A self-adaptive system for improving autonomy and public spaces accessibility for elderly

Sameh Triki, Chihab Hanachi

A self-adaptive system for improving autonomy and public spaces accessibility for elderly

Sameh Triki [a], Chihab Hanachi [b]

[a] 118 Route de Narbonne, 31062 Toulouse
[a] sameh.triki02@gmail.com
[b] 2 Rue du Doyen-Gabriel-Marty, 31042 Toulouse
[b] hanachi@univ-tlse1.fr

Abstract.

Nowadays, there is an increasing need to provide a safe and independent living for cognitively deficient population. Notably, we have to improve seniors' autonomy and their public spaces accessibility. Giving these observations, the aim of this paper is to provide a personalized adaptive assisting system for elderly. More precisely, this paper presents the specification and implementation of a self-organizing multi-agent system able to abstract the different distributed components involved in user's environment. This system is able to detect different possible situations that a user could face in his daily outdoors activities and propose accordingly appropriate actions. This system not only learns user's habits from its perceptions but also improves its recommendations thanks to feedbacks provided by stakeholders (family, doctors …) following a reinforcement learning reasoning. Finally, we present our system evaluation specially its learning capabilities through different scenarios that have been generated automatically.

Keywords: Assisted Living system, Multi-agent system, AMAS theory, Reinforcement learning.

1 Introduction

Currently, the increasing ageing of the population is one of today's major problems. Some difficulties, such as cognitive deficiency among seniors, make the independent access to the city difficult and unfortunately encourage them to stay at home [1, 2]. The increasing population of elder people and their social isolation requires that more activities should be done in order to improve their quality of life. One possible way is to provide them with tools that assist them.

Nowadays, the rapid increasing of electronic components and the reduction of their cost have led to an explosion of the number and functionalities of smart devices. Applications of these devices have reached various domains and a considerable amount of progress was noticed in assisting seniors in their life such as home monitoring, fall

detection and geolocation gadgets. However, the majority of devices on the market, are designed for indoor care or limited to a defined zone or made to assist a limited set of predefined tasks [3, 4 and 5]. Moreover, they do not provide a personalized tool that is able to evolve according to the user needs and to adapt to their gradually cognitive decline or sudden habits changes [6].

This paper presents a research that aims at designing and developing an adaptive accompanying system that enables potentially vulnerable and dependent population to maintain their social life and to ease their access to urban services independently and to unsure user security while being outside. The targeted population is older adults having age-related cognitive deficiencies such as memory loss, difficulties in doing parallel tasks and activities planning.

We propose a self-organizing multi-agent system called Sadikikoi[1] able to perceive its environment and detect conventional or problematic situations. Our system will propose actions to respond to detected situations and will improve its behavior to adapt to the user habits and requirements by a reinforcement learning approach. The system should be able: to evolve using received perceptions coming from different components, to analyze and correlate those perceptions and to propose the action required for assisting the user in his daily life activities outdoors. Moreover, perceptions may be coming from a smartphone sensor, wearable sensors or other devices distributed in the environment. Thus a multi-agent system seems adequate to handle those different components and their interactions and to deal with scalability. Our multi-agent system combines the AMAS ("Adaptive Multi-agent System") theory and a reinforcement learning approach.

Our contribution is threefold. First, we provide a multi-agent architecture: its components and their interactions described with a UML diagram. Second, we define the self-adaptive feature of our system that is based on self-organization following a reinforcement approach. The self-organization feature relies on the automatic creation, modification or deletion of agents without external intervention. We give high-level algorithms and a UML sequence diagram to specify this feature. Third, we provide and discuss evaluations based on simulations.

The paper is organized as follows. Section 2 describes the state of the art of existing researches sharing a similar goal such as assisted living systems and their limitation. Section 3 shows the proposed multi-agent system, its architecture and the specification of its self-adaptive feature. Section 4 illustrates the implemented system and its usefulness. Section 5 concludes the paper.

2 Related Work

Several advanced researches on ambient assisted living (AAL) systems have already been done. In this section, we will compare the most representative AAL systems of the state of the art and specifically the closest ones (same target population, close functions…) to our research.

---

[1] The system build is called Sadikikoi and it is part of the compagnon project funded by the Midi-Pyrénées Region.

We consider eight systems (AMON, WEALTHY, UCS, COACH, PEAT, Autominder, OutCare and KopAL) that are compared considering two disciplines (see Tab. 1): computer science and social human science. Regarding social science, we consider user criteria including functional requirements and user needs. Regarding computer science, we take into account system requirements and the followed methodology (interdisciplinary, participative design)

AMON a wearable multi-parameter medical monitoring and alert system [4] and WEALTHY a Wearable Health Care System Based on Knitted Integrated Sensors [7] consist on wearable sensors that provide continuous monitoring of their user. Both share the same functionality: save user's information and send a signal. The information transmitted is stored in a server and then sent either to family, a neighbor or a helping person. Those devices are certainly interesting, though, is not adaptive but rather reactive device: when a defined condition is detected, a signal is sent. So this system will neither learn nor evolve according to the changing needs of its user.

Japan's Fujitsu UCS has unveiled a prototype cane equipped with a GPS that guides the seniors in their movements, but also to monitor them remotely and monitor their heart rate [8]. As this rod is still in the prototype stage there is not much details about it neither about the used techniques. It is definitely interesting but it relies on sending information and receiving control guidance based on predefined or communicated thresholds.

Many other assisting devices are defined for indoors assistance, some are usually targeting a specified task such as the COACH system (Cognitive Orthosis for assisting aCtivities in the Home) [5] that guide a user through hand washing task based on planning, other cognitive orthotics tools such as PEAT [9] and Autominder [10] also use automated planning to provide generic reminders about daily activities.

Others targets a limited zone: OutCare [11] and KopAL [12] support outdoor wandering issues related to disorientation by alerting the caregiver when leaving a predefined routes or deviating from daily usual routes.

Table.1 classifies the different described ambient assisted living devices AALs according to defined criteria extracted according to the user needs and requirements to create a useful and usable assisting device.

	Disciplines	Criteria	Sub-criteria	Ambient Assisted Living systems (AALs)					
				AMON [4]	WEALTHY [7]	UCS [8]	coach [5]	peat/autominder [9/10]	OutCare/kopAL [11,12]
SHS	USAGER	functional requirements (provided services)	autonomie	-	-	+	+	+	+
			mobility	--	--	++	--	--	+
			protection	--	--	+	--	--	+
		user needs and unfunctionnal requirements	vulerability	+	+	+	+	-	+
			acceptance	+	+	+	+	+	-
			transparency	-	++	-	--	-	-
	ENVIRONNEMENT	interaction with environmental components		--	-	+	-	--	-
		context aware		-	-	-	-	--	+
		covered scope (complete/limited zone)		--	-	+	--	+	--
INFORMATIQUE	METHODOLOGY	interdisciplinarity		-	-	+	-	-	-
		participative design (user centred approach)		+	-	+	-	-	+
	SYSTEM REQUIREMENTS	*adaptation/evolution*		--	--	--	--	--	--
		heteregenity		-	-	-	-	-	-
		ambient		-	-	++	-	--	+

Table 1. Classification of AAL systems

As described in Table.1[2], the majority of AAL systems are designed to respond to predefined situations with predefined conditions and circumstances but they are not targeting unpredictable situations. Even though they are targeting the same population and aiming at assisting elderly, those works differ from our work since they are not adaptive.

We discussed the use of those devices in healthcare based on individuals' medical conditions, such as physical or mental disabilities, chronic disease, or rehabilitation situations. One of the most important shortcomings is that those related works are not adaptive, nor personalized and tailored to the needs of each user.

Thus, it becomes essential to have an evolving system that will adapt to the user change of requirements.

This is not an easy task given that aside social issues and ethics of such a device, the goals set up are not easily reachable. It is very difficult to test exhaustively a device capable of responding to unpredictable situations.

3 A self-adaptive multi-agent system

In order to design and build a self-adaptive system to assist seniors in their daily outdoor activities while respecting their requirement, we have involved users and their surroundings, human social science (HSS) and medical experts during the design and development process following a participative approach.

3.1 Theoretical foundation of our proposed approach

The computing environment where our system should be deployed is complex since it is *distributed*, includes a large number of entities and constraints (devices, sensors …) and *mobile* users and devices. It is *open* since components can enter and leave the system at any time.

To deal with distribution and openness constraints, we use a multi-agent approach. As for unpredictability and adaptation, our agents will follow a reinforcement learning approach taking into account the context.

Let us precise the interest of context and learning in our approach before detailing the proposed approach.

Need of context.

Our system should adapt to the changing contexts of the user and take into account his possible evolving requirements. In other words, we need to build a context-aware software that could self-adapt according to user habits, his dependency level, the changing environment and accessible devices. *The notion of Context* has been widely studied (see for example [14, 15]). In our case, we define the context not only by the

[2] Table legend: The "+ or ++" symbol shows how much the feature is taken into account by each system and shortcomings of each system are spotted with "- or --" symbol

user's location but also the correlation of several parameters such as: network connectivity, noise level, heartbeats rate, and every possible sequence of our system perceptions (events, environment ...). These parameters could be collected through smartphone sensors or connected wearable devices. As stated in [15], the context provides a meaning to a situation and help to choice the more relevant action to trigger. In our case, the context is the sequence of our system perceptions (events, environment ...) that justifies the birth of adequate actions.

Need of learning.

Giving the numerous and possibly unpredictable contexts that could exist in our complex environment, it is difficult to enumerate them a priori. One way to deal with this issue is to use a machine learning approach that enables to build systems that automatically improve their functioning with experiences and learn to adapt to new possible contexts. Machine learning is classified in three categories [16, 17] and in our work, we will use the same reasoning as in the reinforcement learning approach. It is inspired from the natural learning that learns from its actions and mistakes to produce better performance in the future, through rewards. Also, it is the only approach that doesn't require examples and deals with dynamicity and the unpredictability, required for our system. More precisely, our agents will automatically determine the ideal behavior for a specific context based on feedbacks from the environment, and they will keep on adapting their behavior through time. The goal of each agent is to maximize its total reward while being cooperative. We will detail the learning process and their interaction protocol in section 3.2.2.

Proposed Multi-agent Approach: AMAS

We use the Adaptive Multi-Agent Systems (AMAS) [13] approach that aims at solving problems in a dynamic non-linear environment by enabling agents to learn their cooperative behavior in order to make emerge the global function of the system. AMAS theory has presented encouraging results in several context aware application such as user monitoring system indoors [18] and boat behavior detection [19].

In AMAS theory, agents self-organize by cooperation since automatic creations, modifications, merging or deletions of agents are operated without external interventions. Applying this approach to the problem of context learning leads to a specific type of agents, called Context Agents. They are created at runtime and self-adapt on-the fly [20].

To be cooperative means to valorize the global goal over the personal goal. For example, if a context agent becomes useless, it will self-destroy for the benefit of the system. Agents face Non-Cooperative Situation (NCS). For example, a NCS is detected when two context agents propose two different actions in the same situation and the wrong recommendation was selected. In this case, a resolution to solve the NCS situation is conducted collectively by the agents of the system.

3.2 Elderly monitoring and assistance with AMAS: System architecture and self-adaptive feature

3.2.1 Architecture.

We implemented a multi-agent system that offers monitoring and analysis of deviation to detect disorientation situation and assist the user through notifications, alerts and to identify and recommend good practices. This system also keeps a margin of initiative to the user and allows the emergence of innovative processes (set of actions).

Our system contains three main types of agents which are: the percept Agent, The context Agent and the Effector Agent types. Figure 1 describes the architecture of our system and its interactions with the environment.

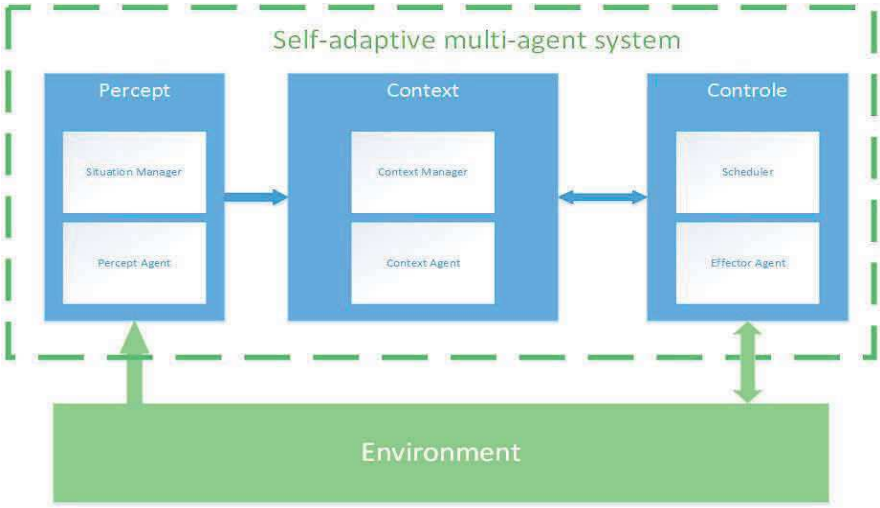

Fig. 1. The different modules of our system and its environment

We have three modules in our system that communicates and collaborates, each composed of a manager and a set of agents. At first, the system contains one situation manager, one context manager and one scheduler. Percept, context and effector agents are created progressively at run time.

The percept module detects, saves and interprets inputs received directly from the environment. Moreover, it guarantees the routing of significant signals to the context module.

The Context module contains context agents that reason on sent perceptions and propose appropriate actions.

The Control module gathers the scheduler (decision maker) and effector agents (performers). It receives context agents' action propositions, select the most appropriate one and execute it. It acts and receives feedbacks from the environment for learning purpose. By environment, we mean both the user behavior and the physical environment (park, ways, restaurants …) and stakeholders.

Testing our system with real users is risky and time consuming at this stage because they are vulnerable, not always available and can't behave normally when observed. So we decided to define a virtual environment perceived through distributed and unpredictable simulated events. These events are organized by scenario. Each scenario represents a succession of user daily possible outdoors activities and system actions that have a relation: done during a given period of time, made on the same place. Figure 2 presents the different components of our MAS (right hand side of the figure) and the composition of the environment. This later corresponds to the structure of the scenario (Left hand side of the figure).

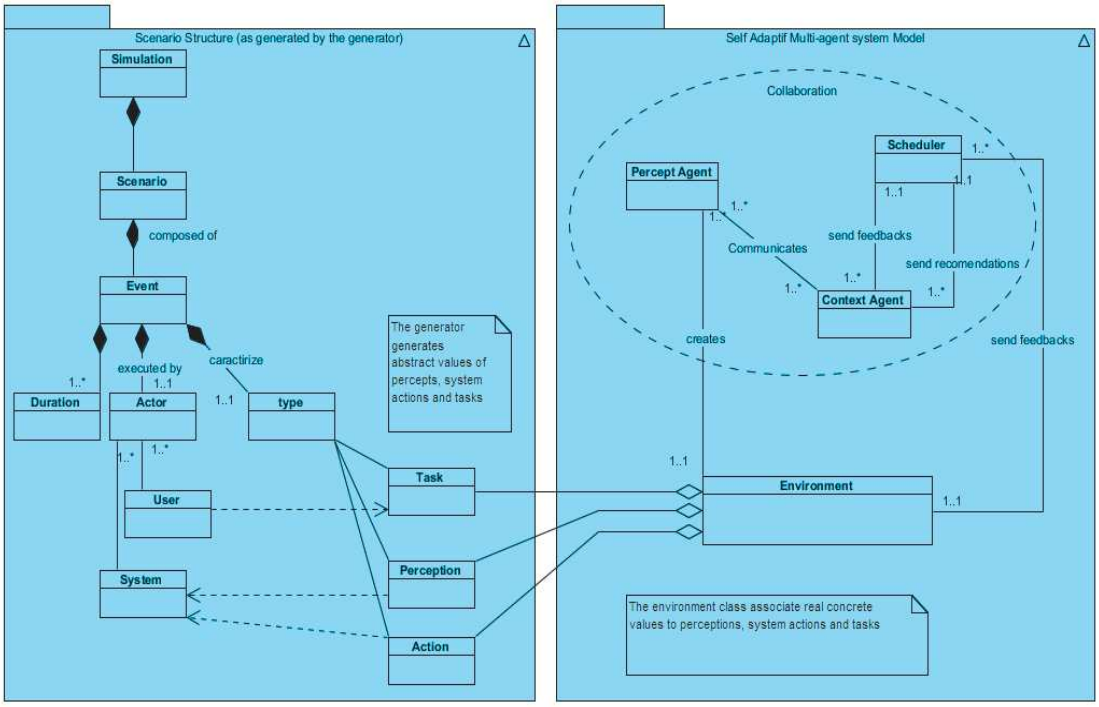

Fig. 2. Different components of the virtual environment and our system (UML diagram)

In the left hand side of Figure 2, we represent the structure of generated scenarios composing our virtual environment. A scenario is a sequence of events (e.g. tasks, perceptions and actions) made by an actor (e.g. user and system) for a defined duration. On the right hand side, we can clearly see interactions between the three types of agents (e.g. percept agents, context agents, effector agents).

3.2.2 The self-adaptive feature of our system.

In this section, we will specify the context agent and his cooperative behavior that represents the key component for the self-adaptive feature of our system.

A. The context Agent specification.

Firstly, each context agent has four attributes: *context*, *state*, *action*, *appreciation*.

Structure of a Context Agent
Context: *{<perception1, valueInf, valueSup>, …, <perceptionN, valueinf, valuesup>}*
State: *(created/validable/valid/selected/dead)*
Action: *action to be executed*
Appreciation: *confidence rate*
End Structure

Let us detail its attributes: Context agent associates to each perceptions an interval of the values [valueInf, valueSup], which we call validity ranges of each perception. The combination of all perceptions' validity ranges composes the *context*. *Appreciation* defines the confidence rate of each context agent.

The context agent state will be said ***valid*** if all received perceptions' values are in its defined the validity ranges. The context is said ***validable*** if the received perception values are in the validity ranges or in their borders.

In Figure 3 we present a context agent validity range for a GPS perception (just the latitude). The yellow interval represents the valid zone, the one with a pattern describes the validable zone. The white box represent the value of the perception in the current situation. The context agent is said valid if all perceptions of the current situation in the yellow zone.

Fig. 3. Context agent validity ranges example of the GPS latitude percept

Each context agent follows a life-cycle, at first it is created. Then it can propose an action if it is in a validable or a valid state. It can be selected by the scheduler and have its action executed. It can be in the dead state if its confidence is too low. The system faces four *Non-Cooperative Situations (NCS)*: ***NCS1***: System unproductivity (No context agent proposing the desired action), ***NCS2:*** A wrong action has been performed, ***NCS3***: Useless context Agent has been detected, ***NCS4***: Conflict between context agents.

To better understand those NCS, the sequence diagram in Figure 4 describes the resolution of NCS1.

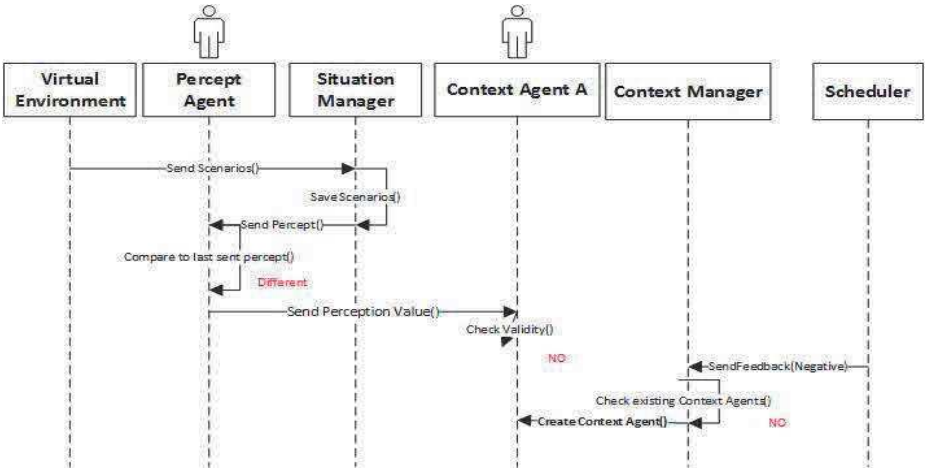

Fig. 4. System unproductivity resolving process (UML sequence diagram)

NCS1 will be managed by the context manager based on the scheduler feedbacks and will create a new context agent that provides the desired action for an interval that contains the perceptions at that time. We will detail behaviour algorithms and how it manages the NCS.

B. *Behavior algorithms ensuring the self-adaptive feature.*

The first algorithm describes the scheduler. It have two phases, the first on is to select the recommendation to be executed and then to send feedbacks according to the performed action and detects different types of NCS. However, it's not responsible of neither their occurrence nor their resolution and it just follows a couple of rules when facing multiples propositions:

— The most confident context agent is selected and its proposition is executed
— The valid context is automatically chosen over a validable context

So context agents should manage to make the context agent proposing the relevant action be valid and the most confident.

Algorithm 1 Scheduler Behaviour Algorithm

```
Require: Sch Scheduler, Propositions are received from each context agent valid or validable
Require: C_s Selected Context Agent, A_s Proposed action of the selected Context Agent C_s
Require: A_d desired Action, C_d Context Agent proposing A_d
 1: while Sch.ExistsPropositions() do                    ▷ Step1: Recommendation Selection and execution
 2:     if Sch.ExistsValidContext() then
 3:         C_s=Sch.selectMostConfident(validContexts);
 4:     else
 5:         C_s=Sch.selectMostConfident(allContexts);
 6:     end if
 7:     Sch.SendActionToExecute(A_s);
 8: end while
 9: while Sch.ExistsFeedbacks() do                       ▷ Step2: Recommendation Evaluation
10:     if (existsCorrectContext()) then
11:
12:         if ( A_s == A_d ) then                      ▷ Case of good recommendation
13:             Sch.sendFeedback(C_s,Positive);
14:         else                                        ▷ Case of failing recommendation
15:             Sch.sendFeedback(C_s,Negative,C_d);
16:         end if
17:     else
18:         if Sch.ExistsSelectedContext() then
19:             Sch.sendFeedback(C_s,Negative);
20:         else
21:             Sch.sendFeedback(Negative);
22:         end if
23:     end if
24: end while
```

There are different three of negative feedbacks, one when there is a context agent that proposes the desired action along received propositions "*sendFeedback(Cs,Negative,Cd)*" ,one where none of the proposed context agents proposes the desired action "*sendFeedback(Cs,Negative*)" and one that no context was proposed when an action was needed "*sendFeedback(Negative*)". In the two latter cases, the context manager will be involved in the resolution of those NCSs. The second algorithm explains the behavior of the context manager whose main role is to dispatch the received feedbacks to the corresponding context agents. Moreover, the context manager resoles NCS1 and NCS2 by creating a new context agent which explains how our system becomes populated with context agents in the first place.

Algorithm 2 Context Manager Behaviour Algorithm

```
Require: CM Context Manager, C_i Context Agent i, Feedbacks are received from the scheduler
while CM.ExistsFeedbacks() do
    CM.DispatchFeedbackToContextAgent();
    if ReceivesFeedback(C_i,Negatif) then
        CM.CreateNewContextAgent(Confidence_i); /*Confidence_newContext > Confidence_i*/
    end if
    if ReceivesFeedback(Negatif) then
        CM.CreateNewContextAgent();
    end if
end while
```

The latter algorithm describes the nominal behavior of the context agent (perceptions receptions, state update and recommendations) and the cooperative behavior when repairing NCS. The context agent ultimate goal is to always be in a cooperative situation.

Algorithm 3 Context Agent Behaviour Algorithm

```
Require: C_i Context Agent i, C_n Context Agent n in conflict with C_i,
Require: Perceptions received from each percept agent,Feedbacks received from the scheduler
 1: while true do                                          ▷ Step1: Recommendation according to received perceptions
 2:     C_i.ReceivesPerceptions();
 3:     C_i.UpdateState()
 4:     if (C_i.isValid() or C_i.isValidable()) then
 5:         C_i.proposeRecommendation (Action_i;Confidence_i;State_i)
 6:     end if
 7:     while C_i.ExistsReceivedFeedbacks() do                          ▷ Step2: Learning from received feedbacks
 8:         if (C_i.ReceivesFeedback(C_i,Positive)) then  ▷ Dealing with positive feedback in case of good recommendation in a
    previous step
 9:             if ( C_i.isValid()) then
10:                 C_i.IncreaseConfidence ();
11:             else
12:                 if C_i.isValidable then
13:                     C_i.IncreaseValidityRange ();
14:                 end if
15:             end if                    ▷ Dealing with negative feedback in case of failing recommendation in a previous step
16:         else                                          ▷ Case of failing recommendation from a valid context
17:             if ( C_i.isValid()) then
18:                 C_i.decreaseValidityRange ();
19:                 C_i.UpdateState() ;
20:
21:                 if C_i.isValid then
22:                     C_i.decreaseConfidence ();
23:                 end if
24:                 if (C_i.checkConfidence () <=0) then
25:                     C_i.Self-destruct (); /*NCS3 resolving*/
26:                     EXIT;
27:                 end if
28:             end if          ▷ Case of failing recommendation while an other context agent proposed the desired action
29:             if (C_i.ReceivesFeedback(C_i,Negatif,C_n)) /*NCS4 resolving*/ then
30:
31:                 if (C_i.isValid() AND C_n.isValid()) then
32:                     C_i.SendIncreaseConfidence (C_n,Confidence_i); /*Confidence_n > Confidence_i*/
33:                 end if
34:                 if (C_i.isValid() AND C_n.isValidable()) then
35:                     C_i.SendIncreaseValidityRange(C_n);
36:                     C_i.SendIncreaseConfidence (C_n,Confidence_i); /*Confidence_n > Confidence_i*/
37:                 end if
38:                 if (C_i.isValidable() AND C_n.isValidable()) then
39:                     C_i.SendIncreaseValidityRangetoValid (C_n);/*Until C_n becomes valid*/
40:                 end if
41:             end if
42:         end if
43:     end while                                                   ▷ Step3: Learning from received messages
44:     while C_i.ExistsReceivedMessages() do
45:         if (ReceivesIncreaseValidityRange()) then
46:             C_i.IncreaseValidityRange ();
47:         end if
48:         if (ReceivesIncreaseValidityRangetoValid()) then
49:             while ( C_i.isValidable()) do
50:                 C_i.IncreaseValidityRange ();
51:             end while
52:         end if
53:         if (ReceivesIncreaseConfidence()) then
54:             C_i.IncreaseConfidence ();
55:         end if
56:     end while
57: end while
```

We have two main hypothesis to ensure cooperative behavior between context agents while dealing with NCSs which are: a ***validable*** context agent does not update its confidence, but updates its validity ranges and a ***valid*** context agent can update both its confidence and its validity ranges.

The context agent behavior algorithm consists on an infinite loop made of three steps. First, it will start its nominal behavior by updating his state according to received perceptions and possibly propose a recommendation (line 2 to 6). Second, the context agent starts a loop to treat received feedbacks from the scheduler in order to learn the user's habit and adapt to his requirement. There are two types of possible feedbacks, positive corresponding to correct recommendation leading the concerned agent to increase its confidence or its validity ranges (line 8 to 15). Negative feedbacks are reactions to failing recommendations which trigger NCS (line 16 to 42). Third, the context agent cooperates with other context agent by executing sent suggestions (line 44 to 56). Feedbacks being sent by the scheduler, the second algorithm details the behavior of the scheduler.

The self-adaptation feature is based on cooperation that reduces NCS and therefore maximizes good recommendations by means of self-organization, consisting in:

1. ***Creating*** new context agents in case of system unproductivity
2. ***Modifying*** existing ones through self-adjustments on the confidence and validity range
3. ***Reducing*** the number of context agents by self-destruction of useless context agent and by merging context agents when they propose the same action for the same situation.

Now that we have an overview on our system process, in the following section we will describe our application and different primary results

4 System Evaluation

As explained in section 3, our system was tested through scenarios generated automatically. The generator simulates a user's activities through a combination of perception, action and tasks. Tasks correspond to user's activities such as "going shopping", "walking", doctor's visit.

Figure 5 illustrates the configuration panel of our generator in which we define various parameters (i.e. number of generated days, number of actions and perceptions in scenario…) of the generator corresponding to user's characteristics (his autonomy level, system acceptance level, …).

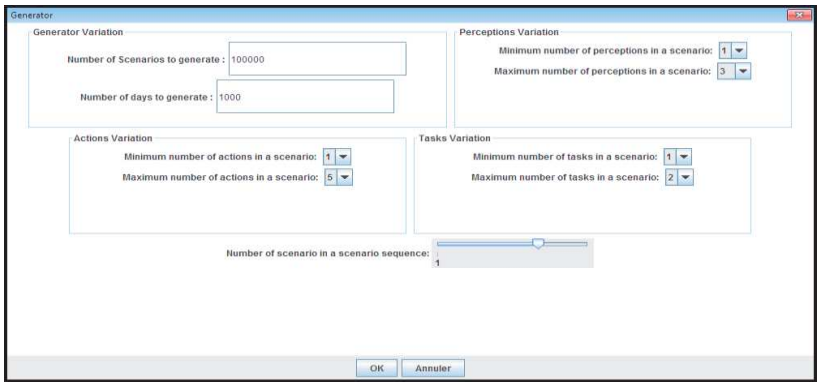

Fig. 5. Configuration panel of the scenarios generator

We generated a set of 100000 scenarios and created a simulation of 1000 days for a single user profile. We observe the behavior of our system to test self-adaptation and self-organization by checking when the system will be able to propose the action required. The generator goal is to create enough scenarios for a given period of time (e.g. a day, a week…) in order to evaluate the learning capabilities of our system.

Fig. 6 describes the system considering four common behaviors of learning process:

- **Right True:** Is when the system proposes the right action
- **Wrong True:** Is when the system proposes an action that should not have been proposed
- **Right False:** Is when the system doesn't proposes any actions and it was the right
- **Wrong False:** Is when the system doesn't proposes any actions but it should have.

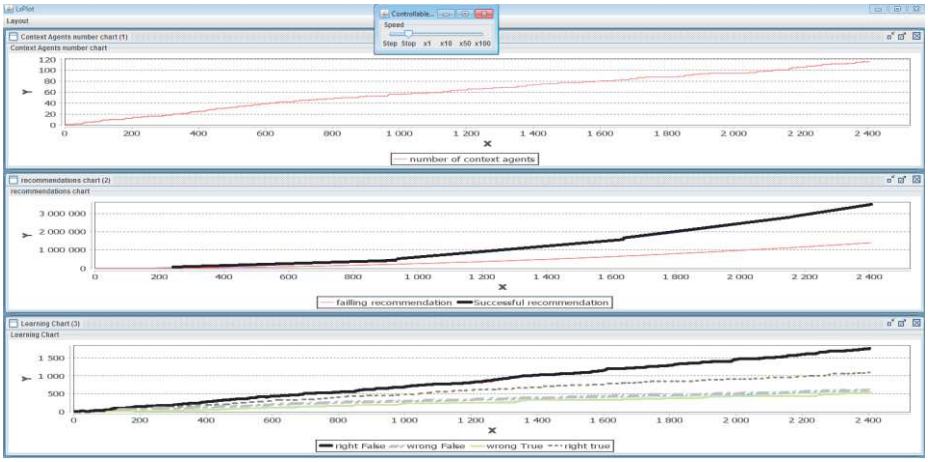

Fig. 6. Variation of right and wrong predictions made by the MAS

These three graphs of Figure 6 are generated automatically throughout the execution of our system. The horizontal axes represent the number of steps (corresponding to the number of received events). We can clearly see that all graphs start at zero since no recommendation can be made when no context agent is created yet. The first graph shows how context agent number evolves with events. We can notice that at the end we have 120 context agents for 2400 events. These numbers show that our system doesn't create an agent for each event but a same agent is able to manage several correlated events (corresponding to a situation) thanks to its adaptation.

In the two last graphs of Figure 6, we can see that our system performs well: Correct recommendations are almost three times greater than the failing ones. The fact that the number of errors keeps increasing comes from the unpredictably of our daily life scenarios. The last graph shows how the correct recommendations (right false and right true) suggesting the right behavior are three times greater than wrong recommendations (wrong false, wrong true).

5 Conclusion and Perspectives

In this paper, we have shown that current Ambient Assisted Living systems (AALs) are facing strong limitations and lacking adaptation since they do not take into account environment and users' requirements evolutions.
We have defined the architecture of a self-adaptive multi-agent system that can assist older adults in their outdoor activities and handle unexpected situations. Our system has the capacity to learn user habits from its perceptions and improve its recommendations thanks to feedbacks provided by stakeholders. A simulator has been implemented and experimentations have shown the feasibility of our system and the efficiency of our learning process.
However, it is clear that there are still a number of issues that require further investigations. The simulator can be improved to take into account the user planning in its recommendations in order to provide recommendation compliant with the user planning. Moreover, we intend to implement a prototype on smartphones and test it with real users.

6 Acknowledgements

We would like to acknowledge both the Midi-Pyrénées Region and the IRIT laboratory that funded the Compagnon project. We also would like to thank Marie-Pierre Gleizes from IRIT Laboratory and and Alice Rouyer from LISST laboratory for helping in specifying the system developed.

7 References

1. United Nations: Population Division. World Population Ageing 2013. Department of Economic and Social Affairs.
2. Fondation of France. 2010-2014: the French increasingly only (in French).
3. P. Rashidi and D. J. Cook, ‘‘Keeping the resident in the loop: Adapting the smart home to the user,’’ IEEE Trans. Syst. Man Cybern. A, Syst. Humans, vol. 39, no. 5, pp. 949–959.
4. Urs Anliker and al.: AMON: a wearable multiparameter medical monitoring and alert system. IEEE Trans. Information Technology in Biomedicine 8(4): 415-427 (2004)
5. 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 Trans. Inf. Technol. Biomed., vol. 8, no. 3, pp. 238–247, Sep. 2004.
6. F. Sadri, ‘‘Ambient intelligence: A survey,’’ ACM Comput. Survey vol. 43, no. 4, pp. 36:1–36:66, Oct. 2011.
7. R. Paradiso, G. Loriga, and N. Taccini, ‘‘A wearable health care system based on knitted integrated sensors,’’ IEEE Trans. Inf. Technol. Biomed., vol. 9, no. 3, pp. 337–344, Sep. 2005.
8. J. Ninomiya, K. Murayama, T. Yamaoka, K. Sakai, “Next-generation ubiquitous device for new mobility society: next generation Cane”, FUJITSU Sci.Tech. J. Vol.51, P8-13, October 2015.
9. R. Levinson, ‘‘the planning and execution assistant and trainer (PEAT),’’ J. Head Trauma Rehabil., vol. 12, no. 2, pp. 85–91, 1997.
10. M. E. Pollack, L. Brown, D. Colbry, C. E. McCarthy, C. Orosz, B. Peintner, S. Ramakrishnan, and I. Tsamardinos, ‘‘Autominder: An intelligent cognitive orthotic system for people with memory impairment,’’ Robot. Autonom. Syst., vol. 44, no. 3-4, pp. 273–282, 2003.
11. J. Wan, C. Byrne, G. M. O’Hare, and M. J. O’Grady, ‘‘Orange alerts: Lessons from an outdoor case study,’’ in Proc. 5th Int. Conf. Perv. Comput. Technol. Healthcare, 2011.
12. S. Fudickar and B. Schnor, ‘‘Kopal a mobile orientation system for dementia patients Intell. Interactive Assistance Mobile Multimedia Comput. vol. 53, pp. 109–118, 2009.
13. Davy Capera, Jean-Pierre Georg´e, M-P Gleizes, and Pierre Glize, The amas theory for complex problem solving based on self-organizing cooperative agents, Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003. WET ICE 2003.
14. M. Bazire, P.Brézillon. “Understanding context before using it. In Modeling and Using Context”, In Modeling and Using Context, Vol.3554 p 29-40, 2005.
15. Salembier P, Dugdale J, Frejus Myriam “A descriptive model of contextual activities for the design of domestic situations” In European Conference on Cognitive Ergonomics 2009
16. Mitchell, T.M. Machine Learning. United State: Mst utilize in cGraw Hill, 1997, 400p. {ISBN 0-07-042807-7}
17. Mitchell, T.M. The discipline of machine learning, volume 9. Carnegie Mellon University, school of computer science, Machine learning department (2006).
18. Valérian Guivarch, Valérie Camps, André Péninou, Pierre Glize, Self-Adaptation of a Learnt Behaviour by Detecting and by Managing User's Implicit Contradictions. WI-IAT (3) 2014: 24-31
19. Nicolas Brax, Eric Andonoff, Marie Pierre Gleizes. A Self-adaptive Multi-Agent System for Abnormal Behavior Detection in Maritime Surveillance. KES-AMSTA 2012: 174-185
20. Jérémy Boes, Julien Nigon, Nicolas Verstaevel, Marie-Pierre Gleizes, Frédéric Migeon. The Self-Adaptive Context Learning Pattern: Overview and Proposal. In : International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT 2015)