# Reactivity and Social Cooperation in a Multi-Robot System

Atef Gharbi[1], Nadhir Ben Halima[2] and Hamza Gharsellaoui[1]
[1]*Department of Computer sciences, INSAT, Tunis, Tunisia*
[2]*College of Computer Science and Engineering, Taibah University, Yanbu Branch, Yanbu Al-Bahr, K.S.A.*

Keywords: Multi-Robot Systems, Distributed Planning, Five Capabilities Model, Benchmark Production System.

Abstract: Multi-Robot System (MRS) is an important research area within Robotics and Artificial Intelligence. The balancing between reactivity and social cooperation in autonomous robots is really considered as a challenge to get an effective solution. To do so, we propose to use the concept of five capabilities model which is based on Environment, Self, Planner, Competence and Communication. We illustrate our line of thought with a Benchmark Production System used as a running example to explain our contribution.

## 1 INTRODUCTION

In this paper, the term Multi-Robot System (MRS) indicates a team of two or more autonomous robots performing in a dynamic environment where uncertainty and unforeseen changes can happen due to robots. Besides, the control is not centralized, but rather distributed over the members of the team. Robots, to achieve their goals successfully in dynamic and unpredictable environments, must be able to generate plans in a timely way, monitor changes in their environment, change and adapt their plans accordingly. In this paper, we regard Multi-Robot System as a particular form of Multi Agent System (MAS), by specifically addressing planning and social abilities. This mapping seems at the first look suprising as a robot is considered a physical entity ensuring a variety of tasks. However, there are many similarities between Agent and Robot which pushes in this way (for example, the robot like the agent has several activities and responds to its environment). Besides, we believe that the multi-agent technology applied to robotic system is able to increase the flexibility of the system as a whole.

This article is concerned with two important matters: how to define the Multi-Robot System in a manner such that it has more utility to deploy it, and how to use such a MRS for the advanced software. The Multi-Robot System must discover the action to be taken by supervising the application and its environment and analyzing the data obtained.

With Multi-Robot System, we face two important matters: (i) the detection of a need for action: the need for action must be discovered by supervising the application and its environment and analyzing data obtained. (ii) the planning of the action: it consists to envisage the action (by proposing which modifications need to be made) and by programming it.

Using a multi-agent approach, the robot's architecture can be decomposed into flexible autonomous subsystems (agents). The architecture can then be described at a higher level, defining the agents that have to be in the system, the role of each of them, the interactions among them, the actions each of them performs, and the resources they need. Since the multi-agent system is inherently multi-threaded, each agent has its own thread of control; each agent decides whether or not to perform an action at the request of another agent (autonomy); agents establish agreements among themselves, while keeping their autonomy sharing their knowledge and acting together to accomplish specific common goals. Agents need to interact to coordinate their activities so that control of the robot is achieved. All of those processes, the agent's own decision making, interaction and coordination need to be highlighted. To do so, we propose the design of a Robotic Agent according to the 5 Capabilities model (5C) proposed by (van Aart, 2004), (C. J. van Aart and Schreiber, 2004). The 5 Capabilities model is separated into five dimensions: Environment, Self, Planning, Competence and Communication. These dimensions are said models where each model represents one specific capability of the robotic agent. First of all, a robotic agent needs to interact with the environment in which it operates thanks to sensors (providing data) and actuators (executing actions) therefore we define the *Environment Model*. To know what tasks to be executed, we define the *Self Model*. The self model is used to know the robotic agent's perception of its own being and state. In other terms, it consists of ongoing tasks. The planning of ongoing tasks is the concern of the *Planner Model*. A planner model is ensuring some kind of reasoning about task selection, execution control, time monitoring and emergency handling.

The *Competence Model* ensures the methods the abilities and the knowledge that enables the robotic agent to execute the task that is designed for it. The multi-robot system has the appropriate *Communication Model* in order to avoid non-feasible, unsecured and fortuitous actions that can provoke undesirable results by a single robot for the whole system, the different robotic agents have to interact together following a specific communication protocol.

This paper introduces a simple Benchmark Production System that will be used throughout this article to illustrate our contribution which is developped as Robot-based application. We implement the Benchmark Production System in a free platform which is JADE (JavaTM Agent DEvelopment) Framework. JADE is a platform to develop multi-agent systems in compliance with the FIPA specifications (Salvatore Vitabile, 2009), (Chuan-Jun Su, 2011), (Bordini and all., 2006).

In the next section, we present the Benchmark Production System. The third section introduces the Environment Model by specifying the sensors, the actuators and the safety requirements. The fourth section presents the Self Model which describes the different tasks to be executed by the robotic agent with a formal specification. We introduce in the fifth section the Planner Model. The sixth section presents the Competence Model based on Fuzzy Logic System. Finally, we study the Communication Model in particular the message exchanged through a communication protocol. We conclude in the last section.

## 2 BENCHMARK PRODUCTION SYSTEM

As much as possible, we will illustrate our contribution with a simple current example called *RARM* (Branislav Hrz, 2007). We begin with the description of it informally, but it will serve as an example for various formalism presented in this article. The benchmark production system *RARM* represented in the figure 1 is composed of two input and one output conveyors, a servicing robot and a processing-assembling center. Workpieces to be treated come irregularly one by one. The workpieces of type $A$ are delivered via conveyor $C1$ and workpieces of the type $B$ via the conveyor $C2$. Only one workpiece can be on the input conveyor. A robot $R$ transfers workpieces one after another to the processing center. The next workpiece can be put on the input conveyor when it has been emptied by the robot. The technology of production requires that first one $A$-workpiece is inserted into the center $M$ and treated, then a $B$-workpiece is added in the center, and last the two workpieces are assembled. Afterwards, the assembled product is taken by the robot and put above the $C3$ conveyer of output. the assembled product can be transferred on $C3$ only when the output conveyor is empty and ready to receive the next one produced.

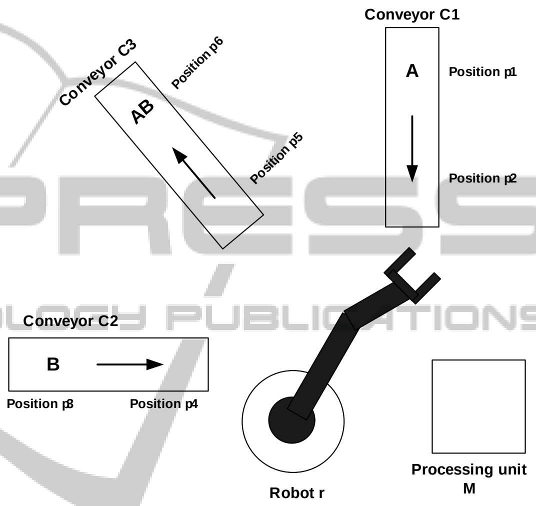

Figure 1: The benchmark production system RARM.

## 3 ENVIRONMENT MODEL

Perception is responsible for collecting runtime information from the virtual environment. The perception component supports selective perception, enabling a robotic agent to direct its perception to its current tasks. The perception component interprets the representation resulting in a percept. A percept consists of data elements that can be used to update the robotic agent's current knowledge.

**Actuators**
The system can be controlled using the following actuators: (i) move the conveyor $C1$ (act1); (ii) move the conveyor $C2$ (act2); (iii) move the conveyor $C3$ (act3); (iv) rotate robotic agent (act4); (v) move elevating the robotic agent arm vertically (act5); (vi) pick up and drop a piece with the robotic agent arm (act6); (vii) treat the workpiece (act7); (viii) assembly two pieces (act8).

**Sensors**
The control program receives information from the sensors as follows: (i) Is there a workpiece of type A

at the extreme end of the position $p1$? (sens1) (ii) Is the conveyor $C1$ in its extreme left position? (sens2) (iii) Is the conveyor $C1$ in its extreme right position? (sens3) (iv) Is there a workpiece of type A at the processing unit $M$? (sens4) (vi) Is the conveyor $C2$ in its extreme left position? (sens5) (vii) Is the conveyor $C2$ in its extreme right position? (sens6) (viii) Is there a workpiece of type B at the extreme end of the position $p3$? (sens7) (ix) Is there a workpiece of type B at the processing unit $M$? (sens8) (x) Is the conveyor $C3$ in its extreme left position? (sens9) (xi) Is the conveyor $C3$ in its extreme right position? (sens10) (xii) Is there a workpiece of type AB at the processing unit $M$? (sens11) (xiii) Is the robotic agent arm in its lower position? (sens12) (xiv) Is the robotic agent arm in its upper position? (sens13)

## 4 SELF MODEL

We represent the whole system (i.e. RARM) with the petri net. A token in the place $p_{A0}$ (resp. $p_{B0}$) represents a situation when the conveyor $C1$ (resp. $C2$) is empty. The event "a workpiece occurred at the system input" changes the input state. Afterwards the workpiece is prepared for manufacturing. The event is represented by firing $t_A$ . Transition $t_A$ (resp. $t_B$) is fireable and on its firing the token from $p_{A0}$ (resp. $p_{B0}$) is removed and is placed into the place $p_{A1}$ (resp. $p_{B1}$). In the Petri net it is not specified when the event happens. If the robotic agent is available (a token is in $p_R$), a workpiece A is available (a token in $p_{A1}$), and the previous machine process is completed (a token in $p_{MF}$) then the robotic transfer of the workpiece A can start (firing transition $t_1$). Place $p_{TA}$ is occupied by a token during the transport. The end of the transport and loading the workpiece into machine M is specified by transition $t_2$. After firing $t_2$ a token is placed in $p_{MA}$ . End of processing of the workpiece A is specified by transition $t_3$ and the start of the transport of B into the machine by $t_4$ . Then a token comes to the place $p_{TB}$, which indicates that the transfer of the workpiece B is in progress; $t_5$ denotes end of the transfer and start of the assembly operation; $t_6$ specifies end of the assembly operation; $t_7$ specifies start of the product transfer from M onto the conveyor C3 ($t_7$ is fired and a token appears in $p_{TO}$). After the transfer of the product AB on C3 and its leaving out the cell, the token moves to the $p_{OF}$ place. The place $p_{MF}$ ensures that the next part A is loaded into machine M only after the assembly process has been finished. The next workpiece A can be loaded at the earliest on the conveyor C1 (firing $t_A$) when the preceding workpiece A is in the transfer to the machine M ($p_{TA}$ marked). Considering it we have: $t_1$ fires and then $t_A$ fires. First, A has to arrive to be processed in M (a token in $p_{MA}$) and after that it waits in the machine (token in $p_{AW}$) and only then B can be loaded into the machine. The place $p_{ABW}$ corresponds to the machine output. A token is in $p_{ABW}$ if the assembled product AB is at the machine output. After the product leaves machine M, M is free again.

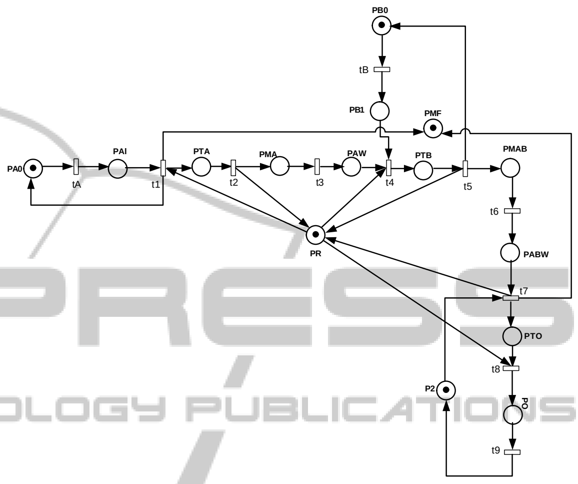

Figure 2: Petri Net of RARM.

## 5 PLANNER MODEL

Distributed planning is considered as a very complex task (David Jung, 1999), (Oscar Sapena, 2008). In fact, distributed planning ensures how the multi-robot system should plan to work together, to decompose the problems into subproblems, to assign these subproblems, to exchange the solutions of subproblem, and to synthesize the whole solution which itself is a problem that the robotic agents must solve (Sergio Pajares Ferrando, 2013), (Pascal Forget, 2008), (Malik Ghallab, 2014). The actions of the other robotic agents can induce a combinatorial explosion in the number of possibilities which the planner will have to consider, returning the space of research and the size of solution exponentially larger.

The Decision making component encapsulates a behavior-based action selection mechanism. Decision making is responsible for realizing the robotic agent's tasks by invoking actions in the virtual environment. To enable a situated multi-robot system to set up collaborations, behavior-based action selection mechanisms are extended with the notions of role and situated commitment. The conceptual model of a robotic agent is constituted by three components including (i) the planner, (ii) the plan-execution agent, and (iii) the world in which the plans are to be executed (the formal representation is based on the work (Malik Ghallab, 2004)).

The planner's input includes descriptions of the state transition system denoted by $\Sigma$, the initial state(s) that $\Sigma$ might be in before the plan-execution robotic agent performs any actions, and the desired objectives (e.g., to reach a set of states that satisfies a given goal condition, or to perform a specified task, or a set of states that the world should be kept in or kept out of, or a partially ordered set of states that we might want the world to go through). If the planning is being done online (i.e., if planning and plan execution are going on at the same time), the planner's input will also include feedback about the current execution status of the plan or policy. The planner's output consists of either a plan (a linear sequence of actions for the robotic agent to perform) or a policy (a set of state-action pairs with at most one action for each state).

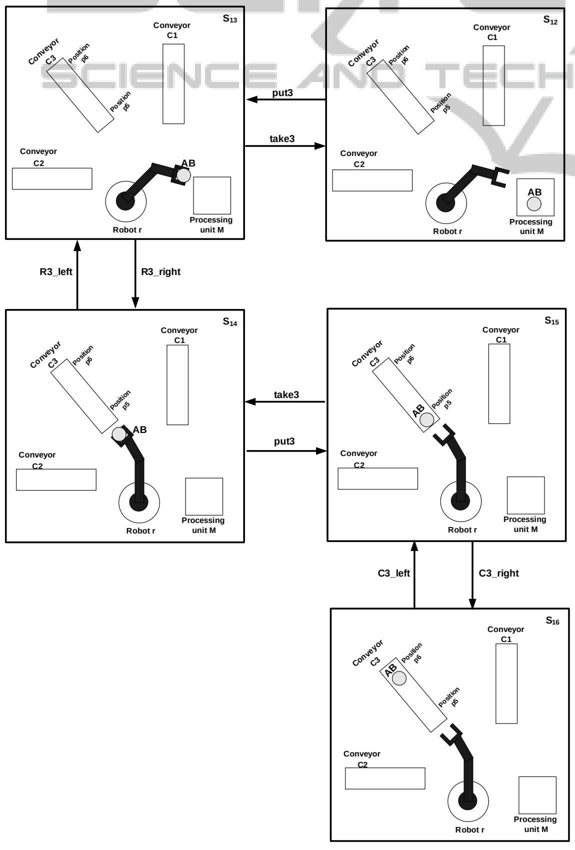

Figure 3: The third state-transition for RARM.

**Running Example**

*According to figure 3 :*

- *A set of positions* $\{p_1, p_2, \ldots\}$ *: A position is used to localise the workpiece A, B or AB;*
- *A set of robotic agents* $\{r_1, r_2, \ldots\}$ *: Each robotic agent transfers a workpiece one after one to be processed;*
- *A set of workpieces of type A* $\{a_1, a_2, \ldots\}$*;*
- *A set of workpieces of type B* $\{b_1, b_2, \ldots\}$*;*
- *A set of workpieces of type AB* $\{ab_1, ab_2, \ldots\}$*;*
- *A set of conveyors* $\{C_{1i}, C_{2i}, C_{3i}\}$ *: A conveyor* $C_{1i}$ *(resp.* $C_{2i}, C_{3i}$*) is responsible for transfering set of workpieces of type A (resp B, AB);*
- *A set of processing Centers M* $\{M_1, M_2, \ldots\}$ *: first one A-workpiece is inserted into M and processed, then one B-workpiece is added into the center M, and last both workpieces are assembled.*

*The set of states is* $\{s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}, s_{12}, s_{13}, s_{14}, s_{15}, s_{16}\}$

*There are nine possible actions in the domain.*

- *a workpiece of type A is trasnported to the left from position p1 to position p2;*
- *the robotic agent transports a workpiece of type A;*
- *the piece is put in the processing unit M;*
- *a workpiece of type B is trasnported to the left from position p3 to position p4;*
- *the robotic agent transports a workpiece of type B;*
- *the piece is put in the processing unit M;*
- *the robotic agent picks up the assembled piece;*
- *the assembled piece is put on the conveyor C3;*
- *a workpiece of type AB is trasnported to the right from position p5 to position p6.*

*The set of actions is* $\{C1\_left, C1\_right, R1\_left, R1\_right, C2\_left, C2\_right, R2\_left, R2\_right, C3\_left, C3\_right, R3\_left, R3\_right, take_1, take_2, take_3, load_1, load_2, load_3, put_1, put_2, put_3, process_1, process_2\}$

*The current configuration of the domain is denoted using instances of the following predicates, which represent relationships that change over time.*

- *occupied*(c1) *(resp. occupied*(c2)*, occupied*(c3)*): conveyor c1 (resp. c2, c3) is already occupied by a workpiece of type A (resp. B, AB);*
- *empty(c1) (resp. empty(c2), empty(c3)): conveyor c1 (resp. c2, c3) is already ready to transport a workpiece of type A (resp. B, AB);*
- *at(r,p2) (resp. at(r,p4), at(r,p5)): robotic agent r is currently at position p2 (resp. p4, p5);*
- *loaded(r, a) (resp. loaded(r, b), loaded(r, ab)) : robotic agent r is currently loading the workpiece a (resp. b, ab) of type A (resp. B, AB);*

- *put(r, a) (resp. put(r, b),put(r, ab)): robotic agent r is currently putting the workpiece a (resp. b, ab) of type A (resp. B, AB);*
- *empty(r): the robotic agent r is empty;*
- *empty(a) (resp. empty(b), empty(ab)): there is no workpiece of type A (resp. B, AB).*

## 6 COMPETENCE MODEL

The Competence Model of a robotic agent is based on a Fuzzy Logic Control. The Fuzzy Logic Control is a methodology considered as a bridge on the artificial intelligence and the traditional control theory (Mria Kukov, 2013). This methodology is usually applied in the only cases when exactitude is not of the need or high importance (Jianhua Dai, 2013). As it is stated in (Marijana Gorjanac Ranitovi, 2014) , Fuzzy Logic is a methodology for expressing operational laws of a system in linguistic terms instead of mathematical equations.

The basic form of a fuzzy logic agent consists of (Zadeh, 2008): Input fuzzification, Fuzzy rule base, Inference engine and Output defuzzification.

**Running Example**

**(i) Fuzzification**
*The number of defected pieces is measured through a sensor related to the system. The range of number of defected pieces varies between 0 to 40, where zero indicates the rate of defected pieces of A that is null (each piece is well) and 40 indicates the rate of defected pieces of A is very high.*
*Now assume that the following domain meta-data values for these variable, VF = very few, F = few, Md = medium, Mc = much, VMc = very much. Assume that the linguistic terms describing the meta-data for the attributes of entities are: VF = [0,..,10], F = [5,..,15], Md = [10,..,20], Mc = [15,..,25] and VMc = [20,..,40].*
*Based on the metadata value for each attribute the membership of that attribute to each data classification can be calculated. In the Figure 4, triangular and trapezoidal fuzzy set was used to represent the state of defected pieces from A classifications (i.e. state of defected pieces from A classification levels: VF , F, Md, Mc, VMc whereas state of defected pieces from B classification levels: F, Md, Mc).*

**(ii) Rule Engine**
*We take as example, the first column from the Table 1: IF number of defected pieces from A is Very Few and number of defected pieces from B is Few Then Production is High.*

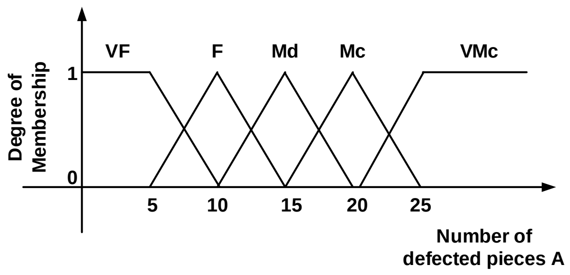

Figure 4: Fuzzy State of defected pieces from A.

*IF number of defected pieces from A is Few and number of defected pieces from B is Few Then Production is High.*
. . .

Table 1: Fuzzy Control rules for the robotic agent.

| A B | **F** | **Md** | **Mc** |
|---|---|---|---|
| **VF** | H | H | M |
| **F** | H | H | M |
| **Md** | H | M | L |
| **Mc** | M | L | N |
| **VMc** | M | L | N |

**(iii) Defuzzification**
*Defuzzification is the conversion of a fuzzy quantity to a precise quantity. There are many methods to calculate it such as Max membership, Centroid method, Weighted average method, Mean max membership, Center of sums, Center of largest area and First (or last) of maxima. Obviously, the best defuzzification method is context-dependant (Zadeh, 2008).*

## 7 COMMUNICATION MODEL

Communication is responsible for communicative interactions within a multi-robot system. Message exchange enables robotic agents to share information directly and set up collaborations. The communication module processes incoming messages and produces outgoing messages according to well-defined communication protocols. To do that, we implement the different robotic agents with the platform JADE (for more details, we refer to (Fabio Bellifemine, 2010b), (Caire, 2009), (Fabio Bellifemine, 2010a)).

**Running Example**
*The robotic agent $RARM_1$ is going to reduce the production. The another robotic agent $RARM_2$ may either decrease the production (in which case the robotic agents can cooperate together) or increase the production (in which case neither robotic agent can*

*cooperate). Thera are two possible actions can be modeled as nondeterministic outcomes.*

## 7.1 Message Exchanged Between Robotic Agents

A multi-robot system never interact through method calls but by exchanging asynchronous messages. Obviously, inter-agent interaction will be very difficult until all robotic agents adopt the same communication language, and fortunately ACL standards ensure this requirement. This format comprises a number of fields and in particular: (1) the sender of the message, (2) the list of receivers, (3) the communicative intention (also called performative) indicating what the sender intends to achieve by sending the message (for example the performative can be REQUEST, INFORM, QUERY_IF, CFP (call for proposal), PROPOSE, ACCEPT_PROPOSAL, REJECT_PROPOSAL, and so on). (4) The content i.e. the actual information included in the message which may be string in simple cases; otherwise we need a content language, a corresponding ontology, and a protocol. (5) The ontology i.e. the vocabulary of the symbols used in the content and their meaning (both the sender and the receiver must be able to encode expressions using the same symbols to be sure that the communication is effective).

### 7.1.1 Sending a Message

Sending a message to another robotic agent is as simple as filling the fields of an *ACLMessage* object and then call the *send*() method of the Agent class.

**Running Example**

```
//The code below informs a robotic agent whose nickname is Robot1 that the production must be decreased.
ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
msg.addReceiver(new AID("Robot1", AID.ISLOCALNAME));
msg.setOntology("Production");
msg.setContent("We must decrease in the production");
send(msg);
```

### 7.1.2 Receiving a Message

A robotic agent can pick up messages from its message queue by means of the *receive*() method. This method returns the first message in the message queue (removing it) or null if the message queue is empty and immediately returns.

**Running Example**

```
ACLMessage msg = receive();
if (msg != null) {
// Process the message
}
```

### 7.1.3 Blocking Behavior Waiting a Message

Some behaviors must be continuously running and at each execution of their *action*() method, must check if a message is recceived and perform some action.

**Running Example**

```
public void action() {
ACLMessage msg = myAgent.receive();
if (msg != null) {
// Message received. Process it
...
}
else {
block();
}
}
```

### 7.1.4 Selecting a Message

When a template is specified, the *receive*() method returns the first message (if any) matching it, while ignores all non-matching messages. Such templates are implemented as instances of the *jade.lang.acl.MessageTemplate* class that provides a number of factory methods to create templates in a very simple and flexible way.

**Running Example**

The *action*() method is modified so that the call to *myAgent.receive*() ignores all messages except those whose performative is REQUEST:

```
public void action() {
MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.REQUEST);
ACLMessage msg = myAgent.receive(mt);
if (msg != null) {
// REQUEST Message received. Process it
...
}
else {
block();
}
}
```

## 8 RELATED WORK

In literature, we can find three kinds of architecture. The first one is Shakey's architecture which is decomposed into three functional elements: sensing, planning, and executing (Nilsson, 1980). The sensing system translates the camera image into an internal world model. The planner takes the internal world model and a goal and generates a plan (i.e., a series of actions) that would achieve the goal. The executor takes the plan and sends the actions to the robot. This approach is called the sense-plan-act (SPA) paradigm.

But the SPA paradigm has problems. First, planning in any realworld domain takes a long time, and the robot would be blocked waiting for planning to complete. Second, and more importantly, execution of a plan without involving sensing is dangerous in a dynamic world.

The second kind of architecture is Reactive planning in which plans are generated quickly and relied more directly on sensed information (Fir, ). The most influential work, however, is the Subsumption architecture of Rodney Brooks of MIT (Brooks, 1986a). A Subsumption architecture is built from layers of interacting finite state machines each connecting sensors to actuators directly. These finite state machines were called behaviors (leading some to call Subsumption ”behavior-based” or ”behavioral” robotics (Arkin, 1998)).

However, it is very difficult to compose behaviors to achieve long-range goals and it is almost impossible to optimize robot behavior.

The last kind of architecture is Layered Robot Control Architectures. Among the many instances of layered architectures, we can distinguish between two fundamental classes: horizontally layered architectures (such as the ones developed by (Brooks, 1986b), (Kaelbling, 1990), and (Ferguson, 1992)) and vertically layered architectures (such as MECCA (D. D. Steiner and Lerin, 1993) and (Muller and Pischel, 1994)). Whereas all the layers of an agent have access both to the perception and action components in horizontal architectures, only one (and normally: the lowest) layer has a direct interface to these facilities in the vertical approach.

Currently two kinds of contributions in the multi-robot systems are considered related to this paper, namely reactivity and social cooperation.
The difference of the architecture proposed in this paper lies in emphasis on reactivity and social cooperation between multiple robots, while most of other architectures analysed by Goodwin (Goodwin, 2008) are concentrating on controlling multiple parts of a single robot.

The main difference of the proposed architecture from STEAM (Tambe, 1997) is in the task allocation mechanism. While STEAM uses the main task coordinator (team leader) that issues orders to team members, the proposed architecture focuses on individual agents being able to apply for tasks, therefore increasing their autonomy.

In ALLIANCE (Simmons, 1994), it is harder to replace actual robots with a software simulator, because single point of interface between deliberative and reactive layers makes it easier to test application.

The main difference from CENTIBOTS is in complexity of each robot CENTIBOT system uses robots with complex sensors and processing units while the proposed architecture focuses on use of low-cost hardware and in order to achieve sufficient environmental data quality, supplements deliberative level with appropriate algorithms and intelligence.

## 9 CONCLUSION

The main aim of this paper is how to ensure a distributed planning in Multi-Robot System composed of several intelligent autonomous robotic agents able to take the initiative instead of simply reacting in response to its environment. Our solution to this problem is the use of the 5 Capabilities Model (as it was presented, 5 levels: Environment, Self, Planner, Competence and Communication). The 5 Capabilities Model can be easily implemented where each model is represented with a process collaborating with the other processes. The 5C Model, based on the principle of separation of concerns, has the following interests: (i) The design is general enough to cope with various kinds of embedded-software application (therefore, the 5C Model is uncoupled from the application); (ii) The robotic agent is represented through five dimensions where each model is independent from the other which permits to change one without having to change the other.

Our future work is the design of an autonomous robot integrating cognitive abilities with other capabilities such as locomotion, prehension and manipulation.

## REFERENCES

Arkin, R. C. (1998). Behavior-based robotics. *MIT Press, Cambridge MA*.

Bordini, R. and all. (2006). A survey of programming languages and platforms for multi-agent systems. *Informatica*, 30(1):33–44.

Branislav Hrz, M. Z. (2007). Modeling and control of discrete-event dynamic systems with petri nets and other tools. page 67.

Brooks, R. A. (1986a). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14–23.

Brooks, R. A. (1986b). A robust layered control system for a mobile robot. *In IEEE Journal of Robotics and Automation*, 2.

C. J. van Aart, B. J. W. and Schreiber, A. T. (2004). Organizational building blocks for design of distributed intelligent system. *International Journal of Human-Computer Studies*, 61(5):567599.

Caire, G. (2009). *Jade Tutorial: Jade Programming For Beginners*.

Chuan-Jun Su, C.-Y. W. (2011). Jade implemented mobile multi-agent based, distributed information platform for pervasive health care monitoring. *Applied Soft Computing*, 11(1):315–325.

D. D. Steiner, A. Burt, M. K. and Lerin, C. (1993). The conceptual framework of mai2l. *In Pre-Proceedings of MAAMAW'93, Neuchatel, Switzerland*.

David Jung, A. Z. (1999). An architecture for distributed cooperative planning in a behaviour-based multi-robot system. *Robotics and Autonomous Systems*, 26(23):149–174.

Fabio Bellifemine, Giovanni Caire, T. T. G. R. (2010a). *Jade Programmers Guide*.

Fabio Bellifemine, Giovanni Caire, T. T. G. R. R. M. (2010b). *Jade Administrators Guide*. The publishing company.

Ferguson, I. A. (1992). Touring machines: An architecture for dynamic, rational, mobile agents. *PhD thesis, Computer Laboratory, University of Cambridge, UK*.

Goodwin, J. (2008). A unified design framework for mobile robot systems. *PhD Thesis. University of the West of England*.

Jianhua Dai, H. T. (2013). Fuzzy rough set model for set-valued data. *Fuzzy Sets and Systems*, 229(3):54–68.

Kaelbling, L. P. (1990). An architecture for intelligent reactive systems. *In J. Allen, J. Hendler, and A. Tate, editors, Readings in Planning*, pages 713–728.

Malik Ghallab, Dana Nau, P. T. (2004). *Automated Planning*.

Malik Ghallab, Dana Nau, P. T. (2014). The actor's view of automated planning and acting: A position paper. *Artificial Intelligence*, 208(3):1–17.

Marijana Gorjanac Ranitovi, A. P. (2014). Lattice representations of interval-valued fuzzy sets. *Fuzzy Sets and Systems*, 236:50–57.

Mria Kukov, M. N. (2013). Principles of inclusion and exclusion for fuzzy sets. *Fuzzy Sets and Systems*, 232(3):98–109.

Muller, J. P. and Pischel, M. (1994). Integrating agent interaction into a planner-reactor architecture. *In M. Klein, editor, Proc. of the 13th International Workshop on Distributed Artificial Intelligence, Seattle, WA, USA*.

Nilsson, N. J. (1980). Principles of artificial intelligence. *Tioga Press, Palo Alto CA*.

Oscar Sapena, Eva Onaindia, A. G. M. A. (2008). Engineering applications of artificial intelligence. *Some Fine Journal*, 21(5):698–709.

Pascal Forget, Sophie DAmours, J.-M. F. (2008). Multi-behavior agent model for planning in supply chains: An application to the lumber industry. *Robotics and Computer-Integrated Manufacturing*, 24(5):664–679.

Salvatore Vitabile, Vincenzo Conti, C. M. F. S. (2009). An extended jade-s based framework for developing secure multi-agent systems. *Computer Standards & Interfaces*, 31(5):913–930.

Sergio Pajares Ferrando, E. O. (2013). Context-aware multi-agent planning in intelligent environments. *Information Sciences*, 227:22–42.

Simmons, R. (1994). Structured control for autonomous robots. *IEEE*, 10(1):34–43.

Tambe, M. (1997). Agent architectures for flexible, practical teamwork. *Proceedings of the National Conference on AI, Providence, Rhode Island, USA*, pages 22–28.

van Aart, C. J. (2004). Organization principles for multi-agent architectures. *PhD thesis, University of Amsterdam, Faculty of Social and Behavioural Sciences*.

Zadeh, L. A. (2008). Is there a need for fuzzy logic? *Information Sciences*, 178(13):2751–2779.