A robotic platform for domestic applications

Francisco Martín, José Mateos, Francisco J. Lera, Pablo Bustos and Vicente Matellán

***Abstract*—During the last years, robots are leaving industrial or research environments and sharing the same space with humans. This paper describes a robotic platform and its behavior architecture created to develop the necessary skills to help a human being in its own environment. This mobile robot is equipped with one robotic arm with 7 degrees of freedom and a RGBD depth camera as main sensor. The software behavior architecture developed for this robot can develop complex behaviors in a simple way, providing real-time features with low computational consumption. This architecture will be validated in a public robot competition called RoCKIn@home.**

***Index Terms*—Mobile robot, robotic manipulator, software architecture, robot competitions.**

I. Introduction

THE presence of robots in our domestic environments is an imminent event. These robots will help us doing the homework, will guard the house when we are out or will take care of our elders. Despite this imminence, there are still many problems to be solved. We need affordable robots that make their common use feasible. The houses, and all that is in them, are designed for humans. It is therefore important to provide robots with similar human characteristics: actuators similar to human hands, human-size, locomotion actuator with legs when stairs are present, collision avoidance methods to avoid colliding with delicate objects, etc... In addition to these hardware size, there are many software problems that need to be solved. The robot must move safely around the house. Hence, it is necessary to know where it is, needing in advanced navigating and self-localization methods. We also need to develop SLAM algorithms with long-term, but dynamical, maps. Also, you must live with a human being without becoming an uncomfortable partner. If we care of elder, we can not introduce another person to care of the robot itself. You must also be able to develop a set of tasks independently without requiring continuous monitoring. All these challenges, and more, have to be addressed before receiving our new family member.

Sharing our environment with robots has been an exciting challenge since mobile robots started to be a fact. The first explored environments were offices, usually at universities. First mobile robots [3][4] began to navigate at these environments. Robot Xavier [5] also demonstrated to navigate avoiding obstacles. CoBot [6] is one of the last office robot. This robot is able to navigate along a multi-floor building. It has not legs to climb stairs, or arms to push the elevator button. It only waits in the elevator for anyone to ask him to push the desired floor button. When lost or confused, it asks for help to the first person it detects. Anyone can ask to CoBot to send a physical mail, or to do anything. The person only has to require CoBot to his room by a web application. Robot starts to

Francisco Martín is with University Rey Juan Carlos.
E-mail: francisco.rico@urjc.es
Francisco J. Lera and Vicente Matellán are with University of León.
E-mail: {fjrodl, vicente.matellan}@unileon.es
Pablo Bustos is with University of Extremadura.
E-mail: pbustos@unex.es
José Mateos is with IaDEx.
E-mail: jose.mateos@gmail.com

be reliable and autonomous after a long time in offices.

Mobile manipulators have been at the center of the robotics community during the last decade. A mobile base coupled with some arrangement of arms and an expressive head are the departure point to social and service robotics. A recent review can be found in [15]. As a necessary following, domestic robots are starting to receive great attention. Beyond entertainment [7] or cleaning [8] robots, we would like to have a complete helper at home. Ambient Assisted Living (AAL) technologies explore how to introduce these type of robots as assistive components [9] or a helpers [10]. Usually these robots are equipped with a touchscreen for interaction with the robot, and they include navigation capabilities. This is enough for providing assistance or telepresence applications, but if we want an effective and versatile domestic robot, we need object manipulation capabilities. Many other affordable robotic platforms have been built during the last decade. Some recent affordable examples are for instance πrobot [12], Maxwell robot developed by Michael Ferguson, software engineer from Willow Garage, or the EL- E: An Assistive Robot from George Tech Healthcare Robotics Lab [11]. All these platforms have been designed for HRI. All of them include a positionable arm, stereo camera mounted on top of the robot and anthropomorphic appearance.

Robot competitions have been used during several years, like AAAI [13] competitions or RoboCup[14]. These competitions foster the research on robotics by providing a challenging scenario where many technologies can be applied and compared. @home the RoboCupRescue league and the RoboCup Junior league or the @home competition focused in the development of new solutions in assistance and personal robotics.

Many other competitions are organized every year, locally, regionally or globally. Some examples are the DARPA challenge, World Robot Olympiad, Robofest, AAAI Grand Challenge, FIRST competition or the RoCKIn Challenge. We will focus on this last one. RoCKIn is an EU project that involves robot competitions, forums, educational camps and workshops. The main goal of this challenge is not only the competition, but to define a testbed able to measure robots capabilities in two well defined environments: @home for better service robotics and @work to improve and measure industrial robotics. Robots that enter in this kind of competitions are brand new platforms with expensive sensors and actuators. We can find robots like REEM robot from PAL, Amigo robot from Tech United or care-o-bot from Fraunhofer IPA.

The aim of the research described in this paper is to present a domestic robot equipped with advanced manipulation capabilities, and the control software designed to carry out complex tasks. We propose an international competitions as a testbed. In particular we have choose RoCKIn, in its @home flavor, to test it.

The second contribution of this research is a component-oriented architecture on top of ROS, running within a single ROS node. Just because to share the same memory address space already provides 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.

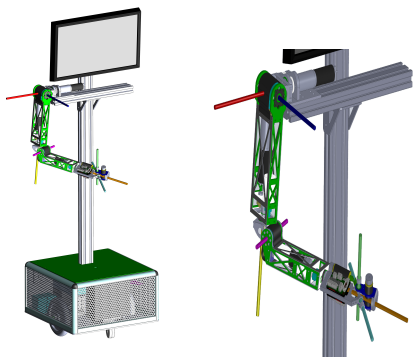
The software schemes sumbsumption paradigm, the behavior of a robot is implemented as the interaction of various software components running at once. This can be implemented as the concurrent execution of several ROS nodes that publish the results of its implementation on topics, which are used by others.

The rest of the paper is organized as fol-

lows: section II describes the robot platform. Section III summarized the software architecture used in the platform. After describing some experiments in section IV, we will comment some conclusions in section V.

II. Robotic platform

The software behavior architecture described in this paper will be tested on a new robotic platform that has been designed specifically for it. Having the opportunity to write down the requirements of a robot for domestic applications and being available some funds to order it, is not a common situation nowadays. The new robot had to have a functional and robust 7 dof humanoid arm (Figure 1a) where a simple gripper could be attached to. Also, it needed a differential autonomous base with batteries, charger, DC and AC buses, a tablet playing the role of an expressive head and an RGBD depth camera. After some comings and goings, the final design boiled down to a robust and powerful enough one arm configuration built using high-end motors, gearboxes and controllers, assembled on top of a light-weight differential base. The final CAD drawing of the robot is depicted in Figure 1a, where the motor axis are shown in blue.



(a) Complete robot (b) Detail of the arm and wrist

Fig. 1. Final CAD drawing of the robot with motor axis in color

The mobile base is a built with the two powered wheels centered sideways, so it can turn with respect to its geometric center occupying very little space. This configuration has the drawback that two caster wheels are needed to complete the support. To avoid that one of the four wheels loses contact, the back caster wheel has been modified to incorporate a spring and absorb most of the floor irregularities. Inside the base, see Figure 2, two 30W Maxon motors with 30:1 planetary gearbox connect through a flexible coupling to the bearing support where the wheels are fixated. Two batteries of 20 Ah provide a 24V DC bus that feeds the motors and their control electronics. Also, a 300W 220V AC inverter has been installed to provide AC energy for external devices. The outer housing is made of aluminum. Attached to the base, a 150cm aluminum bar holds the torso of the robot where the arm is placed.

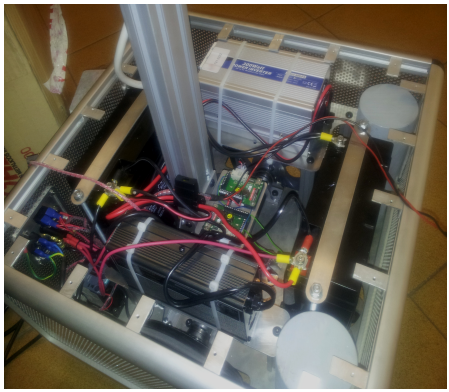
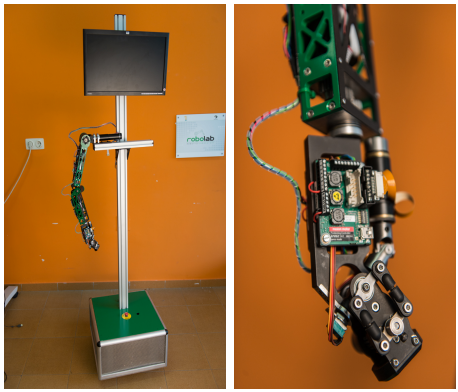


Fig. 2. Detail of the disposition of elements inside the base

So far, the robot has been equipped with one 7 dof arm. The disposition of the motors is anthropomorphic and the wrist has been slightly turned towards the central axis of the body, to facilitate grasping procedures.

The robot is controlled by a set of low-level components defining a Hardware Abstraction Layer. These components control the different interaction devices, such as the tablet in the head, micros and speaker, and also the base and the arm. For the base, both motors are driven by Maxon EPOS



(a) Real robot (b) Detail of wrist construction

Fig. 3. Real robot

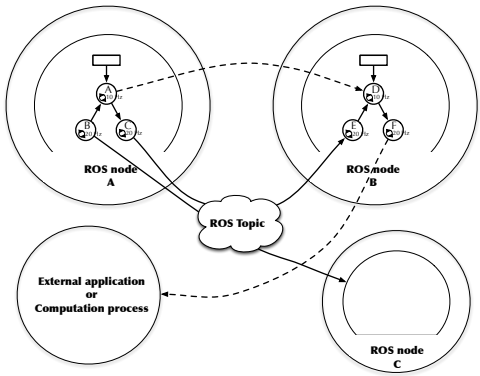


Fig. 4. Distributed scheme of our software architecture. We use both ROS and ICE communications to interoperate with regular ROS nodes, those which also implements our architecture, and another processes as GUIs and computation units.

3 modular controllers connected via Can bus and offering a USB interface to the computer. The arm is also controlled by a bus of EPOS that can be accessed from the computer through a simple C API.

III. Software Architecture

We use a component architecture to generate robot behaviors. We have implemented this architecture inside a ROS node to take advantage of its benefits, but also adding interesting features that are presented below.

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

The behaviors that the robot unfolds are implemented using a components scheme. Each behavior is decomposed into simpler functional units that are executed iteratively at different rate. Figure 5 shows the basic behavior of going to 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 the robot's motors. A third component modulates motor component using the perceptive information.

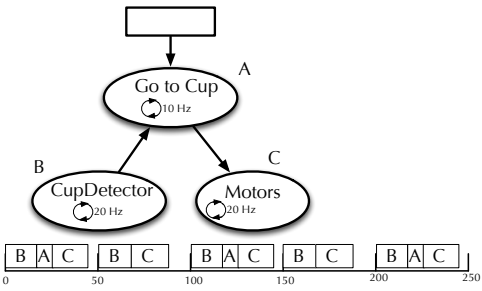


Fig. 5. Relation among components, and the Gantt diagram of the execution.

The basic building block in our architecture is the `component` (Figure 6), which is the basic unit of functionality. The main idea is to define components that only do one thing, but very efficiently. A component is composed of 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.

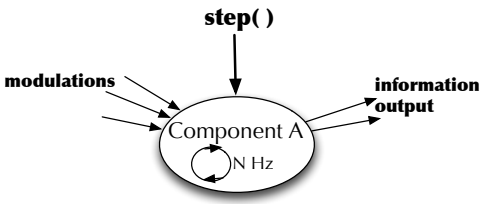


Fig. 6. A component with its interface.

Components can be very simple or very complex. Simple components communicate with the underlying system methods to talk to sensors or motors, or to use some other components. Complex components can be implemented as a finite state machine, so the set of components that are activated depend dynamically of the state. We have developed a useful tool to design these complex components. This tool generates the code for a behavior represented graphically An example of this tool can be seen in Figure 7.

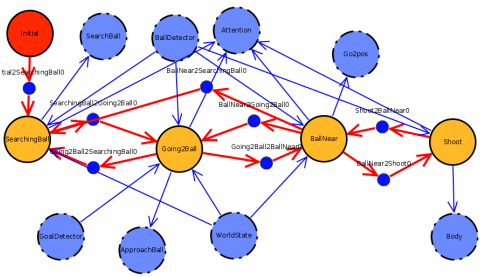


Fig. 7. Visual tool for developing behaviors as finite state machine inside a component. Blue circles are other component dependencies, yellow circles are states (the red circle is the initial state), and red arcs are the transitions between states. This tool generates the complete implementation of 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, do not run, saving computing resources.

IV. Experimental results

The software architecture that we used has been successfully tested in other applications such as robot soccer [1] or application of humanoid robots in Alzheimer therapies [2]. This experience has not only demonstrated the validity of our approach in different and dynamic environments; it has produced lots of tools for developing and debugging robot 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 being used. Our approach does not.

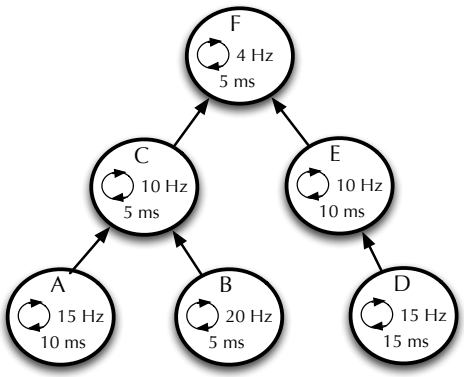


Fig. 8. Experiment set up. A behavior is composed by the execution of several component, running at different rate and with a medium computation time.

Another important feature in our approach is, by design, the optimization of the elapsed time since data is produced until it is used. A component that uses data from another one 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 made up of six components. Component C uses the information obtained from 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 in 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 ROS implementation is bigger than in our approach. This is specially critical in components that needs real time conditions.

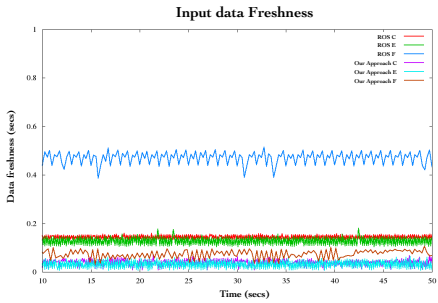


Fig. 9. The elapsed time since a data is produced and it is consumed.

In our second experiment, we demonstrate how this architecture can integrate different skills to get a long term behavior. One of the tests sets out in the RoCKIn@home competition is to serve a human as a waiter. The scenario is shown in Figure 10. It has a principal door, a couple of corridors, a dinning room and a kitchen. The size of this scenario is 30 m^2. The robot has to follow these steps in this scenario:

1) Navigate to the principal door
2) Say hello to a person
3) Walk with that person to the dinning room
4) Ask for a drink
5) Go to the kitchen
6) Take the desired drink

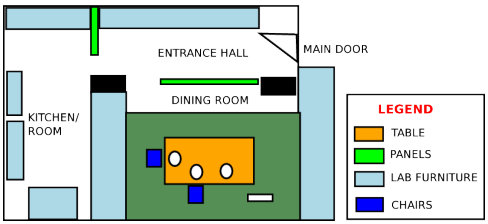


Fig. 10. Scenario for RoCKIn testbed.

7) Go to the dinning room and deliver the drink

This test has a time limit of 15 minutes. To solve this scenario we have decomposed the problem in some components. The high level behavior is shown in Figure 12. It is a finite state machine with the phases of the complete problem. In each state, different components are used. Global (in the world axes) and local (in the robot axes) navigation capabilities are implemented in the `GlobalNavigator` and `LocalNavigator` components. They receive a position (x, y, θ) to be reached. These components can be asked if the point has been reached. `Interactuator` is a component whose task is a human robot interaction. It can make some audible sentences and process the response. `Manipulator` controls the arm actuation. It can grab or deliver an object by setting a (x, y, z) goal position.

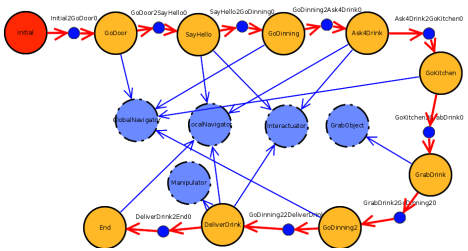


Fig. 11. High level behavior.

The component `GrabObject` is a component also implemented as a finite state machine (Figure 12). In the first state, this component looks for the object turning and using the perceptive component

`ObjectDetector`. When the component is found, the second state makes the robot to approximate to the object. The last state is implemented as a closed loop to grab the object using the perceptive information.

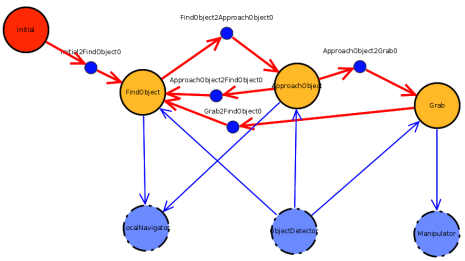


Fig. 12. Search for drink and Grab it behavior.

V. Conclusions

We have made a robotic platform ideal for domestic applications. This robot has a size similar to a human, and it moves using wheels. It is equipped with one arm able to manipulate simple objects. This arm has 7 degrees of freedom. All the processing can be done on-board, in a computer allocated in the lower part of the robot. We can connect multiple sensors to the robot, being a RGBD camera the main source of information.

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, 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.

This robot will participate in the RoCKIn@home competition. During the competition, the robot will be committed to solve some problems in a domestic environment scenario. These problems include manipulating simple objects like cans, navigating along the scenario, interacting with humans, or watching out emergency situations. Some of these skills are also integrated in a single test in which the robot has to show a long term (5 minutes) behavior, as described in the experiments section.

References

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

[2] 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.

[3] Nils J. Nilsson. *Shakey the robot*. Technical Report 323, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, Abril 1984.

[4] Alessandro Saffiotti, Enrique Ruspini, and Kurt Konolige. *A fuzzy controller for flakey, an autonomous mobile robot*. Technical Report 529, AI Center, SRI International, 333 Ravenswood Ave., Menlo Park, CA 94025, Abril 1993.

[5] Reid Simmons, Richard Goodwin, Karen Zita Haigh, Sven Koenig, Joseph O'Sullivan. *A Modular Architecture for Office Delivery Robots*, in Autonomous Agents 1997. February 1997. ACM. Pages 245-252.

[6] Manuela Veloso, Joydeep Biswas, Brian Coltin, Stephanie Rosenthal, Tom Kollar, Cetin Mericli, Mehdi Samadi, Susana Brandao, and Rodrigo Ventura. *CoBots: Collaborative Robots Servicing Multi-Floor Buildings*. In Proceedings of IROS'12, the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal, October 2012.

[7] Fujita, M. and Kitano, H. *Development of an autonomous quadruped robot for robot entertainment*. Autonomous Robots, 5, 7-20.

[8] Jodi Forlizzi, Carl DiSalvo. *Service robots in the domestic environment: a study of the roomba vacuum in the home*. roceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction. 258-265. 2006.

[9] Adriana Tapus, Juan Fasola and Maja J Mataric. *Socially Assistive Robots for Individuals Suffering from Dementia*. In ACM/IEEE 3rd Human-Robot Interaction International Conference, Workshop on Robotic Helpers: User Interaction, Interfaces and Companions in Assistive and Therapy Robotics. 2008.

[10] Joost Broekens, Marcel Heerink, Henk Rosendal. *Assistive social robots in elderly care: a review*. International Journal on the fundamental aspect of technology to serve the ageing society - Gerontechnology 2009; 8(2):94-103.

[11] 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

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

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

[14] RoboCup-97: Robot Soccer World Cup I . editor(s) Kitano, Hiroaki. Springer-Verlag, Berlin; New York, Year 1998.

[15] P. Bustos, I. García-Varea, J. Martínez-Gómez, J. Mateos, L. Rodríguez, A. Sánchez. Loki, a mobile platform for social robotics. WAF2012, Santiago de Compostela, 2012.