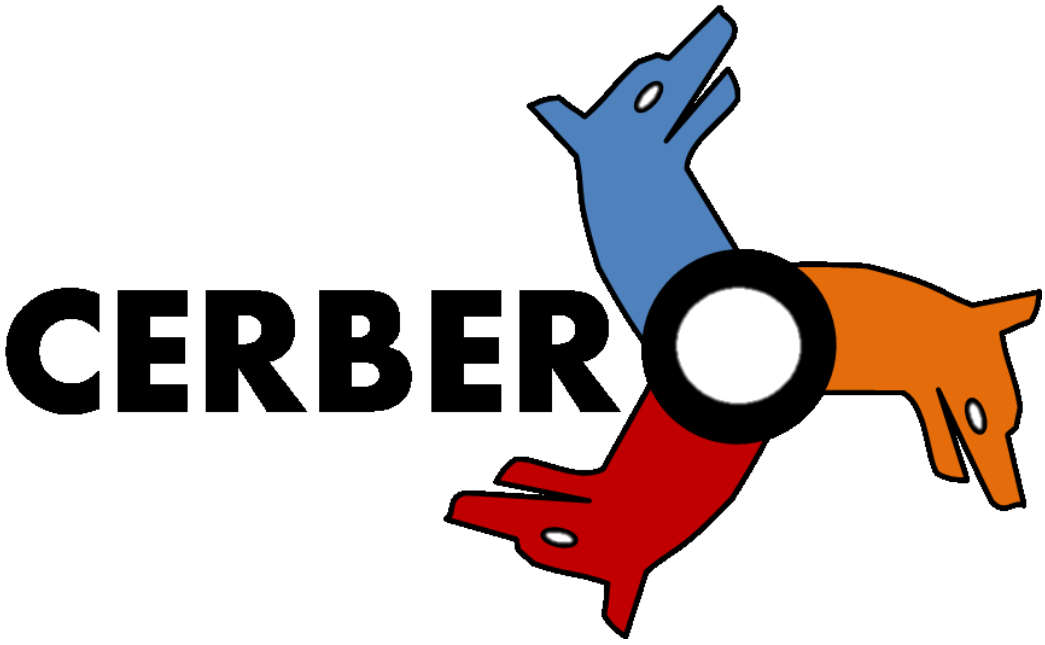
Information and Communication Technologies (ICT) Programme

Project Nº: H2020-ICT-2016-1-732105



D6.5: CERBERO Performance Report

Lead Beneficiary:	TASE
Workpackage:	WP6
Date:	31-12-2019
Distribution - Confidentiality:	Public

Abstract:
This document summarises the achievable performances extracted from the different scenarios. Performance are compared to the initial requirements and, as well, benchmarked against legacy solutions.

Disclaimer

The CERBERO Consortium is the following:

Num.	Beneficiary name	Acronym	Country
1 (Coord.)	IBM Israel – Science and Technology LTD	IBM	IL
2	Università degli Studi di Sassari	UniSS	IT
3	Thales Alenia Space Espana, SA	TASE	ES
4	Università degli Studi di Cagliari	UniCA	IT
5	Institut National des Sciences Appliquees de Rennes	INSA	FR
6	Universidad Politécnica de Madrid	UPM	ES
7	Università della Svizzera italiana	USI	CH
8	Abinsula SRL	AI	IT
9	Ambiesense LTD	AS	UK
10	Nederlandse Organisatie Voor Toegepast Natuurwetenschappelijk Ondeerzoek TNO	TNO	NL
11	Science and Technology	S&T	NL
12	Centro Ricerche FIAT	CRF	IT

For the CERBERO Consortium, please see the http://cerbero-h2020.eu web-site.

Document Authors

The following list of authors reflects the major contribution to the writing of the document.

Name(s)	**Organization Acronym**
Pablo Sánchez de Rojas	TASE
Joost Adriaanse	TNO
Bas Pijnacker Hordijk	S&T
Stuart Watt, Ayse Goker, Leszek Kaliciak, Hans Myrhaug	AS
Antonella Toffetti, Giovanni Turi	CRF
Evgeny Shindin	IBM

Document Revision History

Date	**Ver.**	**Contributor (Beneficiary)**	**Summary of main changes**
21-10-2019	0.1	TASE	First draft
10-11-2019	0.2	TNO	TNO input draft
30-11-2019	0.3	AS	First AS input draft
9-12-2019	0.4	AS	Revised AS input draft
16-12-2019	0.5	UNISS	Scientific Coordinator Review
18-12-2019	0.6	TNO, CRF	Updated ST use case
13-01-2019	0.7	TASE	Revision
05-02-2019	0.8	IBM, TASE	Revision and correction

Table of contents

1. Executive Summary

This document reports the performance achieved in all three use cases of the project CERBERO.

Details on the fulfillment of the initial requirements are given, proving that the final demonstrators successfully validate the needs identified in early stages of the project.

An analysis of the main commercial tools that are considered standard, or at least commonly applied, in the problems tackled by the use cases is included, showing why are them suitable for traditional industry solutions, highlighting their main weak points and remarking potential functionalities they could need for future developments.

Besides, a comparison of the results of the project in M36 with those that could have been achieved through legacy solutions is given, emphasizing the main advantages of CERBERO tools and technologies in most areas for the different scenarios and also showing some of the drawbacks that were found during the work.

Finally, the CERBERO tools are evaluated in the context of the use case, and main advantages and disadvantages are indicated along with any other issue identified by industry partners.

1.1. Structure of Document

The document is organized as follows:

- Section 2 explains the scope and purpose of the document.
- Section 3 presents, for the Smart Travelling use case, the evaluation of the initial requirements, a description of available commercial solutions for the different problems of the scenario, and an assessment
- Section 4 shows the same information than section 3 but focusing on the Planetary Exploration use case.
- Section 5 is equivalent to sections 3 and 4, oriented to the Ocean Monitoring use case.

1.2. Related Documents

This document is related to the deliverables listed below:

- D2.1 - Scenarios Description (Final Version)
- D2.2 - Technical Requirements (Final Version)
- D6.1 - Demonstrator Skeleton (Final Version)
- D6.2 - Space Demonstrator (Final Version)
- D6.3 - Ocean Monitoring Demonstrator (Final Version)
- D6.4 - Smart Travelling Demonstrator (Final Version)
- D6.6 - Gap Analysis and Development Roadmap

1.3. Related CERBERO Requirements

As specified in D2.2, the project sets a requirement (CERBERO-0016) to test tools/technologies versus the state of the art; and a requirement (CERBERO-0017) to provide feedback on the usability of CERBERO framework.

Additional user requirements “TP1: Ensure products will use state-of-the-art technology” and “UCS2: Usability of tools” are related to the content of this deliverable.

2. Scope and purpose

This document provides a revision on the results achieved with respect to the initial requirements of every use case. In order to avoid repetition with other deliverables, this information is summarized in the form of a table. Extensive details on the results of the demonstrators are provided on deliverables D6.2, D6.3 and D6.4, specific for each use case.

Also, an analysis on the state-of-the-art tools in relevant fields of the scenarios is provided, as well as a comparison of these widely used, standard-like solutions with the CERBERO tools and technologies. A tool evaluation section is provided for each one of the use cases highlighting the potential, key benefits and flaws of CERBERO tools for industry applications.

This information serves as input for the gap analysis of deliverable D6.6, where all discrepancies/desiderata are compiled and analyzed, possible solutions for every issue are presented and a road-to-development, or in general towards a wider usability, is proposed for the CERBERO tools.

3. Smart Traveling use case

3.1. Evaluation on initial requirements

Table 2. CERBERO High Level Requirements

Requirement	Validation demonstration	Evaluation
ST1. Development of parametric, modular and extendable cyber-physical co-simulation environment. **Need**: reduction of costs, increase of reuse in different simulation scenarios	Modular communication protocols and time synchronization. Logging application. Building Battery and Motor modules from generic components. Modular and extendable Driver support module. Safe, Secure, and Private Adaptive routing module with energy and cost efficiency and sensitive to drivers needs and environmental status.	System in the loop configuration was successfully implemented using DynAA, in which TNO motor and battery models were integrated and executed. The framework is flexible and allows addition or replacement of new simulation modules. Also, driver support module was added, which monitors both internal and external triggers and allows for adaptive routing using driver preferences.
ST2. Development of an integrated open-source or commercially available toolchain for design space exploration and co-simulation, with system-in-the-loop capabilities. **Need**: reduce development, verification, and integration time and costs by a library of reusable components and metrics integrated by common framework in different levels of abstraction; incremental prototyping.	Software in-the-loop simulation. Interoperability of System Level Design tools.	DynAA and MECA tools (from TNO and S&T) were used as a chain to accomplish the complex prediction and driver support functionality. As CIF was not available yet dedicated interfaces were developed between MECA and DynAA. Design space exploration functionality was used to perform predictions and not for the design of the demonstrator, as the demonstrator was focused on the operational phase. The tooling was also used for co-simulation as most parts of the driving simulator are simulated. This type of co-simulation is often applied however during the design of complex CPS so could well be applied in design phase of other projects.

ST3. Development of a (self-) adaptation methodology with supporting tools. **Need**: efficient support of functional adaptivity, according to system, human and environment triggers.	Re-routing in different simulation scenarios, mainly functional adaptation.	The added driving simulator is providing the self-adaptation through adaptation of the advice to the driver based on internal, human or external triggers. To include human triggers, special sensors were added to monitor status of the driver (e.g. tiredness). Given adaptation functionality could also be applied in fully autonomous cars, where available preferences can be used to select choices.

3.2. Description of available commercial solutions

The tools used during the project for the Smart Traveling scenario focus on system-in-the-loop simulation and planning and support. In order to better highlight the suitability of these tools in autonomous transportation applications, an overview on available solutions targeting these two main areas is provided.

3.2.1. System in the loop simulation

In the most well-known model-based system-oriented simulation tools MATLAB Simulink [Mathworks] and Modelica [Modelica] designers describe their component models, connect them, and start simulation runs. *Modelling of adaptation is very rigid however and within a single simulation run, the models and the system structure can only change if modification is previously described within models during experiment setup.* Although tools like MATLAB Simulink provide options to scale, these options *involve requirements to use specific cloud environments and expensive scaling licenses for every used machine*.

3.2.2. Planning & support

Defining relevant commercially available planning and decision support systems is difficult within the smart travelling context. Commercial planning and decision support tools are generally designed for a specific context and cannot be used off-the-shelf or easily reconfigured for different purposes. More generic planning and support solutions are typically developed as part of research projects. Several such solutions developed within the robotics community are available off-the-shelf as open-source software. Three examples will be discussed here: PLEXIL, T-REX, and ROSPlan.

PLEXIL (PLan EXecution Interchange Language) [PLEXIL] is a planning language designed and maintained by NASA, which is accompanied by *its own execution engine*. It is used in several NASA projects, aiming to represent automation plans in a clear way and providing the means to execute the plans on real or simulated systems.

T-REX [T-REX] originates from the Monterey Bay Aquarium Research Institute and was developed to support sea trials. It is an execution system, generating actions based on a user plan specification and on temporal constraints, as well as providing the means for re-planning in case of failures. Plans for T-REX are specified using EUROPA language. However, *maintenance of T-REX has stopped around 2010.*

ROSPlan [ROSPlan] is a more recently developed system and is actively maintained as of 2019. It is designed specifically for the Robot Operating System (ROS), a commonly used open-source framework in the robotics community. Like the other tools, it provides a plan generation and execution engine. It also includes a knowledge base that stores the modeled domain and state. ROSPlan domains are specified in PDDL (Planning Domain Definition Language).

3.3. Resulting benchmarking

In this section comparison of CERBERO tools/technologies against available commercial tools will be performed. The emphasize will be put on the differences showing how these differences allow to overcome limitations of available tools that were described in the section 3.2., w. r. t. smart travelling use case requirements.

3.3.1. DynAA - System in the loop simulation

DynAA provides simulation capabilities like MATLAB Simulink or Modelica. The simulation engine of DynAA however allows *every component, communication link, or environment model to be included, deleted, and/or modified during simulation*. Such capability makes DynAA specially *interesting for the analysis and simulation of self-adaptive, self-reconfiguring, and self-evolving systems*.

Within the CRF demonstrator it means that there is extensive flexibility compared to other available simulation tools as, during simulations, some models or components can be added, deleted and/or modified. Normally simulations need to be included as code libraries added to simulator control software (such as SCANeR describe below) before the simulation can be started. With usage of DynAA it was possible to *adapt the simulation of components of the car or the driver support predictions by switching simulation models without the need to halt the simulation*.

Although initial version of DynAA used MATLAB as a modelling environment, the updated version of DynAA developed in CERBERO is fully Java based and no longer requires MATLAB software to run. This will allow to run DynAA without MATLAB installed, providing capability of extensive parallel computing without the need for many MATLAB licenses. Furthermore, the Java implementation is much faster than the MATLAB version as it does no longer depend on external MATLAB code (which could not be optimized). This is especially important in order to perform real time simulation, where performance is critical. Potential users (such as CRF) will also *be able to use and adapt DynAA freely once it is open sourced by TNO*.

Another important difference with existing tools is the fact that *DynAA supports distributed simulation as a standard feature* using open source Apache Ignite software [Ignite]. For tools like *MATLAB Simulink special licenses and specific software and hosting functions* are needed for parallel execution.

DynAA was able to fulfill (system-in-loop) simulation tasks similar to commercial tools, while providing a number of benefits. Most relevant are:

- the flexibility on adaptation of the models
- the ability to easily extend and distribute the simulation over multiple servers.

3.3.2. MECA – Planning & support

Several properties of planning and decision support systems such as PLEXIL, T-REX, and ROSPlan are shared by MECA. Like these systems, MECA uses a well-developed planning language (PDDL) to define the high-level domain of a use case, delegating detailed tasks to lower-level programming languages.

However, MECA is more flexible. For example, *both PLEXIL and T-REX require the user to write and compile the lower-level code in C++*. Also, *support and documentation for T-REX are sparsely available*, most likely because maintenance of this tool has been stopped.

The *most comparable alternative to MECA is ROSPlan*, which offers an accessible knowledge base and is designed for the extensibility. However, *users of ROSPlan are required to use ROS*. This fact is substantially bounded usage of ROSPlan. In contrast to ROSPlan, *MECA is purely Python-based, and hence can be easily used by developers because Python is much more commonly used than ROS*. Moreover, MECA allows to define *external interfaces* using HTTP or REST API, *simplifying its integration with external tools*.

Other tools provide adaptation capabilities within the high-level planning domain specification only, making development of runtime monitoring and adaptation triggering very complex. *MECA offers monitoring and reconfiguration through Python modules, which are simpler to develop and test, and thus can perform more complex monitoring calculations with relative ease*.

In addition, *MECA is designed having in mind possibilities of user interaction and human-in-the-loop decision support*. For example, MECA can provide users with a set of options for decisions and the reasoning behind them based on KPIs. Also, *support for modeling and monitoring of user preferences is available*. *Other tools* are generally designed for developers of purely automated systems where only *limited human interaction* is expected at runtime.

In summary, MECA offers similar capabilities to existing generic planning tools, but

- in a more versatile and extensible form
- offering built-in support for human-machine interaction and decision-making.

3.4. Tool evaluation

During the project it was clear that drivers of electric cars would require some sort of **support for optimally charging the battery** during the trip, especially **when adaptation was needed**.

- The **astronaut planning support tool of S&T (MECA)** was found to be very applicable as driving assistant. It already contained functionality to include user preferences and was built to detect triggers for re-planning, and thus perfectly fitted the task of **adaptation manager**. To fit the use case, route planning functionality was added by using open source software, which allowed **MECA to request itinerary predictions using DynAA and provide the driver with guidance on route and charging options**.
- **DynAA** initially **missed** the **real time system in the loop option**, which was added during the project. As the original tool was developed using MATLAB, it was decided to recode the core to Java at the beginning of the project. Performed recoding allowed to **add co-simulation module and parallelism in a flexible manner**, without dependency on commercial MATLAB software. The **new DynAA core** proved to be very flexible and also powerful as the **simulations** can be **executed in parallel**. DynAA also proved to be **fast enough to correctly handle real time system in the loop simulations** (e.g. of battery and motor models).
- **SCANeR** (simulator control software used by CRF) initially **missed the option to simulate electric vehicles (apart from a driver support function)**, due to which the planned driver support on optimal charging could not be tested in the simulator. **By making use of previously developed battery and electric motor models of TNO, this functionality was added to the simulator**.

A considerable amount of results developed during CERBERO project will be used by CRF and by its **Driving Simulator**, beyond the project itself. For example, the **modules** simulating **electric engine and battery**, developed and shared by TNO. One of the main reasons to develop such modules during CERBERO project was their high exploitation potential that achieved by the possibility to use them both in a real time modality and also (more speedy) in a "prediction modality" into the optimizations tools allowing to these tools explore all the possible solutions. This dual modality guaranty consistency between the results of the predictors calculations and the real time outcome. Furthermore, these modules can be integrated at a low level in different development platforms and in different hardware, like microcontrollers, and other ECUs, beyond of course, the classical operative systems, like Windows, or in the classical and well-known real-time OS, that also increase their exploitation potential.

Predictors and optimizers (**DynAA** and **MECA** by TNO and S&T), integrated in the CRF Driving Simulator during CERBERO project and used during the tests, providing to the driver a set of possible choices. This set can be used as a comparative reference, in the context of future development activities aiming to the development of optimizing algorithms making calculation of the best routes for drivers of the electric vehicles based on external context, priorities, unforeseen events, driver preferences and mishaps, and other relevant data.

Also, a tool that monitors the driver and generates a descriptive value of his/her state of drowsiness enrich the data available for decision support algorithms, allowing to make these algorithms sensitive to the driver state.

Moreover, besides its use into the CERBERO Smart Travelling use case, the **HMI** designed by Abinsula and integrated into the driving simulator, can be reused with a same format input data package.

4. Planetary Exploration use case

4.1. Evaluation on initial requirements

Table 3. CERBERO High Level Requirements

Requirement	Validation demonstration	Evaluation
PE1. Enable Dependable Hardware / Software (HW/SW) co-design. **Need**: minimization of energy consumption and costs, while keeping/improving resiliency.	Flexible design of the arm controller using COTS FPGA and considering life-cycle costs, energy efficiency, reliability, etc. Trajectory generation and status monitoring applications.	The Adaption Manager has been implemented in M36 demonstrator, closing the adaptation loop and providing a control of the execution flow where advanced decision-taking takes place according to the different KPIs in order to guarantee the correct calculation of trajectories and commanding the model of the arm under a variety of circumstances.
PE2. Develop integrated open-source or commercially available toolchain environment for the design and assessment of heterogeneous cyber-physical systems. **Need**: provide multi-objective design space exploration and multi-view analysis; reduce development time of complex heterogeneous systems by increasing the level of abstraction; increase quality and verification level to ensure proper operation of the system.	Software and System in-the-loop simulation based on high-level applications abstractions. Interoperability between HW/SW co-design tools on different levels of abstraction.	PiSDF graph combining different implementation and profiles of the motion planning algorithms have been described in PREESM to demonstrate the increase in the level of abstraction, being possible to manage the HW/SW execution of the application thanks to ARTICo^3 and MDC combined architecture.
PE3. Development of a (self-) adaptation methodology and supporting tools. **Need**: efficient support of architectural adaptivity, according to radiation effects and harsh environmental conditions.	Self-healing and run time adaptation features.	Transparent redundancy mechanism and adaption to special circumstances of the scenario (solar storm, sand storm…) are included in the M36 demonstrator.

4.2. Description of available commercial solutions

In the Planetary Exploration scenario state-of-the-art design tools, as well as redundancy and monitoring oriented tools have been considered. The PE use case is focused in autonomous scalability and repair-oriented reconfiguration at computation level, thus commercial tools providing these capabilities have been explored in order to evaluate the real contributions of CERBERO tools and technologies to the space industry.

4.2.1. Model-based design

The space business has always been known for being a very slow market, reluctant to changes and attached in many ways to legacy, well proven solutions. For this reason, the development of software and digital products in this context follows traditional design methodologies.

When model-based approach solution has been explored in this domain, a few open-source toolchains were available to facilitate the design process. Capella [CAPELLA] and Papyrus [PAPYRUS] are both based on Eclipse that guarantees a high usability. Moreover, Papyrus has a robotics extension with code generation features capable of transforming models of software and deployment specifications into actual code. TASTE [TASTE] is also an open-source toolchain that targets embedded, real-time systems created under the initiative of the European Space Agency. It is focused on FPGA development and capable of automatically creating device drivers and VHDL skeletons for the hardware parts of the system.

The major issue with these tools is their level of maturity, especially in the case of TASTE. *All three applications could benefit from extensive documentation, deeper tutorials and example projects*; in other words, factors that *build a larger community around them to foster their continuous improvement and achieve further functionalities* as code instrumentation, scheduling analysis, etc.

4.2.2. Redundancy

Most part of the market with respect to the ruggedization of in-flight digital hardware designs is occupied by two proprietary tools from the main FPGA and ASIC development frameworks: Synopsis Synplify Premier [SYN] and Mentor Graphics Precision Hi-Rel [MENTOR]. These tools can implement different types of redundancy techniques like block-based triple modular redundancy (TMR) or distributed TMR in order to increase the resiliency of the design to radiation failures.

For Xilinx FPGAs, the TMR tool [TMRT] from this FPGA vendor is also an option. The problem with this tool is that *it only targets the current space-qualified FPGAs and it does not work with UltraScale or UltraScale+ FPGAs.* Targeting this technology there exist a specific IP for implementing a triple-redundant voting scheme on the MicroBlaze microprocessor [TMRM].

These tools are constrained to only implement static redundancy, and it is not possible to change the degree of ruggedization or activate and deactivate this feature during the runtime.

4.2.3. Monitoring

In order to get information about the internal status of systems, different tools are typically used in the space industry.

Xilinx FPGAs have special IPs for inspecting some internal parameters of the design: on one hand the Xilinx System Monitor allows the designer to inspect operating conditions such as supply voltage levels or temperature. On the other hand, the Xilinx SEM IP core can perform SEU detection, correction and classification by utilizing Internet content adaptation port (ICAP) and FRAME_ECC primitives, as well as fault emulation by injecting errors in the configuration memory of the device.

Typical hardware debuggers include Xilinx Integrated Logic Analyzer and Synopsis Identify for a general use with a wider set of vendors, both being capable of instrumenting the design and define internal triggers in order to inspect its behavior under operating conditions.

Finally, the RAPITA Verification Suite comprises some of the most powerful tools available in the market for software debugging. Some of them are:

- RapiTime is used for execution time analysis for critical software. It identifies the timing behavior of each function, helping the developer to choose which sections of the code to modify in order to optimize the overall timing behavior.
- RapiTask targets RTOS scheduling visualization, giving information about task-level timing behavior by collecting timing data while the code is running.
- RapiTest is focused on functional testing, providing a framework to easily implement system/unit tests, integration tests, or qualification tests.
- RapiCover: oriented to obtain structural code coverage data.

Hardware and software monitoring tools are well differentiated, since traditionally they are intended to be applied in very separate domains. Besides, *specific tools for the instrumentation of model-based designed systems are not deeply introduced in the space industry design flows*.

4.3. Results benchmarking

CERBERO tools and technologies providing reconfiguration features that allow to implement robotic arm control in an easy and straightforward manner. Implementation of similar control using traditional design flow requires a lot of effort, or even would not be possible because relevant commercial tools do not provide necessary features.

PREESM turned out to be a valuable prototyping tool for dataflow applications with a *remarkable support page.* It extends the basic functionalities of considered frameworks by means of the integration with PAPIFY and PAPIFY VIEWER for *automatic*

instrumentation and with SPIDER for *dynamic mapping and scheduling*, allowing designers to implement optimization in their models.

In summary, PREESM offers similar features to other model-based design frameworks, but with the benefits of being integrated with the rest of the tools:

- HW/SW instrumentation provided by PAPIFY/PAPIFY VIEWER;
- Modeling a reconfigurable behavior thanks to SPIDER, that provides new values for the parameters of the graph at runtime.

None of the evaluated commercial tools can provide Dynamic Partial Reconfiguration capabilities, making development of hardware adaption at runtime impossible. This means that following the traditional approach distributed or block-level TMR could be implemented in the accelerators, but this redundancy will not be dynamically adjustable.

Using ARTICo³ architecture for the execution management of the application comes with a lot of important benefits that have been successfully implemented: *customizable redundancy that adapts the level of ruggedization to the detected amount of radiation failures in operating conditions, transparent scalability of the hardware accelerator, and the possibility of switching between different implementations of the motion planning algorithm*, moving the execution flow between HW and SW, or even of *instantiating MDC accelerators to achieve mixed grain adaptivity*.

We can conclude that the combined use of ARTICo³ with MDC provides unique reconfiguration capabilities, being hardly comparable with commercially available solutions. Its main features are:

- the transparent scalability and redundancy;
- the possibility of supporting/exploiting different flavors of reconfiguration;
- the possibility of supporting algorithmic diversity that benefits from the different trade-offs held by different approaches to the same problem.

Last, the role of the Adaption Manager must be mentioned. It *holds the decision-taking intelligence and performs the evaluation of all Key Performance Indicators trade-offs of the system* in order to trigger adaptation. Such capabilities have making him exceptionally useful with no equivalent in legacy tools. The adoption of the CERBERO tools has facilitated the definition of this Adaption Manager, despite its description has being tailored for this specific scenario.

4.4. Tool Evaluation

CERBERO tools and technologies were found to be a powerful solution to overcome the main challenges of the Planetary Exploration scenario, and they noticeably help the developer to implement essential functionalities and capabilities of the demonstrator. The main considerations and most important issues that were found during the work are listed below:

- PREESM website offers a good number of tutorials showing its integration with PAPIFY and SPIDER in beta exercises along with a set of basic and extended functionalities; but it would be beneficial to update this website with tutorials describing integration with ARTICo³ and MDC. The tutorial of the integrated flow from PREESM down to hardware has been distributed at M33, but it is still limited to one example only.
- The support given in the GitHub page of PREESM is very satisfactory, often receiving an answer by an expert or a member of the community within a couple of hours after posting the issue.
- Due to the nature of the reconfigurable region of ARTICo³ architecture, sometimes slots are too large to show the benefits of transparent scalability or redundancy mechanisms in some of the smaller FPGAs. Addressing this issue would improve portability of target applications to devices other than Zynq UltraScale+.
- ARTICo³ include tutorials for the creation of a reference design for a new board which is a very convenient feature, but it would greatly improve usability to provide native support for a wider set of boards.
- MDC is intended to build coarse-grain reconfigurable systems. Hence, similarities in the high-level modules of the dataflow description must exist. Nevertheless, the tool can be used to provide different form of diversity, i.e. same application with different working points. In this case, it is responsibility of the user to modify the input dataflow to maximize the benefits of using MDC. It would be favourable to have examples on extra-functional oriented reconfigurability too in the MDC support material.
- There are a few frameworks (see section 4.2) typically used for the development of in-flight hardware and software applications. The reason is that traditionally most customers demand strict protocols for architectural design, netlist validation, post-layout verification, I/O timing, code coverage analysis, rule-checking, etc. It would be desirable for CERBERO tools to facilitate integration with commercial frameworks in order to guarantee the mandatory requirements for in-flight HW/SW design.
- It must be noted that CERBERO computing level tool providers decided to keep individual tool identity, thus the design flow still requires intervention from the developer in some steps of the process. To overcome this limit, there are back-ends and explicit links among tools. For example, it is possible to instantiate PAPIFY monitors automatically from PREESM and to have them implemented in MDC compliant architectures, which are fully compliant and can be integrated into ARTICo³ slots using different extensions in MDC backend.

As a summary, along the project time-frame there have been different tutorials on combined usage of all the adopted tools, but still new users may require a learning curve. With respect to the documentation, to engage people outside the project, it would be beneficial to have more documentation with an integrated step-by-step example going

from PREESM down to HW deployment, covering all tools and technologies, in order to better show the capabilities of the framework and have a clearer starting point for new developments.

5. Ocean Monitoring use case

5.1. Evaluation on initial requirements

Table 1. CERBERO High Level Requirements

Requirement	Validation demonstration	Evaluation
OM1. Provide complete design cycle from system level design to mapping over COTS SW and HW components. **Need**: Faster development cycles and cost reductions due to early-stage system characterization.	Models of alternative hardware platforms to predict data throughputs.	Models of various hardware configurations using DynAA have been successfully implemented, which allowed to predict power and video processing data throughput before implementation.
OM2. Develop integrated open-source or commercially available toolchain environment for cyber-physical systems, with focus on fast prototyping due to high-level system characterization. **Need**: Provide multi-objective design space exploration and multi-view analysis at the system level, facilitating development cycles and reducing time to market. Increase reuse among cycles, along with quality and verification level by fast prototyping from high level of abstraction directly to working real time applications.	Building adaptive camera and monitoring hub from generic hardware and software components. Prototyping interface for algorithm assessment and configuration.	COTS hardware and open-source software components have been used to assemble an adaptive camera with an embedded adaptation system. The high-level design allows these open source components to include future acceleration or power-reduction technologies using CERBERO tools. The Java-based prototyping interface has enabled fast iteration and evaluation of video processing technology for inclusion and embedding the monitoring hub and in the adaptive camera.
OM3. Development of a SW (self-)adaptation methodology with supporting tools. **Need**: Efficient support of functional adaptivity, according to system, human and environment triggers.	Adapting to different environment conditions, such as ambient lighting, temperature.	The demonstrator includes implementations of the adaptation cycle in both the adaptive camera and the monitoring component, with feedback connections between them.

5.2. Description of available commercial solutions

In the Ocean Monitoring scenario different image and video processing libraries and tools have been considered in order to gain from the off-the-shelf implementations. One of the central aspects of OM use-case have been the enhancement of situational awareness of the machine and human operator. This is based on the augmentation of data received from different sensors including optical sensors. The OM use-case has a need to efficiently process image frames and build on optimized fundamental implementations to rapidly develop advanced image processing algorithms. This approach follows the CERBERO paradigm of low cost, incremental and rapid prototyping.

5.2.1. Video processing

There are a number of video processing pipeline tools. These broadly divide into three categories: hardware-enabled solutions, interactive tools for offline enhancement, and pipeline-based tools and libraries designed for embedding.

The commercial solutions include the LYYN [LYYN] hardware systems, and interactive tools applications like Amped FIVE [AMPED]. *Neither of these is directly suited to ocean monitoring*: the *interactive tools because they require significant user management in real time*, and *LYYN* because is essentially a *black-box proprietary implementation of algorithms on hardware*, and *does not include any adaptation mechanisms* to respond to environmental conditions.

It is also worth mentioning ffmpeg [FFMPEG], which is among the most widely used video processing systems today, as it is a ubiquitous library and toolset that often powers other applications internally. ffmpeg (and its library components like avfilter) has a natural data-flow architecture that is readily adaptable to CERBERO acceleration, although it *needs care in use as some builds incorporate GPL-licensed components that make commercial usage problematic*.

As with the other video processing systems, *ffmpeg doesn't directly support adaptation* (although a limited amount of adaptation, such as automatic white balance) is often built into the camera sensor chips directly. They are also *poor at load management*: ffmpeg, for example, uses a simple thread tool to distribute processing. On a CPU-based system, this is usually fine, as it delegates to the operating system. On a GPU, adaptation to load usually *requires proprietary knowledge of and access to hardware internal components*.

5.2.2. Image processing

As with video processing, there are commercial tools for image processing, but these are *primarily designed for interactive use*. There are software libraries that are used extensively, notably the commercially supported VisionWorks from NVIDIA, that depends on the CUDA libraries and NVIDIA GPU accelerators.

The other library that is used widely is OpenCV, which includes contributions from Intel, NVIDIA, AMD, and other members of the open source community. Like VisionWorks, it enables pipelines of image processing algorithms to build from relatively standard

components like edge detectors, histograms, and a wide range of image processing primitives, both low-level and high-level.

As with the video processing systems, *adaptation is limited*. GPU-based accelerators, like those used by *VisionWorks through its CUDA library, do adapt load, but do so in a black-box way based on proprietary drivers*. OpenCV also supports CUDA, but a second API for acceleration, the Intel-backed T-API, uses OpenCL to accelerate instead — these are more appropriate for the Qualcomm Dragon boards, which use a proprietary GPU that supports OpenCL. Tools like SPIDER do not exist to handle load adaptation for use of these high-level libraries (and often, they are not necessary, as the drivers provide some automated adaptation built-in). *Some of the more primitive GPU's, like the MALI GPUs on ARM-based systems do require an element of performance tuning, at a very coarse level, like worker queue sizing*.

In the end, *the Ocean Monitoring use case found that a primary design constraint was the video processing capacity, especially for higher resolution video*. Although there is commonly (not always) H264 video compression hardware support, H264 is not an especially good data format for some scenarios, as intermediate frames can be compromised by compression artefacts. MJPG is more commonly used by sensors. Compressing high-resolution video to H264, especially from multiple lenses, is at the limits of the performance capacity of embeddable hardware – and is not always necessary for early-stage prototyping. Our subsequent design performs a minimum of image processing on embeddable devices, sufficient to enable effective adaptation, but not processing every frame at full resolution.

5.2.2.1 Sensors

Sensor systems are important for the OM use case. They come with standards and resulting constraints. This is also the case for adoption of commercial, off-the-shelf technologies (COTS). For example, standard camera sensors might have a CSI interface or a USB interface. Most lower-throughput sensor systems are designed to work with GPIO pins, typically using I2C. This wide standardization meant that *there were many sensor systems with hardware compatibility, although many had additional environmental limitations for the Ocean Monitoring use case, such as temperature and pressure sensors not designed for underwater use*.

5.2.2.2 ROVs

There are a *wide range of ROV platforms, most of which are entirely proprietary*, although crowd-funding and open source are beginning to change the market characteristics. These vary widely in prices from a few hundred euros to several million, depending on the depth characteristics, software, need for autonomy, and so on. Generally speaking, the bigger, the deeper, and the more rugged, the higher the costs. Of note here is the Blue Robotics platform, which uses open source and standardized components extensively, including Arduino and Raspberry Pi hardware components.

5.3. Results benchmarking

With reference to the comparative scenarios, CERBERO-developed tools for image and video processing are qualitatively more costly, due to their requirement for a new

implementation, where in the Ocean Monitoring scenario OpenCV and VideoWorks, among others, have been used to gain from off-the-shelf implementations.

The most immediate gain from the CERBERO tools was using DynAA to assess the algorithm pipeline. This determined early, and correctly, the primary crucial factor of the video encode (and to some extent decode) processes in the data throughput. Having access to this guiding information at a proof-of-concept stage enabled us to move forward with confidence. In summary, DynAA offers similar features to other design-time simulation tools, but with the advantage that its use of Java and open systems for units and measures:

- Provided a speedy development platform for generating the initial models
- Accommodated the data quantity types which were crucial to the model value

Using AOW to build a rough model for predicting the best combination of enhancement techniques also worked well. A large open data set [D1.3] has been prepared, which enabled training data to build a model. Despite there is still need to refine quality measures to accommodate the scarcity of human-judged underwater image assessments, the development process worked very smoothly, and it is clearly remarkable how this process can be productionized and adapted to different customer requirements.

AoW offers a good basis for building models to guide adaptation by optimization.

Finally, the adaptation cycle – with a small set of refinements to handle hierarchical systems and allow for the pre-built AOW models to guide the adaptation manager – was an excellent architectural pattern for the Ocean Monitoring use case. Although this is a framework rather than a tool (and therefore cannot be benchmarked) it noticeably reduced the complexity of early stages of the design process, by giving the use case a proven structure to work within.

5.4. Tool Evaluation

One of the challenges for the Ocean Monitoring use case was adapting the CERBERO tools also to available and appropriate COTS (commercial off-the-shelf) components, where selection was constrained by the video processing needs and by the customer-responsive development process, i.e. for the potential end users.

The detailed requirements for the Ocean Monitoring use case developed through a customer discovery process [BLANK], and with a limited budget. These constraints meant that the use case needed rapid prototyping at a system level rather than a component level. Prior to scaling the technology, increasing processing capacity or reliability, and especially during the earlier stages of development, tools that required

more development effort were not as valuable. Once the design started to converge, it became easier to determine where the value in tools with higher development costs lay.

- **DynAA** was valuable at the early stages of development – it enabled the use case to validate hardware performance capacity, and do a preliminary verification that a particular hardware platform would meet the processing capacity needs. However, we were not able to use DynAA at runtime – this function was developed later in the project, and after refining the specification, we reduced our use of Java at runtime, particularly within the adaptive camera, to optimize data throughput. In fact, using DynAA at design time alerted us to the critical aspect of this KPI.
- **AOW** is used to optimize algorithm combinations for the adaptive camera, allowing the adaptation system to dynamically select an appropriate suite of enhancement techniques (even though the model used to drive this adaptation was developed at design time). We use AOW to build a runtime model that can be used for adaptation.
- The **adaptation cycle** was very well-suited to the OM scenario, and enabled an effective split between a lower-level adaptive camera system and a higher-level monitoring hub. We have proposed small extensions [D6.3] to the cycle to help accommodate the hierarchical structure of the OM use case.

6. References

[Ignite]	https://ignite.apache.org/
[Mathworks]	https://mathworks.com/
[Modelica]	https://www.modelica.org/
[PLEXIL]	http://plexil.sourceforge.net/wiki/index.php/Main_Page
[ROSPlan]	https://kcl-planning.github.io/ROSPlan/
[SYN]	https://www.synopsys.com/implementation-and-signoff/fpga-based-design/synplify-premier.html
[MENTOR]	https://www.mentor.com/products/fpga/synthesis/precision-hi-rel
[TMRT]	https://www.xilinx.com/products/design-tools/tmrtool.html
[TMRM]	https://www.xilinx.com/support/documentation/ip_documentation/tmr/v1_0/pg268-tmr.pdf
[CAPELLA]	https://www.eclipse.org/capella/
[PAPYRUS]	https://www.eclipse.org/papyrus/
[TASTE]	https://essr.esa.int/project/taste
[BLANK]	S. Blank “Four Steps to the Epiphany: Successful Strategies for Products That Win”, K & S Ranch, 5th Edition.
[T-REX]	http://wiki.ros.org/trex_executive
[LYYN]	https://www.lyyn.com
[FFMPEG]	https://ffmpeg.org
[AMPED]	https://ampedsoftware.com/five