AMPERE, “Deliverable D6.4, First version of the AMPERE ecosystem,” September 2022.

[2] European Commission and AMPERE beneficiaries, “Grant Agreement Description of Action,” 2021.

[3] AMPERE, “Deliverable D5.4, Evaluation of the operating systems and hypervisors,” June 2023.

[4] ——, “Deliverable D3.3, Energy optimization framework, predictable execution models and analysis, and software resilient techniques,” September 2022.

[5] ——, “Deliverable D2.3, Programming model extensions and the multi-criteria performance-aware component,” September 2022.

[6] ——, “Deliverable D2.5, Evaluation of performance-aware model transformations,” June 2023.

[7] ——, “Deliverable D3.4, Evaluation of multi-criteria optimizations,” June 2023.

[8] ——, “Deliverable D4.4, Evaluation of run-times,” June 2023.