Agent Abstractions for Engineering IoT Systems: a Case Study in Smart Healthcare

Eloisa Vargiu* and Franco Zambonelli†
* Eurecat Technology Center, eHealth Unit
Barcelona, Spain
eloisa.vargiu@eurecat.org
† Dipartimento di Scienze e Metodi dell'Ingegneria
Università di Modena e Reggio Emilia, Italy
franco.zambonelli@unimore.it

***Abstract*—Despite the rapid progresses in IoT research, a general principled software engineering approach for the systematic development of IoT systems and applications is still missing. In this article, we show that agent-oriented concepts and abstractions can play a key role in the design and development of IoT systems and applications, and could represent the ground on which to shape a new IoT-oriented software engineering discipline. A case study in the area of smart healthcare is adopted as a running example to ground the discussion.**

I. Introduction

Despite the great deal of worldwide researches in the area of the Internet of things (IoT) [9], the technologies to make it a systematic reality are far form being assessed. Early researchers in the IoT area have mostly focussed on communication issues and on enabling interoperability [3]. More recently, great efforts has been devoted at promoting means to facilitate the integration of resources and services towards the provisioning of software-defined distributed services for the IoT. For instance, as in the "Web of Things" (WoT) vision [8], by promoting the provisioning of resources in an IoT network in term of Web Services, and thus making it possible to develop distributed and coordinated IoT services by using standard Web technologies.

WoT is definitely promising and will most likely represent a keystone technology in the future of IoT. Indeed, along the WoT lines, a number of different approaches (in terms of, e.g., supporting middleware [16], [12] and programming approaches [4]) are being proposed to support the development of IoT systems and applications. Yet, a common unifying approach supporting their design and development, grounded on a common set of abstractions, models, and methodologies, is still missing. Also, relying on WoT concepts only for the design and development of IoT systems, one can miss identifying some key characteristics that will necessarily characterize many IoT services, such goal-oriented and autonomous behaviors [10]. Overall, this limits the possibility of promoting a systematic and disciplined approach for the development of complex IoT systems, and thus limits unfolding the full potentials of the IoT vision.

This article attempts at framing some key general characteristics related to the engineering of complex IoT systems and applications, by synthesizing the common features of existing proposals and application scenarios, and by bringing in the lessons of agent-based computing and agent-oriented software engineering. The so analyzed common characteristics are then used to identify some key software engineering abstractions around which the process of developing IoT systems and applications could revolve. Such abstractions – due to the inherent presence in IoT systems and applications of autonomous and goal-oriented behaviours – will exploit some key concept of agent-based computing and agent-oriented software engineering [17], and can be used to define a set guidelines for IoT-oriented software engineering.

To exemplify the analysis, we refer a specific case study, representative of a larger class of IoT scenarios, in the smart helthcare area: IoT enriched houses to support smart health monitoring and care. We assume houses are densely enriched with connected sensors and actuators: light and heat controllers, gas and smoke detectors, presence and motion sensors, door (main doors, internal doors, fridge, kitchen furniture) sensors, electric consume sensors, shutter/curtain controller, as well as sensorized everyday objects (e.g, cup, fork, cane). Moreover, also medical devices (e.g., pulse-oximetry, smart scale) may be provided to patients in order to automatically send health status information and measures. In such a scenario, different actors (from medical doctors to patients and their family members) can contribute to set up a variety of IoT services to support both medical doctors in the monitoring and care activities of individuals, and to help individuals and their family members in their everyday self-managed healthcare activities.

II. Background

The definition of general software engineering principles requires identifying the general features and issues that characterize most current approaches to IoT systems design and development.

A. Things

The "things" in the IoT vision may encompass a large number of physical objects, and also include places and persons.

Physical objects and places can be made trackable and controllable by connecting them to low-cost wireless electronic devices. At the lower end of the spectrum, RFID tags or bluetooth beacons, based on low-cost and short-range communication protocols, can be attached to any kind of objects to enable tracking their positions and status, and possibly to associate some digital information with them. More advanced devices integrating environmental or motion sensors (i.e., accelerometers) can detect the present and the past activities associated with objects or with some place. In addition, one can make objects actuable – enabling the remote control of their configuration/status via proper digitally-controller actuators – and possibly autonomous – delegating them of autonomously direct their activities. In this perspective, autonomous robots and autonomous objects [1] are components that will increasingly populate the IoT universe.

To exemplify, in the smart healthcare scenario: attach RFID to everyday objects in houses, such as a glass to detect the quantity of ingested water; integrate some kind of remote controller (e.g., Arduino-based) to turn on/off the light in a specific room, in order to enable controlling via, e.g., a mobile phone its turning on/off; automatically open and close the shutter/curtain depending on the performed activities, the context (the hour, the day), and/or user's habits; last but not least, robots for home assistance are gaining momentum (e.g., the Giraff plus [**?**]).

Concerning persons, other than simply users of the technology, they can also be perceived at first-class entities of the overall IoT vision. Simply for the fact of having a mobile phone, they can be sensed in their activities and positions, and they can be asked to act in the environment or supply sensing. In the smart healthcare scenario, beside continuously detecting the position and activities of people in order to get ready to manage any possible emergency situation (e.g., fall detection [13]), one can also think at involving them in self-monitoring and supply information to the overall health monitoring system [**?**].

B. Software Infrastructures

To make "things" usable and capable of serving purposes, there is need of software infrastructures (that is, of IoT middleware [14]) capable both of supporting the "gluing" of different things and of providing some means for stakeholders and users to access the IoT system and take advantage of its functionalities.

Concerning the "glue", this involves a variety of technical issues.

There are *interoperability* issues, to enable a variety of very heterogeneous things to interact with each other, via a set of common name spaces, uniform communication protocols and data representation schemes; and *semantic* issues, because a common semantics for concepts must be defined to enable cooperation and integration of things. For both these issues, however, a large body of proposals (dating back to the early years of IoT research) exists. Thus, for our purposes in this article, we assume the existence of proper technical solutions.

Rather, key open "gluing" issues of relevance for software engineering include *discovery, Group Formation, and Coordination*. IoT systems functionalities derive from the orchestrated exploitation of a variety of things, possibly involving a variety of users and stakeholders. In the smart healthcare scenario, it is desirable to automatically configure a given room (e.g. bedroom) for a given context (e.g., time to go to sleep). This requires involving the lightening and shutter system, and consider recommendations by caregivers and clinicians [7]. Thus, it implies to discovery and establish relations between things, between things and humans, and coordinating their activities also accounting for their social relations [2]. Clearly, for the above coordination mechanisms to work, *context-awareness and self-adaptation* are required. In fact, the inherent ephemerality, unreliability, and mobility of system components (e.g., things such as everyday objects at home may come and go, can be moved around, and can be placed in corners without wireless connections) makes it impossible to anticipate which things will be available and for how long during their exploitation. This requires mechanisms for discovery, group formation, and coordination are that are capable of dynamically self-adapting to the general context in which they act, or possibly even self-organize in a context-aware way. [11], [18].

Concerning the "access" to the functionalities and capabilities of individual things by users, the scene is currently dominate by the so called "Web of Things" (WoT) vision [8]. The idea is to expose services and functionalities of individual things in terms of REST services, enabling the adoption of assessed web technologies as far as discovery of things and provisioning of coordinated group services are concerned. Concerning middleware infrastructures, a variety of proposal to support the provisioning of IoT services and applications have appeared [16], [4], [14]. Beside their specificities, most of these proposals rely on: some basic infrastructure to support the WoT approach (i.e., to expose things in terms of simple services); some means to support, in according to a specific coordination model, the discovery of things (and of their associated services), and the coordinated activities of groups of things; and some solutions to make services and applications capable of self-adapting and self-organizing in a context-aware and unsupervised way.

C. Services and Applications

With the term "IoT System" we generally refer to the overall set of IoT devices and to the associated middleware infrastructure devoted to manage their networking and their context-aware interactions. Logically above an IoT system, specific software can be deployed to orchestrate the activities of the system so as to provide:

- A number of specific *services*. That is, means to enable stakeholders and users to access and exploit individual things and direct/activate their sensing/actuating capabilities, but also coordinated services that access groups of things and coordinate their sensing/actuating capabilities. For instance, in a smart home instrumented for healthcare,

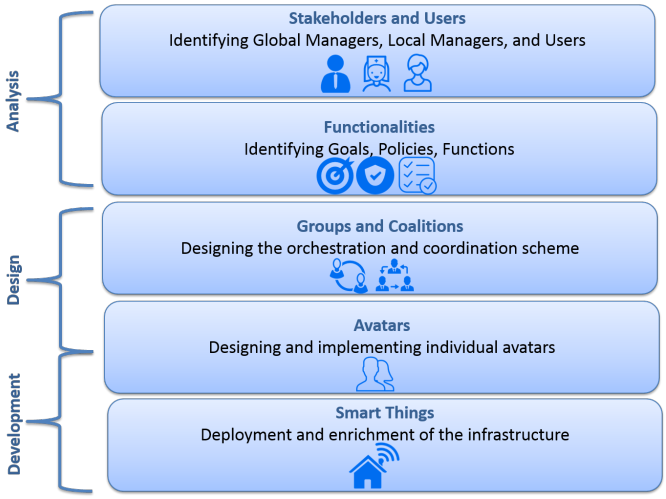

Fig. 1. Key concepts and abstractions for IoT engineering.

other than to services to access and control individual appliances, one can think at providing a coordinated service that, by accessing and directing the lightening system, the light sensors, and the windows obscuring system in a specific room, can modify the overall situation of that room depending on the specific need of the person occupying it.

- A number of more general-purpose *applications* or *suites*, intended as more comprehensive software systems intended to both regulate the overall functioning of an IoT system (or of some of its parts), so as to ensure specific overall behaviour of the system, as well as to provide an harmonized set of services to access the system and (possibly) its configuration. In the smart home scenario, one can think at applications to control the overall heating systems and lightening systems of a set of houses hosting patients with a specific health problem, and giving medical doctors and/or carers the access to services to change the configuration of the associated parameter.

Clearly, depending on the specific scenario, one can think at IoT systems in which services may exist only confined within the context of some general application, but also at scenarios in which there are services that can be deployed as stand-alone software.

III. Software Engineering Abstractions and the Role of Agent-based Computing

Based on the above overview of IoT issues, we now try to synthesize the central concepts and abstractions around which the development of IoT systems (spanning analysis, design, and implementation) should be centered, and discuss how these directly relate to concepts and abstractions developed in the context of agent-based computing [10], [17]. Figure 1 graphically frames such concepts in a logical stack.

A. Actors

The first activity in the analysis of a system-to-be concern identifying the stakeholders and users of the system, aka the "actors". That is, those persons/organizations who will own, manage, and/or use the system and its functionalities, and from which requirements should be elicited.

In the case of IoT systems, the distinction between IoT services and applications, and the presence of an IoT middleware to support them and to manage individual things, naturally leads to the identification of three main abstract classes of"actors":

- *Global Managers*: These are the owners of an overall IoT system and infrastructure, or delegates empowered to exert control and establishing policies over the configuration, structure, and overall functioning of its applications and services. In the smart healthcare scenario, the global manager corresponds to the system manager devoted to control the overall IoT system of the smart houses set according to the directives of the medical doctors, e.g., for deciding heating levels or for surveillance strategies.
- *Local Managers*: These are owners/delegates (whether permanently or on a temporary basis) of a limited portion of the IoT system, empowered to enforce local control and policies for that portion of the system. In the smart healthcare scenario, these could correspond to the house owners, empowered to control the IoT system in their houses and rooms, and tune the local parameters and exploit its services according to own specific needs.
- *Users*: These are persons or groups that have limited access to the overall configuration of the IoT applications and services, i.e., cannot impose policies on them, but are nevertheless entitled to exploit its services. In the smart healthcare scenario, these include the patients with limited abilities, authorized to access specific services (e.g., regulating specific appliances), but not entitled to modify the overall configuration of their houses (in charge of medical doctors and partly of their responsible family members.

The three identified classes of actors are of a very general nature, beside the smart healthcare scenario. For example, in a scenario of energy management in a smart city, they could correspond to, respectively: city managers, house/shop owners, private citizens and tourists. In the area of urban mobility, they could correspond to, respectively: mobility managers, parking owners or car sharing companies, private drivers.

B. Functionalities

Once the key actors are identified, the analysis preceding design and implementation cannot – for IoT systems and applications – simply reduce to elicit from them the functionalities (i.e., the specific services) that things or group of things has to provide, but has to account for a more comprehensive approach. In fact:

- Beside things provided with basic sensing/actuating functionalities, one should consider the presence of smarter

things that can be activated to perform in autonomy some long-term activities associated with their nature and with their role in the socio/physical environment in which they situates. These can range from simply cleaning robots to more sophisticated autonomous personal assistants [1].
- IoT applications are not simply concerned with providing a suite of coordinated functionalities, but they should also globally regulate the activities of the IoT systems on a continuous basis, according to the policies established by its stakeholders and to their objectives.

As a consequence, other than analyzing the specific functionalities to deliver, one also has to identify the *policies* and *goals* to be associated with services and applications, i.e., the desirable "state of the affairs" to strive for in the context of the socio-cyber-physical system where IoT applications and services operate.

In this perspective, the general classes of functionalities to be identified for the development of IoT applications and services include:

- *Policies* express desirable permanent configurations or states of functioning of an overall IoT system (global policies) or portions of it (local policies), and have the aims of regulating the overall underlying IoT system. In the smart healthcare scenario, global policies can be defined, e.g., to specify the maximum sleeping hours, the maximum time for sedentary activities, and have this monitored by not-intrusive sensors in order to invite people to make more activities or to go for resting whenever needed. Policies are meant to be alway active and actively enforced. Although, from the software engineering viewpoint, the focus is mostly on application-level policies, policies can also account for the proper configuration of the underlying hardware and network infrastructures. The definition of global and local policies is generally in charge of the global managers, although local managers can be also entitled to enforce temporary local policies on local portions of the system (provided they do not contrast with the ones imposed by the global managers).
- *Goals* express desirable situations or state of the affairs that, in specific cases, can/should be achieved. The activation of a goal may rely on specific pre-conditions (i.e., the occurrence of specific events or the recognition of some specific configurations in the IoT system) or may also be specifically activated upon user action (e.g., the activation of a goal is invokable "as a service"). The typical post-condition (deactivating the pursuing of a goal) is the achievement of the goal itself. In the smart healthcare scenario, one example could be that of activating an evacuation procedure upon detection of fire by a smoke sensor (pre-conditions), whose goal (and post-condition) is to achieve a quick evacuation of the patient from her/his home. To this end, the activation of a goal can trigger the activities of digital signages and controllable doors in order to rationally guide people towards the exits. Another example could be the case of a fall has been detected. An audio sensor automatically recognizes the help request by the patient (pre-conditions), whose goal is to immediately send assistance at home (e.g., an ambulance) and to communicate with the familiars to make a visit and support the patient (post-condition). To this end, the activation of a goal can trigger the activities of contacting caregivers and familiars. As it was the case for policies, the definition of global and local goals is generally in charge of global, and sometimes of local, managers, whereas users can be sometimes entitled to activate simple local goals (or goals associated to individual things) "as a service".
- *Functions* define the sensing/computing/actuating capabilities of individual things or of group of things, or the specific resources that are to be made available to managers and users in the context of specific IoT application and services. Functions are typically made accessible in the form of services, and can sometime involve the coordinated access to the functions of a multitude of individual things. In the smart healthcare scenario, one can think at the individual functionalities of a door sensor in a fridge (e.g., to control opening/closing), as well as more complex functionalities that can be achieved by orchestrating things (e.g., controlling food in the fridge to verify if the shopping list updating it with needed food). Functions and the associated services are typically defined by global and possibly local managers, but are exploited also by the everyday users of the IoT systems (e.g., the patient and her/his caregivers).).

Clearly, the concepts of goals and policies are central in the research area of agent systems and multiagent systems, and will require, to be realized, components with autonomous and social behaviour, capable of working together towards the achievement of goals and the enforcement of policies.

C. Software Components and Their Coordination

Moving from analysis to the design of an actual system and of its components, one should consider that the "things" to be involved in the implementation of the identified functionalities can correspond to a variety of different objects and devices, other than to places and humans, each relying on a pletora of different technologies and capabilities. Accordingly, from both the gluing software infrastructure and the software engineering viewpoints, it is necessary to define higher-level abstractions to practically and conceptually handle the design and development of application and services, and to harmonically exploit all the components of the IoT system.

Most of the proposal for programming models and middleware acknowledge this need, by virtualizing individual things in some sort of software abstraction [8]. The WoT perspective abstracts things and their functionalities in terms of generic resources, to be accessed via RESTful calls, possibly associating external software HTTP "gateways" to individual things if they cannot directly support HTTP interfacing. Other

approaches suggest adopting a more standard SOA or object-oriented approach. Surprisingly, only a few proposals consider associating autonomous software agents to individual things [15], despite the fact goals to be pursued in autonomy may be associated to things, a feature that service-oriented approaches can hardly accommodate.

In addition, as already stated, some "things" make no sense as individual entities as far as the provisioning of specific services and applications is concerned, and are to be considered part of a group and be capable of providing their services as a coordinated group. This applies both to the cases in which a multitude of equivalent devices must be collectively exploited abstracting from the presence of the individuals [4], and to the cases in which the functionalities of the group complement with each other and needs to be orchestrated [15]. However, due to the dynamic and contextual nature of IoT scenario, traditional service-oriented orchestration methods, although necessary, are not enough to

With these considerations in mind, in an effort of synthesizing from a variety of different proposals and of bringing in as needed agent-oriented concepts, we suggest the unifying abstractions of *avatars* and *coalitions* (See Figure 2).

Avatars. Borrowing the term from [12] (to distinguish from software agents but nevertheless borrowing several features from them) we define an avatar as the general abstraction for individual things and also for group of things (and possibly other avatars) that contribute to define a unique functionality/service. Avatars abstract away form the specific physical/social/technological characteristics of the things their represent, and are defined by means of:

- *Identity*. An avatar has a unique identity and is addressable. An avatar representing a group does not necessarily hides the identities of inner avatars, but it has its own identity.
- *Services*. These represent access point for exploiting the peculiar capabilities of avatars. That is, depending on the kinds of things and functionalities a service abstracts: triggering and directing the sensing/computing/actuating capabilities, or accessing some managed resources.
- *Goals*. Goals, in the sense of desired state of the affairs, can be associated to avatars. A goals have may a pre-condition for autonomous activation, or may be explicitly activated by a user or by another avatar.
- *Events*. Events represent specific state of the affairs that can be detected by an avatar, and that may be of interests to other avatars or to users. Other avatars or users can subscribe to events of interest.

Clearly, for group of avatars, an internal *orchestration scheme* must be defined for coordinating the activities/functionalities of the things (or of the other avatars) it includes. In general terms – and in accord to assessed service-oriented approaches – an orchestration scheme defines the internal workflow of activities among the composing things and avatars, and the constrains/conditions they are subjected to. Orchestration scheme may also account for contextual information, to make the activities of the group of context-aware. The need of defining orchestrations schemes and constraints to rules the access and usage of (group of) things is generally attributed – with specific characteristics and terminologies – in most middleware and programming approaches for IoT [16], [4].

The avatar abstraction is in line, and account for all the typical characteristics, of most existing IoT approaches. However, the stateful concepts of goals and events make avatars go beyond RESTful approaches. Indeed, these concepts make an avatar more than simply a service provider, turning them into autonomous entities capable of goal-oriented and situated behaviour. Although most existing approaches recognize the need to somehow incorporate similar concepts within RESTful architectures [8], a few of them explicitly refer to agent-based computing, where such concepts belong to.

Coalitions. In this case, and without fear of borrowing the term from the area of multiagent systems [6], we define a coalition as a group of avatars that coordinate each other's activities in order to reach specific goals, or enact specific policies. Accordingly, coalitions may be of a temporary or permanent nature. Unlike avatar groups, coalitions does not necessarily have an identity, and does not necessarily provide services.

To define and bring a coalition in action, the abstraction of coalition must be defined (at least) in terms of a *coordination scheme* that should include:

- *Rules for membership*, to specify the conditions upon which an avatar should/could enter a coalitions. From the viewpoint of individual avatars, the act of entering a coalition can be represented by the activation of a specific goal based on pre-conditions that correspond to the rules for membership [5].
- *Coordination pattern*, to define the pattern (interaction protocol and shared strategy) by which the members of the coalition have to interact. The coordination pattern may include an explicit representation of the goal by which the coalition has been activated. However, such goal can also be implicit in the definition of the protocol and of the strategy.
- *Coordination law*, to express constraints that must be enforced in the way the avatars involved in the coalition should act and interact.

In addition, one can consider the possibility to subscribe to events occurring within the coalition.

The view of avatar coalitions can be of use to realize policies, or to aggregate groups of avatar based on similarity, so as to make them work collectively in a mission-oriented way without forcing them to specific identity-centered orchestration scheme. This is coherent with the idea of multiagent societies and, in general, of distributed dynamic coordination [10]. Also, this is in line with nature-inspired approaches [18], and approaches to aggregate programming.

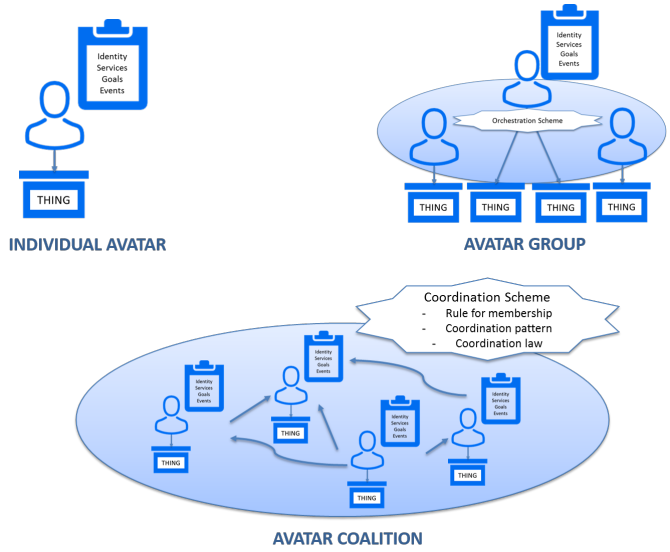

Fig. 2. Avatars, groups, and coalitions.

D. From Design to Implementation

The identification of avatars, avatar groups, and coalitions, abstracts away from implementation issues. However, the implementation of individual avatars associated to actual "things" and of the necessary software for supporting for the orchestration schemes of avatar groups and the coordination patterns of coalitions, has to eventually follow.

In our perspective, and comparing against the state of the art in the area, avatars, groups and coalitions are abstract enough concepts to tolerate implementation – with different efforts – above most existing systems and infrastructures. However, this article at least contributes in suggesting a more deep adoption of multiagent concepts and – consequently – of multiagent languages and middleware infrastructure – in the development of next-generation IoT systems.

IV. Conclusions and Future Work

Despite the large number of research works that attack specific problems related to the design and development of IoT applications and services, a general software engineering approach is still missing. This paper, by having proposed and framed some key conceptual abstractions revolving about the IoT universe, and showing how these related to agent-based computing, can represent a first small step towards a general discipline for engineering IoT systems and applications.

As IoT technologies mature, and real-world experiences accumulate, more research in the area of software engineering for IoT systems will be needed, and these will have increasingly exploit contaminations with the area of agent-oriented software engineering [17] and of software engineering for self-adaptive and self-organizing systems [18], and eventually leading to the identification of a widely accepted general methodology – and associated tools – for the IoT-oriented software engineering.

References

[1] Harshit Agrawal, Sang-won Leigh, and Pattie Maes. L'evolved: Autonomous and ubiquitous utilities as smart agents. In *ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 487–491, New York, NY, USA, 2015. ACM.

[2] Luigi Atzori, Davide Carboni, and Antonio Iera. Smart things in the social loop: Paradigms, technologies, and potentials. *Ad Hoc Networks*, 18:121–132, 2014.

[3] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[4] Jacob Beal, Danilo Pianini, and Mirko Viroli. Aggregate programming for the internet of things. *IEEE Computer*, 48(9):22–30, 2015.

[5] T. Bures, F. Plasil, M. Kit, P. Tuma, and N. Hoch. Software abstractions for component interaction in the internet of things. *Computer*, 49(12):50–59, Dec 2016.

[6] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *Industrial Informatics, IEEE Transactions on*, 9(1):427–438, 2013.

[7] Juan Manuel Fernández, Marc Solà, Alexander Steblin, Eloisa Vargiu, and Felip Miralles. The relevance of providing useful information to therapists and caregivers in tele. In Cristian Lai, Alessandro Giuliani, and Giovanni Semeraro, editors, *Information Filtering and Retrieval. DART 2014: Revised and Invited Papers*, pages 97–117. Studies in Computational Intelligence, Vol. 668, Springer, 2016.

[8] J. Heuer, J. Hund, and O. Pfaff. Toward the web of things: Applying web technologies to the physical world. *Computer*, 48(5):34–42, May 2015.

[9] Marco Iansiti and Karin Lakhani. Digital ubiquity: How connections, sensors, and data, are revolutionizing business. *Harvard Business Review*, 2014.

[10] Nicholas R. Jennings. An agent-based approach for building complex software systems. *Commun. ACM*, 44(4):35–41, 2001.

[11] Alexander Kott, Ananthram Swami, and Bruce West. The internet of battle things. *Computer*, 49(12):70–75, 2016.

[12] M. Mrissa, L. Medini, J.-P. Jamont, N. Le Sommer, and J. Laplace. An avatar architecture for the web of things. *Internet Computing, IEEE*, 19(2):30–38, Mar 2015.

[13] Hammadi Nait-Charif and Stephen J McKenna. Activity summarisation and fall detection in a supportive home environment. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 4, pages 323–326. IEEE, 2004.

[14] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke. Middleware for internet of things: A survey. *IEEE Internet of Things Journal*, 3(1):70–95, Feb 2016.

[15] N. Spanoudakis and P. Moraitis. Engineering ambient intelligence systems using agent technology. *Intelligent Systems, IEEE*, 30(3):60–67, May 2015.

[16] Jonas Ullberg, Amy Loutfi, and Federico Pecora. A customizable approach for monitoring activities of elderly users in their homes. In *International Workshop on Activity Monitoring by Multiple Distributed Sensing*, pages 13–25. Springer, 2014.

[17] Lina Yao, Q.Z. Sheng, and S. Dustdar. Web-based management of the internet of things. *Internet Computing, IEEE*, 19(4):60–67, July 2015.

[18] Franco Zambonelli and Andrea Omicini. Challenges and research directions in agent-oriented software engineering. *Autonomous Agents and Multi-Agent Systems*, 9(3):253–283, 2004.

[19] Franco Zambonelli, Andrea Omicini, Bernhard Anzengruber, Gabriella Castelli, Francesco L. De Angelis, Giovanna Di Marzo Serugendo, Simon Dobson, Jose Luis Fernandez-Marquez, Alois Ferscha, Marco Mamei, Stefano Mariani, Ambra Molesini, Sara Montagna, Jussi Nieminen, Danilo Pianini, Matteo Risoldi, Alberto Rosi, Graeme Stevenson, Mirko Viroli, and Juan Ye. Developing pervasive multi-agent systems with nature-inspired coordination. *Pervasive and Mobile Computing*, 17, Part B:236–252, 2015.

[20] Carme Zambrana, Xavier Rafael-Palou, and Eloisa Vargiu. Sleeping recognition to assist elderly people at home. *Artificial Intelligence Research*, 5(2):p64, 2016.