Software Architectures of the Convergence of Cloud Computing and the Internet of Things: A Systematic Literature Review

Ahmad Banijamali*, Olli-Pekka Pakanen, Pasi Kuvaja, Markku Oivo

Empirical Software Engineering in Software, Systems and Services (M3S), Faculty of Information Technology and Electrical Engineering (ITEE), University of Oulu, Finland

Abstract

Context: Over the last few years, there has been an increasing interest in the convergence of cloud computing and the Internet of Things (IoT). Although software systems in this domain have attracted researchers to develop a large body of knowledge on software architecture designs, there is no systematic analysis of this knowledge.
Objective: This study aims to identify and synthesise state-of-the-art architectural elements including the design patterns, styles, views, quality attributes, and evaluation methodologies in the convergence of cloud computing and IoT.
Method: We used systematic literature review (SLR) methodology for a detailed analysis of 82 primary studies of a total of 1,618 studies.
Results: We extracted six architectural design patterns in this domain; among them, edge connectivity patterns stand out as the most popular choice. The service-oriented architecture is the most frequently applied style in this context. Among all applicable quality attributes, scalability, timeliness, and security were the most investigated quality attributes. In addition, we included nine cross analyses to address the relationship between architectural patterns, styles, views, and evaluation methodologies with respect to different quality attributes and application areas.
Conclusions: Our findings indicate that research on software architectures in this domain is increasing. Although few studies were found in which industrial evaluations were presented, industry requires more scientific and empirically validated design frameworks to guide software engineering in this domain. This work provides an overview of the field while identifying areas for future research.

Keywords: software architecture; complex systems; Internet of Things (IoT); cloud computing; fog computing; edge computing

1. Introduction

The Internet of Things (IoT) allows a wide range of objects to interact with each other via wireless communication technologies, thus enabling smart processes and advanced ser-

*Corresponding author
Email address: ahmad.banijamali@oulu.fi (Ahmad Banijamali)

vices [1, 2]. Pervasive service provision in IoT requires large-scale computing power and the availability of resources [3], which can be provided by other technologies, such as cloud computing. In recent years, researchers have increasingly focused on the convergence of cloud computing and IoT to enable the remote management of IoT devices, anytime and anywhere [2, 4, 5].

Figure 1 describes the concept of the convergence of cloud computing and IoT (hereafter called *CoT* in this paper) and its underlying technologies. CoT leverages advanced services by connecting heterogeneous devices that use communication technologies through convenient and on-demand access to shared and configurable computing resources in the cloud [6]. Within this context, IoT devices interact with remote cloud systems that are in charge of collecting, processing, and making uniform data [7, 8]. The cloud acts as the front end to access IoT devices to create complex processing for the IoT applications [2].

In addition, recent IoT application areas enable nearly real-time services, such as over-the-air (OTA) updates and location awareness via low latency communications [9]. Therefore, it is necessary to bring data processing from the cloud to the edge of the network and closer to the devices through distributed tiny clouds in what is known as fog computing [10], which enables new applications and services by the mass adoption of IoT [11].

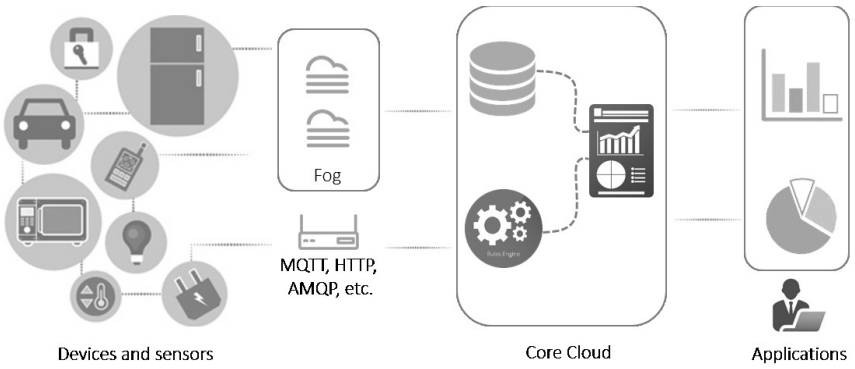

Figure 1: Cloud-IoT convergence (CoT)

Designing software architectures in the context of CoT includes challenges that stem from emerging IoT and cloud technologies, distributed characteristics of CoT, and the ever-growing demand for quality requirements, such as scalability, timeliness, and security [6, 12, 13]. By addressing these challenges, software architecture design has become an essential discipline in this context. Architectures are used to create a comprehensive understanding of the system and guarantee adequate levels of quality [14].

A large body of knowledge of software architectures has been developed in recent years in several published studies that investigate how architectures can address the quality of software systems in CoT. However, no previous study has provided a systematic overview and analysis of the existing architectures of CoT.

This study aims to fill this gap and provide a comprehensive review of the body of existing literature regarding software architectures in CoT. More specifically, the study identifies and analyses previous publications on software architectures of CoT –including design patterns,

styles, views, quality attributes, and evaluation methods– by collecting and synthesising scientific contributions in a systematic manner. The objectives of this study are summarised below:

- To analyse the intensity and characteristics of research pertaining to software architectures of CoT, including bibliographic data, research type, contribution type, and study quality;
- To evaluate software architecture designs in CoT in scientific literature in order to:
  - identify architectural design patterns, styles, and views in this domain;
  - analyse the quality attributes addressed in CoT software architectures and their relation with other architectural elements;
  - review architectural evaluation methodologies in this domain;
- To summarise existing CoT models that have been discussed in the software architecture literature.

This study provides researchers with a structured review of state-of-the-art software architectures in the context of CoT. Specifically, the research aims to identify and analyse primary studies, along with their applied research types and contribution types as well as highlight research gaps in the CoT software architectures. The research findings will also enable practitioners to restructure their design perceptions related to various CoT architectural design elements, such as design patterns, styles, and views. The study strives to review these architectural elements along with different quality attributes and application areas.

The remainder of this paper is organised as follows. Section 2 presents the background and related work on the IoT, cloud computing, and their convergence. Section 3 elaborates on the research methodology used in the research, including research objective and questions, systematic literature review steps in this study, a pilot study, and threats to validity of the results. Section 4 describes the findings from our literature review and nine cross analyses to address the relationship between architectural elements, quality attributes, and application areas. Section 5 describes further discussion of the results and recommendations for future study. Finally, Section 6 concludes the results of the study.

2. Background and related work

This section first provides a brief overview of IoT, cloud computing, and the main motivations to converge these two technologies. Then, it presents a brief overview of relevant literature reviews and surveys in this domain.

2.1. Related research and contribution to CoT

The IoT has impacted numerous aspects of our everyday life and behaviour [1]. It is now a significant part of different domain applications, such as home automation, industrial automation, health care, intelligent energy management and smart grids, and automotive

to seamlessly incorporate a large number of heterogeneous smart devices and provide open access to data for the development of innovative services [15]. The IoT has enabled self-configuring capabilities in devices based on recent communication protocols in which physical and virtual devices are seamlessly integrated into information networks [16].

The design of the IoT has evolved due to the convergence of emerging technologies, real-time analytics, commodity sensors, and embedded systems [17]. In order to address the design and architectural principles for smart objects in the IoT, previous research [18] identified a hierarchy of architectures and described three types of activity-aware, policy-aware, and process-aware devices to demonstrate how the respective architectural abstractions support increasingly complex applications [18]. Another study applied the concept of IoT devices to integrate sensors, actuators, and software agents related to the IoT information service to present a software architecture that leads to the future of the IoT [19].

Despite the benefits of the IoT, it has several technological and research challenges, such as lack of storage and computation power [20], scalability of distributed systems [21], security, and accessibility [1]. As a solution to these issues, industries and researchers have merged the IoT with the virtually unlimited capabilities and resources of the cloud [22]. Cloud computing creates the possibility of ubiquitous and on-demand network access to a shared pool of configurable computing resources [2] that can be rapidly provisioned with minimal management effort [23].

Recent years have seen growing efforts to provide a bridge between the cloud and the IoT [22]. Using the CoT enables the accessing of anything –anywhere and at anytime – without concerns about storage capacities, operational performance, processing capacity, or resources [3]. The CoT indicates a new type of distributed system consisting of a set of smart devices interconnected with a remote cloud infrastructure or software through the Internet [7]. The technology deploys IoT applications in dynamic environments using sufficient computational resources [12]. Fu et al. [12] indicated that cloud computing can help to conduct a system-level comprehensive analysis of IoT spatial-temporal data collection from different locations.

As an important part of software systems, architectures create the fundamentals for a mutual understanding of systems in which relevant decisions have a significant impact on system qualities [14]. They provide reusable abstractions of those systems that are transferable to other systems with similar requirements [24]. In this regard, many European Union projects have been defined to make advances in the architectures of CoT and its underlying technologies that can be tailored for different application areas. For example, IoT-A ("Internet of Things - Architecture")[1] developed an architectural reference model together with the definition of an initial set of key building blocks for the IoT. IoT-A has combined top-down reasoning about architectural principles and designed guidelines with simulation and prototyping in exploring the technical consequences of architectural design decisions. As another example, Cloud for Europe (C4E)[2] carried out a gap analysis to create a clear view of the public sector requirements and usage scenarios for cloud computing. C4E provided a list of obstacles for cloud computing and the services that can improve them. ClouT("Cloud

[1] https://cordis.europa.eu/project/rcn/95713_en.html

[2] https://cordis.europa.eu/project/rcn/109302_en.html

of Things for empowering the citizen clout in smart cities")[3] created a reference architecture to leverage the cloud as an enabler to bridge the IoT with services to make cities smarter and to help them face emerging challenges, such as efficient energy management and economic growth and development.

2.2. Related work

This section provides a brief overview of previous literature reviews and surveys on the software architectures of CoT and its underlying technologies. It gives readers an overview of the results and research approaches to software architectures described in previous secondary studies relevant to this domain.

As a literature review in the domain of IoT, Madakam et al. [25] have investigated IoT architectures and enabling technologies for daily life. According to the authors, seamless IoT architectures generally comprise hardware (e.g., sensors and actuators), middleware (e.g., storage and data analytic tools), and presentations (e.g., visualisation and interpretation tools) [25]. Another study [26] surveyed major enabling technologies and applications of IoT and extracted the relevant architecture, including four layers of sensing, networking, services, and interfaces. The authors explained that IoT requires enabling technologies for identification and tracking, communication, networks, and service management [26].

Whitmore et al. [27] conducted a survey and reported on the current state of research on the IoT. They classified scientific literature into the following six categories: technology (e.g., software, hardware, and architecture), application area (e.g., health and transportation), challenges (e.g., security and privacy), business models, future directions, and overviews/surveys. Their findings can be applied to identify and classify challenges that threaten IoT diffusion [27].

In terms of secondary studies on cloud computing architectures, one study [28] synthesised the previous literature on cloud computing from a business perspective and noted a growing consensus about cloud computing characteristics and design principles. Rimal et al. [29], who developed a taxonomy to describe cloud computing architectures, used the taxonomy to investigate several existing cloud platforms. In another study [30], the authors conducted a survey on mobile cloud computing and provided an overview of the technology's definitions, architectures, and applications. Jamshidi et al. [31] concluded that several research gaps still exist regarding reliable frameworks for migration to cloud platforms [31].

Finally, Cavalcante et al. [32] conducted a mapping study on the interplay of IoT and cloud computing. They identified relevant strategies to form a bridge between the IoT and the cloud, and they provided a brief overview of architectural models in the CoT context. As an interesting result, they noted that architecture is the main research topic in the context of CoT [32].

3. Research methodology

A systematic literature review (SLR) was conducted to identify and analyse the primary studies on software architectures of the convergence of cloud computing and IoT (CoT).

[3] `https://cordis.europa.eu/project/rcn/109108_en.html`

An SLR is defined as a secondary study that applies a well-defined approach to determine, assess and interpret scientific evidence related to a specific research question or topic area in a way that is unbiased and repeatable [33]. This study has adopted the guidelines proposed by Kitchenham and Charters [33]. Additionally, the guidelines by Petersen et al. [34] were taken into consideration, particularly during the study classification process and the identification of the research area. Figure 2 presents the research steps that were adopted from Kitchenham and Charters [33] and customised for this study. For example, this research has conducted an additional pilot study (comparable to Kitchenham and Charters [33]) to investigate possible search strings.

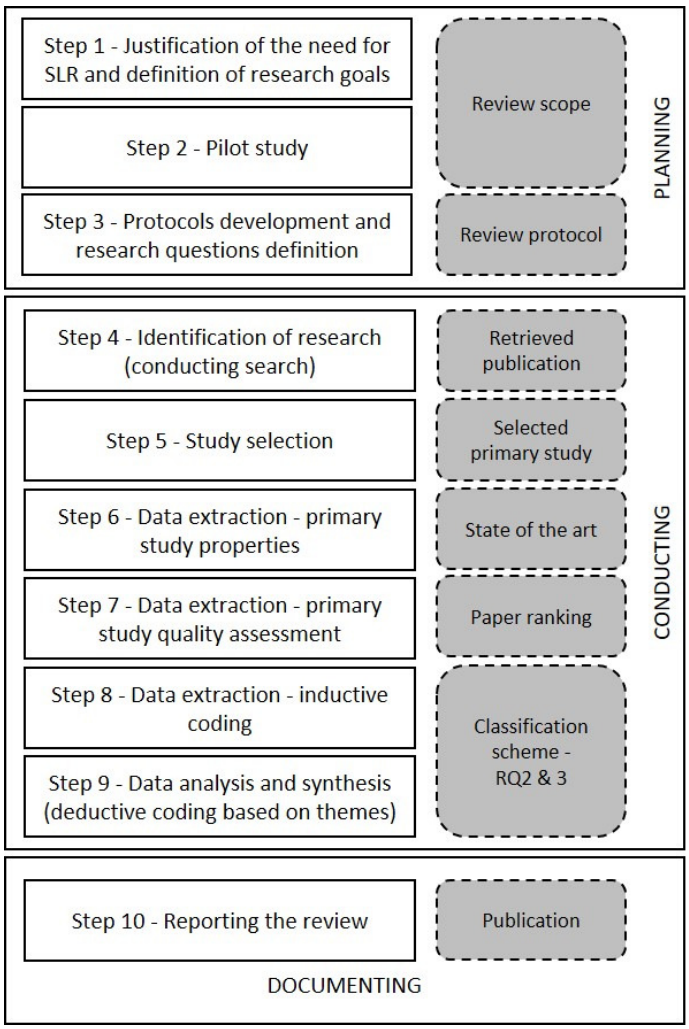

Figure 2: Systematic literature review steps

3.1. Objective and research questions

We adopted the Goal-Question-Metric (GQM) approach [35] to systematically define the research objective in this study. A GQM goal is a measurement goal that is formalised according to certain dimensions [36], which are presented in Table 1.

Table 1: Research Objective

Analyse	software architecture designs
For the purpose of	characterisation
With respect to	research intensity and characteristics, architectural design representation elements, quality attributes, evaluation methodologies, and models
From the viewpoint of	the researcher
In the context of	the convergence of cloud computing and the IoT

The consolidated objective of this research is defined in the following manner: "**Analysing** the software architecture designs **for the purpose of** characterisation **with respect to** research intensity and characteristics, architectural design representation elements, quality attributes, evaluation methodologies, and models **from the viewpoint of** researchers **in the context of** the convergence of cloud computing and the Internet of Things". This objective was translated into three research questions and their rationales, as presented in Table 2.

Table 2: Research Questions

ID	Question	Rationale
RQ1	What are the intensity and characteristics of research pertaining to software architectures in the context of CoT?	Identifying and structuring primary studies in terms of bibliographic data, research type, contribution type, and study quality
RQ2	What software architectures have been investigated most in the context of CoT?	Identifying, evaluating, and synthesising software architectural elements, such as patterns, styles, views, and evaluation methods, considering various quality attributes and CoT application areas
RQ3	What CoT models have been provided in software architecture literature?	To identify the existing CoT models in the software architecture literature.

With RQ1, we aim to obtain insight into the characteristics of and intensity of research in software architectures of CoT. Our analysis includes bibliographic data, research type, contribution type, and study quality (see Appendix B for more information on the classifications). RQ2 addresses the need for a complete analysis of software architecture designs in terms of design patterns, styles, views, application patterns, and evaluation methods that were mostly applied in relation to different quality attributes and application areas in the context of CoT. The last research question summarises existing CoT models that were discussed in software architecture literature.

3.2. Search strategy

An important step in a systematic literature review is the identification of relevant studies that can answer the research questions [37]. In order to develop and evaluate the search strategy, different approaches are available in the literature [38]. In this study, we have adopted and followed an iterative approach that allowed for the analysis and gradual improvement of the search string.

Several iterations of the pilot study were conducted in various bibliographic databases (in January and February 2018) to find the optimum search strategy that would minimise the noise and appropriately retrieve relevant studies. The pilot study started with the following search string: ("Internet of Things" OR IoT OR M2M OR "Machine to Machine" OR "Machine-to-Machine" OR WoT OR "web of things") AND (Cloud OR Fog OR Edge) AND architecture. To address topics related to this research, the string comprises a combination of various synonyms of the three main elements of the topic with the logical operator of "AND".

Using this search string led to a set of 4,310 articles from various bibliographic databases. Our initial hypothesis was that the noise ratio must have been high because of the use of broad separate terms in the search string. Obviously, many articles use these keywords but do not fall within the scope of the present study. To evaluate this hypothesis in an objective manner, an extra search with a similar search string was conducted in Google Scholar. We then analysed the top 100 results to evaluate the effectiveness of the search string. To reduce bias, two authors reviewed the set of 100 papers separately and recorded their votes regarding the relevancy of the papers.

Fleiss' kappa [39] was used to evaluate the consistency of the selection process through an inter-rater agreement calculation. In addition, a publication from Landis and Koch [40] was applied to describe the relative strength of agreement of the kappa statistics, where a kappa value of less than 0 indicates no agreement, 0 to 0.20 is slight agreement, 0.21 to 0.40 indicates fair agreement, 0.41 to 0.60 shows moderate agreement, 0.61 to 0.80 indicates substantial agreement, and finally 0.81 to 1 is in almost perfect agreement. After we followed the above-mentioned approach, the inter-rater agreement showed substantial agreement (Fleiss' kappa $K = 0.78$) for determining the publications' relevancy to the research topic. The authors further discussed any differences in evaluation during a joint meeting to clarify these disagreements.

The final pool of the pilot study comprised 54 studies (54%) that were relevant to the scope of this study, to be investigated thoroughly afterwards. We realised that all relevant studies used *software* at least in the article title, abstract, or keywords and had been published since 2010. A previous study [22] also reported that the combination of *IoT* and *cloud* was included in research titles after 2010.

In the next step of the pilot study, we extracted all keywords from the set of 54 relevant studies to identify and analyse the keywords used by researchers in this field. Figure 3 shows the results of the keyword analysis based on the frequency with which they appeared in the set of 54 relevant studies.

Keyword	Freq.	Keyword	Freq.	Keyword	Freq.	Keyword	Freq.	Keyword	Freq.	Keyword	Freq.
Internet of Things/IoT	48	Cluster	2	Citizen science	1	Intelligent perception	1	Performance	1	System and software infrastructure	1
Cloud computing/Cloud	34	Computer centres	2	Cloud Cover	1	Intelligent transportation systems	1	Pervasive computing	1	Telecommunication network routing	1
CoT/Cloud of Things	7	Manufacturing cloud	2	Cloud-assisted system	1	Internet of services (IoS)	1	Public awareness	1	Telemedicine	1
Fog Computing	7	Orchestration	2	Cloud-sensor architecture	1	JSON	1	Raspberry Pi	1	Topology	1
Architecture/software architecture	5	Platform/platform design	2	Collaboration	1	Low power	1	Representational state transfer	1	User scenarios	1
Cloud manufacturing	5	Semantic (interoperability, technologies)	2	Computer network security	1	Manufacturing resource	1	Resilient cloud	1	WAMP	1
Edge Cloud/Edge computing	5	Ubiquitous computing/Ubiquitous sensing	2	Computerised instrumentation	1	Manufacturing service	1	RESTful Web services	1	Wearable Sensors	1
Healthcare (applications, industrial IoT)	5	Web of Things/WoT	2	Computing-oriented manufacturing	1	Manufacturing system	1	RFID	1	WebSocket	1
Virtualisation/Virtual Functions	5	Web services	2	Concept	1	MapReduce	1	SAaaS	1	Vehicle maintenance services	1
M2M	4	6LowPAN	1	Connected Objects	1	Micro Data Center	1	scalability	1	Vehicular cyber-physical systems	1
Sensor (networks, sensor-centric applications)	4	Access	1	Context-awareness	1	Middleware	1	SDN	1	Vehicular networks	1
Service-oriented architecture/SOA	4	Advanced manufacturing systems	1	Cooperating Smart Objects	1	Mobile	1	Service delivery	1	Wind Farm	1
wireless sensor networks	4	Agent-oriented Computing	1	Decision-making	1	Modelling and Simulation	1	service-oriented manufacturing	1	Zigbee	1
CoAP	3	Agricultural Information Cloud	1	Distributed cloud	1	Multi-cloud	1	Signal watermarking	1		
Container	3	Ambient Aiding Living	1	ECG monitoring	1	Next generation networking	1	Single-board Computer	1		
Gateways/Smart gateway	3	Arduino	1	E-health	1	OMA LwM2M	1	Smart Agriculture	1		
Mobile cloud/Mobile computing	3	Automobile service	1	Event-based architecture	1	OneM2M	1	Smart Grid	1		
Open source	3	Big data	1	Federation	1	OpenStack	1	Smart home	1		
PaaS	3	Biomedical communication	1	Framework	1	Patient Monitoring	1	Smart objects	1		
Resource (allocation, description, management	3	Bluetooth	1	FutureGrid	1	Peer-to-peer computing	1	Software Define Networking	1		

Figure 3: Keywords analysis

This study hence modified and complemented the search string using additional criteria and terms that were found both in the pilot study and in previous SLRs. This new search led to a division of the search string into two combinations of terms, as presented in Table 3. The total number of studies retrieved using these new search strings was 1,618. In comparison to the pilot search string, the revised search string had less noise (the proportion of irrelevant studies to all retrieved records) and a higher accuracy rate (the proportion of relevant studies to all retrieved records).

Table 3: Search Keyword

Search string
Search string A: ((“Internet of things” OR IoT OR M2M OR “Machine to Machine” OR “Machine-to-Machine’ OR WOT OR “Web of Things”) AND (Cloud OR “Fog computing” OR Edge)) AND architecture AND software
Search string B: (“Cloud of Things” OR CoT OR “Cloud manufacturing” OR “Manufacturing Cloud”) AND architecture AND software

An automatic search method was used to retrieve the relevant studies in a number of selected bibliographic databases as shown in Table 4. The selected databases include ACM Digital Library, IEEE Xplore, Scopus, ISI Web of Science and Science Direct. According to Dybå et al. [41] and Kitchenham and Brereton [42], IEEE Xplore and ACM as well as two indexing databases will return the most relevant publications. Five aforementioned databases were selected considering wide coverage of the software engineering literature. In addition, this study reviewed all papers from other venues that were found highly relevant to the field (i.e., “Future Internet of Things and Cloud (FiCloud)”, “Cloudification of the Internet of Things (CIoT)”) to reduce the risk of missing relevant papers.

Table 4: Selected databases and number of retrieved papers (search date: 23.03.2018)

Database	Filter	Papers
ACM Digital Library	None	125
Scopus	Limited to Title/Abstract/Keyword	553
IEEE Xplore	Metadata	613
ISI Web of Science	Limited to Topic (TS)	295
Science Direct	Limited to Title/Abstract/Keyword	32
Total		1,618

3.3. Screening of relevant papers

The screening process of the papers included establishing the inclusion/exclusion criteria, defining the study selection process, and describing the snowballing process to minimise the validity threat of missing relevant primary studies.

3.3.1. Selection criteria

We selected studies to be included in the literature review if they presented a scientific contribution to the body of software architecture knowledge in the context of CoT. Specifically, we included papers that were scientific and clearly stated the architectures in the context of the convergence of cloud computing and IoT.

The selection criteria used in this study included both theoretical and empirical studies [43]. The search results included:

1. studies that addressed the software architectures of CoT at any level of abstraction, including design patterns, styles, views, scenarios, evaluation methods, quality attributes, etc.

2. studies that identified procedures and techniques for software architecture management of the CoT.

Accordingly, the following criteria were considered for exclusion:

1. studies that addressed either IoT or cloud computing but not their convergence;

2. studies that addressed topics other than software architectures;

3. articles that were duplicates;

4. non-peer-reviewed studies, including introductions to special issues, calls for papers, keynote speeches, prefaces, standards, etc.;

5. studies that were not written in English.

As an example, papers that only discussed topics, such as data mining, data analytics, or machine learning, were excluded.

3.3.2. Selection of primary studies

Figure 4 shows the process we followed to screen and select the primary studies. In order to have an integrated list of studies from different databases, a reference management system (RefWorks) was used to import all data into one spreadsheet document.

In the beginning stage, the first author evaluated all retrieved studies (1,618) and removed all duplicate and non-English studies. All non-peer-reviewed studies, such as introductions to special issues, calls for papers, keynote speeches, prefaces, and standards, were identified and removed accordingly. A total of 934 primary studies remained at the end of this stage to be thoroughly reviewed.

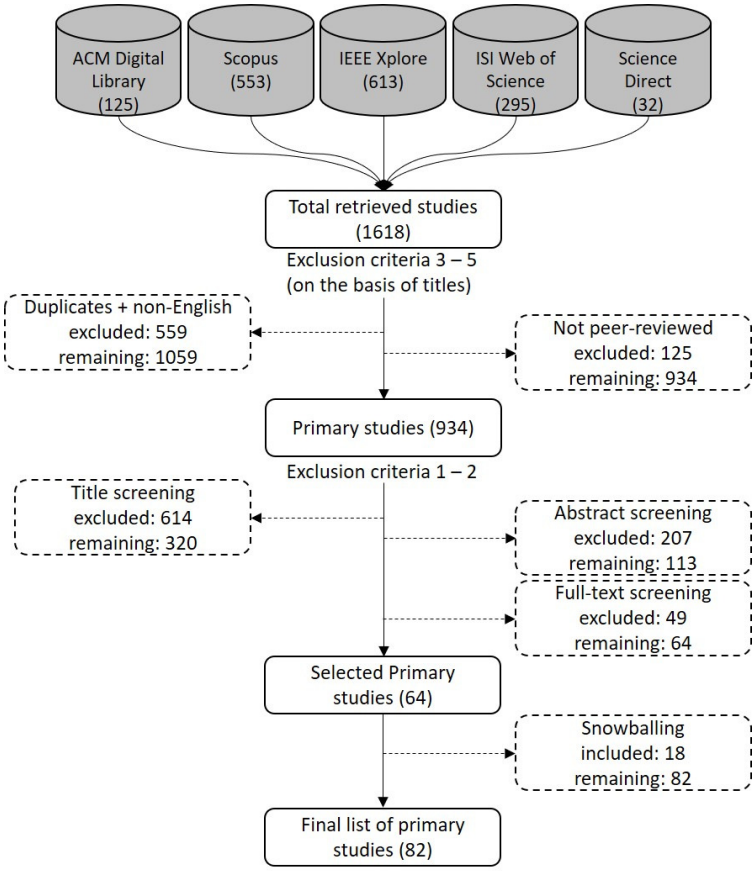

Figure 4: Selection of primary studies

In the second stage, two authors separately analysed the titles of the studies based on the first two exclusion criteria. Both authors concluded with two lists of included and excluded papers. Afterwards, in a joint meeting, all differences in the votes were discussed, and any disagreements were clarified in order to achieve a common understanding of the selection process. A total of 320 papers were included in the next stage of abstract screening. During the next stage, two authors separately reviewed the paper abstracts and provided a study list containing inclusion and exclusion votes. In another joint meeting, the abstracts of the papers on which two authors had voted differently were thoroughly reviewed, and any

disagreements were discussed. After this meeting, we decided to include 113 papers to be reviewed based on the full texts of the papers. The remaining records were then investigated in the next step in which two authors separately reviewed the full text of the articles and recorded their decisions. By crosschecking the votes and discussing any disagreements, the authors agreed that 64 articles addressed the scope and inclusion criteria of this research.

3.3.3. Snowballing

The study selection process was complemented with a backward snowballing process [44, 45]. We then analysed the references of all 64 primary studies yielded by the study selection process. The review included screening the title, publication venue, and (if necessary) the full text to extract as much information as possible from the paper under examination. We also reviewed all recent works from the authors of primary studies whenever they mentioned that their studies were works in progress; we then included those authors' future studies when relevant. Additionally, all references of the new studies found in the snowballing process were checked to reduce the risk of missing relevant papers. At the end of this stage, the authors found and included 18 more papers within the final pool. Appendix A shows a list of the selected primary studies.

3.4. Data extraction

Table 5 represents the properties used during the data extraction process. These properties were used to answer the research questions. The first column indicates the property ID; the second one includes a short description of the property; the third column refers to the cardinality of the relationship between an individual primary study and the data property; and finally, the last column traces the relationship between an individual data property and the relevant research question. The next section provides more information about the data properties.

Table 5: Data properties

ID	Title	Cardinality	RQ
DP1	Publication year	1:1	RQ1
DP2	Publication source	1:1	RQ1
DP3	Research type	1:1	RQ1
DP4	Contribution type	1:*	RQ1
DP5	Primary study quality assessment	1:1	RQ1
DP6	Architecture representation elements	1:*	RQ2
DP7	Quality attribute	1:*	RQ2
DP8	Architectures evaluation method	1:1	RQ2
DP9	Application area	1:1	RQ2
DP10	Existing CoT models	1:*	RQ3

3.4.1. Data properties

In this study, 10 data properties were established in order to answer the research questions. Five generic data properties (DP1-DP5) were used to answer RQ1. The authors ex-

tracted study information, such as publication year, publication source, research type, contribution type, and research quality. A deductive coding approach was then followed to answer RQ2 and RQ3 (DP6-DP10). For this purpose, a qualitative data and research analysis tool (NVivo) was used to support the coding and data extraction process. We created different codes and used NVivo to quote sentences directly from the primary study. Appendix B presents a detailed description of the data properties.

3.4.2. Study quality assessment

Kitchenham et al. [33] proposed that in the quality assessment process in an SLR, it is necessary to investigate whether differences in study quality could provide an explanation for differences in the study results, and if this notion can be used as a means of realising the importance of studies during results synthesis. According to Wohlin et al. [46], there is no universally agreed-on and applicable definition of study quality, although the most practical means for quality assessments are checklists. For this purpose, the authors used Kitchenham et al.'s [33] guidelines to define the quality criteria. In order to increase assessment validity, each primary study was reviewed by two authors. The primary studies included qualitative studies, and quality assessment (QA) criteria were established based on the following questions proposed by Kitchenham et al. [33]:

- QA1. Are the study findings credible?
- QA2. How adequately has the research process been documented?
- QA3. How defensible, scientific, and detailed is the design?
- QA4. How well has the design evaluation been conducted?

We scored each study as Y = 1 (concrete and reliable information), P = 0.5 (partially available information) and N = 0 or Unknown (no information is specified). The studies were scored as follows: QA1 investigates the credibility of the findings and to what extent the findings are important to the domain. QA2 is related to the scientific reporting of the research process and the results. QA3 evaluates how defensible and scientific the approach is, and the level of detail of the primary study's discussion. The last question (QA4) refers to the evaluation process of the findings.

3.4.3. Data synthesis

We adopted descriptive statistics in this SLR to address RQ1. Quantitative descriptions of frequencies were used to evaluate the publication year, publication source, research type, contribution type, and quality of the primary studies. In addition, thematic synthesis, suggested by Cruzes and Dybå [47], was used to answer RQ2 and RQ3. With thematic synthesis, primary studies are coded to label related concepts and findings and then map them to different code categories by drawing recurrent patterns. For this purpose, NVivo was used for the inductive coding to investigate specific software architectural elements (such as design patterns, styles, and views), identify the quality attributes, classify evaluation methods,

present the application areas, and summarise the existing CoT models. These code categories were further evaluated and refined to improve reliability and to obtain higher-level categories through an inductive synthesis approach.

NVivo[4] is a well-established software package with a large number of embedded features to support qualitative data analysis. Numerous systematic literature review studies and guidelines in the software engineering literature (e.g., [48, 49, 50, 51, 52]) reported a positive experience of using NVivo for the analysis of qualitative data in systematic literature reviews. All data within NVivo is arranged in the form of nodes and documents. In our case, documents were the primary studies and nodes were places where we stored code categories of interest to our study extracted from the primary studies. The selection of NVivo enabled us to import and code textual data, retrieve, search, and review the coded data. NVivo is a "method-free" software and the principles employed in structuring code categories are common among numerous methods involving coding [53]. Qualitative analysis of a large amount of data, regardless of the tools available, takes a considerable amount of time and effort. The application of NVivo was useful in shortening the analysis process and making the qualitative data analysis systematic and computer-based. A previous research [52] noted that NVivo saves researchers from time-consuming manual coding and boosts the accuracy and speed of the analysis process. In addition, performing an electronic search may yield more reliable results, as human error is reduced.

3.5. Threats to validity

Several potential threats to the validity of this research were carefully considered as we interpreted the findings. This section elaborates on the strategies we used to minimise the effects of those threats.

One important potential threat to the validity of the research results is related to biases for the identification and selection of the primary studies, as well as the data extraction. Zhang et al. [37] explained that the identification of as many relevant primary studies as possible within the scope of the research is a critical task in an SLR. In practice, however, this could be very challenging, particularly in reviews of broad and trending topics, such as IoT or cloud computing. In addition, the convergence of cloud computing and IoT is a recent approach that requires more maturity in its domain-specific terminologies. To mitigate this threat and to reduce the risk of missing relevant studies, the authors followed an iterative approach using Google Scholar as well as five bibliographic databases (see section 3.2); we employed a systematic search strategy to identify as many relevant primary studies as possible. We carefully adopted guidelines and search strategies that have been widely evaluated and accepted within academic publications. On the basis of the experimental search and pilot study, an appropriate search strategy was designed that was then used to retrieve a reliable number of relevant studies.

Previous studies [38, 54] have acknowledged that it is not always possible to identify and synthesise all relevant studies. But because the aim of SLRs is to find all relevant research on the area of interest, a good sample of the relevant research articles will likely be obtained

[4]https://www.qsrinternational.com/

during the course of an SLR [54]. The objective of the research strategy was to identify as many relevant primary studies as possible while minimising the amount of noise in the selection process. We thus constructed the search string based on the research questions and the relevant literature reviews and mapping studies that have been conducted in this domain. The authors included all peer-reviewed publications that were published from 2010 to 23.03.2018 in order to capture more relevant studies in the research area. Backward snowballing was implemented to reduce the risk of missing any relevant publications. Although the aforementioned activities improved the reliability of the results, the authors are not able to rule out the possibility that relevant studies were missed.

The selection of primary studies that are within the scope of the study is a critical task for the validity of the literature review [33]. We tried to mitigate this threat by creating a research protocol between the authors and by carrying out a pilot study to construct a common understanding of the topic and the search criteria. The authors held several meetings to discuss the selection process before we started in order to reduce the authors' bias, subjective evaluation, or misjudgment of the study under evaluation. In addition, in order to minimise the subjectivity of the selection, each step in the selection process was conducted by two authors. All conflicts and disagreements in two decisions were discussed and resolved afterwards in joint meetings.

Publication bias arises from the problem that positive research outcomes are more likely to be published than negative ones [55]. This usually happen when SLRs compare specific methods or techniques [33]. The effect of this threat in this study should be trivial, as no comparisons were made between methodologies, models, or tools.

Another potential threat to the validity of the research results is related to the researchers' bias in the extraction and interpretation of the data in the primary studies. To minimise the effect of this threat, a data extraction form was created in Microsoft Excel with a detailed description of each data property used in this research. In addition, manual qualitative data analysis is often a demanding and time-consuming process. In a large pool of 82 primary studies, there is a little chance to precisely find, code, re-code, and query a part of texts. NVivo boosted the accuracy and speed of the data extraction and analysis process using a computer-based tool.

Finally, the reliability of the research results requires careful consideration regarding the repeatability of the research process and results [56]. We precisely defined and documented the review protocol, including the steps we implemented, the search string we adopted, the bibliographic databases we used, and the data properties we examined.

4. Findings

From the initial set of 1,618 studies, this SLR selected 82 papers due to their contributions to the topic. This section summarises the findings and results of analysis of the primary studies. Section 4.1 presents a descriptive overview of the studies and analysis of the publications years, research types, contribution types, and study quality assessments in the domain of software architectures of CoT (RQ1). Section 4.2 elaborates on the findings regarding CoT software architectural elements, evaluation methodologies, quality attributes,

and application areas (RQ2). Section 4.3 presents a list of the existing CoT models in the software architecture literature (RQ3).

4.1. RQ1. What are the intensity and characteristics of research pertaining to software architectures in the context of CoT?

This literature review study contains primary studies published until 23.03.2018. The concepts of cloud computing and IoT became popular more than a decade ago. The first combination of these two concepts in research titles appeared in 2010 [22].

Since that time, a growing trend has become visible in the number of publications in this area, as illustrated in Figure 5. This rise is perhaps caused by the increasing technological impact of IoT and cloud computing, as well as the complementary aspects of these two technologies. Emerging new technologies, such as edge and fog computing, have also helped this topic to become discussed in different application areas, such as cyber-physical systems. This study includes only one paper from 2018, as only a few weeks of 2018 were included in the study.

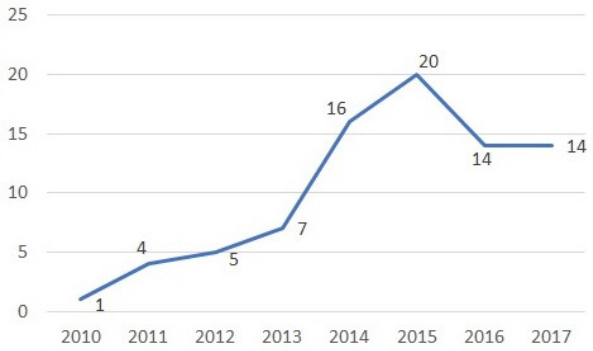

Figure 5: Number of publications per year

For publication sources, most of the primary studies presented in Figure 6 are conference publications (60, or 73%), followed by journal articles (18, or 22%) and workshop proceedings (4, or 5%). For publication venues, the results included a wide variety of conferences and journals, although the "Future Internet of Things and Cloud" conference showed the highest number of contributions on the topic.

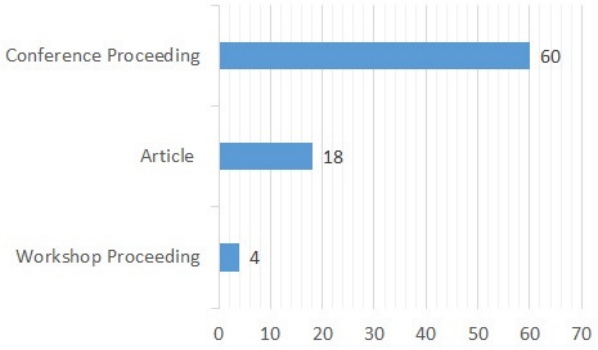

Figure 6: Publication source

We used Wieringa et al.'s [57] classification methodology to analyse the type of primary studies; we then classified them into solution proposals, validation papers, evaluation papers, and experience reports (Figure 7). The findings indicate that the highest number of publications proposed a solution (42, or 51%), followed by validation papers (36, or 44%), evaluation papers (2, or 2.5%), and experience reports (2, or 2.5%). These results indicate a need for more empirical evaluations to increase the validity and applicability of the various software architectures.

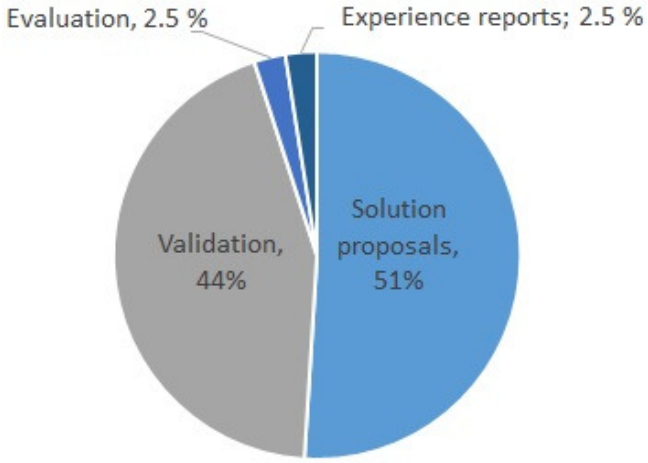

Figure 7: Research type

Based on the contribution classifications of Shaw [58] and Paternoster et al. [59], this study classified the contribution type of the primary studies into model, framework/method/technique, and advice/implications. As Figure 8 indicates, most of the studies (71, or 79%) proposed a model for the CoT architectures. The framework/method/technique category is second (18, or 20%), followed by advice/implications in third place (1, or 1%). This large number of models was expected, as architectures are usually demonstrated in the form of models. These findings suggest a need for tools and guidelines that can be applied when designing software architectures in this domain.

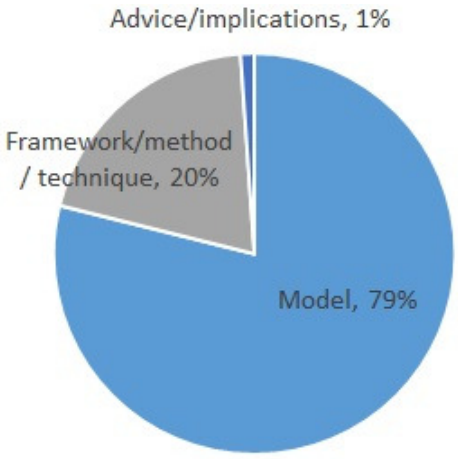

Figure 8: Contribution type

The authors evaluated the quality of the primary studies according to the quality assessment criteria defined in section 3.4.2. Table 6 presents the scores for each primary study. The total score for each study was summed up from individual scores in four questions. The quality assessment showed that all primary studies scored 2 or higher, which was above the average score. Three studies scored 4, eleven studies scored 3.5, and the rest scored between 2 and 3.

Table 6: Quality assessment of the primary studies

Study	QA1	QA2	QA3	QA4	Total score	Study	QA1	QA2	QA3	QA4	Total score
P1	Y	Y	P	P	3	P42	Y	Y	P	N	2.5
P2	Y	P	P	P	2.5	P43	Y	Y	P	P	3
P3	Y	P	P	P	2.5	P44	Y	Y	Y	P	3.5
P4	Y	P	P	P	2.5	P45	Y	P	P	P	2.5
P5	Y	P	P	N	2	P46	Y	Y	P	P	3
P6	Y	Y	P	P	3	P47	Y	Y	Y	N	3
P7	Y	Y	P	N	2.5	P48	Y	Y	P	N	2.5
P8	Y	Y	Y	Y	4	P49	Y	P	P	N	2
P9	Y	P	P	P	2.5	P50	Y	Y	Y	P	3.5
P10	Y	P	Y	N	2.5	P51	Y	Y	Y	P	3.5
P11	Y	Y	Y	Y	4	P52	Y	P	P	N	2
P12	Y	Y	Y	Y	4	P53	Y	Y	Y	P	3.5
P13	Y	Y	Y	N	3	P54	Y	Y	P	N	2.5
P14	Y	Y	P	N	2.5	P55	Y	P	P	N	2
P15	Y	P	P	N	2	P56	Y	Y	P	N	2.5
P16	Y	Y	Y	P	3.5	P57	Y	Y	P	P	3
P17	Y	P	P	P	2.5	P58	Y	Y	P	P	3
P18	Y	Y	Y	P	3.5	P59	Y	P	P	N	2
P19	Y	P	P	N	2	P60	Y	P	P	N	2
P20	Y	P	Y	N	2.5	P61	Y	Y	P	N	2.5
P21	Y	P	P	N	2	P62	Y	P	P	N	2
P22	Y	Y	Y	P	3.5	P63	Y	P	P	N	2
P23	Y	P	P	P	2.5	P64	Y	Y	P	P	3
P24	Y	P	P	P	2.5	P65	Y	P	P	N	2
P25	Y	P	P	P	2.5	P66	Y	Y	Y	P	3.5
P26	Y	Y	Y	P	3.5	P67	Y	Y	P	N	2.5
P27	Y	P	P	N	2	P68	Y	Y	Y	P	3.5
P28	Y	P	P	N	2	P69	Y	Y	P	P	3
P29	Y	P	P	P	2.5	P70	Y	P	P	N	2
P30	Y	Y	P	P	3	P71	Y	P	Y	N	2.5
P31	Y	P	Y	P	3	P72	Y	P	P	N	2
P32	Y	Y	Y	P	3.5	P73	Y	P	P	N	2
P33	Y	P	Y	N	2.5	P74	Y	Y	P	N	2.5
P34	Y	P	P	N	2	P75	Y	P	P	N	2
P35	Y	P	P	N	2	P76	Y	P	P	N	2
P36	Y	P	P	N	2	P77	Y	P	P	N	2
P37	Y	P	P	N	2	P78	Y	Y	P	N	2.5
P38	Y	P	P	N	2	P79	Y	Y	P	P	3
P39	Y	P	P	N	2	P80	Y	Y	P	P	3
P40	Y	Y	P	N	2.5	P81	Y	Y	P	P	3
P41	Y	Y	P	P	3	P82	Y	P	P	N	2

Table 7 presents the relationship between quality scores and publication years. The table includes the number of the paper, the mean quality score, and the standard deviation (SD) of the quality score for each year. No meaningful change in quality scores was noted in this study, based on the year of publication.

Table 7: Average quality scores for studies by publication date

	Year								
	2010	2011	2012	2013	2014	2015	2016	2017	2018
Number of studies	1	4	5	7	16	20	14	14	1
Mean quality score	3	2.4	2.3	2.9	2.7	2.7	2.4	2.7	3.5
Standard deviation of quality score	-	0.61	0.56	0.55	0.60	0.58	0.58	0.61	-

4.2. RQ2. What software architectures have been investigated most in the context of CoT?

The quality of a literature review depends on the classification scheme that is adopted to classify the primary studies [60]. An iterative coding process was used to analyse and classify various CoT architectures to identify the different categories and then map the primary studies to them. The authors structured their work through the information that the primary studies used to describe the software architectures. This information then led to the creation of different categories according to the following questions:

- What architecture representation elements were used to address CoT?
- What quality attributes were investigated in the CoT architectures?
- What application areas do the CoT architectures address?
- How was the validity of the CoT architectures assessed?

The authors discussed each of these questions in detail and then mapped the primary studies to the implied categories. Each category has a number of possible values that were used to design CoT architectures.

4.2.1. CoT architectures analysis

Any architecture design approach may create one or more representations of a system. This representation could be for example architectural models used as "containers" [61] for applying architectural design patterns [62], view points [63, 64], or architectural styles to express a collection of architectural design decisions that are applicable in a given development context [62]. This section of the paper presents the architectural designs used in the primary studies in order to analyse the architectures in terms of their representation elements, quality attributes, evaluation methodologies, and application areas. Table 8 shows a structured quantitative view of all values of the subcategories. In order to assign primary studies to each category in the table, we applied in this SLR whatever the primary studies had clearly mentioned or whatever the authors could deduce.

An **architectural design pattern** includes a package of design decisions that is a reusable solution to a commonly occurring problem describing a class of architectures [14]. Edge connectivity patterns use IoT gateway as an intermediate device to communicate with sensors and actuators using low-level protocols. These patterns use IP-based protocols to connect in CoT. Stream processing creates the ability to run business rules on real-time data streams to enable CoT to make spontaneous decisions, route information, or control objects in real time. Virtual device representation creates device abstraction that is visible to cloud applications and propagates silent state synchronisation. Telemetry ingestion involves offline data processing on the persisted telemetry data: for example, by aggregating the temperature over a period of the last three months to determine the average temperature of a furnace. Brokered communication requires a bidirectional asynchronous communication model using brokers. Device identity and enrollment provides the ability to keep track of all connected devices and information, such as device identifiers, device certificates, and device configuration, among others.

Table 8: Software architectures of CoT: quantitative summary of the results

CoT architectural design pattern			**Quality attribute**		
Edge connectivity (EC)	28	34%	Portability	21	25.3%
Stream processing (SP)	16	19.5%	Performance efficiency	18	21.7%
Virtual device representation (VDR)	9	11%	Security	12	14.5%
Telemetry ingestion (TI)	8	10%	Reliability	9	10.8%
Brokered communication (BC)	6	7%	Maintainability	7	8.4%
Device identity & enrollment (DI)	3	3.5%	Context coverage	6	7.2%
General	12	15%	Usability	4	4.8%
Architectural views			Compatibility	3	3.6%
Development (D)	46	33%	Functional suitability	3	3.6%
Logical (L)	30	22%	**Application area**		
Scenarios (S)	29	21%	Mobility (MO)	12	15%
Physical (Ph)	15	10.5%	Smart city (SC)	10	12%
Process (P)	15	10.5%	Health care (HC)	5	6%
General	4	3%	Smart home (SH)	3	4%
Application patterns for CoT			Manufacturing (MFG)	2	2%
Distributed IoT apps (DIoT)	31	38%	General	50	61%
Social IoT (SIoT)	19	23%	**Architectural styles**		
Cloud-based apps (CBA)	11	13%	Service-oriented (SOA)	27	33%
Digital twin (DT)	8	10%	Multi-layered(ML)	13	16%
Asset-based apps (ABA)	4	5%	Client-server (CS)	10	12%
General	9	11%	Publish-subscribe (PS)	8	10%
Evaluation methodology			Agent-based (AB)	6	7%
Experiment	16	20%	Object request broker (ORB)	2	2.5%
Prototype	14	17%	Event-driven (ED)	2	2.5%
Illustrative example	10	12%	Pipeline (P&F)	1	1%
Simulation	4	5%	Plug-in (PI)	1	1%
Case study	2	2.5%	Lambda (LBD)	1	1%
Hybrid	2	2.5%	General	11	14%
Not presented	34	41%			

Architectural views are defined to portray the architectural elements that are relevant to addressing stakeholder concerns [65]. "4+1 architectural view model" is a common view model to describe software-intensive systems that use multiple and concurrent views [64], as follows:

1. the development view presents a system from a programmer's perspective to describe system components;

2. the logical view describes the functionality the system provides to end-users;

3. the physical view presents the system-level view and is concerned with the topology of software components on the physical layer, as well as the physical connections between these components;

4. the process view is concerned with the dynamic aspects of the system in order to present how the different parts communicate in runtime environments;

5. scenarios describe architectures using a set of use cases or scenarios [64].

In order to create customised architectures, a few papers presented a hybrid **architectural style** (combining more than one styles into one) to represent the architectures of CoT. In this regard, this study classified the architectures according to the most relevant style based on what was directly mentioned in the paper or could be deducted by the authors. Applying service-oriented architectures (SOA), service providers advertise their functionality to a service registry, which service consumers can use to find appropriate services for their needs [66]. As a common architectural style, client-server is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. Multi-layered architecture is a type of client-server in which presentation, application processing, and data management functions are physically separated. The most common type of this style is three-tier architecture, containing data, logic, and presentation tiers. Publish-subscribe is a kind of messaging style where publishers (the senders) do not programme the messages to be sent directly to specific subscribers (receivers) but instead categorise published messages into classes without having any knowledge of the subscribers. The agent-based style comprises modular applications to facilitate injection and distribution through the network using agents. In distributed systems, an object request broker (ORB) is a type of middleware that enables programme calls from one computer to another via a network. The event-driven style is a messaging style that enables production, detection, and reaction to events. Pipeline (or pipes and filters) consists of a chain of processing elements, arranged such that the output of an element is the input of the next element. The plug-in style allows new features to be added to an existing software application through a software component called a plug-in. Lambda architecture is a data-processing architectural style for handling massive quantities of data by taking advantage of both batch- and stream-processing methods.

Prior studies and industrial practices, such as [67, 68, 69, 70, 71, 72] have suggested different **application patterns** that CoT must support, which makes CoT different from other types of cloud. Cloud-based apps are the first pattern that is often found in normal clouds. It is expected that a cloud provides basic support for devices, such as developing responsive web applications. The connection between this application pattern and IoT is when devices connect to clouds and perform basic functions, such as access rights and master data management for users and devices. Asset-based apps are the second category of patterns that enables the autonomous behaviour of devices by supporting application logic and data [70]. An example in this category can be autonomous driving vehicles that must continue to perform all essential functions when they are out of network coverage.

The large amount of data generated from distributed IoT devices cannot be aggregated and analysed in the core cloud; in particular, real-time data analysis implies that data must most often be analysed before it is stored [71]. Distributed IoT apps are the third pattern that enables the integration of an application's logic and data on both devices and clouds.

It supports applications that take the capabilities of local devices that work together with the cloud-based support.

A digital twin serves as a bridge between the physical world and the digital world to map physical devices to their corresponding virtual twins in the cloud [68]. It consists of three parts: physical device, virtual model, and the linkage between them [73]. The data coming from a physical device feeds a virtual model in the cloud. A key advantage in this pattern is that applications in the cloud can work without connecting and extracting the data from devices. They are deployed in a secure sandbox in the cloud, which controls data access for each application to reduce the security risks and development costs [70]. Example applications include predictive maintenance and remote monitoring and service [67].

The last application pattern in CoT is social IoT. The basic concept of this pattern is a social network of intelligent connected devices that can cooperate and exchange data among themselves [72]. Devices not only connect and interact but also socialise and collaborate with each other to perform specific tasks [74]. The data aggregated from such devices is used by multiple applications in the cloud and other devices.

In order to extract the **quality attributes**, the authors adopted the quality characteristics proposed in ISO/IEC 25010:2011 for software product quality [75]. Portability represents the degree of effectiveness and efficiency with which a system, product, or component can be transferred from one hardware, software, or other operational or usage environment to another. Performance efficiency includes characteristics that show the performance relative to the quantity of resources used under the stated conditions. Security indicates the data protection degree of a system or product. Reliability shows how products or systems perform specified functions under specified conditions for a specified period of time. Maintainability is the degree of effectiveness and efficiency with which a product or system can be modified to improve that product or system. Context coverage represents the degree to which a product or system can be used in primary contexts as well as in the contexts beyond those that were initially explicitly identified. Compatibility represents how a product or system can exchange information with other products and systems while sharing the same hardware or software environment. Functional suitability indicates the degree to which a product or system provides functions that meet implied needs when used under specified conditions.

As an **application area**, the smart city uses different technologies to increase the quality of life in urban spaces and to deliver better services to citizens. Smart mobility is a part of the smart city that refers to the technologies used to improve accessibility both within and outside the city as well as the availability of modern transportation systems. Smart health provides health services by using the context-aware network and sensing infrastructure of smart cities. Smart home creates an environment in which many features in our homes are automated, and home appliances can communicate with each other. Smart manufacturing is the pervasive application of networked information-based technologies throughout the manufacturing and supply chains.

As the main scope of this study, all primary studies addressed the IoT and cloud computing. Thirteen studies [9, 10, 11, 13, 76, 77, 78, 79, 80, 81, 82, 83, 84] addressed "edge computing", which shifts processing from data centres to the edge of the network, thus allowing a large class of applications, such as IoT to be deployed in an effective way with

lower latency. Of these studies, ten [9, 10, 11, 76, 77, 80, 82, 83, 84, 85] mentioned "fog computing" to extend cloud computing towards the edge of the network. In addition, ten primary studies [5, 7, 8, 86, 87, 88, 89, 90, 91, 92] discussed the concept of the "hybrid cloud" for those services that utilise both private and public clouds.

4.2.2. Architecture representation elements with respect to quality attributes

Table 9 provides a summary of the architecture representations of CoT that were extracted from the set of primary studies in the literature. In the following, the main results for each subcategory are summarised.

CoT architectural design pattern vs. quality attributes. This section elaborates on design patterns with respect to major quality attributes in CoT. Primary studies have applied edge connectivity and stream processing to present most of the CoT architectures.

The edge connectivity pattern addressed quality attributes, such as portability, security, and performance efficiency in CoT. In this pattern, the IoT gateways work as interface adapters to access a device's data in the cloud [94]. These gateways easily connect low-performance devices to cloud, thereby enabling dynamic adaptation according to the type of device [95]. The portable design of gateways makes them adaptable at different CoTs [94].

Further, the edge connectivity patterns improve security issues in CoT. Smart devices often have limited processing, communication, and memory storage that cannot meet the demand of traditional security technology [114]. Edge connectivity can improve security through secured communications (e.g., transport layer security), safeguards (e.g., firewalls), device authorisation mechanisms, multi-factor authentication (e.g., user, device, and system level), and access policy enforcement (e.g., permission to topics and queues) [86, 96, 115].

Performance is the third important quality attribute addressed in the edge connectivity pattern. A study [96] noted that it is often challenging to optimise the performance of an edge gateway to be used for a large number of IoT devices. Resource allocation models are a relevant solution for dynamically scheduling and allocating resources to different tasks, thereby achieving high throughput and low latency [96, 109].

Stream processing, which was second among the CoT architectural design patterns, primarily addressed performance efficiency and portability. Real-time stream processing in a distributed environment is a new research topic [99]. It exploits parallel processing that runs on multiple computational units [110]. A study [110] proposed an architecture for real-time services in vehicle clouds to enable in-vehicle resource scheduling and improve efficiency in this context. Nevertheless, the real-time data received from IoT devices makes the adaptability and replaceability issues more critical [101], particularly if mashup services have timing constraints for the processing of real-time data [100].

> Summary. IoT gateways is a solution for maintaining lower latency and higher security in CoT.

Architectural style vs. quality attributes. As the first software architectural style in CoT, service-oriented architectures have been applied with respect to all categories of quality attributes; from among these attributes, performance efficiency, portability, and security has

Table 9: Architectural elements and quality attributes

QA	CoT architectural design patterns	Architectural styles	Application patterns for CoT	Architectural views
Portability(21)	**EC(7)** [80, 93, 94, 95, 96, 97, 98], **SP(4)** [12, 99, 100, 101], **TI(1)** [102], **VDR(3)** [7, 9, 13], **BC(2)** [103, 104], **DI(1)** [10], **General(3)** [5, 84, 105]	**SOA(6)** [7, 10, 93, 94, 97, 100], **CS(3)** [12, 96, 102], **ML(2)** [9, 105], **PS(2)** [84, 99], **AB(1)** [95], **ORB(2)** [103, 104], **ED(0)**, **P&F(0)**, **PI(1)** [98], **LBD (1)** [101], **General(3)** [5, 13, 80]	**DIoT(11)** [7, 9, 10, 94, 95, 97, 99, 100, 101, 102, 103], **SIoT(5)** [5, 12, 13, 96, 105], **CBA(1)** [104], **DT(1)** [98], **ABA(0)**, **General(3)** [80, 84, 93]	**D(9)** [5, 9, 10, 12, 97, 98, 100, 102, 105], **L(10)** [7, 9, 12, 13, 84, 94, 98, 100, 103, 105], **S(5)** [12, 93, 97, 101, 105], **Ph(4)** [5, 80, 93, 101], **P(5)** [10, 93, 97, 99, 104], **General(2)** [95, 96]
Performance efficiency(18)	**EC(6)** [85, 96, 106, 107, 108, 109], **SP(6)** [83, 99, 101, 110, 111, 112], **TI(0)**, **VDR(2)** [13, 90], **BC(1)** [104], **DI(1)** [10], **General(2)** [91, 113]	**SOA(7)** [10, 107, 108, 110, 111, 112, 113], **CS(3)** [83, 96, 106], **ML(1)** [91], **PS(1)** [99], **AB(1)** [109], **ORB(1)** [104], **ED(0)**, **P&F(0)**, **PI(0)**, **LBD (1)** [101], **General(3)** [13, 85, 90]	**DIoT(7)** [10, 90, 91, 101, 108, 109, 110], **SIoT(6)** [13, 83, 96, 106, 111, 112], **CBA(2)** [104, 113], **DT(1)** [99], **ABA(2)** [85, 107], **General(0)**	**D(8)** [10, 85, 90, 91, 108, 109, 110, 113], **L(2)** [13, 85], **S(7)** [85, 91, 101, 106, 107, 111, 112], **Ph(3)** [85, 101, 106], **P(7)** [10, 99, 104, 107, 108, 109, 113], **General(2)** [83, 96]
Security(12)	**EC(7)** [2, 86, 91, 96, 114, 115, 116], **SP(1)** [111], **TI(0)**, **VDR(1)** [7], **BC(1)** [117], **DI(2)** [10, 89], **General(0)**	**SOA(6)** [2, 7, 10, 89, 111, 115], **CS(2)** [96, 116], **ML(1)** [91], **PS(0)**, **AB(1)** [117], **ORB(0)**, **ED(0)**, **P&F(0)**, **PI(0)**, **LBD (0)**, **General (2)** [86, 114]	**DIoT(6)** [2, 7, 10, 91, 115, 116], **SIoT(4)** [86, 96, 111, 114], **CBA(1)** [117], **DT(0)**, **ABA(1)** [89], **General(0)**	**D(5)** [10, 86, 89, 91, 117], **L(3)** [7, 89, 116], **S(4)** [91, 111, 115, 116], **Ph(3)** [2, 114, 115], **P(2)** [2, 10], **General(1)** [96]
Reliability(9)	**EC(2)** [96, 108], **SP(3)** [83, 91, 111], **TI(0)**, **VDR(2)** [7, 13], **BC(0)**, **DI(0)**, **General(2)** [84, 91]	**SOA(4)** [7, 108, 111, 118], **CS(2)** [83, 96], **ML(1)** [91], **PS(1)** [84], **AB(0)**, **ORB(0)**, **ED(0)**, **P&F(0)**, **PI(0)**, **LBD (0)**, **General(1)** [13]	**DIoT(3)** [7, 91, 108], **SIoT(5)** [13, 83, 96, 111, 118], **CBA(0)**, **DT(0)**, **ABA(0)**, **General(1)** [84]	**D(3)** [91, 108, 118], **L(4)** [7, 13, 84, 118], **S(2)** [91, 111], **Ph(0)**, **P(1)** [108], **General(2)** [83, 96]
Maintainability(7)	**EC(1)** [93], **SP(3)** [12, 77, 119], **TI(0)**, **VDR(1)** [7], **BC(0)**, **DI(0)**, **General(2)** [91, 120]	**SOA(4)** [7, 77, 93, 119], **CS(2)** [12, 120], **ML(1)** [91], **PS(0)**, **AB(0)**, **ORB(0)**, **ED(0)**, **P&F(0)**, **PI(0)**, **LBD (0)**, **General(0)**	**DIoT(3)** [7, 91, 119], **SIoT(2)** [12, 77], **CBA(1)** [93], **DT(0)**, **ABA(0)**, **General(1)** [120]	**D(4)** [12, 77, 91, 119], **L(2)** [7, 12], **S(3)** [12, 91, 93], **Ph(2)** [93, 120], **P(1)** [93], **General(0)**
Context coverage(6)	**EC(1)** [79], **SP(0)**, **TI(0)**, **VDR(1)** [7], **BC(0)**, **DI(1)** [10], **General(3)** [91, 105, 113]	**SOA(3)** [7, 10, 113], **CS(0)**, **ML(2)** [91, 105], **PS(0)**, **AB(0)**, **ORB(0)**, **ED(1)** [79], **P&F(0)**, **PI(0)**, **LBD (0)**, **General(0)**	**DIoT(3)** [7, 10, 91], **SIoT(1)** [105], **CBA(1)** [113], **DT(0)**, **ABA(0)**, **General(1)** [79]	**D(4)** [10, 91, 105, 113], **L(2)** [7, 79], **S(2)** [91, 105], **Ph(1)** [79], **P(2)** [10, 113], **General(0)**
Usability(4)	**EC(1)** [98], **SP(0)**, **TI(0)**, **VDR(0)**, **BC(0)**, **DI(0)**, **General(3)** [84, 91, 121]	**SOA(1)** [121], **CS(0)**, **ML(1)** [91], **PS(1)** [91], **AB(0)**, **ORB(0)**, **ED(0)**, **P&F(0)**, **PI(1)** [98], **LBD (0)**, **General(0)**	**DIoT(1)** [91], **SIoT(1)** [121], **CBA(0)**, **DT(1)** [98], **ABA(0)**, **General(1)** [84]	**D(2)** [91, 98], **L(2)** [84, 98], **S(1)** [91], **Ph(1)** [121], **P(0)**, **General(0)**
Compatibility(3)	**EC(2)** [86, 93], **SP(0)**, **TI(0)**, **VDR(1)** [7], **BC(0)**, **DI(0)**, **General(0)**	**SOA(2)** [7, 93], **CS(0)**, **ML(0)**, **PS(0)**, **AB(0)**, **ORB(0)**, **ED(0)**, **P&F(0)**, **PI(0)**, **LBD (0)**, **General(1)** [86]	**DIoT(1)** [7], **SIoT(1)** [86], **CBA(0)**, **DT(0)**, **ABA(0)**, **General(1)** [93]	**D(1)** [86], **L(1)** [7], **S(1)** [93], **Ph(1)** [93], **P(1)** [93], **General(0)**
Functional suitability(3)	**EC(0)**, **SP(0)**, **TI(0)**, **VDR(1)** [9], **BC(0)**, **DI(1)** [10], **General(1)** [120]	**SOA(1)** [10], **CS(1)** [120], **ML(1)** [9], **PS(0)**, **AB(0)**, **ORB(0)**, **ED(0)**, **P&F(0)**, **PI(0)**, **LBD (0)**, **General(0)**	**DIoT(2)** [9, 10], **SIoT(0)**, **CBA(0)**, **DT(0)**, **ABA(0)**, **General(1)** [120]	**D(2)** [9, 10], **L(1)** [9], **S(0)**, **Ph(1)** [120], **P(1)** [10], **General(0)**

received the most attention.

The service requests and data in clouds often require real-time processing to control IoT devices [10]. Parallel data processing is an approach to increase the internal throughput of requests in a service in order to improve the overall performance [107, 113]. Cloud platforms support high performance by sharing and coordinating different resources [113]. Multiple processing units minimise the service response time, improve stability of services, and increase cloud reliability and performance [111]. In addition, dynamic service discovery enables routing the request to the fastest service at runtime in order to decrease the response time [112]. Dynamic service migration is another strategy to achieve a fast response time for processing a request with respect to the platform on which a service is located [113].

Modularity and component-based software engineering in SOA increases the portability of a system [100]. Modularity was targeted in the microservices architecture (as a variation of the SOA) that breaks systems down into multiple services [93]. Services are independently replaceable, upgradeable, and deployable [93]. Containerisation is another practice in SOA architecture to simplify the packaging, distribution, installation, and execution of complex applications in cloud and IoT devices [7].

Cloud platforms provide access and redirect requests to the respective applications as soon as the users are authenticated and authorised [115]. The devices may also need to access shared databases on the cloud, which requires identity and authentication management [10]. Although our observation did not reflect a direct relationship between SOA and improving security in CoT, the primary studies [7, 89] mentioned security, privacy, and trust as the capabilities that must be satisfied by the architectures.

Next, client-server architectures primarily discussed portability and performance efficiency in this domain. Modular architectures in this section were to meet constant changes in application requirements and to adapt the architectures in other cloud platforms [102]. In addition, one of the main challenges in CoT is the processing, storing, and querying of big data in an efficient manner; thus, centralised processing is not an efficient solution [96]. Other approaches, such as creating resource-aware allocation models, were proposed to dynamically schedule and allocate resources [96].

Summary. Service-oriented architectures are well adapted to improve performance and portability of CoT architectures.

Application patterns for CoT vs. quality attributes. In the set of primary studies, it was important that CoT support different categories of applications that were represented through different design patterns. Among these patterns, a distributed IoT application, which was the most utilised application pattern, is mainly used to address portability, performance efficiency, and security.

The CoT architectures must rapidly adapt to changes in devices, sensors, and actuators [99]. A study [102] noted that current CoT designs are domain-specific and cannot be easily transferred to other domains. In a distributed data setting, scalable applications that can manage the massive amounts of data generated by the devices are a major challenge [101]. A few primary studies [99, 101] nevertheless have attempted to find solutions for managing

a large amount of data to assist decision-making in this domain.

Due to the need for increased flexibility and portability in CoT architectures, fog computing was proposed as a solution to support dynamic data management [9, 10]. Fog computing technologies have enabled efficient, location aware, and close to the edge cloud applications that helped to increase real-time performance [108]. The data originating from distributed devices varies based on different contexts in which the data must be efficiently managed and utilised by the applications in the cloud [90].

Distributed IoT applications have severe challenges in terms of security, as the data are distributed between the device and the cloud [91]. Hackers can inject malware if they obtain access to physical devices or tamper with cloud data [2]. The communication channels among user, device, and cloud are vulnerable to side-channel information leaks [2]. Prior research [115] proposed different levels of authentication to address security issues in this area.

Social IoT has received attention in terms of performance efficiency, portability, and reliability. As an example, a primary study [111] in the mobility domain emphasised a vehicle's communication with other vehicles or external environments, which results in a large amount of data. Innovative solutions are required to effectively process this data [111] and create efficient services [13]. As a portability feature, users must be able to manage and reuse their own profiles across various social networks [5]. The reliability in this pattern can also be improved by sending queries to the closest healthy node or service [111], thereby obtaining higher reliability in the entire system.

> Summary. Distributing data among clouds and applications requires efficient applications to manage security requirements in different contexts.

Architectural views vs. quality attributes. For the "4+1 architectural view model" [64], we observed that the development view was the most applied view in CoT architectures, followed by the logical and scenarios views.

Using the development view, primary studies addressed all quality attributes in CoT. The logical view is concerned with the functionality that the system provides to end-users [64]; in this view, studies mainly discussed portability of the designed architectures more than other quality attributes. Primary studies used scenarios to identify architectural elements and to illustrate and validate the architecture design with respect to almost all quality attributes; among these, performance and portability of the architectures was discussed more in the scenario view.

> Summary. Various architectural views were used with respect to all categories of quality requirement.

4.2.3. Quality attribute subcategories

Table 10 presents the subcategories in each quality attribute [75]. One primary study may address more than one subcategory in a quality attribute. As the first quality attribute, portability comprised scalability and adaptability issues. Scalability could be horizontal

by using more resources for the logical units for processing purposes [99, 100] (e.g., load-balancing requests) or vertical by adding resources to a physical unit for storage purposes [102] (e.g., database size). With the IoT, it is challenging to extend small-scale systems to the massive scale IoT, which typically involves millions of devices. Scalability is hence an important challenge [7] in designing CoT. Adaptability implies that CoT architectures should be able to evolve to fit changes in their environment or circumstances.

Table 10: Quality attribute subcategories

QA	Subcategory
Portability(21)	Scalability(19) [5, 7, 9, 10, 12, 13, 80, 84, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 104], Adaptability(2) [103, 105]
Performance efficiency(18)	Timeliness(13) [83, 85, 90, 96, 99, 101, 106, 107, 108, 110, 111, 112, 109], Performance(6) [13, 83, 91, 96, 104, 113], Transparency(1) [10]
Security(12)	Security(12) [2, 7, 10, 86, 89, 91, 96, 111, 114, 115, 116, 117], Privacy(2) [89, 91], Confidentiality(1) [10]
Reliability(9)	Reliability(7) [7, 13, 84, 83, 91, 96, 118], Availability(3) [96, 108, 111]
Maintainability(7)	Evolvability(3) [77, 91, 93], Variability management(2) [7, 119], Reconfigurability(1) [120], Re-usability(1) [93], Modularity(1) [12]
Context coverage(6)	Context-awareness(5) [7, 10, 79, 91, 113], Flexibility(1) [105]
Usability(4)	Usability(3) [91, 98, 121], User-participatory(1) [84], Accessibility(1) [91]
Compatibility(3)	Interoperability(3) [7, 86, 93]
Functional suitability(3)	Mobility(2) [9, 10], Energy efficiency(1) [120]

According to this literature analysis, security and timeliness are two other important challenges in this context. Security is a measure of the CoT's ability to protect data and information from unauthorised access while still providing access to authorised users, devices, and platforms [86, 89, 114, 115]. A secure IoT communication environment must provide users with certain access rights within a particular domain [86]. On the other hand, many CoT designs have proposed architectures that provide on-time delivery of information or services, run real-time applications, and interact with different devices in a timely fashion to ensure consistent and updated data exchange [99, 110].

Reliability implies that CoT should remain operational during its use [96], during which CoT must maintain reliability within individual components or at the system level to respect service-level agreements (SLAs) [7]. Context-awareness, which allows for the collection and storage of context information from devices [113], is the main attribute in building adaptive IoT systems and in establishing value from sensed data. Maintainability was briefly presented in the primary studies in the forms of evolvability, variability management, reconfigurability, re-usability, and modularity. Finally, usability, interoperability, and functional suitability

were the three least investigated quality attributes. In the next section, the quality attributes and other architectural elements are used to conduct a cross-analysis of the architectures in this domain.

4.2.4. Cross-analysis

This section extends the analysis of the literature review data across the different categories. Based on the observation from the reviewing process and the categories of topics, the following cross-analysis questions have been defined:

- CAQ1. What CoT architectural design patterns and architectural views have been used with different quality attributes?
- CAQ2. What apps patterns for CoT and architectural styles have been used with different quality attributes?
- CAQ3. What is the relationship between quality attributes and application areas?
- CAQ4. Is there any relationship between the quality attributes and the architecture evaluation methods?
- CAQ5. Are different evaluation approaches used in the different application areas?
- CAQ6. Is there any relationship between the evaluation methodology and CoT architectural design patterns?
- CAQ7. Is there any relationship between CoT architectural design patterns and apps patterns for CoT?
- CAQ8. Is there any relationship between CoT architectural design patterns and architectural styles?
- CAQ9. Is there any relationship between CoT architectural design patterns and architectural views?

CAQ1: CoT architectural design pattern, view, and quality attributes. The results indicated that irrespective of the design pattern selected for CoT architectures, scalability is an issue in this context. Among all design patterns, edge connectivity (37%) was mostly applied to address this quality attribute. Edge connectivity is a gateway-mediated pattern in which the services are not published externally but access is always mediated by a gateway. This design pattern facilitates a scalable deployment of applications by reserving edge deployment for those applications that must be installed closer to sensing devices [94, 95]. A study noted that keeping IoT gateways simple and without the application logic enables them to be highly scalable, thereby supporting the increasing number of IoT devices, without the need to be replaced [97].

Similarly, timeliness and security were two major attributes in studies that applied the edge connectivity design pattern. The data from the devices is often processed in real time on

the devices or the edge [96]. For critical real-time data, there is no time to transfer it to data centers in the core cloud [96]. In numerous cases, edge devices collect and process the data, handle errors locally, and then send the compiled data to central nodes [109]. With security on the edge, data is encrypted before it leaves the devices and the message encryption and multi-factor authentications ensure unique authentication, integrity, confidentiality, and privacy of the users [115].

Further, stream processing was mostly discussed in relation to timeliness and scalability issues. Real-time data processing becomes critical when there are low-latency requirements in stream computation of massive data streams and when the number of devices and amount of data that requires processing increases [12]. Apache Storm and Spark are the most notable solutions to address these issues [99]. The solutions, such as the scalable data storage tuned for high performance computing, lightweight stateless communications, and load balancing mechanisms, can help with improving the scalability in this context [101].

Scalability, timeliness, and security were quality attributes addressed through all design views. The primary studies that we examined, mostly used logical, development, and process views to represent architectures to address scalability. Scenarios were discussed to improve the understandability of architectures with respect to timeliness, security, and scalability. Other quality attributes received less attention in terms of architectural design patterns for CoT and architectural views. Figure 9 illustrates architectural design patterns and views, along with different quality attributes in CoT.

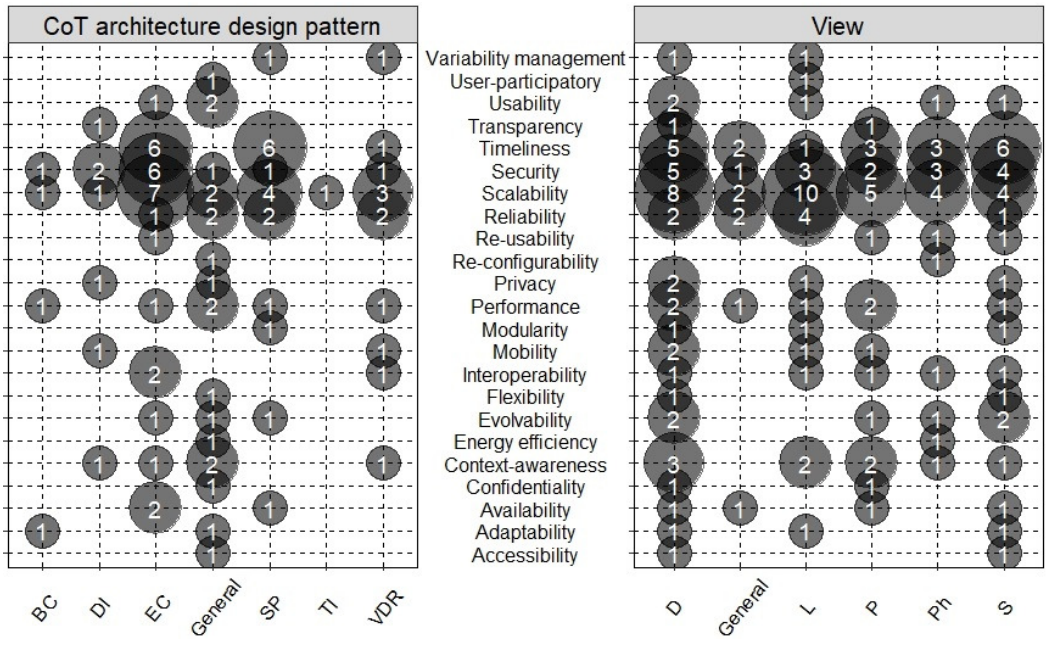

Figure 9: Quality attributes versus CoT architectural design patterns and views

Summary. Data filtering, processing, and encryption on the IoT edge can positively impact scalability, timeliness, and security in CoT.

CAQ2: Apps pattern for CoT, style, and quality attributes. Figure 10 presents the state of the primary studies in terms of the distribution of the architectural design patterns for CoT and architectural styles, along with different quality attributes.

The primary studies that addressed quality attributes mostly used a distributed IoT apps pattern in which an application's logic and data is integrated on devices and the cloud. In this pattern, scalability and security issues were investigated more frequently as compared to other attributes (Figure 10). Data logic and processing applications were deployed and orchestrated from the central cloud into the IoT edge and devices [94]. Based on the latency requirements and the amount of data to be processed, applications are deployed either to the central cloud or to the edge and then migrated from one to the other [102]. Providing end-to-end security in this pattern is difficult; that is confidentiality and integrity cannot be entirely built into architectures by default, as distributed data required by the devices may need to be accessed from shared databases in the cloud [10]. Thus, we must have secure access control to data as well as identity and authentication management [10].

As the second application pattern, social IoT addressed timeliness, reliability, scalability, and security in CoT. A primary study [106] noted that sensor networks can communicate with each other as peers in a peer-to-peer approach. Devices participate in the data collection and social processes in which they are enabled to interact and communicate among themselves and with the environment in real-time [5]. The exchanged data enables them to react autonomously to different events and create services with or without direct human intervention [5]. Our observations in this study did not reveal a direct relationship between this application pattern and security. Other application patterns and quality attributes have received less attention in the literature.

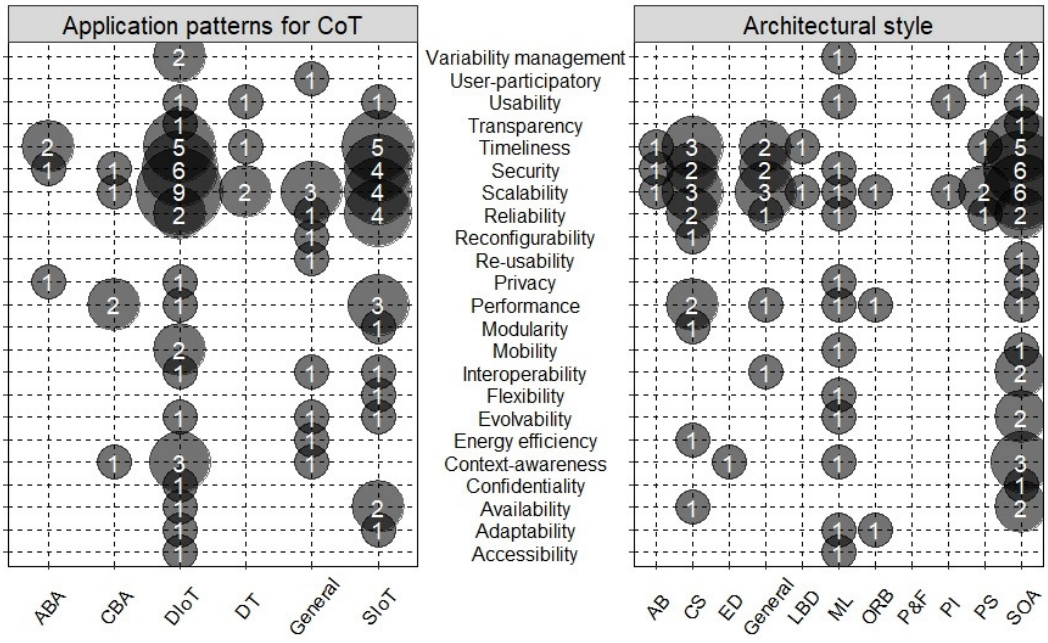

Figure 10: Quality attributes versus application patterns for CoT and architectural style

Service-oriented architecture (SOA) was the most frequently used style for designing CoT architectures for addressing quality attributes. The primary studies addressed a majority of the quality attribute clusters in CoT through this architectural style; among these clusters, scalability, security, and timeliness are important quality attributes. A primary study [12] proposed a cloud model, based on the concept of modularisation and service integration, to improve scalability and reliability of processing units in the smart city context. Another study [89] argued that security requirements are often in contrast to performance requirements, such as timeliness and throughput in SOA, and to optimise these quality requirements, trade-off frameworks may be needed.

The client-server architecture, which was second in terms of number of papers, mostly addressed scalability and timeliness. With regard to higher scalability, client-server architectures allow for increasing computation power on devices and improving hardware and network capacity [96].

Other quality attributes and architectural styles received less attention; for example, according to our observations during the reviewing process, the pipeline style has not been studied alongside the quality attributes in this domain.

Summary. Considering the different quality attributes in CoT, SOA and DIoT are the two most applied architectural alternatives for designing architectures and applications.

CAQ3: Application area and quality attributes. Because the smart city domain has a wide scope, primary studies addressed the majority of the quality attribute clusters in this area. Scalability, timeliness, and evolvability were evaluated more frequently as compared to other attributes in this domain, although a broad range of different quality attributes were found to be relevant to smart cities (Table 11). The primary studies noted several issues, such as necessity for large-scale cloud platforms [12], real-time dynamicity of a city [112], real-time network traffic [106], and extensibility to utilise new emerging technologies [77] that must be addressed by CoT architectures in smart cities.

For the next application area, software architectures addressed quality attributes, such as timeliness, scalability, and security in the mobility (MO) domain. Cloud computing functions as the platform for the integration of large volumes of heterogeneous data originating from vehicles and devices [101]. This data must be collected and analysed on the edge to ensure real-time response [10]. Another issue in large mobility settings is scalability, which impacts other requirements, such as security and mobility [101]. A study [85] noted that named data networking (NDN), as a form of information centric networking, does not need host name resolution and provides scalability and data security in mobility domain. Architectures in the other application areas did not highlight a majority of the quality attributes in CoT, which provides a starting point for future studies in these areas.

The review showed that timeliness (eight papers) was the most addressed quality attribute within different application areas in the literature. Real-time data processing is critical in different CoT application areas, such as mobility and smart cities. Security was highlighted in connected vehicles and smart transportation, health management, and smart

cities management while scalability was more evaluated in the mobility and smart city domains.

Table 11: Quality attributes versus application area

QA	Subcategory	Total	Application area					
			MO	SC	HC	SH	MFG	General
Portability	Scalability	19	2 (10.5%)	2 (10.5%)	-	-	-	15 (79%)
	Adaptability	2	-	-	-	-	-	2 (100%)
Performance efficiency	Timeliness	13	5 (38%)	2 (15%)	-	1 (8%)	-	5 (38%)
	Performance	6	-	1 (17%)	-	-	-	5 (83%)
	Transparency	1	1 (100%)	-	-	-	-	-
Security	Security	12	2 (17%)	1 (8%)	2 (17%)	-	-	7 (58%)
	Privacy	2	-	1 (50%)	-	-	-	1 (50%)
	Confidentiality	1	1 (100%)	-	-	-	-	-
Reliability	Reliability	7	-	1 (14%)	-	-	-	6 (86%)
	Availability	3	1 (33%)	-	-	-	-	2 (67%)
Maintainability	Evolvability	3	-	2 (67%)	-	-	-	1 (33%)
	Variability management	2	-	-	-	-	-	2 (100%)
	Reconfigurability	1	-	-	-	-	-	1 (100%)
	Re-usability	1	-	1 (100%)	-	-	-	-
	Modularity	1	-	1 (100%)	-	-	-	-
Context coverage	Context-awareness	5	1 (20%)	1 (20%)	-	-	-	3 (60%)
	Flexibility	1	-	-	-	-	1 (100%)	-
Usability	Usability	3	-	1 (33%)	-	-	-	2 (67%)
	User-participatory	1	-	-	-	-	-	1 (100%)
	Accessibility	1	-	1 (100%)	-	-	-	-
Compatibility	Interoperability	3	-	1 (33.3%)	1 (33.3%)	-	-	1 (33.3%)
Functional suitability	Mobility	2	1 (50%)	-	-	-	-	1 (50%)
	Energy efficiency	1	-	-	-	-	-	1 (100%)

Summary. In the mobility and smart city domains, certain quality requirements –such as timeliness, scalability, and security– are often considered together when designing the CoT architecture.

CAQ4: Quality attributes and architectures evaluation method. As depicted in Table 12, the primary studies did not evaluate a majority of quality attributes using a specific methodology. In this regard, certain attributes –such as interoperability, adaptability, mobility, transparency, and accessibility– were not evaluated during even one evaluation methodology. This gap creates an opportunity for future empirical research in the CoT domain. In contrast, scalability and timeliness are two quality attributes that were covered throughout the four evaluation methodologies.

Experimentation was the most commonly used evaluation methodology among the primary studies related to different quality attributes; of these attributes, scalability was studied the most. Prototypes, which were second mostly appeared in evaluations of scalability and timeliness. Simulation was an interesting alternative related to reliability, scalability, timeliness, and performance. Case studies, and the combination of more than one methodology (i.e., hybrids), were the last two methodologies in terms of different quality attributes. These findings indicated that major quality attributes, such as scalability or timeliness, have typically been studies in laboratory settings (i.e., experiment or simulation) rather than ac-

tual industrial cases. This provides an opportunity for future studies to assess the designed architectures within industrial settings.

Table 12: Quality attributes versus evaluation methodology

QA	Subcategory	Total	Evaluation methodology						
			Experiment	Prototype	Illustrative example	Simulation	Case study	Hybrid	Not presented
Portability	Scalability	19	5 (26%)	4 (21%)	2 (10.5%)	2 (10.5%)	-	-	6 (32%)
	Adaptability	2	-	-	-	-	-	-	2 (100%)
Performance efficiency	Timeliness	13	3 (23%)	4 (31%)	1 (8%)	2 (15%)	-	1 (8%)	2 (15%)
	Performance	6	2 (33%)	-	1 (17%)	2 (33%)	-	-	1 (17%)
	Transparency	1	-	-	-	-	-	-	1 (100%)
Security	Security	12	2 (17%)	1 (8%)	-	-	1 (8%)	-	8 (67%)
	Privacy	2	-	-	-	-	1 (50%)	-	1 (50%)
	Confidentiality	1	-	-	-	-	-	-	1 (100%)
Reliability	Reliability	7	1 (14%)	1 (14%)	-	3 (43%)	-	-	2 (29%)
	Availability	3	1 (33.3%)	1 (33.3%)	-	-	-	-	1 (33.3%)
Maintainability	Evolvability	3	1 (33%)	-	-	-	-	-	2 (67%)
	Variability management	2	-	-	1 (50%)	-	-	-	1 (50%)
	Reconfigurability	1	-	-	-	-	-	-	1 (100%)
	Re-usability	1	-	-	-	-	-	-	1 (100%)
	Modularity	1	-	1 (100%)	-	-	-	-	-
Context coverage	Context-awareness	5	1 (20%)	1 (20%)	-	-	-	-	3 (60%)
	Flexibility	1	-	-	-	-	-	-	1 (100%)
Usability	Usability	3	1 (33%)	-	-	-	-	-	2 (67%)
	User-participatory	1	-	-	-	1 (100%)	-	-	-
	Accessibility	1	-	-	-	-	-	-	1 (100%)
Compatibility	Interoperability	3	-	-	-	-	-	-	3 (100%)
Functional suitability	Mobility	2	-	-	-	-	-	-	2 (100%)
	Energy efficiency	1	-	-	-	-	-	-	1 (100%)

Summary. There is a scientific shortage of empirical evaluation for architecture designs in CoT.

CAQ5: Evaluation approach and application area. Table 13 presents the distribution of the evaluation methodologies used in each application area. Considering the large number and scope of CoT application areas, a majority of the primary studies only presented proof of concepts for designed architectures by presenting illustrative examples or prototypes. CoT software architectures in the mobility and smart city domains were evaluated the most, among which CoT architectures in the smart city domain were mainly assessed through laboratory approaches, such as illustrative examples (three papers) and prototypes (two papers).

This finding is similar in the mobility area, as three papers developed prototypes, and two studies used a hybrid methodology (a combination of experimentation and prototype) in this application area. The health care and manufacturing areas used barely any methodologies to evaluate software architectures.

The prototype was used as an evaluation methodology in seven papers within four application areas, although the majority of the papers that evaluated the architectures barely considered a specific application area. For example, none of the simulation studies addressed an application area.

Summary. There is a research need for industrial contributions to evaluate CoT architectures in real-world environments.

Table 13: Evaluation methodology versus application area and CoT architectural design pattern

Evaluation methodology	Total	Application area						CoT architectural design pattern						
		MO	SC	HC	SH	MFG	General	EC	SP	TI	VDR	BC	DI	General
Experiment	16	1 (6.2%)	1 (6.2%)	-	1 (6.2%)	1 (6.2%)	12 (70%)	6 (38%)	4 (25%)	2 (12.5%)	1 (6%)	1 (6%)	-	2 (12.5%)
Prototype	14	3 (21.5%)	2 (14.5%)	1 (7%)	1 (7%)	-	7 (50%)	8 (57%)	5 (36%)	1 (7%)	-	-	-	-
Illustrative example	10	-	3 (30%)	-	-	-	7 (70%)	3 (30%)	2 (20%)	-	2 (20%)	1 (10%)	1 (10%)	1 (10%)
Simulation	4	-	-	-	-	-	4 (100%)	-	1 (25%)	-	2 (50%)	-	-	1 (25%)
Case study	2	1 (50%)	-	-	-	-	1 (50%)	-	1 (50%)	-	-	-	1 (50%)	-
Hybrid	2	2 (100%)	-	-	-	-	-	2 (100%)	-	-	-	-	-	-

CAQ6: Evaluation approach and CoT architectural design pattern. Because the majority of the papers adopted the edge connectivity pattern, we expected that most of the evaluations would be part of this design pattern. Prototypes (eight papers) and experimentation (six papers) were the most frequently adopted approaches to evaluate those architectures that use edge connectivity patterns. Stream processing was second in terms of the number of evaluations that were conducted with respect to CoT architectural design patterns (13 papers), while design patterns, such as device identity or brokered communication, came in last in that regard (two papers each). Our findings could not reveal specific reasons for selecting an evaluation methodology with respect to a CoT design pattern.

Experiments and prototypes were the most commonly adopted methodologies within different CoT architectural design patterns (14 papers each), although experiments and illustrative examples had better coverage within different CoT architectural design patterns. Case studies and hybrid approaches were last in this respect. Table 13 presents areas that still lack empirical research, along with different CoT architectural design patterns.

Summary. Experiment and prototype were the most frequently applied methodologies with respect to different design patterns.

CAQ7: CoT architectural design pattern versus apps pattern for CoT. As Table 14 indicates, the majority of the studies applied the distributed IoT as the CoT application pattern (31 papers). This pattern was applied mostly within edge connectivity (32%), stream processing (19%), and virtual device representation (19%). Further, distributed IoT was the most widely used pattern, along with all CoT architectural design patterns. Examples of these applications are content delivery to vehicles [109], analytics of data collected by mobile devices [11], and environmental monitoring through geographically distributed wireless sensor networks [116].

The second application pattern is social IoT in which edge connectivity (32%) and stream processing (32%) were major CoT patterns. Self-configurable gateways that automatically detect and register new devices facilitate communication among smart devices [114]. The use of the digital twin pattern also seems to be widespread with respect to the CoT design patterns, while cloud-based apps are mostly addressed in edge connectivity patterns (45%). Other CoT architectural design patterns were barely adopted in this regard.

Edge connectivity was used with all application patterns for CoT; among these patterns, the highest share of papers were devoted to distributed IoT (10 papers). Stream processing was the second alternative (16 papers) in terms of different application patterns for CoT, while device identity received the least attention in this regard.

Table 14: CoT architectural design pattern versus application pattern for CoT, architectural style, and view

Apps pattern for CoT	Total	**CoT architectural design pattern**						
		EC	SP	TI	VDR	BC	DI	General
Distributed IoT	31	10 (32%)	6 (19%)	4 (13%)	6 (19%)	1 (3%)	1 (3%)	3 (10%)
Social IoT	19	6 (32%)	6 (32%)	-	2 (11%)	1 (5%)	-	4 (20%)
Cloud-based apps	11	5 (45%)	-	2 (18%)	-	2 (18%)	1 (9%)	1 (9%)
Digital twin	8	1 (12.5%)	3 (37.5%)	1 (12.5%)	1 (12.5%)	2 (25%)	-	-
Asset-based apps	4	2 (50%)	-	-	-	-	1 (25%)	1 (25%)
General	9	4 (44%)	1 (11%)	1 (11%)	-	-	-	3 (33%)
Architectural style	Total	EC	SP	TI	VDR	BC	DI	General
SOA	27	7 (26%)	8 (30%)	2 (7%)	4 (15%)	-	2 (7%)	4 (15%)
Multi-layered	13	3 (23%)	3 (23%)	3 (23%)	1 (8%)	-	-	3 (23%)
Client-server	10	6 (60%)	2 (2%)	1 (10%)	-	-	-	1 (10%)
Publish-subscribe	8	1 (12.5%)	1 (12.5%)	-	1 (12.5%)	2 (25%)	1 (12.5%)	2 (25%)
Agent-based	6	3 (50%)	-	1 (17%)	-	2 (33%)	-	-
Object request broker	2	-	-	-	-	2 (100%)	-	-
Event-driven	2	1 (50%)	-	-	-	-	-	1 (50%)
Pipeline	1	-	1 (100%)	-	-	-	-	-
Plug-in	1	1 (100%)	-	-	-	-	-	-
Lambda	1	-	1 (100%)	-	-	-	-	-
General	11	6 (55%)	-	1 (9%)	3 (27%)	-	-	1 (9%)
View	Total	EC	SP	TI	VDR	BC	DI	General
Development	46	15 (33%)	8 (17%)	4 (9%)	5 (11%)	3 (6.5%)	3 (6.5%)	8 (17%)
Logical	30	9 (30%)	5 (17%)	4 (13%)	5 (17%)	3 (10%)	2 (7%)	2 (7%)
Scenario	29	12 (41%)	6 (21%)	3 (10%)	2 (7%)	1 (3%)	1 (3%)	4 (14%)
Physical	15	10 (67%)	1 (7%)	-	-	-	1 (7%)	3 (20%)
Process	15	7 (47%)	2 (13%)	2 (13%)	1 (7%)	1 (7%)	1 (7%)	1 (7%)
General	4	2 (50%)	1 (25%)	1 (25%)	-	-	-	-

Summary. Edge connectivity supports distributed applications with respect to the attributes, such as mobility.

CAQ8: CoT architectural design pattern versus architectural style. SOA was the most applied architectural style (in terms of number of papers), along with different CoT architectural design patterns, as presented in Table 14. SOA was also widely used in a wide variety of CoT architectural design patterns, with the exception of brokered communication; among these patterns, stream processing (30%) and edge connectivity (26%) were the first two alternatives. Modular cloud services advertised in SOA can be preserved through IoT gateways as the interface between the devices and the cloud for data exchange [122].

The multi-layered and client-server styles were next in terms of the number of papers on different CoT architectural design patterns. The client-server style was only addressed in the first three CoT architectural design patterns. A primary study [123] provided a gateway application for transmitting the data received from the sensors and devices to the server in a client-server architecture.

Edge connectivity was used in a wide variety of architectural styles (except for three); in this sense, it is a popular alternative. As the second choice among CoT architectural design

patterns, stream processing was mostly addressed in papers with the SOA, multi-layered, and client-server software architectural styles.

> Summary. IoT gateways can increase flexibility and mobility in SOA and client-server architectures.

CAQ9: CoT architectural design pattern versus architectural view. Out of the five architectural views, the development view was the most frequently used (46 papers). As indicated in Table 14, this view was the most frequently used, along with edge connectivity (33%) and stream processing (17%). In addition, logical (30 papers) and scenario (29 papers) views were the most frequently used, along with edge connectivity patterns. The physical view was frequently used with edge connectivity patterns (67%). Edge connectivity patterns were the most applied CoT patterns among the different views, followed by stream processing and telemetry ingestion.

> Summary. Development view and scenarios were used by a majority of the CoT architectures along with edge connectivity patterns.

4.3. RQ3. What CoT models have been provided in software architecture literature?

In answering this question, we provided a list of 31 existing CoT models in software architecture literature. It is worth mentioning that this list contains the CoT models that were introduced in the literature and the list is not a reflection of the entire CoT model population. We believe that the extraction of this list will enable developers to access relevant academic research material on these CoT models and aid researchers in identifying the research gap in other existing CoT models that are currently available in the market. The complete list is provided in Appendix C.

> Summary. Thirty-one CoT models were collected from academic literature on software architecture.

5. Discussion

The objective of this research is to identify and synthesise software architecture studies in the domain of CoT. An SLR was conducted to analyse the primary studies according to different aspects, including research intensity, research type, contribution type, and study quality. In addition, we have striven to identify, evaluate, and synthesise the academic knowledge related to various software architectural elements of CoT, including design patterns, styles, views, evaluation methods, and quality attributes.

5.1. Overview of findings and their implications

RQ1. What are the intensity and characteristics of research pertaining to software architectures in the context of CoT? To address this question, we have analysed the information

from primary studies as well as determining the research type, contribution type, and quality (see section 4.1).

The findings indicated that the convergence of cloud computing and IoT has received increasing attention since 2010, so the design of relevant architectures that can be used with different application areas will be necessary. The majority of primary studies were published in conference proceedings (73%), followed by journal articles (22%). The results show that a diversified range of journals and conferences have published academic studies within this domain.

More than half the studies did not empirically evaluate the architectures; they primarily provided architectural representations (including building blocks, interfaces, etc.) without providing relevant scientific evaluations.

In addition, we expect that practices regarding the standardisation of CoT architectures will continue to increase over time because of this technology's increasing importance. Still, the rapid pace of emerging new technologies, such as edge and fog computing, as well as the growth in the size and complexity of IoT applications and the number of connected objects, will all bring further challenges. We assume in this study that future architectures will be driven by increasing demand for scalable and secure CoTs, high-quality systems, communication latency, and better context awareness. As a result, software architecture designers will face added challenges in terms of quality concerns and stakeholder needs, all of which will create interesting opportunities for future studies.

RQ2. What software architectures have been investigated most in the context of CoT? To address this question, a deductive coding approach was used to extract architectural design patterns, styles, views, evaluation methodologies, quality attributes, and CoT application areas. We found that barely any methodological approaches addressed the design of software architectures of CoT. This study has shown that previous researchers in this domain have addressed six design patterns for CoT architectures as well as five application patterns for CoT. Accordingly, the architectures were classified into nine architectural styles and 4+1 architectural views. Our literature analysis has indicated that previous researchers have primarily investigated CoT architectures without considering their application areas; though, smart city and mobility were the most interesting areas in this respect. Section 4.2.4 defined nine cross-analysis questions to create a better understanding of the architecture elements as well as the existing research gaps in this domain.

The term "software architecture" can be used in different levels and contexts [14]. For example, it is possible to talk about the technical roles of software architectures, relevant business impacts [124], or representations of different system hierarchy levels [125]. Because the software architectures of CoT are highly interdependent with other systems, such as communication networks and physical things, the primary studies we examined often had to describe the architectures in relation to other domains, such as software applications, communication networks, and IoT resources, among others.

While the approach and perspective of this research are completely different from those of prior secondary studies, certain relationships were observed between the research questions of a few existing secondary studies and the current SLR. For example, a systematic mapping study's [32] third research question was concerned with the existing architectures

that support the construction and execution of cloud-based IoT systems. Those authors characterised and presented an overview of novel concepts in the architectures of CoT. Due to the focus of that mapping study, which was broadly on the realisation of the integration of IoT and cloud computing, the authors were not able to provide a deep analysis of CoT architecture elements.

In another study [126], the authors discussed the key technologies and architectures in the literature in order to find appropriate alternatives to be used in the development of smart cities. They strove to provide an overview of the previous literature on IoT, SOA, event-driven architectures, and relevant technologies. That paper had a broad scope and was not limited to CoT architectural elements.

RQ3. What CoT models have been provided in software architecture literature? To address this question, we collected existing CoT models provided in the primary studies we examined. We found that very few studies have provided thorough analysis on existing CoT models. We believe that more CoT models should exist, compared to the models we have reported in our study, as many of the existing platforms have not been reported in the scientific literature.

5.2. Implications

With this SLR, we provide a systematic synthesis and classification of the software architecture literature on CoT. We have excluded separate architectural knowledge about IoT or cloud computing systems from this study in order to structure the software architecture knowledge on the convergence of cloud computing and IoT. For academia, this work will provide support to continue with software architecture research and help to fill research gaps in terms of research type, contribution type, and empirical evaluation. Practitioners will have access to state-of-the-art architectures, which can be used in design decisions related to architectural elements, such as design patterns, styles, views, and evaluation methodologies that consider quality concerns and application areas.

5.3. Recommendations for future study

The convergence of cloud computing and IoT is in its early phases of research, and researchers still have many opportunities to investigate and improve on CoT models, building blocks, tools, requirements, technologies, inter-dependencies, etc. The findings indicate that the CoT has received increasing attention from several disciplines, including network providers, hardware suppliers, and software developers as well as from academia and standardisation institutes. This section of the paper summarises our observations about various architectural challenges and gaps that might benefit future studies in this domain.

The IoT and cloud computing have evolved due to a convergence of multiple technologies, including wireless communication, real-time analytics, machine learning, commodity sensors, and embedded systems. This study's findings have shown that software architecture engineering has received less attention than other disciplines in this domain. Consequently, there is a gap for investigating the software architectures of CoT, including the models, quality concerns, tool chains, inter-dependencies, and industrial evaluations. This study shows that advances in software engineering aspects will improve the reliability of architectures

in other domains as well as in the whole CoT system. We were able to find several quality attributes that have been discussed in the literature, yet we found many other quality attributes, such as usability, maintainability, and context awareness, which have received less attention from prior researchers and thus could be discussed more in future research.

Our findings indicate that most of previous studies on the software architectures of CoT have not reported any specific research methodologies. For example, our SLR was unable to find any studies in which scientific methodologies suitable for designing architectures were used. The authors believe that the field still has a need for standardised scientific frameworks and methodologies to lead architectural designs in this domain. Existing knowledge about architectural management in other fields could be adopted and customised in future studies.

Designing large-scale distributed systems, such as CoT, evolves according to changes in the underlying technologies and stakeholder requirements. For example, because new emerging technologies, such as edge or fog computing, work based on collaborations between several infrastructure operators and service providers, it is not always clear who operates and manages the infrastructure. Having billions of connected IoT devices also creates research challenges for multi-cloud architectures that support the IoT.

Architectures, which are important contributors to the success of systems, are usually evaluated based on design decisions, stakeholder requirements, and business success, among other factors. Our findings show that the majority of the primary studies we examined did not provide rigorous frameworks or implications for the evaluation of various architectures. We think that there are still gaps in the validity of the existing CoT models and relevant architectures. Very few studies have provided empirical results that show how the proposed architectures work in the real-world. It will also be necessary to establish frameworks and standards that will guide developers and researchers in selecting and evaluating existing architectures, technologies, design process, and tools. Prior software engineering researchers have developed several methodologies and guidelines to evaluate the architectures of software systems in other domains, although the selection and customisation of existing solutions is always a challenge.

Our findings show that the literature on CoT software architectures has yet to reach maturity. We have identified several shortages in standardisation practices for creating reference architectures that can be adopted and customised within different application areas. Several practices exist for the development of reference architectures (i.e., [127, 128]) that can be adopted and applied in CoT domain. Future academic studies can emphasise collaborations with industrial partners and be driven by the specific needs of stakeholders in different contexts to establish reliable, scalable, and secure architectures in the context of CoT, as previous researchers [129, 130] have already realised this need in the automotive domain.

6. Conclusions

The IoT includes challenges, including low-memory devices, network limitations, poor computational capacity, heterogeneity, ubiquity, and mass scalability, among others. The convergence of cloud computing and IoT has recently received attention from the research

and industrial communities, as this convergence can alleviate these challenges using cloud infrastructure. Many companies demand reliable software architectures in order to handle the quality requirements of new emerging technologies and the complex quality challenges that come with these technologies. This study has presented a systematic review of 82 primary studies (of a total of 1,618 studies) on CoT software architectures. We collected, evaluated, and synthesised the existing architecture knowledge in this domain, including design patterns, styles, views, and evaluation methodologies in terms of various quality attributes and CoT application areas.

We classified the architectural design patterns for CoT into six categories; among these categories, edge connectivity and stream processing were found to be the most adopted options. Our SLR also extracted five application patterns for CoT. Distributed IoT apps and social IoT were the patterns that were applied most in the literature. The architectures were classified according to ten software architectural styles, of which service-oriented architectures were the most popular. Nine cross-analyses were conducted to review the relationship between different architectural elements, quality attributes, and application areas. Our findings concluded with a list of 31 existing CoT models in the academic literature, which are summarised in Appendix C.

Current trends show increasing interest towards the design of scalable CoT technologies to be used in different application areas. We have realised that CoT has received less attention from the software architecture engineering field compared to other disciplines, such as network communications, which highlights several opportunities for further academic study.

Acknowledgements

This research was supported by the ITEA3-APPSTACLE research project and funded by Business Finland.[5]

[5]https://www.businessfinland.fi/en

Appendix A. SLR overview

P #	Study (Ref. no.)	Research type	Contribution type
P1	[99]	Validation	Model
P2	[131]	Validation	Model
P3	[132]	Validation	Model
P4	[13]	Validation	Model
P5	[133]	Solution	Framework/Method/Technique
P6	[118]	Validation	Framework/Method/Technique
P7	[80]	Solution	Model
P8	[88]	Evaluation	Framework/Method/Technique
P9	[114]	Validation	Model
P10	[10]	Solution	Model
P11	[89]	Evaluation	Model
P12	[12]	Validation	Model, Framework/Method/Technique
P13	[11]	Solution	Model
P14	[134]	Solution	Framework/Method/Technique
P15	[87]	Solution	Model
P16	[135]	Validation	Model
P17	[104]	Validation	Model
P18	[136]	Validation	Model, Framework/Method/Technique
P19	[116]	Solution	Model
P20	[119]	Solution	Framework/Method/Technique, Model
P21	[7]	Solution	Model
P22	[113]	Validation	Model
P23	[94]	Solution	Model
P24	[83]	Validation	Model, Framework/Method/Technique
P25	[77]	Validation	Model
P26	[90]	Validation	Framework/Method/Technique
P27	[137]	Solution	Model
P28	[121]	Solution	Model
P29	[95]	Validation	Model
P30	[123]	Validation	Model
P31	[96]	Validation	Model
P32	[97]	Solution	Model, Framework/Method/Technique
P33	[138]	Experience	Framework/Method/Technique
P34	[82]	Solution	Model
P35	[117]	Solution	Model
P36	[85]	Validation	Framework/Method/Technique
P37	[139]	Solution	Model
P38	[140]	Validation	Model
P39	[141]	Solution	Model

P40	[92]	Solution	Model
P41	[100]	Validation	Model
P42	[142]	Solution	Model
P43	[143]	Validation	Model
P44	[98]	Validation	Model
P45	[81]	Validation	Model
P46	[101]	Validation	Model
P47	[91]	Solution	Model
P48	[144]	Solution	Model
P49	[105]	Solution	Model
P50	[79]	Validation	Model
P51	[78]	Validation	Model
P52	[145]	Solution	Model
P53	[84]	Validation	Model
P54	[5]	Solution	Model
P55	[146]	Solution	Model
P56	[120]	Solution	Model
P57	[106]	Validation	Model
P58	[107]	Validation	Model
P59	[3]	Solution	Model
P60	[2]	Solution	Model
P61	[147]	Solution	Model
P62	[110]	Solution	Model
P63	[76]	Solution	Model
P64	[108]	Validation	Framework/Method/Technique, Model
P65	[148]	Solution	Model
P66	[115]	Validation	Model
P67	[111]	Solution	Model
P68	[149]	Validation	Framework/Method/Technique
P69	[102]	Validation	Framework/Method/Technique
P70	[8]	Solution	Model
P71	[9]	Solution	Model
P72	[122]	Solution	Model
P73	[86]	Solution	Model
P74	[103]	Solution	Framework/Method/Technique
P75	[150]	Solution	Model
P76	[93]	Experience	Model
P77	[151]	Solution	Model, Advice/Implications
P78	[112]	Solution	Model
P79	[152]	Validation	Framework/Method/Technique, Model
P80	[109]	Validation	Framework/Method/Technique
P81	[153]	Validation	Model
P82	[154]	Solution	Model

Appendix B. Data properties

ID	Data property	Description
DP1	Publication year	Refers to the publication year of the primary study
DP2	Publication source	Refers to publication source, including journal articles, conference proceedings, or workshop proceedings
DP3	Research type; adopted from [57]	• *Solution proposal:* proposes a solution or technique to a particular problem, either novel or a significant improvement of an existing solution, without full validation. • *Validation research:* a novel solution or technique is proposed but not yet implemented in practice. It uses a research approach, such as an experiment, prototyping, formal analysis, simulation, or similar approaches. • *Evaluation research:* investigates a solution or technique in practice, and its evaluation is conducted accordingly. • *Experience report:* reflects the personal industrial experiences of the authors.
DP4	Contribution type; adopted from [58, 59]	• *Framework/method/technique:* proposes a particular framework, method, or technique for CoT software architectures. • *Model:* a representation of an observed reality in concepts or related concepts after a conceptualisation process. • *Advice/implication:* a discursive and generic recommendation based on the personal opinion of the authors
DP5	Quality assessment; adopted from [33]	Each individual study is evaluated according to the quality criteria described in section 3.4.2 (ranging from Yes = 1, Partially = 0.5, and No = 0)

DP6	Software architecture representation elements	Identifies the patterns, styles, and views of software architectures in the context of CoT
DP7	Quality attributes	Identifies the major quality attributes in designing CoT software architectures
DP8	Architecture evaluation methods	Identifies the methods applied to evaluate CoT software architectures
DP9	Application area	Identifies the application areas of CoTs
DP10	Existing CoT models	Summarises the existing CoT models reported in the literature

Appendix C. Existing CoT models

Name	Description	Primary studies
Aneka	A .NET-based application development Platform-as-a-Service (PaaS), which can utilise storage and compute resources of both public and private clouds	[5]
Atlas	An IoT platform that consists of sensor node, hardware or software platform, and service gateway framework	[100]
AWS IoT	Amazon Web Services IoT is a managed cloud platform for the IoT that lets connected devices easily and securely interact with cloud applications and other devices	[122]
Axeda	An IoT cloud platform providing connectivity between devices and objects to enable application services, integration framework, and data management	[95, 141]
CenceMe	A mobile phone sensing system that combines the inference of the presence of individuals using sensor-enabled mobile phones with sharing of this information through social networking applications	[120]
CloudThings	A service platform, developer suite, and operating portal that works over the Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a service (SaaS)	[86]
DARWIN	A cloud service system for automobiles that contains key service components interacting with various services both inside and outside of vehicles to form a comprehensive vehicular cloud	[111]
DIMMER	A service IoT cloud platform aiming at involving different stakeholders to increase the energy efficiency of a city	[93]
Etherios (formerly iDigi)	A hosted device cloud solution that enables the creation of apps that can control devices in real time, schedule operations, and configure alarms	[95, 141, 142]

FIWARE	A platform for the future Internet that would provide a novel service infrastructure built of reusable components (generic enablers)	[93, 138, 122, 154]
FutureGrid	A geographically distributed and heterogeneous cloud test bed	[104]
GSN	Uses virtual sensors to control processing priorities and the management of resources and stored data. By using declarative specifications, virtual sensors can be deployed and reconfigured in GSN containers at runtime	[141, 145]
Hive	An edge-based middleware architecture and protocol to enable heterogeneous edge devices to dynamically share data and resources for enhanced application performance and privacy	[79]
Hourglass	An Internet-based infrastructure for connecting a wide range of sensors, services, and applications in a robust fashion	[141]
iCOMOT	A set of tools and services that simplify the management of such sensors, gateways, and services	[151]
ICWIOT	Intelligent city with an IoT service platform	[12]
IoTCloud	A platform that controls and manages sensors and messages online over the cloud with different modules, e.g., controller, message broker, and sensors	[86]
Kiuas	A cloud environment to make software development for large and highly dynamic topologies of IoT devices as effortless as possible by abstracting away complexities related to connectivity, asynchronous device communication, and physical device location	[142]
LSM2	A platform that bridges the live real-world sensed data and Semantic Web functionalities, such as wrappers for real-time data collection and publishing	[145]
MOSDEN	Supports sensing as a service and is built on top of GSN. MOSDEN improves the scalability and user friendliness of middleware, since plugins for heterogeneous devices are easier to build	[98]
Nimbits	An open source data logging cloud server that provides connectivity between the IoT using data points	[141, 142]
OneM2M	A telecom initiative for interoperability of M2M and IoT devices and applications to develop a common specification of a service layer platform that builds on the existing IoT and Web standards, defining specifications of protocols and service APIs	[93]

OpenIoT	An open-source middleware for IoT applications in a Sensing as a Service model (SaaS), which is available in a cloud environment that can be transparently accessed and configured by users	[86, 145, 148]
SensorCloud	Sensor features include data storage and visualisation and a remote management platform that leverages powerful cloud computing technologies to provide excellent data scalability, rapid visualisation, and user-programmable analysis	[141, 145]
SOCRADES	A middleware that abstracts physical things as services using Devices Profile for WS (DPWS). Its architecture consists of a layer for application services and a layer for device services	[100]
Stack4Things	An infrastructure-oriented two-layer approach that manages policies at the control plane while coping with communication requirements and scalability concerns at the data plane by leveraging cloud-focused design choices and architectural patterns	[144]
ThingSpeak	An open-source IoT application and API for storing and retrieving data from things; ThingSpeak uses HTTP over the Internet or via a local area network	[142]
ThingStore	A platform to bring together the different actors of IoT's cyber-physical environment, such as thing providers, software developers, and end users	[78]
UbiFit Garden	A mobile phone sensing system that uses small, inexpensive on-body sensors and machine learning techniques for activity modelling	[120]
VTrack	A mobile sensing system that tracks traffic delays and congestion	[120]
Xively	a Platform as a Service (PaaS) that provides middleware services to create products and solutions for IoT	[95, 141, 145]

References

[1] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Computer Networks 54 (15) (2010) 2787–2805.

[2] B. P. Rao, P. Saluia, N. Sharma, A. Mittal, S. V. Sharma, Cloud computing for internet of things & sensing based applications, in: Sensing Technology (ICST), 2012 6th International Conference on, IEEE, 2012, pp. 374–380.

[3] L. Liu, X. Liu, X. Li, Cloud-based service composition architecture for internet of things, in: Internet of Things, Springer, 2012, pp. 559–564.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., A view of cloud computing, Communications of the ACM 53 (4) (2010) 50–58.

[5] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, Internet of things (iot): A vision, architectural elements, and future directions, Future Generation Computer Systems 29 (7) (2013) 1645–1660.

[6] P. P. Ray, A survey of iot cloud platforms, Future Computing and Informatics Journal 1 (1-2) (2016) 35–46.
[7] A. Celesti, M. Fazio, M. Giacobbe, A. Puliafito, M. Villari, Characterizing cloud federation in iot, in: Advanced Information Networking and Applications Workshops (WAINA), 2016 30th International Conference on, IEEE, 2016, pp. 93–98.
[8] S. Distefano, G. Merlino, A. Puliafito, Enabling the cloud of things, in: Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 6th International Conference on, IEEE, 2012, pp. 858–863.
[9] F. Bonomi, R. Milito, P. Natarajan, J. Zhu, Fog computing: A platform for internet of things and analytics, in: Big data and internet of things: A roadmap for smart environments, Springer, 2014, pp. 169–186.
[10] M. H. Syed, E. B. Fernandez, M. Ilyas, A pattern for fog computing, in: Proceedings of the 10th Travelling Conference on Pattern Languages of Programs, ACM, 2016, p. 13.
[11] M. S. de Brito, S. Hoque, T. Magedanz, R. Steinke, A. Willner, D. Nehls, O. Keils, F. Schreiner, A service orchestration architecture for fog-enabled infrastructures, in: Fog and Mobile Edge Computing (FMEC), 2017 2nd International Conference on, IEEE, 2017, pp. 127–132.
[12] Y. Fu, S. Jia, J. Hao, A scalable cloud for internet of things in smart cities, Journal of Computers 26 (3).
[13] D. Kelaidonis, A. Rouskas, V. Stavroulaki, P. Demestichas, P. Vlacheas, A federated edge cloud-iot architecture, in: Networks and Communications (EuCNC), 2016 European Conference on, IEEE, 2016, pp. 230–234.
[14] L. Bass, P. Clements, R. Kazman, Software architecture in practice, 3rd edition, Pearson Education, 2013.
[15] A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of things for smart cities, IEEE Internet of Things journal 1 (1) (2014) 22–32.
[16] H. Kopetz, Internet of things, in: Real-time systems, Springer, 2011, pp. 307–323.
[17] M. Zorzi, A. Gluhak, S. Lange, A. Bassi, From today's intranet of things to a future internet of things: a wireless-and mobility-related view, IEEE Wireless communications 17 (6) (2010) 44–51.
[18] G. Kortuem, F. Kawsar, V. Sundramoorthy, D. Fitton, et al., Smart objects as building blocks for the internet of things, IEEE Internet Computing 14 (1) (2009) 44–51.
[19] D. Uckelmann, M. Harrison, F. Michahelles, An architectural approach towards the future internet of things, in: Architecting the internet of things, Springer, 2011, pp. 1–24.
[20] C. Cecchinel, M. Jimenez, S. Mosser, M. Riveill, An architecture to support the collection of big data in the internet of things, in: 2014 IEEE World Congress on Services, IEEE, 2014, pp. 442–449.
[21] D. Miorandi, S. Sicari, F. De Pellegrini, I. Chlamtac, Internet of things: Vision, applications and research challenges, Ad hoc networks 10 (7) (2012) 1497–1516.
[22] A. Botta, W. De Donato, V. Persico, A. Pescapé, Integration of cloud computing and internet of things: A survey, Future Generation Computer Systems 56 (2016) 684–700.
[23] P. Mell, T. Grance, et al., The nist definition of cloud computing, Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology Gaithersburg.
[24] H. Cervantes, R. Kazman, Designing software architectures: A practical approach, Addison-Wesley Professional, 2016.
[25] S. Madakam, R. Ramaswamy, S. Tripathi, Internet of things (iot): A literature review, Journal of Computer and Communications 3 (05) (2015) 164.
[26] L. Da Xu, W. He, S. Li, Internet of things in industries: A survey, IEEE Transactions on Industrial Informatics 10 (4) (2014) 2233–2243.
[27] A. Whitmore, A. Agarwal, L. Da Xu, The internet of things: A survey of topics and trends, Information Systems Frontiers 17 (2) (2015) 261–274.
[28] P. Hoberg, J. Wollersheim, H. Krcmar, The business perspective on cloud computing: A literature review of research on cloud computing, AMCIS 2012 Proceedings.
[29] B. P. Rimal, E. Choi, I. Lumb, A taxonomy and survey of cloud computing systems, INC, IMS and

IDC, 2009. NCM'09. 5th International Joint Conference on (2009) 44–51.
[30] H. T. Dinh, C. Lee, D. Niyato, P. Wang, A survey of mobile cloud computing: Architecture, applications, and approaches, Wireless Communications and Mobile Computing 13 (18) (2013) 1587–1611.
[31] P. Jamshidi, A. Ahmad, C. Pahl, Cloud migration research: A systematic review, IEEE Transactions on Cloud Computing 1 (2) (2013) 142–157.
[32] E. Cavalcante, J. Pereira, M. P. Alves, P. Maia, R. Moura, T. Batista, F. C. Delicato, P. F. Pires, On the interplay of internet of things and cloud computing: A systematic mapping study, Computer Communications 89 (2016) 17–33.
[33] B. Kitchenham, S. M. Charters, Guidelines for performing systematic literature reviews in software engineering, in: Technical Report EBSE-2007-01, sn, 2007, pp. 1–57.
[34] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: International Conference on Evaluation and Assessment in Software Engineering (EASE), Vol. 8, 2008, pp. 68–77.
[35] D. R. van Solingen, E. W. Berghout, The Goal/Question/Metric Method: a practical guide for quality improvement of software development, McGraw-Hill, 1999.
[36] F. Van Latum, R. Van Solingen, M. Oivo, B. Hoisl, D. Rombach, G. Ruhe, Adopting gqm based measurement in an industrial environment, IEEE software 15 (1) (1998) 78–86.
[37] H. Zhang, M. A. Babar, P. Tell, Identifying relevant studies in software engineering, Information and Software Technology 53 (6) (2011) 625–637.
[38] K. Petersen, S. Vakkalanka, L. Kuzniarz, Guidelines for conducting systematic mapping studies in software engineering: An update, Information and Software Technology 64 (2015) 1–18.
[39] J. L. Fleiss, Measuring nominal scale agreement among many raters, Psychological Bulletin 76 (5) (1971) 378.
[40] J. R. Landis, G. G. Koch, The measurement of observer agreement for categorical data, Biometrics (1977) 159–174.
[41] T. Dybå, T. Dingsoyr, G. K. Hanssen, Applying systematic reviews to diverse study types: An experience report, in: Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on, IEEE, 2007, pp. 225–234.
[42] B. Kitchenham, P. Brereton, A systematic review of systematic review process research in software engineering, Information and Software Technology 55 (12) (2013) 2049–2075.
[43] B. Kitchenham, D. Budgen, O. P. Brereton, Using mapping studies as the basis for further research: A participant-observer case study, Information and Software Technology 53 (6) (2011) 638–651.
[44] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: International Conference on Evaluation and Assessment in Software Engineering (EASE), ACM, 2014, p. 38.
[45] D. Badampudi, C. Wohlin, K. Petersen, Experiences from using snowballing and database searches in systematic literature studies, in: Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, ACM, 2015, p. 17.
[46] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in software engineering, Springer Science & Business Media, 2012.
[47] D. S. Cruzes, T. Dybå, Recommended steps for thematic synthesis in software engineering, in: Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on, IEEE, 2011, pp. 275–284.
[48] T. Dybå, T. Dingsøyr, Empirical studies of agile software development: A systematic review, Information and software technology 50 (9-10) (2008) 833–859.
[49] C. Wohlin, A. Aurum, Towards a decision-making structure for selecting a research design in empirical software engineering, Empirical Software Engineering 20 (6) (2015) 1427–1455.
[50] M. Brhel, H. Meth, A. Maedche, K. Werder, Exploring principles of user-centered agile software development: A literature review, Information and software technology 61 (2015) 163–181.
[51] H. Zhang, M. A. Babar, Systematic reviews in software engineering: An empirical investigation, Information and Software Technology 55 (7) (2013) 1341–1354.

[52] F. C. Zamawe, The implication of using nvivo software in qualitative data analysis: Evidence-based reflections, Malawi Medical Journal 27 (1) (2015) 13–15.
[53] P. Bazeley, K. Jackson, Perspectives: qualitative computing and nvivo, Qualitative data analysis with Nvivo, Sage, London (2013) 1–46.
[54] C. Wohlin, P. Runeson, P. A. Neto, E. Engström, I. do Carmo Machado, E. S. De Almeida, On the reliability of mapping studies in software engineering, Journal of Systems and Software 86 (10) (2013) 2594–2610.
[55] B. Kitchenham, T. Dyba, M. Jorgensen, Evidence-based software engineering, in: Proceedings of the 26th International Conference on Software Engineering (ICSE), IEEE Computer Society, 2004, pp. 273–281.
[56] B. Kitchenham, P. Brereton, Z. Li, D. Budgen, A. Burn, Repeatability of systematic literature reviews, in: International Conference on Evaluation and Assessment in Software Engineering (EASE), IET, 2011, pp. 46–55.
[57] R. Wieringa, N. Maiden, N. Mead, C. Rolland, Requirements engineering paper classification and evaluation criteria: A proposal and a discussion, Requirements Engineering 11 (1) (2006) 102–107.
[58] M. Shaw, Writing good software engineering research papers, in: Software Engineering, 2003. Proceedings. 25th International Conference on, IEEE, 2003, pp. 726–736.
[59] N. Paternoster, C. Giardino, M. Unterkalmsteiner, T. Gorschek, P. Abrahamsson, Software development in startup companies: A systematic mapping study, Information and Software Technology 56 (10) (2014) 1200–1218.
[60] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, I. Meedeniya, Software architecture optimization methods: A systematic literature review, IEEE Transactions on Software Engineering 39 (5) (2013) 658–683.
[61] ISO/IEC-42010, Systems and software engineering: Recommended practice for architectural description of software-intensive systems (2007).
[62] N. Medvidovic, R. N. Taylor, Software architecture: Foundations, theory, and practice, in: Proceedings of the 32nd International Conference on Software Engineering (ICSE), ACM, 2010, pp. 471–472.
[63] T. B. C. Arias, P. America, P. Avgeriou, Defining execution viewpoints for a large and complex software-intensive system, in: Software Architecture, 2009 & European Conference on Software Architecture. WICSA/ECSA 2009. Joint Working IEEE/IFIP Conference on, IEEE, 2009, pp. 1–10.
[64] P. B. Kruchten, The 4+ 1 view model of architecture, IEEE Software 12 (6) (1995) 42–50.
[65] N. Rozanski, E. Woods, Software systems architecture: Working with stakeholders using viewpoints and perspectives, Addison-Wesley, 2012.
[66] T. Erl, Service-oriented architecture: Concepts, technology, and design, Pearson Education India, 2005.
[67] S. Oprea, B. G. Tudorica, A. Belciu, I. Botha, Internet of things, challenges for demand side management, Informatica Economica 21 (4) (2017) 59–72.
[68] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, F. Sui, Digital twin-driven product design, manufacturing and service with big data, The International Journal of Advanced Manufacturing Technology 94 (9-12) (2018) 3563–3576.
[69] L. Atzori, A. Iera, G. Morabito, M. Nitti, The social internet of things (siot)–when social networks meet the internet of things: Concept, architecture and network characterization, Computer networks 56 (16) (2012) 3594–3608.
[70] V. Denner, Why a Bosch IoT Cloud?, `https://blog.bosch-si.com/digital-transformation/why-a-bosch-iot-cloud/#` (2016).
[71] A. Akbar, F. Carrez, K. Moessner, J. Sancho, J. Rico, Context-aware stream processing for distributed iot applications, in: 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT), IEEE, 2015, pp. 663–668.
[72] M. Kranz, L. Roalter, F. Michahelles, Things that twitter: social networks and the internet of things, in: What can the Internet of Things do for the Citizen (CIoT) Workshop at The Eighth International Conference on Pervasive Computing (Pervasive 2010), 2010, pp. 1–10.

[73] F. Tao, F. Sui, A. Liu, Q. Qi, M. Zhang, B. Song, Z. Guo, S. C.-Y. Lu, A. Nee, Digital twin-driven product design framework, International Journal of Production Research (2018) 1–19.
[74] B. Afzal, M. Umair, G. A. Shah, E. Ahmed, Enabling iot platforms for social iot applications: vision, feature mapping, and challenges, Future Generation Computer Systems 92 (2019) 718–731.
[75] ISO/IEC-25010, Systems and software engineering – systems and software quality requirements and evaluation (square) – system and software quality models (2011).
[76] M. Aazam, E.-N. Huh, Fog computing: The cloud-iot/ioe middleware paradigm, IEEE Potentials 35 (3) (2016) 40–44.
[77] J. Al-Jaroodi, N. Mohamed, I. Jawhar, S. Mahmoud, Cotware: A cloud of things middleware, in: Distributed Computing Systems Workshops (ICDCSW), 2017 IEEE 37th International Conference on, IEEE, 2017, pp. 214–219.
[78] K. Akpinar, K. A. Hua, Thingstore-an internet of things management system, in: Multimedia Big Data (BigMM), 2017 IEEE 3rd International Conference on, IEEE, 2017, pp. 354–361.
[79] A. Essameldin, K. A. Harras, The hive: An edge-based middleware solution for resource sharing in the internet of things, in: Proceedings of the 3rd Workshop on Experiences with the Design and Implementation of Smart Objects, ACM, 2017, pp. 13–18.
[80] A. Carrega, M. Repetto, P. Gouvas, A. Zafeiropoulos, A middleware for mobile edge computing, IEEE Cloud Computing 4 (4) (2017) 26–37.
[81] A. M. Khan, F. Freitag, On edge cloud service provision with distributed home servers, in: Cloud Computing Technology and Science (CloudCom), 2017 IEEE International Conference on, IEEE, 2017, pp. 223–226.
[82] S. K. Datta, C. Bonnet, J. Haerri, Fog computing architecture to enable consumer centric internet of things services, in: Consumer Electronics (ISCE), 2015 IEEE International Symposium on, IEEE, 2015, pp. 1–2.
[83] Y.-C. Chen, Y.-C. Chang, C.-H. Chen, Y.-S. Lin, J.-L. Chen, Y.-Y. Chang, Cloud-fog computing for information-centric internet-of-things applications, in: Applied System Innovation (ICASI), 2017 International Conference on, IEEE, 2017, pp. 637–640.
[84] W.-S. Kim, S.-H. Chung, User-participatory fog computing architecture and its management schemes for improving feasibility, IEEE Access 6, 2018.
[85] S. K. Datta, R. P. F. Da Costa, J. Härri, C. Bonnet, Integrating connected vehicles in internet of things ecosystems: Challenges and solutions, in: World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2016 IEEE 17th International Symposium on A, IEEE, 2016, pp. 1–6.
[86] A. Sharma, T. Goyal, E. S. Pilli, A. P. Mazumdar, M. Govil, R. C. Joshi, A secure hybrid cloud enabled architecture for internet of things, in: Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on, IEEE, 2015, pp. 274–279.
[87] S. Sotiriadis, E. G. Petrakis, S. Covaci, P. Zampognaro, E. Georga, C. Thuemmler, An architecture for designing future internet (fi) applications in sensitive domains: Expressing the software to data paradigm by utilizing hybrid cloud technology, in: Bioinformatics and Bioengineering (BIBE), 2013 IEEE 13th International Conference on, IEEE, 2013, pp. 1–6.
[88] S.-L. Chen, Y.-Y. Chen, C. Hsu, A new approach to integrate internet-of-things and software-as-a-service model for logistic systems: A case study, Sensors 14 (4) (2014) 6144–6164.
[89] I. D. Addo, S. I. Ahamed, S. S. Yau, A. Buduru, A reference architecture for improving security and privacy in internet of things applications, in: Mobile Services (MS), 2014 IEEE International Conference on, IEEE, 2014, pp. 108–115.
[90] C. K. Dehury, P. K. Sahoo, Design and implementation of a novel service management framework for iot devices in cloud, Journal of Systems and Software 119 (2016) 149–161.
[91] A.-L. Kor, M. Yanovsky, C. Pattinson, V. Kharchenko, Smart-item: Iot-enabled smart living, in: Future Technologies Conference (FTC), IEEE, 2016, pp. 739–749.
[92] J. Mohammed, C.-H. Lung, A. Ocneanu, A. Thakral, C. Jones, A. Adler, Internet of things: Remote patient monitoring using web services and cloud computing, in: Internet of Things (iThings), 2014 IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE

and Cyber, Physical and Social Computing (CPSCom), IEEE, 2014, pp. 256–263.
[93] A. Krylovskiy, M. Jahn, E. Patti, Designing a smart city internet of things platform with microservice architecture, in: Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on, IEEE, 2015, pp. 25–30.
[94] D. Pizzolli, G. Cossu, D. Santoro, L. Capra, C. Dupont, D. Charalampos, F. De Pellegrini, F. Antonelli, S. Cretti, Cloud4iot: A heterogeneous, distributed and autonomic cloud platform for the iot, in: Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on, IEEE, 2016, pp. 476–479.
[95] M. Z. Bjelica, N. Ignjatov, I. Papp, N. Teslic, Device cloud platform with script based agents for "anywhere access" applications development, in: Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on, IEEE, 2014, pp. 1061–1065.
[96] E. Yigitoglu, L. Liu, M. Looper, C. Pu, Distributed orchestration in large-scale iot systems, in: Internet of Things (ICIOT), 2017 IEEE International Congress on, IEEE, 2017, pp. 58–65.
[97] F. Li, M. Vögler, M. Claeßens, S. Dustdar, Efficient and scalable iot service delivery on cloud, in: Cloud Computing (CLOUD), 2013 IEEE 6th International Conference on, IEEE, 2013, pp. 740–747.
[98] C. Perera, P. P. Jayaraman, A. Zaslavsky, D. Georgakopoulos, P. Christen, Mosden: An internet of things middleware for resource constrained mobile devices, in: System Sciences (HICSS), 2014 47th Hawaii International Conference on, IEEE, 2014, pp. 1053–1062.
[99] M. O. Gökalp, A. Koçyigit, P. E. Eren, A cloud based architecture for distributed real time processing of continuous queries, in: Software Engineering and Advanced Applications (SEAA), 2015 41st Euromicro Conference on, IEEE, 2015, pp. 459–462.
[100] J. Im, S. Kim, D. Kim, Iot mashup as a service: Cloud-based mashup service for the internet of things, in: Services Computing (SCC), 2013 IEEE International Conference on, IEEE, 2013, pp. 462–469.
[101] D. Serrano, T. Baldassarre, E. Stroulia, Real-time traffic-based routing, based on open data and open-source software, in: Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on, IEEE, 2016, pp. 661–665.
[102] F. Khodadadi, R. N. Calheiros, R. Buyya, A data-centric framework for development and deployment of internet of things applications in clouds, in: Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE 10th International Conference on, IEEE, 2015, pp. 1–6.
[103] N. Alhakbani, M. M. Hassan, M. A. Hossain, M. Alnuem, A framework of adaptive interaction support in cloud-based internet of things (iot) environment, in: International Conference on Internet and Distributed Computing Systems, Springer, 2014, pp. 136–146.
[104] G. C. Fox, S. Kamburugamuve, R. D. Hartman, Architecture and measured characteristics of a cloud based internet of things, in: Collaboration Technologies and Systems (CTS), 2012 International Conference on, IEEE, 2012, pp. 6–12.
[105] L. Thames, D. Schaefer, Software-defined cloud manufacturing for industry 4.0, Procedia CIRP 52 (2016) 12–17.
[106] R. Jalali, K. El-Khatib, C. McGregor, Smart city architecture for community level services through the internet of things, in: Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on, IEEE, 2015, pp. 108–113.
[107] H. Gu, Y. Diao, W. Liu, X. Zhang, The design of smart home platform based on cloud computing, in: Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on, Vol. 8, IEEE, 2011, pp. 3919–3922.
[108] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, D. Savio, Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services, IEEE transactions on Services Computing 3 (3) (2010) 223–235.
[109] S. Nastic, M. Vögler, C. Inzinger, H.-L. Truong, S. Dustdar, Rtgovops: A runtime framework for governance in large-scale software-defined iot cloud systems, in: Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2015 3rd IEEE International Conference on, IEEE, 2015, pp. 24–33.
[110] J. Wang, J. Cho, S. Lee, T. Ma, Real time services for future cloud computing enabled vehicle networks, in: Wireless Communications and Signal Processing (WCSP), 2011 International Conference on, IEEE,

2011, pp. 1–5.
[111] W. He, G. Yan, L. Da Xu, Developing vehicular data cloud services in the iot environment, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1587–1595.
[112] C. Formisano, D. Pavia, L. Gurgen, T. Yonezawa, J. A. Galache, K. Doguchi, I. Matranga, The advantages of iot and cloud applied to smart cities, in: Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on, IEEE, 2015, pp. 325–332.
[113] D. Seo, C.-S. Jeong, Y.-B. Jeon, K.-H. Lee, Cloud infrastructure for ubiquitous m2m and iot environment mobile application, Cluster Computing 18 (2) (2015) 599–608.
[114] D. Chen, G. Chang, L. Jin, X. Ren, J. Li, F. Li, A novel secure architecture for the internet of things, in: Genetic and Evolutionary Computing (ICGEC), 2011 5th International Conference on, IEEE, 2011, pp. 311–314.
[115] T. D. P. Bai, S. A. Rabara, Design and development of integrated, secured and intelligent architecture for internet of things and cloud computing, in: Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on, IEEE, 2015, pp. 817–822.
[116] G. Suciu, V. Suciu, A. Martian, R. Craciunescu, A. Vulpe, I. Marcu, S. Halunga, O. Fratu, Big data, internet of things and cloud convergence: An architecture for secure e-health applications, Journal of Medical Systems 39 (11) (2015) 141.
[117] B. Manate, T.-F. Fortis, V. Negru, Infrastructure management support in a multi-agent architecture for internet of things, in: Modelling Symposium (EMS), 2014 European, IEEE, 2014, pp. 372–377.
[118] D. Yuan, J. Jin, J. Grundy, Y. Yang, A framework for convergence of cloud services and internet of things, in: Computer Supported Cooperative Work in Design (CSCWD), 2015 IEEE 19th International Conference on, IEEE, 2015, pp. 349–354.
[119] F. Anon, V. Navarathinarasah, M. Hoang, C.-H. Lung, Building a framework for internet of things and cloud computing, in: Internet of Things (iThings), 2014 IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE and Cyber, Physical and Social Computing (CPSCom), IEEE, 2014, pp. 132–139.
[120] X. Sheng, J. Tang, X. Xiao, G. Xue, Sensing as a service: Challenges, solutions and future directions, IEEE Sensors Journal 13 (10) (2013) 3733–3741.
[121] E. Simmon, S. K. Sowe, K. Zettsu, Designing a cyber-physical cloud computing architecture, IT Professional 17 (3) (2015) 40–45.
[122] S. Balampanis, S. Sotiriadis, E. G. Petrakis, Internet of things architecture for enhanced living environments, IEEE Cloud Computing 3 (6) (2016) 28–34.
[123] G. Suciu, A. Scheianu, M. Vochin, Disaster early warning using time-critical iot on elastic cloud workbench, in: Communications and Networking (BlackSeaCom), 2017 IEEE International Black Sea Conference on, 2017.
[124] L. Bass, I. Weber, L. Zhu, DevOps: A software architect's perspective, Addison-Wesley Professional, 2015.
[125] M. Staron, Automotive software architectures, Springer, 2017.
[126] C. Kyriazopoulou, Smart city technologies and architectures: A literature review, in: Smart Cities and Green ICT Systems (SMARTGREENS), 2015 International Conference on, IEEE, 2015, pp. 1–12.
[127] M. Bauer, M. Boussard, N. Bui, J. De Loof, C. Magerkurth, S. Meissner, A. Nettsträter, J. Stefa, M. Thoma, J. W. Walewski, Iot reference architecture, in: Enabling Things to Talk, Springer, 2013, pp. 163–211.
[128] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, L. Badger, D. Leaf, Nist cloud computing reference architecture, NIST special publication 500 (2011) 1–28.
[129] A. Haghighatkhah, A. Banijamali, O.-P. Pakanen, M. Oivo, P. Kuvaja, Automotive software engineering: A systematic mapping study, Journal of Systems and Software 128 (2017) 25–55.
[130] A. Haghighatkhah, M. Oivo, A. Banijamali, P. Kuvaja, Improving the state of automotive software engineering, IEEE Software 34 (5) (2017) 82–86.
[131] G. Baruffa, M. Femminella, M. Pergolesi, G. Reali, A cloud computing architecture for spectrum sensing as a service, in: Cloudification of the Internet of Things (CIoT), IEEE, 2016, pp. 1–5.

[132] G. Di Orio, D. Barata, A. Rocha, J. Barata, A cloud-based infrastructure to support manufacturing resources composition, in: Computing, Electrical and Industrial Systems, Doctoral Conference on, Springer, 2015, pp. 82–89.
[133] X. Ye, J. Huang, A framework for cloud-based smart home, in: Computer Science and Network Technology (ICCSNT), 2011 International Conference, Vol. 2, IEEE, 2011, pp. 894–897.
[134] S. Bhatt, F. Patwa, R. Sandhu, An access control framework for cloud-enabled wearable internet of things, in: Collaboration and Internet Computing (CIC), 2017 3rd International Conference on, 2017.
[135] L. Belli, S. Cirani, L. Davoli, L. Melegari, M. Mónton, M. Picone, An open-source cloud architecture for big stream iot applications, in: Interoperability and Open-Source Solutions for the Internet of Things, Springer, 2015, pp. 73–88.
[136] J. Fiosina, M. Fiosins, J. P. Müller, Big data processing and mining for next generation intelligent transportation systems, Jurnal Teknologi 63 (3) (2013) 23–38.
[137] M. Yang, M. Mahmood, X. Zhou, S. Shafaq, L. Zahid, Design and implementation of cloud platform for intelligent logistics in the trend of intellectualization, China Communications 14 (10) (2017) 180–191.
[138] M. Fazio, A. Celesti, F. G. Márquez, A. Glikson, M. Villari, Exploiting the fiware cloud platform to develop a remote patient monitoring system, in: Computers and Communication (ISCC), 2015 IEEE Symposium on, IEEE, 2015, pp. 264–270.
[139] G. Fortino, A. Guerrieri, W. Russo, C. Savaglio, Integration of agent-based and cloud computing for the smart objects-oriented iot, in: Computer Supported Cooperative Work in Design (CSCWD), Proceedings of the 2014 IEEE 18th International Conference on, IEEE, 2014, pp. 493–498.
[140] W. Na, Internet of things based on the cloud computing architecture, in: Measuring Technology and Mechatronics Automation (ICMTMA), 2015 7th International Conference on, IEEE, 2015, pp. 585–587.
[141] K. Misura, M. Zagar, Internet of things cloud mediator platform, in: Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on, IEEE, 2014, pp. 1052–1056.
[142] P. Selonen, A. Taivalsaari, Kiuas: Iot cloud environment for enabling the programmable world, in: Software Engineering and Advanced Applications (SEAA), 2016 42th Euromicro Conference on, IEEE, 2016, pp. 250–257.
[143] A. Ghose, C. Bhaumik, D. Das, A. K. Agrawal, Mobile healthcare infrastructure for home and small clinic, in: Proceedings of the 2nd ACM International Workshop on Pervasive Wireless Healthcare, ACM, 2012, pp. 15–20.
[144] G. Merlino, D. Bruneo, F. Longo, A. Puliafito, S. Distefano, Software defined cities: A novel paradigm for smart cities through iot clouds, in: Ubiquitous Intelligence and Computing and 2015 IEEE 12th International Conference on Autonomic and Trusted Computing and 2015 IEEE 15th International Conference on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom), 2015 IEEE 12th International Conference, IEEE, 2015, pp. 909–916.
[145] R. Petrolo, V. Loscri, N. Mitton, Towards a smart city based on cloud of things, in: Proceedings of the 2014 ACM International Workshop on Wireless and Mobile Technologies for Smart Cities, ACM, 2014, pp. 61–66.
[146] C. Perera, A. Zaslavsky, P. Christen, D. Georgakopoulos, Sensing as a service model for smart cities supported by internet of things, Transactions on Emerging Telecommunications Technologies 25 (1) (2014) 81–93.
[147] A. Alshehri, R. Sandhu, Access control models for cloud-enabled internet of things: A proposed architecture and research agenda, in: Collaboration and Internet Computing (CIC), 2016 IEEE 2nd International Conference on, IEEE, 2016, pp. 530–538.
[148] J. Soldatos, N. Kefalakis, M. Hauswirth, M. Serrano, J.-P. Calbimonte, M. Riahi, K. Aberer, P. P. Jayaraman, A. Zaslavsky, I. P. Žarko, et al., Openiot: Open source internet-of-things in the cloud, in: Interoperability and Open-Source Solutions for the Internet of Things, Springer, 2015, pp. 13–25.
[149] L. Jiang, L. Da Xu, H. Cai, Z. Jiang, F. Bu, B. Xu, An iot-oriented data storage framework in cloud computing platform, IEEE Transactions on Industrial Informatics 10 (2) (2014) 1443–1451.

[150] Z. Y. Mohamed, D. M. Elsir, M. O. Elbasheer, J. Lloret, Architecture proposal for mcloud iot, in: International Conference on Future Intelligent Vehicular Technologies, Springer, 2016, pp. 135–145.
[151] H.-L. Truong, S. Dustdar, Principles for engineering iot cloud systems, IEEE Cloud Computing 2 (2) (2015) 68–76.
[152] S. Nastic, S. Sehic, D.-H. Le, H.-L. Truong, S. Dustdar, Provisioning software-defined iot cloud systems, in: Future Internet of Things and Cloud (FiCloud), 2014 International Conference on, IEEE, 2014, pp. 288–295.
[153] S. Nastic, S. Sehic, M. Vogler, H.-L. Truong, S. Dustdar, Patricia: A novel programming model for iot applications on cloud platforms, in: Service-Oriented Computing and Applications (SOCA), 2013 IEEE 6th International Conference on, IEEE, 2013, pp. 53–60.
[154] K. Stravoskoufos, S. Sotiriadis, E. Petrakis, Iot-a and fiware: Bridging the barriers between the cloud and iot systems design and implementation, in: Cloud Computing and Services Science, Proceedings of the 6th International Conference on, 2016, pp. 146–153.