International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 14, No. 3, November 2025, pp. 745~753
ISSN: 2089-4864, DOI: 10.11591/ijres.v14.i3.pp745-753

Enhancing cross-cutting concerns in the internet of things with applying aspect oriented programming

Khalifa Fatiha, Guelta Bouchiba
Département d'informatique, Faculté des Mathématiques et Informatique, Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf (USTO-MB), Oran, Algeria

Article Info

Article history:

Received Feb 18, 2025
Revised Aug 12, 2025
Accepted Sep 9, 2025

Keywords:

Aspect-oriented programming
Crosscutting concerns
Internet of things
Microservices
Petri net

ABSTRACT

Aspect oriented programming (AOP) is a new programming model that provides new concepts to handle cross-cutting concerns about code. The idea of introducing AOP in the internet of things (IoT) is inherited from the complexity of sensor operations involving data acquisition, processing, and communication, the need to support multiple simultaneous services for users particularly security services such as authentication, authorization, data traceability, and transaction management, and the challenges posed by the IoT deployments, the treatment of these data volumes lead to problematic code redundancy and cross-cutting concerns that compromise system maintainability. In this context, AOP enables the separation of core functionalities, data management, and cross-cutting concerns, allowing them to be developed and reused independently within the same codebase. To address these issues, this paper proposes an AOP model for IoT systems based on the Petri net representations. The model strategically integrates the core AOP advantages of modularity, reusability, and extensibility, microservices based architectural decomposition and specialized handling of sensor-specific requirements in IoT environments.

Corresponding Author:

Khalifa Fatiha
Département d'informatique, Faculté des Mathématiques et Informatique
Université des Sciences et de la Technologie d'Oran Mohamed Boudiaf (USTO-MB)
BP 1505, El M'naouer, Oran, Algeria
Email: fatiha.khalifa@univ-usto.dz

1. INTRODUCTION

The internet of things (IoT) [1] is a contemporary technology that makes it possible to build a connected world based on physical objects that can be used in smart vehicles, water systems, smart home appliances, and other extensions that are integrated with software, electronics, and various sorts of sensors and networks that allow these objects to communicate and share data [2], [3]. Nowadays, IoT is an important and widely used low-area network characterized by low energy consumption, low memory, and the ability to use a large number of sensors.

Development of these different systems in a single IoT application can be done by microservices [4], [5]. Each sensor can offer only one microservices due to its low memory; on the other side, one sensor can deal with several microservices [6]. This will cause redundancy codes and many cross-cutting concerns that affect various IoT device codes. To tackle this problem using aspect oriented programming (AOP) [7], a novel abstraction design technique in IoT technology was required to solve cross-cutting issues, provide entities that are simultaneously reusable, modular, and adaptable, produce code that can be dynamically deployed, and improve the data exchange between various sensors and the caliber of networked devices.

AOP is a methodology that provides separation of crosscutting concerns by introducing a new unit of modularization an aspect. Each aspect focuses on a specific crosscutting functionality [8]. These aspects are designed to be reusable and allow adding at any time a new behavior to a source code without altering or interacting with the other aspects [9]. In addition to the notion of aspect, aspect- oriented programming offers some important keys concepts [10], [11]:

- Joinpoint is a precise definition of a point within an application code where it can be either a function or a procedure where it will be a cross-cutting concern.
- Pointcut is an expression used to be matched with join points. The pointcut use three kinds of expressions (before, after, and around).
- Advice is a part of the application code defining a cross-cutting concern that relies on a joinpoint.
- Aspect is the most important mechanism in AOP; it is a unit allowing modularization of the code related to a final application code.
- Weaving is the process that allows advice to be inserted into the functional application exactly at the join points.

Applying AOP as a programming approach is an important focus area for developers especially in IoT paradigm. However, there is a very few of literature on the use of AOP in the IoT paradigm.

- Maingret *et al.* [12] suggested a dynamic external behaviors issue. They separate and interconnect the context tracking from the control process in an IoT application. The use of AOP was to manage the dynamic external behaviors defined as aspects. In this work, the authors did not discuss the problems of microservices or data transactions.
- Balakrishnan and Sangaiah [13] discussed some points based on the need for the IoT domain in distributed applications. They proposed an aspect oriented framework for IoT context to solving the limitation of the existing protocol in service discovery.
- Velan [14] presented a literature review of service discovery in IoT applications. In this work, the AOP was used to evaluate the reconfiguration service in terms of aspect management, updating the parameters required for the service, and adding and removing components.
- Balakrishnan and Sangaiah [15] proposed an AOP approach to address the issues involved in the analysis and treatment of data in IoT research.
- Bansode [16] employs AOP to enhance system security and optimize performance within IoT environments.
- Khalifa and Guelta [17] proposed an AOP model in an IoT-based Petri net graph that involves the use of an aspect entity to satisfy the requirements of the web services composition.

To highlight the contribution of the use of AOP by microservices in IoT. There are some important points shared by aspect-oriented design in the goals of microservices utilized in IoT technology, namely:

- Lightweight communication: communication between different sensors in IoT technology needs lightweight communication, which is offered by aspect-oriented design.
- Independent deployable units: for microservices uses in IoT technology, it is very important that the services offered by each sensor be independent and deployable. Aspect-oriented programming offers "units" named aspects that are deployable independently.
- Centralized management: both IoT technology and aspect-oriented programming are based on central management.
- Independent development technologies.

In this paper we present a first attempt to applying AOP in IoT technology for consider crosscutting concerns in IoT technology based Petri net model presenting in detail the hardware and software implementation, thus our contribution focuses on separating all crosscutting concerns from de microservices used by the IoT application and the solution found to integrate them in Aspect code. The method and proposed algorithm, discussion with a conclusion are proposed.

2. PROPOSED ALGORITHM

In this section, we introduce an algorithm that applies the AOP concept to separate crosscutting concerns occurring in microservices used for IoT systems, using Petri net-based formal semantics. Algorithm 1 illustrates the proposed method. Given as input three set L (representing sensors), A (representing crosscutting concerns implementing as an aspect code) and μS (representing the decomposed services requested by users).

The algorithm is based on generating the data collected by the sensors:

$$Function\ Get - data(ID_i, OD_i) \quad (1)$$

First the data is collected and recorded based on the type and requirements of the sensors, we have: $ID_i \subset OD_i \subset D_i$. This function allow data to be stored in a database according to the sensor's ID, type, and specific requirements:

$$Function\ AddServices(\mu S'_i, S_i) \quad (2)$$

This function enables to create services as needed. For example, if a temperature control service is required, this service will be automatically added:

$$Function\ AspectADD(Weaved, \mu S_i, A) \quad (3)$$

After data collection and service processing the algorithm finalizes the transition points in the Petri net graph (lines 15–20).

$$Graphend = graphend.proceed \quad (4)$$

Algorithm 1. AOP for IoT based Petri net graph

```
Input: Graphstart (Petri graph net start),
 μS (Set of available microservices) → μS = {μS1, μS2 , μS3,.., μSn},
A (Set of Aspects) → A = {A1, A2, A3,.., An},
L (Set of Sensors) → L={S1, S2, S3, ……, Sn}
n (maximum number of Sensor)
 A Sensor Sa = (Da, Ta, Aa, Wa, IDa, ODa , μSa), Data Da = {D1, D2, D3,.., Dn}
Output: Graphend (completed or failure)
 1:     Graphend=empty;
 2:           IDa ∈ Da
 3:        m = ∑Services ∈ μS //A microservice can intervene in one or more sensors.
 4:         μS0=null
 5:        for i=1 to n do L={S1, S2, S3, ……, Sn}
 6:             for each sensor Si ∈ L do:
 7:                     function Get-data (IDi, ODi)
 8:                      function AddService ( μSi, Si)
 9:                      for each A ∈ R do
10:                           if (μSi.CCC) = (A.CCC) then // CCC : Crosscutting Concerns
11:                             function AspectADD(Weaved, μSi, A)
12:                          end if
13:                       end for
14:              end for
15:       OutputParameters=OutputParameters ∪ Si.Po
16:       AddService ( μSi, Si)
17:       Graphend= Graphend.proceed
18:      end for
19:      Graphend= Graphend.Completed
20:      return failure
```

2.1. Layer diagram

Figure 1 shows our proposed AOP layer-based IoT architecture, where aspects for microservices that are readily reusable in IoT applications can be generated, the device layer, communication layer, data acquisition layer, aspect-oriented abstract layer, and application layer are the five layers that make up this layer diagram. The device layer consists of sensors and actuators that interact with the IoT area network to collect data e.g., temperature and humidity, which will be sent to the data acquisition layer through the communication layer via the transmission control protocol (TCP) and user datagram protocol (UDP) transfer protocols. The data acquisition and processing layer consists of a microcontroller that allows read data from the sensors, controls the actuators, and creates microservices [18].

The device layer consists of physical devices belonging to the IoT environment, which collect and they can also change the state of the environment. The collected data will be transmitted to the data acquisition system through the communication layer using the two transfer protocols, TCP and/or UDP [19].

In this layer diagram, our proposed system utilizes a weaving process using the AspectJ of the AOP programming language, which is an underlay between the aspect layer and microservices layer. This underlayer allows you to create and add aspects in the aspect layer to separate cross-cutting concerns that occur in different microservices of the IoT application through the weaving process. Finally, the information will be sent to the client by the application layer in the form of web services via http, rest, coap, and message queue telemetry transport (MQTT) protocols [20].

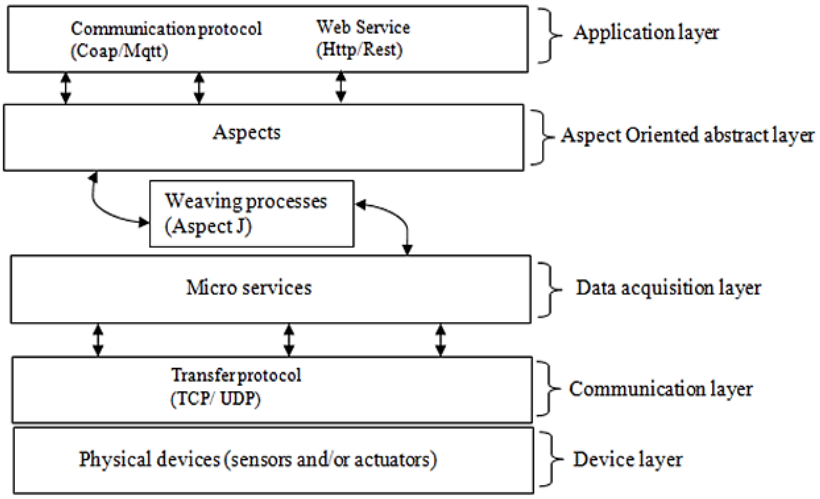

Figure 1. Layer diagram of AOP proposed model in IoT

3. METHOD

In this section, we explain how IoT technology uses the AOP paradigm to consider cross-cutting concerns in sensor microservices by employing a Petri net graph as the conceptual model. We give some definitions to help illustrate the concepts that are necessary background information.

3.1. Aspect oriented model for internet of things

Introducing aspect-oriented programming in IoT technology for interpreting cross-cutting concerns in sensor microservices using the Petri net model focuses on five points: as shown in Figure 2.

- Data entry: IoT sensors collect data from their cover area. The collection of data depends on some factors, such as the range of area needs and the user request. For example, in the application of IoT technology in agriculture, we need the use of data on temperature, humidity, and watering.
- Data analysis: to utilize the data collected in the IoT system, the data must be within the acceptable range. The user request sends requests to the sensor network to detect temperature values within a predefined interval (TempMIN and TempMAX). Each device can issue several requests with different parameters. An example of a query sent by a monitoring device: find all sensors with temperature values in the range {10–12}.
- Aspects creation: this step permits aspects to be added. Aspects are the most important entities in aspect-oriented programming because they permit the separation and reuse of the cross-cutting concerns existing in the microservices of the sensor code.
- Data weaving with aspect: aspects will be weaved dynamically with the final code before the transmission of data to the sink.
- Data transmission: the collected and analyzed data must undergo the processing required by the microservices, and finally, it will be transmitted to the web service.

Definition 1: a sensor (S) is defined by a 7 tuple.

- $S = (D, TP, A, DA, ID, OD, \mu S)$, where,
- $D = \{ D_1, D_2, D_3, \dots, D_n \}$ represents the data collected,
- $TP = \{ TP_1, TP_2, TP_3, \dots, TP_n \}$ represents the protocol used to transmit data to the sink,
- $A = \{ A_1, A_2, A_3, \dots, A_n \}$ represents aspects that consider crosscutting concerns in microservices used by the sensors,
- $DA \subseteq (D \times TP) \cup (TP \times D)$ directed arcs,
- ID: input data,
- OD: output data after analysis and treatment,
- μS: sensor microservices.

Definition 2: weaving processes (WP).

Each aspect $A_i \in A$ is represented by the tuple $A_i =< CCC, Jpoint, Pcut, Adv >$,

- CCC: represents the crosscutting concerns that occur in microservices,
- Adv: is a functionality designed to encapsulates CCC,
- Jpoint: specific points in the microservices code that correspond to the aspect's pointcuts,

– Pcut: is a function that connects a joinpoint to the advice.

A weaving process establishes an explicit connection between each μS and these cross-cutting concerns functionality CCC.

$$WP=ID_i \rightarrow \mu S_i \, \Delta \, \{A_1, A_2, A_3.., A_n\} \rightarrow OD_i$$

– The WP have as input: ID_i (input data of sensor number "i"),
– and as output parameter: ID_o(output data of sensor number "i").

The WP permits to weave all aspects $\{A_1,A_2,A_3..,A_n\}$ with each μS offered by sensors. The weaving process can be executed in three types:

– A WP type «before»: in this type the advice is executed before the the microservice μS_i.
– An WP type «after»: the advice is executed after the execution of the microservice μS_i.
– An WP type «around»: the advice exeuted around execution of the microservice μS_a.

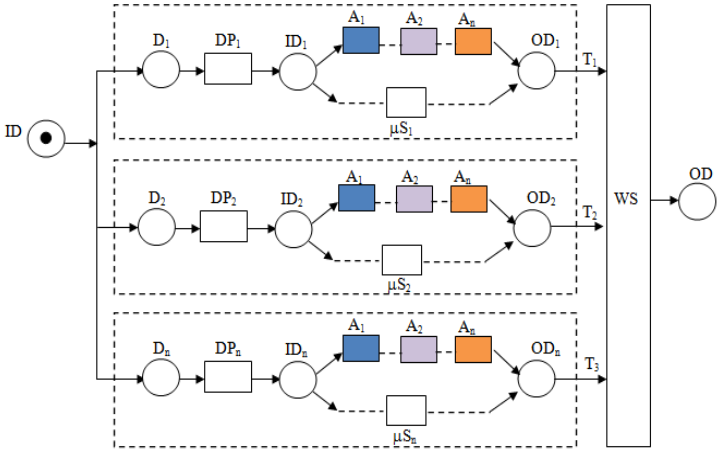

Figure 2. AOP architectural IoT Petri net graph

In Figure 3, we illustrate a diagram of our AOP model proposed for IoT technology, this functional diagram shows how our model uses AOP to manage core functionality. Data is initially collected by IoT sensors according to the parameters required (e.g., temperature readings). The data is then filtered and analyzed by the microservices layer according to contextual demands or threshold levels. Aspects hold crosscutting concerns occured in the application, and they are implemented and deployed separately to promote modularity and code reusability. Weaving process, provided by the AOP, integrates those concerns into the primary application by introducing their behavior at specific points in the execution (jointpoint, pointcut and advice), based on the information logged and on the setup of the final application.

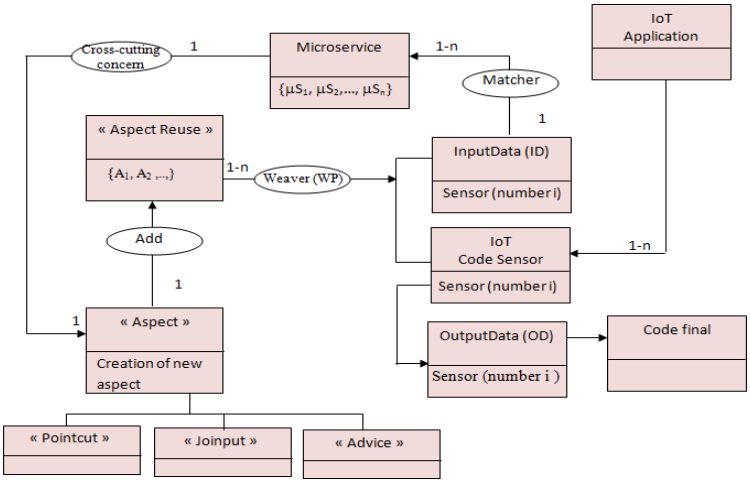

Figure 3. Functionality diagram of a basic AOP model for IoT

4. RESULTS AND DISCUSSION

4.1. Experimental analysis

Our AOP model IoT is implemented and developed by the Java language using the software Eclipse Kura, inspired from [21], which utilizes a gateway for the IoT, the AspectJ [22] language for aspect-oriented programming, and by using the Eclipse Paho MQTT library, the software Kura provides a means of communication for those applications to take the data gathered from the sensors to the gateway by the MQTT brokers and web services [23]-[25].

4.2. Illustration and discussion

An example will be given in this section to better illustrate the proposed AOP model for the IoT-based Petri net graph. Consider, for example, a smart agriculture area equipped with data collection for temperature control, gas control, smart water service, smart weather service, and energy management service. Consider the following services that can be used in this agriculture area:

a. Temperature monitoring service: manages and processes the temperature monitoring (TM) data.
b. Weather service: manages and processes the wind speed (WS) data.
c. Watering service: manages and processes the soil moisture (SM) data.
d. Lighting service: manages and processes the natural light measured (NLM) data.
e. Energy management service: manages and processes the electrical voltage measured (EVM) data.

Let consider five types of sensors: S_1, S_2, S_3, S_4, and S_5 and let remember that sensor is defined by: $S\,(D, TP, DA, , ID, OD, \mu S)$. All information is given in Table 1.

Table 1. Sensors information

Sensor type	Data (D)	ID	μS	Aspect cross-cutting concerns
S_1	TM	TM∈{0-100}/°	Data centered on temperature	If Tem <10° or Tem >40° then call A1
S_2	WS	WS∈{0,.15}/ms	Data centered on wind speed	If weather state=WS call A2
S_3	SM	SM∈{0–100%}	Data centered on soil measure	If SM<30° call A3 (alert threshold)
S_4	NLM	NLM∈{1, 10, 30, 100} mille lux	Lighting servcie	If NLM<10 call A4 for supplemental lighting
S_5	EVM	EVM∈{0-250 v}	Energy management servcie	If EVM> 250 call service A5

Understanding the contribution of aspect-oriented programming to IoT applications necessitates the careful definition of aspect entities and their role in separating crosscutting concerns within the microservices. The aspects are programmed with the AspectJ. These aspects have been utilized to the information in Table 1. Although both TCP and UDP transmission protocols can be used in IoT systems, we have used only the TCP protocol in this work. The MQTT protocol was employed to enable communication between the sensors and the end user.

In order to understand the contribution of applying aspects for separate crosscutting concerns in the IoT application, Figure 4 displays a bar graph that provides a statistical representation of the use of aspects in an IoT application. To sum up, applying aspect-oriented programming in IoT applications with the AsecptJ language has shown a considerable increase in reusability and modularity, which are key factors contributing to a higher level of programming quality.

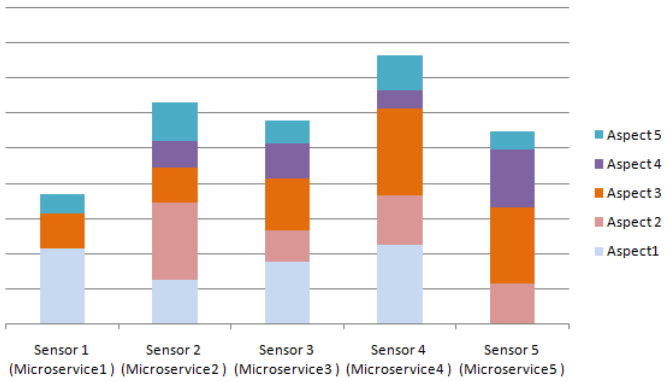

Figure 4. Code reusability in AspectJ

4.3. Relative performance analysis of programming methods in internet of things

IoT applications often rely on a large number of sensors operating under limited resources particularly memory and energy. Among these constraints, memory and energy consumption are especially critical. when these sensors are deployed using object-oriented programming (OOP), the OOP handle the crosscutting concerns by duplicating the Business logic across multiple modules. This leads to lager codebase, increased CPU usage and consequently higher memory consumption and energy consumption, values per percentage is shown in Figure 5 and Table 2.

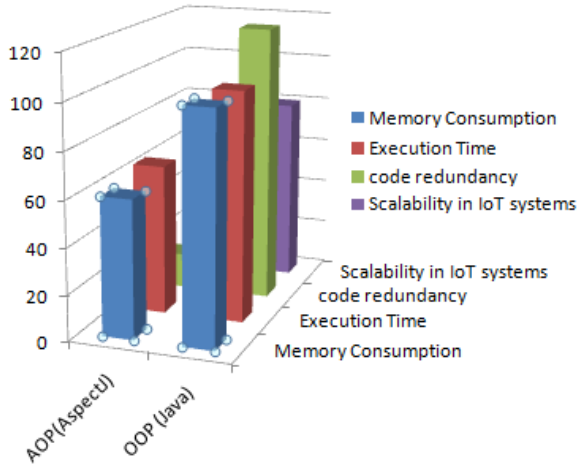

Figure 5. The benchmark methods comparison

Table 2. Comparison of memory consumption

Methods	OOP (%)	AOP (%)
Memory consumption	100	60
Execution efficiency	100	80
Code redundancy	120	15.3
Scalability in IoT systems	60	80

Using the current AOP programming method, the memory consumption is lower due to its modular aspect design up to 60% compared by OOP which is higher because of the duplicated logic across final code approaching to 100% use of memory. The execution efficiency in a simulation of an IoT application, lasting 10 minutes and using 500 sensors with a reading every 5 seconds, was evaluated using Java for OOP and AspectJ for AOP. The results showed that execution time was significantly higher in the OOP model due to redundant logic. In contrast, the increase in reusability and modularity in the AOP approach enabled a reduction in execution time, decreasing from 100% to 80%.

5. CONCLUSION

Aspect AOP enables better separation of crosscutting concerns from the main application code by encapsulating them into modular aspects. In this research, we demonstrate that applying AOP in IoT applications compared to OOP can lead to a reduction in memory and energy consumption by up to 50%, while also improving code reusability by eliminating redundant code in the final application. Furthermore, execution time was improved by up to 65%. In future work, we propose to explore the dynamic deployment of aspect-oriented components to further enhance adaptability and efficiency.

FUNDING INFORMATION

The authors confirm that no funding supported this research.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Khalifa Fatiha	✓	✓	✓	✓	✓	✓			✓	✓			✓	
Guelta Bouchiba		✓			✓	✓			✓	✓	✓	✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

REFERENCES

[1] P. Goyal, A. K. Sahoo, and T. K. Sharma, "Internet of things: Architecture and enabling technologies," *Materials Today: Proceedings*, vol. 34, pp. 719–735, 2019, doi: 10.1016/j.matpr.2020.04.678.

[2] Y. Qian, D. Wu, W. Bao, and P. Lorenz, "The Internet of Things for Smart Cities: Technologies and Applications," *IEEE Network*, vol. 33, no. 2, pp. 4–5, Mar. 2019, doi: 10.1109/MNET.2019.8675165.

[3] C. V. Mahamuni, "Exploring IoT-Applications: A Survey of Recent Progress, Challenges, and Impact of AI, Blockchain, and Disruptive Technologies," in *7th International Conference on Electronics, Communication and Aerospace Technology, ICECA 2023 - Proceedings*, Nov. 2023, pp. 1324–1331, doi: 10.1109/ICECA58529.2023.10395064.

[4] T. Cerny, M. J. Donahoo, and M. Trnka, "Contextual Understanding of Microservice Architecture: Current and Future Directions," *ACM SIGAPP Applied Computing Review*, vol. 17, no. 4, pp. 29–45, 2017, doi: 10.1145/3183628.31836.

[5] T. Cerny, "Aspect-oriented challenges in system integration with microservices, SOA and IoT," *Enterprise Information Systems*, vol. 13, no. 4, pp. 467–489, Apr. 2019, doi: 10.1080/17517575.2018.1462406.

[6] B. Butzin, F. Golatowski, and D. Timmermann, "Microservices approach for the internet of things," in *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, Sep. 2016, pp. 1–6, doi: 10.1109/ETFA.2016.7733707.

[7] Z. Chen, Y. Zhu, and Z. Wang, "Design and Implementation of an Aspect-Oriented C Programming Language," *Proceedings of the ACM on Programming Languages*, vol. 8, no. OOPSLA1, pp. 642–669, Apr. 2024, doi: 10.1145/3649834.

[8] A. Kumar, A. Kumar, and M. Iyyappan, "Applying Separation of Concern for Developing Softwares Using Aspect Oriented Programming Concepts," *Procedia Computer Science*, vol. 85, pp. 906–914, 2016, doi: 10.1016/j.procs.2016.05.281.

[9] O. A. Abdulhameed, A. Y. Yousuf, and R. H. Abbas, "Aspect oriented programming: Concepts, characteristics and implementation," *Periodicals of Engineering and Natural Sciences*, vol. 7, no. 4, pp. 2022–2033, 2019, doi: 10.21533/pen.v7i4.975.

[10] S. R. Raheman, H. B. Maringanti, and A. K. Rath, "Aspect oriented programs: Issues and perspective," *Journal of Electrical Systems and Information Technology*, vol. 5, no. 3, pp. 562–575, Dec. 2018, doi: 10.1016/j.jesit.2017.06.003.

[11] A. A. Magableh, H. B. Ata, A. A. Saifan, and A. Rawashdeh, "Towards improving aspect-oriented software reusability estimation," *Scientific Reports*, vol. 14, no. 1, pp. 1-20, Jun. 2024, doi: 10.1038/s41598-024-62995-z.

[12] B. Maingret, F. Le Mouël, J. Ponge, N. Stouls, J. Cao, and Y. Loiseau, "Towards a decoupled context-oriented programming language for the internet of things," in *International Workshop on Context-Oriented Programming, COP 2015 - co-located with the 29th European Conference on Object-Oriented Programming, ECOOP 2015*, pp. 1–6, Jul. 2015, doi: 10.1145/2786545.2786552.

[13] S. M. Balakrishnan and A. K. Sangaiah, "Aspect oriented middleware for internet of things: A state-of-the art survey of service discovery approaches," *International Journal of Intelligent Engineering and Systems*, vol. 8, no. 4, pp. 16–28, Dec. 2015, doi: 10.22266/ijies2015.1231.03.

[14] S. S. Velan, "Introducing aspect-oriented programming in improving the modularity of middleware for internet of things," *Advances in Science and Engineering Technology International Conferences, ASET 2020*, 2020, doi: 10.1109/ASET48392.2020.9118238.

[15] S. M. Balakrishnan and A. K. Sangaiah, "Aspect oriented modeling of missing data imputation for internet of things (IoT) based healthcare infrastructure," in *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications, Elsevier*, pp. 135–145, 2018, doi: 10.1016/B978-0-12-813314-9.00006-2.

[16] R. S. Bansode, "Enhancing IoT Security and Performance Using Aspect- Oriented Programming in Java Applications," *International Journal of Scientific Research and Engineering Trends*, vol. 3, no. 1, pp. 25–29, 2017, doi: 10.61137/ijsret.vol.3.issue1.143.

[17] F. Khalifa and B. Guelta, "Aspect Oriented Web Service Composition Based Petri Net Model," in *Lecture Notes in Networks and Systems*, vol. 591, pp. 148–159, 2023, doi: 10.1007/978-3-031-21216-1_16.

[18] M. A. Jarwar, S. Ali, M. G. Kibria, S. Kumar, and I. Chong, "Exploiting interoperable microservices in web objects enabled Internet of Things," in *International Conference on Ubiquitous and Future Networks, ICUFN*, Jul. 2017, pp. 49–54, doi: 10.1109/ICUFN.2017.7993746.
[19] R. Mehta, J. Sahni, and K. Khanna, "Internet of Things: Vision, Applications and Challenges," *Procedia Computer Science*, vol. 132, pp. 1263–1269, 2018, doi: 10.1016/j.procs.2018.05.042.
[20] D. Bilal, A.-U. Rehman, and R. Ali, "Internet of Things (IoT) Protocols: A Brief Exploration of MQTT and CoAP," *International Journal of Computer Applications*, vol. 179, no. 27, pp. 9–14, Mar. 2018, doi: 10.5120/ijca2018916438.
[21] F. Khalifa and S. Chouraqui, "Applying aspect oriented programming in distributed application engineering," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 7, pp. 226–232, 2020, doi: 10.14569/IJACSA.2020.0110729.
[22] A. Przybyłek, "An empirical study on the impact of AspectJ on software evolvability," *Empirical Software Engineering*, vol. 23, no. 4, pp. 2018–2050, Aug. 2018, doi: 10.1007/s10664-017-9580-7.
[23] Rupali Atul Mahajan, "Enhancing MQTT Security in the Internet of Things with an Enhanced Symmetric Algorithm," *Journal of Electrical Systems*, vol. 20, no. 1s, pp. 126–137, Mar. 2024, doi: 10.52783/jes.758.
[24] M. Trnka, J. Svacina, T. Cerny, and E. Song, "Aspect oriented context-aware and event-driven data processing for internet of things," in *Proceedings of the 2018 Research in Adaptive and Convergent Systems, RACS 2018*, Oct. 2018, pp. 319–323, doi: 10.1145/3264746.3264761.
[25] M. Michaelides, C. Sengul, and P. Patras, "An Experimental Evaluation of MQTT Authentication and Authorization in IoT," in *WiNTECH 2021 - Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental Evaluation and CHaracterization, Part of ACM MOBICOM 2021*, Jan. 2022, pp. 69–76, doi: 10.1145/3477086.3480838.

BIOGRAPHIES OF AUTHORS

Khalifa Fatiha Lecturer at the Department of Computer Science at the University of Sciences and Technology of Oran Mohamed Boudiaf (USTO-MB), Algeria. She received his Ph.D. in Computer Science at the same university in 2020. Her research interests include issues related to computer engineering, network communication, network security, web semantic; aspect oriented programming language, internet of things, and pattern extraction. She is author of a set of research studies published at National and International Conferences, and International Journals. She can be contacted at email: fatiha.khalifa@univ-usto.dz.

Guelta Bouchiba currently working as an assistant professor in the Faculty of Mathematics of Computer Science at the University of Sciences and the Technology of Oran (USTO-MB). He has received his Licence diploma in computer sciences in 2011, then his master degree in system information and network 2013 and he obtained a Ph.D. degree from the same university. The Thesis was about multimodal recognition. He presented new versions of multimodal recognition based on ECG and GAIT in several journal papers and one book. He can be contacted at email: bouchiba62@univ-usto.dz.