Biologically Inspired Cognitive Architectures II:
Papers from the AAAI Fall Symposium (FS-09-01)

Towards a Methodology for Designing Artificial Conscious Robotic Systems

Antonio Chella[1], **Massimo Cossentino**[2] **and Valeria Seidita**[1]

[1]Dipartimento di Ingegneria Informatica - University of Palermo, Viale delle Scienze 90128 Palermo, Italy
[2]Istituto di Calcolo e Reti ad Alte Prestazioni, Consiglio Nazionale delle Ricerche, Viale delle Scienze, 90128 Palermo, Italy
chella@dinfo.unipa.it, cossentino@pa.icar.cnr.it, seidita@dinfo.unipa.it

Abstract

Engineering artificial conscious robotic systems, able to perceive, think and act in an unstructured environment is a very challenging issue. Basing on the results of the experiences made in the latest years about modeling the perception loop of a robot and about the creation of ad-hoc methodologies for engineering complex systems, we developed an initial model of an artificial conscious system and extended a well known methodology (PASSI) for engineering the elements we identified as composing such a system.

Introduction

Perception, also including memory, is one of the most important features a robotic system must present. In (Chella and Manzotti 2007) it is argued that a perception process can be modelled and implemented as a continuous interaction loop among brain, body and environment; by continuously comparing actual and expected "data" coming from the environment the robot achieves the ability to gain perceptual experience and to react by simulating a conscious behaviour to the external stimuli. The current perception loop has been implemented in a hybrid architecture (Chella and Macaluso 2009), integrated with a reactive system where a set of predetermined behaviours had been set; this architecture does not provide means for being used in a fully dynamic environment.

The problem of interacting with a dynamic and unstructured environment is one of the most important topics of today's research in artificial consciousness (Chella and Manzotti 2007); one possible solution is to provide the robot with the ability of reflecting upon itself and the world around it, its actions and perceptions while performing actions; in other words to provide the robot with a kind of *self-conscious ability*.

We present the results of our studies towards developing a design methodology for artificial conscious robotic systems where the conscious perception model is implemented in order to provide the robot with the ability of acting in a unstructured environment. This part of the work already has a partial result in the identification of a very general metamodel for an artificial conscious system that has been mapped onto the already proposed conscious perception model. The metamodel is the starting point for us to apply a Situational Method Engineering (SME) approach for the creation of the portions of methodologies used for extending PASSI (Process for Agent Society Specification and Implementation), a known agent oriented design process for developing robotic systems.

Theoretical Background

Specific design methodology is required each time one wants to develop a robotic system and today there are not standard and reusable methodologies. In the past years we developed several design processes (Chella et al. 2006)(Cossentino and Seidita 2004)(Cossentino, Gaglio, and Seidita) following the approach based on Situational Method Engineering paradigm we fixed in these years (Cossentino et al. 2007)(Seidita et al. 2009). In the following subsections an overview on the used SME approach, the PASSI design process, and the robot perception loop will be given.

PRoDe - A Situational Engineering Approach

Situational Method Engineering (Brinkkemper, Lyytinen, and Welke 1996)(Kumar and Welke 1992) is the discipline devoted to the construction of ad-hoc design processes manly following an approach based upon the reuse of techniques, methods and guidelines coming from portions of already existing design processes.

The main element used in such an approach is what we call a *process fragment* (Cossentino et al. 2007). The whole process for the creation of a new design process pivots on the process fragment concept; well defined techniques for the definition of fragments from existing design processes, for the selection of the right ones and for assembling them in a right way to form a new design process are needed.

The process we developed for doing that (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 MAS metamodel for carrying out the selection and the assembly of fragments.

Starting from process requirements analysis the metamodel of the systems, that will be constructed with the new design process, is created; we identify requirements in terms

of *development context*, the available resources and competencies of the team will use the new process, *problem type*, the specific solution strategies for a class of problems and *organization maturity*, from SEI-CMMI (SEI).

The PASSI Process

PASSI (Process for Agent Societies Specification and Implementation) (Cossentino 2005) is a design methodology for developing and implementing multi-agent systems (MAS), it uses models from object oriented software engineering, artificial intelligence approaches and the UML notation, and it is able to drive the designer from the requirements analysis phase to the code implementation through five models composed by twelve sequential and iterative activities used to produce the MAS specification.

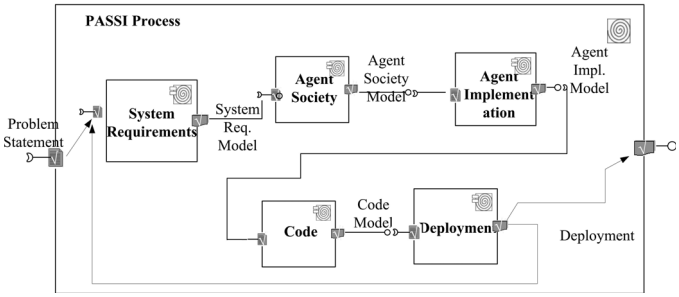

Figure 1: The PASSI Design Process

Figure 1 shows, through a SPEM (SPEM) process component diagram, the models and phases of PASSI. The **System Requirements Model**, produces a model of the system requirements and is composed of four different activities: *Domain Description*, a functional representation of the system through use cases. *Agent Identification*, by grouping one or more use cases agents are identified, each agent have a set of requirements assigned to it. *Role Identification*, describes agents' interactions by using traditional scenarios. *Task Specification*, represents the agent's behavior by showing relationships among received stimuli and its behavior.

The second model is **Agent Society Model**. It is a model of the agents' society under the point of view of their knowledge in the environment they live in and the communication they perform. It is composed of: *Domain Ontology Description* where the elements occurring in the system domain are represented in term of concepts, predicates, activities and relationship among them. *Communication Ontology Description*, here each agent is described in terms of its ontology and agent's communications represent their relationships with specific language and protocol. *Role Description* describes the roles played by agents in the society and all the tasks involved in each role.

The third model is **Agent Implementation Model**. It is a model of the solution architecture, it is composed of: *Agent Structure Definition*, describes the internal structure of the system both at the single-agent and multi-agent level of abstraction. *Agent Behavior Description*, describes the agent's behavior by showing the flow of events among main agent classes and their inner classes.

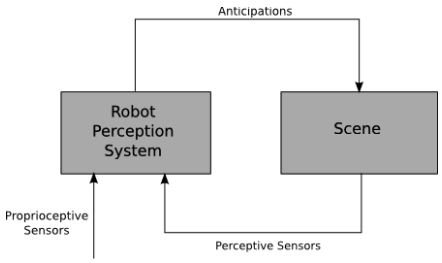

Figure 2: The Robot Perception Loop - redrawn from (Chella and Macaluso 2009)

The **Code Model** is the model of the solution at the code level. It is largely supported by patterns reuse and automatic code generation.

Finally **Deployment Model**. It is a model of the distribution of the parts of the system across hardware processing unit. *Deployment Configuration* describes the allocation of agents in the units and any constraint on migration and mobility.

PASSI proved to be very useful for engineering, developing and implementing robotic applications.

The Robot Perception Loop

The robot perception loop described in (Chella and Macaluso 2008)(Chella and Macaluso 2009) and presented in Figure 2 is mainly 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 robot's prediction/knowledge/detection about the relevant events around it.

Not all the perceived scenes (hence occurring stimuli and events in the environment) are useful for generating the anticipation of scenes, the attention mechanism takes care of generating expectations and hypothesis to be verified and adjusted, see (Chella and Macaluso 2008) for further details.

As it can be seen from the figure, a loop there exist among the perception and the anticipation, so each time some parts of a perceived scene, in what it is called the current situation, matches with the anticipated one, then the anticipation of other parts of the same scene can be generated.
The presented perception loop can model the robot's ability of sensing the word around it; in (Chella and Macaluso 2008) it is argued that in a real operating robot, we may have many perception loops in action, thus realizing robot self-consciousness, the robot's inner world conscious perception.

The New Methodology Requirements and Metamodel

According to the PRoDe process, the new methodology should be defined on the basis of the requirements it should

fulfill. Actually, we adopted the perspective proposed by Osterweil in (Osterweil 1987).

According to that, and as prescribed by PRoDe, the requirements our new process should satisfy the following four categories: *(i)* Problem Type: requirements related to the specific category of problems the system, developed with the new methodologies, will aim to solve; *(ii)* Development Context: requirements related to the context where the software will be developed; these include also considerations about development environment and personnel skills; *(iii)* Organization Maturity: requirements related to the maturity of the organization in developing systems that are similar in nature or developed with similar processes. Further details about that can be found in the SEI specifications (SEI). With regards to the Problem Type, the new methodology should support:

- **System Architecture**. The development of 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, . . .).
- **Robot Structure**. A robot is composed of: *(i)* Rational agents: agents with reasoning/planning capabilities and a knowledge of the world; *(ii)* Reactive agents: agents adopting the stimulus-reaction loop; *(iii)* Devices: artefacts (Omicini, Ricci, and Viroli 2008) representing robot's hardware components. *(iv)* a Conscious agent: an agent providing self-consciousness features to the robot
- **Interactions**. Each robot should be able to interact with other robots, the objects in the world (regarded as artifacts) and external agents; several robots can form a society of robots.
- **Perception**. The robot has to be endowed with the capability of perceiving the environment around it and its own inner world. 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). Perception is supervised by means of perception loops. Several perception loops can be active at the same time for taking care of different aspects of robot management.
- **Robot Knowledge**. The robot moves in an unstructured environment and it is able to autonomously interact with it. This entails that: *(i)* The robot has a model of the environment; *(ii)* The robot owns a model of the "self"; *(iii)* 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.
- **Expectation Design**. The metaphor of software testing is adopted for detecting differences between expected and observed behaviour. This detection process represents the robot conscious reasoning. The scenario is perceived by the robot (the observed behaviour) and it is compared with the expected behaviour (anticipation of the perception loop) in order to detect a not-expected situation and to activate all the fixing actions. The test planning result is the simulation of the actions the robot performs in order to perceive a goal; this defines the expected behaviour that is the test *oracle*.

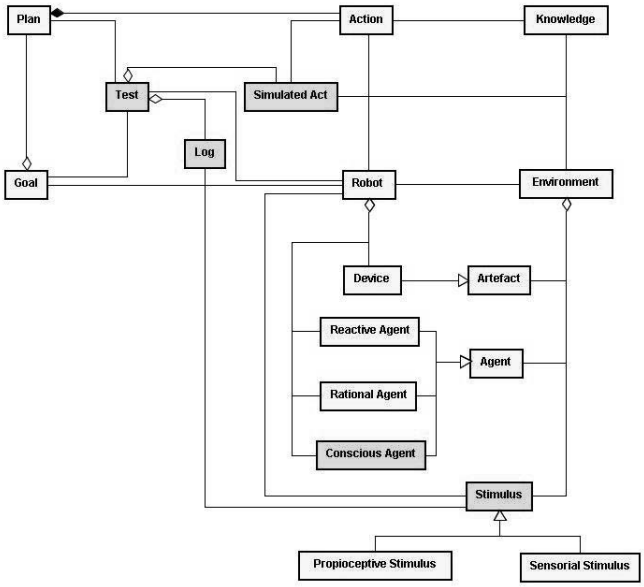

Figure 3: The Conscious System Metamodel

As regards the Development Context and The Organization Maturity, the robotic system will be developed in a research lab by people skilled with robot programming and multi-agent system concepts, but the adoption of a perception-driven perspective is new to all lab members and specific guidelines will be necessary to improve product quality.

The Metamodel

The result of the process requirements analysis is a metamodel where all the above mentioned requirements have been translated into a set of metamodel elements, each other related. In this section we show (see Figure 3) only a portion of the obtained result, the one concerning the conscious part of our robotic systems useful for engineering the perception loop. The definitions of some elements are reported in table 1.

The core of the metamodel is composed of the *Robot* and the *Environment* concepts; *Environment* represents the world where the robot lives and it interacts with, we also consider the inner robot's world as part of the environment, in fact environment is composed of three concepts: *Artifact*, *Agent* and *Stimulus*.

Artefact is the part of environment (external to the robot) that does not offer any autonomous capability, it can be used as a resource or can simply be an inanimate part of the world, it can also be a *Device*, one of the physical robot's component. The *Agent* concept - *Reactive* or *Rational Agent* - represents the society of agents taking care of all the robot's functionalities or each external autonomous entity (on the

Term	Definition
Robot	The whole system (hardware and software) endowed with reasoning capabilities.
Artefact	It is a part of environment that does not offer any autonomous capability, it can be a resource, an inanimate part of the world, or it can also be a *Device*, one of the physical robot's components.
Environment	The world the robot is located in and interacts with, it includes the inner robot's world. It is composed of three concepts: *Artefact*, *Agent* and *Stimulus*.
Agent	An agent is capable of acting in its environment, it is autonomous (it has control over its own behaviour based on its internal or external stimuli), it can communicate with other agents, it is capable of perceiving its environment
Reactive Agent	A Reactive Agent acts according to a perception-act loop. It implements no specific decision process and reactions to stimuli are cabled in the agent code.
Rational Agent	A Rational Agent purposefully pursuit its own goal. It exhibits proactivity and it uses a representation of the environment to decide about the best action to be done.
Conscious Agent	The Conscious Agent implements robot's awareness of selfness.
Stimulus	It is an event coming from the environment and perceived by the robot's sensors or a robot internal event, for instance changed wheels position.
Test	It is the self-consciousness essential component. It has a goal (testing the plan adopted by the robot for pursuing its objective), a plan (selected for pursuing the objective) a set of Simulated Acts and a Log.
Simulated Act	It constitutes the expectation. Once a plan is defined, an expectation about plan results can be calculated by simulating the plan actions and estimating their results.
Log	It is the result of the comparison between the simulated acts and the ongoing situation as it is reconstructed by using propioceptive and sensorial stimuli.

Table 1: Metamodel Elements Definiton

contrary of artefact) the robot interacts with in order to pursue a *Goal*.

In this metamodel we consider the robot as the unique entity (an individual) endowed with autonomous reasoning capabilities about the self and the environment. The robot has one or more goals to be reached, they are realised by means of *Plans* each of them is accomplished by a set of actions. The *Knowledge* the robot has about the environment and about itself is affected by each action performed during its lifetime. An *Action* is a kind of act - a physical robot act or a communication between the robot and an agent - resulting in a state change of the environment (either agents or artefacts); each state change is traced in the knowledge.

The robot has conscious capabilities in the sense that it can reason about the required and observed behaviour by means of a *Test* composed of a set of *Simulated Act*s and a *Log*; 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 compares the simulated acts with its current situation coming from propioceptive and sensorial stimuli and produces the log, hence the set of correlations between the observed and required behaviour.

The log is then used for activating all the required actions in order to correctly follow the established plan. The elements used for modelling robot conscious perception come from the fourth requirement; in the following subsection it will shown how, using the metaphor of software testing, we were able to create a set of design process activities that we integrated with PASSI in order to make a first step towards the definition of a design process for the development of robot conscious systems.

Extending PASSI - Implementing the Perception Loop

The metamodel we described in the previous section is the result of the PRoDe process requirements analysis. In this section, for the scope of this paper, we will now consider only a portion of the metamodel elements - the green coloured ones. They realize the portion of the system exploiting the following features: *(i)* generating expectations about the state of the world (therefore also about itself) as results of a plan execution, starting from the goals and the knowledge on the environment, *(ii)* comparing this "simulated state" with the effective reality after plan execution.

In order to design these elements we decided to implement the *perception loop* as a testing activity and for that we referred to one of the most known design processes in the object oriented area - the Unified Process (UP) (Jacobson, Booch, and Rumbaugh 1999)(Jacobson).

The result was the extraction of two portions of UP (i.e. process fragments) that we modified in order to meet the needs of our problem; the fragments are: *test plan and design* and *test execution*. In UP the first activity aims at identifying the system functionalities to be tested, the resources and the objective of test, whereas the second one aims at executing the test in order to investigate the results and to identify the defects.

In Figure 4 the new test fragments, their activities and the inputs and outputs they need, are shown; these fragments had been integrated in the PASSI Agent Society Model. In the following paragraph we will provide a brief description of them.

The planning activity results in a *test case*, composed of a name, a location, a set of inputs, an oracle and a log. The latter two, respectively represent the expected outcome and the correlation between expected and observed behavior. The execution activity results in a report listing the differences among expected and observed behavior.

According to the portion of design process we are defining, (see the system metamodel in Figure 3) the designer starts its work from system requirements, the goals, the en-

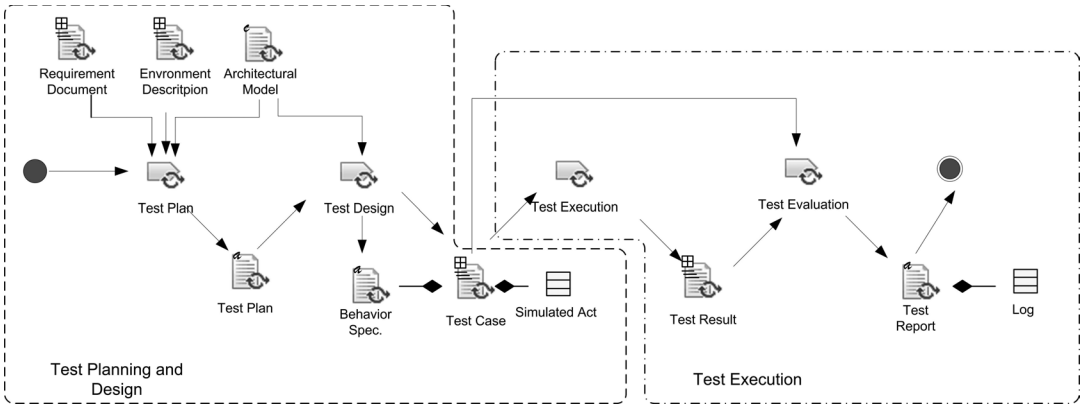

Figure 4: The Testing Activities Extending PASSI

vironment description, the system architecture and all the remaining robotic system components. Based on this, the designer produces a test plan including the definition of the expected behavior; then he/she has to create a test report where the log is designed through a comparison between the expected/simulated behavior and the real observed one.

Therefore we reused from UP the test plan and design portions of process by only modifying the resulting test case; this because we need it to be composed of behavioral specification and not to contain the classic test log.

Planning Test. The aim of the test plan activity is to produce the anticipation, represented by the *Simulated Act* in the metamodel, that is the starting point for conducting the reasoning activities. The Planning activity is composed of two tasks: *Test Plan* and *Test Design*, the first task receives the architectural model as an input; this represents the architecture of the agent society (the robot's components); from this model the designer can identify the portion of the system under test, it is possible to think that each architectural component (or at least the most strategic ones) is managed by a perception loop, thus allowing us to decide which kind of robot functionality is "consciously" valuated. The Requirement Document and the Environment Description are structured documents detailing goals, actions, requirements and environment elements affecting the situation under simulation.

The purpose of the design is to identify and describe the robot 's actions while it interacts with the environment in order to pursue an objective, the result of this task is a *test case* containing, for each module to be tested; the name, the location, all the input data, the oracle, and the expected results, estimated by means of a set of simulated acts.

Executing Test. The aim of this fragment is to design the portion of robotic system devoted to produce the *log* metamodel element, this is the element resulting from the comparison between expected and observed system behaviour. The test case generated from the "Planning Test" fragment is the input document.
The fragment is composed of two activities: *Test Execution* and *Test Evaluation*, both performed by the Designer. During Test Execution, starting from all the simulated acts detailed in the test case, a document describing the test results is produced and then used by the Test Evaluation activity for producing a Test Report describing the *Log* element of the metamodel.

Conclusions

A specific design methodology is required each time one wants to develop a robotic system and today there are not standard customizable or reusable methodologies.

Engineering artificial conscious robotic systems endowed with the *robot perception model* requires modeling and developing a set of robotic components, and above all deeply analyzing and designing how they have to interact. Moreover, according to the kind of application, some specific phases are required, for instance if we were developing a robotic system acting in a static environment we could model it in a way that is different from the case we were modeling a dynamic one.

By following a Situational Method Engineering approach we created two portions of design process to be integrated with PASSI in order to implement the self-conscious loop in the robot.

Acknowledgements This work has been partially supported by the EU project FP7-Humanobs.

References

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

Chella, A., and Macaluso, I. 2008. Higher order robot perception loop. In *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. 2007. *Artificial Consciousness*. Imprinting Academic, Exter, UK.

Chella, A.; Cossentino, M.; Sabatucci, L.; and Seidita, V. 2006. Agile PASSI: An Agile Process for Designing Agents. *International Journal of Computer Systems Science & Engineering. Special issue on Software Engineering for Multi-Agent Systems* 133–144.

Cossentino, M., and Seidita, V. 2004. Composition of a New Process to Meet Agile Needs Using Method Engineering. *Software Engineering for Large Multi-Agent Systems* 3:36–51.

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.; Gaglio, S.; and Seidita, V. Adapting passi to support a goal oriented approach: a situational method engineering experiment. In *Proc. of the Fifth European workshop on Multi-Agent Systems (EUMAS'07)*, 729–743.

Cossentino, M. 2005. From requirements to code with the PASSI methodology. In *Agent Oriented Methodologies*. Hershey, PA, USA: Idea Group Publishing. chapter IV, 79–106.

Jacobson, I.; Booch, G.; and Rumbaugh, J. 1999. *The unified software development process*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

Jacobson, B. Rumbaugh, The Unified Software Development Process. *Addison-Wesleym ISBN 0-20-157169-2*.

Kumar, K., and Welke, R. 1992. Methodology engineering: a proposal for situation-specific methodology construction. *Challenges and Strategies for Research in Systems Development* 257–269.

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. Special Issue on Foundations, Advanced Topics and Industrial Perspectives of Multi-Agent Systems.

Osterweil, L. 1987. Software processes are software too. In *Proceedings of the 9th international conference on Software Engineering*, 2–13.

SEI. CMMI for Development Version 1.2. *Technical Report of the Software Engineering Institute of the Carnagie Mellon University. August 2006*.

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).*

SPEM. Software Process Engineering Metamodel. Version 2.0.final adopted specification ptc/07-03-03. (omg) Object Management Group.March 2007.