




# Edge computing

## A grounded theory study

Jorge Pérez<sup>1</sup> · Jessica Díaz<sup>1</sup>  · Javier Berrocal<sup>2</sup> · Ramón López-Viana<sup>1,3</sup> · Ángel González-Prieto<sup>4</sup>

Received: 25 February 2022 / Accepted: 2 July 2022 / Published online: 20 July 2022  
© The Author(s) 2022

### Abstract

IoT edge computing is a new computing paradigm “in the IoT domain” for performing calculations and processing at the edge of the network, closer to the user and the source of the data. This paradigm is relatively recent, and, together with cloud and fog computing, there may be some confusion about its meaning and implications. This paper aims to help practitioners and researchers better understand what the industry thinks about what IoT edge computing is, and the expected benefits and challenges associated with this paradigm. We conducted a survey using a semi-structured in-depth questionnaire to collect qualitative data from relevant stakeholders from 29 multinational companies and qualitatively analyzed these data using the *Constructivist Grounded Theory (Charmaz)* method. Several researchers participated in the coding process (collaborative coding). To ensure consensus on the constructs that support the theory and thus improve the rigor of qualitative research, we conducted an intercoder

---

✉ Jessica Díaz  
yesica.diaz@upm.es

Jorge Pérez  
jorgeenrique.perez@upm.es

Javier Berrocal  
jberolm@unex.es

Ramón López-Viana  
ramon.lopez.viana@alumnos.upm.es

Ángel González-Prieto  
angelgonzalezprieto@ucm.es

- <sup>1</sup> Departamento de Sistemas Informáticos, ETSI Sistemas Informáticos, Universidad Politécnica de Madrid, C. Alan Turing s/n (Carretera de Valencia Km 7), 28031 Madrid, Spain
- <sup>2</sup> Universidad de Extremadura, Avda. de la Universidad, 10001 Cáceres, Spain
- <sup>3</sup> Capgemini Engineering, C. Campezo, 1, 28022 Madrid, Spain
- <sup>4</sup> Departamento de Álgebra, Geometría y Topología. Facultad CC. Matemáticas, Universidad Complutense de Madrid, Plaza Ciencias 3, 28040 Madrid, Spain

agreement analysis. From the analysis, we have derived a *substantive and analytic* theory of what companies perceive about IoT edge computing, its benefits and challenges. The theory is substantive in that the scope of validity refers to the 29 surveys processed and analytic in that it analyzes “what is” rather than explaining causality or attempting predictive generalizations. A public repository with all the data related to the information capture process and the products resulting from the analysis of this information is publicly available. This study aims to strengthen the evidence and support practitioners in making better informed decisions about why companies are adopting edge computing and the current challenges they face. Additionally, the testing theory phase shows that the results are aligned with the ISO/IEC TR 30164 standard.

**Keywords** Edge Computing · IoT · Grounded Theory · Inter-rater Reliability · Inter-rater Agreement

**Mathematics Subject Classification** 68-02, 68Qxx

## 1 Introduction

During the last few years, we have witnessed a massive increase in the number of devices connected to the Internet. In particular, it is expected that the number of connected devices will be more than three times the global population by 2023. Indeed, Machine-To-Machine (M2M) connections will correspond to half of such connected devices (up to 14.7 billion M2M connections) in that year [7]. This deployment has fostered the deployment of the Internet of Things (IoT) paradigm, a comprehensive network of intelligent objects that have the capacity to automatically organize, share information, data, and resources, react and act in the face of situations and changes in the environment [35]. This has led to an exponential growth in the amount of data traffic flowing through the network.

The computation of shared data can be intensive and can usually not be completed by the IoT devices themselves, due to limited resources (memory, battery, etc.) [2]. The deployment and exploitation of the cloud paradigm has helped companies address this capacity limitation by offloading intensive computing tasks to the cloud. However, this offloading has a penalty on the quality of service (QoS) offered, such as the increase in latency imposed by the distance between the cloud and the end devices, the network overhead, and the increase in the security and privacy risk that this offloading entails.

To reduce this penalty, over the last few years, the edge computing paradigm has been proposed. Edge computing allows the offloading of the computing task to nodes closer to the end devices (at one hop from them). Therefore, these tasks are closer to the source of data and the consumer of data, increasing the quality of service offered.

However, the exact definition and coverage of edge computing are unclear. Some researchers indicate that edge computing addresses the offloading of computing tasks from the cloud to the last hop before smart devices, others indicate that it only covers the set of devices at one hop from end devices, some also include IoT devices, etc. [60]. In addition, different researches recommend the application of edge computing for different domains, applications with a strict response time, for those who want

to reduce the infrastructure cost or to improve the privacy management. Therefore, there is currently a lack of consensus on the specific coverage, targeted domains, and benefits of edge computing.

For this paradigm, to have an adoption by the industry similar to that of other proposals, such as cloud computing, all stakeholders should share a common vision, clearly knowing the different elements and concepts involved in it, what are the main problems that allow them to address, what are the main benefits provided, and for which domains.

This paper presents an analysis of the vision of edge computing in the industry. To this end, 29 international companies have been surveyed to identify what they understand by edge computing, what problems it addresses, and what benefits it provides. This analysis allows us to bring the vision of the business world closer to the academy, allowing both of them to better focus their efforts to improve its adoption.

The remainder of the paper is structured as follows. Section 2 analyzes the main concerns that led us to carry out this study. Section 3 is devoted to the research methodology applied in this work. The development and final description of the created theory on IoT edge computing is presented in Sect. 4. In Sect. 5 we analyze the threads to validity of this theory and its limitations. A description of existing work related to our proposal is provided in Sect. 6, information that is complemented by the information shown in B. Finally, Sect. 7 draws the main conclusions of this work.

## 2 Motivation

Cloud computing is an architecture based on accessing centralized computing resources ubiquitously and on demand by making use of the network. This paradigm was standardized by the National Institute of Standards and Technology (NIST). The concept of cloud computing is to have hugely powerful servers in data centers connected to the network. The resources of cloud servers are then virtualized and offered to clients. Cloud computing has been the de facto solution over the past decade. One of the main reasons for its rise and wide adoption is the clarity and common vision that the entire industry has about the advantages and disadvantages that it provides [22]. For Edge Computing (EC), and other similar paradigms, to be successfully incorporated by the industry, it is necessary for them to clearly share its vision and the benefits it brings.

Specifically, there are some key requirements of QoS-stringent IoT applications that cannot be met only by applying a pure cloud paradigm, such as response time, cost, sensitivity, data volume, bandwidth limitation, resilience, etc. [27]. The Internet Research Task Force (IRTF) in its draft on *IoT Edge Challenges and Functions* [31] concluded that these limitations should be overcome by applying EC. The basis for EC is to use different computing devices closer to the end user to distribute the workload of the application to them [36]. However, there is no consensus on the definition of this paradigm in the literature, nor on the border with other closely related paradigms, which jeopardize the adoption of these paradigms by the industry [60].

ISO defines the term Edge Computing (EC) as “*a form of distributed computing in which processing and storage takes place on a set of networked machines which are*

near the edge, where the nearness is defined by the system's requirements" [30]. Nevertheless, the NIST also introduces a couple of closely related terms, Fog Computing (FC) and Mist Computing (MC), which makes it more difficult to clearly identify the borders of EC and its benefits. FC is defined by the OpenFog Consortium as "a horizontal system-level architecture that distributes computing, storage, and networking functions closer to the user along a cloud-to-thing continuum" [20]. Moreover, FC can be multitiered: fog nodes do not need to be placed at a single point or a single network tier and can be placed on multiple tiers. Instead, MC is "a lightweight computation distribution proposal that resides directly at the edge of the network fabric, bringing the FC layer closer to the smart end devices".

Different proposals have been proposed with concrete nuances to support these paradigms. For instance, the ETSI's Multi-access Edge Computing (MEC) [9]. MEC is an architecture based on the provision of computing and storage resources closer to users, similar to FC, but with resources directly connected to a 4G or 5G base station. In addition, the concept of Cloudlet proposes to bring cloud servers to the edge, instead of computing and storage resources closer to the edge with intermediate devices [49]. Aside from the previously defined architectures, some works present different proposals contributing to the EC. In [4], the authors present Mobile Edge Clouds, a three-tier architecture for IoT that contains IoT devices, a middle tier that contains mobile devices, which can run services, and the cloud. During the execution time, the bottom tier requests a service to the middle tier, which can be executed either within the middle tier or it can be offloaded to the top tier. The osmotic computing platform is another proposal that is also closely related to edge computing [56]. Osmotic computing bases its model on *microelements*. These microelements are deployed and provisioned on the basis of the concept of osmosis: initially, microelements are deployed in the cloud. As requests for microelements arrive at the infrastructure, the osmotic computing platform is in charge of detecting the requirements from the requests. If these requirements require that the microelement be deployed on the edge, the platform automatically provisions them at the appropriate edge node and maintains them as long as required.

Regarding MC, some works define different proposals detailing its own architecture [42, 53], while others define it as a layer of fog or edge computing [28]. Thus, although it is possible to combine MC with other architectures [28, 42, 53], it is not clear whether such a combination is allowed by design or not. Other architectures aim at combining different architectures into a single one by merging their proposals and concepts. This is the case of [3], which proposes a combination of MEC and FC. In the literature, EC in some cases is described as a concept implemented in FC, MC, or MEC as described by K. Dolui et al. [11]. In other cases, it is used interchangeably as pointed out by [60].

Therefore, there is a plethora of approaches that define frameworks to apply the edge computing paradigm. However, each approach addresses the paradigm from a different point of view, defines different elements, and tries to meet different goals and challenges. Therefore, there is no clear and unanimous agreement on what EC is, which is crucial for its adoption in the industry, and more importantly, this view should come from the industry to share a common language and better transmit the benefits of this paradigm.

### 3 Research methodology

This paper presents empirical research on practicing IoT edge computing. It is based mainly on the constructivism model as an underlying philosophy (epistemological and ontological positions) [12]. Constructivism or interpretivism states that scientific knowledge cannot be separated from its human context and that a phenomenon can be fully understood by considering the perspectives and the context of the involved participants. Therefore, the most suitable methods to support this approach are those collecting rich qualitative data, from which theories (tied to the context under study) may emerge.

One of these methods is Grounded Theory (GT), which aims at the iterative development of a theory from qualitative data [16] and encourages deep immersion in the data [46]. “*In grounded theory, initial analysis of the data begins without any preconceived categories. As interesting patterns emerge, the researcher repeatedly compares these with existing data, and collects more data to support or refute the emerging theory*” [12]. Thus, GT is adequate for our purposes, and according to our philosophical stance, we used *Constructivist Grounded Theory* (aka Charmaz’s GT variant [6]). Specifically, we applied a novel process that extends the Charmaz GT variant to allow multiple researchers to participate in the coding process, i.e., *collaborative coding*, while ensuring consensus on the constructs that support the theory and, thus, improving the rigor of qualitative research (cf. [10]). To this end, we conducted an analysis of the intercoder agreement (ICA) to measure the extent to which different raters assign the same precise value (code or category) for each item being rated (qualitative data item or quotations) [14].

GT allows in-depth analysis of the phenomenon to be studied, that is, the perception that companies have of IoT edge computing. Once the data had been collected through a survey, GT is the methodology (in the field of qualitative research) that seemed most appropriate. Others, such as ethnographic or action research, require the researcher to enter the context to be studied for a prolonged period of time, which is unfeasible for this study due to the circumstances surrounding it, that is, software companies jealous of the way they work. We did not find appropriate to use focus groups (given that the individuals work for different organizations) or content/thematic analysis (subsumed in GT but which do not generate any theory).

#### 3.1 Initial research questions

We began by asking what the industry thinks about what IoT edge computing is, and the expected benefits and challenges associated with this paradigm.

#### 3.2 Data collection

GT involves iteratively performing interleaved rounds of qualitative data collection and analysis to lead to a theory (e.g., concepts, categories, patterns) [47]. The selection of participants is also iterative and can be considered a combination of “convenience sampling” as we are restricted to organizations and relevant stakeholders to which we

had access; “theoretical sampling”, in the sense that we chose which data to collect based on the concepts or categories that were relevant to the emerging theory, i.e., data from organizations that have been adopting IoT edge computing; and “maximum variation sampling”, in the sense that we tried to choose highly diverse people and organizations in our sample, strengthening the transferability of our theory.

According to the *purposed sampling strategy*, we initially collected data from a set of participants from several leading international organizations in the Internet of Things domain, which are currently committee members of the Master’s Degree in Distributed and Embedded Systems Software<sup>1</sup> and Master’s Degree in IoT<sup>2</sup> at Universidad Politécnica de Madrid (Spain), and international industrial contacts of the Universidad de Extremadura (Spain). Then we moved on to *theoretical sampling* and iteratively collected more data based on the concepts or categories that were relevant to the emerging theory until the ICA value exceeded a given threshold and the theoretical saturation was reached. Table 1 lists the organizations involved in the study, their ID, scope (international or national), size<sup>3</sup>, business core, and role and experience of the respondent. A total of 29 responses were collected from an open-ended questionnaire available in <https://es.surveymonkey.com/r/PMWD7ZM>.

### 3.3 Qualitative data analysis

GT is a technique for iteratively developing theory from qualitative data [16] that encourages a deep immersion in the data [46]. “*In grounded theory, initial analysis of the data begins without any preconceived categories. As interesting patterns emerge, the researcher repeatedly compares these with existing data and collects more data to support or refute the emerging theory*” [12]. To conduct a constructivist GT, we will follow the following steps: initial/open coding, selection of core categories, selective coding, sorting, theoretical coding, and write-up. These steps are detailed in the next section.

## 4 A Theory on IoT edge computing

This section describes a theory on how the IoT industry perceives the edge computing paradigm, as well as the benefits they expect from adopting this paradigm and the challenges they face. To analyze the data from the survey responses of 29 companies and construct the theory, we followed the steps described in the previous section. It is important to keep in mind two concerns: i) the theory to be developed is a substantive theory; and ii) the theory is about how companies perceive the IoT edge computing

---

<sup>1</sup> <http://msde.etsisi.upm.es/>

<sup>2</sup> <https://masteriot.etsist.upm.es/?lang=en>

<sup>3</sup> Spanish Law 5/2015 indicates that a micro enterprise is one that has less than ten workers and an annual turnover of less than two million euros or a total asset of less than two million euros; a small company is one that has a maximum of 49 workers and a turnover or total assets of less than ten million euros; medium-sized companies are those with less than 250 workers and a turnover of less than fifty million euros or an asset of less than 43 million euros; and large companies are those that exceed these parameters.

**Table 1** Description of organizations

ID	Organization			Stakeholder	Experience
	Scope	Size	Business	Role	
1	Internat.	S	installation and maintenance services	Software architect	15
2	Nat.	mi	Consultancy	Director	–
3	Nat.	S	Academic system	Student	2
4	Nat.	mi	Forestal	Platform engineering	2
5	Internat.	L	Telco	CTO	–
6	Internat.	M	Cellular IoT connectivity	Director product	–
7	Internat.	L	Food	Director	15
8	Internat.	L	Testing and Certification	Manager	15
9	Internat.	L	Public cloud	Cloud architect	+20
10	Internat.	L	Security	Manager	+30
11	Internat.	M	IoT and M2M	Field Application Engineer	20
12	Internat.	L	IoT simulation and machine learning	Product manager	5
13	Internat.	S	Industry 4.0	System engineer	–
14	Internat.	S	Control & Monitoring Electronics for refrigeration	Product manager	>5
15	Internat.	L	Software development	Software architect	18
16	Internat.	L	ITC	Sales director	20
17	Internat.	L	Consultancy / On-Demand Solutions	Analyst-programmer	3
18	Nat.	S	Providing technological solutions for the livestock sector	IoT development engineer	1
19	Internat.	S	Software development	CEO	5
20	Nat.	S	Software	Consultant	18
21	Internat.	mi	Industry 4.0	Manager	+20
22	Internat.	M	Energy efficiency	Firmware engineer	8
23	Internat.	L	Smartcities, Industry 4.0, Business Analytics	Presales manager	8
24	Internat.	S	Embedded systems R&D: New product development IoT	Lead developer	8
25	Internat.	M	Big Data & AI	Head of Big Data & Governance	+20
26	Internat.	L	IT Software and Services	Manager	20
27	Internat.	M	IoT Projects	IoT BDM	4
28	Internat.	S	PV energy	COO	12
29	Internat.	L	Services	Edge architect	6

Scope: International (Internat.) / National (Nat.)

Size: Large (L) / Medium (M) / Small (S) Experience (in years): No data (–)

paradigm and not so much how the paradigm has been defined in other scientific sources and/or standards.

#### 4.1 Initial/Open coding

This activity aims to discover the concepts underlying the data and instantiate them in the form of codes. Thereby, at each iteration of the open coding,  $n$  documents of the survey are analyzed, that is, chopped into quotations that are assigned to either a previously discovered code or a new one that emerges to capture a new concept.

##### Iteration 1

In the first iteration of the open coding process, researchers R1, R2, and R3 analyzed 6 documents. R1 created a codebook with 29 codes that was subsequently refined by R2 and R3. As a by-product of this process, 40 codes were discovered and divided into 7 semantic domains (denoted by S1, S2, ..., S7) (see Table 2).

After completing the coding process, Krippendorff's  $\alpha$  coefficients [26, 32] were computed (see also [18] for a thorough introduction to these techniques). Specifically,  $Cu-\alpha$ <sup>4</sup> and  $cu-\alpha$ <sup>5</sup> coefficients were computed and their values are shown in Table 3. As we can observe from this table, the value of the global coefficient  $Cu-\alpha$  did not reach the acceptable threshold of 0.8, as fixed in the literature [32]. For this reason, a review meeting was necessary to discuss disagreements and the application criteria of the different codes. The results of this meeting are documented in the *disagreements diary* file of the *open coding* folder in the public repository.

To highlight problematic codes, we used the coefficients  $cu-\alpha$  calculated per semantic domain. For Table 3, we observe that domain S3 had a remarkably low value of the coefficient  $cu-\alpha$ . A thorough look at the particular codes within S3 shows that this domain includes codes related to the functionality of the system. This is particularly a fuzzy domain in which several concepts can be confused. During the review meeting, clarifications about these codes were necessary to avoid misconceptions. After this, a new codebook was released. In this new version, memos and comments were added, and a code was removed, so 39 codes (and 7 semantic domains) proceeded to the second iteration of the open coding.

##### Iteration 2

Researchers R1, R2, and R3 analyzed six other documents. Since the coders agreed on a common codebook in the previous iteration, we can expect a greater agreement that materializes as a higher value of ICA. As a by-product of this second iteration, 8 new codes arose, leading to a new version of the codebook with 47 codes and 7 semantic domains (see Table 4).

<sup>4</sup> This coefficient measures the degree of agreement in the decision to apply different semantic domains, independent of the chosen code.

<sup>5</sup> This coefficient is computed on a specific semantic domain S. It indicates the degree of agreement with which coders identify codes within S.



**Table 2** Domains and Codes resulting from Open Coding - Iteration 1

Domain: Benefits (S1)	Domain: Conceptualization (S2)
B01 better user experience	C01 (physical/virtual) device characterization: intelligence, reliability, security, efficiency, availability, status, load
B02 less response time	C02 restricted capabilities (limited computational capabilities)
B03 greater efficiency and speed	C03 devices type: sensors, actuators, constrained devices, gateways, micro-controllers, miniPCs, servers
B04 save energy	C04 devices type: medical devices
Domain: Functionality (S3)	Domain: Management (S4)
F01 local processing in device	G01 cloud-managed (remote) over the air updates
F02 reliable services	G02 continuous integration (CI) and continuous delivery/deployment (CD)
F03 devices take over part of the data center/cloud workload	G03 remote and local over-the-air updates
F04 data collection	G04 automated provisioning, monitoring, deployment, build, testing, maintaining
F05 data aggregation and filtering, data analytic, video processing, artificial intelligence, control logic	G05 bringing agile methodologies with customers
F06 communication with other devices	G06 servers (remote) over-the-air updates
F07 decision making	G07 https requests (remote) over the air updates
F08 bringing infrastructure closer to the consumer	
Domain: Network (S5)	Domain: Challenges (S6)
N01 (less) bandwidth	R01 time to market
N02 (less/low) latency	R02 speed up delivery
N03 speed up communications	R03 supervision and management, certifications
N04 data exchange between multiple nodes	R04 deployment time
	R05 scalability
Domain: Technology (S7)	R06 security
T01 containers	R07 vendor lock-in
T02 virtual environments (machines, networks, servers)	R08 (maintenance) cost
T03 downlinks of wireless communication networks	R09 reliability
T04 orchestration layer	

**Table 3** Values of the different Krippendorff's  $\alpha$  coefficients in the iteration 1 of the open coding. In bold, the values above the acceptability threshold ( $\geq 0.80$ )

<i>cu-<math>\alpha</math></i> per semantic domain							<i>Cu-<math>\alpha</math></i>
S1	S2	S3	S4	S5	S6	S7	
<b>0.81</b>	<b>0.98</b>	0.59	<b>0.80</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	0.56

**Table 4** Domains and Codes resulting from Open Coding - Iteration 2

Domain: Benefits (S1)	Domain: Conceptualization (S2)
B01 better user experience	C01 (physical/virtual) device characterization: intelligence, status, load
B02 less response time	C02 restricted capabilities (limited computational capabilities)
B03 greater efficiency and speed	C03 devices type: sensors, actuators, constrained devices, gateways, micro-controllers, miniPCs, servers
B04 save energy	C04 devices type: medical devices
B05 better performance	C05 servers, mini-datacenters
B06 business needs	C06 IoT SIM cards
	C07 distributed architecture
Domain: Functionality (S3)	Domain: Management (S4)
F01 local processing in device	G01 cloud-managed (remote) over the air updates
F02 reliable services	G02 continuous integration (CI) and continuous delivery/deployment (CD)
F03 devices take over part of the data center/cloud workload	G03 remote and local over-the-air updates
F04 data collection	G04 automated provisioning, monitoring, deployment, build, testing, maintaining
F05 data aggregation and filtering, data analytic, video processing, artificial intelligence, control logic	G05 bringing agile methodologies with customers
F06 communication with other devices	G06 servers (remote) over-the-air updates
F07 decision making	G07 https requests (remote) over the air updates
	G08 billing data in real time
	G09 locally-managed over the air updates
Domain: Network (S5)	Domain: Challenges (S6)
N01 (less) bandwidth	R01 time to market
N02 (less/low) latency	R02 speed up delivery
N03 speed up communications	R03 supervision and management, certifications
N04 data exchange between multiple nodes	R04 deployment time
N05 disconnected mode	R05 scalability
Domain: Technology (S7)	R06 security
T01 containers	R07 vendor lock-in
T02 virtual environments (machines, networks, servers)	R08 (maintenance) cost
T03 downlinks of wireless communication networks	R09 reliability
T04 orchestration layer	

**Table 5** Values of the different Krippendorff's  $\alpha$  coefficients in the iteration 2 of the open coding. In bold, the values above the acceptability threshold ( $\geq 0.80$ )

<i>cu-<math>\alpha</math></i> per semantic domain							<i>Cu-<math>\alpha</math></i>
S1	S2	S3	S4	S5	S6	S7	
0.72	<b>0.97</b>	<b>0.88</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>0.80</b>

The ICA values for this second iteration are shown in Table 5. From the results of this table, we observe that after this refinement of the codebook, *Cu- $\alpha$*  reaches the acceptable threshold of agreement. In this way, the open coding process can stop: There exists consensus in the interpretation of the codes presented in the codebook, and we can proceed with the selection of core categories and selective coding.

## 4.2 Selection of core categories

In this activity, R1 and R2 selected the core categories, that is, the most relevant codes from the 47 codes obtained in the open coding. To this end, we focused on the groundedness of the codes and semantic domains (i.e., the number of quotations coded by a code) and the density of the codes and semantic domains (i.e., the number of relationships between codes, that is, the cooccurrence of codes in the same quotation). Table 6 shows these values. The detailed analysis is documented in the *selection of core categories* file of the *selection of core categories* folder in the public repository. As a result of the analysis, four semantic domains (S1, S2, S3 and S6) and 29 codes were selected for the next activity. This codebook is available in the *selection of core categories - codebook* file of the public repository.

## 4.3 Selective coding

This is an inductive-deductive process in which new data are labeled with the codes of selected categories (semantic domains). The coders only focused on the core categories, but the number and definition of their inner codes were modified according to the analysis of new data. The researchers R1, R2 and R3 analyzed 6 documents using S1, S2, S3 and S6, which comprise a total of 29 codes. After coding, 9 codes were

**Table 6** Groundedness of the codes per semantic domain

Semantic Domain	Groundedness	Density
Benefits (S1)	20	29
Conceptualization (S2)	33	19
Functionality (S3)	43	21
Management (S4)	27	13
Network (S5)	19	9
Challenges (S6)	36	12
Technology (S7)	17	10

**Table 7** Values of the different Krippendorff's  $\alpha$  coefficients in the selective coding phase. In bold, the values above the acceptability threshold ( $\geq 0.80$ )

$Cu-\alpha$ per semantic domain				$Cu-\alpha$
S1	S2	S3	S6	
<b>1.00</b>	<b>0.95</b>	<b>0.87</b>	<b>1.00</b>	<b>0.80</b>

added to the codebook, representing a total of 38 core codes (see Appendix A). The results of the ICA coefficients obtained after coding are shown in Table 7.

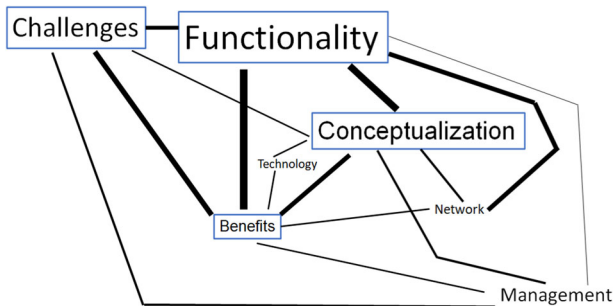
As we can observe from this table, the value of  $Cu-\alpha$  reached the acceptable reliability threshold of 0.8. This evidences that there exists a consensus among the coders on the meaning and limits of the codes within the core categories. Additionally, the coders also agreed that adding new data did not lead to new information, so the theoretical saturation had been reached. Therefore, since after this first iteration, the value of  $Cu-\alpha$  was compelling and the coders agreed that the theoretical saturation had been reached, the GT process could proceed to the next activity.

#### 4.4 Sorting procedure

From the analysis of the memos together with the co-occurrence tables, we drew the relationships between the different categories (see Fig. 1). The core categories are boxed, while the font size of each category, as well as the thickness of the lines that relate them, correspond to the groundedness of semantic domains and the density of codes, respectively.

#### 4.5 Theoretical coding

Theoretical coding is defined as “the property of coding and constant comparative analysis that yields the conceptual relationship between categories and their properties as they emerge” [17]. According to Gregor's taxonomy [19], we develop an analytic theory: “Theories of this type include descriptions and conceptualizations of 'what is'”. Taxonomies, classifications, and ontologies, as defined by Gruber [21], are also included. In fact, Gregor says “Some examples of grounded theory can also be



**Fig. 1** Relations between Categories

examples of Type I theory, where the grounded theory method gives rise to a description of categories of interest.”. Type I refers to “Analytic theories analyze ‘what is’ as opposed to explaining causality or attempting predictive generalizations”. These types of theory are valuable when little is known about the phenomena they describe. This is the case for the edge computing paradigm, which is relatively new. That is, the theory to be built will answer “What is edge computing?” We do not attempt to answer questions related to “why edge computing is used” (explanation theory), nor do we intend to develop mathematical/probabilistic models to support predictions (prediction theory), nor do we intend to describe “how to do” things (design and action theory or prescription theory).

To develop the theory, we follow the following steps, which are thoroughly described in the following sections.

- Determining the scope of the theory
- Defining the constructs of the theory
- Defining the propositions of the theory
- Providing explanations to justify the theory
- Testing the theory

### 4.5.1 Determining the scope of the theory

Figure 2 shows, using UML 2.0, the elements and relations that determine the scope of this theory. We describe the theory scope through four archetype classes: Actor, Technology, Activity, and Software\_System (“An actor applies technologies to perform certain activities on an (existing or planned) software system” [52]). The four archetype classes have been represented as abstract classifiers (classes), and the relationships between these archetypes are as stated in [52]. We added several classifiers (subclasses) to indicate that the activities of IoT edge systems are performed on the edge side (see the enumerated type Location). Specifically, the subclass IoT\_Edge\_Computing rep-

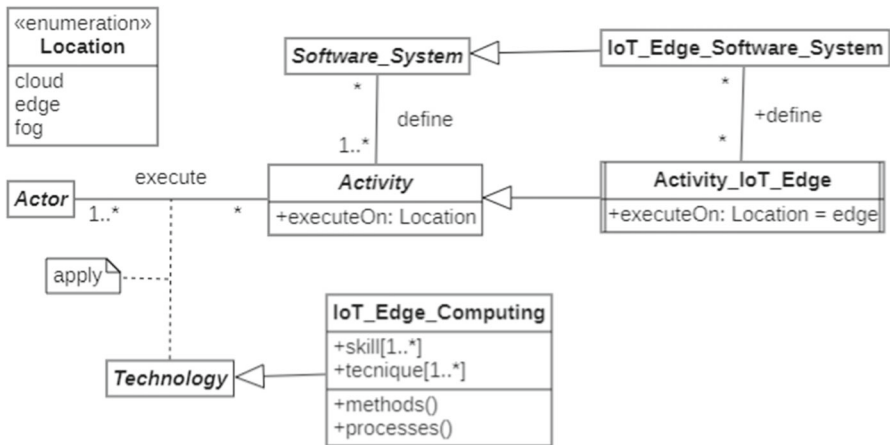


Fig. 2 Scope of the theory

represents a technology understood as a set of skills, techniques, methods, and processes, all specialized for the IoT edge computing paradigm. The subclass *Activity\_IoT\_Edge* represents an activity performed at the edge; in addition, this class has been declared as active, since, by its very nature, its instances will have their own control flows. Finally, the subclass *IoT\_Edge\_Software\_System* represents a software system in the IoT Edge computing domain.

Since the construction of the theory was based on the qualitative analysis of data obtained from a set of surveys of 29 companies in the sector, this theory must necessarily be limited to a substantive (local) theory, as opposed to what could be a formal (all-inclusive) theory. However, the scope of the theory will be revealed during the testing phase.

#### 4.5.2 Defining the constructs of the theory

The constructs have been derived from the codes associated with the core categories. The codes of each core category are described in Appendix A. Table 8 shows the constructs and the code(s) from which they are derived.

Figure 3 shows the relations between the elements of the scope and the constructs. Regarding this figure, class *Device\_in\_the\_Edge* and its subclasses represent the constructs C1 to C4. The classes *Sensor* and *Actuator* represent the construct C5. *Distributed\_Architecture* represents the construct C6. The classes *Advantage* and *Problem* represent constructs C7 and C8, respectively. Construct C9 (represented by the classifier *Activity\_IoT\_Edge*) is an artifice (it is not central to the theory, although it is part of the scope of the theory) that allows us to establish two levels of abstraction in the operations performed by an *IoT\_Edge\_Software\_System*. These high-level operations generate the benefits of the *IoT\_Edge\_Computing* technology. For example, the storage and analytics in the *Device\_in\_the\_Edge*, and the filtering and artificial intelligence techniques enable local data processing and avoid sending raw data to fog/cloud, gaining better bandwidth throughput. Table 9 relates the constructs to the classifiers in Fig. 3.

#### 4.5.3 Defining the propositions of the theory

The propositions of the theory are derived from the relationships between the constructs that make up the theory. In this sense, we extract the relationships described in Fig. 3 as propositions. We characterize each proposition by means of three elements: i) the actual textual statement of the proposition; ii) its formalization by means of the OCL language; and iii) an excerpt, as an example, obtained from the surveys that mention this relationship. The code of each proposition is composed of the letter P, followed by an order number and optionally in square brackets the classifiers to which it relates, a comma character, and the constructs. Additionally, we use a hyphen to indicate several items of a range and & for a sequence of items. Thus, the code P1 [1&6, C1-C4 & C9], refers to Proposition 1, which states the relationship between classifiers 1 and 6 (*Device\_in\_the\_Edge* and *Activity\_IoT\_Edge*) that support the relationship of constructs C1-C4 and C9.

**Table 8** Building constructs from codes

Construct	Code(s)
C1. <i>Device_in_the_Edge</i> . It is an artificial (abstract) concept that encompasses any device (physical or virtual) that is located at the edge	Conceptualization
C2. <i>Device_in_the_Edge</i> :type. Devices such as servers, gateways, microControllers, miniPcs, medical devices, miniDataCenters, IoT SIM cards	C03, C04, C05, C06
C3. <i>Device_in_the_Edge</i> ::attributes. Attributes shared by the devices: limited/restricted computational capabilities, intelligence, status, load, and disconnected mode	C01, C02, C09
C4. <i>Device_in_the_Edge</i> :functionalities. Operations performed by devices (local processing): data collection, data aggregation, filtering, storage and analytics, video processing, virtual (augmented) reality, artificial intelligence, control logic, data exchange and connectivity, decision making, cloud shadowing and protocol transformation	F01, F04, F05, F06, F07, F08, F09, F10
C5. Physical elements. Sensors and actuators	C03
C6. Architecture. Modes of interaction and organization of these devices with the rest of the software system (distributed, modular)	C07, C08
C7. Concerns:Benefit. Advantages of using the technology IoT_edge: better bandwidth, throughput, better performance, better user experience, business needs, customer has the control of their data, greater efficiency, less cloud overhead, less response time, lower latency, and save energy	B01, B02, B03, B04, B05, B06, B07, B08
C8. Concerns:Challenge. The challenges that the IoT_edge technology faces: complexity, deployment time, latency, (maintenance) cost, reliability, scalability, security & privacy, speed up delivery, supervision & management certification, time to market, and vendor lock-in	R01, R02, R03, R04, R05, R06, R07, R08, R10, R11, R12
C9. Activity. Any high-level operation in the IoT Edge computing domain that requires the participation of several instances of <i>Device_in_the_Edge</i> given its functional properties	-

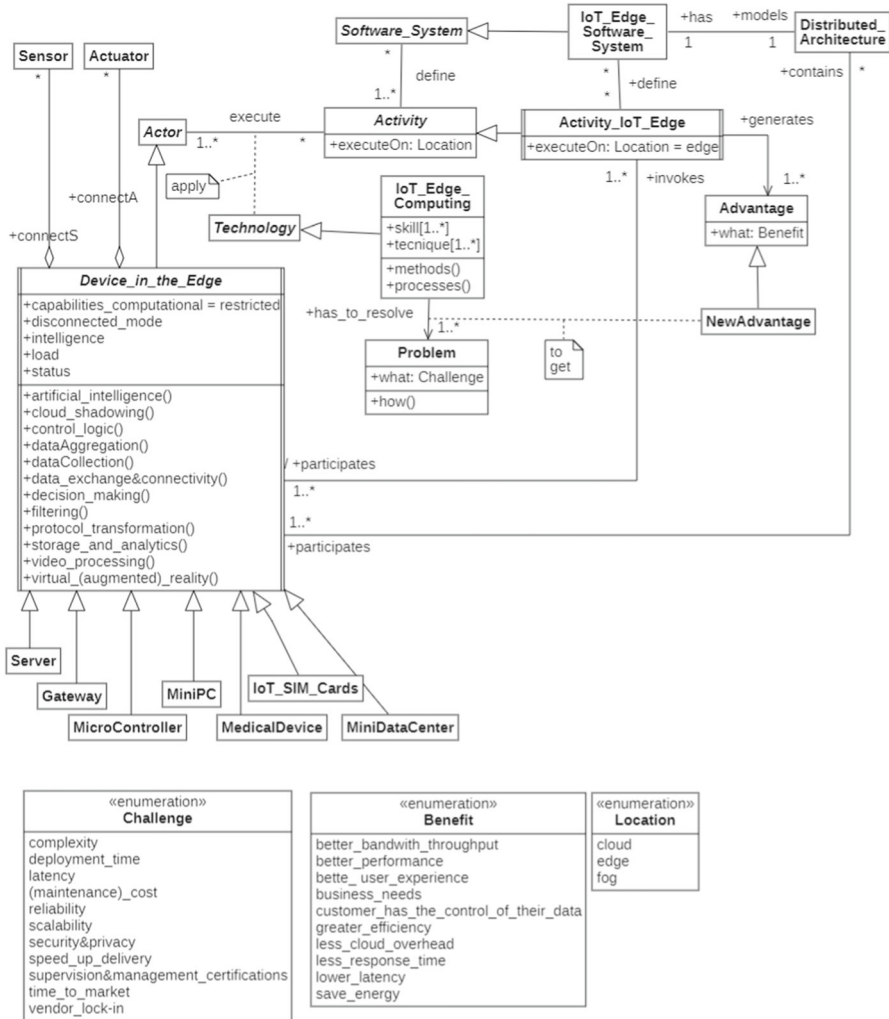


Fig. 3 Constructs & scope of the theory

P1 [1&6, C1-C4 & C9]. A device located at the edge (i.e., an instance of one of the subclasses of the class *Device\_in\_the\_Edge*) /participates in the execution of one or more *Activity\_IoT\_Edge*. Since classifier 6 (see Table 9) is an artifice, there is no excerpt to support it. The OCL syntax is as follows:

```
Context Device_in_the_Edge inv:
    self.invokes->notEmpty()
```



**Table 9** Relations between constructs and classifiers

Construct	Classifier
C1. <i>Device_in_the_Edge</i>	1. <i>Device_in_the_Edge</i>
C2 <i>Device_in_the_Edge</i> :: type	
C3 <i>Device_in_the_Edge</i> :: attributes	
C4 <i>Device_in_the_Edge</i> :: functionalities	
C5 Physical elements	2. Sensor, Actuator
C6 Architecture	3. <i>Distributed_Architecture</i>
C7 Concerns::Benefit	4. Advantage
C8 Concerns::Challenge	5. Problem
C9 Activity	6. <i>Activity_IoT_Edge</i>

P2 [1&6, C1-C4 & C9]. An *Activity\_IoT\_Edge* invokes the operations of one or more instances of one of the subclasses of *Device\_in\_the\_Edge* to carry out its responsibilities. Since classifier 6 (see Table 9) is an artifact, there is no excerpt to support it.

```
Context Activity_IoT_Edge inv:
  self.participates->size()>0
```

P3 [1&3, C1-C4 & C6]. A device located at the edge (that is, an instance of one of the subclasses of the class *Device\_in\_the\_Edge*) participates in a *Distributed\_Architecture*. The OCL syntax and excerpts are as follows:

```
Context Device_in_the_Edge inv:
  self.contains->size() >= 0
```

*ID14* “Edge computing is a distributed computing framework that brings enterprise applications closer to data sources such as IoT devices or local edge servers”

P4. An *IoT\_Edge\_Software\_System* has one and only one architecture, and this architecture is unique. This relationship is established by knowledge of the problem domain: every software system has an associated architecture, whatever its type. The OCL syntax is as follows:

```
Context IoT_Edge_Software_System inv:
  self.models->size() = 1
Context Distributed_Architecture inv:
  self.has->size()=1
```

P5 [4&6, C9 & C7]. The execution of *Activity\_IoT\_Edge* instances generates advantages defined by the enumeration type named *Benefit*. The OCL syntax and excerpts are as follows:

```

Context Activity_IoT_Edge inv:
  (self.participates->exists
    (d | d.allOperations() ->
      exists (op |
        op.name = 'dataAggregation'))
  )
implies
  (self.generates ->
    exists (ob | ob.what =
      Benefit::better_bandwidth_throughput)
  )

```

*ID20 “Additionally, some analysis and data aggregation can be done on edge nodes reducing the network traffic and disc usage on the centralized system”*

P6 [1&5, C1-C4 & C8]. IoT\_Edge\_Computing technology (represented by the IoT\_Edge\_Computing classifier) has a number of problems (defined by the Challenge type) to be solved, the solution of which would bring new benefits (NewAdvantage classifier). The OCL syntax and excerpts are as follows:

```

Context Device_in_the_Edge inv:
  self.IoT_Edge_Computing.has_to_resolve -> notEmpty()

```

*ID02 “Mostly, the big concerns are about security, scalability and avoiding any vendor locks-in. In a second step, the cost of maintenance is usually a big issue to consider”*

*ID03 “Security and compliance is also a concern because we want to offer security and peace of mind to our customers. The deployment time is also important, and we want to reduce and simplify it as much as we can”*

In both cases, reference is made to different challenges faced by using IoT Edge technology, such as security, scalability, vendor lock-in, and deployment time.

P7 [1&2, C1-C4 & C5]. An instance of type *Device\_in\_the\_Edge* (or any of its subtypes) is connected to an arbitrary number of instances of the types Sensor and Actuator. The OCL syntax and excerpts are as follows:

```

Context Device_in_the_Edge inv:
  (self.connectS->size() >= 0) and
  (self.connectA->size() >= 0)

```

*ID01 “Sensors, actuators, gateways and constrained servers (ARM based)”*

*ID02 “Sensors, constrained devices, gateways, servers”*

P8. The use of Edge computing techniques (represented by the IoT\_Edge\_Computing classifier), such as the use of containers, faces some of the challenges (described in the enumeration type Challenge), such as scalability. The OCL syntax and excerpts are as follows:

```
Context IoT_Edge_Computing inv:
  (self.technique -> exists (t|t = 'containers'))
  implies
  (self.has_to_resolve -> exists (c|c.what = Challenge::scalability))
```

*ID09 “Mostly containers due to the scalability”*

*ID10 “Actually, there are several project to containerize the VM into containers to reduce complexity increasing the performance and the scalability”*

P9. The use of Edge computing techniques (represented by the IoT\_Edge\_Computing classifier), such as downlinks of wireless communication networks, enables some of the benefits (described in the enumeration type Benefit) such as *save energy*.

```
Context Device_in_the_Edge inv:
  (self.IoT_Edge_Computing.technique
  -> exists (t| t = 'downlinks_of_wireless_communication_networks'))
  implies
  (self.invokes.generates
  -> exists (c| c.what = Benefit::save_energy))
```

*ID03 “To update the functionalities of our devices and to save energy, we use the downlinks of wireless communication networks and responses to https requests”*

The rest of the associations, indicated in Fig. 3 (other than those of generalization), refer to the scope of the theory (Fig. 2) and are, therefore, outside the scope of the propositions. For example, the relations “define” and “execute”. The generalization (inheritance) relations are implicitly shown in the propositions listed above (indicated by the OCL expressions relating to navigation between classifiers).

#### 4.5.4 Providing explanations to justify the theory

An explanation is a relation between constructs and other categories that are not central enough to become constructs. The code of each explanation comprises the letter E, followed by a number referring to its order, and, optionally between brackets, the number of the proposition related to the explanation separated by hyphens. In this way, the code “E1 [1-2]” refers to Explanation 1 about Propositions 1 and 2.

E1 [1-2]. An activity (instance of the classifier Activity\_IoT\_Edge) may involve several instances of the classifier *Device\_in\_the\_Edge* and one of the latter may intervene in several activities. These activities are high-level operations whose results are sensible to being analyzed to measure the benefits of this paradigm. However, it is complicated to analyze these benefits in operations of a smaller scope carried out by a single type of device.

E2 [3]. The very nature of an IoT application makes it a strong candidate to be based on an architecture with distributed and interconnected elements. In this architecture, many types (instances of the subclasses) of *Device\_in\_the\_Edge* may appear arbitrarily.

E3 [4]. It may seem that the same architecture can support different IoT edge software systems. However, this rarely occurs since the number and types of components involved are typically characteristic of a particular system. However, several IoT edge software systems may share a reference architecture (comprised of a reference model and an architectural style).

E4 [5]. The explanation of some of the benefits captured by the enumeration type Benefit is the following (it is an abductive reasoning):

- E4.1 “better\_bandwidth\_throughput”. Since part of the post-processing of the data ingestion process is done locally, a large amount of bandwidth is saved by transmitting only the data in “cooked” format instead of “raw” format.
- E4.2 “better\_performance”. If all the context (elements needed to carry out a computation) is saved locally, then much time is saved in service requests to other computational nodes, leading to an increase in performance.
- E4.3 “better\_user\_experience”. Since the performance of the system has been improved, better response times may be expected to user queries, leading to an improvement of the quality of the user experience.
- E4.4 “customer\_has\_the\_control\_of\_his/her\_data”. When we transmit the data in “cooked” format, the customer retains control of the “raw” data that were generated in the IoT on the Edge devices and were not sent through the network.
- E4.5 “greater\_efficiency”. We should understand efficiency as the fundamental reduction in the amount of wasted resources that are used to produce a given number of goods or services. In other words, to produce the same results, fewer bandwidth requests and service requests to remote notes are needed.
- E4.6 “less\_cloud\_overhead”. Since a large amount of processing is done locally, the cloud is not responsible for this task.
- E4.7 “less\_response\_time”. This is strongly related to the time invested in communications. When we reduce this time due to local processing, we also decrease the response time observed by the user.
- E4.8 “lower\_latency”. The latency is related to the use of the network. If remote service requests are needed, we must send the request through the network and wait for a response from the server. These lead to an increase in the waiting time to get a response, i.e. the latency of the net. In this manner, the fewer service requests that are issued, the lower the global latency observed.

E5 [6]. The explanation of some of the benefits captured by the enumeration type Challenge is the following (it is an abductive reasoning):

- E5.1. “complexity”. The complexity of these systems is determined by: i) the heterogeneity of the devices to be connected, regarding their properties and functions, but also in the definition of their interfaces; ii) the requirements of real-time operation; iii) the costs of developing and maintaining the system to achieve a permanent operation; iv) the financial and human consequences of a malfunctioning of the system.
- E5.2 “deployment\_time”. The time needed to deploy the system in production environments must be as low as possible if we want to compete with similar products. This also implies that we must deploy new functionalities and fix errors quickly. The complexity of these systems, as pointed out previously, as well as the

necessary automation of the CI and CD process, requires a continuous effort to update on the air infrastructure (hardware and software) to exploit the potential of its new functionalities.

- E5.3 “latency”. Under strict real-time conditions, the latency of the network remains an issue. Maybe the 6G system will cushion this problem, but removing it is quite unlikely. The higher the traffic flow, the higher the expected demand. This phenomenon is analogous to the well-known problem with RAM memory, in which programs tend to occupy all the available space.
- E5.4 “maintenance\_cost”. This cost refers not only to hardware infrastructure (devices and networks to be maintained), but also to the software infrastructures that have to be updated and the applications that require more and more resources.
- E5.5 “reliability”. These types of applications often have strong requirements on the expected reliability. In this context, reliability must be understood as the “degree to which a system, product, or component performs specified functions under specified conditions for a specified period of time” [29]. The lack of this attribute may jeopardize customers and their resources. However, to get reliability, one must balance cost and risk. It is not possible, in very complex systems, to achieve a reliability of 100%, but reaching levels close to this value is feasible.
- E5.6 “scalability”. In general terms, a system is scalable if it can grow to adapt to new and more exigent demands of service, without requiring a change of architecture and only increasing the invested amount of resources. For instance, an intelligence system for agricultural tasks is scalable if it can be adapted to new croplands (with a new area to be screened with new types of crops) by only increasing the number of resources (devices, communications) without altering the architecture or the implementation.
- E5.7 “security&privacy”. In IoT systems in domains such as health, the privacy of the used data and the mechanisms applied to meet these constraints are crucial for the success of the system.
- E5.8 “time\_to\_market”. The speed with which a new version of an IoT edge software system is released is crucial to the survival of any organization.

E6 [7]. A device located at the edge will be connected with sensors (to obtain data from the context) and actuators (to modify the context). The device is fed with sensor data, processes them locally or remotely, and uses the results to command the actuators.

E7 [8-9]. The use of techniques from the IoT Edge computer domain (containers, virtual environments -machines, networks, servers, downlinks of wireless communication networks, orchestration coordination) allows developers to address problems like scalability and to obtain benefits like saving energy or response time.

#### 4.5.5 Testing the theory

The last step of the theory-building process involves examining the validity of the theory. To this aim, we examine the following elements:

1. The data from the surveys not used in the previous steps to contrast how the theory fits to the new data.

2. The standard ISO/IEC TR 30164 (Internet of things (IoT) - Edge computing) to validate the alignment of the theory developed with this standard.
3. The clarity and precision of the elements that are part of the theory.
4. The extent to which a theory has been validated.
5. The scope of the theory.

**Analysis of the remaining surveys.** The remaining 11 surveys were analyzed to test whether the propositions established in Sect. 4.5.3 are aligned or contradict the data contained in the surveys. Recall that, as the previously analyzed surveys, only the answers to questions 7, 10, 13, 14, 15, 16, 19, and 21 were parsed.

This analysis confirms that no new constructs emerged, apart from those described in Sect. 4.5.2, no new relations were needed and therefore no new propositions were added. Furthermore, the previously formulated propositions were validated, clarifying the conclusions.

**Analysis of the standard ISO/IEC TR 30164.** Sect. 1 (Scope) of that document says *“This document describes the common concepts, terminologies, characteristics, use cases and technologies (including data management, coordination, processing, network functionality, heterogeneous computing, security, hardware/software optimization) of edge computing for IoT systems applications”*.

For this reason, it makes sense to compare this standard with our theory to validate the theory. The main conclusions raised were the following.

- The main motivations for edge computing pointed out by the standard (latency, disconnected operations, need to minimize the volume of data transmitted upstream, and data providence) are reflected in the theory developed (Benefit::lower latency, *Device\_in\_the\_Edge*::disconnected mode, Benefit::better bandwidth throughput, and Challenge::security & privacy).
- Our theory encompasses the main classifiers (constructs) indicated in the conceptual viewpoint of the standard as follows. We associate the classifier IoT\_System of the standard with the classifier IoT\_Edge\_Software\_System of the theory, as well as the classifier IoT\_Component of the standard with the classifier *Device\_in\_the\_Edge* of the theory. However, it is worth mentioning that the theory does not distinguish between physical and digital entities, whereas the standard does include this distinction.
- The functional viewpoint of the standard claims that *“An edge computing entity can have but is not limited to the functions mentioned in 6.3.”* The functions described in Section 6.3 of the standard are subsumed in the methods of the *Device\_in\_the\_Edge* classifier. We should understand that these functions are those extracted from the surveys and do not represent an exhaustive list of the functions that can be assigned to an IoT\_Edge\_Software\_System.
- Regarding the deployment viewpoint, the standard defines two deployment models: three levels vs. four layers. In both models, a distributed architecture underlies, as pointed out in the theory developed.

In summary, for the aforementioned reasons, we consider that the theory is perfectly aligned with the standard.

**Clarity and precision.** The constructs and propositions of a theory should be clear and precise so that they are understandable, internally consistent, and free of ambiguities. In our case, the definitions and descriptions of both constructs and propositions have been expressed in UML and, in the case of the propositions, also in OCL. The semantics of each of the elements that appear in the UML/OCL diagrams are described in their respective specifications [41] and [40] (see also [48]). Due to the formal language applied, there is no room for ambiguity or inconsistency, problems that would have been detected by the tool used to draw the diagrams [37].

It is worth mentioning that the semantics of some of the operations/attributes defined in some of the classifiers may be misleading, but if we would like to clarify them, this information would be artificially added from the researchers' knowledge, since it is not reflected in the documentation analyzed. In this sense, we limited ourselves to define/characterize the elements that arise in the theory only based on the data extracted from the surveys, trying to not include any extra knowledge.

**Extent to which a theory has been validated.** Following [52], we must differentiate between two terms: scope of interest and scope of validity of a theory. In our case, the scope of interest was explained in Sect. 4.5.1.

On the other hand, quoting [52], "*The theory's scope of validity refers to that part of the scope of interest in which the theory has actually been validated. The scope of validity of a theory is the accumulated scopes of validity of the results of the studies that have tested the theory, or the studies from which the theory has been generated*". In our case, the scope of validity is for the 18 surveys used to generate the theory, plus the remaining surveys (11) used to validate it.

**The scope of the theory.** In general, this concept refers to the fact that conditions must be explicitly and clearly specified, so that the domain or situations in which the theory should be (dis)confirmed and applied are clear. In our case, the scope was set in Sect. 4.5.1 and graphically depicted in Fig. 2. Roughly speaking, the theory can be applied to IoT edge software systems.

## 4.6 Discussion

As noted by Glaser [15], "*The task of the GT researcher is to generate a theory within the chosen data boundaries, not a formal theory*". The same author also highlights "*The researcher, if using the classical GT method, is set up to write – and must – to conclude a substantive GT. He/she should stop, write.*"

Independently of the description in the standard ISO/IEC TR 30164, the theory developed in this work is based on the perception that the professionals involved in the surveying process have about what edge computing is. Indeed, the standard is relatively recent (April 2020), so its adoption by the industry, if finally reached, will take some time.

IoT edge computing is a computational paradigm within the IoT framework characterized by the aim of moving the computations as close as possible to the data source. This computation is held in edge devices that frequently have severe limitations of

computing speed and storage resources. These limitations are common to several types of devices (gateways, servers, microcontrollers, etc.) and condition the functionality that they can host (filtering, video processing, storage and analytics, etc.). As a result of carrying out the computation at the edge, we obtain a set of benefits (better performance, less response time, greater efficiency, etc.). However, a set of problems related to this paradigm must also be addressed if it intends to be applicable to the IoT framework (deployment time, reliability, scalability, etc.). Finally, we would like to mention that all the applications supported by this paradigm must present a highly distributed architecture with interconnected remote nodes in different topologies.

From the analysis conducted, it is possible to deduce that all the companies identify that there are several dimensions that must be taken into account, or that are affected by, the application of edge computing in the design of IoT applications. The dimensions identified by the vast majority of companies are computing, networking, functionality, and technology. Indeed, all of these dimensions are affected because all organizations highlight that the application of a distributed architectural design is crucial for this paradigm and that greater control of this distribution is also necessary to achieve the desired QoS. However, with respect to the challenges highlighted, a lower consensus may initially be seen, but a more thorough analysis shows otherwise. The organizations interviewed identify different challenges, but they are closely related. Thus, an important challenge is the complexity in the management of these applications, which is highly related to other challenges such as the need to automate this management and the deployment of functionalities, which also lead to better control of the scalability, reliability, and maintenance cost of the systems.

Furthermore, the deployment and monitoring of highly distributed applications, where quality can be affected by several highly related dimensions, entails greater complexity in management. Companies demand tools that allow them to automate this process, to detail their needs in a simpler way, to automate how applications should scale in these highly distributed environments, and how the operational cost can be kept under control. Therefore, one of the key aspects that can be deduced from this study is that methodologies, techniques, and tools are needed in this direction, so that organizations can apply this paradigm more boldly and confidently. In this regard, the GT study has detected some of the challenges that have not been described in the literature so far, such as those related to the delivery and deployment of IoT edge applications. This could indicate that further research in that area is required to introduce or adapt already existing paradigms that have been proven successful when dealing with highly distributed workloads such as DevOps or GitOps. In such a case, new research is required to analyze which other benefits and challenges arise when adopting these paradigms in the domain of IoT edge computing.

## 5 Threats to validity and reliability. Limitations

Criteria for judging the quality of the research design are the key to establish the validity, that is, the accuracy of the findings and the reliability, i.e., the consistency of the procedures and the researcher's approach, of most empirical research [8, 59].



We considered the quality criteria defined by Lincoln and Guba's [34] for qualitative research as follows:

- *Credibility* is also referred to as trustworthiness, i.e., the extent to which conclusions are supported by rich, multivocal evidence. The strategy to mitigate this threat was data triangulation. We received surveys from 29 companies, which means that we collected data at different times and locations and from different populations, as can be seen in Table 1.
- *Resonance* is the extent to which a study's conclusions make sense to (i.e., resonate with) participants. A key strategy to that end is member checking, so some participants received preliminary results to ensure the correctness of our findings.
- *Usefulness* is the extent to which a study provides actionable recommendations to researchers, practitioners, or educators and the degree to which the results extend our cumulative knowledge. The usefulness of this study is to validate that the vision of IoT companies aligns with the standards generated in the IoT domain. We assume that the industry has also defined the constructors and relationships in these standards.
- *Transferability* shows whether the findings could plausibly apply to other situations. Data were iteratively gathered from 29 companies, a number large enough to build a complete picture of the phenomenon. This multiplicity is what provides the basis for “theoretical generalization”, where the results are extended to cases that have common characteristics and hence for which the findings are relevant [58]. Furthermore, it is necessary to consider that the theory is substantive (i.e., local to the analyzed surveys). Like any grounded theory study, the result is only applicable to the domain and context being studied and therefore cannot be assumed to be applicable to other contexts or in general.
- *Dependability* shows that the research process is systematic and well documented and can be traced. The public repository contains all the data and procedures used in this research so that other researchers can replicate it.
- *Conformability* assesses whether the findings emerge from the data collected from cases and not preconceptions. As explained in Sect. 4.5.5, we deliberately omitted any interpretation of the analyzed data, even if this may lead to ambiguities or a vague interpretation of the data. Additionally, as pointed out in Sect. 4.5.4, to explain some phenomena, we applied abductive reasoning: we assume the premise to be true and seek the most probable explanation.

## Limitations

As with any research methodology, there are limitations to our choice of research methods. The first limitation of our study lies in the number of surveys. The goal of our study was not to generalize a phenomenon observed in a sample to a population: instead, we are generating a theory about a complex phenomenon from a set of observations obtained through theoretical sampling. Grounded Theory does not support statistical generalization. Although the proposed theory appears widely applicable, organizations with different software development cultures in the IoT edge computing domain could have different perspectives.

## 6 Related work

During the last few years, different work has analyzed the main characteristics of edge computing, its application to specific domains (such as IoT), and the open challenges that should be addressed to increase its adoption by the industry.

Specifically, focused on the application of Grounded Theory to create hypotheses and theories through the analysis of the perception of edge computing by the industry, few resources can be found, as is also stated by [13]. Some works, such as the one presented by Mengru Tu [55] studied the intention to embrace IoT in Indian organizations, focused on the logistics and supply chain management area, identifying that benefits and cost perception were important over technology trust. Furthermore, Radanliev P. et al. [45] use Grounded Theory to identify current gaps in cyber risk standards and policies, defining the design principles of the future cyber risk impact assessment of the IoT. The same authors use Grounded Theory to build a conceptual cascading model for the future integration of cognition in Industry 4.0 [44] where current and future challenges are identified in the use of Artificial Intelligence in cyber-physical systems.

However, a greater number of works can be found in the literature that analyze these paradigms, through surveys or systematic literature reviews, to characterize edge computing, its benefits and challenges. Some of these works have been analyzed in order to, first, better outline some of the questions presented in this work to the industry; and, second, compare the conclusions obtained from applying Grounded Theory and the conclusions obtained by analyzing the related works. A summary of the analyzed works can be found in B.

As a summary, the reviewed related works highlight some benefits of edge computing that has also been identified in this work analyzing the responses of the industry, such as the improvement in the quality of service, the performance of the applications, a better user experience, the increase in data privacy, the decrease of network and cloud overhead, and, also, the decrease in the energy consumption. Nevertheless, a notable key benefit for the industry is to better satisfy business needs. This is a crucial benefit for any company that is usually addressed by related work at the second level (Table 13). Only [39], identifies that one of the benefits of edge computing is the ability to create more innovative solutions. This is because academia usually focuses on more technical aspects, which require greater coordination between both worlds to address and provide more business-related benefits. Therefore, the research community needs to invest efforts to show how this paradigm can be applied by the industry to create solutions that better meet the needs of businesses and their customers.

Linked to these benefits, from the companies' responses, we can also identify a series of challenges that still need to be addressed more deeply, some of them are also highlighted in the related works analyzed. For example (as can be seen in B), both companies and academia identify complexity, latency, cost, scalability, security and privacy, and certifications as challenges that need to be further addressed. This alignment between the two worlds will allow us to address these challenges in an agile and successful way and will provide greater usefulness.

However, we have identified some challenges that are relevant to companies and have not been described in the literature so far (Table 12), such as: the complexity of

the systems using edge computing, the time required to deploy these hyperdistributed systems, the speed in the delivery of the product, how to decrease the time to market applying these solutions, and whether certifications are needed to guarantee the quality of these distributed systems and also those people developing them. As can be seen, these challenges are closely related to the development and maintenance of IoT applications, which is the main concern of companies. On the industry side, companies are the ones that first detect these challenges as relevant because they need to solve them to develop massively systems that apply this paradigm. On the research side, they are currently not core challenges because, although they are important, they are addressed once they are demanded by the industry. This gap between the two worlds may show that the industry's need of applying this paradigm is closer than expected. Therefore, more effort must be invested to make the application and adoption of edge computing smoother. Furthermore, by solving these challenges, a success similar to that provided by cloud computing can be obtained, in which these challenges were also addressed.

Therefore, although the main benefits and challenges of edge computing are similar in both the research and industry contexts, there are some issues mainly related to the impact in business, such as the improvement of the quality of the applications and the decrease in the time-to-market, which have to be deeply addressed in both contexts to increase the adoption of edge computing.

## 7 Conclusion

In recent years, the expression “edge computing” has become familiar in the IoT domain. However, not all stakeholders seem to share the same semantics for this expression, leading to confusion in its implementation and application.

The aim of this work has been to develop comprehensive qualitative research that sheds some light on the meaning of edge computing for the industry. The theory developed in this work comprises nine constructs and nine propositions that define the ingredients that, according to the companies interviewed, are central in the edge computing paradigm. From this point of view, the theory satisfies the parsimony criterion (the degree to which a theory is economically constructed with a minimum of concepts and propositions) and is a substantive and analytic theory.

The main contribution of this work is to show, by construction of a theory, that industry and the standard ISO/IEC TR 30164 are mostly aligned. This makes us expect that in the near future the interoperability issues experienced in the edge computing world will vanish or, at least, will be less predominant. It is worth mentioning that the best alignment between industry and the standard is achieved in the discussion of what the functionalities that the edge computing paradigm should support. Our results show unanimity in the expected benefits of the paradigm in terms of resource consumption, security, performance, etc.

There exists also a clear agreement in the sketch of the challenges that the paradigm must address, even though many of them are not appropriate for the paradigm itself but for software engineering. To highlight some of these shared challenges, concerns were detected about how to address the construction and maintenance of complex systems and reliability and efficiency issues. Other challenges are more characteristic of the IoT environment, such as how to increase the computational performance and storage of devices at the edge and how to improve the use of communication networks.

**Acknowledgements** The work reported here has been partially sponsored by *Comunidad de Madrid* under *Convenio Plurianual* with *Universidad Politécnica de Madrid* in the actuation line of “*Programa de Excelencia para el Profesorado Universitario*” and the Agencia Estatal de Investigación PID2020-118969RB-I00.

**Funding** Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

**Data Availability** Link to supplementary materials in a long-term repository: <https://doi.org/10.5281/zenodo.5034244>. The data of the GT study on EdgeOps were collected from an open-ended questionnaire available at the link <https://es.surveymonkey.com/r/PMWD7ZM>.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## A Core categories after the selective coding phase

In this appendix, we present Table 10 in which we summarize the collection of cores categories, and well as their inner codes, obtained after the selective coding phase.

## B Analyzed related works

This appendix presents three tables with a summary of the different surveys, systematic review of the literature, and related work that has been analyzed. Table 11 summarizes, for each work, the benefits it provides and the challenges highlighted that must be addressed. Table 12 compares the challenges detected with those highlighted by the related works. Finally, Table 13 identifies the benefits that are also directly defined by the analyzed works.

**Table 10** Core categories and codes after the selective coding phase

Domain: Benefits	
B01	Better user experience
B02	Less response time
B03	Greater efficiency
B04	Save energy
B05	Better performance
B06	Business needs
B07	Less cloud overhead (better bandwidth throughput and lower latency)
B08	Customers has the control of their data
Domain: Conceptualization	
C01	(physical/virtual) device characterization: intelligence, status, load
C02	Device capability (limited/restricted computational capabilities)
C03	Sensors, actuators, constrained devices, gateways, micro-controllers, miniPCs
C04	Medical devices
C05	Servers, mini-datacenters
C06	IoT SIM cards
C07	Distributed architecture
C08	Modular architecture
C09	Disconnected mode
Domain: Functionality	
F01	Local processing (computation) in device
F03	Devices take over part of the data center/cloud workload
F04	Data collection and processing
F05	Data aggregation, filtering, storage and analytics
F06	Video processing, virtual (augmented) reality, artificial intelligence, and control logic
F07	Data exchange & connectivity
F08	Decision making
F09	Cloud shadowing
F10	Protocol transformation
F11	Serverless
Domain: Challenges	
R01	Time to market
R02	Speed up delivery
R03	Supervision and management, certifications
R04	Deployment time
R05	Scalability
R06	Security and privacy
R07	Vendor lock-in
R08	(maintenance) cost
R10	Reliability
R11	Latency
R12	Complexity

**Table 11** EC Challenges & Benefits

Authors	Contribution	Requirements	Challenges	Benefits
K. Dolui et al. [11]	This work presents a comparison of different implementations of EC (FC, MEC, and cloudlet) to provide a decision tree for selecting the best implementation based on the features provided by the studied implementations	Proximity Context awareness Access mediums Power consumption Computation time	-	-
P. Bellavista et al. [2]	After an exhaustive literature review that analyzes the main IoT application requirements, this work defines a unified model of a fog platform and a taxonomy to compare different solutions and how these solutions work in specific domains	Scalability Interoperability Real-time Data quality Security Location-awareness Mobility Reliability (resilience)	Standardization Security Privacy Safety Data abstraction Device management Virtualization Logic location	-
Dapeng Lan et al. [33]	This work discusses the characteristics of fog programming and its requirements based on the common fog architectural and application model and using several fog programming frameworks and finally presents several open research questions in fog programming	Heterogeneity Scalability Quality of Service Context awareness Mobility Unreliable access	Service Migration Virtualization Resource Reconfiguration Workload offloading Usage optimization Security and privacy	Better performance Autonomous Privacy Security Context-awareness
Yousefpoor, A. et al. [60]	Extensive Literature Review identifying relevant papers on FC and its related computing paradigms. It describes some distributed computing paradigms (FC, EC, and MEC) and identifies concerns and opportunities for all of them. Finally, it provides a taxonomy of current research topics in FC	QoS Cost Energy Bandwidth Security Foundation RAS Mobility Scalability Heterogeneity Management Programmability	Bandwidth savings Workload offloading Scalability Resiliency Authentication Privacy Resource provisioning	-

Table 11 continued

Authors	Contribution	Requirements	Challenges	Benefits
S. N. Swamy S. R. Kota [54]	This survey provides a comprehensive study of system-level aspects and enabling technologies in IoT with a focus on interoperability, application layer protocols, and security. It reviews IoT application requirements, challenges, and EC, FC, cloudlet and CC classification	Interoperability Device management Data Availability Reliability Scalability Security and privacy	Reliability Scalability Adaptation Context awareness Interoperability Embedded intelligence Privacy & security	-
N. Hassan et al. [25]	It highlights the role of EC in the IoT context through the study of advances in EC from the IoT perspective. It includes a taxonomy for categorizing and classification of edge computing literature. It outlines the key requirements for the successful deployment of EC in IoT including some scenarios of it. Finally, this work identifies and describes several research challenges	Latency Reliability Mobility support Real time interaction Security Interoperability	Heterogeneity Standard Protocols Standard Interfaces Availability Data abstraction Security & Privacy	Latency minimization Cost optimization Network management Resource management Data management
Hamdan S. et al. [24]	This work provides an overview of edge, cloud, and IoT technologies with a focus on challenges to enable future studies on new computing architectures for IoT	Security Performance Bandwidth Data Integrity Complexity Memory Constraints	Security Confidentiality Privacy Scalability Interoperability Device management Data management Availability Data Integrity	-
Carlo Puliafito et al. [43]	This work conducts a survey on FC and provides an overview of existing FC platforms for IoT, highlighting IoT application domains that can benefit from FC based on the existing literature for these domains. Finally, it describes how FC has addressed the requirements of IoT applications	Mobility support Service orchestration Deployment models Revenue scenarios Security and privacy Heterogeneity Computing power Network performance	Security Confidentiality Privacy Scalability Interoperability Device management Data management Availability Data Integrity	Low latency Bandwidth consumption Privacy Security Context awareness

Table 11 continued

Authors	Contribution	Requirements	Challenges	Benefits
Botta A. et al. [5]	Early survey to identify the main enablers for integrating CC and IoT into a unique environment and the complementary aspects to fulfill the IoT application requirements	-	Security and privacy Heterogeneity Performance Reliability Scalability Standardization Power efficiency Security privacy Integration Pricing & billing Network communications	-
M. Mukherjee et al. [38]	This work provides a comprehensive survey of fundamental and recent advancements towards fog, summarizing the key research challenges and how state-of-the-art consider these issues helping to identify the main aspects to consider when designing fog computing networks are also presented. Finally, it highlights some of the open research challenges and discusses future research directions	-	Workload offloading Resource management Mobility virtualization Heterogeneity Standardization	-
S. N. Shirazi et al. [51]	This work describes the influence and impact of EC on existing communication and networking service models based in the cloud, focusing on security and resilience, studying the newly posed requirements of MEC and FC. It includes the examination of models and architectures supporting EC discussing some of the most evident characteristics associated with them. Finally, it analyzes them with respect to security and resilience requirements	Low latency Location awareness Mobility Security Resilience	Standardisation Security Resiliency	-



Table 11 continued

Authors	Contribution	Requirements	Challenges	Benefits
Xiaofei Wang et al. [57]	This work analyzes the application of deep learning in typical IoT scenarios and identifies the benefits, challenges, and future trends of using EC implemented with FC, MEC or Cloudlets	Cost Latency Reliability Privacy	Cost Customized hardware offloading Limited communication Privacy & security	-
Alli AA & Alam MM [11]	This survey, based on several studies that involve IoT, FC and Cloud ecosystems, studies the building blocks FC mainly the concepts, architectures, standards, and tools to identify the requirements of IoT solutions in the FC context and describes the current trends in the research	-	Performance Robustness Security and trust Resource Management Workload offloading	Latency Location of service Geo-distribution Mobility Location awareness Connectivity
Sabireen H. and Neelamarayanan V. [23]	Detailed survey on different computing paradigms along with a comprehensive discussion on the framework for FC with a detailed discussion on different implementations. It also includes a brief overview of the different research work carried out on FC for IoT and various algorithms used for job planning, resource allocation, and workload distribution. Finally, it describes several research challenges related to FC	-	Networking resources Device heterogeneity Distributed computation Provisioning Security Resource management Cloud Integration	Adaptability Real-time Latency Position awareness Heterogeneity Compatibility
Naveen S. et al. [39]	This work provides insight into IoT with CC and EC discussing benefits, challenges and differences between EC and CC	-	Resource constraints Remote management Security and privacy Scheduling Load balancing Reliability	Quick response time Bandwidth consumption Cost Reliability Better user experience Innovative solutions Development support Real-time analytics

**Table 12** EC Challenges

	[2]	[60]	[39]	[33]	[54]	[25]	[57]	[24]	[43]	[5]	[38]	[51]	[1]	[50]
Complexity						x				x	x			x
Deployment time														
Latency		x					x			x	x			x
Maintenance cost		x								x				
Reliability	x			x										
Scalability		x			x		x	x	x	x				
Security and privacy	x	x	x	x	x	x	x	x	x	x		x	x	x
Speed up delivery														
Supervision and management, certifications	x	x	x	x	x		x	x	x		x		x	x
Time to market														
Vendor lock-in														

**Table 13** EC Benefits

	[2]	[60]	[39]	[33]	[54]	[25]	[57]	[24]	[43]	[5]	[38]	[51]	[1]	[50]
Better bandwidth					x		x			x				
Better performance					x	x								
Better user experience					x									
Business needs					x									
Customers have the control of their data							x							
Greater efficiency			x			x				x			x	x
Less cloud overhead			x				x							
Less response time			x		x									x
Lower latency							x			x			x	x

## References

1. Alli AA, Alam MM (2020) The fog cloud of things: A survey on concepts, architecture, standards, tools, and applications. *Internet of Things* 9:100177. <https://doi.org/10.1016/j.iot.2020.100177>
2. Bellavista P, Berrocal J, Corradi A, Das SK, Foschini L, Zanni A (2019) A survey on fog computing for the internet of things. *Pervasive Mob Comput* 52:71–99. <https://doi.org/10.1016/j.pmcj.2018.12.007>
3. Bellavista, P., Foschini, L., Scotece, D.: Converging mobile edge computing, fog computing, and iot quality requirements. In: 2017 IEEE 5th international conference on future internet of things and cloud (FiCloud), pp 313–320. IEEE (2017)
4. Borgia, E., Bruno, R., Conti, M., Mascitti, D., Passarella, A.: Mobile edge clouds for information-centric iot services. In: 2016 IEEE symposium on computers and communication (ISCC), pp 422–428. IEEE (2016)
5. Botta A, de Donato W, Persico V, Pescapé A (2016) Integration of Cloud computing and Internet of Things: A survey. *Futur Gener Comput Syst* 56:684–700. <https://doi.org/10.1016/j.future.2015.09.021>
6. Charmaz K (2014) *Constructing Grounded Theory*, 2nd edn. Sage
7. CISCO: Annual internet report (2018–2023). Tech. rep., Cisco (2020)
8. Creswell JW, Creswell JD (2018) *Research design: qualitative, quantitative, and mixed methods approaches*, 5th edn. SAGE, Los Angeles
9. Dahmen-Lhuissier, S.: ETSI - Multi-access Edge Computing - Standards for MEC (2020). <https://www.etsi.org/technologies/multi-access-edge-computing>
10. Díaz, J., Pérez-Martínez, J.E., Gallardo, C., González-Prieto, Á.: Applying inter-rater reliability and agreement in grounded theory studies in software engineering. *CoRR arXiv:abs/2107.11449* (2021)
11. Dolui, K., Datta, S.K.: Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In: 2017 Global Internet of Things Summit (GIoTS), pp 1–6. IEEE (2017)
12. Easterbrook, S., Singer, J., Storey, M.A., Damian, D.: *Selecting Empirical Methods for Software Engineering Research*, pp 285–311. Springer London, London (2008). [https://doi.org/10.1007/978-1-84800-044-5\\_11](https://doi.org/10.1007/978-1-84800-044-5_11)
13. Ghaffari K, Lagzian M, Kazemi M, Malekzadeh G (2019) A comprehensive framework for Internet of Things development: A grounded theory study of requirements. *JEIM* 33(1):23–50. <https://doi.org/10.1108/JEIM-02-2019-0060>
14. Gisev N, Bell JS, Chen TF (2013) Interrater agreement and interrater reliability: key concepts, approaches, and applications. *Res Social Adm Pharm* 9(3):330–338
15. Glaser BG (2016) Stop. write! writing Grounded Theory Rev: An Int J 11 (1). Sociology Press
16. Glaser B, Strauss AL (1967) *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine de Gryter, New York
17. Glaser BG (1992) *Emergence vs forcing?: basics of grounded theory analysis*. Sociology Press, Mill Valley, CA
18. González-Prieto Á, Perez J, Diaz J, López-Fernández D (2021) Reliability in Software Engineering Qualitative Research through Inter-Coder Agreement: A guide using Krippendorff's  $\alpha$  & Atlas.ti. [arXiv:2008.00977](https://arxiv.org/abs/2008.00977) [cs, stat]
19. Gregor S (2006) The nature of theory in information systems. *MIS Q* 30(3):611–642
20. Group, O.C.A.W., et al.: Openfog reference architecture for fog computing. *OPFRA001* 20817, 162 (2017)
21. Gruber TR (1993) A translation approach to portable ontology specifications. *Knowl Acquis* 5(2):199–220
22. Gupta P, Seetharaman A, Raj JR (2013) The usage and adoption of cloud computing by small and medium businesses. *Int J Inf Manage* 33(5):861–874. <https://doi.org/10.1016/j.jinfomgt.2013.07.001>
23. Sabireen H, Neelanarayanan V (2021) A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges. *ICT Express* 7(2):162–176. <https://doi.org/10.1016/j.icte.2021.05.004>
24. Hamdan S, Ayyash M, Almajali S (2020) Edge-Computing Architectures for Internet of Things Applications: A Survey. *Sensors* 20(22):6441. <https://doi.org/10.3390/s20226441>
25. Hassan N, Gillani S, Ahmed E, Yaqoob I, Imran M (2018) The Role of Edge Computing in Internet of Things. *IEEE Commun Mag* 56(11):110–115. <https://doi.org/10.1109/MCOM.2018.1700906>
26. Hayes AF, Krippendorff K (2007) Answering the Call for a Standard Reliability Measure for Coding Data. *Commun Methods Meas* 1(1):77–89. <https://doi.org/10.1080/19312450709336664>

27. Herrera JL, Galán-Jiménez J, Berrocal J, Murillo JM (2021) Optimizing the response time in sdn-fog environments for time-strict iot applications. *IEEE Internet Things J* 8:17172–17185
28. Iorga, M., Feldman, L., Barton, R., Martin, M.J., Goren, N.S., Mahmoudi, C.: Fog computing conceptual model (2018)
29. ISO/IEC: Iso/iec 25010 - systems and software engineering – systems and software quality requirements and evaluation (square) – system and software quality models. Tech. rep., ISO/IEC (2011)
30. ISO/IEC: Tr 30164:2020 - internet of things (iot) -edge computing. Tech. rep., ISO/IEC (2020)
31. Kovatsch, M., Schooler, E., Kutscher, D.: Iot edge challenges and functions draft-hong-t2trg-iot-edge-computing-05 (2020)
32. Krippendorff K (2004) Reliability in Content Analysis: Some Common Misconceptions and Recommendations. *Hum Commun Res* 30(3):411–433 <https://doi.org/10.1111/j.1468-2958.2004.tb00738.x>. <http://doi.wiley.com/10.1093/hcr/30.3.411>
33. Lan, D., Taherkordi, A., Eliassen, F., Horn, G.: A Survey on Fog Programming: Concepts, State-of-the-Art, and Research Challenges. In: *Proceedings of the 2nd International Workshop on Distributed Fog Services Design - DFSD '19*, pp 1–6. ACM Press, Davis, CA, USA (2019). <https://doi.org/10.1145/3366613.3368120>
34. Lincoln YS, Guba EG (1985) *Naturalistic inquiry*. Sage Publications, Beverly Hills, Calif
35. Madakam S, Lake V, Lake V, Lake V et al (2015) Internet of things (iot): A literature review. *Journal of Computer and Communications* 3(05):164
36. Mell, P., Grance, T., et al.: The nist definition of cloud computing (2011)
37. MKLab: Staruml. <http://staruml.io/>
38. Mukherjee M, Shu L, Wang D (2018) Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges. *IEEE Commun. Surv. Tutorials* 20(3):1826–1857. <https://doi.org/10.1109/COMST.2018.2814571>
39. Naveen, S., Kounte, M.R.: Key Technologies and challenges in IoT Edge Computing. In: *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pp 61–65. IEEE, Palladam, India (2019). <https://doi.org/10.1109/I-SMAC47947.2019.9032541>
40. Object Management Group: Object Constraint Language (version 2.4). <https://www.omg.org/spec/OCL/2.4>
41. Object Management Group: Unified Modeling Language (version 2.5). <https://www.omg.org/spec/UML/>
42. Preden JS, Tammemäe K, Jantsch A, Leier M, Riid A, Calis E (2015) The benefits of self-awareness and attention in fog and mist computing. *Computer* 48(7):37–45
43. Puliafito C, Mingozzi E, Longo F, Puliafito A, Rana O (2019) Fog Computing for the Internet of Things: A Survey. *ACM Trans Internet Technol* 19(2):1–41. <https://doi.org/10.1145/3301443>
44. Radanliev P, De Roure D, Van Kleek M, Santos O, Ani U (2021) Artificial intelligence in cyber physical systems. *AI & Soc* 36(3):783–796. <https://doi.org/10.1007/s00146-020-01049-0>
45. Radanliev P, De Roure DC, Nurse JRC, Mantilla Montalvo R, Cannady S, Santos O, Maddox L, Burnap P, Maple C (2020) Future developments in standardisation of cyber risk in the Internet of Things (IoT). *SN Appl. Sci.* 2(2):169. <https://doi.org/10.1007/s42452-019-1931-0>
46. Ralph P (2019) Toward methodological guidelines for process theories and taxonomies in software engineering. *IEEE Trans Software Eng* 45(7):712–735. <https://doi.org/10.1109/TSE.2018.2796554>
47. Ralph, P., et al.: Empirical standards for software engineering research (2021). [arXiv:2010.03525v2](https://arxiv.org/abs/2010.03525v2) [cs.SE]
48. Rumbaugh J, Jacobson I, Booch G (2004) *Unified Modeling Language Reference Manual, The, 2nd edn*. Pearson Higher Education
49. Satyanarayanan, M.: Mobile computing: the next decade. In: *Proceedings of the 1st ACM workshop on mobile cloud computing & services: social networks and beyond*, pp 1–6 (2010)
50. Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge Computing: Vision and Challenges. *IEEE Internet Things J* 3(5):637–646. <https://doi.org/10.1109/JIOT.2016.2579198>
51. Shirazi SN, Gouglidis A, Farshad A, Hutchison D (2017) The Extended Cloud: Review and Analysis of Mobile Edge Computing and Fog From a Security and Resilience Perspective. *IEEE J. Select. Areas Commun.* 35(11):2586–2595. <https://doi.org/10.1109/JSAC.2017.2760478>
52. Sjøberg, D.I., Dybå, T., Anda, B.C., Hannay, J.E.: Building theories in software engineering. In: *Guide to advanced empirical software engineering*, pp 312–336. Springer (2008)

53. Small, N., Akkermans, S., Joosen, W., Hughes, D.: Niflheim: An end-to-end middleware for applications on a multi-tier iot infrastructure. In: 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA), pp 1–8. IEEE (2017)
54. Swamy SN, Kota SR (2020) An Empirical Study on System Level Aspects of Internet of Things (IoT). *IEEE Access* 8:188082–188134. <https://doi.org/10.1109/ACCESS.2020.3029847>
55. Tu M (2018) An exploratory study of Internet of Things (IoT) adoption intention in logistics and supply chain management: A mixed research approach. *IJLM* 29(1):131–151. <https://doi.org/10.1108/IJLM-11-2016-0274>
56. Villari M, Fazio M, Dustdar S, Rana O, Jha DN, Ranjan R (2019) Osmosis: The osmotic computing platform for microelements in the cloud, edge, and internet of things. *Computer* 52(8):14–26
57. Wang X, Han Y, Leung VCM, Niyato D, Yan X, Chen X (2020) Convergence of Edge Computing and Deep Learning: A Comprehensive Survey. *IEEE Commun. Surv. Tutorials* 22(2):869–904. <https://doi.org/10.1109/COMST.2020.2970550>
58. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A (2012) *Experimentation in software engineering*. Springer Science & Business Media
59. Yin RK (2018) *Case study research and applications: design and methods*, 6th edn. SAGE, Los Angeles
60. Yousefpour A, Fung C, Nguyen T, Kadiyala K, Jalali F, Niakanlahiji A, Kong J, Jue JP (2019) All one needs to know about fog computing and related edge computing paradigms: A complete survey. *J Syst Architect* 98:289–330. <https://doi.org/10.1016/j.sysarc.2019.02.009>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.