An Approach for the Design of Self-Conscious Agent for Robotics

Antonio Chella[§], Massimo Cossentino[†], Valeria Seidita[§] and Calogera Tona[§]
[§]*Dipartimento di Ingegneria Informatica*
Universitá deli Studi di Palermo, Palermo, Italy
Email: chella@unipa.it, seidita@dinfo.unipa.it, tona@dinfo.unipa.it
[†]*Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche*
Palermo, Italy
Email: cossentino@pa.icar.cnr.it

***Abstract*—Developing complex robotic systems endowed with self-conscious abilities and subjective experience is a hard requirement to face at design time. This paper deals with the development of robotic systems that do not own any a-priori knowledge of the environment they live in and proposes an agent-orientd design process for modelling and implementing such a systems by means of implementing the perception loop occurring between environment, body and brain during subjective experience. A case study dealing with a robocup setup is proposed in order to describe the design process activities and to illustrate the techniques for making the robot able to autonomously decide when an unknown situations occurs and to learn from experience.**

***Keywords*-Design Process; Self-Conscious System; Perception Loop.**

I. Introduction

The design and development of complex robotic systems require a great effort in terms of techniques, models and methods in order to catch the specific features the system to be developed has to implement and in order to relate them and to realize them onto a specific robotic platform.

In the past several software engineering techniques have been proposed for developing complex robotic systems also using the well know agent paradigm [2] [13]; using that the robotic system is considered as a collection of agents, each agent is responsible of a functionality or is committed to reach a goal.

The authors carried on in the past several experiments in the creation of ad-hoc design processes and methods for different kinds of application; in this paper they present an agent oriented design process for the design and development of self-conscious robotic systems able to autonomously act in an unknown and unstructured environment, in a human like fashion.

The presented work is mainly based on the assumption that self-conscious ability can be implemented by a continuos interaction between brain, body and environment; hence the subjective experience, realized by means of what is called the *perception loop* allows the robotic system to anticipate the results of the mission it is performing and to realize if it was successful by means of a continuous comparison with the perceived environment. A perception loop is continuously and instinctively performed by a human being, when he is starting a mission he imagines his state (himself) at the end of the mission; when the mission is over, if what he has imagined is different from what he perceives then he realizes that his mission fails otherwise it was successful; in the first case he has to adopt some "corrective" actions.

In this paper we focus on the implementation of the perception loop in a robotic system using an agent oriented design process and on how a not pre-programmed system can be made able to take decisions and learn each time it encounters an unknown situation for which it was not designed.

The work embraces two different levels of abstraction, one concerning the creation, and then the use, of a design process for developing self-conscious systems and the higher one concerning the identification and the definition of the whole process for the development of conscious system, from the definition of the problem domain to the execution of the loop and the management of the robot parameters tuning phase.

The proposed design process (PASSIC) has been created greatly exploiting the experiences made in the past with PASSI (Process for Agent Society Specification and Implementation) [10] that has been extended and integrated with a set of portion of design processes for developing and implementing the reflective part of the robotic system. We mainly reused features from PASSIG [20] that offers the possibility of performing a goal oriented analysis of the system in the same way of what is proposed in [3] [22].

PASSIC phases are shown along the proposed paper through an experiment made about a Robocup setup and using the NAO platform.

The rest of the paper is organized as follows: in section II the background and motivation of the proposed work is illustrated, in section III an overview on the PASSIC design process and the complete process for the development of a self-conscious robotic system is given, section IV deals with the robocup experiement and finally in section V some conclusions are drawn.

II. Background and Motivations

The robot perception loop described in [5] [9] (see Figure 1) is composed of three parts: the *perception system*, the *sensor* and the *comparative component*; through the *proprioceptive* sensors the perception system receives a set of data regarding the robot such as its position, speed and other information. These data are used from the perception system for generating the *anticipation* of the scenes and are mapped on the effective scene the robot perceives, thus generating the robots prediction about the relevant events around it.

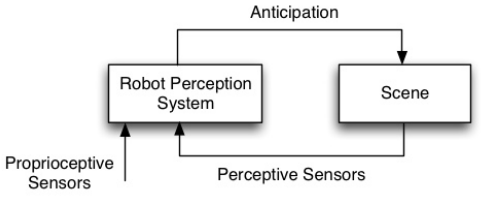

Figure 1. The Robot Perception Loop

As it can be seen from Figure 1, a loop there exists among the perception and the anticipation; each time some parts of a perceived scene, in what it is called the current situation, matches with the anticipated one the robot realizes experiences about what it is happening around it. According to [19], the perception loop realizes a loop among "brain, body and environment" that is the base for the *externalist* point of view of subjective experience; the subjective experience supposes a processual unity between the activity in the brain and what is perceived from the external world. We start from these considerations for developing and implementing our robotic systems.

The experiments we made aim at verifying the usability of the perception loop into a software design process ad-hoc created in order to support that. Our aim is to create a software system able to control a robot by means of perception loops.

The robot we consider does not own an *a-priori* knowledge of the possible obstacles it can encounter. It is able to self-localize, to recognize a situation preventing it to reach its goal (in this case to detect and identify an object as an obstacle) and to decide which is the best action to be performed in order to solve the problem.

The robot is not equipped with "pre-compiled" pre-planning abilities: we want to study the situation in which the robot does not know what to do in a given case, and so it queries his memory in search of a ready solution or it tries to find a novel solution exploiting his knowledge about itself, its capabilities and the surrounding world.

At the beginning of our experiment, the robot does not own any sophisticated behaviour: it can walk, move arms and legs and this is what it does, sometimes in a scrambled fashion, whenever it wants to reach a goal. For instance, let us suppose the robot wants to move on the floor and to pick up an object far in front of him. Then, it randomly tries to execute some behaviours from the set of "primitive" it owns until it does reach its objective. So, if the robot encounters a small obstacle, it could move on its right or left in order to go over, or it could move his arms (randomly) in order to cause the displacement of the obstacle. Each time a set of actions revealed to be successful for solving a problem, the robot is able to learn it, and when necessary it will apply again this successful strategy.

In particular, we experimented a humanoid robot endowed with a set of primitive behaviours. The learning phase consisted in letting the humanoid robot try some sets of behaviours, analyze the results and store them. Each time the robot, while reaching a goal, does not know how to solve a problem, it may query its database of previous solutions and retrieve past cases experienced as successful. When this strategy fails, the robot activates random behaviours.

The design process and the model we fixed for designing such a systems is general and it can be applied to different kinds of robot. Indeed, it aims at designing concepts such as goals, behaviours and actions.

We employed the humanoid robot NAO, a tall humanoid robot developed by Aldebaran Robotics[1]. NAO has 25 degrees of freedom, two cameras, inertial sensors and other sensors allowing it to interact with its environment. It is equipped with a set of basic behaviours that can be linked in a time-based or event-based fashion in order to create complex behaviours.

In order to implement the anticipation step of the NAO perception loop, we adopted the 3D robot simulator Webots from Cyberbotics[2]. By means of Webots, we may simulate NAO's movements and behaviours as well as its surrounding environment.

Figures 2 show the design and implementation of the perception loop in NAO. We exploited the fact that we use NAO and at the same time the NAO simulator, so the perception loop among brain, body ad environment [19] corresponds to the loop among NAO (the real robot), the virtual NAO (the robot simulator) and the NAO's world.

Both NAO and the virtual NAO use the *Behavioural Specification* resulting from the design process phases: the former for executing a sequence of behaviours, and the latter for producing the corresponding simulated behaviours. More in details, the Behavioural Specification is the work product resulting from the analysis phase. Here, the robot behaviours, to be put into practice for reach its goal, are fully specified; each behaviour is composed of a set of simple robot's actions.

Each time NAO encounters a stop condition - for instance an unexpected object in the path - while pursuing its goal, the

[1]http://www.aldebaran-robotics.com
[2]http://www.cyberbotics.com

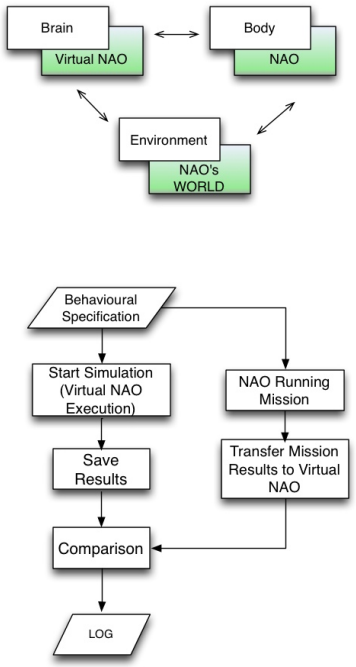

Figure 2. The Loop Brain-Body-Environment vs the NAO Mission

anticipated scene is compared with the real NAO parameters and the resulting *log*, in form of proprioceptive and sensorial values, is used for the tuning and learning phase.

It is worth noting that the use we made of the simulation is quite different from the common one: in fact, it is not used for investigating and anticipating the robot behaviour in a specific working condition, but instead for producing, starting from the designed behaviours, the expected results of a mission. The simulator and NAO work separately, only when a stop condition is identified the simulation results are compared with the real NAO parameters.

III. The Self-Conscious Robotic System Design Process

In [7] [8] the creation of PASSIC design process and a model for perception loop has been presented. The process for creating the new design process follows the *Situational Method Engineering* paradigm [4] [18] and extends the PASSI2 [11] and PASSIG [20] developed by the authors in the latest years for creating ad-hoc design processes [21]. One of the main points is the definition of a *metamodel* for the perception loop, see [7] for further details, where elements of perception loop have been identified and reflected onto a robotic system.

The creation of PASSIC allowed us to formalize the design of the aforementioned kind of robotic applications, it has been then included in what we think it would be the whole self-Conscious System Development Process (CSDP).

The complete development process used for developing self-conscious systems is composed of three different phases: *Problem*, *Design and Configuration* and *Execution*.

The Problem phase is composed of all the activities devoted to focus the problem domain hence to elicit the system requirements and to identify the mission the robot performs to reach its goals. During these activities the designer considers a database where the set of abilities the robotic system possesses are stored (*Cases* and *Configuration*).

A *Case* is composed of the goal description, the set of tasks performed in order to reach it (a plan), some preconditions, and the list of parameters needed for successfully applying the plan. With reference to NAO, task correspond to the action it is able to do.

A *Configuration* is a specific set of parameter values that has proved to be successful to instantiate one specific case; it also includes the number of total used and positive outcomes this configuration produced in pursuing one case.

During the *Design and Configuration* phase, the designer defines a software solution in order to accomplish the required goals; these activities are made with the aid of the PASSIC design process during which two fundamental deliverables are collected: *i)* the design of the robotic system to be built using a society of agents each of which committed to realize a particular goal of the system; *ii)* the mission configuration including all the elements of the database of cases and configurations related to the particular robotic platform chosen. In the next section an example reporting a part of a case study will be given.

In the *Execution* the running system is considered, here the robotic system has to execute a mission, a goal to satisfy following a plan, hence a specific sequence of tasks to be executed. For each goal the related set of tasks is decided at design time, the specifications are at the same time sent to the part of the system devoted to generate the anticipation and to the part (the robot itself) that really executes the mission - it is shown in Figure 2.

Once both have terminated the results are compared, if they match then the goal has been reached and the configuration (task, parameter,...) can be saved for future reuse, the system has learnt. On the contrary if the results do not match the robotic system has to, without the human intervention, select another mission configuration.

One of the most important part of the CSDP is the PASSIC design process, it has been ad-hoc created in order to the need of perception loop based self-conscious system design.

PASSIC is composed of three phases and it is based on a iterative/incremental life cycle (see Figure 3). The *System Requirement* phase deals with a goal oriented analysis of the system and one of the main model it produces is the *Agent Diagram* where the society of agents is identified together with the roles ad the tasks of each agent. Besides a description of agent structure- in the *Agent Structure Exploration* - in terms of tasks required for accomplishing

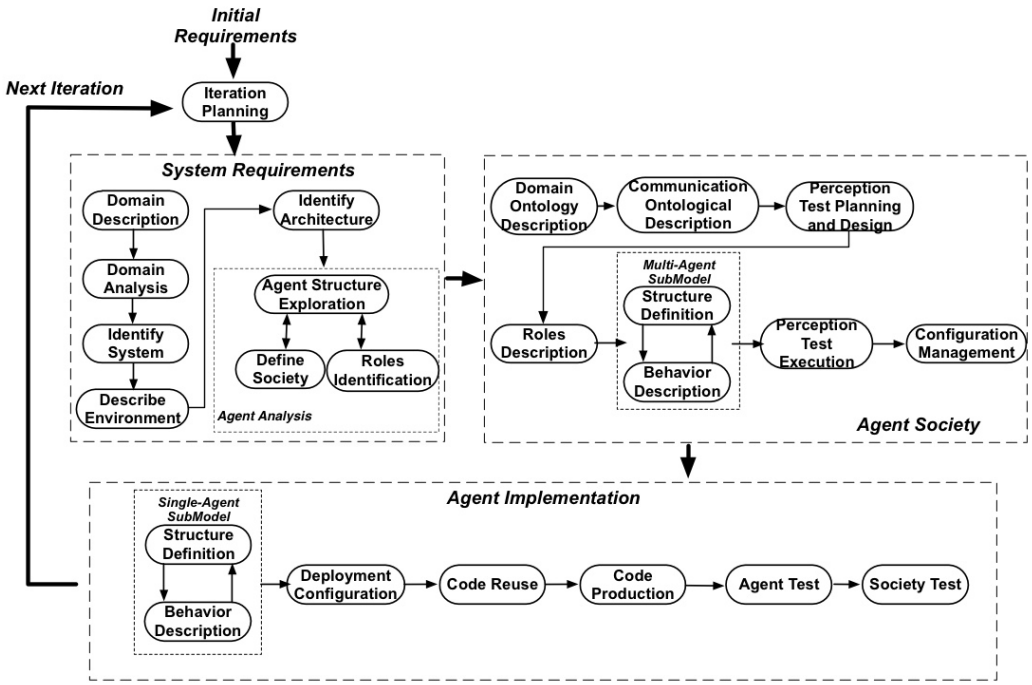

Figure 3. The PASSIC Design Process

the agents' functionalities is given.

In the *Agent Society* phase an agent based solution is introduced, in this phase the ontological description of the domain categories and the actions having effect on their states allows to establish the agents' communications. Then the agents are described in terms of roles, services and resources dependencies. Once the agent society has been designed the autonomous part of the system devoted to implement the perception loop is designed; during the the *Perception Plan and Design*, starting from the knowledge about the environment, the agents' society architecture and the requirements, the anticipation is produced. Then the *Perception Test Execution* aims at designing the criteria for evaluating the results of the running loops and in the *Configuration Management* the rules for enhancing the Case and Configuration database, hence the rules for tuning the system parameters, are designed.

Finally in the *Implementation Phase* the model of the solution architecture in terms of classes, methods, deployment configuration, code and testing directives is realized. In this phase, the agent society defined in the previous models and phases is seen as a specification for the implementation of a set of agents that should be now designed at the implementation level of details, then coded, deployed and finally tested.

IV. Applying PASSIC - a Robocup Based Case Study

In this section the self-Conscious System Development Process (CSDP), including some PASSIC activities, will be shown through a case study in order to make some fundamental aspects of the proposed work clear. We will focus on the possibility of designing the mission configuration phase allowing the robot to effectively implement the perception loop. Therefore to provide the robot with the capability of: *i)* becoming conscious of what it is happening around it, *ii)* efficiently selecting a set of behaviours letting it to reach its goal, even if unexpected situations occurs, and finally *iii)* learning from experience.

NAO is the official platform used in the standard league of Robocup [3]. Robocup aims at realizing a team of humanoid robots that in an autonomous fashion were able of challenging the world soccer champion team and winning over.

Robocup is the ideal workbench for encouraging study and research on all the methods and technologies for dealing with themes and aspects such as localization, environment mapping, obstacle identification etc., the whole made in guided or autonomous way. For these reasons we decided to use a Robocup workbench in order to carry on our tests.

The experiment made was about the development of a multi-agent system for managing two robots (two NAOs) engaged in a soccer match where one NAO (from now on

[3]www.robocup.org

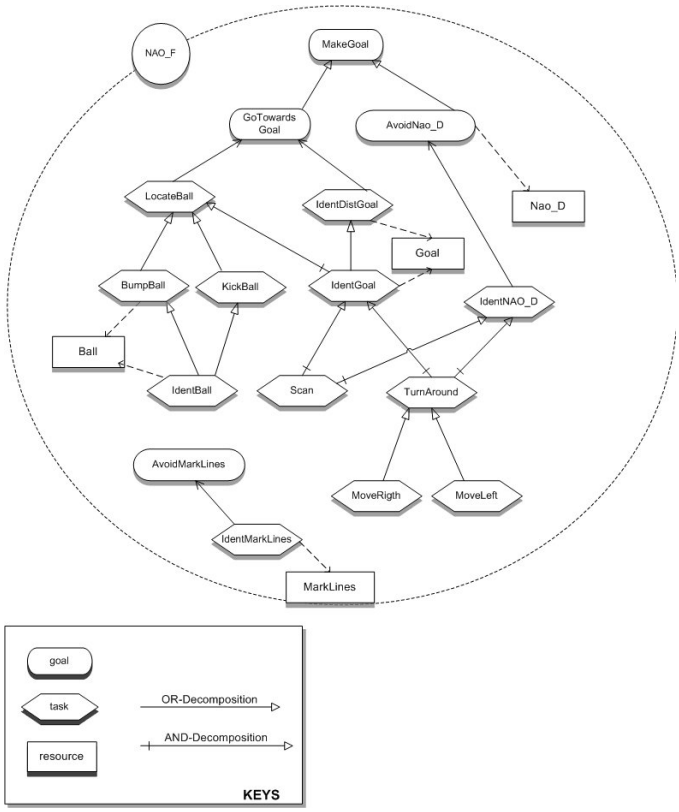

Figure 4. The Goal Diagram Portion of the Robotic System

we will call it NAO_F) serves as a forward with the main scope of making a goal and the second one as a defender (we will call it NAO_D) with the main scope of preventing the NAO_F from making the goal.

As already said, NAO platform is endowed with a predetermined set of actions that proved to be very useful for Robocup setups. The capabilities the NAO offers has been analyzed during the Problem Domain phase and let us to identify the right tasks to be used for realizing the goals identified during the PASSIC design phases. The developed system includes goals such as identifying the ball, identifying the goal and the boundaries of the game field, interfacing with the simulator, managing the results of the comparison and managing the communications with the databse. Some of the previous goals are clearly NAOs' goals but it is to be noted that their are committed to one or more agents of the society. These goals are designed using PASSIC and from the design activities it results a set of behavioural specification useful for NAOs and the simulator (as it is shown in Figure 2).

Referring to Figure 1 and to what we said in section II the term "scene" includes a whole set of parameters including the surrounding world and the robot state identified through the set of specific parameters related to robotic platform used - NAO in our case.

As it can be seen from PASSIC, designing the mission configuration exploits the analysis of the system's goals. Figure 4 shows a portion of the goal diagram resulting from the Domain Analysis activity which aim is to identify each actor's tasks and applying means-end-analysis - a task (mean) can be used to achieve a goal (end). The NAO_F actor has been identified with some of its goals and tasks. The Goal Diagram results from the portion of Tropos [3] [14] we used when we created PASSIG; Tropos principally adopts a requirements driven software development approach, exploiting goal analysis in order to identify actor dependencies. Tropos's, hence PASSIC analysis activities, main concepts are: the *Actor* that models an entity having strategic goals and represents a physical, a social or a software agent; the *Goal*, that is the strategic interest of an actor, satisfied through the *Task*, hence a particular course of action that can be executed. Another important element is the *Resource*, an entity without intentionality that can be physical or informational.

In the Figure 4 the goal *MakeGoal* is considered, it is decomposed in two goals *GoTowardsGoal* and *AvoidNAO_D*, each of them is related to one or more tasks through the means-end-analysis and uses one or more resources; for instance bumping the ball implies the use of the ball as a resource.

With reference to the proposed case our main aim is to have a robot exploiting all that we call its innate capabilities in a human like fashion. For the sake of brevity in this section we point our attention to a sub-case, the one concerning the following scenario: the NAO_F is in front of the ball and its objective is to kick the ball into the goal also autonomously managing all the possible unknown or unexpected situations.

In the following all the design results of this sub-system will be shown and it will be illustrated how the robot acts as the result of design time activities and how it act as the results of the self-conscious abilities it has been provided.

In order to express the pre and post-condition in the configuration we assume for now a high level of abstraction and we consider that the robot interacting with its environment, whatever its goal, has to see the world, the object in the world, to touch them if necessary, hence it has to sensorially perceive and then it can act.

During the System Requirements phase the set of actors involved in the system together with all their related goals are identified and analyzed; the result is a set of work products including goal diagrams and agent diagrams as it is proposed in PASSIC [4] and in [20] [3].

In Figure 5 a portion of the agent diagram for the *TakeBall* goal to be pursued is provided, it relates to the case the robot has to reach the ball in order to kick it towards the goal.

[4]More details can be found in http://www.pa.icar.cnr.it/passi/PassiExtension/extensionIndex.html

This artefact results from the Identify Architecture activity where the system-to-be is decomposed into sub-actor and the agents of the system are identified.

It can be seen that the main actor involved in this part of the system is the NAO_F, and the agent identified as responsible for this goal is the *MoveManager* agent. The *TakeBall* goal has been decomposed, through the means-end-analysis, into two tasks: *Walk* and *Turn*; in this case the two tasks are present in the database of cases analyzed and identified during the problem domain phase of the CSDP, besides the used resources are that provided by NAO platform.

Starting from the agent diagram and from all the identified relationships between each goal and the set of tasks to be used to reach it, the designer can perform the *Configuration Management* phase during which the database of Configurations and Cases is enhanced with all the behaviours established at design time.

In Figure 6 the portion of the Case and Configuration database related to the presented experiment is shown. During the analysis phase we considered only two possible cases; as previously said the robot interacts with the environment through its sensors and actuators, so we established two kinds of pre-conditions: the vision and the sensorial perception. If the robot's goal is to take the ball it has to walk towards the ball only if two conditions occur: it sees the ball but it has not still touched it, this means that the ball is in front (or more generally in the robot's vision field). If, instead, the ball is not in the vision filed, the robot has to turn around in order to identify its position and then it can walk towards it. Therefore the pre-conditions for this case are {vision=Y AND sens_perception=N} or {vision=N AND sens_perception=N}.

In order to describe the experiment in a more complete fashion, let us suppose that the database also presents two

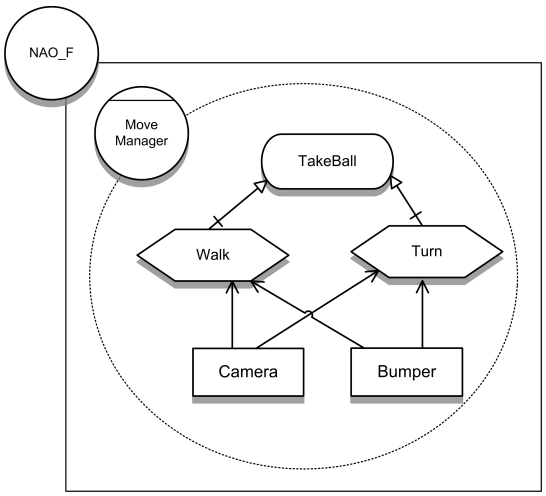

Figure 5. The Agent Diagram Portion of *TakeBall* Goal

CASE

ID	*Goal*	*ID_Task*	*Pre*
1	Go towards the ball.	6	{vision=Y AND sens_perception=N}
2	Go towards the goal.	6	{vision=Y AND sens_perception=Y}
3	Locate the ball	3	{vision=Y AND sens_perception= Y/N}
4	Kick the ball	7	{vision=Y AND sens_perception=Y}

a)

CONFIGURATION

ID	*CASE*	*Post*	*Param. Value*	*Succ.*	*Total*
1	1	{vision=Y AND sens_perception=Y}	v=3; ns=5	3	6
2	2	{vision=Y AND sens_perception=Y}	v=2; ns=8	4	6
3	4	{vision=Y/N AND sens_perception=N}	Signal=True	5	7

b)

TASK

ID	*Elementar Task*	*Description*	*Parameter Type*
1	ArmCircle	An arm of the robot makes a circle	Duration, Steps
2	StandUp	The robot stand up, if it is standing on its front or on its back. Otherwise it will do nothing.	Signal
3	NaoMark	Start naoMark extractor and subscribe on mark value	MarkList
4	FaceDetect	Start naoMark extractor and subscribe on mark value	FaceList
5	Turn	The robot will turn around itself	Angle
6	Walk	This Box should make your robot walk straight	Velocity, Steps
7	Bumper	Listens to bumpers sensors. Stimulate left or right output depending on what bumper has been stimulated	Signal

c)

Figure 6. The Portion of Case and Configuration Related to the *TakeBall* Goal

others cases, one related to the *Turn-Left* task and the other to the *Turn-Right* with the related pre conditions.

For the *Execution* phase we used three different setups - the goal is the same: *a)* the NAO_F in in front of the ball, *b)* the NAO_F on the left of the ball and *c)* the NAO_D has gone between the NAO_F and the ball.

Case a): the NAO_F sees the ball - {vision=Y and sens_perception=N}) - the first Case is selected and the NAO_F walks towards the ball for a maximum of 4 steps (this parameter is imposed at design time in order to stop the mission and to start the perception loop comparison). Each time the mission stops the comparison between the perceived scene and the anticipated one is performed and in this case, it is obvious that if the ball is not reached the NAO keeps going as it was designed to do for.

Case b): the NAO_F does not see the ball - {vision=N and sens_perception=N}) - the second case is selected and the NAO_F turns of 5 degree (this parameter too is fixed at design time in order to stop the mission); when after a certain number of turns it sees the ball it is in the a) case

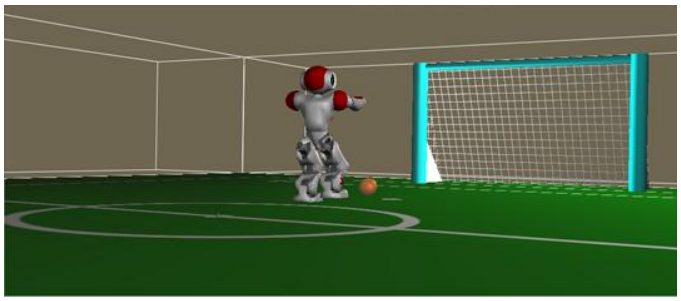

Figure 7. The Anticipation Generated for the Case a)

and it selects the first CASE moving towards the ball.

Case c: while the NAO_F is performing the actions for the case a) the NAO_D has come in its trajectory, the stop mission condition is revealed and the comparison results in a mismatch between the expected and real situation; this is an unexpected situation and the NAO_F is not designed for it; our aim is to provide means for autonomously deciding what to do in these cases without the human intervention. The NAO_F has to retrieve from its knowledge base, the Case and Configuration database, that in some senses emulates the human behaviour based on the classic case based reasoning [16] [1], the most useful Case and the related task, basing on the goal it has to satisfy.

The case c) can have two possible consequences: the NAO_F finds in the database a Case to be used or not, in this second case following our rationale for providing the robotic system with self-conscious ability the NAO_F should randomly select a Case, the one containing the closer goal and it performs the related task. Obviously, considering a large populated database of Cases and a long list of tasks (representing the innate ability of the robot in general and the NAO_F in this particular experiment), it is not efficient to let the NAO_F selects each kind of task. If the goal is the one said above, it is for sure unfruitful to move the arm or to make some flexions so we needed a way for creating a taxonomy of the goal.

In order to solve this problem we reused our previous experience in providing a robot [6] with the ability of planning a path in a dynamic indoor environment by using Cyc and the common sense reasoning [17] [12]. This part of the system is designed in the *Perception Test Execution* phase and for space concerns is out of the scope of this paper. It is only worth noting that by using the common sense reasoning we are able to create at design time a kind of goal taxonomy letting the robot select, when necessary, a Case in a set of Cases related with the one it had before the unknown situation and avoiding the risk to move the arm when it has move its legs.

In Figure 7 the anticipation of the *TakeBall* mission in the case a) is shown. The behavioural specifications generating by the designed behaviour in the Webots are the same given to the real NAO, the environment is exactly the same in both the cases and the NAO_F at the end of its mission does not reveal any differences in its paramenters against the virtual NAO one. When, in the real environment, the NAO_D is between the NAO_F and the goal, the expected situation is different from the real one and NAO_F started to retrieve the most useful case from the database.

In the presented experiment the NAO_D lied still while the NAO_F was performing its mission; even if this part of the experiment is hardly reduced in the paper, it is worth nothing that it let us to verify the PASSIC design process activities and the way of saving the winning configuration causing the robot learning.

V. Conclusion

The authors developed in the past some agent oriented design processes realizing the possibility of designing systems working in different application contexts mainly exploiting the fact that agent oriented processes can be used as a design paradigm.

In this paper an ad-hoc created design process (PASSIC) is exploited in order to develop and implement a robotic system able to manage a soccer match in an autonomous fashion. The meaning of autonomy in this case is the capability of the robot to perceive its objectives and execute its tasks without the human intervention when it encounters situation not established at design time.

The perception loop and the way how a robotic system with only its innate capabilities, is able to manage and interact with unstructured environment.

PASSIC has been created basing on the experiences made in the latest years in the construction of ad-hoc design processes and on the use of agent oriented methodologies for developing and implementing robotic systems. PASSIC allows the design and implementation of the perception loop thus letting the system able to move in its environment and decide by continuously comparing the differences between expected and real situation. In this context the agent paradigm proved to be very useful in the sense that using PASSIC we were able to identify a society of agents in which a set of agent was devoted to manage a perception loop; moreover agents' peculiarities such as autonomy, proactivity and situadness perfectly fitted our case.

The proposed experiment allowed us to test the usability of the whole development process and in particular of PASSIC in the part regarding the parameters and configurations tuning; a lot of problems to be taken into account in the proposed setup such as real time contingencies and interaction among a society of robots will be treated in the future as well we are now deepening and fixing what is related to the taxonomy of the goals in order to improve and speed up the selection of cases.

ACKNOWLEDGMENT

This research has been partially supported by the EU project FP7-Humanobs and by the FRASI project managed by MIUR (D.M. n593).

REFERENCES

[1] A. Aamodt and E. Plaza. Case-based reasoning. *Proc. MLnet Summer School on Machine Learning and Knowledge Acquisition*, pages 1–58, 1994.

[2] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. An Architecture for Autonomy. *The International Journal of Robotics Research*, 17(4):315, 1998.

[3] Paolo Bresciani, Paolo Giorgini, Fausto Giunchiglia, John Mylopoulos, and Anna Perini. Tropos: An agent-oriented software development methodology. *Autonomous Agent and Multi-Agent Systems (8)*, 3:203–236, 2004.

[4] S. Brinkkemper, K. Lyytinen, and R. Welke. Method engineering: Principles of method construction and tool support. *International Federational for Information Processing 65*, 65, 1996.

[5] A. Chella. Towards robot conscious perception. In A. Chella and R. Manzotti, editors, *Artificial Consciousness*. Imprinting Academic, Exter, UK, 2007.

[6] A. Chella, M. Cossentino, M. Fisco, M. Liotta, A. Rossi, and G. Sajeva. Simulation based planning and mobile devices in cultural heritage robotics. In *Proc. of the Robotics Workshop in the Ninth Conference of the Associazione Italiana per l'Intelligenza Artificiale*, Sept. 2004.

[7] A. Chella, M. Cossentino, and V. Seidita. Towards a Methodology for Designing Artificial Conscious Robotic System. In A. Samsonovich, editor, *Proc. of AAAI Fall Symposium on Biologically Inspired Cognitive Architectures BICA '09*, Menlo Park, CA., 2009. AAAI Press.

[8] A. Chella, M. Cossentino, and V. Seidita. Towards The Adoption of a Perception-Driven Perspective in the Design of Complex Robotic Systems. *Proc. Of the 10th Workshop on Objects and Agents (WOA09)*, 2009.

[9] A. Chella and I. Macaluso. The perception loop in Cicerobot, a museum guide robot. *Neurocomputing*, 72:760 – 766, 2009.

[10] M. Cossentino. From requirements to code with the PASSI methodology. In *Agent Oriented Methodologies* [15], chapter IV, pages 79–106.

[11] M. Cossentino and V. Seidita. Passi2 - going towards maturity of the passi process. *Technical Report ICAR-CNR*, (09-02), 2009.

[12] TX. Cycorp Inc., Austin. Cyc home page. http://www.cyc.com.

[13] AC Dominguez-Brito, D. Hernandez-Sosa, J. Isern-Gonzalez, and J. Cabrera-Gamez. Integrating robotics software. *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, 4, 2004.

[14] Paolo Giorgini, Manuel Kolp, John Mylopoulos, and Jaelson Castro. Tropos: A requirements-driven methodology for agent-oriented software. [15], chapter II, pages 20–45.

[15] Brian Henderson-Sellers and Paolo Giorgini. *Agent Oriented Methodologies*. Idea Group Publishing, Hershey, PA, USA, June 2005.

[16] J. Kolodner. *Case-Based Reasoning*. Morgan-Kaufmann Publishers, Inc., San Mateo, CA., 1993.

[17] Douglas B. Lenat. Cyc: a large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):33–38, 1995.

[18] J. Ralyté. Towards situational methods for information systems development: engineering reusable method chunks. *Procs. 13th Int. Conf. on Information Systems Development. Advances in Theory, Practice and Education*, pages 271–282, 2004.

[19] W.T. Rockwell. *Neither brain nor ghost*. MIT Press, 2005.

[20] V. Seidita, M. Cossentino, and S. Gaglio. Adapting passi to support a goal oriented approach: a situational method engineering experiment. 2007.

[21] V. Seidita, M. Cossentino, V. Hilaire, N. Gaud, S. Galland, A. Koukam, and S. Gaglio. The metamodel: a starting point for design processes construction. *International Journal of Software Engineering and Knowledge Engineering. (in printing).*, 2009.

[22] E. Yu. Towards modeling and reasoning support for early-phase requirements engineering. In *Proc. RE-97 - 3rd Int. Symp. on Requirements Engineering*, pages 226–235, Annapolis, 1997.