Information and Communication Technologies (ICT) Programme

Project Nº: H2020-ICT-2016-1-732105

D6.1: Demonstration Skeletons (Ver. II)

Lead Beneficiary:	8 - AI
Work Package:	6
Date:	19-08-2019
Distribution - Confidentiality:	[Public]

Abstract:
This document takes as input the technical requirements and scenario specifications prepared in WP2 and the results of the modeling and adaptation activities of WP3 and WP4 respectively, to prepare demonstration skeletons with a consistent set of tools and procedures applicable to the proposed use-case scenarios. This set undergoes a first compatibility test that guarantees the proper interconnection of the different entities prior to its customization for each of the use cases. The goal is to provide demonstration skeletons on top of which the different models, processes and applications of each use-case shall be built and tested.

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 Appliques de Rennes	INSA	FR
6	Universidad Politecnica 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**
Francesca Palumbo	UNISS
Antonella Toffetti	CRF
Francesco Palma	CRF
Joost Adriaanse	TNO
Giuseppe Meloni	AI
Maria Katiuscia Zedda	AI
Leszek Kaliciak	AS
Stuart Watt	AS
Ayse Goker	AS
Hans Myrhaug	AS
Pablo Sanchez de Roja	TASE
Sergio Tardón	TASE
Alfonso Rodriguez	UPM
Eduardo de la Torre	UPM

Document Revision History

Date	**Ver.**	**Contributor (Beneficiary)**	**Summary of main changes**
01-July-19	0.1	AI	1st Draft
03-July-19	0.2	TNO	Update ST UC
08-July-19	0.3	CRF	Update ST UC
15 July-19	0.4	Thales	Update PE UC
24 July-19	1.0	AI	Section 1, 2, 3,4 revised
25 July 19	1.1	AS	Update OM UC
24 July-19	1.1	AI, UNISS	Section 5 revised
26 July 19	1.2	UPM	Tools details for PE UC

Table of Contents

1. Executive Summary

This document is the final release of the demonstrator skeletons for prototyping activities in CERBERO use-cases. The document is an updated version of D6.7 released and accepted at M15. The aim of the deliverable is to present full-fledged demonstrators of CERBERO use-case scenarios.

1.1. Structure of Document

This document presents the general skeleton for demonstrating the final iteration of the CERBERO use-case scenarios, the customization of such skeleton for each use case, the testing environment and the planned prototypes expected to be delivered at the end of the project.

Use case challenges, goals, and system architectures have been described in D2.1 (and its previous version D2.3 and D2.4), while the present document clarifies and better analyses how CERBERO tools/technologies are used within the use-cases. The idea, given a generic skeleton description in Section 2, is to provide an overview of how the generic skeleton is going to be customized into the different demonstrators (see the Customization of the Skeleton in Sections 3-5). For each use case we also describe:

1. how CERBERO tools and technologies are going to be used to design, verify and access (sub-)parts of the demonstrators;
2. the interface and components that are part of the demonstrators (including off-the-shelf ones);
3. the validation set-up that we are putting in place for assessment purposes.

As happened in the previous project iteration (M1-M18), the final set-up and assessment purposes of each demonstrator, the toolchain feedback, and the achieved results are further elaborated on by respective use-case demonstrator deliverables, namely the D6.2 for planetary exploration, D6.3 for ocean monitoring, and D6.4 for smart traveling at M36.

Compared to the previous version of the deliverable D6.7, the CERBERO-compliant self-adaptive system skeleton (Section 2) has not be changed, whereas the customization of the skeleton for each use case has been revised and aligned with the foreseen final demonstrators. Furthermore, some potential functionality, that is not going to be included in the final demo, has been added to the use cases, in order to give a clear overview of the full set of features achievable with the proposed approach.

In order to simplify the reading and maintain the deliverable self-consistent, the section that has not been changed in the new version is in gray.

1.2. Related Documents

- D2.1 – D2.3 – D2.4 – Description of Scenarios (Ver. I, II, III)
  Customization of each skeleton follows the use case scenario description defined in D2.1 – D2.3 -D2.4
- D5.2 - D5.6 – Framework Components (Ver. I, II)
  Adopted components of the CERBERO framework are described in D5.2 and D5.6

- D6.2 - D6.8 - Space Demonstrator (Ver. 1, final), D6.3 - D6.9 - Ocean Monitoring Demonstrator (Ver. 1, final), D6.4 - D6.10 - Smart Travelling Demonstrator (Ver. 1, final)

Skeletons will be used to guide implementation in demonstration activities to be described in D6.2-4 and D6.8-10.

1.3. Related CERBERO Requirements

Deliverable D2.2 of the CERBERO project defines the final list of CERBERO Technical Requirements (CTRs) the project should achieve. Each of them is referenced with a unique identifier ranging from 0001 to 0020.

The CERBERO demo skeleton aimed to assess CERBERO technologies, does not cover directly the technical requirements that have been fully covered by the other WPs (2, 3, 4 and 5) more focus on developing these technologies. The only non-technical requirement covered by the demo skeleton is CERBERO-0017 'CERBERO Use Case providers SHOULD check and provide timely feedback on the usability of CERBERO tools and framework', in fact the incremental development of the demonstrators starting from the skeleton, which covers by definition all the technologies and tools developed in the project, certainly allows for providing feedback on the utility of the different parts.

2. CERBERO-compliant Self-Adaptive Systems Skeletons

CERBERO main outcome is its continuous design framework for self-adaptive cyber-physical systems (CPS). One-fit-all solutions would certainly not apply in different contexts. Nevertheless, it is possible to define a common frame, explaining:

1. what are the generic elements of CERBERO self-adaptive systems (see Figure 2.1);
2. what techniques and tools (of the CERBERO framework in particular) are used to design and deploy the different composing elements of each system;
3. what methodologies guarantee the proper interconnection of the different entities prior to the customization of each of the use case.

CERBERO targeted self-adaptive CPS range from stand-alone to system of systems, with the same continuous design framework. The skeleton below depicts an overview of the deployable self-adaptive systems. It includes basically the runtime elements. Design time strategies, leading to the definition of the different sub-parts, are described in each use case paragraph since they are necessarily target and scenario specific. A demonstration of the design time support offered by the CERBERO framework and framework components is presented in D5.2, D5.6, D5.1 and D5.7.

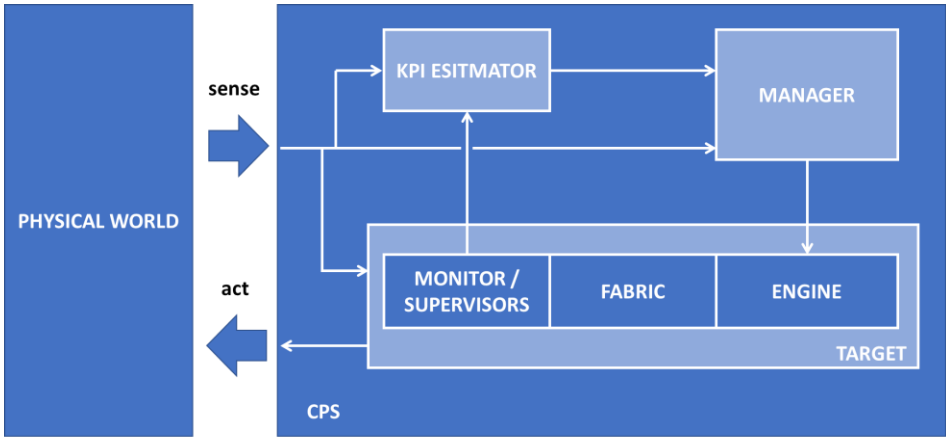

Figure 2.1: CERBERO skeleton

The elements composing the skeleton in the Figure above are:

- The ***physical word***: it can be user, environment or a physical model;
- The ***CPS*** contains the following abstract functions (while the actual realization could be distributed between CPS sub-systems):
    1. ***KPI ESTIMATOR***: The Key Performance Indicators (KPIs) library is defined in WP3, which gives also way of modeling and evaluating KPIs at runtime. As stated in D3.1 and D3.4 the library is then customized on use case needs. KPIs represent a way to evaluate how close the CPS currently is to its expected goals/performance.

2. ***MANAGER***: The goal of this element is to define if adaptation is needed, according to the distance between the evaluated KPIs and the final system goal. Self-adaptivity management is an outcome of WP4, but it uses runtime models of the system and its performance from WP3 and runtime tools/toolchain developed and aggregated in WP5.
3. ***TARGET***: It is a generic element, i.e., it can be hardware, software or hybrid, and it can be also intended as a system or as a system of systems. It is composed of three abstract elements: the engine, the fabric and the monitors. The *engine* is a target dependent element that physically puts in place the actions to execute the decisions of the *MANAGER* in terms of adaptivity (it could be a piece of hardware or a software agent). The reconfigurable *fabric* carries out the functional tasks according to the taken decisions. The HW *monitors* and the SW supervisors (monitors) sense the fabric status. The fabric, engine, and monitors/supervisors are tightly linked and are normally technology dependent. They are use case specific and generally defined in WP2 and WP6 as a result of the use case needs. Fabric, engines and monitors have also been implemented/extended in WP4, if not already available as off-the-shelf components.

In the following sections, which are use case specific, we are going to customize the skeleton above mentioned according to the CERBERO technologies, strategies and components used, starting from the classification of the types of triggers and adaptivity used.

Adaptivity can be classified as follows based on the reasons for the required adaptivity:

- **FUNCTIONALITY-ORIENTED:** Adaptation needed to offer different functionalities over the same substrate or to maintain correct functionality, e.g., because the CPS mission changes, several functions running on the same HW interchangeably, or the data being processed changed and adaptation is required. It may be parametric (a constant changes) or fully functional (algorithm changes). For example, you can change the type of filtering according to the type of noise in order to provide the required functionality.
- **NON-FUNCTIONAL REQUIREMENTS-ORIENTED:** Functionality is fixed, but system requires adaptation caused by non-functional requirements, such as performance or available energy. For example, filtering precision could be reduced in case of low battery.
- **REPAIR-ORIENTED:** For safety and reliability purposes, adaptation may be used in case of faults. Adaptation may add self-healing or self-repair features. For example, HW task migration for permanent faults, or scrubbing (continuous fault verification) and repair.

Adaptivity triggers can come from several sources:

- **ENVIRONMENTAL AWARENESS**: Influence of the environment on the system, e.g. daylight vs. nocturnal, radiation level changes, inspection of surroundings (including status of users/operators) etc. Sensors are needed to interact with the environment and capture conditions variations.

- **USER-COMMANDED**: System-User interaction, e.g. user preferences. Proper human-machine interfaces are needed to enable interaction and capture commands.
- **SELF-AWARENESS**: The internal status of the system varies while operating and may lead to reconfiguration needs, e.g. chip temperature variation, low battery. Status monitors are needed to capture the status of the system.

3. Smart Traveling for Electric Vehicles

3.1. *Customization of the Skeleton*

Context definition: supported adaptivity type(s) and related triggers.

Self-Awareness	In the Smart travelling use-case, an electric vehicle is simulated. Sensors and monitors are used to keep trace of system parameters: - GPS coordinates of the car - Speed of the car - Current level of the battery - Current energy consumption (engine, heating, lights)
User-Commanded	Driver's intentions are monitored. The user will input its intentions like destinations and preferences on specific KPIs (such as cost, time for charging, scenery, requested facilities at charging stations, etc.) via an essential User Interface (Abinsula HMI). The users should also select preferred itinerary if multiple itineraries are possible to reach the destination.
Environmental Awareness	Environmental conditions are monitored. The system is aware of its current position via a GPS sensor and is optionally aware of the availability and status of charging stations (which will require Internet connection to central charging system to retrieve status of each relevant station). Furthermore, the Internet connection could be used to retrieve traffic jams or road blockage on possible and/or planned routes. Possible routes and charging facilities will be preloaded in the system and optionally updated via an Internet connection to a central system. To monitor the status of the driver in order to adapt the advice according to this status, a special sensor will be added to the CRF Driving Simulator. The sensor will detect the opening and closure of the eye lids and trigger "driver tired" signal in the case the eye lids are closed for a certain amount of time. This functionality could be used in the scenarios to signal the driver and perform re-planning so driver can rest at the closest charging station or resting place.

Functionality Oriented	The functional parameters that can trigger adaptation in the Smart Travelling use case are: - Battery load (low detected battery level can for example trigger to charge earlier than planned) - GPS coordinates (can trigger re-planning of route in case driver does not follow advised route) - User request to plan route to destination - Update on charging station status (occupancy or out of service messages from charging infrastructure could trigger re-planning of route in case station was planned to be used) - Update on road blockage or traffic jams (could trigger re-planning in case road segment was on planned route) See also adaptation strategies in D2.1 section 2.2

Non-Functional Requirements Oriented	Non-functional parameters may trigger adaptation (see battery level above), but normally in this scenario the implemented adaptation is typically functional, since it requires re-planning.
Repair Oriented	There is no need for fault tolerance support in the Smart Travelling use case, except from handling failures in the external environment (road and charging infrastructure), for which messages such as failures in charging stations or road congestions/traffic jams are received. In any case, as the non-functional requirements above, such conditions lead to functional adaptation from the vehicle perspective, meaning re-planning.

<u>CERBERO skeleton implementation in the Use Case: description of the runtime elements.</u>

Element	Involved Components	Interconnection Strategy
Physical Element Target	In this scenario the target and the physical element are basically coincident. In fact, the physical environment is simulated by Driving Simulator, which represent the target we are using. Currently the Target is composed of the SCANeR Driving Simulator (containing most of the sensors), DynAA (to directly communicate with the car via provided APIs), MECA (to calculate predictions on routes to take) and the connected HMI (Human Machine Interface), which is used to interact with the driver and reactively or proactively provide suggested routes to the driver. The Driving Simulator also is representative of a real electric vehicle in case the same sensor interfaces would be supported as provided by SCANeR.	The CERBERO tools are "connected" to the physical environment via the provided SCANeR interfaces. Between the CERBERO tools DynAA and MECA direct (HTTP REST) interfaces are defined. The demonstrator will not yet use the CIF interface, but there is a separate trial performed using DynAA to interface via CIF towards AOW, explained in the D5.3.
Used KPIs	The supported KPIs are (see D2.1 for generic KPIs): - Response time to trigger and Latency (used in system-in -the-loop simulation mode) - Energy (load of battery and usage of energy by car) - Cost (of electricity at charging station) - Travel time - Possible other user preferences - Driver status (if driver is tired the system should try to ensure the driver gets rest as soon as possible) Within the Smart Travelling use case evaluation of the KPIs will be performed by MECA. Collection and prediction on KPIs will be performed by DynAA (using engine and battery models).	Information for the KPIs is derived from sensors in the car (SCANeR), information from the environment (charging infrastructure info) and information from the driver (commands via HMI and driver status sensor). There also will be KPI related user preferences defined in MECA, used for itinerary ranking. In the final demonstrator an additional HMI is added to visualize the dynamic info received from SCANeR, plot the map with calculated itinerary and collect use case specific commands form the driver (e.g. destinations and itinerary selection).
Manager	MECA and DynAA are used as Manager of the CPS, where MECA makes decisions on routes based on available KPIs and DyNAA provides support to MECA by calculating predictions on	The interfaces between tools will be direct (HTTP REST).

	possible routes to take. For interaction with the driver the Abinsula HMI is used.	

3.1.1. CERBERO Design Time support

In the Smart Travelling use case, the CERBERO tools are used mainly for system deployment. CERBERO tools allow for implementing complete CPS with minimal effort. Using DynAA and MECA tools directly connected to SCANeR avoids the need of building the demonstrator from scratch. Simulation in the loop allows early stage verification of new system modules (i.e. battery modules or the user interfaces), which facilitates to take decisions for the real implementation helping in the overall design process.

3.2. Testing Environment

The main testing environment for the Smart Travelling use case is the CRF Virtual Driving Simulator test site in Turin. To optimize the activities and minimize the travelling, beyond the CRF site, a second test site with SCANeR driving simulator software is set up at TNO (in The Hague) where DynAA, MECA and the HMI are tested. For this purpose, Oktal/AVSimulation has provided the project with a test license of SCANeR.

This second test site is used to develop and validate the message interface used by SCANeR. Once components were validated, they were shipped and integrated in the test site in Turin. In the meanwhile, Smart Travelling Use Case partners worked together remotely to eliminate possible incompatibilities and difficulties during the integration phase, defining specifications carefully.

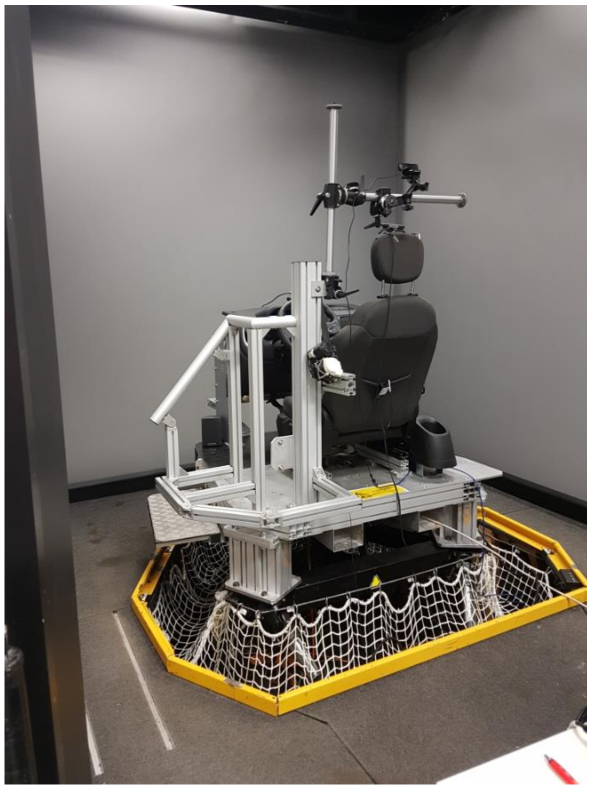

Figure 3.1: Driving simulator at CRF in Turin

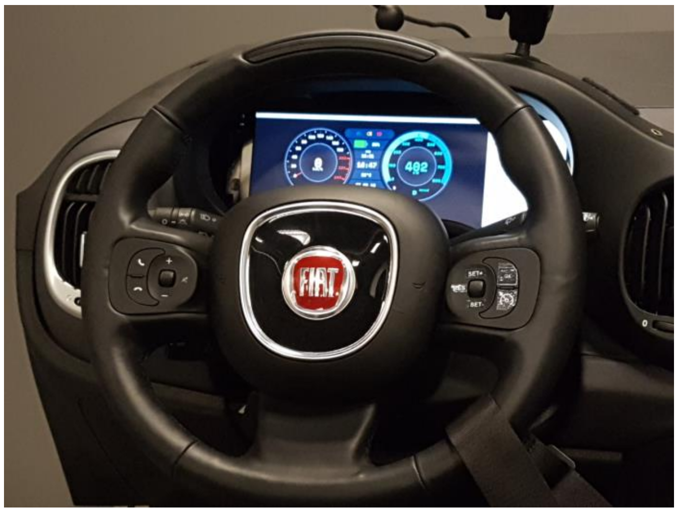

Figure 3.2: One of the two HMI screens showing speed, voltage and state of charge

For integration of the CERBERO software components (DynAA, MECA and HMI) and HMI hardware into the CRF Driving Simulator, a number of integration sessions were planned in the first half of 2019. During these sessions the software was installed on added hardware servers and end-to-end integration tests were performed.

In the scenario description document (D2.1), three different individual use cases were described. In the final demonstrator set up the scenarios are mapped on the actual geographic location of Turin and its close surroundings.

The tests that will be performed include:

1. Simulation of electric motor during a driving simulation run.
2. Simulation of battery performances during a driving simulation run.
3. Monitoring of vehicle and user data and triggering of interaction between SCANeR, the HMI, MECA and DynAA based on those data. The set up will allow CRF to perform the defined scenarios, where MECA, with the support of DynAA, will calculate itineraries and the HMI will perform required driver interaction for the scenarios.
4. Synchronization and fusion of generated logging data of data recorded using SCANeR and Eye Tracker system into a single (synchronized) simulation output file.

The test scenario will include a number of driving simulation runs in which the electric vehicle is simulated, using the CERBERO simulation modules, and where the indicated test items above mentioned will be verified.

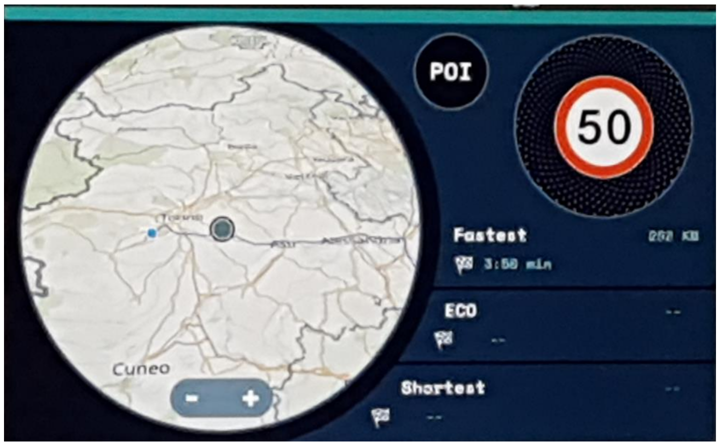

Figure 3.3: The HMI screen showing the geographic map and interaction options with MECA

Itinerary calculation will be done by DynAA, were both the HMI and MECA will use a geographic map of Turin they retrieve from OpenStreetMap and SCANeR also uses OpenStreetMap as input for the configuration of routes and sceneries.

3.3. Prototype Validation

For validation of the prototype Smart Travelling scenarios as defined in D2.1 (Scenario Description) will be used. During the prototype validation, the functionalities of the developed interfaces in the skeleton will be tested by executing the relevant parts of the test scenario.

For the data fusion application real log files from a test run performed on the CRF simulator will be merged into an overall logfile regarding SCANeR and Eye Tracker system during pilot test sessions. The benefit for CRF provided by the usage of the CERBERO tool is tangible, CRF had to dismiss the tool they used before because of the huge computational time and precision of alignment which caused mistakes difficult to be highlighted.

4. Self-Healing System for Planetary Exploration

4.1. *Customization of the Skeleton*

Context definition: supported adaptivity type(s) and related triggers.

Self-Awareness	In the Planetary Exploration use case, the status of the robotic arm computing platform is described through hardware monitors internal to the computing infrastructure. The main information that produces self-adaption comes from: - Latency monitors - Power consumption monitors - Fault monitors
User-Commanded	The motion planning performs in an autonomous way and no mandatory parameters must be provided during operation. Starting and ending position of the movement in Cartesian coordinates may be specified by the user in order to customize the trajectory.
Environmental Awareness	The system is capable of adapting to its surroundings by means of proximity sensors that are mounted on the arm. When an obstacle is detected, the trajectory is re-calculated in order to avoid it, achieving a collision-free movement.

Functionality Oriented	The functional parameters that can trigger adaptation are: - Obstacles: Robot Control Unit will be able to re-calculate motion planning points in order to avoid obstacles in the robotic arm path. - User: requests may change the robotic arm position to a different goal position. - A change in the threshold values for the acceptable power consumption (battery) levels.
Non-Functional Requirements Oriented	Transparent scalability and virtual reconfiguration are enabled in order to perform different implementations of the motion planning algorithm different balance between area, processing time, energy consumption, accuracy, etc.
Repair Oriented	In the Planetary Exploration use case, the computing fabric may be damaged by radiation effects. The ARTICo^3 tool provides on-demand hardware redundancy and hardware monitors to check error status and ensure the correct operation of the system.

CERBERO skeleton implementation in the Use Case: description of the runtime elements.

Element	Involved Components	Interconnection Strategy
Physical Element	WidowX robotic arm	The six actuator joints present in the Physical Element are commanded via UART through the PMOD connectors of the board. Besides, environmental information is provided to the CERBERO tools by sensor interfaces.

Used KPIs	The supported KPIs include: - Latency - Throughput - Resources utilization - Availability - Energy consumption - Accuracy	Information for KPIs is derived from cyber and physical monitors. - Latency and throughput: time to perform kinematics algorithms will be measured by PAPIFY and displayed by PAPIFY VIEWER. Moreover, studies towards execution time minimization and improved computational robustness will take into account: - Resources utilization in the form of reconfigurable logic is monitored by ARTICo3 through the number of slots used. A trade-off evaluation is necessary between this parameter and the total power consumption of the system. - Availability: number of faults produced by radiation effects.
Manager	The adaption manager is implemented as part of the software application. This manager uses the ARTICo^3 runtime management software, which offers execution rescheduling capabilities similar to SPiDER but with enhanced flexibility and productivity for hardware-accelerated systems	It is embedded in the host application of the ARTICo3 processing architecture.
Target	The target for the demonstrator is an heterogenous MPSoC to be fully developed and customized using CERBERO computing-layer framework components capable of automatically deploying the performance and fault monitors over the computing fabric.	ARTICo3 and MDC are already working together through an integrated, novel design flow. At the same time, they are linked to PAPIFY and SPIDER to provide, respectively, monitoring and runtime re-allocation and re-scheduling capabilities.

4.1.1. CERBERO Design Time support

In the Planetary Exploration use case, the following CERBERO tools will be used at design time:

- SAGE-ReqV uses formal methods at design time in order to ensure the correct construction of the specification by translating a set of requirements in natural language.
- SAGE-HyDRA can be used to derive goal-oriented, correct by construction strategies from high-level model of Hybrid Systems by means of state-of-the-art techniques in AI Planning, Satisfiability Modulo Theories and Numeric Optimization.

- PAPIFY is a tool that implements an event-based performance monitoring. It integrates the Performance API (PAPI). A library called eventLib adds a new abstraction layer to PAPI enabling this way a transparent access to standard PMCs in processor cores and specific HW monitoring infrastructure (specifically added in the HW accelerators coming from MDC and ARTICo^3). At design time, it will assist in rapid prototyping by gathering runtime information that can assist in the DSE done by PREESM.
- PAPIFY Viewer is a visualization tool to monitor the actions of actors. Fired actors can be analyzed chronologically from either an actor or a processing element point of view.
- The ARTICo^3 toolchain includes a set of automation scripts capable of building already-partitioned hardware/software systems from accelerator specifications in HDL or C + HLS, and application specification in C code. The ARTICo^3 toolchain outcome is represented by the binaries required to program the final platform (FPGA bitstreams, application executable).
- The MDC toolchain combines different dataflow descriptions of the application into a unique, reconfigurable multi-dataflow model, combining the common resources of each one of them.

Further details on these tools are provided in deliverable D5.2 and D5.6.

4.2. Testing Environment

The testing environment for the robotic arm is composed by the system architecture and components already presented in D2.1 which must be validated in laboratory environment. There will be three main scenarios to perform the testing of the architecture:

- Motion planning in free space without obstacles.
- Collision-free motion planning with identified obstacles.
- Collision-free motion planning with obstacles in unknown positions.

In all of these scenarios, reconfiguration and KPI monitoring techniques are implemented.

4.2.1. Motion planning in free space

The goal of this test environment is to perform a linear movement from an initial point to a final point with the Robotic Arm. This test will verify the proper operation of the software algorithms.

By introducing a couple of points, initial and final position, the software algorithms must provide a motion planning interpolating these points. The Robotic Arm end effector must be able to reach all the points. These points are shown as a blue line in Figure 4.1.
Once all the points are calculated, inverse kinematics and iterative optimization algorithms must calculate dynamixel actuators position for each point.

Finally, dynamixel actuators position must be sent to the Robotic Arm, which must perform a movement as shown in Figure 4.1: Python Simulation of Linear Motion Planning in Free Space.

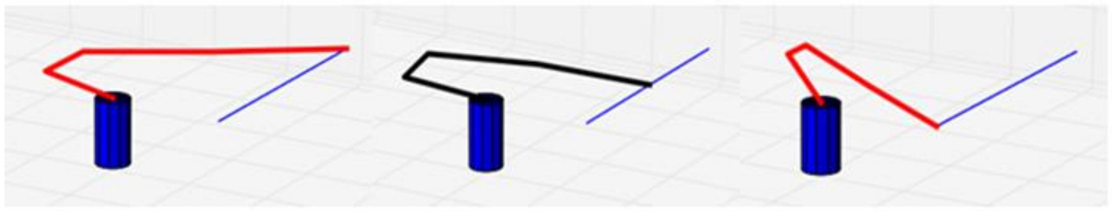

Figure 4.1: Python Simulation of Linear Motion Planning in Free Space

4.2.2. Collision-free motion planning

The goal of this test environment is complementary to the previous one. In this case, another data input is required: the obstacle position. This obstacle is shown as a red cylinder in Figure 4.2 below.

Once all the points are calculated by interpolation algorithms, they must be recalculated in order to avoid a trajectory with obstacle collision. These new recalculated points are shown in Figure 4.2 as a blue line.

Inverse kinematics and iterative optimization methods must calculate dynamixel actuators position. These positions must be sent to the Robotic Arm, which must perform a movement avoiding the obstacle as shown in Figure 4.2.

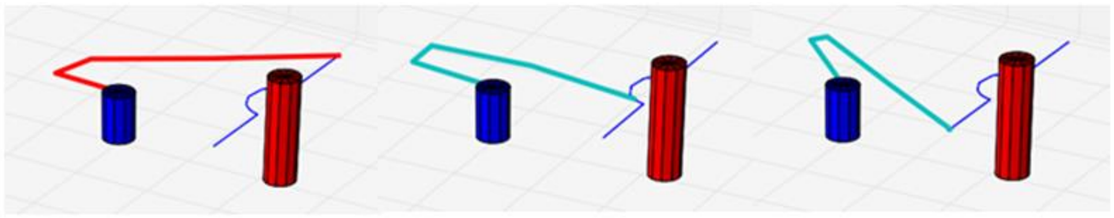

Figure 4.2: Python simulation of Collision-Free Motion Planning

4.2.3. Collision-free motion planning with obstacles in unknown positions

Additionally, the robotic arm will be equipped with proximity sensors in order to sense its physical environment and make it capable of reacting to changes in its surroundings.

This test is similar to the previous one, but the data input regarding the obstacle position is not needed. The algorithm would autonomously adapt to the presence of the object blocking the generated trajectory and will generate a new one in order to avoid the collision while maintaining a smooth movement.

4.3. Prototype Validation

The Planetary Exploration prototype will be validated by executing the Testing Environment presented in Section 4.2. During this test scenarios, the following functionalities will be validated:

- Motion planning in free space to validate calculated trajectories.
- Collision-free motion planning with obstacles in known and unknown positions in order to validate adaptation of the system.
- Monitoring of the different KPIs.
- Sensing of changes in the environment through measurements of proximity sensors mounted on the arm.
- Scalable parallelism through different flavors of reconfiguration
- Functional fault injection (multiplexor-based faulty/correct behavior selection) to trigger redundancy mechanisms at component level.

5. Ocean Monitoring

5.1. Customization of the Skeleton

<u>Context definition: supported adaptivity type(s) and related triggers.</u>

Self-Awareness	The OM system is aware of its internal status, including battery level and achieved video quality. The monitors used include the following: - current image/video quality based on pre-defined quality levels and detected from image/video illumination levels, red color deficiency, etc. - current battery status - current location based on GPS measurements - speed and thrust - processing load - storage capacity - system temperature
User-Commanded	The user can communicate and interact with the robot via its user interface. The interactions will take the form of: - manual vehicle control, e.g.: steering, adjustment of speed and thrust - controlling the autopilot settings, e.g.: setting the destination, desired speed, preferred route - requesting images/videos of specific quality
Environmental Awareness	The OM system is aware of its environment, specifically of: - surface and underwater visibility conditions - light measurement based on the light sensor - depth - obstacles - moving objects - its location and location of shores and other boats (publicly available map with GPS locations) - presence of data connectivity (Wi-Fi/GSM) - temperature - humidity (for above-water deployments) The environmental awareness is enabled by the following sensors: - GPS - surface and underwater cameras - depth - light - laser, sonar (subsea), and humidity (for above-water deployments) are optional sensors - temperature

Functionality Oriented	The Camera system adapts to changes in the environment conditions: it adapts the image quality to the visibility conditions based on the measurements of light. The fusion model for image enhancement

	adaptively changes the weights associated with the influence of the edge-detected image and the original one. The poorer the visibility conditions, the higher the level of image enhancement. The fusion model for object detection adapts the color information based on object's motion. The image quality can be assessed by visual inspection and/or automatically. There are several methods for automatic measurement of image quality. The functional requirements that can trigger adaptation in the OM use-case are: - changing visibility levels - changing illumination levels - user requesting image of certain quality - depth in relation to color compensation - changes of temperature and humidity changing user preferences in the autopilot - collision or obstacle avoidance
Non-Functional Requirements Oriented	The non-functional requirements that can trigger adaptation in the OM use-case are: - power loss that can result in emergency signaling for example - low battery level resulting in triggering the re-charge or change of image resolution for example - system temperature exceeding acceptable values
Repair Oriented	The OM use-case is going to: 1. Obtain fault tolerant power supply to the system components from multiple batteries. 2. Use fault tolerant information storage

CERBERO skeleton implementation in the Use Case: description of the runtime elements.

Element	Involved Components	Interconnection Strategy
Physical Element	Physical prototype of the marine robot including the hull, relevant motor components, battery module, and sensors and cameras will constitute the M36 demonstrator.	All candidate hardware platforms support direct connection for camera, GPS, battery, and other sensors, and enable Java for runtime use. Java interfaces are defined between DynAA and the adaptation components, using an API pattern based on the RCS-4 reference model.
Used KPIs	OM use-case uses the following KPIs: - throughput - energy - power - response time - cost - ranked feature (image quality)	Information for the KPIs is derived from sensors in the marine robot (such as camera sensors) and information from the environment (such as location of other vessels). For the M18 demonstrator we focused on the sensor information received from the camera sensors. For M36 demonstrator, relevant

		marine robot aspects will be shown along with further vision enhancements.
Manager	DynAA is used to model the CPS, with custom logic used to make decisions based on available KPIs. DynAA provides support to the decision components by calculating predictions on possible future impacts on storage, energy, processing load, and image quality. Back up periodic assessments of individual model aspects are also possible. AOW use and investigation is currently aimed for M36.	The components will be connected through abstract interfaces encapsulating toolchain interfaces.
Target	The target will consist of Java components conforming to a reference model API derived from the RCS-4 hierarchical framework. Wrappers for components like DynAA (to model power usage, lighting, and video processing demand) will connect them into the overall framework. C/C++ is used in sub-systems for low-level sensor integration and acceleration purposes.	The components will be connected through abstract interfaces encapsulating toolchain interfaces.

5.1.1. CERBERO Design Time support

DynAA has been used to assess the hardware and software design configurations, to ensure that a given design is capable of handling the video processing data throughput KPIs, and to evaluate the battery demands.

For the development of tool extensions and use-case-specific functionalities the Java programming language is used.

For longer term demonstration activities, if further degrees of freedom are added or if the demonstrator setup scales in terms of number of adopted cameras, we have already discussed the adoption of AOW to cope with larger design time decisions.

5.2. Testing Environment

Testing facilities/environment to be used by the Ocean Monitoring use case:

- **Software simulation (AS)** – before hardware design is finalized, a pure Java simulation environment will connect the primary software components to simulated but realistic video and other sensor data sources. This helps measure the required computation load of the algorithms used in the final use case design and ensure that a viable and cost-effective final design can be developed. In essence it provides mocks and stubs for the sensors and data sources and adds instrumentation to the main functional blocks so that the power and computation resource demands can be measured effectively. This simulation is transitioning from “bare bones” on-shore prototype testing to meeting operation under waterproofing requirements.

- **Subsea light simulator (AS)** - this facility is located at the premises of AS in Aberdeen. In this test environment we can digitally and very accurately simulate and control the representative light conditions for subsea technologies at different depths. This is because the subsea light conditions vary widely both in terms of available color spectrum, and light intensity, depending on the ocean depth. This test environment helps to both research: develop and test the camera sensors and information fusion methods that together help enhance the situational awareness through adaptive sensing. It simulates representative light conditions from surface to deepsea. In essence, it is a subsea dark room for enhancing visibility aspects of subsea imagery. It can also be used for surface purposes to test and simulate differences between day and night vision too. The subsea light simulator has been used in the M18 demonstration. Relevant aspects will be used for M36 demonstration.
- **Oceanlab** - this research lab located in Newburgh, Aberdeenshire, is available for industry to test surface and subsea technologies. The use of this facility for further testing, as necessary, will be considered for M36. The following relevant test environments are available for controlled testing and simulation of ocean environments:
    1. Environmental chamber - allows for testing towards operational requirements by simulating temperature and humidity. It allows temperature ranges from -40°C to +180°C, with varying air humidity conditions.
    2. Indoor immersion tank - allows new technologies to be immersed into a 5x5x5 meter freshwater tank, suitable to test subsea inspection equipment, smaller robots, and so forth. A saltwater tank is also available with acoustically neutral walls, intended for acoustic testing, such as for sonars, and can also be used to expose components to high corrosion.
    3. Hydrostatic pressure tank - The maximum pressure available is 700Bar; deep ocean temperatures can also be simulated, if required.
- **Real ocean testing (AS)** - the depth and conditions of the ocean environment naturally varies. By testing the components directly in real ocean environments, aspects of both surface and underwater technologies that need improvements can quickly be identified. Seeing as the physical/mechanical parts are mobile, and easily transportable, the actual test places can vary from time to time. The real ocean tests of the camera system have already started. The camera prototypes have undergone underwater ruggedization testing in the ocean in the Arctic region (FiguresFigure 5.1,Figure 5.2 andFigure 5.3), according to requirements. Testing within this environment will continue for M36.

Figure 5.1: the Arctic region where the cameras have been tested

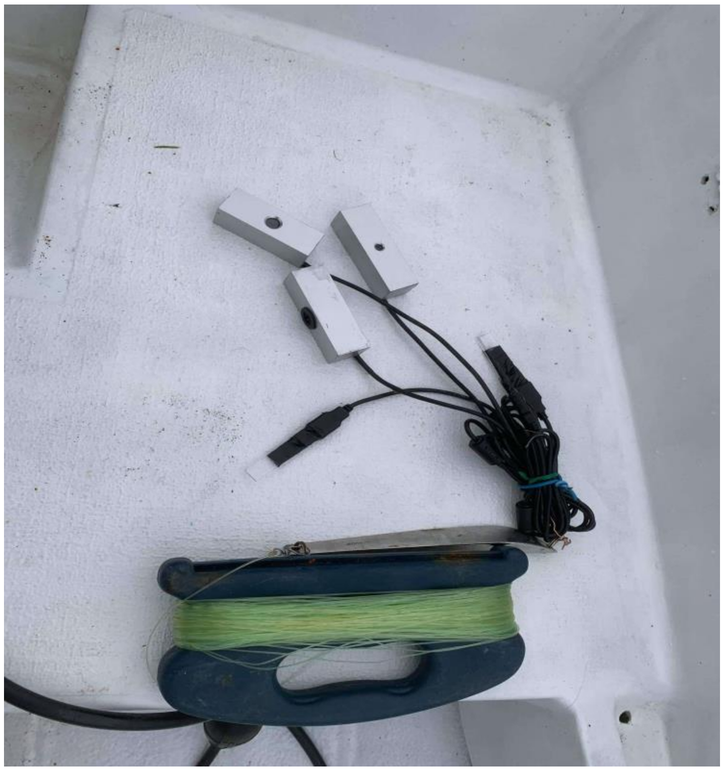

Figure 5.2: the cameras are submerged to different depths with fishing lines in still waters.

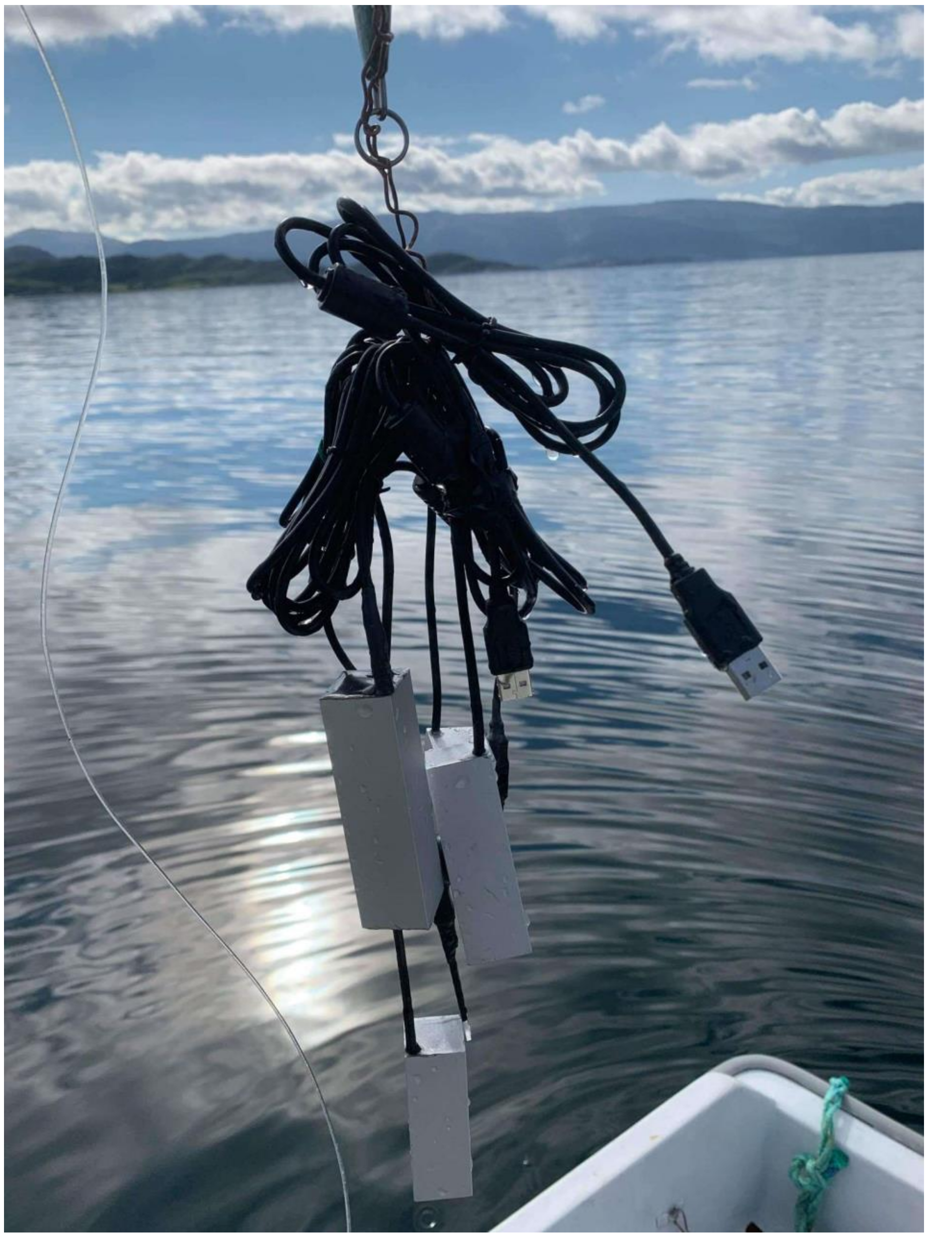

Figure 5.3: the cameras are lowered into deep Arctic waters

The following demonstrators address the challenges and goals of enhanced vision and sensing capabilities. They will be tested within the aforementioned test environments:

- The adaptive fusion model for the automatic enhancement of underwater images. The assessed visibility conditions will be the illumination level and the blurriness of an image which can be measured automatically. The fusion model will update its weights depending on those measurements.
- The adaptive model for the automatic compensation of the loss of light at different depths starting with the red color. Currently, for example, underwater

photographers need to use and physically change different camera filters at different depths. The subsea light simulator is being used for this purpose.

- Fusion of color and motion-based approaches for object detection. The color information is automatically adapted based on object's motion.
- Background subtraction based on optical flow for the moving camera. It requires compensation for the camera movement in order to detect objects based on their motion.

5.3. Prototype Validation

To validate the prototype, a test scenario will be defined, containing the early functionality elements from the Ocean Monitoring scenarios defined in D2.4. During this validation, functionalities of the component interfaces in the skeleton will be assessed by executing the relevant parts of the test scenario.

The primary validations of the prototypes will be:

- Physical prototype including subsea / on-sea movement and vision
- Camera system test in a variety of light conditions e.g. dark areas, day light, night, deep waters that fits with requirements.
    - Test of enhanced vision and object detection capabilities.
- Ruggedization tests including underwater real ocean trials of the camera and marine robot prototypes.

6. Conclusion

In this document the demonstration skeleton for prototyping activities has been described in detail. Starting from the general skeleton of the CERBERO approach, each use case has been further developed identifying tools, technologies, interface and components that are planned to be used in the demonstration activities.

The skeleton and the customization for each use case has allowed to:

- Identify a homogenous approach for all the 3 use cases, giving them guidelines and recommendations for planning the prototypes.
- Identify the CERBERO tools and their interfaces that are requested to be developed the demonstrator.
- Analyze the components and their compatibilities for the demo.

7. References

[CERBERO 2018]	http://www.cerbero-h2020.eu