Key Abstractions for IoT-Oriented Software Engineering

Franco Zambonelli, University of Modena and Reggio Emilia

A synthesis of current Internet of Things (IoT) research can help frame the key abstractions related to systematic development of IoT systems and applications. Such a framework could be the basis for guidelines for IoT-oriented software engineering.

Despite the considerable research on the Internet of Things (IoT), the technologies to make the IoT a systematic reality are far from being assessed. Early research focused mostly on communication and interoperability.[1] More recently, researchers have tried to facilitate the integration of resources and services toward provisioning software-defined distributed services. This is the case, for example, for the Web of Things (WoT) vision,[2] which aims to employ standard Web technologies to help develop coordinated IoT services.

The WoT—possibly integrated with concepts from agent-based computing[3,4]—will likely represent a keystone technology in the IoT's future. Along such lines, several approaches (for example, in terms of supporting middleware[5,6] and programming[7]) exist to aid IoT system and application development. Yet, a unifying approach to this development, grounded on a common set of abstractions, models, and methodologies, is still missing.[8]

Here, I try to frame key general characteristics related to the engineering of complex IoT systems and applications, by synthesizing the common features of existing proposals and scenarios. I then use these features to identify the key software engineering abstractions around which IoT system and application development could revolve. That is, these abstractions could be the basis for a general IoT-oriented software engineering discipline, including detailed models, methodologies, and guidelines (whose analysis and definition, though, is outside this article's scope).

To exemplify the analysis, I present a specific case study scenario: an IoT-enabled hotel and conference center. I assume the hotel infrastructure (for example, lighting and heating) and 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 guests or conference attendees) can contribute to set up a variety of IoT services. Although this scenario seems simple, it exhibits characteristics and engineering challenges representative of more complex scenarios such as energy management in smart cities and urban transportation management.

Common Features

I identified the following general features of current approaches to IoT system design and development.

Things

The "things" in the IoT vision might include a large number of physical objects, along with places and persons.

Objects and places can be made trackable and controllable by being connected to low-cost wireless electronic devices. Simple RFID tags or Bluetooth beacons, based on low-cost communication protocols, can be attached to any kind of object to enable tracking its position and status and possibly to associate some digital information with it. More advanced devices integrating environmental or motion sensors can detect the past and present activities associated with objects or places. In addition, objects can incorporate digital actuators that enable remote control of their configuration or status, or they can perhaps be made autonomous.

For example, the hotel scenario could have the following features. Objects in rooms, such as remote controls, could contain RFID tags to communicate their presence and location. Projection screens in conference rooms could have some kind of Arduino-like controller to let people use mobile phones to raise and lower them. Conference room window curtains could autonomously regulate the lighting on the basis of the activities in the room. Walls could change the meeting rooms' shape on demand. From this perspective, autonomous robots (or robotized objects[9]) could be considered the highest end of the smart-thing spectrum.

People, even when they aren't actively using IoT technology, can still be first-class entities in the IoT vision. Simply because they have a mobile phone, their activities and positions can be sensed and they can be asked to act in the environment or supply sensing. In the hotel scenario, hotel management could continuously detect people's locations and activities in order to manage emergency situations in the most efficient way.

Software Infrastructures

To make things capable of serving purposes, we need software infrastructures (IoT middleware[1]) that support the "gluing together" of different things and provide ways to access the IoT system and its functionalities.

The gluing together involves four main technical issues. The first is interoperability. The interaction of heterogeneous things requires a set of shared communication protocols and data representation schemes, which involves more than just a way to identify things.[1] The study of this issue dates back to early IoT research; several proposals exist, and the road toward assessed standards is well paved.

The second issue is semantics. A common semantics for concepts must be defined to enable the cooperation and integration of things. Proposals for this exist that are grounded on standard Web technologies and on ontologies suited to the physically and socially embedded nature of the scenario.[10]

The third issue is discovery, group formation, and coordination. IoT system functionalities derive from the orchestrated exploitation of a variety of things, possibly involving a variety of users and stakeholders. In the hotel scenario, configuring conference rooms for slide presentations involves a projector, the lighting system, and the conference organizers and speakers. This requires a way to discover and establish relations between things and between things and humans and to coordinate their activities.[11]

The final issue is context awareness and self-adaptation. System components' inherent unreliability and mobility (for example, chairs or flip charts in the hotel conference center could come and go, be moved, or be placed in areas without wireless connections) make it impossible to anticipate which things will be available and for how long. This requires discovery and coordination mechanisms that can dynamically adapt to their context.[4]

Approaches to user access to things' functionalities and capabilities are currently dominated by the WoT vision. The idea is to expose things' services and functionalities in terms of REST (Representational State Transfer) services, enabling the adoption of web technologies to discover things and provision coordinated group services. Concerning middleware infrastructures, most proposals recognize the need to integrate three key features. The first is some basic infrastructure to support the WoT vision (that is, to expose things in terms of simple services). The second is some way to support, according to a specific coordination model, the discovery of things (and their associated services) and the coordinated activities of groups of things. The third is solutions that enable services and applications to adapt in a context-aware, unsupervised way.[1,5,7,12]

Services and Applications

The term "IoT system" generally refers to a set of IoT devices and the middleware infrastructure that manages their networking and interaction. Specific software can be deployed logically above an IoT system to orchestrate system activities to provide both specific services and general-purpose applications (or suites of applications).

Providing specific services means enabling stakeholders and users to access and exploit things and direct their sensing or actuating capabilities. This includes coordinated services that access groups of things and coordinate their capabilities. For instance, in a hotel conference room, besides providing access to and control of individual appliances, a coordinated service could, by accessing and directing the lighting system, the light sensors, and the curtains, change the room from a presentation configuration to a discussion configuration.

General-purpose applications or suites, on the other hand, are more comprehensive software systems that

- regulate the functioning of the overall IoT system (or parts of it) to ensure its specific overall behavior and
- provide a harmonized set of services to access the system and (possibly) its configuration.

In the hotel scenario, applications could control the overall heating and lighting systems and give hotel clerks tools to reconfigure services and associated parameters.

Depending on the scenario, you can think of IoT systems in which services either exist only in the context of some general application or are deployed as stand-alone software.

The Key Abstractions

From the analysis of the common features, I identified the following key abstractions for the development of IoT systems (see Figure 1).

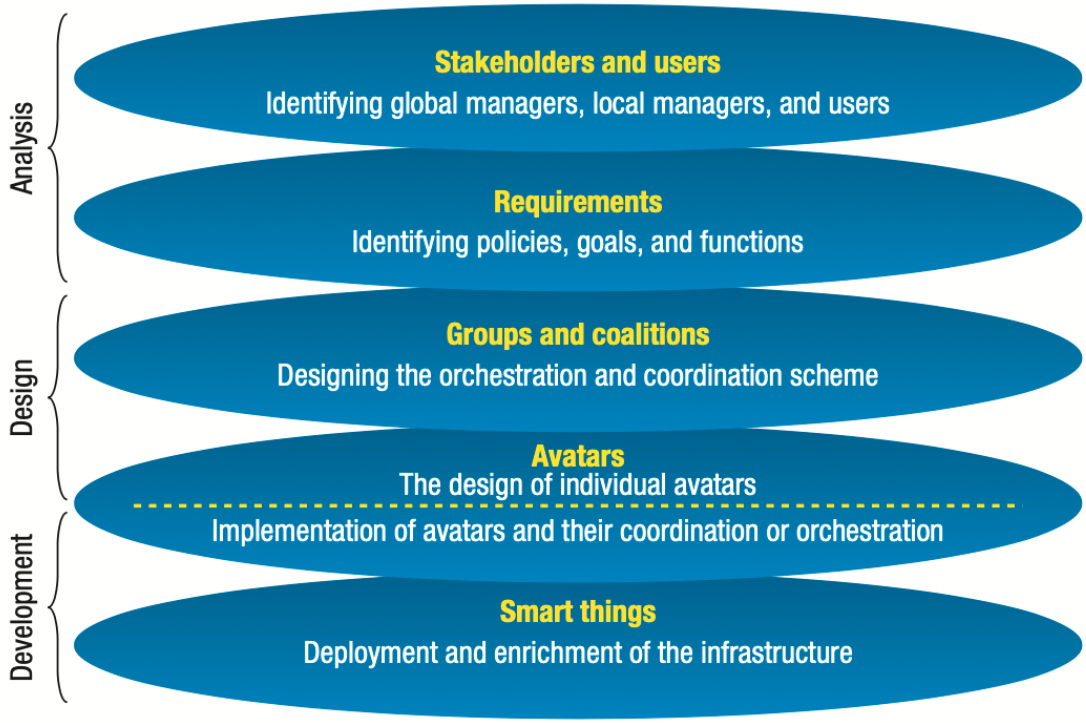

Figure 1. Key abstractions for the engineering of Internet of Things (IoT) systems. These abstractions could be the basis for a general IoT-oriented software engineering discipline, including detailed models, methodologies, and guidelines.

Stakeholders and Users

The first activity in the analysis of a system-to-be is to identify the actors: the system stakeholders and users. These persons or organizations will own, manage, or use the system and its functionalities; the requirements should be elicited from them.

In IoT systems, the distinction between IoT services and applications, and the presence of a middleware to support them and manage things, leads to the identification of the following three key abstract classes of actors.

First, *global managers* are the owners of an IoT system and infrastructure or are delegates empowered to exert control over and establish policies regarding the configuration and functioning of its applications and

services. In the hotel scenario, the global manager corresponds to the system manager who controls the hotel's IoT system according to the hotel management's directives—for example, for deciding heating levels or surveillance strategies.

Second, *local managers* are the owners of, or delegates (permanent or temporary) given control over, a portion of the IoT system. They're empowered to enforce local control and policies on that portion. In the hotel scenario, these could be hotel guests, who can control the IoT system in their room, tuning its local parameters and exploiting its services according to their needs. Or, they could be conference organizers in charge of managing and configuring the conference rooms' services.

Finally, *users* are persons or groups that have limited access to the overall configuration of the IoT applications and services. That is, they can't impose policies on the IoT but nevertheless are entitled to exploit its services. In the hotel scenario, these include conference delegates who are authorized to access the conference facilities (for example, uploading presentations to the projector) but aren't entitled to modify the conference rooms' configuration.

These three classes of actors are of a general nature, outside the hotel scenario. For example, in the scenario of energy management in a smart city, they could correspond to, respectively, city managers, house or shop owners, and private citizens and tourists. In the urban-transportation-management scenario, they could correspond to, respectively, mobility managers, parking-lot owners or car-sharing companies, and private drivers.

Requirements: Policies, Goals, and Functions

Once the key actors are identified, analysis of the IoT system-to-be can't simply reduce to understanding from them the services that things or group of things must provide. The analysis must also account for a more comprehensive approach to requirements elicitation.

Regarding functional requirements, the analysis should consider—beside those things that have basic sensing or actuating functionalities—the presence of smarter things that can be activated to autonomously perform long-term activities associated with their nature and their role in their sociophysical environment. These things can range from simple cleaning robots to more sophisticated autonomous personal assistants.[9]

Regarding nonfunctional requirements, IoT applications shouldn't just provide a suite of coordinated functionalities. They should also globally regulate IoT system activities continuously, according to the system stakeholders' policies and objectives.

So, in addition to analyzing the specific functionalities to deliver, the analysis must also identify the policies and goals to be associated with services and applications—that is, the desirable state of affairs to strive for in the context of the system's operational scenario.

Accordingly, the key abstract classes of requirements to be identified are *policies*, *goals*, and *functions* (see Figure 2).

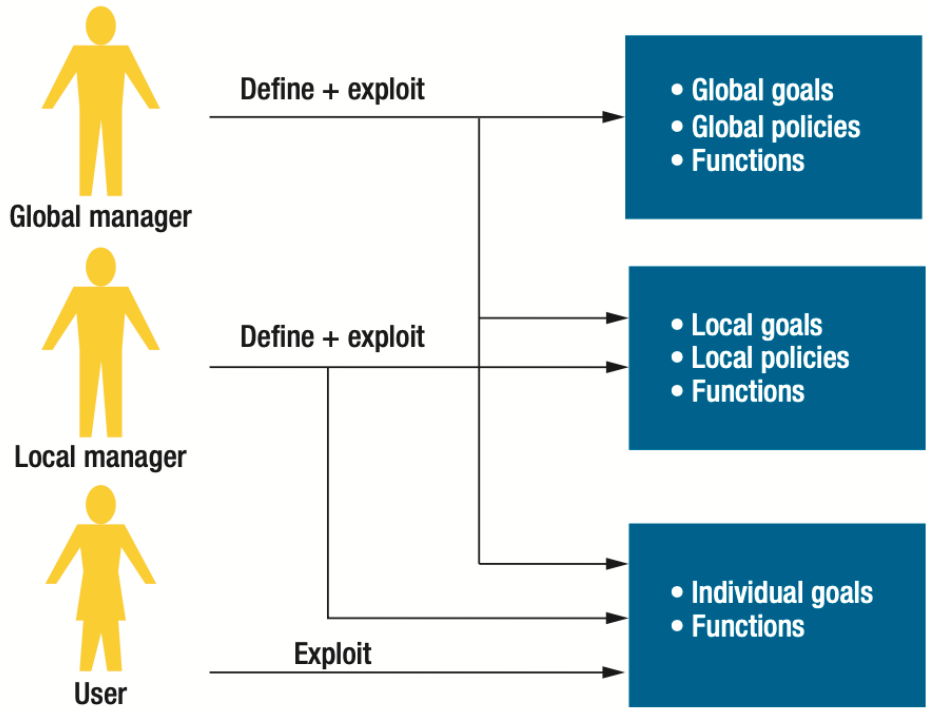

Figure 2. IoT actors and their role in defining and exploiting IoT system requirements. The general classes of requirements are policies, goals, and functions.

Policies express desirable permanent configurations or states of functioning of an overall IoT system (global policies) or portions of it (local policies). In the hotel scenario, global policies can be defined, for example, to specify each room's maximum occupancy and have local cameras monitor it, to suggest that people move to different rooms when needed. Policies are always active and actively enforced. The definition of global and local policies is generally determined by the global managers, although local managers can be authorized to enforce temporary local policies on local portions of the system (if this doesn't conflict with the policies the global managers enforce).

Goals express desirable situations that, in specific cases, can or should be achieved. A goal's activation might rely on specific preconditions (the occurrence of specific events) or a specific user action (that is, a goal's activation is invokable as a service). The typical postcondition (deactivating the pursuit of a goal) is the achievement of the goal itself. For example, in the hotel scenario, an evacuation procedure could activate when a fire is detected (the precondition); its goal (and postcondition) would be the quick evacuation of all the people in the building. Toward that goal, the procedure's activation could trigger digital signage and controllable doors to rationally guide people toward the exits. The definition of global and local goals is generally assigned to global, and sometimes local, managers. Users can sometimes be authorized to activate simple local goals (or goals associated with individual things) as a service.

Functions define the sensing, computing, or actuating capabilities of individual things or a group of things, or specific resources that are to be made available. Functions are typically accessible as services and can sometimes involve coordinated access to the functions of a multitude of things. In the hotel scenario, examples include the individual functionalities of the appliances in a conference room (for example, opening or closing a curtain or displaying or changing a slide in a projector), as well as more complex functionalities achieved by orchestrating things (for example, setting up a room for presentation by closing curtains and switching off lights). Functions and the associated services are typically defined by global and possibly local managers but are also exploited by the IoT system's everyday users (for example, the hotel guests and conference attendees).

Avatars and Coalitions

The things involved in implementing the functions, goals, and policies can correspond to a variety of objects and devices (besides places and humans), each relying on a plethora of technologies and capabilities. Accordingly, from both the gluing-software-infrastructure and software engineering viewpoints, it's necessary to define higher-level abstractions to handle the design and development of applications and services and to harmoniously exploit IoT system components.

Most proposals for programming models and middleware acknowledge this need by virtualizing individual things in some sort of software abstraction.[2] The WoT perspective abstracts things and their functionalities in terms of generic resources, to be accessed through RESTful calls, possibly associating external HTTP gateways to individual things if they can't directly support HTTP interfacing.[13] Other approaches suggest adopting a standard service-oriented architecture (SOA).[14] Also, some proposals consider associating autonomous software agents with individual things,[3] which particularly suits the fact that some goals to be pursued in autonomy might be associated with things.

In addition, regarding the provisioning of specific services and applications, some things make no sense as individual entities, as I already stated. Those things are to be considered part of a group and provide their services as a coordinated group. This applies to cases in which the group's functionalities complement each other and must be orchestrated.[3] It also applies to cases in which a multitude of equivalent devices must be collectively exploited, abstracting from the presence of the individuals.[4,12]

With these considerations in mind, to create a synthesis from a variety of proposals, I suggest the unifying abstractions of *avatars* (or groups of avatars) and *coalitions* (see Figure 3).

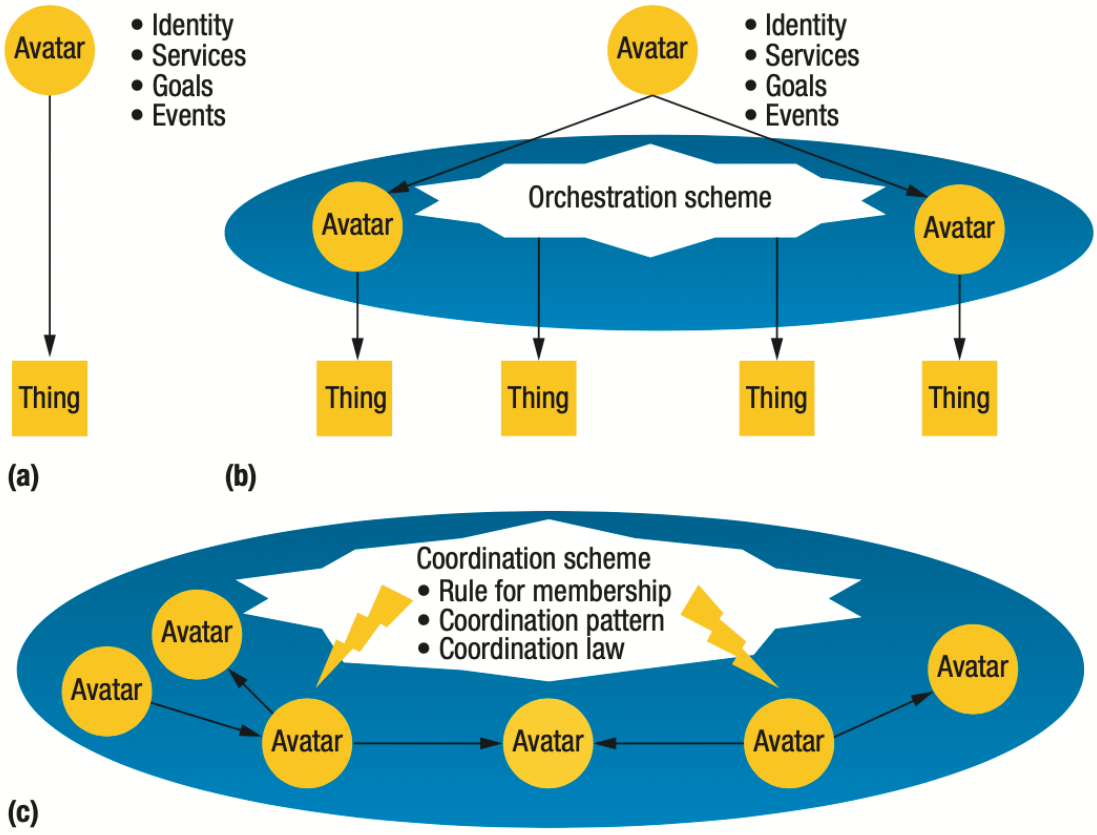

Figure 3. Combinations of avatars. (a) An individual avatar. (b) An avatar group. (c) An avatar coalition. An avatar is a general abstraction for individual things or a group of things (and possibly other avatars) that contribute to define a unique functionality or service.

Avatars and groups. Borrowing the term from Michael Mrissa and his colleagues,[6] I define an avatar as the general abstraction for individual things and for a group of things (and possibly other avatars) that contribute to define a unique functionality or service. Avatars abstract away from the specific physical, social, or technological characteristics of the things they represent. They have four main characteristics:

- An avatar has a unique *identity* and is addressable. An avatar representing a group doesn't necessarily hide the inner avatars' identities, but it has its own identity.
- *Services* represent access points for exploiting avatars' unique capabilities. Depending on the things and functionalities a service abstracts, this exploitation could involve triggering and directing the sensing, computing, or actuating capabilities or accessing managed resources.
- *Goals*, in the sense of desired states of affairs, can be associated with avatars. A goal might have a precondition for autonomous activation or might be explicitly activated by a user or another avatar.
- *Events* represent specific states of affairs that an avatar can detect and that might be of interest to other avatars or users. Those avatars or users can subscribe to those events.

In the hotel scenario, individual avatars could be associated with conference room appliances and objects—for example, the projector and window curtains. Groups of avatars could help set up a conference room for a presentation by coordinating several appliances or objects. Such a group might be able to perceive events in the room (such a door opening) and react accordingly (for example, to preserve specific lighting conditions).

Clearly, a group of avatars requires an *internal orchestration* scheme for coordinating the activities or functionalities of the things (or other avatars) it includes. Such a scheme defines the internal workflow of activities among the things and avatars and the constraints or conditions they're subjected to. The scheme might also account for contextual information, to make the group's activities context-aware. (For example, the service for setting up a conference room for a presentation must adapt to the current lighting conditions.) Most IoT middleware and programming approaches address the need to define orchestration schemes and constraints regarding the access and use of things (or groups of things).[4,5,14]

Coalitions. Borrowing the term from the area of multiagent systems,[3] I define a coalition as a group of avatars that coordinate each other's activities to reach specific goals or enact specific policies. Accordingly, coalitions might be temporary or permanent. Unlike avatar groups, coalitions don't necessarily have an identity or provide services.

A coalition must be defined in terms of a *coordination scheme*, which includes the following:

- *Membership rules* specify the conditions upon which an avatar could or should enter a coalition. From the viewpoint of individual avatars, the act of entering a coalition can be represented by the activation of a specific goal whose preconditions correspond to the membership rules.
- A *coordination pattern* defines the interaction protocol and shared strategy by which the coalition members must interact. This pattern might include an explicit representation of the goal that activates the coalition. However, such a goal can also be implicit in the definition of the protocol and strategy.
- A *coordination law* expresses constraints on how the avatars in the coalition act and interact. This includes the possibility to subscribe and react to events occurring in the coalition.

The view of avatar coalitions can be useful to realize policies or aggregate groups of avatars based on their similarity, so as to make them work collectively in a mission-oriented way without forcing them into specific identity-centered orchestration schemes. This is coherent with the idea of collective self-organized programming in sensor networks and IoT systems,[4,12] enabling the dynamic formation of service ensembles focused on short-term goals.

In the hotel scenario, a permanent coalition of all the thermostats and air-conditioning fans in the hotel's public areas could continuously coordinate to preserve specific climate conditions, according to the hotel management's policies. Also, when a fire alarm activates, a temporary coalition of the hotel's digital signage could provide evacuation directions in a coordinated way. For these two examples, membership in these coalitions would rely not on the avatars' unique identities but on their attributes (for example, public-area =

true) and preconditions.

From Design to Implementation

The design of avatars, groups, and coalitions abstracts from implementation details. From my perspective, and on the basis of my analysis of the state of the art, such design concepts are abstract enough to tolerate implementation above most existing systems and infrastructures.

Although a detailed discussion of implementation issues is outside this article's scope, the following general considerations are worth reporting.

Infrastructural needs. Increasingly, new IoT applications and services will have to be deployed over an existing IoT hardware infrastructure, possibly with some associated middleware. So, analyzing an infrastructure's characteristics and limitations will be a prerequisite for developing software over it. This is because deployment of the desired IoT services and applications might require enriching and updating the hardware infrastructure and its appliances.

For instance, in the hotel scenario, the conference room light switches might require updating to make them wirelessly controllable and make it possible to virtualize them as avatars with associated set-light-level services.

Implementing avatars. The abstraction of avatars is of a very general nature, but the specific implementation model clearly depends on the adopted middleware infrastructure and programming model. By adopting an agent-based model and associated agent-based middleware, developers can straightforwardly implement an avatar as a software agent, with its services, goals, and ability to sense events.[3,15] The adoption of a RESTful WoT approach is viable as well. Although RESTful architectures can't directly accommodate the stateful concepts of avatar goals and events, most WoT models and middleware recognize the need to somehow incorporate similar concepts.[2]

Implementing coordinators. The implementation of orchestration of avatar groups and coordination schemes for coalitions depends strongly on the adopted middleware and its underlying coordination models. Yet, as a general guideline, for relatively small groups and coalitions with well-defined, predictable coordination needs (such as in setting up a conference room for a presentation), the best solution might be assessed SOA orchestration models[14] or rule-based orchestration engines.[5] For large-scale groups and coalitions in dynamic situations (such as the coalition of digital signage for emergency evacuation), coordination models based on aggregate programming[12] or nature-inspired mechanisms[4] are preferable.

As technologies mature and real-world experiences accumulate, more research on software engineering for IoT systems will be needed. This research might exploit overlap with related engineering areas,[3,4] eventually leading to the identification of general models and methodologies—and associated tools—for IoT-oriented software engineering.

References

1. M.A. Razzaque et al., "Middleware for Internet of Things: A Survey," *IEEE Internet of Things J.*, vol. 3, no. 1, 2016, pp. 70–95.
2. J. Heuer, J. Hund, and O. Pfaff, "Toward the Web of Things: Applying Web Technologies to the Physical World," *Computer*, vol. 48, no. 5, 2015, pp. 34–42.
3. N. Spanoudakis and P. Moraitis, "Engineering Ambient Intelligence Systems Using Agent Technology," *IEEE Intelligent Systems*, vol. 30, no. 3, 2015, pp. 60–67.
4. F. Zambonelli et al., "Developing Pervasive Multi-agent Systems with Nature-Inspired Coordination," *Pervasive and Mobile Computing*, vol. 17, part B, 2015, pp. 236–252.
5. L. Yao, Q.Z. Sheng, and S. Dustdar, "Web-Based Management of the Internet of Things," *IEEE Internet Computing*, vol. 19, no. 4, 2015, pp. 60–67.
6. M. Mrissa et al., "An Avatar Architecture for the Web of Things," *IEEE Internet Computing*, vol. 19, no. 2, 2015,

pp. 30–38.

7. E. Latronico et al., “A Vision of Swarmlets,” *IEEE Internet Computing*, vol. 19, no. 2, 2015, pp. 20–28.
8. P. Patel and D. Cassou, “Enabling High-Level Application Development for the Internet of Things,” *J. Systems and Software*, May 2015, pp. 62–84.
9. H. Agrawal, S. Leigh, and P. Maes, “L’evolved: Autonomous and Ubiquitous Utilities as Smart Agents,” *Proc. 2105 ACM Int’l Joint Conf. Pervasive and Ubiquitous Computing* (UbiComp 15), 2015, pp. 487–491.
10. C. Perera et al., “Context Aware Computing for the Internet of Things: A Survey,” *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, 2014, pp. 414–454.
11. L. Atzori, D. Carboni, and A. Iera, “Smart Things in the Social Loop: Paradigms, Technologies, and Potentials,” *Ad Hoc Networks*, July 2014, pp. 121–132.
12. J. Beal, D. Pianini, and M. Viroli, “Aggregate Programming for the Internet of Things,” *Computer*, vol. 48, no. 9, 2015, pp. 22–30.
13. G. Bovet and J. Hennebert, “Offering Web-of-Things Connectivity to Building Networks,” *Proc. 2013 ACM Conf. Pervasive and Ubiquitous Computing Adjunct Publication*, 2013, pp. 1555–1564.
14. C. Sarkar et al., “A Scalable Distributed Architecture Towards Unifying IoT Applications,” *Proc. 2014 IEEE World Forum on Internet of Things*, 2014, pp. 508–513.
15. J.P. Jamont, L. Médini, and M. Mrissa, “A Web-Based Agent-Oriented Approach to Address Heterogeneity in Cooperative Embedded Systems,” *Trends in Practical Applications of Heterogeneous Multi-agent Systems: The PAAMS Collection*, Springer, 2014, pp. 45–52.

Franco Zambonelli *is a professor of computer science at the University of Modena and Reggio Emilia. His research interests include pervasive computing, multiagent systems, collective adaptive systems, and software engineering for large-scale systems. Zambonelli received a PhD in computer science and engineering from the University of Bologna. He is the co-editor in chief of* ACM Transactions on Autonomous and Adaptive Systems *and is on the editorial boards of the* Journal of Pervasive and Mobile Computing*, Computer Journal, and* Journal of Agent-Oriented Software Engineering*. He’s on the steering committee of the IEEE International Conference on Self-Adaptive and Self-Organizing Systems. He’s an ACM Distinguished Scientist, a member of Academia Europaea, and an IEEE Fellow. Contact him at franco.zambonelli@unimore.it.*