International Journal of Humanoid Robotics
Vol. 16, No. 6 (2019) 1950043 (13 pages)

DOI: 10.1142/S0219843619500439

World Scientific
www.worldscientific.com

Development of Applications for Humanoid Robots Using Multiple Platforms, Tools, and Cloud Data Sharing

Santiago Martínez*, Juan Miguel Garcia-Haro, Concepcion Alicia Monje and Carlos Balaguer

Robotics Lab, University Carlos III of Madrid,
Avda. de la Universidad 30, Leganes, 28911, Madrid, Spain
**scasa@ing.uc3m.es*

Received 31 March 2019
Revised 16 September 2019
Accepted 27 November 2019
Published 30 January 2020

This paper describes the procedure followed for using third-party tools and applications, avoiding the development of complex communication software modules for data sharing. A common practice in robotics is the use of middlewares to interconnect different software applications, hardware components, or even complete systems. It allows code and tool reuse minimizing the development effort. In this way, applications developed for one middleware can be shared with others by means of establishing communication bridges among them. The most extended procedure is the development of software modules that use the low-level communication resources that middlewares provide. This procedure has many advantages but a clear disadvantage: the complexity of development. The procedure proposed is based on the use of cloud technologies for data sharing without the development of middleware bridges. The way of interrelate different middlewares is by means of the development of a compatible robot model. This procedure has enabled the use of the ArmarX middleware tools and the application of the results obtained to the humanoid robot TEO, that uses the YARP middleware, in an easy and fast way.

Keywords: Middleware; cloud computing; humanoid robot.

1. Introduction

The complexity of the research in robotics makes it desirable to use the best tools available to improve and develop applications. Many tools exist and new developments are continuously launched into the market. All these applications help the researchers to achieve their investigation goals. But they usually find the barrier of the time to develop or integrate new tools in the existing system they are working

*Corresponding author.

with. Another barrier is the incompatibility among software systems. Due to this, the researchers cannot easily use the existing tools for improving their works. But nowadays, there are many ways of collaboration and one of them is the knowledge sharing through new ICT technologies, like the cloud data sharing and computing.[1] But this data sharing is performed inside their own working environment.

In the robotics field, the main software environment that provides intercommunication and integration of tools is the middleware. This special kind of program was defined by Bakken in 2001[2] as "a class of software technologies designed to help to manage the complexity and heterogeneity inherent in distributed systems. It is defined as a layer of software above the operating system but below the application program that provides a common programming abstraction across a distributed system."

The middleware behaves like an operating system for the distributed components usually found in robotic platforms. The analogy refers that, in the same way, an operating system allows communication between different components at very low level and organizes them, a middleware is responsible for establishing communication and coordination between the different distributed elements of a robotic platform that can include numerous sensors and actors, like motors, cameras and several PCs.

It is evident that the use of a middleware solves a large number of problems related to robotics development, but it has several inconveniences as well. On one side, the middleware provides the basic infrastructure to enable robotic applications, tools for development, communication channels, etc. In summary, it provides an adequate environment to operate a robotic system. But sometimes, the use of a specific middleware can be a cause of research and development delays. There are many factors that favor this kind of problem: lack of standardization, lack of appropriate tools for application development, missing desirable properties of the tools, etc. These problems force the researchers to develop their own tools to continue with the work, even when tools with the same features are available for other middlewares.

But luckily, it is possible to interconnect middleware to share tools and data. Mainly it can be done developing bridges between systems, but this is a complex programming task. An alternative way is the use of cloud systems for data sharing, and the appropriate robot software model (kinematics, dynamics, etc.) acting like the bridge or mean of inter-relation for systems.

2. Development of Humanoid Tasks Using Multiple Software Systems

In the past years, various middlewares have been developed such as ROS[3] or RT-Middleware,[4] some of them rely on standard communication libraries (e.g., CORBA or ICE[5]). However, robotic platforms are far from standardization, thus making sharing knowledge very difficult. Another important question is the duplicity of efforts to develop tools for each middleware. It is easy to state, reviewing the existing tools for each middleware, the existence of many tools with the same function available in them (simulator, planning tools, etc.). Researchers could benefit

from using the existing tools but usually it cannot be done. The main reason is the barriers of interoperability (communication channels, protocols, etc.) among tools and systems.

One possible option when we have two different middlewares is to make a software bridge between them.[6] Connecting two middlewares is not just solving the communication differences, it is also important to consider the discrepancies in performance or total possible incompatibilities. Also, we have to keep in mind that the task is not to open a socket in one middleware and connect it to another socket in the second, but to translate the structures to be understandable between completely different communication protocols.

An example of middleware-bridging that was studied previously for this work has been the bridge made between ArmarX and YARP.[7] It was used for transferring knowledge between the robots ARMAR and iCub.[8] The bridge allowed the use of the ArmarX environment without the need to develop a complete working platform of this middleware in the target robot ArmarX-iCub.[7] This work successfully allowed the iCub robot to use Armar-III *prior knowledge* to implement a grasping skill. The bridge was responsible for preparing the input data from iCub to be used in the ArmarX middleware. Then, ArmarX will use its internal tools to create an output response. This answer will be translated by the bridge again to the robot, in order to be executed. The workflow of this approach is simplified in Fig. 1.

Analyzing the work performed to obtain the intercommunication depicted in Fig. 2, we come to several conclusions. The first one is the huge amount of work needed to develop the intercommunication between both sides. This is caused not only by the necessity of adapting the communication channels, but also of establishing common protocols and structures. Second, the goal of establishing the infrastructure depicted is the interconnection of systems in real-time, but rarely the

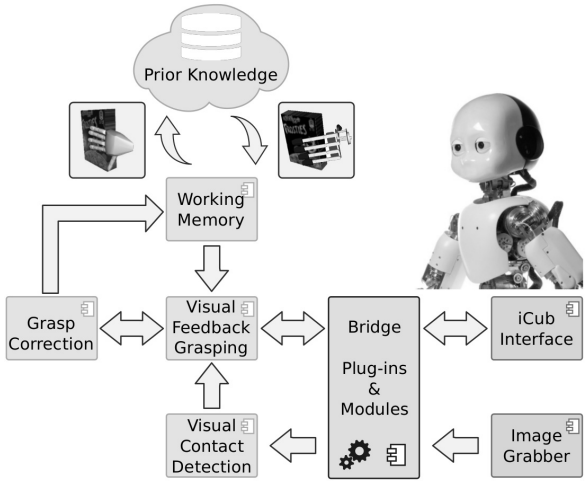

Fig. 1. Armar-III-iCub approach.

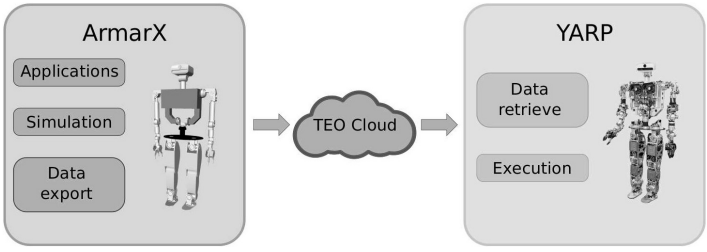

Fig. 2. Proposed approach for TEO.

feature is needed in a development phase. It is most commonly used once an application has been developed or if it is an imposed requirement. Finally, the last conclusion is that the bridge is usually unique and it cannot be used to interface with other systems, or even with other tools inside the same middleware, without adaptation in an easy way.

3. Enabling Middleware Interoperability

Therefore, this work exposes a solution based on trending technologies, like cloud data sharing, to solve the intercommunication of systems and make it easier and increase resources for researching. This last reason is one of the main motivations to foster the use of tools already developed and fully operative. Humanoid robotic researchers can really benefit from the use of interesting and powerful applications existing in different middlewares. In the authors' case, they can be used to improve abilities, capacities, and tasks of the robot platform Task Environment Operator (TEO). TEO humanoid robot software infrastructure is based on YARP middleware. It provides intercommunication tools, libraries and applications that enable the research in biped walking, body balance, manipulation, etc. Thanks to these capabilities many applications have been developed for this humanoid robot (Fig. 3), such a waiter application in which the robot is able to balance bottles on a tray[9] while using the robot vision system to analyze the object movement,[10] as can be observed in `https://youtu.be/Ai1WZgg6aKw`;[11] or the ironing application in which TEO robot is able to iron wrinkles of a shirt.[12] These applications have been developed not only by using the existing resources in the YARP middleware,[13] but also by applying a huge effort to develop new tools. This effort could be avoided using the existing tools in other software architectures and the results could be obtained faster.

YARP middleware enables the operation and integrates the computation system of the robot TEO. The hardware architecture is composed of several computers dedicated to different tasks. TEO integrates three computers, each one in charge of the *vision*, *manipulation*, *locomotion*; all connected by means of the YARP infrastructure. Moreover, YARP provides the mechanisms to interface with other subsystems such as motor drivers, sensors, and even remote control devices. TEO's architecture is depicted in Fig. 4.

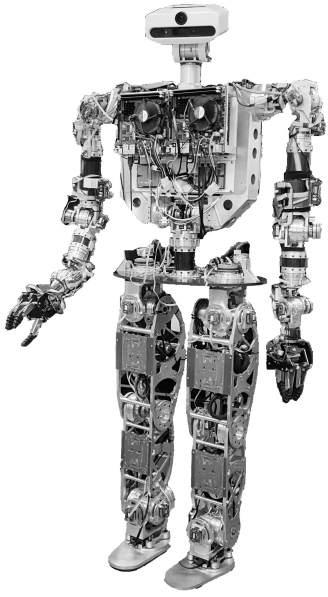

Fig. 3. The humanoid platform TEO from the RoboticsLab.

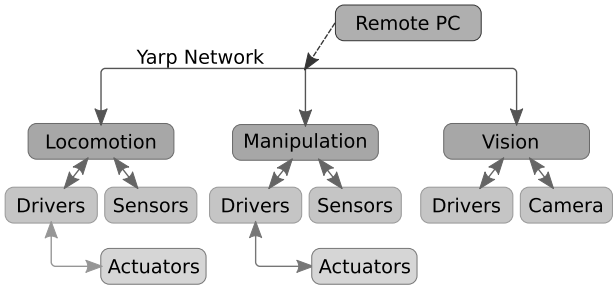

Fig. 4. TEO computer architecture.

Due to the previous collaboration between the High Performance Humanoid Technologies Lab (H^2T) from the Karlsruhe Institute of Technology (KIT) and the RoboticsLab from the University Carlos III of Madrid (UC3M), authors have knowledge about the existing tools in the middleware ArmarX. It is a framework that implements simulation, working management, dynamic and kinematic calculations, robot-task management[14] and finally, a middleware built over ICE.[15] ARMAR robots' family has distributed architecture for operating and controlling robotic systems.

It is detailed in Fig. 5.[16] The architecture is organized in three different layers. The top layer is dedicated to high-level task planning. The middle layer contains all modules and tools needed to enable the processing of information from the robotic

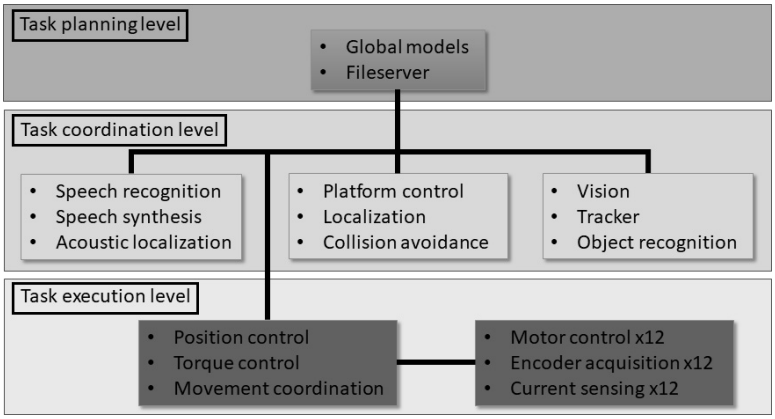

Fig. 5. ARMAR computer architecture.

hardware such as vision processing, grasping planners, localization, etc. The bottom layer contains the low-level controller that interfaces directly with the robotic hardware.

4. Development of a Multisystem Application

The most interesting layer for this work is the coordination layer and the tools integrated inside it. Due to the experience of the H^2T laboratory, many powerful tools for planing, grasping, and simulation are available but they depend on the ArmarX middleware. But the knowledge of both systems (TEO and ARMAR) has been the key to find an alternative way of interfacing between them: the robot model.

4.1. *TEO robot model*

TEO is an anthropomorphic robot designed by the UC3M, it is 1.70 m high and weights 56 k due to its aluminum structure. This robot is the next link in the RH family: TEO is the upgraded version of its predecessors RH-0 and RH-1. With 28 DOF and multiple sensorial systems, it is able to perform elaborated tasks resembling humans, including grasping and manipulation of objects.[20] TEO is composed of several mechanics elements, from top to bottom. It has a head with RGB and depth cameras. The trunk, where the main processors and power supply are located, has 2 DOF allowing upper body movement. The arms have 6 DOF each. They have a force–torque sensor at the wrist and a gripper allows grasping capabilities. The legs have also 6 DOF, similar to the arms, and have force–torque sensors in the ankles. This kinematic structure has been designed using CAD software whose outputs are 3D computer models of its parts. The relation of these 3D models is the kinematic chain. Besides, these models contain the dynamics of the parts (inertia axes, mass, etc.) that are very easy to compute in the CAD software. Then, these 3D models and

the related information are the basis for building any kind of model for any of the middlewares commented.

In order to use TEO in a new software environment, it is necessary to create a compatible computer model for it. YARP and ArmarX share the same file format for modeling robots: XML files in which the kinematic structure is defined and any individual part of a robot is described in a *RobotNode*. The concatenation of these *RobotNodes* will create a kinematic chain. The union of different chains creates finally the complete robot structure. This architecture also holds additional information such as the visualization, dynamics, or joint declarations.

The H^2T has developed several robot development tools, not only the ArmarX middleware. These include Simox, a C++ tools package that provides algorithms for 3D simulation. It also calculates grasping positions and collision-free trajectories.[17] They have also created the Master Motor Map (MMM). This project includes several applications for capturing human motions and transform them to be reproduced by humanoid robots.[18] MMM provides the tools to capture human motions and the KIT has created a database with over 3000 recorded human motions.[19] These tools use as main input the XML robot model which constitutes effectively the way of collaboration between platforms. The difference between the XML definition in ArmarX environment and OpenRAVE, the one previously used in TEO, can be seen in Listings 1 and 2.

The model hierarchy and structure starts from the *root* of the robot placed at the center of the hip. It coincides with the location of the Center of Mass (CoM) and allows the study of its behavior, specially for gait control. From the hip, there are three child sub-assemblies: one for the body and two for the legs. Each leg is a 6 DOF open chain that begins in the axial hip joint. The arms open chains are child of the body, in the same way than the legs. The XML file of the robot model represents this hierarchy.

```
<RobotNode name="LeftAxialShoulder">
  <Joint type="revolute">
  <Axis z="-1" y="0" x="0" />
  <Limits units="degree" hi="55" lo="-50" />
  </Joint>
  <Transform>
    <DHunits="deg" alpha="0" a="0" d="-329.01" theta="0" />
  </Transform>
  <Visualization>
    <Filetype="Inventor">models/2.2^leftArmLink.wrl</File>
  </Visualization>
    <Child name="LeftFrontalElbow" />
```

Listing 1. Part of the XML definition in Simox (Left Axial Shoulder).

```
<Body name="r22" type="dynamic">
  <offsetfrom>r21</offsetfrom>
  <Translation>0 0 -0.329</Translation>
  <RotationAxis>0 0 1 0</RotationAxis>
  <Geom type="trimesh">
    <Render> models/2.2^leftArmLink.wrl
    </Render></Geom>
  </Body>
  <Joint circular="false" name="q22" type="hinge">
    <Body>r21</Body>
    <Body>r22</Body>
    <offsetfrom>r22</offsetfrom>
    <limitsdeg>-50 55</limitsdeg>
    <axis>0 0 -1</axis>
```

Listing 2. Part of the XML definition in OpenRAVE (Left Axial Shoulder).

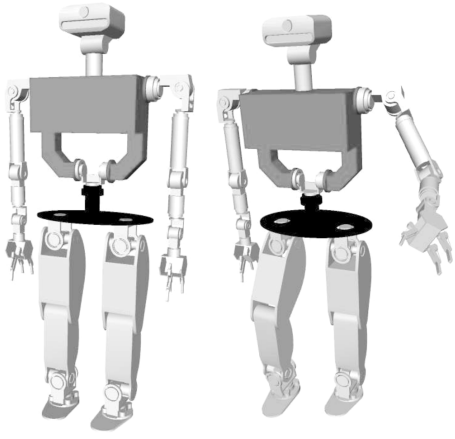

Fig. 6. TEO loaded in the Simox environment. The second image shows a test to check the joint articulations.

One of the capabilities from ArmarX framework that has motivated its use is the capacity of performing powerful dynamic simulations for TEO robot. With that purpose, all dynamic information about TEO (e.g., Com, inertia matrices) have been added to the XML definition. Besides, the robot model supports direct definition of the robot's sensors and cameras.[a] Figure 6 shows the results of TEO loaded in the Simox environment.

[a]https://github.com/roboticslab-uc3m/teo-simox-models.

5. Implementation of Multisystem Interactive Applications

The following procedure, once a model of the actual robot is defined, is to create a task to be executed by TEO. For illustrating the process, a movement task has been designed, simulated in the ArmarX framework environment, and shared to the real environment through TEO's cloud.

In this simple task, one movement pattern from the H^2T database has been adapted to TEO kinematics by means of the MMM application. This task consists of moving the robot's arm waving.[b] This back and forth movement is repeated four times and when done, TEO will return its arm to its initial resting position.

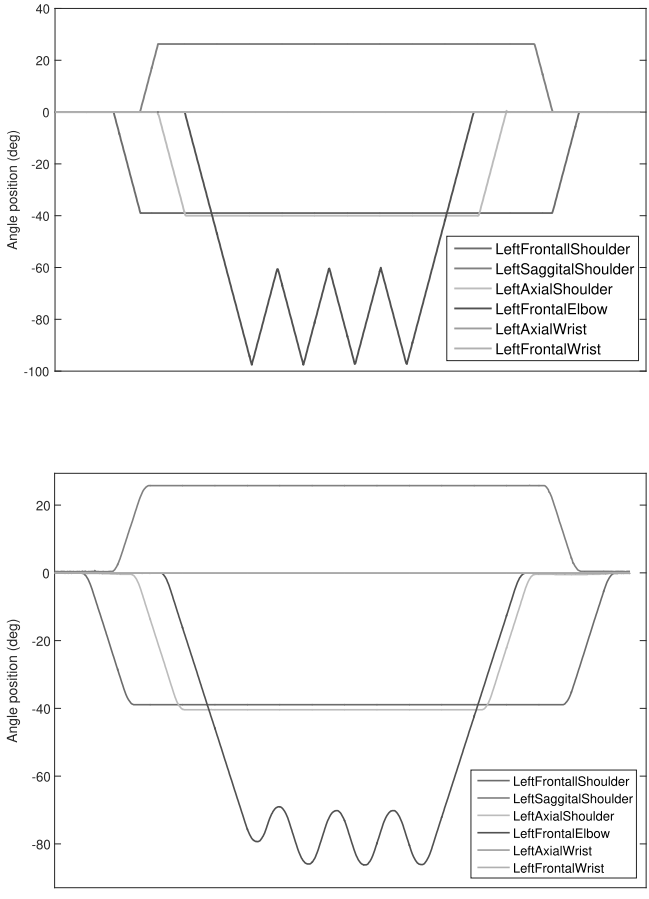

Fig. 7. Joint angle values comparison between the simulated model and the real robot. (a) Joint angles values during the simulation in ArmarX. (b) Joint angles values exported after executing the task in TEO.

[b]`https://youtu.be/KeIoJu5xmDk`.

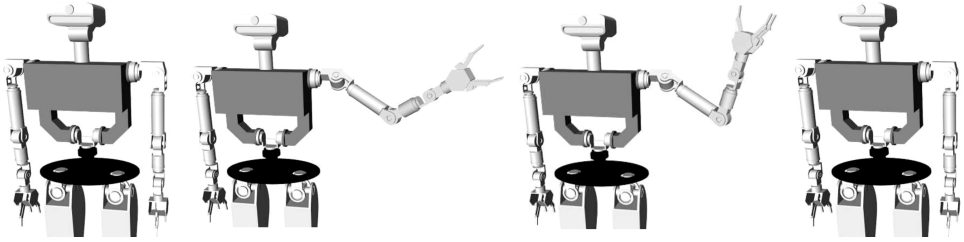

Fig. 8. TEO waving simulation in ArmarX. From left to right, the simulation of TEO waving.

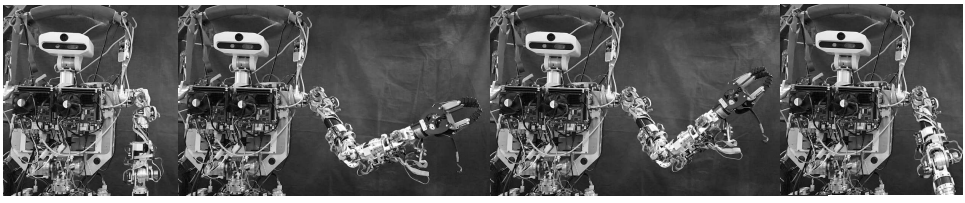

Fig. 9. TEO waving simulation in ArmarX (from left to right, the complete sequence).

Therefore, this tool links a general movement captured from humans with the TEO robot model developed in ArmarX format.

Then, to create an executable task inside ArmarX architecture, it is necessary to create a state chart to define its behavior, its inputs, and its outputs. This operation has been programmed using the StateChart application. This step has two main objectives. The first one is to check the task using the simulation environment in ArmarX, and the second one is to generate a text file with the low-level information that can be executed directly by TEO humanoid robot [Fig. 7(a)].

The StateChart is a graphic way to program finite states machines, that is based on Harel's formalisms[21] and allows high modularity and reusability, in addition disclosing the system state.[22] After creating the logic states, they have been implemented in the ArmarX simulation environment using TEO as a proxy. The successful simulation sequence can be seen in Fig. 8.

The output from the StateChart application is then exported to a text file as list of joint movements including timestamps. The file is automatically uploaded to TEO's cloud and it can be read directly from TEO's interpreter and then the movements are executed. This process is done directly from the YARP network already implemented in TEO [Figs. 7(b) and 9].

6. Conclusions

This work has presented a method for enabling the use of multiple tool from multiple robotic software platforms to be applied in a humanoid robot. It opens the possibility of using fully developed tools even when they belong to incompatible systems. This

procedure allows to extract data and share it to the robot without creating a new software interface. This is similar to the concepts exposed in the fourth industrial revolution (Industry 4.0) regarding cloud computing, data sharing, and big data. In Industry 4.0, the use of the cyberspace is key for the development of technologies and process data. In the same way, robotic environment can gain the advantage of using the capabilities that the cyberspace offers.

ArmarX has been presented as an outstanding robot framework. It provides a huge range of opportunities to the developer: database of movement patterns, task planners, simulators, etc. In this work, only a slight fraction of its potential was used. YARP middleware is also a good framework for developing the software for a humanoid robot system. But the maturity of the tools developed for both systems is very different. Then, the possibility of using the best tools from both middlewares is a very good reason to research in methods of intercommunication. The idea of data sharing between the software development environment and robotic hardware through a cloud provides an easy way of intercommunication. But, actually, it is not performed in real-time. Future development and technologies will enable the communication using a cloud in real-time applications. Then, remote control loops running in other environments could be used in the humanoid robot TEO for new researches.

Finally, this methodology is a way of empowering the collaboration between research groups and a very good way of sharing knowledge.

References

1. B. Kehoe, A. Matsukawa, S. Candido, J. Kuffner and K. Goldberg, Cloud-based robot grasping with the google object recognition engine, in *Proc. IEEE Int. Conf. Robotics and Automation* (Karlsruhe, Germany, 2013), pp. 4263–4270.
2. D. E. Bakken, Z. Zhan, C. C. Jones and D. A. Karr, Middleware support for voting and data fusion. in *Proc. Int. Conf. Dependable Systems and Networks* (Göteborg, Sweden, 2001), pp. 453–462.
3. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, Leibs, J. ... A. Y. Ng, ROS: An open-source Robot Operating System, in *ICRA workshop on Open Source System*, Vol. 3 (Kobe, Japan, 2009), p. 5.
4. N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W. K. Yoon, RT-Middleware: Distributed component middleware for RT (Robot Technology), in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (Edmonton, Alberta, Canada, 2005), pp. 3933–3938.
5. M. Henning, A new approach to object-oriented middleware, *IEEE Internet Comput.* **8**(1) (2004), pp. 66–75.
6. V. Issarny, A. Bennaceur, and Y. D. Bromberg, Middleware-layer connector synthesis: Beyond state of the art in middleware interoperability in *Formal Methods for Eternal Networked Software Systems* (Springer, Berlin: Heidelberg, 2011), pp. 217–255.
7. A. Paikan, D. Schiebener, M. Wächter, T. Asfour, G. Metta and L. Natale, Transferring object grasping knowledge and skill across different robotic platforms, in *2015 Proc. 17th Int. Conf. Advanced Robotics* (Istanbul, Turkey, 2015), pp. 498–503.
8. G. Metta, G. Sandini, D. Vernon, L. Natale and F. Nori, The iCub humanoid robot: An open platform for research in embodied cognition, in *Proc. 8th Workshop on Performance Metrics for Intelligent* (Gaithersburg, Maryland, USA, 2008), pp. 55–56.

9. J. M. Garcia-Haro, S. Martinez and C. Balaguer (2018). Balance computation of objects transported on a tray by a humanoid robot based on 3D dynamic slopes, in *2018 IEEE-RAS 18th Int. Conf. Humanoid Robots* (Beijing, China, 2018), pp. 704–709.
10. J. Hernandez-Vicen, S. Martinez, J. M. Garcia-Haro and C. Balaguer, Correction of visual perception based on neuro-fuzzy learning for the humanoid robot TEO, *Sensors* **18**(4) (2018) 972.
11. J. M. Garcia-Haro, E. Daniel Ona, S. Martinez, J. Hernandez-Vicen and C. Balaguer, Waiter robot application: Balance control for transporting objects, in *IEEE/RSJ International Conference on Intelligent Robots* (1–5 October 2018) (Madrid, Spain), p. 5036.
12. D. Estevez, J. G. Victores, R. Fernandez-Fernandez and C. Balaguer, Robotic ironing with 3D perception and force/torque feedback in household environments, in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (Vancouver, Canada, 2017), pp. 6484–6489.
13. G. Metta, P. Fitzpatrick and L. Natale, YARP: Yet another robot platform, *Int. J. Adv. Robot. Syst.* **3**(1) (2006) 8.
14. R. Zollner, T. Asfour and R. Dillmann, Programming by demonstration: Dual-arm manipulation tasks for humanoid robots, in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems* (Edmonton, Alberta, Canada, 2005), pp. 479–484.
15. N. Vahrenkamp, M. Wächter, M. Kröhnert, K. Welke and T. Asfour, The robot software framework ArmarX. *It - Inform. Technol.* **57**(2) (2015) 99–111.
16. T. Asfour, K. Regenstein, P. Azad, J. Schröder, A. Bierbaum, N. Vahrenkamp and R. Dillmann, ARMAR-III: An integrated humanoid platform for sensory-motor control, in *Proc. 2006 6th IEEE-RAS Int. Conf. Humanoid Robots* (Cancun, Mexico, 2006), pp. 169–175.
17. N. Vahrenkamp, M. Kröhnert, S. Ulbrich, T. Asfour, G. Metta, R. Dillmann and G. Sandini, Simox: A robotics toolbox for simulation, motion and grasp planning in *Intelligent Autonomous Systems* (Springer, Berlin: Heidelberg, 2013), pp. 585–594.
18. Ö. Terlemez, S. Ulbrich, C. Mandery, M. Do, N. Vahrenkamp and T. Asfour, Master Motor Map (MMM) — Framework and toolkit for capturing, representing, and reproducing human motion on humanoid robots, in *IEEE-RAS Int. Conf. Humanoid Robots* (Madrid, Spain, 2014), pp. 894–901.
19. C. Mandery, Ö. Terlemez, M. Do, N. Vahrenkamp and T. Asfour, The KIT whole-body human motion database, in *Proc. 17th Int. Conf. Advanced Robotics* (Istanbul, Turkey, 2015), pp. 329–336.
20. S. Martinez, C. A. Monje, A. Jardon, P. Pierro, C. Balaguer and D. Muñoz, TEO: Full-size humanoid robot design powered by a fuel cell system, *Cybernet. Syst.* **43**(3) (2012) 163–180.
21. D. Harel, Statecharts: A visual formalism for complex systems, *Sci. Comput. Program.* **8**(3) (1987) 231–274.
22. M. Wächter, N. Vahrenkamp, M. Kröhnert, T. Asfour and S. Ottenhaus, The ArmarX statechart concept: Graphical programing of robot behavior, *Front. Robot. AI* **3** (2016) 33.

Santiago Martinez received his Ph.D. in Electric, Electronics and Industrial Automation Engineering from the University Carlos III of Madrid in 2012. He is currently an Associate Professor at the Systems Engineering and Automation Department and he belongs to the RoboticsLab Research Group. He has participated in several EU and National research projects mainly related with Assistive Robotics and Construction Robotics. He is also responsible for the development of the humanoid robot TEO.

Juan Miguel Garcia-Haro received his M.S. from the University Carlos III of Madrid, in 2014. Currently, he is a Ph. D. Candidate and Researcher at the RoboticsLab research group, and works with the Humanoid Robot TEO. His research is focused on whole-body postural control for humanoid robot tasks. Since 2011, he is a member of the RoboticsLab and the ASROB robotic association, and has helped to develop different robotic platforms such as the humanoid robot TEO, the assistive robot ASIBOT or the soft robotic neck from the HUMASOFT project.

Concepcion Alicia Monje received her M.Sc. in Electronics Engineering from the Industrial Engineering School of the University of Extremadura, Spain, in 2001, and her Ph.D. Degree in Industrial Engineering from the University of Extremadura, Spain, in 2006. In September 2006, she joined the University Carlos III of Madrid as a Visiting Professor in the Department of Electronics and Electromechanical Engineering, where she is currently an Associate Professor. Her research focuses on Control Theory and Applications of Fractional Calculus in Control Systems and Robotics. She has been working in these areas for over 15 years, and has published over 100 technical papers mostly related to Control and Robotics.

Carlos Balaguer received his Ph.D. in Automation from the Polytechnic University of Madrid (UPM), Spain, in 1983. From 1983–1994, he was with the Department of Systems Engineering and Automation of UPM as Associated Professor. Since 1996, he has been a Full Professor of the RoboticsLab at the University Carlos III of Madrid. He was the Vicechancellor for Research at the University. His research has included Humanoid and Assistive and Service Robots, among others. He has participated in numerous EU projects since 1989, such as Eureka projects SAMCA, AMR and GEO, Esprit projects ROCCO and CEROS, Brite project FutureHome, IST project MATS, and other numerous 6FP and 7FP projects. He has published more than 200 papers in journals and conference proceedings, and several books in the field of Robotics.