The iCub humanoid robot: an open platform for research in embodied cognition

(Special Session on EU-projects)

Giorgio Metta
Giulio Sandini
Italian Institute of Technology and University of Genoa
Via Morego, 30
16163 Genoa, Italy
+39 010 71781411
{giorgio.metta, giulio.sandini}@iit.it

David Vernon
Khalifa University
P.O. Box 573
Sharjah
david@vernon.eu

Lorenzo Natale
Francesco Nori
Italian Institute of Technology
Via Morego, 30
16163, Genoa, Italy
+39 010 71781420
{lorenzo.natale, francesco.nori}@iit.it

ABSTRACT

We report about the iCub, a humanoid robot for research in embodied cognition. At 104 cm tall, the iCub has the size of a three and half year old child. It will be able to crawl on all fours and sit up to manipulate objects. Its hands have been designed to support sophisticate manipulation skills. The iCub is distributed as Open Source following the GPL/FDL licenses. The entire design is available for download from the project homepage and repository (http://www.robotcub.org). In the following, we will concentrate on the description of the hardware and software systems. The scientific objectives of the project and its philosophical underpinning are described extensively elsewhere [1].

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics – *Commercial robots and applications, Kinematics and dynamics, Manipulators, Operator interfaces, Sensors.*

General Terms

Experimentation, Standardization.

Keywords

Humanoid robotics, cognitive systems, open source, software modularity.

1. INTRODUCTION

RobotCub is a collaborative project funded by the European Commission under the sixth framework programme (FP6) by Unit E5: Cognitive Systems, Interaction and Robotics. It has the two-fold goal of: i) creating an open hardware/software humanoid robotic platform for research in embodied cognition, and ii) advancing our understanding of natural and artificial cognitive systems by exploiting this platform in the study of the development of cognitive capabilities.

The RobotCub stance on cognition posits that manipulation plays a fundamental role in the development of cognitive capability [1-4]. As many of these basic skills are not ready-made at birth, but developed during ontogenesis [5], RobotCub aims at testing and developing this paradigm through the creation of a child-like humanoid robot: the iCub. This "baby" robot will act in cognitive scenarios, performing tasks useful for learning while interacting with the environment and humans. The small (104cm tall), compact size (approximately 22kg and fitting within the volume of a child) and high number (53) of degrees of freedom combined with the Open Source approach distinguish RobotCub from other humanoid robotics projects developed worldwide.

In this paper, we focus on the description of the iCub, both in terms of hardware and software. In particular, we will briefly discuss the rationale of the hardware design, the modularity and reuse of software components, and the consequences of the Open Source distribution policy. RobotCub has, in parallel, the goal of advancing the science and engineering of cognitive systems. This part of the research has been discussed elsewhere in greater detail [4, 6] and will not be reported here.

The hardware of iCub has been specifically optimized and designed somewhat holistically: modularity in this case had to be traded for functionality and overall size. Software, on the other hand, has been designed with modularity and component reuse in mind [7]. Both the hardware and software of the iCub have been released under the GPL and FDL licenses. The mechanical drawings, the electronics, schematics and documentation are available from the RobotCub website. The iCub software is available as well from the same repository together with a basic dynamical simulator of the robot.

Additional initiatives are aiming at promoting the iCub as the platform of choice for research in embodied cognition. Fifteen robots are expected to be delivered by the end of the project (end of 2009) as part of RobotCub and of other EU funded projects.

2. THE ICUB

The iCub has been designed to allow manipulation and mobility. For this reason 30 degrees of freedom (DOF) have been allocated to the upper part of the body. The hands, in particular, have 9 DOF each with three independent fingers and the fourth and fifth to be used for additional stability and support (only one DOF). They are tendon driven, with most of the motors located in the forearm. The legs have 6 DOF each and are strong enough to allow bipedal locomotion.

From the sensory point of view, the iCub is equipped with digital cameras, gyroscopes and accelerometers, microphones, and force/torque sensors. A distributed sensorized skin is under development using capacitive sensor technology.

Each joint is instrumented with positional sensors, in most cases using absolute position encoders. A set of DSP-based control cards, designed to fit the iCub, take care of the low-level control loop in real-time. The DSPs talk to each other via CAN bus. Four CAN bus lines connect the various segments of the robot.

All sensory and motor-state information is transferred to an embedded Pentium based PC104 card that handles acquisition and reformatting of the various data streams. Time consuming computation is typically carried out externally on a cluster of machines. The communication with the robot occurs via a Gbit Ethernet connection.

The overall weight of the iCub is 22kg. The umbilical cord contains both an Ethernet cable and power to the robot. At this stage there is no plan for making the iCub fully autonomous in terms of power supply and computation (e.g. by including batteries and/or additional processing power on board).

The mechanics and electronics were optimized for size, starting from an evaluation and estimation of torques in the most demanding situations (e.g. crawling). Motors and gears were appropriately sized according to the requirements of a set of typical tasks. The kinematics was also defined following similar criteria. The controllers were designed to fit the available space. Figure 5 shows the prototype of the iCub.

2.1 Mechanics

The kinematic specifications of the body of the iCub, the definition of the number of DOF, their actual locations as well as the actual size of the limbs and torso were based on ergonomic data and x-ray images.

The possibility of achieving certain motor tasks is favored by a suitable kinematics and, in particular, this translates into the determination of the range of movement and the number of controllable joints (where clearly replicating the human body in detail is impossible with current technology). Kinematics is also influenced by the overall size of the robot which was imposed *a priori*. The size is that of a 3.5 years old child (approximately 100cm high). This size can be achieved with current technology. QRIO[1] is an example of a robot of an even smaller size although with less degrees of freedom. In particular, our task specifications, especially manipulation, require at least the same kinematics of QRIO with the addition of the hands and moving eyes. Also, we considered the workspace and dexterity of the arms and thus a three degree of freedom shoulder was included. This was elaborated into a proper list of joints, ranges, and sensory requirements at the joint level.

Considering dynamics, the most demanding requirements appear in the interaction with the environment. Impact forces, for instance, have to be considered for locomotion behaviors, but also and more importantly, developing cognitive behaviors such as manipulation might require exploring the environment erratically. As a consequence, it is likely that high impact forces need to be sustained by the robot mechanical structure. This requires strong joints, gearboxes, and more in general powerful actuators and appropriate elasticity (for absorbing impacts). In order to evaluate the range of the required forces and stiffness, various behaviors were simulated in a dynamical model of the robot. These simulations provided the initial data for the design of the robot. The simulations were run using Webots[2] and were later cross-checked by traditional static analysis.

At a more general level, we evaluated the available technology, compared to the experience within the project consortium and the targeted size of the robot: it was decided that electric motors were the most suitable technology for the iCub, given also that it had to be ready according to the very tight schedule of the overall project. Other technologies (e.g. hydraulic, pneumatic) were left for a “technology watch” activity and were not considered further for the design of the iCub.

From the kinematic and dynamic analysis, the total number of degrees of freedom for the upper body was set to 38 (7 for each arm, 9 for each hand, and 6 for the head). For the legs the simulations indicated that for crawling, sitting and squatting a 5 DOF leg is adequate. However, it was decided to incorporate an additional DOF at the ankle to support standing and walking. Therefore each leg has 6 DOF: these include 3 DOF at the hip, 1 DOF at the knee and 2 DOF at the ankle (flexion/extension and abduction/adduction). The foot twist rotation was not implemented. Crawling simulation analysis also showed that for effective crawling a 2 DOF waist/torso is adequate. However, to support manipulation a 3 DOF waist was incorporated. A 3 DOF waist provides increased range and flexibility of motion for the upper body resulting in a larger workspace for manipulation (e.g. when sitting).

The neck has a total of 3 DOF and provides full head movement. The eyes have further 3 DOF to support both tracking and vergence behaviors.

The actuation solution adopted for the iCub is based on a combination of a harmonic drive reduction system (CSD series, 100:1 ratio for all the major joints) and a brushless frameless motor (BLM) from the Kollmorgen frameless RBE series (Figure 1). The harmonic drive gears provide zero backlash, high reduction ratios on small space with low weight while the brushless motors exhibit the desired properties of robustness, high power density, and high torque and speed bandwidths (especially when compared with conventional DC brushed motors). The use of frameless motors permits integration of the motor and gears in an endoskeletal structure that minimizes size,

[1] http://www.sony.net/Fun/design/history/product/2000/sdr-4x.html

[2] http://www.cyberbotics.com/products/webots/webots5.pdf

weight and dimensions. Smaller motors (brushed-DC type) were used for the hands and head joints.

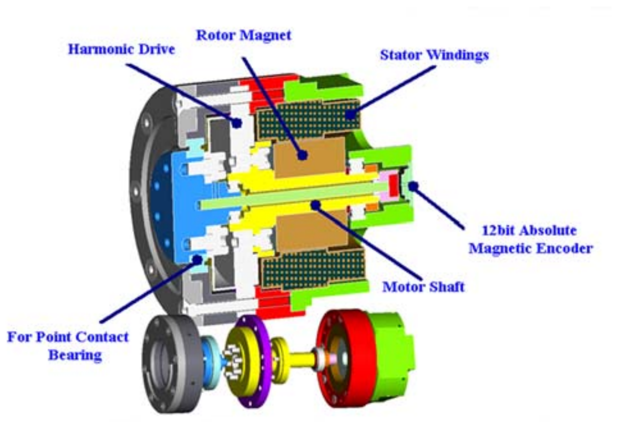

Figure 1: section of the standard brushless motor group of the iCub. Positioning of the motor and gears can be noted (as indicated). Figure from [8]. Note the compact assembly of the frameless motor and harmonic drive gear.

An example on the use of this structure is depicted in Figure 2, which shows the shoulder of the iCub with details of the motor enclosure and tendon-driven pulley mechanisms.

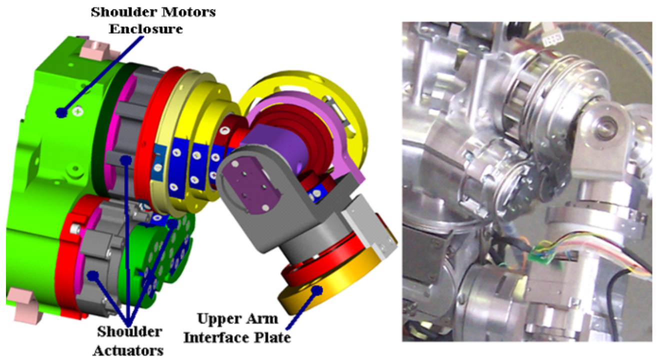

Figure 2: the shoulder of the iCub. Left: CAD schematics. Right: the implementation. Note the three DOF of the shoulder with intersecting axes of rotation and the placement of the actuators in the chest as indicated. 1.75mm steel cables join the movement of the motors with the pulleys actuating the joints.

Certain features of the iCub are unique. Tendon driven joints are the norm both for the hand and the shoulder, but also in the waist and ankle. This reduces the size of the robot but introduces elasticity that has to be considered in designing control strategies where high forces might be generated.

The hand, for example, is fully tendon-driven. Seven motors are placed remotely in the forearm and all tendons are routed through the wrist mechanism (a 2 DOF differential joint). The thumb, index, and middle finger are driven by a looped tendon in the proximal joint. Motion of the fingers is driven by tendons routed via idle pulleys on the shafts of the connecting joints. The flexing of the fingers is directly controlled by the tendons while the extension is based on a spring return mechanism. This arrangement saves one cable per finger. The last two fingers are coupled together and pulled by a single motor which flexes 6 joints simultaneously. Two more motors, mounted directly inside the hand, are used for adduction/abduction movements of the thumb and all fingers except the middle one which is fixed with respect to the palm. In summary, eight DOF out of a total of nine are allocated to the first three fingers, allowing considerable dexterity. The last two fingers provide additional support to grasping.

Joint angles are sensed using a custom designed Hall-effect-magnet pair. In addition room for the electronics and tactile sensors has been planned. The tactile sensors are under development [9].

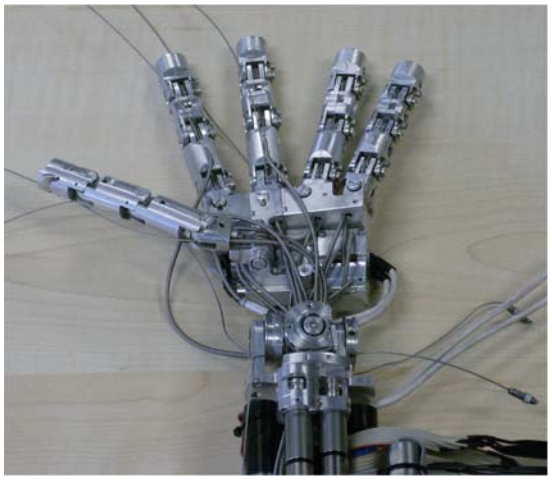

Figure 3: the hand of the iCub, showing the routing of the tendons through the wrist and some of the DOF before full assembly is completed (the palm is missing). Tendons are made of Teflon-coated cables sliding inside Teflon coated flexible steel tubes.

The overall size of the palm has been restricted to 50mm in length; it is 34mm wide at the wrist and 60mm at the fingers. The hand is only 25mm thick.

2.2 Electronics

The generation of motor control signals and sensory data acquisition is fully embedded into the iCub electronics. Further control layers are implemented externally. The interface between the iCub and the outside world occurs through a Gbit Ethernet cable. The robot contains motor amplifiers, a set of DSP controllers, a PC104-based CPU, and analog to digital conversion cards.

The low-level controller cards are of two types for the brushless and the brushed-DC motors respectively. They are based on the same DSP (Freescale 56F807). The controller of the brushless motors is made of two parts (logic and power) and can deliver a current of 6A continuous (20A peak) at 48V. All supply voltages are generated internally. The CAN bus is employed to communicate with the PC104 CPU. Logic and power are

58x42mm each and can control up to two motors. The power stage mounts also a metal heatsink that is then connected to the external shell of the robot for dissipation.

Similarly the controller of the brushed-DC motors is made of two parts. One card acts as power supply; the other contains the CPU and amplifiers to control up to four motors. In this case the maximum continuous current is limited to 1A at 12V.

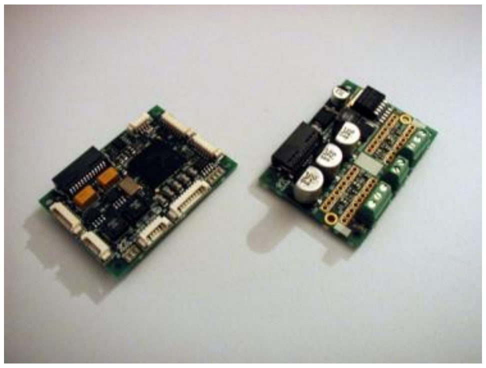

Figure 4: the brushless motor control logic and power amplifier of the iCub. Transistors and heatsinks are not shown. The size of the two PCBs is 58x42mm.

More development is in progress to interface tactile and force/torque sensors as discussed in [8].

2.3 Sensors

Given the size of the iCub, sensors were evaluated for performance but also weight and interface standards. The following table contains the list of available sensors and their status of maturity (i.e. integration into the robot hardware):

Component	**Model/type**	**Notes**
Cameras	PointGrey Dragonfly 2 640x480 30fps	Firewire cameras, support also higher resolution
Microphones	MICRO POM-2746L	Condenser electrect type
Inertial sensors	XSense MTx	3 gyroscopes, 3 linear accelerometers, compass
Force/torque sensors	Custom	Mechanically compatible with the ATI Mini-45
Position sensors	AS5045	12bit, absolute magnetic encoder
Position sensors	Faulhaber	Integrated position sensing for DC motors
Position sensors	Honeywell SS495A	Finger position sensing
Tactile sensors	Custom	Based on the AD7147, capacitive sensing

All sensors are fully integrated apart from the force/torque sensor whose control electronics is still under development and the skin whose entire technology is under testing. More information can be found in [8, 9].

3. SOFTWARE

The iCub software was developed on top of Yarp [7]. RobotCub supported a major overhaul of the Yarp libraries to adapt to a more demanding collaborative environment. Better engineered software and interface definitions are now available in Yarp.

Yarp is a set of libraries that support modularity by abstracting two common difficulties in robotics: namely, modularity in algorithms and in interfacing with the hardware. Robotics is perhaps one of the most demanding application environments for software recycling where hardware changes often, different specialized OSs are typically encountered in a context with a strong demand for efficiency. The Yarp libraries assume that an appropriate real-time layer is in charge of the low-level control of the robot and instead takes care of defining a soft real-time communication layer and hardware interface that is suited for cluster computation.

Yarp takes care also of providing independence from the operating system and the development environment. The main tools in this respect are ACE [10] and CMake[3]. The former is an OS-independent communication library that hides the quirks of interprocess communication across different OSs. CMake is a cross-platform make-like description language and tool to generate appropriate platform specific project files.

Yarp abstractions are defined in terms of protocols. The main Yarp protocol addresses inter-process communication issues. The abstraction is implemented by the *port* C++ class. *Ports* follow the observer pattern by decoupling producers and consumers. They can deliver messages of any size, across a network using a number of underlying protocols (including shared memory when possible). In doing so, *ports* decouple as much as possible (as function of a certain number of user-defined parameters) the behavior of the two sides of the communication channels. *Ports* can be commanded at run time to connect and disconnect.

The second abstraction of Yarp is about hardware devices. The Yarp approach is to define interfaces for classes of devices to wrap native code APIs (often provided by the hardware manufactures). Change in hardware will likely require only a change in the API calls (and linking against the appropriate library). This easily encapsulates hardware dependencies but leaves dependencies in the source code. The latter can be removed by providing a “factory” for creating objects at run time (on demand).

The combination of the *port* and *device* abstractions leads to remotable device drivers which can be accesses across a network: e.g. a grabber can send images to a multitude of listeners for parallel processing.

Overall, Yarp’s philosophy is to be lightweight and to be “gentle” with existing approaches and libraries. This naturally excludes hard real-time issues that have to be necessarily addressed elsewhere, likely at the OS level.

[3] http://www.cmake.org

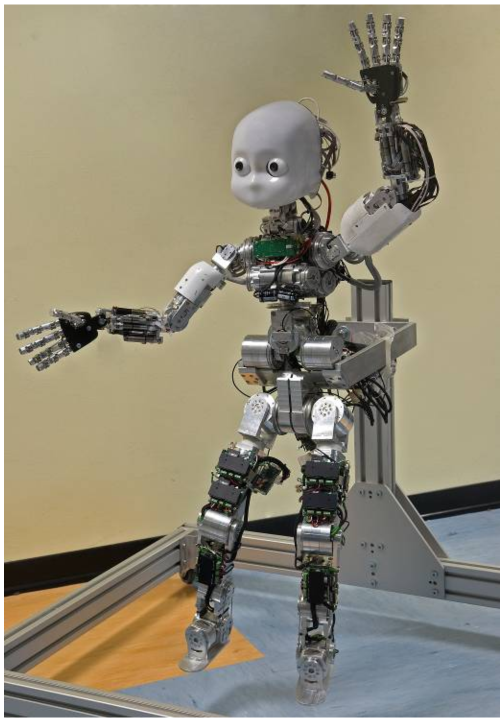

Figure 5: the complete iCub prototype.

3.1 Yarp example

For the purposes of YARP, communication takes place through connections between named entities called *ports*. These form a directed graph, the YARP Network, where ports are the nodes, and connections are the edges. Each port is assigned a unique name, such as "/icub/camera/right". Every port is registered by name with a "name server". The goal is to ensure that if you know the name of a port, that is all you need in order to be able to communicate with it from any machine. The YARP name server converts from symbolic names to all the details necessary to make a connection with a specific resource. The YARP name server is designed to be easily used by clients who are not themselves using the YARP libraries or executables.

The purpose of ports is to move data from one thread to another (or several others) across process and machine boundaries. The flow of data can be manipulated and monitored externally (e.g. from the command-line) at run-time. It can also be accessed without using the YARP libraries or executables, since the relevant protocols are documented.

A port can send data to any number of other ports. A port can receive data from any number of other ports. Connections between ports can be freely added or removed, and may use different underlying transports. The use of several different transports and protocols allows us to exploit their best characteristics. TCP is reliable; it can be used to guarantee the reception of a message. UDP can be faster than TCP, but without guarantees. Multicast is efficient for distributing the same information to large numbers of targets. Shared memory can be employed for local connections.

Figure 6 shows a very simple network of ports for a visual tracking application. Machine 1, in this example, grabs images which are sent to another application (the tracker proper). The output of the tracker consists of two parts: the image coordinates of the tracked object and an image with a graphic overlay showing how good the tracker is doing. The output is sent to a control process on another machine (Machine 2) and for visualization to yet another machine. Different protocols can be used for reasons of efficiency.

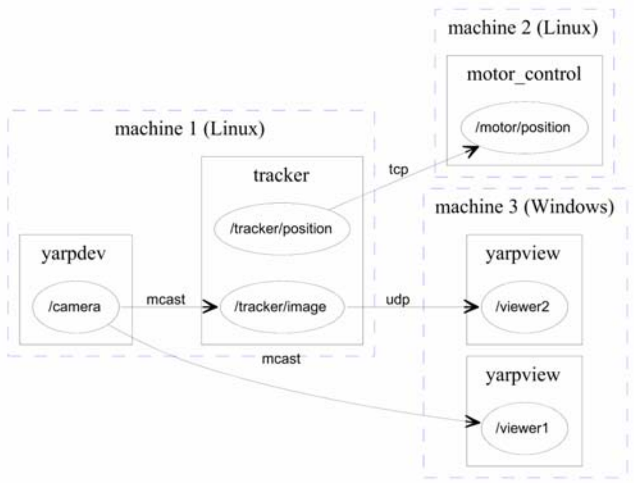

Figure 6: example of a YARP network for a simple visual control loop.

4. OPEN SOURCE ROBOTICS

RobotCub is Open Source both for software and hardware. While the phrase "Open Source software – OSS" is clear, "Open Source hardware" might sound strange, but in fact it is a plain transfer of the open source philosophy to the entire design of the RobotCub platform. The design of the robot started from the preparation of specifications (e.g. estimation of torque, speed, etc.), a typical 3D CAD modeling, and eventually in the preparation of the executive files which are used to fabricate parts and for assembly. Without good documentation it is very complicated to build and assemble a full robot. This means that documentation (as for software) is particularly important.

The CAD files, in some sense, can be seen as the source code, since they are the "preferred form of the work for making modifications to it", in the language of the GPL. They get "compiled" into 2D drawings which represent the executive drawings that can be used by any professional and reasonably well-equipped machine shop either to program CNC machines or to manually prepare the mechanical parts. This compilation process is not fully automated and requires substantial human intervention. There is a clear dependency of the 2D drawings on the original 3D CAD model. To enable the same type of virtuous development cycle as occurs in open source software, the 3D CAD is required, since changes happen in 3D first and get propagated to 2D later. In addition, assembly diagrams, part lists, and all the material produced during the design stage should be included to guarantee that the same information is available to new developers.

One difference between software and the hardware design is that there are currently no effective formats for interchange of 3D models. Proprietary systems such as SolidWorks and Pro/E can import and export a range of formats, but going from one to another is lossy, destroying the information needed for production and leaving just the basic geometrical shape. So in practice, designs are tied to tools produced by a particular vendor, and interoperability between hardware design tools is limited. In RobotCub we were forced to choose a specific set of tools for mechanical and electronic CAD and future upgrades will have to strictly adhere to these standards. Due to the absence of open source professional design tools, RobotCub uses proprietary products. This is an unfortunate situation, but there is no practical alternative at the moment. The “C++” and “gcc” of CAD do not exist yet.

As a practical matter, the simple duplication of RobotCub parts does not require the use of any of these tools since we provide all executive drawings and production files (e.g. Gerber files for the PCBs). For modification, the design tools are somewhat expensive (although educational discounts or educational releases exist). Free of charge viewers are currently available for all file types in question.

For RobotCub, we decided to license all the CAD sources under the GPL which seems appropriate given their nature. Associated documentation will be licensed under the FDL. These will be made available through the usual source code distribution channels (e.g. repositories, websites).

5. CONCLUSION

The design process of RobotCub has been a distributed effort as for many open source projects. Various groups developed various subcomponents and contributed in different ways to the design of the robot including mechanics, electronics, sensors, etc. In particular, a whole design cycle was carried out for the subparts (e.g. head, hand, legs) and the prototypes that have been built and debugged. The final CAD and 2D drawings were discussed and then moved to the integration stage. Clearly, communication was crucial at the initial design stage to guarantee a uniform design and a global optimization.

The distributed design broke down at the integration stage where the industrial partner (Telerobot Srl. – Genoa) stepped in to carry out integration, verification and consistency checks. The design and fabrication of the control electronics was also subcontracted to a specialized company. It is important to stress the collaboration with industry for a project of this size and with these goals and requirements. For many reasons building a complete platform involves techniques and management that is better executed following industrial standards. One example that applies to RobotCub is the standardization of the documentation.

A further strategy used in RobotCub is that of building early. Each subsystem was built and copied as soon as possible. In several cases debugging happened either because the copies of the robot did not work as expected or because easy-to-fix problems were spotted. Sometimes the documentation had to be improved. Unfortunately, this strategy was applied less extensively to some of the subparts which are or were still under design and debugging. The design stage will be completed with the realization of the fifteen copies of the iCub.

This will further test the documentation and in general the reliability of the overall platform including software, debugging tools, electronics, etc. The first release of the iCub will be consolidated after this final fabrication stage.

The actual design of the robot had to incorporate manipulation by providing sophisticated hands, a flexible oculomotor system, and a reasonable bimanual workspace. On top of this, the robot has to support global body movements such as crawling, sitting, etc. These many constraints were considered in preparing the specifications of the robot and later on during the whole design process.

Both the iCub design and its software architecture are distributed as Open Source. This is not enough to guarantee success. Additional initiatives are required. RobotCub is giving away six copies of the iCub to the winners of an Open Call for proposals to use the iCub (recently concluded). In addition a structure called the Research and Training Site (RTS) has being created to support visiting researchers to work on the iCub prototypes in Genoa.

6. ACKNOWLEDGMENTS

The authors would like to thank the RobotCub Consortium. The authors were supported by European Union grant RobotCub (IST-2004-004370) and by euCognition (FP6 Project 26408). Paul Fitzpatrick is gratefully acknowledged for the continuous support to Yarp.

7. REFERENCES

[1] L. Fadiga, L. Craighero, and E. Olivier, "Human motor cortex excitability during the perception of others' action," *Current Biology*, vol. 14 pp. 331-333, 2005.

[2] L. Fadiga, L. Craighero, G. Buccino, and G. Rizzolatti, "Speech listening specifically modulates the excitability of tongue muscles: a TMS study," *European Journal of Neuroscience*, vol. 15, pp. 399-402, 2002.

[3] G. Rizzolatti and L. Fadiga, "Grasping objects and grasping action meanings: the dual role of monkey rostroventral premotor cortex (area F5)," in *Sensory Guidance of Movement, Novartis Foundation Symposium*, G. R. Bock and J. A. Goode, Eds. Chichester: John Wiley and Sons, 1998, pp. 81-103.

[4] D. Vernon, G. Metta, and G. Sandini, "A Survey of Cognition and Cognitive Architectures: Implications for the Autonomous Development of Mental Capabilities in Computational Systems," *IEEE Transactions on Evolutionary Computation, special issue on AMD*, vol. 11, 2007.

[5] C. von Hofsten, "On the development of perception and action," in *Handbook of Developmental Psychology*, J. Valsiner and K. J. Connolly, Eds. London: Sage, 2003, pp. 114-140.

[6] G. Sandini, G. Metta, and D. Vernon, "RobotCub: An Open Framework for Research in Embodied Cognition," presented at IEEE-RAS/RJS International Conference on Humanoid Robotics, Santa Monica, CA, 2004.

[7] P. Fitzpatrick, G. Metta, and L. Natale, "Towards Long-Lived Robot Genes," *Journal of Robotics and Autonomous Systems, Special Issue on Humanoid Technologies*, vol. 56, 2008.

[8] N. G. Tsagarakis, G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, J. Santos-Victor, A. J. Ijspeert, M. C. Carrozza, and D. G. Caldwell, "iCub – The Design and Realization of an Open Humanoid

Platform for Cognitive and Neuroscience Research," *Advanced Robotics*, vol. 21, 2007.

[9] M. Maggiali, G. Cannata, P. Maiolino, G. Metta, M. Randazzo, and G. Sandini, "Embedded Distributed Capacitive Tactile Sensor," presented at The 11th Mechatronics Forum Biennial International Conference 2008, University of Limerick, Ireland, 2008.

[10] S. D. Huston, J. C. E. Johnson, and U. Syyid, *The ACE Programmer's Guide*: Addison-Wesley, 2003.