Towards a Methodology for creating Internet of Things (IoT) Applications based on Microservices

Edwin Cabrera, Paola Cárdenas, Priscila Cedillo, Paola Pesántez-Cabrera
Department of Computer Science
Universidad de Cuenca
Cuenca-Ecuador
{edwin.cabrera, paola.cardenas0108, priscila.cedillo, paola.pesantezc}@ucuenca.edu.ec

***Abstract*—The Internet of Things (IoT) represents the new industrial revolution, in which physical and virtual objects are interconnected. On the other hand, microservices architectures have broken the monolithic and centralized way to build software, and provide systems with high-quality characteristics (e.g., resilience, availability, modularity, and portability). Therefore, the idea of merging those technologies can constitute a powerful strategy to be applied in environments that demand the distribution and management of many IoT devices using high-quality software. In this context, several studies that integrate IoT with microservices solutions have been analyzed. However, most of these studies aim to satisfy the functional requirements related to software and hardware, without taking into account software engineering methodologies and good practices that allow the creation of software for IoT devices considering their distributed nature. Thus, this paper presents the first approach to an agile methodology that i) contemplates the main characteristics of the IoT and ii) guides the development of appropriate software solutions based on microservices architectures to manage IoT environments acknowledging the serious difficulties that microservices imply.**

Keywords*—Methodology, Microservices, Internet of Things, Domain-Driven Design, Agile, MicroIoT.

I. INTRODUCTION

The Internet of Things has revolutionized how several organizations perform their activities by interconnecting things to provide comfortability, availability, and functionality to users [1]. However, most IoT solutions consider only the functionality of devices, without thinking about the appropriate software architecture for managing the associated devices. The IoT involves multiple devices with different requirements, so it needs to be highly scalable due to new requirements that the environment demands.

On the other hand, a microservice is defined as "a small application that can be implemented, scales independently and has a unique responsibility" [2]. Thus, microservices can be applied in IoT solutions because they are endowed with quality characteristics such as interoperability, resilience, scalability, maintainability, among others [3].

Studies that propose methods, processes, or techniques that relate IoT domains to MicroServices Architecture (MSA) are few [4]. Additionally, IoT applications have a common problem in the software development process that is the lack of interest in the methods and processes related to software engineering; this is due to the importance given to immediate functionality without analyzing the software quality, evolution, and scalability. This fact leads to the creation of monolithic applications where all services are developed in a single shared code base among multiple developers, when these developers want to add, modify, or update services they must guarantee that all other services will continue working. Then, complexity increases as more services are added, limiting the ability of companies to innovate with new versions and features [5]. Therefore, this study proposes the creation of Microservices for the Internet of Things (MicroIoT), a methodology that allows the development of microservices-based applications. It can be implemented in any domain related to IoT.

Finally, the paper is structured as follows: Section II presents an overview of different studies examining an approach to a microservices architecture from the perspective of IoT software development. Section III presents the MicroIoT methodology and defines the roles played by those involved in the development of the methodology's life cycle. Section IV details the stages and activities that make up the methodology. Lastly, Section V draws some conclusions and examines potential future directions of research.

II. RELATED WORK

Many studies have focused on microservices and have associated them with IoT, it is the case of [6], which explains how microservices architectures can offer robust, low-cost solutions to manage the high frequency of data and real-time requirements in an IoT environment. It also proposes, as future work, to use microservices in a methodology employing the automation of DevOps. On the other hand, [5] establishes how microservices allow companies to manage large applications using a methodology in which small teams make incremental improvements and independent implementations. However, companies must consider the challenges of distributed systems and good management practices that help to assure characteristics such as agility, cost reduction, and granular scalability. Thus, [5] proposes, as future work, to establish processes to i) define the number of microservices and gateways in an IoT environment and ii) automate the implementation and management of microservices. Nonetheless, no research provides an overview of the software development process, the structural organization of software development companies or groups, and the tasks taking part in microservices. Thus, this paper proposes MicroIoT as a software development methodology offering the benefits of microservices to face the challenges of a distributed environment such as the Internet of Things.

III. THE MICROIOT METHODOLOGY

Software applications usually tend to follow an architecture with a monolithic approach, where all business logic is highly coupled. A monolithic system shows severe limitations when considering the distribution of systems to multiple users, the rapid deployment cycles, and the prevention of cascade failures (not provided by centralized software systems) [2]. On the other hand, IoT requires a specific approach to solve the problem of providing scalability and performance, clearly pointing to the distribution of effort among many small and

2474-2473/20/$31.00 ©2020 IEEE
DOI 10.1109/SCC49832.2020.00072

specialized services [7]. The development of software in IoT environments requires special considerations in the design and development of the system to be scalable and maintainable. Therefore, a microservice architecture requires many changes: a) the way of developing traditional software is replaced by decoupled programs which work together like a whole system, b) the organizational scheme changes according to the requirements that microservices imply, c) the way of managing the system becomes more complex because the system is formed by many microservices, and d) the adoption of tools to guarantee the best performance of microservices. Also, this architecture can be considered as an agile software development, where a general vision of the system is generated from the beginning. For this, the DevOps organizational approach will deal with the process of software development from the planning phase to the operations phase, including monitoring of microservices and system maintenance [8].

Thus, MicroIoT identifies the need for combining both concepts (i.e., DevOps, Agile) because agile methods often end up before the transition phase (i.e., delivery of the software to the users), whereas DevOps covers the operations defined for software deliveries; this is the interval in which the software is available to the users, but is continuously monitored and maintained by the developers. Consequently, the MicroIoT methodology builds on these concepts to execute the whole life cycle of an IoT software system, using microservices. It can be considered an agile methodology, where the development, testing, deployment, and operations activities are strictly aligned with their DevOps counterparts. This aspect has been addressed because quality deliveries require a high degree of automation within a short time, using tools and technologies [9]. Additionally, this methodology defines: i) *internal roles* related to the organization's development staff such as software architect, requirements analyst, development team: developer, tester, designer, database administrator, quality assurance team (Qa), and operations team (Ops); and ii) *external roles* addressing targeted stakeholders such as clients (e.g., manager/director of an organization, end-users) and domain experts.

IV. Stages And Activities Of The MicroIoT Methodology

The MicroIoT methodology will help organizations to leverage their key development resources. It has seven stages: a) analysis of requirements, b) domain-driven delivery design, c) architecture of the solution, d) validation, e) development and testing, f) deployment, and g) operations. Fig. 1 illustrates the life cycle that each delivery follows and shows the alignment of the MicroIoT stages with the corresponding phases of the DevOps reference architecture. Below, these seven stages and their activities are described.

A. *Analysis of requirements*

It is vital for planning the development of a software system or solution. Since MicroIoT focuses on agile development, it considers an iterative life cycle that allows defining continuous deliveries in a short time and thus reducing setup time in production. The activities in this stage are: i) *elicitation of requirements* to identify the high-level requirements of the system without worrying about the implementation specifications, but with a sufficient level of

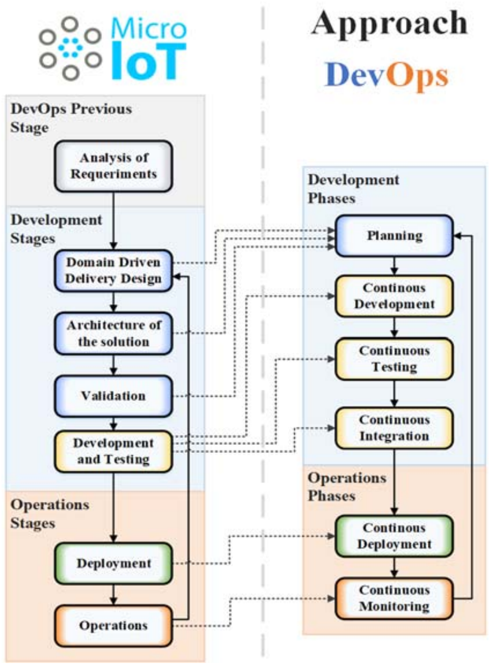

Fig. 1. Stages of the MicroIoT methodology aligned with the corresponding phases proposed by the DevOps organizational approach.

detail to establish the size of the system and define the number of deliveries necessary to obtain the complete system, based on the needs of the client; ii) *prioritization of requirements* to make appropriate decisions regarding the definition of the number of deliveries [10]. The decisions are made by the interested parties, with the help of the requirements analyst, considering: the importance of the requirements, penalties for the non-compliance, necessary resources (e.g., time, cost, personnel), risks according to their probability and impact, and volatility of the requirements due to constant changes; and iii) *definition of deliveries* to establish the software deliveries based on the priorities previously defined.

B. *Domain-Driven Delivery Design*

An effective and efficient design allows the correct application of the microservices architecture to support the level of availability, scalability, and performance of a system. Therefore, this is the most important part of the life cycle of a microservices application. MicroIoT proposes the design of the delivery following the recommendations of [11]. The activities suggested for this stage are: i) *analysis of the domain* to obtain a technical perspective of the system according to the requirements. It considers system specifications, hardware controllers involved, and properties of a specific context; ii) *delimitation of the context* to decouple the system into different contexts where each one is resilient and accomplished with specific software requirements; iii) *definition of entities, aggregates, and services* to identify entities within the delimited context. The domain model will include representations of real-world things (e.g., hardware devices, embedded devices, sensors). Here are some suggestions for identifying them: a) break down the application requirements by nouns; this helps to identify entities (*objects*) and valuables (*attributes*); b) break down the application requirements by verbs and identify services (*functionalities*) that implement a unique use case; and c) identify aggregates to reveal the services that must be implemented in the system (i.e., root entity, whose lifetime defines the lifetime of its leaves [12], a persistence limit [11]); iv) *identification, definition, and verification of microservices and their components* to define the schemas of microservices and their components (front-end, back-end, database, and external IoT systems). This activity finishes with the validation of the microservices to address aspects like low latency, integrity, and independence between them; v) *constitution of cross-functional teams* to divide members performing internal roles into cross-functional teams, according to the number of microservices to be developed. As stated in [13], there must be a global cross-functional team monitoring and verifying the progress of the rest of the teams; and vi) *establishment of data models* to define the database model with which each microservice will work.

C. Architecture of the solution

This stage defines a general architecture of IoT based on microservices to obtain a scalable and maintainable system over time, achieving satisfaction from both the functional and non-functional requirements specific to the solution domain.

D. Validation

The system design is presented to the stakeholders, indicating the microservices that have met the requirements. The goal is to assess whether the delivery design meets the user's expectations before coding it. If the stakeholders do not approve the design, the team will identify, adapt, and correct the flaws found and include any suggested improvements.

E. Development and testing

It involves three software development practices: i) *continuous development* where each work team is responsible for the implementation or modification of a microservice so that it complies with the requested functionalities; ii) *continuous testing* where tests are developed to verify whether microservices work correctly or not; and iii) *continuous integration* where the members of a team integrate their work frequently in an organized and controlled fashion. The MicroIoT methodology recommends the adoption of tools that allow the developers to achieve the integration of their work, automatically, as soon as possible (e.g., Jenkins, Codeship, or Bamboo). This integration requires a) checking source code in a repository, b) executing unitary tests, and c) integrating the new functionalities with the existing ones to look for any problem.

F. Deployment

During this stage, the new functionalities go into the production environment. Several tools such as Docker, Kubernetes, or Rancher are used to manage the containers in the production environment, while Puppet or Chef are used to configure the applications within it.

G. Operations

This stage includes two practices: i) *continuous monitoring* to identify and solve the problems of the Information Technology (IT) infrastructure (i.e., how launched applications perform in production). Tools like Nagios or Zabbix monitor aspects of the system, such as CPU load, RAM allocation, network traffic statistics, memory consumption, and the availability of free disk space [9]; and ii) *receiving feedback from the stakeholders* regarding usability and accessibility to make decisions, implement changes, and improve the applications. In case of pending deliveries, a new iteration of MicroIoT must be performed.

V. Conclusions and Future Work

This study proposes an agile methodology, MicroIoT, for the creation of Internet of Things applications based on microservices. It can be adapted to many domains such as Ambient Assisted Living, among others. The stages and activities of this methodology allow stakeholders to obtain a general architecture for IoT based on microservices; also, some recommendations on the management of microservices infrastructure were presented. The development, testing, deployment, and operations stages must be automated using the practices and technological tools recommended in the DevOps organizational development approach. This array of techniques will allow better version control, helping with the maintenance of the system and integrating the independent components with ease. Furthermore, the proposed methodology could effectively be applied in more complex software architectures covering other aspects such as human-computer interaction or management in the cloud. Thus, as future work, the MicroIoT methodology will be evaluated to address if it adapts adequately to these types of architectures.

VI. References

[1] B. Butzin, F. Golatowski, and D. Timmermann, “Microservices approach for the internet of things”, 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), Berlin, 2016, pp. 1-6.

[2] G. Mazlami, J. Cito and P. Leitner, “Extraction of Microservices from Monolithic Software Architectures”, 2017 IEEE International Conference on Web Services (ICWS), Honolulu, 2017, pp. 524-531.

[3] A. Krylovskiy, M. Jahn and E. Patti, “Designing a Smart City Internet of Things Platform with Microservice Architecture”, 2015 3rd International Conference on Future Internet of Things and Cloud, Rome, 2015, pp. 25-30.

[4] H. Vural, M. Koyuncu, and S. Guney, “A Systematic Literature Review on Microservices”, International Conference on Computational Science and Its Applications (ICCSA), Springer, Cham, 2017, pp. 203–217.

[5] M. Villamizar et al., “Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud”, 2015 10th Computing Colombian Conference (10CCC), Bogota, 2015, pp. 583-590.

[6] K. Tserpes, “stream-msa: A microservices’ methodology for the creation of short, fast-paced, stream processing pipelines”, ICT Express vol. 5, no. 2, 2019, pp. 46–149.

[7] T. Vresk and I. Čavrak, “Architecture of an interoperable IoT platform based on microservices”, 2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, 2016, pp. 1196-1201.

[8] E. F. Cabrera Alvarado and P. J. Cárdenas Cárdenas. "Metodología para la creación de aplicaciones basadas en microservicios para soluciones de internet de las cosas en ambientes de vida asistidos," B.S. Thesis. Cuenca, 2018-10-29. [Online]. Available: http://dspace.ucuenca.edu.ec/handle/123456789/31511

[9] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, “Devops,” IEEE Software, vol. 33, no. 3, pp. 94–100, 2016.

[10] C. Wohlin et al., Engineering and Managing Software Requirements. Springer Science & Business Media, 2005.

[11] M. Wasson, “Using domain analysis to model microservices” Microsoft, 2019.

[12] S. Sharma, Mastering Microservices with Java 9: Build domain-driven microservice-based applications with Spring, Spring Cloud, and Angular. Packt Publishing Ltd, 2017.

[13] A. Balalaie, A. Heydarnoori, and P. Jamshidi, “Microservices architecture enables devops: Migration to a cloud-native architecture”, IEEE Software, vol. 33, no. 3, 2016, pp.42–52.