**Information and Communication Technologies (ICT) Programme**

**Project Nº: H2020-ICT-2016-1-732105**


# *D6.7: Demonstration Skeletons (Ver. I)*

| | |
|---|---|
| **Lead Beneficiary:** | 8 - AI |
| **Work Package:** | 6 |
| **Date:** | 16-Feb-18 |
| **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 |
|---|---|
| Gasser Ayad | AI |
| Francesca Palumbo | UNISS |
| Antonella Toffetti | CRF |
| Joost Adriaanse | TNO |
| Antonio Solinas | AI |
| Maria Katiuscia Zedda | AI |
| Leszek Kaliciak | AS |
| Stuart Watt | AS |
| Ayse Goker | AS |
| Hans Myrhaug | AS |
| Sergio Tardón | TASE |



# Document Revision History

| Date | Ver. | Contributor (Beneficiary) | Summary of main changes |
|---|---|---|---|
| 8-Feb-18 | 0.1 | AI | 1st Draft |
| 16-Feb-18 | 0.2 | AI | 2nd Draft |
| 26-April-18 | | CRF, TNO | Update Smart Travelling use case |
| 04-May-18 | 0.3 | UNISS, AI | 3nd Draft |
| 09-May-18 | 0.4 | AI, TNO | Minor revision to the structure and update Smart Travelling Use case |
| 17-May-18 | 0.4 | AS | Add Ocean Monitoring information to new table templates |
| 28-May-18 | 0.5 | AS | Updates in Ocean Monitoring use case and prototyping |

| | | | |
|---|---|---|---|
| 31-May-18 | 0.6 | TNO | Updated Smart Travelling use case chapter (and corrected some typos) |
| 31-May-18 | 0.7 | TASE | Update Planetary Exploration use case |
| 14-June-18 | 1.0 | AI | Complete revision |
| 30-June-18 | 1.0 | IBM , UNISS | Review and consistency check. |

Table of Contents

# 1. Executive Summary

This document is a partial release of the demonstrator skeletons for prototyping activities in CERBERO use-cases. The document is a preliminary version of D6.1 due by M32 where the consortium presents full-fledged demonstrators of CERBERO use-case scenarios.

## *1.1. Structure of Document*

This document presents the general skeleton for demonstrating the first iteration of the CERBERO use-case scenarios, the customization of such skeleton for each use case, the testing environment and the planned prototypes. At the moment, demonstrators are under integration and preliminary assessment is scheduled for M18.

Use case challenges, goals, and system architectures have been described in 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 M18 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.

The final set-up and assessment purposes of each demonstrator, the toolchain feedback, and the next steps would be further elaborated on by respective use-case demonstrator deliverables, namely the D6.8 for planetary exploration, D6.9 for ocean monitoring, and D6.10 for smart traveling at M18.

## *1.2. Related Documents*

- D2.4 – Description of Scenarios (Ver. II)
  Customization of each skeleton follows the use case scenario description defined in D2.4
- D5.6 – Framework Components (Ver. I)
  Adopted components of the CERBERO framework are described in D5.6
- D6.8 - Space Demonstrator (Ver. 1), D6.9 - Ocean Monitoring Demonstrator (Ver. 1), D6.10 - Smart Travelling Demonstrator (Ver. 1)
  Skeletons will be used to guide implementation in demonstration activities to be described in D6.8-10.

# 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 components of CERBERO self-adaptive systems (see Figure 2-1);
2. what techniques and tools 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 is presented in D5.7.

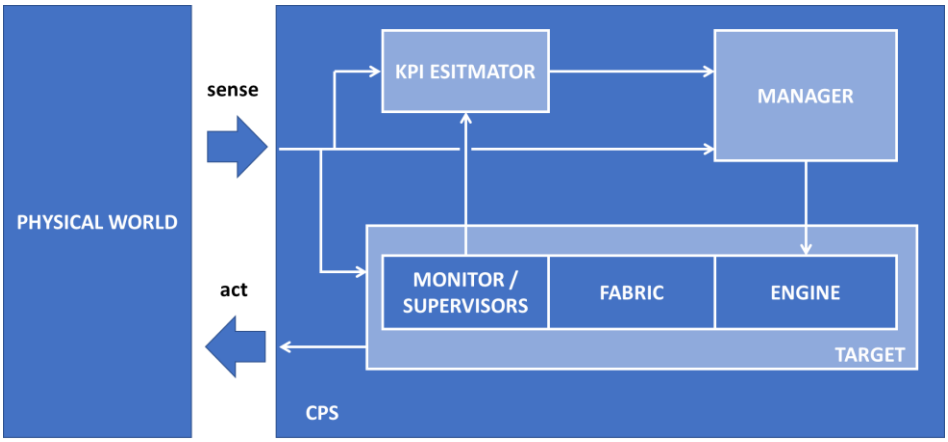


**Figure 2-1: CERBERO skeleton**

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 D3.4 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 which 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 may be implemented/extended in WP4 if not already available as off-the-shelf components.

The strategy to interconnect the three above-mentioned elements and their sub-parts can follow two different approaches: toolchain-based and intermediate format based, which have been defined as outcomes of WP3 and WP5.

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, 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:<br>- GPS coordinates of the car<br>- Speed of the car<br>- Current level of the battery<br>- 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 a special user interface. The users should also select preferred route if multiple routes 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.<br>Possible routes and charging facilities will be preloaded in the system and optionally updated via an Internet connection to a central system. |

| | |
|---|---|
| Functionality Oriented | The functional parameters that can trigger adaptation in the Smart Travelling use case are:<br>- Battery load (low detected battery level can for example trigger to charge earlier than planned)<br>- GPS coordinates (can trigger re-planning of route in case driver does not follow advised route)<br>- User request to plan route to destination<br>- 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)<br>- Update on road blockage or traffic jams (could trigger re-planning in case road segment was on planned route)<br>See also adaptation strategies in D2.4 section 2.3. |
| 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. |

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

| Element | Involved Components | Interconnection Strategy |
|---|---|---|
| Physical Element<br><br>Target | In this scenario the target and the physical element are basically coincident. In fact, the physical environment is simulated by Car simulator, which represent the target we are using. Currently the Target is composed of the SCANeR car simulator (containing most of the sensors), DynAA (to directly communicate with the car via provided APIs) and the MECA User Interface (which is used to interact with the driver and reactively or proactively provide suggested routes to the driver). The Car 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. In a later phase the interfaces will be migrated to CERBERO toolchain interfaces (using the intermediate format). During the development of the M18 demonstrator, an initial (stand-alone) version of CERBERO intermediate format software will be developed by TNO. |
| Used KPIs | The supported KPIs are (see D2.4 for generic KPIs):<br>- Response time to trigger and Latency (used in system-in -the-loop simulation mode)<br>- Energy (load of battery and usage of energy by car)<br>- Cost (of electricity at charging station)<br>- Travel time<br>- Possible other user preferences (not yet included in M18 demonstrator)<br><br>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) and information from the environment (charging infrastructure info, road congestion info). For the M18 demonstrator we will focus on sensor information received or retrieved from the car.<br><br>At M18 direct interactions among tools are expected, while at M36 the CERBERO intermediate format will be adopted. |
| 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 possible routes to take. | The interfaces between tools will be direct (HTTP REST). In a later phase the interfaces will be migrated to CERBERO toolchain interfaces (using the intermediate format). |

### *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 implementing complete CPS with minimal effort. Using DynAA and MECA tools directly connected to SCANeR avoids 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 will be the CRF 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 will be set up at TNO (in The Hague) where DynAA and MECA will be tested. For this purpose, Oktal will provide the project with a test license of SCANeR. This second test site will be used to develop and validate the message interface used by SCANeR. Once components are validated, they will be shipped and integrated in the test site in Turin. Meanwhile, Smart Travelling Use Case partners will work together remotely to eliminate possible incompatibilities and difficulties during the integration phase, defining specifications carefully.

In the scenario description document (D2.4), three different individual use cases were described. As the demonstrator of M18 does not include all required integration of the different components, only specific parts or functionalities of these scenarios can be tested.

The tests that will be performed for the first demonstrator (M18) will 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 MECA and DynAA based on those data. These actions simulate the actions needed for route optimization. The actual calculation of best route, using predictions (run in DynAA), will not be part of the first demonstrator (M18) but will be provided in the second demonstrator (M36).
4. Synchronization and fusion of generated logging data of all simulation modules 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.

It is expected that the route calculation and charging optimization (using AOW) will not be integrated and completed before M18. The executed test scenario will therefore not include rerouting or re-planning (as indicated in D2.4). These adaptation functionalities will be part of the second demonstrator (of M36).

## *3.3. Prototype Validation*

For validation of the prototype a test scenario will be defined, which will contain the most important elements from the Smart Travelling scenarios as defined in D2.4 (Scenario Description). 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 form a test run performed on the CRF simulator will be merged into an overall logfile. This logfile will be compared with the (partly)

manually generated merged logfile produced by CRF for the given test run. The validation should prove that the automatically fused logfile provides correct results with similar or even higher quality than the file produced manually.

# 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 a robotic arm is simulated how it adapts to harsh environmental conditions. The system will be able to ensure proper operation, adaptation and failure detection.<br>Monitors used are:<br>- Performance monitors (throughput).<br>- Fault monitors (HW and SW). |
| User-Commanded | The user can interact with the system by sending the final position of the robotic arm. If no problems arise, the motion planning will be performed on the robotic arm and will also be shown on the screen. |
| Environmental Awareness | The system is aware of its position by solving forward kinematics equations and is also aware of known obstacles in the planning path. In future deliverables (D6.1 and D6.2) the system will be aware of unknown obstacles by sensor feedback. |

| | |
|---|---|
| Functionality Oriented | The functional parameters that can trigger adaptation are:<br>- Obstacles: based on the shortest path from the starting point to the goal point, the Robot Control Unit (RCU) will be able to recalculate motion planning points in order to avoid obstacles in the robotic arm path.<br>- User: requests may change the robotic arm position to a different goal position. |
| Non-Functional Requirements Oriented | The non-functional parameters that can trigger adaptation are:<br>- Execution time: Parallels methods of computation and HW accelerators will be developed in order to improve execution time. |
| Repair Oriented | In the Planetary Exploration use case, fault tolerance support is very important due to radiation and harsh environmental conditions. Hardware and software monitors are meant to trigger reconfiguration when necessary. At component level the ARTICo$^3$, used to accelerate specific computational tasks, provides on-demand hardware redundancy and hardware monitors to check error status. |

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

| Element | Involved Components | Interconnection Strategy |
|---|---|---|
| Physical Element | WidowX robotic arm with 6 Dynamixel actuators. | The Physical Element provides information to the CERBERO tools by sensor interfaces, which will be developed for M36 demonstrator. |
| Used KPIs | The supported KPIs are (see D2.4 for generic KPIs):<br>- Latency | Information for KPIs is derived from cyber and physical monitors. |

| | | |
|---|---|---|
| | - Throughput<br>- Resources utilization<br>- Availability | In D6.8 the following monitors will be provided:<br>- Latency and throughput: time to perform kinematics algorithms will be measured and displayed by PAPIFY.<br>Moreover, studies towards execution time minimization and improved computational robustness will take into account:<br>- Resources utilization: ARTICo3 monitors the number of slots used. In the long term, resource utilization should be sustainable wrt energy consumption.<br>- Availability: number of errors. |
| Manager | CERBERO tools are used as manager of the CPS. In particular, at M36, we will exploit SPIDER (connected to PAPIFY) to take decisions. | The interface with the robotic arm will be a serial communication. |
| Target | At M18, the target will be composed of the robotic arm, which will physically execute the RCU. Execution is planned to be performed over the reconfigurable logic. In particular, ARTICO + PAPIFY in the short term will constitute fabric, engine and monitors. | The components will be connected as follows:<br>- ARTICO + PAPIFY / PAPIFYVIEWER<br>There will be also another demonstrators as proof of concept between CERBERO tools:<br>- ARTICO + MDC<br>- PREESM + PAPIFY + SPIDER<br>At M36 these PoC will be used as the basis of the final demonstrator. |

### *4.1.1. CERBERO Design Time support*

In the Planetary Exploration use case, the following CERBERO tools will be used at Design Time support:

- 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 ARTICo³). 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 analysed chronologically from either an actor or a processing element point of view.
- The ARTICo$^3$ toolchain is a set of automation scripts to build 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).

Details on these tools are provided in D5.6.

## *4.2. Testing Environment*

The testing environment for the robotic arm is composed by the system architecture and components already presented in D2.4, 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.
- Hardware accelerators and time monitoring.

### *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.

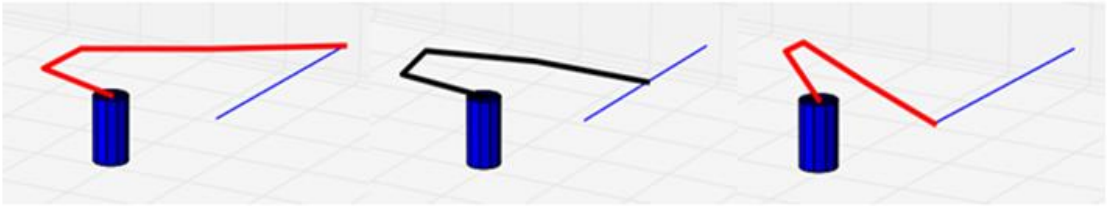

**Figure 4-1: Python Simulation of Linear Motion Planning in Free Space**

### *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.

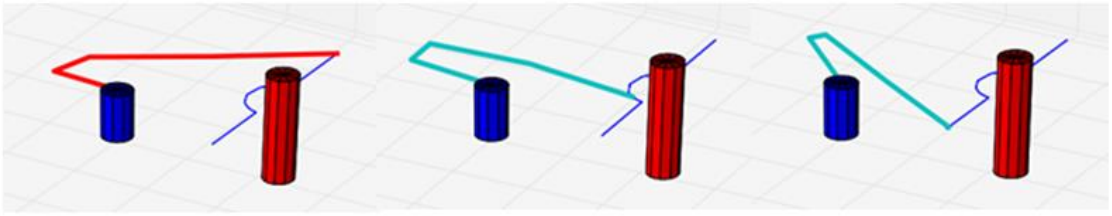

**Figure 4-2: Python simulation of Collision-Free Motion Planning**

### *4.2.3. Hardware accelerators and time monitoring*

The goal of this test environment is to perform the inverse kinematics and inverse optimization algorithms in parallel by HW accelerators. ARTICo³ will migrate algorithms from software to hardware in order to achieve the most efficient way to solve these algorithms in parallel. This will improve the execution time and thus the latency and throughput of the system (see used KPIs).

PAPIFY and PAPIFY VIEWER will be monitoring the system´s KPIs (e.g. execution time). Figure 5-3 shows the PAPIFY VIEWER interface divided in four cores running Sobel edge detector with 12 actors, which shown parallelism of the system and execution time of each actor.

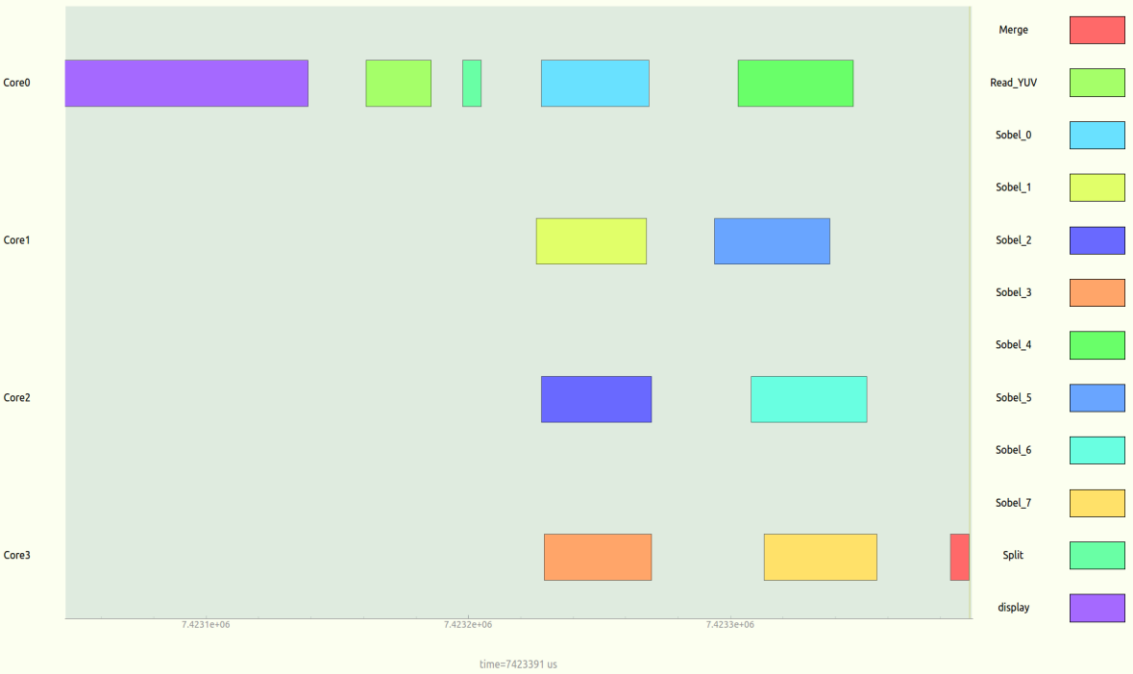


**Figure 4-3 PAPIFY VIWER interface**

## *4.3. Prototype Validation*

The Planetary Exploration prototype will be validated by executing the Test Scenario presented in Section 4.2. During this test scenario, the following functionalities will be validated:

- Motion planning in free space to validate inverse kinematics algorithms.
- Collision-free motion planning with known obstacles to validate adaptation of the system.
- Time monitoring for different parallelism of the algorithms.
- Functional fault injection (multiplexor-based faulty/correct behavior selection) to prove that the system is fault tolerant due to Triple Module Redundancy at component level (ARTICo$^3$).

# 5. Ocean Monitoring

## *5.1. Customization of the Skeleton*

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

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

| | |
|---|---|
| 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 image quality can be assessed by visual inspection and/or |

| | |
|---|---|
| | automatically. The functional requirements that can trigger adaptation in the OM use-case are:<br>- changing visibility levels<br>- user requesting image of certain quality<br>- depth in relation to color compensation<br>- user changes battery configuration to achieve certain voltage<br>- GPS location information can trigger vehicle's re-routing<br>- changing user preferences in the autopilot<br>- collision or obstacle avoidance |
| Non-Functional Requirements Oriented | The non-functional requirements that can trigger adaptation in the OM use-case are:<br>- power loss that can result in emergency signaling for example<br>- low battery level resulting in triggering the re-charge or change of image resolution for example |
| Repair Oriented | The OM use-case is going to:<br>1. Obtain fault tolerant power supply to the system components from multiple batteries.<br>2. Use fault tolerant information storage (M36) |

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, engine, battery module, and sensors like GPS and cameras will constitute the M36 demonstrator. The M18 demonstrator consists of the physical prototype of the camera system. | All candidate hardware platforms support direct connection for camera, GPS, battery, and other sensors, and enable Java for runtime use.<br>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:<br>- throughput<br>- energy<br>- power<br>- response time<br>- cost<br>- ranked feature (image quality) | Information for the KPIs is derived from sensors in the marine robot (like camera sensors) and information from the environment (like location of other vessels). For the M18 demonstrator we will focus on the sensor information received from the camera sensors. |
| 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 Java abstract interfaces encapsulating toolchain interfaces. In a later phase the interfaces may be migrated to CERBERO 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. | The components will be connected through Java abstract interfaces encapsulating toolchain interfaces. In a later phase the interfaces may be migrated to CERBERO 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 languages is used.

For longer term demonstration activities, if further degrees of freedom are added or if the demonstrator set up 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 energy and computation resource demands can be measured effectively. It also provides an effective early test and development environment for the use case before the physical platform has been completed.
- **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 colour 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 deep sea. 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 will be used in the M18 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 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. Vibration table - an industry testing standard for automotive, aerospace, electronics, and subsea vibration testing. Components can be tested/ exposed to mechanical shocks and vibrations over time, i.e. test for robustness.
4. Hydrostatic pressure tank - The maximum pressure available is 700Bar; deep ocean temperatures can also be simulated, if required.

- **Real ocean testing (AS)** - the marine robot parts are tested/exposed on the sea, which means the equipment is launched either directly from shore, near shore, in sheltered and/or open seas. 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 will vary from time to time. Testing within this environment will be for M36.

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

- The adaptive model for the automatic compensation of the loss of light at different depths starting with the red color. Currently the underwater photographers need to use and physically change different camera filters at different depths. The subsea light simulator will be used for this purpose.
- 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.
- Depending on the status and timing of the other scheduled tests, it might be possible to work also on the fusion model for the combination of frame difference and color based movement detection and tracking methods. The model should make use of the strengths of both methods towards a demo.

The first mid-way demonstrator will be in M18.

## *5.3. Prototype Validation*

To validation 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 at M18 will be:

- Enhanced vision and sensing capabilities: evaluating adaptive vision with simulated data in a software-only environment.
- Interoperability of CPS components: evaluating configuration predictions against datasheet specifications for alternative architectures.

# 6. Conclusion

In this document the demonstration skeleton for prototyping activities has been described in details. 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