Australian Journal of Science and Technology

ISSN Number (2208-6404)
Volume 7; Issue 4; December 2023

Original Article

Remodeling the oneM2M common services layer with core logic services for enhanced IoT reuse

Bassey Asuquo Ekanem[1]*, Kehinde K. Agbele[2]

[1]Department of Computer Science, Delta State University of Science and Technology, Ozoro, Delta, Nigeria, [2]Department of Mathematics and Computer Science, Elizade University, Ondo, Nigeria

ABSTRACT

Reuse of core logic functions across domains in internet of things (IoT) app development is a challenging task due to inadequate models and architectures to support this process. Due to this, reuse in IoT systems is limited to infrastructure services such as device management, connectivity, message communication, and others while core logic components are developed at the application layer in a highly domain specific approach. This research undertook a study of core logic functions in source codes of selected IoT apps downloaded from GitHub. The study revealed four core logic functions common to IoT apps across domains that can be repackaged as microservices and incorporated into the common service layer where they can be reused across domains in IoT solutions. With this approach, apps development time and efforts will be reduced as developers' productivity and products quality are enhanced.

Keywords: Common service layer, IoT common core logic services, oneM2M Architecture, software reuse

Submitted: 25-09-2023, **Accepted:** 06-11-2023, **Published:** 30-12-2023

INTRODUCTION

In recent times, the world has witnessed a great digital transformation following the introduction of internet of things (IoT), a technology that is swiftly penetrating every sphere of human endeavors with great benefits. IoT is a network of devices called "things" – that are connected through the internet, enabling the collection, exchange, and analysis of generated information.[1] This concept was first introduced by Peter T. Lewis in Washington DC in September, 1985 as a technology for the future.[1-3] Today, it's a global masterpiece adopted by businesses and individuals to improve operational efficiency, grow revenue, and enhance the quality of life.[4]

IoT is widely applied in our daily life. The next generation of mobile connection technology called 5G is expected to boast the application of IoT in our daily living through massive connected devices. Research findings reported in Statista[5] reveals that, by 2030, 29.42 billion IoT devices will be connected globally across domains from automotive to smart-homes, Fintech to wearables, and many others. The report further affirms that, the massive deployment of IoT devices globally is expected to generate huge revenue of about $1 trillion by 2030 to the IoT industry which will, in turn, encourage innovative use of IoT solutions. Furthermore, it is predicted that the lion share of this industry revenue will be from innovative IoT software solutions developed and deployed for use by businesses and individuals across domains. This trend is expected to ignite a growing pressure on software developers to build and deploy massive innovative IoT software solutions to meet users' demands.[5]

Considering the complexity of IoT software solutions, enormous tasks and costs involved in the development process, IoT software development projects could drag longer than expected thereby delaying the deployment of the needed solution. In this case, software reuse becomes a viable option to fast-track the process.[2,3,6] Software reuse is the process of reusing software components from existing software projects in developing new solutions, instead of developing entire software from scratch. This approach is advantageous in terms of increased productivity, shorter time to market, minimized risks, and enhanced software quality with integration capabilities.[3,4,7,8] Reusable software components

Address for correspondence: Bassey Asuquo Ekanem, Department of Computer Science, Delta State University of Science and Technology, Ozoro, Delta, Nigeria. E-mail:basseyekanem99@gmail.com

are interchangeable software parts with interfaces that make it easier to use them in other software projects and they are usually presented in the following forms, namely, APIs, data access objects, plugins, classes, or models.[9-11]

Being that IoT software engineering is a relatively new area, application of software reuse in this area is highly challenged[12] as most IoT solutions are often bespoke and highly domain specific.[13] Moreover, inadequate models, frameworks, and architecture needed to support loose coupling of core logics of IoT apps into microservices for reuse are a more serious challenge.[7,14-17] To this end, reuse in IoT systems is limited to infrastructure services such as device management, connectivity, and data messaging as provided by most widely used architectures like onemachine-to-machine (M2M) while core logics reuse across domains remains highly challenged.

In view of the above, this research was designed to study reuse in IoT software engineering with the view to proffering solutions to inherent challenges with respect to core logic reuse. The remainder of this article is organized into nine sections as follows. Section 2 presents a review of related works while Section 3 is for overview of oneM2M model. Section 4 presents findings from the review. Materials and methods are discussed in Section 5 while Section 6 presents a review of the downloaded source codes. In Section 7, results and discussions are covered while Section 8 presents implementation of the remodeled common service layer. Conclusion and recommendations are presented in Sections 9 and 10, respectively.

RELATED WORKS

There are many research efforts in IoT software reuse documented in the literature. However, for the purpose of this research, emphasis was placed on the review of research efforts relating to decoupling of IoT Apps into common services for reuse and the reuse of IoT core logics in software engineering projects.

In Nastic *et al*.,[18] an IoT programming model called PatRICIA is presented as a guide on how to decouple physical devices from user applications. PatRICIA uses Intent and IntentScope Abstractions. While intent represents desired tasks to be performed in a physical environment, IntentScope represents logical groups of physical devices. The open source OM2M project named autonomic ETSI-compliant M2M service platform is presented in Alayaa *et al*.[19] The project uses autonomic computing paradigm and semantic models to addresses the M2M complexity issues that hampers reuse in IoT software engineering. With these models, dynamic discovery and reconfiguration mechanisms for IoT systems are provided.

The report presented in Swetina *et al*.[20] states that, in horizontal IoT standards, oneM2M is the most popular; its goal is to define independent accessibility interfaces for M2M services that can be reused across domains founded on architectural outlay comprising three layers, namely, applications, services, and networks layers. In Bonino *et al*.,[21] microservice-based IoT platform for building smart cities is proposed with a distributed functionality of large systems organized into different reusable microservices performing the tasks of data collection, data aggregation, and data analytics on live streaming data. Furthermore, in Newman,[22] composability is emphasized as apps attribute that facilitates fine-grained arrangement of microservices using RDF and JSON payload in a manner that increases reusability in various application domains. Al-Fuqaha *et al*.[23] presents a review of IoT enabling technologies, protocols, and applications as well as key IoT challenges faced in IoT deployment of which decoupling and reuse of microservices are revealed as the key IoT deployment challenges.

In Alaya *et al*.,[24] oneM2M based ontology is presented as a solution for semantic data interoperability at the services level in heterogeneous IoT environment. IoT Reuse across domains is demonstrated in[25] with lysis platform using the iCore vertical architecture and social relationships defined among virtual objects that support horizontal IoT. In Spalazzese *et al*.,[26] a review of solutions for technical interoperability is made which reveals the use of global sensor networks Framework, SENSEWEB Platform, and FOSSTRAK tools in most IoT projects.

The need for semantic interoperability as a tool for aligning and mapping objects among different application domains in Web of Objects (WoO) based IoT environment through microservices and semantic technologies is emphasized in OEP.[27] Semantic interoperability in this context means enabling different agents, services, and applications to exchange information, data, and knowledge in a meaningful way, on and off the Web. WoO architecture for IoT Provisioning is presented in Jarwar *et al*.[14] to support reuse of objects with microservices and maintain their relationships with the reused objects. It also presents an algorithm for microservices and related objects discovery for reuse purposes. Using Xively platform, a technique for sharing firmware of common devices is presented in LogMeIn,[28] although the work does not cover reusability of service and data at service level. In SimfonyBlog,[29] oneM2M's horizontal architecture framework is presented as a global framework for building IoT devices, gateways, and platforms. Furthermore, it provides for common services layer that can be used by developers to deploy IoT solutions and data-exchange systems.

In Martín *et al*.,[30] the categorization of IoT platforms into vertical layered platforms and horizontal layered platform is provided with pros and cons of each category enumerated.

Vertically layered platforms are those that address problems in single domain or industry while horizontally layered platforms address problems across domains. The work in Ahmad *et al*.[31] presents appdaptivity as a system that enables the development of portable decoupled applications with adaption to changing contexts. Using appdaptivity, application developers can intuitively create portable and personalized applications, disengaging from the underlying physical infrastructure.

In the work reported in Alwis *et al*.,[32] a cloud-centric IoT application store that hosts virtual objects of different IoT domains was designed and implemented with a provision for technology tinkerers to consume the virtual objects and integrate them in new IoT applications. The said application store presents decoupled apps with exposed virtual objects of different IoT domains that can be reused in similar use cases with little or no modifications. Arumugam *et al*.[13] identified a set of decoupled, flexible independent reusable AI-centric components across IoT domains that can be adapted and reused in IoT apps development instead of building full stack static IoT solutions. These components include smart contracts, AI planner, condition monitor, and analytics components. This approach is advantageous as its capable of shortening the time to market, reducing development costs of IIoT solutions, and enhancing reusability.

Reuse potentials of IoT application frameworks were examined in Smiari *et al*.[6] Results of the study revealed that the most reused functionalities are those related to device management layer implemented as black-box reuse. In Wehlitz *et al*.,[7] a device abstraction model is presented for defining business processes across heterogeneous devices without the need for dealing with their technical implementations. It also proposes a system architecture for modeling, deployment, execution, and reuse of IoT-aware business processes that are not bound to specific device types.

In oneM2M,[33] software re-modularization technique for discovering fine-grained microservices from enterprise system that can be reused to create IIoT solutions is presented. Findings reported in Elloumi[34] reiterates the benefits of the common service layer of the oneM2M architecture in IoT reuse. However, it further maintains that when the object data are exchanged and reused among various application domains in WoO enabled IoT environment, such data exchange should be in a well-defined formats such as JSON, XML, and RDF to derive the full benefits of the architecture.

OVERVIEW OF ONEM2M MODEL

The entity, oneM2M was launched in July 2012 as a global organization by eight World's leading information and communications technology (ICT) standards development organizations (SDOs) with the mandate to ensure the most efficient deployment of M2M communications systems. Consequent on this, oneM2M Model was developed to support effective use of the emerging technology.[34] The eight ICT SDOs that founded oneM2M are as follows: ARIB (Japan), ATIS (United States), CCSA (China), ETSI (Europe), TIA (USA), TSDSI (India), TTA (Korea), and TTC (Japan).

The oneM2M model takes the form of a middleware service layer consisting of a suite of common service functions (CSFs).[34,35] The middleware service layer lies between applications layer and connectivity layer, as shown in Figure 1. The oneM2M CSFs are exposed to applications and IoT devices through RESTful APIs.

These CSFs are represented as common services which reside within a common service entity (CSE). Services are provided to the application entities (AEs) by CSF through the Mca Reference point and to other CSEs through the Mcc Reference Point.[35,36]. The oneM2M's common service layer is implemented as a software layer with IP transport in the connectivity layer. However, non-IP transports are also supported in the model through interworking proxies. The oneM2M service layer provides 14 common functions that can be reused in IoT application development and deployment projects.[34-36]

The 14 functions as shown in Figure 1 include device management, registration, security, discovery, group management, communication management, data management and registry, subscription, and notification. Others include application and service management, network service exposure, location, service charging and accounting, semantics, and transport management. Because oneM2M follows a modular standardization roadmap, it allows for future IoT requirements and new common service functions to be added.[34]

FINDINGS FROM THE REVIEW

The review reveals the importance of reuse in IoT software engineering and great successes recorded so far especially with the introduction of WoO architecture for IoT provisioning;[14] device abstraction model for defining business processes across heterogeneous devices;[7] software re-modularization technique for discovering fine-grained microservices for reuse in creating IIoT Solution; and the oneM2M Model with 14 common services that can be reused in IoT solution projects.[34]

However, it is worth mentioning that these models, architectures, and techniques only address basic infrastructure services across domains with no provision for reuse of core logic services and interoperability across domains.[34-36] This encourages vertical approach to IoT core logic development, in which case, services are developed and updated separately in a recurring manner for different domains, a situation that increases the overall cost of

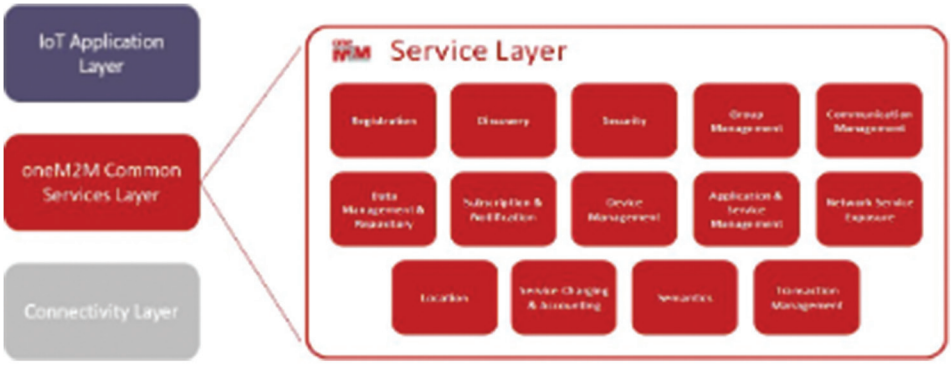

Figure 1: The oneM2M common service layer

application development, deployment, and maintenance.[6,13,34] In view of the above, decoupling of IoT core logics into microservices and reusing them across domains remain a challenging task in IoT Software engineering that calls for solutions. Therefore, architectures, models, and techniques designed specifically to address these challenge by supporting reuse of core logics across domains will advance IoT software engineering.[15,23,33,37]

In realizing this challenge, oneM2m model which is one of the most widely used models provides for modular standardization roadmap that allows for addition of future IoT requirements and new common service functions.[34] Relying on this provision, this research was designed to identify new common service functions, in this case core logics functions that could be added to the oneM2M model to further advance the course of IoT reuse.

MATERIALS AND METHODS

The research work was designed as a case study research to study software reuse in IoT software engineering. The work comprises multiple-case study from different IoT use cases, reviewed in the context of their structures, core logics functions, and reuse. The cases are Gits for IoT projects from GitHub in different use cases, namely, smart home security, Smart road traffic management, Smart waste Management, and Smart health-care system.

Source code of IoT applications in the aforementioned use cases was downloaded and analyzed accordingly with emphasis on their structures, core logics functions, and reuse.

Research Methodology

The research was conducted in the following stages:

i. Download the gits of IoT Apps in the selected use cases from GitHub
ii. Review the codes to identify classes, modules/code blocks that implement the core logics of the application
iii. Study the identified classes, modules/code blocks to understand the core function(s) they perform and reuse of such core functions in the blocks
iv. Compare the core logic functions identified in the reviewed gits for IoT solutions from different use cases to spot common functions if anyName the identified common functions accordingly based on the type of functions they perform.
v. Recommend how such common functions can be packaged as microservices for reuse in IoT projects.

Dataset

The data for this research are gits of IoT Apps in the selected use cases downloaded from GitHub. A total of 33 IoT apps spread across different use cases were reviewed. A summary of the reviewed IoT apps categorized according to use cases and number of apps reviewed in each case follow thus: IoT-based home automation systems (8), IoT-based air pollution monitoring systems (5), IoT-based road traffic management system (5), IoT-based waste management system (5), IoT-based Health monitoring system (5), and IoT-based weather forecast system (5).

Table 1 presents a brief description of each of the IoT Apps which source code were downloaded and used for the research work with respect to languages used in developing the app, lines-of-code (LOC), number of forks, number of commits, and last updated date/latest release of the apps.

REVIEW OF DOWNLOADED SOURCE CODES

A review of the downloaded source codes of the selected Apps was undertaken. In this case, classes in each App were identified and reviewed to understand the type of functionality they provided in the App. To undertake a comprehensive review, reused classes were also considered through a search for "import" directives in the source code with their paths identified as pointers to the source of the reuse components and reviewed accordingly.

Table 1: Description of IoT apps selected for review

S. no	IoT application	Description	Languages used	Lines-of-code	# Forks	# Commits	last updated date /latest release
			IoT-based home automation systems				
1	Wallpanel-Android (Jschollenberger/ TheTimeWalker	Android app for home automation.	Kotlin 85.4% Java 12.4% Other 2.2%	9K	22	2	Updated: April 10, 2023 Releases: 7 Latest: V0.10.5 Released on January 15, 2023
2	Smart-Home-App (Lakhankumawat)	Mobile app for home automation to controls all home gadgets at one tap	Dart 97.3% C++ 1.8% Other 0.9%	2,973	88	227	Updated: June 26, 2023 Release: No release published
3	Domotics (Gizmocuz)	Home automation system to monitor different parameters like temperature, rain, wind, gas, and alert users accordingly	C++ 67.6% JavaScript 16.0% Lua 7.5% HTML 4.7% Python 1.3% CSS 1.7% Other 1.2%	6K	1.1K	360	Updated: June 23, 2023 Releases: 14 Latest: released on February 14, 2023d
4	Home-automation (Danionescu)	Home automation system using raspberry Pi.	Python 56.6% SCSS 22.0% JavaScript 14.6% C++ 6 Z.6% Other 0.2%	6,271	8	2	Updated: May 16, 2023 Releases: No release published
5	Night-patrolling -robot (RakshanaG)	IoT system that detects the presence of intruders and send live feeds of the camera and real-time data to Microsoft azura for processing.	Python 52.5% HTML 47.5%	133	1	1	Updated: April 27, 2020 Release: No Release Published
6	Quxilo2	Home automation system	C++ 94% Make file 2.3% Qmake 3.7%	163		1	Updated: January 30, 2018
7	Mylisabox/ lisa-box – LISA	Home automation system to control home gadgets	JavaScript 99.8% Other 0.2%	5,194	6	5	Updated: July 8, 2022 Releases: No release published
8	Smart security system (Cyb3rG0dzilla)	Home automation system to detect intruders and remotely monitor parameters such as temperature, humidity, gas, and flame.	Python 99.9% Other 0.1%	187	21	15	Updated: January 2, 2022 Releases: No release published

(Contd...)

Table 1: ***(Continued)***

S. no	IoT application	Description	Languages used	Lines-of-code	# Forks	# Commits	last updated date /latest release
			IoT-based air pollution monitoring systems				
1	IOT-based-air-pollution-monitoring -system (ro0t×4mit)	IoT-based air pollution monitoring system to measure and track air quality in real-time.	C++ 100%	349	0	3	Updated: July 14, 2023 Release: No release published
2	IOT-based-air-pollution-monitoring-system (Abhilash)	IoT system to detect air pollution.	C++ 100%	243	0	13	Updated: December 30, 2019 Release: No release published
3	Air-pollution -monitor (chelseabai)	IoT-based application to monitor air pollution.	JavaScript 82.5% HTML 6.4% C++ 11.1%	1,019	0	56	Updated: December 21, 2021 Release: No release published
4	Air-pollution -monitoring (Sarvesh2K)	IoT-based mobile air pollution monitoring system	Jupyter Notebook 89.1% HTML 5.2% C++ 5.2% CSS 0.5%	3,746	1	15	Updated: December 4, 2021 Release: No release published
5	IoT-based-smart-helmet-for-industrial -workers (iamkishandadhania)	IoT app to monitoring and detect air quality and hazardous events like radiation leakage, gas explosion in industrial work environment.	C++ 100%	108	nil	1	Updated: September 16, 2021 Release: No Release Published
			IoT-based road traffic management systems				
1	IoT-based-smart-traffic-light-management -system (ajeetpandeyy)	IoT-based system for road traffic management to alerts users of alternative pathways where congestion occurs	Python 100%	45	4	1	Updated: September 13, 2017
2	Aztecs-logiTraffic	IoT based traffic management and theft detection system	Jupyter Notebook 98.7% Other 1.3%	1,714	1	15	Updated: May 1, 2021 Release: No release published
3	Density-based-two-way -traffic-control -with-IoT (varshita-21)	IoT-based solution to trigger traffic lights based on traffic density at road junctions and roundabouts.	C++ 95.4% C 4.6%	130	0	8	Updated: June 16, 2021 Release: No release published
4	Smart-road -system-using-IoT (aparajitahalder)	IoT-based system to run traffic signal according to the density of the road.	Jupyter Notebook 100%	512	0	1	Updated: November 28, 2019 Release: No Release Published

(Contd...)

Table 1: ***(Continued)***

S. no	IoT application	Description	Languages used	Lines-of-code	# Forks	# Commits	last updated date /latest release
5	Traffic-control-for-narrow-roads	IoT system for traffic control on narrow roads using tensorflow	JavaScript 82.2% Python 17.8%	133	0	2	Updated: June 12, 2019 Release: No release published
			IoT-based waste management systems				
1	IBM smart-waste-management-system (IBM-Project-35221-1660282887)	Smart waste management system for cities to detect garbage level/weights in bins and alert authorized person to empty the bins accordingly.	Python 93.9% CSS 2.3% HTML 1.7% JavaScript 1.2% Nix 0.7% Yacco 0.1% Shell 0.1%	1,590	67	5	Updated: May 25, 2023 Releases: No release published
2	Smart-waste-management-system (MatthewBrandon21)	Smart waste management system for smart cities	CSS 55.3% JavaScript 38.5% C++ 5.4% Other 0.8%	7,954	2	2	Updated: December 19, 2021 Releases: No release published
3	Ecobin (shubhajitml)	IoT based waste management system for smart cities	PHP 44.2% CSS 13.0% Python 28.1% HTML 12.5% JavaScript 1.3% Hack 0.9%	9,638	7	1	Updated: April 8, 2019 Releases: No release published
4	Smart-waste-management-system (sumanth1974)	IoT based waste management System	Dart 93.2% JavaScript 1.6% Shell 0.3% Kotlin 0.1% Objective-C 29% Java 1.6% Swift 0.3%	4,152	4	17	Updated: October 7, 2022 Releases: No release published
5	Smart waste management system (matthewbrandon21)	Smart waste management system for cities	CSS 55.3% JavaScript 38.5% C++ 5.4% Other 0.8%	2,709	2	36	Updated: December 19, 2021 Releases: No release published
			IoT-based health monitoring system				
1	IoT-based-healthcare-system (MohanadSinan)	IoT-based patient health monitoring system	C++ 100%	537	7	1	Updated: September 23, 2023
2	HealthFog	Smart healthcare system based on deep learning for automated diagnosis of heart diseases	PHP 56.1% Python 37.4% Shell 6.5%	450	8	44	Updated: December 16, 2020 Releases: No release published

(Contd...)

Table 1: ***(Continued)***

S. no	IoT application	Description	Languages used	Lines-of-code	# Forks	# Commits	last updated date /latest release
3	IoT-based-health-monitoring-system-in-azure (hectorTa1989)	Smart healthcare system to monitor vital signs such as heart-rate and temperature. alert patients/doctors where abnormality exist.	Python 100%	42	0	11	Updated: March 16, 2023 Releases: No release published
4	Smart-healthcare-patient-monitoring-system	Smart health monitoring system to detect vital signs	C++ 69.0% Python 31.0%	219	4	5	Updated: June 8, 2018 Releases: No release published
5	Smart-health-monitoring-system (hkush8289)	Android application to calculate patient's BMI and provide step counter, sleep suggestions and other relevant remedies for health challenges.	Java 100%	1563	1	8	Updated: January 31, 2020 Releases: No release published
			IoT-based weather forecast system				
1	iot-zambretti-weather-forecasting	IoT-based system for weather forecast	Shell 100%	16	1	6	Updated: April 1, 2021Releases: No release published
2	IoT-based-weather-reporting-system (Rashmika-B)	IoT-based system to monitor pressure, temperature and humidity, then alert users accordingly on possibility of rainfall	Python 100%	169	3	2	Updated: July 15, 2021Release: No Release Published
3	Smart-agriculture-system-using-IoT	IoT-based app to monitor temperature, humidity and soil moisture parameters to provide guides to farmers	Python 100%	44	5	24	Updated: August 21, 2020Release: No release published
4	Smart-weather-monitoring-system	Smart weather monitoring system	JavaScript 43.9% HTML 24.4% C++ 25.2% CSS 6.5%	1,273	0	3	Updated: October 8, 2022 Release: No release published
5	IoT-based-Smart-farming-using-machine-learning	Intelligent system to monitor temperature, humidity and other conditions in the farm and advice farmers accordingly.	Python 100%	211	1	1	Updated: September 13, 2022 Release: No Release Published

Based on the type of functionality provided by the identified classes, they were categorized into two groups, namely, *classes providing basic* infrastructure functions and classes providing core logic functions. Basic infrastructure functions include those for setting up and managing the system infrastructure, namely, device management, network connectivity, message communication, and others. Core logic functions on the other hand are those that perform the core functions of the app which could be switching a lightbulb on/off, blowing a security alarm or emitting dangerous items to scare intruders in the case of smart home security or alerting a patient/doctor of abnormality in patient's body vitals in the case of smart health monitoring system.

Using Smart-Home-App (Lakhankumawat) – one of the case Apps as a case in point, some of the classes identified as providing basic infrastructure functions include MqttMessage for message communication, ESP8266WiFi and WiFiClient for network connectivity and Arduino for device management. Furthermore, some of the classes identified as providing core logic functions include SmartFanViewModel, SmartACViewModel, ChangeNotifier, BaseModel, intensity, PopUpAlert, and PopUpWarning performing different core logic functions, respectively. In line with the aim of this research, emphasis was placed on classes providing core logic functions.

In view of the above, the process of class identification and classification was undertaken for all gits downloaded for the research. Core logic classes and the type of functions provided in the apps were recorded and compared for similarities and differences accordingly. Being that our research interest is in common core logics, emphasis at this point was placed on similarities among core logic functions in the reviewed Gits.

After a careful review as well as comparison and classification of the core logic classes, four core logic functions were found to be common among the different use cases although they are applied differently in the use case. These are, functions relating to:

i. Analysis of generated data from IoT devices to retrieve those data that are relevant to the use case for further processes. In this research, these category of functions are termed Smart Agent Function
ii. Continuous monitoring of the system for changes and notifying other components of changes for corresponding actions. Accordingly, these category of functions are designated as Condition Monitoring Function
iii. Prediction of systems behavior in line with the expectations of the actors. For the purpose of this research, these categories of functions are termed Predictive Analytics Function.
iv. Planning and execution of the specified tasks using optimization technique. Finally, these functions are designated as AI Tasks Planner & Executor.

Table 2 shows the core logic functions and some of the classes identified to be associated with them in the reviewed use cases.

RESULTS AND DISCUSSION

At present, there are architectures and models that support reuse of IoT functions in IoT software development projects of which oneM2M model is considered the most widely adopted.[1,29,34,35] However, these models and architectures only support reuse of infrastructural services such as device management, connectivity, data messaging, and others thereby making it difficult to harness the full benefits of software reuse in IoT software engineering. With this, the need for architectures and models beyond reuse of basic infrastructure services, to support the reuse of core logic functions is overwhelming.

Research findings reveal similarities in core logic implementation of IoT apps within and across domains which can be repackaged as common services for reuse. In view of this, remodeling of existing common service layer models to accommodate core logic functions will advance the course of IoT reuse. In this research, remodeling of oneM2M model with the identified core logic services is illustrated. Choice of this model is due to its global acceptability and use among practitioners.

Figure 2 presents the proposed remodeled oneM2M common service layer with two extended IoT basic services and four IoT core logic services added to the existing fourteen services in the oneM2M model. A brief description of these services follows thus:

IoT Basic Services

The 14 oneM2M Common Services are maintained in the remodeled common service layer where they are designed as IoT basic services. They are to serve the purpose for which they were intended and in a manner designed in the existing oneM2M model.

IoT Core Logic Services

The core logic services are services needed to perform the core functions of IoT apps. Research findings reveal that these functions are applicable to all gits in different use cases reviewed and implemented using classes in the source code without reused potentials across domains since they are not provided in the common service layer. Having these services repackaged into microservices as a collection of common core services in the common service layer will provide for easy reuse of these important functions across domains. Accordingly, the four core logic functions identified in this research are provided for in the remodeled layer as IoT core logic services. These services are explained thus:

Table 2: Identified core logic functions for gits reviewed in different use cases

IoT use case	Common core logic 1 (smart agent)	Common core logic 2 (condition monitor)	Common core logic 3 (Predictive Analytics) with real-life scenario	Common core logic 4 (AI tasks planner and executor) with real-life scenario
IoT-based home automation system	Function performed: Analyze generated data and retrieve data relevant to automated home in the covered area. Some of the classes identified in reviewed gits: SmartFanViewModel SmartACViewModel SmartLightViewModel SmartSpeakerViewModel SmartTV UploadImage DeviceList ColorPickerSheet	Function performed: Monitor the system and notify other components of changes in the home and required actions to be taken. Some of the classes Identified in reviewed gits: ChangeNotifier BaseModel TimeContainer ElectricityUsage	Function Performed: Predict systems behavior inline with the expectations of the actors, for example, 1. suspicious movement detected, home might be attacked 2. increasing smoke intensity, home might be ablaze Some of the classes Identified in reviewed gits: intensity WeatherContainer Consumption	Function Performed: Plan and execute the specified tasks using optimization technique, for example, Suspicious movement: - Notify house owner/ security agents - Raise security alarm - emit dangerous substances to scare/ dispel intruders Increasing Smoke Intensity: - Notify house owner/fire service - Raise fire alarm - Activate fire extinguisher to quench fire Some of the classes Identified in reviewed gits: PopUpAlert; PopUpWarning SavingsContainer AddEventDialog; SplashScreen
IoT-based air pollution monitoring system	Function performed: Analyze generated data and retrieve data relevant to air pollution monitoring in the covered area. Some of the classes identified in reviewed gits: Get_value Upload PostpollutionData FetchPollutionData	Function performed: Monitor the system and notify other components of changes in air pollutants and required actions to be taken. Some of the classes identified in reviewed gits: loop ParseData MonitorPollutantLevel	Function performed: predict systems behavior inline with the expectations of the actors, for example, Increasing contents of pollutants in the air like carbon monoxide, nitrogen dioxide (NO_2), Sulfur Dioxide (SO_2), fine particulates ($PM_{2.5}$) 1. Air Pollution is imminent Some of the classes identified in reviewed gits: setPollution pollutionModel computeAirQuality calculate_aqi	Function performed: plan and execute the specified tasks using optimization technique, for example, Increasing contents of pollutants in the air like - alert concerned authorities - block source of pollutants where possible - refresh the air Some of the classes identified in reviewed gits: pollutionPieChart SensorLineChart pollutionLineChart printAirQuality gradient_descent

(Contd...)

Table 2: ***(Continued)***

IoT use case	Common core logic 1 (smart agent)	Common core logic 2 (condition monitor)	Common core logic 3 (Predictive Analytics) with real-life scenario	Common core logic 4 (AI tasks planner and executor) with real-life scenario
IoT-based road traffic management system	Function performed: Analyze generated data and retrieve data relevant to Road Traffic management of the covered area. Some of the classes identified in reviewed gits: videoCapture car_cascade	Function performed: Monitor the system and notify other components of changes in the road traffic and required actions to be taken. Some of the classes identified in reviewed gits: detectMultiScale getStructuringElements reshape_ipts	Function performed: predict systems behavior inline with the expectations of the actors, for example, 1. long queues of vehicles detected; traffic gridlock is imminent 2. road blockage by faulty trucks detected; gridlock is imminent 3. road accident detected; gridlock is imminent Some of the classes identified in reviewed gits: PredictTtrafficStatus	Function performed: plan and execute the specified tasks using optimization technique e.g. Detected long queues of vehicles - Alert commuters of alternative routes - Alert Road Safety Agency - extend duration of traffic flow on affected road to ease traffic Some of the Identified classes identifies in reviewed gits: VideoCapture Car_cascade DataExtract DataVisualise
IoT-based smart waste management system	Function performed: Analyze generated data to retrieve data that are relevant to Waste Management of the covered area. Some of the classes identified in reviewed gits: HX711 DistanceSensor GPS_info TopContainer BinDetails	Function performed: Monitor the system and notify other components of changes in deposited waste and required actions to be taken. Some of the classes identified in reviewed gits: myCommandCallback myCommandPublsihCallback BinDetailsState AvailableBins	Function performed: predict systems behavior inline with the expectations of the actors, for example, Increasing level of generated waste - waste management safety might be exceed Some of the classes identified in reviewed gits: myOnPublishCallback DirectionMaps Message	Function performed: plan and execute the specified tasks using optimization technique, for example, Increasing level of generated waste - Alert concerned authorities - empty waste bins Some of the classes identified in reviewed gits: cleanAndExit Splash mensajeToken
IoT-based health monitoring system	Function performed: Analyze generated data and retrieve data relevant to health monitoring of the covered area. Some of the classes identified in reviewed gits: Read_temp detectMultiScale readVitals	Function performed: Monitor the system and notify other components of changes in patient's health status and required actions to be taken. Some of the classes identified in reviewed gits: arduinoSerialMonitorVisual SerialOutputWhenBeatHappens loop	Function performed: predict systems behavior inline with the expectations of the actors, for example, Abnormal status of body vitals: - increasing blood pressure, sugar level; health challenge is imminent Some of the classes identified in reviewed gits:	Function performed: plan and execute the specified tasks using optimization technique, for example, Abnormal status of body vitals: - Alert Doctor/patient Some of the classes identified in reviewed gits: sendDataToSerial SerialOutput filterWarnings imShow

(Contd...)

Table 2: ***(Continued)***

IoT use case	Common core logic 1 (smart agent)	Common core logic 2 (condition monitor)	Common core logic 3 (Predictive Analytics) with real-life scenario	Common core logic 4 (AI tasks planner and executor) with real-life scenario
			setup heartModel train_test_split BaggingClassifier cascadeClassifier	
IoT-based weather forecast system	Function performed: Analyze generated data to retrieve data relevant to weather forecast for the covered area. Some of the classes identified in reviewed gits: createTemperatureGauge createHumudityGauge readBMP180	Function performed: Monitor the system and notify other components of changes in weather conditions and corresponding actions. Some of the classes identified in reviewed gits: *loop* *weatherMonitor*	Function performed: predict systems behavior inline with the expectations of the actors, for example, Abnormal weather data: - increasing volume of rainfall, flood may occur - cloudy atmosphere, rain may fall Some of the classes identified in reviewed gits: predict predictWeather	Function performed: plan and execute the specified tasks using optimization technique, for example, Abnormal weather recorded data: - Alert concerned authorities Some of the classes identified in reviewed gits: createTemperatureChart createHumudityChart sendNotification

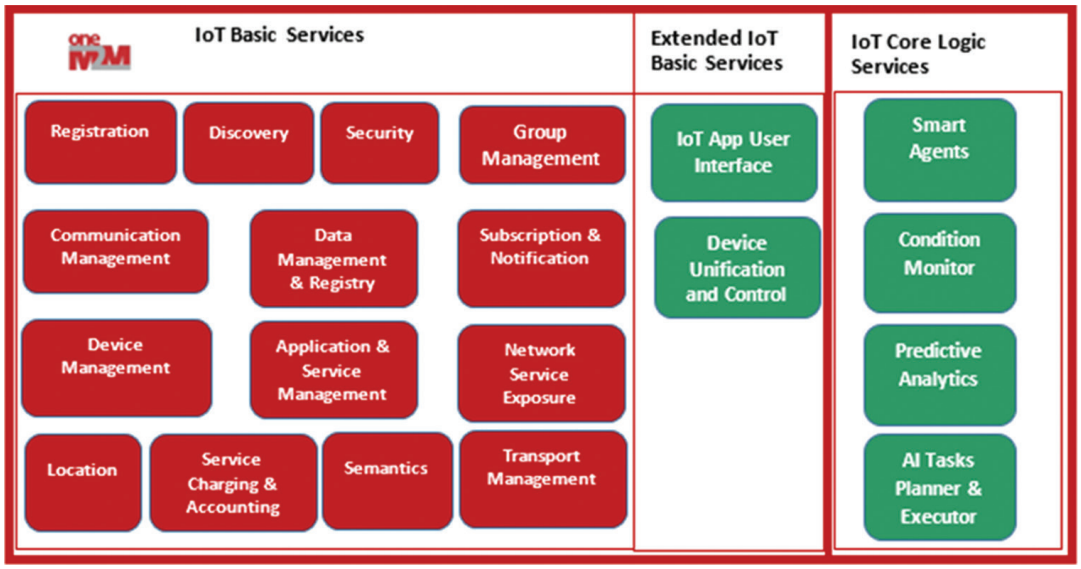

Figure 2: The proposed remodeled oneM2M common service layer

Smart agent

This service is responsible for preliminary analysis of generated data to retrieve those data that are relevant to the use case. Since this function is performed by every IoT App, presenting it as a common service that can be reused by developers will further reduce development time and efforts.

Condition monitor

It refers to service that monitors the IoT system continuously to detect changes and notify smart agent and AI task planner and executor of the changes and corresponding actions to be taken. In every IoT use cases, there is need to monitor changes and respond to it accordingly. For instance, in the case of

smart home security, continuous monitoring of the system will detect the presence of an intruder and notify other components accordingly. Furthermore, presenting this as a common service that can be reused is advantageous as it's capable of reducing the development time and efforts while enhancing product quality.

Predictive analytics

This service defines/predicts systems behavior inline with the expectations of the actors. It uses predictive analytics for its predictions. Using analytical models, the service could defines/predicts the possible state of the systems based on generated data. Developers will benefit more from having this as a common service that can be reused in IoT development and deployment projects.

AI tasks planner and executor

This services uses optimization method to plan the task to be executed and execute the optimized solution accordingly. Following the predicted state of the system, this service will generate the possible course of actions needed in the circumstance and execute the ones considered to be the optimized solution. For instance, in the case of smart home security, taking suspicious movement as an example, possible course of actions could be notify house owner/security agents, raise security alarm, and emit dangerous substances to scare/dispel intruders.

In this circumstance, the AI Task planner and executor will execute the one considered to be optimal solution say raise security alarm; in some cases, all of the generated options could necessary in the given circumstance and executed accordingly. Having a common service of this category in the common service layer for reuse will be of great advantage.

Extended IoT Basic Services

In terms of extended basic IoT services, two services are proposed namely, IoT app user interface and device unification and control which are explained thus:

IoT app user interface

This service will provide the required components for creating the application user interface which include maps, things card, things list, measured values, event list, timeline, and sensor charts. Although these components are available in application enablement platforms as plugins that can be reused[1,6] providing them as services in the common services layer will increase developers productivity and reduces development time.

Johnson *et al*.[38] maintains that by decoupling the UI and APIs from the logic and data sources of a microservice, the location of data processing or function performance may be conveniently and easily changed without modifying the implementation of use cases that utilize these functions. In other words, data processing may be performed anywhere without affecting how it is subsequently requested and/or used, and without constraining the performance to a particular location. This is an important benefit that this proposed remodeling seeks to achieve.

Device unification and control

The key challenge of IoT development is the difficulty in developing IoT apps that can easily connect with a plethora of technologies and devices from different manufacturers that can work together as a unified system. Device unification and control service is needed to address systems interoperability challenge. Having a dedicated service for device unification and control in the common layer that standardizes the use heterogeneous devices from different manufacturers in a unified system will address this interoperability challenges. In this case, this service will be a collection of software components that can easily detect heterogeneous devices connected to the system to enable them work as a unified system that promote interoperability.

IMPLEMENTATION OF THE PROPOSED REMODELED COMMON SERVICES LAYER

The remodeled common service layer is implemented as a middleware service layer where the three categories of functions are presented as microservices. The functions can be developed using suitable IoT programming languages, repackaged into containers, and exposed as services that are accessible through oneM2M REST API (i.e., Representational state transfer) API. This can be achieved in the following simple steps:

i. Using a suitable microservice framework, for example, spring boot and flask create the microservice say smart agent that implements the smart agent function across domain. Choice of the framework depends on the programming language to be used in coding the IoT logic function.
ii. Containerize the microservice together with the supporting REST API using a container tooling kit like Docker or any as may be considered suitable by the developer.
iii. Point the container image to the cloud pointer using appropriate configurations and deployment rules required to make it accessible through the REST API.

The oneM2M RESTful API provides the means of communication between the CSE and the AEs as indicated in the oneM2M Restful Architecture given in Figure 3. AEs refers to any entity in the application layer that implements an M2M service logic. Examples of AEs are an instance of smart home system, remote health monitoring system, and smart waste management system. With the proposed remodeling of the oneM2M model, implementation of the service logic at AE will be easier through reuse of common core logics from the CSE. The CSE represents an instance of the set of "common service functions" of the oneM2M service layer. The remodeled common service layer provides a richer set of common services which are the three

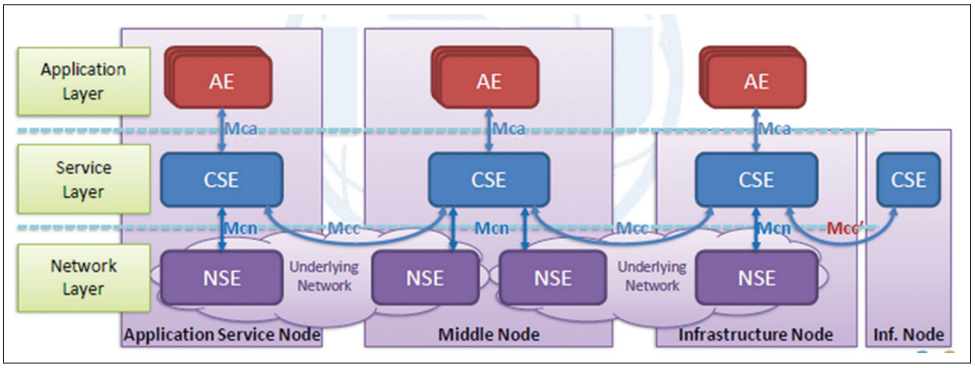

Figure 3: RESTful architecture (Source:[35])

categories of common services, namely, the 14 IoT basic services, 2 extended IoT basic services, and 4 IoT core logic services.

The network services entity (NSE) of the architecture provides the following services from the underlying network to the CSEs, namely, location services, device triggering, and sleep modes. The figure also indicates reference points, which are interfaces between service providers. There reference points are Mca – interfacing between CSE and AE, Mcn – interfacing between CSE and NSE, Mcc – interfacing between CSE and CSE, and Mcc' – interfacing between CSE and CSE in different network infrastructures.

Furthermore, the communication between CSEs and AEs: Communication between the CSEs and AEs is through the oneM2M RESTful APIs. The oneM2M RESTful APIs handle create, retrieve, update, delete, and notification operations between the layers. Communication can originate from an AE or a CSE depending on the type of operation which may occur through the exchange of primitives across three reference points, namely, Mca, Mcc, and Mcc'.

CONCLUSION

Reuse in IoT software engineering is currently challenged by lack of models and architectures to support reuse of core logic functions within and across domains. Hence, IoT reuse is limited to the reuse of infrastructure services. In this research, four core logics functions common to IoT solutions across different use cases that can be repackaged as common layer services for reuse are identified. These services include smart agent, condition monitor, predictive analytics, and AI tasks planner and executor. Furthermore, two other services needed to extend the IoT basic services have been identified, namely, device unification and control as well as IoT UI services. To advance the course of IoT reuse, remodeling of existing common service layer models like oneM2M model to incorporate these identified services is proposed.

In view of this, since the contributions of oneM2M model are significant in IoT reuse and widely adopted, it's further remodeling as proposed in this research to support the reuse of core logics will be of great benefits in terms of enhanced developers' productivity and products quality as well as reduced development time and costs.

RECOMMENDATIONS

The following recommendations are necessary:

i. Implementation of the remodeled oneM2M common service layer to include the proposed four common core logic services and the two extended basic services as additional services is highly recommended.
ii. Further, research efforts aimed at enhancing the process of creating common core logic services and extended basic services for reuse are also recommended.
iii. Furthermore, further research efforts aimed at discovering more core logic functions from IoT systems to boast the quality of core logic reuse is highly recommended.

REFERENCES

1. Bhavana BC, Vathsala GC, Rakshitha BH. A survey: Internet of things (IOT) technologies, applications. Int J Res Appl Sci Eng Technol 2022. [Last accessed on 2023 Mar 15].
2. Krishna V, Padshah KS. Current internet of things: A modernity. Int J Creat Res Thoughts (IJCRT) 2023. [Last accessed on 2023 Jul 12].
3. Sorri K, Mustafee N, Seppänen M. Revisiting IoT definitions: A framework towards comprehensive use. Technol Forecast Soc Change 2022;179:121623.
4. Patel A. Unlocking the Power of IoT for Your Business. Forbes Technology Council Post; 2023. [Last accessed on 2023 Jul 12].
5. Statista. Number of Internet of Things (IoT) Connected Devices Worldwide from 2019 to 2021, with Forecasts from 2022 to 2030; 2016. [Last accessed on 2022 May 10].
6. Smiari P, Bibi S, Feitosa D. Examining the reuse potentials of IoT application frameworks. J Syst Software 2020;169:110706.
7. Wehlitz R, Jauer F, Rößner I, Franczyk B. Increasing the Reusability of IoT-Aware Business Processes. Vol. 22. Annals of Computer Science and Information Systems, ACSIS; 2020. p. 17-22. [Last accessed on 2022 May 10].

8. Ekanem BA, Woherem E. Dealing with Components Reusability Issues as Cutting-edge Applications Turn Legacy. In: SAI Computing Conference Proceedings 2016, London, UK.
9. Mäkitalo N, Taivalsaari A, Kiviluoto A, Mikkonen T, Capilla R. On opportunistic software reuse. Computing 2020;102:2385-408.
10. Capilla R, Gallina B, Cetina C, Favaro J. Opportunities for software reuse in an uncertain world: From past to emerging trends. J Software Evol Pract 2019;31:e2217.
11. Ekanem BA, Agbele KK. A review of software components identification methods and quality assessment criteria. Eur J Appl Sci 2021;9:194-209.
12. Smiari P, Bibi S, Feitosa D. Examining the Reusability of Smart Home Applications: A Case Study on Eclipse Smart Home. In: Conference Proceeding, 18th International Conference on Software and Systems Reuse; 2019. [Last accessed on 2022 May 10].
13. Arumugam SS, Badrinath R, Herranz AH, Höller J, Azevedo CR, Xiao B, *et al*. Accelerating Industrial IoT Application Deployment through Reusable AI Components. In: Computer Science 2019 Global IoT Summit (GIoTS); 2019. [Last accessed on 2022 May 10].
14. Jarwar MA, Kibria MG, Ali S, Chong I. Microservices in web objects enabled IoT environment for enhancing reusability. Sensors (Basel) 2018;18:352.
15. Industry IoT Consortium. Advancing the Industrial Internet of Things. Industry IoT Consortium White Paper; 2023. [Last accessed on 2023 Jul 12].
16. Naghib A, Navimipour NJ, Hosseinzadeh M, Sharifi A. A comprehensive and systematic literature review on the big data management techniques in the internet of things. Wirel Netw 2023;29:1085-144.
17. Lekidis A, Stachtiari E, Katsaros P, Bozga M, Georgiadis CK. Model-based design of IoT systems with the BIP component framework. Softw Pract Exp 2018;48:1167-94.
18. Nastic S, Sehic S, Vogler M, Truong HL, Dustdar S. PatRICIA-A Novel Programming Model for IOT Applications on Cloud Platforms. In: Proceedings of the IEEE 6th International Conference on Service-Oriented Computing and Applications (SOCA), Koloa, HI, USA; 2013. p. 53-60.
19. Alayaa MB, Banouara Y, Monteila T, Chassota C, Driraa K. OM2M: Extensible ETSI-compliant M2M service platform with self-configuration capability. Proc Comput Sci 2014;32:1079-86.
20. Swetina J, Lu G, Jacobs P, Ennesser F, Song J. Toward a standardized common M2M service layer platform: Introduction to oneM2M. IEEE Wirel Commun 2014;21:20-6.
21. Bonino D, Alizo M, Alapetite A. Almanac: Internet of Things for Smart Cities. In: Proceedings of the 2015 3rd International Conference on Future Internet of Things and Cloud (FiCloud), Rome, Italy; 2015.
22. Newman S. Building Microservices: Designing Fine-Grained Systems. Sebastopol, CA, USA: O'Reilly Media, Inc.; 2015.
23. Al-Fuqaha A, Guizani M, Mohammadi M, Aledhari M, Ayyash M. Internet of things: A survey on enabling technologies, protocols, and applications. IEEE Commun Surv Tutor 2015;17:2347-76.
24. Alaya MB, Medjiah S, Monteil T, Drira K. Toward semantic interoperability in one M2M architecture. IEEE Commun Mag 2015;53:35-41.
25. Girau R, Martis S, Atzori L. Lysis: A platform for IoT distributed applications over socially connected objects. IEEE Internet Things J 2017;4:40-51.
26. Spalazzese R, Pelliccione P, Eklund U. Intero: An interoperability model for large systems. IEEE Softw 2017;37:38-45.
27. OEP. Achieving Semantic Interoperability Using RDF and OWL-v10; 2017. Available from: https://www.w3.org/2001/sw/bestpractices/oep/semint [Last accessed on 2017 Sep 19].
28. LogMeIn. Best IoT Platform Solution: Xively by LogMeIn. Available from: https://www.xively.com/xively-iot-platform [Last accessed on 2018 Jan 11].
29. SimfonyBlog. IoT Platforms: Vertically versus Horizontally Layered Architecture, Simfony Mobile; 2018. [Last accessed on 2022 Apr 08].
30. Martín C, Hoebeke J, Rossey J, Díaz M, Rubio B, Van den Abeele F. Appdaptivity: An internet of things device-decoupled system for portable applications in changing contexts. Sensors (Basel) 2018;18:1345.
31. Ahmad S, Mahmood F, Mehmood A, Kim D. Design and implementation of decoupled IoT application store: A novel prototype for virtual objects and discovery. Electronics 2019;8:285.
32. Alwis AA, Barros A, Fidge C, Polyvyanyy A. Microservice Remodularization of Monolithic Enterprise Systems for Embedding in Industrial IoT Networks. In: Advanced Information Systems Engineering 33rd International Conference, CAiSE 2021, Melbourne, VIC, Australia; 2021. p. 432-448.
33. oneM2M. oneM2M Basics. [Last accessed on 2022 Apr 15].
34. Elloumi O. OneM2M and Its Role in Achieving Interoperability in IoT. In: Regional Standardization Forum for Bridging the Standardization Gap (BSG), Riyadh, Saudi Arabia; 2017.
35. Mediawiki. OneM2M Overview; 2023. [Last accessed on 2023 Jul 12].
36. Babaria U. Why IoT Development Needs Microservices and Containerization. eInfoChips Publication; 2019. [Last accessed on 2020 Aug 11].
37. oneM2M. oneM2M Is the Global Standards Initiative for Machine to Machine Communications and the Internet of Things; 2018. [Last accessed on 2023 Apr 12].
38. Johnson C, Maes HS, Kim W. Microservice with Decoupled User Interface. United States Patent Application Publication; 2017. [Last accessed on 2020 Jun 18].