International Journal of Machine Consciousness
© World Scientific Publishing Company

From Modelling to Implementing the Perception Loop in Self-Conscious Systems

Valeria Seidita
Dipartimento di Ingegneria Informatica, Universitá degli Studi di Palermo
Viale delle Scienze, 90128 Palermo, Italy
seidita@dinfo.unipa.it

Massimo Cossentino
Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche
Viale delle Scienze, 90128 Palermo, Italy
cossentino@pa.icar.cnr.it

Received 1 April 2010
Revised 9 April 2010

Engineering self-conscious robotic systems is a challenge issue because of the intrinsic complexity of such systems; a self-conscious robot has to acquire knowledge, to understand its world and to autonomously interact with its environment. In this paper the externalist point of view is used for developing a complete process for the design and implementation of a conscious robotic system that is able to interact with a dynamic environment in a human like fashion without possessing detailed knowledge about the environment and pre-programmed tasks and algorithms. The paper mainly focusses on the configuration part of the whole process that make the robot able to decide and to learn from experiences.

Keywords: Self-Conscious System; Design Process; Goal; Perception.

1. Introduction

How to realize machine consciousness, what the meaning of consciousness is in a human being and what in a robot, how can a robot be able to reflect about itself are hard problems in the field of machine consciousness; they have been faced through different points of view, from philosophical to psychological and to scientific ones.

Starting from the assumption that with the term *machine consciousness* [Chella & Manzotti, 2009] it is intended to refer to a wide range ot aspects of the human being that can be reproduced, emulated, in a robot, we aim at exploring and developing software engineering techniques for designing and implementing self-conscious robotic system.

We base our work on the externalist [Manzotti, 2006] point of view that sees the self-conscious ability of a system, just like that of a human being, as underpinned by the subjective experience of what there is in the outer world and in the inner world. A system is able to reflect about itself and the world around it by continuously comparing the subjective with the objective experience - hence the continuos interaction among brain, body and environment through what it has been called the *perception loop*.

The perception loop [Chella, 2009] realizes a continuous interaction with the external environment by means of continuously comparing the expected behaviour with the real one. In a real robotic system there may be different perception loops contemporarily in action, being each of them related to different sensor modalities or considering different parameters and aspects of the same sensor modality.

Higher order perceptions make the robot able to reflect about itself, in the sense that the higher order loops allow the robot to make inferences about how to act in the scene. In [Chella, 2009] is argued that higher order perception loops are responsible of the robot self-consciousness.

Implementing generalized higher orders of perception loops in a robotic system is a hard issue, in this paper we propose a self-Conscious Systems Development Process (CSDP) composed of three main phases: the *problem domain* analysis for identifying the robot's mission, the *design and configuration* activities for realizing the robotic system and the *execution* activities entailing the mission execution and if/when necessary robots parameters tuning.

The PASSIC [Chella *et al.*, 2009b][Chella *et al.*, 2009a] design process is one of the main elements of the proposed process. More specifically, PASSIC is employed in the *design and configuration* phase and it provides means for designing the robotic system. PASSIC has been created by exploiting the experiences made in the past with using and creating agent-oriented design processes. For the purposes of creating PASSIC, PASSI (Process for Agent Society Specification and Implementation) [Cossentino, 2005] has been extended (PASSI2 [Cossentino & Seidita., 2009]) and integrated with a set of portions of design processes for developing and implementing the reflective part of the robotic system. In so doing, we also reused features from PASSIG [Seidita *et al.*, 2007] that offers the possibility of performing a goal oriented analysis of the system in the same way of what is proposed in [Bresciani *et al.*, 2004][Yu, 1997].

The rest of the paper is organized as follows: in section 2 the proposed process is illustrated, in section 3 all the elements of the CSDP process are detailed, in section 4 an experimental setup is proposed in order to shed light on a specific aspect of the process (the one related to the mission configuration), and finally in section 5 some conclusions and future works are drawn.

2. The Self-Conscious System Development Process

Starting from the externalist point of view we want to design and develop systems able to interact with an unknown environment and to create a model of their inner and outer world by continuously comparing the subjective with the objective experience.

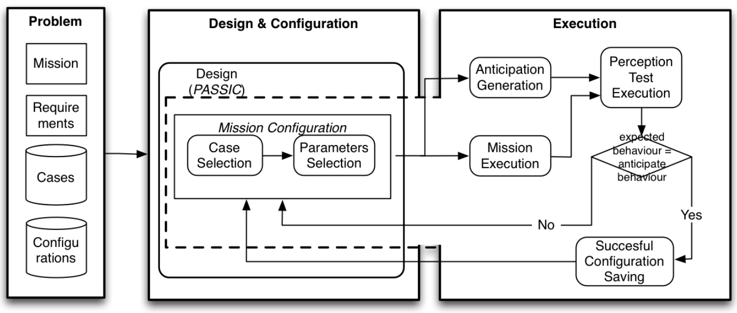


Fig. 1. The self-Conscious System Development Process

In our work we consider robotic systems possessing a set of "innate" abilities which they can refer to in order to solve problems equal or similar to already faced ones. In this way we can endow the system with the ability of using previous experience and learning a new one when it proves to be successful. In the attempt to give autonomy and self-conscious ability to the robotic system we are drawing on case-based reasoning theories [Kolodner, 1993][Aamodt & Plaza, 1994]: retrieving the most similar case (or set of cases) for solving a specific problem and reusing the information from the retrieved case to solve the new problem. Finally, if the case was successfully applied, the robot saves the parts of this experience likely to be useful for future problem solving.

Since a solution to the problem may not be available the robotic system has to be able to try to use its own abilities and then to learn whether it has success or not, in the same way the human being does.

In such a way, the main elements to be used by self-conscious robotic systems are: the *case* (i.e. the set of actions to be performed for pursuing a well determined goal), and the knowledge about the application of a specific case by means of the set of parameters to be used for instantiating the *case* (what we call *configuration*).

In order to engineer the development of a self-conscious system in its whole we identified the three different areas reported in Fig. 1. Let us start from the latest area, the *Execution* one, where the running system is considered; the robotic system has to execute a mission, this means it has a goal and it can satisfy that by following a plan; the plan may be regarded as 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. Once both have terminated, their results are compared (*Perception Test Execution* rounded box in Fig. 1); if they positively match then the goal has been reached and the configuration (task, parameter,. . .) can be saved for future reuse, the system learns that. On the contrary if the results do not match, the robotic system has to select, without any human intervention, another mission configuration.

All the elements involved in the *Execution* area have to be designed and configured, so the *Design and Configuration* area deals with all the elements devoted at producing the running system; it is composed of the design activities prescribed by the PASSIC design process [Chella *et al.*, 2009b][Chella *et al.*, 2009a]. The process starts with the inputs collected during the activities involved in the Problem area (the details of this part is out of the scope of this paper) and according to them it aims at defining two fundamental deliverables: the design of the robotic system to be built and the design of the perception test that will drive the robot's behavioural choices. This latter artefact, also includes the specification of the rules that will be used for tuning system parameters when the executed behaviour results do not match the anticipation.

PASSIC is an agent-oriented design process and the reasons for us to choice such a kind of design process instead of, a more usual object-oriented one, will be illustrated in the following section.

At design time a mission, in terms of cases and parameters, is configured for each goal; let us suppose the goal of the robot is: *"go towards object x"*, the designer has elicited this goal during the analysis phase and has identified an agent committed to execute the related plan (composed of tasks), so the configured mission is composed of the goal (i.e. *"go towards object x"*), the plan-tasks (i.e. *walk ahead*) and the parameters (i.e. *speed, number of steps*). During the design phase a case can be selected from the repository or can be generated from scratch and inserted in the repository before the robotic system begins its life, in a sort of initial memory generation.

In the first area - *Problem* - the designer describes the problem statement under the point of view of the missions to be executed by the robot, the requirements the whole system has to fulfil and the repository of cases and configurations the adopted robotic platform natively includes, and that can be augmented after robot missions; by querying the repository (dynamically) the robot may tune some of the mission execution parameters, or it may decide to adopt another behaviour or to save the successful one in the repository for a future reuse; in the following section 4 an example using the humanoid NAO-Aldebaran[a] is given.

3. Detailing the Elements of the Self-Conscious System Development Process

The robotic systems we want to model with the previous described development process are based on the externalist point of view; under this hypothesis, subjective experiences are supposed to be caused by the interaction between the brain and the environment; brains activities and external perceived events realize a sort of processing unit.

[a]http://www.aldebaran-robotics.com

In this section we will detail two main elements of the CSDP: the perception loop and the PASSIC design process used for designing it.

3.1. *The Perception Loop*

The robot perception loop described in [Chella, 2007; Chella & Macaluso, 2009] (see Fig. 2 a)) 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. This data are used from the perception system for generating an *anticipation* of the scenes and it is mapped on the effective scene the robot perceives, thus generating the robot's prediction about the relevant events around it, hence the subjective and the objective experience.

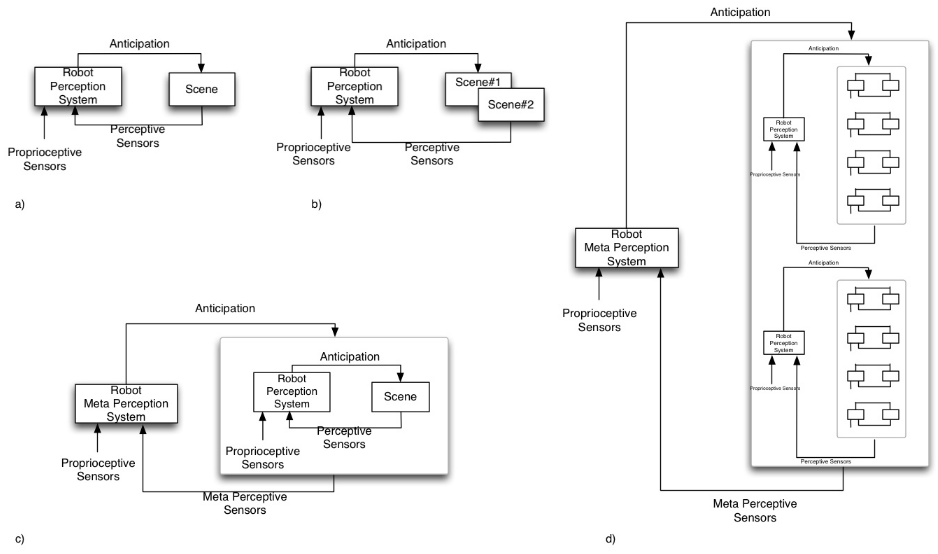


Fig. 2. Different Perception Loops Order in a self-Conscious System

As it can be seen from the above cited figure, a loop there exist among the perception and the anticipation, so that if a part of the perceived scene (the current situation), matches with the anticipated one (within the "Scene" block), then the anticipation of other parts of the same scene may be generated (Fig. 2 b)). According to [Rockwell, 2005], such a perception loop realizes a loop among "brain, body ad environment".

The generalized perception loop has been tested and implemented on *Cicerobot*, an indoor robot offering guided tours in the Archaeological Museum of Agrigento [Chella & Macaluso, 2009], and on *Robotanic*, an outdoor robot offering guided tours in the Botanical Garden of the University of Palermo [Barone *et al.*, 2008].

By implementing the perception loop the robot is endowed with the ability to sense (perceive) the word around it; besides in [Chella & Macaluso, 2008] [Chella, 2009] it is argued that in a real operating robot there can be different perception loops contemporaneously in action, thus realizing robot self-consciousness involving the robot's inner world perception (see Fig. 2 c) and d)). Each of them is applied to different abilities of sensing and reacting to external stimuli; and all of them can be managed at an higher level allowing the lower order loops to perceive the environment and the higher order loops to perceive the self thus providing the robot with a wide autonomous control about its own capabilities, actions and behaviours.

Moreover each of them can be managed by one or more agents arranged in a hierarchical structure where the higher-level agent is aware of its self and at the same time, through the law level agents, of its environment.

3.2. *Designing the Robotic System*

As already discussed, we created a specific process (PASSIC) for supporting the development of self-conscious robotic systems (see Fig. 1). PASSIC has been created by extending and integrating two existing processes, PASSI2 [Cossentino & Seidita., 2009] and PASSIG [Seidita *et al.*, 2007]. Both the two are evolutions of PASSI (Process for Agent Society Specification and Implementation [Cossentino, 2005]).

PASSI has been widely used over the years for developing different kinds of agent applications (also including robotic ones) mainly exploiting the fact that it provides means for: *i)* decomposing the system requirements in functionalities which can be committed to an agent or a society of agents, *ii)* for representing the environment the robot lives and *iii)* for managing the communications among the different parts of the system.

During the last years researchers in our laboratory have been experimented the possibility of creating a design processes framework, centred on PASSI as the "core", for being applied to a wide range of multi agent systems development. One of the most important elements of the framework is the natural evolution of PASSI, labelled PASSI2, that fundamentally provides a greater attention towards organizations and the possibility of early defining the structural description of the identified agents.

An ongoing description of the other well defined design processes can be found in the PASSI website[b]. For the purpose of this paper we consider only PASSIG that was created for providing activities for a goal oriented analysis of the features the system has to accomplish.

On of the pillars of our approach is that in designing a system, each designer refers to a formalized (or not) system metamodel explicitly related to the adopted design process. A metamodel contains a vocabulary of concepts and a set of rela-

[b]http://www.pa.icar.cnr.it/passi/PassiExtension/exstensionsIndex.html

tionships among them to be instantiated at design time. For instance let us consider the well known UML metamodel [UMLR-Revision-Taskforce, 2009]; it includes elements such as class, attribute and method. Using a design process based on the UML metamodel implies to instantiate such a concepts while producing the model of the system. In other words, the designed system may include the *Sensor* class that is an instance (at the model level of abstraction) of the *Class* type defined in the metamodel. In the same way most agent oriented design processes underpin a metamodel of concepts (such as *agent, role, task* and so on) to be instantiated, hence designed in, at least, one activity of the design process.

Moreover the metamodel of the system (or of the class of systems) to be developed with one design process is at the base of the approach we follow for developing new ad-hoc design processes [Seidita *et al.*, 2009]. Specifically the authors made several experiences in ad-hoc design process construction and in the past developed a Situational Method Engineering approach here used for creating PASSIC [Chella *et al.*, 2009a] [Chella *et al.*, 2009b].

Situational Method Engineering [Brinkkemper *et al.*, 1996][Brinkkemper *et al.*, 1999][Harmsen *et al.*, 1994][Ralyté, 2004][Henderson-Sellers, 2006] is the discipline aiming at creating design processes mainly basing them on the reuse of portion of existing design processes; the born of Situational Method Engineering follows up the more and more increasing request for ad-hoc design processes arisen from the acknowledgement that it does not exist an unique standard design process fitting all possible situations. Very often today the (wrong) trend is to construct home methods with high costs for developing them and training the involved personnel. Situational Method Engineering provides means in order to solve these problems.

Our work is mainly focussed on the use of SME in the construction of customized multi-agent oriented design processes. In literature the *Method Fragment* is the core concept of SME and different well known approaches [Cossentino *et al.*, 2007][Henderson-Sellers, 2006][Mirbel & Ralyté, 2006] present different definitions and descriptions of that but they all share the assumption that whatever design process can be decomposed into (or it is composed of, if we use a bottom-up point of view) self-contained components. By following a SME approach the *method designer* (the person devoted to create the new design process) has at his disposal a rich repository of components (or method fragments or simply fragments), coming from existing design processes, from which he can retrieve the best fragments for his own needs.

Since our target is about multi-agent systems design processes during the latest years we specialized the SME approach for our aims. The solution we found (PRoDe - PRocess for the Design of Design PRocesses) [Seidita *et al.*, 2009] follows the principles of SME and it is mainly based on the adoption of a multi-agent system (MAS) metamodel for carrying out the selection and the assembly of fragments and the main element used is what we call a *process fragment* [Cossentino *et al.*, 2007]. The whole PRoDe process pivots on the process fragment concept and it is composed inthis way: the metamodel of the systems that will be constructed with the new design process is created starting from process requirements analysis; requirements are identified in terms of the *development context* (available resources and the skills of people involved in using the new process), *problem type* (the specific solution strategies for a class of problems), and *organization maturity* (for instance evaluated by using SEI-CMMI [SEI, August 2006]).

A design process is created by assembling process fragments stored in a repository or ad-hoc created from scratch; it is of fundamental importance to have a technique for establishing how to link and to realize the matching between the inputs of one fragment and outputs of another one. For this purpose, we use the MAS metamodel; by analyzing how the elements are each other related we create a list allowing to establish the priority and the fragments sequence realizing what we call the component diagram that at the end of the process will define1 with the new design process.

PASSIG [Seidita *et al.*, 2007] is the design process we created for experimenting a goal oriented design process, specifically we decided to modify PASSI [Cossentino, 2005] and to make it suitable for a goal oriented requirements analysis. Our main requirements were to have an agent design process including the use of ontology and communications with a FIPA compliant structure and to have a requirements analysis in a goal driven fashion, being focussed on starting from PASSI we decided to use some process fragments coming from Tropos [Bresciani *et al.*, 2004] for merging them with those from PASSI.

3.2.1. *PASSIC - Requirements and Resulting Metamodel, the SME approach*

By following the PRoDe approach the first commitment is to analyze the new process requirements; this phase results in the core metamodel useful for the fragments selection and then their assembly.

As described in section 2 our aim is to engineer, design and develop robotic systems able to detect the differences between expected and real behaviour and to autonomously learn and eventually tune parameters in order to react to novel situations. Hence robotic systems base their behaviour on the continuous interaction among brain, body and environment, and the continuous comparison of the real and the expected data, the robot is able to gain perceptual experiences and to simulate a self-conscious behaviour reacting to external stimuli.

The set of requirements elicited from the above specifications are reported in Table 1; a detailed description of the process used to elicit and define strategies for requirements and then to create the metamodel elements is out of the scope of this paper, it has been already discussed in [Chella *et al.*, 2009a].

The right column of the table shows the list of metamodel elements resulting from the requirements (reported in the left-most column). They are related to the self-conscious portion of the robotic system. Once this set of elements with their definition and features has been identified the new core metamodel is created. This latter is the result of the integration of elements allowing a goal oriented analysis,

Table 1. Some of the Design Process Requirements and the Resulting Metamodel Elements

Requirement	**Sub-Requirement**	**MAS Metamodel Element**
Developing a robotic architecture composed of two main levels of abstraction: one or more robots, and inside each robot, a society of agents responsible for the basic robot's functionalities (for instance sensors management, vision, . . .)	A robot is composed of: • Rational agents: agents with reasoning/planning capabilities and a knowledge of the world. • Reactive agents: agents adopting the stimulus-reaction loop. • Devices: artifacts [Omicini *et al.*, 2008] representing robot's hardware components. • Conscious agent: an agent providing self consciousness features to the robot	Robot, Reactive Agent, Rational Agent, Device, Artefact.
Each robot can interact with other robots, the objects in the world (regarded as artifacts) and external agents; several robots can form a society of robots.	The robot has the ability of recognizing and distinguishing stimuli coming from the outer word (sensorial perception) and stimuli coming from the robot body (proprioceptive sense).	Conscious Agent, Stimulus, Goal, Plan, Action.
	Perception is supervised by means of perception loops. The conscious agent is responsible for the execution of the perception loop that is composed of the following step: • The robot perceives the outer world (sensors) and the inner world (propioceptive perceptions) • Perceptions are used to build a 3-D simulation of the mission (anticipation) • Anticipation is compared with the perceived scene during mission execution • Parameters are tuned according to results of that matching	Stimulus, Propioceptive Stimulus, Sensorial Stimulus, Goal.
	Several perception loops can be active at the same time for taking care of different aspects of robot management.	Conscious Agent.
The robot moves in an unstructured environment and it is able to autonomously interact with it.	• The robot has a model of the environment; • The robot owns a model of the "self"; • The environment is composed of objects that can be agents and artifacts - an artifact is a passive, function-oriented entity with no means of autonomy and control encapsulation.	Knowledge, Artefact, Agent, Stimuli.
The robot detects the differences between required and observed behavior	Using the metaphor of testing.	Perception Test, Simulated Act, Log.
	Supporting the creation of a simulated environment (anticipation) in order to compare it with the perceived situations.	Simulated Act.

environment representation and self-conscious ability; because of its size only a portion of it is shown in Fig. 3. In this portion of the metamodel we highlighted some of the most important elements of the table for which we used the metaphor of software testing and for which we were able to create the set of design process activities, successfully integrated with PASSI2 and PASSIG in order to create our design process for robot self-conscious system (PASSIC). Besides this portion of metamodel clearly represents a first attempt towards the creation of a conscious system metamodel.

Some of the most important elements of the metamodel are: the *Robot*, that is the whole system endowed with reasoning capabilities, it is composed (among the others) of *Rational Agent*s purposefully able to pursuit their own goals. Each Rational Agent is proactive and it uses a representation of the environment in order to decide the best action to be done in a given situation. The Robot is also composed of *Conscious Agents* implementing the robot's awareness of the self.

The Robot perceives several stimuli, the *Stimulus* is an event coming from the environment and perceived by the robots sensors (*Sensorial Stimulus*) or it is a robot internal event (*Propioceptive Stimulus*), for instance the information about a change in wheels position.

Each time a robot has to reach a goal, it establishes a plan, based on the knowledge it has about the environment, the goal itself and the set of stimuli coming from the environment, it produces expectations about the result of the plan to be activated, hence the simulation of its actions and the related results. Once the plan has been terminated the robot has to "test" (*Perception Test* element in the metamodel is the robot's self-consciousness essential component) hence to compare the simulated acts with its current situation coming from propioceptive and sensorial stimuli and produces the Log. Therefore the Simulated Act constitutes the expectation whereas the Log is the result of the comparison between the simulated acts and the ongoing situation as it is reconstructed by using propioceptive and sensorial stimuli.

As mentioned during the design process construction with PRoDe, once the metamodel has been created, one process fragment has to be identified for designing one (or more) element(s) of the metamodel; the PASSIC metamodel contains many elements that can be realized by using parts of process coming from PASSI and PASSIG; this because we did not start from scratch in constructing this new process but we were well grounded on our previous works.

For the new elements we found useful, on the base of the elements definition and the relationships between them, to reuse portions of process coming from UP (the ones relating to software testing); do not let the latest statement surprise, what's the matter of using the metaphor of software testing? We have to design a portion of system able to produce an anticipation of the mission execution results and to compare this result with the real outcome of the system itself.

When facing the test design phase designers perform activities aiming at identifying the system functionalities to be tested and also the resources and the objective

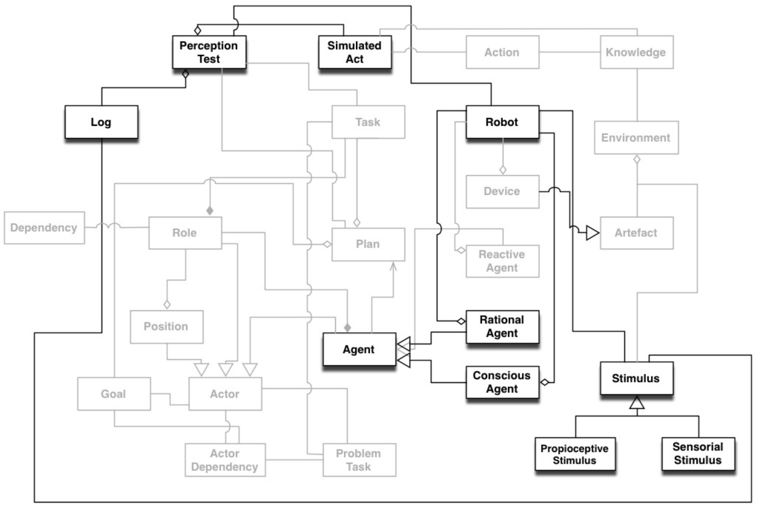


Fig. 3. A Portion of the PASSIC Metamodel

of test, then while the test is executed the real performances of the system are compared with the oracle, the expected outcome, in order to investigate and to identify defects and faults, the result is the log.

Only by simply modifying this portion of process and the related outcome we were able to produce a process fragment through which the designer can create a test plan starting from the system requirements/goals, the environment description, the system architecture, all the robotic components and how they are interconnected. In this process fragment, the element we called *Simulated Act* is defined, this is composed of its name, the location of the robotic module to be tested, a set of inputs and finally the definition of the expected system behaviour.

The portion of process designing the *Log* element of our metamodel results in an artefact where the criteria for evaluating the results of the comparison are reported.

Fig. 4 shows the whole PASSIC design process, as it can be seen it follows an iterative/incremental life cycle and it is composed of three phases (and related resulting models): *i) System Requirements*,*ii) Agent Society* and *iii) Agent Implementation*. Each of them is devoted at producing a model, in the first one a model able to give information about the goals of the system and the agents involved in realizing them is given. In the second model, all the elements of the society of agents are described together with an ontological description of the environment the agents live in, their communications and the description of the autonomous part of the system devoted to create the expectation about the results of the mission. The third is the model of the system's architecture in terms of classes and methods, the structure and the behaviour of each agent and how agents are deployed.

Each phase produces a document that is usually composed aggregating UML

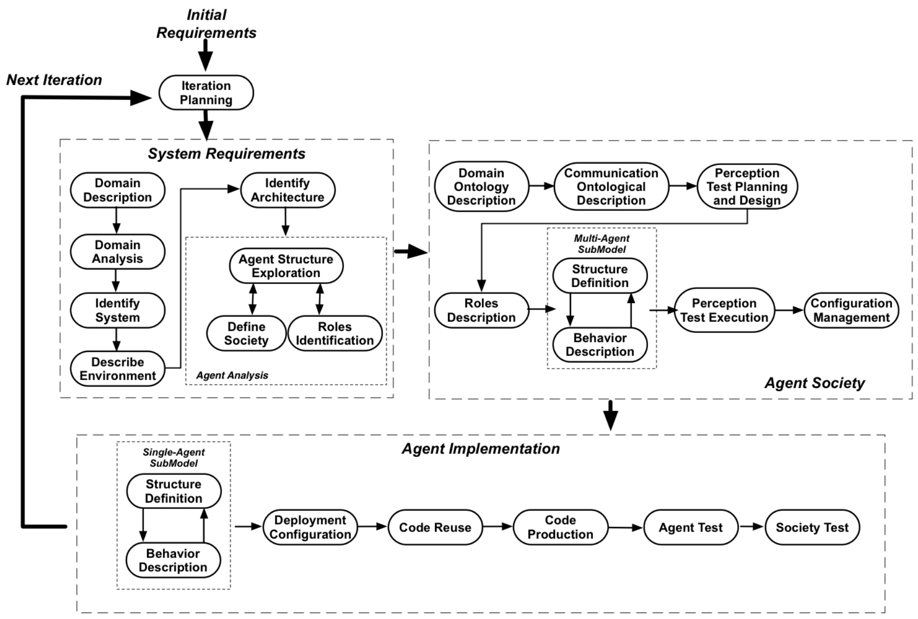


Fig. 4. The PASSIC Design Process Phases and Activities

models and work products produced during the related activities and it is composed of one or more sub-phases each one responsible for designing or refining one or more artefacts that are parts of the corresponding model.

In the next section an experiment will be reported where the described PASSIC process has been employed, tested and verified.

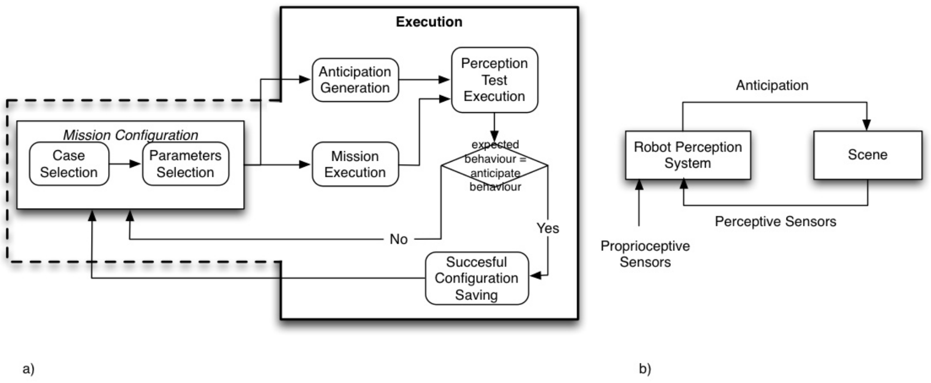


Fig. 5. The CSDP in Relation with the Perception Loop

4. Implementing the Perception Loop - an Experimental Setup

PASSIC and the self-Conscious System Development Process have been used for developing a robotic system composed of two robots performing a football match. The experiment served as a testbed for bug-fixing the design process and for testing the validity of the whole process; in particular in the following some discussions about how the robots have been provided with the capability of taking conscious decisions while interacting with the environment will be provided. More specifically, the implementation of the perception loop will be shown.

For our experiments we used the NAO humanoid by Aldebaran Robotics and because of NAO is the official robotic platform for the Robocup league we found useful to exploit it for realizing a match where one NAO has the role of *forward* and the other one the role of *defender*; the forward's main objective is to make a goal whereas the defender has to try to prevent the forward from that. The two NAOs' environment consists of the game field with the goal and the boundaries, each NAO has its own knowledge of the environment and it has been designed for pursuing a different objective (according to the football-player role). The self-conscious part has been realized by means of the perception loop; it has to be in control each time a situation, the NAO has not been designed for, occurs.

The whole system has been designed using PASSIC and goals and requirements (such as identifying the ball, interfacing with the simulator, avoiding the boundaries, communicating with the database, . . .) have been identified. In the following we will detail only the part of experiment dealing with the perception loop and how the database of cases and configurations operates in the learning and deciding actions.

As it can be seen in Fig. 5 b) by exploiting the perceptive and the propioceptive sensors the robot can generate the anticipation of the mission that realizes a particular goal, when the mission is finished the perceived scene is compared with the anticipated one thus making the robot able to experiment the subjective experience about a specific case; the same is seen in Fig. 5 a) under the point of view of the whole development process.

Starting from a designed mission configuration a set of specifications are sent to the robot and to the system generating the anticipation at the same time; in our experiment we used the 3D robot simulator by Cyberbotics: Webots. By using Webots the NAOs' movements and their environment can be simulated. In our case the simulator is not used for analyzing and virtually investigating the robot behaviour but only for generation the anticipation, the simulator represents the human mind that continuously imagines the results of an action and that compares it with what is really perceived. The two systems, the NAO and the Webots, work separately and they exchange data only at the end of each mission. Of course the robots reasoning system is able to decompose the mission in sub-missions according to sub-plans available for intermediate goals achievements. In this case what above reported is similarly applied to sub-missions.

When the mission terminates or when an unknown situation occurs (in this case a stop condition is revealed by the NAO and it stops its mission) the results, in form of parameters and post-conditions determining NAO's state, from the Webots are compared with the real ones, this means what we called the perception test is executed; the test is the base for the subjective experience and for the self-conscious ability, in the same way a human being does.

If the comparison result is true the anticipated scene matches with the real one and the robot has reached its objective; in this situation the adopted configuration can be considered successful with regard to the goal pursued and it can be saved in the robot memory for future reuse; each time the robot has to pursue the same goal it can retrieve the configuration that once made it able to realize its mission.

If the comparison faults then it is like if the robot does not know how it can behave to solve a problem. For instance this can happen when the robot suddenly meets an unexpected obstacle and so on; in the same situation a conscious human being would use his mind and his memory in order to retrieve the most used or the most useful action done in a case like the faced one. In our approach the robot is designed for emulating the human being that instinctively tries to apply corrective actions to an unexpected result of another action by using his own experiences, memory etc. In a similar the robot is designed for querying the cases and configurations database in order to retrieve the most useful couple of cases and configurations, hence a specific mission configuration. If it does not exist the robot undertakes a random action among the set of actions that can be useful for pursuing a specific goal. For instance if the goal is to go towards a door the possible actions to be proved can be all those related to legs movements. This part of the system can be implemented by using common sense reasoning and the Cyc ontology with its inference engine; this portion of the work is still in progress and we are exploiting our previous experiences in developing a robotic museum guide able to plan a path and to avoid obstacles by using Cyc [Macaluso *et al.*, 2005].

When the activation of random actions brings to the right solution it can be saved in the database. According to the presented arguments, mission configuration is a part of the self-conscious system belonging to both the design phase and the execution one.

The configuration phase starts with the Case selection activity where the case that most likely will accomplish the mission (or sub-mission) is to be chosen. As a result of the design phase, at execution time the robot will be able to execute several different tasks. Some of them are native tasks (a kind of inner abilities with whom the robot has been built and micro-programmed), some others have been purposefully developed in order to enable the specific mission accomplishment.

Some cases are reported in Fig. 6 a). Each case concurs to the achievement of a specific goal and it can be successfully applied only when some pre-conditions are verified. Each case contains one task that describes the behaviour the robot may exhibit to achieve its aim (a kind of plan). Usually a task (Fig. 6 c)) is a parameterized behaviour whose instantiation at runtime is subject to the selection of the correct values for some parameters. A task may also include other tasks in a hierarchical composition from simplex to more complex behaviours.

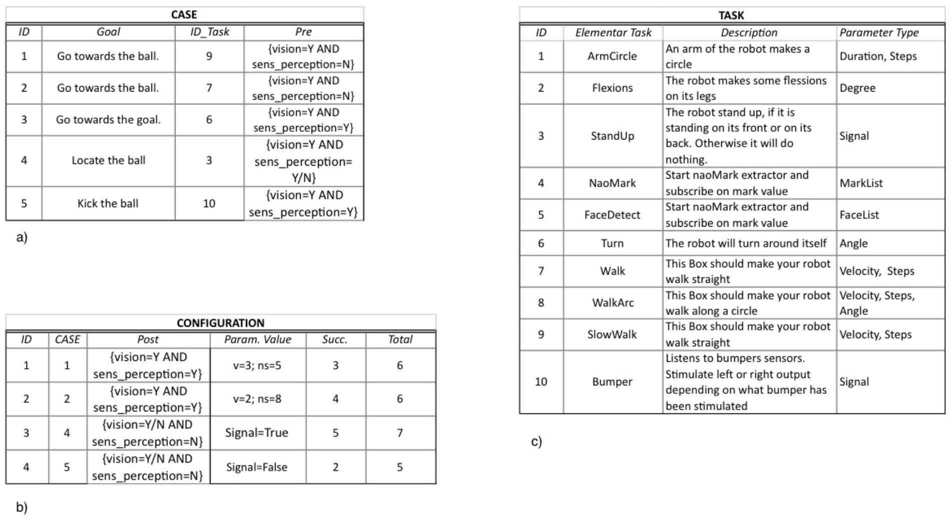
Just to provide an example of configuration, the Walk task requires the specification of a *distance* parameter. When the execution of a new mission starts, the robot uses its reasoning capabilities to select the first goal to be achieved. If this goal is not present in the list of goals related to cases (Fig. 6 a)) the robot tries to decompose the goal until some goals from table in Fig.6 a) are found. Generally speaking each goal may be achieved through different cases. As a first attempt, the robot selects the case that most likely will succeed by accessing the results of its past executions reported in Fig. 6 b). The rationale that is behind that is in the commonsense consideration that if in a specific situation (detailed by the pre-conditions) the application of a specific choice (case selection) proved useful to achieve a specific goal, then this is worth to be tried again. Once the case is selected, the next step consists in the definition of the execution parameters for the task linked to the case. The same case (and related task) may have been applied several times for the same goal (for instance reaching the ball) with different parameters because of different environment conditions that are not exactly reported in the pre-conditions or whose physical laws are not known. For instance, it is sufficient to consider that the case *2 (Going towards the ball)* may be successful in reaching the ball if this latter is in front of the robot at a distance of 10 centimetres and we execute the *Walk* task with a *distance* value of 0,10 (robot's specifications require such a distance in meters). The values of parameters used to instantiate a case (and its related task) are stored in the table Configuration reported in Fig. 6 b). This table also reports how many times the case has been instantiated with the specific set of configurations and the corresponding number of successes.

The next time the robot perceives the ball in front of itself, it tries to use the same case with the same parameters. This time the ball may be at a different distance (let us suppose 15 centimeters). The execution of the case is not successful but the robot may appreciate an improvement in its situation (it is now nearer to the goal).

5. Conclusion

In this paper we presented a process for engineering the development of self-conscious robotic systems from analysis to implementation. The proposed process is based on the assumption that the robot self-knowledge can be achieved through the perception loop among brain, body and environment. The continuous interaction with the environment lets the robot reflect about itself and be engaged in autonomous missions.

There can be different perception loops contemporarily in action in a robot, each of them taking care of a different aspect of the subjective experience and besides there can be different orders of perception loops, the lower one relates to the experience about the external world and the higher one to the inner world



CASE

ID	Goal	ID_Task	Pre
1	Go towards the ball.	9	{vision=Y AND sens_perception=N}
2	Go towards the ball.	7	{vision=Y AND sens_perception=N}
3	Go towards the goal.	6	{vision=Y AND sens_perception=Y}
4	Locate the ball	3	{vision=Y AND sens_perception= Y/N}
5	Kick the ball	10	{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
4	5	{vision=Y/N AND sens_perception=N}	Signal=False	2	5

b)

TASK

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

c)

Fig. 6. The Cases and Configurations Database

knowledge.

The perception loop has been, in the past, tested and implemented on *Cicerobot* and on *Robotanic* robots respectively offering guided tours in the Archeological Museum of Agrigento and in the Botanical Garden of Palermo [Chella & Macaluso, 2009][Barone *et al.*, 2008].

The work presented in this paper represents a way of formalizing and engineering the implementation of the perception loop onto a robotic system mainly exploiting the use of agents. We aim at realizing a robotic system where different perception loops are in action at the same time and each of them is managed by a society of agents that cooperates in order to achieve the robot global goals, in the same way the sensorial parts of a human body do. At the same time we claim the use of different orders of perception loops managed by the society of agents at different level of a social hierarchy.

The self-Conscious System Development Process (CSDP) has been shown together with PASSIC activities. PASSIC offers means for developing and implementing the reflective part of the robotic system since it has been created for this scope by integrating design process parts coming from PASSI2 and from PASSIG with other portions of process ad-hoc created for the loops realization.

The whole CSDP process and PASSIC have been tested and fixed through an experiment resembling a Robocup setup. This experiment principally allowed us to define and to fix the *Case* and *Configuration* database, the portion of process regarding parameters tuning during an incoming unknown situation and the learning phase. We also provided means for creating the database in a way that allows to use different robotic platforms, the form of cases and of the configuration has been established but it can be customized on the base of the robotic platform one wants to use.

Acknowledgments

Authors kindly wish to thank Prof. Antonio Chella for his invaluable suggestions and constant support.
This research has been partially supported by the EU project FP7-Humanobs.

References

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

Barone, R., Macaluso, I., Riano, L. and Chella, A. [2008] "A brain inspired architecture for an outdoor robot guide," in A. Samsonovich (ed.), *Proc. of AAAI Fall Symposium on Biologically Inspired Cognitive Architectures BICA '08* (AAAI Press, Menlo Park, CA.).

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

Brinkkemper, S., Saeki, M. and Harmsen, F. [1999] Meta-modelling based assembly techniques for situational method engineering, *Information Systems, Vol. 24* **24**.

Brinkkemper, S., Welke, R. and Lyytinen, K. [1996] *Method Engineering: Principles of Method Construction and Tool Support* (Springer).

Chella, A. [2007] Towards robot conscious perception, in A. Chella & R. Manzotti (eds.), *Artificial Consciousness* (Imprinting Academic, Exter, UK).

Chella, A. [2009] A robot architecture based on higher order perception loop, in A. Hussain (ed.), *Brain Inspired Cognitive Systems 2008* (Springer Science+Business Media), p. (in press).

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

Chella, A., Cossentino, M. and Seidita, V. [2009b] 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)* .

Chella, A. and Macaluso, I. [2008] "Higher order robot perception loop," in B. E. Springer-Verlag (ed.), *BICS 2008 Brain Inspired Cognitive Systems.*

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

Chella, A. and Manzotti, R. [2009] Machine consciousness: A manifesto for robotics, *International Journal of Machine Consciousness* **1**(1), 33 – 51.

Cossentino, M. [2005] From requirements to code with the PASSI methodology, in *Agent Oriented Methodologies*, chap. IV (Idea Group Publishing, Hershey, PA, USA), ISBN 1-59140-581-5, pp. 79–106, `http://www.idea-group.com/books/details.asp?id=4931`.

Cossentino, M., Gaglio, S., Garro, A. and Seidita, V. [2007] Method fragments for agent design methodologies: from standardisation to research, *International Journal of Agent-Oriented Software Engineering (IJAOSE)* **1**(1), 91–121.

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

Harmsen, A., Brinkkemper, S. and Oei, H. [1994] "Situational method engineering for information system projects," in *Methods and Associated Tools for the Information Systems Life Cycle, Proceedings of the IFIP WG8. 1 Working Conference CRISí94*, pp. 169–194.

Henderson-Sellers, B. [2006] "Method engineering: Theory and practice. in D. Karagiannis & e. Mayr, H. C. (eds.), *Information Systems Technology and its Applications.*, pp. 13–23.

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

Macaluso, I., Ardizzone, E., Chella, A., Cossentino, M., Gentile, A., Gradino, R., Infantino, I., Liotta, M., Rizzo, R. and Scardino, G. [2005] Experiences with cicerobot, a museum guide cognitive robot, in *Lecture Notes in Computer Science* (Springer-Verlag GmbH), pp. 474–482.

Manzotti, R. [2006] A process oriented view of conscious perception, *Journal of Consciousness Studies* **13**(6), 7 – 41.

Mirbel, I. and Ralyté, J. [2006] Situational method engineering: combining assembly-based and roadmap-driven approaches, *Requirements Engineering* **11**(1), 58–78.

Omicini, A., Ricci, A. and Viroli, M. [2008] Artifacts in the A&A meta-model for multi-agent systems, *Autonomous Agents and Multi-Agent Systems* **17**(3), 432–456, doi:`10.1007/s10458-008-9053-x`, `http://www.springerlink.com/content/l2051h377k2plk07/`, special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems.

Ralyté, J. [2004] 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* , 271–282.

Rockwell, W. [2005] *Neither brain nor ghost* (MIT Press).

SEI [August 2006] Technical report of the software engineering institute of the carnagie mellon university. SEI. CMMI for Development Version 1.2.

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

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

UMLR-Revision-Taskforce [2009] Omg uml specification v. 2.2, Object Management Group.

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