

# Smarts Agents and Smarts Environments: a Predictive Approach to Replanning

Alfredo Garro<sup>1</sup> and Sergio Greco<sup>2</sup> and Fabio Palopoli<sup>3</sup>

**Abstract.** There is an increasing interest in the design and development of smart environments as they can be used to implement valuable applications aiming at improving the quality of life. However, another important issue, which has not been yet fully investigated, concerns the design and development of agents able to effectively and efficiently act in the new and emerging smart environments characterized by an increasing complexity and openness. This paper proposes a novel approach for dynamic (re)planning for agents which come into and would act in such kind of smart environments. The flexibility of the approach makes it exploitable also in the design of the replanning capabilities of smart environments both conceived as single rational entities that as a multi agent systems.

## 1 INTRODUCTION

A Smart Environment can be defined as “a region of the real world that is extensively equipped with sensors, actuators and computing components” [23]; it can sense what occurs within itself and its surroundings and adapts its actions accordingly to achieve its specific goals (e.g. to make inhabitants’ lives more comfortable [4]). There is an increasing interest in the development of smart environments as they can be used to implement valuable applications aiming at improving the quality of life [5]. However, suitable paradigms and technologies are required for tackling the development of smart environments due to their ambitious goals [4]. Moreover, an important feature which makes more difficult their development is their *openness*: the entities which *inhabit* the environment are heterogenous, not known *a priori* and may change across time. Several research efforts deal with the design and development of smart environments [1, 7, 22]. However, another important issue, that has not been yet fully investigated, concerns with the design and development of autonomous (artificial) entities (agents) able to effectively and efficiently act in such kind of open environments [2]. In fact, as a smart environment is endowed with goal-directed and autonomous behaviors [5], autonomous entities, which come into and would act in such an environment have to be able to properly interact with the environment and to constantly adapt their behavior to the behavior of the environment itself. Even if a smart environment provides several advanced services which can be used by these entities for achieving their goals in a more effective manner, the goal-directed and autonomous behavior of the environment makes it a complex and dynamic environment in which to act so requiring to

these entities more *smart* capabilities. In particular, a key capability is planning. To support this statement, it can be useful to envisage a scenario concerning an autonomous entity (agent) which come into an open smart environment and is engaged in executing a plan for reaching a goal. Due to the complex and dynamic nature of the environment, during the plan execution it may occur an event which does not let the agent to continue the execution of the plan so it becomes necessary to build a new plan for keeping the goal reachable. Typically, during the replanning phase the agent reacts to the occurred event by applying reactive rules. After a certain amount of time the new plan is ready for the execution so the agent can switch from the reactive behavior to the execution of the new plan. However, mainly due to the complexity and the autonomous behavior of the environment, the state of the environment at the switching time can be very different from the state in which the event has occurred. If this possible evolution of the environment is not accurately considered in building the new plan, the computed plan could not be executable.

This paper proposes a novel approach for dynamic (re)planning which is based on a hybrid agent architecture and is specifically conceived for agents acting in smart environments. In particular, a prediction module is used to predict the environment state in which the agent will switch from the reactive behavior to the execution of the new plan. In addition, as at the switching time the conditions in which the agent should start the execution of the new plan could not be exactly the predicted one, it is computed and executed a “pre-plan” able to bring the agent in the conditions in which it is possible to start the execution of the new plan.

Although there are several approaches for dynamic (re)planning [12, 14, 24, 26, 28], the proposed predictive approach differs from the others in the way that the deliberation is evaluated with respect to the future state in which the deliberation output (i.e. the plan) will be executed. As far as the authors are concerned, there are few similar approaches reported in the literature [3, 11].

Moreover, although there are several studies and proposals which aim at providing smart environments with planning capabilities [8, 15, 16, 17, 18, 20, 21, 25], the problem concerning the planning capabilities of (artificial) autonomous entities which come into and would act in smart environments has been not explicitly and deeply investigated. Furthermore, it is worth noting that the flexibility of the proposed approach make it exploitable also in the design of the (re)planning capabilities of a smart environment both conceived as a single rational entity [5] that as a multi agent system [22]. In the former case the smart environment generates and executes plans to reach its own goals. Again, due to the possible events which could occur during the execution of these plans, it

<sup>1</sup> Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, Italy, email: garro@unical.it

<sup>2</sup> Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, Italy, email: greco@unical.it

<sup>3</sup> Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, Italy and Exeura S.r.L., email: f.palopoli@exeura.com

can be necessary to replan to keep the goals reachable, so dealing with the problems highlighted in the above described scenario. In the second case, the autonomous agents, which compose the smart environment and cooperate for achieving the environment goals, should be provided with suitable planning and replanning capabilities as they have to deal with several potentially unexpected events which must be properly managed.

This paper is organized as follows: in Section 2 the proposed replanning approach is presented and contextualized for smart agents which act in smart environments; Section 3 reports an example application concerning an Agent which acts in a (smart) office; eventually, in Section 4 conclusions are drawn and future works delineated.

## 2 A PREDICTIVE APPROACH TO REPLANNING

In this section, a new approach for dynamic replanning based on a novel agent architecture ( $\mathcal{HPA}$  - Hypersphere and Prediction based Architecture) is presented. In particular, this approach has been conceived for dealing with the scenario discussed in Section 1 concerning an autonomous entity (hereafter  $\mathcal{HPA}$  Agent) which come into an open smart environment and is engaged in executing a plan for reaching a goal.

The  $\mathcal{HPA}$  Agent is supposed to be benevolent [10] and to act not in contrast with the smart environment rules and goals. The smart environment allows the  $\mathcal{HPA}$  Agent to gather knowledge about its goals and behaviors in order to facilitate both actual and future interactions between the Agent and itself.

### 2.1 Basic Definitions

The considered scenario is as follows: the goal of an  $\mathcal{HPA}$  Agent is to transform a state  $s_0 \in S_0$  (the set of the Agent's initial states), into a state  $s_g \in S_g$  (the set of the Agent's goal or final states). In order to get the state  $s_g$ , starting from the state  $s_0$ , it executes a plan  $p$ . During the plan execution the Agent's state changes due: (i) to the effects of the executed actions which strongly depends on the environment in which they are performed; (ii) to the events which concurrently happen in the environment in which the Agent is acting.

Given a set of possible states  $S = \{s_1, \dots, s_n\}$  and a set of actions  $A = \{a_1, \dots, a_m\}$ , a plan  $p = \langle a_i, \dots, a_j \rangle$  is a sequence of actions belonging to  $A$ . The set of actions define a transition function and the application of an action  $a_i$  to a state  $s_j$ , denoted by  $a_i(s_j)$  gives a new state  $s_k$ . The notation  $a_1.a_2(s_j)$  will be used in substitution of  $a_2(a_1(s_j))$ .

**Definition 1** An  $\mathcal{HPA}$  Agent is a tuple  $\mathcal{AG} = (S_0, S_g, S, P, R)$  where  $S$  denotes the set of its states,  $S_0 \subseteq S$  denotes the set of its initial states,  $S_g \subseteq S$  the set of its final states,  $P$  the set of plans, where for each  $p \in P$  there is a  $s_0 \in S_0$  such that  $p(s_0) \in S_g$ , and  $R$  denotes the set of reactive rules, where each  $r \in R$  is a sequence of actions such that there is a  $s \in (S - S_0)$  and  $r(s) \in S$ .

An  $\mathcal{HPA}$  Active Agent  $\mathcal{AAG} = (s_0, s_g, s, p, q, R)$ , derived from an  $\mathcal{HPA}$  Agent  $\mathcal{AG} = (S_0, S_g, S, P, R)$ , is characterized by an initial state  $s_0 \in S_0$ , a final states  $s_g \in S_g$ , the current (or observed) state  $s$ , the already executed plan  $p$  and the plan to be executed  $q$ .  $\square$

An  $\mathcal{HPA}$  Active Agent  $\mathcal{AAG} = (s_0, s_g, s, p, q, R)$  is in a correct state if  $q(s) = s_g$ ; otherwise, it is in a wrong state if  $q(s) \neq s_g$ .

**Definition 2** An influential unexpected event  $e$  is an action, not necessarily belonging to  $A$ , which is executed before  $q$  such that  $e.q(s) \neq s_g$ .  $\square$

Thus, when an Active Agent  $\mathcal{AAG} = (s_0, s_g, s, p, q, R)$ , evolves into an Agent  $\mathcal{AAG} = (s_0, s_g, s', p, q, R)$ , where  $s'$  is a wrong state, it is supposed that an influential unexpected event  $e$ , modifying the correct state  $s$  of the Agent into the wrong state  $s'$  (i.e.  $e(s) = s'$ ), has occurred. Moreover, if after the execution of an action  $a$ , belonging to  $A$ , the Agent evolves in a wrong state  $s' \neq a(s)$ , it is assumed that there has been an influential unexpected event  $e$  such that  $a.e(s) = s'$ .

**Definition 3** Given an  $\mathcal{HPA}$  Active Agent  $\mathcal{AAG} = (s_0, s_g, s, p, q, R)$ , an uninfluential unexpected event  $e$  is an action, not necessarily belonging to  $A$ , which is executed before  $q$  such that  $e.q(s) = s_g$ .  $\square$

Thus, when an Active Agent  $\mathcal{AAG} = (s_0, s_g, s, p, q, R)$ , evolves into an agent  $\mathcal{AAG} = (s_0, s_g, s', p, q, R)$ , such that  $q(s') = s_g$ , it is supposed that an uninfluential unexpected event  $e$ , modifying the state  $s$  of the agent into another correct state  $s'$  (i.e.  $e(s) = s'$ ), has occurred. Moreover, if after the execution of an action  $a$ , belonging to  $A$ , the Agent evolves in a correct state  $s' \neq a(s)$ , it is assumed that there has been an uninfluential unexpected event  $e$  such that  $a.e(s) = s'$ .

The sequence of actions which compose a reactive rule is different from a plan (i.e.  $P \cap R = \emptyset$ ) as it is not conceived for reaching a goal state starting from an initial state but for reacting to an influential unexpected event during the replanning process as described in Section 2.2.

### 2.2 The Replanning Process

The main activities which characterize the  $\mathcal{HPA}$  Agent behavior are showed in the Figure 1. Each activity is executed by a dedicated module as described in details in Section 2.3.

The  $\mathcal{HPA}$  Agent behavior is as follows. At time  $t_0$ , in the state  $s_0$ , the  $\mathcal{HPA}$  Agent starts to interact with the smart environment by executing a plan  $p_0$ , such that  $p_0(s_0) = s_g$ , in order to get the goal state  $s_g$ . At time  $t_j$  an influential unexpected event happens (see Definition 2). Thus, the Agent evolves in a state  $s'$  from which, by keeping executing  $p_0$ , it is not possible to get the goal state  $s_g$ .

The  $\mathcal{HPA}$  Agent perceives the event, analyzes it and recognizes that an influential unexpected event happened, so it stops executing the current plan (which is became ineffective) and, concurrently, it both adopts a reactive behavior and starts replanning.

The reactive behavior consists in the selection and execution of reactive rules from the set  $R$  of the Agent (see Definition 1). More in detail  $R$  can be conceived as a table in which reactive rules are selected by using the current state as an index. In other words, from this time, until no switch to any new plan is performed, from an external observers point of view, the  $\mathcal{HPA}$  Agent is characterized by a purely reactive behavior.

Concurrently to the execution of the reactive behavior, the  $\mathcal{HPA}$  Agent starts replanning by performing the following activities:

1. *Time bounds definition.* The  $\mathcal{HPA}$  Agent defines the computational and execution time bounds for the other activities which compose the  $\mathcal{HPA}$  replanning process (see Section 2.2.1 for a discussion).

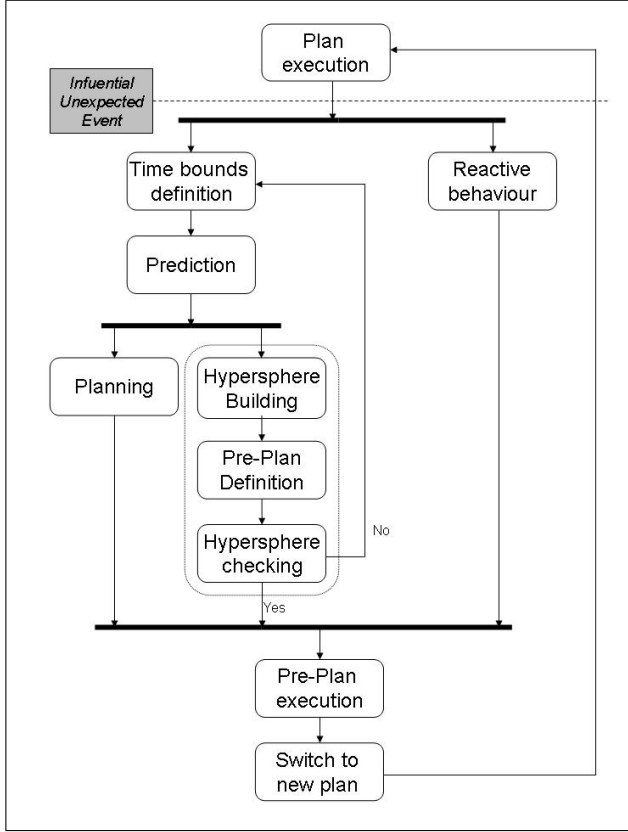


Figure 1. The  $\mathcal{H}PA$  approach to replanning

2. *Prediction.* The  $\mathcal{H}PA$  Agent computes the future state  $s_e$  in which it should be at the time  $t_e$  at which the new plan execution should start. The prediction must be computed in  $\Delta T_{prd}$  seconds ( $t_e$  and  $\Delta T_{prd}$  have been set during step 1).
3. *Planning.* The  $\mathcal{H}PA$  Agent computes the new plan which has as initial state  $s_e$  and final state  $s_g$ . The new plan must be computed in  $\Delta T_{pld}$  as set during step 1.
4. *Pre-planning, Hypersphere Definition & Checking.* This activity, which is executed in parallel with *Planning*, is required because, due to the dynamics and complexity of the environment, there is no guarantee that the Agent state will be exactly equivalent to  $s_e$  at time  $t_e$ . The idea is to compute a set of states in the *neighborhood* of  $s_e$  from which it is possible to reach  $s_e$ , at latest at time  $t_e$ , by executing a set of actions (pre-plan). Specifically, the *Pre-planning, Hypersphere Definition & Checking* activity consists of the following consecutive steps:
  - (a) *Hypersphere Building.* The  $\mathcal{H}PA$  Agent computes the set of states (*Approximation Hypersphere*) from which it is possible to reach the state  $s_e$  in a certain amount of time ( $\Delta T_{ppe}$ ) by executing a pre-plan. The Hypersphere must be computed in  $\Delta T_{hsd}$  defined during step 1.
  - (b) *Pre-plan Definition.* The  $\mathcal{H}PA$  Agent defines the set of actions to be performed before executing the plan in order to reach the state  $s_e$  from a state belonging to the *Approximation Hypersphere*. The pre-plan must be computed in the time interval  $\Delta T_{ppd}$  defined during step 1.

- (c) *Hypersphere Checking.* The  $\mathcal{H}PA$  Agent tests at time  $t_e - \Delta T_{ppe}$  if it is in a state which belongs to the *Approximation Hypersphere*.

The subsequent behavior of the  $\mathcal{H}PA$  Agent depends on the outcome of the *Hypersphere Checking* step as follows:

- *case 1:* at time  $t_e - \Delta T_{ppe}$  the  $\mathcal{H}PA$  Agent has reached a state which belongs to the *Approximation Hypersphere*. In this case it stops executing the reactive behavior and starts the execution of the computed pre-plan. After the execution of the pre-plan, the  $\mathcal{H}PA$  Agent, which is now in  $s_e$  (if no other influential unexpected event has occurred in the meantime), can start executing the new plan.
- *case 2:* at time  $t_e - \Delta T_{ppe}$  the  $\mathcal{H}PA$  Agent has reached a state which does not belong to the *Approximation Hypersphere*. In this case it does not stop executing the reactive behavior and restarts the execution of the replanning process from the *Time bounds definition* activity which consequently remodulate the previously computed time bounds.

It is worth noting that if during any phase of the described replanning process another influential unexpected event occurs it causes the immediate ending of the process and the starting of a new one for managing the occurred event (the  $\mathcal{H}PA$  Agent starts reacting to the new event and replans again). However, this case is risk free, also if a large number of influential unexpected events occur, since the Agent simply resets the replanning process while continuing the execution of the reactive behavior. In the following subsections further details on some relevant aspects of the proposed approach to replanning are discussed.

### 2.2.1 Time bounds definition

The *time bounds definition* activity defines the time bounds for the execution of the above mentioned activities. In particular, it allows us to define the following time bounds:

- $\Delta T_{prd}$ : maximal time to compute the prediction output;
- $\Delta T_{hsd}$ : maximal time to define the Approximation Hypersphere;
- $\Delta T_{ppd}$ : maximal time for defining the pre-plan;
- $\Delta T_{pld}$ : maximal time for defining the new plan;
- $\Delta T_{ppe}$ : maximal time for executing the pre-plan.

It is worth recalling that the temporal execution sequence is as follows: first, the *Prediction* activity is executed; next, the Agent executes in parallel the *Pre-planning, Hypersphere Definition & Checking* activity and the *Planning* activity. The *Hypersphere Checking* execution time is negligible. From these considerations it is possible to compute the execution time  $t_e$ , the time at which the new plan can be executed:

$$t_e \geq t_{bde} + \Delta T_{ppe} + \max(\Delta T_{hsd} + \Delta T_{ppd}, \Delta T_{pld})$$

where  $t_{bde}$  is the time at which the execution of the *time bounds definition* activity ends.

The *time bounds* are properly tuned on the basis of the results of the previous replanning processes in which the Agent has been engaged. Two typical situations which cause the adjustment of the *time bounds* are the following:

- at  $t_e - \Delta T_{ppe}$  the Agent is in a state which does not belong to the computed *Approximation Hypersphere*, and thus it is necessary to

restart replanning. Probably, in this case, the dynamic of the environment are increasing so requiring a more fast replanning ( $t_e - t_j$  decreases and thus the *time bounds* need to be consequently re-modulated);

- a large number of influential unexpected events occur while the agent is replanning, so causing frequent resets of the replanning process and forcing the agent executing a reactive behavior. In this case can be necessary to re-modulate the *time bounds* to reduce the duration of the replanning process so trying to execute a new plan before the next influential unexpected event occurs.

These tuning capabilities make the *HPA* approach suitable for highly dynamic environment and able to scale in situations characterized by a large number of influential unexpected events, possibly due to a large number of agents acting in the same environment.

### 2.2.2 Prediction

After the definition of the time bounds, the prediction module starts; its output is a prediction of the state  $s_e$  at the time  $t_e$  at which the agent should start to execute the new plan.

The inputs of the prediction procedure are:

- a description of the influential unexpected event;
- a description of the state at time  $t_j$  - the time at which the agent has observed the influential unexpected event;
- the set  $R$  of reactive rules of the Agent;
- information about the (future) behavior of the smart environment (both in function and not in function of the occurrence of the influential unexpected event).

On the basis of these inputs, the prediction procedure calculates  $s_e$ . It is worth noting that the smartness of the environment makes it possible to gather useful information about its future behavior which can increase the accuracy of the prediction respect to what happen in *classic* environments where the Agent can only make assumptions on the environment evolution.

### 2.2.3 Hypersphere Building

Pre-plan and Hypersphere Definition are introduced in the *HPA* approach because there is no guarantee that the agent state will be exactly equivalent to  $s_e$  at time  $t_e$ . In fact, the output of the prediction procedure might be imprecise, for several reasons:

- the computational time available to the prediction activity is limited, and this fact influences the prediction degree of accuracy;
- an *influential* unexpected events might have occurred after  $t_j$  - it is worth recalling that:
  - the prediction is based on the state at time  $t_j$ ;
  - the occurrence of an *influential* unexpected event causes the restart of the replanning process;
- the reaction time may vary, and the response of the environment to a reaction could be slightly different from the prediction module's output;
- in the same way, even if the agent has acquired information on the (future) behavior of smart environment, it is very difficult to predict the exact effects of the actions that the smart environment will perform after  $t_j$ .

Thus, it is very difficult to predict exactly the agent state at time  $t_e$ . However, it is easier to predict that at time  $t_e - \Delta T_{ppe}$  the Agent could be in the *neighborhood* of the state  $s_e$ . If it is the case, the agent can perform a sequence of corrective actions (a pre-plan) in order to get the state  $s_e$ . The distance of the neighbor states of  $s_e$  must be properly limited so that it is possible to perform correcting actions (pre-plan) for getting state  $s_e$  no later than time  $t_e$ . More formally:

**Definition 4** An *Approximation Hypersphere* of a state  $s_e$  and time bound  $\Delta T_{ppe}$  is a set of states  $S$  such that for each state  $s_i \in S$  there is a pre-plan which is able to transform the state  $s_i$  in a state  $s_e$  in a time less than or equal to  $\Delta T_{ppe}$ .  $\square$

It is worth noting that in the *HPA* approach each state  $s_e$  has associated a unique pre-plan (defined on the basis of the properties of  $s_e$ ) and that each neighbor state  $s_i$  is such that the application of the pre-plan brings the Agent to state  $s_e$  before time  $t_e$ . Observe that it is possible to generalize by considering more alternative neighbors and pre-plans which could be defined in parallel, but this is outwith the aims of this paper.

## 2.3 The Hypersphere and Prediction based Architecture

The replanning process described in Section 2.2 is supported by the Hypersphere and Prediction based Architecture (*HPA*) for replanning shown in figure 2.

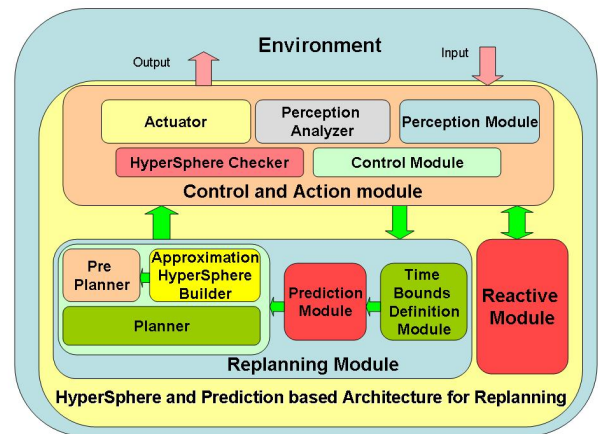


Figure 2. HPA Architecture for Replanning.

The *HPA* architecture consists of three macro modules: a *Control and Action Module*, a *Reactive Module* and a *Replanning Module*.

The *Control and Action Module* consists of the following submodules:

- *Perception Module*, which percepts the environment through a set of sensors;
- *Perception Analyzer*, which processes the perceptions received by the *Perception Module*;
- *Actuator*, which executes actions received both from the *Reactive* and the *Replanning Module*;

- *Control Module*, which controls the other submodules and determines the behavior of the agent by coordinating the *Reactive Module* and the *Replanning Module*. In particular, if the *Input Analyzer* identifies an influential unexpected events, the *Control Module*:

- asks to the smart environment information about the future behavior of the environment (which can also depend on the occurred event);
- stops the execution of the current plan;
- activates both the *Reactive* and the *Replanning Module*.

During the reaction of the Agent (see Section 2.2) the *Control Module*:

- requires the *Actuator* the execution of reactive actions received from the *Reactive Module*;
- stops the current replanning process and restarts the *Reactive* and *Replanning Module* in case of a new influential unexpected event;
- at time  $t_e - \Delta T_{ppe}$  requires the *Hypersphere Checker* to check if the Agent state belongs to the *Approximation Hypersphere*, and on the basis of the result requires the *Actuator* to execute the computed pre-plan or it restarts the *Replanning Module* and requires the *Actuator* to continue executing the reactive actions (see Section 2.2);

The *Replanning Module* is responsible of the replanning process described in Section 2.2. It consists of the following submodules:

- *Time Bounds Definition Module*, which defines time bounds for the replanning process as described in Section 2.2.1;
- *Prediction Module*, which computes the future state in which the Agent should be when the new plan execution should start (see Section 2.2.2);
- *Planner*, which computes the (new) plan for reaching the goal state starting from the future state predicted by the *Prediction Module*. It is possible to adopt several algorithms for the planning task [9, 13];
- *Approximation Hypersphere Builder*, computes the set of states in the neighborhood of the state predicted by the *Prediction Module* from which it is possible to reach the predicted state by executing a pre-plan in a given amount of time (see Section 2.2.3);
- *Pre-Planner*, which computes the pre-plan for reaching the state predicted by the *Prediction Module* starting from a state which belongs to the *Approximation Hypersphere*. Again, it is possible to adopt several algorithms for the pre-planning task [9, 13];

The *Reactive Module* is responsible of the reaction to an expected influential event; in particular, it selects a sequence of actions and sends them to the *Actuator* for their execution.

### 3 AN APPLICATION EXAMPLE

The application example envisages a scenario similar to that presented in [19, 27]. It consists of a mobile robot (Worker) which acts in a warehouse to properly store goods on the basis of their typologies. The warehouse is a smart environment. In particular, the Worker's goal is to move a certain quantity of goods from a point  $A$  to a point  $B$ . The map of the environment, its expected behavior, and a plan  $p_0$  to move from  $A$  to  $B$  are available. The internal architecture of the Worker is the *HPA* architecture presented in Section 2.3. At some time  $t_j$ , the Worker's *Perception Analyzer* detects that an unexpected obstacle lies on its way to  $B$  (e.g. a shelf is fallen down). Assume that

it categorizes the obstacle as an influential unexpected event. At this point, the Worker's *Control Module* starts two concurrent processes:

- it stops the execution of the plan  $p_0$  and starts the execution of the reactive behavior;
- it starts replanning.

For simplicity, the only trivial reactive rule for obstacle avoidance could be as follows:

RULE1:

```

if (there is an obstacle on your trajectory) then
    send an alert message to the environment;
    turn 90 degrees to your left;
    repeat
        move ahead at a speed of 2 Km/h;
    until (no obstacle is detected)
end if

```

The input to the replanning process is composed by relevant data about: (i) the environment and the Worker state at time  $t_j$ ; (ii) the expected behavior of the environment (e.g. the environment may unlook some doors along some alternative paths to keep point  $B$  reachable). Those data also includes a description of the obstacle and its location, the location of the Worker, its speed, and so on.

After the *Time bounds definition*, the next main step of replanning is the *Prediction*. The *Prediction Module* of the Worker, on the basis of the above described information, predicts the (future) state at time  $t_e$  (the time at which the new plan's execution must start). For instance, if the parameter  $t_e - t_j = 10$  seconds the execution of the new plan (that does not exist yet) must start 10 seconds after the time  $t_j$  in which the influential unexpected event occurred. The *Prediction Module* accesses the warehouse map, the set  $R$  of Worker's reactive rules, and the information regarding the expected behavior of the environment. Firstly, it matches its knowledge of the influential unexpected event - *there is an obstacle on the Worker's way* - against the set of reactive rules, deducing that the Worker should choose RULE1. Then it predicts, also taking into account the fact that the environment may unlook some doors along some alternative paths to keep point  $B$  reachable, that the Worker at time  $t_e$  should be in a point of the warehouse of coordinates  $(x_e, y_e)$ . At this point, if the environment was static, nothing more is necessary: the *Planning Module* knows that at time  $t_e$  the Worker will be in a well determined point of the warehouse so it is sufficient to return a new plan (the initial state and the goal state (unchanged) are known). But the environment is complex and dynamic, and the amount of time to predict the Worker's future location is bounded, thus the prediction has a fixed level of accuracy. There is no guarantee that the Worker really will be at  $(x_e, y_e)$  at time  $t_e$ . However, in the *HPA* approach, it is sufficient that the state reached by the Worker is in the neighborhood of the predicted state at time  $t_e - \Delta T_{ppe}$ ; more specifically, it is sufficient that the Worker's location is inside a "circle" whose centre is  $(x_e, y_e)$ : the circle is the *Approximation Hypersphere*. According to the *Approximation Hypersphere* definition, the circle's size has to be large enough to allow the robot to move from every point inside the circle to the point  $(x_e, y_e)$  in an amount of time equal (or less than)  $\Delta T_{ppe}$ .

If, at time  $t_e - \Delta T_{ppe}$ , the current state belongs to the *Approximation Hypersphere* (the Worker is in the circle), the Worker stops the execution of the sequence of reactive actions and initiates the pre-plan's execution. If the Worker reaches in useful time the point  $(x_e, y_e)$ , it starts the execution of the new plan.

If anything goes wrong, the Worker would continue working according to the reactive behavior and it starts to replan.

## 4 CONCLUSIONS AND FUTURE WORK

In recent years enormous interest has grown around the design and development of smart environments. This paper has presented a novel approach to (re)planning that provides effective planning capabilities to agents that come into and act in smart environments. The main novelty is that, while standard approaches compute the plans with respect to the current state, in the  $\mathcal{HPA}$  approach the computation of the plan is based on the (future) state in which the plan execution will start. In order to effectively replan, the agent exploits information received by the smart environment about its current and future behavior. Due to its flexibility, the  $\mathcal{HPA}$  approach can be easily adapted in order to provide smart environments themselves with effective (re)planning capabilities in the case in which they are modelled and implemented as (multi)agent systems.

Efforts are currently underway to: (i) further investigate the interrelationships among the different *time bounds* which drive the replanning process in order to define specific strategies for their (optimal) setting; (ii) extensively experiment the  $\mathcal{HPA}$  approach in real testbeds and more complex scenarios.

## REFERENCES

- [1] J. C. Augusto and C. D. Nugent (Editors). *Designing Smart Homes: The Role of Artificial Intelligence*. Lecture Notes in Artificial Intelligence, vol. 4008. Springer, Berlin, 2006.
- [2] M. Broxvall, M. Gritti, A. Saffiotti, B. S. Seo, and Y. J. Cho, "PEIS Ecology: Integrating Robots into Smart Environments", *In Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, Florida, May 2006.
- [3] A. Coddington and M. Luck, "A Motivation-based Planning and Execution Framework", *In International Journal on Artificial Intelligence Tools*, vol. 13(1), pp. 5-25, 2004.
- [4] D.J. Cook and S. Das, "Smart Environments: Technology, Protocols and Applications", Wiley-Interscience, 2004.
- [5] D. J. Cook, M. Youngblood, E. Heierman, K. Gopalratnam, S. Rao, A. Litvin, and F. Khawaja, "MavHome: An Agent-Based Smart Home", *In proceedings of the Conference on Pervasive Computing*, 2003.
- [6] D. J. Cook, M. Youngblood, and S. K. Das, "A Multi-agent Approach to Controlling a Smart Environment". *Designing Smart Homes* 165-182, 2006.
- [7] S. K. Das, and D. J. Cook, "Designing Smart Environments: A Paradigm Based on Learning and Prediction". *Mobile, Wireless, and Sensor Networks*. John Wiley & Sons, Inc., 2006.
- [8] S. K. Das, D. J. Cook, A. Battacharya, E. O. Heierman, and L. Tze-Yun, "The role of prediction algorithms in the MavHome smart home architecture", *IEEE Wireless Communications*, vol. 9, pp. 77-84, 2002.
- [9] M. Ghallab, D. Nau, and P. Traverso, "Automated Planning: Theory & Practice", Morgan Kaufmann, 2004.
- [10] N. Jennings and M. Wooldridge, "Agent Theories, Architectures, and Languages: A Survey", *Intelligent Agents: ECAI-94 Workshop on Agent Theories, Architectures, and Languages*, Springer-Verlag, Berlin, 1995.
- [11] M. Hadad, S. Kraus, Y. Gal, and R. Lin, "Temporal reasoning for a collaborative planning agent in a dynamic environment", *Annals of Mathematics and Artificial Intelligence*, vol. 37(4), pp. 331-379, 2003.
- [12] X. H. Hu and J. W. Dang, "Hybrid Agent Model to Design Real-time Distributed Supervising and Control Systems", *In Proceedings of the 9th International Conference on Control, Automation, Robotics and Vision*, pp. 1-5, Singapore, 2006.
- [13] M. LaValle, "Planning Algorithms", Cambridge University Press, 2006.
- [14] K. H. Low, W. K. Leow, and M. H. Ang, "Hybrid Mobile Robot Architecture with Integrated Planning and Control", *In Proceedings of the 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)*, 2002.
- [15] R. Levinson, "The planning and execution assistant and trainer (PEAT)", *Journal of Head Trauma Rehabilitation*, vol. 12, pp. 85-91, 1997.
- [16] A. Mihailidis, G. Fernie, and J. C. Barbenel, "The use of artificial intelligence in the design of an intelligent cognitive orthotic for people with dementia", *Assistive Technology*, vol. 13, pp. 23-39, 2001.
- [17] A. Mihailidis, B. Carmichael, and J. Boger, "The use of computer vision in an intelligent environment to support aging-in-place, safety, and independence in the home", *IEEE Transactions on Information Technology in Biomedicine*, vol. 8, pp. 238-247, 2004.
- [18] A. Mihailidis, B. Carmichael, J. Boger, and G. Fernie, "An intelligent environment to support aging-in-place, safety, and independence of older adults with dementia", *In Proceedings of the 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications*, Seattle, Washington, 2003.
- [19] F. Mizoguchi, H. Hiraishi, and H. Nishiyama, "Human-robot collaboration in the smart office environment," *In Proceedings of the 1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, Atlanta, USA, 1999.
- [20] M. C. Mozer, "Lessons from an adaptive house," in *Smart environments: Technologies, protocols, and applications*, D. Cook and R. Das, Eds. Hoboken, NJ: Wiley & Sons, pp. 273-294, 2005.
- [21] M. C. Mozer, "An intelligent environment must be adaptive", *IEEE Intelligent Systems and Their Applications*, vol. 14, pp. 11-13, 1999.
- [22] A. Omicini, A. Ricci, and G. Vizzari, "Building Smart Environments as Agent Workspaces", *In Proceedings of the 16th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, Paris, France. June 18-20, 2007.
- [23] P. Nixon, S. Dobson, and G. Lacey, "Managing Smart Environments", *Workshop on Software Engineering for Wearable and Pervasive Computing*, Limerick, Ireland, 2000.
- [24] T. Patkos, A. Bikakis, G. Antoniou, M. Papadopoulou, and D. Plexousakis, "Distributed AI for Ambient Intelligence: Issues and Approaches", *Ambient Intelligence*, Lecture Notes in Computer Science, Springer Berlin, vol. 4794, pp. 159-176, 2007.
- [25] M. E. Pollack, "Intelligent technology for an aging population", *AI Magazine*, vol. 26, pp. 9-24, 2005.
- [26] H. Sarmadi, "Fuzzy Hybrid Deliberative/Reactive Paradigm", *Formal Approaches to Agent-Based Systems*, Lecture Notes in Computer Science, vol. 3228, pp. 281-286, 2004.
- [27] D. Weyns, K. Schelfhout, T. Holvoet, and T. Lefever, "Decentralized control of E'GV transportation systems", *In Proceedings of AAMAS Industrial Applications*, pp. 67-74, 2005.
- [28] D. E. Wilkins, K. L. Myers, J. D. Lowrance, and L. P. Wesley, "Planning and Reacting in Uncertain and Dynamic Environments", *Journal of Experimental and Theoretical AI*, vol. 7, no. 1, pp. 197-227, 1995.