A feature-based survey of Fog modeling languages

Abdelghani Alidra, Hugo Bruneliere, Thomas Ledoux

A Feature-based Survey of Fog Modeling Languages

Abdelghani Alidra[a,b,*], Hugo Bruneliere[a], Thomas Ledoux[b]

[a] *NaoMod Team - IMT Atlantique, LS2N (UMR CNRS 6004) - Nantes, France*
[b] *STACK Team - IMT Atlantique, Inria, LS2N (UMR CNRS 6004) - Nantes, France*

Abstract

Fog Computing is a new paradigm aiming at decentralizing the Cloud by geographically distributing away computation, storage and network resources as well as related services. In order to design, develop, deploy, maintain and evolve Fog systems, languages are required for properly modeling both their entities (e.g., infrastructures, topologies, resources configurations) and their specific features such as the locality concept, QoS constraints applied on resources (e.g., energy, data privacy, latency) and their dependencies, the dynamicity of considered workloads, the heterogeneity of both applications and devices, etc. This paper provides a detailed overview of the current state-of-the-art in terms of Fog modeling languages. We relied on our long-term experience in Cloud Computing and Cloud Modeling to contribute a feature model describing what we believe to be the most important characteristics of Fog modeling languages. We also performed a systematic scientific literature search and selection process to obtain a list of already existing Fog modeling languages. Then, we evaluated and compared these Fog modeling languages according to the characteristics expressed in our feature model. As a result, we discuss in this paper the main capabilities of these Fog modeling languages and propose a corresponding set of open research challenges in this area. We expect the presented work to be helpful to both current and future researchers or engineers working on/with Fog systems, as well as to anybody genuinely interested in Fog Computing or more generally in distributed systems.

Keywords: Fog Computing, Cloud Computing, Internet of Things, Modeling Language, Survey

1. Introduction

Fog Computing [1, 2, 3] is a new paradigm aiming at decentralizing the Cloud by geographically distributing away computation, storage and network resources as well as related services. Instead of considering a huge centralized Cloud system, data centers of various sizes in a core Cloud network can be combined with smaller data-centers located at the Edge of the network and with related IoT devices. All these resources can be collaboratively used together to form a single *large-scale geo-distributed system*. Thanks to resource locality, such a Fog system [4] allows for better performance in terms of service latency, power consumption, network traffic or content distribution. The main gains are twofold: i) avoid network bottlenecks and single points of failure; ii) keep the data as close as possible to their sources (e.g., IoT devices) and to their final usage (i.e., end-users). This new type of system, integrating resources coming from a central Cloud to the edge of the network, is expected to facilitate many aspects of our daily lives by improving decision-making processes in various fields such as industry, transportation, health, education, cities, etc.

However, managing resources in such heterogeneous and dynamic large systems is highly challenging and requires fully decentralized solutions for scalability and reliability reasons. In order to support this, we need simultaneously new abstractions, languages, distributed algorithms and system mechanisms capable of operating and exposing a large number of diverse resources in a unified, efficient and sustainable way. One of the first challenge is notably to be able to properly model such large and complex Fog systems. To this end, appropriate modeling languages are required to be used by operators, DevOps and architects when designing, developing, deploying, maintaining or evolving their Fog systems. In other words, Fog modeling languages should help to better manage the Fog system's life cycle, including the digital infrastructure as well as the hosted applications.

The main objective of this paper is to provide orientation to the researchers or engineers genuinely interested in Fog Computing and Fog systems modeling. To this end, we propose a detailed study of the current state-of-the-art in terms of already existing Fog Modeling Languages, i.e., languages allowing to model Fog systems possibly having different types of architectures and different application domains. In practice, we first worked on a feature model to describe what we believe to be the main general characteristics of Fog modeling languages (i.e., independently of any specific language). In parallel, we set up

*Corresponding author
Email addresses: abdelghani.alidra@imt-atlantique.fr (Abdelghani Alidra), hugo.bruneliere@imt-atlantique.fr (Hugo Bruneliere), thomas.ledoux@imt-atlantique.fr (Thomas Ledoux)

and performed a systematic paper selection process from the scientific literature in order to obtain a list of existing approaches proposing Fog modeling languages. Then, we evaluated the selected Fog modeling languages according to the characteristics expressed in our feature model. Finally, we analyzed and discussed further the obtained results.

As a summary, we contribute to the community i) the description of what we believe to be the main general characteristics (expressed as a feature model) of Fog modeling languages ; ii) a presentation of how these characteristics are currently covered or not by already existing Fog modeling languages from the scientific literature; iii) the identification of present and future research challenges concerning such Fog modeling languages.

The remainder of the paper is organized as follows. In Section 2, we first introduce the general terminology that we illustrate via one concrete example of a Fog system. In Section 3, we describe the methodology we used for searching, identifying and selecting approaches providing relevant Fog modeling languages. In Section 4, we present our feature model summarizing the main characteristics of Fog modeling languages. Then, in Section 5, we evaluate and compare the selected Fog modeling languages thanks to the previously proposed terminology and feature model. In Section 6, we both discuss our analysis of the obtained results and identify open research problems or challenges related to Fog modeling languages. We close the paper with further discussion on the related work in Section 7 and a general conclusion in Section 8.

2. Background

2.1. General Definitions

We start by proposing general definitions of the main big concepts behind Fog modeling languages, as they will be used along the paper.

A **system** is a unit consisting of multiple interdependent components designed and implemented by different engineers. These components can be software or hardware ones, as well as any other artefacts created during the system development and running process.

A **Cloud system** is a particular type of system offering various kinds of computer system resources on-demand. These resources can be infrastructure, platform or software ones and are commonly centralized over data centers.

An **Internet of Things (IoT) system** is a particular type of system providing a network of physical components. These components (i.e., "things") can be devices, sensors, software or any technical component to connect and share data over networks.

A **Fog system** is a particular type of Cloud and IoT system where computation and storage can be distributed between different locations at the edge of the Network (i.e., closer to IoT devices providing data) rather than systematically centralized in data centers.

A **model** provides a representation of a given system. It specify the model elements describing this system according to the element's and relationship's types and constraints provided by a **metamodel** (i.e., a model conforms to a metamodel).

A **modeling language** has three main components: 1) An **abstract syntax** defining the concepts of the language (quite often expressed as a metamodel), 2) one or several **concrete syntaxes** (graphical and/or textual) for users to be able to specify corresponding models with the language and 3) **semantics** providing meaning to the language.

A **Fog modeling language** is a particular kind of modeling language dedicated to the creation of models of Fog systems and their specific characteristics.

2.2. Illustrative Example: a Fog system in the context of University Education

To concretely illustrate these general definitions and further motivate the need for Fog modeling languages, we describe in this section one example of a Fog system in the context of a hypothetical college campus consisting of a main department and a dormitory.

For the sake of campus modernization, the campus management board decided to deploy two distributed applications: one for smart surveillance and another for smart bell notification. A particular Fog system has been designed in order to maximize local resources utilisation, optimize latency and response time, and minimize the monetary costs relative to Cloud resources exploitation. while enforcing some privacy and data protection guaranties.

Firstly, the campus administrators designed the physical infrastructure (i.e., the blue elements in Figure 1) intended to host the virtual infrastructure. This physical infrastructure is spread across three layers. The *IoT layer* is made of small IoT devices (in our example WiFi cameras, alarms and users' personal devices) acting mostly as sensors and/or actuators communicating with the environment and users. They can be static or mobile, their network connection is often scarce and volatile. As the use of some of these devices (e.g., cameras) may imply privacy issues, their locations have been carefully chosen to respect users privacy, related information panels have also been placed wherever required. The *Fog layer* consists of physical equipment offering computation and/or storage capabilities, distributed across the network. In our example, there are a network gateway, two small data centers (one located in the main department and the other in the dormitory) and a cluster of several Raspberry Pis. The *Cloud layer* contains, in our example, a public Cloud provider with a very large amount of available resources (depending on budget and latency constraints) that can be exploited in an Infrastructure as a Service (IaaS) or a Platform as a Service (PaaS) way.

Secondly, on top of this physical infrastructure, the system administrators described the virtual platform and its characteristics in terms of Virtual CPU (VCPU), RAM

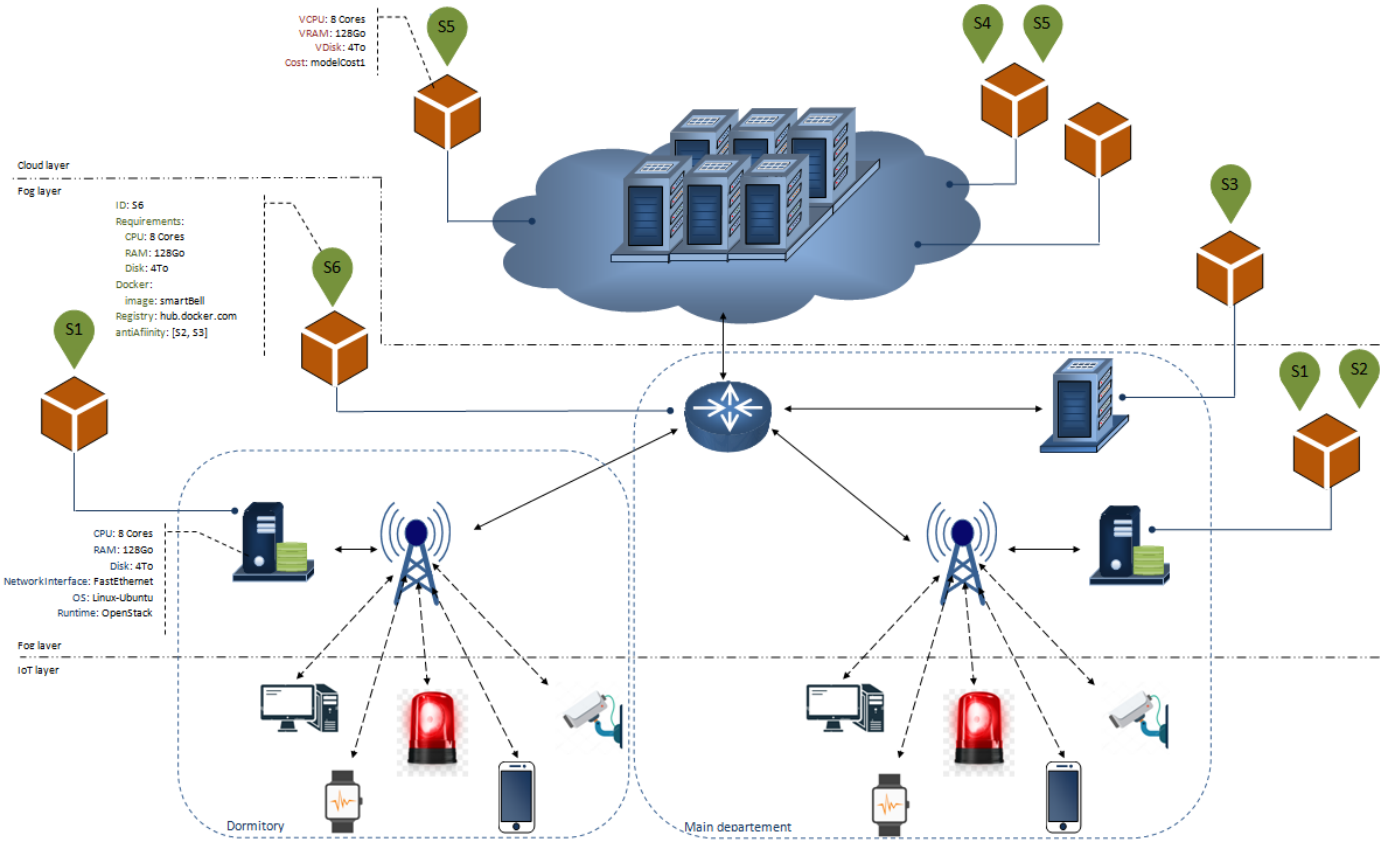

Figure 1: One example of a Fog system in the context of hypothetical college campus

(VRAM) and storage (VDisk). The corresponding set of Virtual Machines (VMs) and lightweight containers (i.e., the orange elements in Figure 1) aims at hosting the target services/applications.

Thirdly, the software architects model the two applications architecture (i.e., the green elements in Figure 1) as a set of loosely coupled micro-services that can be mutually shared between the two applications. This also includes their requirements/constraints (e.g., computation power, storage capacity, location) and application-specific objectives (e.g., latency, response time, cost, bandwidth usage). For the sake of privacy, hard constraints have been imposed on services placement. As a result, the raw data is only stored in local nodes and systematically anonymized (and eventually randomized) when required to be processed outside of the campus.

The *smart surveillance* application is composed of five micro-services. The low-level ones are (S1) Feature extraction micro-service to extract human faces in video or captured images, (S2) Audio analysis micro-service that identifies specific keywords in record audios and (S3) Licence plate recognition micro-service to identify and track cars. The high-level services include (S4) Gesture and facial recognition micro-service that recognizes and links specific persons or subjects to extracted faces and (S5) Behavior analysis micro-service to infer and detect suspicious behavior and malicious intent.

The *smart bell* application [5] is composed of four micro-services. The low-level ones are (S1) Feature extraction micro-service and (S6) Bell ringing decision micro-service making reaction decision for each received visitor. The high-level services are (S4) Gesture and facial recognition micro-service and (S7) Visitor habits analysis micro-service identifying different visitors habits by analyzing past recorded visits and visitation patterns.

In addition to modeling these physical resources, virtual platform and application micro-services, Fog modeling languages are also about describing how resources (physical or virtual) are affected to application micro-services.

The scheduling or placement of these services is of primary importance, as it will largely determine how well the system fulfills the designers' original goals. It must notably take into account the application requirements and constraints related to the infrastructure capacity or allocated budget (for instance). Other popular system reconfiguration techniques may also be considered such as network reconfiguration, auto-scaling or requests offloading [6].

3. Survey Method

3.1. Goal of the Survey and Research Questions

The objective of this paper is to identify, classify, compare and then analyze the core characteristics and related

support of already existing Fog modeling languages, in order to identify a number of open and relevant challenges from a research perspective.

To this end, we propose to answer to the following three main research questions (RQs):

RQ1 - What is the scope of the current Fog modeling languages?

RQ2 - How are they defined from a language engineering perspective?

RQ3 - What features and tooling support do they already offer?

3.2. Approach Selection Process

The paper selection process was conducted by exploring bibliographic data sources, following the general guidelines by Kitchenham and Charters [7] concerning literature search and selection strategy. At the end of the process, we performed an additional snowballing step thus double-checking the references cited in the finally selected publications. The overall selection process is summarized in Figure 2 and detailed in the following subsections[1].

3.2.1. General Search Strategy and Data Source (Step 0)

We used DBLP[2] as our core database to search for primary studies on Fog modeling languages. We took this decision because the main journals and conferences in the targeted scientific domain (i.e., distributed systems, Fog Computing) have their publications systematically indexed in DBLP. This notably covers papers from the most common publishers (IEEE, ACM, Springer, Elsevier, etc.). The quality of DBLP references is also generally acknowledged in the Computer Science community, contrary to other sources whose indexation process is less transparent.

As Fog Computing is a relatively recent paradigm, we decided to limit the search to the last decade (2012 corresponds to first appearances of the term Fog Computing). We decided to focus on approaches already published in journals, conferences and in serious workshops eventually (though we gave preference to related journal and conferences publications when possible). Thus, we generally removed publications of other kinds or coming from other types of venues. For automating the search process described here, we used the DBLP search facilities to select a first set of publications that we refined as detailed below. Based on the topic of this survey, we defined the terms of the search query as follows.

3.2.2. Whitelist-based Keyword Search (Step 1)

We considered the terms "Fog" and "model" and "language" as the main constituents of our search query. We also included the terms "architecture", "description" and "specification" as quite often used as alternatives to the term "model", as well as the terms "profile" and "domain" as sometimes associated to the modeling activity. We could have included other search terms, but most of them are already covered by the general term "model" (e.g., the related term "modeling"). As we used a very general search query, we expected a large result set requiring special treatment to select the most appropriate studies. When executing the search query over the DBLP dump, we received a complete result set consisting of 785 references. Based on this initial set of references, we determined the set of relevant publication sources by a semi-automated pruning process detailed in what follows.

3.2.3. Pruning Process (Steps 2 to 4)

In what follows (as proposed by Kitchenham and Charters [7]), we present the pruning steps applied to identify the final set of relevant papers. The selection of these primary studies was carried out based on the initial set of records we obtained from executing the search query against the DBLP data sources (cf. Section 3.2.2). In each one of these steps, the number of studies was significantly reduced compared to the result of the previous step.

Blacklist-based Keyword Search (Step 2). After the whitelist-based keyword search determining the initial set of studies, we conducted a blacklist-based keyword search. This step was needed as we found out that many papers which were part of the initial selection were actually not from the distributed systems domain (which is our target scientific domain). Notably, we automatically excluded all the papers from the Artificial Intelligence area, i.e., having the keywords "AI" or "learning", "neural" (neural networks are sometimes referred to as neural models, but these are out of the scope of our study), "stochastic" (stochastic models are mathematical models that can be used to represent some network or monitoring control phenomenons, but they are also out of the scope of our work). This resulted in a set of 421 papers to be investigated during the next step. Please note that the reduction to a human-manageable number of studies at this step was important. Indeed, the two remaining steps are hardly achievable automatically, and thus, were conducted manually.

Manual Selection Based on Title and Abstract (Step 3). It is often difficult to determine the relevance of a study by considering its title only. Thus, we decided to evaluate both the title and abstract of each study against inclusion and exclusion criteria. Studies relevant for this survey must clearly and explicitly describe a proposed Fog modeling language. We decided to not put particular restrictions on how such a language is specified and then implemented (all possible technological stacks are allowed). In other words, in this paper we are not interested in the Fog-supporting approaches/tools themselves but rather in the Fog modeling languages that they can provide.

From this overall principle, the **inclusion criteria** are: (1) Proposed approaches that report on or contribute related sets of concepts which are specifically intended to

[1] Further resources are also available from `https://cloud.imt-atlantique.fr/index.php/s/zeYJHTSwYyYQMkJ`.

[2] `http://www.dblp.org/search`

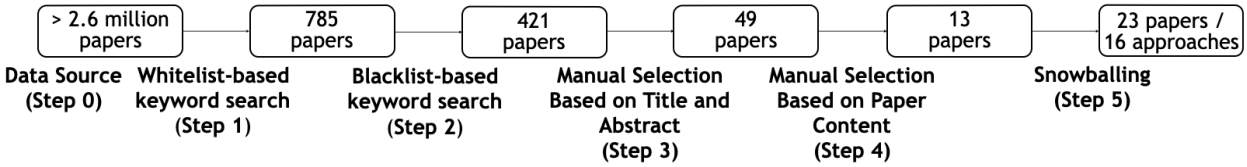

Figure 2: Overview of the used selection process.

model Fog systems. (2) Proposed approaches that allows to create models for different kinds of Fog systems, Fog (technical) environments or application domains (e.g., Smart City, Smart Health, Smart Home, etc.). (3) Proposed approaches that provide support to a Fog modeling language (tooling, documentation, etc.).

The **exclusion criteria** are: (1) Proposed approaches that are proposing to simply reuse Cloud modeling languages "as is" in order to model Fog systems. (2) Proposed approaches that are purely theoretical or mathematical, i.e., not providing a language nor corresponding support (in the Software Language Engineering sense).

Generally, we acted in a conservative manner. In some cases, more information than the title and abstract is actually required to determine whether a study is relevant for this review. This gave us a reduced set of 49 papers to be investigated further in the final pruning step.

Manual Selection Based on Detailed Content (Step 4). In the final pruning step, we carefully read the remaining papers considering both our objective and the defined inclusion/exclusion criteria. As a result, we selected 12 relevant approaches corresponding to 13 different papers. After this step, we are confident that this initial set of approaches already covers a significant portion of published approaches proposing Fog modeling languages.

3.2.4. Snowballing (Step 5) and Final Result

Following the application of our methodology, we wanted to double-check that we did not miss any potentially relevant approaches. Indeed, some papers could have been missed during the search process for different reasons, or some existing solutions are tools and thus do not necessarily have indexed publications on DBLP.

As a consequence, based on the 13 selected relevant papers from our literature search, we performed a snowballing phase [8] by investigating the references these papers cite (approaches these paper authors were already aware of as domain experts). We have been able to identify a few more approaches after applying the aforementioned inclusion/exclusion criteria to these additional references.

The final result of the overall process (including this last snowballing step) is presented in Table 1. This table provides, in addition to a representative name for each selected approach, the main publication(s) from which the data has been extracted out for this survey. At the end, we considered 23 different papers corresponding to 16 approaches as relevant for this survey.

Table 1: Finally selected approaches.

Approach	**References**
Smada-Fog	[9]
Khebbeb et al.	[10]
Sahli et al.	[11]
FogDirSim	[12][13]
AcOP	[14]
DITAS	[15]
Engelsberger et al.	[16][17]
MobileFog	[18]
iFogSim	[19][20][21][22]
FogNetSim++	[23]
COMPSs	[24] [25]
Extended TOSCA	[26]
CloudPath	[27]
Distributed Node-RED	[28] [29]
YAFS	[30]
Fogify	[31]

3.3. Data Extraction Process

To study the selected approaches, the data extraction process itself has been divided in different big stages concerning the terminology, feature model and evaluation of the selected approaches proposing Fog modeling languages.

The survey results have been principally achieved by having regular face-to-face meetings between the authors. We believe that frequent brainstorming sessions between our three different but complementary profiles and expertise (going from distributed and Cloud systems to software engineering and modeling) already allowed to obtain relevant results. However, as getting external feedback is also important, we presented both the feature model and our intermediate evaluation results during a couple of SeMaFoR ANR project's internal meetings. In these occasions, we received interesting remarks from our academic and industrial partners that ultimately led us to further discussions and corresponding adjustments of our work.

The main general objectives were (1) to agree on an overall common terminology to be used all along the work presented in this paper and (2) to produce a relevant version of the feature model to be presented and used in this

study. During several months, we also worked in parallel on applying the survey methodology described in this section. Then we deeply studied all the selected approaches, resulting from our application of the methodology, in order to individually evaluate them thanks to the characteristics expressed in our feature model. After having performed these evaluations, we discussed altogether several times the obtained results before reaching the current consolidated version we are now confident with.

4. A Feature Model for Characterizing Fog Modeling Languages

Based on our own experiences working on modeling Cloud systems in the past years, and on a deep study of the related state-of-the-art (cf. Section 3 for the used method), we propose in this paper a feature model (see Figure 3) describing what we consider to be the main characteristics of a Fog modeling language. This feature model will be used in Section 5 to better describe and compare the existing solutions we have identified and selected.

We have defined two main categories of features that Fog modeling languages are supposed to cover. The first one concerns the general characteristics of the *language*, i.e., its *scope* and *definition*. These two features and their sub-features directly relate to RQ1 and RQ2 (respectively) as stated in Section 3.1. The second main category of features concerns the corresponding *support* that can come along with a given language. This feature and its sub-features directly relate to RQ3 as stated in Section 3.1.

4.1. Language

4.1.1. Scope

A Fog modeling language can first be characterized by its **Scope** and the following characteristics.

Dimension: A given language can possibly allow representing elements describing the **structure** of a Fog system, its **behavior** or both of them.

Layer: A given language can cover one or several of the layers typically encountered in Fog system's standard representation [4]. These layers are namely **Cloud** (i.e., covering traditional IaaS, PaaS, SaaS elements), **Fog** (i.e., for resources more at the Edge of the system), and **IoT** (i.e., for devices, related protocols, etc.).

ArchitectureType: A given language can target various kinds of Fog systems architectures based on distinct **layers**, on the existence of various Fog **areas** or on several complementary **views** over the modeled Fog system.

Control Type: A given language can be oriented towards two main kinds of control management for Fog systems: **centralized** (such as in more traditional Cloud systems) or **decentralized**.

Resources: A given language can allow representing various kinds of Fog system resources. It can provide language concepts to model **physical resources** (e.g., physical machines, devices, network infrastructures), **software resources** (e.g., virtual resources, application or service resources, but also data, files, protocols) and/or **workflow resources** (e.g., processes, actors, activities).

Properties: A given language can provide language concepts targeting in particular one or several important properties of Fog systems. We have currently identified key ones such as **privacy / security**, **health** (in terms of availability, stability, consistency, etc.), **performance** (related to CPU / memory, latency, bandwidth or other), or **energy**. Any other property can also be considered if it appeared to be relevant.

Genericity: A given language can be either **domain-specific** (i.e., target particular application domains such as Smart City, Smart Home or Smart Health) or **domain-independent** (i.e., completely generic in this sense).

4.1.2. Definition

A Fog modeling language can also be primarily characterized by its **Definition**. According to the principles and best practices of Software Language Engineering [32], a given (modeling) language is normally defined by the three following elements:

Abstract Syntax: A given language is based on a structured set of concepts related together. They can be **standard-based** (e.g., coming from a standard metamodel), or defined in a **custom** way (from scratch).

Concrete Syntax: A given language provides to users one or several ways of expressing corresponding models. The offered interfaces or notations can be **graphical**, **textual** or a combination of both.

Semantics: A given language comes with a particular meaning attached to its concepts and their associated syntactical elements. This can be expressed in a way that is **formal** (i.e., mathematical) or **semi-formal** (e.g., specifications in natural language).

Moreover, such a language can also come with an **Extension mechanism**. A given language can be natively defined in a way that it can be refined and/or extended when needed, e.g., in order to address different kinds of Fog systems or corresponding ranges of problems.

4.2. Support

Finally, a Fog modeling language can be characterized by its associated **Support**.

Implementation: The implementation of a given language can be **proprietary** or the corresponding source code can be made available publicly in **open source**.

Capabilities: The tooling supporting a given language can provide different kinds of capabilities traditionally associated to modeling languages. We can notably consider **editing** capabilities (i.e., for creating or modifying models), **model transformation** or **code generation** features (e.g., targeting various modeling standards or programming languages, respectively), the support for **verification and validation** (V&V), **simulation** or **execution** of the defined models. An important Fog-related

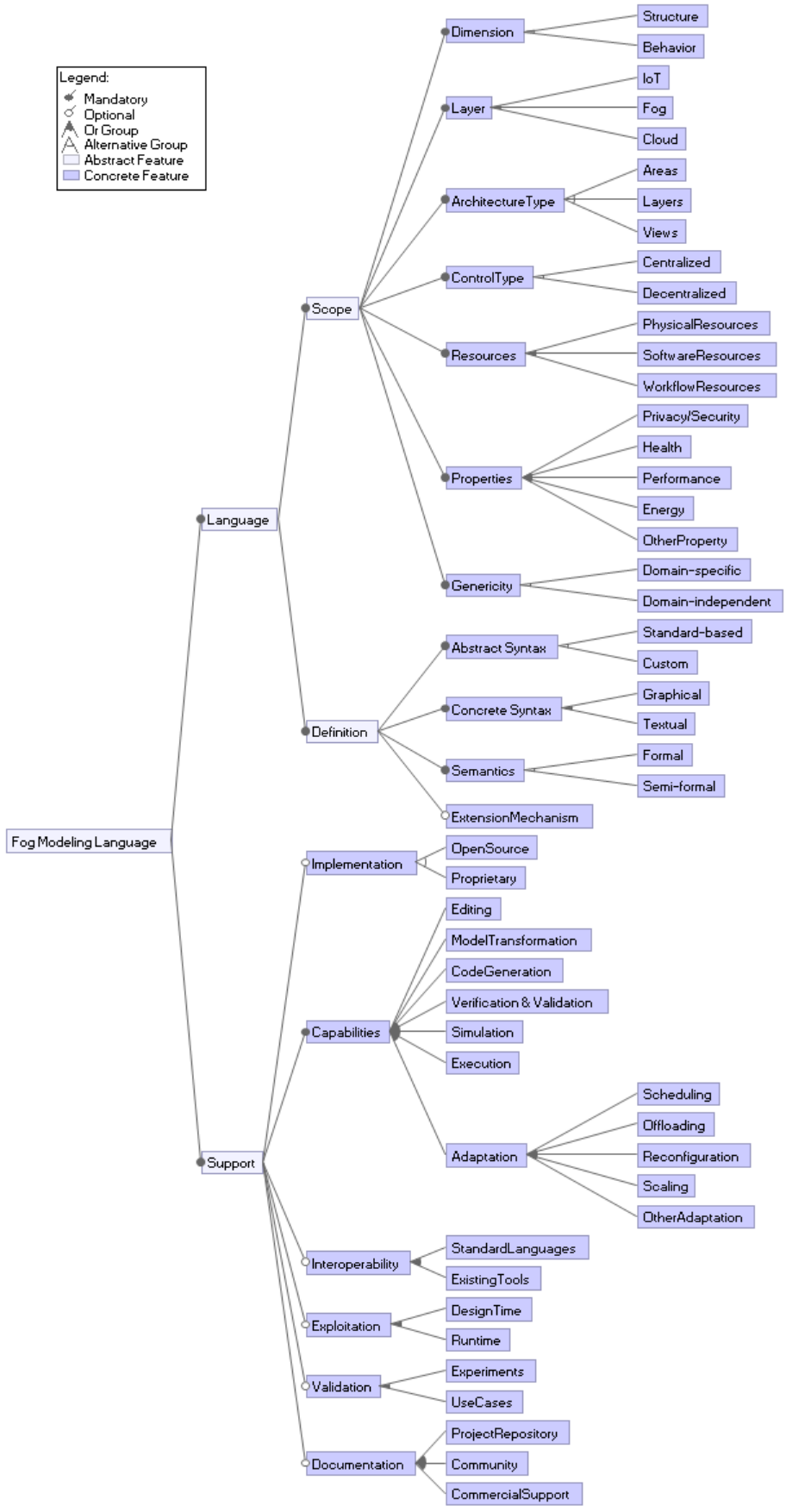

Figure 3: A feature model for characterizing Fog modeling languages.

capability concerns the support for Fog system's **adaptation** in terms of self-reconfiguration, self-optimization, self-healing (fault tolerance), etc. This can be realized via techniques such as **service scheduling**, **scaling** (vertical or horizontal), **network reconfiguration** or **requests offloading** [6] (please also refer to Subsection 2.2 for further explanations).

Interoperability: A given language can possibly be interoperable with either well-known **standard languages** in the domain (notably modeling ones, but also data sharing formats such as XML or JSON) or **existing tools** providing relevant capabilities.

Exploitation: A given language and its tooling support can be mostly exploited at **design time**, i.e., when designing the modeled Fog system, and/or at **runtime**, i.e., when running the modeled Fog system.

Validation: A given language may have been validated/evaluated by various kinds of **experiments** (e.g., coverage or performance benchmarks) or via its concrete application in the context of different **use cases** (which are not necessarily industrial ones but could also be conceptual examples of usage).

Documentation: A given language can come with corresponding documentation support taking several forms. A dedicated **project repository** (e.g., on GitHub) may be provided featuring tutorial, examples, etc. A related user **community** may also have been developed and provide practical support via forums, blogs, etc. Moreover, an ecosystem of companies may sell a related language/tool-specific **commercial support**.

5. Description of Selected Fog Modeling Languages

We now describe each one of the selected approaches by focusing on the main characteristics of the Fog modeling languages they provide (cf. Section 3). To realize this, we notably rely on the previously introduced terminology and feature model. Table 2 summarizes the overall results of our study. The more detailed cross analysis of this table is presented later in Section 6.

Smada-Fog. This solution proposes a semantic model-based approach intended to support Fog systems deployment and autonomic adaptation [9].

To support this approach, two complementary languages are introduced in the paper. The *deployment model* is centered on the notions of devices and tasks (or services). Devices can be consumer devices, network devices or servers while tasks represent the application modules as well as network functions. The *adaptation model* allows the representation of the various adaptation strategies. An adaptation strategy consists of a number of Event-Condition-Action rules. The currently supported reconfiguration actions are services duplication, re-scheduling, network rules configuration and horizontal scaling.

In terms of related support, the authors reported on a Node-RED-based implementation. This implementation is based on a semantic model consisting of an ontology encoded as an RDF Schema. The authors provide a model transformation to generate (part of) this ontology from the users models. The implementation also supports code generation for two target platforms, namely Docker commands and SDN (Software-Defined Networking) rules. In addition, the authors provide a web-based simulator. Part of the described support is available in open source[3].

Khebbeb et al. The main objective of this solution is to provide a formal framework to reason about the correctness of Fog system management policies [10].

The Maude language [33] has been chosen as the basis for the proposed solution. The Fog system's structure is modeled as a Maude configuration and roughly consists in a Fog and a Cloud layer. The adaptation scenarios are modeled via Maude rewriting rules, as a rich and expressive mechanism provided by Maude to express system dynamic behavior. The described adaptation scenarios are mainly related to latency and resource usage optimization. However, no explicit mention is made of mobility, security/privacy, self healing or optimization related to fluctuating network capabilities (for example). Finally, the supported adaptation capabilities cover services/VMs creation and/or deletion, service/VM relocation, VMs scaling and requests load-balancing (offloading).

Regarding the related support, the language is based on the open source Maude execution engine which offers textual editing, simulation and model-checking capabilities. The authors did not provide a code repository nor advanced documentation. However, some source code snippets are provided in appendix of their paper.

Sahli et al. This solution promotes the use of Bigraphs [34] in order to formally model and verify Fog system's adaptative behavior [11].

The authors introduce a formal language based on the Bigraphical Reactive Systems (BRS for short). BRSs are a formal and compositional process algebra that provides both a textual and graphical representation of systems structure and behavior [34]. In particular, the proposed formalism provides the necessary building blocks to describe a Fog system consisting of Cloud and Fog layers. Moreover, reaction rules are used to model the Fog system behaviour as Event-Condition-Action rules. Finally, the formalism also allows the expression of adaptation scenarios related to performance (CPU, memory, energy, latency) and self-healing. The proposed adaptation actions essentially cover service rescheduling or VMs/Containers scaling, workers (VMs/Containers) switching On/Off and network reconfiguration.

The authors did not report on any existing tooling support for their language. They rather described the theoretical process of reasoning on and checking the correctness of online reconfiguration of modeled Fog systems.

[3] `https://github.com/penenadpi/smada-fog`

			Smada-Fog	Khebbeb et al.	Sahli et al.	FogDirSim	AcOP	DITAS	Engelsberger et al.	MobileFog	iFogSim	FogNetSim++	COMPSs	Extended TOSCA	CloudPath	Distributed Node-RED	YAFS	Fogify
Language																		
Scope																		
	Dimension	Structure	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		Behavior	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓		✓	
	Layer	IoT	✓		✓		✓	✓	✓	✓	✓	✓		✓		✓	✓	✓
		Fog	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
		Cloud	✓	✓	✓			✓	✓	✓	✓	∼		✓	✓	✓	✓	✓
	Architecture Type	Areas			✓		✓			✓			∼					
		Layers	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		Views							∼									
	Control Type	Centralized	✓	✓	✓	✓		✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
		Decentralized					∼						∼					
	Resources	Physical Res.	✓	✓	✓	✓	✓	∼	✓	✓	✓	✓				✓	✓	✓
		Software Res.	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	∼	✓
		Workflow Res.	✓	✓	✓	✓	✓	✓		✓	✓	✓				✓		
	Properties	Privacy/Security	∼					∼								∼		
		Health			✓	✓								✓			✓	✓
		Performance	✓	✓	✓	✓	✓		✓	✓	✓	∼	✓	✓	∼	✓	✓	✓
		Energy	✓		✓	✓					✓	✓						
		Other properties						✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
	Genericity	Domain-specific				✓	✓		✓				✓				✓	
		Domain-independent	✓	✓	✓			✓		✓	✓	✓		✓	✓	✓		✓
Definition																		
	Abstract Syntax	Standard-based						✓						✓		✓		
		Custom	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓		✓		✓	✓
	Concrete Syntax	Graphical	✓		✓			✓	✓		✓	✓		✓		✓		
		Textual		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓		✓	✓
	Semantics	Formal		✓	✓	✓			✓									
		Semi-formal	✓				✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
	Extension Mechanism						✓	✓	✓	✓	∼			✓		✓	✓	
Support																		
Implementation	Open Source		✓			✓	✓				✓	✓	✓	✓		✓	✓	✓
	Proprietary																	
Capabilities	Editing		✓	✓							✓	✓	✓	✓		✓		
	Model Transformation		✓											∼				
	Code Generation		✓															
	V&V			✓	✓													
	Simulation		✓	✓		✓					✓	✓					✓	✓
	Execution					✓	✓			✓			✓		✓	✓		
	Adaptation	Scheduling	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	
		Offloading		✓			✓					✓						
		Reconfiguration	✓		✓										✓		✓	
		Scaling	✓	✓	✓	✓				✓				✓	✓			
		Other adaptation		✓	✓			✓	✓		✓						✓	
Interoperability	Standard Languages		✓			✓					✓			✓	✓		✓	✓
	Existing Tools		✓			✓								✓		✓	✓	✓
Exploitation	Design Time		✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓
	Runtime		✓	✓	✓		✓	✓	✓	✓			∼	✓	✓			
Validation	Experiments		✓			✓	✓	✓		✓	✓	✓	✓	✓	✓		✓	✓
	Use Cases		✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Documentation	Repository		∼			✓					✓	✓	✓	✓		✓	✓	✓
	Community										✓		✓					
	Commercial Support																	

Table 2: A comparison of existing Fog modeling languages (✓=feature supported, ∼=feature partially supported).

FogDirSim. This solution proposes a simulator of the CISCO control panel, called FogDirector, that aims at deploying, monitoring, managing and troubleshooting Fog systems realized on CISCO devices [13, 12].

To this end, the authors propose a modeling language with operational semantics based on (a feature-complete subset of) the FogDirector API [13]. Then, they extend this language with simulation primitives, allowing to model erroneous situations (for example). The language notably includes API-provided features for adding, removing and tagging CISCO devices, or deploying applications (as VMs, Docker or Linux containers, etc.) on these devices. Additionally, configuration files allow to describe the application requirements (in terms of resource usage for instance) as well as devices capabilities and network configuration. The proposed language also covers monitoring and alerting calls to retrieve infrastructure and application data, as well as to compute Key Performance Indicators (KPIs) accordingly. The approach is strongly bound to the CISCO ecosystem and deployment is only supported on Fog nodes. Thus, the language does not support layered Fog systems.

Regarding the related support, the authors provide an open source implementation of the language with a Web-based simulator (and the associated documentation) [4].

AcOP. This solution proposes a new paradigm to write and execute coordinated, proactively and pervasively initiated multi-device applications in Fog environments [14].

The provided modeling language has it roots in coordination languages [35, 36] and in the join actions theory [37]. The central concepts are the notions of actions and collective executions (which can be assimilated to applications). An action is triggered by a sensation (an event) and consists in a guard (preconditions), a role (a set of devices with necessary capabilities and status) and a body (the execution logic of the action). The proposed language is textual, imperative and focuses only on the software-related aspects of Fog systems. Contrary to most of the solutions identified in our study, AcOP does not rely

[4] `https://github.com/di-unipi-socc/FogDirSim`

on micro-services and adaptation is supported through services (tasks) scheduling and offloading.

Regarding the related support, the authors propose a Web-based infrastructure and dedicated IDE (with language editor) partially available under the MIT licence, along with the source code of a simple use case[5].

Ditas. This solution comes from the DITAS H2020 project and calls for the development of Data as a Service (DaaS) platforms in the Cloud/Fog continuum [15].

To model data movement scenarios, the authors propose to extend a goal-based modeling language, the Business Intelligent Model (BIM), with annotations related to data movement actions and their impact on some system goals (e.g., response time or data consistency). BIM is a graphical and hierarchical representation of the goals of a system under study. The authors first identify the main categories of data movement (*edge2cloud, edge2edge, cloud2edge* and *cloud2cloud*), the associated transformations (*aggregation, pseudonymization, encryption*) and corresponding metrics (execution time, cost, etc.). Then, *contribution links* are introduced to map the data movement actions to (sub)goals in the BIM. When a goal violation is detected by the monitoring infrastructure, the annotated BIM together with the *contribution links* are used to determine the data movements that will recover it. Regarding the supported adaptation mechanisms, the formalism is restricted to Data movement actions (moving/duplication) of data between Fog/Cloud nodes.

Finally, the paper does not mention any support related to the (goal-based) modeling language.

Engelsberger et al. This solution proposes a set of models and an execution framework specifically targeting Fog-based Cyber Physical Production Systems (CPPS) [16] [17].

The authors introduce three formal complementary languages to represent three main aspects of a CPPS: (1) The *Production-Step Model* for the production steps, their temporal interdependencies as well as links to their supporting IT services; (2) The *Service-Dependency Model* for the service instances and their mutual dependencies together with requirements on the underlying computing nodes; (3) The *Node-Link Model* for the Fog/Cloud computing nodes, their logical links and respective properties. Defined models can be assimilated to different views on the system. They are formal and can have graphical representations based on finite state machines and undirected graphs. They can be annotated with information that links them together, as well as with the requirements and criteria that need to be optimized. They also have an equivalent textual algebraic representation that allows automatic reasoning. The proposed languages and the accompanying algorithms offer the necessary building blocks to express the CPPS behavior as a multi-objectives optimization problem including in the presence of failures or faulty behaviors. Regarding adaptation mechanisms, the approach supports service embedding (selection of the most appropriate service instance) and service placement or (re)scheduling.

The authors describe a case study to illustrate the feasibility and effectiveness of the proposal. However, no mention is made of any tool support for the languages.

MobileFog. This solution is one of the first attempts to provide a modeling language supporting the development of applications for the Fog computing paradigm [18].

The proposed language is based on a programming language-agnostic API intended to be used directly within the application business code. It allows application developers to model and query information about the application and Fog/Cloud environment (location, topology, available resources, etc.) to realize higher level objectives such as resource optimization (CPU, memory, bandwidth or latency) and response time optimization (proximity). The offered mechanisms mainly cover dynamic scaling and application re-scheduling. It targets tree-based Fog systems where the root node is the Cloud, leafs are (mobile) IoT devices and intermediate nodes are Fog devices.

Although the paper describes a deployment, adaptation and code execution platform, such a platform is not implemented in practice. Still, the language has been evaluated on a network simulator with realistic traffic patterns.

iFogSim. This solution is among the first and most popular simulation tools for Fog/Cloud/IoT applications. It provides a dedicated language to model simulation scenarios thanks to a Java API. A graphical syntax and interface is also available to model Fog system's topologies (and then export the models in JSON format). The *iFogSim* modeling language mainly focuses on resources management related aspects: optimization of CPU, memory, storage, latency, energy consumption, network congestion and operational costs. In terms of attached mechanisms, it is mainly coming with module placement (service scheduling) through the provisioning of placement algorithms. The language does not come with specific extension mechanisms to address additional objectives and/or implement additional adaptation mechanisms. However, it is possible to extend it because of its openness, documentation and modular design. Thus, different extensions have already been proposed to address various concerns. For instance, *MobFogSim* in an extension targeting devices mobility [20] while *MyiFogSim* is another one for VM migration [21]. Naas et al. present an extension to model and evaluate Data placement strategies [22].

iFogSim (and its various extensions) is an open source project published under the Apache Version 2.0 license[6].

FogNetSim++. This solution is a simulator for Fog environments with a special focus on networking aspects [23].

[5] `https://github.com/ProgrammingModel/AcOP`

[6] `https://github.com/Cloudslab/iFogSim`

This simulator notably provides a graphical language to model fine-grained network configurations and comes with a predefined set of supported protocol, mobility, energy, pricing and data transmission models. It also implements predefined management policies. The *Broker Node* that centralizes the management of the system under test is at the heart of the used Fog system's models. *FogNetSim++* comes with a predefined offloading and scheduling algorithms that can be overwritten in order to consider nodes characteristics in term of processing load, energy consumption, cost, distance from request source (for latency optimization), bandwidth, etc.

FogNetSim++ is available under the GPL-3.0 License[7]. The authors reported a detailed use case of a Fog-based traffic management system, along with extensive scalability experiments and network performance benchmarks.

COMPSs. This solution provides a framework for the development and execution of parallel applications that run on distributed infrastructures [24] [25].

The proposed modeling language has its roots in the *distributed computing programming models family* which aims at discharging application developers from distribution concerns. It extends the functional Java, C++ or Python code with annotations to model candidate methods and services to parallelization, as well as the information to build their dependency graph and (optionally) their required hardware and software resources. COMPSs is exclusively focused on the capability of scheduling services and methods to distribute computing nodes in the Fog/Cloud. It does not seem to offer much primitives for model placement preferences, except through required resources. Interestingly, the paper describes a partially decentralized control mechanism coming with the language. COMPSs is available under the Apache 2 licence[8].

Extended TOSCA. This solution proposes to provide support for optimizing TOSCA deployment templates in a Fog/Multi Cloud environment [26].

The authors propose an extension of the TOSCA modeling language [38] focusing on deployment and runtime optimization. This extension relies on a two-level TOSCA specification process. The "type-level" model is partial and produced at design-time to reflect Fog system requirements, while the "instance-level" runtime model is optimized, complete and thus actually deployable. Moreover, two kinds of nodes are explicitly distinguished. *Processing nodes* represent the Cloud VMs or Fog PMs that will host the application tasks or services (called Fragments). They are modeled as a set of constraints on CPU cores, available memory, storage, operating system and available sensing capabilities (e.g., a camera or a microphone on the hosting node). *Fragment nodes* represent containerized tasks and their respective characteristics (number of replicas, Docker image and repository, etc.), as well as a link to the Processing node candidate for hosting the task. The proposed language focuses on the self optimization capability of TOSCA deployment regarding cost, distance form the centroid of edge devices (reflecting latency) and preferences towards Cloud providers. The supported adaptation capabilities are essentially service/tasks (re)scheduling and (horizontal and vertical) scaling through the generation of new instance type specifications whenever a change is detected in runtime condition.

Regarding the related support, the authors describe a partially tool-supported process for generating optimized "instance-level" TOSCA deployment models from "type-level" models. The proposed language and related tooling are available under the Apache-2.0 License[9].

CloudPath. This solution is a framework following the Path Computing paradigm, i.e, a multi-layer architecture supporting processing and storage on a set data centers span over a Cloud/Fog infrastructure [27].

The central notion of the modeling language associated to this framework is the one of function. The language syntax is associated to applications written as Java servlets. It uses the *web.xml* file to describe, as a tree-based Fog system's model, how and where each function will be placed whether explicitly (Edge, Fog or Cloud) or implicitly by a constraint on the tolerated latency of the function. Concerning supported Fog capabilities, the paper describes a deployment infrastructure that schedules functions according to developer preferences and to previously fixed internal parameters. The authors also mention network reconfiguration, rescheduling or the horizontal scaling of functions at runtime. However, the proposed language does not provide any mean to model preferences or to customize such mechanisms.

Regarding the related support, the authors did not mention any source code repository or documentation.

Distributed Node-RED. This solution is a realization of the *Distributed Data Flow* paradigm that supports the development of IoT applications to be run over a Fog/Cloud architecture [28, 29].

The proposed modeling language is an extension of *Data Flow*, a coordination language for developing WSNs (Wireless Sensor Networks) and IoT applications [39]. A *Data Flow* (DF) model is a directed graph of nodes where each node represents an independent processing unit consuming inputs and producing outputs. The proposed language extends DF by integrating various Fog Computing-specific elements related to heterogeneous nodes' capabilities, location, etc. In the current version of the solution, this language and resulting models are mostly intended to be exploited at design time.

[7] `https://github.com/rtqayyum/fognetsimpp`
[8] `https://github.com/bsc-wdc/compss`
[9] `https://gitlab.com/prestocloud-project/application-fragmentation-deployment-recommender`

Regarding the related support, the implementation of Distributed Node-RED (D-NR) provides a deployment platform based on Node-RED as well as an editing graphical interface allowing application developers to model Dataflow nodes and corresponding placement constraints. For deployment purposes, D-NR can be connected to Web of Things Toolkit (WoTKit)[40]. The project source code is available on GitHub under the Apache-2.0 License[10].

YAFS. This solution (*Yet Another Fog Simulator*) is a discrete event simulator for Fog systems [30].

YAFS provides a JSON-based modeling language that insures compatibility with third party tools and/or can be used by non-expert users quite easily. For advanced scenarios and policies, an extended Python-based API complements the language. The language allows modeling highly flexible scenarios, including user movement by modifying workload sources. Health status modeling is also supported as well as resource optimization regarding performance parameters: CPU, memory, bandwidth and link propagation. Additionally, users can model their own parameters and objective functions. The offered mechanisms are service scheduling and network configuration.

The YAFS source code is available in open source under the MIT licence[11].

Fogify. This solution provides a Fog Computing system emulator offering modeling, deployment, measurement and evaluation facilities [31].

The Fogify modeling language is Python-based, RESTful and extends the Docker Compose language. It allows to model the physical/virtual infrastructure of a Fog system (i.e., Fog nodes), the (micro-)services that will be deployed on top of it, and the supporting networks. Even though "what-if" scenarios such as devices mobility or failure can be modeled by using the proposed language, no primitives are provided to model the dynamic behavior of the Fog system in order to adapt to changing execution conditions. Moreover, the underlying infrastructure does not provide any support for runtime adaptation. Instead, the adaptation logic must be manually implemented and handled at the application's source code level.

Fogify is open source and available under the Apache 2.0 licence[12]. The repository also includes some documentation and illustrative examples.

6. Discussion

6.1. Analysis of the Collected Results

Interesting findings can be made by analyzing the aggregated Table 2 resulting from our evaluation of different Fog modeling languages and associated approaches (as described in Section 5). In the context of our survey, these findings notably allow proposing answers to the three research questions stated in Section 3.1.

6.1.1. General Observations

Overall, we can observe that almost all the features appears to be supported (even if partially) by at least one solution. This highlights an already quite important coverage offered by existing languages and the support coming with them. There are only two notable exceptions: *Documentation - Commercial Support* and *Implementation - Proprietary*.

Even if the overall coverage is globally significant, individually most of the identified solutions tend to focus on a given set of features (according to their specific objectives). For example, on average, a selected language addresses 47.50% of the identified core properties for Fog systems (i.e., privacy/security, health, performance, energy or others). Similarly, still on average, a selected language provides support for 42.50% of the identified adaptation techniques (i.e., scheduling, offloading, reconfiguration, scaling or others). In both cases, the covered features are often very different from one solution to another. This gives an impression of relative heterogeneity.

6.1.2. Language Characteristics

As an answer to RQ1 (cf. Section 3.1), we can observe immediate similarities related to the scope of the selected languages. For example, most of the approaches allow a certain level of modeling at the structural level (for 93.75% of them) and behavioral level (for 81.25% of them) of Fog systems. Moreover, they generally allow to model the IoT, Edge and Cloud usual constituent layers of Fog Systems (for 75%, 93.75% and 81.25% of them respectively).

Quite surprisingly, only one language (6.25% of the selected languages) provides some kind of support for the *Architecture - Views* feature. Moreover, 87.50% of the selected languages assume a centralized control type for the modeled Fog systems while only 12.50% allows (partially) for a decentralized control type. Related to the possibly modeled resources, there is a general consensus to systematically address software resources (whatever their type) as well as also physical and workflow resources to a lower extent (for 81.25% and 62.50% of the selected languages respectively).

However, when it comes to the modeled properties, there are much more differences between the proposed languages. Indeed, while performance appears to be a key property to be modeled (for 93.75% of the selected languages), the coverage for the other properties is rather heterogeneous and somehow limited. For instance, important expected characteristics of Fog systems such as the health status or energy-related issues are only addressed by one third (31.25%) of the identified languages. Even more surprisingly, only three languages explicitly offer some kind of partial support for modeling privacy / security aspects of Fog systems. Such aspects being fundamental for the

[10] `https://github.com/namgk/dnr-editor`
[11] `https://yafs.readthedocs.io/en/latest/`
[12] `https://ucy-linc-lab.github.io/fogify/`

current and future development of Fog systems in the industry, and even more in our society (e.g., health, cities), there is a clear gap to be filled in this domain when it comes to modeling languages.

As an answer to RQ2 (cf. Section 3.1), i.e., concerning the language definitions themselves, the available solutions are also quite heterogeneous. 81.25% of the identified languages are based on custom abstract syntaxes (i.e., few of them are relying on existing standards) and 81.25% of them propose textual concrete syntaxes. However, half of the identified languages (50% exactly) also propose alternative graphical syntaxes, e.g., to allow for an easier information visualization (read-only mode). Interestingly, only 25% of the identified languages have formalized semantics while 75% of them come with semi-formal semantics. Moreover, only half of the identified languages (exactly 50%) come with an explicit extension mechanism that would allow to customize them in the context of particular ranges of problems or for specific application domains. We consider this as another clear gap to be filled for allowing a wider use and dissemination of a Fog modeling language.

6.1.3. Related Language Support

As an answer to RQ3 (cf. Section 3.1), the situation in terms of the support associated to the identified languages is globally very heterogeneous. A notable exception is the common use of the modeling languages at design time (for 87.50% of them) which is rather natural, though we could have also expected a more widespread use of such models at runtime as well (currently for 62.50% of them only). Another exception is the significant validation of these languages via research experiments (for 68.75% of them) and even more via concrete use cases (for 93.75% of them), as a commonly accepted practice in the language engineering community. It appears that providing support for scheduling algorithms to be used for model/system (re)configuration is also rather common (in 87.50% of the selected languages).

For the other features, the actual support seems to be rather limited and sparse. 62.50% of the identified languages are available as open source while the remaining 37.50% are not made publicly available (or we were not able to find the information from the corresponding papers). In terms of modeling capabilities, less than half of them (43.75% exactly) provide an extended editing and simulation support. While this level of simulation support was predictable, as simulation is a relatively common practice in the distributed systems community, not having a more generalized dedicated support for editing is rather surprising. To a lower extent, model execution capabilities are also quite frequently encountered with the identified languages (for 37.50% of them). However, few languages natively come with transformation (12.50%), verification & validation (12.50%) or even code generation (6.25%) capabilities. This is surprising since transformation and code generation are normally among the key features bringing a significant added value or return on investment (ROI) to modeling approaches. This shows that there are still significant progresses to be made in these directions in order to provide an extended language support in the context of Fog modeling languages.

Concerning the adaptation capabilities, apart from the previously mentioned case of scheduling, the support for different kinds of mechanisms or algorithms is rather limited at the time of writing. Indeed, this ranges from 43.75% for scaling features down to 25% for reconfiguration or even 18.75% for offloading. This is also surprising as these are expected to be important capabilities related to Fog systems. Again, it appears that there is still an important gap to be filled in this area. In terms of language interoperability, the current support is also limited: only a subset of the identified languages can actually interoperate with common standards (43.75%) or tools (37.50%) in the domain. In this area as well, there is a large room for improvements as interoperability is usually crucial for facilitating the adoption of modeling approaches.

Finally, the documentation and related practical support associated to the identified languages can be considered as globally insufficient. While 56.25% of the solutions do provide access to a dedicated repository where users can find documents, manuals, tutorials or other resources, only a couple of them (i.e., 12.50% of the total) already offer an actual user community, active forums, blogs, etc. More effort should be made in the future in terms of user support in order to foster the use and dissemination of such Fog modeling languages.

6.2. Identified Open Challenges

As a follow-up of the survey results analysis, we now highlight some research challenges we believe to be worth investigating in the coming years. Some of them are directly connected to well-known problems in the Fog Computing area. Some others are more specifically related to our modeling language context and have been identified by studying deeper the different selected solutions (cf. Section 5) and our comparison (cf. Table 2).

Multi-domain Context. One notable information appearing in our descriptions of the selected solutions is that current Fog modeling languages are heavily influenced by different existing sub-domains of Fog Computing. Indeed, the proposed languages borrow various characteristics and features coming from modeling approaches in the Cloud, DevOps, IoT and Autonomic Computing (AC) domains notably. These domains already come with their own modeling practices, constraints or standards (eventually) and may overlap in other contexts (e.g., Cloud and Autonomic Computing). For instance, Infrastructure As Code [41] or textual-based declarative description of requirements are often preferred by engineers with a Cloud and DevOps background while graphical "component and connector" based ADLs are generally adopted by engineers with an IoT background. Moreover, while TOSCA [38] is generally perceived as a current reference standard in

Cloud Computing, Node-RED and its underlying model are more widespread in the IoT community [42]. Similarly, Autonomic Computing comes with its own standards centered around the MAPE-K control loop [43]. As a consequence, at the Fog Computing level, there is still an important homogenization and integration effort to be made between these different domains, both conceptually and technically (e.g., by relying on federation techniques such as model views [44]). This paper and survey can be considered as part of a first step towards filling this gap.

Separation of Concerns. As seen in the previous challenge, Fog Computing is at the crossroad of different domains with high complexity and heterogeneity. Unlike in Cloud Computing, different stakeholders may come from various domains with different competencies, responsibilities or requirements. For instance, this is especially true in the IoT domain [28] or in Production Systems as requiring different views over a same system [17]. Thus, we envision modeling solutions such as Multi-Paradigm Modeling (MPM) [45] to be particularly relevant in this multi-domain Fog Computing context. Moreover, as in Cloud Computing, Fog system engineers can have different visions or needs in terms of supporting infrastructure, services, applications, networks, etc. For instance, one engineer may work only at the application level without wanting to refer to underlying virtual and physical infrastructures. While the large majority of the reviewed languages (except for Engelberger et al. [16]) do not provide a Multi-View Modeling (MVM) [46] support for multi-dimensional Fog systems, we argue that this is a very important feature to be taken into account in the future. This will notably requires collaboration and synchronization mechanisms allowing Fog system engineers to define their own viewpoints and customize their views accordingly [47] so that they can be better assisted in they daily work.

Multiple Representations and Abstractions. Related to the previous challenge, another feature that emerges from our study is the need for multiple representations and abstractions to model the same Fog system's concepts. For instance, to model the CPU power of a given node in a Fog system, different engineers might need to characterize them in terms of number of cores (for instance in SmadaFog) or number of micro-instructions per second (like is the case with iFogSim). This is also true for localization modeling, where some approaches use GPS to characterize mobile devices (MobileFog) while others simply use the position of the device in the network topology (CloudPath). Another example is the fact that different technological concepts may refer to the same abstractions in a Fog system, such as Containers, VMs or stateless functions. Thus, we believe that an ideal Fog modeling language should allow different representations or abstractions of a same concept to co-exist in the same model, but should also provide means to relate these representations and abstractions (at least to a certain extent). To this end, latest advances in the recently coined area of Blended Modeling [48, 49], as advocating for a better support for multiple notations over single models, could be studied in the future.

Extensibility and Refinement. Another important feature extracted from our study is the need for extensibility and refinement capabilities as far as Fog modeling languages are concerned: we have observed that less than half of the selected languages do provide such a support (at least partially), thus there is still room for significant improvements in this respect. Indeed, multiple areas and related concepts need to be modeled in the context of Fog computing. As a relatively novel paradigm under development, new concepts are also constantly emerging. For instance, data-oriented approaches (e.g., DITAS [15]) bring new needs in term of modeling the data related information such as locality, transformation cost or consistency. Moreover, some concepts to be modeled in Fog systems can be specific to particular application domains and their specific requirements. As an example, this can be clearly noticed in some Fog-based production systems [16]. In order to be able to better adapt to this heterogeneity, we believe that Fog modeling languages should provide common concepts that can then be extended and/or refined with a minimum effort by the Fog engineers. To this end, the reuse of modeling solutions available in the area of metamodel extension [50, 51] or more generally language extension [52] could be envisioned.

Security and Privacy. One of the most noticeable and probably surprising findings from our survey is that the modeling of security and privacy aspects is barely addressed by the studied Fog modeling languages. Such features are of primary importance in a Fog Computing context, even more than in Cloud Computing because of the highly distributed and heterogeneous nature of Fog nodes (among other reasons). However, none of the studied languages appears to provide dedicated elements to properly model Fog system's capabilities and/or requirements related to security and privacy (though other types of language elements have been derived to this end in some cases). What we have observed in our study is that these features are eventually supported at the technical level using conventional mechanisms such as packets filtering or authentication, or are only implicitly represented via constraints related to services, VMs or data placement. This partial support is usually based on very general assumptions about security/privacy in a Fog context, e.g., Edge nodes are more respectful of privacy or Cloud is more secure. Thus, there are still limitations when it comes to precisely model these aspects. We believe that these limitations are also somehow inherent to the nature of security and privacy. Indeed, contrary to other more easily quantifiable properties, security and privacy are complicated to explicitly express and measure. Since security and privacy can depend on various parameters and on multiple dimensions of the Fog systems, one open question is notably how to derive actual metrics or heuristics from the modeled information.

Behavior Modeling. By nature, Fog Computing calls for modeling capabilities covering its dynamic aspects: Fog

systems are extensively open and are composed of highly volatile resources forming continuously evolving environments. We have observed that a significant number of the selected languages allows for a certain level of modeling concerning the behavioral aspects of Fog systems. However, when it comes to dynamic adaptation, we have seen that the situation is much more heterogeneous. Despite the abundant literature on this subject, there is still a lack of standards, widely adopted common strategies, algorithms or mechanisms. As a result, a majority of the reviewed languages are provided along with one or more adaptation capabilities. For example, while some languages propose the use of Event-Condition-Action rules (SMADA-Fog, Sahli et al., Khebab et al., MobileFog), others prefer to model applications requirements and/or constraints and to delegate the adaptation process to external optimizers or solvers (SMADA-Fog, Engelsberger et al., CloudPath, Extended TOSCA, Ditas). As an alternative, some languages and their supporting frameworks (FogDirSim, AcOP, iFogSim, FogNetSim++ and YAFS) provide a low-level API to imperatively describe custom adaptation policies based on elementary monitoring events and reconfiguration actions. In addition to these complementary approaches, we strongly advocate for a better description and support of these dynamic adaptation aspects directly at the modeling level. Behavioral modeling has also been a long-term concern in Software Engineering as well [53]. Thus, we believe that Fog modeling languages could benefit from the research effort in related areas such as Models@Runtime [54] for example.

Decentralized Control. As stated before, Fog systems are intended to be composed of very heterogeneous resources going from public clouds, private clouds, autonomous data centers, varied network equipment to multiple user devices. These systems are also massively distributed and are meant to be subject to important constraints on latency and privacy. As a consequence, within these systems, it can be reasonably expected that different self-managed clusters collaborate in the best and the most efficient manner. Thus, from a modeling perspective, it is important to provide the right abstractions to represent and then enable such capabilities. However, what we have observed in the results of our survey is that, contrary to centralized control architectures, modeling concepts targeting decentralized control are almost nonexistent at the time of writing. We believe this raises a double challenge: 1) Relevant abstractions need to be identified and 2) Adequate representations must be proposed accordingly. For instance, this notion of cluster or more generally of Fog Area should be explicitly represented in Fog modeling languages. The same is also true for suitable functions that aim at "building" Fog Areas by aggregating more concrete modeling concepts such as physical or virtual machines. In addition, we advocate for the identification and support of inter-areas collaboration mechanisms directly at the modeling language level (e.g., consensus, negotiation).

Tooling Support. As previously mentioned in Section 6.1, all the studied languages do not come with supporting tools and, even if they do, the capabilities they provide are still relatively limited at the time of writing. For example, while simulation is supported by less than half of the selected solutions, Verification and Validation (V&V) capabilities appear to be under represented. Specifically, formal verification is supported only by two approaches (Khebab et al. and Sahli et al.) while no language provides any kind of support for model testing. We have also noted that an actual Integrated Development Environment (IDE) covering multiples phases of the Fog system's life cycle is only offered by one solution (namely COMPSs). Moreover, no solution explicitly describes the exploitation of Fog system models in an CI/CD pipeline, despite their large popularity in the Cloud community. As a consequence, going further than simple Fog modeling languages, we believe that a substantial effort should be dedicated to the design and development of Fog IDEs incorporating popular editing and automated deployment facilities but also monitoring and V&V facilities, CI/CD support, etc. In addition, an important objective could be to consider a more systematic reuse of the designed Fog system models both for simulation and deployment purposes (as it has been demonstrated in SMADA-Fog, FogDirSim and Fogify for instance).

7. Related Work

7.1. Fog Computing

With the emergence of Fog Computing during the past years, several works have been published with the objective to specify the main principles, concepts and possible architecture(s) of Fog Computing [55, 56, 4, 57, 58]. This effort also produces a significant number of survey and reviews on various Fog Computing related aspects (as this can be observed in Section 7.3).

In parallel to these numerous attempts to come up with a common general taxonomy for Fog Computing, there has also been a few initiative to provide a more formal definition of Fog systems in the general case [59, 60, 61]. In addition to these initial conceptual efforts, some technical solutions have been proposed to deal with the simulation of Fog/IoT environments [62]. Notably, this includes tools whose modeling languages have been selected and analyzed in the present survey. Some more specific solutions, relying on existing domain-specific standards or protocols, also exist in particular application domains such as telecommunications [63] (for example).

In this paper, we clearly differ from all these works as we are not aiming at identifying Fog Computing common concepts nor providing any (reference) Fog architecture. Our goal is rather to study modeling languages and their capabilities, in the Software Engineering sense, as currently available within the Fog Computing community.

7.2. Modeling Languages

Modeling languages or Architecture Description Languages (ADLs), have been widely used in several engineering disciplines since a long time. This is notably the case in System and Software Engineering [64] where they have already proven to be practically needed by the industry [65]. In the last couple of decades, the development of and support for such languages have been notably supported by the Model Driven Engineering (MDE) / Model Driven Development (MDD) community [66].

Quite recently, based on these good practices and following the advent of Cloud Computing, different languages have been designed and developed in order to model Cloud systems [67]. Among others, a standard called TOSCA [38] for modeling portable Cloud applications and supporting their life-cycle management is getting more attention. For example, TOSCA can be possibly used for specifying the basic constructs of IoT systems [68]. From our side, we already worked in the past on a model-based approach for heterogeneous Cloud systems [69], notably by proposing a corresponding Cloud modeling language that can directly interoperate with TOSCA [70].

On the IoT side, a significant research effort has also been made when it comes to related languages. For example, ThingML [71] is a well-know DSL targeting the modeling of IoT components and their interactions or communications. There have been other initiatives of the same kind, extending UML for instance [72]. Moreover, for particular types of IoT systems, some efforts have been devoted to the definition of an IoT Reference Model [73]. The use of (visual) programming languages have also been studied in an IoT context [74], but we have not found any general survey on modeling languages for IoT.

Overall, up to our current knowledge, we are not aware of any work focusing on studying the current state-of-the-art in terms existing modeling languages particularly intended to the Fog domain. We intended to fill this gap with the survey presented in this paper.

7.3. Secondary Literature

As already mentioned in Section 7.1, the definition effort around Fog Computing during the past years resulted in a significant number of general surveys or (systematic) literature reviews in the area.

Most of these works were focusing on identifying the main Fog Computing trends [75], involved concepts [76, 77], supported architectures [78, 75, 79] or corresponding research challenges [78, 80, 75, 77]. In addition to these works, some initiatives were targeting more the direct relation between Fog Computing and other underlying paradigms such as the IoT [1] or the Edge [3].

Moreover, we have found several surveys studying the state-of-the-art of Fog Computing in specific application domains such as Smart Cities [81] or Health [82] (among others). We have also identified reviews targeting the state-of-the-art concerning particularly challenging aspects of Fog Computing such as security [83] or dependability [84] (for example).

Nevertheless, we have not found any work focusing on the study of existing approaches providing actual Fog modeling languages (and related capabilities) independently from any specific application domain or concern. Complementary to all the works mentioned in this section, our goal is to provide to the Fog Computing community a suitable overview similarly to what is already available for Cloud modeling languages [67].

8. Conclusion

In this paper, we presented a collection of research approaches dealing with the problem of providing Fog modeling languages and related support. Although the domain under study is still relatively young and not mature yet, we do believe this research line is becoming more and more crucial for the Fog Computing community, and also from a general distributed systems perspective. This is especially true given the current momentum around Fog Computing and the development of more and more complex Fog systems that need to be properly modeled, in areas such as Smart Cities, Smart Home, Smart Health, etc.

Via the list of identified solutions, we have showed that modeling Fog systems is and will be important in order to support more efficiently their design, development, deployment and running/monitoring. We have described what are these current solutions and their corresponding modeling languages, as well as how they compare to each other. To this end, we have classified them according to a feature model that aims to shed more light on this topic and to be possibly used as a reference for future research in related areas. In total, we have reviewed more than 420 academic papers from which we have identified nearly 50 candidate solutions to finally select 16 Fog modeling languages (after snowballing) representing a current spectrum of available initiatives in this domain.

As future work from our side, we plan to directly tackle some of the challenges we identified in this paper. To this end, we expect to actively collaborate with our industrial partner Alter Way (a Smile group company)[13], notably in the context of the SeMaFoR collaborative research project (funded by ANR, France)[14] which started recently. More generally, we also hope this paper will stimulate further discussions within the Fog Computing community. This could allow both enriching the proposed feature model and extending the approach comparison in the future. But more than anything, this could raise the global awareness on the research challenges associated to Fog system modeling and its practical usages in the industry.

[13] https://www.alterway.fr/
[14] https://semafor.gitlab.io/

Acknowledgments

The authors acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-20-CE25-0017 (SeMaFoR project).

References

[1] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, A. Zanni, A Survey on Fog Computing for the Internet of Things, PMC 52 (2019) 71–99.

[2] C.-H. Hong, B. Varghese, Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms, ACM CSUR 52 (5).

[3] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, J. P. Jue, All One Needs to Know About Fog Computing and Related Edge Computing Paradigms: A Complete Survey, JSA 98 (2019) 289–330.

[4] M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. S. Goren, C. Mahmoudi, Fog Computing Conceptual Model, Tech. rep., NIST (march 2018).

[5] Y. Xia, X. Etchevers, L. Letondeur, T. Coupaye, F. Desprez, Combining Hardware Nodes and Software Components Ordering-based Heuristics for Optimizing the Placement of Distributed IoT Applications in the Fog, in: SAC 2018, 2018, pp. 751–760.

[6] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, A. Ahmed, Edge computing: A survey, FGCS 97 (2019) 219–235.

[7] B. Kitchenham, S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, Tech. Rep. EBSE 2007-001, Keele University and Durham University Joint Report (2007).

[8] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: EASE'14, 2014, pp. 1–10.

[9] N. Petrovic, M. Tosic, Smada-fog: Semantic model driven approach to deployment and adaptivity in fog computing, Simulation Modelling Practice and Theory 101 (2020) 102033.

[10] K. Khebbeb, N. Hameurlain, F. Belala, A maude-based rewriting approach to model and verify cloud/fog self-adaptation and orchestration, JSA 110 (2020) 101821.

[11] H. Sahli, T. Ledoux, É. Rutten, Modeling self-adaptive fog systems using bigraphs, in: SEFM 2019, Springer, 2019, pp. 252–268.

[12] S. Forti, A. Pagiaro, A. Brogi, Simulating fogdirector application management, Simulation Modelling Practice and Theory 101 (2020) 102021.

[13] S. Forti, A. Ibrahim, A. Brogi, Mimicking fogdirector application management, SICS 34 (2) (2019) 151–161.

[14] N. Mäkitalo, T. Aaltonen, M. Raatikainen, A. Ometov, S. Andreev, Y. Koucheryavy, T. Mikkonen, Action-oriented programming model: Collective executions and interactions in the fog, JSS 157 (2019) 110391.

[15] P. Plebani, M. Salnitri, M. Vitali, Fog computing and data as a service: A goal-based modeling approach to enable effective data movements, in: CAiSE 2018, Springer, 2018, pp. 203–219.

[16] M. Engelsberger, T. Greiner, Dynamic management of cloud- and fog-based resources for cyber-physical production systems with a realistic validation architecture and results, in: ICPS 2018, IEEE, 2018, pp. 109–114.

[17] M. Engelsberger, T. Greiner, Dynamic reconfiguration of service-oriented resources in cyber–physical production systems by a process-independent approach with multiple criteria and multiple resource management operations, FGCS 88 (2018) 424–441.

[18] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, B. Koldehofe, Mobile fog: A programming model for large-scale applications on the internet of things, in: SIGCOMM'13 - MCC 2013, 2013, pp. 15–20.

[19] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, R. Buyya, iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments, SPE 47 (9) (2017) 1275–1296.

[20] C. Puliafito, D. M. Gonçalves, M. M. Lopes, L. L. Martins, E. Madeira, E. Mingozzi, O. Rana, L. F. Bittencourt, Mobfogsim: Simulation of mobility and migration for fog computing, Simulation Modelling Practice and Theory 101 (2020) 102062.

[21] M. M. Lopes, W. A. Higashino, M. A. Capretz, L. F. Bittencourt, Myifogsim: A simulator for virtual machine migration in fog computing, in: UCC 2017, 2017, pp. 47–52.

[22] M. I. Naas, J. Boukhobza, P. R. Parvedy, L. Lemarchand, An extension to ifogsim to enable the design of data placement strategies, in: ICFEC 2018, IEEE, 2018, pp. 1–8.

[23] T. Qayyum, A. W. Malik, M. A. K. Khattak, O. Khalid, S. U. Khan, Fognetsim++: A toolkit for modeling and simulation of distributed fog environment, IEEE Access 6 (2018) 63570–63583.

[24] F. Lordan, D. Lezzi, J. Ejarque, R. M. Badia, An architecture for programming distributed applications on fog to cloud systems, in: Euro-Par 2017, Springer, 2017, pp. 325–337.

[25] F. Lordan, E. Tejedor, J. Ejarque, R. Rafanell, J. Alvarez, F. Marozzo, D. Lezzi, R. Sirvent, D. Talia, R. M. Badia, Servicess: An interoperable programming framework for the cloud, Journal of grid computing 12 (1) (2014) 67–91.

[26] A. Tsagkaropoulos, Y. Verginadis, M. Compastié, D. Apostolou, G. Mentzas, Extending tosca for edge and fog deployment support, Electronics 10 (6) (2021) 737.

[27] S. H. Mortazavi, M. Salehe, C. S. Gomes, C. Phillips, E. De Lara, Cloudpath: A multi-tier cloud computing framework, in: ACM/IEEE SEC 2017, 2017, pp. 1–13.

[28] N. K. Giang, M. Blackstock, R. Lea, V. C. Leung, Developing iot applications in the fog: A distributed dataflow approach, in: IoT 2015, IEEE, 2015, pp. 155–162.

[29] M. Blackstock, R. Lea, Toward a distributed data flow platform for the web of things (distributed node-red), in: WoT 2014, 2014, pp. 34–39.

[30] I. Lera, C. Guerrero, C. Juiz, Yafs: A simulator for iot scenarios in fog computing, IEEE Access 7 (2019) 91745–91758.

[31] M. Symeonides, Z. Georgiou, D. Trihinas, G. Pallis, M. D. Dikaiakos, Fogify: A fog computing emulation framework, in: IEEE/ACM SEC 2020, IEEE, 2020, pp. 42–54.

[32] A. Kleppe, Software Language Engineering: Creating Domain-Specific Languages Using Metamodels, Addison-Wesley Professional, 2008.

[33] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C. Talcott, All About Maude-A High-Performance Logical Framework: How to Specify, Program, and Verify Systems in Rewriting Logic, Vol. 4350, Springer, 2007.

[34] R. Milner, The space and motion of communicating agents, Cambridge University Press, 2009.

[35] D. Gelernter, N. Carriero, Coordination languages and their significance, Communications of the ACM 35 (2) (1992) 96.

[36] P. Ciancarini, Coordination models and languages as software integrators, ACM CSUR 28 (2) (1996) 300–302.

[37] R.-J. R. Back, F. Kurki-Suonio, Distributed cooperation with action systems, ACM TOPLAS 10 (4) (1988) 513–554.

[38] OASIS, Topology and Orchestration Specification for Cloud Applications (TOSCA) (Nov. 2013).
URL `http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html`

[39] W. M. Johnston, J. P. Hanna, R. J. Millar, Advances in dataflow programming languages, ACM CSUR 36 (1) (2004) 1–34.

[40] M. Blackstock, R. Lea, IoT Mashups with the WoTKit, in: IoT 2012, IEEE, 2012, pp. 159–166.

[41] K. Morris, Infrastructure as Code: Managing Servers in the Cloud, O'Reilly Media, Inc., 2016.

[42] M. Lekić, G. Gardašević, IoT Sensor Integration to Node-RED Platform, in: INFOTEH 2018, IEEE, 2018, pp. 1–5.

[43] P. Arcaini, E. Riccobene, P. Scandurra, Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptation, in:

IEEE/ACM SEAMS 2015, IEEE, 2015, pp. 13–23.
[44] H. Bruneliere, F. M. de Kerchove, G. Daniel, S. Madani, D. Kolovos, J. Cabot, Scalable Model Views Over Heterogeneous Modeling Technologies and Resources, SoSyM 19 (4) (2020) 827–851.
[45] H. Vangheluwe, J. De Lara, P. J. Mosterman, An Introduction to Multi-Paradigm Modelling and Simulation, in: AIS'2002, 2002, pp. 9–20.
[46] A. Cicchetti, F. Ciccozzi, A. Pierantonio, Multi-View Approaches for Software and System Modelling: A Systematic Literature Review, SoSyM 18 (6) (2019) 3207–3233.
[47] H. Bruneliere, E. Burger, J. Cabot, M. Wimmer, A Feature-based Survey of Model View Approaches, SoSyM 18 (3) (2019) 1931–1952.
[48] F. Ciccozzi, M. Tichy, H. Vangheluwe, D. Weyns, Blended Modelling - What, Why and How, in: MODELS-C 2019, IEEE, 2019, pp. 425–430.
[49] L. Addazi, F. Ciccozzi, Blended Graphical and Textual Modelling for UML Profiles: A Proof-of-Concept Implementation and Experiment, JSS 175 (2021) 110912.
[50] H. Bruneliere, J. Garcia, P. Desfray, D. E. Khelladi, R. Hebig, R. Bendraou, J. Cabot, On Lightweight Metamodel Extension to Support Modeling Tools Agility, in: ECMFA 2015, Springer, 2015, pp. 62–74.
[51] P. Langer, K. Wieland, M. Wimmer, J. Cabot, et al., EMF Profiles: A Lightweight Extension Approach for EMF Models, JoT 11 (1) (2012) 1–29.
[52] M. Völter, E. Visser, Language Extension and Composition with Language Workbenches, in: OOPSLA-C 2010, 2010, pp. 301–304.
[53] R. France, B. Rumpe, Model-Driven Development of Complex Software: A Research Roadmap, in: FOSE'07, IEEE, 2007, pp. 37–54.
[54] G. Blair, N. Bencomo, R. B. France, Models@ run. time, IEEE Computer 42 (10) (2009) 22–27.
[55] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, R. Buyya, Fog computing: Principles, architectures, and applications, in: Internet of Things, Elsevier, 2016, pp. 61–75.
[56] Y. Liu, J. E. Fieldsend, G. Min, A Framework of Fog Computing: Architecture, Challenges, and Optimization, IEEE Access 5 (2017) 25445–25454.
[57] R. Mahmud, R. Kotagiri, R. Buyya, Fog Computing: A Taxonomy, Survey and Future Directions, in: Internet of Everything, Springer, 2018, pp. 103–130.
[58] M. Aazam, S. Zeadally, K. A. Harras, Fog Computing Architecture, Evaluation, and Future Research Directions, IEEE Communications 56 (5) (2018) 46–52.
[59] S. Sarkar, S. Misra, Theoretical modelling of fog computing: a green computing paradigm to support IoT applications, Iet Networks 5 (2) (2016) 23–29.
[60] A. Brogi, S. Forti, QoS-aware deployment of IoT applications through the fog, IoT-J 4 (5) (2017) 1185–1192.
[61] H. Sahli, T. Ledoux, É. Rutten, Modeling Self-Adaptive Fog Systems Using Bigraphs, in: FOCLASA 2019, 2019, pp. 1–16.
[62] A. Markus, A. Kertesz, A survey and taxonomy of simulation environments modelling fog computing, Simulation Modelling Practice and Theory 101 (2020) 102042.
[63] R. Vilalta, V. Lopez, A. Giorgetti, S. Peng, V. Orsini, L. Velasco, R. Serral-Gracia, D. Morris, S. De Fina, F. Cugini, et al., TelcoFog: A unified flexible fog and cloud computing architecture for 5G networks, IEEE Communications 55 (8) (2017) 36–43.
[64] N. Medvidovic, R. N. Taylor, A Classification and Comparison Framework for Software Architecture Description Languages, IEEE TSE 26 (1) (2000) 70–93.
[65] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, A. Tang, What Industry Needs from Architectural Languages: A Survey, IEEE TSE 39 (6) (2012) 869–891.
[66] M. Brambilla, J. Cabot, M. Wimmer, Model-driven Software Engineering in Practice, Synthesis lectures on software engineering 3 (1) (2017) 1–207.
[67] A. Bergmayr, U. Breitenbücher, N. Ferry, A. Rossini, A. Solberg, M. Wimmer, G. Kappel, F. Leymann, A Systematic Review of Cloud Modeling Languages, ACM CSUR 51 (1) (2018) 22.
[68] F. Li, M. Vögler, M. Claeßens, S. Dustdar, Towards Automated IoT Application Deployment by a Cloud-Based Approach, in: SOCA 2013, 2013, pp. 61–68.
[69] Z. Al-Shara, F. Alvares, H. Bruneliere, J. Lejeune, C. Prud'Homme, T. Ledoux, CoMe4ACloud: An end-to-end framework for autonomic Cloud systems, FGCS 86 (2018) 339 – 354.
[70] H. Bruneliere, Z. Al-Shara, F. Alvares, J. Lejeune, T. Ledoux, A Model-based Architecture for Autonomic and Heterogeneous Cloud Systems, in: CLOSER 2018, SciTePress, 2018, pp. 201–212.
[71] B. Morin, N. Harrand, F. Fleurey, Model-based Software Engineering to Tame the IoT Jungle, IEEE Software 34 (1) (2017) 30–36.
[72] T. Eterovic, E. Kaljic, D. Donko, A. Salihbegovic, S. Ribic, An Internet of Things Visual Domain Specific Modeling Language Based on UML, in: ICAT 2015, IEEE, 2015, pp. 1–5.
[73] F. Ciccozzi, I. Crnkovic, D. Di Ruscio, I. Malavolta, P. Pelliccione, R. Spalazzese, Model-driven Engineering for Mission-Critical IoT Systems, IEEE software 34 (1) (2017) 46–53.
[74] P. P. Ray, A Survey on Visual Programming Languages in Internet of Things, Scientific Programming 2017.
[75] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, R. Ranjan, Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions, IEEE Access 6 (2018) 47980–48009.
[76] S. Yi, C. Li, Q. Li, A Survey of Fog Computing: Concepts, Applications and Issues, in: Mobidata 2015, 2015, pp. 37–42.
[77] M. Mukherjee, L. Shu, D. Wang, Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges, IEEE COMST 20 (3) (2018) 1826–1857.
[78] P. Hu, S. Dhelim, H. Ning, T. Qiu, Survey on Fog Computing: Architecture, Key Technologies, Applications and Open Issues, Journal of network and computer applications 98 (2017) 27–42.
[79] P. Habibi, M. Farhoudi, S. Kazemian, S. Khorsandi, A. Leon-Garcia, Fog Computing: A Comprehensive Architectural Survey, IEEE Access 8 (2020) 69105–69133.
[80] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, P. A. Polakos, A comprehensive survey on fog computing: State-of-the-art and research challenges, IEEE COMST 20 (1) (2017) 416–464.
[81] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, A. V. Vasilakos, Fog Computing for Sustainable Smart Cities: A Survey, ACM CSUR 50 (3) (2017) 1–43.
[82] H. J. de Moura Costa, C. A. da Costa, R. da Rosa Righi, R. S. Antunes, Fog Computing in Health: A Systematic Literature Review, Health and Technology 10 (2020) 1025–1044.
[83] P. Zhang, M. Zhou, G. Fortino, Security and Trust Issues in Fog Computing: A Survey, FGCS 88 (2018) 16–27.
[84] Z. Bakhshi, G. Rodriguez-Navas, H. Hansson, Dependable Fog Computing: A Systematic Literature Review, in: SEAA 2019, IEEE, 2019, pp. 395–403.