Service Oriented Computing for Humans as Service Providers

Sergio Laso[1], Javier Berrocal[1(✉)], José Garcia-Alonso[1], Carlos Canal[2], and Juan M. Murillo[1]

[1] Universidad de Extremadura, Cáceres, Spain
{slasom,jberolm,jgaralo,juanmamu}@unex.es
[2] ITIS Software, Universidad de Málaga, Málaga, Spain
canal@lcc.uma.es

Abstract. For the past twenty years, Service Oriented Computing has changed the way in which information technology was understood. The approach involves not only technological advances that have influenced the development of Software Engineering, such as Service Oriented Architecture, Web services, Service Choreography, or Microservices. In addition, it has also provided the pillars for the development of Cloud Computing, which has transformed how the business in Information and Communication Technology is developed. In that context, this work focuses on how Service Oriented Computing can also drive the integration of humans in the Internet of Things and Crowd Sensing loops by enabling them to act as service providers. The key to this is the deployment of services on mobile devices, in particular smartphones. The enormous penetration of these devices in today's society, together with the personal nature of the information they handle, open a new horizon for the development of services. Through them individuals are able to make personal information available to others. This paper depicts Human Microservices, an architecture that allows humans to be considered as service providers, and discusses the open challenges in the field that conforms one of the next frontiers for Service Oriented Computing.

Keywords: Service Oriented Computing · Human service providers · Smartphones

1 Introduction

Service Oriented Computing (SOC) [16] has been a driving force behind innovation in computer science for the last decades [19]. From Service Oriented Architectures [21] to Cloud Computing [17], SOC has had a deep impact, both in research and industry.

Additionally to this background, SOC is more active than ever [18], as there is still a lot of challenges to be addressed [8]. Recently, the advances in this area led to the paradigm of Everything as a Service [4], where any component in a system can be handled as a service.

M. Aiello et al. (Eds.): Papazoglou Festschrift, LNCS 12521, pp. 111–122, 2021.
https://doi.org/10.1007/978-3-030-73203-5_9

This situation is complemented by the enormous penetration and the increasing capabilities of smartphones and other smart devices. The constant presence of this kind of devices in everyday life has led to a more direct involvement of humans in SOC. On the one hand, most of the companies and services offered by the so called "collaborative economy" highly depend on service-based applications that need humans to perform some task in the real world [12], such as delivering some food, or driving somewhere. On the other hand, the presence of smart devices around people are producing an amount of information never seen before, allowing to provide new kinds of services related to these people.

Due to their nature, a significant amount of the information gathered by smartphones and other smart devices is personal. There is no doubt of the value that personal information has for the development of context-oriented services. There is also no doubt about the privacy and ethical problems associated with the management of such information. Proposals like People as a Service [11] address this issue by keeping personal information in their owners' devices, giving them back control over their data. However, to take advantage of the information stored in end user devices, there must be a standardised way to offer such information to approved applications.

We will refer to these services—based on smart devices owned by individuals and providing personal information about their owners to approved applications—as Human Microservices. In this paper we present the technological architecture needed to manage and provide them. The ultimate benefit is that humans and their context can be servitized and included in service oriented applications in a completely smart way through their mobile devices.

The rest of the paper is organised as follows. Section 2 presents the motivations and technological foundations behind this work, while Sect. 3 describes the architecture of the proposal and its implementation. Section 4 develops a use case in which we have applied Human Microservices. And finally, Sect. 5 discusses the findings of this work and the conclusions of this paper.

2 Motivations and Technological Foundations

The SOC paradigm uses services to support the development of rapid, low-cost, interoperable, evolvable, and massively distributed applications [18]. During the last few years this paradigm has changed the way in which software is developed, paving the way for the proliferation of smart devices and pervasive and ubiquitous applications [8].

Indeed, the emergence of technological foundations such as Service Oriented Architecture (SOA) [24], and the Semantic Web [5], and the development of standards and recommendations such as OpenAPI [15] and W3C Thing Description [13] has made easier the specification of services, and their deployment. Upon them, Service Composition allows the aggregation of multiple services or microservices for offering higher level services supporting complex tasks or business processes.

The driving goal of these technological advances is to facilitate the flexibility and seamlessly integration of distributed applications, which is also a key issue

in IoT applications. Typically, the business process flow of IoT systems depends on the services offered by several smart devices, that may even come from different manufacturers [21]. Consider for instance how the microservices of light bulbs of different trademarks in an office could be composed to be turned on or off depending on the brightness of the environment and the preferences of the employees in the room.

Up to now, the services run by IoT devices have been limited to sensing and changing the state of the environment. More complex tasks (such as storing or computing information), or the orchestration of services were limited to cloud environments, in which research efforts have being invested to address scalability [20] and vendor lock-in problems [14], in order to improve the execution of distributed applications.

Nowadays, the massive deployment of Internet-connected devices has fostered the consumption of services deployed on cloud environments for retrieving, storing or computing the sensed information. However, the specific requirements of IoT applications and the increasing computing capabilities of these devices is changing the way these services are deployed and consumed. Indeed, paradigms such as Edge or Mist Computing are fostering the deployment of microservices on end or near-to-the-end devices. This way, the Quality of Service (i.e., response time, network overload, data traffic, etc.) can be improved [7]. Similarly to cloud-based services, these microservices can be composed for executing complex tasks, and the same technologies for addressing scalability and vendor lock-in problems can be reused to solve some of the issues of these platforms. For instance, by increasing the horizontal deployment of services on the edge layer in order to increase the computational capabilities.

Nevertheless, the nature of IoT devices and the information they handle open new challenges on the services offered and how they are composed. For instance, paradigms such as Human-in-the-loop or User-in-the-loop [22] promote services centred on people and aware of their context [23], mobility or preferences, in order to personalise the behaviour of the environment. Microservices focused on offering human-related information (i.e., the preferences, needs or contextual situation of the users) are needed in order to allow other devices or even third-party entities to consume that information, and to compose services and business flows adapted to the users' needs. These services could be deployed in cloud environment but, as has been discussed before, some stringent requirements may benefit from their deployment near the user [7].

In the next section we present the enabling technologies for such a "close-to-the-user" deployment. First, we introduce a conceptual model, that we have called People as a Service (PeaaS), for storing, computing and providing the user's contextual information within his/her smartphone. Then, we present the current implementation of the model. This implementation is based on Human Microservices, a framework which provides a set of tools for designing, implementing and deploying APIs focused on offering (or storing) contextual information of the users from their mobile devices.

3 Humans as Service Providers

From our point of view, companion devices, more specifically smartphones, can take a much more active role in the integration of humans in the Internet of Things through the use of SOC. During the last few years we have witnessed how these devices have increased their computing and storage capabilities, and the number of built-in sensors in order to gather more information about their owners. Usually, the destination of these data is some storage infrastructure in the cloud. Instead, in order to address the computational requirements of modern service oriented applications, companion devices should be able of capturing, storing and processing information about the users in the device itself, in particular when this information is to be consumed in environments close the user.

3.1 People as a Service

PeaaS is a mobile-centric computing model which proposes using the smartphone's sensors and interactions with other devices in order to gather large amounts of information about the user's context. This information is processed by using the smartphone's computational capabilities in order to infer the virtual profile of the owner. The computed virtual profile is kept in the device and provided by means of services. This allow owners to keep their virtual identity under their own control and, at the same time, the consumers of such services are allowed to get fresh and updated personalised information.

The PeaaS model is based on four principles:

- Mobile devices as interfaces to people. Smartphones are usually Internet-connected. Therefore, they are the interface of humans to the virtual word –they support the virtual links with other people and devices.
- Virtual profiles. Smarthpones have a large number of sensors that collect information about their surroundings. PeaaS allows to compute all this information in order to create and store locally the virtual profile of the owner.
- Virtual profiles as a service. Building a virtual profile of the smartphone's owner is particularly useful if it can be queried by external entities. Virtual profiles are provided as a service to those who might wish to access that information (such as IoT devices or interested enterprises).
- User privacy. PeaaS guarantees that an individual's virtual profile is always exclusively kept in the owner's device. PeaaS allows the users to control and monitor the external entities consuming their information. By means of user-defined privacy rules, PeaaS empowers users to manage their privacy.

Serving virtual profiles allows the integration of humans in the loop by applying SOC together with mobile computing technologies. In addition, PeaaS allows a variety of information to be collected in order to infer higher-level knowledge about the users, such as their mood, the kind of place their current location corresponds to, or the people who are with them. Such analysis requires specific algorithms to process the data and to provide them as part of the virtual profile.

PeaaS first use case consisted in an advertising and a commercial platform called nimBees [2] based on Google Cloud Messaging and Apple Push Notification Service. This implementation had some restrictions due to the limitations of the mobile operating systems and its orientation to the mobile marketing domain. The current implementation of PeaaS is called Human Microservices, and it eases the deployment of microservices on smartphones. In the coming subsections we describe both implementations of the PeaaS model.

3.2 nimBees

NimBees [2] is a smart push notification system with advanced segmentation capabilities based on the user's virtual profile. Figure 1 shows the nimBees architecture compared with the reference architecture of PeaaS.

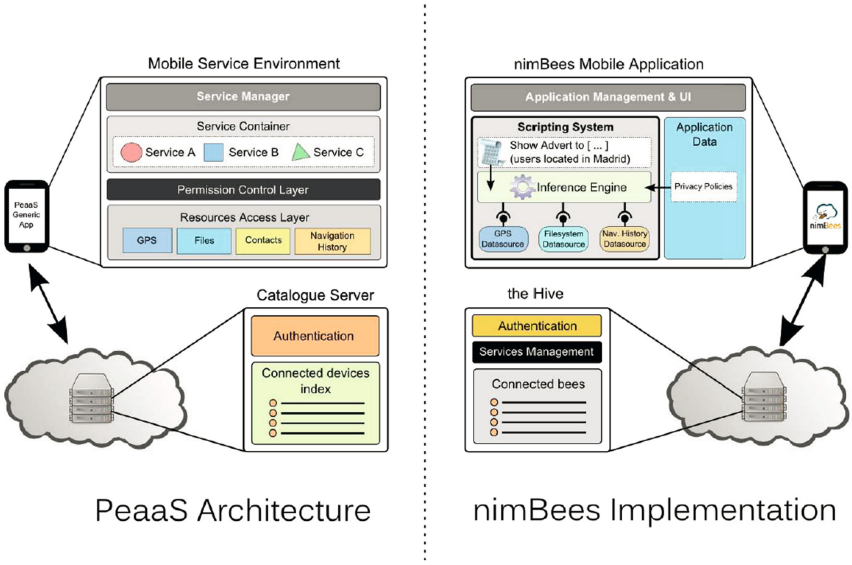

Fig. 1. Architectures of PeaaS and nimBees.

The system consists of a library that can be imported in almost any mobile application. This library allows these applications to receive segmented push notifications (i.e., notifications that are only shown or processed by the smartphones of the users meeting some specific requirements). Once the push notification reaches the smartphone, nimBees checks the owner's virtual profile in order to decide if s/he meets the requirements indicated in the notification. Only in that case, the notification is processed. Otherwise, it is ignored. The whole workflow of push notifications is processed transparently to the mobile applications finally receiving them.

More importantly, nimBees is also in charge of building the virtual profile by getting information from the different sensors of the smartphone and by processing it by means of inference rules [6]. It also allows consulting the virtual profile through push notifications. The richer the profile, the greater the segmentation can be. In addition, every personal datum stays in the owner's device. nimBees has a server-side, but only to manage the connected devices, the nimBees-based applications, and the delivery of the push notifications.

nimBees was a successful commercial implementation exploiting some of the features of PeaaS that allowed us to see its real potential. Thus, more recently we have been working on an implementation applying the SOC technologies and to directly provide the user's profile as a service.

3.3 Human Microservices

Human Microservices are services integrating a human in the loop, and focused on providing very personalised and updated contextual information about this person and his/her context and surroundings.

In order to implement Human Microservices, we have built a framework based on SOC technologies for the development and deployment of APIs on companion devices (mobile devices, smartwatches, etc.) for providing the owner's virtual profile. All the information is obtained at runtime through the device's sensors or from the profile stored in the smartphone.

Differently from nimBees, Human Microservices is not a library to be imported by third-party application developers. Instead, it proposes a set of tools and a development process that can be easily followed by any developer to design and implement the APIs that can be deployed on top of the virtual profile and the device's resources. Please, note that in this paper we will not focus on how the virtual profile or the information is computed, but only on how it is exposed by means of microservices.

Deployment Process. First, a development process has been defined in order to provide a guideline to developers about the different activities that should be performed and their sequencing. This process is based on technologies and standards already used for designing and implementing APIs that will deployed on cloud environments. Figure 2 shows the proposed steps.

- **API Definition.** First, the characteristics of the API are defined through the OpenAPI Specification (OAS) [15] following the same notation as if it were developed for a cloud environment. During the design of the interface, developers must bear in mind that the microservices will provide personal information about one single user. In that sense, the API could be a little different with respect to its design as a cloud microservice.
- **Generate Source Code.** In this step, the source code of the API is generated. Currently, different tools, such as OpenAPI Generator [3] or Guardrail [1], support the generation of source code (mainly the skeleton

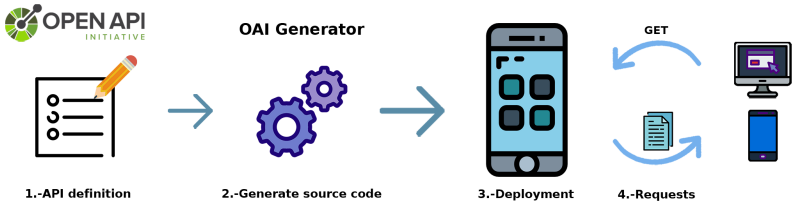

Fig. 2. Process for the development and deployment of Human Microservices.

and the schema of the API) from an OAS design. This scaffolding is based on its deployment on cloud environments. Therefore, as detailed bellow, an extension of the OpenAPI Generator tool has been developed to support the deployment of Human Microservices in smartphones and other companion devices.

- **Deployment.** At this point developers implement the business logic for each endpoint/microservice and configure the communication protocol. As it will be explained bellow, mobile devices present several limitations due to the restrictions imposed by the operating system and the mobile nature of the devices. Therefore, during the API scaffolding asynchronous communication protocols are supported. In this step, these protocols have to be configured, and the API is deployed following the procedure defined by the manufacturer.
- **Service consumption.** Finally, once the API is deployed, any device or third party can invoke the provided Human Microservices to request and consume information about the user, and to adapt the behaviour of the system accordingly.

In the next subsections, the three last steps and the related tools implemented to support them are described in more detail.

API Generator for End Devices - APIGEND. APIGEND is an extension of OpenAPI Generator that allows the generation of APIs that can be deployed on end devices. Currently, we provide support for its scaffolding and deployment on Android-based devices and other devices base on Esp32 Microcontrollers. APIGEND is available to any developer[1].

Figure 3 shows an example of generating an API using APIGEND. As it can be seen, the tool provides a website in which developers have to specify the framework (or the operating system) for which they want to generate the skeleton. Subsequently, In the *parameters* section, they have to indicate the following:

- *"openAPIUrl"*, which is the public URL to the specification of the API with OpenAPI. This specification can be stored in a git repository, a Dropbox folder, or any others web environment.

[1] https://openapi-generator-spilab.herokuapp.com.

– *"options"* allows the definition of different parameters that are not mandatory for the API scaffolding, but they help developers to adapt the source code generated to their needs. For instance, developers can specify the communication protocol they want to use to consume the endpoints. Currently, developers can choose between MQTT or FCM.

Fig. 3. Parameters to generate an Android API using the MQTT library.

In order to generate the API, the developer must click the *Execute* button and if there are no errors, a JSON response containing a link for downloading the generated API is produced.

Deployment. For deploying the API on end devices, the developer first have to configure the selected communication protocol. For MQTT, she only needs to indicate the connection parameters for the MQTT broker used to send and receive the messages consuming the microservices. For FCM, there is a guide for configuring it either using the Android IDE or manually[2]. Then, the developer has to implement the behaviour of the different endpoints defined (to get personal information from the stored virtual profile, to access information from the device's sensors, etc.). Finally, the developed API is deployed following the procedure recommended by the manufacturer.

Consuming Human Microservices. Due to their mobile nature and to limitations of the operating system, not every companion device supports the provision of services following synchronous communication. As indicated above, several asynchronous protocols (currently, MQTT and FCM) are supported by the framework.

For instance, for MQTT the content of the request is defined in JSON. The different parameters that must be specified are:

[2] https://firebase.google.com/docs/cloud-messaging/android/client.

- **Resource**: it indicates the *Tag* associated with the end point to be invoked.
- **Method**: it corresponds to the Id of the endpoint.
- **Sender**: ID or topic of the consumer performing the invocation, used for sending the reply.
- **Params**: parameters associated to the endpoint.

By default, the **MQTT topic** for sending requests to the API is the *title* of the specification without spaces. Listing 1.1 shows an example of an API request.

Human Microservices represent a shift in the role of companion and mobile devices in service oriented applications. It allows them to take full responsibility for storing, processing and exposing users' personal information, having greater control than they currently do, offering updated information, and seamlessly integrating humans in the loop.

4 Case Study

This section presents a case study where Human Microservices are deployed on an Android device and a smartwatch (based on the ESP32 microcontroller). To that end, the development process and the tools described in the Sect. 3.3 have been used.

The case study consists of an API that allows to obtain contextual and health information about people. The API will be deployed on a smartwatch to monitor an elderly person. This same API will be deployed on an Android device, acting as the caregiver's device. The mobile device can obtain information about the elderly person, such as location, body temperature, etc. by invoking the deployed API. In addition, the smartwatch is able to send messages or alerts by invoking the microservice deployed on the mobile device. This case study is fully available on Github[3].

The first step is to design and define the API. It is composed of four main microservices. **Get User** allows to obtain personal data stored in the device (such as name, age, etc.). **Get Body-Temperature** provides the body temperature of the elderly person thanks to the sensors of the smartwatch. **Get Location** provides the location through the built-in GPS sensor. Finally **Post Alert** allows to invoke the caregiver's microservices in order to send alerts (e.g., when an elder falls down and cannot get up).

The second step is to generate the skeleton of the API for both the Android device and the smartwatch, in this case MQTT is used as the communication protocol.

The third step is to deploy the APIs on both devices. On the one hand, it is necessary to configure the MQTT communication protocol for the connection between both devices. On the other hand, the microservices have to be implemented with the behaviour described above.

The last step is to invoke the deployed microservices. Listing 1.1 shows an example of invocation of the API developed to obtain the location. The request

[3] https://github.com/rurentero/HealthAlerts_M5Stick-C.

is sent to the main topic *'HealthAlerts'* (which is the name of the application) indicating the user id from whom the service consumer wants to get the information. Other topics schemes can be configured, for instance, per endpoint, tag, user, etc.

```
{
    "resource": "Status",
    "method": "getLocation",
    "sender": "caregiver293",
    "params": {}
}
caregiver.publish('HealthAlerts/user2234',request)
```

Listing 1.1. Content of the request to obtain the location by MQTT.

Thus, using Human Microservices, the elderly person virtual profile is connected to the Internet to be consumed by trusted external entities.

5 Discussion and Conclusions

Current information systems and applications are more focused on the users (e.g., their preferences, their context and their needs). From simple IoT applications [10], such as a smart light-bulb that is automatically turner on or off depending on the luminosity of the environment and the user's preferences, to complex business processes and systems [9], such as a Supply Chain or Smart Manufacturing Systems in which the socio-technical integration is crucial. The integration of people in the Internet is key but, in order to obtain the maximum benefit, this integration should follow standards and mechanisms that already exists.

PeaaS and its implementation Human Microservices allows the definition of APIs and services that can be deployed on companion devices following a specification broadly adopted by the industry. First, this reduces the learning curve, since developers can design the API and generate the source code by using tools already known by them. Secondly, it facilitates the integration of these Human Microservices in the business processes of the information systems because they can be consumed without requiring any knowledge about their implementation in the "server" side (in our case, the companion device).

In addition, the deployment of microservices near the user has positive implications with respect to resource consumption, latency, response times, and privacy. With a client-server model, end devices act as simple clients which constantly collect and send information to a cloud environment. This implies a significant consumption of some of their resources such as battery and data traffic, including heavy use of the network increasing latency and response times. In addition, this architecture can pose a risk to the privacy of users since the information is stored on servers and is beyond their control.

Finally, Human Microservices allows smartphones to change their role from pure consumers of information to also become providers of information. The APIs deployed on companion devices can be consumed by different information systems and third-party entities. For instance, the information gathered by the device's sensors and provided as Human Microservices can be consumed by a smart factory system, a smart city application and by a smart home IoT systems in order to adapt their behaviour to the current context and needs of the user. As future work, we currently work on evaluating the defined process and the related tools developed.

This work is founded on previous work on the Service Oriented Computing and Mobile Computing paradigms. We would like to thank Mike P. Papazoglou for his substantial contributions to these areas and, specially, for inspiring us in the development of the proposal presented in this paper.

Acknowlegments. This work was supported by the projects RTI2018-094591-B-I00, PGC2018-094905-B-I00 (MCI/AEI/FEDER, UE), the RCIS research network (RED2018-102654-T), the 4IE+ Project (0499-4IE-PLUS-4-E) funded by the Interreg V-A España-Portugal (POCTEP) 2014-2020 program, by the project UMA18-FEDERJA-180 (FEDER/Junta de Andalucia), by the Department of Economy and Infrastructure of the Government of Extremadura (GR18112, IB18030), and by the European Regional Development Fund.

References

1. Guardrail. https://github.com/twilio/guardrail
2. nimBees. http://www.nimbees.com
3. Openapi Generator. https://github.com/OpenAPITools/openapi-generator
4. Banerjee, P., et al.: Everything as a service: powering the new information economy. Computer **44**(3), 36–43 (2011)
5. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Sci. Am. **284**(5), 34–43 (2001)
6. Berrocal, J., García-Alonso, J., Murillo, J.M., Canal, C.: Rich contextual information for monitoring the elderly in an early stage of cognitive impairment. Pervasive Mob. Comput. **34**, 106–125 (2017). https://doi.org/10.1016/j.pmcj.2016.05.001
7. Berrocal, J., et al.: Early analysis of resource consumption patterns in mobile applications. Pervasive Mob. Comput. **35**, 32–50 (2017). https://doi.org/10.1016/j.pmcj.2016.06.011. http://www.sciencedirect.com/science/article/pii/S1541192163007797
8. Bouguettaya, A., et al.: A service computing manifesto: the next 10 years. Commun. ACM **60**(4), 64–72 (2017)
9. Cimini, C., Pirola, F., Pinto, R., Cavalieri, S.: A human-in-the-loop manufacturing control architecture for the next generation of production systems. J. Manuf. Syst. **54**, 258–271 (2020). https://doi.org/10.1016/j.jmsy.2020.01.002. http://www.sciencedirect.com/science/article/pii/S0278612520300029
10. Flores-Martín, D., Berrocal, J., García-Alonso, J., Murillo, J.M.: Towards a runtime devices adaptation in a multi-device environment based on people's needs. In: IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2019, Kyoto, Japan, 11–15 March 2019, pp. 304–309. IEEE (2019). https://doi.org/10.1109/PERCOMW.2019.8730859

11. Guillen, J., Miranda, J., Berrocal, J., Garcia-Alonso, J., Murillo, J.M., Canal, C.: People as a service: a mobile-centric model for providing collective sociological profiles. IEEE Softw. **31**(2), 48–53 (2013)
12. Huang, K., Yao, J., Zhang, J., Feng, Z.: Human-as-a-service: growth in human service ecosystem. In: 2016 IEEE International Conference on Services Computing (SCC), pp. 90–97. IEEE (2016)
13. Kaebisch, S., Kamiya, T., McCool, M., Charpenay, V.: Web of Things (WoT) thing description. First Public Working Draft W3C (2017)
14. Nguyen, D.K., Lelli, F., Papazoglou, M.P., van den Heuvel, W.: Blueprinting approach in support of cloud computing. Future Internet **4**(1), 322–346 (2012). https://doi.org/10.3390/fi4010322
15. OpenAPI Initiative: The OpenAPI Specification. https://github.com/OAI/OpenAPI-Specification
16. Papazoglou, M.P., Georgakopoulos, D.: Service-oriented computing. Commun. ACM **46**(10), 25–28 (2003)
17. Papazoglou, M.P., van den Heuvel, W.J.: Blueprinting the cloud. IEEE Internet Comput. **15**(6), 74–79 (2011)
18. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: state of the art and research challenges. Computer **40**(11), 38–45 (2007)
19. Papazoglou, M.P.: Service-oriented computing: concepts, characteristics and directions. In: Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003, WISE 2003, pp. 3–12. IEEE (2003)
20. Papazoglou, M.P.: Cloud blueprint: a model-driven approach to configuring federated clouds. In: Abelló, A., Bellatreche, L., Benatallah, B. (eds.) MEDI 2012. LNCS, vol. 7602, pp. 1–1. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33609-6_1
21. Papazoglou, M.P., Van Den Heuvel, W.J.: Service oriented architectures: approaches, technologies and research issues. VLDB J. **16**(3), 389–415 (2007)
22. Petrov, V., et al.: When IoT keeps people in the loop: a path towards a new global utility. IEEE Commun. Mag. **57**(1), 114–121 (2018)
23. Rosenberger, P., Gerhard, D.: Context-awareness in industrial applications: definition, classification and use case. Procedia CIRP **72**, 1172–1177 (2018)
24. World Wide Web Consortium: Web services architecture (2004). http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/