Developing Agent-based Smart Objects for IoT Edge Computing: Mobile Crowdsensing Use Case

Teemu Leppänen[1], Claudio Savaglio[2], Lauri Lovén[1], Wilma Russo[2], Giuseppe Di Fatta[3], Jukka Riekki[1], Giancarlo Fortino[2]

[1] Center for Ubiquitous Computing, University of Oulu, Finland
teemu.leppanen@oulu.fi, lauri.loven@oulu.fi, jukka.riekki@oulu.fi
[2] Department of Informatics, Modelling, Electronics and Systems, University of Calabria, Italy
csavaglio@dimes.unical.it, w.russo@unical.it, g.fortino@unical.it
[3] Department of Computer Science, University of Reading, United Kingdom
g.difatta@reading.ac.uk

Abstract. Software agents have been exploited to handle the inherent dynamicity in the Internet of Things (IoT) systems, as agents are capable of autonomous, reactive and proactive operation in response to changes in their local environment. Agents, operating at the network edge, enable leveraging cloud resources into the proximity of the user devices. However, poor interoperability with the existing IoT systems and the lack of a systematic methodology for IoT system development with the agent paradigm have hindered the utilization of software agent technologies in IoT. In this paper, we describe the development process and the system architecture of a mobile crowdsensing service, provided by an agent-based smart object that comprises agents in both edge and user devices. Mobile crowdsensing is an example of such an application that relies on large-scale participatory sensor networks, where participants have active roles in producing information about their environment with their smartphones. This scheme introduces challenges in handling dynamic opportunistic resource availability, due to mobility and unpredicted actions of the participants. We present how ACOSO-Meth (Agent-oriented Cooperative Smart Object-Methodology) guidelines the development process systematically from the analysis to the actual agent-based implementation of a crowdsensing service. The implementation is done with the ROAgent framework that utilizes resource-oriented architecture and REST principles to integrate agent-based smart objects seamlessly with the programmable Web.

Keywords: Smart Object · Edge computing · Internet of Things · Agent-based computing · Programmable Web

1 Introduction

The Internet of Things (IoT) vision of a seamless integration of the cyber and physical worlds can be realized with Smart Objects (SO) as the fundamental

building blocks [1]. These are common objects of any context (industry, entertainment, healthcare, etc.) able to interact with conventional computer systems, human users and the physical environment, thanks to their embedded sensing, processing, communication and actuation units. Moreover, SOs exhibit intelligence, autonomy, reactivity, proactivity and social skills in their operation, thus enabling, in theory, straightforward provision of advanced, decentralized, and cyberphysical services. In practice, however, the development of SO's requires established, effective and flexible metaphors, techniques, methods and tools for systematically conceptualizing, designing and implementing complex, autonomous and interactive SOs. In addition, interoperability is required between the SOs and the existing IoT systems.

The SO features listed above can be achieved with the agent-based computing paradigm [2]. Agents are natively autonomous, reactive, proactive, social and, in some cases, mobile, being able to relocate themselves at runtime. With agents' capabilities, SOs can observe their operational environment, react to changes and interact with other components and with each other as a multiagent system (MAS). In such a way, agent-based SOs are enabled to concretely bring smartness, autonomy and interactivity into the operation of IoT systems [3].

Recently, edge computing [4] brings computational power and data storage into the proximity of user devices. The aim is to improve IoT application execution by reducing latencies and providing more bandwidth locally. Benefits are seen in robustness and security, and also, data traffic on the backbone network is reduced. However, such a decentralized model requires capabilities for autonomy, interoperability and smartness in application execution that takes the local circumstances into account. Here, agent-based SO's can be exploited in the development and implementation of such distributed applications. Mobile crowdsensing [5] is an example of a distributed IoT application, which is beneficial to leverage into the edge and where agent-based SO enables context-aware operation between the edge and user devices. Application components in the edge perform computationally heavy tasks and interact with resource-constrained user devices, i.e., smartphones, that perform data collection in dynamic and opportunistic settings as guided by agents.

In this paper, we analyze, design and implement a mobile crowdsensing SO as an IoT edge application. We tackle the methodology and interoperability challenges by following the guidelines provided by the agent-oriented IoT development methodology ACOSO-Meth [6] and the resource-oriented architecture (ROA) [7] principles. We utilize the ROAgent framework [8,9,10] to develop the agent-based SO, where the functionality is distributed as a MAS across the edge devices and user devices. The SO exposes a service on the edge platform that, first, provides a set of smartphones that meet specific criteria for participation into crowdsensing campaigns and, second, interacts with mobile agents in the smartphones to execute the campaign, as in [11]. The analysis, design and implementation of such a multiagent-based SO leads to a real-world prototype.

The rest of the paper is organized as follows. In Section 2, we describe the operation of crowdsensing SO as a MAS. In Section 3, we present how the crowd-

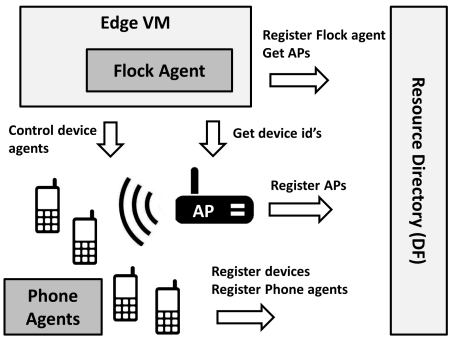


Fig. 1: System architecture of crowdsensing SO with the ROAgent framework.

sensing SO is developed accordingly exploiting both the ACOSO-Meth agent-oriented approach and ROA resource-oriented paradigm. Section 4 then discusses implications of the approach and concludes the paper.

2 Agent-based Crowdsensing System

Crowdsensing frameworks are typically cloud-based and centrally-controlled [5]. However, by following such a model, it is difficult to react to changes in the opportunistic environment and to the participants' behavior, possibly resulting in a significant reduction of data quality. Conversely, as shown in previous works, e.g., [11,12,13], agents and MAS can be effectively exploited to address the dynamicity in crowdsensing applications. In fact, agent-based development also allows enhancing privacy and energy efficiency in context-aware way, particularly beneficial for resource-constrained devices, such as smartphones [11].

Figures 1 and 2 present the agent-based crowdsensing SO system architecture within the ROAgent framework. The framework fully complies with ROA and the fundamental idea is to facilitate Representational state transfer (REST) principles [7] for agent-based applications in IoT in a standardized way [8,9,10]. As discussed in [10], the ROA approach can be optimized for resource-constrained IoT devices, while still enabling agent operations and interactions with other system components. The main abstraction is a resource that can be anything that has a value, e.g., an IoT device, its sensors and data, an edge device in the network infrastructure running VMs, an external data source (e.g., a Web service), or a software agent. The resources are accessed through a RESTful uniform interface that is based on the combined semantics of HTTP methods, resource URLs and HTTP response codes. The ROAgent framework extends this uniform interface to agent operations and interactions. In addition, the framework follows FIPA specifications that enable a standardized way to integrate agents into a MAS. A set of FIPA components are implemented, i.e., Directory Facilitator (DF), Agent Management System (AMS) and Message Transport System (MTS). The DF is implemented as Distributed Resource Directory (DRD) [14].

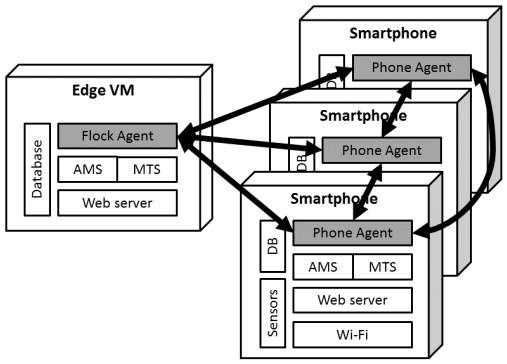


Fig. 2: The SO components in the system devices.

The framework operation requires system devices to register their resources into the DRD, which enables runtime lookup for available resources in a given location. MTS is enabled through a Web server with HTTP as the communication and interaction protocol that implements the uniform interface.

Figure 2 illustrates the agent-based crowdsensing SOs internal architecture and depicts the agents interactions as a MAS. Both Flock and Phone agents are implemented with the ROAgent platform [10]. The Flock agent platform is implemented within a Edge Virtual Machine (VM), which enables typical edge computing operations to instantiate and transfer the platform between edge devices. In the smartphones, the ROAgent platform runs an AMS to enable agent operations in the device and provides a database (DB) for data storage including agent knowledge base, i.e., SO data, and its interaction results. In detail, the agent's roles are the following. First, the Flock agent detects suitable participants, with the required sensor type, in a given area from the list its connected phones provided by the Wi-Fi access points (AP in Figure 1) and the DRD. Flocks of participants, with similar behavior, can be detected from these data that meet the campaign requirements. For example, a set of participants moving into the same direction can be utilized to provide data with better quality than with a single participant. Moreover, movement patterns of participants can be detected, to get information about their performance and to control participation. Second, the Flock agent and Phone agents, in the participants smartphones, interact as a MAS to execute the campaign aiming to fulfill its goals and reacting to participants' behaviors.

3 Systematic approach to Crowdsensing SO development

In this section, we apply the ACOSO-Meth [6] with ROA general guidelines [7] for the development of ROAgent-based crowdsensing SO. Initially, ACOSO-Meth and the ROAgent framework have been conceptually integrated in [15] by re-engineering the ROAgent framework according to the methodology.

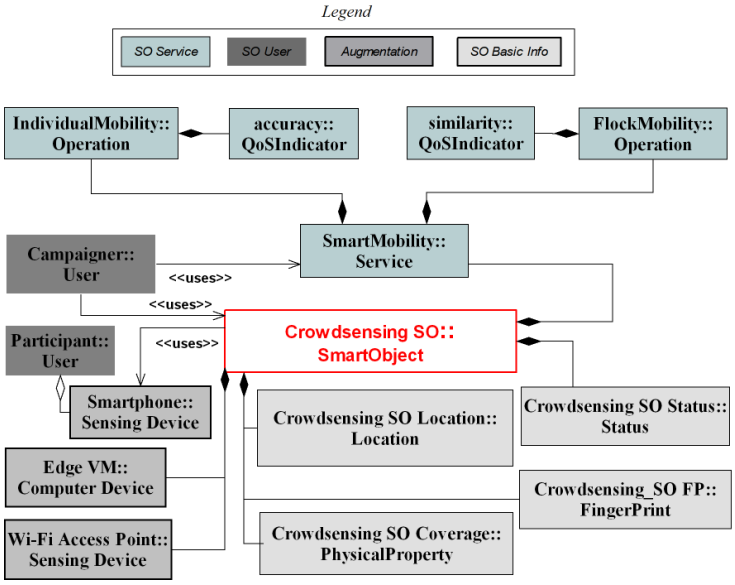


Fig. 3: Crowdsensing SO High-Level metamodel of the analysis phase.

We provide an actual real-world development of an agent- and SO-based crowdsensing service according to ACOSO-Meth and ROAgent framework guidelines. Regarding the service functionality, crowdsensing campaign participants are recruited based on their reputation, that can be monitored by an agent [11]. However, integrating participation into the existing traveling routes improves possibilities of participation [16], which justifies the presented crowdsensing service. In this paper, our focus is not on developing novel methods for agent-based crowdsensing participant recruitment nor for flock detection, e.g., [17,18].

3.1 Analysis phase

At the analysis phase, the ACOSO-Meth exploits of the SO High-Level metamodel to specify SO basic features and information. This is a very general metamodel, supporting the preliminary SO description regardless of any technological or behavioral specification. Indeed, the SO High-Level metamodel outlines how the SO can be identified, the provided services with related composing operations and needed augmentation devices (e.g., sensors, actuators, processing units), and who are the SO users [19,20].

The SO High-Level metamodel instantiated on the *Crowdsensing SO* is shown in Figure 3. The SO provides the SmartMobility *Service* for crowdsensing *Campaigners* as the users, who launch campaigns and need to recruit *Participants* meeting the campaign-specific requirements. For example, participants need to be in the target location in a specified timeframe. The SmartMobility Service is realized through two *Operations*: IndividualMobility to get the movement trace

of an individual participant and FlockMobility to detect participant flocks in the specified area. As service content, the SO provides a list containing timeseries of movement traces of individual participants and detected flocks that meet the campaign parameters. This list provides the campaigners a selection of possible participants to recruit. The quality of SmartMobility Service is expressed in terms of two metrics or *Quality of Service (QoS) Indicators*: the detection accuracy of the individual trace and the similarity of participants' movements within a flock. In order to provide SmartMobility Service, the Crowdsensing SO leverages augmentation devices such as smartphones (*Sensing Device*), the Edge VM and Wi-Fi APs (*Computer Device*). The lists of APs in the target area and their currently connected smartphones constitute the Crowdsensing SO *Status*. The distinctive SO information are reported by its *FingerPrint*: the SO is identified by its resource URL as the unique identifier and the service creator. Physical *Location* is in the premises of the Center for Ubiquitous Computing, University of Oulu, Finland, and the SO Coverage Area (*Physical Property*) is the center zone of city of Oulu (about 1 km x 1 km).

3.2 Design phase

At the design phase, the ACOSO-Meth guides the refinement of the SO High-Level Metamodel with the goal of obtaining a design metamodel. This, independently of technological specifications or low level details, highlights the functional components the SO, including communication, augmentation, service provision and information management, and their interactions, through the adopted computing paradigms and enabling mechanisms. By complying with the ROAgent framework design specifications [8,9,10], the SO High-Level Metamodel has been refined in the the ROA SO Metamodel [15].

The instantiated ROA SO Metamodel on the Crowdsensing SO is shown in Figure 4. Following the ROAgent framework, the Crowdsensing SO is based on a lightweight and platform-neutral agent. The agent functionality is defined through resource abstractions, interactions realized via RESTful uniform interface and the agent operations are event-driven.

The Crowdsensing SO lifecycle is specified in terms of Behaviors. A *Behavior* consists of one or more *Tasks*, which are coordinated by the *Crowdsensing SO Manager*. The tasks refer either to internal system operations (*SystemTask*) or to SO application-specific functionalities (*ServiceTask*). The latter category contains IndividualTrace tasks that provide individual movement traces, and the FlockTrace task that provides flock movement traces. Both tasks acquire input through the uniform interface and resource URLs, which guarantees transparent access regardless of resource type and location, according to ROA.

Within the ROAgent framework, each *Resource* is identified with a URL that is registered to the DRD. Resources needed for the presented crowdsensing application are listed in Table 1. With respect to the Crowdsensing SO, the resources are: (i) *Internal Resources* which are logically located within the devices hosting the SO ROAgents and their internal components, i.e. Edge VMs host the Flock agent and the smartphones host the Phone agents, (ii) *External*

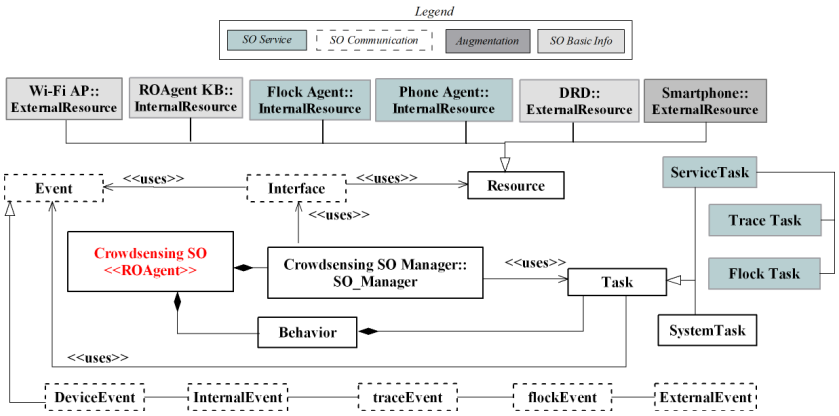


Fig. 4: Crowdsensing SO ROA Metamodel of the design phase.

Resources which are physically hosted in devices that are not part of the Crowdsensing SO, i.e., smartphones' sensors, or other devices co-located within the smart city infrastructure (i.e., DRD and Wi-Fi APs), and (iii) *Device Resources* which are SO sensors, actuators, and computing units, i.e., smartphones as agent platforms and their sensors. The SO ROAgentKB contains the current results of the agent programs, the agent's internal data, and the campaign parameters including required sensor data type, area, timeframe and the ending criteria.

The SO service resources in Table 1, including the resources provided by agents, are utilized as follows. The DRD, through the resource *drd*, allows retrieving a list of APs and information about individual devices connected to the framework. The WI-Fi APs, through the resource *wifi*, provide real-time information of smartphone availability, namely lists of their currently connected smartphones. The Flock Agent provides participant and flock movement traces, through the resource *flock*. The Phone Agents, through the resource *phone*, enable control the campaign execution. Smartphone sensors, through the resource *sensor_type*, provide requested sensor data for the campaigners.

As illustrated in Figure 4, each resource request arriving to the SO through the uniform *Interface*, the Crowdsensing SO Manager executes the operation defined in the request URL and transports the obtained data through a specific *Event*. Individual trace retrievals and flock detections are notified through the corresponding Trace and Flock *ServiceEvents*. The retrievals of Internal, External and Device resources (e.g., information about Crowdsensing SO status or smartphone availability) is performed through corresponding *InternalEvent*, *ExternalEvent* and *DeviceEvent* events.

Internal Flock agent architecture follows ROAgent specifications [8,9,10] as illustrated in Table 2. The architecture comprises four elements. The agent name is derived from the resource listing (URL) to be *flock*. The agent programs, that implement the SO functionality, are defined as *flock* and *trace*. The resources that agent utilizes as data sources or to interact with are defined as local, i.e.,

Table 1: The resources for the Crowdsensing SmartMobility Service.

External Resource		
/drd/{sensor_type}	Lookup for particular sensor in the DRD	*Infrastructure*
/wifi/devices	Connected devices (MAC addresses) in an AP	*Infrastructure*
Internal Resource		
/trace/{MAC_addr}	Movement traces	*Edge VM*
/flock	List of detected flocks	*Edge VM*
/flock/map	Visualization of detected flocks in a map	*Edge VM*
/flock/{flock_id}	List of phones in the identified flock	*Edge VM*
/flock/{flock_id}/trace	Movement trace of the identified flock	*Edge VM*
/phone	Control phone operation	*Smartphone*
Device Resource		
/{sensor_type}	Specific sensor and its data	*Smartphone*

Table 2: The Flock agent architecture.

Agent name	flock		
Code	flock	*Agent program code*	
	trace	*Agent program code*	
Resource	**Remote**	wifi	*List of AP URLs*
		phone	*List of Phone agent URLs*
State	**Knowledge base**	campaign	*Campaign parameters*
	Service content	flock	*List of detected flocks*
		trace	*List of individual traces*

in the hosting device, or remote, i.e., in other system component. Lastly, the agent state exposes the results of the agent program, e.g., flocks. The Phone agent architecture is similar as in [10], containing the agent program to process the sensor data for the particular campaign task, where the local resource is the utilized phone sensor.

3.3 Implementation phase

At the implementation phase, ACOSO-Meth guides the exploitation of a metamodel that can elicit the programming paradigms and technology solutions which concretely realize the designed SO functionalities of communication, augmentation, service provision and information management. The ROA SO Metamodel of the design phase is implemented with regard to the heterogeneous ROAgent platforms, resulting in the ROAgent-based SO metamodel. Its instantiation on the Crowdsensing SO is shown in Figure 5.

The Edge VM hosting the Crowdsensing SO is implemented with Ubuntu 16.04 LTS, where MySQL database is installed for the ROAgentKB and Node.js provides the ROAgent platform and a Web server for interactions. The Flock agent program is implemented with Python, which is run in Node.js using the library Python-shell. The Phone agents also follow the ROAgent architecture

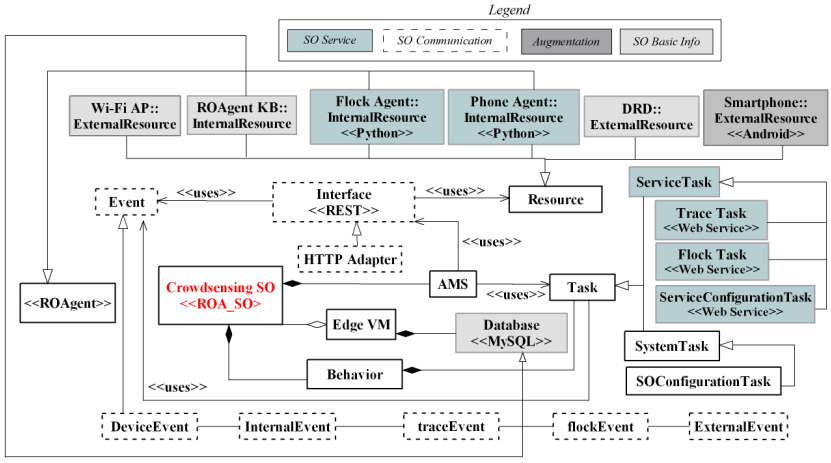


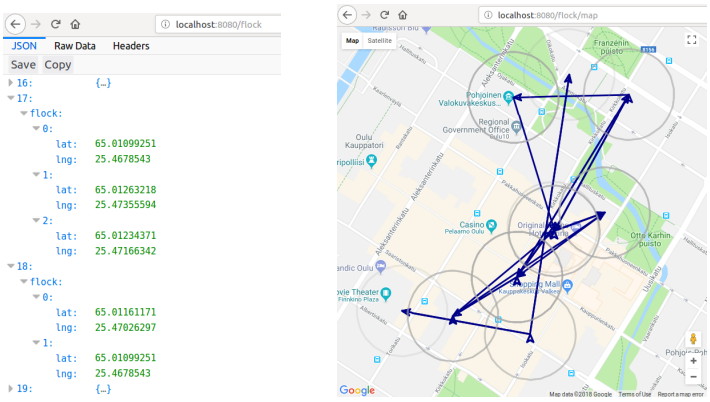
Fig. 5: Crowdsensing SO ROAgent Metamodel of the implementation phase.

and are implemented for the particular campaign task with Python for Android platform, as described in [11].

In addition to Figure 4, the Figure 5 reports the *ServiceConfigurationTask* that allows interacting with the ROAgents in the smartphones and Edge VM, for example, in order to set the campaign parameters. The *SOConfigurationTask* allows setting the Crowdsensing SO internal parameters, such as its resource URL as the identifier. The ROAgent platform *AMS* implements the uniform interface, presented in Table 1, for utilizing both internal and external resources in the service execution.

The SO REST-based uniform interface is implemented with HTTP, complying with the generic agent uniform interface presented in [8,9,10]. HTTP method GET is used to retrieve resource representations, i.e., state, where the content-type is a JSON object. The method POST is used to control sensors in smartphones, e.g., turn on/off and set sampling rates, as in [11]. For example, to get a list of APs in given area, the request is following: *GET <drd_IP_address>://drd/wifi?area=...*, which returns a JSON object containing list of AP URLs. To get the movement trace of a flock #123, the request is the following: *GET <so_IP_address>://flock/123/trace*, which returns a JSON object containing a list of APs in order that the smartphones in this flock have been connected to. Figure 6 illustrates, atop Google Maps, a set of flocks detected at given time by analysing smartphone connection patterns.

In this paper, the SO service part is implemented, but the actual campaign execution with agents (studied in [11]) is not considered. To provide data for movement trace and flock detection, we utilized the existing smart city infrastructure [21] that includes a set of 1300 Wi-Fi APs and their data of connected devices during the years 2007-2015 in the city of Oulu. This dataset provides information about Wi-Fi connection data from the APs, containing a list of



(a) JSON object (b) Google Maps visualization

Fig. 6: Illustration of the SmartMobility Service results.

smartphone MAC addresses of connected devices for each AP as time series. A subset of 100 traces during one day (February 3rd, 2015) in the city centre area was imported for the Crowdsensing SO. The data was pre-filtered to remove outliers, i.e., APs with strong signal strength, which appeared in almost all traces across the city.

The Crowdsensing SO service provides the following content to campaigners to assist in campaign implementation: (i) the flocks, consisting in movement patterns of participants, i.e., individual smartphones, which behave similarly for a period in the target area. This data are dynamic, i.e., flocks appear and disappear and their participants may change any time, and (ii) the movement traces of individual participants retrieved from the APs data in the target area. To detect flocks, the algorithm presented in [18] was used. A strict threshold of 85% similarity was used between the individual traces to be considered a flock. Individual traces have low accuracy due to original data collected from APs that cover a large area. The flock detection and trace accuracy could be improved with Wi-Fi AP signal strength data from the smartphones, as in [18].

4 Discussion and conclusion

Within the open and dynamic IoT edge computing scenario, the agent paradigm has been found useful [2]. Indeed, the paradigm allows exhibiting autonomous smart behavior in the collaborative execution of distributed IoT applications with both conventional computers and resource-constrained devices [10]. In mobile crowdsensing, data collection is a strictly coordinated effort in which the participants behaviors, such as unexpected actions or departures, can significantly affect the achievable results. The agent paradigm provides an approach

to handle such situations in runtime, while aiming to save resources in the participating devices.

However, the lack of systematic methodology, that leads from initial analysis to actual agent-based implementation, has been an obstacle in agent-based IoT application development. Another obstacle has been poor interoperability, in general with the variety of existing system infrastructure, but also from the existing agent framework point-of-view. To address these both issues, we integrated the ROAgent framework with the development methodology ACOSO-Meth. As result, the ROAgent framework provides in a systematic way for interoperability through a programming language- and platform-independent agent architecture and exposing the agent as a Web service with standardized Web technologies.

Overall, this paper contributed to the full-fledged development of an agent-based SO under the form of a MAS. Jointly exploiting agent-oriented and resource-oriented paradigms, the ROAgent framework enabled heterogeneous, resource-constrained agent-based SOs interoperating in IoT systems in a standardized way. This made the agent-based SOs' resources and services browsable and searchable within the Internet as for any non-agent based service, integrating the agent-based SO's into the programmable Web for machines as well. The ACOSO-Meth approach drove the SO development across the phases of analysis, design and implementation through a set of metamodels, featured by different levels of abstraction and aimed at seamlessly supporting IoT developers in such a complex process.

Our future work aims at developing further the agent-based SO concepts for programmable Web under the umbrella of the integrated ACOSO-Meth and ROAgent framework.

Acknowledgments. This work has been carried out under the framework of INTER-IoT, Research and Innovation action - Horizon 2020 European Project, Grant Agreement #687283, financed by the European Union.

References

1. Kortuem, G., Kawsar, F., Sundramoorthy, V., & Fitton, D.: Smart objects as building blocks for the internet of things. IEEE Internet Computing, 14(1), pp. 44-51 (2010).
2. Savaglio, C., Fortino, G., Ganzha, M., Paprzycki, M., Badica, C., and Ivanovic, M.: Agent-Based Computing in the Internet of Things: A Survey, In: Intl. Symposium on Intelligent and Distributed Computing, pp. 307-320 (2017), Springer, Cham.
3. Savaglio, C., and Fortino, G.: Autonomic and cognitive architectures for the Internet of Things, In: Intl. Conf. on Internet and Distributed Computing Systems, pp. 39-47 (2015), Springer, Cham.
4. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: Vision and challenges. IEEE Internet of Things Journal **3**(5), 637-646 (2016).
5. Liu, J., Shen, H., Narman, H. S., Chung, W., Lin, Z.: A survey of mobile crowdsensing techniques: A critical component for the internet of things. ACM Transactions on Cyber-Physical Systems **2**(3) (2018).

6. Fortino, G., Russo, W., Savaglio, C., Shen, W., Zhou, M.: Agent-Oriented Cooperative Smart Objects: From IoT System Design to Implementation, IEEE Transactions on Systems, Man, and Cybernetics: Systems, pp. 1-18 (2017) https://doi.org/10.1109/TSMC.2017.2780618
7. Richardson, L., Ruby, S.: RESTful Web services. O'Reilly (2008).
8. Leppänen, T., Liu, M., Harjula, E., Ramalingam, A., Ylioja, J., Närhi, P., et al.: Mobile Agents for Integration of Internet of Things and Wireless Sensor Networks. In: IEEE Intl. Conf. on Systems, Man and Cybernetics, pp. 14-21 (2013). https://doi.org/10.1109/SMC.2013.10
9. Leppänen T., Riekki J., Liu M., Harjula E., Ojala T.: Mobile Agents-Based Smart Objects for the Internet of Things. In: Fortino G., Trunfio P. (eds) Internet of Things Based on Smart Objects, 29-48, Springer (2014). https://doi.org/10.1007/978-3-319-00491-4_2
10. Leppänen, T.: Resource-oriented mobile agent and software framework for the Internet of Things. Doctoral dissertation, University of Oulu, Finland, ISBN 978-952-62-1813-7 (2018).
11. Leppänen, T., Álvarez Lacasia, J., Tobe, Y., Sezaki, K., Riekki, J.: Mobile Crowdsensing with Mobile Agents. Auton Agent Multi-Agent Syst **31**(1), 1-35 (2017). https://doi.org/10.1007/s10458-015-9311-7
12. Bosse, S., Pournaras, E.: An Ubiquitous Multi-Agent Mobile Platform for Distributed Crowd Sensing and Social Mining. In: 5th IEEE Intl. Conf. on Future Internet of Things and Cloud, pp. 280-287 (2017).
13. Hu, X., Liu, Q., Zhu, C., Leung, V., Chu, T. H., Chan, H. C.: A mobile crowdsensing system enhanced by cloud-based social networking services. In: First Intl. Workshop on Middleware for Cloud-enabled Sensing, no. 3 (2013).
14. Liu, M., Leppänen, T., Harjula, E., Ou, Z., Ramalingam, A., Ylianttila, M., et al.: Distributed resource directory architecture in Machine-to-Machine communications. In: IEEE 9th Intl. Conf. on Wireless and Mobile Computing, Networking and Communications, pp. 319-324 (2013). https://doi.org/10.1109/WiMOB.2013.6673379
15. Savaglio, C., Russo, W., Fortino, G., Leppänen, T., Riekki, J.: Re-engineering IoT systems through ACOSO-Meth: the IETF CoRE based agent framework case study, In: 19th Workshop from Objects to Agents (WOA 2018), June 28-29, Italy, 2018.
16. Chon, Y., Lane, N., Kim, Y., Zhao, F., Cha, H.: Understanding the coverage and scalability of place-centric crowdsensing. In: Proceedings of the 2013 ACM Intl. joint Conf. on Pervasive and ubiquitous computing, pp. 3-12 (2013).
17. Kjaergaard, M. , Wirz, M., Roggen, D., Tröster, G.: Mobile sensing of pedestrian flocks in indoor environments using wifi signals. In: IEEE Intl. Conf. on Pervasive Computing and Communications, pp. 95-102 (2012).
18. Álvarez Lacasia, J., Leppänen, T., Iwai, M., Kobayashi, H., Sezaki, K.: A method for grouping smartphone users based on Wi-Fi signal strength. In: Forum on Information Technology **12**(3), 449-452 (2013).
19. Fortino, G., Rovella, A., Russo, W., Savaglio, C.: Towards cyberphysical digital libraries: integrating IoT smart objects into digital libraries. Management of Cyber Physical Objects in the Future Internet of Things, pp. 135-156. Springer 2016.
20. Fortino, G., Gravina, R., Russo, W., Savaglio, C.: Modeling and simulating internet-of-things systems: a hybrid agent-oriented approach. In: Computing in Science & Engineering, **19**(5), pp. 68-76 (2017).
21. Kostakos, V., Ojala, T., Juntunen, T.: Traffic in the smart city: Exploring city-wide sensing for traffic control center augmentation. IEEE Internet Computing, **17**(6), 22-29 (2013).