Cloud-based modeling in IoT domain: a survey, open challenges and opportunities

Felicien Ihirwe[¶ §], Arsene Indamutsa[§], Davide Di Ruscio[§], Silvia Mazzini[¶], and Alfonso Pierantonio[§]
[§]Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, L'Aquila, Italy
[¶]Innovation Technology Services Lab, Intecs Solutions S.p.A, Pisa, Italy
[§]{*arsene.indamutsa,davide.diruscio,alfonso.pierantonio*}@*univaq.it*
[¶]{*felicien.ihirwe,silvia.mazzini*}@*intecs.it*

***Abstract*—The current evolution of cloud-based computing opens up a lot of possibilities for software development. In the near future, complex systems in various domains such as Space, Automotive, Internet of Things, and Smart Cities, will be designed, developed, and deployed from cloud-based environments, hence lowering production and maintenance costs. However, in the IoT domain, parts of the system have to run on Edge, Fog, or Cloud, posing significant difficulties in determining what, where, and when to develop. Therefore, this paper conducted a thorough review to investigate where the IoT domain community stands concerning the current trend of moving traditional modeling infrastructures to the cloud. Following an examination of 625 articles, we focus on 22 different cloud-based IoT system development approaches. Moreover, we highlight various opportunities and challenges related to the adoption of cloud-based modeling tools and platforms in the IoT domain.**

***Index Terms*—Model-driven engineering, low-code, cloud-based modeling, Internet of Things.**

I. Introduction

Cloud-based modeling is one of the relevant topics in the model-driven engineering community due to the induced possibilities of designing, developing, analyzing and deploying applications seemingly with reduced efforts. This has also been recently favored by the increasing adoption of low-code development platforms (LCDP). The recent Lowcomote project[1] aims at shifting the use and development of LCDPs to advanced levels by bridging together different research fields, including Model-Driven Engineering, Cloud Computing, and Machine Learning. Ideally, domain-specific low-code development platforms have to run on cloud infrastructures, even though in some industrial settings such as IoT, domain-specific modeling environment tends to be local-based [1]. Nowadays, industries and companies are trying to migrate their modeling infrastructures to the cloud. However, especially in industrial contexts, the existing modeling infrastructures are implemented in complex environments in which the migration cost can be far more expensive and very complicated.

The future of modeling will forcefully be cloud-based [2]. Several initiatives, including Visual Studio Code[2], Eclipse Che[3], Theia[4], and others have shown a lot of potential in shifting modeling environments from local-based and monolithic installations to cloud-based platforms in order to eliminate accidental complexity and expand the variety of available functionalities [2].

In the IoT domain, modeling and development infrastructures need to consider several heterogeneous aspects of the system's data, communication, and implementation layers. This paper looks at what has been done so far in the IoT domain to support IoT systems' development through cloud-based modeling approaches. In particular, we conducted a thorough investigation to see where the IoT community stands concerning the current trend of moving traditional modeling infrastructures to the cloud. Following an examination of 625 articles, we identified 22 different cloud-based IoT system development tools and platforms. We perform an analysis of the various issues that the IoT community is encountering while implementing cloud-based modeling tools. As a result, we take a deeper look at a few options and discuss the research and development opportunities enabled by adopting cloud-based modeling approaches in the IoT domain.

The remainder of this paper is organized as follows: Section II provides an overview of cloud-based modeling approaches and highlights motivations and needs for modeling IoT systems by means of dedicated cloud-based environments. Section III presents the research methodology we have used to conduct the survey. Sections IV–VI discuss the findings of the performed analysis, which has been performed to answer three dedicated research questions. Section VII discusses the related work, whereas Section VIII concludes the paper.

II. Background

The significant advancements in computing power, data storage and processing are revolutionizing the development and research of complex systems in several domains, including that of the Internet of Things (IoT) [3]. IoT systems enable the integration of intelligent features into daily human activities through the automation of services. In particular, such systems allow automation of low-level services that used to be error-prone if done by humans. Moreover, they increase efficacy in

This work has received funding from the Lowcomote project under European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement n° 813884.

[1]https://www.lowcomote.eu/

[2]https://code.visualstudio.com

[3]https://www.eclipse.org/che/

[4]https://theia-ide.org

current engineering solutions and connect a range of many devices that render our environment smart. Recent reports predict that more than 100 billion devices will be connected by 2025 and 11 trillions dollars of global market capital will be reached [4]. However, to unleash the full potential of these systems, it is necessary that also citizen developers can take part in the development of custom IoT applications [1].

The development and consumption of IoT systems are becoming way more complex, and involving end-users is more challenging due to the heterogeneity of the hardware and required expertise [1]. This complexity originates from various sources. IoT applications are complex systems that use heterogeneous devices and data sources. Besides, IoT systems require enormous efforts and investments both in their implementation and maintenance. Moreover, the systems are implemented using code-centric approaches that make it challenging to foster the inclusion of IoT domain experts and other stakeholders with less IoT programming skills [1].

Due to the ever-changing requirements and the shortage of engineering experts that develop these systems robustly and securely [5], the way forward entails the need to pave the way for domain experts and other stakeholders to integrate IoT capabilities in their daily tasks [1]. Several approaches are being discussed, while practical solutions are finding a way to facilitate IoT application usage and development very accessible. Model-driven engineering (MDE) promotes the systematic use of models as the primary abstraction entities all along the development of complex systems by fostering abstraction and automation [6]. Models in the context of MDE are not sketches, drawings that serve purpose only in design, but they prevail until the end of the development cycle of these systems as machine-readable and processable abstractions [7]. MDE favors collaboration of engineers and stakeholders, as both work together toward the completion of the conceived products and foster integration of different engineering processes [5].

However, MDE itself has faced challenges that have shifted the focus of the development of such complex and heterogeneous systems from local environments to the cloud [8]. Modeling-as-a-Service is gaining momentum as the MDE research community is migrating modeling tools and services to the cloud. This migration is encouraged by several out-of-box benefits in cloud computing, such as easy discovery and reuse of services and artifacts [6]. It has enabled efficient self-healing mechanisms to detect, diagnose, and countermeasure threats and foster collaboration among stakeholders and engineers [9]. Furthermore, migrating modeling artifacts and services on the cloud can facilitate end-users easy accessibility, hence supporting sustainable management and disaster recovery of model artifacts and tools [10].

III. Study design

This paper aims to analyze how the IoT domain is coping with the trend of moving existing modeling and development infrastructures to the cloud. To this end, we followed the process shown in Fig. 1 according to the methodology presented in [11]. In particular, the search and selection process was mainly conducted into four main phases. In the first phase, we formally and explicitly represented the problem to get a head start on the search. Second, we defined a search string and selected well-known academic search databases. Third, we performed a search to gather papers to answer properly defined research questions. Fourth, we narrowed down the potential papers and mapped them based on their similarity and variability. Finally, we analyzed the collected papers and elaborated some recommendations on the identified difficulties.

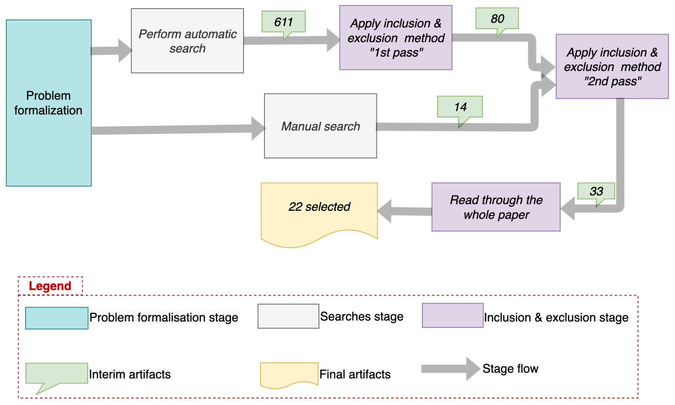
Problem formalization
Perform automatic search
611
Apply inclusion & exclusion method "1st pass"
80
Apply inclusion & exclusion method "2nd pass"
Manual search
14
33
Read through the whole paper
22 selected
Legend
Problem formalisation stage
Searches stage
Inclusion & exclusion stage
Interim artifacts
Final artifacts
Stage flow

Fig. 1. Search and selection process

Phase 1: Problem formalisation - This phase mainly focused on formalizing the problem we wanted to solve by looking at the current model-driven engineering tendency. One of the sources of inspiration for this study was the work in [2] which addresses the topic of *"what is the future of modeling?"*. Thus, we came up with the formulation of the following research questions:

- ***RQ1:*** *How is the IoT community adopting cloud-based modeling approaches?*
- ***RQ2:*** *What challenges do researchers face when developing cloud-based IoT modeling and development infrastructures?*
- ***RQ3:*** *What are the main potential opportunities laying ahead for future researchers and developers in the IoT domain?*

Phase 2: Automatic & manual search: In this phase, we applied a search string to different academic databases, i.e., Scopus (Elsevier)[5], IEEE Xplore[6] and ACM library[7] by limiting the search on the last 10 years. Additionally, we also performed a manual search, primarily using Google Scholar. The query string we used for the automatic search was: `("MDE" OR "Model Driven Engineering") AND ("IoT" OR "Internet of Things") AND ("Cloud" OR "Web")`.

[5]https://www.elsevier.com/
[6]https://ieeexplore.ieee.org
[7]https://dl.acm.org/

Table I shows the number of papers we managed to collect in this phase.

TABLE I
RESULTS TABLE

Database	Results
Scopus (Elsevier)	233
IEEE Xplore	263
ACM library	115
Manual search	14
Total	**625**

Phase 3: Inclusion & exclusion, 1st pass: Table I shows that 611 publications were initially discovered from different sources, in addition to the 14 papers that were manually found and considered relevant for the study. At this point, we have reviewed the paper's title, keyword, and abstract to exclude papers that were not satisfying the following criteria:

- Studies published in a peer-reviewed journal, conference, or workshop.
- Studies written in English.
- Papers that focus explicitly on the Internet of Things (IoT) topic.
- Studies that propose a cloud-based modeling approach, either explicitly or implicitly.

At the end of this step, we had 80 papers to be added to additional 14 documents manually retrieved from Google Scholar.

Phase 4: Inclusion & exclusion, 2nd pass: In this phase, we read the introduction and the conclusion of the papers previously collected. We also removed some duplicates. Various documents were rejected during this phase for a variety of reasons, for instance, because the presented approach is not explicitly offering an IoT-based cloud-based development environment. At the end of this phase, we ended up with 33 documents.

Phase 5: Reading of the whole paper text: We've gone over the entire articles in this phase, focusing on the proposed approaches and their evaluation sections. Several documents were discarded because of different reasons. For instance, papers that presented hybrid solutions (e.g., enabling local modeling with the possibility of storing models on remote repositories) were discarded. In addition, the approaches that claim to build web-based IoT data-wrangling platforms by reusing existing IoT data storage platforms were also discarded. Finally, we selected 22 documents that leverage a cloud-based modeling environment to design, develop, or deploy IoT applications.

Figure 2 shows the distribution of the selected approaches with respect to their corresponding sources. As you might notice from Fig. 2, a portion of the selected approach (4 out of 22) was found manually. This is because of our previous knowledge on this topic in terms of framework and tools. In the following, the research questions presented in Sec. III are answered singularly by analyzing the research papers that have been collected as previously described.

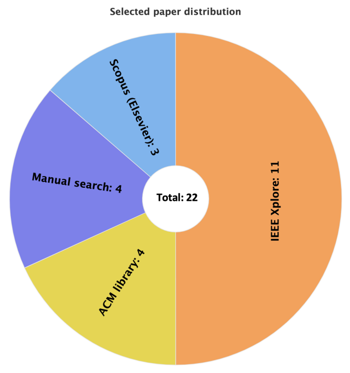

Fig. 2. Selected paper distribution

IV. Cloud-based modeling approaches (RQ1)

This section goes over different cloud-based modeling approaches that target the IoT domain. We organized the analysed approaches into three categories according to their main focus of interest i.e., modeling IoT structural aspects, service-oriented approaches, and deployment orchestrations. The aim is to answer the research question ***RQ1:*** *How is the IoT community adopting cloud-based modeling approaches?*

Modeling IoT structural aspects: DSL-4-IoT [12] is a cloud-based modeling tool for the IoT domain, which comprises a JavaScript-based graphical frontend programming language and a runtime "OpenHAB" execution engine. DSL-4-IoT provides a multistage model-driven approach for the design of IoT applications that supports all stages of the life cycle of these systems. Automatic model transformations are provided to refine abstract models elements into concrete ones. Those transformation results formatted as JSON-Arrays are passed to the OpenHAB runtime engine for execution.

BIoTA [13] offers a cloud-based modeling approach for IoT architectures. A graphical DSL and supporting tools allow users to perform syntax and semantic analysis. BIoTA renders it possible to computationally formalize a software architecture suggested by a user according to formal automata techniques. The component & connectors are created following specific rules to meet IoT-specific scenarios while exporting the resulting software architecture towards a Docker-based deployment infrastructure.

Node-RED [14] is a popular and extensible model-driven framework for tying together IoT devices, APIs, and web services in a homogenous manner. It provides users with a Web-based graphical editor with drag-and-drop facilities. In Node-RED, the user can also create and deploy real-time dashboards.

AutoIoT [15] is a Web-based platform with a Graphical User Interface (GUI) that allows programmers to deploy and configure IoT systems quickly. The final system-generated artifact is a Flask project[8] (a python micro-framework) that can be run as is or extended to meet the needs of developers that might require more complex functionalities. The final system can also connect to an MQTT Broker, storing and querying data in a database, presenting data to users, and exchanging them with other systems. AutoIoT had later been extended in [16] to allow users to model their IoT systems in terms of JSON files.

In [17], a cloud-based textual language and tool for Event-based Configuration of Smart Environments (ECSE) had been proposed. The tool enables the end-user, being expert or not, to configure a smart environment by employing an ontology-based model. In their approach, the authors used the Resource Description Frameworks (RDF) to define the event-action rules.

AtmosphericIoT [18] is a cloud-based domain-specific language and tools for building, connecting, and managing IoT systems. AtmosphericIoT Studio is a free online IDE that lets you create all kinds of device firmware, mobile apps, and cloud dashboards. It links devices via Wi-Fi, Bluetooth, BLE, Sigfox, LoRa, ZigBee, NFC, satellite, and cellular networks.

In [19], authors propose a model-based approach for creating responsive and configurable Web of things user interfaces. Models@Runtime are used to produce runtime interfaces based on a formal model named Thing Description (TD). TD's goal is to expose Web Things (WT) attributes, actions, and events to the outside ecosystem. The modeling language has been developed in JavaScript, using the VueJS framework, and it is publicly available[9].

In [20], the authors presented a model-driven approach to the development of IoT system interfaces. In their work, they proposed a design pattern and the required components for designing such interfaces. The authors implemented a platform for the development of IoT mobile and web applications based on WebRatio,[10] a generic cloud-based model-driven development and code generation framework.

FloWare Core [21] is a model-driven open-source toolchain for building and managing IoT systems. FloWare supports the Software Product Line and Flow-Based Programming paradigms to manage the complexity in the numerous stages of the IoT application development process. The system configures the IoT application following the IoT system model supplied by the IoT developer. A Node-RED engine [14] is integrated in FloWare.

Vitruvius [22] is an MDD platform that allows users with no programming experience to create and deploy complex IoT web applications based on real-time data from connected vehicles and sensors. Users can design their ViWapplications straight from the web using a custom Vitruvius XML domain-specific language. Furthermore, Vitruvius provides a variety of recommendation and auto-completion features that aid in creating applications by reducing the amount of XML code to be written.

[8]https://flask.palletsprojects.com

[9]https://github.com/smar2t/td_interface_builder

[10]https://www.webratio.com/

Service-oriented approaches: This category includes approaches providing users with cloud-based modeling environments targeting service-oriented architectures. Thus, different services are connected to build the final IoT systems.

MIDGAR [23] is an IoT platform specifically developed to address the service generation of applications that interconnect heterogeneous objects. This is achieved by using a graphical DSL in which the user can interconnect and specify the execution flow of different things. Once the desired model is ready, it gets processed through the service generation layer, generating a tree-based representation model. The model is then used to generate a Java application that can be compiled and run on the server.

IADev [24] is a model-driven development framework that orchestrates IoT services and generates software implementation artifacts for heterogeneous IoT systems while supporting multi-level modeling and transformation. This is accomplished by converting requirements into a solution architecture using attribute-driven design. In addition, the components of the produced application communicate using RESTful APIs.

LogicIoT [25] offer a textual web-based DSL to ease data access and processing semantics in IoT and Smart Cities settings. LogicIoT is implemented as a set of custom Jakarta Server Pages (JSP)[11] in which different custom JSP tags have been implemented to define the modeling semantics. The language consists of seven constructs: relations, triggers, endpoints, timers, facts, rules, and modules. Using the custom tags, the user can define the application's operations required to enable the communication between process instances and sensors without being concerned with low-level programming details.

glue.things [26] offers a cloud-based mashup platform for wiring data of Web-enabled IoT devices and Web services. glue.things take care of both the delivery and maintenance of device data streams, apps, and their integration. In this regard, glue.things rely on well-established real-time communication networks to facilitate device integration and data stream management. The glue.things modeling tool combines device and real-time communication, allowing users to describe element's triggers and actions and deploy them in a distributed manner.

In [27], the authors proposed a framework for scalable and real-time modeling of cloud-based IoT services in large-scale applications, such as smart cities. IoT services are modeled and organized in a hierarchical manner.

In [28], a portable web-based graphical end-user programming environment for personal apps is proposed. This tool allows the users to discover smart things in their environment and create personalized applications that represent their own needs. Each of the defined smart objects can provide various features that can be published via a well-defined API. The graphical representation of the system is then generated from the constructed JavaScript objects in which the user can interact with the system on the fly.

[11]https://en.wikipedia.org/wiki/Jakarta_Server_Pages

E-SODA [29] is a cloud-based DSL under the Cloud-Edge-Beneath (CEB) architecture ecosystem. In E-SODA, a cloud sensor comprises a set of Event/Condition/Action (ECA) rules that define the sensor service life-cycle. It allows the user to be abstract and simulate sensor behavior through an events-based fashion. This is achieved by having the ECA rules listen for the occurrence of a predetermined "event" and respond by performing the "action" if the rule's "condition" is met. Finally, the generated cloud sensor application can be used in any cloud-based application which needs sensor data.

In [30], the authors introduced an integrated graphical programming tool based on a goal-driven approach, in which end users are only required to specify their purpose in a machine-understandable manner, rather than designing a service architecture that fulfills their goal. This allows a smart environment's ultimate purpose to be graphically represented, but the complexities of the underlying semantics are hidden. A reasoning component uses the provided goal statement and analyses whether the goal can be achieved given the set of available services and infers whatever user actions (i.e., requests involving REST resources) are required to achieve it.

InteroEvery [31] promotes a microservice-based architecture to deal with interoperability issues of the IoT domain. First, an IoT system is configured through a web-based graphical interface showing each microservice's functionalities. A universal broker connects a dedicated interoperability microservice with various adaption microservices depending on employed choreography patterns.

Model-based deployment orchestration: As IoT system deployment happens at different layers of abstractions, this section presents the identified approaches, which aim to orchestrate the deployment mechanisms of IoT systems using cloud-based modeling environments.

DoS-IL [32], [33] is a textual domain scripting language for resource-constrained IoT devices. It allows changing the system's behavior after deployment through a lightweight script written with the DoS-IL language and stored in a gateway at the fog layer. The gateway hosts an interpreter to execute DoS-IL scripts.

TOSCA (Topology and Orchestration Specification for Cloud Applications) [34] aims at improving the reusability of service management processes and automate IoT application deployment in heterogeneous environments. In TOSCA, common IoT components such as gateways and drivers can be modeled. In addition, the gateway-specific artifacts necessary for application deployment can also be specified to ease the deployment tasks.

GENESIS (Generation and Deployment of Smart IoT Systems) [35] is a textual cloud-based domain-specific modeling language that supports continuous orchestration and deployment of Smart IoT systems on edge, and cloud infrastructures. GENESIS uses component-based approaches to facilitate the separation of concerns and reusability; therefore, deployment models can be regarded as an assembly of components. The GENESIS execution engines support three types of deployable artifacts, namely ThingML model [36], Node-RED container [14] and any black-box deployable artifact (e.g., an executable jar). The created deployment model is subsequently passed to GENESIS deployment execution engine, which is in charge of deploying the software components, ensuring communication between them, supplying the required cloud resources, and monitoring the deployment's status.

Discussion: As previously presented, several approaches are available to support cloud-based modeling in the IoT domain. Table II shows an overview of the analyzed approaches; half of them are concerned with structural issues, whereas only a few deal with deployment concerns. The current state of the art suggests that there is no predominant common language, although graphical syntax is preferred.

Most of the analyzed approaches are supported by tools, which are not open source. This goes hand in hand with the public availability of the methodologies. We can observe that all the tools that are not open-source are also not publicly accessible. When looking at industrial settings, this is especially true when it comes to internal proprietary tools.

While analyzing each approach, we also looked at the supporting infrastructures and their ability to generate deployable artifacts. In this regard, we have discovered that JavaScript-based environments like Node.js and Angural.js are widely used for tool development. This might be due to the fact they are among the modern languages for front-end technology implementation. On the other hand, it appears that the majority of techniques generate artifacts, even though few of them are standalone deployable components. It is also worth noting that the generated deployable artifacts can only be deployed within the same original environment in most of the cases. To ensure interoperability, scalability, and reusability of the tools, the generated artifacts should generally be deployed anywhere.

V. Open challenges (RQ2)

Multiple issues have arisen as a result of the expansion of connected smart and sensor devices, as well as the increased usage of cloud-based models [9]. As a typical IoT system consists of multiple complex sub-systems, having an all-in cloud-based environment can become even more complicated. On the other hand, overcoming these barriers is worth the effort because it opens up more opportunities. This section elaborates on the current challenges IoT systems face while developing and integrating such tools in a cloud-based environment. Essentially, we are answering the research question ***RQ2****: What challenges do researchers face when developing cloud-based IoT modeling and development infrastructures?*

Extensibility mechanisms: Extensible platforms allow the addition of new capabilities without having to restructure the entire ecosystem. Because IoT systems are distributed, a typically recommended architecture would be to use the micro-service architecture throughout the development process [3]. Aside from that, IoT systems may require additional interactions with third-party technologies. As a result of the previous scenario, developing tools to design and develop

TABLE II
ANALYZED APPROACHES

Tool name	Category	Language syntax	Open-source	Tool availability	Underlying infrastructure	Generated artifact
DSL-4-IoT	Structure	Graphical	no	no	js, OpenHAB	JSON config
BIoTA	Structure	Graphical	no	no	Apache Tech. GraphQL	YAML file
IADev	Service	Textual	no	no	ASR,REST,ATL	REST app
Node-RED	Structure	Graphical	yes	yes	Node.js	Node-RED app
AutoIoT	Structure	Graphical & textual	no	no	Python, js	Flask app
[17]	Structure	Textual	no	no	Smart-M3	–
AtmosphereIoT	Structure	Graphical	no	yes	Multi-platform	Multi-platform apps
[19]	Structure	Textual	yes	yes	js,VueJS	UI code
[20]	Structure	Graphical	no	no	WebRatio, IFML	UI code
FloWare Core	Structure	Graphical	yes	yes	JavaScript	Node-RED Config file
Vitruvius	Structure	Textual	yes	no	XML,HTML,js	HTML5 with JavaScrit app
MIDGAR	Service	Graphical	no	no	Ruby,js,HTML, Java	Java app
LogicIoT	Service	Textual	no	no	JSP	-
glue.things	Service	Graphical	yes	no	AngularJS,Meshblu PubNub	NodeRED service
[27]	Service	Textual	no	no	Firebase&Node.js	–
TOSCA	Deployment	Textual	yes	yes	Multi-platform	Config files
[28]	Service	Graphical	no	no	-	-
E-SODA	Service	Textual	yes	no	OSGI cloud	OSGI java bundles
[30]	Service	Textual	yes	yes	ClickScript,AJAX	REST services
InteroEvery	Service	Graphical	no	no	Spring Boot,Rest RabbitMQ,Angular	–
DoS-IL	Deployment	Textual	no	no	js,HTML,DOM	Config files
GENESIS	Deployment	Textual	no	no	multi-platform	Genesis dep. agents

such distributed applications on the cloud need efficient tools that traditional domain specialists may not have. Accessibility mechanisms are presented through tools like [14], [21], but there is still a lot to be done. Currently, domain experts must provide cloud-based automation mechanisms and tools to allow citizen developers to add new features without requiring sophisticated knowledge or changing existing architectures.

Heterogeneity: It is an important challenge of the IoT domain, which involves different players developing various applications running at different layers, namely the edge, fog, and cloud [3]. In addition, deployments and data consumption methods are very diverse, increasing the complexity of traditional code-centric approaches [37]. Cloud-based modeling in IoT brings even more sophistication regarding the environment in which the system should be designed and developed. The typical cloud-based modeling platform should foster the integration of heterogeneous technological implementations, promoting reusability and developing solutions close to the problem domain. Approaches such as [23], [24], [34] have presented different strategies to tackle such issues, but much more have to be investigated.

Scalability: IoT systems are expected to handle a wide range of users, perform demanding computations, and share enormous amounts of data among nodes. Therefore, supporting cloud-based modeling approaches must be implemented in such a way that scalability concerns are mitigated. One of the approaches to tackle such challenges is to adopt container-based orchestration tools such as Kubernetes. The use of such tools can offer out-of-box features such as self-healing, fault-tolerance, and elasticity of containerized resources [4]. This will also help automate cognitive processes that can detect scalability needs and adjust autonomously without human intervention.

Interoperability: The interoperability of various tools, services, and resources is critical in the IoT domain. The interoperability of cloud-based modeling platforms, particularly in the IoT area, is currently limited since different tools run in different environments and have different natures. A tool like [31] promotes the micro-service architecture by allowing all parts of the system to communicate with each other. Several regulations, such as standardization, will need to be implemented to achieve interoperability among different cloud-based modeling environments. To address interoperability concerns, technologies like [14], [12], [15], [21] promote a common format based on JSON to encode models. It is worth noting that adopting Model-as-a-Service (MaaS) architectures could also promote the interoperability of services and artifacts.

Learning curve: It is not easy to find professionals who can master and combine the different sophisticated technologies involved in developing and managing IoT systems. IoT domain experts may lack modern programming expertise, whereas experienced software programmers may lack modeling domain expertise. For instance, conceiving a cloud-based code generator requires understanding different model transformation techniques and particular programming abilities; Implementing a visual mashup tool will necessitate knowledge of modern languages such as JavaScript, HTML, and CSS.

Security concerns: Current IoT systems suffer from security concerns as data are collected from a wide distribution of private and public nodes. Furthermore, the data is transferred using remote IoT gateways, which might get exposed in the process. This heterogeneity of secured and unsecured data might favor attackers to target devices and compromise the integrity of data and operations [38]. Therefore, proper abstractions and automation techniques are needed to help target users that might not necessarily have the required knowledge of the security practices to be employed.

VI. Research and development opportunities (RQ3)

In this section, we examine several opportunities that we think researchers and developers can leverage to improve the cloud-based development and management of IoT systems. Therefore, we aim at answering the research question ***RQ3:*** *What are the main potential opportunities laying ahead for future researchers and developers in the IoT domain?*.

A. Tools and platforms

Numerous tools and platforms are being built to tackle cloud-based modeling concerns. Thus, now is the right moment to suggest powerful and extensible tools that the IoT community may harness to solve their domain-specific issues. In this section, we look at various open-source and highly extensible platforms that are popular among the modeling community and that we would recommend for the IoT domain.

Cloud-based development tools based on Eclipse: We believe that a significant part of the MDE community, or at least for research purposes, uses Eclipse-based technologies. This is because most Eclipse projects and technologies are open-source, making them more accessible and encouraging individuals to participate. As of March 2021, the Eclipse Foundation hosts over 400 open source projects, 1,675 committers, and over 260 million lines of code have been contributed to Eclipse project repositories [39]. Through the Eclipse Cloud Development (ECD)[12] effort, the Eclipse community has demonstrated its willingness to transit a part of its ecosystem to the web. Eclipse's ECD Tools working group strives to define and construct a community of best-in-class, vendor-neutral open-source cloud-based development tools and promote and accelerate their adoption. Some of the best cloud-based technologies that the IoT community can benefit from are the following:

- *EMF.cloud, GLSP, Theia* - Independently from the Eclipse modeling framework (EMF), the EMF.cloud community recently expressed a strong desire to migrate the Eclipse-based modeling infrastructure to the cloud. This project aims to develop a web-based environment for creating modeling tools that can support the editing mechanisms of EMF-based models. EMF.cloud allows users to interact with models through the EMF.cloud model server, which coordinates the use of GLSP for graphical modeling, and LSP for textual modeling. Code generation infrastructures based on Eclipse Xtend are also included, while Eclipse Theia provides a web-based code editing and debugging infrastructure. Several resources are available in the community for extending those tools, and we believe that IoT developers may use such technologies to construct cloud-based IoT DSLs.
- *Sirius Web*[13] - It is an Eclipse Sirius-based modeling tool that provides a powerful and extensible graphical modeling platform for users to design and deliver modeling tools on the web. In Sirius Web, the ability to create your modeling workbench in a configuration file is supported. In this case, no code generation is required because everything is interpreted at run-time [40]. Furthermore, being open-source, Sirus Web provides greater accessibility and customizability than the desktop version, making it easier for the IoT community to get started with their cloud-based solutions.

Another alternative, such as Eclipse Che[14] makes Kubernetes development accessible for developer teams. Che is an in-browser IDE that allows you to develop, build, test, and deploy applications from any machine. Finally, Epsilon playground[15] has been recently launched to offer cloud-based tools for run-time modeling, meta-modeling, and automated model management.

Low-code development platforms: Looking at the LCDPs, the only powerful cloud-based open-source platform we would recommend is Node-RED [14]. Due to its high extensibility and accessibility, Node-RED offers an excellent IoT system mashup environment in which IoT systems can be designed, developed, and deployed on the fly. The Node-RED platform is open, and IoT system developers can build their custom nodes, compile, test, and deploy them in the Node-RED ecosystem. Several extensions have been made, such as [41] tackling the reusability issues in cloud-based modeled components, [42] to tackle the heterogeneity and complexity challenges found in the Fog based development. Finally, in [43], the authors presented SHEN to enable self-healing capabilities of applications based on Node-RED. In terms of interoperability, Node-RED models are represented as JSON objects, which any third-party tools can easily consume. Some of the tools in this domain, such as FloWare [21] and GENESIS [35] already support the Node-RED models, which shows a great sign of its high impact. Table III outlines the essential characteristics of the recommended platforms.

B. Benefits of cloud-based modeling

Although there are difficulties in adopting a cloud-based modeling approach in the IoT domain, various opportunities will emerge, making the investment worth it. This section outlines various opportunities that will emerge once cloud-based modeling is widely adopted in the IoT domain.

[12]https://ecdtools.eclipse.org/

[13]https://www.eclipse.org/sirius/sirius-web.html

[14]https://www.eclipse.org/che/

[15]https://www.eclipse.org/epsilon/live/

TABLE III
RECOMMENDED TECHNOLOGIES

	EMF.cloud	GLSP	Theia	Che	Node-RED
Open-source	✓	✓	✓	✓	✓
Extensible	✓	✓	✓	✓	✓
Scalable	✓	✓	✓	✓	✓
IoT-specific	—	—	—	—	✓
Application	Web-based EMF modeling tools	Graphical language-server-editor	Web-based code editor	Kubernetes-native IDE for DSL deployment	Flow-based programming

1) User communities: Adopting cloud-based modeling in the IoT domain will have the potential of attracting more citizen developers, and it will unravel a lot of modeling opportunities on different devices such as tablets and mobile devices [44], [28], [45].

2) Collaborative modeling: Once IoT modeling infrastructures are moved to the cloud, it can be necessary to introduce collaborative modeling features to simplify the interaction of both developers and stakeholders. Unfortunately, none of the discussed approaches provide collaborative modeling functionalities. However, collaborative modeling can harness the power of real-time information, artifacts, and service exchange.

3) Productivity: Empowering users to develop their applications by embracing cloud-based modeling is a head-start toward high production and reducing time to market [1]. Users can create applications that cater to their problems, and engineers focus on developing features that facilitate the user for a smooth development at an appropriate abstraction level. In addition, the participants will focus on problem-solving in their particular domain and avoid wasting time and resources on solving problems that are outside their competencies.

4) Maintenance: Traditional code-centric methodologies necessitate a significant investment in the ongoing maintenance of developed systems. In addition, systems require regular upgrades and installations, which can be error-prone and time-consuming. During upgrades or troubleshoots times of the system, sometimes system downtime is necessary, impacting production. Furthermore, the growing need for software systems in our daily lives and constantly changing user requirements required an agile approach for addressing these issues quickly without compromising system availability or user access. In many cases, such challenges are handled by cloud providers, leaving developers and engineers to focus on developing applications that directly impact customer demands [46].

5) Monitoring and debugging: Cloud-based modeling enables monitoring of activities and their archive through its cloud providers. This is a head-start when debugging distributed applications because developers can track down the microservices, which are the root of the detected problems. Without appropriate cloud infrastructures, it would be challenging to solve these issues, even with features such as self-healing and repair strategies. Current cloud-based solutions come bundled with monitoring tools that assist in problem diagnosis and monitor the usage of the applications.

VII. RELATED WORK

We identified several surveys papers on MDE and DSL in the IoT domain throughout our paper selection process. Still, very few of them focuses explicitly on cloud-based MDE approaches ([47], [1], [48], [49] to mention a few). In this paper, we are interested in examining the possible approaches helping in migrating the classical local-based MDE in IoT technologies to the cloud and its adoption.

Our previous study [50] looked at the current state of low-code engineering (LCE) adoption in the IoT domain. LCE combines LCDPs, MDE, machine learning, and cloud computing to facilitate the application development life-cycle, namely from design, development, deployment, and monitoring stages for IoT applications. A comparable set of features has been identified by examining sixteen platforms to represent the functionalities and services that each of the investigated platforms could support. We discovered that just 7 of the 16 could be deployed on the cloud, with the majority of them being LCDPs, whereas classical MDE approaches rely on a local-based design paradigm.

In [51], the authors conducted a comprehensive assessment of model-based visual programming languages in general before narrowing their focus to 13 IoT-specific visual programming languages. The research was carried out based on their characteristics, such as programming environment, licensing, project repository, and platform support. According to a comparison of such features, 72% of open-source projects are cloud-based, whereas only 17% percent of closed-source platforms are cloud-based, which confirms a strong uptrend of cloud-based systems in open-source IoT projects.

In [52], the authors discussed tools and methods for creating Web of Things services, in particular, mashup tools as well as model-driven engineering approaches. The techniques regarding expressiveness, suitability for the IoT domain, and ease of use and scalability have been analysed. Although this study is related to this paper, it solely focuses on mashup tools and only includes a few approaches. We can observe from the preceding discussion that only a few techniques attempted to explore cloud-based MDE approaches implicitly. According to this, and to the best of our knowledge, this is the first study analyzing the status of cloud-based modeling in the IoT domain.

VIII. CONCLUSION

To develop IoT applications, developers must overcome various challenges, including heterogeneity, complexity, and scalability. Moving development infrastructure to the cloud will open up plenty of new opportunities regarding accessibility, productivity, maintenance, and monitoring. In this paper, we conducted a systematic study to assess the current state of the art on cloud-based modeling approaches in the IoT domain. We looked at 22 papers proposing cloud-based modeling environments in the IoT domain. The considered approaches have been analyzed to assess their strengths and weaknesses concerning many characteristics, including their modeling focus, accessibility, openness, and artifact generation. Throughout the paper, we have discussed many challenges that IoT developers encounter while adopting such tools. We also discussed various generic technologies and tools, which can be adopted in the IoT domain.

References

[1] A. J. Salman, M. Al-Jawad, and W. A. Tameemi, “Domain-Specific Languages for IoT: Challenges and Opportunities,” *IOP Conference Series: Materials Science and Engineering*, vol. 1067, no. 1, p. 012133, 2021.

[2] A. Bucchiarone, F. Ciccozzi, L. Lambers, A. Pierantonio, M. Tichy, M. Tisi, A. Wortmann, and V. Zaytsev, “What is the future of modeling?” *IEEE Software*, vol. 38, no. 02, pp. 119–127, mar 2021.

[3] M. Marjani, F. Nasaruddin, A. Gani, A. Karim, I. A. T. Hashem, A. Siddiqa, and I. Yaqoob, “Big IoT Data Analytics: Architecture, Opportunities, and Open Research Challenges,” *IEEE Access*, vol. 5, pp. 5247–5261, 2017.

[4] L. Farhan, S. T. Shukur, A. E. Alissa, M. Alrweg, U. Raza, and R. Kharel, “A survey on the challenges and opportunities of the Internet of Things (IoT),” *Proceedings of the International Conference on Sensing Technology, ICST*, vol. 2017-Decem, no. December, pp. 1–5, 2017.

[5] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*, 2012.

[6] A. Indamutsa, D. D. Ruscio, and A. Pierantonio, “A Low-Code Development Environment to Orchestrate Model Management Services,” *Advances in Production Management Systems*, 2021.

[7] L. Berardinelli, A. Mazak, O. Alt, and M. Wimmer, *Model-Driven Systems Engineering: Principles and Application in the CPPS Domain*, 05 2017, pp. 261–299.

[8] D. Di Ruscio, M. Franzago, I. Malavolta, and H. Muccini, “Envisioning the future of collaborative model-driven software engineering,” *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017*, no. May, pp. 219–221, 2017.

[9] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, “A vision of IoT: Applications, challenges, and opportunities with China Perspective,” *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 349–359, 2014.

[10] Á. Hegedüs, G. Bergmann, C. Debreceni, Á. Horváth, P. Lunk, Á. Menyhért, I. Papp, D. Varró, T. Vileiniskis, and I. Ráth, “Incquery server for teamwork cloud: Scalable query evaluation over collaborative model repositories,” *21st ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings, MODELS-Companion 2018*, pp. 27–31, 2018.

[11] J. Biolchini, P. G. Mian, A. C. C. Natali, and G. H. Travassos, “Systematic review in software engineering,” *System Engineering and Computer Science Department COPPE/UFRJ, Technical Report ES*, vol. 679, no. 05, p. 45, 2005.

[12] A. Salihbegovic, T. Eterovic, E. Kaljic, and S. Ribic, “Design of a domain specific language and ide for internet of things applications,” in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 996–1001.

[13] F. F. Borelli, G. O. Biondi, and C. A. Kamienski, “Biota: A buildout iot application language,” *IEEE Access*, vol. 8, pp. 126 443–126 459, 2020.

[14] Node-RED, “Node-red: Low-code programming for event-driven applications,” https://nodered.org/, 2020, last accessed May 2020.

[15] T. Nepomuceno, T. Carneiro, T. Carneiro, C. Korn, and A. Martin, “A gui-based platform for quickly prototyping server-side iot applications,” in *Smart SysTech 2018; European Conference on Smart Objects, Systems and Technologies*, 2018, pp. 1–9.

[16] T. Nepomuceno, T. Carneiro, P. H. Maia, M. Adnan, T. Nepomuceno, and A. Martin, “Autoiot: A framework based on user-driven mde for generating iot applications,” in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, ser. SAC ’20. New York, NY, USA: Association for Computing Machinery, 2020, p. 719–728.

[17] J. Kiljander, J. Takalo-Mattila, M. Etelapera, J.-P. Soininen, and K. Keinanen, “Enabling end-users to configure smart environments,” in *2011 IEEE/IPSJ International Symposium on Applications and the Internet*, 2011, pp. 303–308.

[18] AtmosphereIoT, “Fast time to first data,” https://atmosphereiot.com/, 2020, last accessed May 2020.

[19] J. D. H. Bezerra and C. T. de Souza, “A model-based approach to generate reactive and customizable user interfaces for the web of things,” in *Proceedings of the 25th Brazillian Symposium on Multimedia and the Web*, ser. WebMedia ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 57–60.

[20] M. Brambilla, E. Umuhoza, and R. Acerbis, “Model-driven development of user interfaces for IoT systems via domain-specific components and patterns,” *Journal of Internet Services and Applications*, vol. 8, no. 1, p. 14, 2017.

[21] F. Corradini, A. Fedeli, F. Fornari, A. Polini, and B. Re, “FloWare: An Approach for IoT Support and Application Development,” in *Enterprise, Business-Process and Information Systems Modeling*, A. Augusto, A. Gill, S. Nurcan, I. Reinhartz-Berger, R. Schmidt, and J. Zdravkovic, Eds. Cham: Springer International Publishing, 2021, pp. 350–365.

[22] G. Cueva-Fernandez, J. P. Espada, V. García-Díaz, C. G. García, and N. Garcia-Fernandez, “Vitruvius: An expert system for vehicle sensor tracking and managing application generation,” *Journal of Network and Computer Applications*, vol. 42, pp. 178–188, 2014.

[23] C. González García, B. C. Pelayo G-Bustelo, J. Pascual Espada, and G. Cueva-Fernandez, “Midgar: Generation of heterogeneous objects interconnecting applications. a domain specific language proposal for internet of things scenarios,” *Computer Networks*, vol. 64, pp. 143–158, 2014.

[24] W. Rafique, X. Zhao, S. Yu, I. Yaqoob, M. Imran, and W. Dou, “An application development framework for internet-of-things service orchestration,” *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4543–4556, 2020.

[25] M. Sneps-Sneppe and D. Namiot, “On web-based domain-specific language for internet of things,” in *2015 7th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2015, pp. 287–292.

[26] R. Kleinfeld, S. Steglich, L. Radziwonowicz, and C. Doukas, “Glue.things: A mashup platform for wiring the internet of things with the internet of services,” in *Proceedings of the 5th International Workshop on Web of Things*, ser. WoT ’14. New York, NY, USA: Association for Computing Machinery, 2014, p. 16–21.

[27] A. Taherkordi and F. Eliassen, “Scalable modeling of cloud-based iot services for smart cities,” in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, 2016, pp. 1–6.

[28] Y. Valsamakis and A. Savidis, “Personal Applications in the Internet of Things Through Visual End-User Programming,” in *Digital Marketplaces Unleashed*, C. Linnhoff-Popien, R. Schneider, and M. Zaddach, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018, pp. 809–821.

[29] Y. Xu and A. Helal, “Scalable cloud–sensor architecture for the internet of things,” *IEEE Internet of Things Journal*, vol. 3, no. 3, pp. 285–298, 2016.

[30] S. Mayer, R. Verborgh, M. Kovatsch, and F. Mattern, “Smart configuration of smart environments,” *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 3, pp. 1247–1255, 2016.

[31] B. El Khalyly, M. Banane, A. Erraissi, and A. Belangour, “Interoevery: Microservice based interoperable system,” in *2020 International Conference on Decision Aid Sciences and Application (DASA)*, 2020, pp. 320–325.

[32] B. Negash, T. Westerlund, A. M. Rahmani, P. Liljeberg, and H. Tenhunen, “DoS-IL: A Domain Specific Internet of Things Language for Resource Constrained Devices,” *Procedia Computer Science*, vol. 109, pp. 416–423, 2017.

[33] B. Negash, T. Westerlund, P. Liljeberg, and H. Tenhunen, “Rethinking ‘Things’ - Fog Layer Interplay in IoT: A Mobile Code Approach,” in *11th International Conference on Research and Practical Issues of Enterprise Information Systems (CONFENIS)*, vol. LNBIP-310. Shanghai, China: Springer International Publishing, Oct. 2017, pp. 159–167.

[34] F. Li, M. Vögler, M. Claeßens, and S. Dustdar, “Towards automated iot application deployment by a cloud-based approach,” in *2013 IEEE 6th International Conference on Service-Oriented Computing and Applications*, 2013, pp. 61–68.

[35] N. Ferry, P. Nguyen, H. Song, P.-E. Novac, S. Lavirotte, J.-Y. Tigli, and A. Solberg, “Genesis: Continuous orchestration and deployment of smart iot systems,” in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1, 2019, pp. 870–875.

[36] N. Harrand, F. Fleurey, B. Morin, and K. E. Husa, “Thingml: A language and code generation framework for heterogeneous targets,”

in *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 125–135.

[37] M. Noura, M. Atiquzzaman, and M. Gaedke, "Interoperability in Internet of Things: Taxonomies and Open Challenges," *Mobile Networks and Applications*, vol. 24, no. 3, pp. 796–809, 2019.

[38] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of Things security and forensics: Challenges and opportunities," *Future Generation Computer Systems*, vol. 78, pp. 544–546, 2018. [Online]. Available: http://dx.doi.org/10.1016/j.future.2017.07.060

[39] E. Foundation, "2020 annual eclipse foundation community report," 2020, last accessed July 2021. [Online]. Available: https://www.eclipse.org/org/foundation/reports/annual_report.php

[40] M. Bats and S. Begaudeau, "Sirius web: 100platform," 2020, last accessed July 2021. [Online]. Available: https://www.eclipsecon.org/2020/sessions/sirius-web-100-open-source-cloud-modeling-platform

[41] A. Restivo, H. S. Ferreira, J. P. Dias, and M. Silva, "Visually-defined real-time orchestration of iot systems," 2020.

[42] N. K. Giang, M. Blackstock, R. Lea, and V. C. Leung, "Developing iot applications in the fog: A distributed dataflow approach," in *2015 5th International Conference on the Internet of Things (IOT)*, 2015, pp. 155–162.

[43] J. P. Dias, A. Restivo, and H. S. Ferreira, "Empowering visual internet-of-things mashups with self-healing capabilities," *arXiv preprint arXiv:2103.07395*, 2021.

[44] L. Brunschwig, E. Guerra, and J. de Lara, "Towards access control for collaborative modelling apps," in *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, ser. MODELS '20. New York, NY, USA: Association for Computing Machinery, 2020.

[45] Y. M. Antorini and A. M. Muñiz, "The Benefits and Challenges of Collaborating with User Communities," *Research-Technology Management*, vol. 56, no. 3, pp. 21–28, 2013. [Online]. Available: https://doi.org/10.5437/08956308X5603931

[46] O. David, W. Lloyd, K. Rojas, M. Arabi, F. Geter, J. Ascough, T. Green, G. Leavesley, and J. Carlson, "Model-as-a-service (MaaS) using the Cloud Services Innovation Platform (CSIP)," *Proceedings - 7th International Congress on Environmental Modelling and Software: Bold Visions for Environmental Modeling, iEMSs 2014*, vol. 1, pp. 243–250, 2014.

[47] G. Fortino, C. Savaglio, G. Spezzano, and M. Zhou, "Internet of things as system of systems: A review of methodologies, frameworks, platforms, and tools," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 223–236, 2021.

[48] A. Wortmann, B. Combemale, and O. Barais, "A systematic mapping study on modeling for industry 4.0," in *Proceedings of the ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS '17. IEEE Press, 2017, p. 281–291.

[49] S. Teixeira, B. A. Agrizzi, J. G. P. Filho, S. Rossetto, and R. de Lima Baldam, "Modeling and automatic code generation for wireless sensor network applications using model-driven or business process approaches: A systematic mapping study," *Journal of Systems and Software*, vol. 132, pp. 50–71, 2017.

[50] F. Ihirwe, D. D. Ruscio, S. Mazzini, P. Pierini, and A. Pierantonio, "Low-code engineering for internet of things: a state of research," in *MODELS '20: ACM/IEEE 23rd International Conference on Model Driven Engineering Languages and Systems, Virtual Event, Canada, 18-23 October, 2020, Companion Proceedings*, E. Guerra and L. Iovino, Eds. ACM, 2020, pp. 74:1–74:8. [Online]. Available: https://doi.org/10.1145/3417990.3420208

[51] P. P. Ray, "A Survey on Visual Programming Languages in Internet of Things," *Scientific Programming*, vol. 2017, p. 1231430, 2017.

[52] C. Prehofer and I. Gerostathopoulos, "Chapter 3 - Modeling RESTful Web of Things Services: Concepts and Tools," in *Managing the Web of Things*, Q. Z. Sheng, Y. Qin, L. Yao, and B. Benatallah, Eds. Boston: Morgan Kaufmann, 2017, pp. 73–104.