Prototyping Ubiquitous Multi-Agent Systems: A Generic Domain Approach with Jason

Carlos Eduardo Pantoja[1,2(✉)] and José Viterbo[2]

[1] Centro Federal de Educação Tecnológica (CEFET/RJ),
Av. Maracanã 229, Tijuca, RJ, Brazil
pantoja@cefet-rj.br
[2] Universidade Federal Fluminense (UFF),
Av. Gal. Milton Tavares de Souza, São Domingos, Niterói, RJ, Brazil
viterbo@ic.uff.br

Abstract. This work presents a generic domain approach for programming ubiquitous Multi-Agent Systems using Jason framework and ARGO in electronic prototypes. The approach aims to provide a ready-to-use platform that is heterogeneous and independent from the hardware selected to be used in several domains. In order to validate the approach, two examples in distinct domains and based on case studies were implemented, prototyped and discussed. The results show that the approach is adequate to develop such kind of systems.

1 Introduction

Agents are intelligent and autonomous entities that can be implemented in both hardware and software. They are proactive and able to communicate to each other in organizations. A Multi-Agent Systems (MAS) is a system composed of agents acting upon an environment to achieve mutual or conflicting goals [7]. Ubiquitous Systems and Ambient Intelligence (AmI) are electronic ambient that aids humans in common or complex situations in a pervasive way aided by intelligent systems. Accordingly to [2], the characteristics of the MAS approach can be exploited for the development of such kind of systems.

Applying the MAS approach in prototyping is not a simple issue since several limitations can occur when integrating hardware devices and the software responsible for the reasoning. One of these limitations provides tied solutions integrating MAS platforms and prototypes where the software is tied to the hardware technology employed, such as [3]. When it happens, the software is coupled to the hardware and it is not possible to change it for another one from a different type without rework. Besides, in most of the cases, it is only possible to use one kind of controller. Another limitation is that several works, such as [5], provide solutions where the software is strictly developed to one specific domain, do not offering generic constructions for programming robotic agents.

ARGO[1] [4] is a Jason's customized architecture that tries to facilitate the development of MAS for robotic platforms by allowing agents to control

[1] http://argo-for-jason.sourceforge.net.

DOI: 10.1007/978-3-319-59930-4_34

heterogeneous microcontrollers. Jason [1] is a well-known agent-oriented programming language. We assert that Jason and ARGO agents can be employed in the development of ubiquitous Multi-Agent Systems (uMAS) in a generic domain approach, uncoupled and independent from the type of controllers used.

2 Main Purpose

The main objective of this work is to provide an easy way of prototyping uMAS using BDI agents in a generic domain approach without concerning with the hardware technology employed in the prototype. The approach aims to be used in ubiquitous prototyping using Jason and ARGO independently of the domain chosen for the development of uMAS. In other words, in combining Jason and ARGO agents, it is possible to create agents capable of controlling hardware devices by means of microcontrollers. The designer of the prototype just has to concern with the agents' programming and the functionalities of the devices. All infrastructure (e.g. middleware) for transferring the perceptions from hardware to agent's knowledge base is inserted into the reasoning cycle of ARGO agents.

ARGO counts with a mechanism capable of processing sensorial information as perceptions directly into the belief base of specific kind of agent and it allows hardware controlling without concerning with the technology. Several internal actions are available to program some specific behaviors of an ARGO agent, which is able to decide: whether or not perceive the environment using its sensors; to act upon the environment using its actuators; the time interval between each environment sensing; if it is necessary to filter information for the sake of performance and; select which device to control in a specific moment. All of these characteristics can be exploited at runtime, offering a dynamic solution for programming and prototyping ubiquitous systems based on the agent approach.

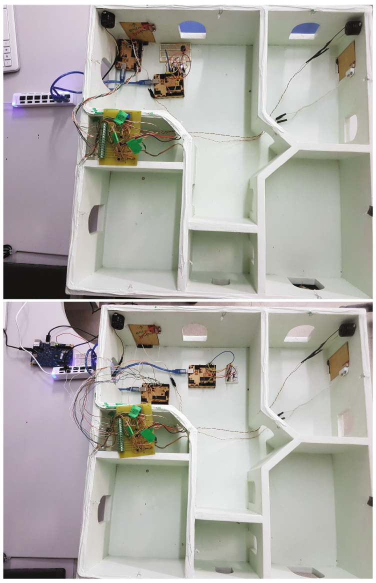
In order to clarify the proposed approach, two examples in complete distinct domains are shown: in the first example it will be used a smart home prototype controlled by several ATMEGA controllers in a situation with a hearing impaired person living at a house and the second example will present an autonomous vehicle capable of identifying a wall and stop based on its sensors. Complementing the approach, the controllers from the first example will be changed for a different type and the example will be executed again.

3 Demonstration

This section shows the examples[2] using the generic domain approach employing Jason along with ARGO in two practical examples in distinct domains. The first example, based on [6], presents a smart home where a hearing impaired person is living in. In this smart home, if someone presses the door bell in front of the main door, the hearing impaired is not able to hear it and the smart home warns the person blinking the lights of the house. A prototype represents the smart

[2] https://youtu.be/9osZIMKvftA and https://youtu.be/0QzXHwzLSj8.

home using two ATMEGA328 (Arduino): one for controlling the bell and another one responsible for the lights of the house. The MAS responsible for controlling the prototype has an ARGO agent, which is responsible for identifying if exists somebody at the front door and another one responsible for blinking the lights of the house. For instance, agent Kate is responsible for the bell and agent Bob is responsible for the lights. Figure 1 depicts the prototype, Kate and Bob.

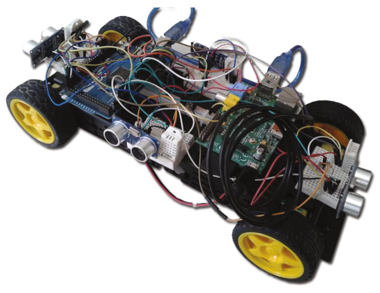


```
!start.

+!start : true <-
        .port(COM11);
        .percepts(open);
        .percepts(block);
        !ligar_campainha.

+!ligar_campainha: campainha(Value) & Value == 1 <-
        .print("Ligando campainha");
        .act(ligarCampainha);
        .send(bob,achieve,turnOnLights);
        !start.

+!ligar_campainha: campainha(Value) & Value == 0 <-
        .print("Ningu m na porta");
        .act(desligarCampainha);
        .send(bob,achieve,turnOffLights);
        !start.

-!ligar_campainha <-
        .print("A campainha est  desligada!");
        !start.

value(40).
!config.
```

```
!start.

+!start : true <-
        .port(COM5).

+!turnOnLights <-
        .act(ligaTodasLuzes).

+!turnOffLights <-
        .act(desligaTodasLuzes).
```

Fig. 1. The prototype employing Arduino (top left); agent Kate (top right); the prototype employing Galileo (bottom left) and; agent Bob (bottom right).

The second example is an autonomous unmanned vehicle, which is able to stop before it crashes into a wall. The vehicle is a 4WD platform with 4 distance sensors on each side of the prototype plugged in an Arduino board and an



```
value(40).
!config.

+!config: true <-
        .port(COM5);
        .limit(20);
        .filter(byValue);
        .percepts(open);
        !start.

+!start : true <-
        .act(front);
        +status(front);
        !moving.

+!moving: dist(f, X) &
                value(J) & X>J &
                status(front) <-
        .act(front);
        !moving.

+!moving: dist(f, X) &
                value(J) &
                X<=J & status(front) <-
        .act(stop).

-!moving <-
        !!moving.
```

Fig. 2. The autonomous vehicle (left) and the agent code (right).

intelligent agent is responsible for perceiving the environment and to move until it perceives the wall. Figure 2 depicts the vehicle and the agent code. Finally, the first example was repeated using an Intel Galileo board instead of the Arduino board for agent Kate (Fig. 1) without modifying the agent's code. Bob still controls the other Arduino board. It is possible to see that the demonstration combines two different controllers in the same prototype in a heterogeneous approach. The result shows no difference in both executions.

4 Conclusions

This paper presented a generic domain approach using Jason and ARGO for prototyping uMAS and two practical examples in different domains were presented. The proposed approach is generic since it is possible to program agents for different domains without being aware or bonded to the type of controller employed. Because the software layer is independent of the controller employed it is allowed to use different types of controllers (even together or separated). Besides, they can be replaced without changing the MAS. For adding new controllers, they must comply with the protocol used in ARGO which uses serial ports as the communication channel between low-level layers and the software. For instance ARGO accepts ATMEGA and PIC controllers. These characteristics of the approach can be exploited to develop ubiquitous systems, where heterogeneous hardware are employed and intelligent agents can be used for providing an autonomous behavior of the system.

References

1. Bordini, R.H., Hübner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak using Jason. Wiley, Chichester (2007)
2. Chaouche, A.C., Seghrouchni, A.E.F., Ilié, J.M., Saïdouni, D.E.: A higher-order agent model with contextual planning management for ambient systems. In: Kowalczyk, R., Nguyen, N. (eds.) Transactions on Computational Collective Intelligence XVI. LNCS, vol. 8780, pp. 146–169. Springer, Heidelberg (2014). doi:10.1007/978-3-662-44871-7_6
3. Cook, D.J., Youngblood, G.M., Heierman, E.O., Gopalratnam, K., Rao, S., Litvin, A., Khawaja, F.: Mavhome: an agent-based smart home. PerCom. **3**, 521–524 (2003)
4. Pantoja, C.E., Stabile, M.F., Lazarin, N.M., Sichman, J.S.: ARGO: an extended jason architecture that facilitates embedded robotic agents programming. In: Baldoni, M., Müller, J.P., Nunes, I., Zalila-Wenkstern, R. (eds.) EMAS 2016. LNCS (LNAI), vol. 10093, pp. 136–155. Springer, Cham (2016). doi:10.1007/978-3-319-50983-9_8
5. Sun, Q., Yu, W., Kochurov, N., Hao, Q., Hu, F.: A multi-agent-based intelligent sensor and actuator network design for smart house and home automation. J. Sens. Actuator Netw. **2**(3), 557–588 (2013)
6. Villarrubia, G., De Paz, J.F., Bajo, J., Corchado, J.M.: Ambient agents: embedded agents for remote control and monitoring using the pangea platform. Sensors **14**(8), 13955–13979 (2014)
7. Wooldridge, M.: An Introduction to MultiAgent Systems. Wiley, New York (2009)