Advances in deployment and orchestration approaches for IoT - A systematic review

Phu H. Nguyen, Nicolas Ferry, Gencer Erdogan, Hui Song
SINTEF, Oslo, Norway
firstname.lastname@sintef.no

Stéphane Lavirotte, Jean-Yves Tigli
Université Côte d'Azur, CNRS, I3S
Nice, France
firstname.lastname@unice.fr

Arnor Solberg
TelluIoT
Asker, Norway
firstname.lastname@tellu.no

***Abstract*—To unleash the great potential of the Internet of Things (IoT), it is critical to facilitate the creation and operation of IoT systems across IoT, edge and cloud infrastructures with vast heterogeneity, scalability and dynamicity. What is the current landscape of the existing approaches and tools that attempt to cope with this complexity? The work presented in this paper contributes to this picture. This paper presents the results of our systematic literature review (SLR) on research approaches and tools for the deployment and orchestration of IoT systems (DEPO4IOT). From thousands of relevant publications, we systematically identified and reviewed *seventeen* (17) primary studies for data extraction and synthesis to answer our predefined research questions. The results of our SLR show the technical details of the primary DEPO4IOT studies. A main finding is that most approaches do not properly support software deployment and orchestration at the tiny IoT device level. Moreover, there is a lack in terms of properly addressing the trustworthiness aspects in approaches for IoT systems deployment and orchestration. In this paper, we suggest some potential research directions to address the gaps found.**

***Keywords*-IoT; deployment; orchestration; review; survey**

I. INTRODUCTION

By 2020, Gartner envisions around 20 billion Internet-of-Things (IoT) endpoints will be in use[1], representing great business opportunities. However, to unleash the full potential of the IoT, it is critical to facilitate the creation and operation of IoT Systems [1]. IoT systems are typically complex, large scale, and distributed. Coordinated behavior across IoT, edge and cloud infrastructures need to be managed [2]. Besides, the trustworthiness of such systems is critical[2], ranging from business critical to safety critical. The ability to continuously evolve and adapt these systems is decisive to ensure and increase their trustworthiness, quality and user experience.

Recently, different approaches and tools have emerged to support the orchestration and deployment of IoT systems, hereafter referred to as deployment and orchestration of IoT systems (DEPO4IOT) approaches. Software deployment typically refers to a post-development activity performed once a piece of software has been developed. In this study, approaches are considered as supporting deployment when they explicitly offer mechanisms enabling the software deployment process, which typically consists of the following stages: (i) release, (ii) installation, (iii) activation, (iv) update, (v) adaptation, and (vi) un-deployment [3]. An orchestration is also often represented as a graph describing relationships between software elements or processes. Contrary to deployment configurations, these relationships may represent an order of operations required to realize a behavior. Using this definition, any service composition approach that targets IoT domain is considered as an orchestration approach for IoT. However, there is no clear picture of what are the current primary DEPO4IOT approaches, and how advanced they are in supporting the continuous evolution of IoT systems and keep them trustworthy. For example, we would wonder if there exists any advanced approach to safely control evolving actuator behaviours and secure any sensitive data at any point in time.

We aim to clarify the DEPO4IOT research landscape by identifying the most significant DEPO4IOT approaches, performing analyses on them and discussing their technical details. Based on the analyses, we discuss the existing technical challenges and suggest directions for addressing these challenges. To achieve this goal, we conducted a systematic literature review (SLR) following the guidelines in [4]. From thousands of relevant publications, we systematically identified and reviewed 17 primary studies for data extraction and synthesis to answer our research questions. The main contributions of this work are our answers to the following research questions. **RQ1**: *What are the technical details of the primary* DEPO4IOT *approaches?* **RQ2**: *How do the existing primary* DEPO4IOT *approaches address the trustworthiness aspects?* **RQ3**: *What are the current technical challenges to be further investigated?*

In the main content, Section II presents our SLR approach. Section III describes our classification schemes for the primary studies applied to facilitate the data extraction and comparison. We give the results of our SLR in Section IV and discuss related work in Section V. Finally, Section VI concludes the paper with the major findings.

II. OUR SYSTEMATIC REVIEW APPROACH

We conducted the SLR by following the guidelines in [4]. Based on the context and motivation presented in Section I, we give the research questions (RQs) in Section II-A. To explicitly specify the scope of our SLR and reduce

[1]http://www.gartner.com/newsroom/id/3598917

[2]https://www.enisa.europa.eu/publications/baseline-security-recommendations-for-iot

possible bias in our selection process, Section II-B clarifies the inclusion and exclusion criteria for selecting primary studies. Section II-C shows our strategy to find and select the primary studies for answering the research questions.

A. Research questions

To answer the general research questions given in Section I, we refine them into the following sub-questions. RQ1's sub-questions: **RQ1.1** - *What are the publication status of* DEPO4IOT *research?* **RQ1.2** - *How do the primary* DEPO4IOT *approaches support IoT deployment and orchestration?* **RQ1.3** - *What are the design support aspects of the primary* DEPO4IOT *approaches?*

Moreover, we examine how the trustworthiness aspects, which are important in the development and operation of IoT systems, are supported by the primary DEPO4IOT studies. RQ2's sub-questions: **RQ2.1** - *How do the existing primary* DEPO4IOT *approaches address the trustworthiness aspects?* **RQ2.2** - *Have any primary* DEPO4IOT *approaches provided advanced supports such as monitoring, adaptation, and shared access to resources?* **RQ2.3** - *How mature are the approaches in terms of tool support and evaluation?*

Finally, we want to discuss the open issues that would deserve more research attention and some potential directions to tackle these issues (RQ3).

B. Inclusion and exclusion criteria

Based on the research questions and the scope of our study presented in Section I, we clearly predefined the inclusion and exclusion criteria to reduce bias in our process of search and selection of primary studies. The primary studies must meet ALL the following inclusion criteria (IC):
- (IC1) A primary study must contain a deployment (with or without orchestration) approach for IoT systems.
- (IC2) A primary study must be explicitly in IoT area, either in general or in a specific application domain of IoT.
- (IC3) A primary study must have software engineering (SE) approaches as deployment is typically SE activity.

We excluded non-peer-reviewed or unpublished paper, white paper, technical report, thesis, patent, presentation, book chapter, and paper not written in English.

C. Search strategy and selection process

In Section II-C1, we present our database search process. To complement for the database search process, we have also conducted a manual search process presented in Section II-C2. Fig. 1 shows an overview of the search and selection process with the results for each step, which we describe in the rest of the section.

1) Database search: We used four popular publication databases IEEE Xplore[3], ACM DL[4], Science Direct[5], and

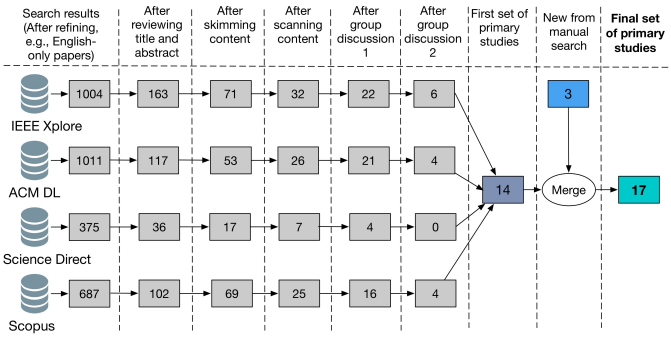

Figure 1. Overview of the search and selection steps

Scopus[6] to search for primary studies. We did not use Google Scholar and SpringerLink. Scopus and ACM DL already index SpringerLink[7]. Google Scholar returns all kinds of papers, in which peer-reviewed articles should have been covered by our four chosen databases. Worse, Google Scholar also returns many non-peer-reviewed and non-English papers, which should have been excluded at the first place in our search process. The four chosen databases contain peer-reviewed articles, and provide advanced search functions, especially search in meta-data such as title, abstract, keywords that we used. Based on the research questions, we identified the search keywords. Basically, we used the following search query: *("Internet of Things" OR IoT OR "Web of things" OR WoT) AND (orchestration OR deployment OR choreography OR topology OR composition OR dataflow) AND (Tool OR Middleware OR Service OR Framework)*. The search string was applied according to the search functions provided by the four databases.

For each candidate paper, we first read the paper's title, keywords and abstract. If the title, keywords and abstract are insufficient for us to decide to exclude it, we further skimmed and scanned the paper's full content. Note that we rather kept any candidate paper in doubt at one point for further checks later. In the end, we hold discussions among reviewers to crosscheck the candidate papers in doubt and agreed on final decisions to include or exclude them. In the first round (depicted as group discussion 1 in Fig. 1), we discussed a list of studies focusing on either deployment OR orchestration. However, we found that most of the discussed studies only focus on orchestration that do not have any research contributions in deployment for IoT. In the second round of group discussion (depicted as group discussion 2 in Fig. 1), from the first list, we shortlisted 14 primary studies that have research contributions in deployment for IoT (using IC1), which are in the scope of this study.

2) Manual search: We conducted a manual search process by first initiating a set of the DEPO4IOT studies that we have known of such as [S7-S9, S11, S14, S17] in Table

[3]http://ieeexplore.ieee.org
[4]https://dl.acm.org
[5]https://www.sciencedirect.com

[6]https://www.scopus.com
[7]https://www.springer.com/gp/computer-science/lncs/information-on-abstracting-and-indexing/799288

I. This was the test set to fine-tune the search query in the automatic search. Moreover, we checked the latest work of the authors of these primary DEPO4IOT studies and their related work to find more DEPO4IOT studies (using Google scholar). We found three new primary studies that have not been found in the automatic search process (because the common keywords were not in their titles and abstracts). In total, we obtained the final set of 17 primary studies for data extraction and synthesis to answer the research questions.

III. A TAXONOMY FOR CLASSIFICATION

To classify, analyze, and compare the primary studies, we have developed a taxonomy of DEPO4IOT (Fig. 2). Our taxonomy consists of the key aspects of deployment and orchestration for IoT that we know of, *e.g.*, from analyzing the initial set of primary studies, and extracting relevant concepts from [5]. We present them in the following categories: deployment and orchestration support (Section III-A), design support (III-B), and advanced support (III-C).

A. Deployment and orchestration support

Here are the key deployment and orchestration aspects.

1) Deployment support:

***Deployment type**:* Deployment typically comes in two flavors: imperative and declarative ([S17], Table I). The imperative approach requires a "deployment plan" that details how to reach the desired deployment, usually using a workflow language. In contrast, the declarative approach only requires a specification of the desired deployment which a "deployment engine" then computes how to reach.

***Target infrastructure**:* IoT systems are typically running over heterogeneous infrastructures ranging from tiny devices (the *device layer*) over gateways (the *edge/fog layer*) to cloud resources (the *cloud layer*).

***Network**:* Connected software components, deployed over the whole IoT, edge and cloud space, must properly interact with each other, *e.g.*, by custom addressing, segmentation of launched Virtual Machines (VMs), or software-defined networking, virtual network. Besides, we consider if an approach supports for specific communication protocols among IoT devices, such as Wifi, Bluetooth.

***Cloud Provisioning**:* It is not only important to distinguish tools that actually support cloud resources provisioning from others but also to identify which virtualization layer (*e.g.*, IaaS) their provisioning engine supports.

***Bootstrap**:* A bootstrap is a basic executable program on a device, or a runtime environment, which the system in charge of the deployment relies on (*e.g.*, Docker).

2) Orchestration support:

***Communication support**:* we classify five different kinds of communication support in orchestration [6]: MessagePassing, SharedSpaces, MessageQueue, Publish-Subscribe, and RemoteInvocations (*e.g.*, Java RMI, Microsoft DCOM, CORBA).

***Integration support**:* The interactions between the software components that compose an orchestration or a deployment can be specified and managed at different levels of abstractions: logical ports (*e.g.*, port 22 should be open for SSH) or methods (*e.g.*, remote methods invocation).

3) Tool scope: If tool support is provided and the level of automation (auto, semi-auto, manual).

B. Design support

1) Specification language:

***Representation and Language type**:* A language may have one or more textual and graphical syntaxes, and can be general-purpose (GPL) or domain-specific (DSL).

***Programming Model**:* The selection of a programming model typically depends on the pragmatic of a tool. For instance, reactive and event-driven programming is often used to design IoT systems, flow-based programming is well suited for orchestrating data flows between IoT devices and services, whilst component oriented programming is typically used to specify deployment models.

2) Specification Capabilities:

***Application and Deployment Structures**:* Modelling a deployment typically overlaps with the specification of the system's structure. Indeed, to actually allow for the deployment of a system on a selected target infrastructure, its application components (hereafter called *entities*) need to be allocated to IoT, edge or cloud resources.

C. Advanced supports

To address trustworthiness aspects, the DEPO4IOT approaches often require advanced supports such as adaptation, monitoring, and shared access to resources.

1) Trustworthiness: refers to the preservation of security, privacy, safety, reliability, and resilience of systems [7].

2) Adaptation: There are two main approaches for dynamic adaptations [8]: (i) parametric adaptation and (ii) compositional adaptation. Parametric adaptation allows modifying the system's behavior by tuning parameter values. This type of adaptation requires the adaptation parameters to be defined at design-time. In contrast, compositional adaptation allows the "hot" deployment and binding of software components that were not necessary foreseen before the initial deployment of the system. It is worth noting that both types of adaptation can be seamlessly combined. Orchestration and deployment tools can support these two types of adaptation for modifying: (i) the application itself (*e.g.*, replacing one software component by another) and (ii) its infrastructure (*e.g.*, bursting from one cloud to another).

3) Monitoring: Monitoring is a key activity to reason on the state of a system and for controlling and managing hardware as well as software infrastructures [9]. Monitoring probes are used to deliver information and indicators characterizing the system and the context in which it is running. In this study, we identify the following purposes of monitoring:

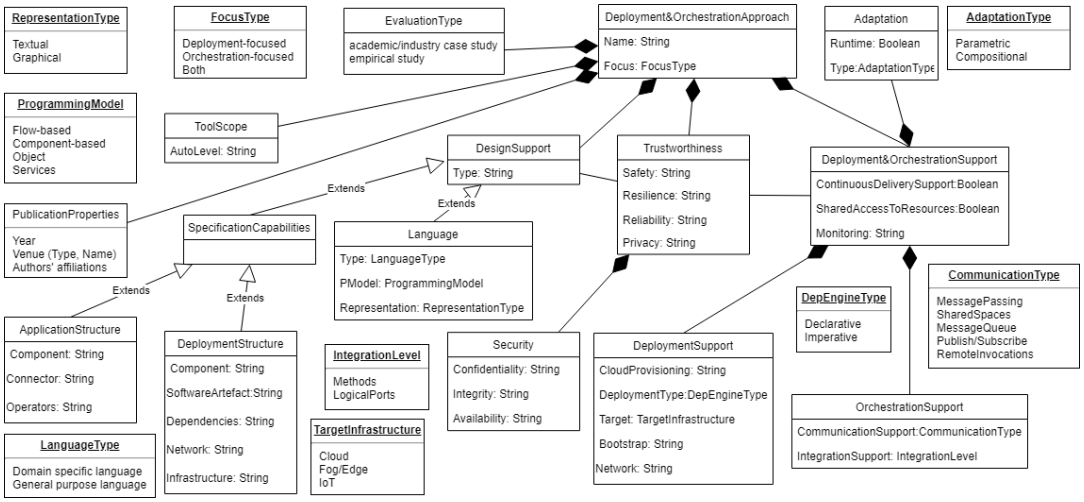

Figure 2. A Taxonomy of Deployment and Orchestration for IoT

(i) measuring QoS of the system, (ii) capturing the status and health of a deployment, (iii) depicting the state of the environment, and (iv) observing the execution flow of the system, for instance for debugging purpose.

4) Shared access to resources: Because IoT systems may involve actuators, it is important to control the impact these actuators can have on the physical world and to manage conflicting actuation requests. More generally, this applies to the management of shared accesses to resources, which can be of two types: (i) direct concurrent access to resources (*e.g.*, several entities accessing to the same service/actuator) or (ii) indirect shared access to a resource (*e.g.*, actuators from different applications are modifying the temperature with possibly conflicting goals) [10].

IV. Results

Table I shows an overview of the 17 primary DEPO4IOT studies. We have conducted in-depth analyses on these studies based on the taxonomy in Section III. These analyses give the answers to our research questions as follows.

A. RQ1 - On techniques and approaches

Answering RQ1.1, Fig. 3 shows a sharp rise in the number of primary DEPO4IOT studies very recently. More than half of the studies (nine out of 17) were just published in 2017, which could indicate this research area is taking off to match with the important role of deployment and orchestration for IoT. One reason that DEPO4IOT challenges are only receiving more attention recently is that so far IoT research might have focused more on fundamental IoT technologies, *e.g.*, approaches and tools for the development of IoT software components, communication protocols. But, once the IoT development approaches have advanced, the challenges of deployment and orchestration of IoT have popped up, which require more systematic approaches in deployment and orchestration.

Table I
THE PRIMARY DEPO4IOT STUDIES*

#	Year	Study	Title (clickable to open the corresponding publication)	PV
S1	2017	FogTorch	QoS-Aware Deployment of IoT Applications Through the Fog	J
S2	2017	ARCADIA	A Middleware for Mobile Edge Computing	J
S3	2017	Chen et al.	A Dynamic Module Deployment Framework for M2M Platforms	C
S10	2017	SoPIoT	A Novel Service-Oriented Platform for the Internet of Things	C
S11	2017	Calvin	Calvin Constrained: A Framework for IoT Applications in Heterogeneous Environments	C
S14	2017	Niflheim	Niflheim: An end-to-end middleware for applications on a multi-tier IoT infrastructure	C
S15	2017	Verba et al.	Platform-as-a-service gateway for the Fog of Things	J
S16	2017	Foggy	Foggy- A Framework for Continuous Automated IoT Application Deployment in Fog Computing	C
S17	2017	TOSCA-BMWi	A TOSCA-based Programming Model for Interacting Components of Automatically Deployed Cloud and IoT Applications	C
S12	2016	Cloud4IoT	Cloud4IoT: A Heterogeneous, Distributed and Autonomic Cloud Platform for the IoT	C
S7	2015	D-NR	Developing IoT Applications in the Fog: a Distributed Dataflow Approach	C
S9	2015	WComp	A Generic Service Oriented Software Platform to Design Ambient Intelligent Systemss	C
S13	2015	xWoT	A component based approach for the Web of Things	W
S5	2014	BeC3	BeC3: Behaviour Crowd Centric Composition for IoT applications	J
S8	2014	glue.things	glue.things - a Mashup Platform for wiring the Internet of Things with the Internet of Services	W
S6	2013	SAaaS	Application deployment for IoT: An infrastructure approach	C
S4	2011	D-LITE	D-LITE: Distributed logic for internet of things services	C

*PV: Publication venue; J: Journal; C: Conference; W: Workshop; *Sorted by publication year*

On another note, IoT is a heterogeneous research area that spans in multiple relevant research domains such as Software Engineering (SE), Cloud or Service-Oriented Architecture (SOA), Network, and IoT itself. Fig. 4 shows the times each research domain is the main topic in the calls for papers of the publication venues where the primary DEPO4IOT studies are published. It is not surprising to see the dominance of IoT topic in the publication venues of the primary DEPO4IOT studies (nine in total in Fig. 4). But, the other related research domains are sharing publication venues with IoT (SE: four, Clould/SOA: four, Network: four). Note that some venues do not really have a specific dominant research domain, thus classified as multiple-domain publication venues. Even not absolute, these numbers do reflect the heterogeneous nature of IoT research, especially regarding DEPO4IOT.

Answering RQ1.2, Because of our selection criteria, most of the primary studies have their primary contributions in deployment for IoT. Even though a few primary studies

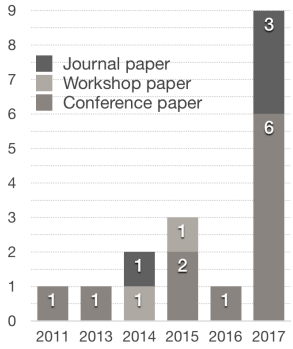

Figure 3. Publication of the primary studies per year

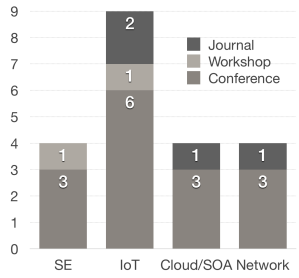

Figure 4. Research topics per publication venue

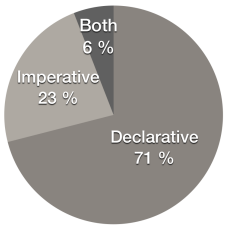

Figure 5. Deployment engine

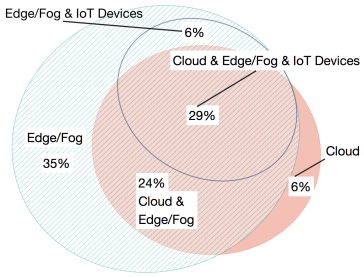

Figure 6. Target infrastructure

(six out of 17) such as ARCADIA [S2], SoPIoT [S10], or WComp [S9], have orchestration as their primary focus, deployment is still present in these approaches, *e.g.*, in forms of mechanisms for the dynamic loading (deployment) of WComp or OSGi components.

We would expect modern IoT systems leveraging cloud computing should have used cloud provisioning techniques in the DEPO4IOT approaches. However, only five approaches mention cloud provisioning ([S1, S2, S10, S14, S15, S17]). This observation illustrates that so far there is a lack of approaches/tools specifically designed for supporting the vertical depth of different IoT layers, *i.e.*, from cloud to fog/edge and to IoT devices.

After analyzing the data about deployment engine, we observe that more than two-third (71%, Fig. 5) of the primary studies have declarative deployment type vs. less than one-fourth (23%) have imperative type. We would argue this is because imperative approaches are typically more complicated to specify and reuse (as they require the specification of a deployment plan). On the other hand, they give full control over the deployment process, thus allowing its optimization. By contrast, declarative approaches better supports evolving and reusing deployment models, however the deployment engine may not compute the optimal deployment process. One primary study [S17] has leveraged both declarative and imperative deployment types. This is because the authors of TOSCA-BMWi [S17] have focused on understanding and providing both declarative and imperative deployment types to TOSCA [11].

Fig. 6 shows how the DEPO4IOT studies focus on the different layers of infrastructure: cloud, edge, or IoT devices. Most studies (65% in total) have the deployment and orchestration focus on cloud (6%) or edge/fog (35%) or both (24%). Few studies (29% + 6%) mention orchestrating and deploying software on IoT devices. We would argue that deployment and orchestration on IoT devices are the most challenging research problems in DEPO4IOT because of the diversity of IoT devices, their network protocols, and their different computing resource constraints [1]. To really support for modern IoT systems in which trustworthy aspects are crucial, DEPO4IOT studies must advance to the technical details of edges and IoT devices. Only in this way, IoT engineering can control the whole chain of IoT software deployed from cloud until IoT devices. Besides the heterogeneity, a huge challenge to bring IoT software engineering down to the low level of IoT devices could be the involvement of "black-box" software components or devices in IoT systems. Even worse, among the few studies that support DEPO4IOT at IoT devices level, we find the technical details at IoT devices level very limited. SAaaS [S6], BeC3 [S5] and D-LITE [S4] mention about managing and supporting deployment at IoT devices but no technical details are given. Niflheim [S14] also focuses more on the technical details of cloud and edge levels. Only Calvin Constrained (called Calvin for short)[8] [S11] and SoPIoT [S10] provide some technical details at IoT devices level such as the service abstraction of devices in SoPIoT or hardware-specific features supported by Calvin.

Most of the studies provide information about the bootstraps that the deployment approaches rely on. Besides common (open source) runtime environments such as OSGi, Docker, Node-RED, Node.js, Python, there are solution-specific runtime environments that have been developed together with the DEPO4IOT approaches such as Calvin runtime [S11], WComp container [S9], or D-LITE [S4]. In other words, we find two trends of using bootstraps in DEPO4IOT. One trend uses main-stream execution environments, *e.g.*, Docker, Node.js, SSH and OS, which make them somehow easier to adopt (in the sense that it is easier to find a target with these main-stream execution environments). Another trend relies on more specific execution environments, *e.g.*, Node-RED (D-NR [S7]), Calvin [S11], which can offer extra services compared to generic solutions. It is worth to note that this is typically the case of middleware, where the middleware itself must be running or installed on the target host. In this case, the mechanisms such as dynamic component loading or class loading are typically used, *e.g.*, Node-red, WComp [S9], OSGi.

Network specification among IoT elements is an important part of deployment and orchestration. Most of the primary

[8]https://github.com/EricssonResearch/calvin-base/wiki/Tools

approaches use straightforward network addressing. Only one approach provides an advanced support of software defined networking [S15]. Less than half of the approaches provide some information about the supported communication protocols of the IoT devices such as wifi, bluetooth, ZigBee, Xbee. These approaches do not really have explicit network specification and device communication protocols support but rather just briefly mention them. This observation is understandable because very few primary studies really support at IoT device level.

We can see that the communication types in orchestrating IoT components are diverse and have a fair share in use, excepts *Shared space (SS)*, which is rarely used. Overall, the approaches offering the best decoupling in time and space [6] (between subscribers and publisher), *i.e.*, message passing, message queues and publish/subscribe, are largely adopted. Two studies TOSCA-BMWi and Cloud4IoT, which tick all the boxes for five communication types, are independent from communication used.

The integration level at methods for orchestration is more common than at logical port level. Only three studies Niflheim [S14], Verba *et al.* [S15], and TOSCA-BMWi [S17] support both levels, which often needed for orchestrating more complex IoT systems. Indeed, orchestration is typically concerned with the behavior of the system, which often has the integration at the method level based on the understanding of the semantics behind the methods. Vice versa, a deployment approach may not need to care about the actual behavior of the system being deployed but just assumes the behavior is correct. Here, the objective of deployment is to enable the communication between the elements of the system, which can be done at logical port level, not necessary at method level.

Answering RQ1.3: The language support is important to make DEPO4IOT approaches more practical. Using domain-specific languages (DSL) is much more popular than using general purpose languages (GPL). Two studies FogTorch and Foggy use GPL such as Java to implement their deployment algorithms together with the IoT systems (FogTorch) or some components of the deployment framework (Foggy). Among the studies that use DSLs, there are some DSLs that are common such as the graphical DSL for flow-based programming in the approaches based on Node-RED (*e.g.*, D-NR [S7], glue.things [S8]). We can also see that textual form and graphical form are equally popular in language support. There are some approaches that do propose both graphical and textual formats [S4, S5, S7, S9, S11, S12, S17]. Component-based model and service-oriented model are the most common programming models in the DEPO4IOT approaches, followed by flow-based. Indeed, DEPO4IOT needs to provide a modular, loosely-coupled specification of the orchestration and its deployment so that the modules can be seamlessly substituted and reused.

In most of the primary DEPO4IOT studies, there is often an overlap between deployment and orchestration specifications. Even though, when the focus is deployment, orchestration specification stays at a higher level of abstraction (*e.g.*, microservice vs. node in ARCADIA [S2]) and the opposite for the approaches focusing on orchestration, deployment specification stays at a higher level of abstraction. Checking the deployment structure, we find that most approaches do provide some details about the software artifacts used ranging from node implementation (abstract) to jar file or docker image (detail). It is worth noting that the deployment mechanisms of DEPO4IOT approaches are not all technology agnostic as for some it is tight to the underlying platform (*i.e.*, for some it is only possible to deploy instances of nodes or components implemented using the DEPO4IOT approaches – *e.g.*, using D-NR [S7] the deployable software artifact has to be an implementation of Node-RED node). Plain support for network specification (simply addressing) is common. Verba et al. [S15] is the only approach that provides software defined networking to support the configuring of multiple networking connections that not only allow message passing but management and application deployment as well.

B. RQ2 - On the support for trustworthiness

Answering RQ2.1: Very few primary DEPO4IOT studies address trustworthiness aspects: security (three studies), privacy (two), resilience (four), reliability (four), safety (zero). Worse, the trustworthiness aspects are only briefly mentioned in these few studies, neither with details nor in explicitly systematic manner. For example, regarding security, we do not find any primary studies that have put their approaches into the context of a Security Development Lifecycle[9]. Among three studies that mention security, Calvin [S11] needs other runtimes to provide security infrastructure. But, Calvin does have application access control in form of token data to decide whether the action of an actor (*e.g.*, sensor or actuator) is allowed to run or not. Moreover, Calvin runtime uses Distributed Hash Table based registry by default for metadata about runtimes, actors, ports, etc. Security is more visible in glue.things [S8], which uses OAuth as open standard for authorization. Based on OAuth, glue.things provides fine grained policy and visibility management to define API tokens for controlling API access. With API tokens, glue.things supports individual views, access policies and privacy enforcement. It is not clear how privacy enforcement is done in glue.things though. D-NR [S7] briefly considers Authentication in the deployment, *i.e.*, an Authentication node must be deployed in the local network. The Authentication node authenticates the SensorTag based on its MAC address so that, *e.g.*, only a specific SensorTag can control the lights.

[9]Microsoft's SDL, https://www.microsoft.com/en-us/sdl

Reliability and resilience each is mentioned in four studies. Only FogTorch [S1] explicitly addresses both reliability and resilience. The other studies SoPIoT [S10], Niflheim [S14], D-NR [S7], WComp [S9], and Verba *et al.* [S15] only slightly mention reliability and/or resilience. None addresses safety, which should be crucial for critical IoT systems. In the IoT context, safety is often linked with actuation conflicts, which can occur, *e.g.*, when concurrent applications have a shared access to the actuators.

Answering RQ2.2: Shared access to resources is a rare feature because only two studies [S9, S14] have mentioned it. Even so, these two approaches have not gone further to provide real technical solutions. This is a hard problem that requires more extensive research work. The uncertain, dynamic, and partially known nature of the physical environment makes it very difficult to assess at runtime the conformity of the effects of actions in this environment with deterministic models.

All the studies ARCADIA [S2], Foggy [S16], and WComp [S9] that mention monitoring support also provide adaptation support. ARCADIA [S2], Foggy [S16] have provided technical details of their monitoring approaches, such as the Resource Monitor module of Foggy can monitor the resource usage and dynamically adapt container placement, whilst ACADIA [S2] monitors the application to trigger lifecycle management actions (such as relocation, scaling, and termination). WComp [S9] simply monitors the appearance and disappearance of smart things in the environment, and then allows to automatically and dynamically compose multiple applications sharing common services according to the context evolution. Among the approaches that have adaptation support without monitoring, the purposes of adaptation vary. For example, Calvin can update the actor placements and auto-scale to handle a varying workload or to replicate actors onto the available runtimes. TOSCA-BMWi [S17] leverages the OSGi framework Equinox, which allows to add and start new plugins, even during the runtime of the service bus. Some other approaches provide adaptation support for dynamic orchestration such as FogTorch [S1], SoPIoT [S10], Cloud4IoT [S12], BeC3 [S5] and D-LITE [S4]. SAaaS [S6] does not touch trustworthiness aspects and advanced supports at all.

Answering RQ2.3: Eight studies ([S2, S3, S4, S5, S8, S11, S12, S17]) have provided tool support for full automation in deployment and orchestration process. Other eight ([S1, S6, S7, S9, S10, S13, S14, S16]) have semi-auto level in tool support, *e.g.*, the bootstraps cannot be installed automatically. Only one study ([S15]) has manual level in tool support, which means the tool has not been implemented but only the tool framework proposed.

Finally, looking at the affiliations of the authors, we find that a majority of the authors working on DEPO4IOT are academics (14 primary studies). Three papers have the authors from both academia and industry. Regarding the types of case studies used for evaluating the DEPO4IOT approaches, we also find the similar dominance of academic approaches. We classify the case studies that are not from industry as academic ones, *e.g.*, motivational examples, prototypes or simulations developed by researchers for discussing or evaluating their DEPO4IOT approaches. Only WComp [S9] was driven by industrial case study. All the other primary studies are evaluated using academic case studies. Thus, the collaboration between industry and academia in this research is still very limited.

C. RQ3 - On the research challenges

Our answers to RQ3 are based on the findings presented in the answers to RQ1 and RQ2. Even though there is a big jump in the number of primary DEPO4IOT studies in the year 2017 compared to the previous years, our analyses show that DEPO4IOT research is still in its infancy. There are fundamental technical gaps and open issues to be tackled.

The trend of compute moving from cloud towards the edge and "things" is obvious but regarding DEPO4IOT, cloud provisioning is rare and does not cover the vertical depth of different IoT layers, *i.e.*, from cloud to fog/edge, to IoT devices. It is worth to note that most of the primary DEPO4IOT approaches do not support deployment and orchestration at IoT devices. For example, network communication and IoT device communication protocols are not considered, or only briefly mentioned in a few primary studies. Not covering the vertical depth of all IoT layers is a problem because in practice, supporting deployment and orchestration at (resource-constrained) IoT devices/gateways is very challenging [12]. Only by going into low-level details, IoT engineering can really control the whole chain of IoT software deployed from cloud until IoT devices. In this way, it will be more likely that the trustworthy aspects and advanced supports can be addressed more systematically and efficiently. The current status shows that the existing approaches are immature in addressing trustworthy aspects and advanced supports. Based on these observations, we would propose to do DEPO4IOT research focusing on the real, low level technical details of deployment and orchestration, especially deployment. New DEPO4IOT research needs to address more thoroughly and systematically the trustworthy aspects and advanced supports for modern IoT systems.

The dominance of academia-only in DEPO4IOT research shows that there is a big gap to make the proposed DEPO4IOT approaches more practical and closer to the needs in industry. There should be more collaboration between academia and industry in DEPO4IOT research.

V. RELATED WORK

Our work so far is the only systematic review of the deployment and orchestration approaches for IoT. From the software architecture view, an IoT middleware provides a layer between application software and system software.

DEPO4IOT approaches are more vertical along the IoT engineering life-cycle from development to operation of IoT systems. Thus, DEPO4IOT approaches are not necessary about middleware. Research on IoT middleware highly overlaps with DEPO4IOT because DEPO4IOT approaches often leverage middlewares for integrating heterogeneous computing and communications devices, and supporting interoperability within the diverse applications and services running on these devices. The authors of [13] and [14] conducted two different (not systematic) surveys of the existing middlewares to classify them and find out the main challenges. In [14], the authors analyzed some main approaches of IoT middleware categorized into three key architectures, *i.e.*, consumer-centric cloud-based, light-weight actor-based, and heavy-weight service-based. The results of these studies not only address the functional aspects of IoT middleware but also quality aspects such as security, adaptation that we also considered in our work. [15] provides a gap analysis on the well-known IoT platforms. In particular, it identifies gaps related security (fine grained access control), cross-platform and cross-layer DSL to reduce threats on privacy and security. This study is not a systematic one and focuses on platforms maturity and usability. [16] is a systematic study on the key IoT architectural concerns. It highlights a set of challenges that is a combination of technical, human, financial and ethical aspects. The topic scope of [16] is complementary to our SLR. Even though not focusing on deployment, orchestration and trustworthiness, it shares some of the findings with our SLR, *e.g.*, the lack of support for security, or the need for runtime adaptation.

VI. CONCLUSIONS

Deployment and orchestration approaches for IoT should be advanced enough to support distributed processing and coordinated behavior across IoT, edge and cloud infrastructures and cope with vast heterogeneity, dynamicity and required scalability. In this paper, we have examined the existing deployment and orchestration approaches for IoT, by conducting a systematic review. After systematically identifying and reviewing 17 primary studies out of thousands relevant papers in this field, we have found out that 1) there is a sharp rise in the number of primary studies published in 2017; 2) however, there are still different challenges that seem to remain to be properly addressed such as proper support for the deployment and orchestration on the IoT devices and sensors level; 3) future IoT deployment and orchestration research should also focus on addressing the trustworthiness challenges and advancing its support for modern IoT systems. Moreover, there appears to be a lack of approaches and tools that are efficient and practical. More applied research as well as more research collaborations between academia and industry can be a way to improve on this aspect.

ACKNOWLEDGMENT

This work is supported by the H2020 programme under the grant agreement number 780351 (ENACT).

REFERENCES

[1] NESSI, "SOFTWARE CONTINUUM Recommendations for ICT Work Programme 2018," 2016.

[2] A. M. (Ed.), "Cyber physical systems: Opportunities and challenges for software, services, cloud and data," 2015.

[3] A. Dearie, "Software deployment, past, present and future," in *Future of Software Engineering, 2007. FOSE'07*. IEEE, 2007, pp. 269–284.

[4] D. Budgen and P. Brereton, "Performing systematic literature reviews in software engineering," in *Proceedings of the 28th international conference on Software engineering*. ACM, 2006, pp. 1051–1052.

[5] A. Bergmayr, U. Breitenbücher, N. Ferry, A. Rossini, A. Solberg, M. Wimmer, G. Kappel, and F. Leymann, "A systematic review of cloud modeling languages," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 22:1–22:38, Feb. 2018. [Online]. Available: http://doi.acm.org/10.1145/3150227

[6] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The many faces of publish/subscribe," *ACM computing surveys*, 2003.

[7] E. R. Griffor, C. Greer, D. A. Wollman, and M. J. Burns, "Framework for cyber-physical systems: Volume 1, overview," Tech. Rep., 2017.

[8] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and B. H. Cheng, "A taxonomy of compositional adaptation," *Rapport Technique numéro MSU-CSE-04-17*, 2004.

[9] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, 2013.

[10] M. Ma, S. M. Preum, W. Tarneberg, M. Ahmed, M. Ruiters, and J. Stankovic, "Detection of runtime conflicts among services in smart cities," in *Smart Computing (SMARTCOMP), IEEE International Conference on*, 2016.

[11] U. Breitenbücher, T. Binz, K. Képes, O. Kopp, F. Leymann, and J. Wettinger, "Combining declarative and imperative cloud application provisioning based on tosca," in *Cloud Engineering (IC2E), 2014 IEEE International Conference on*. IEEE, 2014, pp. 87–96.

[12] M. Vögler, J. M. Schleicher, C. Inzinger, and S. Dustdar, "A scalable framework for provisioning large-scale iot deployments," *ACM Transactions on Internet Technology (TOIT)*, vol. 16, no. 2, p. 11, 2016.

[13] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things: a survey," *IEEE Internet of Things Journal*, vol. 3, no. 1, pp. 70–95, 2016.

[14] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "Iot middleware: A survey on issues and enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017.

[15] J. Mineraud, O. Mazhelis, X. Su, and S. Tarkoma, "A gap analysis of internet-of-things platforms," *Computer Communications*, vol. 89, 2016.

[16] A. Q. Gill, V. Behbood, R. Ramadan-Jradi, and G. Beydoun, "Iot architectural concerns: a systematic review," in *Proceedings of the Second International Conference on Internet of things and Cloud Computing*. ACM, 2017, p. 117.