CHAPTER

5 BUILDING A WEB OF THINGS WITH AVATARS

A COMPREHENSIVE APPROACH FOR CONCERN MANAGEMENT IN WOT APPLICATIONS

Lionel Médini[*], Michael Mrissa[*], El-Mehdi Khalfi[†], Mehdi Terdjimi[*], Nicolas Le Sommer[‡], Philippe Capdepuy[§], Jean-Paul Jamont[†], Michel Occello[†], Lionel Touseau[‡]

Univ Lyon, Université Claude Bernard Lyon 1, CNRS, LIRIS UMR5205, F-69622, Villeurbanne, France[*] Laboratoire LCIS, Université Grenoble Alpes, Valence, France[†] Laboratoire IRISA, Université de Bretagne Sud, Vannes, France[‡] Génération Robots, Bruges, France[§]

We propose the notion of avatar as a software architecture that extends a thing, in order to gain benefits from Web standards. Avatars achieve interoperability with things and expose high-level functionalities as RESTful resources, collaborate with each other and external services to form standard-compliant WoT applications.

CHAPTER POINTS

In this chapter, we highlight the following contributions:

- A generic software extension of things, called avatar. Avatars exploit things low-level capabilities, using their own protocols and encodings, to provide high-level, Web-compliant functionalities, while taking into account various concerns such as ensuring security and privacy, optimizing avatar-thing communications, locating application code or collaborating with other avatars. We present the component-based architecture of an avatar, explain its life-cycle and show how it relies on semantic Web technologies to fulfill its objectives. We also detail its components, some of them along with the architecture, the others as separate contributions.
- A RESTful disruption-tolerant support for the WoT. Resources exposed by connected things are identified by URIs and accessed through stateless services. Service requests and responses are forwarded using the store-carry-and-forward principle. We provide a complete service invocation model, allowing to perform unicast, anycast, multicast and broadcast service invocations using HTTP or CoAP.

Managing the Web of Things. DOI: 10.1016/B978-0-12-809764-9.00007-X

- A multi-level, multi-dimensional context model and domain-independent adaptation process. This allows avatars to reason about contextual information and solve various adaptation requests. We show how to pre-process stable context representations, so that the adaptation process can both cope with constantly changing environments and be optimized for reducing computing load at request time.
- An approach for inter-avatar collaboration. We introduce the notion of interaction situation as the reciprocal influence that avatars have on the actions of others when they are interrelated. We also provide avatars with social and human-inspired characteristics, such as collective identity (motivations and relationships among participants) and inter-subjectivity (the avatar's ability to consider other avatars' mental states in its beliefs). We show how, using these concepts, an avatar can decide whether to participate in an avatar community.

5.1 INTRODUCTION

The Web of Things (WoT) promotes the idea of using Web technologies to support interactions with things. The benefits of this idea are twofold. First, it aims at accessing all kinds of things through standard technologies and tools, thus achieving interoperability among things and breaking silos between thing manufacturers' proprietary technologies. This will allow reusing proven platforms, technologies and user interaction techniques, ease application development and reduce time-to-market. Second, by giving things an existence as Web resources, the WoT aims at enriching the possibilities of Web applications by bridging the gap between the virtual and the physical world. Indeed, being able to control one's smart home temperature through a social network is already a reality, as well as making a flower pot tweet when its soil is getting dry. Conversely, representing a thing as a Web resource gives it an existence in the informational world. Even inanimate objects can now be added a QR-code, so that people can post comments to a bottle of wine they drank together and get back all these comments in the bottle's blog. Of course, more powerful – and useful – applications are now built to make use of things through Web technologies. But the fact is that building such applications remains a handcrafted activity.

One of the reasons is the numerous domains and technologies that a complete WoT application can require: connected things and robots are subject to specific constraints that strongly impact applications, such as physical phenomena (gravity, friction, etc.), energy management and network disconnections. As classical applications, WoT applications have functional and non-functional requirements, called concerns in the remainder of this paper, such as reliability, performance, security, privacy and usability. Moreover, "intelligent" technologies can now be added to these applications, to help objects collaborate with one another to achieve a common goal, or to learn new knowledge to be used in the application. All these interrelated concerns have to be studied, modeled and integrated in applications in order to fulfill the WoT promises. Cyber-Physical Systems (CPS) propose a global approach that aims at taking into account such various concerns. CPS applications are designed in a domain-specific, performance-driven manner; however most of them are built with a bottom-up approach and their architectures are hardly reusable. Reusable frameworks, dedicated to Machine-to-Machine (M2M) and Internet of Things (IoT) applications have been created, but they fail to meet all these challenges.

We think that WoT platforms should be able to mix these various concerns in a manner that allows all specialties to connect together and form complex and efficient, though well-architectured WoT applications. To this end, we introduced in [50] the notion of avatar developed in the ASAWoO project.[1] An avatar represents the software part attached to a thing in a CPS-inspired approach. It embeds the components that implement the concerns required for the thing to participate in WoT applications through standard Web interfaces. In this chapter, we detail the notions of avatar and avatar-based WoT infrastructures that gather the common characteristics of WoT platforms able to expose things on the Web and run WoT applications using avatars. We illustrate using a realistic scenario, how avatars address concerns at different levels, such as physical constraints (network disconnections), transversal requirements (contextual adaptation) and application-level processes (inter-avatar collaboration).

CHAPTER ORGANIZATION

The chapter structure includes:

- A **motivating scenario** in the smart agriculture domain, that highlights the need for avatar-based infrastructures in the WoT
- A **description of avatar-based WoT infrastructures** that depicts the different elements of our approach: avatar architecture, common features of WoT platforms able to deploy and run avatars, and structure of interoperable WoT applications
- A **focus on contributions in three different domains**: disruption-tolerant networks, contextual adaptation and multi-agent systems; each contribution includes its own problem description, contribution and evaluation
- A **general conclusion**, including a discussion and outlook on the future of the Web of Things and avatar-based architectures

5.2 MOTIVATING SCENARIO

The WoT infrastructures presented in this chapter have applications in multiple scenarios ranging from simple home automation to more complex smart factories and service robotics settings. In this chapter, we focus on a motivating scenario in agricultural robotics that leverages the different aspects of the proposed infrastructures.

Agricultural robotics is an important trend that aims at decreasing the reliance on large machinery and human labor in favor of fleets of small autonomous robots and distributed sensors working in collaboration [58]. The objective is twofold: im-

[1] http://liris.cnrs.fr/asawoo/.

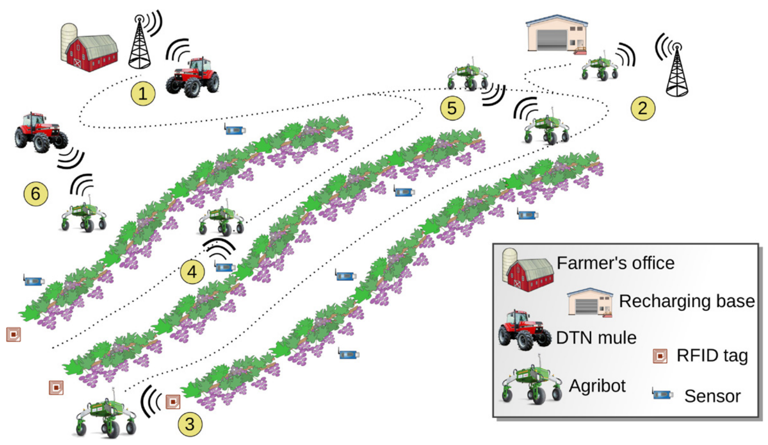

FIGURE 5.1

Illustration of the viticulture scenario

proving productivity and economic efficiency by reducing labor costs; decreasing the ecological footprint thanks to better energy efficiency and parcimonious usage of resources such as water, fertilizers and pesticides. The last aspect relies mostly on obtaining information for multiple sensors, and using smart processes for deciding how the robots should act upon these data. This is generally referred to as *precision farming* and we will refer to the involved robots as *agribots*. Such applications are already studied in the context of viticulture [54].

Fig. 5.1 depicts the application scenario that will be used for illustrating the different elements of avatar-based WoT infrastructures. This scenario takes place in a vineyard and is composed of the following elements:

1. The farmer's office from which the overall system is monitored and controlled. We assumed that it has enough available computational power for hosting a WoT infrastructure and that it provides a WiFi access point (potentially also connected to a larger network). The operator interacts with the system through a Web interface provided by the precision viticulture WoT application.
2. Recharging bases for the agribots that provide short-range radio connections.
3. Active RFID tags that mark rows and allow to know if they have been processed. These tags can be read by agribots or human workers, facilitating their interaction.
4. Low energy sensors spread all over the vineyard that monitor environmental parameters such as temperature, humidity, etc. They rely on inexpensive short-range radio communication to send their measurements.
5. Agribots of various natures that perform tasks such as weeding, irrigation, fertilization. They are equipped with sensing capabilities, computing power to autonomously achieve WoT application tasks, and network connectivity (RFID and short-range radio) to communicate with other connected things.

6. Agricultural machinery such as tractors that we herein consider as data mules. They use short-range radio to communicate with sensors and agribots and WiFi to exchange with the farmer's office.

In the next sections, we present these infrastructures and detail how they cope with constraints such as wireless network malfunctions, adaptation to resource and environment conditions, and autonomy and coordination constraints.

5.3 AVATARS AND AVATAR-BASED WOT PLATFORMS

The WoT imposes challenges regarding the representation, interoperability and collaboration of radically different things (in our scenario: RFID tags, low energy sensors, agribots, tractors, etc.), embedding various hardware (sensors, actuators, processing and storage units, network interfaces). We herein advocate *avatars* as virtual representations of things, so that a thing and its representation form a "Web-based cyber-physical object" that proposes homogeneous and user-understandable functionalities. In this context, WoT infrastructures are platforms able to deploy and execute WoT applications by creating, using and interlinking avatars. In this section, we overview WoT challenges and show how they can be tackled with our approach.

5.3.1 REQUIREMENTS FOR WOT PLATFORMS

In the IoT field, numerous platforms have been implemented [75]. Some of them rely on existing standards (e.g. ETSI's OneM2M[2]) related to the way they connect with things. However, there is currently no standard solution to address concerns at the level of WoT software platforms. In [50], we proposed a list of design requirements for WoT software platforms to summarize the WoT main challenges. We herein complete and extend this list. A WoT platform should:

- **R1: discoverability** – allow to discover heterogeneous things [29,48], to be able to plug and unplug things to the platform
- **R2: connectivity** – take into account several communication models (request/response, message-oriented, event-based, publish-subscribe, streaming, etc.) in order to allow applications to interact with various things [40], as well as support connectivity disruptions for mobile wirelessly connected things [42]
- **R3: reactivity** – adapt its structure and behavior to its environment at runtime, such as the CityPulse[3] platform or the work in the INCOME project [4], to react to changes in the environment
- **R4: safety** – be reliable and secure so that things and applications are harmless and avoid privacy issues [29,35]

[2]http://www.onem2m.org/.
[3]http://www.ict-citypulse.eu/.

- **R5: interoperability** – allow WoT applications to run across heterogeneous objects [29,24], so that users can seamlessly interact with things
- **R6: delegation** – identify the most suitable location to execute each code module and deploy these modules on the thing processing unit or on a cloud infrastructure [48], instead of completely delegating computation tasks to cloud-based infrastructures (see IFTTT[4])
- **R7: scalability** – cope with high numbers of things, heavy calculation processes and/or high quantities of data [40]), as the number of things is expected to increase
- **R8: collaboration** – allow a set of things to exhibit a collective behavior [18], as seen in the SensorMeasurement[5] framework, to achieve complex functionalities
- **R9: usability** – provide high-level services, so that applications match end-users' needs [30,10]
- **R10: marketability** – design software applications and components that implement different functionalities for heterogeneous things, so that developers and industrial companies can distribute them on open online marketplaces.[6]

To the best of our knowledge, a WoT platform able to handle all these requirements for a given WoT application is missing. We think that it is possible to define such comprehensive yet realistic infrastructures for the WoT by introducing an intermediary abstraction level between the things and the platform. This way, a WoT infrastructure can delegate the management of different requirements to software artifacts at this intermediate level.

5.3.2 RELATED WORK

Since the development of Programmable Logic Controllers and robots, programming elements have been associated with things. With the advent of (wireless) networked communications, they have evolved to distributed control systems, embedded systems and more recently ambient intelligence and distributed robotics [25]. The IoT is a direct consequence of this evolution and aims at exploiting Internet's communication capabilities, nearly unlimited computing power of cloud infrastructures and modern user interfaces to provide end-users with helpful applications. The WoT builds on top of the IoT and promotes the use of Web standards.

But as soon as physical things come into play, programming such applications becomes more complex and less deterministic. Indeed, sensor data suffer from imprecisions, actuators require feedback loops, time-critical and synchronization processes need constant attention and network communications may lose data or get interrupted. CPS upholders claim that such difficulties originate from physical phenomena and must be modeled together with computer-based models and processes [44]. This

[4]https://ifttt.com/.
[5]http://sensormeasurement.appspot.com/.
[6]http://www.compose-project.eu/sites/default/files/publications/COMPOSE_v2_factsheet.pdf.

way, embedded system applications will allow to take into account various concerns such as reliability, safety, adaptability, scalability and usability.[7] Among the numerous CPS architectures that have been designed, many of them are component-based and include software entities that model these concerns. Some of them include semantic languages to manage their interaction workflows [3,55]. Others took advantage of the multi-agent paradigm to build more autonomous and scalable CPS [67] and mixed this paradigm with semantic technologies [45]. In these works, agents embed the algorithms that control the things and are able to communicate together to perform collaborative tasks. However, these tasks are implemented in the frameworks at design time, which makes them hardly reusable across applications.

In the WoT community, we proposed in [33] the notion of avatar to denote software artifacts attached to a thing and aggregating the necessary code to execute WoT applications. Avatars are software agents that allow collaboration between things and distribution of application code between avatars. As detailed below, avatars rely on an internal component-based architecture, so that all necessary concerns from a CPS point of view can be described. More recently, the World Wide Web Consortium (W3C) Web of Things Interest Group[8] (WoT IG) proposed the notion of "Servient" that is currently being defined to standardize software objects in WoT applications. Servients are very close to avatars: they provide access to things, can be executed on them, on gateways or on cloud infrastructures and can interact with other servients. Both also rely on semantic technologies to exchange machine-understandable data. As WoT standards must cope with a variety of use cases and platforms, servient architecture only specifies building blocks ("runtime environment", "resource model", etc.). Avatars can be seen as a specialization of servients, more focused on WoT application deployment and execution, and relying on a component-based architecture to take advantage of advances related to specific concerns and requirements in various fields.

5.3.3 AVATARS

Some components of the avatar architecture (Fig. 5.2) are dedicated to thing control and others implement the autonomous, self-adaptive and collaborative behavior of avatars. The physical setup is decoupled from its logical architecture: an avatar can dynamically adapt the distribution of its components to different locations (see below) to improve their efficiency. We grouped the avatar components in 8 functional modules.

The **Core Module** includes components that are used in several steps of the avatar lifecycle. The component deployment manager defines which avatar components will be instantiated wrt. the thing capabilities, and where.[9] Each avatar embeds a Rea-

[7]https://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503286.

[8]https://www.w3.org/WoT/IG/.

[9]Avatar components can be located on the thing if it has enough computing capabilities or for time-constrained code modules, on the gateway for processes that involve inter-avatar communication, or on

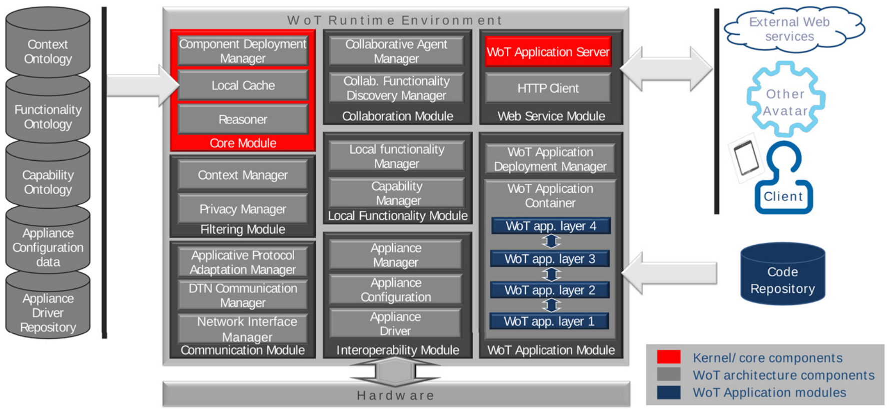

FIGURE 5.2

Architecture of the avatar software platform

soner, used by other components to process semantic information pertaining on the capabilities, functionalities and context. So is the Local Cache, that stores semantic information from diverse sources (thing, repositories, external context) and reflects the current state of the avatar. In particular, the cache loads concepts from the semantic repositories, in order to make them available to other modules through the reasoner, as shown in Section 5.5. This module is essential to address the multiple concerns targeted by the application through the avatar, while avoiding allocating unnecessary resources. As such, it participates in addressing most of the requirements, and especially (R6).

The **Interoperability module** provides the other avatar modules with a uniform interface to interact with the thing it is attached to (R1, R5). This interface consists of a set of *capabilities* that represent the thing API. It loads drivers from a platform repository and uses them to identify the communication schemes understood by the thing; eventually, it uploads onto the thing the appropriate configuration.

The **Filtering module** restricts functionality exposition and data exchanges. If, for privacy or security reason, some functionalities should not be achieved by the avatar, they will be filtered by the Privacy manager. The Context Manager has a more complex role, which is explained in Section 5.5.

The **Communication module** ensures reliable communication with the thing. It selects the appropriate network interface (Ethernet, Wi-Fi, Zigbee, etc.) and protocols (CoAP, HTTP, etc.) according to communication purposes and performance needs (throughput/energy consumption). It also supports connectivity disruptions, as explained in Section 5.4 (R4).

the cloud for calculation-intensive processes. This way, application components that model different CPS aspects and address different application concerns can be executed at an optimal location.

The **Web service module** allows avatars to communicate with other avatars and with the external world wrt. Web standards. By this means, avatars can: interact with the WoT platform to query repositories, respond to client requests regarding the functionalities they expose as RESTful resources, exchange data with other avatars to achieve collaborative functionalities and query external Web services to enrich their own data.

The **Local Functionality module** handles high-level *functionalities* achievable using the thing capabilities[10] (R9). It relies on semantic technologies to map the thing layer (capabilities) with the application layer (functionalities) in a declarative and loosely coupled manner, ensuring application interoperability with various things [51] (R5). When the avatar is created, the CapabilityManager queries the Interoperability module for the thing capabilities and the platform capability ontology for their semantic descriptions. It is queried by the LocalFunctionalityManager, which also loads the descriptions of functionalities and uses the reasoner to infer the avatar local functionalities.[11] For each inferred functionality, the LocalFunctionality-Manager queries the Context Manager to decide if it should be exposed to clients. Exposed functionalities are bound to a registry, so that users and other avatars can find them.

The **Collaboration module** handles functionalities that require collaboration between several avatars. The CollaborativeFunctionalityDiscoveryManager queries the reasoner as described above to identify, from the local functionalities, in which higher-level ones it could participate. Then, it queries the platform functionality directory to search for the locally missing functionalities. If such functionalities are available from other avatars, it calls the Collaborative Agent Manager, which handles negotiation with these other avatars, as explained in Section 5.6 (R8). Here, (R5) and (R9) are addressed at a multi-thing level.

The **WoT Application module** provides and controls "WoT application containers" that execute code modules implementing the different aspects of a WoT application (R9). Such containers can be replicated on the thing, on the gateway and on the cloud infrastructure thanks to the deployment manager, so that modules are executed on the appropriate location (R6).

5.3.4 AVATAR-BASED INFRASTRUCTURE

As requirements R1 to R4 and R7 are also addressed by IoT platforms, existing IoT solutions can be used as a lower layer to connect things, and WoT platforms implemented as "WoT application servers" on top of these solutions. To allow deploying and running WoT applications on ubiquitous computing environnements, WoT plat-

[10]Functionalities provide user-understandable compositions of capabilities. For instance, a user will prefer to tell a robot to move to another part of the field, rather than to pilot each of its wheels individually.

[11]Inference processing relies on the Capability and Functionality classes, and relationships between them, expressed in our own OWL (http://www.w3.org/TR/owl2-overview/) vocabulary. Individuals expressed in other vocabularies [31] can be used and "rdf:typed" as capabilities or functionalities.

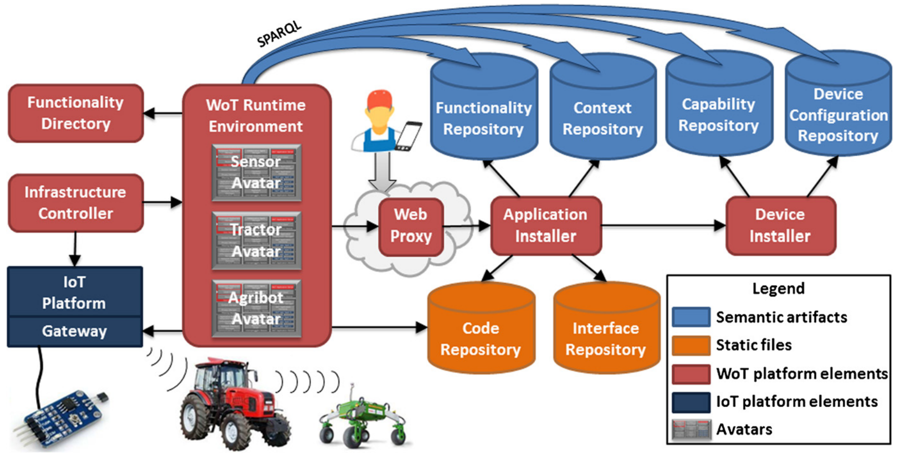

FIGURE 5.3

An Avatar-based infrastructure for the WoT

forms must provide access to information and knowledge storage facilities and to additional computing power (cloud). As we herein promote avatar-based WoT platforms, they should also support managing, executing, and (de)serializing avatars. The serialization mechanism allows scaling horizontally by replicating platforms and moving avatars between them. Vertical scaling is ensured by our multi-layer infrastructure (thing, gateway, cloud) (R7). Fig. 5.3 depicts the infrastructure of such platforms.

The main elements of this infrastructure are the *Infrastructure Controller* and *WoT Runtime Environment*. The former interacts with the IoT platform and is in charge of deciding to create,[12] update[13] or remove avatars, as things are plugged and unplugged from the IoT layer (R1, R3). The latter is the container that isolates avatars (R4) and handles their lifecycle. For performance reasons, it is also connected to the gateway, so that avatars can directly interact with things without traversing the IoT platform stack at each request (R2). Inside the container, avatars can also: access external Web resources through the *Web Proxy*, share information about the functionalities each of them exposes using the *Functionality Directory*, query the different *Semantic Repositories* to access the semantic descriptions they need to operate, and retrieve application code modules in the *Code Repository*. The *Device Installer* and *Application Installer* are in charge of populating the different repositories and are independent of the notion of avatar. Users securely interact with the platform to download thing

[12]Each avatar is built so that it can access the thing through the gateway. The thing capabilities are injected and the avatar components instantiated.
[13]By periodically checking the IoT platform.

drivers, install WoT applications from a marketplace (R10) or execute these applications using their Web browsers, through the proxy that blocks unidentified incoming requests (R4).

5.3.5 WOT APPLICATIONS

In order to ease WoT application design (R11) and keep it independent from the characteristics of available things (R5), a WoT application only deals with the functionality level. Hence, it mainly describes a hierarchy of functionalities, the end nodes of which graph are terminal functionalities (i.e. that have to be implemented by a thing capability[14]), and all others nodes are composed functionalities (i.e. that require sub-functionalities and query them using code modules). Some of these composed functionalities may require the capabilities of several things, and therefore, a collaboration between several avatars (R8). The top-level functionality then corresponds to the application that the end-user wishes to use (R9).

A WoT application is packaged in a compressed file, composed of: a *Manifest* file, describing its contents; the above mentioned *hierarchy of functionalities*; the *Code modules* corresponding to the algorithms that implement composed functionalities[15]; the *application context model*, containing a semantic description of the application domain and a set of adaptation rules (see Section 5.5); a set of static files that constitute the *application interface* and allow end-users to execute and control the application through their Web browser by querying avatar functionalities using Web standards (RESTful resources, WebSockets, etc.).

5.3.6 VALIDATION PROTOTYPE

The above sections show that all requirements are met by the different elements of our avatar-based WoT infrastructures. In order to validate our approach, we implemented its different components as follows. For the **IoT platform**, we chose OM2M,[16] as it relies on ETSI standards,[17] is supported by the Eclipse community and is extensible. We actually extended it by developing a module capable of introspecting things and sending semantically annotated capabilities.[18] Two versions of those **semantic annotations** have been experimented: one relying on Java annotations and one using JSON-LD files served as RESTful resources using the Hydra specification.[19] A prototype of the latter has been presented in [49], to demonstrate how avatars can

[14]Capabilities are semantically mapped with these terminal functionalities during avatar initialization.

[15]The R5 (interoperability) requirement imposes to these applications to be generically described, in order to be deployable and executable on diverse thing setups. Hence, we recommend describing them using declarative languages, such as state chart of finite state machines – and at installation time – pre-transpiling them into the languages of the thing, gateway and cloud platform they can be deployed on.

[16]http://www.eclipse.org/om2m/.

[17]http://www.etsi.org/technologies-clusters/technologies/m2m.

[18]http://liris.cnrs.fr/asawoo/doku.php?id=cima.

[19]http://www.hydra-cg.com/.

use semantic reasoning to dynamically compose functionalities. The **ASAWoO platform** has been developed in Java and runs avatars inside an OSGi (Felix[20]) container. Avatar components are implemented as OSGi services. Some of them are necessary for the avatar to function, and others depend on the concerns addressed by the application. The next sections highlight our contributions to certain concerns and detail how a given concern can be modeled and implemented in an avatar component.

5.4 DISRUPTION-TOLERANT COMMUNICATIONS

It is quite common that things involved in WoT applications cannot be connected permanently, firstly for energy saving reasons but also because of mobility. Such things – which are often equipped with short-range radio interfaces (Wi-Fi, Buetooth, ZigBee, etc.) – can be connected to other things or to Internet gateways while they are moving or when mobile things are passing near them. As shown in the scenario depicted in Section 5.2, communication links of such things are thus subject to frequent and unpredictable disconnections.

To cope with this issue, the communication module of avatars implements disruption-tolerant communication techniques to transport HTTP or CoAP requests and responses in partially or intermittently connected networks. These techniques do not assume that there always exists an end-to-end path between two things in the network. Mobile things can store messages, carry these messages while moving and deliver them to other things when possible.

5.4.1 RELATED WORK

The provision of REST services in partially or intermittently connected networks has been addressed in a few number of works, most of them in opportunistic networks. Opportunistic networks can be considered as a sub-category of disruption-tolerant networks, since only opportunistic contacts between mobile things are exploited to transfer data. In [20,56], the authors propose analytical models to determine the optimal number of parallel executions required to minimize the service time, without saturating the computational resources of the service providers, as well as to select the best service composition among alternative compositions based on the local knowledge of the network collected by a node through its opportunistic contacts with other nodes. The exploitation of the presence of several providers in the network and of their parallel invocation has also been investigated in [46], but with a publish/subscribe and content-based approach. The objective is to reduce the service delivery delay, and thus to provide a better response time to end-users. To reduce the response time, intermediate nodes can be used as service proxies has shown in [43]. Intermediate nodes are expected to respond on behalf of service providers, as long as that

[20]http://felix.apache.org/.

these intermediate nodes have in their local cache the responses for the requests they receive and the services are stateless-services. Intermediate nodes do not forward the requests towards the providers, but send back the responses to the clients directly. This solution allows to reduce drastically the network load and the response time. All these research works address the service provision in general, but do not conform to the REST architectural style, that ensures its performance, scalability and simplicity [23].

5.4.2 OVERVIEW OF THE RESTFUL DTN COMMUNICATION SUPPORT

The avatar communication module (CM) complies with the six constraints below – the sixth one is optional – defined by REST. Avatars are indeed designed as a set of distributed clients and stateless services, that interact through well-defined schemes (i.e. negotiation, functionality calls). Since the communication path between a service provider and its clients may be subject to disruptions, maintaining a session with a client can become difficult for a provider. It is consequently preferable to maintain a state on the client side. This is thus consistent with the REST approach, which is based on a loosely-coupled *client/server* (constraint 1) architecture where servers do not maintain any session state (servers must be *stateless*) (constraint 2) and are accessed through *uniform interfaces* (constraint 3).

To improve overall scalability, REST promotes a *layered system* (constraint 4) and *cacheable* responses (constraint 5). Layers consist in intermediate servers in charge of non-functional concerns such as security, load balancing or shared caches provision. By implementing the "store, carry and forward" principle, the CM stores both the service requests and service responses that have been sent in the network in the cache of clients and of intermediate nodes. Following a proxy-based approach, intermediate hosts can respond on behalf of a server if they have the response in their local cache, when this one is still valid. Such an approach allows to improve the performance and the scalability of the system, because it naturally performs load balancing and data caching on intermediate hosts, and thus fulfills the two above-mentioned requirements.

WoTApps can be partially or fully developed in Javascript, thus allowing to execute on the client side a part of the application and to reduce the computation load on the things, which complies with REST optional *code-on-demand* constraint (constraint 6).

The RESTful disruption-tolerant communication part of CM is designed to be as versatile as possible. It allows to invoke services using the HTTP and CoAP application-level protocols. It is composed of two main elements, namely an HTTP/CoAP proxy and a DTN adapter (see Fig. 5.4). Thanks to this proxy, programmers can develop HTTP and CoAP WoTApps using regular HTTP and CoAP libraries. Moreover, standard HTTP and CoAP servers do not need to be modified. This proxy can also invoke remote REST services using Internet-legacy routing protocols (i.e., TCP/IP). Application-level messages (i.e., HTTP and CoAP messages) can be encapsulated in UDP datagrams, in TCP segments or in messages of a given

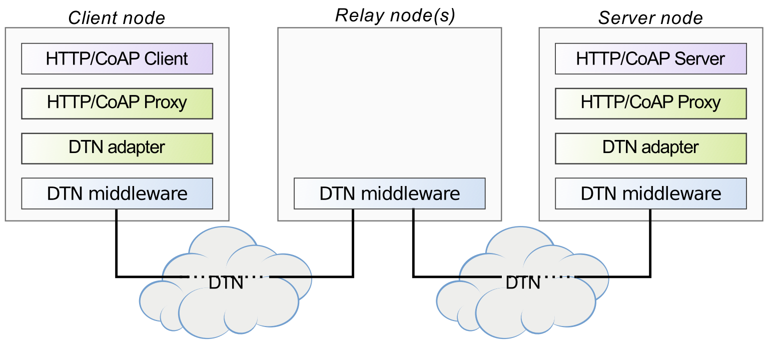

FIGURE 5.4

Overview of the architecture of the communication module

disruption-tolerant communication middleware in order to be transmitted to their destination. Different wireless technologies (e.g., Bluetooth, Wi-Fi) can be used to communicate with things.

As for the DTN adapter, it binds the proxy and the disruption–communication middleware in charge of forwarding messages in the network. Hence, the DTN adapter depends of the underlying communication system and is specifically developed for each different system. The Bundle Protocol (BP) [61], which is the standard message-based protocol over the DTN architecture [17], has been chosen as the default disruption–communication system in our current implementation. Another implementation has also been done using an opportunistic communication middleware we have developed, and which is called C3PO [42].

5.4.3 COMMUNICATION MODES

Besides providing the traditional point-to-point client/server communication model used in the Web, CM proposes alternative communication models, that are not necessarily relevant for traditional Web applications, but that are suitable for the WoT, such as anycast, multicast and broadcast transmission models. Disruption-tolerant communication systems implementing a multiple copy forwarding strategy can take advantage of these message transmission models to increase the message delivery probability and to reduce the response time. Indeed, if it exists several providers offering the same service in the environment, these providers can indifferently be invoked with the same request. If an anycast communication model is used, the communication system will only return the first received response to the client. Similarly, several sensors can simultaneously be invoked using a multicast transmission model without naming them explicitly. All responses returned by these sensors will be transmitted to the client.

In order to remain consistent with the RESTful approach, these different transmission models are specified in the scheme of the URIs used to access REST Web

Table 5.1 Examples of URIs supported by ADTRS

	URI	Method	Parameters
1	coap+dtn://agribot7/JobManager/back_to_charging_station	POST	
2	coap+dtn+acast://agribots/JobManager/weed	POST	rows=1–3
3	http+mcast://soilsensors/moisture	GET	

services. The first part of the scheme indicates the application-level protocol (i.e., HTTP, HTTPS, or CoAP) used to communicate with a remote service. +dtn must additionally be specified in the scheme if CoAP and HTTP messages must be forwarded by a disruption–tolerant communication system. By default, messages are transmitted using a unicast communication model. +acast, +mcast and +bcast specify respectively that an anycast, a multicast and a broadcast transmission model must be used in the forwarding process.

Table 5.1 gives examples of URIs that could be used in the smart vineyard scenario presented in Section 5.2. URI number 1 could be sent in unicast to order robot whose ID is agribot7 to go back to its recharging base after its current mission. Concretely, such an order is given by submitting a new job back_to_charging_station to the job manager of the avatar. This job manager is exposed as a Web service and is invoked via a POST CoAP method. Similarly, URI number 2 could be used to ask robots belonging to the agribots anycast group to weed around vine feet from rows 1 to 3 (additional parameters specified in the body of the POST request). Finally, URI number 3 allows to get the soil moisture from avatars of sensors monitoring the soil. As sensor avatars are hosted in a cloud infrastructure, they do not need to be reached in a disruption-tolerant way.

5.4.4 NON-FUNCTIONAL PARAMETERS FOR DELAY-TOLERANT COMPUTING

In delay-tolerant networks, service messages are forwarded following the “store, carry and forward” principle. The propagation delay and transmission area of messages is likely to be bounded not only to fulfill application requirements, but also to reduce the network load. Four additional parameters can be specified by service clients and service providers regarding the service delivery conditions on the one hand, and that can be exploited by disruption–tolerant communication systems in the message routing process on the other hand.

Caching parameter: Service clients can specify if their requests can be cached by intermediate nodes or not. If so, intermediate nodes will store in their cache both the request and the response associated with this request until they expire. Thus, they can reply later to a similar request sent by any node on behalf of the service provider (i.e., by returning immediately the cached response, instead of forwarding the request towards the service provider). For instance, environmental data such as air temperature at the vine feet, which is not likely to change frequently, can be cached.

Time parameters: Temporal constraints can be expressed in URIs as query strings. Two temporal constraints are considered: the message creation time, and the message expiration time, expressed relatively to the creation time.

In a service invocation request, these constraints express the fact that the client wants to get a response before the expiration time specified in the request message. The request can be forwarded in the network, and a provider of the service can answer to this request until it expires. When specifying time constraints for a response, these constraints express the lifetime of the response, and the validity duration of the data it contains. These temporal constraints are also used by the disruption–tolerant communication systems to determine how long a message can be stored in a cache or forwarded in the network.

Spatial parameters: A number of hops can additionally be specified in application-level messages to circumscribe at a coarse grain the area in which the messages can be forwarded, and to avoid that a message eternally roams in the network. Nevertheless, this limitation is not exact, and does not guarantee that a message cannot be transferred outside an expected area. As shown in [41], geographical areas can be specified in order to limit more precisely the propagation of messages in the physical environment.

Asynchronous communication: Service clients can add a callback parameter to define the URI that must be used by service providers to return the response. Clients can thus process the responses they receive asynchronously without being blocked by the reception of responses.

In our vineyard scenario, a sprinkler agribot that does not know the soil composition on the third row can ask other agribots to analyze it and send back the result. While waiting for the analysis, the agribot continues its current task (e.g., irrigating the first row). Once the soil composition analysis is received, the robot can process it and if the soil is dry, it will proceed to irrigate the row.

5.5 CONTEXT MODELING AND MANAGEMENT

An avatar must take decisions all along the lifecycle of the thing it extends, to adapt its behavior for different purposes (choose a protocol to communicate with the thing, a location to execute applicative components, etc.) and with respect to various parameters. Such parameters form the avatar context and come from various sources (thing sensors, environment, user's preferences, etc.). In this section, we build a multi-purpose context adaptation framework. We discuss related work on context modeling and adaptation to show the need for such framework, and present and evaluate our context meta-model and multi-level context adaptation solutions.

5.5.1 RELATED WORK

Context is often modeled as multi-dimensional views [60,79,1] that include environmental and geospatial information, such as location, co-location (i.e. what is nearby), time, etc. [59] but also privacy [21], computing resources (availability, remaining battery power) and network information (types of connection, services in reach, distances, disconnection rates) [47,53]. Most context-aware applications give a major importance to user profiles and preferences [14,74,11] sometimes combined with network elements [73,57,76]. Context helps users with decision making [13], based on different knowledge sources [12] described in situations [19,9]. Most context information is domain-specific [52], or related to the application architecture [72,39]. Hence, existing work highlights the diversity and complexity of context that includes heterogeneous information, thus motivating our need to build a single framework that deals with any kind of context information.

Numerous self-adaptive and autonomic systems support dynamic contextual adaptation in pervasive and mobile computing environments. Most are based on a control loop, such as the autonomic computing architecture MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) [32] or the Rainbow framework [26]. Reflexive mechanisms are also used for dynamic adaptation such as in the ReMMoC [28] and CARISMA [15] middleware platforms. These centralized, problem-specific solutions are not adapted to our distributed, multi-focus setup. Different work from the IoT community perform multi-level adaptation in environments composed of cloud infrastructure and things [5,2] as well as networked cyber-physical systems [65] but does not benefit from the advantages of the Web and the collaboration between things as we provide in our avatar-based approach [33].

5.5.2 PROBLEM STATEMENT

As seen above, existing work mostly bind context models to specific scenarios and do not design them with adaptation in mind. It is currently difficult to reuse or extend these models due to their specificity, and they sometimes do not cope with WoT application requirements. Furthermore, adaptation solutions proposed in the literature do not provide abstraction for extending things and enabling collaboration between them, or do not provide multi-purpose adaptation. Some of them are also not compatible with decentralized architectures. Thus, there is a need for a solution to support flexible and interoperable context models that can be reused across applications and use-cases, as well as autonomic, decentralized and multi-purpose context adaptation. The following questions, illustrated with our agricultural scenario, must be answered to provide multi-purpose adaptation:

1. **Which protocols should be used to communicate with things?** Agribots, machineries and sensors must exchange information using adequate protocols, with respect to their heterogeneity and the network status.
2. **Where should the application code be executed?** Applicative modules can be executed either locally on agribots or remotely on the office system, which depends on their computational resources, location and availability.
3. **Which local capability should be involved in a given high-level functionality?** To move across grapefruit lines, agribots can rely on processing data from their GPS sensors or embedded cameras.

4. **Which other avatars functionalities should compose a collaborative functionality?** A sensor can select an agribot to water specific parts of the field.
5. **Which functionality should be exposed to application end-users and to other avatars?** The watering functionality would not be displayed in case of drought.

We next present our solution to instantiate reusable and flexible context models, based on both the literature and a typical WoT application architecture. We then describe how the components that compose an avatar handle, update and query an instantiated context model in order to be able to respond to the above adaptation questions.

5.5.3 CONTEXT META-MODEL

The adaptation process relies on physical, security, privacy, or other constraints, which vary according to the use-case. We have designed a context meta-model that allow avatars to instantiate domain-specific models, capable of answering different adaptation questions for a given WoT application [69]. We have organized our meta-model into levels that characterize the different parts of a WoT application.

The *Physical* level describes things, their internal states and their physical (sensed) environment. The *Application* level describes the application architecture, its state, and its configuration (*e.g.* components or services, locally stored or distributed). The *Communication* level describes network link context between application and clients, things and data sources (*e.g.* network states, bandwidth, latency, connection types). The *Social* level includes cognitive aspects related to the client's environment, which involves roles, organizations, as well as behavioral concepts and type of users (human, software).

Our meta-model is a two-dimensional matrix, that crosses pre-determined levels with a set of application-defined *Dimensions* based on the literature. Thus, it not only provides flexibility to WoT application developers by allowing the creation of any number of dimensions, but also provides reusability across WoT applications through the level/dimension view.

5.5.4 MULTI-LEVEL CONTEXT ADAPTATION

The above context meta-model allows for a single adaptation mechanism to operate across application domains. Each instantiated model supports data integration and query answering. For each dimension, data integration applies *transformation rules* to contextual data (numerical values coming from sensors or external services) in order to turn them into high-level contextual representations corresponding to the four levels of the model. Transformation rules act as filters that allow these representations to change only when the data go under significant changes. Query answering relies on SPARQL SELECT queries to get the accurate information to each adaptation question. Then, making the adaptation decisions consists in applying *adaptation rules* on the SPARQL query results. Our avatar architecture includes a context manager that populates the context model and handle the adaptation questions, as defined in [69].

5.5.5 EVALUATION

We have conducted two types of evaluations. First, we have evaluated our meta-model in [69] according to the methodology proposed in [27] and have shown its relevance and reusability across different scenarios as well as its usability in terms of performance based on the DL language it relies on. Second, we have evaluated our adaptation process in [70]. This evaluation consisted in adapting the location of application code (question #2). This code performed semantic reasoning about 2 ontologies: Fipa-Device[21] with 126 entities (schema + axioms) and IoT-O[22] with 328 entities. We have used the HyLAR [68] architecture to migrate the code between a thing (laptop mocking a drone) and a cloud (virtual machine) and to measure reasoning times. Evaluation results detailed in [70] show that our solution globally improves avatar performances for the code execution adaptation question, and that reasoning is up to 82% more effective than fixed implementations in disrupted environments for an average domain-specific ontology. This shows that our solution enables multi-purpose adaptation of avatar-based WoT applications and allows developers to declaratively define adaptation processes using high-level concepts and standardized rules.

5.6 A SOCIAL VISION OF THE WEB OF THINGS

To execute a WoT application (*i.e.* a composition of functionalities), the end-user requests its top-level functionality to an avatar that exposes it. Each avatar is responsible for the correct execution of the functionalities it exposes. To do so, it satisfies its individual *goals*. A goal can be functional or non-functional. In the first case, it corresponds to the objective of a local (resp. collaborative) functionality, handled by the *Local Functionality Module* (resp. *Collaborative Functionality Module*). In the second, it is related to other concerns, such as energy management or response time to other functionalities. The hierarchy of functionalities can be considered as a global plan to orchestrate local functionalities and produce collaborative ones. An avatar decides to provide a collaborative functionality by semantically matching the requirements of the functionality with functionalities exposed by itself and other avatars. If all sub-functionalities are available, it can start a *coalition formation* process (Fig. 5.5). A coalition is a set of self-interested avatars that agree to cooperate for executing a task [36]. The result of this process is recruiting a set of avatars in a temporary *ad-hoc* cooperation to perform a collective goal. A collective goal is the objective of a coalition of avatars (namely, the functionality that requires the collaboration). In this section, we describe how avatars can autonomously expose and fulfill collaborative functionalities and therefore, WoT applications.

[21] www.fipa.org/specs/fipa00091/PC00091A.html.
[22] http://www.irit.fr/recherches/MELODI/ontologies/IoT-O.owl.

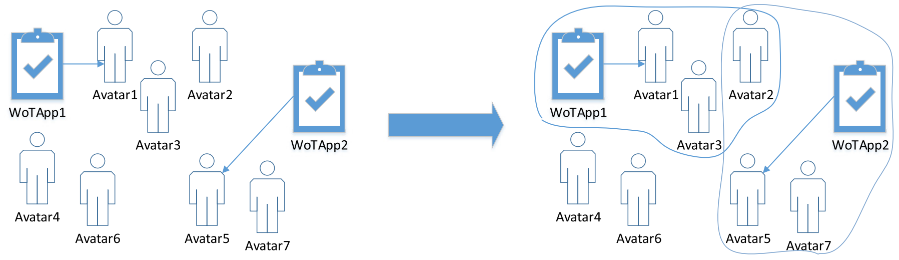

FIGURE 5.5

Coalition formation

5.6.1 TOWARDS SOCIAL CONSCIOUSNESS OF AVATARS

The smartness of a WoT system lies typically in its service composition, context-awareness, human-machine interaction, automation, semantic reasoning, data integration and analysis to name a few [8]. Still, we can more attractively exploit the potential of the Web of Things and go beyond its user-centric technical smartness. Currently, we are observing a progressive shift from things with a user-understandable smartness to things with an actual social consciousness [6]. Motivated by the same social concern, we believe that the technical smartness offered by WoT platforms and architectures is mature enough to take a further step in the evolution of everyday things into socially intelligent ones. Nonetheless, it remains difficult for a WoT system designer to manage or anticipate the heterogeneity and the complexity of interactions between things. Hence the need for a thing-centered social vision for the Web of Things. This vision cannot be easily achieved as many pieces are still missing.

While efforts to meet WoT specific challenges (listed above) are partially made, we still need an abstract high-level perspective. Recently, there has been a paradigm shift towards a social vision for the IoT/WoT. Many terms like “Social Web of Things”, “Social Internet of Things”, “Internet of Social Things”, “Smart-its”, “Everything is alive”, “Cyber Physical Social Systems”, “Wisdom Web of Things” have flourished attempting to make smart things as first-class citizens of the Web. However, little attention was paid to “human-inspired” social interactions between smart things. Approaching the social vision from a collective intelligence perspective [37,34,38], we consider smart things as (i) autonomous entities, i.e. operating and deciding independently according to their own agenda, (ii) exhibiting goal-directed behavior, (iii) and capable of cooperating to solve problems that go beyond the individual capabilities or knowledge of each entity.

5.6.2 COOPERATION AS AN INTENDED SOCIAL INTERACTION

Avatars can engage in many types of social interactions. The simplest form is, for example, communicative action such as sending requests or information. Cooperation is a very general form of interaction that is studied in multi-agent systems. It is established by the delegation/adoption of tasks, coordination of actions, and conflict resolution [16]. From an individual point of view, cooperation is a deliberate attitude of an avatar who decides to carry out a joint action with one or many avatars. More concretely, cooperation works through an avatar allocating a task (or sub-task) to another avatar via a specific request (offer, proposal, announcement, etc.) satisfying a commitment (help, contract, etc.) [16].

Avatars need to cooperate when they do not have the necessary skills or knowledge to accomplish their individual goals, to respond effectively to users' and other avatars' requests, but also to meet objectives of the overall system. The challenge of cooperation is the establishment and maintenance of mutually advantageous relations between avatars. This challenge is more present in the coexistence of WoT systems, unknown to each other, in the same physical environment. An avatar that belongs to one or more coalitions and that is requested by another avatar must ensure that the response to this request does not interfere in achieving its individual goals and those associated to its various coalitions.

There are several cases where "interferences" can occur:

- Incompatible goals between two avatars. An avatar A_1 delegates an action to an avatar A_2 with consequences on the environment that are not acceptable by A_2. It should be noted that this conflict has no place in a situation of competition or adversity between avatars, but occurs between avatars with collaborative attitudes.
- Cooperation with avatars having conflicting goals. When an avatar A_1 accepts requests from two avatars A_2 and A_3 in conflict, the external conflict between A_2 and A_3 will become an internal conflict to A_1. From a collective point of view, this interference transforms an external conflict between coalitions with incompatible goals into an internal conflict to the avatar agreeing to join these coalitions.
- Resource conflict between avatars. When two avatars need to use the same resource at the same time. In a cooperative setting, this interference can be considered as a scheduling constraint.

5.6.3 TOWARDS SOCIAL CONSCIOUSNESS OF AVATARS: UNDERSTANDING SOCIAL INTERACTIONS

In an open environment like the Web, systems are designed by different vendors, making it difficult for all the coexisting groups of socially conscious avatars to share the same mental attitudes (cooperative, cautious, selfish, precautious, etc.). Indeed, such coexistence can show a wide range of behaviors that may either fit the label of mutualism, cooperation, antagonism, parasitism, etc. One can also think about considering some kind of exploitive behaviour that smart things can encounter. This difference in attitudes and intentions of avatars influences the interactions between coexisting smart things. Therefore, an avatar must be able to understand the nature of its social interactions for a successful cooperation.

Identification of social interactions One of the most important prerequisites to implement a social vision of the Web of Things is to investigate the types of social

Table 5.2 Classification of Interaction Situations ([22])

Resources	Skills	Goals	Interaction Situation	
			Type	**Category**
Sufficient	Sufficient	Compatible	Independence	Independence
	Insufficient		Simple Collaboration	Cooperation
Insufficient	Sufficient		Obstruction	
	Insufficient		Coordinated Collaboration	
Sufficient	Sufficient	Incompatible	Pure Individual Competition	Antagonism
	Insufficient		Pure Collective Competition	
Insufficient	Sufficient		Individual Resource Conflict	
	Insufficient		Collective Resource Conflict	

interactions among things [6]. According to Ferber [22], except for independence situations where goals are compatible and the resources and skills are sufficient, the possible interaction situations are cooperation or antagonism (Table 5.2).

There have been efforts to present a taxonomy for relationships between smart things [7]: Parental object relationship, Co-location object relationship, Co-work object relationship, Ownership object relationship, Social object relationship. However, they are meant to classify agents into communities (smart things sharing the same manufacturer, environment, owner, or goal) so they can be easily navigable in social networks like the ones created by humans.

Intersubjectivity as a social process to understand social interactions Communication is the essence of multi-avatar systems. Indeed, expressing desires, setting goals and committing to joint actions require that all avatars communicate. The formation of coalitions needs, as well, some kind of communication. In this context, we point out a concept that is the basis of human communication called *intersubjectivity* [71]. Intersubjectivity – between two or more avatars in interaction – consists in considering the recognition of the intended meaning of actions and requests. From a multi-agent perspective, intersubjectivity refers to the interconnection of the self, the others and the environment. In other words, the avatar ability to take into account the mental states [62] of other avatars in its perception of the environment. Yet, we can never access the mental states of other avatars, we can only infer their existence based on what we observe, namely their performed actions, exchanged messages, requests, social links, etc. When an avatar is requested to contribute to a coalition, it is not enough to assess the cost of the provided services. The avatar should include the involvement of goals, both individual and derived from the coalitions in which the requester is involved. Additionally, in connection with its own coalitions, the avatar should construct a social representation of what we call its *Collective Identity*, *i.e.* any self-representational mode adopted by most of the avatars in a coalition, that they must integrate into their personal identity. Collective identity can simply be the coalition members and their common goal, or can be extended to role hierarchical structure between avatars.

5.6.4 INTENT RECOGNITION AS A MEANS FOR INTERSUBJECTIVITY

As we stated earlier, for a successful cooperation, an avatar needs to identify the type of interaction situations with its potential partners. We introduce intersubjectivity as a social process to this identification. This is where intent recognition [66] will play a very important role. There are many sub-areas that contribute to the growing field of recognizing other agents' behavior: intent recognition, goal recognition, plan recognition, activity recognition, mental state abduction, etc. But they share similar challenges and applications. This division is not meant to separate recognition into isolated research areas, they are closely related and they can transform into one another. Better than that, the resolution of one can automatically resolve others. For example, many works use goal and intention interchangeably. Likewise, intent recognition and plan recognition may have the same meaning, while some other works consider goal/intent recognition as a particular case of plan recognition.

Intent Recognition has proven useful for many research areas such as robotics, ambient intelligence, computer vision, human-machine interaction, traffic monitoring, military applications, video games and many others [66]. Several difficulties are encountered like the cost of handcrafted plan libraries [77], and partial observability of agents actions. In the absence of a plan library, agents can use action models [78] to recognize the plans that may be executed by an observed team. These techniques require a team trace, an initial state and a goal. The objective is to generate all possible plans to go from the initial state to the goal state, and keep the plans corresponding to the observed team trace. There are other techniques that do not rely on plan libraries or action models. [64] offers an approximation of a team intention considering the activities that take place between the roles of agents in a hierarchical structure (called *RoleGraph*). Agents can then match those activities between roles with rolegraphs that they already know.

5.6.5 APPLICATION TO THE VITICULTURE SCENARIO

We consider a set of agribots, RFID tags, sensors and a tractor in a vineyard. These smart things participate – through their avatars – in the satisfaction of three functionalities, grouped under a *Vineyard Management* WoT application (Fig. 5.6).

A user requests a moisture sensor avatar to achieve a *Watering* functionality, which requires several sub-functionalities, among which *water-vineyard*. To recruit an avatar that performs this functionality, the moisture sensor avatar initiates the Contract Net Protocol [63] to form a coalition. This protocol comprises four steps: (i) **Announce**: the moisture sensor avatar requests every avatar exposing a *water-vineyard* functionality, (ii) **Proposition**: the avatar of an agribot already involved in a *3D Data Analysis* functionality receives the request to provide *water-vineyard*. It knows that participating to a coalition for realizing another functionality may interfere in achieving the objective of *3D Data Analysis*. However, it can postpone the accomplishment of *3D Data Analysis* for a functionality of higher priority. For this reason, the agribot avatar requests an external service to use an intent recognition algorithm (like MARS [77] and DARE [78]) to identify the moisture sensor intention.

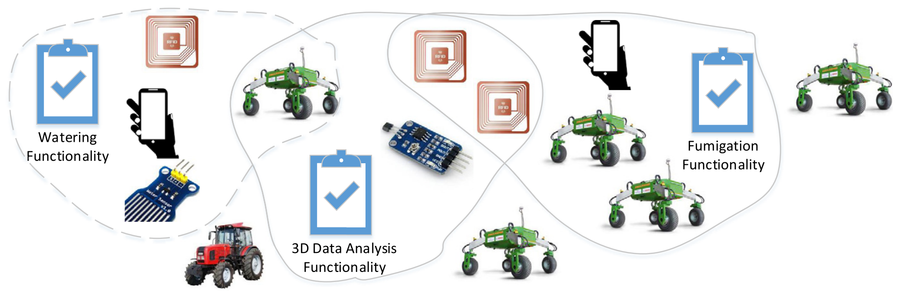

FIGURE 5.6

Coalition formation

After having determined that the moisture sensor avatar is likely seeking to realize the priority *Watering* functionality, the agribot avatar makes the decision to join the moisture sensor avatar coalition. It abandons its original plan of realizing *3D Data Analysis*, saves it for later, and responds positively to the sensor avatar proposal, (iii) **Decision**: after analysis of the positive responses, the moisture sensor avatar chooses the agribot avatar which seems to be the best candidate. (iv) **Contract**: the contract between the moisture sensor avatar and the agribot avatar that accepted to join the coalition is made. The sensor avatar can then inform the user that the collaborative functionality is being achieved.

5.7 CONCLUSION

Web of Things applications target various domains, address different concerns and will only meet global adoption by offering smart behaviors and services to their users. WoT platforms need to be sufficiently flexible to both enable interoperability between heterogeneous things and host such diverse and complex Web applications. In this chapter, we claim that such flexibility requires an abstraction layer between things and applications, and that this layer should represent things in a convenient, extensible and standard manner. To this end, we present the notion of avatar developed in the ASAWoO project and highlight several contributions.

Avatars are autonomous software artifacts that apply semantic technologies to both achieve interoperability among things and expose high-level RESTful functionalities from combinations of elementary thing capabilities. They can be connected to things using different network protocols withstanding connectivity disruptions. They can be executed and can deploy application code on the thing itself, on the gateway on which it is connected or on cloud infrastructures. They integrate a multipurpose adaptation mechanism that allows optimizing functional and non-functional processes. They can autonomously collaborate and form coalitions to provide applications composed with functionalities from avatars of multiple and heterogeneous things.

All these possibilities are provided in the avatar component-based architecture described in this chapter. We show how these contributions are put into work in the avatar architecture and more generally, how this architecture is able to efficiently handle various concerns at different levels: thing-specific constraints (*e.g.* network management), transversal requirements (*e.g.* adaptation) or application behavior (*e.g.* collaboration). This architecture has been designed to be extensible, and additional components are currently being implemented, such as a (de)serialization mechanism that will allow avatars to be saved, restored and moved between WoT platforms.

Together with the avatar architecture, we have specified an infrastructure that gathers the characteristics of avatar-based WoT platforms. Such platforms can be implemented on top of existing IoT solutions. They add support for avatar life-cycle, and act as "WoT application servers". We have also specified how to package WoT applications in a way that copes with WoT requirements and allows application designers to address domain-specific concerns while reusing existing functionalities. However, the notion of avatar does not appear in an application description; this saves developers' time and allows other WoT application specifications to be imported in our infrastructure. In any case, deploying an application in an avatar-based platform seamlessly leverages all advantages of the concerns already addressed in our avatar approach (interoperability, fine-grained deployment, network control and adaptation, collaboration). This is particularly useful when using complex things (such as robots) or heterogeneous setups. We have implemented our approach and describe the different parts of our prototype. We illustrate our contributions with a scenario inspired from agricultural robotics.

The Web of Things is currently a relatively new field. The community is large and expects promising business opportunities. Therefore, it will for sure continue to evolve and new standards will emerge. It has to deal with a multi-layered vertical stack and multiple use cases, and to combine contributions from different disciplines. This is why we have built our approach on stable and comprehensive visions from as various disciplines as cyber-physical systems, software engineering, semantic Web and artificial intelligence. In this chapter, we show first that the notion of avatar helps building multidisciplinary, efficient and user-understandable WoT applications, and second that the theoretical foundations of our approach are reusable across the languages, standards and components implemented in the avatars.

REFERENCES

[1] Abowd GD, Dey AK, Brown PJ, Davies N, Smith M, Steggles P. Towards a better understanding of context and context-awareness. In: Handheld and ubiquitous computing. Springer; 1999. p. 304–7.

[2] Alaya MB, Matoussi S, Monteil T, Drira K. Autonomic computing system for self-management of machine-to-machine networks. In: Proceedings of the 2012 international workshop on self-aware Internet of things. New York, NY, USA: ACM; 2012. p. 25–30.
[3] Arbab F. A channel-based coordination model for component composition. Rep Softw Eng 2002;3:1–35.
[4] Arcangeli J-P, Bouzeghoub A, Camps V, Canut M-F, Chabridon S, Conan D, et al. Income–multi-scale context management for the internet of things. In: Ambient intelligence. Springer; 2012. p. 338–47.
[5] Athreya A, DeBruhl B, Tague P. Designing for self-configuration and self-adaptation in the internet of things. In: First international workshop on internet of things (C-IOT), 9th international conference on collaborative computing: networking, applications and worksharing (CollaborateCom 2013). Oct 2013. p. 585–92.
[6] Atzori L, Iera A, Morabito G. From "smart objects" to "social objects": the next evolutionary step of the internet of things. IEEE Commun Mag 2014;52(1):97–105.
[7] Atzori L, Iera A, Morabito G, Nitti M. The social internet of things (siot)–when social networks meet the internet of things: concept, architecture and network characterization. Comput Netw 2012;56(16):3594–608.
[8] Bandyopadhyay D, Sen J. Internet of things: applications and challenges in technology and standardization. Wirel Pers Commun 2011;58(1):49–69.
[9] Bazire M, Brézillon P. Understanding context before using it. Modeling and using context. 2005.
[10] Bischof S, Karapantelakis A, Nechifor C-S, Sheth A, Mileo A, Barnaghi P. Semantic modelling of smart city data. In: W3C workshop on the web of things. Jun 2014.
[11] Brézillon P. Context in artificial intelligence: II. Key elements of contexts. Comput Artif Intell 1999:1–27.
[12] Brézillon P. Representation of procedures and practices in contextual graphs. Knowl Eng Rev 2003:1–26.
[13] Brézillon P, Pomerol J. Contextual knowledge sharing and cooperation in intelligent assistant systems. Le Travail Humain 1999:1–33.
[14] Cao H, Hu DH, Shen D, Jiang D, Sun J-T, Chen E, Yang Q. Context-aware query classification. ACM Sigir 2009;106(3):3.
[15] Capra L, Blair GS, Mascolo C, Emmerich W, Grace P. Exploiting reflection in mobile computing middleware. SIGMOBILE Mobile Comput Commun Revue Oct. 2002;6(4):34–44.
[16] Castelfranchi C, Falcone R. Conflicts within and for collaboration. In: Conflicting agents. Springer; 2002. p. 33–61.
[17] Cerf VG, Burleigh SC, Durst RC, Fall K, Hooke AJ, Scott KL, et al. Delay-tolerant networking architecture. In: IETF RFC 4838. Nov. 2007.
[18] Cervantes F, Occello M, Ramos F, Jamont J. Toward self-adaptive ecosystems of services in dynamic environments. In: Proceedings of the international conference on systems science 2013, ICSS 2013. Advances in intelligent systems and computing, vol. 240. Springer; 2014. p. 671–80.
[19] Chaari T, Laforest F, Flory A, Einstein AA, Cedex V. Adaptation des applications au contexte en utilisant les services web. In: Proceedings of the 2nd French-speaking conference on mobility and uibquity computing – UbiMob '05. 2005. p. 3–6.
[20] Conti M, Marzini E, Mascitti D, Passarella A, Ricci L. Service selection and composition in opportunistic networks. In: 9th international wireless communications and mobile computing conference (IWCMC 2013). July 2013. p. 1565–72.

[21] Dey AK, Salber D, Abowd GD, Futakawa M. The conference assistant: combining context-awareness with wearable computing. In: The third international symposium on wearable computers, 1999. Digest of papers. IEEE; 1999. p. 21–8.

[22] Ferber J. Multi-agent systems: an introduction to distributed artificial intelligence, vol. 1. Reading: Addison-Wesley; 1999.

[23] Fielding RT. Architectural styles and the design of network-based software architectures. Ph.D. thesis. Irvine: University of California; 2000.

[24] Fuhrhop C, Lyle J, Faily S. The webinos project. In: Proceedings of the 21st international conference companion on world wide web. ACM; 2012. p. 259–62.

[25] Galloway B, Hancke GP. Introduction to industrial control networks. IEEE Commun Surv Tutor 2013;15(2):860–80.

[26] Garlan D, Schmerl B, Cheng S-W. Software architecture-based self-adaptation. In: Autonomic computing and networking. Springer; 2009. p. 31–55.

[27] Gómez-Pérez A. Knowledge sharing and reuse. Handbook of applied expert systems. 1998. p. 10–1.

[28] Grace P, Blair GS, Samuel S. Remmoc: a reflective middleware to support mobile client interoperability. In: On the move to meaningful Internet systems 2003: CoopIS, DOA, and ODBASE. Nov. 2003. p. 1170–87.

[29] Guinard D, Trifa V, Mattern F, Wilde E. From the internet of things to the web of things: resource-oriented architecture and best practices. In: Uckelmann D, Harrison M, Michahelles F, editors. Architecting the internet of things. New York, Dordrecht, Heidelberg, London: Springer; 2011. p. 97–129.

[30] Gyrard A. 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. p. 371–6.

[31] Gyrard A, Serrano M, Atemezing GA. Semantic web methodologies, best practices and ontology engineering applied to internet of things. In: 2015 IEEE 2nd world forum on Internet of things (WF-IoT). IEEE; 2015. p. 412–7.

[32] IBM. An architectural blueprint for autonomic computing. 2004.

[33] Jamont J, Médini L, Mrissa M. A web-based agent-oriented approach to address heterogeneity in cooperative embedded systems. In: Pérez JB, Rodríguez JMC, Mathieu P, Campbell A, Ortega A, Adam E, Navarro E, Ahrndt S, Moreno MN, Julián V, editors. The 12th international conference on practical applications of agents and multi-agent systems PAAMS 2014. Advances in intelligent systems and computing, vol. 293. Salamanca, Spain: Springer; Jun 2014. p. 45–52.

[34] Jamont J-P. Multi-agent approach, models and tools to collective cyber-physical system engineering. In: Habilitation thesis. Université Grenoble Alpes; 2016.

[35] Mattsson J, Göran Selander GAE. Object security in web of things. In: W3C workshop on the web of things – enablers and services for an open web of devices. June 2014.

[36] Kerr R, Cohen R. Detecting and identifying coalitions. In: International foundation for autonomous agents and multiagent systems. Proceedings of the 11th international conference on autonomous agents and multiagent systems, vol. 3. 2012. p. 1363–4.

[37] Khalfi EM, Jamont J-P, Cervantes F, 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. ACM; 2014. p. 1–2.

[38] Khenifar A, Jamont J-P, Occello M, Ben-Yelles C-B, Koudil M. A recursive approach to enable the collective level interaction of the web of things applications. In: Proceedings

of the 2014 international workshop on web intelligence and smart sensing. ACM; 2014. p. 1–5.

[39] Kirsch-Pinheiro M, Gensel J, Martin H. Representing context for an adaptative awareness mechanism. In: Groupware: design, implementation, and use. Springer; 2004. p. 339–48.

[40] Kovatsch M, Lanter M, Duquennoy S. Actinium: a restful runtime container for scriptable internet of things applications. In: 3rd IEEE international conference on the internet of things, IOT 2012. IEEE; 2012. p. 135–42.

[41] Le Sommer N, Ben Sassi S, Guidec F, Mahéo Y. A middleware support for location-based service discovery and invocation in disconnected MANETs. Studia Inform Universalis Sep. 2010;8(3):71–97.

[42] Le Sommer N, Launay P, Mahéo Y. A framework for opportunistic networking in spontaneous and ephemeral social networks. In: 10th workshop on challenged networks (CHANTS'2015). Paris, France: ACM; Sep. 2015.

[43] Le Sommer N, Said R, Mahéo Y. A proxy-based model for service provision in opportunistic networks. In: 6th international workshop on middleware for pervasive and ad-hoc computing (MPAC'08). Louvain, Belgium: ACM; Dec. 2008.

[44] Lee EA. Cyber physical systems: design challenges. In: 2008 11th IEEE international symposium on object oriented real-time distributed computing (ISORC). IEEE; 2008. p. 363–9.

[45] Lin J, Sedigh S, Miller A. Modeling cyber-physical systems with semantic agents. In: Computer software and applications conference workshops (COMPSACW), 2010 IEEE 34th annual. IEEE; 2010. p. 13–8.

[46] Mahéo Y, Said R. Service invocation over content-based communication in disconnected mobile ad hoc networks. In: 24th international conference on advanced information networking and applications (AINA'10). Perth, Australia: IEEE; Apr. 2010. p. 503–10.

[47] Mascolo C, Capra L, Emmerich W. Mobile computing middleware. In: Advanced lectures on networking. Springer; 2002. p. 20–58.

[48] Cuenca M, Da Cruz M, Morin R. Programming device ensembles in the web of things. In: W3C workshop on the web of things – enablers and services for an open web of devices. June 2014.

[49] Médini L, Terdjimi M. An avatar-based workflow for the semantic web of things. Communication in the WWW2016 W3C track, https://www.w3.org/2016/04/w3c-track.html, April 2016.

[50] Mrissa M, Médini L, Jamont J-P, Le Sommer N, Laplace J. An avatar architecture for the web of things. IEEE Internet Comput 2015;19(2):30–8.

[51] Mrissa M, Médini L, Jamont J-P. Semantic discovery and invocation of functionalities for the web of things. In: IEEE international conference on enabling technologies: infrastructure for collaborative enterprises. Jun. 2014.

[52] Munnelly J, Fritsch S, Clarke S. An aspect-oriented approach to the modularisation of context. In: Fifth annual IEEE international conference on pervasive computing and communications, 2007. PerCom'07. IEEE; 2007. p. 114–24.

[53] Musolesi M, Mascolo C. Car: context-aware adaptive routing for delay-tolerant mobile networks. IEEE Trans Mob Comput 2009;8(2):246–60.

[54] Neves dos Santos F, Sobreira H, Campos D, Morais R, Moreira AP, Contente O. Towards a reliable monitoring robot for mountain vineyards. In: International conference on autonomous robot systems and competitions (ICARSC 2015). Vila Real, Portugal: IEEE; Apr. 2015. p. 37–43.

[55] Papadopoulos GA, Stavrou A, Papapetrou O. An implementation framework for software architectures based on the coordination paradigm. Sci Comput Program 2006;60(1):27–67.

[56] Passarella A, Kumar M, Conti M, Borgia E. Minimum-delay service provisioning in opportunistic networks. IEEE Trans Parallel Distrib Syst 2010;22(8):1267–75.

[57] Raverdy P-G, Riva O, de La Chapelle A, Chibout R, Issarny V. Efficient context-aware service discovery in multi-protocol pervasive environments. In: 7th international conference on mobile data management, 2006. MDM 2006. IEEE; 2006. p. 3.

[58] Redhead F, Snow S, Vyas D, Bawden O, Russell R, Perez T, et al. Bringing the farmer perspective to agricultural robots. In: 33rd annual ACM conference on human factors in computing systems (CHI EA '15). Seoul, Republic of Korea: ACM; Apr. 2015. p. 1067–72.

[59] Schilit BN, Adams N, Gold R, Tso MM, Want R. The parctab mobile computing system. In: Fourth workshop on workstation operating systems, 1993. Proceedings. IEEE; 1993. p. 34–9.

[60] Schmidt A. Ubiquitous computing-computing in context. Ph.D. thesis. Lancaster University; 2003.

[61] Scott K, Burleigh S. Bundle protocol specification. In: IETF RFC 5050. Apr. 2007.

[62] Sindlar MP, Dastani MM, Dignum F, Meyer J-JC. Mental state abduction of bdi-based agents. In: Declarative agent languages and technologies VI. Springer; 2008. p. 161–78.

[63] Smith RG. The contract net protocol: high-level communication and control in a distributed problem solver. IEEE Trans Comput 1980;12:1104–13.

[64] Soon S, Pearce A, Noble M. A teamwork coordination strategy using hierarchical role relationship matching. In: Agents and computational autonomy. Springer; 2003. p. 249–60.

[65] Stehr M, Talcott CL, Rushby JM, Lincoln P, Kim M, Cheung S, Poggio A. Fractionated software for networked cyber-physical systems: research directions and long-term vision. In: Agha G, Danvy O, Meseguer J, editors. Formal modeling: actors, open systems, biological systems – essays dedicated to carolyn talcott on the occasion of her 70th birthday. Lecture notes in computer science, vol. 7000. Springer; 2011. p. 110–43.

[66] Sukthankar G, Geib C, Bui HH, Pynadath D, Goldman RP. Plan, activity, and intent recognition: theory and practice. Newnes; 2014.

[67] Sztipanovits J, Koutsoukos X, Karsai G, Kottenstette N, Antsaklis P, Gupta V, Goodwine B, Baras J, Wang S. Toward a science of cyber-physical system integration. Proc IEEE 2012;100(1):29–44.

[68] Terdjimi M, Médini L, Mrissa M. HyLAR+: improving hybrid location-agnostic reasoning with incremental rule-based update. In: WWW'16: 25th international world wide web conference companion. Apr. 2016.

[69] Terdjimi M, Médini L, Mrissa M. Towards a meta-model for context in the web of things. In: Karlsruhe service summit workshop. Feb. 2016.

[70] Terdjimi M, Médini L, Mrissa M, Le Sommer N. An avatar-based adaptation workflow for the web of things. In: WETICE 2016. Jun. 2016.

[71] Tomasello M, Kruger AC, Ratner HH. Cultural learning. Behav Brain Sci 1993;16(03):495–511.

[72] Truong H-L, Dustdar S, Baggio D, Corlosquet S, Dorn C, Giuliani G, et al. Incontext: a pervasive and collaborative working environment for emerging team forms. In: International symposium on applications and the internet, 2008. SAINT 2008. IEEE; 2008. p. 118–25.

[73] Wei Q, Farkas K, Prehofer C, Mendes P, Plattner B. Context-aware handover using active network technology. Comput Netw 2006;50(15):2855–72.

[74] Xiang B, Jiang D, Pei J, Sun X, Chen E, Li H. Context-aware ranking in web search. Sigir 2010;2010:451.
[75] Yao L, Sheng QZ, Benatallah B, Dustdar S, Shemshadi A, Wang X, Ngu AH. Up in the air: when homes meet the web of things. arXiv preprint arXiv:1512.06257, 2015.
[76] Yu Z, Zhou X, Zhang D, Chin C-Y, Wang X, et al. Supporting context-aware media recommendations for smart phones. IEEE Pervasive Comput 2006;5(3):68–75.
[77] Zhuo HH, Li L. Multi-agent plan recognition with partial team traces and plan libraries. In: IJCAI, vol. 22. 2011. p. 484.
[78] Zhuo HH, Yang Q, Kambhampati S. Action-model based multi-agent plan recognition. In: Advances in neural information processing systems. 2012. p. 368–76.
[79] Zimmermann A, Lorenz A, Oppermann R. An operational definition of context. In: Modeling and using context. Springer; 2007. p. 558–71.

ACKNOWLEDGEMENTS

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