



# ASAWoO ANR Project

## Deliverable 2.1: Specification and implementation of WoT constrained object architecture

Grant Agreement: N° ANR-13-INFR-0012-04

---

### Abstract

This document presents avatars as an extension of physical objects in virtual worlds. Physical objects can therefore be coupled with avatars, to form cyber-physical objects that are compatible with WoT constraints and infrastructures. Avatars are implemented using a distributed software platform that can be fully deployed on powerful objects, or distributed over resource-constrained objects and a cloud infrastructure.

---

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Physical Object Constraints</b>	<b>2</b>
<b>3</b>	<b>Overview of the Avatar Architecture and of the WoT Infrastructure</b>	<b>3</b>
<b>4</b>	<b>Extending Objects with Avatars</b>	<b>4</b>
<b>5</b>	<b>And About Seriously Constrained Objects?</b>	<b>5</b>
<b>6</b>	<b>Conclusion</b>	<b>8</b>

# 1 Introduction

The “Web of Things” (WoT) extends the Internet of Things so that physical object can be accessed and controlled using Web standards. Objects are expected to expose logical interfaces through Web services, to describe Web contents and services using semantic Web languages and annotations, and to communicate together through standard protocols in order to provide software interoperability between objects. Although the number and types of connected objects increases quickly, the WoT is not yet a reality, as several issues must be addressed in order to seamlessly interconnect physical objects, and to make these objects accessible on the Web. Indeed, objects are heterogeneous and are rarely able to communicate with each other as they do not share the same communication protocols or the same data formats.

In project ASAWoO, we introduce a new kind of software artifact called “avatar”. Avatars provide a virtual extension to physical objects. Physical objects can therefore be coupled with avatars, to form cyber-physical objects that are compatible with WoT constraints and infrastructures. Avatars are implemented using a distributed software platform that can be fully deployed on powerful objects, or distributed over resource-constrained objects and a cloud infrastructure. This platform is designed and developed in the ASAWoO project to address the research challenges of the WoT.

## 2 Physical Object Constraints

Physical objects (e.g., robots, sensors, camera, mobile phones, connected-watches) can have radically different functional and physical capabilities (i.e., processing, acting, sensing and storage). Some objects are fixed and can be connected to the Internet using wired links, while others are mobile and can suffer from connectivity disruptions due to their mobility and the short communication range of their wireless interface.







	Fixed objects	Mobile objects
Resourceful objects		
Resource constrained objects		
Resourceless objects		

Figure 1: Examples of physical objects considered in project ASAWoO.

From our point of view, connected objects can be classified in 6 different categories, depending if they are fixed or mobile and depending on their physical capabilities. Resourceful objects embed a Web of Things (WoT) plat-

form that hosts all the services they provide. Installation of these objects is often simple since they are standalone and do not require additional infrastructure elements to run. Resource-constrained objects cannot embed the whole WoT software platform due to restricted resources, but it is possible to link them to distant hosts that can embed missing parts of the platform. Resourceless Objects are passive objects, detected using unique identifiers such as QR codes or RFID tags. They do not have any computation, storage or memory capability. They can be extended with software deployed on the cloud or on the local network gateway.

### 3 Overview of the Avatar Architecture and of the WoT Infrastructure

In order to address the issues posed by the interoperability, the management, the communication of these heterogeneous devices, we have designed both an autonomous distributed platform, called Avatar, and an WoT infrastructure allowing to access and to deploy instances of the Avatar platform on the connected objects or/and on hosts forming a cloud infrastructure.

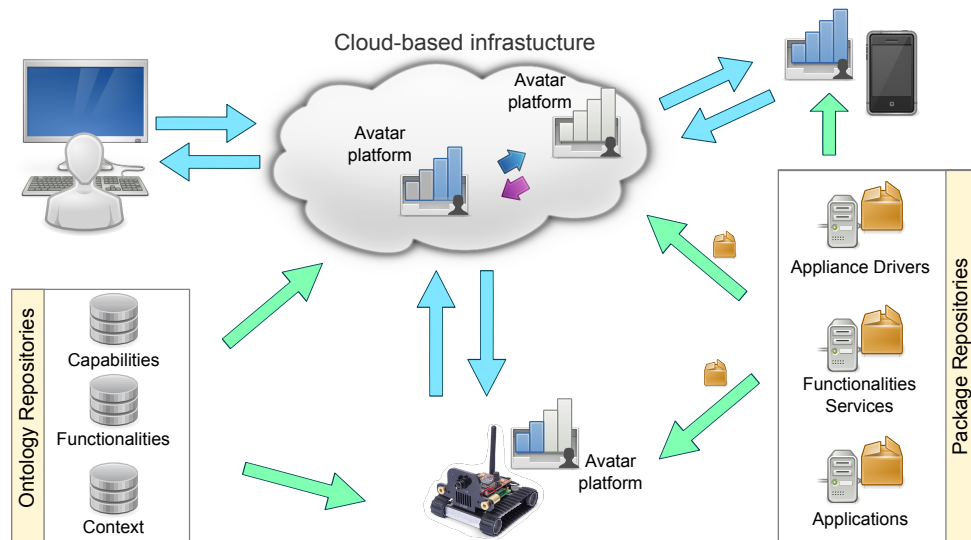


Figure 2: WoT Infrastructure designed in project ASAWoO

The WoT infrastructure illustrated in Figure 2 is composed of ontology repositories, software package repositories and runtime environments that can host avatar platforms. Ontology repositories contain RDF-based ontologies that can be used by avatar platforms to infer from their runtime context, and from the capabilities and functionalities they provide, which additional functionalities and capabilities they could offer by deploying new services locally, or by exploiting remote services provided by other avatar platforms. These ontologies are processed by a specific element, called Reasoner (see Figure 3). This reasoner relies on three other main managers of the core module, namely the functionality manager, the deployment manager and the context manager. The functionality manager is used to maintain an up-to-date list of the functionalities provided by the avatar (i.e., by the physical object). These functionalities can appear and disappear dynamically according to the service that are deployed, started and stopped by the avatar platform itself according to its execution context. For example, due to power budget limitation, a mobile object (e.g., a robot) can deactivate some functionalities that are known for their high

energy consumption. The deployment manager is especially responsible for 1) obtaining semantic descriptions of services from the software package repositories, 2) providing these descriptors to the reasoner, 3) downloading from the repositories the software packages identified by the reasoner, and 4) deploying these packages locally. These packages are the service deployment units. They contain pieces of code and additional resources (configuration files, ...). These deployment units are detailed in the remainder of this document.

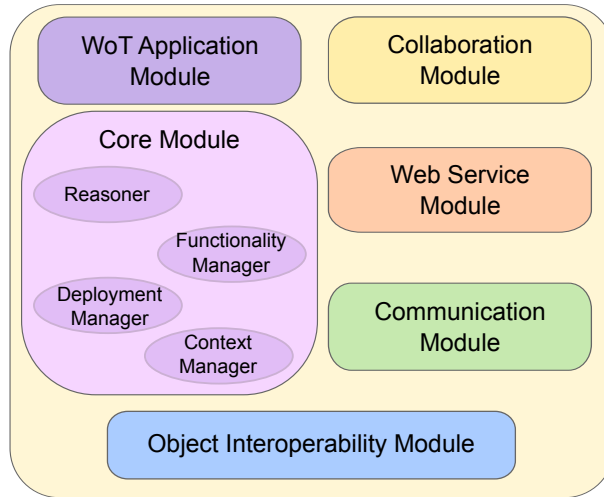


Figure 3: General Architecture of the Avatar Platform.

In our platform, the Web Service module allows to declare local services as REST Web Services. Doing so, it is possible to interact with avatars (and the physical objects) using the standard HTTP or COAP protocols, and to ensure the interoperability between the avatars/objects. This Web Service module is also detailed in the rest of this document. The three other modules shown in Figure 3, are not in the topic of this document, and therefore are not presented.

## 4 Extending Objects with Avatars

To introduce the concept of avatar [2, 5], we need to define *object* and *proxy* (see [4, 3]).

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

*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  $P = \langle Kp, Sp \rangle$  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. 4). We call such an representation an avatar. Avatars are not simple proxies. An avatar is an autonomous entity (i.e. an agent) which has its own 4-tuple  $A = \langle Ga, Ba, Ka, Sa \rangle$  with  $G \subset Ga, B \subset Ba, K \subset Ka$  and  $S \subset Sa$ . 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.

## 5 And About Seriously Constrained Objects?

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 ([1, 6]).

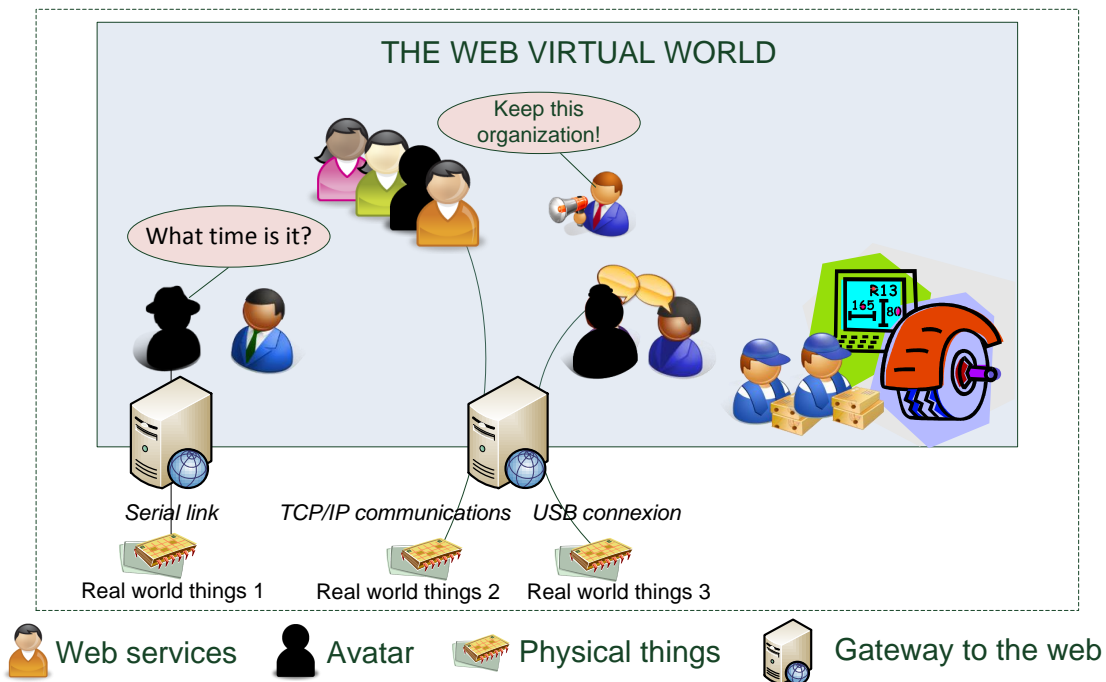


Figure 4: Using avatars in the Web

**Application of our approach** To illustrate our approach, we consider the following production chain scenario (inspired by [2]) (fig. 5).

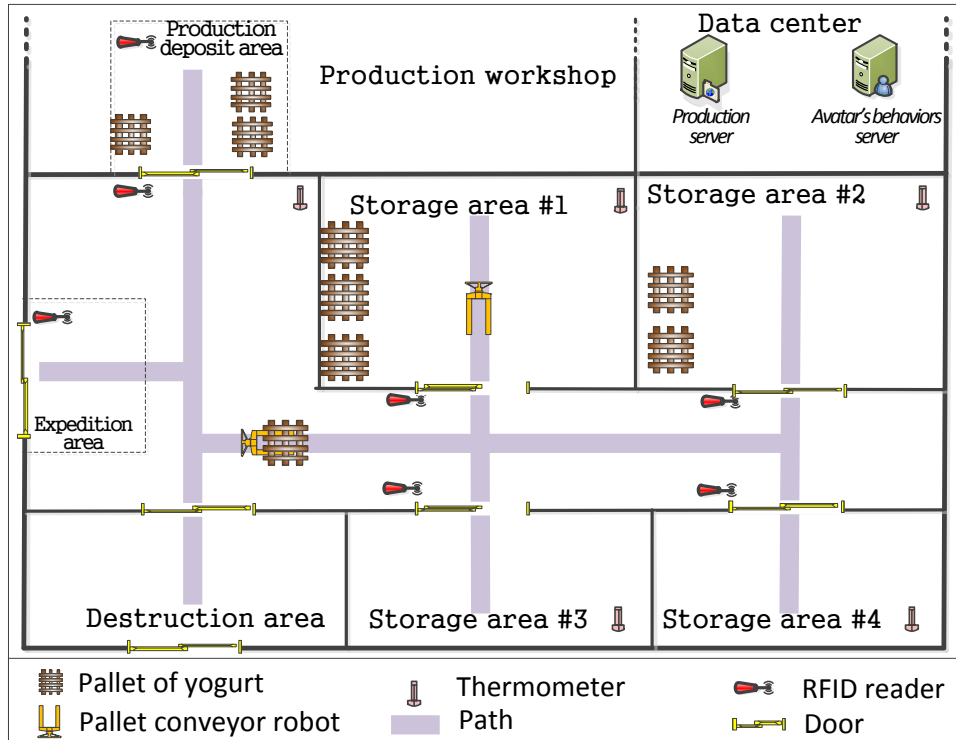


Figure 5: Illustration of the scenario

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°C, 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.
  - (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 area 1*.
  - (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°` 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°C` 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) ...

## 6 Conclusion

We present here how constrained 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. Extended objects can be resourceless : neither a CPU nor RAM chips have to be embedded.

## References

- [1] FIPA. Review of fipa specifications, Dec. 2006.
- [2] GAUTIER, P., AND GONZALEZ, L. *L'Internet des objets:Internet,mais en mieux*. AFNOR, 2011.
- [3] JAMONT, J., MÉDINI, L., AND MARISSA, M. A web-based agent-oriented approach to address heterogeneity in cooperative embedded systems. In *Trends in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection, 12th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2014) Special Sessions, Salamanca, Spain, June 4-6, 2014*. (2014), pp. 45–52.
- [4] KHALFI, E., JAMONT, J., CERVANTES, F., AND BARHAMGI, M. Designing the web of things as a society of autonomous real/virtual hybrid entities. In *Proceedings of the 2014 International Workshop on Web Intelligence and Smart Sensing, IWWISS '14, Saint Etienne, France, September 1-2, 2014* (2014), pp. 16:1–16:5.
- [5] MARISSA, M., MÉDINI, L., JAMONT, J., SOMMER, N. L., AND LAPLACE, J. An avatar architecture for the web of things. *IEEE Internet Computing*, (accepted) (2015), 30–38.
- [6] WEISS, G., Ed. *Multiagent Systems, 2nd edition*. MIT Press, Cambridge, USA, 2013.