Information and Communication Technologies (ICT) Programme

Project Nº: H2020-ICT-2016-1-732105

D6.8: Planetary Exploration Demonstrator (Ver. I)

Lead Beneficiary: TASE
Work Package: 6
Date: 14-June-18
Distribution - Confidentiality: [Public]

Abstract:
This document contains the description of the M18 Planetary Exploration demonstrator. The description is based on the skeleton defined in D6.7 and includes the scope and purpose of the demonstrator, the description of the demonstrator itself and the accomplished results till now.

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 Onderzoek 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**
Sergio Tardón	TASE
Pablo Sánchez	TASE
Manuel Sánchez	TASE
Eduardo Juarez	UPM
Daniel Madroñal	UPM
Francesca Palumbo	UNISS

Document Revision History

Date	**Ver.**	**Contributor (Beneficiary)**	**Summary of main changes**
14-June-18	0.1	TASE	1st Draft
20-June-18	0.2	TASE	Adding sections 2, 3 and 4.
26-June-18	0.3	UPM	Adding sections 3.1.2 and 3.1.3.
27-June-18	0.4	UNISS	1st Review
27-June-18	0.5	TASE	New release
29-June-18	0.5	IBM, AS	Final Review

Table of contents

1. Executive Summary

This document describes the M18 Planetary Exploration demonstrator, and it is a preliminary version of D6.2, due by M36, which will describe the final Planetary Exploration demonstrator as one of the CERBERO use-case scenarios.

At the moment, the general infrastructure of the demonstrator and the CERBERO technologies that will be assessed through it have been completely identified. The skeleton and purposes of the demonstration have been already discussed in D2.4 and D6.7. More details are provided in this document with a complete overview of the current status and the expectations for the second reporting period.

1.1. Structure of Document

Section 2 describes the scope and purpose of the demonstrator. Section 3 provides details the developed demonstrator. In Section 4 the tests, results and feedback are presented. Section 5 concludes our insights so far and Section 6 provides the references.

1.2. Related Documents

The CERBERO deliverables related to this document are:

- D2.4 – Description of Scenarios (Ver. II)
  The planetary exploration demonstrator is based on the use case scenario description defined in D2.4
- D2.7 – Technical Requirements (Ver. II)
  Assessment of the demonstrator will contribute to the validation of the requirements listed in D2.7
- D3.4 – Modeling of KPI (Ver. I)
  The addressed KPIs are based on the generic list of KPIs as defined in D3.4
- D4.3 – CERBERO Multi-Layer Runtime Adaptation Strategies (ver. 1)
  The hardware adaptation strategies, in particular ARTICo3 and MDC-compliant reconfiguration, are described in D4.3
- D4.4 – Self-Adaptation Manager
  Runtime adaptation strategies exploited in the planetary exploration demonstrator are described in D4.4
- D5.6 – Framework Components (Ver. I)
  The adopted components of the CERBERO framework are described in D5.6
- D6.7 – Demonstration Skeleton (Ver. 1)
  The generic skeleton used to build the planetary exploration demonstrator is described in D6.7

2. Scope and purpose

In this document the M18 demonstrator of the Planetary Exploration (PE) use case is described. The demonstrator is developed to demonstrate and validate the CERBERO runtime support and the tool-chain for the deployment of cross-layered and adaptive CPS. A commercial robotic arm, WidowX, with a similar structure to the intended scenario is used in the demonstrator to validate CERBERO tools deployment and also KPI measurements and adaptation.

As the WidowX robotic arm does not perform harsh environment adaptation and reconfiguration due to radiation. CERBERO technologies (in particular the autonomous self-adaptation manager and the sel-adaptation loop described in D4.4 and D4.3) and tool-chain are going to be used to offer that kind of support providing advanced adaptation capabilities for CPS. In the long term (M36), autonomous reconfiguration for self-healing purposes is going to be demonstrated. Moreover, as WidowX robotic arm does not include space qualified motors, Brushless DC (BLDC) motor control will be tested at M36.

The M18 demonstrator is a preliminary version of the complete one, where the development was focused on:

1. validating algorithms and motion planning;
2. exploring the adaptation support to be provided in this scenario and mapping it to the available CERBERO technologies, to understand if current envisioned self-adaptation infrastructure would suffice in the long term or if additional strategies have to be developed in the second part of the project;
3. measuring the robotic arm KPIs by CERBERO technologies (the KPIs will be optimized at runtime in M36).

For the final demonstrator the functionality will be further extended using additional tools from the CERBERO toolchain, sensor fusion from proximity sensors and path planning with unknown obstacles using, e.g., potential field algorithms, to implement better adaptation to the environment.

The drivers of demonstration activities have been defined in D2.7. In Table 2-1 we provide an excerpt of Table 4 of D2.7. In Table 4 of D2.7 we mapped user requirements with technical requirements, while we are discussing here how (Validation within the PE demonstrator) and when (Planned Month) the validation is going to take place. As you can see, and discussed above, the coverage of the various needs is distributed between M18 and M36. The main reason is that some technologies required stand-alone integration before being applied to a complete complex use case such as the PE one.

Table 2-1. User to Technical requirements mapping with related assessment strategy.

User Requirements	Technical Requirements	Validation within the PE demonstrator	Planned Month

PE1. Development of a Hardware / Software (HW/SW) co-design for Rad-Tolerant control of robotic arm for planetary exploration using adaptable COTS FPGAs.	5, 6, 7 and 8 (see Section 4.5 and Table 4 of D2.7)	Integrate CERBERO tools and technologies in the Robot Control Unit in order to provide HW/SW co-design.	M18 integration of HW accelerators. MDC and ARTICo^3 have been successfully integrated (see D5.7). ARTICo^3 has been already used in the context of this use case to achieve a fault tolerant behavior. Migration from SW to HW to accelerate path planning has been performed. The exploration of the possible runtime trade-offs is still on-going, and will be completed at M36. Complete HW-SW self-reconfiguration is expected for M36.
PE2. Develop integrated open toolchain environment for development of robotic arms for space missions with focus on multi-viewpoint system-in-the-loop virtual environment.	1, 2 and 4 (see Section 4.5 and Table 4 in 4.6 in D2.7)	Use CERBERO tools/technologies for development of robotic arm applications both at design time and at run time.	At M36 we intend to assess in the PE demonstrator the self-adaptation loop, including SPIDER, PAPIFY, MDC and ARTICo^3. Moreover, design time tools will be adopted as well to facilitate the adaptive fabric deployment.
PE3. Development of a (self-)adaptation methodology with supporting tools.	6 and 20 (see Section 4.5 and Table 4 in 4.6 in D2.7)	Integrate CERBERO tools for adaptation methodologies (e.g. energy efficiency, reconfiguration, etc).	At M18 we achieved integration of the different adaptation strategies (we already combined dynamic partial reconfiguration and virtual reconfiguration) among each other and exploration of the adaptation potentials of the free path planning. At M36, we expect to finalize KPIs runtime assessment and optimization, while providing autonomous self-reconfiguration.

3. Description of the Planetary Exploration demonstrator

The PE use case aims at validating motion planning algorithms and assessing adaptation to harsh environments (see D2.4). Different CERBERO tools and technologies will be used to address these challenges. In the final demonstrator, they will be integrated in the Robot Control Unit (RCU) and in the Servo Control Unit (SCU) to provide KPI measurements, optimization and self-reconfiguration.

A high level view of the Planetary Exploration use case is presented in Figure 3-1. After iterations and discussions between partners in developing the demonstrator, this figure has been modified, with respect to D2.4, to better represent the Planetary Exploration use case, and better fit the mapping with the skeleton. In the M18 demonstrator the focus is on the RCU, the Adaptive Motion Planning, the Monitoring Infrastructure and the Reconfiguration. Please note that the already implemented/under assessment elements are identified in the figure below with a green box. The whole system with its functionalities including SCU, Self-Healing, Reinforcement Learning and Encryption will be presented at M36.

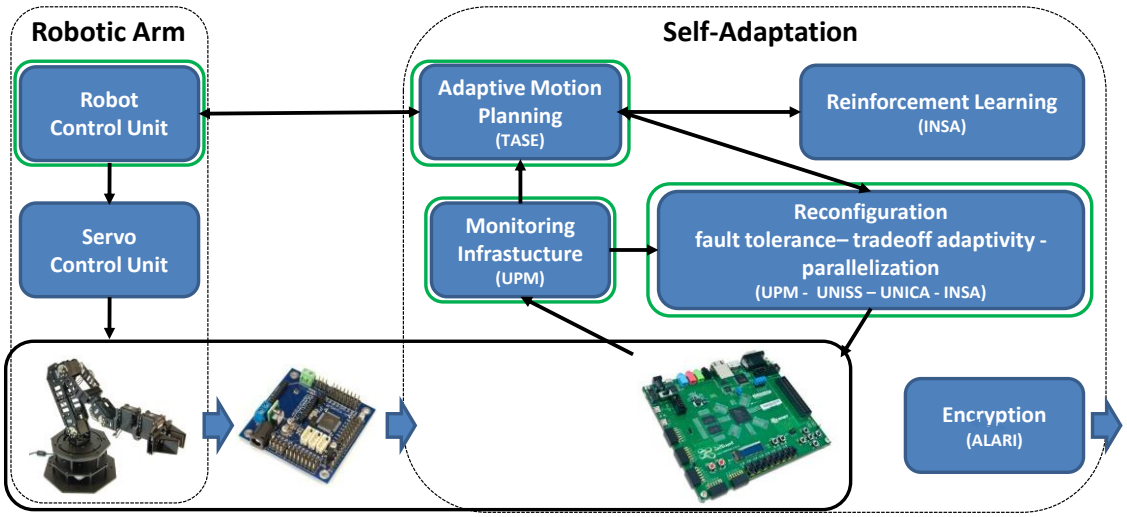

Figure 3-1. Schematic overview of the M18 PE components.

As reported in Table 2-1, the high-level CERBERO requirements related to the PE use case are:

1. (PE1) Enable **Hardware / Software (HW/SW) co-design** for Rad-Tolerant control of robotic arm for planetary exploration using adaptable COTS FPGAs.
2. (PE2) Develop **integrated "open" toolchain environment** for development of robotic arms for space missions with focus on multi-viewpoint system-in-the-loop virtual environment.
3. (PE3) Development of a **(self-)adaptation methodology** with supporting tools.

Derived from these requirements, we have defined the following long term goals:

1. **Fault tolerance** to single event effecting errors. In M18 demonstration ARTICo³ has already proved this capability.
2. **Environment adaptation by reinforcement learning** to the harsh physical environment. This goal is expected to be addressed at M36.
3. **Power measurement and optimization** performed by SW and HW monitors on the cyber part of the considered CPS. For M18 we have instrumented the code with Performance Monitoring Counters, but the next short term goal is to achieve simultaneous HW/SW monitoring.

PE1 and PE3 requirements will be partly covered at M18 demonstrator. **(PE1)** HW/SW co-design will be implemented by means of mixed HW/SW implementations, combining different CERBERO tools (i.e., ARTICo³ and MDC). Parallelization and fault tolerance capabilities are going to be demonstrated at M18. In parallel, we have started to study the possible runtime trade-off that characterize the planetary exploration use case. The idea is to present, at the review, a study on the different algorithms exploitable for solution of the inverse kinematics problem, together with their mapping to CERBERO technologies. The concurrent adaptive implementation of different algorithms could contribute to achieve the requested QoS versus computational efficiency and energy trade-off. Such study contributes to both PE1 and PE3.

(PE3) implies the development of a multi-layer adaptation methodology. As a result the first 18 months of CERBERO activities we are going to assess the integration (see D5.7) of different HW and SW adaptation strategies and toolchains which compose CERBERO adaptation loop (see D4.3). Monitoring capabilities are going to be demonstrated at M18. The complete assessment of these strategies/toolchains and the respective adaptation loop will be completed at M36.

The demonstrator we are presenting in this document is derived from the skeleton we generically defined for all the use case, and then customized for the PE use case, in D6.7. In Figure 3-2, a graphical mapping is done between the use case architecture and the CERBERO skeleton.

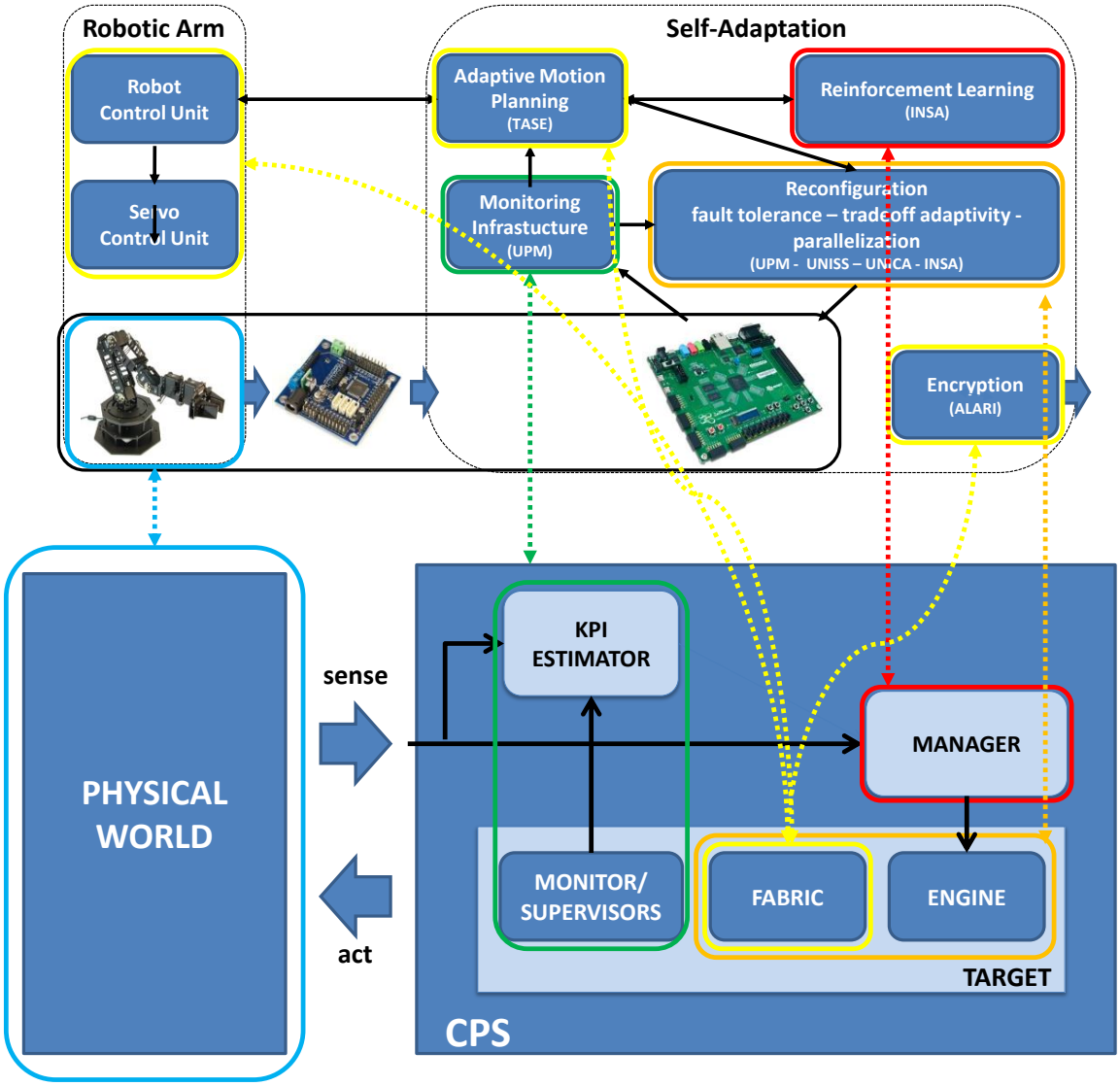

Figure 3-2. Planetary Exploration use case: CERBERO skeleton mapping.

Here we can see as the WidowX robotic arm and the environment compose the physical world. The Robotic Control Unit (RCU) and the Servo Control Unit (SCU), provided by TASE, represent the functionalities implemented by the demonstrator. At M36 both of them are meant to be implemented on the reconfigurable target, while we have used the RCU for the demonstration activities of this first review period. The rest of the mapping among the CERBERO skeleton and the use-case complete overview is as follows:

- The **KPI Estimator and the Monitor / Supervisor** features are provided by the Monitoring Infrastructures (UPM) using PAPIFY / PAPIFY VIEWER. SW monitors are available and already integrated within PREESM and SPIDER tools (see D5.7), while HW extensions are on-going. Motion planning adaptation leverages on these infrastructures to select the optimal trajectory at run-time.

- The computing **Fabric** is based on ARTICo^3 (see D4.3, D4.4 and D5.6), which has been extended to host also MDC-based accelerators (see D4.3, D4.4 and D5.6). Specific stand-alone demonstration of this integration is provided in D5.7. ARTICo^3 and MDC have their own engines to put in place the actions needed for reconfiguration. PAPI-compliant monitors are going to be developed, to include HW-monitoring capabilities in the reconfigurable architecture (see D4.4).
- MDC and ARTICo^3 have their own reconfiguration **Engines**, which are already implemented (see D4.3). In the long term, these engines will be able to execute the decisions of the **Manager** adapting the system at runtime given environmental and system triggers.

Self-adaptation (intended for self-healing and adaptive planning) is going to be demonstrated at M36. The **Manager** is in charge of enabling both physical environment and self-awareness adaptivity through Reinforcement Learning techniques (INSA). The self-adaptation is triggered when needed by SPIDER tool, depending on the measured KPIs and the system goal/needs. This is going to be demonstrated at M36.

3.1. Functionalities

The implementation of the M18 Planetary Exploration demonstrator functionalities (according to goals and requirements set in D2.7) is split into the following three parts:

1. Physical environment adaptation.
2. Mixed hardware/software implementation with hardware accelerators.
3. KPIs monitoring and display.

In the following sub-sections, the developed functionalities of different parts of the demonstrator are described.

3.1.1. Physical environment adaptation

The PE demonstrator must be able to adapt to physical obstacles. Based on an initial and final points, the system will calculate a set of interpolation points and inverse kinematics solutions to provide collision-free path planning of the robotic arm. All relevant steps of the PE scenario are described in D2.4.

There will be two main scenarios to verify physical environment adaptation:

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

3.1.1.1 Motion planning in free space without obstacles

The purpose of this test environment is to validate the software algorithms in order to provide motion planning of the robotic arm.

When a final position is sent to the RCU, the system will interpolate the linear trajectory between the actual point and the desired point with a fixed number of interpolation points. Once the interpolation points are calculated, inverse kinematics algorithms using Nelder-Mead optimization method[1] will provide the angle of each joint actuator in order to reach the final end effector position.

These joint actuator angles will be sent to Python for verification and visualization by a 3D simulation of the motion planning and then, they will be sent to the robotic arm in order to perform a real motion planning. A 3D Python motion planning simulation in free space is shown in Figure 3-3:

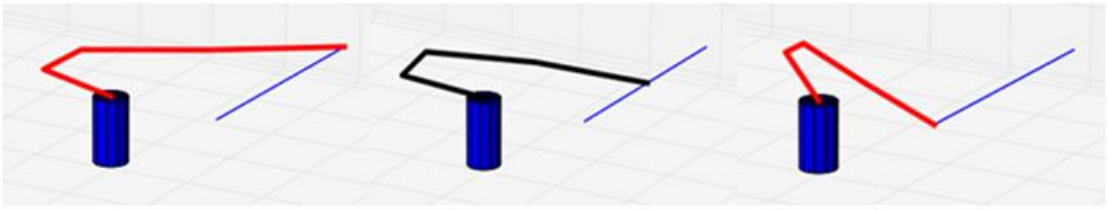

Figure 3-3 - Python simulation of linear motion planning in free space

3.1.1.2 Collision-free motion planning with identified obstacles

The purpose of this test environment is to validate the adaptation software algorithms in order to avoid identified obstacles in the path.

This test environment is complementary to the previous one. Once the interpolation points are calculated, they will have to be compared with the obstacle coordinates and then, recalculated if necessary with a defined distance between the new points and the obstacle.

The inverse kinematics algorithms and Nelder-Mead optimization method will provide each joint actuator angle, which will be sent to Python and also to the robotic arm.

A 3D Python collision-free motion planning simulation is shown in Figure 3-4.

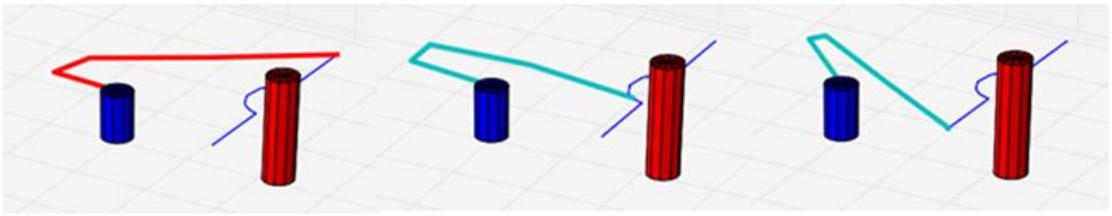

Figure 3-4 - Python simulation of collision-free motion planning

[1] Nelder-Mead algorithm is a numerical algorithm to solve Inverse Kinematics problems. It does not use derivatives; therefore, it is a good candidate for non-linear optimization problems. Moreover, TASE had available an in-house (C and Python) implementation, which is why it has been used as a starting point to implement the PE use case.

3.1.2. Hardware accelerators

In D4.3, D4.4 and D5.7 we have discussed in detail the different options, stand-alone and combined, available for runtime hardware adaptation in the CERBERO project. As we have already said, to maximize the flexibility of the computing fabric while minimizing the costs of reconfiguration, we leverage on Dynamic Partial Reconfiguration (made available by ARTICo^3) and coarse-grain virtual reconfiguration (made available by MDC). Currently, these techniques are fully integrated (see D5.7), and their assessment within the PE case is on-going. The HW acceleration technologies discussed below will be connected to the co-design environment that will assess autonomous HW-SW exchangeability at M36.

3.1.2.1 Parallelization

Parallelization will be demonstrated in M18 by using the ARTICo^3 architecture to implement the identified hardware-friendly components of the robotic arm controller.

After an initial analysis of the inverse kinematics algorithm based on Nelder-Mead optimization, which also involved a thorough profiling of the application, the first approach to manual hardware/software partitioning was proposed. In this proposal, the cost function, which is the most time-consuming operation (called 6 times in every iteration of the Nelder-Mead optimization algorithm), was selected for implementation.

The implementation of the cost function evaluator in hardware has been further evaluated by using a model-based implementation in Simulink (see Figure 3-1). An ARTICo^3-compliant version of the hardware core has been generated, including input and output memory banks as well as a specific Finite State Machine (FSM) FSM controller.

Two different levels of parallelism have been identified to be considered for the demonstrator in M18: evaluation of the cost function during each iteration of the Nelder-Mead optimization algorithm, and inverse kinematics solution for each point in the trajectory. The former can be addressed at design time by increasing the amount of cost-function evaluators inside each ARTICo^3 accelerator, while the latter can be addressed only at run time by dividing the whole trajectory to be described by the robotic arm in different sections, and processing each one in a separate hardware accelerator. For parallelizing the inverse kinematics solution, data independence is assumed to exist between trajectory sections, which will eventually lead to sharp transitions between them during actual robot movement (the initial point for each section will not come from previous iterations of the optimization algorithm).

The demonstrator will allow the runtime evaluation of a tradeoff between processing performance and smoothness in the movement of the arm. The trajectory of the robotic arm will be split in a variable number of sections. Each of those sections will be evaluated in an ARTICo^3 accelerator. The more sections trajectory is divided in, the faster will be the computation of the trajectory but the less smooth the resulting dynamic of the arm will be. It will also allow the analysis of convergence problems derived from the data-independent partitioning scheme required in ARTICo^3 implementations.

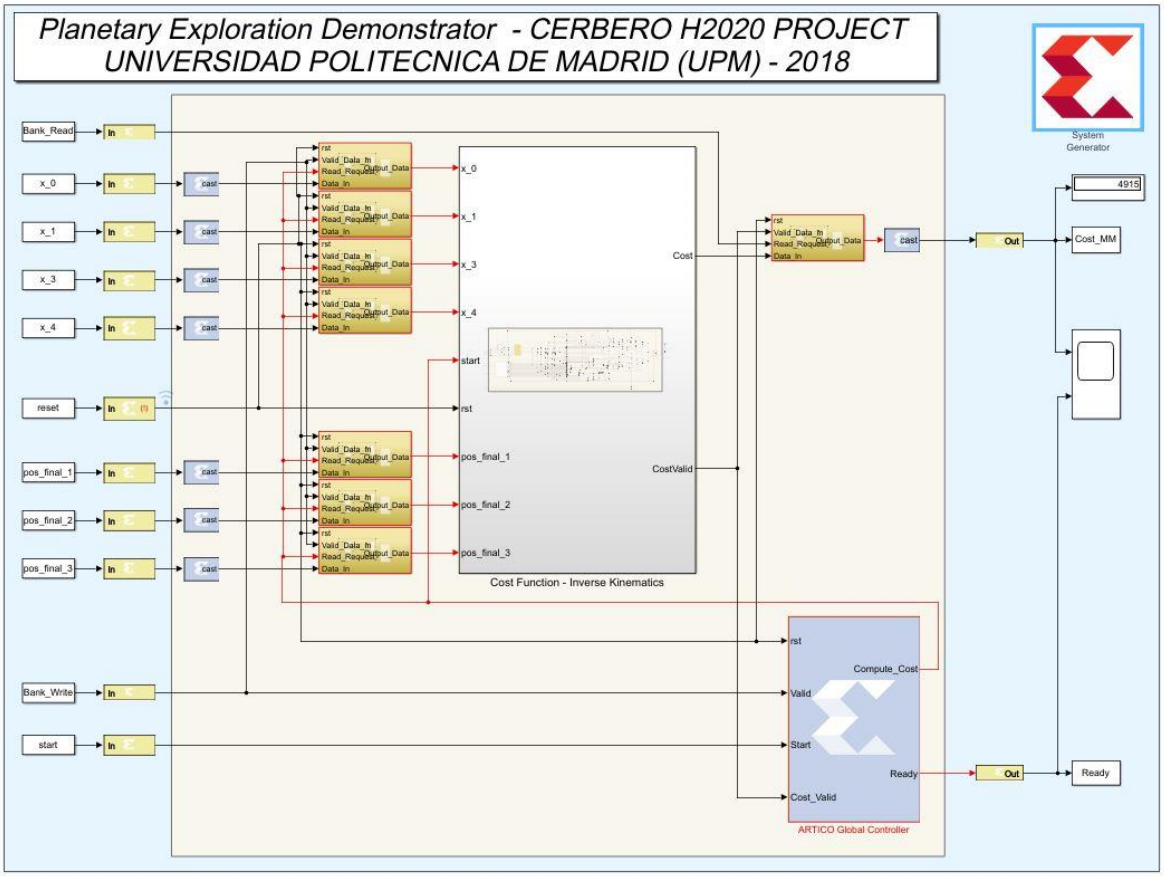

Figure 3-5 - Simulink implementation of the ARTICo3 compliant Robotic Arm cost function to be integrated in ARTICo^3

3.1.2.2 Fault Tolerance

The ARTICo^3 architecture supports configurable fault-tolerant execution with on-demand hardware redundancy. Its internal datapath can be dynamically altered to deliver the same input data to one (Simplex), two (DMR, Double Modular Redundancy) or even three (TMR, Triple Modular Redundancy) copies of the same hardware accelerator and, after each processing round, gather results through a voter unit. This infrastructure enables fault masking against Single-Event Upsets (SEUs) only when working in TMR, thus the voter unit has been enhanced so that error reports are generated whenever a fault is detected (either in DMR or TMR). A memory-mapped register interface allows external agents to access the error counts per slot at any given point in time.

It is important to highlight that ARTICo^3 provides fault tolerance at component level with no additional cost, since it is an already existing architectural feature. For the M18 demonstrator, only the inherent fault masking capabilities of the ARTICo^3 computing fabric will be shown (hardware redundancy and voter unit), leaving the fault detection and optional recovery processes as tasks to be performed by the Adaptation Manager (by reading error status registers and deciding on whether to trigger adaptation or not) that will be developed for the M36 demonstrator.

In any case, within the CERBERO project, faults will be always emulated in both demonstrators (M18 and M36) using functional fault injection: each hardware accelerator will have a multiplexed output and two different operation modes (normal, faulty) that will be controlled using configuration registers also present in the accelerator logic. Hardware accelerators will show functionally correct behavior, generating valid output values, until their operation mode is changed to generate random or tied-off values instead. Extensive system-level assessment of fault tolerance (i.e., not only at component-level and using more advanced fault injection mechanisms) is not envisioned.

3.1.2.3 Runtime Trade-Off Management with MDC

In this section, we report some initial consideration of the inverse kinematics computation problem based on [2017_Artistidou]. Inverse kinematics (IK) techniques can be divided in four main categories:

1) *Analytic solutions*: used for mechanisms with low degree of freedom.
2) *Numerical solutions*: they require various iterations to converge over a solution, but are better capable of dealing with degree of freedom, and multiple end effectors (e.g. fingers of a hand or arms of a body).
3) *Data-Driven methods*: these methods are based on learning-based techniques. They require having available large datasets, and using pre-learned postures to match the position of the end-effector. These characteristics make them suitable for complex problems such as human-like structures.
4) *Hybrid methods*: these methods decompose the IK problems into *analytical* and *numerical* components, to reduce complexity. As data-driven methods, the hybrid methods are suitable for complex problems, such human-like structures or multiple end-effectors ones.

The studies on-going, to be presented in at the review meeting, are focusing on the second category due to the fact that we do not need to deal with body-like mechanisms. In this family lie all of the Jacobian methods (based on inversed, pseudo-inversed and transposed Jacobian matrices), the Newton methods (based on second-order Taylor expansion), Heuristic methods and Cyclic coordinate descendant. The Nelder-Mead Simplex method adopted by TASE is one of the most common numerical methods applied in multi-dimentional spaces. It does not require calculating derivatives, thus it is suitable for problems with non-smooth functions. However, Nelder–Mead convergence properties have been proven only for problems of small dimensions [1998_Lagarias], and Spensieri et al. [2016_Spensieri] run some tests applying Nelder-Mead using several starting points. If the weld point cannot be reached the search gets easily stuck.

For these reasons, we are currently investigating different numerical algorithms to understand which feature and trade-off they expose. As a preliminary result, we have discovered that the Selective Damped Least Squares (SDLS) method requires fewer iterations to converge, with no ad hoc damping constants, while returning the best results in terms of lack of oscillation, as demonstrated in [BK05]. Nevertheless, it has a high

computational cost, presenting the slowest performance time among all Jacobian methods.

In the next few months we plan to implement different algorithms and enable fast, low overhead, switching among them, leveraging on virtual-reconfiguration capabilities offered by MDC. We intend to present the results of this investigation, currently still on going, during the M18 review meeting; while MDC coarse-grained virtual reconfiguration will be integrated in M36 demonstrator.

3.1.3. SW KPIs monitoring and display

In order to integrate Performance Monitoring Counter (PMC) instrumentation based on the PAPI library, PAPIFY has been developed[2]. PAPIFY, as described in D5.7 and previous, has been integrated within CERBERO with both PREESM (which generates instrumented code) and SPIDER (which adopts runtime measurements to trigger adaptation). This code generator aims at instrumenting C code automatically, including both timing and event monitoring using PAPI capabilities, hereafter called *papification*. Although automatic instrumentation is currently generated from the integration of PREESM and PAPIFY, for M18 demo, given that no PREESM implementation of the Nelder-Mead algorithm adopted by TASE is available, the instrumented code of the optimization algorithm is generated manually only for the purpose of the demo. Besides, as indicated in D6.7, the specific KPIs monitored are throughput and latency.

Papify includes function calls for monitoring each function of the demo individually. These extra functions have been developed and included in a new library called *eventLib*, adding a new level of abstraction to PAPI. As a result, the user is able to decide whether to characterize or not each function in design time. During the execution, a csv file is generated for each instrumented function storing both timing and PMC values. *PAPIFY-VIEWER* reads this file to visualize, from either a function or a processing element point of view, the activity of the instrumented function.

The procedure to automatically instrument each actor is the following. If a function is being *papified*, a set of extra function calls are included during code generation. In this case, the individual monitoring information related to the *papification* of each actor is stored. The specific information is the following: (1) the list of events being monitored during the execution; (2) the processing element executing the function (ARM processor, etc.), which isolates the function monitoring to a specific core; (3) the values obtained for each PAPI event being monitored; (4) the timings, i.e., start and stop times. A proof of concept of this methodology is provided as part of D5.7, and will be used as a basis for M36 demonstrator with the Nelder-Mead algorithm.

[2] https://github.com/preesm/preesm
https://github.com/dmadronal/papify
http://preesm.insa-rennes.fr/website/index.php?id=beta-papify-tutorial
https://gitlab.citsem.upm.es/papify/papify

PAPIFY functions leverages on the *eventLib* Library. The resulting extra abstraction layer aims at unifying the procedure of monitoring each function independently, hence, to configure the instrumentation in terms of the hardware resource that is executing the function (the so-called PAPI components) and/or the specific events being monitored. This is a preliminary step for self-adaptation where both heterogeneous platforms (including several PAPI components for HW monitoring) and some additional Key Performance Indicators (KPI), such as energy consumption estimation, will be supported in M36.

3.2. Tools

The tools used for the development of the demonstrator are standard development environments. The RCU architecture will be based on a development board with a Zynq-7000 for implementing HW and SW solutions. The commercial Xilinx design suite has been adopted to provide configuration of the RCU. Moreover, Python Spyder will be used to provide serial communication and 3D representation of the calculated path planning before sending to the physical system.

In the PE demonstrator, the CERBERO tools are used both at deployment and during the design phases. The lower part of the CERBERO stack is mainly used. In particular, in M18 demonstrator ARTICo³ computing infrastructure and PAPIFY are the adopted tools.

3.3. Development and deployment environment

The development and deployment for M18 PE demonstrator is composed by the WidowX robotic arm, a development board with a Zynq-7000 and a PC. The following functions and validations are planned:

- Proper operation of the robotic arm and the RCU in two scenarios:
  - Motion planning in free space.
  - Collision-free motion planning with known obstacles.
- Integration of the CERBERO tools with the robotic arm:
  - Parallelization and Fault-Tolerance (ARTICo³).
  - Event monitoring (PAPIFY / PAPIFY VIEWER).

At M36 the final demonstrator will include assessment for the following CERBERO technologies too.

- ARTICo^3 + MDC
- SPIDER + HW accelerators
- PREESM + PAPIFY + SPIDER

Please note that separate proof of concept of these latter are already provided as part of D5.7.

4. Tests, Results and Feedback

4.1. Tests

For testing and validating the M18 Planetary Exploration demonstrator, a test scenario with a known obstacle will be implemented. This scenario will contain some functionalities defined in D2.3 and D2.4. These functionalities will be enhanced in M36 demonstrator to cope all Planetary Exploration requirements and goals.

The development of the M18 demonstrator is split into three parts:

- Proper operation of the robotic arm and RCU.
- Integration of the CERBERO tools/technologies with the robotic arm.
- Proof of concept.

Proper operation of the robotic arm and integration with the CERBERO tools has been tested with different trajectories and configuration parameters in order to verify proper operation of the system.

4.2. Results

In the table below the results from the demonstrator development activity were added (in *italic*) to the expected results as specified in D2.4.

Table 4-1 Planetary Exploration Goals and Results

ID	Goal	Results M18 demonstrator
PE1	Development of a Hardware / Software (HW/SW) co-design for Rad-Tolerant control of robotic arm for planetary exploration using adaptable COTS FPGAs.	Minimization of energy consumption and costs, while keeping/improving resiliency. - *At M18 migration of algorithms from Software to Hardware, time monitoring and display will be tested to improve execution time of the system. Energy consumption monitoring and optimization will be implemented at M36.*
PE2	Develop integrated open toolchain environment for development of robotic arms for space missions with focus on multi-viewpoint system-in-the-loop virtual environment.	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. - *At M18 stand-alone tools have been used, the integrated toolchain will be assessed at M36.*
PE3	Development of a (self-)adaptation methodology with supporting tools.	Efficient support of architectural adaptivity, according to radiation effects and harsh environmental conditions. - *Adaptation strategies in this scenario are used to speed-up computation (parallelization over ARTICo^3 slots) when energy permits it, to achieve a fault-*

		tolerant behaviour (exploiting redundancy over ARTICo³ slots), and to guarantee different functional execution trade-offs (studies exploiting MDC adaptivity support are on-going). - *Based on SPIDER an adaptation manager will be developed for M36 demonstrator to trigger adaptation of the system by using the ARTICo³ and MDC. To close and assess the adaptation loop PAPIFY/PAPIFY-VIEWER will be used to sense running system conditions (exploration on software has already been performed).*

4.3. Feedback

Based on the high level functionalities described in D2.4 and the Demonstrator Skeleton described in D6.7, this is what we achieved in the Planetary Exploration demonstrator:

- Verification of the proper operation of the inverse kinematics and optimization algorithms to provide collision-free motion planning.
- Execution time is improved by using parallel methods of computation.
- Optimal parallelization to find a trade-off between execution time and soft motion planning.
- Component level fault tolerance is achieved thanks to Double and Triple Modular Redundancy built in support.

According to the results obtained from the PE demonstrator, a more detailed description of the PE scenario will be presented in D2.5 CERBERO Scenarios Description – Version 3.

5. Conclusion

Considering that the long term goals of this demonstrator are 1) *failure tolerance*, 2) *self-healing* (by means of self-adaptation), *environment adaptation* and *self learning* to the harsh physical environment, and 3) *power measurement and optimization*, we are satisfied with M18 achievements, since we were able to implement already (by means of CERBERO technologies and tools) *failure tolerance*, *performance enhancements,* and also *event monitoring*.

We managed to validate the motion planning and to connect CERBERO technologies to the robotic arms. Current analysis took into account free path planning algorithms. In addition, the integration of HW accelerators allowed to migrate SW functionalities to HW resources in order to adapt the RCU to different scenarios in terms of execution time.

Event monitoring represents the basis to achieve runtime KPIs measurements at M36. KPIs addresses in the M18 demonstrator were: latency, throughput and resource utilization.

Additional tools from the CERBERO toolchain will be integrated in the next iteration (M36) to guarantee better adaptation to the environment and more functionalities will be added:

- Sensor fusion (proximity sensors) and path planning improvements
- Self-Healing
- Reinforcement Learning
- Encryption

6. References

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

[D2.7] CERBERO D2.7 Technical Requirements (Ver. II)

[D2.4] CERBERO D2.4 Description of Scenarios (Ver. II)

[D3.4] CERBERO D3.4 Modeling of KPI (Ver. I)

[D3.6] CERBERO D3.6 Cross Modelling Methodology for CPS (Ver. I)

[D5.6] CERBERO D5.6 Framework Components (Ver. I)

[D6.7] CERBERO D6.7 Demonstration Skeleton (Ver 1)

[2017_ Artistidou] Aristidou, A. , Lasenby, J. , Chrysanthou, Y. and Shamir, A. (2017), Inverse Kinematics Techniques in Computer Graphics: A Survey. Computer Graphics Forum. . doi:10.1111/cgf.13310

[1998_ Lagarias] J. C. Lagarias, J. A. Reeds, M. H. Wright, P. E. Wright, Convergence properties of the Nelder-Mead simplex simplex method in low dimensions, SIAM Journal of Optimization, Vol. 9, No. 1, pp 112-147, 1998

[2016_Spensieri] Domenico Spensieri, Johan S. Carlson, Robert Bohlin, Jonas Kressin, Jane Shi, Optimal Robot Placement for Tasks Execution, Procedia CIRP, Volume 44, 2016, Pages 395-400, ISSN 2212-8271, https://doi.org/10.1016/j.procir.2016.02.105.

[BK05] BUSS S. R., KIM J.-S.: Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools 10*, 3 (2005), 37–49.