A Web-based agent-oriented approach to address heterogeneity in cooperative embedded systems

Jean-Paul Jamont[1], Lionel Médini[2], and Michael Mrissa[2]

[1] Université de Grenoble Alpes, Laboratoire LCIS, Valence, France
jean-paul.jamont@lcis.grenoble-inp.fr
[2] Université de Lyon, LIRIS - UFR FST, CNRS, 43, boulevard du 11 novembre 1918, 69622 Villeurbanne cedex
{michael.mrissa,lionel.medini}@univ-lyon1.fr

Abstract. This paper presents a MAS-oriented approach to enable emergence and execution of complex functionality among a fleet of heterogeneous connected objects. It relies on the Web of Things paradigm, in which such objects communicate using Web standards. In order to homogenize the objects and extend their capabilities, our approach is based on agents that can be deployed either on objects or in the cloud. Such agents can embody the object behaviors and perform negotiation to achieve collaborative functionalities.

Keywords: avatars, multi-robot systems, embedded systems, Web of things

1 Introduction

Developments in wireless technologies have strongly impacted embedded systems. Embedded systems have become sets of small interconnected heterogeneous devices strongly related to their environment. Devices need to cooperate when they do not have certain necessary skills or knowledge to accomplish their individual goals but also to meet objectives of the overall system. Such systems can be then observed at two levels: an individual level (the device layer) and a social level (the global system).

Designing a cooperative embedded systems require to meet numerous challenges like heterogeneity, scalability, openness etc. Heterogeneity means the diversity of the devices in terms of platform, connectivity, operating systems etc. A device can be a complex robots reasoning with abstract symbolic representations of its own environment, a mobile phone, a hard resources constrained sensors or chipless things like a RFID tagged books. The heterogeneity management, which is the ability to describe data and services exposed by the devices, is a hard point.

In such applications, multiagent systems (MAS) are used because they offer many relevant models to implement collective and adaptive behaviors. The contribution of this paper is a Web-based agent-oriented approach to address the previously exposed challenges. This approach is based on the concept of avatar which is a Web abstraction of a real world device. According to a given Avatar architecture, the result is an open, generic and Web-based solution to deliver high-level, user-understandable functionality, while interacting with a set of various physical objects, enabling cooperation between them.

Section 2 the background of this work i.e. works integration of heterogeneous agents in a same MAS using Web. Section 3 describes our decentralized approach and defines the notion of avatar, before illustrating the interest of our approach in the context of the WoT with a motivating scenario. Before to conclude, we introduce in Section 4 the internal architecture of avatar agents.

2 Background

The integration of heterogeneous agents in a same MAS is the subject of numerous studies since the late 90s. FIPA[3] has proposed many specifications to address the problem as the standardization of interactions. FIPA proposes some standard to treat heterogeneity as a communication problem(FIPA-ACL), from an architectures point of view (Agent Abstract Architecture), as a middleware integration problem etc.

The work of the most popular standardization in IT and most universally prevalent are those of the Web. It is therefore natural to focus on these standards to ensure interoperability of heterogeneous agents.

Works at the intersection of MAS and WS can be divided into three categories [1]. The first one concerns the use of a MAS as a mediator in the WS functional model like in [2–5]. The last categories use WS to make available the MAS through the Web according two different approaches :

– An integrated approach: WS are developed following an agent model to perform complex tasks such as management commercial transactions or interactions [6, 7] or, on the other hand, WS are accessible through a multiagent framework [8, 9].
– A decoupled approach: Starting from a given MAS, a WS layer enable agents to interact together according to Web interfaces [1, 10–12].

3 Our Approach

We provide a decentralized approach that enables proactive and cooperative intelligence through the concept of avatar. We then explain how avatars enable collective intelligence. We then illustrate our approach through.

Extending Objects with Avatars To introduce the concept of avatar, we need to define *object* and *proxy*.

Objects Applications developed for the WoT put into play many heterogeneous. These objects are hardware or software entities. Establishing a relationship between these different types of entities creates the added-value the

[3] Foundation for Intelligent Physical Agents - `http://www.fipa.org/`

WoT brings to the user community. We can consider an object as a 4-tuple $O =< G, B, K, S >$, where $G = \{G_1, G_2, ..., G_n\}$ is its set of goals (what the object want fulfill), $K = \{K_1, K_2, ..., K_n\}$ its knowledge i.e. the set of re-usable abstractions about its environment and about other devices, $S = \{S_1, S_2, ..., S_n\}$ is the set of device's features and $B = \{B_1, B_2, ..., B_n\}$ its behavior i.e. a set of rules to define the logic of actions/reactions in response to internal/external stimuli.

We identify three types of physical objects:

1. Complex Objects: These objects provide software services and embed a Web server that offers service interfaces. It is then often trivial to link these objects together or with other software services. The only difference with traditional WS is the connection to physical objects that interact with their environment.
2. Lightweight Objects: These objects cannot embed Web servers due to restricted computing capacity (low memory, limited energy resources, restricted bandwidth) but it is often easy to link them to proxies. A proxy can embed a Web server. The (object,proxy) couple can be seen as a complex object that is physically distributed.
3. Bare Objects : These objects are passive objects that can be detected such as pallets with RFID (Radio Frequency IDentifier) tags. When such an object is in the range of a RFID reader, the reader receives a byte array. A logical link can then be established between the physical object and the byte array.

Proxy In our context, a proxy is a projection of a physical object into the Web. Concretely, it is a Web intermediary for requests from clients seeking resources from other objects. We can define a proxy as a 2-tuple $P =< Kp, Sp >$ with $Kp \subseteq K$ and $Sp \subseteq S$. The selection of what is exposed depends of different strategies (energy management , privacy preserving, etc.).

Avatar Our approach consists in extending objects with a virtual representation on the Web (Fig. 1). We call such an representation an avatar. Avatars are not simple proxies. An avatar is an autonomous entity (i.e. an agent) which has it own 4-tuple $A =< Ga, Ba, Ka, Sa >$ with $G \subset Ga$, $B \subset Ba$, $K \subset Ka$ and $S \subset S_a$. The increase of its knowledge and its skills comes from (1) the Web which is the avatar environment (so an avatar can access to the Web of data and WS) and (2) others avatars. Through their avatars, physical objects can be in interaction and particularly in cooperation.

Considering a bare object "pot of yogurt", its avatar should be able to identify its location and state by accessing and analyzing the history of crossed RFID readers. The avatar can therefore try to interact with another avatar that extends a temperature sensor at the same place, to determine whether or not the yogurt pot could be degraded. This example shows that more than having extended the capacities of things, we have, individually and globally, enriched their behavior. Interactions between avatars can lead to exhibit collective behaviors. We can concretely reuse all the works exhibited in MAS in the context of avatars : an avatar also is an autonomous agent.

In order to plan how to achieve collaborative functions, avatars must negotiate with one another. This requires both a negotiation model and a communication protocol. These well-known models of the multiagent community are not developed here ([13, 14]).

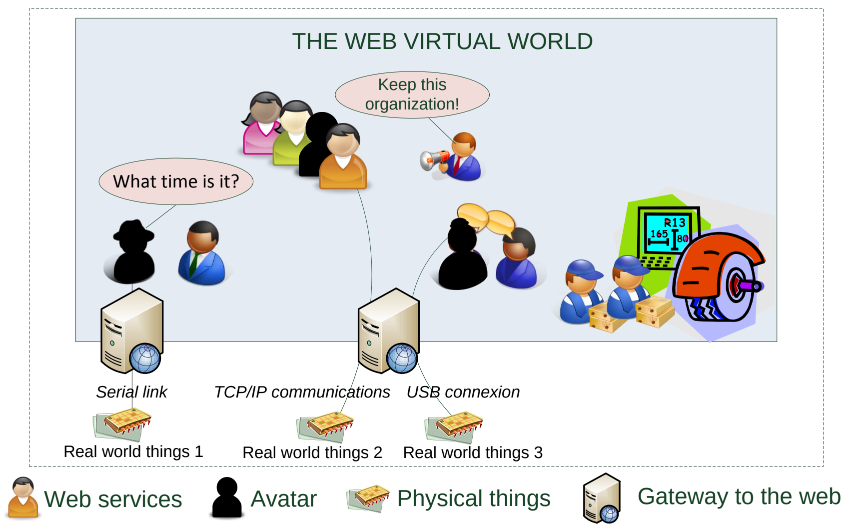

Fig. 1. Using avatars in the Web

Application of our approach To illustrate our approach, we consider the following production chain scenario (inspired by [15]) (fig. 2).

We have to manage the flow of yogurt produced in a firm. These goods are packed in boxes. Boxes are stacked on RFID tagged plastic pallet. When a pallet is filled to its maximum capacity, a human operator places the pallet in the production deposit area. The pallets will now be handled by mobile robots. Pallets of yogurt have to be stored in any storage inside the warehouse. We assume that a yogurt should never stay more than 10 minutes in a place where the temperature is higher than 10^oC, otherwise it becomes inappropriate for human consumption and an alternative scenario must be considered (i.e. recycling process or destruction).

Our approach leads to build WoT adaptive applications. We illustrate our approach in the context of the previously i scenario.

1. Production chain produces a pallet of yogurt: a pallet leaves the production line with a load of yogurt. It is deposited in the *production deposit area.*
   (a) the pallet RFID tag is read by the deposit area RFID reader.
   (b) the *avatar creation service* queries the *production server* to know what is the associated thing.

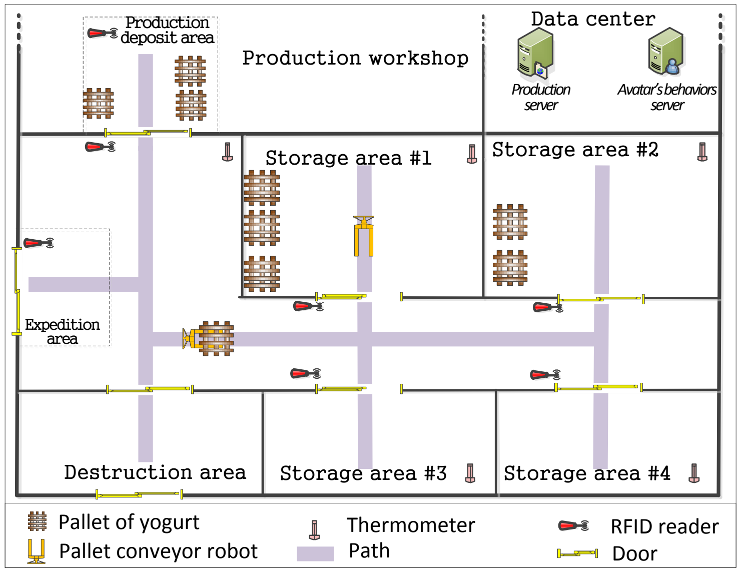

Fig. 2. Illustration of the scenario

  (c) the *production server* informs it is a pallet of yogurt.
  (d) the *avatar builder service* downloads from the *code repository* the behavior associated to an abstraction of pallet of yogurt and generates an instance of avatar for the yogurt pallet.
  (e) the *avatar builder service* creates an *avatar* (`rfid_tag="#50 41 4c 4c 45 54 31 32 33 34 35 36 37 38", type="#Pallet", parameter_list=("#Yogurt") )`.
  (f) the association between the avatar's url (`#yogurt_pallet_1`) is memorized in the *context server*.
2. The pallet is deposited in *Storage area 1*.
  (a) The tag is read by the *RFID reader* device associated to *storage area1*.
  (b) Treatment of the tag by the RFID reader (`#RFID_reader_1`) avatar associated to this room.
  (c) The contextual event `You enter in #StorageArea_1` is send by the avatar `#RFID_reader_1` to the avatar `#yogurt_pallet_1`.
3. Avatar `#yogurt_pallet_1` has an introspection : it inspect its behaviour to find rules linked to a room modification.
  (a) He searches rules linked to the event `room modification`
    i. He searches rules linked to the instance `#StorageArea_1`
    ii. No rules are found then he search rules linked to concept behaviour introspection `#Area`
    iii. No rules are found then he search rules linked to super-concepts of `#Area` (here it is #Place).
    iv. A rule is founded : `if temperature(#Place) > 10`^{o} `then "find a solution to be sure the storage period at this temperature is under 10 minutes"`.

(b) Application of rules:
   i. The `#yogurt_pallet_1` avatar have a behaviour introspection to try to find functionality `getTemperature()`
   ii. The function is not found then the avatar search another one which can provide functionality `getTemperature(p : place = #StorageArea_1)`.
   iii. The thermometer avatar associated to `#StorageArea_1` inform it is able to provide this functionality.
   iv. The pallet of yogurt avatar request functionality `getTemperature(#StorageArea_1)` of `#temperature_sensor_StorageArea_1`
   v. The result is 12^oC then the pallet of yogurt avatar applies rule `search a solution`. The 1st solution is `self-move : move(p : pallet = #Pallet_1)`
   vi. Introspection to find functionality `move(p : pallet = #Pallet_1)`.
   vii. The functionality is not found then the avatar search another one which can provide functionality `move(thing : pallet, from : place, to : place )`
   viii. Avatars `#Pallet_conveyor_1` and `#Pallet_conveyor_2` inform they can provide this functionality. `#Pallet_conveyor_3` has not answer because if is under maintenance.
   ix. Avatars `#Yogurt_pallet_1`, `#Pallet_conveyor_1` and `#Pallet_conveyor_2` negotiate together : `#Pallet_conveyor_1` proposes the best quality of service because it can supply the service earlier. In fact, it is free of load contrary to `#Pallet_conveyor_2`.
   x. `#Yogurt_pallet_1` requires this functionality of `#Pallet_conveyor_1`
   xi. `#Pallet_conveyor_1` accepts.
4. Transportation of `#Yogurt_pallet_1`
   (a) ...

This scenario shows how objects can benefit from a Web-based and agent-oriented approach that extends the interactions possibilities and enhance their life cycle, allowing better and natural integration into their environment.

4 Avatar architecture

The avatar architecture is referred to as "WoT Runtime Environment", as depicted in figure 3. It is composed of a framework in which are plugged a set of components called managers, which are grouped in modules. Each manager takes charge of a specific concern and interacts with other components using a specific API. The core module of the avatar architecture allows deploying the components into this framework and furnishes low-level components that are required by several managers, to perform caching and reasoning tasks. The applicative code that implements the functionalities that the objects can realize is dynamically deployed in a second internal framework, called the "WoT Application Container". Each logical component of the avatar architecture can be executed either on the object or in the cloud.

The "Collaborative Agent Manager" is responsible of the MAS aspects of the avatar architecture and embeds the communication and negotiation models. It queries the "Collaborative Functionality Discovery Manager" to retrieve a list of potential functionalities that cannot be locally achieved by an object, but require collaboration. It interacts with other avatars through the Web service module: the "WoT Application Server" component exposes both the available functionalities on each object and the negotiation protocol as REST resources; each avatar thus uses the HTTP client module to query these resources in the WoT infrastructure.

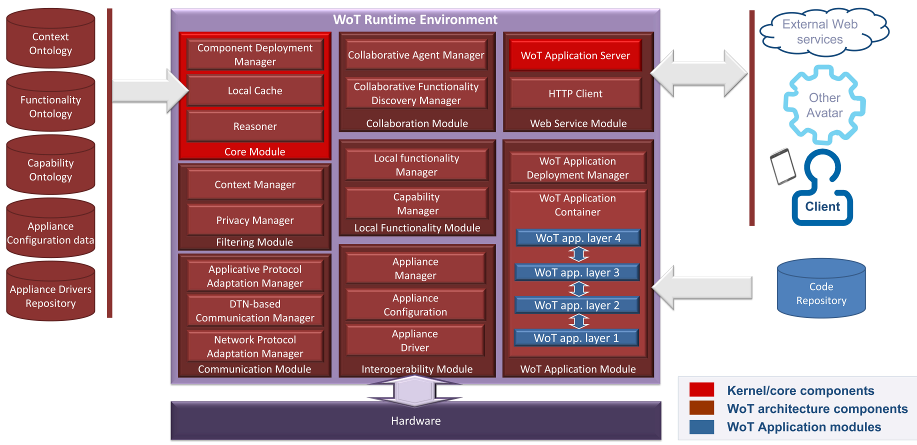

Fig. 3. Architecture of an avatar

5 Conclusion

We proposed an approach to enable cooperative intelligence for the Web of Things. Our approach is decentralized, Web-based and agent-oriented to enable proactive and cooperative intelligence between heterogeneous objects with the help of Web-based languages and protocols. Its builds on an agent abstraction called avatar that extends a real world device on the Web.

Implementing avatars according the introduced architecture enables to demonstrate the applicability of our work in the context of a production chain scenario. The result is an open, generic and Web-based solution to deliver high-level, user-understandable functionality, while interacting with a set of various physical objects through their avatars, enabling cooperation between them.

As future work, we envision to develop MAS protocols between avatars, and to extend our architecture to make it adaptable to different aspects such as MANET environments and scalability problems.

Acknowledgment

This work was supported by the french *Agence Nationale de la Recherche* through contract <ANR-13-INFR-012>.

References

1. Melliti, T., Haddad, S., Suna, A., Fallah-Seghrouchni, A.E.: Web-masi: Multi-agent systems interoperability using a ws based approach. In: Proc. of the IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology, IEEE (2005) 739–742
2. Richards, D., Splunter, S., Brazier, F., Sabou, M.: Composing web services using an agent factory. In: Extending Web Services Technologies. Volume 13 of Multiagent Systems, Artificial Societies, and Simulated Organizations. Springer US 229–251
3. Maamar, Z., Mostéfaoui, S.K., Yahyaoui, H.: Toward an agent-based and context-oriented approach for ws composition. IEEE Trans. Knowl. Data Eng. **17**(5) (2005) 686–697
4. Morais, A.J.: A multi-agent approach for web adaptation. In: 7th International Conference on Practical Applications of Agents and Multi-Agent Systems. Volume 55 of Advances in Intelligent and Soft Computing., Springer (2009) 349–355
5. Gutiérrez-García, J.O., Corchado, F.F.R., Koning, J.L.: An obligation-based framework for web service composition via agent conversations. WIAS **10**(2) (2012) 135–150
6. Jin, T., Goschnick, S.: Utilizing web services in an agent based transaction model (abt). In: Workshop on Web services And Agent-based engineering. (2003)
7. Falou, M.E., Bouzid, M., Mouaddib, A.I., Vidal, T.: A distributed multi-agent planning approach for automated web services composition. WIAS **10**(4) (2012) 423–445
8. Ma, B., Chen, B., Bai, X., Huang, J.: Design of bdi agent for adaptive performance testing of web services. In: Proceedings of the 10th International Conference on Quality Software, IEEE Computer Society (2010) 435–440
9. Coria, J.A.G., Castellanos-Garzón, J.A., Corchado, J.M.: Intelligent business processes composition based on multi-agent systems. Expert Syst. Appl. **41**(4) (2014) 1189–1205
10. Ramírez, E.H., Brena, R.F.: Multi-agent systems integration in enterprise environments using web services. IJIIT **2**(3) (2006) 72–88
11. Goschnick, S., Balbo, S., Sterling, L., Sun, C.: Tandem - a design method for integrating web services into multi-agent systems. In: 5th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2006), ACM (2006) 939–941
12. Karaenke, P., Schuele, M., Micsik, A., Kipp, A.: Inter-organizational interoperability through integration of multiagent, web service, and semantic web technologies. In: Agent-Based Technologies and Applications for Enterprise Interoperability. Volume 98. Springer (2012) 55–75
13. FIPA: Review of fipa specifications (December 2006)
14. Weiss, G., ed.: Multiagent Systems, 2nd edition. MIT Press, Cambridge, USA (2013)
15. Gautier, P., Gonzalez, L.: L'Internet des objets:Internet,mais en mieux. AFNOR (2011)