Towards An Avatar Architecture for the Web of Things

Michael Mrissa, Lionel Médini, Jean-Paul Jamont, Nicolas Le Sommer, Jérôme Laplace

Towards An Avatar Architecture for the Web of Things

Michael Mrissa, Lionel Médini, Jean-Paul Jamont, Nicolas Le Sommer and Jérôme Laplace

***Abstract*—The Web of Things (WoT) extends the Internet of Things considering that each physical object can be accessed and controlled using Web-based languages and protocols.**

In this paper, we summarize ongoing work promoting the concept of avatar as a new virtual abstraction to extend physical objects on the Web. An avatar is an extensible and distributed runtime environment endowed with an autonomous behaviour. Avatars rely on Web languages, protocols and reasoning about semantic annotations to dynamically drive connected objects, exploit their capabilities and expose their functionalities as Web services. Avatars are also able to collaborate together in order to achieve complex tasks.

I. Introduction

The future Internet has been envisioned as an Internet of Things[1], in which billions of heterogeneous objects will be connected to the Internet using wired or wireless links. The "Web of Things" (WoT) extends the Internet of Things in order to enable access and control of physical objects 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[2], the WoT is not yet a reality, as several issues must be addressed in order to seamlessly interconnect physical objects, and make these objects accessible on the Web. On the one hand, objects are heterogeneous and rarely able to communicate together because most of them implement proprietary communication protocols instead of Web standard protocols [1]. Yet, end-users can combine their capabilities, thus providing meaningful and complex functionalities. On the other hand, objects usually respond to basic requests using their sensors and actuators, whereas users require comprehensible and usable services to achieve their goals.

We argue that an open market of software components and applications dedicated to connected objects that rely on Web standards will help the WoT to become a reality. Such a WoT marketplace should permit developers and industrial companies to distribute their software applications and components, and should provide end-users with software pieces allowing them to implement different functionalities into their objects in order to perform various tasks.

To reach this objective, a WoT runtime environment (WoT RE) must be defined. The runtime environment must bridge the gap between resource-constrained objects and the Web, which offers complex and heavyweight, but interoperable and user-friendly technologies. We identified the following requirements for such an environment:

- (R1) Autonomy and resource management: Connected objects can be autonomous devices with limited resources (battery, CPU and memory). Thus, the WoT RE must be able to estimate the global cost of physical actions in terms of device usage, computation and networking, to determine if an object can realize a given action.
- (R2) Live reactivity: The WoT RE must be able to adapt its behavior to its environment at runtime, regarding functional and QoS aspects. It must be able to provide a service with graceful degradation if typical service computation time exceeds time requirements.
- (R3) Computation delegation: to promote autonomy, resource management and live reactivity, the WoT RE should allow deploying code modules on the object processing unit or on a cloud infrastructure and should identify the most suitable location to execute each module.
- (R4) Safety: The WoT RE must ensure the physical and informational harmlessness of the object for people and assets, using risk analysis and regulation criteria before realizing an action.
- (R5) Disconnection tolerance: The WoT RE must be able to support the connectivity disruptions of between the mobile objects themselves and between the mobile objects and the access points of an infrastructure-based network.
- (R6) Interoperability: The WoT RE must be able to handle heterogeneous objects in terms of size, OS and protocols. It must also allow objects with similar physical properties to fulfill the same actions (device independence).
- (R7) User-understandable interfaces: The WoT RE must provide entry points to handle requests, and provide applications that correspond to users' high-level goals.

Michael Mrissa, Lionel Médini are members of the LIRIS Laboratory, Université de Lyon, France, e-mail: {michael.mrissa,lionel.medini}@univ-lyon1.fr

Jean-Paul Jamont is member of the LCIS Laboratory, Université de Grenoble Alpes, France, e-mail: jean-paul.jamont@lcis.grenoble-inp.fr

Nicolas Le Sommer is member of the IRISA Laboratory, Université de Bretagne Sud, France, e-mail: nicolas.le-sommer@univ-ubs.fr

Jérôme Laplace is head of the company "Génération Robots", France, e-mail: jl@generationrobots.com

[1]http://www.itu.int/osg/spu/publications/internetofthings/

[2]See http://www.ifr.org/service-robots/statistics/, or http://www.ericsson.com/res/docs/whitepapers/wp-50-billions.pdf and http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

The ASAWoO project[3] aims at providing an distributed, open and generic architecture for WoT REs along with a WoT infrastructure that provides means to execute, secure and link WoT RE instances, and a WoT application marketplace. WoT REs are called avatars and are endowed with an autonomous behavior and on Web languages and protocols. An avatar provides a virtual abstraction of physical objects in the Web. It can expose object basic capabilities as high-level functionalities on the Web, connect to and interact with objects using the most appropriate protocols and languages, perform different reasoning processes to discover, build and adapt such functionalities, query external Web services, interact with users and other objects and execute generic WoT applications. In the remainder of this paper, we show how our architecture design meets the requirements above. The (Rn) notation indicates a specific design to answer the nth requirement. Our preliminary results show good insight on the relevance of our approach.

The remainder of the paper is organized as follows. Section II describes the approach and gives detail on the avatar architecture and its lifecycle. Section III gives details on the different contributions involved in the avatar architecture. Section IV presents our prototype to show the feasibility of our work. Section V discusses related work and highlights the novelty of our contribution. Section VI discusses our ongoing results and gives some guidelines for future work.

II. Avatar Architecture and Lifecycle

Avatars possess a complex component-based architecture, allowing to take into account additional information that is not *a priori* visible to connected objects but is available from other Web sources and/or avatars to add intelligence to object behaviors. The main components that we propose to leverage such high-level behaviors in WoT applications rely on Web standards to provide interoperability among objects and on advances in various domains, such as component-based programming, embedded systems, cloud computing, delay-tolerant networks, Web and semantic Web technologies and multi-agent systems to leverage single and collective intelligence in object behaviors. As illustrated in Figure 1, an avatar can be distributed on an object and in a cloud infrastructure depending on the resources offered by the object it extends. We identify three categories of connected objects:

1) Resourceful Objects: provide software services and embed a Web server that offers service interfaces. It is often simple to link these objects with other objects or software services, and to deploy all the avatar components on them.
2) Resource-constrained Objects: cannot embed all avatar components due to restricted resources but it is possible to link them to distant hosts that can embed missing components.
3) Resourceless Objects : These objects are passive objects, detected using unique identifiers such as QR codes or RFID tags. They do not have any computation, storage and memory capability. Their avatars are deployed on the cloud or on the local network gateway.

[3]Project homepage: https://liris.cnrs.fr/asawoo/

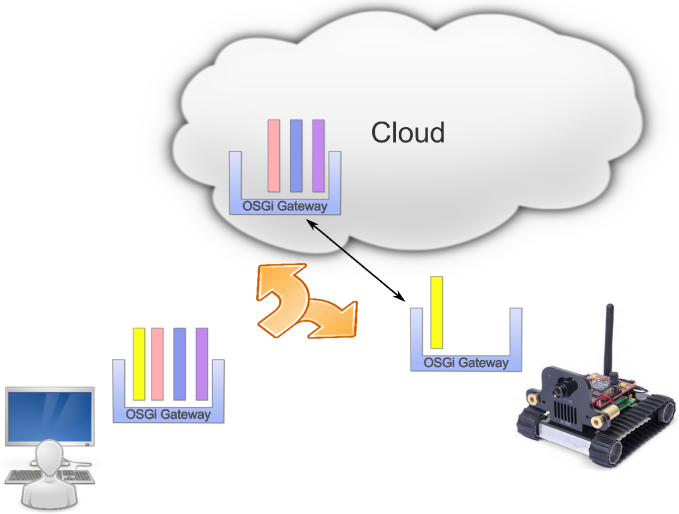

Fig. 1. Distribution of the Avatar architecture on a resource-constrained physical object and cloud infrastructure.

According to these categories, it is possible to adapt the deployment of software components of the avatar to different places to globally improve its operation. In the following, we present the avatar architecture and its lifecycle.

A. Avatar Architecture

Our avatar runtime environment has been designed as an OSGi service-oriented architecture. The implementation of the runtime environment is entirely decoupled from its logical architecture. Consequently, it is possible to adapt the avatar to different types of objects dynamically (R2), as well as to adapt the distribution of the services implementing the avatar to different places - object, local network gateway and cloud - in order to improve the avatar execution (R1, R3).

Therefore, the avatar architecture is structured as a set of "manager" components (OSGi services) with each a particular role. Each service can be transparently (R3) and at runtime (R2) deployed on any physical location depending on the object and application context. Services interact with each other according to the principles that guide the behavior of the avatar during its lifecycle presented below (Sec. II-B). With the architecture come the necessary inter-service middleware and service deployment and communication schemes to build the avatar as a common logical entity, sometimes requiring distant communication. Figure 2 shows the services available in the architecture, grouped into functional modules as follows.

1) Core module: The core module includes components that are central to the architecture and reused in different steps of the avatar lifecycle. The **reasoner** allows reasoning about knowledge representation and is useful to the local functionality manager, the context manager and the privacy manager. The **local cache** improves middleware performance and speeds up data exchange between the different services of the architecture. The **Component deployment manager** is a core component that decides when and where to deploy the other components in the architecture. This component is essential to respect the (R1), (R2), (R3) and (R5) requirements.

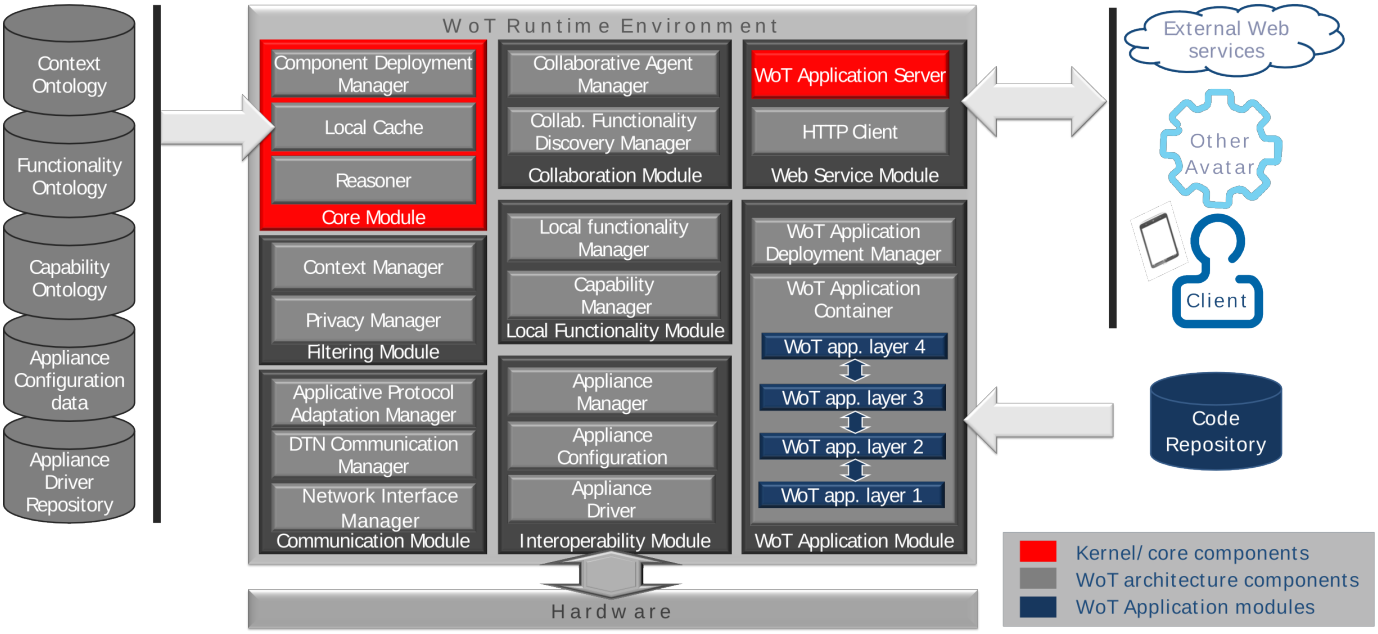

Fig. 2. Web architecture of avatar agents

2) Web service module: The **WoT Application Server** is the endpoint that exposes simple (one involved) or complex (multiple involved) functionalities available as applications. Both local (one avatar/object involved) and collaborative (multiple avatars/objects involved) applications are available to other avatars and end-users. The **HTTP Client** allows the avatar to interact with an external Web service available on the Web, including the applications another avatars provide. These two components are also in charge of implementing the inter-avatar negotiation processes, using a Web service-based communication scheme.

3) WoT Application module: A **WoT application** (i.e. end-user understandable, high-level object behavior) is executed inside a **WoT Application Container** that can be physically distributed over the physical layers of the architecture (object, gateway, cloud) thanks to the **WoT Application Deployment Manager** (R7). The different parts of the application are implemented as "code modules" that are cross-compiled to be either executed on the object or on the gateway/cloud wrt. contextual adaptation decisions (R2).

4) Local functionality module: The **Capabilities Manager** exchanges with the object to discover and identify its capabilities (see Sec. III-A). The **Local Functionality Manager** deducts available functionalities from the set of capabilities of the capabilities manager[4] (see Sec. III-C). To do so, it also gets helped by the context and privacy managers and the reasoner to reason about exposable functionalities according to the current situation.

5) Collaboration module: An avatar must identify other avatars that can provide functionality. To enable object collective behavior, the **Collaborative Functionality Discovery Manager** allows to look for external functionality in the avatar community[5]. By observing the activity of other avatars in its immediate environment, the **Collaborative Agent Manager** can identify if its goals are compatible with the goals of other avatars. It can also note if a conflict with other avatars occurs (resource/function access). According these interaction situations (obstruction, independence, collaboration...), negotiations with other avatars could be achieved to expose collaborative functionality in the WoT Application server (cf Sec. III-D).

6) Communication module: The **Network Interface Manager** and the **Application Protocol Adaptation Manager** respectively select the right network (Wi-Fi, Bluetooth, Zigbee,...) and application protocols (CoAP, HTTP) according to available communication interfaces and performance needs (throughput and energy consumption). In order to support connectivity disruptions due to mobile contexts, we have introduced in the communication module of the avatars a **DTN Communication Manager**, responsible for initializing and configuring the opportunistic communication protocol we have defined and that relies on the "store, carry and forward" principle (R5). These managers are described in Sect. III-B.

7) Filtering module: The **Context Manager** aggregates data from domain ontologies, external services and environment events into contextual situations [2], in order to perform semantic multi-level adaptation, to 1) identify on which avatar to expose a collaborative function; 2) decide which functionalities to expose wrt. object context; 3) choose where (on which layer) to deploy each architecture component and application code module; 4) determine the most appropriate protocol stack for the current communication scheme and contextual conditions. The **privacy manager** will rely on models developed in previous work [3] that describe a query in terms of user role, purpose of the query and data queried

[4]An observer design pattern can be implemented here for dynamic updates to answer the (R2) requirement.

[5]For simplicity purpose we assume in this paper that an avatar community is delimited to the local network.

to reason about privacy constraints and protect data (R4).

8) Interoperability module: The **Appliance communication manager** is the high level component of the interoperability module. It communicates with other components in the architecture through its high level interface and as well works with the appliance configuration manager and the appliance driver described below to communicate with the object. The **Appliance configuration manager** relies on a database of object configuration tools to associate communication methods to objects. For instance, when a Lego Mindstorms sends its ID on the USB wire, the configuration manager sends the required drivers to load to enable communicating with the object. The **Appliance driver** loads and uses the drivers to send and receive messages to and from the object. Thanks to its high level uniform interface drivers are dynamically loaded and abstracts low level object communication (R6).

All the managers get involved at different, sometimes overlapping stages during the avatar lifecycle. We describe the lifecycle in the following section.

B. Avatar lifecycle

The avatar begins its lifecycle with its instantiation from the avatar builder that creates an avatar instance. The avatar builder is designed to be located on the local network gateway or in layer so that it becomes possible to detect the arrival of an object in the network (new wifi connection, new bluetooth device detected, new USB device plugged, etc.).

Upon creation, the avatar deploy the main components connects to and and exchanges a set of messages with the object it extends to discover its actuators and sensors (core communication and interoperability module). Once the sensors and actuators have been discovered, they form a list of capabilities (local functionality module). Based on this list the avatar decides with the help of a reasoning engine which capabilities to expose as functionalities (filtering module). Functionalities are exposed as Web services and can be discovered and invoked by other avatars in the context of WoT applications [4] (communication, WoT application, collaboration and Web service module).

During the lifetime of the avatar and object, services are queried by other avatars and end users. Sometimes pluggable devices are added to or removed from the object, or environmental information changes (day/night, weather, etc.). In such case the avatar is notified of the change (via polling or observer pattern) and updates its capabilities and exposed functionalities accordingly, which answers the live reactivity requirement.

When the lifecycle comes to an end (object disconnection), the avatar notifies its community that the services it exposed are not available anymore and terminates all the processes in memory it is attached to.

III. Avatar Components

A. Avatar/Object Introspection

Physical objects can have different capabilities in terms of processing, memory, communication, sensing and action. In order to discover an object resources and amongst others decide how to deploy the avatar of a given object, we perform an introspection of the object using SAJE (System-Aware-Java Environment)[6]. SAJE is a part of the hardware abstraction layer of the ASAWoO middleware platform. SAJE makes it possible to give information about the capabilities of physical objects, and to control some components of these objects such as the communication interfaces. The discovery of resourceless objects is not performed directly on the objects, but instead on the devices they are attached to. This discovery is performed continuously on some objects, because sensors or actuators can be plugged (or unplugged) dynamically.

Based on the information returned by SAJE, the deployment manager of the ASAWoO middleware platform is able to deploy dynamically from a remote repository the OSGi bundles that allow to monitor and to control the hardware components (e.g., sensors, actuators) of the physical objects. These bundles also allow to have a semantic description of the capabilities of the objects. These capabilities are then used by the ASAWoO middleware platform in order to decide which functionalities can be deployed dynamically on the object (or on the cloud).

B. Communication protocols

Depending on the capabilities and on the execution context of the objects, the avatar runtime environments are able to select the most suited communication protocols dynamically. Thus, they can either use a communication protocol based on HTTP (over TCP), or a standard UDP-based version of COAP or a disruptions tolerant based version of COAP, which implements the ”store, carry and forward principle” in order to support the connectivity disruptions. Such connectivity disruptions can be frequent and impredictable in use cases involving mobile devices (e.g., robots) equipped with short range wireless communication interfaces such as Bluetooth, Wi-Fi or Zigbee.

Opportunistic and disruption/delay tolerant (DT) communications have been studied in several research works and projects over the last years [5], but the issues introduced by the service-oriented opportunistic (ort DT) computing have been addressed only in few works [6], [7] The disruption tolerant COAP based protocol implemented in avatars currently relies on the solution we proposed in [7].

C. Semantic processing

Semantic processing is a major feature of our architecture. An avatar needs to reason about capabilities and functionalities, while taking into account several aspects from privacy to security and context. In [4], we proposed a generic model to describe and exploit the semantic relationships between a functionality used from the application perspective, and a capability that expresses the possibility for an object to realize an action. Our model enables domain-dependent instances to populate the ontology and provide the means for avatars to get the information about which simple or complex functionalities can be exposed, being given a set of available capabilities. As well, it allows identifying complex functionalities that are partially implemented with the help of existing objects,

[6]https://www-casa.irisa.fr/saje/

opening the possibilities for additional application when other avatars enter the community.

As well, the context manager aims at performing multi-level adaptation [8]. It relies on a semantic context model [9] that can be processed at different abstraction levels and populated with functional and QoS data provided by object sensors and external resources, and on an adaptation engine [10], both compatible with environmental data and high-level functionalities.

D. Enabling Collective Behavior between Avatars

An avatar inherits goals, knowledge, sensors and actuators from its physical object. Its capabilities and its knowledge are extended with Web information and services but also through its community. To meet a goal, an avatar need resources (energy, storage, cpu, bare objects, ...) and skills (Web services, avatar's skills, ...).

If local resources and skills are available to accomplish a goal, an avatar do not generally need collective features. If an avatar has all the skills to accomplish a task but if the resources are not sufficient, it will be in a situation of obstruction. A coordination mechanism will be necessary to avoid harmful interactions. If resources are sufficient but if an avatar does not have all the required skills, collaboration mechanisms will be needed. In these three cases, we assume that the goals of all the avatars are compatible. If this is not the case, depending on the availability of resources and skills, avatars could be in antagonism (individual or collective conflicts if there are insufficient resources, individual or collective competition if the skills are not available). A possible solution is to establish coalition against a subset of other avatars. Avatars, resources and services could appear or disappear dynamically, so an avatar must continuously be aware of the situation of interaction in which it operates.

IV. Prototype

Our current prototype is divided in the following parts:

- A WoT physical infrastructure that contains a gateway to connect objects and a WoT Processing Unit (e.g. a cloud infrastructure) that hosts avatar parts outside of the objects.
- A WoT logical infrastructure that contains an avatar container and the different ontologies and repositories depicted in Section 2
- An avatar architecture implemented in Java/OSGi and designed that can locally or remotely instantiate and invoke the avatar components and compose the core module of the avatar architecture
- A WoT Runtime Environment that implements the WoT component framework on the object layer; its implementation language depends on the object OS and bridges this framework with the object hardware
- A discovery module implemented using the SAJE library
- A Web service module using the JAX-RS library
- An interoperability module based on the AllJoyn framework[7]
- A set of OWL functionality and capability classes that describe domain knowledge according to different scenario
- A semantic local functionality module [4] implemented using the Java OWL API and operated using the HermiT reasoner
- A preliminary collaboration module adapting the ABT algorithm into a set of HTTP exchanges between RESTful resources.
- A communication module that implements COAP and disruption-tolerant protocols, based on the solution proposed in [7]

We are currently implementing the filtering and WoT application modules.

V. Related Work: Web of Things infrastructures

The Web of Things integrates various research and application fields, among which embedded systems, wireless networks, software infrastructure, Web technologies and artificial intelligence. According to [11] and more recently[8], a Web of Things infrastructure should: allow discovering objects without configuration, dynamically adapt to its environment, be secured so that things and applications are harmless and avoid privacy issues, allow manual or semi-automatic service composition and provide services that make sense for the users. From a more technical point of view, it should:

- rely on Web standards to achieve interoperability [12]
- take into account several communication models (request/response, message-oriented, event-based, publish-subscribe, streaming...) [13], [14]
- allow executing code on objects or delegating it to the cloud
- semantically deduce available functions and enrich data [15]
- open an easy way for developing marketable applications[9]
- encourage developers to respect good practices[10]

Several ongoing projects (Webinos[11], Compose[12], SensorMeasurement[13], CityPulse[14]...) and infrastructures ([12], [13], [15]) are related to the Web of Things. Each one highlights a specific point of view or different properties. For instance, the COMPOSE project is oriented towards standardizing WoT marketable applications, CityPulse focuses on event processing, the SensorMeasurement project proposes a reasoning toolkit to reason on sensor data and other work focuses on object security[15].

However, if the lack for a standard specification for developing WoT infrastructures can only be solved by organizations

[7] https://www.alljoyn.org/

[8] http://www.w3.org/2014/02/wot/papers/ricardo.pdf, http://www.w3.org/2014/02/wot/papers/karapantelakis.pdf

[9] http://www.compose-project.eu/sites/default/files/publications/COMPOSE_v2_factsheet.pdf

[10] http://iot-datamodels.blogspot.fr/2014/05/design-patterns-for-internet-of-things.html, https://www.w3.org/wiki/Web_of_Things_Workshop_Breakout_Sessions

[11] http://webinos.org/

[12] http://www.compose-project.eu/

[13] http://sensormeasurement.appspot.com/

[14] http://www.ict-citypulse.eu/

[15] http://www.w3.org/2014/02/wot/papers/mattsson.pdf

such as the World Wide Web Consortium[16], it is possible to define a comprehensive architecture for software objects that represent physical ones on the Web, such as avatars do. Such architectures can be contained in a WoT infrastructure that will cope with yet-to-come WoT standards, assuming that avatar communication schemes follow state of the art principles in terms of services and protocols.

A comprehensive architecture for software objects that can cope with multiple points of view has been proposed for the IoT in the FI-Ware[17] project. But to the best of our knowledge, a similar architecture that targets WoT standards is missing. Therefore, such an architecture can take advantage of advances in each field and one can develop modules related to specific concerns, as long as these works can be encapsulated in components. Using this approach, our avatar architecture proposes different modules that allow plugin heterogeneous objects, communicating with them using different paradigms and protocols, deduce and reason about their functionalities, adapt their behavior according to semantized context representations, collaborate with one another and expose standard services to the users.

VI. Conclusion

The connection between the Web and physical objects is not yet a reality. In this paper, we propose an avatar architecture that enables connecting objects to the Web and improving their skills with additional intelligence. Avatars receive data from the objects they extend and provide reasoning capability to drive object towards a cleverer behavior, thus naturally improving object intelligence and rising object possibilities to a new level.

Future work includes developing multi-agent communication protocols for effective exchange and creation of value-added functionality to be exposed to end-users, as well as studying the limitations of the different parts of our architecture.

Acknowledgement

This work is supported by the French ANR (Agence Nationale de la Recherche) under the grant number <ANR-13-INFR-012>.

References

[1] Dominique Guinard, *A Web of Things Application Architecture ? Integrating the Real-World into the Web*, Ph.D. thesis, PhD thesis No. 19891, ETH Zurich, August 2011.

[2] Luca Buriano, Marco Marchetti, Francesca Carmagnola, Federica Cena, Cristina Gena, and Ilaria Torre, "The role of ontologies in context-aware recommender systems," in *Mobile Data Management, 2006. MDM 2006. 7th International Conference on*. IEEE, 2006, pp. 80–80.

[3] Salah-Eddine Tbahriti, Chirine Ghedira, Brahim Medjahed, and Michael Mrissa, "Privacy-enhanced web service composition," *IEEE T. Services Computing*, vol. 7, no. 2, pp. 210–222, 2014.

[4] Michaël Mrissa, Lionel Médini, and Jean-Paul Jamont, "Semantic Discovery and Invocation of Functionalities for the Web of Things," in *IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises*, June 2014.

[16]http://www.w3.org/

[17]http://www.fi-ware.org/

[5] Vinicius F. S. Mota, Felipe D. da Cunha, Daniel F. Macedo, José Marcos S. Nogueira, and Antonio A. F. Loureiro, "Protocols, mobility models and tools in opportunistic networks: A survey," *Computer Communications*, vol. 48, pp. 5–19, 2014.

[6] Marco Conti, Emanuel Marzini, Davide Mascitti, Andrea Passarella, and Laura Ricci, "Service selection and composition in opportunistic networks," in *IWCMC*, Roberto Saracco, Khaled Ben Letaief, Mario Gerla, Sergio Palazzo, and Luigi Atzori, Eds. 2013, pp. 1565–1572, IEEE.

[7] Ali Makke, Yves Mahéo, and Nicolas Le Sommer, "Towards Opportunistic Service Provisioning in Intermittently Connected Hybrid Networks," in *4th International Conference on Networking and Distributed Computing (ICNDC 2013)*, Honk Kong, China, Dec. 2013, IEEE CS.

[8] Kurt Geihs, Paolo Barone, Frank Eliassen, Jacqueline Floch, Rolf Fricke, Eli Gjorven, Svein Hallsteinsen, Geir Horn, Mohammad Ullah Khan, Alessandro Mamelli, et al., "A comprehensive solution for application-level adaptation," *Software: Practice and Experience*, vol. 39, no. 4, pp. 385–422, 2009.

[9] Jacqueline Floch, Svein Hallsteinsen, Erlend Stav, Frank Eliassen, Ketil Lund, and Eli Gjorven, "Using architecture models for runtime adaptability," *Software, IEEE*, vol. 23, no. 2, pp. 62–70, 2006.

[10] Mounir Beggas, Lionel Médini, Frederique Laforest, and Mohamed Tayeb Laskri, "Towards an ideal service qos in fuzzy logic-based adaptation planning middleware," *Journal of Systems and Software*, vol. 92, pp. 71–81, 2014.

[11] Dominique Guinard, Vlad Trifa, Friedemann Mattern, and Erik Wilde, *From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices*, chapter 5, pp. 97–129, Springer, New York Dordrecht Heidelberg London, 2011.

[12] Dominique Guinard, Vlad Trifa, and Erik Wilde, "A resource oriented architecture for the web of things," in *Proceedings of Internet of Things 2010 International Conference (IoT 2010)*, Tokyo, Japan, Nov. 2010.

[13] Matthias Kovatsch, Martin Lanter, and Simon Duquennoy, "Actinium: A restful runtime container for scriptable internet of things applications," in *IOT*. 2012, pp. 135–142, IEEE.

[14] Feng Gao, Edward Curry, and Sami Bhiri, "Complex event service provision and composition based on event pattern matchmaking," in *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems*. ACM, 2014, pp. 71–82.

[15] Amelie Gyrard, "A machine-to-machine architecture to merge semantic sensor measurements," in *Proceedings of the 22nd international conference on World Wide Web companion*. International World Wide Web Conferences Steering Committee, 2013, pp. 371–376.