Self-Conscious Robotic System Design Process - from Analysis to Implementation

Antonio Chella and Massimo Cossentino and Valeria Seidita

Abstract Developing robotic systems endowed with self-conscious capabilities means realizing complex sub-systems needing ad-hoc software engineering techniques for their modelling, analysis and implementation. In this paper the whole process (from analysis to implementation) for modelling the development of self-conscious robotic systems is presented and the new created design process - PASSIC supporting each part of it - is fully illustrated.

1 Introduction

One of the most important topics in the today robotic research is to provide a robotic system with self-conscious abilities.

Our work starts from the hypothesis, also endorsed by several studies in the field of neuroscience, psychology and philosophy, that basic conscious behaviour perception can be modelled and implemented by means of a continuous loop between the activity in the brain and the events perceived in the outer world (see [6]). The perception loop 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 contemporaneously in action, being each of them related to different sensor modalities or considering different parameters and aspects of the same sensor modality.

Antonio Chella and Valeria Seidita
Dipartimento di Ingegneria Informatica, Universitá degli Studi di Palermo, Viale delle Scienze, 90128 Palermo, Italy
e-mail: {chella,seidita}@dinfo.unipa.it

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

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 acting in the scene. We argue that higher order perception loops are responsible of the robot self-consciousness.

Implementing generalized higher orders perception loops in a robotic system is a hard issue. We are investigating how to cope with modelling and engineering these robotic systems.

Nowadays literature proposes several different software engineering techniques for developing complex robotic systems, and in the past the agent paradigm [1, 18, 10] has proved to be successful for developing robotic applications by considering the robotic system as a collection of agents each of them responsible for a specific functionality.

In this context the PASSI (the Process for Agent Societies Specification and Implementation) [13] design process provides means for developing multi-agent system used within different kinds application domain, for instance software for embedded robotics and agent-based information systems.

In the presented work our aim is to model the development of self-conscious robotic system in its entirety and to adopt proper software engineering techniques for conceiving its parts in order to obtain a multi agent system where each agent (or a set of agents) is committed to manage the different order of perception loops; agents' peculiarities and characteristics such as autonomy, proactivity and situadness let a multi agent system be suitable to implement such systems.

In the past we developed and experimented an approach for the creation of ad-hoc agent design processes following the (Situational) Method Engineering paradigm [32, 16]; this approach is principally based on the use of the metamodel, describing the set of elements to be instantiated during system development. The results of the experiments realized in the past include agent design processes like Agile PASSI [7] and PASSIG [17, 31], both are based and developed starting from PASSI by extracting the essential characteristics useful for being applied to specific application contexts; the former is an agent oriented design process developed taking into account what Agile Manifesto [25] prescribes and was thought for being used as the agile development tool for robotic systems, the latter is the evolution of PASSI ad-hoc created and used for performing a goal oriented analysis of the problem domain. This latter, together with PASSI2, the main evolution of PASSI, presents features we found useful for being integrated in a new agent design process for developing self-conscious robotic systems.

Therefore the work presented in this paper is based on the extension of the PASSI (Process for Agent Society Specification and Implementation) process and metamodel in order to include the activities and elements needed for the construction of self-conscious robotic system. In [9] and [8] the metaphor of test has been used for developing and implementing the reflective part of a robotic system; that work resulted in two different design activities that in this paper are integrated in the new process (PASSIC) built on the basis of two previous evolutions of PASSI: PASSI2 and PASSIG.

In the rest of the paper an overview on the previous work is given and the PASSIC design process is illustrated together with the whole self-conscious robotic system development process it is part of.

The paper is organized as follow: section 2 gives some hints about the theoretical background of the presented work, in section 3 the whole self-conscious system development process is shown and in section 4 its central point, namely the PASSIC design process, is detailed. Finally in section 5 some conclusions are drawn.

2 Theoretical Background

The aim of our work is to create multi agent software systems able to control a robot by means of different perception loops; in this section an overview about the key elements of our research is given: the perception loop, PASSI, the agent design process we extended for our needs and the techniques used for creating ad-hoc design processes.

2.1 The Robot Perception Loop

The robot perception loop described in [5, 12] (see Fig. 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 robot's prediction about the relevant events around it.

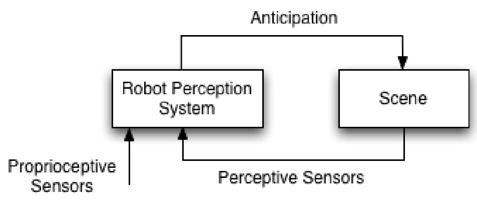

Fig. 1 The Perception Loop

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. According to [29, 26, 23] the perception loop realizes a loop among "brain, body ad environment".

The generalized perception loop was tested and implemented on *Cicerobot*, an indoor robot offering guided tours in the Archaeological Museum of Agrigento [12], and on *Robotanic*, an outdoor robot offering guided tours in the Botanical Garden of the University of Palermo [2].

By implementing the perception loop the robot is endowed with the ability to sense (to perceive) the word around it; besides in [11, 6] it is argued that in a real operating robot there can be different perception loops contemporaneously in action, thus realizing robot self-consciousness, the robot's inner world perception. 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, behaviors.

2.2 The PASSI Design Process

PASSI (Process for Agent Society Specification and Implementation)[13] is the design process developed several years ago in our laboratory; it is devoted to modelling and implementing different kind of multi-agent software systems mainly exploiting the possibility it offers of decomposing the system requirements into functionalities that can be assigned to a set of agents each of which can interact with another one by exchanging knowledge about the environment they live in.

PASSI has been principally used for developing robotic systems and besides during the last years we experimented the possibility of creating a process framework, whose core is PASSI, for developing a wide number of agent software kinds (see the following site for a more detailed overview:
http://www.pa.icar.cnr.it/passi/PassiExtension/extensionsIndex.html).

In the following PASSI main phases and its lifecycle are illustrated.

2.2.1 The PASSI Lifecycle

The PASSI process covers all the phases from requirements analysis to deployment configuration, coding, and testing. PASSI has been designed for developing systems to be applied in the areas of robotics, workflow management, and information systems.

Designers involved in the design process are supposed to have experiences of object-oriented design, processes like UP [24] and of concepts like a functionality-oriented requirement analysis. PASSI principally uses models from object oriented software engineering and UML notation for most artefacts resulting from the activities it is composed of.

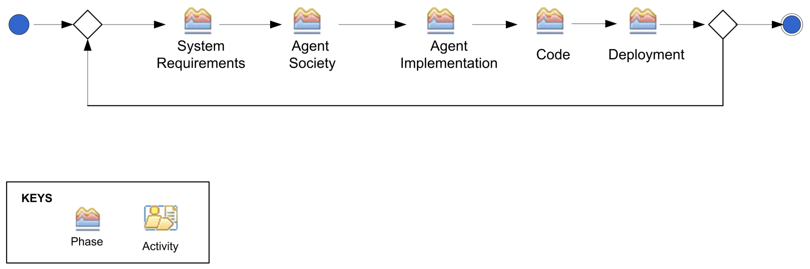

Fig. 2 Phases of the PASSI Design Process

Fig. 2 [1] shows an high level phases-decomposition of PASSI, each phase is decomposed in activities (and then in tasks) resulting in the production of one artefact[2].

1. The *System Requirements* phase is devoted to produce a model of the system requirements that can be committed to agents, the activities involved in this phase are: *Domain Description*, *Agent Identification*, *Role Identification*, *Task Specification*.
2. The *Agent Society* phase's aim is to model the agents society knowledge and the communications the agents take part in; it also produces models describing the structure of role played by agents and the protocol used for communicating. The activities involved are: *Domain Ontology Description*, *Communication Ontology Description*, *Role Description* and *Protocol Description*
3. The *Agent Implementation* phase deals with the solution architecture both in terms of single agent view and multi agent one, the activities it is composed of are: *Multi-Agent Structure Definition*, *Multi-Agent Behavior Description*,*Single-Agent Structure Definition*, *Single-Agent Behavior Description*.
4. The *Code* phase provides a model of the solution at the code level. It is largely supported by patterns reuse and automatic code generation. The activities are: *Code Reuse* and *Code Completion*.
5. The *Deployment* phase describes the distribution model system's parts across hardware processing unit and the allocation of agents.

Several extensions to PASSI have been developed for specific application contexts. The work presented in this paper starts from two of them: PASSI2 and PASSIG; the former was the natural evolution of PASSI after a few years of experience with that; whereas the latter is the result of a PASSI modification in order to support goal oriented analysis [3].

One of the most important features exploited from PASSI2 is the possibility of early identifying, during the analysis phase, the structural description of the identi-

[1] The notation used in this diagram is the one proposed by SPEM 2.0 (Software and Systems Process Engineering Metamodel) specification [27].

[2] For a detailed description of the PASSI design process refer to [13] and http://www.pa.icar.cnr.it/passi/

[3] For more details see [17] and http://www.pa.icar.cnr.it/passi/PassiExtension/exstensionsIndex.html

fied agents. PASSIG was used for it provides means for performing a goal oriented analysis of the features the system have to accomplish and we found it principally useful for the identification and description of the goals the robot has to execute.

2.3 Agent Oriented Situational Method Engineering

The development of a multi agent system always requires great efforts in learning and using an existing design process. Today it is recognized that it does not exist only one standard design process (or also a methodology or a method) for developing every kind of systems able to solve every kind of problems and therefore there is the need for creating techniques and tools for a designer to develop an ad-hoc design process prior to use it on the base of his own needs.

In order to solve this problem and to give means for one to develop an agent system using the "right" design process we adopted and extended the Situational Method Engineering (SME)approach [4, 20, 33, 28] by creating techniques and tools [15, 14, 32] that gave us the possibility of creating design processes for developing and implementing whatever class of systems.

SME is mainly based on the concept of "reuse"; each time the method engineer, the person devoted to create and develop methodologies, wants to create his own methodology he has to reuse portions of existing design processes already used and tested, in a certain sense in the same way a software designer does when he is developing software. SME root element, often called method fragment, chunk, process fragment or simply fragment is generally extracted from an existing design process; this requires using the right techniques, also for its description. Once extracted, the fragment is stored in a repository from which it can be selected when necessary and assembled with other one to form a complete, new, design process.

3 The Proposed Development Process for Self-Conscious Systems

The perception loop forms the base of the development and implementation process for self-conscious behaviour in a robotic system because it provides the starting point for the system to be able to activate all the proper behaviours sprung from the mismatch between the expected situation and the real perceived one while pursuing a goal.

In our approach, the robot can (dynamically) tune some of the mission execution parameters, decide to adopt another behaviour or to save the successful one in a repository of cases for a future reuse. Besides we consider robotic systems that, in the same way biological systems do, are endowed with a set of innate capabilities; these capabilities found their practical realization in the set of activities (i.e. tasks) the robot is able to perform with a precise set of parameters configuration; each set of parameters allows the robot to reach one specific goal.

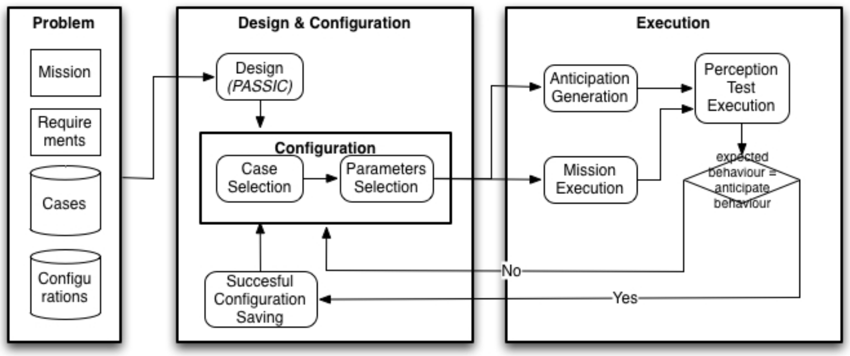

Fig. 3 The Proposed Self-Conscious System Development Process

Fig. 3 shows the complete development process used for developing self-conscious systems; the figure depicts the three different areas the designer has to deal with while implementing such systems, they are: *(i)* Problem, *(ii)* Design and Configuration, *(iii)* Execution.

3.1 The Three Development Areas

The **Problem** area is composed of all the activities devoted to elicit system requirements and to identify the mission the robot has to perform in order to reach its goals. During these activities the designer considers a database where the set of abilities the system possesses are stored (the *Cases*); the proposed development process is applied to systems owning pre-determined abilities. More in details the process considers two different archives: *Cases* and *Configurations*. A *Case* is composed of the goal description, the set of actions performed in order to reach it (a plan), pre-conditions, and the list of parameters needed for successfully applying the plan (only their names, not useful values). A *Configuration* is a specific set of parameter values that has proved to be successful for instantiating one specific case; it also includes the number of positive outcomes this configuration produced in pursuing the case goal.

The **Design and Configuration** area deals with the definition of the robotic system that will accomplish the required mission while successfully fulfilling the requirement constrains. After the design has been completed, the system has to be configured in order to obtain an optimal performance.

The first activity in this area is the Design activity. This corresponds to the usual application of a system design process. During this activity, the designer defines a software solution that could accomplish the required mission. This activity corresponds to the application of the PASSIC design process (see section 4).

The process starts with the inputs collected during the previous phase and according to them 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.

Once the system is designed, one case has to be selected from the Cases database. This will be used for producing both the anticipated behaviour and in the meanwhile to start the mission execution. Cases selection is done on the basis of the goal(s) to be pursued. In such a selection, it is to be considered that sometimes the pursued goal cannot be satisfied by any of the cases in the database. This situation is solved by creating a new case (usually by reusing and composing existing cases).

In the current implementation of the system, new cases are created by randomly selecting existing ones, we plan to adopt a more rigorous and smart approach in the future. Usually, cases are described in terms of some parameters that deeply affect the expected outcome.

Such set of parameter values define a configuration. In other words, a configuration is a set of records reporting instantiation data for cases in the database, with the corresponding scores, that reports successful applications of the case (with that configuration) versus total applications of it (with that configuration). If the results obtained by the application of the selected configuration are not correct (this check is performed after the Perception Test Execution), a new configuration can be tried (either by selecting a new set of values for parameters or by selecting a new case). If the results of the perception test are satisfying, the new configuration is saved in the Configurations Database as a successful one.

The perception test is performed within the activities in the Execution area during which the running system produces the *Anticipation Generation* and executes the mission. After a case has been selected, a part of the system generates the anticipations about the mission to be performed using the case itself. For instance, if the goal is "reaching object O", the plan might be "go from point A to point B" and the corresponding expectation is "the robot position at the end of the plan execution is (x,y)".

Once the anticipation is produced, the robot starts the execution of its mission. Referring to the above example, it moves and continuously compares its real behaviour with all the parameters involved (for instance wheels position, proprioceptive sensors and so on) in the anticipated case by means of the *Perception Test Execution*. If it finds some differences it activates the tuning phase by changing the initial configuration, instead if the expected behaviour perfectly matches with the anticipated one then the used configuration has been successful and it can be saved in the database for future reuse.

4 The PASSIC Design Process

PASSIC is the design process that has been created by extending PASSI for developing and implementing self-conscious behaviour onto a robotic systems; it provides design activities for the design of each portion of the system presented in the previous section.

4.1 The Definition of PASSIC Design Process

In [8, 9] an experiment concerning the creation of a design methodology and a model for perception loop has been presented. The process for creating the new methodology follows the *Situational Method Engineering* paradigm [4, 28, 21], extends and modifies the PASSI [13], PASSI2 [17] and the PASSIG [31] processes developed by the authors in the latest years by exploiting process fragments also coming from Tropos [3, 19].

As already said (section 2.3) Situational Method Engineering is the discipline developed in the field of information systems with the aim of creating, exploiting and evaluating techniques, methods and tools for the creation of design processes to be used in specific application context. The SME paradigm has been extended to the agent field and a well defined approach for the creation of agent design process has been developed [32]; this one is called PRoDe (Process for the Design of Design Processes) and its main elements is the so called *process fragment* [14]. The whole process is composed of three main phases, the process requirements, the fragments selection and the fragments assembly, the first concerns with the requirements analysis of the design process under construction, the second with the selection of the right process fragments to be selected from the repository and to be assembled in the following phase.

PRoDe mainly exploits the use of the multi agent system (MAS) metamodel for performing the tasks within the selection and the assembly phases. The MAS metamodel contains all the elements to be designed for developing a specific system following one specific design process. For instance the PASSI MAS metamodel contains elements such as agent, role, task etc., an agent plays some roles in order to reach an objective and has some capabilities under the form of tasks it is able to perform; each of this elements has to be designed in, at least, one activity of the design process. In PRoDe the MAS metamodel is the results of the process requirements phase and is used as the base for the selection and assembly of fragments.

In [9] and [30] an extended analysis and description of the set of requirements leading to the creation of PASSIC is reported; the analysis resulted in the definition of the *metamodel* for the perception loop, see [8] for further details, where elements of perception loop have been identified and reflected onto a robotic system.

Briefly some central elements of the metamodel are: the *robot* having the responsibility of pursuing one or more *goals* composed of *plans* and *actions*, i.e. physical or communicative acts between the robot and external objects that result in the change of the surrounding environment. The robot also has capability by means of *test*, *simulated act* and *log* that implement the robot's inner and outer reflections (i.e., the perception loop).

In order to cope with the aforementioned elements of the conscious metamodel two process fragments coming from the Unified Process (UP) [24] (*Test Plan and Design* and *Test Execution*) have been reused, modified and integrated; the former's aim is to identify the system functionalities to be tested, the available system resources and the test objective in order to design the *Anticipation Generation*. The latter aims at defining the *Execution Test* in order to identify defects and analyze the results also by means of defining criteria for evaluating perception test results.

4.2 The PASSIC Process Lifecycle

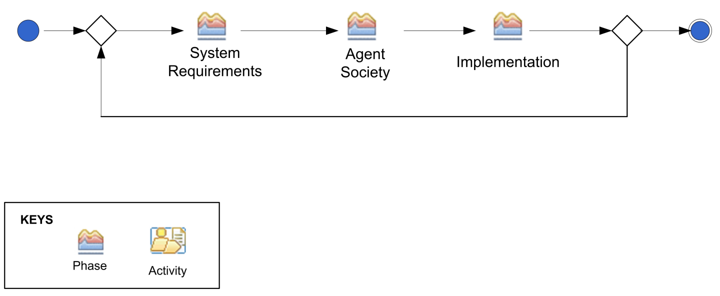

Fig. 4 The PASSIC Design Process - Phases

PASSIC includes three phases arranged in an iterative/incremental process model (see Figure 4):

- System Requirements: it covers all the phases related to a goal oriented requirements analysis and agents/roles identification.
- Agent Society: where all the aspects of the agent society are faced.
- Implementation: A view on the system's architecture in terms of classes and methods to describe the structure and the behavior of single agent, reusable code and source code for the target system, how the agents are deployed and which constraints are defined/identified for their migration and mobility.

Each phase produces a document that is usually composed aggregating UML models and work products produced during the related activities; moreover each phase is composed of one or more sub-phases responsible for designing or refining one or more artefacts that are part of the corresponding model. The details of each phase will be discussed in the following subsections.

4.2.1 The System Requirements Phase

The System Requirements phase aims at analyzing the problem domain through a goal oriented analysis in order to produce the model of the system in terms of agency, the set of actors involved in the system under construction and the related goals.

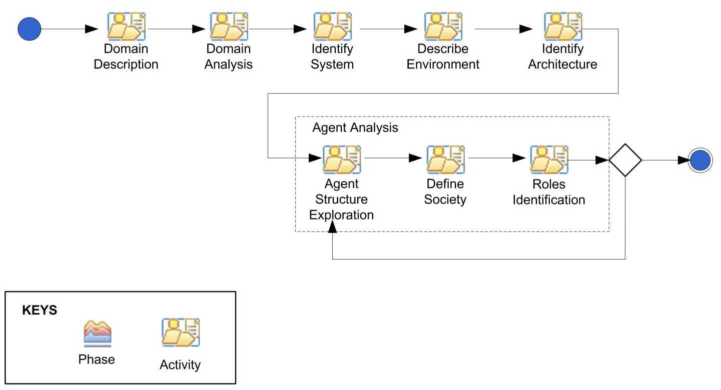

Fig. 5 The Activities of the System Requirements Phase

Developing this phase involves eight activities:

1. Domain Description provides means for analyzing the problem statement, that is the description of the problem to be faced, in order to identify the actors involved in the system and their goals; actor is an intentional entity that can be external or internal and that has a strategic interest, i.e. the goal.
2. Domain Analysis aims at identifying the tasks, each actor has to perform in order to pursue a goal, and applying means-end-analysis in order to relate each task to (at least) one goal; task is a specific set of actions performed in order to pursue a goal, or a sub-goal.
3. Identify System where the System-to-be actor is identified; the System-to-be actor represents the system under construction together with the dependencies with all the other actors of the environment.
4. Agent Structure Exploration where an analysis-level description of the agent structure in terms of tasks required for accomplishing the agent's functionalities is performed.
5. Describe Environment produces the system's actors and goals that can be assigned to the System-to-be actor thus identifying the dependencies between the System actor and all other actors.
6. Identify Architecture for decomposing the System-to-be into sub-actors, to which goals are assigned, and for identifying agents; generally each sub-actor can be mapped onto an agent.

7. Define Agent Society aims at identifying a set of capabilities for each agent in order to establish which plans they have to follow.
8. Roles Identification provides means for identifying the roles each agent plays and the dependencies among agents. The role represents the social behaviour of an agent.

4.2.2 The Agent Society Phase

The Agent Society phase introduces an agent-oriented solution for the problem described in the previous phase. This phase presents an ontological description of both the domain where agents will live and their communications, then agents are described in terms of the roles they play, services provided by roles, resource dependencies and finally their structure and behaviors. Once an agent solution has been identified the autonomous part of the system devoted to create the expectation about the results of plans application and the related configuration management is designed.

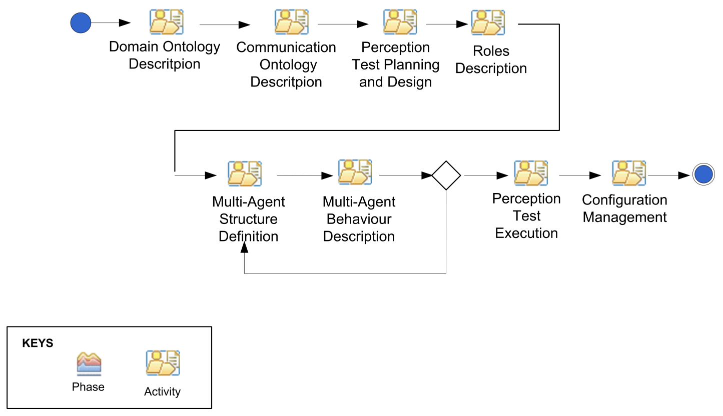
Domain Ontology Descritpion
Communication Ontology Descritpion
Perception Test Planning and Design
Roles Description
Multi-Agent Structure Definition
Multi-Agent Behaviour Description
Perception Test Execution
Configuration Management
KEYS
Phase
Activity

Fig. 6 The Activities of the Agent Society Phase

Developing this phase involves eight activities:

1. Domain Ontology Description aims at identifying and describing the ontological elements the system will deal with in order to define the pieces of knowledge of each agent and their communication ontology. The domain categories are: concepts, actions that could affect their state and propositions about values of categories.
2. Communication Ontology Description for describing agents' communications in terms of previously determined ontology, interaction protocol and message content language.

3. Perception Test Planning and Design where the anticipation is produced, starting from the agent society architecture, the knowledge about the environment and requirements. The set of tasks each agent has to pursue is modeled from a structural point of view. The purpose is to describe the robot's actions while interacting with the environment.
4. Role Description aims at modeling the whole lifecycle of each agent formalized by the distinct roles played, the tasks involved in the roles, communication capabilities and inter-agent dependencies in terms of services.
5. Multi-Agent Structure Definition (MASD) describes the structure of solution agent classes at the social level of abstraction.
6. Multi-Agent Behavior Description describes the behavior of individual agents at the social level of abstraction.
7. Perception Test Execution aims at designing the portion of system devoted at producing the results of the comparison between the observed and the expected robot's/system's behavior and the criteria for evaluating them.
8. Configuration Management designs the rules for tuning the system parameters. This activity is obviously strictly related to the specific robotic platform to be used for deploying the designed multi-agent system.

4.2.3 The Implementation Phase

Implementation Phase results in the model of the solution architecture in terms of classes, methods, deployment configuration, code and testing directives. 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.

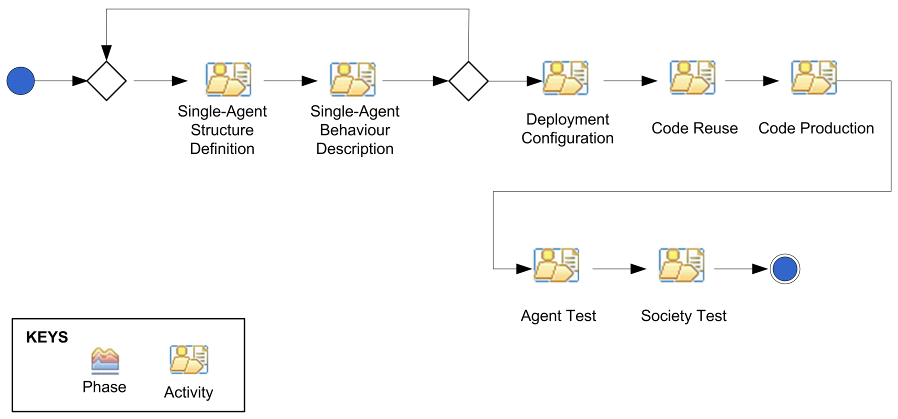

Fig. 7 The Activities of the Implementation Phase

The Implementation Phase is composed of seven activities:

1. Single-Agent Structure Definition describes the structure of solution agent classes at the implementation level of abstraction.
2. Single-Agent Behavior Description describes the behavior of individual agents at the implementation level of abstraction.
3. Deployment Configuration describes the allocation of agents to the available processing units and any constraints on migration, mobility and configuration of hosts and agent-running platforms.
4. Code Reuse uses a library of patterns with associated reusable code in order to allow the automatic generation of significant portions of code.
5. Code Completion where source code of the target system is manually completed.
6. Agent Test is devoted to verifying the single behavior with regards to the original requirements of the system solved by the specific agent.
7. Society Test where the validation of the correct interaction of the agents is performed in order to verify that they actually concur in solving problems that need cooperation.

More details about PASSIC phases and activities and how it has been created starting from its metamodel by using PRoDe approach can be found in [8, 9, 30]; [30] also provides the description of the experiment made in order to test the usability of PASSIC

5 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. The work presented here focuses on the creation of a complete process for the development of a self-conscious robotic system and starts from the hypothesis that self-consciousness in a robot may be reached by means of different orders of perception loops. Each loop can be managed by an agent, or a society of agents.

The experiences made in the latest years in the creation of ad-hoc design processes allowed the identification and the analysis of the requirements for the creation of a design process, realizing such a system, by following a perception driven approach; the continuous loop between perceived events and activities in the brain is the core of the self-conscious behaviour we want to emulate in a robotic system.

The result was the extension of PASSI design process, by integrating it with new techniques for designing the robot perception loop, thus creating PASSIC whose activities are fully described in this paper.

PASSIC contains all the activities for the complete development of self-conscious robotic system and allows to design and implement the perception loop thus making a robotic system able to move in a dynamic environment, by continuously detecting the differences between the expected and the real behaviour, and tuning its parameters also learning successfully experienced behaviours for later reuse in novel situations.

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

References

1. 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.
2. R. Barone, I. Macaluso, L. Riano, and A. Chella. A brain inspired architecture for an outdoor robot guide. In A. Samsonovich, editor, *Proc. of AAAI Fall Symposium on Biologically Inspired Cognitive Architectures BICA '08*, Menlo Park, CA., 2008. AAAI Press.
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. A robot architecture based on higher order perception loop. In A. Hussain, editor, *Brain Inspired Cognitive Systems 2008*, page (in press). Springer Science+Business Media, 2009.
7. A. Chella, M. Cossentino, L. Sabatucci, and V. Seidita. Agile PASSI: An Agile Process for Designing Agents. *International Journal of Computer Systems Science & Engineering. Special issue on Software Engineering for Multi-Agent Systems*, pages 133–144, 2006.
8. 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.
9. 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.
10. A. Chella, A. Frixione, and S. Gaglio. An architecture for autonomous agents exploiting conceptual representations. *Robotics and Autonomous Systems*, 25:231–240, 1998.
11. A. Chella and I. Macaluso. Higher order robot perception loop. In Berlin Eiderlberger Springer-Verlag, editor, *BICS 2008 Brain Inspired Cognitive Systems*, June 24-27 2008.
12. A. Chella and I. Macaluso. The perception loop in Cicerobot, a museum guide robot. *Neurocomputing*, 72:760 – 766, 2009.
13. M. Cossentino. From requirements to code with the PASSI methodology. In *Agent Oriented Methodologies* [22], chapter IV, pages 79–106.
14. M. Cossentino, S. Gaglio, A. Garro, and V. Seidita. Method fragments for agent design methodologies: from standardisation to research. *International Journal of Agent-Oriented Software Engineering (IJAOSE)*, 1(1):91–121, 2007.
15. M. Cossentino, S. Galland, S. Gaglio, N. Gaud, H. Hilaire, A. Koukam, and V. Seidita. A mas metamodel-driven approach to process composition. In *Proc. of the Ninth International Workshop on Agen-Oriented Software Engineering (AOSE-2008) at The Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008),*, 2008.
16. M. Cossentino and V. Seidita. Composition of a New Process to Meet Agile Needs Using Method Engineering. *Software Engineering for Large Multi-Agent Systems*, 3:36–51, 2004.
17. M. Cossentino and V. Seidita. PASSI2 - going towards maturity of the PASSI process. *Technical Report ICAR-CNR*, (09-02), 2009.
18. 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.

19. Paolo Giorgini, Manuel Kolp, John Mylopoulos, and Jaelson Castro. Tropos: A requirements-driven methodology for agent-oriented software. [22], chapter II, pages 20–45.
20. AF Harmsen, S. Brinkkemper, and H. Oei. 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*, pages 169–194, 1994.
21. A.F. Harmsen, M. Ernst, and U. Twente. *Situational Method Engineering*. Moret Ernst & Young Management Consultants, 1997.
22. Brian Henderson-Sellers and Paolo Giorgini. *Agent Oriented Methodologies*. Idea Group Publishing, Hershey, PA, USA, June 2005.
23. S. J. Hurley. Varieties of externalism. In ed. In R.Menary, editor, *The Extended Mind*. Ashgate, in press.
24. B. Jacobson. Rumbaugh, The Unified Software Development Process. *Addison-Wesleym ISBN 0-20-157169-2*.
25. Agile Manifesto. *http://http//agilemanifesto.org*.
26. R. Manzotti and V. Tagliasco. An externalist process-oriented framework for artificial consciousness. *AI and Consciousness: Theoretical Foundations and Current Approaches, eds. A. Chella and R. Manzotti (AAAI Press, CA, 2007)*.
27. OMG Object Management Group. Software Process Engineering Metamodel. Version 2.0. Final Adopted Specification ptc/07-03-03. March 2007.
28. 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.
29. W.T. Rockwell. *Neither brain nor ghost*. MIT Press, 2005.
30. V. Seidita and M. Cossentino. From modeling to implementing the perception loop in self-conscious systems. *International Journal of Machine Consciousness (IJMC)*, 2(2):289–306, 2010.
31. V. Seidita, M. Cossentino, and S. Gaglio. Adapting passi to support a goal oriented approach: a situational method engineering experiment. 2007.
32. 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).*, June 2010.
33. ter Hofstede A.H.M. and Verhoef T.F. On the feasibility of situational method engineering. *Information Systems.*, 22(6/7):401–422, 1997.