ESROCOS: DEVELOPMENT AND VALIDATION OF A SPACE ROBOTICS FRAMEWORK

Miguel Muñoz Arancón [1], Malte Wirkus[2], Killian Hoeflinger[3], Nikolaos Tsiogkas[4], Saddek Bensalem[5] Olli Rantanen[6], Daniel Silveira [7], Jérôme Hugues [8], Mark Shilton [9], Herman Bruyninckx [10]

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

ABSTRACT

The H2020 ESROCOS project has built an open-source framework for the development of space robotics software. The ESROCOS framework provides a model-based engineering approach and a collaborative development process. It integrates proven and new technologies, providing modelling and verification tools as well as reusable components that facilitate the production of high-quality robot control software. While other robotics frameworks exist, ESROCOS is designed from the ground up to fulfil the needs of the space robotics community, which requires sound software engineering processes. This paper presents the outcomes of the ESROCOS project, the evaluation activities performed in three case studies (on-orbit servicing, planetary exploration and nuclear industry), and the results of this evaluation.

1 INTRODUCTION

The development of software for robotic systems has become easier with the emergence of robotics frameworks, with ROS, the "Robot Operating System" [1], being the most popular among them. The term Robot Control Operating System (RCOS) has evolved, describing frameworks like ROS and others with a similar aim, such as OROCOS [2], GenoM [3], ROCK [4] and CLARAty [5]. It can be defined as a framework for the development of robotics applications that may provide a component model, a runtime environment and a set of tools and reusable components to build robotics software. An RCOS allows the user to easily combine generic and application-specific components to build applications, and provides tools to support development and testing.

The RCOS has become an essential tool in robotics research and in industry. The standardization of components and interfaces through a component model, together with the use of common tools and practices, fosters a software ecosystem that facilitates the development of applications. However, existing RCOS are not suitable for critical applications with stringent Reliability, Availability, Maintainability and Safety (RAMS) requirements. It is not feasible to modify an existing RCOS to comply with the requirements of critical software development. Therefore, to leverage the benefits of the RCOS concept in critical domains, specifically developed frameworks are needed. Yet, past efforts to develop a standard robot control software at ESA failed to be adopted by the community, even if they achieved their immediate technical objectives.

To address the issues that prevented a wider adoption, the PERASPERA project (Plan European Roadmap and Activities for Space Exploitation of Robotics and Autonomy) [6] identified the need for an open-source RCOS as one of the "building blocks" to be produced in the frame of the first call of activities of the H2020 Space Robotics Technologies Strategic Research Cluster (SRC). The ESROCOS project [7] was selected to develop this building block, and it has aimed to answer the aforementioned needs by:

- Developing an RCOS specifically designed for space software processes and technologies.

- Integrating advanced modelling technologies to support the design and verification of the architectural and behavioural properties of the system from the early stages.
- Focusing on the space robotics community, addressing their needs and use cases.
- Coping with the complexity of robotics applications, both at development and at execution time.
- Avoiding proprietary solutions and vendor lock-in by using open-source licenses.
- Leveraging existing tools and reusable components, benefiting from their maturity and user base.
- Exploring the needs of other domains with stringent RAMS requirements, such as nuclear robotics.

The ESROCOS project was executed by a consortium of ten industrial companies and academic institutions led by GMV, and spanned over 27 months from Nov-2016 to Jan-2019.

The following sections present an overview of the ESROCOS framework, detail the supported development approach and the tools and components provided by the framework, present the evaluation of ESROCOS on three case studies, and summarize the status of the framework and the future work.

2 OVERVIEW OF THE ESROCOS FRAMEWORK

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.

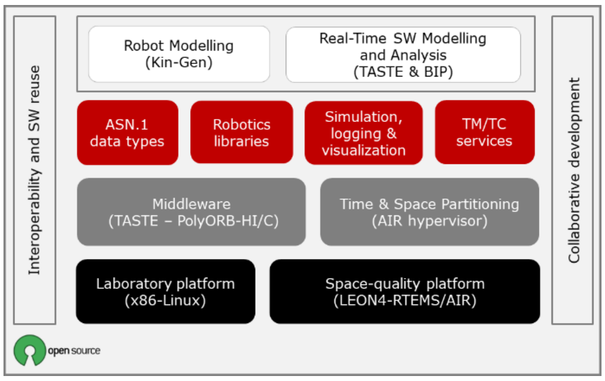
Interoperability and SW reuse
Robot Modelling (Kin-Gen)
Real-Time SW Modelling and Analysis (TASTE & BIP)
ASN.1 data types
Robotics libraries
Simulation, logging & visualization
TM/TC services
Middleware (TASTE – PolyORB-HI/C)
Time & Space Partitioning (AIR hypervisor)
Laboratory platform (x86-Linux)
Space-quality platform (LEON4-RTEMS/AIR)
Collaborative development
open source

Fig. 1. The ESROCOS Framework.

The ESROCOS framework is intended to support the development of software following the ECSS standards for space software. 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 Reliability, Availability, Maintainability and Safety (RAMS) requirements of critical systems.

Fig. 1 summarizes the main elements of the ESROCOS framework. At the top of the figure there are tools for robot and software modelling. These are supported by the common data types for component interfacing, basic libraries for robotics functions, test tools, and monitoring and control services. The middleware layer allows for the management and communication of software components at runtime. Time and space partitioning capabilities are provided to isolate application components of at runtime and prevent failure propagation between components. Finally, the applications may run laboratory platforms, space-quality hardware or a combination of both.

The boxes on the sides of the figure represent orthogonal concerns. Firstly, ESROCOS integrates with third-party frameworks to leverage existing code and tools. Secondly, ESROCOS supports a collaborative development approach based on component reuse. Finally, the figure highlights that ESROCOS is open-source and relies on non-proprietary technologies in order to encourage usage and contributions from the community.

3 DEVELOPMENT WITH ESROCOS

The requirements of space robotics systems may lead to complex software and hardware architectures that combine heterogeneous computer nodes and software functions. Capabilities such as fast control loops, sensor data processing, autonomy functions or monitoring and control, all of which may be needed in a typical application, have very different requirements. In addition, the design and validation of the application often requires specific tools for simulation, data recording and replaying, or visualization.

The ESROCOS framework provides a heterogeneous set of tools and reusable components that address different needs of the robotics applications. The framework relies on a model-based engineering approach, and in particular on the TASTE toolchain (see § 3.1), to link together the different elements.

The model-based approach was a requirement of the ESROCOS project. It provides separation of concerns between the different design aspects and the deployment details, and enables early formal verification activities to reduce the amount and impact of design errors.

ESROCOS supports model-based development in two dimensions:

- Modelling of the robot system, in particular of the robot kinematics, using the Kin-Gen tool (see § 3.3).
- And modelling and validating the software architecture and behaviour, using TASTE (see § 3.1) and BIP (see § 3.2).

In addition, ESROCOS provides an infrastructure to maintain software implementations divided in a number of individual packages, where each package can have its own development cycle. Combined with a set of common robotics data types for modelling the interfaces between components, this mechanism allows the user to easily combine existing and new software blocks to design an application. By enabling the reuse of components developed by the community, and facilitating that newly developed components are published and contributed back, ESROCOS makes it possible that a user community of commercial vendors and open-source projects emerges and supports a lively software ecosystem around the framework.

The following subsections describe in more detail the main tools and components provided by the ESROCOS framework.

3.1 The TASTE Toolchain

TASTE (The ASSERT Set of Tools for Engineering) [8] is a toolchain developed by the European Space Agency (ESA) to model heterogeneous real-time distributed systems. It is the backbone of ESROCOS. At development time, it allows the user to model the hardware and software architecture of the system.

The user models the system in TASTE using four views, each addressing a different aspect of the system:

- Data view: describes the data definitions of the system using the ASN.1 standard, as well as data encoding aspects.
- Interface view: models the functional aspects of the system, structured in components with defined real-time behaviour and interfaces; it is described using the AADL standard [9].
- Deployment view: describes the hardware architecture of the system as a set of nodes and data buses, and the allocation of software components and interfaces to these; it is also described with AADL.
- Concurrency view: generated from the previous two, this view represents the system in terms of real-time software artifacts (tasks, subprograms, etc.), enabling the schedulability analysis of the system by automated tools and the tuning of different parameters by the user.

The TASTE editor allows the user to model the system using these four views, and to validate the model. The interface and deployment views are edited graphically, while the data view is textual. From the model, TASTE generates code using the Ocarina tool. This code manages the system initialization, task execution and communication aspects, using the PolyORB-HI middleware. The user must then provide the implementation of the individual components. TASTE supports different languages such as Ada, C, C++, SDL state machines, VHDL (for supported FPGAs) or Simulink. TASTE supports several hardware platforms, including RTEMS on LEON processors and Linux on x86, and communication links such as TCP/IP and SpaceWire.

Individual TASTE components can be exported for reuse in different TASTE applications and deployed to different architectures and communication buses. The toolchain thereby takes care of the compilation, communication and runtime details. To allow the independent development of compatible software components, the ESROCOS framework provides a library of common data types and provides support in development with specific components, e.g., for monitoring and control and data visualization (see § 3.5).

3.2 The BIP Tools

ESROCOS complements the TASTE modelling capabilities with BIP (Behaviour, Interaction, Priority) [10], a formal language for modelling and analysing heterogeneous real-time systems. The BIP language and tools go further than TASTE in capturing the behaviour of the software and the interaction between the components, allowing for new scalable formal verification techniques of the system model.

The following BIP tools can be used to support the modelling and analysis of robotics applications built with ESROCOS:

- BIP Compiler & Engines: the BIP compiler generates C++ code that can be linked to different engines that support model execution and simulation for analysis.
- iFinder/iChecker: it allows for the verification of requirements (safety, performance, etc.) on BIP models.
- Statistical Model Checking (SMC-BIP): it runs a representative number of simulations and

statistically checks whether a requirement (safety, performance, etc.) is satisfied.

- FDIR tool: permits the generation of fault detection and recovery software components that can be embedded in the target system.

The integration of BIP in ESROCOS is done via an automated model transformation from TASTE to BIP models.

3.3 The Kin-Gen Kinematics Modelling Tool

The Kin-Gen tool [11] has been developed within the ESROCOS project to model the characteristics of the robotic platform. Kin-Gen allows the user to model the robot kinematics using an innovative approach based on model composability. The robot is modelled as a set of kinematic trees and an arbitrary frame. The model can be verified with semantic checks. Additionally, custom solvers can be defined as queries to the model, and code can be generated for these custom solvers, ensuring correctness.

The tool provides a complete toolchain for forward and inverse kinematics. Future evolutions of the tool will cover other aspects of the robot system and its environment.

3.4 The AIR Hypervisor

Robotics software must combine functions ranging from real-time control loops to non-deterministic planning algorithms, which differ in their criticality level and execution constraints. This may be an issue for the development of critical robotics software, which must satisfy the safe execution of certain critical functions.

The Time and Space Partitioning (TSP) paradigm consists of separating different applications running on the same on-board computer by enclosing them in isolated partitions with their own assigned memory and processor utilization. The aim is the execution of applications with different criticality in such a way that a failure in one cannot affect the others. ESROCOS includes the AIR hypervisor [12], an ARINC 653 [13] compliant hypervisor that uses paravirtualisation to enable the execution of partitioned applications and supports RTEMS and ARINC 653 APEX as guest operating systems.

3.5 Support Tools and Components

In addition to the elements described above, the ESROCOS framework includes other tools and components to support the development and testing of space robotics applications.

The most important is a collection of common robotics data types modelled in ASN.1. These types, derived from the ROCK and ROS frameworks, represent common robot and sensor data and allow modelling the interfaces between reusable components.

In addition, a set of libraries that support common robotics and space software functionalities are provided. These libraries are prepared for easy integration in TASTE applications.

- Transformer: a C++ library to model and compute at runtime geometric frame transformations.
- Stream Aligner: a C++ library to manage the synchronization of multiple data streams.
- Data Logger: a C++ library to allow for data logging and replay to support testing.
- Data Visualization: a C++ library for robot and sensor data visualization in 3D based on Vizkit3D.
- PUS Library: a missionizable C library that provides a subset of the ECSS Packet Utilization Standard (PUS) [14] for monitoring and control of space assets. The library implements the services at logical level (the packet encoding uses ASN.1 instead of the standard PUS frames). It includes an On-Board Control Procedure (OBCP) engine based on MicroPython [15], allowing for the simultaneous execution of several OBCPs that can interact with the on-board software system.

ESROCOS includes tools for interoperability with the ROS and ROCK frameworks so that prospective users can leverage the rich ecosystems existing around these frameworks as well as their existing software assets. The aim is to facilitate the transition from the laboratory to the level of quality and verification demanded by space systems. To this end, ESROCOS provides:

- Tools to import data type definitions from ROS and ROCK to the ASN.1 format used by TASTE.
- Tools to generate bridge components that enable seamless communication between TASTE functions and ROS or ROCK components running on a separate middleware environment.

Finally, ESROCOS integrates with external tools and libraries commonly used in robotics software development, such as the OpenCV image processing library, the Eigen linear algebra library, the Gazebo simulator or the RVIZ data visualization tool.

3.6 Installation of ESROCOS

ESROCOS is available as open-source software in public repositories (see § 5 below). The framework relies on the Autoproj tool to manage both the components of the framework itself and those of the user application. Each component is hosted in its own repository and includes a manifest file declaring its dependencies. The Autoproj tool handles the building and installation of the framework and the applications using this information. The fact that the framework and the user application components are indistinguishable from each other makes it possible that individual components can be contributed back to the ESROCOS ecosystem, and that the users of the framework can seamlessly mix components provided by the framework, by other suppliers and by their own organisation.

The installation of ESROCOS creates a workspace hosting both framework components and user applications. The setup is organized in three package sets that define the components available in the workspace:

- Core: the components provided by the framework.
- External: existing tools and libraries that the core components depend on.
- Universe: contributed packages (drivers, algorithms, etc.) contributed by third parties.

As an ecosystem develops around the ESROCOS framework, the Universe package set will be extended with new capabilities and functions that can be reused in new applications.

4 CASE STUDIES

The validation in the space reference scenarios took place in two test facilities provided by the H2020 FACILITATORS project [16] in the frame of the Space Robotics Technologies SRC. In addition, the Finnish Technology Research Centre VTT was part of the project consortium and provided a nuclear robotics test facility to perform a case study based on a terrestrial application.

The ESROCOS framework was evaluated in three case studies by developing representative applications targeting on-orbit satellite servicing, planetary exploration and nuclear robotics. The focus of the evaluation was the production and testing of a set of robotics applications that exercise the different elements of ESROCOS.

The reference applications addressed the functional layer of the robotics systems, specifically for a rover and two manipulator arms. The aim of the tests was not to build fully autonomous systems, but to demonstrate the software capabilities that will allow to do so in the future, considering that the future activities of the Space Robotics Technologies SRC will integrate other building blocks providing higher level functions, such as data fusion and autonomy.

4.1 Planetary Scenario

The planetary exploration scenario was demonstrated using the BRIDGET rover in Airbus DS Mars Yard facility at Stevenage (UK). The tests included the tele-operation of the rover, logging and replay capabilities and control of a CAN device. As in the orbital scenario, both laboratory (Linux PC) and space-representative (GR740) hardware were used.

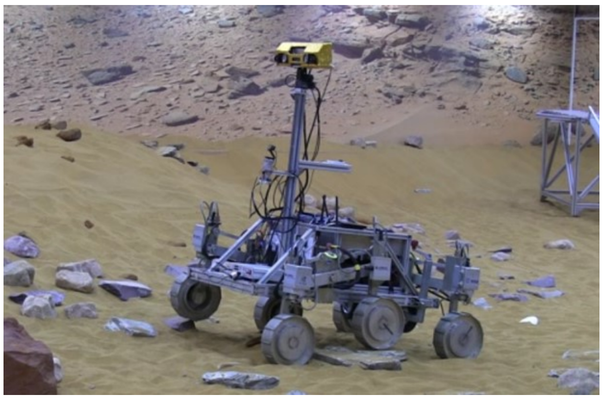

Fig. 2. Testing in the planetary exploration reference scenario: BRIDGET rover in the Marys Yard facility.

The case study consisted of four test cases. For each of them, an application was modelled with TASTE and implemented using the different components provided by the framework. The test cases addressed:

- Rover teleoperation and control.
- Rover trajectory data logging and replay.
- CAN bus device management.
- Integration of data processing functions.

The test campaign demonstrated the functionality of the software modules Stream Aligner, Data Logger, Transformer, as well as the possibility to use BIP for modelling and executing safety routines for robotic systems.

4.2 Orbital Scenario

The on-orbit satellite servicing scenario was demonstrated at GMV's platform-art facility in Tres Cantos (Spain). The tests consisted in the teleoperation of an UR-5 manipulator with a camera as end effector to inspect a satellite mock-up. Different variants of the scenario were tested, including simulation aspects and targeting laboratory-type hardware (Linux PC) and space-representative avionics (GR740 board running RTEMS).

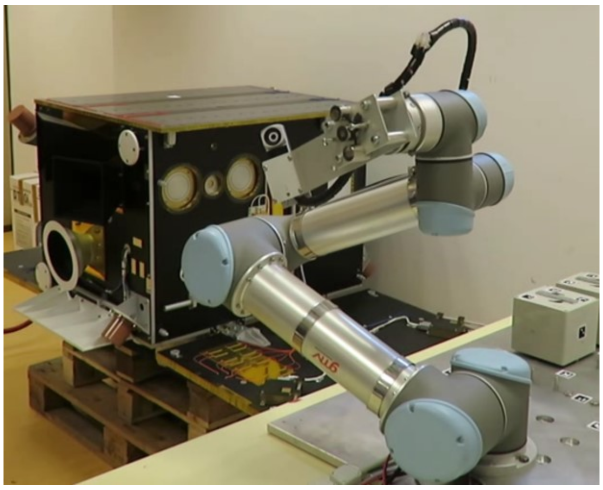

Fig. 3. Testing in the on-orbit servicing reference scenario: UR-5 manipulator and satellite mock-up in GMV's robotics laboratory.

The case study consisted of four test cases with their respective ESROCOS applications:

- Control of the manipulator on simulation, using a laboratory platform (Linux on x86).
- Control of the actual manipulator and camera using a laboratory platform (Linux on x86).
- Control of the actual manipulator using space-representative hardware (GR740 board running RTEMS) and SpaceWire data bus.
- EtherCAT driver testing (however, this test case could not be performed, see § 4.4).

The test case applications were modelled using TASTE, BIP and Kin-Gen, and demonstrated the capabilities of the Data Visualization, integration of the Gazebo simulator, PUS Services and interoperability with ROS and ROCK provided by ESROCOS.

4.3 Nuclear Scenario

The nuclear robotics demonstration took place at the VTT Diverter Test Platform 2 at Tampere (Finland). A control application for the Cassette Multifunctional Mover (CMM) robot was developed with ESROCOS and deployed at the control centre. The CMM robot is a robotics demonstrator for the ITER experimental fusion reactor, intended for the replacement of the elements that compose the bottom part, so called divertor, of the toroidal enclosure of the reactor.

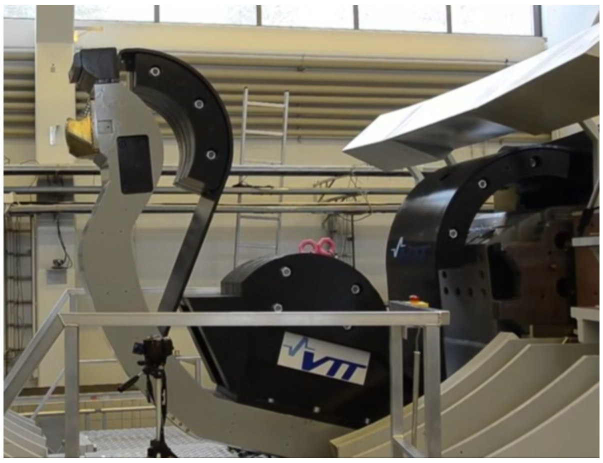

Fig. 4. Testing in the nuclear robotics scenario: CMM robot in the DTP2 facility at VTT.

The case study included two test cases with their respective ESROCOS applications:

- Manipulator trajectory planning and control using interpolation in Cartesian space.
- Manipulator trajectory planning and control using interpolation in joints space.

Both applications ran on laboratory hardware (Linux on x86) and interacted with the CMM controller through an UDP bus. The applications were modelled in TASTE and included a kinematic solver generated with Kin-Gen. The tests also demonstrated the data visualization functionality and the integration with the Gazebo simulator.

4.4 Test Results, Constraints and Limitations

The results of the evaluation of ESROCOS have been in general satisfactory, proving that the framework is functional and useful to support the development of space robotics applications following a sound modelling, implementation and verification approach.

Nevertheless, not all the technical objectives of the project have been achieved, and the case studies have shown some limitations that should be taken into account by prospective users of the system:

- The installation of the ESROCOS framework is not fully managed by Autoproj, and some dependencies and components need to be manually installed.
- The TASTE toolchain has some limitations for large application models, exhibiting excessive memory usage and long build times.
- The driver configuration in TASTE is complex, with some elements configured at model level and some others at middleware source code level.

- The transformation of TASTE models to BIP is partially automated, and some concepts in the deployment view must be manually modelled.
- The integration of the AIR hypervisor in TASTE is not complete, and currently it is not possible to generate code for models that combine AIR and other types of hardware nodes.
- The SOEM EtherCAT driver could not be ported to the GR740 board selected as space-representative avionics board due to the use of a deprecated network stack in RTEMS for LEON4.
- The ESROCOS runtime components have not been verified to the level required by flight software in terms of unit testing, coding rules or code review.

Despite these shortcomings, the results of the validation of ESROCOS have been positive, and the framework is usable in its current state to support the development and verification of robotics applications.

5 STATUS AND FUTURE WORK

The ESROCOS project was successfully completed in January 2019. The public results from the project, including the open-source software, can be found at the project's website [17] and GitHub page [18]. The evolution of ESROCOS continues in the second phase of the Space Robotics Technologies SRC that will use and extend the framework to support the development of several application demonstrators. The H2020 MOSAR project, in particular, is in charge of the maintenance of the ESROCOS building block during this phase of the SRC, coordinating the corrections and extensions of the framework in the different projects.

In addition to the usage and maintenance foreseen within the SRC, the fact that ESROCOS is based on technologies that have a trajectory outside the project will also contribute to the success of the platform. The ESROCOS framework may benefit from the further development of its constituent tools, and likewise improvements in the framework may reflect on the tools, creating a live ecosystem.

6 CONCLUSION

The ESROCOS project has developed a robotics framework that addresses the needs of the space robotics community. The framework was validated by developing robotic applications for three reference scenarios: on-orbit servicing, planetary exploration and nuclear robotics. The resulting RCOS will be used as a building block in future Space Robotics Technologies SRC activities, and is available for wider use by the community under open-source licenses.

7 REFERENCES

1. Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A. (2009, May). ROS: an open-source Robot Operating System. *ICRA workshop on open source software*, Vol. 3, No. 3.2, p. 5.
2. Bruyninckx, H. (2001). Open robot control software: the OROCOS project. *IEEE International Conference on Robotics and Automation (ICRA).* Vol. 3, pp. 2523-2528. IEEE.
3. Ceballos, A., De Silva, L., Herrb, M., Ingrand, F., Mallet, A., Medina, A., & Prieto, M. (2011). GenoM as a Robotics Framework for Planetary Rover Surface Operations. *11th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*. ESA.
4. The Robot Construction Kit (ROCK). Online at: http://rock-robotics.org.
5. Nesnas, I. A. (2007, June). CLARAty: A collaborative software for advancing robotic technologies. *Proceedings of NASA Science and Technology Conference.* Vol. 2. NASA.
6. PERASPERA (Plan European Roadmap and Activities for Space Exploitation of Robotics and Autonomy). Online at: https://www.h2020-peraspera.eu/.
7. Muñoz Arancón, M., Montano, G., Wirkus, M., Hoeflinger, K., Silveira, D., Tsiogkas, N., Hugues, J., Bruyninckx, H., Dragomir, I., Muhammad, A. (2017). ESROCOS: A robotic operating system for space and terrestrial applications. *14th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA).* ESA.
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. *Embedded Real Time Software and Systems (ERTS2).*
9. Architecture Analysis & Design Language (AADL). (2017). SAE International. AS5506C.
10. Basu, A., Bozga, M., & Sifakis, J. (October 2016). Modeling Heterogeneous Real-Time Components in BIP. *Fourth IEEE conference on Software Engineering and Formal Methods (SEFM)*, pp. 3-12. IEEE.
11. Frigerio, M., Scioni, E., Pazderski, P., & Bruyninckx, H. (2019). Code generation from declarative models of robotics solvers. *Third IEEE International Conference on Robotic Computing (IRC)*, pp. 369-372. IEEE.
12. The AIR Hypervisor. GMV. Online at: http://www.gmv.com/en/Products/air/.

13. RTCA. (October, 2019). ARINC653, Arinc Specification 653-1. Avionics Application Software Standard Interface. RTCA.
14. Space Engineering: Telemetry and Telecommand Packet Utilization. ECSS-E-70-41C (April 15, 2016). European Cooperation for Space Standardization (ECSS).
15. MicroPython. Online at: https://micropython.org/.
16. H2020 FACILITATORS project: Facilities for Testing Orbital and Surface Robotics Building Blocks. Online at: https://www.h2020-facilitators.eu/.
17. The ESROCOS project: European Space Robotics Control and Operating System. Online at: https://www.h2020-esrocos.eu/.
18. ESROCOS at GigHub. Online at: https://github.com/ESROCOS.

ACKNOWLEDGMENTS

We would like to thank the European Commission 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 ESROCOS activity. We also thank all the participants in the ESROCOS project for their contributions to the results presented here.

This project has received funding from the European Union´s HORIZON 2020 research and innovation programme under grant agreement No 730080.