# Design an evaluation of RoboCup humanoid goalie

Juan F. García, Francisco J. Rodríguez, Camino Fernández, and Vicente Matellán

*Abstract*—In this article we describe the ethological inspired architecture we have developed and how it has been used to implement a humanoid goalkeeper according to the regulations of the two-legged Standard Platform League of the RoboCup Federation. We present relevant concepts borrowed from ethology that we have successfully used for generating autonomous behaviours in mobile robotics, such as the use of ethograms in robotic pets or the ideas of schemata, or the use of fixed actions patterns to implement reactivity. Then we discuss the implementation of this architecture on the Nao biped robot. Finally, we propose a method for its evaluation and validation and analyse the results obtained during RoboCup real competition, which allowed us to test first hand how it worked in a real enviroment.

*Index Terms*—reactive, attention, vision, humanoid, schema

## I. ROBOTICS CONTROL ARCHITECTURES IN LITERATURE

GENERATING autonomous behaviours in mobile robotics is really a complex problem. In this section we present a non in-depth outline about those robotics control architectures close to our research. We are going to be neither exhaustive, mainly because it would be impossible to describe all control architectures in just one section, nor hierarchycal, since there would be too many criteria to take into account.

Summarizing the complex history of the AI, we can state that two main schools of thought have coexisted, the subsymbolic one, interested on modeling intelligence in a level similar to neurons; and the symbolic AI, which models knowledge and planning in data structures that make sense to the programmers that build them. Another way of explaining the difference between both schools is referring to their foundations: Biology in the subsymbolic AI, and cognitive psychology in the symbolic AI [9]. Hybrid systems are a pragmatic approach, where ethology based systems can be included because they sucesfully integrate deliberative and reactive perspectives in natural autonomous systems.

Hybrid architectures intend to combine reactive and deliberative control, and usually consist of three components: a reactive layer, a planner, and a layer that links the other two. Well known examples of this kind of architecture are AuRA[1], which integrates a $A^*$ planner with schema-based controllers [2], and PRS (Procedural Reasoning System) [6] based on least commitment via plan elaboration postponement.

Teleo-Reactive (TR) program formalism proposed by Nilsson [28] falls also under Hybrid control category. Teleo-reactivity in dynamic environments implies a short sense - act cycle. Robots are able react rapidly to communly occuring situations (such as crash avoidance or refueling) but their behaviours are also influenced by their goals (hence "teleo").

Juan, Francisco, Camino, and Vicente are with Departamento de Ingeniería Mecánica, Informática y Aeroespacial Escuela de Ingenierías Industrial e Informática, Universidad de León, 24071 León Web: http://robotica.unileon.es e-mail:{jfgars, fjrodl, camino.fernandez, vicente.matellan}@unileon.es

Opportunistic architectures are a subset of the hybrid architectures that take its name from Barbara Hayes-Roth approach to hybrid control [7]. The agents architecture on her system was also made up by three components: an event-triggered reactive level, an strategic planner, and a control process in charge of matching triggered actions with the generated plan. A similar architecture is used in O-Plan [8] where the term "agent" is used to name each of the three modules of the system.

Another implementation of these ideas are RAPs (Reactive Action Packages) proposed by Firby [5]. RAPs were designed to allow the reactive execution of symbolic plans. In this way, each RAP defines different alternatives of execution depending on the environment, and an agenda is used to select the next action to execute. Another approach is the TCA (Task Control Architecture) by Simmons [18], which integrates symbolic plans with real-time restrictions as well as reactive behaviors triggered as exceptions.

In the RoboCup domain, Saffiotti [21] presented the *ThinkingCap* architecture. This architecture was based in a fuzzy approach, extended in [24]. The perceptual and global modelling components managed information in a fuzzy way and were used to generate the next actions.

Also in the RoboCup domain, the architecture proposed by Manuela Veloso et al[22] shows an hybrid hierarchical behaviour-based architecture. This architecture was divided in levels. The upper levels set goals that the bottom level had to achieve using information generated by a set of virtual sensors, which were an abstraction of the actual sensors.

Another successful approach in the RoboCup was the one used in the German Team[23] that proposed a four levels architecture: perception, object modelling, behaviour control, and motion control. The execution starts in the upper level perceiving the environment and finishes at low level sending motion commands to actuators. The behaviour level was made up of several basic behaviours implemented as finite state machines. These finite state machine were written in XABSL language [25], that was interpreted at runtime and let change and reload the behaviour during the robot operation.

The foundation of the work presented in this paper is JDE (*Jerarquía Dinámica de Esquemas*) [9], an etho-inspired architecture where behaviour is organized as a dynamic hierarchy of independent schemata.

Many other concepts borrowed from Ethology have been used in robotics. For instance, homeostasis, proposed as mechanisms for action selection by T. Tyrrell [16] ; or the movements of bees [14], capable of returning to their beehive using the sun as compass for global navigation; or the flies balancing optical flow in both eyes to local navigation [4]; gestalt perception, and the use of visual perceptive invariants, as the ones discovered in the cormorant fishing [15], that can

make easier the goal of developing robotic behaviours, etc. These works have also been applied to modern humanoids [3].

Besides the theoretical description of the architecture, which we will make in the next section, we made a development on a robot in a real environment. The choosen place was the RoboCup (Robotic soccer WorldCup) is an international research and education initiative, which has put forward a standard problem to promote the research on artificial intelligence and intelligent robotics.

In particular, the work described in this paper has been tested during the German Open 2009[1] in April 2009 and Robocup 2009[2] in June-July 2009. In this league all teams use the same hardware platform, the Nao robot (see figure 1). This robots are manufactured by Aldebaran Robotics, so the focus of this competition is on the software controlling the robot.
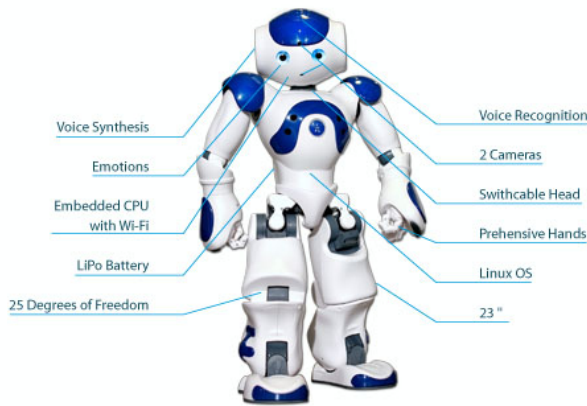


Fig. 1.    Nao robot (figure copyrighted by Aldebaran Robotics)

Nao robot is a 21 degrees of freedom humanoid, whose height is 57 cm. and its weight is around 4.5 Kg. It has two 30 fps video cameras located in the forehead and in the mouth, each one with a maximum resolution of 640x480, but they cannot be used simultaneously. The switch between cameras takes too long and the field of view is scarcely overlaped so they are not capable of stereo vision.

Control is managed onboard using a x86 AMD Geode chip at 500 MHz, 256 MB of SDRAM memory and a standard 1 Gb in flash memory that can be upgraded. It also has got WiFi (802.11g) and Ethernet connections. Concerning the sensors, apart from the cameras, it has 2 gyroscopes and 3 accelerometers, 2 bumper sensors in the feet, and 2 ultrasonic sensors in the chest.

The rest of the paper is organized as follows. Second section describes the architecture we propose. In the third section we present a software implementation for our architecture. In the fourth section, we propose a method to analyse and validate our proposal. Finally, in the last section, the results obtained with this architecture and its performance in the RoboCup

[1]http://www.robocup-german-open.de/en

[2]http://www.robocup2009.org/

German Open and in the Robocup 2009 Graz are analysed and also future works are enumerated.

## II. AN ETHOLOGICAL INSPIRED ARCHITECTURE

Our architecture is based on etholgical principles that exhibit the same features of the hybrid ones previously described, that is, deliberative and reactive capabilities. The two main principles of this architecture are the decomposition of the control problem into behavioural units named components, and the generation of behaviour by building dynamic hierarchies. Both are detailed in next sections.

### A. Components

Our approach is based on the assumption that complex behaviour can be obtained by combining simpler "components" inspired by ethological schemata as defined in [10]. These components perform a specific task in an iterative way and at a controlled frequency. They may send commands to actuators, process data from sensors, or activate/deactivate and modulate other components creating a hierarchy.

When activated, a component creates its data and processing structures and starts its iterations. It can keep its state from one iteration to another or change it depending on its functionallity and the system stimuli (internal or environmental information). When a component is deactivated, all its descendants (all the components the currently component becoming inactive had activated) must also be finished.

A group of components which perform subroutines of the same task are grouped in so called controllers which functionality is explained in section II-C and their implementation in section III.

### B. Dynamic Hierarchy

Components are organized in hierarchy in order to generate more complex behaviours. High level components activate low level components, and all of them run concurrently. The hierarchy is dynamic in the sense currently active modules will be different depending on the situation. Only one schema per level can be active at any given time and, before activating any component, an ancestor of it in the inmediatly superior level has to be already active. If a component is deactivated, all its descendants will also become inactive. Each schema or a whole branch of linked schemata can be activated (or deactivated) at any given time to achive the desired functionality, completly deactivating a previously working set of schemata if nessesary. This makes an improvement to the initial JDE assumtion which establishes that every single schema in a certain active hierarchy has to be deactivated one by one before starting a new one.

Components use a common shared memory space to read its inputs and write its outputs. The upper level component connects the output with the inputs of the modules it activates. This way a low level component could be reused by another high level components which could decide to connect the low level components in a different way. All these inputs and outputs define the system information flow, which basically

consists of internal (component generated) or external (from the environment) stimuli. All components output are then interal stimuli, while their input can be either an internal or external stimuli depending where it comes from.
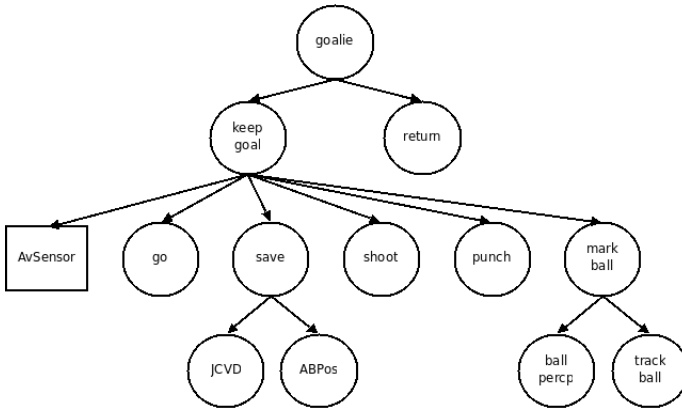


Fig. 2. Goalkeeper modules

Figure 2 shows an example of hierarchy, the goalkeeper behaviour schemata.

### C. Controllers

As already explained in section II-A, a controller is a group of schemata which perform subroutines of the same task. The components are grouped to simplify the overall structure of the architecture: it is an effective way to reduce the information flow present in the system.

Information, as we explained in the previous section, consists of external or system internal stimuli wich would cause either activation or deactivation of a given component. The main reason to have controllers and not individual ungroupped components is not having to consider an input and output information channel per component of the system at any given time. Instead, information is brought to each controller, and it will then be redirected to the concrete component which is desgined to react to it.

Basically, a controller oversees the activation and deactivation of its components redirectering the information flow it receives and produces. Each controller is able to communicate with other controllers coexisting in the system to which it is directly connected the samw way isolated components do.

Besides the conceptual simplicity explained, there is no real difference among a controller and a group of components. We describe the controllers we use, its functionality ,and its implementation in section III.

### D. The architecture working

The presented architecture shows both deliberative and reactive properties, so it is a hybrid architecture in the classic definition. The set of all possible conections among components and their organization in different levels, as shown for example in figure 2, give the system its deliberative nature.

The architecture is reactive during the hierarchy activation phase previously explained in II-B: the set of active components will vary depending on the situation, with only those useful for the current behaviour being active. We will give two examples of its reactive nature in next section.

Please note that even if the hierarchy defined by active schemata in a given situation is dynamic (varies depending on the task at hand) - reactive behaviour - the available conections among components and their organization in levels is fixed and previously stablished - deliberative architecture -.
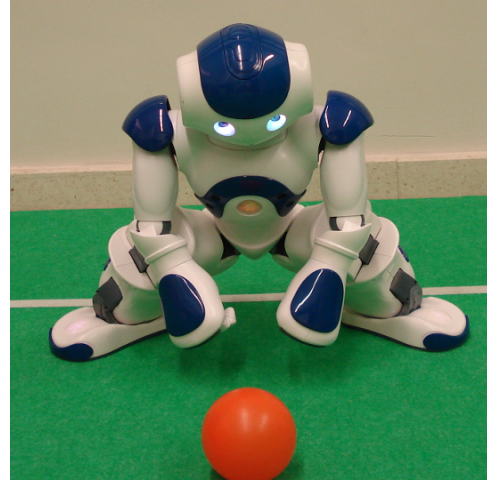


Fig. 3. JCVD defensive movement schema implemented on real Nao robot

## III. IMPLEMENTATION

We are interested in testing our architecture in order to prove its functionality. To do so we have chosen to model a goalkeeper behaviour.

### A. Components

The components are the ethological schemata which model all the posible actions the goalkeeper needs. We have the following components (with each component's name being pretty much self-explanatory about their functionality):

- *Goalie*: Represents all kind of high level behavioural decisions which a goalkeepr would perform during a match, either specific to its role (eg: pucnch out the ball), or not (perception tasks).
- *KeepGoal*: Represents all kind of high level behavioural decisions which are specific to a goalkeeper's role.
- *MarkBall*: Tryes to keep the ball inside the robots visual field.
- *BallPerception*: Looks for the ball in a given image.
- *TrackBall*: Moves the robots head so that the ball stays in the center of its visual field.
- *Go*: Makes the robot walk.
- *Return*: Makes the robot walk to the center of its keep.
- *Save*: Performs a defensive move to try to stop the ball.
- *JCVD*: A wide-area but slow defensive move intended to prevent a goal.
- *ABPos*: A fast but small-area defensive move intended to prevent a goal.
- *Shoot*: Performs a kick to clear the ball.
- *PunchOut*: Punches out the ball.

*B. Controllers and NaoQi Layer*

Task related components are grouped into controllers. The controllers we have implemented, its main functionality, and the components they include are:

- `Goalkeeper Controller`: Takes high level decisions about what to do at any given time: look for the ball, move or try to prevent a goal. Includes *Goalie* and *KeepGoal* components.
- `Scanner Controller`: Moves the robot head in order to look for the ball and gets and analises images from the robots camera. Includes *MarkBall*, *BallPerception* and *TrackBall* components.
- `Walk Controller`: Allows the tobot walk in different directions. Includes *Go* and *Return* components.
- `Save Controller`: Performs defensive positions intended to stop or clear the ball. Includes *Save*, *JCVD*, *ABPos*, *Shoot* and *PunchOut* components.

Those controllers have been used, as already introduced in section II-C, to reduce the system complexity. By grouping components which take part in the same task we reduce the amount of information channels to be considered at any given time. For instance: *BallPerception* and *TrackBall* components are meant to work with visual information (the first one will look for the ball in any image obtained by the robot camera while the second one will try to center it in the field of view once it has been found). There is no reason then to use two different information channels carrying the same visual information, so we group them in a controller which we call `Scanner Controller` which will receive this information and then redirect it to the component which actually needs it.

To be able to control the robot, we will use the software layer it provides, called NaoQi. NaoQi is a propetary SDK which allows us to access robot sensors and actuators by using the modules it provides. We can not consider that NaoQi modules define a real controller since they implement very different functions, from internal memory management to servo motors control, and thus are not really task-related. However, for simplicity reasons, we will represent all these modules together grouped in what we call `NaoQi Layer`. Also, the only conceptual difference between NaoQi modules and the rest of our components is that components in the NaoQi layers are platform specific, that is, they are part of Nao robot's software. The hierarchical relation between all the controllers can be seen in figure 4.

*C. Hierarchy*

The whole static hierarchy, which consists of all implemented components, can be seen in figure 2, and the controllers which they are part of are shown in figure 4. This static hierarchy represents the deliberative nature of the architecture. The components are organized in levels, with those realted to high level tasks occupying the top ones while other more being in the lower levels. The lines connecting components represent the hierarchical relation between them.

The hybrid nature of our architechture can be better ilustrated by two examples of generation of autonomous behaviour for our RoboCup goalkeeper.
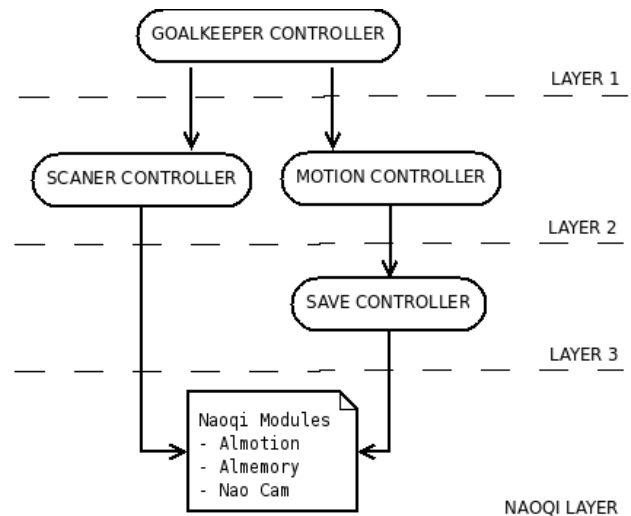


Fig. 4.   Goalkeeper controllers

*Example I.* Let's assume only scanning and basic saving positions modules are available during a real match (deliberative architecture). Given this situation, the goalkeeper would just activate the *TrackBall* component (after activating the needed ascendant schemata to reach it) - reactive architecture - (see figure 2), thus activating *KeepGoal* and *MarkBall* on its way down to it. It would also eventually try to stop it if it comes too close to the keep by going down the hierarchy activating *KeepGoal*, *Save* and finally *ABPos* (a static defensive position)..

*Example II.* Let's suppose all schemata are available during the match. In this case, the goalie, once the ball has been found, would perform side steps to position itself in front of the ball, activating *Keep_goal* and *go* schemata. It could even clear the ball activating the *punch-out* component (which is also connected to *Keep_goal*) if the ball comes close enough.

We have two videos[34] in our web that show these examples working in a real Nao humanoid. Both videos show the whole tree of components, with the active components displayed in a ligther color. The Nao robot appears to the left of the media player, performing each action enumerated in both scenarios, while we can observe its internal architechture displayed at the right side. The relation among components and the dynamic hierarchy resulting from their synchronization are also represented: some schemata are activated when required by the situation while others no longer neccesary become inactive.

*D. Real world restrictions*

When implementing our architecture in a real robot some issues were raised. For instance, not all actions can be instantly stopped (specially those related to movement) to start a new one. This affects not only robots also humans: just imagine you are running and suddenly decide to lay on the floor; you better slow down and stop moving before trying to do so or

---

[3] http://robotica.unileon.es/mediawiki/videos/save.swf

[4] http://robotica.unileon.es/mediawiki/videos/movementAndSave.swf

you will en up rolling on the floor. Applied to our component based design, this means component deactivation will always have a time cost. It is not possible to model this cost because it depends on the current situation.

In most cases, deactivation times are so brief that can be ignored, such is the case of decision related componente like *KeepGoal*, *Save*, *MarkBall* or *TrackBall* (not much time is neccesary to decide you are no longer interested in defending your keep or tracking the ball). However, time cost for movement related ones like *Go*, *ABPos* or *JCVD* (see figure 2) are not negligible, as shown by figure 5. Althought these time costs may seem to high, imagine for a moment the time it would take to a human to stop running, fall to the floor to try to stop the ball and then get up and start running again.

This actions performs in a real robot on a real enviroment offers this results which are not son long like their appears, if we compare with a human (non professional athletes) it is similar to a person getting up in a normal situation.

We need to perform some actions before fully deactivating any of these components so that the robot is not left in an unstable position and thus becomes suitable to fall. We could consider that some sort of cooldown timer is set preventing any new schema activation until the last deactivated schema ensures the robot has reached a stable state. As a result, all schemata have an asociated deactivation cost.
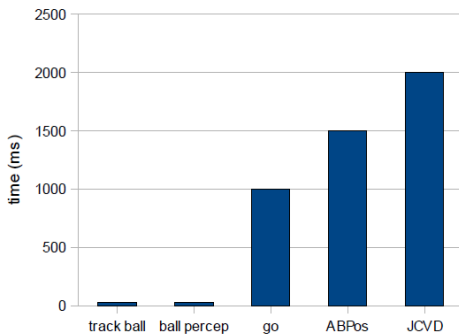


Fig. 5. Time cost for some schemata

Taking these times into account, some sort of high level deliberative mechanism is neccesary for behaviour planning: It is neccesary to evaluate the advantages (goal achived) and drawbacks (in terms of time cost) to deactivate a schema in order to activate a new one. Should I really stop running to comb my hair if I am running to try to catch the bus?, should I stop running to tie my shoes if I am, again, trying not to lose the bus? The answer to first question is obviously "no" since my goal is to get in time before the bus leaves, but it would probably be "yes" to the second one, since it may not be worth taking the risk of falling.

In the robotics world, and specifically in the Robocup environment, we also have plenty of situation which illustrate this kind of situations. For instance, imagine the robot is moving sideways (see figure 6, (1)). When close enough to the ball, the `Goalkeeper Controller` decides it should stop and try to block it by using a fast defensive move (*ABPos* or *JCVD*). The *Go* schema should be deactivated, which would

consume 1 second (see figure 5). As soon as *go* is inactive, *Save* is activated, and then *JCVD* becomes active too (labeled as (2) in figure 6). If the ball suddenly moves away from the goalie (for whatever reason), it will have to stand up and move again. *JCVD* schema would be deactivated, which would consume 2 seconds (it takes some time to get up from the floor). Then, *Save* schema would get inactive (barely instantly since it is just a decision related schema) and finally *go* could be reactivated (see (1) in the same figure). So basically, a "move - stop - save - get up - move" sequence would take more than 3 seconds to perform in reality (without taking into account the time the save would take per-se), while theorically those times were neglected.

It is important then to evaluate advantages and disadvantages of a decision (in terms of schemata deactivation times) before putting it to practice. This was one of our flaws in the last Robocup competition held in June 2009 at Graz, where our controllers didn't have a deliberative mechanism to evalute these costs. For instance, there was a match in which we were scored a goal after our goalkeeper decided to deactivate the defensive *JCVD* component to get up and move in order to better position itself between the ball and the net; it took it too much time to get up, move and go back to its *JCVD* defensive position, leaving space between itself and the keep post while moving, which the rival team used to score a goal.

We are currently working into adding such functionality to one of the components of our `Goalkeeper Controller`.

## IV. ARCHITECTURE ANALYSIS AND VALIDATION

For the Standard Platform League, with the Naos being a relatively new addition, the level of play of many teams is not yet that sophisticated. Our goalkeeper was not called upon to save many goals and so it is difficult to assess its effectiveness relaying just on the results obtained during the championship. It is also difficulty to assess how well would the robot perform without this architecture in place since it is the only one fully tested during our research. Keeping this limitations in mind, we are trying to evaluate the architecture the most objective way possible from three different points of view: quality, development, and performance.

To evaluate these criteria we review all points presented in the "4+1 View Model of Architecture" by Krutchen [26], using the norm ISO 9126, an international standar for software quality evaluation:

1) **Logic level**. High level programming allowed by schemata and behaviour units usage and low level details being hidden thanks to NaoQi both improve abstraction. The architecture makes it easier to understand already developed behaviours and actions and simplyfies the process of adding new ones, so it complies with "usability" characteristic of ISO 9126. Although it doesnt directly guarantees efficiency, it makes it easier to achive it since every module can be tested and improved its own.

2) **Processes level**. The goalkeeper behaviour is splited in different levels, with every level performing actions independently form the rest (movement, vision, etc.).
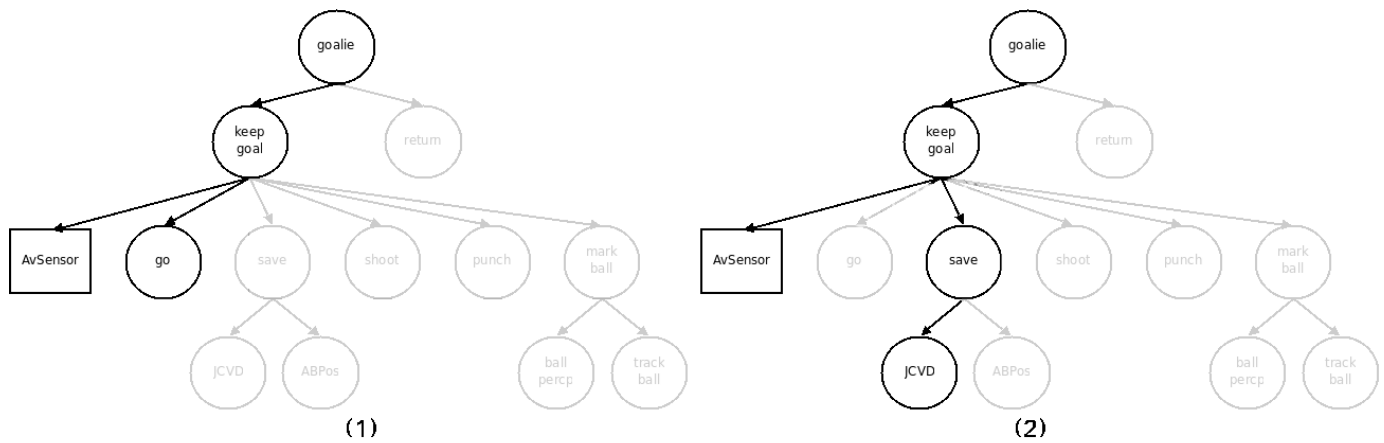
Fig. 6.   Schemata active during a "move (1) - save (2) - move (1)" sequence

This makes concurrency much easier and at the same time it simplyfies the process of adding a new beahviour (for instance, a new kick o new scan mode) or even a whole new controller (for instance, a localization controller) without interfering with the already existing ones. ISO 9126 "Security", "interoperatbility", and "stability" cathegories are then maximized when using this architecture.

3) **Development level**. To evaluate development advantages, that is, how this architechture makes the programming of the Nao easier or how behaviours are more quickly developed when using it, we use cocomo (COnstructive COst MOdel), a mathematical empirical model for software costs stimation [27]. ¿Poner tabla sin más con y sin arquitectura?.¿Quitar lode cocomo?¿calcularlo?

4) **Hardware**. The most interesting aspect of our architecture about hardware is that all paltform specific calls and functions are contained inside NaoQi layer. Since this layer is developed and maitained by the Nao's company (Aldebaran), hardware optimized usage is taken outside of the architecture and solely relies on their external development. The existance of the NaoQi layer also means that we could use this very same implementation except for that layer for any other robot, which complies with the "potability" cathegory of ISO 9126.

5) **Performance of the four previous levels when working together**. The best way to evaluate performance of architecture as a whole is put it to the test. For that reason, in section IV-A a battery of goalkeeper specific tests is proposed. The matches played during Hannover German Oppen and Graz Robocup are also a good benchmark themselves and the result obtained are reviewed in section aún por determinar.

hay que enlazar lo anterior con las pruebas más claramente entiendo que "lo anterior" se refiere al artículo en general. De ser así, creo que lo mejor es añadir un parrafo diciendo que pretenden medir concretamente las pruebas y por qué. Creo que queda mejor ponerlo en el apartado de pruebas, el siguiente

### A. Goalkeeper behaviour specific tests

The architecture described intends to model a goalkeeper behaviour. The best way to test its elements, that is, hybrid control based on reactive and deliberative actions, is to check how well does a robot with an implementation of it perform the goalkeeper rol. The most relevant tasks fulfilled by a goalkeeper are then put to the test: fast movement over the field, ball perception, ball tracking, proper positioning (deliberative part), and goal saving (reactive part). Since movement speed, and movement in general, is dependant on NaoQi primitives, as it was previously stated in section III-B, this particular characteristic is not taken into account. In order to test the rest of them, the following benchmark is proposed:

Only one side of the field is used. A set of markers are placed on it: they are placed at 4 rows and 5 columns, first row at 50 cm from the keep line and the rest 50 cm from the previous one. The markers are labeled as $(m, n)$, with $m$ (rows) ranging from 0 to 3 and $n$ (columns) from 0 to 4, with marker $(0, 0)$ being the one at the top-left corner when looking at the keep and marker $(0, 2)$ being positioned exactly in front of the robot, at 50 cm from the keep's line. All markers in the same row are also positioned 50 cm from the adjacent ones. Fig. 7 shows a top view of the benchmark proposed.

A 130 cm length ramp with variable inclination from 6% to 24% is used to perform the test. The ball can be placed on three different positions on top of it, at 50, 100 and 130 cm from its lower part, which is placed on every marker in the field. For every marker, the angle to the keep can be modified from $-\frac{\pi}{4}$ rad to $\frac{\pi}{4}$ rad with steps of $\frac{\pi}{16}$ rad, which results in nine possible orientation for every marker: $-\frac{\pi}{4}$ rad, $-\frac{5\pi}{16}$ rad, $-\frac{\pi}{8}$ rad, $-\frac{\pi}{16}$ rad, 0 rad, $\frac{\pi}{16}$ rad, $\frac{\pi}{8}$ rad, $\frac{5\pi}{16}$ rad, and $\frac{\pi}{4}$ rad. This allows us to test nine different orientations and three different ball starting position for every marker; since twenty markers are placed in the field, it means the benchmark includes a total of 540 different shots. Shot speed can also be modified depending on the ramp's inclination. To perform the test, the goalkeeper stands in the middle of the keep when started. The ramp is then positioned at every marker (different ball positions on top of it, and different angles and inclination are used for all of them). Once the goalkeeper locates the ball and
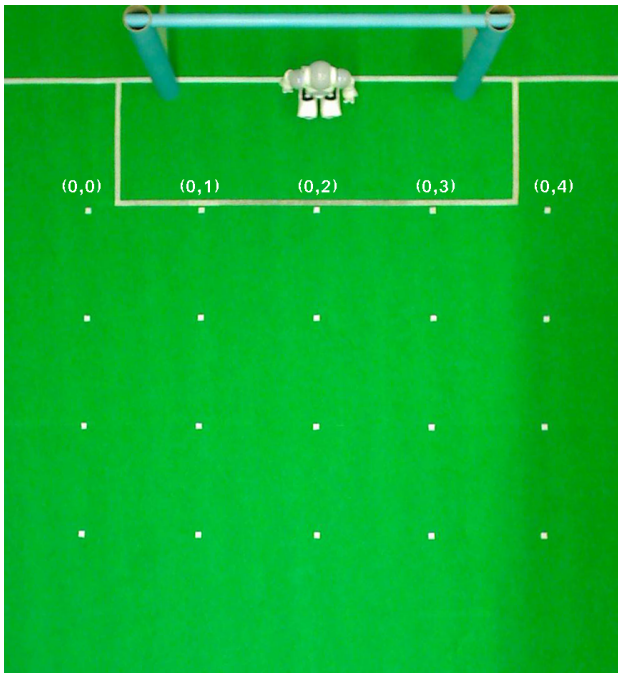
Fig. 7. Markers used for testing

starts tracking it, the ball is realeased from its initial position and thus approaches the keep. For every shot, the goalkeeper sucess rate in every category evaluated (ball perception, ball tracking, proper positioning, and goal saving) is measured.
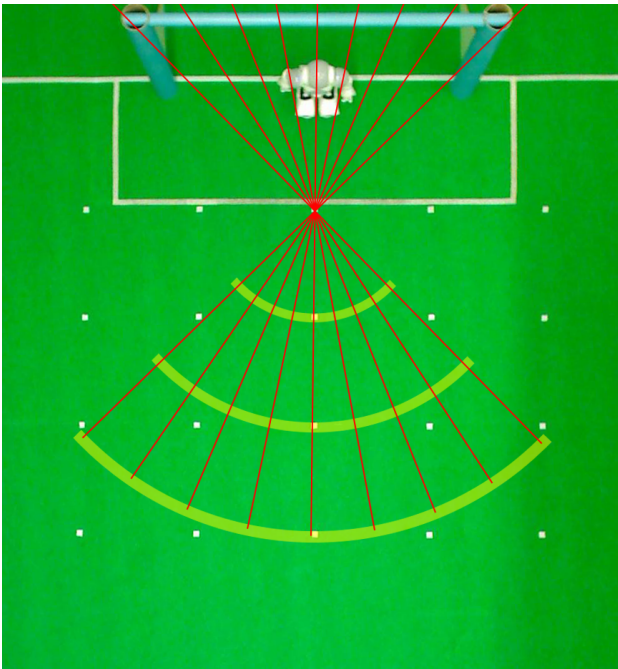


Fig. 8. Possible orientations from marker $(0, 2)$

To reduce the amount of measures to be taken, only markers $(0, n)$ from first row were used to test the architecture. Ramp inclination was also fixed at 6%. All nine possible angles and the three starting ball positions on top of the ramp were used. This means 135 different shots were used to test our

behaviour. Every shot was repeated ten times to ensure more reliable results, which gives us a total of 1350 measures. Fig. 8 shows the different trajectories possible from marker $(0, 2)$ and Table I shows the sucess rates (in %) obtained for every characteristic measured and every test performed. Ball perception and tracking is almost flawless, and positioning is also good, specially for central and side markers, that is, $(0, 0)$, $(0, 2)$, and $(0, 4)$. Positioning being not very accurate for markers $(0, 1)$ and $(0, 3)$ (see Fig. 7) is a consequence of far post shots from these positions: the goalkeeper tries to always position itself in front of the ball, which is better to intercept shots aimed at the near post but worse for far post shots, which mostly end up in a goal.

## V. CONCLUSIONS AND FURTHER WORK

In this paper we have presented a hierarchical architecture, borrowing concepts such as components (schemata) or dynamic hierarchies from ethology and adding others like the controller concept to simplify the architecture design and the information flow inside the system.

This architecture has been implemented on the software infrastructure offered by the NaoQi development environment for the Nao humanoid. We have shown two videos of this robot operating with our ethological architecture. The videos are intended to help understanding the hybridness of our approach, showing both the hierarcical structure of all available connections among components (deliberative part) and its dynamical variation with the activation and deactivation of components to create different hierarchies depending on external stimuli (reactive part).

The implementation has been evaluated using the "4+1 View Model of Architecture" by Krutchen [26], and it has been proved to comply with norm ISO 9126. To further validate its performance in real gameplay environemnt, a benchmark of goalkeeper specific behaviours has been proposed. The architecture was also tested druing RoboCup German Open (April 2009) and RoboCup Official tournament (July 2009). The results obtained in both the benchmark and the competitions show the correctness of the approach:

- The architecture mantains a goalkeeper behaviour for the whole test or match (20 minutes for the former and 10 minutes for the later, both without human intervention).
- The architecture adapts to dynamic match situations, mainly ball position.
- The architecture implements the goalkeeper behaviour using a deliberative structure for planification (self-positioning) and a reactive control mechanism to respond to environmental and internal stimuli (goal saving).
- The architecture allowed the robot to track the ball nearly 100% of the time, position itself properly 84% of the time, and save goals from 62% of the trajectories tested.

Future works envisioned are to include in the architecture deliberative mechanisms to evaluate advantanges and drawbacks of deactivating a component (in terms of time cost) depending on the situation, improving reactive nature of the system via gaze control implementation to react to other elements apart from the ball, and testing architecture's suitability

| marker | ball perception | ball tracking | proper positioning | goal saved |
|--------|-----------------|---------------|--------------------|------------|
| (0,0)  | 100%            | 100%          | 100%               | 78%        |
| (0,1)  | 100%            | 97%           | 61%                | 50%        |
| (0,2)  | 100%            | 99%           | 98%                | 58%        |
| (0,3)  | 100%            | 97%           | 63%                | 54%        |
| (0,4)  | 100%            | 100%          | 100%               | 72%        |

TABLE I
GOALKEEPER'S BEHAVIOUR TEST RESULTS

to more complex behaviours that include collaboration with other robots. Minor improvements in goalkeeper positioning behaviour to make it less vulnerable to far post shooting would also be interesting given the results obtained in the tests.

## REFERENCES

[1] Arkin, R.C., Balch, T.: AuRA: Principles and Practice in Review. Journal of Experimental and Theoretical Artificial Intelligence, **9(2-3)** (1997) 175–188

[2] Arkin, R.C.: Motor Schema-Based Mobile Robot Navigation. The International Journal of Robotics Research, **8(4)** (1989) 92–112

[3] Chernova, S., Arkin, R., 2007. From deliberative to routine behaviours: a cognitively-inspired action selection mechanism for routine behaviour capture. Adaptive Behaviour Journal, Vol. 15, No. 2, pages 199-216, 2007.

[4] Duchon, A. P., Harren, W. H., Kaelbling, L. P. Ecological robotics. Adaptive Behaviour 6 (3/4), 473-507, special Issue on Biologically Inspired Models of Spatial Navigation, 1998.

[5] Firby, R. J., Task networks for controlling continuous processes. In: Proceedings of the 2nd International Conference on AI Planning Systems AIPS?94. Chicago, IL (USA), pp. 49-54. 1994.

[6] Michael P. Georgeff and Amy L. Lansky, Reactive Reasoning and Planning",Proceedings of AAAI-87 Sixth National Conference on Artificial Intelligence, pp. 677-68, Seattle, WA (USA), 1987.

[7] , Barbara Hayes-Roth, Opportunistic control of action in intelligent agents, IEEE Transactions on Systems, Man, and Cybernetics, Vol 23:6, pp. 1575-1586, 1992.

[8] , Ken Currie and Austin Tate,O-Plan: The Open Planning Architecture", "Artificial Intelligence", Vol. 52:1, pp. 49-86, 1991.

[9] José María Cañas y Vicente Matellán. *From Bio-inspired vs. Psycho-inspired to Etho-inspired robots*. Robotics and Autonomous Systems. Vol.55, Num. 12, pp. 841-850. DOI:10.1016/j.robot.2007.07.010

[10] J.M.Cañas, J. Ruíz-Ayúcar, C. Agüero, F. Martín. *JDE-neoc: component oriented software architecture for robotics*. . Journal of Physical Agents, Volume 1, Number 1, pp 1-6, 2007.

[11] Juan F. García, Francisco J. Rodríguez, Camino Fernández, and Vicente Matellán. *Designing a minimal reactive goalie for the RoboCup SPL*. WAF 2009 (Workshop de Agentes Físicos) Cáceres (Spain).

[12] Pilar Bachiller, Pablo Bustos and Luis J. Manso. *Attentional Selection for Action in Mobile Robots*. I-Tech, pp. 472, October 2008.

[13] Allen Newell Allen, J.C. Shaw, Simon, Herbert A. Simon. *Report on a general problem-solving program*. Proceedings of the International Conference on Information Processing. pp. 256-264, 1959.

[14] Lorenz, Konrad, *Foundations of ethology*. Springer Verlag, New York, 1981.

[15] D. McFarland, T. Bösser, Intelligent Behaviour in Animals and Robots. MIT Press, ISBN: 0-262-13293-1, 1993.

[16] Toby Tyrrell, *An evaluation of Maes' bottom-up mechanism for behaviour selection*. Journal of Adaptive Behaviour, Vol. 2, Num 4, pp. 307-348, 1994.

[17] Walter, Grey. *An imitation of life*. Scienti?c American, 1950.

[18] Reid Simmons, R. Goodwin, K. Haigh, S. Koenig, Joseph O'Sullivan, and Maria Manuela Veloso, *Xavier: Experience with a Layered Robot Architecture,* Agents '97, 1997.

[19] Alexander Stoytchev and Ronald C. Arkin, *Combining Deliberation, Reactivity, and Motivation in the Context of a Behaviour-Based Robot Architecture*. In Proceedings 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation. 290-295. Banff, Alberta, Canada. 2000.

[20] Ronald C. Arkin. *Motor Schema Based Mobile Robot Navigation*. The International Journal of Robotics Research, Vol. 8, No. 4, 92-112 (1989).

[21] Saffiotti, Alessandro and Wasik, Zbigniew, *Using hierarchical fuzzy behaviours in the RoboCup domain*. Autonomous robotic systems: soft computing and hard computing methodologies and applications. pp. 235-262. Physica-Verlag GmbH. Heidelberg, Germany, 2003.

[22] Scott Lenser and James Bruce and Manuela Veloso, *A Modular Hierarchical Behaviour-Based Architecture*, Lecture Notes in Computer Science. RoboCup 2001: Robot Soccer World Cup V. pp. 79-99. Springer Berlin / Heidelberg, 2002.

[23] T. Rofer and H. Burkhard and O. von Stryk and U. Schwiegelshohn and T. Laue and M. Weber and M. Juengel and D. Gohring and J. Hoffmann, B. Altmeyer and T. Krause and M. Spranger and R. Brunn and M. Dassler and M. Kunz and T. Oberlies and M. Risler and M. Hebbela and W. Nistico and S. Czarnetzkia and T. Kerkhof and M. Meyer and C. Rohde and B. Schmitz and M. Wachter and T. Wegner and C. Zarges. *German team: Robocup 2005*. Technical report, Germany, 2005.

[24] Antonio Gómez Skarmeta, y Humberto Martínez Barberá, *Fuzzy Logic Based Intelligent Agents for Reactive Navigation in Autonomous Systems*, Fitth International Conference on Fuzzy Theory and Technology, Raleigh (USA), 1997

[25] M. Loetzsch, M. Risler, and M. Jungel. *XABSL - A pragmatic approach to behaviour engineering*. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006), pages 5124-5129, Beijing, October 2006.

[26] Philippe Kruchten, *Architectural Blueprints The "4+1" View Model of Software Architecture*, IEEE Software 12 (6), pages 42-50, November 1995.

[27] Barry W. Boehm, *Software Engineering Economics*, Prentice-Hall, 1981.

[28] N. Nilsson, *Teleo-Reactive Programs for Agent Control*, Journal of Artificial Intelligence Research, pages 139-158, 1994.