Landscape of Architecture and Design Patterns for IoT Systems

Hironori Washizaki, *Member, IEEE*, Shinpei Ogata, *Member, IEEE*, Atsuo Hazeyama, *Member, IEEE*, Takao Okubo, *Member, IEEE*, Eduardo B. Fernandez, *Senior Member, IEEE*, and Nobukazu Yoshioka, *Member, IEEE*

***Abstract*—Due to the widespread proliferation of today's Internet of Things (IoT), a system designer needs the IoT system and software design patterns to assist in designing scalable and replicable solutions. Patterns are encapsulations of reusable common problems and solutions under specific contexts. Many IoT patterns have been published, such as IoT design patterns and IoT architecture patterns to document the successes (and failures) in IoT systems and software development. However, because these patterns are not well classified, their adoption does not live up to their potential. To understand the reasons, we conducted a systematic literature review. From the 32 identified papers, 143 IoT architecture and design patterns were extracted. We analyzed these patterns according to several characteristics and outlined directions for improvements when publishing and adopting IoT patterns. Of the extracted patterns, 57% are non-IoT patterns, suggesting that IoT systems and software are often designed via conventional architecture and design patterns that are not specific to IoT design. Although most IoT design patterns are applicable to any domain, IoT architecture patterns tend to be domain specific, implying that the unique nature of IoT adoption in specific domains appears at the architecture level. As more domains adopt IoT, the number of domain-specific IoT design patterns should increase. In terms of quality attributes, many IoT patterns address compatibility, security, and maintainability.**

***Index Terms*—Architecture, design, Internet of Things (IoT), patterns, survey, systematic literature review (SLR).**

I. Introduction

THE INTERNET of Things (IoT) is expected to bridge diverse Internet collaborative technologies to enable new services and applications by connecting physical objects (i.e., devices, such as sensors and actuators that are tied with the physical entities to be monitored and manipulated), together in support of intelligent decision making to empower teams across the world [1].

Thus, IoT aims to bring connectivity to almost every electric device in physical space. Although IoT extends connectivity to everyday things, this increase in connectivity creates many challenges [2]. Since the application spread in today's IoT is wide and is typically structured in market-oriented groups, a system designer needs the IoT system and software design patterns to assist in designing for scalable and replicable solutions [3]. Patterns are encapsulations of reusable common problems and solutions under specific contexts. To document the successes (and failures) in IoT systems and software development, IoT patterns, including IoT design patterns and IoT architecture patterns, have been published.

In general, systems and software design processes have two major phases [4] with different abstraction levels: 1) architecting (i.e., architectural design) and 2) design (i.e., detailed design). These two phases can be classified into two corresponding types: 1) architecture patterns and 2) design patterns. Moreover, architecture patterns that do not emphasize problems and rationales are called architecture styles. Although some IoT architecture styles have been studied [5], IoT architecture and design patterns at different abstraction levels are not well classified or researched. Consequently, adopting such patterns may not resolve problems or have the desired impact.

The abstraction level can be important when describing, examining, and reusing IoT patterns. The same is true for domain specificity and quality attributes. IoT is basically constituted of the traditional technical fields, such as wireless sensor networks, and embedded and control systems. Thus, "IoT patterns" may not be exclusively IoT patterns. Non-IoT patterns as well as some domain-specific IoT can be utilized for IoT systems and software design. Moreover, it is important to identify which quality attributes are addressed by the target IoT pattern to be reused. IoT architecture and design patterns should address interoperability since, by definition, IoT is about ensuring interoperability among objects. However, other attributes may also be addressed. Based on the definition in ISO/IEC 25010:2011 [6], interoperability for IoT systems means the degree that two or more IoT devices and systems exchange and use information.

The contribution of this article is an overview of the current landscape of IoT architecture and design patterns to identify

Manuscript received November 16, 2019; revised May 14, 2020; accepted June 3, 2020. Date of publication June 18, 2020; date of current version October 9, 2020. This work was supported in part by JSPS KAKENHI under Grant 16H02804 and Grant 17K00475, and in part by SCAT and enPiT-Pro Smart SE. *(Corresponding author: Hironori Washizaki.)*

Hironori Washizaki is with the Faculty of Science and Engineering, Waseda University, Tokyo 1698555, Japan, also with the National Institute of Informatics, Tokyo 101-8430, Japan, also with System Information Company Ltd., Tokyo 104-0054, Japan, and also with eXmotion Company Ltd., Tokyo 141-0032, Japan (e-mail: washizaki@waseda.jp).

Shinpei Ogata is with the Graduate School of Engineering, Faculty of Engineering, Shinshu University, Nagano 390-8621, Japan.

Atsuo Hazeyama is with the Department of Information Science, Tokyo Gakugei University, Tokyo 184-0015, Japan.

Takao Okubo is with the Institute of Information Security, Kanagawa 221-0835, Japan.

Eduardo B. Fernandez is with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, FL 33431 USA.

Nobukazu Yoshioka is with the Information Systems Architecture Research Division, National Institute of Informatics, Tokyo 101-8430, Japan.

Digital Object Identifier 10.1109/JIOT.2020.3003528

shortcomings and suggest improvements when publishing and adopting IoT patterns. Specifically, a complete set of IoT patterns available in the literature is analyzed. The authors found 32 papers published from 2014–2018. The four research questions below are intended to constructively determine the direction for improvement.

RQ1. How does academic literature address IoT architecture and design patterns? To answer this question, we conducted a systematic literature review (SLR) of the academic literature. We analyzed the 32 identified papers and extracted 143 patterns.

RQ2. Are all existing IoT architecture and design patterns really IoT patterns? To answer this question, we distinguished between architecture and design patterns specific to IoT systems and non-IoT patterns that are applicable to any system or software design. Of the 143 patterns, 61 addressed IoT-specific problems and solutions, and the remaining 82 were not IoT-specific patterns.

RQ3. Can IoT architecture and design patterns be classified? To answer this question, we classified these IoT patterns with respect to three characteristics: 1) abstraction level; 2) domain specificity; and 3) quality attributes.

RQ4. What IoT architecture and design patterns exist? To answer this question, we showed that IoT patterns not only exist but are related to different abstraction levels, domain specificities, and quality attributes. We also provided examples of such patterns.

The remainder of this article is organized as follows. Section II summarizes related work. Section III presents our SLR and its results. Section IV discusses our results. Section V concludes this article and provides a future direction.

II. Related Work

Surveys have been conducted on general architecture and design patterns, e.g., [7]–[9]. Most focus on the object-oriented design. Moreover, surveys on architecture and design patterns exist for specific domains and quality attributes, such as multiagent systems [10], machine learning systems [11], or secure systems [12].

Ahmadi *et al.* [13] conducted an SLR of IoT-specific to the healthcare domain. Asghari *et al.* [14] conducted an SLR of IoT applications. Giudice [15] conducted a literature review on the role of IoT on the business process management in terms of promotion of knowledge flow, innovation, and competitiveness. None of these reviews focused on IoT patterns. Ray [16] surveyed existing IoT cloud platforms. However, Ray's work is not formalized and the scope of it is limited to the domain concerning IoT clouds.

For the domain of IoT systems and software design, case studies, best practices, and patterns are mostly available as independent documents. To grasp the entire picture, several surveys have been reported [1], [5]. Muccini and Moghaddam [5] conducted a systematic mapping study of IoT architecture styles. They identified seven architecture styles, including layered architecture and service-oriented architecture. In addition, Aly *et al.* conducted an SLR of both IoT interoperability issues and state-of-practices of IoT technologies in the industry, which highlighted integration challenges related to IoT that have significantly shifted the landscape of Internet-based collaborative services and applications.

However, previous studies did not classify IoT architecture and design patterns at different abstraction levels. This study is the first comprehensive survey on IoT architecture and design patterns.[1]

III. Systematic Literature Review

A. Process and Query

We performed an SLR of the academic literature to collect architecture and design patterns for IoT systems and software. An SLR aims to assess scientific papers and group concepts around a topic. We chose Scopus[2] as the search engine since it is effectively used in SLRs of software engineering and the search results can be exported. The database covers many major publishers, including IEEE, ACM, Springer Nature, Wiley Blackwell, Taylor & Francis, and Elsevier. Furthermore, the database provides a mechanism to perform keyword searches.

Fig. 1 overviews the process adopted to identify relevant papers. Our process has four steps as follows.

1) *Initial Search:* We executed the following query on titles, abstracts, and keywords of papers regardless of time and subject area. We used no publications period restrictions. We found 63 papers published from 2014 to 2018.

```
"IoT" AND ( "design pattern" OR "
   architecture pattern" )
```

2) *Impurity Removal:* Due to the nature of the involved data source, the search results included elements that are clearly not research papers, such as abstracts and international standards. Removing such results left 56 papers.
3) *Inclusion and Exclusion Criteria:* For each paper, two of the authors vetted whether they should be included in our SLR by applying the following criteria. First, the titles and abstracts followed by the entire paper were read to determine whether the paper pertained to IoT architecture and design patterns. Using the definition of our query, 32 scholarly papers [18]–[49] were identified.
   a) *Inclusion:* Papers addressing patterns to design IoT systems and software that are written in English.
   b) *Exclusion:* Papers focusing on IoT but not explicitly dealing with architecture and design patterns or duplicate papers of the same study.
4) *Data Extraction:* The following information was collected from each paper to answer the research questions: publication title, publication year, publication venue, types of patterns proposed or used, pattern names,

[1]This article is an extension of a paper presented at the 1st International Workshop on Software Engineering Research & Practices for the IoT (SERP4IoT 2019) [17]. In this article, we extend research questions and pattern analysis as well as corresponding discussions. We also added related work.

[2]https://www.scopus.com/

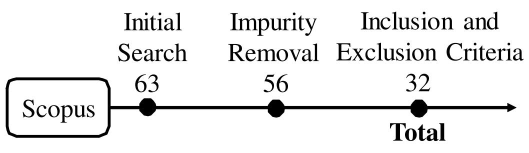


Fig. 1. Selection process and the number of papers remaining after each activity.

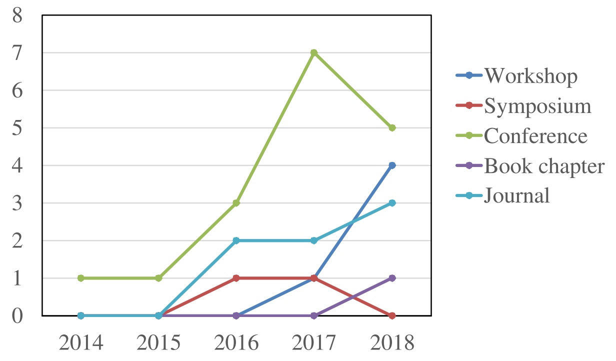


Fig. 2. Numbers of documents per year.

domain names in the case of domain-specific IoT patterns, and quality attributes addressed.

B. RQ1. How Does Academic Literature Address IoT Architecture and Design Patterns?

Our SLR revealed that IoT architecture and design patterns are very popular due to the promotion of IoT systems and software in recent years. Fig. 2 shows the annual trend in the number of papers related to IoT architecture and design patterns by publication type.

The most common publication types are conference papers (17), journals (7), workshops (5), symposiums (2), and a refereed book chapter (1). The most common publication type is conference papers followed by journals, suggesting that certain IoT patterns are maturing. However, the high number of conference papers suggests that the entire topic of IoT architecture and design patterns is in its early stage. Since 2016, IoT patterns have garnered increased research attention each year.

RQ1. How does academic literature address the IoT architecture and design patterns? There are 32 academic papers related to IoT architecture and design patterns. Most are conference papers followed by journal publications. The high number of conference papers indicates that the entire topic of IoT architecture and design patterns is in its early stage, but the presence of journal articles suggests that some types of IoT patterns are maturing.

C. RQ2. Are All Existing IoT Architecture and Design Patterns Really IoT Patterns?

Five of the authors plus two researchers indicated in the acknowledgment read a seventh of the papers. For each paper, patterns were extracted and the specificity of the content to IoT was analyzed. All patterns were independently vetted by another author. Overall, the 32 papers contained 143 patterns.

Among the 143 patterns, 82 (57%) were considered non-IoT patterns in terms of domain specificity. Table I shows the list of extracted non-IoT architecture and design patterns and their abstraction levels (i.e., architecture style, architecture pattern, or design pattern). 11 non-IoT patterns appeared in multiple papers: “publish-subscribe” [21], [22], [39], [48], [49], “client-server” [39], [48], [49], “peer-to-peer” [39], [48], “representational state transfer” (REST) [48], [49], “service-oriented architecture” (SOA) [39], [49], “role-based access control” (RBAC) [27], [30], “model-view-controller” (MVC) [35], [43], “reflection” [23], [42], “blockchain architecture style” [22], [24], “strategy” [23], [30], and “observer” [30], [40]. The other 71 non-IoT patterns appeared in one paper only.

Fourteen papers [18], [21], [23], [26], [27], [35], [37], [39], [41]–[43], [46], [48], [49] only used non-IoT patterns. These results indicate that IoT systems and software are often designed via conventional architecture and design patterns that are not specific to IoT design. This is not unexpected since IoT is constituted of traditional technical fields such as wireless sensor networks, embedded and control systems, and other supports. However, an alternative possibility is that practitioners are unaware of the existing IoT patterns. The existing IoT patterns support practitioners to plan and design their own IoT systems and software.

There are 61 IoT patterns (i.e., 43%) in 18 papers [19], [20], [22], [24], [25], [28]–[34], [36], [38], [40], [44], [45], [47] that address specific problems and solutions in IoT. The details are discussed in the subsequent section.

RQ2. Are all existing IoT architecture and design patterns really IoT patterns? Of the extracted patterns, 57% are non-IoT patterns, suggesting that IoT systems and software are often designed via conventional architecture and design patterns that are not specific to IoT design.

D. RQ3. Can IoT Architecture and Design Patterns Be Classified?

Through our SLR and while reading the documents, we noted various characteristics that could help classify patterns. We observed that IoT patterns are often presented in a context with an abstraction level, domain specificity, and quality attribute to be addressed.

1) Abstraction Level: Patterns to design IoT systems and software can be classified into two types: 1) architecture patterns and 2) design patterns. In addition, there are two different terms in the literature with respect to the nature of architecture patterns: 1) “architecture style” and 2) “architecture pattern.” Both refer to recurring solutions that solve problems at the architecture design level and provide a common vocabulary to facilitate communication [7]. The key difference is that architecture patterns address problem–solution pairs with contexts and rationales behind particular solutions, while architecture styles address the structure with constraints without explicit attention to the problem [7]. By definition, architecture styles are located at a higher abstraction level compared to architecture patterns since architecture styles have less information.

TABLE I
LIST OF EXTRACTED NON-IOT PATTERNS (AS: ARCHITECTURE STYLE, AP: ARCHITECTURE PATTERN, AND DP: DESIGN PATTERN)

Type	Pattern name	Paper
AS	Blockchain	[22], [24]
AS	Clean Architecture	[35]
AS	Microservices	[35]
AS	Data integration	[39]
AS	Event-based	[39]
AS	IaaS-based	[39]
AS	Intelligent gateway	[39]
AS	Protocol Integration via Middleware	[39]
AS	Multi-gateway	[39]
AS	PaaS-based	[39]
AS	SaaS-based	[39]
AS	Traditional gateway	[39]
AS	Distributed Multi-Gateway	[39]
AS	Web-service multiprotocol	[39]
AS	Peer-to-Peer	[39], [48]
AS	Client-Server	[39], [48], [49]
AS	Service Oriented Architecture (SOA)	[39], [49]
AS	Lambda-style architecture	[46]
AS	Kappa-style architecture	[46]
AS	Representational State Transfer (REST)	[48], [49]
AS	Microkernel	[49]
AS	Service bus	[49]
AP	Publish-Subscribe	[21], [22], [39], [48], [49]
AP	Model-View-Presenter (MVP)	[35]
AP	Model-View-ViewModel (MVVM)	[35]
AP	Model-View-Controller (MVC)	[35], [43]
AP	Pipes & Filters	[39]
AP	Layered	[49]
AP	Entity-Component-Attribute (ECA)	[32]
DP	Exception manager	[18]
DP	Input validation	[18]
DP	Secure adapter	[18]
DP	Secure directory	[18]
DP	Secure logger	[18]
DP	Trusted gateway	[22]
DP	Dependency injection	[23]
DP	Strategy	[23], [30]
DP	Reflection	[23], [42]
DP	Added noise measurement obfuscation	[26]
DP	Aggregation gateway	[26]
DP	Aggregation of data	[26]
DP	Data isolation at different entities	[26]
DP	Decoupling content and location information visibility	[26]
DP	Personal data store	[26]
DP	Single Point of Contact	[26]
DP	File authentication	[27]
DP	Reference monitor	[27]
DP	Access Matrix Role authorization rule	[27]
DP	Remote Authenticator/Authorizer	[27]
DP	Role Based Access Control (RBAC)	[27], [30]
DP	Constraint based Role Based Access Control (CRBAC)	[30]
DP	Abstract Factory	[30]
DP	Observer	[30], [40]
DP	Frame Buffer	[37]
DP	Slot Buffer	[37]
DP	Asynchronous data synchronizations	[38]
DP	Basic search	[38]
DP	Cascade selection fields	[38]
DP	Complete storage	[38]
DP	CRUD	[38]
DP	Data lookup	[38]
DP	Default selection	[38]
DP	Full transfer	[38]
DP	Location-aware search	[38]
DP	Login	[38]
DP	Master details and multi-details	[38]
DP	Mathematical transfer	[38]
DP	Multi-field form	[38]
DP	Multi-level master details	[38]
DP	Partial storage	[38]
DP	Permission and Access configuration	[38]
DP	Pre-assigned selection field	[38]
DP	Preloaded field	[38]
DP	Synchronous data synchronization	[38]
DP	Timestamp transfer	[38]
DP	User management	[38]
DP	Coordinated control	[41]
DP	Hierarchical control	[41]
DP	Information sharing	[41]
DP	Master-worker	[41]
DP	Regional planning	[41]
DP	Bridge	[42]

Thus, IoT design patterns can be classified into the following three types in terms of abstraction level.

1) *High Abstraction Level:* Architecture styles are patterns that specify architectural elements and connections at a very high abstraction level. These are often used in early phases, such as analysis and architecture design. For example, "layered architecture for IoT applications" [44] addresses a general layered IoT architecture without any concrete problem or rationale. Hence, architecture styles are regarded as highly abstract.
2) *Medium Abstraction Level:* Unlike architecture styles, medium recommends concrete architecture designs of IoT systems and software to address recurrent architectural problems such as ensuring interoperability among heterogeneous devices. These architectural elements and connections are often documented as architecture patterns that encapsulate contexts, recurring problems, and corresponding solutions. The abstraction level of architecture patterns is between high and low since architecture patterns contain more information than architecture styles (i.e., highly abstract design descriptions) while still addressing the entire software or system design rather than specific parts. On the other hand, design patterns address specific parts of the system design (i.e., low abstract design descriptions). For example, "entity-component-attribute (ECA) on linked data platform" [32] recommends a specific architecture to improve the changeability and reusability of IoT software components over different domains on the linked data platform by establishing the structural mapping from ECA to the platform as semantic Web of Things (WoT). Hence, architecture patterns are regarded as medium abstract.
3) *Low Abstraction Level:* There are recommended detailed designs to address recurrent detailed design problems such as enabling proper communications among software modules while keeping high extensibility. Since these patterns target specific modules or limited parts

and not the entire software or system, the abstraction level of the design patterns is regarded as low. These are often used in the detailed design and construction phases. For example, "pull information" [38] recommends a detailed design of the communication structure between IoT devices and gateways. Hence, design patterns are regarded as low abstract.

2) Domain Specificity: Domain specificity is important to examine the applicability and reusability of each IoT pattern. It is divided into three types: 1) non IoT; 2) general IoT; and 3) domain-specific IoT.

1) *Non-IoT Patterns:* General systems and software architecture patterns as well as design patterns that can be adopted to design IoT systems and software if the contexts and problems match the patterns' contexts and problems. There are 82 non-IoT patterns, such as MVC [35], [43] and RBAC [27], [30], which are well-accepted general architecture and design patterns.
2) *General IoT Patterns:* IoT architecture and design patterns, which are applicable to any IoT system or software. Examples include "IoT gateway event subscription" [29] and "pull information" [38] since these are originally described in the context of IoT systems and software, and not specific to a certain problem or technical domain.
3) *Domain-specific IoT Patterns:* IoT architecture and design patterns that address specific problem domains (such as healthcare) and technical domains (such as brain–computer interactions). For example, "operator-controller-module (OCM)" [19], [47] is a problem-domain-specific pattern since it addresses a specific problem and solution in the cyber–physical control domain such as operating organic Rankine cycle turbines.

3) Quality Attribute: All systems and software design patterns are expected to address one or more quality attributes. For example, IoT design patterns should mostly address interoperability, which is defined as a subattribute of compatibility in ISO/IEC 25010:2011 [6] since, by definition, IoT is about ensuring interoperability among objects. To classify IoT patterns, we use all quality attributes except for functional suitability defined in ISO/IEC 25010:2011, which is a well-accepted quality model system, and select terms from software engineering: performance, compatibility, usability, reliability, security, maintainability, and portability. We excluded functional suitability because certain functional requirements are often satisfied by concrete system and software design decisions, including reuse of IoT platforms and software libraries, instead of reuse of abstract architecture or design patterns. Without concrete functional requirements, it is difficult to determine whether a pattern contributes to functional suitability.

Additionally, there are emerging quality attributes that are not defined in ISO/IEC 25010:2011 but are common in IoT development and operation. Possible candidates are scalability and privacy.

We observed that some IoT patterns are dedicated to one or few quality attributes, while others address many attributes.

TABLE II
PATTERNS BY ABSTRACTION LEVEL AND DOMAIN SPECIFICITY (AS: ARCHITECTURE STYLE, AP: ARCHITECTURE PATTERN, AND DP: DESIGN PATTERN)

Type	Domain specificity			Total
	Non-IoT	General IoT	Domain-specific IoT	
AS	22	2	1	25
AP	7	1	15	23
DP	53	38	4	95
Total	82	41	20	143

For example, the IoT design patterns described in [38] such as "application launch" are dedicated to usability only, while "edge orchestration" [34] addresses many attributes, including reliability and maintainability.

RQ3. Can IoT architecture and design patterns be classified? Patterns for IoT systems and software can be divided along three main characteristics: 1) abstraction level; 2) domain specificity; and 3) quality attributes.

E. RQ4. What IoT Architecture and Design Patterns Exist?

Table II shows the distribution of IoT and non-IoT patterns by abstraction level and domain specificity. Table III lists the 61 IoT architecture and design patterns. Table III can be a guide for practitioners to identify available IoT patterns in terms of abstraction level, domain specificity, and quality attributes.

Surprisingly, only two patterns "OCM" [19], [47] and "computation offloading" [22], [33] are mentioned in multiple papers. The rest appear in one paper, demonstrating that IoT patterns are not shared or recognized by different research groups. This may be due to their short history. To avoid confusion, potential pattern authors should check the existing IoT patterns carefully before publishing their own "new" patterns.

In terms of abstraction level, many IoT design patterns (i.e., $42/61 = 69\%$) and some IoT architecture patterns ($16/61 = 26\%$) exist, but only a few represent IoT architecture styles ($3/61 = 5\%$).

In terms of domain specificity, 41 patterns (i.e., 67%) are general IoT, while the remaining 20 patterns (33%) are specific to a problem or technical domain (Table II).

Reviewing the combinations of abstraction level and domain specificity, most of the IoT design patterns are applicable to any domain. In contrast, many IoT architecture patterns exist for specific domains, implying that the unique nature of IoT adoption in specific domains often appears at the architecture level. Design details seem to be commonly addressed by general IoT design patterns or non-IoT design patterns. This is not surprising since IoT is constituted of traditional technical fields. In the future, the number of specific IoT design patterns may increase as more domains adopt IoT.

In terms of quality attributes, more than 80% of IoT patterns address compatibility (including interoperability as a subattribute), security, and maintainability. This finding is reasonable since major concerns in IoT adoption revolve around these attributes. As expected, most IoT architecture and design

TABLE III
LIST OF IoT PATTERNS (AS: ARCHITECTURE STYLE, AP: ARCHITECTURE PATTERN, DP: DESIGN PATTERN, PE: PERFORMANCE, C: COMPATIBILITY, U: USABILITY, R: RELIABILITY, SE: SECURITY, M: MAINTAINABILITY, PO: PORTABILITY, SC: SCALABILITY, AND PR: PRIVACY)

Type	Pattern name	Specific domain	Pe	C	U	R	Se	M	Po	Sc	Pr	Paper
AS	Layered architecture for IoT applications		X			X	X			X	X	[44]
AS	Semantic Web Thing Architecture			X				X	X			[45]
AS	Security Architecture for Smart Water Management System	Smart Water Management					X	X		X		[22]
AP	Fog computing		X	X			X	X		X		[20]
AP	Operator-Controller-Module (OCM)	Cyber-Physical Control	X					X				[19], [47]
AP	Blockchain-based Architectural Pattern for Trusted Orchestration Management	Blockchain	X			X	X	X		X		[24]
AP	Actor	Blockchain		X			X					[40]
AP	Closed-Loop: Classical Closed-Loop Control	Industrial IoT	X			X						[31]
AP	Cloud-in-the-Loop: Closed-Loop Control via the Cloud	Industrial IoT	X			X						[31]
AP	Cloud-on-the-Loop: Cloud-configured Control	Industrial IoT	X			X						[31]
AP	Device-to-Device (D2D): Local Coordination	Industrial IoT	X			X						[31]
AP	Open-Loop: Classical Open-Loop Control	Industrial IoT	X			X						[31]
AP	Publisher: Sensor Data Publication	Industrial IoT		X				X				[31]
AP	Entity-Component-Attribute on Linked Data Platform	Semantic Web of Things		X				X	X			[32]
AP	Landline Interception	Emergency Information Delivery			X							[36]
AP	SIM Equipped Device	Emergency Information Delivery			X							[36]
AP	SMS to Display over Bluetooth/Wi-Fi	Emergency Information Delivery			X							[36]
AP	SMS to Mobile Application	Emergency Information Delivery			X							[36]
AP	Web System (for Emergency Information Delivery)	Emergency Information Delivery			X							[36]
DP	Computation offloading		X					X	X			[22], [33]
DP	Stateless authentication						X					[22]
DP	Alignment-based Translation			X				X				[29]
DP	AS2AS Discovery of IoT Services			X				X				[29]
DP	AS2AS Flow-based Service Composition			X				X				[29]
DP	AS2AS Service Orchestration			X				X				[29]
DP	D2D REST Request/Response			X				X				[29]
DP	IoT Artifact's Middleware Message Broker						X	X				[29]
DP	IoT Artifact's Middleware Message Translator			X				X				[29]
DP	IoT Artifact's Middleware Self-contained Message							X				[29]
DP	IoT Artifact's Middleware Simple Component					X		X				[29]
DP	IoT Gateway Event Subscription			X				X				[29]
DP	IoT Pattern for Orchestration of SDN Network Elements			X				X				[29] [29]
DP	IoT SSL CROSS-Layer Secure Access			X			X	X				[29]
DP	Translation with Central Ontology			X				X		X		[29]
DP	Sensor Design Pattern (SDP)						X	X				[30]
DP	Edge Code Deployment							X	X			[34]
DP	Edge Diameter of Things (DOT)							X				[34]
DP	Edge Orchestration					X		X	X	X		[34]
DP	Edge Provisioning					X		X				[34]
DP	Application launch				X							[38]
DP	Get details of a device				X							[38]
DP	Get Information for one category				X							[38]
DP	Get information from the device				X							[38]
DP	Get state of the device				X							[38]
DP	More devices more operations				X							[38]
DP	More devices one operation				X							[38]
DP	Nearby devices				X							[38]
DP	One category more operations				X							[38]
DP	One device more operations				X							[38]
DP	One device one operation				X							[38]
DP	One device one program				X							[38]
DP	Pull information				X							[38]
DP	Push information				X							[38]
DP	Search device				X							[38]
DP	Event interaction							X				[45]
DP	Action interaction							X				[45]
DP	Property interaction							X				[45]
DP	Ontology Design Pattern for IoT Device Tagging Systems	Building Automation, Ontology		X				X				[25]
DP	Actuation-Actuator-Effect (AAE) ontology design	Brain-Computer Interaction, Ontology		X				X				[28]
DP	Participation foundational design	Brain-Computer Interaction, Ontology		X				X				[28]
DP	Stimulus-Sensor-Observation (SSO) ontology design	Brain-Computer Interaction, Ontology		X				X				[28]
Number of patterns that address the corresponding quality attribute			21	57	20	18	51	54	11	28	12	

patterns address interoperability. On the other hand, connectivity is at the core of IoT, and every communication channel needs to be secured against attacks. Moreover, the number and heterogeneity of objects can increase the attack surface. Hence, IoT security patterns have been required and published to mitigate vulnerabilities. Maintainability is well addressed in IoT patterns since IoT systems and software often address various devices and their different lifecycles.

TABLE IV
OVERVIEW OF PAPERS MENTIONING IoT PATTERNS (**REFERENCES** INDICATING PAPERS OF DOMAIN-SPECIFIC IoT PATTERNS)

Type	Influence on quality in use for primary users	Influence on quality in use for maintenance tasks
AS	[44] **[22]**	[44], [45] **[22]**
AP	[20] **[19], [24], [31], [36], [40], [47]**	[20] **[19], [24], [31], [32], [40], [47]**
DP	[22], [29], [30], [33], [34], [38]	[22], [29], [30], [33], [34], [45] **[25], [28]**

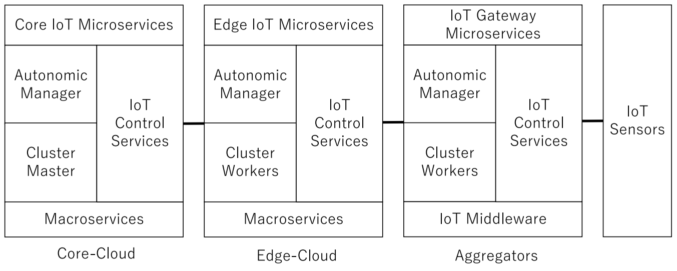


Fig. 3. Layered architecture style for IoT applications (adapted from [44]).

In addition, some IoT patterns address performance, usability, reliability, and scalability. We observed that these attributes are also important to address in IoT systems and software.

Consequently, other quality attributes are less researched. For example, only a few IoT patterns address portability and privacy. In the future, IoT patterns addressing these attributes are anticipated by accumulating more design cases focusing on these attributes since they are also important.

As an additional guide for practitioners to find papers about IoT patterns, Table IV shows the distribution of papers containing IoT patterns by abstraction level, domain specificity, and influence of the quality attributes addressed by the patterns. According to ISO/IEC 25010:2011 [6], performance, usability, reliability, and security significantly influence the quality in use for primary users, while compatibility, maintainability, and portability greatly impact quality in use for secondary users who maintain the system. We classify these additional quality attributes as privacy and scalability. The former is an important concern of primary users, while the latter is about ease of extending a system by maintainers in terms of performance.

In Table IV, most papers use IoT patterns to address quality attributes that influence the quality in use for primary users as well as those that influence the quality in use for maintenance tasks. Because this implies that IoT systems should be designed with good quality for both primary users and maintainers, the identified IoT patterns should help support architecting and design.

RQ4. What IoT architecture and design patterns exist? IoT architecture patterns and design patterns exist. Many IoT patterns address compatibility (including interoperability as a subattribute), security, and maintainability. Most IoT design patterns are applicable to any domain. On the other hand, many IoT architecture patterns are domain specific, implying that the unique nature of IoT adoption in specific domains appears at the architecture level.

IV. DISCUSSION

We describe extracted IoT patterns, possible use cases, and threats to validate our results.

A. Examples of IoT Pattern

Here, we describe three extracted IoT patterns having different abstraction levels: 1) "layered architecture for IoT applications" [44] as an example of IoT architecture style; 2) "ECA on linked data platform (ECA2LD)" [32] as an example of IoT architecture pattern; and 3) "IoT gateway event subscription" [29] as an example of IoT design pattern. For brevity, participants, collaborations, implementation, and known uses are omitted. The intent section of each pattern can be a guide for practitioners to understand and consider reusing the corresponding content.

1) Example of IoT Architecture Style:

a) Pattern name: Layered architecture for IoT applications [44].

b) Intent: Support the construction of hierarchical, programmable, and autonomic IoT applications.

c) Solution: The IoT platform providing resource virtualization using lightweight virtualization (i.e., containerization) for multilayer applications (Fig. 3).

d) Consequences: By implementing the respective requirements of an IoT application to the appropriate layer of the three layers shown in Fig. 3, nonfunctional properties, such as performance, security and privacy, reliability, elasticity, and scalability can be treated flexibly with service orchestration through the layers.

2) Example of IoT Architecture Pattern:

a) Pattern name: ECA2LD [32].

b) Intent: Support the design of changeable and maintainable software components for large-scale IoT applications.

c) Context: The Web is considered as an IoT convergence platform to realize WoT. Software built for IoT environments must be adaptable to changes and interoperable with others on the platform.

d) Problem: ECA-based software design (Fig. 4) is particularly well suited to improve the changeability and reusability of IoT software components. However, seamless cross-domain interoperability between independently developed IoT applications and platforms is not directly addressed.

e) Solution: It establishes a structural mapping from ECA to the Linked Data Platform so that the interoperability of independently developed IoT applications can be seamless over different domains on the platform.

f) Consequences: This data-oriented approach should significantly improve the changeability of entities and reuse of IoT software components. Mapping the entire architecture makes it easy to implement large-scale IoT applications to Semantic WoT.

3) Example of IoT Design Pattern:

a) Pattern name: IoT gateway event subscription [29].

b) Intent: Provide interoperability between two heterogeneous IoT devices, while simultaneously ensuring that the IoT gateway has flexibility.

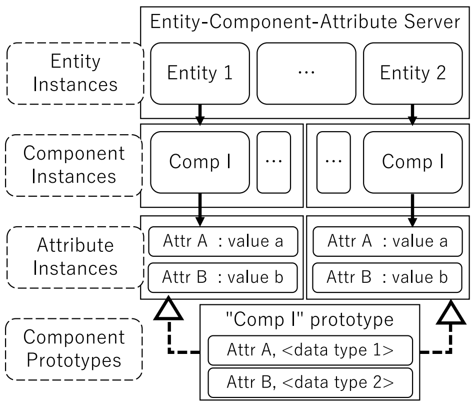


Fig. 4. ECA (adapted from [32]).

c) Context: This pattern is used within event-based communication when the data are pushed (pulled) to (from) the IoT gateway asynchronously. The IoT gateway allows for data forwarding.

d) Problem: Interoperability between two heterogeneous IoT devices requires bidirectional, asynchronous communication with the ability to publish, filter, and consume data.

e) Solution: Employ a subscription mechanism into the IoT gateway, which allows asynchronous and mutual transmissions of data obtained by sensors at the destination and the message between artifacts. Transmitters of messages (i.e., publishers) can publish messages using defined classes without knowledge of subscribers. Meanwhile, subscribers can express interest in one or more classes, and receive messages of interest without knowledge of publishers. The IoT gateway works flexibly in two parts. The physical part deals with network access and communication protocols, while the virtual part deals with the remaining gateway operations and services. The former is platform specific. It depends on the network communication protocols and devices deployed in physical space. In contrast, the latter is platform-independent.

f) Consequences: Encouraging asynchronous messaging improves the compatibility of IoT applications using heterogeneous IoT devices. Enhancing the loose coupling between publishers and subscribers improves the maintainability of IoT applications. In addition, decoupling the IoT gateway into two parts realizes flexibility in the device-to-device layer.

g) Related patterns: "D2D REST request/response" pattern [29], and "publish-subscribe" pattern [21], [22], [39], [48], [49].[3]

B. Use Cases

The results of our SLR are expected to guide practitioners and researchers in the following possible use cases UC1–UC3.

UC1 (To Publish New IoT Patterns): When practitioners (and researchers) want to write and publish their new IoT architecture and design patterns, they can be aware of the existing IoT patterns by referring to our classification results. The characteristics of IoT patterns identified in our SLR can also support writing new patterns to consider appropriate abstraction levels, domain specificity, and quality attributes to be addressed. In addition, the practitioners and researchers eventually extend the existing IoT architecture and design patterns.

UC2 (To Resolve IoT Design Problems): When practitioners and researchers want to resolve problems in the IoT design, our classification results and the characteristics of IoT patterns help them to compare the existing IoT patterns, and then select and reuse the appropriate one according to their objectives. Developers can utilize our classification scheme and results in different development phases.

1) To consider the appropriate high-level IoT system architecture in the analysis phase as well as the early architecting phase in IoT system development projects, developers can first review non-IoT or IoT architecture styles of the given projects by examining the relevance between the contexts, architectural elements, and their connections.
2) To design concrete architectures of the target IoT systems and software in the architecting phase, developers can also consider reusing (non-) IoT architecture patterns by examining the relevance between projects' specific requirements, contexts, and problems of the architecture patterns.
3) To design limited parts of the target IoT systems and software in the design phase, developers can consider reusing (non-) IoT design patterns by examining relevance between specific detailed design problems and contexts of the design patterns.

UC3 (To communicate and Research IoT Patterns): Our classification results and the characteristics of IoT patterns can serve as a reference for the IoT pattern engineering community, including practitioners and researchers. Our results can be extended by peers, providing the community with an important body of knowledge to guide future communications and research on IoT patterns.

C. Threats to Validity

As an empirical study, the results of SLRs are vulnerable to internal validity and reliability [50]. Internal validity arises from the cause–effect conclusion drawn from the SLR process and its results. To alleviate this, we used the data to answer each research question.

Reliability concerns arise from the quality and rigor that the SLR was conducted. To demonstrate a sound process, Section III explains the steps in our SLR and reports the number of papers in each step. In addition, all of our data are available online.[4]

Another threat to reliability is that an independent third party has yet to vet all the identified patterns. The Pattern Languages of Programs (PLoP) conference series,[5] such as PLoP[6] and AsianPLoP,[7] which is sponsored by the Hillside

[3]"IoT gateway event subscription" can be regarded as a general IoT design pattern since it is originally described as a slight extension of "publish-subscribe" in the context of IoT [29].

[4]http://www.washi.cs.waseda.ac.jp/iot-patterns/

[5]https://hillside.net/conferences/

[6]https://www.hillside.net/plop/

[7]http://asianplop.org

Group, focuses on pattern writing groups to improve patterns through group exposure. We intend to participate in the conference series to receive community feedback about each pattern prior to publication.

Most authors extracted and classified patterns. All patterns were independently vetted by another author. Although our rigorous SLR noted the characteristics of IoT patterns, other characteristics to be used for the classification of IoT patterns may be omitted. It is possible that our classification results are not completely correct. To analyze the extent of this threat to reliability in terms of pattern extraction and classification, we asked two uninvolved researchers (R_1 and R_2) to extract and classify patterns from [32] and [44] studied in our SLR. We selected these papers [32], [44] since they do not describe patterns in any explicit structured pattern format. Thus, there was a possibility that different examiners may extract different patterns or classify them differently. From [44], R_1 extracted the same architecture style as our result (i.e., layered architecture for IoT applications) while R_2 extracted a similar but more concrete architecture pattern. In terms of quality attributes, R_2 commonly identified performance, reliability, security, scalability, and privacy, which are also identified by us in Table III. In contrast, R_1 identified compatibility, maintainability, portability, and scalability; these attributes except for scalability are different from our result. From [32], R_1 and R_2 extracted the same or similar patterns as our result (i.e., ECA2LD), but classified them as design patterns unlike our classification. In terms of quality attributes, R_2 identified compatibility, maintainability, and portability, which are also identified by us in Table III. R_1 also identified compatibility and portability but missed maintainability. Based on these independent analysis results, we believe that our pattern extraction results can be generally consistent. However, our classification process can be somewhat inconsistent resulting in partially different classification results by different examiners. To mitigate this threat, we have shared our classification results with the public to call for comments on our website in the future.

We used Scopus as the initial document base of the SLR. Although many other SLRs, such as [11] and [51]–[53] have adopted it, relevant papers (such as IoT security pattern papers [54]) may have been missed. To mitigate this threat, we plan to use other databases, extend our SLR, and elicit public review of the results.

V. Conclusion

To overview the current landscape of IoT architecture and design patterns, we conducted an SLR of the academic literature and identified the 143 patterns mentioned in 32 papers. Of the extracted patterns, 57% are non-IoT patterns, suggesting that IoT systems and software are often designed via conventional architecture and design patterns that are not specific to the IoT design. Although most IoT design patterns are applicable to any domain, IoT architecture patterns tend to be for specific domains, implying that the unique nature of IoT adoption in specific domains appears at the architecture level. In the future, the number of domain-specific IoT design patterns may increase as more domains adopt IoT. In terms of quality attributes, many IoT patterns address compatibility, security, and maintainability. Consequently, other quality attributes have yet to be investigated.

Our future work includes further analysis of IoT patterns using additional characteristics, such as the relationships among patterns and writing quality of patterns (as discussed in [55] for security patterns). We also plan to increase our survey scope to include gray literature.

We plan to share the revised survey and analysis results to obtain reviews from the public. We expect that the research community will further validate the SLR results from the viewpoints of practitioners and researchers. Public input should extend the classification to include new characteristics and data sets.

Acknowledgment

The authors would like to thank Dr. Takehisa Kato and Prof. Haruhiko Kaiya for their initial pattern analysis, and Dr. Takafumi Tanaka and Hideyuki Kanuka for their additional pattern analysis. They also would like to thank the anonymous reviewers for their insightful comments and suggestions.

References

[1] M. Aly, F. Khomh, Y. Guéhéneuc, H. Washizaki, and S. Yacout, "Is fragmentation a threat to the success of the Internet of Things?" *IEEE Internet Things J.*, vol. 6, no. 1, pp. 472–487, Feb. 2019. [Online]. Available: https://doi.org/10.1109/JIOT.2018.2863180

[2] M. Aly, F. Khomh, M. Haoues, A. Quintero, and S. Yacout, "Enforcing security in Internet of Things frameworks: A systematic literature review," *Internet Things*, vol. 6, Jun. 2019, Art. no. 100050. [Online]. Available: https://doi.org/10.1016/j.iot.2019.100050

[3] J. Höller, V. Tsiatsis, and C. Mulligan, "Toward a machine intelligence layer for diverse industrial IoT use cases," *IEEE Intell. Syst.*, vol. 32, no. 4, pp. 64–71, Aug. 2017. [Online]. Available: https://doi.org/10.1109/MIS.2017.3121543

[4] "Software engineering—Guide to the software engineering body of knowledge (SWEBOK)," ISO/IEC, Geneva, Switzerland, Rep. TR 19759:2015, 2015.

[5] H. Muccini and M. T. Moghaddam, "IoT architectural styles—A systematic mapping study," in *Proc. 12th Eur. Conf. Softw. Archit. (ECSA)*, Madrid, Spain, Sep. 2018, pp. 68–85. [Online]. Available: https://doi.org/10.1007/978-3-030-00761-4_5

[6] "Systems and software engineering—Systems and software quality requirements and evaluation (SQuaRE)—System and software quality models," ISO/IEC, Geneva, Switzerland, Rep. 25010:2011, 2011.

[7] P. Avgeriou and U. Zdun, "Architectural patterns revisited—A pattern language," in *Proc. 10th Eur. Conf. Pattern Lang. Programs (EuroPLoP)*, Irsee, Germany, Jul. 2005, pp. 431–470.

[8] A. Ampatzoglou, S. Charalampidou, and I. Stamelos, "Research state of the art on GoF design patterns: A mapping study," *J. Syst. Softw.*, vol. 86, no. 7, pp. 1945–1964, 2013. [Online]. Available: https://doi.org/10.1016/j.jss.2013.03.063

[9] B. B. Mayvan, A. Rasoolzadegan, and Z. G. Yazdi, "The state of the art on design patterns: A systematic mapping of the literature," *J. Syst. Softw.*, vol. 125, pp. 93–118, Mar. 2017. [Online]. Available: https://doi.org/10.1016/j.jss.2016.11.030

[10] J. Juziuk, D. Weyns, and T. Holvoet, "Design patterns for multi-agent systems: A systematic literature review," in *Agent-Oriented Software Engineering—Reflections on Architectures, Methodologies, Languages, and Frameworks*. Heidelberg, Germany: Springer, 2014, pp. 79–99. [Online]. Available: https://doi.org/10.1007/978-3-642-54432-3_5

[11] H. Washizaki, H. Uchida, F. Khomh, and Y.-G. Gueheneuc, "Studying software engineering patterns for designing machine learning systems," in *Proc. 10th Int. Workshop Empirical Softw. Eng. Pract. (IWESEP)*, Tokyo, Japan, 2019, pp. 1–6.

[12] P. Ponde and S. Shirwaikar, "An exploratory study of the security design pattern landscape and their classification," *Int. J. Syst. Syst. Eng.*, vol. 7, no. 3, pp. 26–43, 2016. [Online]. Available: https://doi.org/10.4018/IJSSE.2016070102

[13] H. Ahmadi, G. Arji, L. Shahmoradi, R. Safdari, M. Nilashi, and M. Alizadeh, "The application of Internet of Things in healthcare: A systematic literature review and classification," *Univ. Access Inf. Soc.*, vol. 18, pp. 1–33, May 2018.

[14] P. Asghari, A. M. Rahmani, and H. H. S. Javadi, "Internet of Things applications: A systematic review," *Comput. Netw.*, vol. 148, pp. 241–261, Jan. 2019.

[15] M. D. Giudice, "Discovering the Internet of Things (IoT) within the business process management: A literature review on technological revitalization," *Bus. Process Manag. J.*, vol. 22, no. 2, pp. 1–9, 2016.

[16] P. P. Ray, "A survey of IoT cloud platforms," *Future Comput. Informat. J.*, vol. 1, no. 1, pp. 35–46, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2314728816300149

[17] H. Washizaki *et al.*, "Landscape of IoT patterns," in *Proc. 1st Int. Workshop Softw. Eng. Res. Pract. Internet Things, (SERP4IoT@ICSE)*, Montreal, QC, Canada, May 2019, pp. 57–60. [Online]. Available: https://dl.acm.org/citation.cfm?id=3354013

[18] W. Lee and P. Law, "A case study in applying security design patterns for IoT software system," in *Proc. Int. Conf. Appl. Syst. Innov. (ICASI)*, May 2017, pp. 1162–1165.

[19] C. Wolff, M. Knirr, K. Priebe, P. Schulz, and J. Strumberg, "A layered software architecture for a flexible and smart organic rankine cycle (ORC) turbine–solutions and case study," *Inf. Technol. Control*, vol. 47, no. 2, pp. 349–362, 2018. [Online]. Available: https://doi.org/10.5755/j01.itc.47.2.19681

[20] M. H. Syed, E. B. FernÁndez, and M. Ilyas, "A pattern for fog computing," in *Proc. 10th Travelling Conf. Pattern Lang. Programs VikingPLoP*, Leerdam, The Netherlands, Apr. 2016, pp. 1–10. [Online]. Available: https://doi.org/10.1145/3022636.3022649

[21] L. Roffia *et al.*, "A semantic publish-subscribe architecture for the Internet of Things," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1274–1296, Dec. 2016. [Online]. Available: https://doi.org/10.1109/JIOT.2016.2587380

[22] N. Ntuli and A. M. Abu-Mahfouz, "A simple security architecture for smart water management system," in *Proc. 7th Int. Conf. Ambient Syst. Netw. Technol. (ANT) 6th Int. Conf. Sustain. Energy Inf. Technol. (SEIT) Affiliated Workshops*, Madrid, Spain, May 2016, pp. 1164–1169. [Online]. Available: https://doi.org/10.1016/j.procs.2016.04.239

[23] E. Jung, I. Cho, and S. M. Kang, "An agent modeling for overcoming the heterogeneity in the IoT with design patterns," in *Proc. Mobile Ubiquitous Intell. Comput. (MUSIC) FTRA 4th Int. Conf. Mobile Ubiquitous Intell. Comput.*, Gwangju, South Korea, Sep. 2013, pp. 69–74. [Online]. Available: https://doi.org/10.1007/978-3-642-40675-1_11

[24] C. Pahl, N. E. Ioini, S. Helmer, and B. Lee, "An architecture pattern for trusted orchestration in IoT edge clouds," in *Proc. 3rd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Barcelona, Spain, Apr. 2018, pp. 63–70. [Online]. Available: https://doi.org/10.1109/FMEC.2018.8364046

[25] V. Charpenay, S. Käbisch, D. Anicic, and H. Kosch, "An ontology design pattern for IoT device tagging systems," in *Proc. 5th Int. Conf. Internet Things (IoT)*, Seoul, South Korea, Oct. 2015, pp. 138–145. [Online]. Available: https://doi.org/10.1109/IOT.2015.7356558

[26] S. Pape and K. Rannenberg, "Applying privacy patterns to the Internet of Things' (IoT) architecture," *Mobile Netw. Appl.*, vol. 24, no. 3, pp. 925–933, 2019. [Online]. Available: https://doi.org/10.1007/s11036-018-1148-2

[27] I. Ali and M. Asif, "Applying security patterns for authorization of users in IoT based applications," in *Proc. Int. Conf. Eng. Emerg. Technol. (ICEET)*, Feb. 2018, pp. 1–5.

[28] S. J. R. Méndez and J. K. Zao, "BCI ontology: A context-based sense and actuation model for brain-computer interactions," in *Proc. 9th Int. Semantic Sensor Netw. (SSN) Workshop Colocated 17th Int. Semantic Web Conf. (ISWC)*, Monterey, CA, USA, Oct. 2018, pp. 32–47. [Online]. Available: http://ceur-ws.org/Vol-2213/paper3.pdf

[29] R. Tkaczyk *et al.*, "Cataloging design patterns for Internet of Things artifact integration," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, Kansas City, MO, USA, May 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICCW.2018.8403758

[30] K. Periyasamy, V. S. Alagar, and K. Wan, "Dependable design for elderly health care," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, Sep. 2017, pp. 803–806. [Online]. Available: https://doi.org/10.15439/2017F261

[31] G. Bloom, B. Alsulami, E. Nwafor, and I. C. Bertolotti, "Design patterns for the industrial Internet of Things," in *Proc. 14th IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, Jun. 2018, pp. 1–10. [Online]. Available: https://doi.org/10.1109/WFCS.2018.8402353

[32] T. Spieldenner, R. Schubotz, and M. Guldner, "ECA2LD: From entity-component-attribute runtimes to linked data applications," in *Proc. Int. Workshop Semantic Web Things Ind. Extend. Semantic Web Conf. (ESWC)*, Jun. 2018, pp. 1–12.

[33] S. Chen, B. Liu, X. Chen, Y. Zhang, and G. Huang, "Framework for adaptive computation offloading in IoT applications," in *Proc. 9th Asia–Pac. Symp. Internetware*, Shanghai, China, Sep. 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1145/3131704.3131717

[34] S. Qanbari *et al.*, "IoT design patterns: Computational constructs to design, build and engineer edge applications," in *Proc. 1st IEEE Int. Conf. Internet Things Design Implement. (IoTDI)*, Berlin, Germany, Apr. 2016, pp. 277–282. [Online]. Available: https://doi.org/10.1109/IoTDI.2015.18

[35] M. P. Shopov, "IoT gateway for smart metering in electrical power systems–software architecture," in *Proc. 40th Int. Convention Inf. Commun. Technol. Electron. Microelectron. (MIPRO)*, May 2017, pp. 974–978. [Online]. Available: https://doi.org/10.23919/MIPRO.2017.7973565

[36] A. Q. Gill, N. Phennel, D. Lane, and V. L. Phung, "IoT-enabled emergency information supply chain architecture for elderly people: The australian context," *Inf. Syst.*, vol. 58, pp. 75–86, Jun. 2016. [Online]. Available: https://doi.org/10.1016/j.is.2016.02.004

[37] S. Vorapojpisut, "Model-based design of IoT/WSN nodes: Device driver implementation," in *Proc. Int. Conf. Embedded Syst. Intell. Technol. Int. Conf. Inf. Commun. Technol. Embedded Syst. (ICESIT-ICICTES)*, May 2018, pp. 1–5.

[38] M. Brambilla, E. Umuhoza, and R. Acerbis, "Model-driven development of user interfaces for IoT systems via domain-specific components and patterns," *J. Internet Services Appl.*, vol. 8, no. 1, pp. 1–21, 2017. [Online]. Available: https://doi.org/10.1186/s13174-017-0064-1

[39] B. Tekinerdogan and Ö. Köksal, "Pattern based integration of Internet of Things systems," in *Proc. 3rd Int. Conf. Internet Things (ICIOT) Services Conf. Federat. (SCF)*, Seattle, WA, USA, Jun. 2018, pp. 19–33. [Online]. Available: https://doi.org/10.1007/978-3-319-94370-1_2

[40] M. A. Walker, A. Dubey, A. Laszka, and D. C. Schmidt, "PlaTIBART: A platform for transactive IoT blockchain applications with repeatable testing," in *Proc. 4th Workshop Middleware Appl. Internet Things (M4IoT@Middleware)*, Las Vegas, NV, USA, Dec. 2017, pp. 17–22. [Online]. Available: https://doi.org/10.1145/3152141.3152392

[41] V. Cardellini, T. G. Grbac, M. Nardelli, N. Tankovic, and H. L. Truong, "QoS-based elasticity for service chains in distributed edge cloud environments," in *Autonomous Control for a Reliable Internet of Services—Methods, Models, Approaches, Techniques, Algorithms, and Tools*. Cham, Switzerland: Springer, 2018, pp. 182–211. [Online]. Available: https://doi.org/10.1007/978-3-319-90415-3_8

[42] M. Mongiello, G. Boggia, and E. D. Sciascio, "ReIOS: Reflective architecting in the Internet of objects," in *Proc. 4rd Int. Conf. Model Driven Eng. Softw. Develop. (MODELSWARD)*, Rome, Italy, Feb. 2016, pp. 384–389. [Online]. Available: https://doi.org/10.5220/0005800603840389

[43] M. A. Al-Taee, W. Al-Nuaimy, Z. J. Muhsin, and A. Al-Ataby, "Robot assistant in management of diabetes in children based on the Internet of Things," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 437–445, Apr. 2017. [Online]. Available: https://doi.org/10.1109/JIOT.2016.2623767

[44] H. Khazaei, H. Bannazadeh, and A. Leon-Garcia, "SAVI-IoT: A self-managing containerized IoT platform," in *Proc. 5th IEEE Int. Conf. Future Internet Things Cloud (FiCloud)*, Prague, Czech Republic, Aug. 2017, pp. 227–234. [Online]. Available: https://doi.org/10.1109/FiCloud.2017.27

[45] A. Mazayev, J. A. Martins, and N. Correia, "Semantic Web thing architecture," in *Proc. 4th Exp. Int. Conf.*, Faro, Portugal, Jun. 2017, pp. 43–46. [Online]. Available: https://doi.org/10.1109/EXPAT.2017.7984368

[46] A. Auger, E. Exposito, and E. Lochin, "Sensor observation streams within cloud-based IoT platforms: Challenges and directions," in *Proc. 20th Conf. Innov. Clouds Internet Netw. (ICIN)*, Paris, France, Mar. 2017, pp. 177–184. [Online]. Available: https://doi.org/10.1109/ICIN.2017.7899407

[47] C. Wolff *et al.*, "Software architecture for an ORC turbine—Case study for an intelligent technical system in the era of the Internet of Things," in *Proc. 23rd Int. Conf. Inf. Softw. Technol. , (ICIST)*, Druskininkai, Lithuania, Oct. 2017, pp. 226–237. [Online]. Available: https://doi.org/10.1007/978-3-319-67642-5_19

[48] P. M. Jacob and P. Mani, "Software architecture pattern selection model for Internet of Things based systems," *IET Softw.*, vol. 12, no. 5, pp. 390–396, 2018. [Online]. Available: https://doi.org/10.1049/iet-sen.2017.0206

[49] V. Taratukhin, Y. Yadgarova, and J. Becker, “The Internet of Things prototyping platform under the design thinking methodology,” in *Proc. 125th ASEE Annu. Conf. Expo. Amer. Soc. Eng. Educ.*, 2018, pp. 1–9.

[50] X. Zhou, Y. Jin, H. Zhang, S. Li, and X. Huang, “A map of threats to validity of systematic literature reviews in software engineering,” in *Proc. 23rd Asia–Pac. Softw. Eng. Conf.*, Dec. 2016, pp. 153–160.

[51] H. Washizaki *et al.*, “Taxonomy and literature survey of security pattern research,” in *Proc. IEEE Conf. Appl. Inf. Netw. Security (AINS)*, Nov. 2018, pp. 87–92.

[52] A. Dadwal, H. Washizaki, Y. Fukazawa, T. Iida, M. Mizoguchi, and K. Yoshimura, “Prioritization in automotive software testing: Systematic literature review,” in *Proc. 6th Int. Workshop Quantitative Approaches Softw. Quality Colocated 25th Asia–Pac. Softw. Eng. Conf. (APSEC)*, Nara, Japan, Dec. 2018, pp. 52–58. [Online]. Available: http://ceur-ws.org/Vol-2273/QuASoQ-07.pdf

[53] A. B. Marques, R. Rodrigues, and T. Conte, “Systematic literature reviews in distributed software development: A tertiary study,” in *Proc. IEEE 7th Int. Conf. Global Softw. Eng.*, Aug. 2012, pp. 134–143.

[54] E. B. FernÁndez, N. Yoshioka, and H. Washizaki, “Abstract and IoT security segmentation patterns,” in *Proc. 8th Asian Conf. Pattern Lang. Programs (AsianPLoP)*, 2019, pp. 1–9.

[55] T. Heyman, K. Yskout, R. Scandariato, and W. Joosen, “An analysis of the security patterns landscape,” in *Proc. 3rd Int. Workshop Softw. Eng. Secure Syst. (SESS)*, Minneapolis, MN, USA, May 2007, p. 3. [Online]. Available: https://doi.org/10.1109/SESS.2007.4

Hironori Washizaki (Member, IEEE) received the Doctoral degree in information and computer science from Waseda University, Tokyo, Japan, in 2003.

He is a Professor and the Associate Dean of the Research Promotion Division, Waseda University, and a Visiting Professor with the National Institute of Informatics. He also works in industry as Outside Director of System Information and eXmotion. He has led many academia–industry joint research and large-funded projects in software analysis and quality assurance. Since 2017, he has been the lead on a large-scale grant at MEXT, called enPiT-Pro SmartSE, which encompasses professional education in IoT, AI, software engineering, and business. Since 2015, he has been the Convener of ISO/IEC/JTC1 SC7/WG20 to standardize bodies of knowledge and professional certifications. He has published more than 120 research papers in refereed international journals and conferences, including IoT-J, TETC, EMSE, SCICO, ICSE, and ASE. His research interests include systems and software engineering.

Dr. Washizaki has received various awards and honors, including the IWESEP Best Paper Award and the IJSEKE Most Read Article. He has served as the Program Chair of multiple IEEE conferences, including ICST, CSEE&T, and SIoT/SISA of COMPSAC. He is the Program Chair of ICPC Programming Education Track and SCAM Engineering Track, the Workshop Chair and the Publicity Chair of ASE, a Local Chair of COMPSAC, and the Chair of IEEE CS Japan Chapter. He serves as the Chair of the IEEE Computer Society Professional and Educational Activities Board Engineering Discipline Committee. He is spearheading the Guide to the Software Engineering Body of Knowledge (SWEBOK) evolution. He serves as an Associate Editor for the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, a Steering Committee Member of the IEEE Conference on Software Engineering Education and Training, and an Advisory Committee Member of the IEEE CS flagship conference COMPSAC. He is a Professional Member of IEEE-Eta Kappa Nu. Since 2019, he has been a Steering Committee Member of APSEC.

Shinpei Ogata (Member, IEEE) received the M.E. degree in electrical engineering and computer science and the Ph.D. degree in functional control systems from the Shibaura Institute of Technology, Tokyo, Japan, in 2009 and 2012, respectively.

He is an Associate Professor with Shinshu University, Nagano, Japan. His current research interests include model-driven engineering for information system development.

Dr. Ogata is a member of ACM, IEICE, IPSJ, and JSSST.

Atsuo Hazeyama (Member, IEEE) received the Doctoral degree in information engineering from Shinshu University, Nagano, Japan, in 1999.

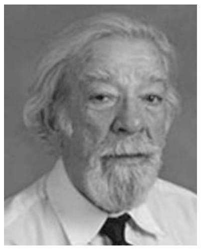
He is a Professor with the Department of Information Science, Tokyo Gakugei University, Tokyo, Japan. His research interests are support of secure software development, collaborative software development, and project-based learning for software development.

Dr. Hazeyama has served as a Program Committee Member for some international conferences, including the International Conference on Software Engineering Education and Training, Asia–Pacific Software Engineering Conference, and Knowledge Based and Intelligent Information and Engineering Systems.

Takao Okubo (Member, IEEE) received the M.S. degree in engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 1991, and the Ph.D. degree in informatics from the Institute of Information Security, Kanagawa, Japan, in 2009.

He is a Professor with the Institute of Information Security. From 1991 to 2013, he worked as a Researcher in software engineering and software security with Fujitsu Laboratories. In 2013, he moved to the Institute of Information Security as an Associate Professor. His current interests are secure development and threat analysis.

Dr. Okubo is a member of IEICE, IPSJ, ACM, and IEEE CS.



Eduardo B. Fernandez (Eduardo Fernandez Buglioni) (Senior Member, IEEE) received the B.S. degree in electrical engineering from Universidad Técnica Federico Santa Maria, Valparaíso, Chile, the M.S. degree in electrical engineering from Purdue University, Lafayette, IN, USA, in 1963, and the Ph.D. degree in computer science from the University of California Los Angeles (UCLA), Los Angeles, CA, USA, in 1972.

He is a Professor with the Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL, USA. He is an active consultant for industry, including assignments with IBM, Allied Signal, Panasonic, Motorola, Lucent, and Huawei. He has published numerous papers as well as several books on computer security and software architecture, and numerous papers on authorization models, object-oriented analysis and design, cloud computing, and security patterns. He has written four books on these subjects, the most recent being a book on security patterns.

Nobukazu Yoshioka (Member, IEEE) received the B.E. degree in electronic and information engineering from Toyama University, Toyama, Japan, in 1993, and the M.E. and Ph.D. degrees in information science from the School of Information Science, Japan Advanced Institute of Science and Technology, Nomi, Japan, in 1995 and 1998, respectively.

He is a Researcher with the National Institute of Informatics, Tokyo, Japan. From 1998 to 2002, he was with Toshiba Corporation, Tokyo. From 2002 to 2004, he was a Researcher, and since August 2004, he has been an Associate Professor with the National Institute of Informatics. His research interests include Security and privacy software engineering and software engineering for machine learning-based systems.

Dr. Yoshioka was a Board Member of JSSST from 2011 to 2015, and has been the Auditor since 2018. He was the Chair of the IEEE CS Japan Chapter from 2015 to 2017.