EXOSKELETON CONTROL OF THE ROBONAUT THROUGH RAPID AND ROS

Thomas Krüger and André Schiele
ESA Telerobotics & Haptics Laboratory
ESA/ESTEC
Noordwijk, The Netherlands
Thomas.Krueger@esa.int Andre.Schiele@esa.int

Kimberly Hambuchen
Robotic Systems and Technology Branch
NASA/Johnson Space Centre
Houston, Texas, USA
Kimberly.A.Hambuchen@nasa.gov

***Abstract*— In this paper, the integration between the ESA human arm exoskeleton and NASA's Robonaut 2 is described. The integration has been performed through a NASA provided communication middleware called RAPID (Robot API Delegate). In ESA, RAPID was extended to allow real-time communications as required for bilateral control applications. In combination with automatic code generation of the control and mapping algorithms, a stand-alone application can be created and deployed on a real-time system based on Xenomai Linux.**

The paper will describe the technical implementation of the interfaces between the Exoskeleton and Robonaut 2. A brief review of the RAPID framework is presented and its advantages and shortcomings for future bilateral control applications. The paper describes how a RAPID- bridge for the ESA X-Arm-2 exoskeleton has been implemented and outlines some preliminary experiments that have been done between the exoskeleton and a simulation system of Robonaut 2. The simulation environment VERVE, provided by NASA has been used for this purpose. A test-campaign between ESTEC Telerobotics & Haptics Laboratory and the Dexterous Robotics Laboratory at Johnson Space Centre has been set up for the uni-lateral control of the Robonaut 2 simulator from the ESA Exoskeleton at ESTEC.

The paper also provides an outlook on the implementation with the Robonaut 2 simulation available in ROS.

Keywords*— *METERON, Telerobotics, DDS, Data distribution service, ROS, Robot Operating System, RAPID, Robonaut, R2, ESA, X-ARM-2, EXARM, Exoskeleton

I. INTRODUCTION

In the frame of the METERON project [1] robots on ground will be teleoperated from space onboard the ISS. One of the main input devices is the ESA Exoskeleton Robot 1 (XR-1), an evolution of the EXARM [2] and the X-Arm-2 [3] exoskeletons. The XR-1 is an exoskeleton for the right arm. Among others the Robonaut 2 (R2) [4] is one of the targeted slave systems.

Furthermore, in the frame of the CCSDS (The Consultative Committee for Space Data Systems), a new telerobotics working group has been established, aimed at designing a new standard for space telerobotics. The goal of the standard is to lead to interoperability between robotic assets. The operations experiments presented in this paper directly contribute to finding optimal solutions for defining such standards.

It is the goal of the presented paper to describe the experiments performed for controlling R2 using the ESA Exoskeleton. In preliminary steps, i.e. before connecting the exoskeleton, a simple joystick input device was used to control a simulation of R2. The inputs from the joystick were used to move a single or multiple joints of the simulated robot in Cartesian coordinates. This was done to firstly develop and test the communications interface, before controlling the real robot. All tests were performed using the NASA R2 simulation, which provides the same inverse kinematics algorithms as R2, over a RAPID bridge, while VERVE visualized the simulation. The major objective of the presented experiments was to learn about the functional principle of RAPID ("Robot API Delegate) [5] and to realize a robot- to-input device-type communication based upon this framework.

II. RAPID IMPLEMENTATION OVERVIEW

A. RAPID - Robot API Delegate

While the ESA exoskeleton message structure for communication with external devices was developed using custom UDP messages, R2 and other NASA robots use RAPID for communicating with remote tools. The usage of data-centric communication middleware is an important step towards interoperability between robots and controller/monitor interfaces. RAPID (Robot Delegate API) is a set of defined messages and message templates which are exchanged using DDS (Data Distribution Service) [6].

As such, RAPID itself is providing the "concept" of how different robotic systems (called "assets") can communicate with each other. The formats and contents of the messages that are sent and received by a robot are described by a set of Interface Definition Language (IDL) files. In principle, each participating robot can derive the commanding and monitoring messages from these IDLs. Generally used command and telemetry contents are provided in RAPID. Commands that are used across robots, e.g. "move to 6DOF location" are predefined for ease of use across multiple robots. Telemetry, e.g. joint values and pose/orientation are also predefined. Due to the generic nature of RAPID, though, these are only

suggestions. Individual users can define their own message content.

B. *RAPID on top of DDS*

While RAPID can be used over any middleware dependent on an IDL format for code generation, current consumers of RAPID generally use DDS (Data Distribution Service). DDS is a definition of a middleware of the OMG (Object Management Group). The specific DDS implementation which is given preference in RAPID is RTI-DDS, which is a commercial version. RTI DDS has been designed specifically for the use on different platforms and for real-time applications where timing constraints can be in the range of microseconds. In addition, the RAPID developers established a defined set of Quality of Service (QoS) profiles for their needs, which are described in XML files.

III. IMPLEMENTATION OF THE R2 RAPID COMMUNICATION

A. *R2 Publishers & Subscribers*

A rough overview of the communication messaging is provided in Figure 1. It is easily visible which messages are exchanged but their content is not standardized, since each robot can and should have a derived sub-set message from the full templates. The following tables contains the various publishers and subscribers that R2 can provide and accept. Again, the "config" topics contain the information about how the data is provided (e.g. in text form) while the real data is published via the "sample" topics. For commanding, R2 simply subscribes to the command topic.

Table 1: R2 publishers

(*These have been defined ad-hoc in this project, since no RAPID message for force information had been available)

Message type & topics	
AssetConfig ASSET_CONFIG_TOPIC	Forces definitions*
AssetState ASSET_STATE_TOPIC	Force values*
JointConfig JOINT_CONFIG_TOPIC	Joint definitions
JointSample JOINT_SAMPLE_TOPIC	Joint values
PositionConfig POSITION_CONFIG_TOPIC_LEFTARM, POSITION_CONFIG_TOPIC_RIGHTARM	Position definitions
PositionSample POSITION_SAMPLE_TOPIC_LEFTARM, POSITON_SAMPLE_TOPIC_RIGHTARM	Position values
CommandConfig COMMAND_CONFIG_TOPIC	Definition and specification of commands controllig the robot

Table 2: R2 Subscribers

Message type	Message content
Command COMMAND_TOPIC	Command in the specification of the content of CommandConfig to control the robot

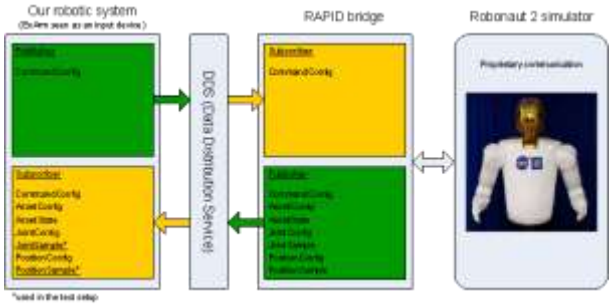

Figure 1: Basic publishers and subscribers of a rapid system

B. *The RAPID config messages*

In principle, since the config messages contain the command set or parameter set of the robot itself, they should be received and analysed automatically. However, in our case, the message content has been derived by the developers. In principle a message of the "command config" contains following information:

- The Header with timestamp, id, priority, etc.
- The name of the command
- The subsystems accepting this command
- The number of parameters it expects
- The type of the parameters

An example is given in Table 3 below.

Table 3 Example commands and addressable subsystems

Instruction	Addressable Subsystems
Setpoint for tool (cartesian)	LeftArm, RightArm
Setpoint for single joint	LeftArm, RightArm, Neck,
Open the Hand	LeftHand, RightHand
Close the Hand	LeftHand, RightHand

For the types of parameters, besides the standard bool, integer, float, double and string there are also:

- RAPID_VEC3d (a 3 x 1 vector)
- RAPID_MAT33f (a 3 x 3 matrix)

An example message to move the hand of the left arm to a certain location would have to be published into the command topic containing the following information

Table 4 Content of the *Setpoint for Tool* Command

Topic Name	**" Command "**
Command Name	**" Setpoint for Tool"**
Addressed subsystems	**" Left Arm"**
Description & values	Type
X, Y, Z coordinates [SI meter]	RAPID_VEC3d
Roll, pitch, yaw [SI rad]	RAPID_VEC3d
Frame, e.g. "R2OriginFrame"	RAPID_STRING

The format of these messages is depended on the specific command even if published into the same topic. For example commanding the joints of the leftarm directly the message would have to have this form and require more parameters:

Table 5: Content of the *Setpoint for single joint* command

Topic Name	**"Command"**
Command Name	**"Setpoint for single joint"**
Addressable Subsystem	**"Left Arm"**
Parameter & Type	Description & values
1 - Float	abduction [rad]
2 - Float	extension [rad]
3 - Float	upper arm outer rotation [rad]
4 - Float	elbow flexion [rad]
5 - Float	forearm outer rotation [rad]
6 - Float	wrist flexion [rad]
7 - Float	wrist ulnar flexio [rad]

C. *The sample messages*

The sample messages contain the information about the robots telemetry, status or measurement data from its instruments. They are published only by the asset (robot) itself. The principle is similar to the config messaging, where the robot publishes how the sample message looks like. For example to retrieve the position of the tool (the hand) in Cartesian coordinates there are two topics, one for the right and one for the left hand. Each of these topics will provide a message in the following form:

Table 6: Content of the PositionSample message

Datafield1	Datafield2	Content
Pose	xyz	RAPID_VEC3d
Pose	rot	RAPID_MAT33f

D. *Message Headers*

For ensuring ensure safe and secure operations as well as proper data logging, each message also contains a header, which provides information about:

- Asset name – address the right robot
- Command source – to identify the sender
- Timestamp
- Status codes
- Priority
- Command Name
- Name of the subsystem

The header is always required and provided. This way, during commanding, the robot on-board software can detect via the command source who is commanding and if the sender has enough privileges to do so. For telemetry and status messages the receiver has information about the timestamp and from which subsystem the message originates.

IV. EXPERIMENT 1: EXOSKELETON – R2A CONTROL

A. *ESA Exoskeleton*

The ESA exoskeletons are a family of different devices for the right human arm. There are 3 different models, called EXARM [2], SAM and X-Arm-2 [3]. The different prototypes aim at studying different aspects of a complete bilateral teleoperation chain, e.g. the investigation of bilateral control, ergonomics, appropriate kinematic mapping, etc. and on the usage from within a micro-gravity environment. They are controlled by different real-time systems and provide and receive data to/from external systems. Following versions are suitable to exchange following data:

- EXARM:

  Output: Joint-level (position, velocity); Cartesian-level (location)

- SAM & X-Arm-2:

  Output: Joint-level (position, velocity, torque); Cartesian-level (location)

  Input: Joint-level (position, velocity, torque);

B. *NASA Robonaut 2*

R2 is a humanoid robot that is designed to assist humans and operate in human environments that may be too dangerous or impossible for humans to work in. R2 unit B is currently assisting astronauts on the International Space Station during IVA activities, with the direction of eventually performing EVA activities. Robonaut consists of a torso with 2 dexterous arms and a 3DoF head.

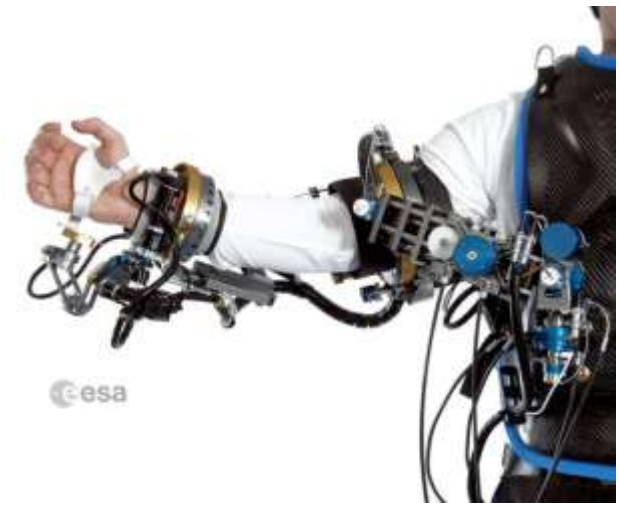

Figure 2 ESA X-Arm-2 Exoskeleton prototype

Figure 3: Impression of the NASA Robonaut2 (source: NASA [7])

C. *The R2 RAPID bridge*

For receiving commands and sending telemetry to the R2 controllers, NASA uses a RAPID bridge. The bridge provides bi-directional access for commands and telemetry. VERVE is used to visualize multiple NASA robotics, including R2. VERVE is fully RAPID compliant. Figure 4 provides an overview of the setup in terms of communication, while Figure 5 provides a schematic overview of the setup components.

D. *VERVE*

Visual Environment for Robotic Virtual Exploration or short VERVE [8] is a software tool developed by NASA, based on the Eclipse framework. It uses Java and can display COLLADA models.

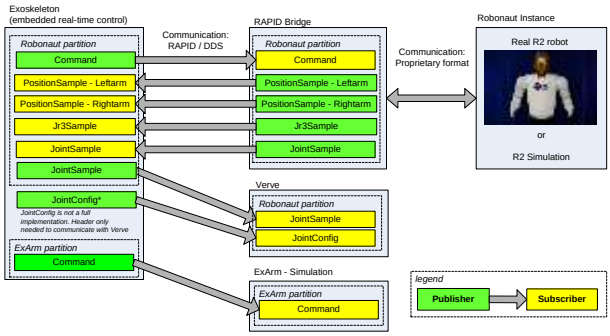

Figure 4: Communication overview from RAPID/DDS perspective including RapidBridge and VERVE

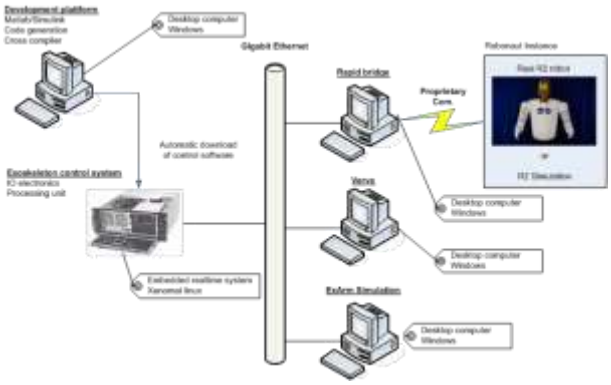

Figure 5: Systems involved in the experiment between X- Arm-2 and R2 Simulator

VERVE's plugins provide rich visual interfaces for real-time situational awareness required for robotic commanding in a multitude of contexts. VERVE is used mainly for robotic activities, such as: exploration system planning, monitoring, coordination, as well as to perform simulations and visualisations for the multitude of NASA robots. In our context, however, only the features for display of the Robonaut torso and harms were used.

As seen in Figure 4, the Command is send from the Exoskeleton to the RAPID-Bridge. The bride calculates internally the dynamics and kinematics and publishes the resulting joint positions via joint sample messages. VERVE uses these as input to visualize the Robonaut motion.

E. *The X-Arm-2 SPAN visualization*

The X-Arm-2 also has its own visualisation tool [9]. For the experiment, the tool was equipped with a subscriber that listens to the joint commands sent by the exoskeleton to the Robonaut, for visualization of the exoskeleton posture being commanded. This way, both X-Arm-2 and the Robonaut arms can be displayed simultaneously. This is useful for further developments to improve the joint mapping algorithms.

Having a simulation and visualization of both devices enables also the use of recorded data for post-hoc analyses or automatic algorithm evaluation. For this work, special recorded movements have been used to tune optimal mappings without using the actual exoskeleton.

F. *Controlling R2 using the X-Arm-2*

Next, the X-Arm-2 was used as input device for controlling the R2 simulation. This way, an additional computer of the X-Arm-2 joins the network. The X- Arm-2 software is generated out of a Simulink model including the Command-Publisher in form of an S- function.

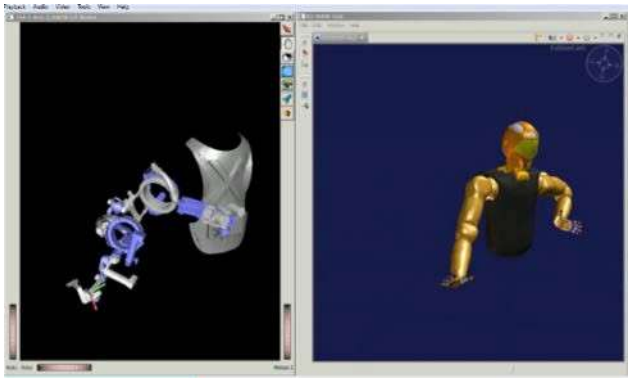

Figure 6: Visualization of the joint values of the X-Arm-2 and resulting pose command for R2

Figure 7: Controlling R2 via XArm2 in Verve

The R2 simulator does not provide capabilities for collision detection and the calculation of collision forces. Thus, with the simulators of R2, no force feedback implementation is possible.

V. EXPERIMENT 2: EXOSKELETON – R2A CONTROL OVER INTERNET EUROPE/USA

After controlling the simulated R2 successfully a test to control it over the distance from ESTEC in the Netherlands to the Johnson Space Center in Texas, USA was performed.

All the components involved before were reused. However when sending DDS message over the internet it should be done using TCP instead of UDP as carrier for the RTPS2. The DDS framework allows this. The advantage of TCP, the transmission control, is favoured over the faster UDP due to router/firewall issues.

In the case of the RTI-DDS implementation the service is called routing service. This service basically establishes a TCP connection to a defined second DDS- router. The router is another DDS node in the local network. It selects and sends/receives the messages which are relevant from its distant counterpart.

In the current stage the results are proof of functioning only. It was possible to send commands to move the tool of R2 in Cartesian coordinates from ESTEC. The R2 bridge calculated the corresponding joint values and sent them back to the Netherlands where they have been displayed in VERVE. Due to the prototypical character of this experiment no performance analyses of latency and jitter of the connection have been recorded.

VI. EXPERIMENT 3: EXARM – R2A CONTROL IN ROS

Later in the collaboration, it was decided by the R2 development team to provide the R2 simulation to a wider public by making it available as a ROS [10] (Robot Operating System) package [11]. This includes the kinematic simulator and a simulation/visualization package in the ROS internal simulator Gazebo. Therefore nodes and topics for the ROS publish/subscribe system have been developed by NASA. Changing to ROS also meant in that case that RAPID was no longer used. ROS uses its own messaging system which also is based on publish/subscribe and works on distributed systems. To interface the system an API is available in different programming languages. The most commonly used ones are C++ and Python. The ESA Exoskeletons were not part of ROS and no nodes did exist.

Figure 8 Scheme of an optimal NASA – ESA remote connection

Figure 9 Controlling the R2 via the EXARM in ROS

For the purpose of a "proof of principle" a ROS bridge was programmed in Python which receives UDP messages from an exoskeleton and publishes them into the concerning ROS topic. For a quick evaluation, the X,Y,Z coordinates and the quaternion of the centre of the right palm of the exoskeleton are published for R2 into, "/r2_controller/right/pose_command".

Figure 9 shows the postural matching between the exoskeleton and R2 during control.Force information can also no be retrieved form the simulation in ROS, since no dynamics model is present. Gazebo however provides a contact model and in general the calculation of collisions forces could be added in the future.

VII. DISCUSSION

Several ESA Exoskeletons were used successfully to control a variety of R2 simulations. NASA Frameworks, such as around RAPID and Verve as well ROS have been used.

While such simulations allow to understand the interfaces and to develop suitable control strategies, for the full control of a real robotic asset of NASA, it would be required to understand better the state-changes and state-machines used for coordination tasks in the system. This information is needed, in order to safely link two independent systems (e.g. considering safe/active modes, alignment mode, etc.)

RAPID has clear advantages as an approach to a control framework, also due its smart usage of the DDS communication middleware. This allows to define system characteristics, such as Quality of Service, safe interfaces, auto discovery and, in principle, real-time capabilities. A framework, such as RAPID allows good reusability since certain commands are keep in a uniform manner. For example if all mobile robots understand the command move forward in the same way, there's a high reusability. Following features can clearly be seen as an advantage of the RAPID framework:

- A well thought-of set of defined Messages & Commands and Data-types
- The availability of DDS as middleware, enabling:
  - Clear modular definition of QoS
  - Real-time capabilities
  - Platform independence
  - RTI with tools for NAT/routing

However, despite the sophisticated concept of the "self-describing" interface, it still remains a challenge for the developers to interface different robotics systems if no full set of documentation is available. This means in practice, that despite the command interface published via RAPID, it is required to have knowledge about the robot, in terms of:

- Its general capabilities & functions (sensors types, number, location, sample rates; system hierarchy, sub-systems; etc.)
- Its Kinematics & Control concept
- Its Interfaces & Interface configurations
- Its Coordination layer and state-machines

Of course, a framework alone can't cover all of these points. Even if the full interface publishing is automated, analysed and linked/mapped via algorithms automatically, there is still behaviour that has to be programmed/coded. It seems unlikely that all aspects concerning system safety and security can be covered by one framework alone.In our work, we did not reach full RAPID compliance with the exoskeleton, since for that, we'd need to understand and implement the full set of messages and also to introduce our command configurations properly to other assets by publishing the *config files. This, however, was not yet the scope of this evaluation work but could be interesting to complete in a next step. The ROS framework is widely used throughout academic institutes, we see following advantages from using ROS:

- ROS can interface already many hardware components and robots
- ROS offers simulation, visualization, messaging services and a rather extensive algorithms library (including mapping, kinematics, computer vision, point clouds, etc.)
- ROS is open source, with a growing community

However, ROS does not provide any real-time capabilities since the system sits on top of a common Ubuntu with a response time in the range of hundreds of milliseconds. Even with up to date hardware (Quad-core Intel i7 CPU's) the Gazebo simulator was not running smoothly during our experiments. Moreover, ROS is not suitable for real-time distributed applications (even if extended with the OROCOS framework). We feel, that ROS is still targeted at programmers mainly and not at users, which is a big disadvantage in real operational contexts.

VIII. CONCLUSIONS & FUTURE WORK

During the described ESA / NASA collaboration, ESA has extended the NASA RAPID framework by a possibility to do commanding and messaging in real- time, through a Xenomai Linux based RAPID implementation. The accomplished work is a first step towards interfacing respective robots on operational level. The work on RAPID has provided valuable insights in the workings and needs for robotic frameworks and has put the involved teams in a close and fruitful collaboration. Findings from this work will influence the decision-taking process of new telerobotics standards for space applications. Therefore recently a CCSDS working-group has been established to create a standard for space telerobotics operations. Further testing is planned, which will involve:

- Controlling R2 directly with the X- Arm-2 exoskeleton (at NASA Johnson Space Center)
- Establishing a bilateral control between X- Arm-2 and R2 between the Telerobotics & Haptics Lab. at Noordwijk and NASA JSC
- Demonstrating the bilateral control of R2 on Station by the XR-1 (Exoskeleton Robot 1) exoskeleton currently in development at ESA for ISS (fully on board ISS)

At the moment, there is still no complete solution for a 'user oriented' framework that can handle distributed real-time operations between robotic assets. However, the current work on RAPID and ROS has provided valuable insights for the needs of a complete architecture. Some of the lessons learned during this experiment provide useful insights about:

- Required interface definitions
- Required interface propagation
- Suitability of communication layers
- Categorization (real-time and non-time critical)
- Required framework services

RAPID already tackles this and provides in combination with DDS a possible solution for the robotic control. It makes clear that certain data types and message structures have to be defined and made available to participating system. Another promising approach is the use of interface definition files and methods of automatic code generation that the Telerobotics & Haptics Laboratory is currently following [12]. It is the goal at ESA to create a complete distributed real-time framework software that is truly user centric.

REFERENCES

[1] Schiele, A. (2011) METERON – validating orbit-to- ground telerobotic operations technologies, 11th symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)

[2] Schiele A., (invited), 'Case Study: The ergonomic EXARM exoskeleton', in: Wearable Robots: Biomechatronic Exoskeletons, Edited by J.L. Pons, John Wiley & Sons, Ltd., (2008), pp. 248 – 255

[3] Schiele A., Hirzinger, G. (2011) A new generation of ergonomic exoskeletons - The high-performance X- Arm-2 for space robotics telepresence, International Conference on Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ, pp 2158 – 2165

[4] Diftler, M.A., Mehling, J.S.; Abdallah, M.E.; Radford, N.A.; Bridgwater, L.B.; Sanders, A.M.; Askew, R.S.; Linn, D.M.; Yamokoski, J.D.; Permenter, F.A.; Hargrave, B.K.; Piatt, R.; Savely, R.T.; Ambrose, R.O. (2011) Robonaut 2 - The first humanoid robot in space, 2011 IEEE International Conference on Robotics and Automation (ICRA), pp. 2178-2183

[5] Torres, J., Allen, M., Hirsh, R., Wallick, M.N. (2009) RAPID: Collaboration results from three NASA centers in commanding/monitoring lunar assets, IEEE Aerospace conference, pp. 1 – 11

[6] Pardo Castellote, G. (2003) OMG Data Distribution Service: architectural overview, Distributed Computing Systems Workshop

Proceedings 23rd International Conference on Digital Object Identifier, pp. 200 - 206

[7] R2 picture
http://www.nasa.gov/topics/technology/features/robona ut_photos.html

[8] VERVE:
https://ti.arc.nasa.gov/blog/irg/?p=30

[9] Kimmer, S., Rebelo, J., Schiele, A. (2013) SPANviewer - A Visualization Tool for Advanced Robotics Applications, 12th symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)

[10] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. ICRA workshop on open source software, (Vol. 3, No. 3.2)

[11] ROS-R2-Sim:
http://ros.org/wiki/nasa_r2_simulator

[12] Schiele, A., Krueger T., Aiple, M., Kimmer, S. (2013) Initial Thoughts About a Model-based & Code- Generation Framework for Robotic Control, 12th symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)