Towards a Discipline of IoT-Oriented Software Engineering

Franco Zambonelli
Dipartimento di Scienze e Metodi dell'Ingegneria
Universitá di Modena e Reggio Emilia
Reggio Emilia, Italia
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, by synthesizing form the state of the art in the area, we attempt at framing the key concepts and abstractions that revolve around the design and development of IoT systems and applications, and that could represent the ground on which to start shaping the guidelines of a new IoT-oriented software engineering discipline.**

I. Introduction

The dramatic future impact of IoT in society, industry, and commerce is already widely recognized [14]. However, despite the great deal of worldwide researches in the area, the technologies to make IoT 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 [13], 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 [29], [17] and programming approaches [5], [16]) 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. This undermines 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.

Against this background, 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. Such common characteristics are then used to identify the key software engineering abstractions around which the process of developing IoT systems and applications could revolve, and via which to organize a set of guidelines towards a general IoT-oriented software engineering discipline.

To exemplify the analysis, we refer a specific case study, representative of a larger class of IoT scenarios: an IoT enabled hotel with conference center. We assume the hotel infrastructures (e.g., lightening, heating, etc.) and its facilities (guest rooms, conference rooms, and their associated appliances) are densely enriched with connected sensors and actuators. There, different actors (from hotel managers to hotel/conference guests) can contribute to set up a variety of IoT services to support both the hotel management and the activities of its guests.

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.

To exemplify, in the hotel scenario: attach RFID tags to objects in rooms, such as to a remote control in order to detect its presence and location in the room; integrate some kind of Arduino-link controller to a roll-up board in the conference room, in order to enable controlling via, e.g., a mobile phone

its rolling-unrolling; have the window obscuring systems autonomously regulate lightening conditions depending on the kind of activities detected in the conference room, and possibly actuable walls that can dynamically change the shape and dimensions of meeting rooms depending on needs [21]. In this perspective, autonomous robots (or robotified objects [1]) can be somehow considered the highest end of the spectrum in the world of smart "things".

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 hotel scenario, one may think continuously detecting the position and activities of people, in order to get ready to manage any possible emergency situation in the most efficient way.

B. Software Infrastructures

To make "things" usable and capable of serving purposes, there is need of software infrastructures (that is, of IoT middleware [22]) 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:

- *Interoperability*. To enable a variety of very heterogeneous things to interact with each other, a set of shared tele-communication protocols and data representation schemes must be put in place [20], other than means to identify things [24]. The study of these issues dates to the very early stages of IoT researches, a number of different proposals exists, and the way towards assessed standards in well paved.
- *Semantics*. Beyond mere interoperability, a common semantics for concepts must be defined to enable cooperation and integration of things [4]. Also for this issue, a number of proposals grounded on standard Web technologies, and ontologies and schemas specifically suited for the physically and socially embedded nature of the , exists [20].
- *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 hotel scenario, configuring a conference rooms for slide presentation requires involving the beam projector, the lightening system, other than the conference organizers and the speakers. This requires means to discovery and establish relations between things, between things and humans, and coordinating their activities also accounting for their social relations [2].
- *Context-awareness and self-adaptation*. The inherent ephemerality, unreliability, and mobility of system components (e.g., things such as chairs or flipboards in the hotel conference centre can 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 [31].

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 [13]. 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 [29], [5], [16], [22]. 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 conference room of the hotel, 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, can modify the overall situation of the room from "presentation state" to "discussion state" and viceversa.
- 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 hotel scenario, one can think at applications to control the overall heating systems and lightening systems, and giving to hotel clerks

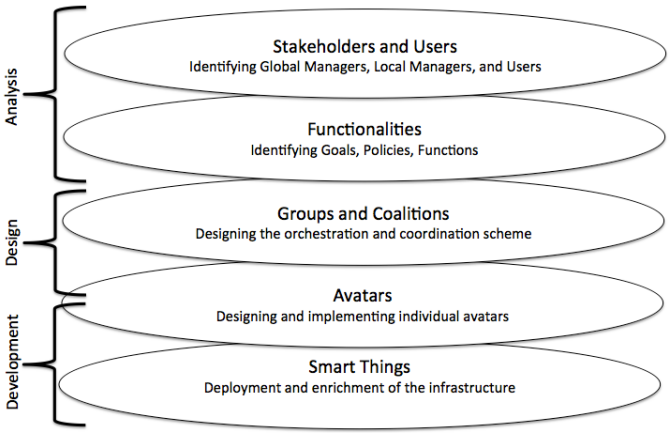

Fig. 1. Key concepts and abstractions for IoT engineering.

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. Key Software Engineering Concepts and Abstractions

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. Figure 1 graphically frames such concepts in a logical stack.

A. Stakeholders and Users

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 hotel scenario, the global manager corresponds the system manager devoted to control the overall IoT system of the hotel according to the directives of the hotel management, 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 hotel scenario, these could correspond to hotel guests, empowered to control the IoT system in their room, and tune the local parameters and exploit its services according to own specific needs. Or they can be conference organizers in charge of managing and configuring the services of the rented conference rooms.
- *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 hotel scenario, these include conference delegates authorized to access the conference facilities (e.g., uploading presentations in the projector), but are not entitled to modify the configuration of the conference rooms.

The three identified classes of actors are of a very general nature, beside the hotel 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 [28], 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 hotel scenarios, global policies can be defined, e.g., to specify the maximum occupancy levels in each room and have this monitored by local cameras in order to invite people to move in different rooms 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 hotel scenario, the clearer example could be that of activating an evacuation procedure upon detection of fire by some sensors (pre-condition), whose goal (and post-condition) is to achieve a quick evacuation of all people inside the building. 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. 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 hotel scenario, one can think at the individual functionalities of the appliances in a conference room (e.g., open/close a curtain, display slide / change slide in a projector), as well as more complex functionalities that can be achieved by orchestrating things (e.g., set up room for presentation by closing all curtains and switching off all lights). 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 hotel guests and the conference attendees).

Figure 2 shows the different roles of IoT actors in defining and exploiting the above framed functionalities.

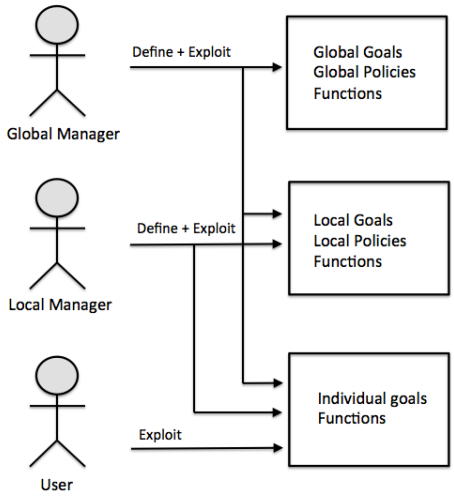

Fig. 2. IoT actors and the functionalities of IoT systems.

C. Avatars and Coalitions

Moving from analysis to design, 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 [13]. 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 [6]. Other approaches suggest adopting a more standard SOA or object-oriented approach [23]. Also, some proposals consider associating autonomous software agents to individual things [27], which we think well suits the fact that goals to be pursued in autonomy may be associated to things.

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 [5], and to the cases in which the functionalities of the group complement with each other and needs to be orchestrated [27].

With these considerations in mind, in an effort of synthesizing from a variety of different proposals, we suggest the unifying abstractions of *avatars* and *coalitions* (See Figure 3).

Avatars. Borrowing the term from [17], 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, an orchestration scheme defines the internal workflow of activities among the composing thungs 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 [23], [29], [5].

More in general, the avatar abstraction is in line, and account for all the typical characteristics, of most existing IoT approaches. Although the idea is not fully in line with that of RESTful WoT approaches, because of the stateful concepts of goals and events, most of them recognize the need to somehow incorporate similar concepts within RESTful architectures [13], to suit the dynamic and contextual nature of IoT systems and applications.

Coalitions. Borrowing the term from the area of multiagent systems [7], 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.
- *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.

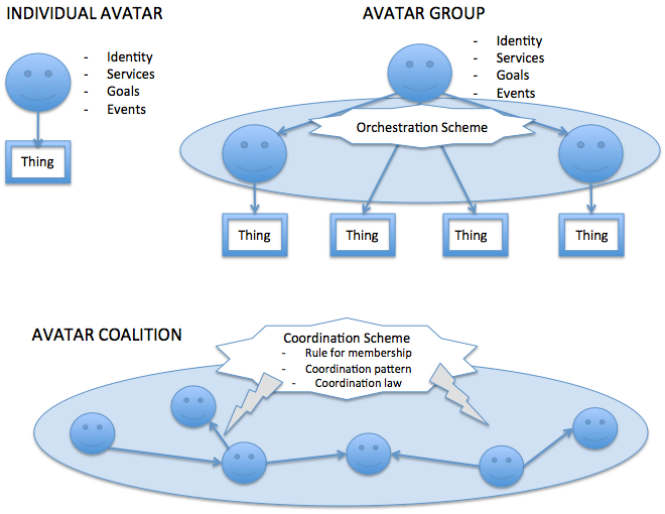

Fig. 3. Avatars, groups, and coalitions.

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 aggregate programming in sensor networks and in spatial computing systems [5], to realize nature-inspired coordination schemes [31], to enable the dynamic formation of service ensembles focused on short-term goals or sharing specific attributes [9].

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 above most existing systems and infrastructures. If not, this article at least contributes proposing a starting point from where to reason further on the expressiveness and necessity of the identified abstractions, and on the desirable features of IoT programming systems and middleware.

IV. Related Work

In the past few years, research in the area of IoT has exploded. Nevertheless, a few research work has explicitly attacked the problem of defining new software engineering approaches specifically conceived for the IoT.

Some proposals for development frameworks for the IoT or for the WoT (whether middleware architecture [17] or programming models [5], [16]), are also accompanied by guidelines towards the development of applications. However, such guidelines are not grounded on general abstractions and haven't a general applicability beside the specific framework in which they are conceived. Similar considerations apply to the area of smart cities and urban computing [30], where middleware and programming approaches are being proposed – mostly of a special-purpose nature and focussed on specific application scenarios such as participatory sensing [11], [12] or mobility management [25] – but without accounting for the issue of defining general design and development methodologies.

Agent-oriented software engineering research is strictly related to IoT engineering [32]. Indeed, AOSE tackles the problem of engineering large-scale systems, goal-oriented entities, possibly including robots [26] and humans [15] with conflicting goals and a multitude of stakeholders. This is somehow related to the IoT problems of accommodating services and a multitude of goals [27]. Indeed, the idea of goal-oriented groups we have introduce somehow borrow from the agent-oriented software engineering area. However, IoT requires the introduction of specific concepts and abstractions that AOSE, in general terms, do not address.

General approaches for the engineering of self-organizing computing systems have been proposed [19], [33], [18], [31], [10]. There, the key issue is to engineering complex distributed behaviours in large-scale systems lacking centralized control. These two characteristics are mostly shared by IoT systems, and indeed the problems of enabling self-organization of specific behaviors have been outlined in the previous sections.

Mainstream software engineering researches have recently put great attention to the problem of promoting self-adaptive features in software [8], to attack the problem of increased dynamically and impredictability of operational environments. Such dynamics also affects IoT systems, in which the problem of ensuring continuity in functionalities requires the embedding of close control loops (along similar lines of those promoted in self-adaptive systems researches) to continuously monitor the activities of the system and its environment, and eventually plan corrective actions.

V. 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, 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, possibly exploiting contaminations with the related areas of agent-oriented software engineering [32] and software engineering for self-adaptive and self-organizing systems [8], 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] Payam Barnaghi, Wei Wang, Cory Henson, and Kerry Taylor. Semantics for the internet of things: early progress and back to the future. *International Journal on Semantic Web and Information Systems*, 8(1):1–21, 2012.

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

[6] Gérôme Bovet and Jean Hennebert. Offering web-of-things connectivity to building networks. In *ACM Conference on Pervasive and Ubiquitous Computing - Adjunct Publication*, pages 1555–1564, New York, NY, USA, 2013. ACM.

[7] 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.

[8] B. H. C. Cheng and al. Software engineering for self-adaptive systems: A research roadmap. In *Software Engineering for Self-Adaptive Systems*, volume 5525 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2009.

[9] Rocco De Nicola, Diego Latella, Alberto Lluch-Lafuente, Michele Loreti, Andrea Margheri, Mieke Massink, Andrea Morichetta, Rosario Pugliese, Francesco Tiezzi, and

Andrea Vandin. The SCEL language: Design, implementation, verification. In *Software Engineering for Collective Autonomic Systems - The ASCENS Approach*, pages 3–71. Springer Verlag, 2015.

[10] J.L Fernandez-Marquez, G. Di Marzo Serugendo, S. Montagna, M. Viroli, and J. Arcos. Description and composition of bio-inspired design patterns: a complete overview. *Natural Computing*, 12(1):43 – 67, 2013.

[11] Sara Hachem, Animesh Pathak, and Valérie Issarny. Service-oriented middleware for large-scale mobile participatory sensing. *Pervasive and Mobile Computing*, 10:66–82, 2014.

[12] Dries Harnie, Theo D'Hondt, Elisa Gonzalez Boix, and Wolfgang De Meuter. Programming urban-area applications for mobility services. *ACM Transactions on Autonomous and Adaptive Systems*, 9(2), 2014.

[13] 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.

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

[15] N. R. Jennings, L. Moreau, D. Nicholson, S. Ramchurn, S. Roberts, T. Rodden, and A. Rogers. Human-agent collectives. *Commun. ACM ACM*, 57(12):80–88, December 2014.

[16] E. Latronico, E.A. Lee, M. Lohstroh, C. Shaver, A. Wasicek, and M. Weber. A vision of swarmlets. *Internet Computing, IEEE*, 19(2):20–28, Mar 2015.

[17] 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.

[18] Andrea Omicini and Mirko Viroli. Coordination models and languages: From parallel computing to self-organisation. *The Knowledge Engineering Review*, 26(1):53–59, March 2011.

[19] Van Parunak. Go to the ant: Engineering principles from natural multi-agent systems. *Annals of Operations Research*, 75:69–101, 1997.

[20] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *Communications Surveys Tutorials, IEEE*, 16(1):414–454, First 2014.

[21] M. Phillips. The slothbot moving wall projects, http://arch-os.com/projects/slothbot/.

[22] 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.

[23] C. Sarkar, S.N.A.U. Nambi, R.V. Prasad, and A. Rahim. A scalable distributed architecture towards unifying iot applications. In *IEEE World Forum on Internet of Things*, pages 508–513, March 2014.

[24] Amardeo C Sarma and João Girão. Identities in the future internet of things. *Wireless personal communications*, 49(3):353–363, 2009.

[25] Andrea Sassi and Franco Zambonelli. Coordination infrastructures for future smart social mobility services. *IEEE Intelligent Systems*, 29(5):78–82, 2014.

[26] Nathan Schurr, Janusz Marecki, Milind Tambe, and Paul Scerri. Towards flexible coordination of human-agent teams. *Multiagent and Grid Systems*, 1(1):3–16, 2005.

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

[28] Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Fifth IEEE International Symposium on Requirements Engineering*, pages 249–262. IEEE, 2001.

[29] 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.

[30] Franco Zambonelli. Toward sociotechnical urban superorganisms. *IEEE Computer*, 45(8):76–78, 2012.

[31] Franco Zambonelli and et al. Developing pervasive multi-agent systems with nature-inspired coordination. *Pervasive and Mobile Computing*, 37, 2015.

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

[33] Franco Zambonelli and Mirko Viroli. A survey on nature-inspired metaphors for pervasive service ecosystems. *Journal of Pervasive Computing and Communications*, 7:186–204, 2011.