**2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)
May 14-15, Espinho, Portugal**

MYRABot+: A Feasible robotic system for interaction challenges

Francisco Martín Rico
Technical School of Telecommunication Engineering
Rey Juan Carlos University
Fuenlabrada, Madrid (Spain)
Email: francisco.rico@urjc.es

Francisco J. Rodríguez Lera, Vicente Matellán Olivera
School of Industrial and Computer Engineering
University of León
León (Spain)
Email: {francisco.lera, vicente.matellan}@unileon.es

Abstract—**This paper describes the development of a low cost robotic platform named MYRABot+, which is able to make the required tasks in order to assist a human in a domestic environment. We also believe that the best way to measure its performance and validate its behavior is taking part in a robotic challenge. This robotic platform explicitly uses low cost components, making it an accessible platform in monetary terms. In addition, we have designed a component-oriented software architecture that allows an easy implementation of simple HRI tasks. We demonstrate the feasibility of this proposal by taking part in the competition RoCKIn@home. There we show the basic abilities needed to work in a house environment, such as self-localization, navigation, human dialog, and object manipulation. Our goal is to be able to evaluate and to compare it with other robotic platforms.**

I. Introduction

Competition is a well known catalyst for innovation. It has been used in the robotic field for a long time, for instance, AAAI competitions [21] are decades old. Lately, competitions are gaining even more attention within the robotics research community.

Another positive aspect of competitions is its attractiveness for general audiences. This point has particularly well exploited by The RoboCup Federation that organizes the RoboCup competition since 1997.

Since then, they have been organizing challenges in different categories and performing Academic Conferences. They are not only focused on robotic soccer, but organize competitions in other domains: @work competition, a sponsored category with the aim of developing new solutions for industrial environments; @home RoboCupRescue league and RoboCup Junior league or @home competition focused on the development of new solutions in assistance and personal robotics.

Many other competitions are organized every year, locally, regionally or globally. Some examples are DARPA challenge, World Robot Olympiad, Robofest, AAAI Grand Challenge, FIRST competition or RoCKIn Challenge.

On this paper we will focus on RoCKIn, which is a robotic event organized by a consortium funded mainly by an EU project. This event is made up by robot competitions, forums, educational camps and workshops. The main target of this challenge is not only the competition, but to define a test-bed able to measure robots capabilities in two well defined environments: RoCKIn@home for service robotics and RoCKIn@work for industrial robotics.

Most robots that enter in this kind of competitions are state of the art platforms with expensive sensors and actuators, as, for instance, REEM robot made by PAL; Amigo robot from Tech United, or Care-o-Bot from Fraunhofer IPA.

However, we think that economic cost has to be taken into account. In this way, we have built a low-cost robotic platform. The current version of our robot is named MYRABot+ (figure 1). It is based on Turtlebot platform, but we have improved its hardware for @home competitions. We also have developed a software architecture for platform control based on ROS.

During the last decade, many other affordable robotic platforms have been built. Some recent affordable examples are, for instance, πrobot [17], Maxwell robot developed by Michael Ferguson, software engineer from Willow Garage, or the EL-E: An Assistive Robot from George Tech Healthcare Robotics Lab [16]. All these platforms have been designed for HRI. All of them includes a positionable arm, stereo camera mounted on top of the robot and anthropomorphic appearance.

MYRABot+ has been built on top of the Turtlebot. Turtlebot is one of the most successful affordable platforms [15]. It was initially designed at Willow Garage lab and developed by Tully Foote and Melonee Wise in 2011.

Major contributions of MYRABot+ when compared with other platforms as previously mentioned,there are four: first, the perception system that is based on MS-Kinect which reduces the cost from some thousand euros to less than 150. Second one is possibility of including a low cost arm. Third, the use of ROS (Robotic Operative System) that simplified the way that the technical people develop solutions for robots, and fourth, the frame oriented chassis that let us improve the human-robot interaction capabilities, because the original Turtlebot robot is not an appropriated solution for HRI as we saw in previous experiences [18].

Another major contribution of this research is a component-oriented architecture. It can be implemented on several open source frameworks, such as ROS [9] by Willow Garage, RoboComp [10] by Universidad de Extremadura, CARMEN [11] by Carnegie Mellon, and Miro [12] by University of Ulm. All of them use some component-based approach to organize robotic software using ICE, IPC, and CORBA, respectively, to communicate their modules.

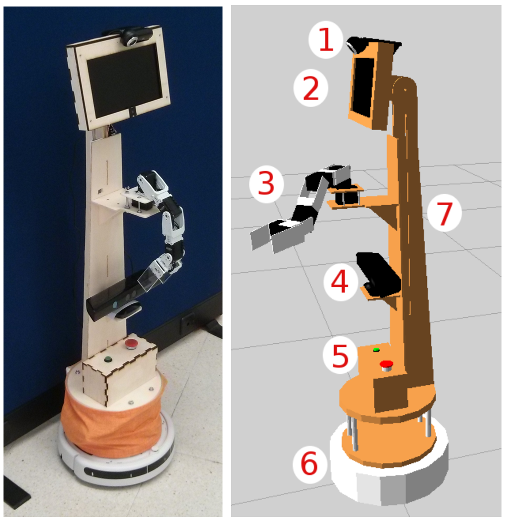


Fig. 1. MYRABot+ Robot Models (real and Rviz)

To sum up, the aim of the research described in this paper is to evaluate if it is feasible to create a low cost platform that could be able to join international competitions. In particular, we have chosen RoCKIn, in its @home flavor, to test it.

The rest of the paper is organized as follows: section II discusses MYRABot+ platform. Section III summarized the software architecture used in the platform. Section IV presents the experiments made and last section V discuss all the research, first results and future work.

II. MYRABot+

The robot presented in this research is the result of the evolution of our first MYRABot solution, a platform for basic elderly assistance. The main motivation guiding the design of our platform was to provide a low-cost tele-presence platform capable of providing augmented reality for pills dose management. We wanted to develop an available platform for less than 1500 euros (less than 2k euros with and arm) in order to be affordable for elderly homes. The figure 1 identifies the main components of the architecture.

The component labelled as 1 is the webcam camera attached on top of the platform. It is used for simple HRI tasks and object perception for grasping tasks. The component 2 is the display, we decided to use the netbook screen located in the bottom of the robot structure and numbered as 7 in the figure. This computer manages all the robot behaviors. The component named 3 is the robot arm, used for simple manipulation and robot grasping. The component labelled 4 is a kinect RGB-D camera used for navigation tasks. It gets the power from Roomba base (component 6) that is the famous iRobot Roomba vacuum cleaner. Finally the element number 5 has the start/stop button and also the emergency button, which is a must to take part in a robotic competition.

A. Arm

Regarding the arm (figure 2 a)) we designed a solution meeting three conditions: low cost, low weight and able to be powered by Roomba base.

From the hardware point of view, the robotic arm was built using Bioloid, the educational kit. It is controlled by an Arduino Mega 2560 board, and is fed by Roomba battery after a small modification in the base.

From the software point of view, we initially developed an embedded solution but in order to make the integration with the system easier, we finally developed a ROS module in charge of controlling it.

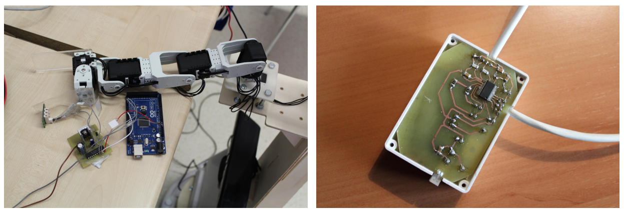


Fig. 2. a) Arm + Arduino Mega 2560 b) Control Board (PC-Roomba)

B. Frame

During our research we have built different prototypes for robot frame, all of them made in wood. We used a 5 mm laminated wood to make the frame. This is not a common material used to make a robot, but it is one of the cheapest way to do it.

The major problem was that the frame suffers small shakiness during navigation, and we solved it decreasing linear an angular velocities and adding an extra castor wheel.

C. Base

As a mobile platform we have a roomba vacuum cleaner robot. Similarities with iCreate (from iRobot too) let us use ROS and turtlebot packages.

The communication interface used in this platform between the robot and the laptop, is a custom designed board developed by the team supported in the classical serial port features. It is presented in figure 2 b).

The base supplies the power for the two components, the kinect and the arm. We made a modification in the interface to power the kinect (12V). We get it from 5 pin connector on top of the base. The power for the arm is got from vacuum motor.

D. Simulator

In order to be able to test all our developments prior to be deployed on the platform, we have developed an initial model of our platform for the Gazebo simulator. The figure 3 presents the model. We also developed a model for the rviz, the 3D visualization tool available in ROS, it is depicted in the right part of figure 1.

III. Component Oriented Software Architecture

We use a component architecture to generate robot behaviors. This architecture has been implemented in a single ROS node to take advantage of its benefits, but providing interesting features as presented below.

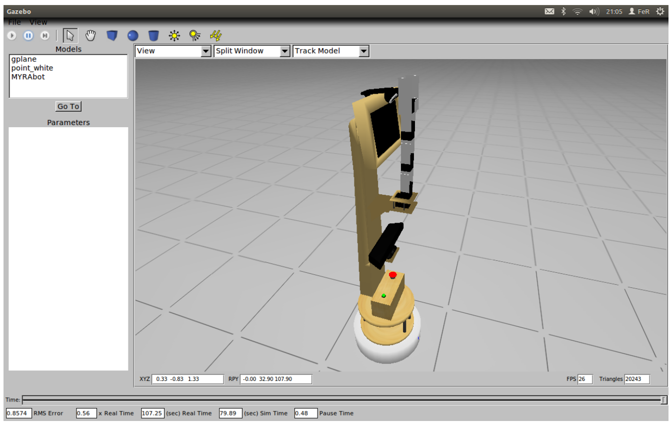


Fig. 3. MYRABot+ model for Gazebo simulator

Figure 4 describes the implementation scheme of a robotic application using our software architecture. There can be multiple ROS nodes containing components of our architecture, which communicates with other ROS regular nodes to take advantage of existing ROS software. Besides ROS and ICE communications can be used to interconnect any component to other processes or even debugging graphics applications.

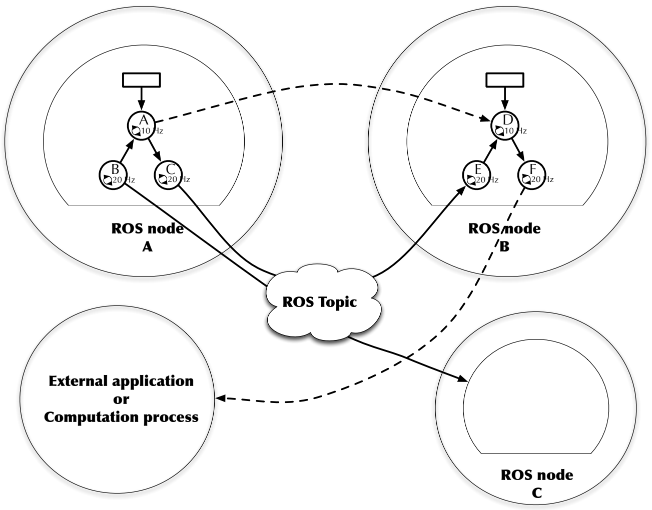


Fig. 4. Distributed scheme of our software architecture. We use both ROS and ICE communications to interoperate with regular ROS nodes, those which also implements our architecture, and another processes as GUIs and computation units.

The behaviors that the robot carried out are implemented using a component-based scheme. Each behavior can be decomposed into simpler functional units that are executed iteratively at different rate. Figure 5 shows the basic behavior of looking for a cup. The overall behavior is formed by the iterative execution of three components. One component analyzes the image looking for a cup. Another component controls robot's motors. A third component modulates motor component using the perceptive information.

The software schemes sumbsumption paradigm, the behavior of a robot is implemented as the interaction of various software components running at once. This can be implemented as the concurrent execution of several ROS nodes that publish the results of its implementation on topics, which are used by others.

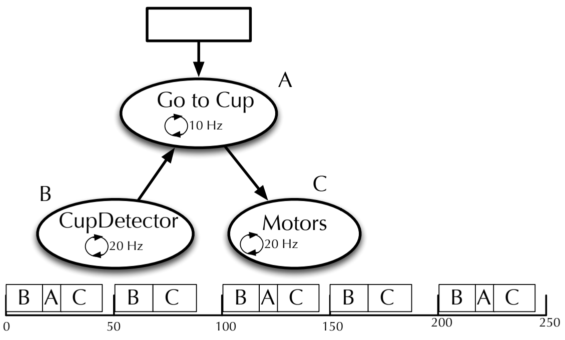


Fig. 5. Relation among components, and the Gantt diagram of the execution.

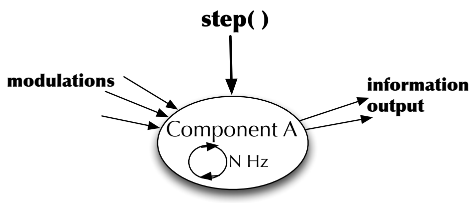


Fig. 6. A component with its interface.

The basic building block in our architecture is the component (figure 6), that represents the basic unit of functionality. The main idea is building component that it does only one thing, but efficiently.

A component is composed by three main parts:

- Modulations: The modulation methods set operation modes or set up the next component iterations.
- Execution: All the components inherit from the virtual class `component`, which defines the mandatory methods to be implemented. The most important method is `step()`. This method performs an iteration of this component. This is the entry point for a component-explicit execution.
- Output: The results method is used to get the information produced in the last iteration.

Components run in the same process, using a single thread. This eliminates race conditions and has benefits for component synchronization. Sharing the same memory address space also gives certain advantages from the point of view of implementation, such as being able to use design patterns (such as singleton) or reduce the time spent on the interconnection of components. Instead of operations between ROS nodes, there are direct calls to local procedures. ROS provides a good mechanism to avoid race conditions, in our architecture most of these problems are solved using a single thread implementation of all components.

If a component requires information from another component, it explicitly calls its `step()` function before collecting the data. If a component modulates the execution of another component, it first modulates, and then it calls its `step()`

method. The function `istime2run()` evaluates to true if the time elapsed since last execution is longer than $\frac{1}{f}$, where f is the frequency. Depending on this, `step()` method carries out its work, or not . Below is a basic example of a typical implementation of a component A that uses the information from B to modulate C.

```
//Two components used by A
B A::b;
C A::c;

A::A()
{
        b = B::GetInstance();
        c = C::GetInstance();
}

A::step()
{
        b->step();

        if(isTime2Run())
        {
                int info = b->getInfo();
                c->setInfo(info);
        }

        c->step();
}
```

Components can be very simple or very complex. Simple components communicate with the underlaying system methods to communicate with sensors or motors, or use a fixed number of components. Complex components can be implemented as finite state machine, changing the set of components it activates dynamically depending of the state. We have developed a useful tool for designing these complex components. This tool generates the code of the graphically represented behavior. An example of this tool can be seen in figure 7.

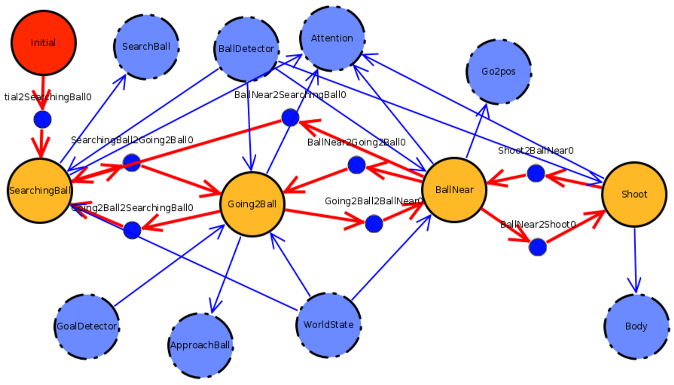


Fig. 7. Visual tool for developing behaviors as finite state machine inside a component. Blue circles are other component dependencies, yellow circles are states (the red circle is the initial state), and red arcs are the transitions between states. This tool generates the complete implementation of a component.

An interesting aspect of our approach is the use of resources. When a component uses another, explicitly calls his `step()` method. In this way, components that are not being used by any other component, does not run, saving computing power. There are not explicit activations or deactivations, or situations where components are running, but its result is not being used.

IV. Experiments

Our architecture intensively uses the ROS resources. As described in the previous section, we have used the simulator Gazebo and rviz visualization tool. In addition, ROS provides all drivers to access the hardware of the robot, presenting it in a compact and easily way as topics. PCL and Bullet libraries allow to handle complex structures such as RGB point clouds in 3D and have operations such as reference axes changes. We tested the full system in two environments: a test bed environment restricted to our lab and the RoCKIn challenge. All the tests performed in our lab started using the Gazebo simulator and our models. The tests related with the RoCKIn carry many steps, disassembling/assembling of the platform, software integration during camp and work with the system under stress or restricted conditions related with the competition environment as for instance use a special network.

A. *Software architecture experiments*

The software architecture that we used has been successfully tested in other applications such as robot soccer [19] or application of robot in Alzheimer therapies with humanoid robots [20]. This experience has not only demonstrated the validity of our approach in varied and dynamic environments, it has produced lots of tools for developing and debugging behavior.

In addition to this successful experience, there are more reasons to use this architecture. We could have implemented each component in a different ROS node running a certain frequency, and using ROS communications. Previously we have stated that it would not be appropriate to run nodes with components that are not used. Our approach does not.

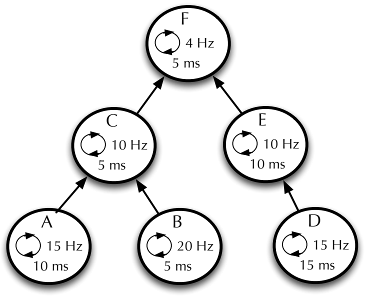


Fig. 8. Experiment set up. A behavior is composed by the execution of several component, running at different rate and with a medium computation time.

Another important feature in our approach is, by design, that the time between a data is produced and used is reduced. A component that uses data from another component requires the freshest information possible. To demonstrate this feature, we designed an experiment and we measured the time from a data occurs until used. In figure 8 we show a behavior composed by six components. Component C uses the information produced in the execution of component A and B; component E uses the information from D; and component F uses information from C and E. We have implemented this scheme both in ROS (each component in a different node) and our architecture, and we have measured the time elapsed since the result of each component is produced until it is used. Figure 9 shows the results of this experiment. The age of the data consumed by components C, F and E in the ROS implementation is bigger than in our approach. This is specially critical in components that needs real time conditions.

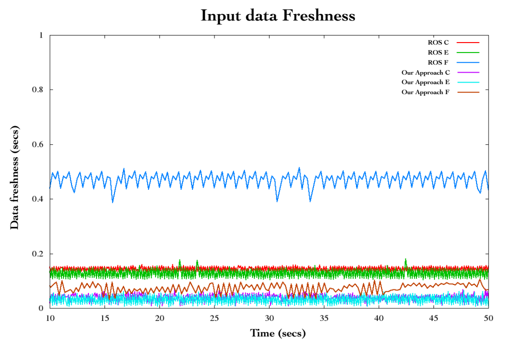


Fig. 9. The elapsed time since a data is produced and it is consumed.

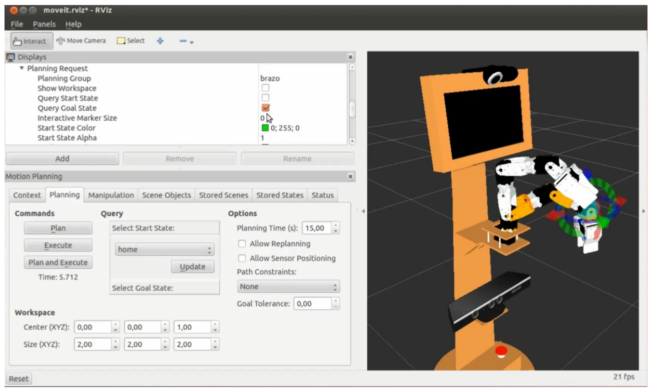
B. *RoCKIn Camp 2014*

We participated in the first RoCKIn camp 2014 in Rome Fig.rockincamp. The organization has divided the challenge in two phases, the first oriented to robot set-up and team collaboration. The second is the challenge itself against other teams.

This first camp was not the real competition phase, it was presented as the initial set up phase. The schedule was separated in two sessions: training in the morning and hands on in the afternoon.

We were able to identify some problems in our platform. We also deployed new solutions to apply in our solution during this camp.

The software problems identified during the camp were: the platform was working with the old ROS Fuerte distribution and some of the solutions given in the camp for grasping or perception were not prepared for our platform, so we decided to upgrade our version to ROS hydro. We migrated everything in two days and we were ready to integrate some of the camp solutions.



Planning Request
Planning Group: brazo
Show Workspace: ☐
Query Start State: ☐
Query Goal State: ✓
Interactive Marker Size: 0
Start State Color: 🟩 0; 255; 0
Start State Alpha: 1

Time: 5.712
Select Start State: home
Select Goal State:
Planning Time (s): 15,00
☐ Allow Replanning
☐ Allow Sensor Positioning
Path Constraints: None
Goal Tolerance: 0,00
Center (XYZ): 0,00 0,00 1,00
Size (XYZ): 2,00 2,00 2,00

Fig. 10. MoveIt! integration

The improvements done during these days were: a) the integration of our model with the MoveIt! solution (figure10) provided by RoCKIn organization. b) A new component to our system to manage the object recognition with depth cameras. The problem related with this new component was the computational cost. This task has to be done off-board because the computational cost needed is too high for the computer (a netbook PC) mounted on-board. Everything can be seen in the web [1] of our project.

We also find two problems were related with hardware. The first one was related with the robot morphology, we were not able to deploy some perception solutions for grasping because our 3D camera is situated in a lower position and only for navigation tasks. In this way we decided to put an extra Xtion Camera on top of the robot as is shown in right picture of figure 11.

The last problem was related with power supply, we burn our battery and were not able to feed kinect or arm without wires. The reason is that we carry our first prototype to the camp, and it was extensively used for labs experiments.

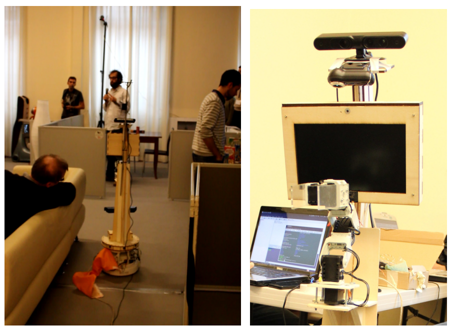


Fig. 11. RoCKIn Camp: a) Team demo b) Improved robot perception with a RGB-D camera on top of the platform

V. Conclusion

We have made a low cost robotic platform for @home competitions. We built a first prototype for RoCKIn@home challenge that was tested in the first Rome camp (left picture of figure 11).

The robot works under ROS environment but also a high level control architecture using cognitive features was developed. We have implemented a component oriented software architecture which uses all the resources of ROS, but implemented in a single ROS node. This behavior-oriented architecture is thread safe. The scheduler does not create multiple threads to execute components. Only one thread calls sequentially to the scheduler list of components. This thread executes in cascade the components, in the order defined depending on the relation of the components (modulation or results), as we presented in figure 5.

If any of the components sporadically spends more time than that desired, the systems suffers from what in real-time literature is called ”graceful degradation”. The execution of the other components is delayed, but no executions are canceled or overlapped. In the development phase of the components,

[1]http://robotica.unileon.es/mediawiki/index.php/RoCKIn2014

offender components are detected because `istime2Run()` methods of each component periodically test if the frequency is achieved, generating a warning if not. The set of of components that a component can activate varies dynamically. As there is no explicit deactivation method, its step function is simply not called anymore, we have to design the component having in mind that a component does not know when it is going to be called again. This is called "quiet shutdown". The information produced by a component can be used by several components.

This is very common in components that extract information from the sensors, and which is used by several components. It is especially critical in complex sensors such as images, in which the processing time is not negligible. In our architecture, these perceptual components are set to the frequency at which the information is completely valid between executions. Thus, separate components requiring the same sensory information does not require additional executions of perceptual components.

We found a some problems during the fist robot camp but related with hardware, to be precise with the prototype power, not with the software or architecture. All software was tested before, during and after de camp and new solutions were developed during these days and can be seen in the project page. We also migrated all the platform to hydro ROS environment.

As a future development we have two well define development lines. In one hand we want to solve the hardware issues: the one founded with the power supply, as we have developed all the platform supported in the Roomba battery, an improvement should to be done. The second, is to prepare the platform for the integration of a depth camera, in this way again the power supply needs to be taking into account. By the other hand we want to perform stress tests to know the robot limits.

ACKNOWLEDGMENT

This work was partially supported by University of León and Cátedra-Telefónica.

REFERENCES

[1] K. Konolige, K. Myers, E. Ruspini, A. Saffiotti, *The Saphira architecture: A design for autonomy*, Journal of experimental & theoretical artificial intelligence 9 (2-3), 215-235. 1998.

[2] K. Konolige, *Saphira robot control architecture*, Technical Report, SRI International, Menlo Park, Calif, USA, 2002.

[3] R. A Brooks, *Intelligence Without Representation*, Artificial Intelligence 47 (1991) 139-159.

[4] A. Brooks, T. Kaupp, A. Makarenko, A. Orebck, S. Williams, *Towards Component-Based Robotics*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005) p. 163–168. 2005.

[5] A. Makarenko, A. Brooks, T. Kaupp, *On the Benefits of Making Robotic Software Frameworks Thin*, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007). Workshop on Evaluation of Middleware and Architectures. 2007.

[6] B. Gerkey, R. T. Vaughan, A. Howard, *The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems*, In Proceedings of the 11th International Conference on Advanced Robotics (ICAR 2003), pages 317-323, Coimbra, Portugal, June 2003.

[7] H .J. Collett, B. A. MacDonald, B. Gerkey, *Player 2.0: Toward a Practical Robot Programming Framework*, In Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2005), Sydney, Australia, December 2005.

[8] R. T. Vaughan, *Massively multi-robot simulations in Stage*, Swarm Intelligence, 2(2-4):189-208, 2008.

[9] M. Quigley, C. Ken, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, Y. Andrew, *ROS: an open-source Robot Operating System*. Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics, 2009.

[10] R. Cintas, L. J. Manso, L. Pinero, P. Bachiller, P. Bustos, *Robust Behavior and Perception using Hierarchical State Machines: A Pallet Manipulation Experiment*. Proceedings, Journal of Physical Agents, ISSN 1888-0258. Vol. 5, No. 1, pp 35-44. March 2011.

[11] M. Montemerlo, N. Roy, S. Thrun, *Perspectives on standardization in mobile robot programming: the Carnegie Mellon Navigation (CARMEN) Toolkit*. IROS 2003: 2436-2441. 2003.

[12] G. K. Kraetzschmar, H. Utz, S. Sablatng, S. Enderle, and G. Palm, *Miro - Middleware for Cooperative Robotics*. Proceedings of RoboCup-2001 Symposium, volume 2377 of Lecture Notes in Artificial Intelligence, pages 411-416, Berlin, Heidelberg, Germany, 2002. Springer-Verlag.

[13] M. Henning, *The Rise and Fall of CORBA*, ACM Queue Magazine (Vol 4, Issue 5, June 2006).

[14] J.M. Cañas, V. Matellán, *From bioinspired vs psychoinspired to ethoinspired robots*. Robotics and Autonomous Systems, Volume 55, pp 841-850, 2007.

[15] Gerkey, B.; Conley, K., *Robot Developer Kits [ROS Topics],* Robotics & Automation Magazine, IEEE , vol.18, no.3, pp.16,16, Sept. 2011 doi: 10.1109/MRA.2011.942483

[16] Chih-Hung King, Marc D. Killpack, and Charles C. Kemp,*Effects of Force Feedback and Arm Compliance on Teleoperation for a Hygiene Task* , Eurohaptics, 2010

[17] Patrick Goebel, *The Chronicle of Pi Robot, Dynamixel using ROS*, Robotis, Available online:http://www.robotis.com/xe/158765

[18] Francisco J. Lera, Víctor Rodrguez, Carlos Rodríguez and Vicente Matellán, *Augmented Reality in Robotic Assistance for the Elderly*, In International Technology Robotics Applications. ISBN: 978-3-319-02331-1, 2013

[19] F. Martín, C. Aguero, J. M. Cañas, E. Perdices, *Humanoid Soccer Player Design*. Robot Soccer. Ed: Vladan Papic, pp 67-100. IN-TECH, 2010.

[20] F. Martín, C. Aguero, J. M. Cañas, P. Martínez, M. Valenti, *RoboTherapy with Alzheimer Patients*, International Journal of Advanced Robotic Systems: Humanoid. Vol. 9, pp 1-7. 2012.

[21] R. Bonasso, T. Dean, *A Retrospective of the AAAI Robot Competitions*, AI Magazine, 18(1), 11-23. 1997.