ESROCOS: A ROBOTIC OPERATING SYSTEM FOR SPACE AND TERRESTRIAL APPLICATIONS

M. Muñoz Arancón [1], G. Montano [2], M. Wirkus [3], K. Hoeflinger [4], D. Silveira [5], N. Tsiogkas [6], J. Hugues [7], H. Bruyninckx [8], I. Dragomir [9], A. Muhammad [10]

[1] GMV Aerospace and Defence, Isaac Newton 11 PTM, Tres Cantos, 28760 Madrid, Spain, mmunoz@gmv.com
[2] Airbus Defence and Space Ltd, Gunnels Wood Road, SG1 2AS Stevenage, United Kingdom, Giuseppe.Montano@Airbus.com
[3] Deutsches Forschungszentrum für Künstliche Intelligenz GmbH - Robotics Innovation Center, Robert-Hooke-Straße 1, 28539 Bremen, Germany, malte.wirkus@dfki.de
[4] Deutsches Zentrum für Luft - und Raumfahrt Ev, Linder Höhe, 51147 Köln, Germany, Kilian.Hoeflinger@dlr.de
[5] GMVIS Skysoft SA, Av. D.Joao II Lote 1.17.02, Torre Fernao Magalhaes 7°, 1998025 Lisboa, Portugal, Daniel daniel.silveira@gmv.com
[6] Intermodalics BVBA, Gaston Geenslaan 9, 3001 Leuven, Belgium, Nikolaos Tsiogkas nikolaos.tsiogkas@intermodalics.eu
[7] Institut Superieur de l'Aeronautique et de l'Espace, Avenue Edouard Belin 10, 31055 Toulouse, France, Jerome.HUGUES@isae-supaero.fr
[8] Katholieke Universiteit Leuven, Oude Markt 13, 3000 Leuven, Belgium, Herman.Bruyninckx@kuleuven.be
[9] Universite Grenoble Alpes, 700 Avenue Centrale, 38401 Saint Martin d'Heres, France, iulia.dragomir@univ-grenoble-alpes.fr
[10] VTT Technical Research Centre of Finland Ltd., Tekniikankatu 1, 33720 Tampere, Finland, ali.muhammad@vtt.fi

ABSTRACT

ESROCOS (http://www.h2020-esrocos.eu) is a European Project in the frame of the PERASPERA SRC, (http://www.h2020-peraspera.eu/), targeting the design of a Robot Control Operating Software (RCOS) for space robotics applications. The goal of the ESROCOS project is to provide an open-source framework to assist in the development of flight software for space robots, providing adequate features and performance with space-grade Reliability, Availability, Maintainability and Safety (RAMS) properties. This paper presents the ESROCOS project and summarizes the approach and the current status.

1. INTRODUCTION

In the industrial robotics domain, it is common practice for robotics manufacturers to adapt proprietary Real-Time Operating Systems (RTOS) regarding specific functional and business demands. This results in the development of non-standardized, proprietary solutions. The space robotics industry has been following a similar path, with the additional constraints imposed by the operating environment and the particularities of each mission, leading to minimal reuse across developments.

On the other hand, open-source robotics frameworks such as ROS [1], ROCK [2] or GenoM [3] have flourished in the academic world, enabling the growth of reusable functional block libraries and allowing for faster application development. However, these open-source frameworks are not developed with sufficient level of quality assurance, and they are not suitable for space or critical terrestrial robotics applications with demanding RAMS properties.

ESROCOS will provide an open-source robotics framework designed from the beginning to support the development of space robotics. ESROCOS will provide the foundation of a robotics software development eco-system not only relevant for the space industry, but also fulfilling the requirements of other industries like underwater, nuclear or medical robotics. To gain this large and multi-domain user base (including industry), the resulting tools have to be simple to use and a diverse offering of modules has to be available.

The paper is structured as follows: § 2 enumerates the specific objectives and challenges, identified for the project; § 3 describes the ESROCOS framework; and § 4 summarizes the project status and planning.

2. PROJECT OBJECTIVES AND CHALLENGES

In order to meet the goal of providing to the robotics community an open-source framework that is the base of future space robotics applications, the ESROCOS project has defined several objectives:

1. **Develop a space-oriented RCOS:** the development of ESROCOS follows the ECSS standards [4,5] and implements certain capabilities specific to space communications (ECSS PUS services [6]) and hardware support (LEON processor and bus drivers) on top of a space RTOS such as RTEMS [7]. ESROCOS

covers three levels of quality: laboratory, high-reliability and critical software. Each level has different quality assurance and hardware representativeness.

2. **Integrate advanced modelling technologies:** ESROCOS will include a complete model-based methodology, including robotic-specific modelling semantics, and supporting the design and integration of software components, as well as the verification of the structural and behavioural properties at system level. By relying on formal verification and automatic code generation, this methodology will help to reduce the number of defects that are introduced in the software. ESROCOS will provide a kinematic modelling tool, based not just on models, but semantic models, making possible model-to-model transformation, and not just the traditional model-to-text transformations.

3. **Focus on the space robotics community:** actors that have a leading role in state-of-the-art robotics missions have participated in the definition and review of the ESROCOS requirements, ensuring that they are aligned with the needs of current and future missions.

4. **Allow for the integration of complex robotics applications:** ESROCOS will support mixed-criticality applications using time and space partitioning, which allows running applications with different levels of quality on the same on-board computer, ensuring no propagation of failures among them.

5. **Avoid vendor lock-in:** ESROCOS will integrate existing open-source tools as well as new developments, and will be distributed as open-source software to the robotics community.

6. **Leverage existing assets:** instead of attempting to develop a new framework from scratch, ESROCOS builds on existing technologies such as the TASTE framework [8] developed and maintained by ESA and partners. It will incorporate tools, libraries and approaches from well-established robotics software ecosystems such as ROCK and ROS. ESROCOS will also offer interoperability capabilities with these robotics frameworks, so that critical software components can be tested in combination with existing algorithms or tools (data viewers, simulators, etc.).

7. **Cross-pollinate with non-space solutions and applications:** the design of ESROCOS will benefit from the experience gathered from the nuclear robotics domain, with very stringent RAMS requirements.

The development of ESROCOS is not an isolated activity. The PERASPERA SRC has launched six Operational Grants (OG) to target different aspects of space robotics (software, autonomy, data fusion, sensors, manipulation and validation). The activities of the six OGs are coordinated, and an integrated Interface Control Document (ICD) is being developed. ESROCOS provides the software infrastructure and methodology that will enable the other OGs to comply with the software RAMS requirements of space systems. Therefore, the coordination with these parallel activities is essential to ensure mutual success.

The results of the ESROCOS project will be validated taking as reference two representative applications: a planetary rover mission, and an in-orbit servicing scenario using a robotic arm. In addition, validation in a nuclear robotics scenario is also foreseen. The validation will follow a two-stepped approach, with an internal validation phase against ESROCOS' requirements, followed by a demonstration of the system in two facilities that will be provided by the Facilitators project (OG6) [9].

The objectives of ESROCOS are ambitious and present several challenges. The most relevant are:

- Successfully combine innovative and well-established modelling approaches at different levels (system architecture, physical configuration, software architecture and behaviour, etc.).
- Integrate tools and libraries in a consistent way, providing a unified view to the final user.
- Coordinate with the parallel Operational Grant activities to ensure that they are compatible at interface level and that together they cover all the capabilities required by space robotics systems.
- Provide a powerful and flexible framework that can be adopted by the community, mainly in the space robotics domain, but also for terrestrial applications with strict RAMS requirements.

The following section presents the ESROCOS framework, as well as the approach selected to overcome these challenges and accomplish the project objectives.

3. DESCRIPTION OF THE FRAMEWORK

3.1. Scope and workflow

ESROCOS is a framework for developing robot control software applications. It includes a set of tools that support different aspects of the development process,

from architectural design to deployment and validation. In addition, it provides a set of core functions that are often used in robotics or space applications.

The ESROCOS framework supports the development of software following the ECSS standards. It does not by itself cover all the development phases and verification steps, but it facilitates certain activities and ensures that the software built can be made compatible with the RAMS requirements of critical systems.

Figure 1 summarizes the main activities supported by the ESROCOS framework. The rounded white boxes indicate activities, grey rectangles denote software artefacts (models, source code, applications, etc.), and dashed boxes group related items. The software artefacts are either product of the activities (identified in *italics)*, or directly provided by the framework as functional blocks to use in the activities. The figure illustrates how the activities, their products and the functional blocks combine to form a consistent workflow for the production of distributed robot application software.

The starting point of the workflow is the formal modelling of the robot and the application. The model-based approach facilitates the early verification of the system properties, in particular for RAMS. The modelling activities encompass the following aspects:

- The robot's kinematic chain, in order to produce a formal model of the robot motion, from which software can be automatically generated.
- The hardware and software architecture of the application, including non-functional properties (real-time behaviour, resource utilisation, etc.).

The models allow for different analyses to verify the non-functional properties of the system and iteratively refine the system architecture. ESROCOS relies on both existing and newly developed tools to support the different modelling aspects.

The model of the application may include functional building blocks, either provided by ESROCOS or specifically generated from the models (e.g. a hybrid dynamics instantaneous motion solver). This model can then be used to automatically generate the software scaffolding for the application, consisting of the skeleton of the application components and the glue code that enables the inter-component communication. The application-specific behaviour is implemented and integrated in this structure, making use of libraries to support the required functionalities.

The application binaries can then be built and deployed in a runtime platform. Distributed applications are possible, with components running in separate nodes or partitions. ESROCOS supports SPARC/RTEMS and x86/Linux platforms. The former is intended for usage in space-quality systems, while the latter aims towards laboratory setups, as well as validation and debugging.

The framework includes the *autoproj* [10] package and build management system to handle software builds and component dependencies. The management system allows the developer to seamlessly combine ESROCOS, 3rd-party and own components to build an application.

ESROCOS can be used to model applications using time and space partitioning, in order to build mixed-criticality systems in which components with different

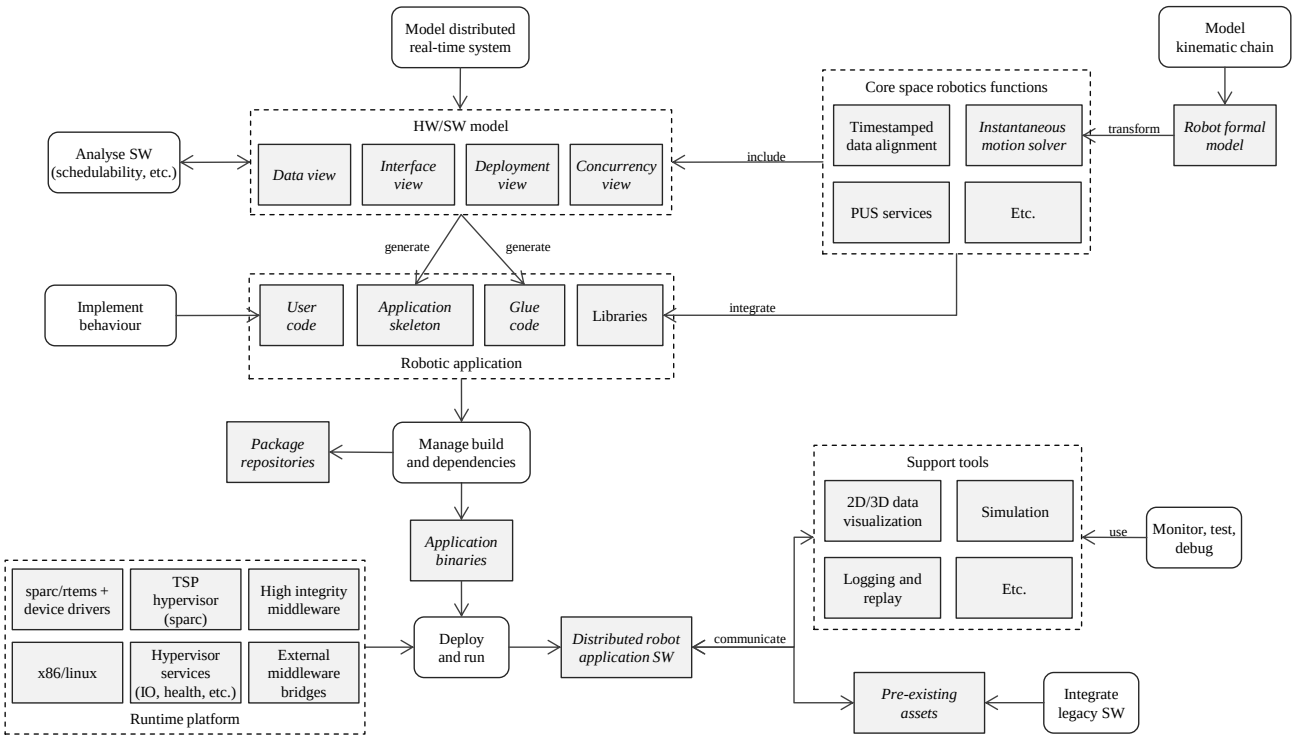

Figure 1. Development of a robot control application with ESROCOS

RAMS levels can safely coexist. These applications can be deployed on a SPARC (LEON) platform using the AIR hypervisor.

The communication between the application components at runtime is enabled by the PolyORB-HI middleware. ESROCOS will provide also bridge components that enable the communication with external middleware for ROS and ROCK. This will allow the robotics engineer to use tools from these ecosystems (data visualizers, simulators, etc.) for testing and debugging the application. A selection of tools will be provided ready to use with ESROCOS, with all the required data types and interfaces. In addition, middleware bridges will allow the user to integrate existing software assets and run them together with newly built software in a distributed environment.

The following sections explain in detail the main elements of the ESROCOS framework.

3.2. Robot and software modelling tools

Kinematic chain models. ESROCOS includes tools to formally model the kinematics and dynamics of ideal, lumped-parameter robots of all possible configurations (serial, mobile, parallel, hybrid, multi-DOF joints, multi-articular actuation), with a structured set of interdependent modelling languages: geometry (e.g., line, point, pose), kinematics (e.g., joint, link, inverse dynamics, Jacobian, singularity), mathematical representations (e.g., frame, quaternion), numerical representations (e.g., homogeneous transformation matrix, n-vector), digital representations (e.g., 16-bit integers, IEEE floats), and physical dimensions (e.g., length, meter, energy, Joule). For all properties and transformations that are physically relevant for robots (e.g., forward kinematics, hybrid dynamics), code-generators will be provided, that take a specification in a formal modelling language as input, and generate code with verifiable properties (e.g., no dynamic allocation).

Distributed real-time system models. ESROCOS relies on the TASTE framework to model robotics applications from a real-time software perspective. TASTE is an open source framework that allows the development of embedded, real-time systems. It relies on technologies such as standardized modelling languages (e.g., ASN.1 [11] and AADL [12]), code generators and real-time systems, and allows for the generation of application skeletons and the production of the system executable. The designers implement their embedded systems using a set of views, abstracting the user from the implementation details of the underlying platform (e.g., operating system, drivers) and guaranteeing the fulfilment of real-time properties.

Model analysis and verification. The TASTE framework supports the analysis of the real-time behaviour and resource utilisation of the software. ESROCOS will complement these capabilities with BIP (Behaviour, Interaction, Priority) [13] formal models, which offer additional possibilities to analyse the software and verify properties at a behavioural level.

The verification and validation of TASTE models is done with the BIP framework via a model translation between the two formalisms using the TASTE2BIP tool, currently under development. The BIP framework offers several analysis tools: iFinder [14] verifies the satisfaction of safety properties, SMC [15] evaluates a system's performance metrics, and user-guided/automated simulation validates the given requirements. The workflow of these tools is illustrated in Figure 2. The model translator, the simulator and the SMC are being developed/expanded in the ESROCOS framework to consider more complex systems with hard real-time constraints, such as robot controllers.

BIP can be used to formally model not only the nominal operation of the system but also its faults, enabling the correct construction of fault tolerant systems. The nominal behaviour and the Fault Detection, Isolation and Recovery (FDIR) strategy are given by the user in a state machine formalism such as SDL or BIP. The faults model is given explicitly by the user in the BIP formalism, targeting mainly the erroneous behaviour of hardware. Then the two models – nominal and faulty – are put together and the user-designed FDIR component can be validated by simulation in the BIP framework.

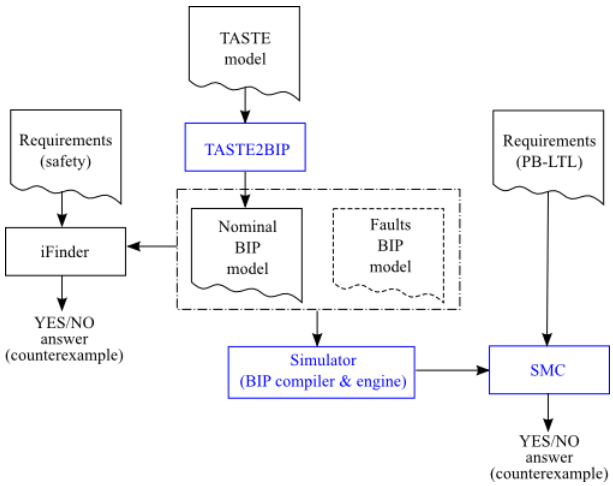

Figure 2. Verification and validation workflow of TASTE models with the BIP framework. Components in blue are new/extended tools developed in the ESROCOS framework.

3.3. Runtime platforms

The ESROCOS framework supports the development of applications for three target quality levels: laboratory,

high-reliability and space. Laboratory applications focus on reduced development times with light quality assurance activities. Space quality applications have demanding RAMS requirements and must follow a strict development process. Finally, the high-reliability level is a balance between the two.

ESROCOS uses the TASTE toolset, which supports different hardware and software platforms. For laboratory applications, ESROCOS targets x86/Linux systems and provides a set of tools for logging and data visualization, among others, to facilitate the development and debugging of applications. For space-quality applications, the framework targets SPARC/RTEMS systems and includes formal modelling tools that enable correct-by-construction software development and verification of RAMS properties.

Space robotics applications are complex and they may combine functions with different degrees of criticality and real-time requirements. For instance, the functional layer may rely on hard real-time control loops, while higher-level functions may require a varying amount of time and memory to complete an operation. In order to safely support such diverging requirements, the ESROCOS framework offers the capability to design time- and space-partitioned systems using TASTE and the AIR/XKY hypervisor.

The TSP (Time and Space Partitioning) concept, also known as IMA (Integrated Modular Avionics), emerged as an opposing concept to the federated architecture, and offered the possibility of integrating multiple functions into partitions of the same set of physical resources, allowing the aeronautical industry to manage software growth in functionality and in efficiency. Partitioning keeps applications from inadvertently influencing each other by enforcing strict separation, segregating computing resources in space and time [16].

In a system operating under the specification ARINC 653 [17], all the inter-partition communications take place over statically defined ports and channels. Ports can be configured for a queuing or sampling discipline. A message-passing technique is used to communicate through a channel between two partition ports.

The architectural concept of IMA was transposed from the aeronautical to the space domain by ESA and other agents of the European space industry in the IMA-SP (IMA for Space) activity [18]. The IMA-SP platform defines a two-level software executive. The system executive level is composed by a software hypervisor, also called partitioning kernel, which segregates computing resources between partitions. A second level, the application level, is composed by the user's applications running in a partition. Each application can implement a system function by running several tasks managed by the Partition Operating System (POS), which is modified to operate along with the underlying hypervisor. In the context of IMA-SP, the selected POS was the space-qualified version of the Real-Time Executive for Multiprocessor System (RTEMS). A graphical representation of an IMA-SP executive is depicted in Figure 3.

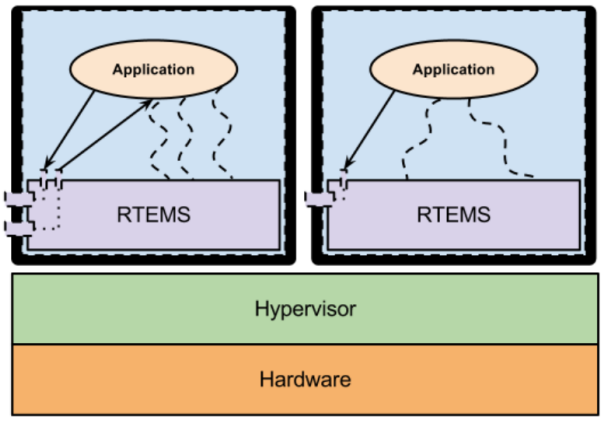

Figure 3. The IMA-SP executive composed by a hypervisor and two partitions, each one running its version of RTEMS.

The ESROCOS framework includes the AIR/XKY hypervisor, also known as ARINC Interface in RTEMS (AIR) separation kernel [19]. It is an ARINC 653 compliant time and space partitioned operating System that was originally based on RTEMS technology.

3.4. Functional building blocks and tools

One important factor in the success of robotics frameworks is that they provide ready-to-use components (algorithms, sensor and actuator controllers, etc.) and tools that greatly speed up application development. In order to address the specific needs of space systems, ESROCOS will offer a set of reusable components for common robotics functions compatible with the space quality level, as well as a set of TM/TC services. In addition, the framework will integrate popular visualization and simulation tools, and provide support for the reuse of legacy software developed in ROS and ROCK.

Generic robotics building blocks. ESROCOS is intended as a software development framework for the space robotics domain. It must therefore support common development tasks within the envisioned domain by means of tools, libraries and software components. An example of this has been introduced in Section 3.2, with the kinematics modelling tool. In addition to this tool, we intend to provide software that supports the developer in calculating geometric transformations at library level. This tool will allow

configuring a graph with named geometric reference frames as nodes and frame transformations generated by user code as edges. Also from user code, transforms between arbitrary frames within the specified graph can be queried. Similar functionality is provided within ROCK with the *transformer* [20] and in ROS with the *tf* mechanism [21].

Another common problem in the robotic domain is to handle data that might be created and processed asynchronously. To support the handling of such data, we intend to implement a mechanism to buffer multiple asynchronous data streams for the selection of best corresponding samples at a given time point, inspired by the *stream aligner* [22] in ROCK.

These two proposed tools stand exemplary as first developments toward a potentially growing tool box for robotic developments. The pool of user components developed within the ESROCOS framework and the support for 3rd party frameworks (as described above) work into the direction in making ESROCOS a valuable tool for robotic application development.

TM/TC services building blocks. The Telemetry and Telecommand Packet Utilization Standard (PUS) defines a set of on-board services for ground monitoring and control of spacecraft, including space robotics missions. The ESROCOS framework provides a set of PUS services, implemented at logical level, that can be integrated in applications to emulate the operation of the robot from a ground control station.

Support tools from 3rd-party frameworks. ESROCOS will integrate existing tools to facilitate the development of robotics applications. One of the areas that will be supported by these tools is the visualization of robotics data. The *vizkit3d* [23], from the ROCK ecosystem, and the *RViz* [24] tool, from ROS, will be integrated in ESROCOS. These tools have proven to be extremely useful in the development process, allowing the user to have a full 3D view of the system status, which increases the understanding of the system functionality. This in turn can increase the pace of development and debugging process.

Another tool to be integrated is the Gazebo simulator [25]. Although technically not part of ROS, Gazebo is highly integrated with it, making it an ideal addition to ESROCOS. Simulation tools are essential for the development of any robotics application. By eliminating the need to use real hardware in certain phases of the development, they allow fast iterations and lower costs.

The combination of visualization and simulation tools can vastly improve the efficiency of the development of any space robotics application. It can provide a stable platform to simulate and test software developed in a cost and time effective manner.

Interaction with 3rd-party frameworks. The ESROCOS framework will include support for the integration with 3rd-party frameworks, in particular for ROCK and ROS, which are popular within the scientific robotics community. The support is established by providing a mapping for the abstractions of the different frameworks, conversion functions between basic data types from the particular frameworks, and a set of software components that serve as bridge between the ESROCOS middleware (PolyORB-HI) and an external middleware at runtime.

During execution, the bridge component is visible in both middleware domains as a regular component. It performs the data conversion on-line and passes the respective data from its source to the sink, bridging between both middleware domains.

This bridge mechanism will potentially allow for the integration of a larger pool of available components and development tools.

3.5. Build and package management

In ESROCOS, many individual software developments are arranged to work together and build the overall development framework. Many of these software developments also have relevance out of the scope of ESROCOS and have their own maintenance structure or development road-map. For example, the TASTE tool, the kinematics modelling tool or the AIR/XKY hypervisor are software packages that are used in different contexts than ESROCOS. Also, an infrastructure for reusing other developments made within the framework should be provided.

To allow this, we use the *Autoproj* tool, which allows specifying a set of software packages from different sources, such as Git or SVN repositories, archives such as tar or zip, or system package management software such as apt. It can also handle different kinds of packages like source code packages with different languages and build systems, or binary packages.

3.6. Validation approach

The validation of the ESROCOS software products will be done in two steps. The software will be fully validated within the project with a combination of unitary, integration and end-to-end system tests. Then, in order to demonstrate the capability to integrate with the developments of the other PERASPERA OGs, the ESROCOS framework will be used to develop two applications that will leverage the validation facilities developed by the Facilitators project (OG6).

The ESROCOS framework will be validated according to the two reference scenarios defined by the PERASPERA SRC as representatives of future space robotics missions: a planetary exploration rover mission and an in-orbit assembly mission. To validate the ESROCOS concept, these use cases will be oriented to cover the laboratory (x86/Linux) and the space representative (SPARC/RTEMS) environments. The ESROCOS application will be deployed on a space-representative on-board computer. For the orbital scenario, the application will control a robotic arm and a camera, perform Cartesian and joint space real-time motion control, and acquire and display telemetry. For the planetary scenario, the application will drive a rover platform in Ackermann and point-turn modes, acquiring images and platform telemetry during the traverse.

In addition, in order to demonstrate the benefits of the ESROCOS framework for other critical robotics applications beyond space systems, ESROCOS will be validated in a nuclear robotics scenario. VTT hosts the DTP2 platform of ITER since 2007. The purpose of DTP2 is to develop and demonstrate maintenance of the ITER divertor with remotely operated robots with high degree of RAMS attributes. The demonstration equipment at DTP2 includes the Cassette Multifunctional Mover (CMM) robot and control system along with the infrastructure such as the full scale Divertor Region Mockup. The system reproduces the particular constraints of the domain. For example, due to the fusion-induced radiation, the operations cannot be relayed on video feedback from cameras and must rely on a high-quality calibrated virtual model that provides the operators with the missing information about the CMM state.

An ESROCOS application will be executed on the CMM robot to control a subset of functionalities. The performance of the CMM will be benchmarked against the existing control system. The control system shall be able to avoid failure or ensure a successful recovery without damaging the task or the robot.

4. PROGRESS AND PLANNING

The ESROCOS project started in November 2016 and successfully passed its System Requirements Review (SRR) in February 2017. The planned duration of the activity is 27 months.

Updates on the activity can be found at the project's website (http://www.h2020-esrocos.eu). The ESROCOS consortium plans to make the framework available through GitHub (https://github.com/ESROCOS), once the software reaches a sufficient level of maturity for widespread usage.

Figure 4. DTP2 facility (top) and five-degrees-of-freedom CMM robot at the DTP2 facility at VTT Tampere (bottom)

5. REFERENCES

1. The Robot Operating System. Online at http://www.ros.org
2. Robot Construction Kit. Online at http://rock-robotics.org
3. Ceballos (2011), A., et al. GenoM as a robotics framework for planetary rover surface operations. ASTRA, 2011, p. 12-14.
4. European Cooperation for Space Standardization (ECSS), Space Engineering: Software. ECSS-E-ST-40C, 6-Mar-2009
5. European Cooperation for Space Standardization (ECSS), Space Product Assurance: Software Product Assurance. ECSS-Q-ST-80C, 6-Mar-2009
6. European Cooperation for Space Standardization (ECSS), Space Engineering: Telemetry and Telecommand Packet Utilization. ECSS-E-ST-70-41C, 15-Apr-2016
7. RTEMS Real-Time Operating System. Online at: https://www.rtems.org/
8. Perrotin, M., Conquet, E., Dissaux, P., Tsiodras, T., Hugues, J. (2010), The TASTE Toolset: turning human designed

heterogeneous systems into computer built homogeneous software. In: ERTS 2010, Toulouse (2010)
9. H2020 Facilitators project: Facilities for Testing Orbital and Surface Robotics Building Blocks. Online at: http://www.h2020-facilitators.eu/
10. Autoproj. Online at https://github.com/rock-core/autoproj
11. International Telecommunications Union, Recommendation X.680-X.693 (08/2015): Information Technology - Abstract Syntax Notation One (ASN.1) & ASN.1 encoding rules.
12. SAE International, Architecture Analysis & Design Language (AADL), AS5506C (18/01/2017).
13. A. Basu, M. Bozga, J. Sifakis (2006), Modeling heterogeneous real-time components in BIP, in SEFM '06: Proceedings of the Fourth IEEE International Conference on Software Engineering and Formal Methods. Washington, DC, USA: IEEE Computer Society, 2006, pp. 3–12.
14. Ben-Rayana S., Bozga M., Bensalem S., Combaz J. (2016), RTD-Finder: A Tool for Compositional Verification of Real-Time Component-Based Systems. In: Chechik M., Raskin JF. (eds) Tools and Algorithms for the Construction and Analysis of Systems. TACAS 2016. Lecture Notes in Computer Science, vol 9636. Springer, Berlin, Heidelberg
15. Nouri, A., Bensalem, S., Bozga, M. et al. Statistical model checking QoS properties of systems with SBIP. International Journal of Software Tools for Technology Transfer (2015) 17: 171.
16. C. Silva. Integrated Modular Avionics for Space applications: Input/Output Module. Master's thesis, Instituto Superior Técnico, October 2012.
17. ARINC653, Arinc Specification 653-1. Avionics Application Software Standard Interface. Published: October, 2003 by RTCA.
18. IMA-SP (Ima for Space Activity). Online at: http://esaconferencebureau.com/docs/12c25_2310/sa1440_deredept.pdf?sfvrsn=2
19. AIR. Online at: http://www.gmv.com/en/Products/air/
20. ROCK Transformer. Online at http://rock-robotics.org/stable/documentation/data_processing/transformer.html
21. ROS tf. Online at http://wiki.ros.org/tf
22. Stream aligner. Online at http://rock-robotics.org/stable/documentation/data_processing/stream_aligner.html
23. Graphical User Interfaces in ROCK. Online at http://rock-robotics.org/stable/documentation/graphical_user_interface/index.html
24. RViz. Online at http://wiki.ros.org/rviz
25. Gazebo Simulator. Online at: http://gazebosim.org/

ACKNOWLEDGEMENTS

We would like to thank the European Commission / Research Executive Agency and the members of the PERASPERA Programme Support Activity (ESA as coordinator, ASI, CDTI, CNES, DLR and UKSA) for their support and guidance in the H2020 ESROCOS activity. The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 730080.