8th International Conference on Modelling, Identification and Control (ICMIC-2016)
Algiers, Algeria- November 15-17, 2016

A Survey of Development Frameworks for Robotics

Abdelfetah Hentout[1], Abderraouf Maoudj[2] and Brahim Bouzouia[3]
[1,2,3]Centre de Développement des Technologies Avancées (CDTA)
Division Productique et Robotique (DPR)
BP 17, Baba Hassen, Algiers 16303, Algeria
[2]University of Sciences and Technology Houari Boumediene (USTHB)
Laboratory of Robotics, Parallelism and Electro-energy (LRPE)
BP32 El Alia, Bab Ezzouar, Algiers 16111, Algeria
Email: [1]ahentout@cdta.dz, [2]amaoudj@cdta.dz, [3]bbouzouia@cdta.dz

***Abstract*—The growing interest in robotics has originated, in these recent years, the development of a high variety of intelligent and autonomous robots, and a great number of software frameworks for robotics. This paper surveys some representative of the existing of these development frameworks. In addition, it describes and compares their principal features and limitations. The paper provides also information on some of the existing robots supported by the studied frameworks all over the world. Finally, the paper discusses the main requirements that a robotics framework must satisfy while being easy to understand and to use.**

***Index Terms*—Robotics, Development frameworks, Requirements for robotics frameworks, Survey.**

I. Introduction

Robotics is the set of sciences and technologies that deals with the development of autonomous and intelligent robots. It is based on various disciplines and technologies such as electronics, mechanics, mechatronics, computer science, wireless communication and automation [1] [2].

Over the years, research in robotics has moved from a very limited capability robot(s) to very sophisticated robot(s). In addition, current robots are equipped with heterogeneous interconnected hardware components, a large variety of sensors (cameras, encoders, effort sensors, laser and ultrasonic sensors, GPS, temperature and humidity sensors, etc.) and actuators (DC servomotor, pneumatic motor, gripper, etc.). These components are controlled by software modules, developed by different manufacturers using various programming languages and tools. Software modules are also required to process sensors information and control actuators for performing vision processing, mapping, navigation, trajectory planning, user interaction, etc. [1].

Depending on their types, current robots must be able to accomplish new applications inside their environments. *Field robots* carry out tasks in hazardous, harsh or dangerous environments or inaccessible terrains to save and preserve human lives. *Service robots* perform human-assistance for the elderly or physical-disability persons, medical or surgical tasks, etc. to support human daily life in unstructured dynamic environments. *Industrial robots* accomplish tedious, painful and dirty tasks in manufacturing environments. Current researches in robotics aim to endow robots more skills. Consequently, next generation robots should be modular for easy and rapid implementation, flexible, customizable, self-configuring, and able to interact with other systems such as senor networks, enterprise information systems, etc. [1].

Industrial robots are extensively used in industrial manufacturing environments. However, few of *service robots* became commercially available, and *field robots* are not widely used [3]. This is due, in large part, to the long development process times and high costs for such systems that are generally designed for specific tasks (ad-hoc solutions) [4]. Furthermore, each manufacturer develops and uses its own operating systems, middlewares and development tools (low-level communication libraries, etc.), which further complicate the development process for such robots. The process of developing and programming current robots is not a simple task for engineers. It requires, per contra, knowledge, effort and expertise. To support developing new intelligent and autonomous robots, numerous common open source and commercial robotics frameworks (operating systems, middlewares, programming languages, etc.) have been developed all over the world.

Software frameworks are universal and reusable software environments that attempt to provide integral solutions through a set of generic tools, off-the-shelf libraries, application programming interfaces (APIs) with algorithms and controllers useful to create general-purpose robotic systems [5]. Their main aims consists of (*i*) assisting developers in the design, implementation, debugging and execution of robots [6], (*ii*) significantly decreasing the development time and required resources, (*iii*) making developing robots easier, faster and cheaper [7], and finally, (*iv*) making reuse and sharing of existing software [8]. Some robotic development frameworks include off-the-shelf sub-programs for usual robotic tasks such as trajectory generation, vision processing, mapping, localization, remote control, etc.

The first robotic applications have been developed using standard real time operating systems such as *VxWorks*, *OS9*, *RTX*, etc. Currently, we note an expanding number of highly competitive development frameworks for robotics. Kramer and Scheutz [9] investigated nine open-source frameworks. They proposed a conceptual evaluation framework based on four categories of criteria (specification, platform support, infrastructure and implementation). Michal and Etzkorn [6] compared

978-0-9567157-6-0

two well-known frameworks (*i*) *Player/Stage/Gazebo*, which is free widely-used framework and (*ii*) *Microsoft Robotics Developer Studio* which is its commercial association. Biggs and MacDonald [10] focused more on robot programming environments characteristics and evaluations. Mohamed and colleagues [1] explored different robotic middleware projects and provided a discussion of the existing approaches. Colon [8] described and compared *CoRoBA* and *Microsoft Robotics Developer Studio* based on the identified requirements. Iigo-Blasco et al. [5] focused in their study on multi-agent robotic frameworks where some of them were enumerated, analyzed and compared. The rest of this paper is structured as follows. The second section enumerates and analyzes six of the most well-known development frameworks of robotic software. The third section explains and discusses the most significant characteristics and requirements that a robotics framework must include. The fourth section provides a comparative study between the six development frameworks. Finally, the fifth section concludes the paper.

II. Development frameworks for robotics

This section is the focus of this study. It describes some illustrative of the existing software frameworks proposed for the development of robotic applications:

A. Microsoft ® Robotics Developer Studio (Microsoft RDS)

Microsoft RDS is a commercial Windows-based (*MSDN license*) programming environment system developed by *Microsoft* for building robotics applications. It can be used by both professional and non-professional developers as well as hobbyists [11]. *Microsoft RDS* includes, also, the *3D Visual Simulation Environment* (*VSE*), which provides useful data to the user [6] [12]. This framework is only compatible with Windows. On the other hand, it supports various programming languages including *C#*, *VB.NET* and *Iron Python*.

Various robots are compatible with *Microsoft RDS* such as *Eddie* Robot Platform [13], Japanese ZMP' *e-nuvo WALK* humanoid robot [14], *Pioneer DX* and *Pioneer AT*, *CoroWare CoroBot* and *Explorer*, *Nao* of Aldebaran Robotics [15], etc.

B. Robot Operating System (ROS)

ROS is a completely open source robot system (*creative commons attribution 3.0 license*) developed by *Willow Garage* (robotics incubator) in collaboration with the *University of Stanford* for writing robots software. It provides collections of hardware abstraction, device drivers, libraries, visualizers, message-passing, package management and tools that aim to simplify the task of creating complex and robust robotic applications across a wide variety of robotic platforms. These libraries and tools are enriched by the robotics community all over the world [16]. *ROS* runs on Unix-based systems (*Linux*, *OSX*, etc.). Moreover, it supports different programming languages such as *C++*, *Python*, *Java*, *Matlab* and *Lisp*. *ROS* uses *rviz* (2D visualizer), *Stage* (2D simulator) and *Gazebo* (3D simulator).

ROS supports many robots such as Willow Garage *PR2*, Thecorpora' *Qbo*, *Lego Mindstorms NXT*, Fraunhofer institute *Care-O-bot* mobile manipulator, *Nao* of Aldebaran Robotics, Kawada *HRP2-V* [17], Shadow's *robotic hand* [18], etc.

C. Open Robot Control Software (OROCOS)

OROCOS is a free software (*GNU/GPL license*) composed of a set of open source libraries for real-time control of industrial robots. It is developed by the *European Robotics Research Network* (*EURON*) and is implemented mainly by the *Katholieke Universiteit Leuven* (Belgium). *OROCOS* is delivered with its own optimized run-time environment for real-time applications. It consists of a set of reusable components including *RTT* (The *Real-Time Toolkit*), *KDL* (*Kinematics and Dynamics Library*), *BFL* (*Bayesian Filtering Library*) and robotic equipment drivers. However, *OROCOS* includes neither a graphical environment for drag-and-drop development nor a simulation environment [19] [20].

OROCOS runs on *Linux*, *Windows* and *MacOS* operating systems and supports real-time kernels such as *RTAI* [21] and *Xenomai* [5]. It supports various programming language such as *C/C++* (as the main language), *Tcl/Tk* for graphical interfaces [22] and other languages such as *Python* and *Simulink OROCOS Scripting Language* [5].

OROCOS supports various industrial robots such as *Wam* robot of *Barret Technology Inc.* [23], NASA' *Motoman SIA10D* industrial robot [24], autonomous cars in Berlin highways [25], *KUKA lightweight* robot (*LWR*) [26], etc.

D. Player/Stage

*Player/Stag*e is an open source (*GNU/GPL license*) hardware-abstraction layer (middleware platform) developed at the *Robotics Research Lab* of the *University of Southern California* (USA). *Player/Stage* is currently supported by *Source Forge*. This middleware has two major components (*i*) *Player* and (*ii*) *Stage*.

Player provides an interface for managing robot devices and services (reading sensors) and actuators (writing on actuators) on the network and can support multiple devices into a single interface. It also includes data communication and control programs. The communication interfaces are based on a client/server architecture that uses *TCP/IP* sockets. *Stage* is a plugin to *Player*, which simulates a population of robots, sensors and objects in 2D environments [12] [6]. It provides different models for robot sensors and actuators (sonar and infrared rangers, scanning laser rangefinder, color-blob tracking, fiducial tracking, bumpers, grippers, etc.) and mobile bases (with odometric localization, global localization, etc.) [27].

This robotics framework can run on both regular and embedded *Linux*, *Solaris*, and *BSD Unix*. In addition, it supports multiple programming languages such as *C/C++*, *Tcl*, *Java* and *Python*.

Player/Stage supports many robotic systems such as *Summit RB-Rtk-01* off-road robot of Robotnik, *mini-ATRV* mobile robot of iRobot, *RWI B21* robot [28], *Koala* robot [29], *Pioneer 3-DX* [30], etc.

E. Universal Robotic Body Interface (URBI)

URBI consists of an open-source (*BSD license*) software built by *Gostai* (*Aldebaran Robotics group*). It is based on a *TCP/IP* client/server side type architecture. The server side handles the low-level control of the robot such as robots joints and sensors. The client side is used to control the robot from a higher-level, that is, the sending of higher-level actions such as walk method, etc. [31]. *URBI* includes also *Webots*, which is its official 3D robot simulator.

URBI is multi-platform able to run on *Windows*, *Linux* and *MacOS*. Moreover, it includes *C++* component libraries and is provided with its own *Urbiscript* parallel script language, which runs on top the *Urbi Virtual Machine* (*UVM*). *URBI* also provides libraries collectively called *libUrbi* for different languages such as *Java*, *Matlab*, *Lisp* and *C++* [32].

URBI supports different robots, such as *Jazz* of Gostai, *Aibo* from Sony, *HRP-2* of Kawada, *Nao* of Aldebaran Robotics, *Pioneer P3-DX* mobile robot [33], etc.

F. Generator of Modules (GenoM)

GenoM is an open-source (*BSD license*) tool to design real-time software architectures for autonomous and terrestrial mobile robots [34]. It provides the programmer with some built-in system primitives such as inter-process communication, data-pool capabilities with external access, a state machine for algorithms and automatic client code generation [35]. This robotics framework allows encapsulating the operational functions on independent modules that manage their execution. The functions are dynamically started, interrupted or (re)parameterized upon asynchronous requests sent to the modules (control of sensors and actuators, servo-controls, monitoring, data processing, trajectory computation, etc.). A final reply that qualifies how the service has been executed is associated to every request [34].

GenoM generates software modules running under various operating systems such as *Linux*, *Unix*, *VxWorks*, *BSD* and *Xenomai*. However, it supports only *C/C++* and *Tcl/Tk* programming languages.

GenoM modules are present on various robots such as *Rackham* autonomous interactive mobile robot [36], *Jido* mobile manipulator [37], *HRP-2* humanoid robot [38], the two rough terrain *ATRV* mobile rovers *Dala* and *Mana* [39], *RobuCar* of the CDTA [40], etc.

III. Requirements for robotics development frameworks

It is not efficient enough to evaluate development frameworks for robotics on traditional metrics, such as speed and responsiveness. The best measure of the effectiveness of such a framework is how much it is used amongst robotics researchers [41]. Unfortunately, this measure is not possible. As pointed out in [1] and [9], it is clear that we cannot provide undisputed set of distinct criteria that are able to distinguish between the different frameworks and, consequently, offering a solid basis for comparisons. However, in this section we try to define a set of the most important requirements that a robotics framework must satisfy while being easy to understand and to use.

A. License

The license is a crucial factor for the success of development frameworks [5]. It is preferable that robotics frameworks be open source and available free of charge (under *BSD* or *GNU/GPL license*) to have large distribution amongst robotics community.

B. Installation

A software framework for robotics is generally installed manually on each embedded PC of the multi-robot team involved in the application; this method is time consuming and error prone [8]. Therefore, development engineers need tools that facilitate software installation (wizards, etc.), tools for supporting distributed installation, integration tools, etc.

C. Operating systems

To be more usable and adopted by the roboticists, a robotics framework must be multi-platform able to run on various operating systems such as *Windows*, *Linux*, *MacOS*, *Solaris*, *Android*, *iOS*, etc.

D. Programming languages

A robotics framework must offer the possibility to use several programming languages such as *C/C++*, *Java*, *Python*, *Tcl/Tk*, etc. [8]. It is also preferable that the framework provides high-level graphical programming languages.

E. Simplifying and enhancing the development process

The development of robotic applications is not easy. Consequently, the framework should simplify the development process by providing high-level abstractions, modular design mechanisms, component-based development, off-the-shelf software modules, etc. [1].

F. Usability (Ease of use)

The usability of a software framework for robotics is defined as its degree fitting to the characteristics of their users.

G. Interoperability and transparency

Heterogeneous hardware components developed by different manufacturers could coexist inside a single robotic system. Moreover, software modules, designed and implemented by different programmers, are required to process sensors information and control actuators. Thus, the framework must provide an abstraction layer assuring that these different modules work harmoniously all together, while hiding their complexity and heterogeneity.

H. Communication

Robotic frameworks should integrate object-oriented communication libraries offering synchronous and asynchronous mechanisms and supporting different communication models (master/slave, client/server, peer-to-peer, etc.) [8].

I. Efficient utilization of robot components and available resources

In order to be able to carry out real-time processing-intensive tasks (vision processing, mapping, navigation, etc.), robots are generally equipped with embedded PCs with single or multiple microprocessors, one or more interconnection networks and several other resources [1]. Accordingly, a robotics framework should assure efficient utilization of all these components and resources.

J. Multi-robot control

When several robots need to cooperate and coordinate their operations to carry out assigned tasks, the robotics framework should offer services and possibilities to manage the multi-robot team efficiently.

K. Multi-user control

In case of some specific applications, the control of the robot(s) could be shared between several operators or between operator(s) and robot(s) at the same time (traded control, shared control, etc.). Thus, a robotics framework should provide such possibility.

L. Graphical user interface

It is preferable that the robotics framework provides user-friendly (well-designed, well-ergonomic, simple, easy-to-understand and easy-to-use) graphical interface. This should be independent of the robot, the application, and the operating system [1]. It is also desirable that the framework provides drag-and-drop components, graphical programming languages, etc.

M. Simulation capability

A robotics framework must offer 2D/3D simulators in order to test and debug developed modules, strategies and applications without accessing the real robotic system (single- or multi-robot system).

N. Reusability

The software framework for robotics must provide multi-platform off-the-shelf components and software modules (motion control, vision processing, navigation strategies, localization approaches, etc.) that have been developed and tested by the roboticists all over the world. In addition, once improved and approved, these components have to be published and shared on the web to avoid to be rewritten several times.

O. Documentation

A development framework for robotics should include detailed documentations, tutorials and explained examples for all programmers and developers levels (novice, competent, proficient, expert, etc.).

IV. Comparison between the studied development frameworks for robotics

The comparative study is summarized in three tables (1, 2 and 3). For each group of the requirements identified in the previous section, a table of results is shown. The rows represent the selected software frameworks; the columns provide the described features/requirements.

A. Comparison between the studied software frameworks

Table 1 reviews the general aspects for all the development frameworks for robotics considered in this work.

Tables 2 and 3 present a comparison of some of the available characteristics, detailed in section III, for all the previous development frameworks for robotics [22] [43] [44] [45]. We can see that *Microsoft RDS*, *ROS* and *Player/Stage* are the best frameworks over all the other ones considered in this study:

- The installation of *Microsoft RDS* is very simple. In addition, it has a very user-friendly graphical interface. This framework includes detailed samples, tutorials and documentation. Furthermore, it integrates a very powerful 3D simulator (*VSE*). However, the major weaknesses of *Microsoft RDS* is that it is not free; it is a commercial framework released under *MSDN license* except for the express edition. It is also only compatible with *Windows* and does not support any other operating system.
- *Player/Stage* is released with *GNU/GPL license*, compatible with most of the operating systems and programming languages. It is very adapted for multi-robot team control. *Player/Stage* provides a 2D simulator (*Stage*) and can integrate another 3D simulator (*Gazebo*). In this case, this framework is called *Player/Stage/Gazebo* instead of *Player/Stage* only. However, its installation is not simple. Moreover, it does not have a simple graphical interface. Finally, *Player/Stage* has poor tutorials and documentation.
- *ROS* is completely open source released under *BSD license*. Furthermore, it provides very rich documentation and tutorials. As *Player/Stage*, *ROS* integrates the same 2D (*Stage*) and 3D (*Gazebo*) simulators. However, the most important disadvantage with this framework is that it is very complex; it is designed to be used by more-experienced programmers to produce solutions for more-complex robotic problems [6] [12]. In addition, its installation is more difficult compared with the other frameworks. Finally, unlike *Microsoft RDS*, *ROS* is compatible only with *Unix*-based systems and does not support other operating systems (especially *Windows*).

B. Discussions

We should note that all the considered development frameworks for robotics are different. In addition, none of them fits all the fifteen requirements detailed in the previous section. However, each of the frameworks provides its own specificities and tries to bring up its benefits that make it the most suited for a specific applications field. For example, *Player/Stage* is most suitable for multi-robot team control, *ROS* for service

TABLE I
GENERAL ASPECTS OF THE DEVELOPMENT FRAMEWORKS FOR ROBOTICS CONSIDERED IN THIS WORK.

Robotics Framework	Originating institution	First version released on	Latest version		More Information
			Version	Released on	
Microsoft RDS	Microsoft (USA)	18^{th} December 2006	4.0.261.0	6^{th} March 2012	[11]
ROS	Willow Garage in collaboration with the University of Stanford (USA)	1^{st} November 2007	ROS Indigo Igloo (8^{th})	22^{nd} July 2014	[16]
OROCOS	K. U. Leuven (Belgium) LAAS (France) and KTH Stockholm (Sweden)	2000	Toolchain: 2.6.0 KDL: 1.0.2	3^{rd} December 2012	[19]
Player/Stage	University of Southern California Robotics Research Lab (USA), then Source Forge	1999	Player: 3.0.2 Stage: 4.1.0	28th June 2010 5th July 2013	[27]
URBI	Gostai (France)	2003	2.7.4	17th November 2011	[42]
GenoM	LAAS (France)	1996	2.10	24 October 2012	[34]

TABLE II
PRINCIPLE FEATURES OF THE DEVELOPMENT FRAMEWORKS FOR ROBOTICS STUDIED IN THIS PAPER (1).

Robotics Framework	1. License	2. Installation	3. Operating system	4. Programming language	5. Simplify and enhance	6. Usability	7. Interoperability and transparency
Microsoft RDS	MSDN	Very simple	Windows	C#, VB.Net, JScript, Iron Python	Very good	Simple	Support
ROS	BSD	Difficult	Linux, OSX, Windows (limited and experimental)	C++, Java, Python, Matlab, Lisp, Octave	Very good	Very complex	High support
OROCOS	GNU/GPL	Difficult	Linux, Windows, Mac OS, RTAI, Xenomai	C/C++, Tcl/Tk, Python, Simulink Orocos Scripting Language	Good	Very complex	Support
Player/Stage	GNU/GPL	Difficult	Linux, Solaris, BSD Unix, Mac OS	C/C++, Java, Tcl, Python	Good	Complex	High support
URBI	BSD (Parts of URBI)	Difficult	Windows, Linux, Mac OS	C++, Java, Matlab, Lisp, Urbiscript	Good	Complex	?
GenoM	BSD	Difficult	Linux, Unix, VxWorks, BSD, Xenomai	C/C++, Tcl/Tk	Good	Complex	Support

robots control, *OROCOS* for real-time control of industrial robot, *URBI* for humanoid and android robots, etc. In addition, we found that currently several universities and research laboratories all over the world use *ROS* for developing their robotic applications. Furthermore, it was found that *C/C++* and *Java* are the most used programming languages for developing robotic applications, since they offer access to devices, sensors and actuators at low level, while still offering a sufficient level of abstraction to create complex systems [5].

As it can be noted, some of the principle features of the studied frameworks could not be identified (Multi-user control, etc.). This is due mainly to the lack of the available information and to the difficulty to test all of these requirements on all the frameworks.

V. CONCLUSION

This paper surveyed six well-known frameworks for developing robotic applications. These software systems provide design tools and a large selection of implemented powerful components. For each framework, few recent and typical examples of robotic platforms have been mentioned. We also highlighted fifteen main requirements facing researchers and developers in the field of robotics frameworks. The end of the paper made comparison between the selected frameworks based on the identified requirements.

Throughout this study, we found that many robotic systems only require a sub-set of the functionalities and features of a development framework; most of them do not need all the functionalities together. Finally, we arrived at a general conclusion that it is currently very difficult (even impossible) to develop a framework able to solve all problems and issues of robotics.

REFERENCES

[1] N. Mohamed, J. Al-Jaroodi and I. Jawhar, *Middleware for robotics: A survey*, IEEE Conference on Robotics, Automation and Mechatronics (RAM2008), pp. 736-742, Chengdu, China, 21-24 September 2008.
[2] Ch. Nicot, L. Lacheny, D. Atlan and L. Bernasconi, *La robotique de service en Midi-Pyrnes*, 2012.
[3] H. Tanaka, M. Yoshikawa, E. Oyama, Y. Wakita and Y. Matsumoto, *Development of assistive robots using international classification of functioning, disability, and health: concept, applications, and issues*, Hindawi Journal of Robotics, Vol. 2013, pp. 1-12, 2013.
[4] S. Farritor, S. Dubowsky, N. Rutman and J. Cole, *A Systems-Level Modular Design Approach to Field Robotics*, The International Conference on Robotics and Automation (ICRA96), pp. 2890-2895, USA, April 1996.
[5] P. Iigo-Blasco, F. Diaz-del-Rio, C. Romero-Ternero, D. Cagigas-Muiz and S. Vicente-Diaz, *Robotics software frameworks for multi-agent robotic systems development*, Robotics and Autonomous Systems, 60, pp. 803-821, 2012.
[6] D. S. Michal and L. Etzkorn, *Comparison of Player/Stage/Gazebo and Microsoft Robotics Developer Studio*, The 49th ACM Southeast Conference, pp. 60-66, Kennesaw, GA, USA, 24-26 March 2011.
[7] K. Jensen, M. Larsen, S. H. Nielsen, L. B. Larsen, K. S. Olsen and R. N. Jrgensen, *Towards an Open Software Platform for Field Robots in Precision Agriculture*, Robotics, 3(2), pp. 207-234, 2014.
[8] E. Colon, *Robotics Frameworks*, Introduction to Modern Robotics II, iConcept Press, ISBN 978-1-463789-442, 2014.
[9] J. Kramer and M. Scheutz, *Development Environments for Mobile Autonomous Robots: A Survey*, Autonomous Robots, pp. 101-132, 2007.

TABLE III
PRINCIPLE FEATURES OF THE DEVELOPMENT FRAMEWORKS FOR ROBOTICS STUDIED IN THIS PAPER (2).

Robotics Framework	8. Communication	9. Efficient use of components	10. Multi-robot control	11. Multi-user control	12. Graphical user interface	13. Simulation capability	14. Reusability	15. Documentation
Microsoft RDS	DSSP	Yes	Support	?	Very user-friendly interface	3D: VSE	No support	Very rich
ROS	Topics, Properties, Services	Yes	Support	Support	No	2D: Stage, rviz (visualizer) 3D: Gazebo	High support	Very rich
OROCOS	Ports, Services, Events, Properties, CORBA	Yes	?	?	No	No	Medium support	Poor
Player/Stage	Client/Server	Yes	High support	?	User-friendly interface	2D: Stage 3D: Gazebo	Support	Poor
URBI	Client/Server	Yes	?	?	Very user-friendly interface	3D: Webots	?	Very poor
GenoM	Asynchronous event/Poster	Yes	?	?	No	No	Support	Rich

[10] G. Biggs and B. MacDonald, *A Survey of Robot Programming Systems*, The Australasian Conference on Robotics and Automation (CSIRO), Brisbane, Australia, 1-3 December 2003.
[11] http://www.microsoft.com/en-us/download/details.aspx?id=29081
[12] J. Kerr and K. Nickels, *Robot Operating Systems: Bridging the Gap between Human and Robot*, The 44th South-Eastern Symposium on System Theory, University of North Florida, USA, 11-13 March 2012.
[13] http://www.parallax.com/product/28992
[14] http://robotzeitgeist.com/tag/microsoft-robotics-studio
[15] http://emereo.net/success/microsoft-robotics-developer-studio-supported-robots/
[16] http://www.ros.org/
[17] https://www.willowgarage.com/pages/software/ros-platform
[18] http://singularityhub.com/2010/11/08/publish-monday-robot-operating-system-celebrates-3rd-birthday-with-exponential-growth-video/
[19] http://www.orocos.org/
[20] http://people.mech.kuleuven.be/ orocos/pub/documentation/rtt/v1.12.x/doc-xml/orocos-overview.html
[21] P. Mantegazza, *DIAPM RTAI for linux: Whys, whats and hows*, Real Time Linux Workshop, Vienna University of Technology, Austria, 1999.
[22] H. Bruyninckx, *Open Robot Control Software: the OROCOS project*, The International Conference on Robotics and Automation (ICRA'01), Vol. 3, Seoul, Korea, pp. 2523-2528, 21-26 May 2001.
[23] D. Ioris, W. F. Lages and D. C. Santini, *Integrating the OROCOS framework and the Barrett Wam robot*, The 5th Workshop on Applied Robotics ad Automation, Bauru, Brazil, pp. 1-8, 2012.
[24] http://www.orocos.org/orocos/using-orocos-visually-track-free-floating-target-micro-gravity
[25] http://www.orocos.org/node/632
[26] G. Schreiber, A. Stemmer and R. Bischoff, *The Fast Research Interface for the KUKA Lightweight Robot*, The International Conference on Robotics and Automation (ICRA2010), pp. 15-21, USA, 3-8 May 2010.
[27] http://playerstage.sourceforge.net/
[28] R. B. Rusu, A. Maldonado, M. Beetz and B. Gerkey, *Extending Player/Stage/Gazebo towards Cognitive Robots Acting in Ubiquitous Sensor-equipped Environments*, The International Conference on Robotics and Automation (ICRA2007), Workshop for Networked Robot Systems, Roma, Italy, April 2007.
[29] M. Anderson, L. Thaete and N. Wiegand, *Player/Stage: A Unifying Paradigm to Improve Robotics Education Delivery*, Workshop on research in robots for education at robotics: science and systems conference, 2007.
[30] F. Espinosa, M. Salazar, F. Valds and A. Bocos, *Communication architecture based on Player/Stage and sockets for cooperative guidance of robotic units*, The 16th Mediterranean Conference on Control and Automation, pp. 1423-1428, Ajaccio, France, 25-27 June 2008.
[31] D. A. Alexander, *Linking an HTN Planner to a Universal Robot Controller for High-Level Activity Control*, M.Sc. thesis in Artificial Intelligence, School of Informatics, University of Edinburgh, 2006.
[32] J.-Ch. Baillie, A. Demaille, Qu. Hocquet, M. Nottale and S. Tardieu, *The Urbi Universal Platform for Robotics*, The International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR 2008), Venice, Italy, pp. 580-591, 3-4 November 2008.
[33] J.-Ch. Baillie, A. Demaille, G. Duceux, D. Filliat, Qu. Hocquet and M. Nottale, *Software architecture for an exploration robot based on Urbi*, The 6th National Conference on Control Architectures of Robots, Grenoble, France, pp. 1-12, May 2011.
[34] https://www.openrobots.org/wiki/genom
[35] A. Mallet and M. Herrb, *Recent developments of the GenoM robotic component generator*, The 6th National Conference on Control Architectures of Robots, Grenoble, France, May 2011.
[36] G. Bailly, L. Brthes, R. Chatila, A. Clodic, J. Crowleyo, P. Dans, F. Elisei, S. Fleury, M. Herrb, F. Lerasle, P. Menezes and R. Alami, *HR+: Towards an interactive autonomous robot*, 3me Journes du Programme ROBEA (ROBEA'2005), France, pp. 39-45, 29-31 Mars 2005.
[37] A. Ceballos, L. De Silva, M. Herrb, F. Ingrand, A. Mallet, A. Medina and M. Prieto, *GenoM as a Robotics Framework for Planetary Rover Surface Operations*, The 11th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA 2011), The Netherlands, April 2011.
[38] E. Yoshida, A. Mallet, F. Lamiraux, O. Kanoun, O. Stasse, M. Poirier, P-F. Dominey, J-P. Laumond and K. Yokoi, *Give me the Purple Ball He said to HRP-2 N.14*, The 7th IEEE-RAS International Conference on Humanoid Robots, pp. 89-95, USA, 29 November-1 December 2007.
[39] O. Lefebvre, F. Lamiraux, C. Pradalier and Th. Fraichard, *Obstacles Avoidance for Car-Like Robots Integration and Experimentation on Two Robots*, The International Conference on Robotics and Automation (ICRA'04), Vol. 5, pp. 4277-4282, USA, 18-22 April 2004.
[40] N. Ouadah, O. Azouaoui and B. Khiter, *Modular and Reusable Embedded Control Architecture: Case of a Car-like Robot*, The International Conference on Robotics and Automation (ICRA 2013), Workshop SDIR, Karlsruhe, Germany, 06-11 May 2013.
[41] W. D. Smart, *Is a Common Middleware for Robotics Possible?*, The International Conference on Intelligent Robots and Systems (IROS 2007), Workshop on Measures and Procedures for the Evaluation of Robot Architectures and Middleware, USA, 29 October-02 November 2007.
[42] http://www.gostai.com/products/jazz/urbi/index.html
[43] G. Markou and I. Refanidis, *MSRS: Critique on its usability via a path planning algorithm implementation*, The 5th IFIP Conference on Artificial Intelligence Applications and Innovations (AIAI09), pp. 311-320, Greece, 23-25 April 2009.
[44] http://playerstage.sourceforge.net/doc/Player-2.0.0/player/group__utils.html
[45] http://linuxdevices.com/articles/AT9631072539.html