Autonomous IoT Device Management Systems: Structured Review and Generalized Cognitive Model

Anders Eivind Braten, Frank Alexander Kraemer, *Member, IEEE,* David Palma, *Member, IEEE*

***Abstract*—Research on autonomous management for large-scale deployments of constrained devices is still a maturing field in the Internet of Things (IoT). Although much research has been conducted on how to achieve autonomous management in specific cases, there is a need for literature investigating which mechanisms can achieve such behavior in a generalized way. In this review, we present a comprehensive and structured study of the mechanisms for autonomous device management of constrained IoT devices in the light of management tasks, operational environment, network topology, resource constraints, scalability and management categories. Data extracted from 32 relevant cases is first organized and analyzed according to a synthesized taxonomy of observed adaptation mechanisms, and then combined with state-of-the-art models of autonomous operations, identifying common patterns for autonomous management. Based on our findings we substantiate best practices for designing and implementing solutions around adaptation mechanisms. We then present a generalized model for autonomous device management that describes and explains the processes required for autonomous operation, unifying the insights from previous works as one cohesive archetype.**

***Index Terms*—Autonomous device management, adaptive management, self-management, situation awareness, internet of things, iot architecture, constrained devices, artificial intelligence, cognitive computing, machine learning, adaptation mechanisms.**

I. Introduction and Background

Large-scale deployments of IoT devices are held together by device management platforms. These systems aggregate data collected by the devices and also monitor and control their operation. They are critical since insufficient device management can increase the need for expensive manual interventions and cause downtimes, waste system resources or reduce the reliability and functionality of the system by not appropriately detecting and reacting to problems [1].

Due to the scale, heterogeneity and constraints inherent to IoT systems, the architecture of device management systems is itself critical, and we observe an increasing interest in the use of principles from autonomic computing for device management over the last decade. Traditional management theory defines autonomy as *"the degree to which one may make significant decisions without the consent of others"* [2]. Applied to the IoT, this emphasizes the use of autonomously acting devices. However, the actions of autonomous agents are usually guided by a strategy or objective set by a manager. Autonomous IoT device management hence implies a division of tasks, with a central manager controlling the strategic direction and policies of the system, and the devices or agents acting on their behalf, deciding how to reach their goals [3]. The autonomy of devices can help to overcome challenges with the scale and heterogeneity of the systems. Autonomy can also reduce the need for communication, which can make the system more dependable in cases of intermittent communication. On the other hand, the support by a central management can free resource-constrained devices from complex analysis tasks, provide them with context and guide their operation.

Although an increasing amount of research has been conducted on how to achieve autonomous management of IoT devices, current research on the design and implementation of these systems is chaotic and sporadic, and does not account for the variance in adaptation mechanisms found within this domain. In addition, existing solutions are often highly specialized toward solving one or two particular tasks within a single use case. Conversely, these solutions often discuss architectural challenges in general terms, and the proposed management systems are only partially implemented. The nature of the employed mechanisms that allow autonomous behavior is rarely discussed, and alternative approaches are seldom considered. In fact, we have not found any papers that study the specific mechanisms that are used to achieve this goal in a generalized way, a challenge also identified in [4]. This shows that research on autonomous management for constrained IoT devices is still a maturing field, and that there is a need for a standardized, unified view or methodology that can advance the goal of achieving management systems for IoT that require a minimum of human intervention [5].

This is the first structured review on the topic of *autonomous* IoT device management. Existing literature within device management discusses only niche topics, lacking an overarching perspective. Sinche et al. conducted a survey on IoT management [6], where they identify key requirements for IoT device management, and give an overview of management frameworks and protocols. They stress that management solutions must be able to control IoT devices efficiently with regard to constraints and complexity. They also identify that there is a strong need for a common IoT management architecture. Chowdury et al. surveyed resource management in IoT [7]. One of their contributions is the classification of three different management activities found within this domain, namely resource discovery, resource provisioning and resource scheduling. Further, they state that ensuring automatic management and handling different architectural requirements are among the biggest challenges in distributed computing in general and IoT in particular. However, neither of these studies

A.E. Braten (anders.e.braten@ntnu.no), F.A. Kraemer (kraemer@ntnu.no), and D. Palma (david.palma@ntnu.no), are with the Department of Information Security and Communication Technology, Norwegian University of Science and Technology, NTNU, Norway.

look at the mechanisms that are needed for autonomous device management. Colakovic et al. published a comprehensive review on enabling technologies, challenges, and open research issues within IoT [8]. Although their study is too broad to investigate in detail the challenge of autonomous IoT device management, they discuss aspects of management related to monitoring, control and configuration. They found that due to the inherent complexity of IoT, autonomous adaptation to changes in the environment requires the presence of context-aware management mechanisms.

The lack of a comprehensive review on autonomous management for constrained IoT devices means that there is a need to identify 1) which problems within this domain are addressed by the current state of the art; 2) which mechanisms are most useful for solving these problems; 3) best practices to use when designing and implementing solutions around these mechanisms; and 4) a generalized model that describes and explains the processes needed for autonomous operation. To investigate the issues pertaining to autonomous IoT device management, we chose to study models of architectures presented in previous research in the field of IoT management with the goal of unifying used techniques and lay the foundation for a structured methodology for handling autonomous device management. In particular, our review contributes to the research and development of device management for IoT with the following:

- An overview of different aspects that must be considered when designing IoT device management systems.
- A taxonomy of adaptation mechanisms that are used to allow autonomous IoT device management.
- An overview of different patterns and models that are used to achieve adaptive, autonomous management in the reviewed cases and in general literature on autonomous behavioral systems, respectively.
- The five best identified practices for designing and implementing autonomous IoT device management systems.
- A generalized cognitive model for autonomous IoT device management that unifies the key requirements identified in literature with approaches found in general autonomic computing, with an emphasis on adaptive control loops.

The rest of the paper is structured as follows: In Sect. II we describe the research methodology that we used for the literature review. We continue with an overview of different aspects to consider when designing and implementing IoT device management in Sect. III. In Sect. IV we describe different adaptation mechanisms that are used to solve specific IoT management problems, before we take a closer look at patterns and models used to achieve autonomous management in Sect. V. Afterwards, we present the five best practices for designing solutions for autonomous management for constrained IoT devices that we identified through the review process in Sect. VI. Finally, in Sect. VII we present a generalized cognitive model for adaptive, autonomous management for constrained IoT devices, based on the mechanisms, patterns and models discussed earlier, followed by our conclusions.

TABLE I
SELECTION PROCESS OF PAPERS INCLUDED IN THE STUDY

Step	Included Papers
Relevant papers from former research	123
Total papers after electronic search	2037
Inclusion after removing duplicates	1703
Inclusion based on title	322
Inclusion based on abstract	188
Inclusion based on skimming through text	111
Final inclusion, based on criteria	**32**

II. METHODOLOGY

This review generally followed the guidelines for performing systematic literature reviews in software engineering [9]. This process includes developing a review protocol, identifying and selecting primary studies based on pre-defined inclusion and exclusion criteria, and defining the data extraction and data synthesis activities.

Our search process is shown in Table I. First, we identified relevant papers (123) that were already in our possession through previous research. We then conducted a search in the digital libraries offered by IEEE Xplore, ACM Digital Library, ScienceDirect, SpringerLink and Wiley Online. The search was conducted by putting together phrases using the terms in Table II. We used the same phrases for all 5 libraries. If a search returned more than 100 papers, the result was sorted by relevance and the first 100 papers were included for further assessment. This initial search resulted in 1703 unique papers. The third step was to read titles and exclude papers that were irrelevant for the study. This left us with 322 papers, of which we read abstracts to identify relevant papers. The resulting 188 papers were then browsed for relevance, which yielded 111 papers that we studied in detail using the inclusion and exclusion criteria presented next. The whole process resulted in 32 relevant papers, listed in Table III.

The review includes articles published between 2009 and 2019 on the topic of *Autonomous or adaptive device management for constrained IoT devices*. Articles with any of the following traits were excluded:

- Articles that do not provide a detailed model describing the components involved in device management.
- Articles that do not include an explicit description of adaptation mechanisms.
- Articles where there is no communication between nodes or between a sensor node and a central node.
- Articles with a dominating focus on security, robotics or autonomous vehicles.
- Articles describing systems where human intervention is a part of the device management process.
- Articles shorter than 5 pages or not subject to peer review.

If a topic was published in several journals or conferences by the same authors, we selected the version that contained the most detailed description of the underlying model.

All papers included after browsing through them were subject to a data extraction process. From the 111 relevant papers we first documented the main author, publisher, year

TABLE II
TERMS USED IN SEARCH FOR RELEVANT PUBLICATIONS

Mechanisms				Abstraction level		Awareness level	Subject
autonomous	adaptive	organizing	energy harvesting	architecture	system	self-aware	internet of things / iot
dynamic	machine-learning	optimizing	management	framework	wireless	context aware	constrained device
intelligent	resource(-allocation)	configuring	smart	platform		situation aware	constrained sensor
cognitive	orchestration	healing		cyber physical		environment aware	constrained network

published and the search term that was used for identification. We then carefully read each paper and recorded the main management problem being addressed; if the authors addressed challenges related to *resource constraints*, *scalability and technical heterogeneity*; a classification of the *environment*, the *network topology* and the *management task*; and finally a short description of the *reasoning, learning and planning mechanism* employed to solve the problem. For the 32 papers included in the review, we also classified the architectural models according to their detail level. We synthesized the data into Table III. This process was done incrementally, as some patterns emerged during the extraction phase.

We identified one threat to construction validity. Initially, we planned to conduct a snowball search strategy to identify relevant publications that were not caught in the manual or automated search. However, initial rounds using this strategy failed to show any papers from the selected publishers that were not already identified. Based on this we assumed that a sufficient sample of papers was already available, and therefore elected to cancel this strategy.

We did not identify any threats to internal or external validity. Any causal relationships involved in the study are discussed in an open-ended manner. We do emphasize though that the study can only be generalized within the domain that is defined through the inclusion criteria. For the assessment of conclusion validity, the chosen methodology helps to ensure that the collection procedure is repeatable. Regardless, there is a risk that relevant papers were overlooked when browsing through the titles or abstracts, since the selection is partly based on subjective reading. That said, for both identified threats we surmise that the sample of selected papers is large enough to capture the main patterns within the studied domain.

III. OVERVIEW OF IOT DEVICE MANAGEMENT

Table III summarizes the reviewed use cases [10-41] and lists the different aspects to contemplate when designing IoT device management systems. We observe that autonomous management of constrained IoT devices is a composite problem related to the context in which the devices operate, the device topology, available resources, the scale of the deployment and the problem that the system solves, as stipulated in [32]. In the following, we will introduce and explain these aspects in detail. The synthesized data will then aid our analysis and guide the discussion.

A. Operational Context and Environment Type

All papers included in the final review cope with problems related to management of devices operating in settings where conditions change over time. Local conditions can vary considerably between individual devices within the same network, too. Device management therefore address operation in a context that is dynamic in temporal and spatial dimensions [20]. This means that it becomes complicated to plan proper corrective actions to a previously unseen event, since the same corrective action applied to two different devices can have different outcomes [42]. A high variance in environmental conditions thus implies that it is necessary to *individually* manage each device, to allow for operational adaptation in accordance to the varying conditions that each device experiences. We observed two types of dynamic temporal environments:

- **Stationary:** In a stationary environment, the variance is within a known distribution, that is, the changes in variation can usually be predicted stochastically. An example of device management in a stationary environment can be seen in [14], where Sahni et al. demonstrate energy-aware task allocation.
- **Non-stationary:** Non-stationary environments are characterized by dynamic statistical properties, that is, unstable conditions and distributions that change over time. An example of device management in a non-stationary environment can be seen in [28], where Alam et al. demonstrate adaptive computational offloading.

As we will show in Sect. VI, this difference plays a significant role for the employed adaptation mechanism. We therefore classified the reviewed cases according to the type of environment in which they operate, as shown in column 1 of Table III.

B. System Topology

The system topology describes how the devices are connected. This is a key design decision that influences the organization of the management processes, which are organized in three different ways, as shown in column 2 of Table III.

- **Fully distributed topology (D):** In a distributed topology, each device is responsible for all actions needed to operate and adapt, including storing knowledge and initiating learning processes. Even though a central node is usually present, its only task is to collect data that is sensed by the devices. This means it has no power to manage the operation of the device. In the reviewed papers, we found only one case with a distributed topology [10].
- **Clustered topology (C):** In a clustered topology, two or more parent nodes share the responsibility for storing data and managing processes on behalf of separate subgroups of devices. Each parent node sends instructions to the devices belonging to designated subgroups. The devices

TABLE III
COMPARISON OF ADAPTIVE MANAGEMENT MECHANISMS FOR IOT DEVICE MANAGEMENT

1	2	3	4	5	6	7	8	9	10	11	12
Environment[1]	Topology[2]	Constraints[3]	Scale[3]	Heterogeneity[3]	Management task	Category[4]	Reasoning mechanism	Learning mechanism	Planning mechanism	Detail level of model	Reference
S	D	●	○	○	Adaptive energy management	R	model-driven (linear prog.)	—	predictive controller model	high	[10]
S	C	●	●	○	Energy-aware network management	N, R	model-driven (stoch. geometry)	—	policy-based (obj. engine coord.)	low	[11]
S	C	●	●	●	Adaptive comp. offloading	N, R	semantic (knowledge-graph)	—	dependency-tree (directed graph)	low	[12]
S	C	●	●	●	Context-aware self-management	A, R	semantic (rule-based)	—	policy-based inference	high	[13]
S	C	●	●	●	Energy-aware task allocation	R	model-driven (nonlinear algorithm)	—	model-driven task allocation matrix	low	[14]
S	S	●	●	○	Adaptive config. management	N	semantic (rule-based algorithm)	—	inference-based algorithm	low	[15]
S	S	◐	●	●	Adaptive access control	N	semantic (ontology-based)	semantic (new rules)	event-based inference engine	high	[16]
S	S	◐	○	●	Dynamically change exec. environment	A	semantic (ontology-based context search)	semantic (stored context info.)	query built from context search	low	[17]
S	S	●	◐	●	Context-aware QoS-management	A	model-driven (stoch. model checker)	semantic (update context model	goal-directed MAPE-K control loop	high	[18]
S	S	●	●	○	Autonomous policy determination	A, R	data-driven (ML classifier)	semantic & data-driven (text-mining & SVM)	event-triggered action plan	high	[19]
N	C	●	◐	○	Dynamic energy balancing	R	model- & data-driven (game theory + RL)	data-driven (RL)	reward-based utility function	high	[20]
N	C	●	◐	●	Context-aware self-management	N	semantic (ontology-based)	—	event-based inference engine	high	[21]
N	C	●	●	●	Network lifetime optimization	N, A	semantic & data-driven (CBR + RL)	data-driven (RL)	goal-directed action plan	high	[22]
N	C	●	●	●	Context-aware self-management	N, R	semantic (ontology/context-based)	model-driven (game theor. learning)	decision based on game theory	low	[23]
N	C	●	●	●	Adaptive device orchestration	N, R	semantic (ontology-based)	semantic (learned facts)	goal-directed action plan	high	[24]
N	C	●	●	●	Context-aware self-management	N, R	model-driven (game theory)	model-driven (weighted obs.+ univ. approx.)	maximized utility function	low	[25]
N	C	●	◐	●	Adapt to recognized activity	A	model-driven (fuzzy logic)	data-driven (machine-learning)	goal-directed action plan	high	[26]
N	C	●	◐	○	Resource-aware data collection	A, R	data-driven (RL)	data-driven (RL)	utility look-up table	high	[27]
N	C	●	●	●	Adaptive comp. offloading	R	model-driven (markov dec. proc.)	data-driven (deep Q-learning)	learned policy	low	[28]
N	C	●	○	○	Energy-aware self-management	R	model-driven (modal logics)	—	goal-directed action plan	high	[29]
N	C	●	●	◐	Energy-aware QoS-management	R	semantic & data-driven (dynamic prog. + SVM class.)	data-driven (machine-learning)	QoS- and policy-based service provisioning	low	[30]
N	C	◐	●	●	Context-aware self-management	R	semantic (rule-based inf. engine)	data-driven (machine learning)	event-triggered action plan	high	[31]
N	S	●	●	●	Energy-aware self-management	A, R	semantic (rule-based inf. engine)	data-driven (RFR)	policy-based action plan	high	[32]
N	S	●	●	●	Autonomous network resource discovery	N	model-driven (MAPE-K)	semantic (stored context info.)	policy-based MAPE-K control-loop	high	[33]
N	S	◐	●	●	Adaptive config. management	A, R	semantic (pattern recogn.)	data-driven (machine-learning)	policy-based action plan	high	[34]
N	S	◐	●	●	Adaptive config. management	A, R	data-driven (RL)	data-driven (RL w/back-propg.)	goal-directed control loop)	high	[35]
N	S	●	●	○	Adaptive appl. management	A, R	semantic (semantic modeling)	data-driven (deep-learning+RL)	goal-directed action plan	high	[36]
N	S	●	○	○	Energy-aware task allocation	R	model-driven (smart persistence)	—	semantic task allocation algorithm	high	[37]
N	S	●	○	●	Autonomous service discovery	R	model-driven (prob. reasoning)	model-driven (prob. distr. learning)	event-based service provisioning	high	[38]
N	S	●	○	○	Adaptive energy management	R	data-driven (RL)	data-driven (RL w/back-propg.)	reward-based utility function	high	[39]
N	S	●	●	○	Energy-aware self-management	R	data-driven (ML classifier)	data-driven (RFR+ANN)	policy-based utility function	high	[40]
N	S	◐	●	●	Context-aware self-management	R	semantic (context ontology)	semantic (stored episodes)	goal-based action plan	high	[41]

1) Environment: S... stationary, N... non-stationary
2) Topology: D... distributed, C... clustered, S... star
3) addresses the concern ●... directly, ◐... indirectly, ○... not at all
4) Category: N... network A... application, R... resource

Abbreviations:
machine learning (ML), case-based reasoning (CBR),
reinforcement learning (RL), support vector machine (SVM),
random forrest regressor (RFR), artificial neural network (ANN)

may have some autonomous responsibilities, such as basic reasoning, but learning is usually offloaded to the parent node. Often, a parent node can share knowledge with other parent nodes. Communication between devices within each own subgroup is allowed, but uncommon. We found this topology in 16 cases.

- **Star topology (S):** In a star topology, a single central node stores all data and manages all processes for all devices. The central node sends instructions to the managed devices, which are responsible for receiving and storing instructions, sensing and sending data, and executing actions. There is usually no direct communication between the managed devices. We found 15 cases with a star topology.

C. Resource Constraints

Column 3 of Table III indicates if the cases address resource constraints directly or indirectly. Despite advances in capabilities of IoT devices, they are constrained in terms of available energy, memory and processing capability. In addition, they usually have limited access to contextual information. These constraints make it hard for the devices to solve their own problems, since they lack resources needed to analyze the current situation and predict future events that might influence their operation. To ensure that the devices are able to operate at their optimum and plan corrective and adaptive actions, such tasks are therefore often moved to nodes with better access to resources [43]. The problem of resource constraints are thus often tied to the system topology.

D. Scale and Technical Heterogeneity

Traditionally, device management for wireless IoT nodes has been done manually, where the devices have been configured and updated individually or in bulk, either on-site or over a communication channel. However, this method does not work well in large-scale deployments characterized by many devices and high heterogeneity, which means that the devices and the networks connecting them vary in form, function and functionality [44]. Maintenance throughout the full device lifecycle (planning, configuring, deploying, operating, repairing, and recycling) is a major challenge. Any architecture or framework that supports large-scale device management must therefore be able to operate autonomously with a minimum of human intervention [8]. We indicate to which degree the papers address challenges related to resource constraints, scale and heterogeneity in columns 3 to 5 of Table III.

E. Management Tasks

All the systems in the reviewed cases are directed toward management of constrained IoT devices. Within this domain we find a broad spectrum of operations. Usually, the main problem that is addressed in an article maps to a specific management task. We describe the main management task performed by each reviewed system in column 6 of Table III. Each of these specific tasks can be further mapped to three distinct categories. We chose to categorize device management in the reviewed cases as network, application or resource management, or a combination of these. This categorization is in contrast to Gurgen et al. [45], who divide management of networked sensing devices in network-, system- and application management. This is due to the fact that we did not find any cases that focused on system management, while many cases went beyond application management and focused on how to manage the resources that are available for the devices directly. The category that the main management task belongs to is shown in column 7 of Table III.

- **Network management** is concerned with how to initiate, monitor and maintain the infrastructure of a wireless sensor network, to ensure that the devices are connected and able to send the collected data. Some typical management tasks include discovery, that is, registering devices when connected to a wireless sensor network for the first time [33], ensuring they maintain a stable connection [46], and reconnecting them when they drop out of the network or move between base stations [27]. In many networks the connections between the devices, or even the topology of the network itself, change over time. Such networks can be regarded as a dynamic environment with non-stationary properties. Research related to this problem area is often referred to as *Cognitive IoT* (CIoT). The main idea of CIoT is that interconnected devices are able to analyze their context, learn from experience and develop hypotheses based on their knowledge base with a minimum of human intervention [36]. Typical management tasks within CIoT are aimed at maximizing network performance by analyzing current network conditions, and then deciding and executing adaptive actions [47].
- **Application management** runs processes to configure, monitor and maintain the applications that run on the devices. These are typically extrovert processes, that is, they focus on the purpose of the system that is managed. The main input for application management is the sensor data acquired by the IoT devices. Typical management tasks are off-loading, distributed computing and data collection, in addition to high-level tasks like processing and analyzing environmental data sensed by the devices. Two examples of application management processes are activity adaptation [26] and action recommendation [19].
- **Resource management,** in contrast to application management is an introvert process, as it typically looks at internal processes related to the maintenance and optimization of the operation itself. The main input for resource management is operational data about the devices and their current status. The focus is typically on low-level tasks like ensuring an acceptable quality of service, managing the energy consumption and scheduling sensing cycles. In a dynamic environment, resource management needs to be context-aware, to accommodate for variations in the environment [48]. Examples of resource management are resource-aware data collection [27] and energy balancing [20].

IV. Adaptation Mechanisms

Autonomy is a complex behavior characterized by the capacity an agent has to achieve a goal while adapting to changes in the environment without human intervention [49]. Further, Sifakis et al. [49] list five complementary aspects required to achieve full autonomy: 1) *perception*, or interpretation of stimuli from the environment; 2) *reflection*, that is, building a model of the environmental context; 3) *goal management*, i.e., choosing the best among possible goals given the environmental model; 4) *planning*, or deciding which actions to take to achieve the chosen goal, and 5) *self-adaptation*, i.e., to adjust the autonomous behavior through learning and reasoning.

Self-adaptation and self-management are core concepts in autonomous systems [50]. According to Kephart et al., the goal of self-management is to free system administrators from the tasks of system operation and maintenance and provide systems with the ability to configure, optimize, heal and protect themselves [51]. However, for systems that operate under conditions that vary over time, this means that the devices have to adapt, i.e., adjust operation in accordance with the current situation. Sheth et al. argue that adaptive decision mechanisms under such conditions require situation awareness, that is intelligent mechanisms that can convert raw data into something that is contextual meaningful [52].

Vernon goes further in [53], discussing the concept of self-awareness, i.e., the extent to which a system can reflect about itself. He states that self-awareness can be seen as a device's ability to see itself in relation to its context, learn from experience, predict the outcome of future events and act to pursue goals. Preden et al. support this view in [26], claiming that for devices operating in a dynamically changing environment, self-awareness is needed for devices to understand their own state in relation to the environmental conditions that influence their operation. Thus, a device manager can achieve autonomous self-management by combining situation-awareness with adaptation mechanisms to dynamically select its behavior, taking previous experience, contextual parameters, internal status and designated policies into account, as discussed by Foteinos et al. [34] and Sezer et al. [54].

A. A Taxonomy of Observed Adaptation Mechanisms

We see the considerations above confirmed in the reviewed cases, and observe a general pattern with autonomous device management based on adaptation mechanisms that analyze input data, reason about the current situation and produce some output data that result in a corrective action or plan, when needed. Many models include learning mechanisms as well, to expand the knowledge base of the system. These mechanisms are often encapsulated in separate modules, to reduce complexity and separate concerns. We will discuss this aspect further in Sect. VI. In particular, we identified three distinct types of adaptation mechanisms, indicated in columns 8, 9 and 10 of Table III:

- **Reasoning mechanisms** analyze sensed events and device states, reflect upon the current situation and control the internal data flow of the manager node. They also decide if a perceived situation requires adaptation.

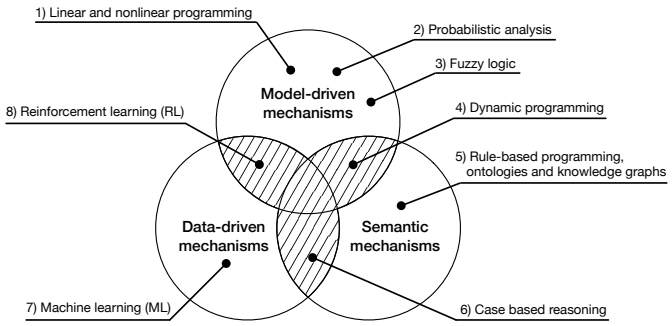

Fig. 1. Reasoning and learning mechanisms used in device management

- **Planning mechanisms** produce corrective actions or action plans, send instructions to the managed devices, and make sure the manager and the devices are in sync.
- **Learning mechanisms** make sure that the knowledge base is up to date, that it reflects the state of the devices in the context of the environment in which they operate.

In the following, we will take a closer look at reasoning mechanisms, since these are paramount for autonomous behavior. Perera et al. [55] classify reasoning mechanisms into six categories: supervised learning, unsupervised learning, rules, fuzzy logic, ontological reasoning and probabilistic reasoning. However, this classification neither differs between reasoning and learning mechanisms, nor covers all the different adaptation mechanisms we found in the reviewed papers. We therefore chose to categorize the different reasoning and learning mechanisms according to the underlying principle that the mechanisms use to infer an understanding of the situation, as shown in Fig. 1.

- **Model-driven mechanisms** capture knowledge and derive decisions through representation and rules that are declared explicitly. This means they are based on logics that are inherent to the model, which also implies that all variables that are part of the reasoning must be declared within the model itself. Thus, the output is a result of a logical analysis of a context change. Linear and nonlinear programming, probabilistic analysis and fuzzy logic can usually be placed in this category.
- **Semantic mechanisms** analyze structures where meaning is associated with the data. This allows data to be interpreted in context, regardless of differences in syntax or structure [52]. Knowledge is usually stored in a knowledge base that holds facts about the world, often in the form of ontologies, knowledge graphs or episodes. The semantic reasoning mechanism is an inference engine that applies logical rules to the knowledge base to deduce new information. Its output is thus based on inferring a deeper meaning or finding similarities to the input variable. Case-based reasoning, ontology-based inferring and rule-based programming are often placed in this category.
- **Data-driven mechanisms** are based on some statistical analyses that attempt to identify patterns in the data. This can be historical data collected by the devices themselves, or it can be other relevant contextual data collected from

external sources. The output is statistically inferred from the data itself. Most machine-learning methods can be placed in this category.

We will now take a closer look at the particular mechanisms used in literature with examples for their usage.

1) Linear and Nonlinear Programming: Linear and nonlinear programming are mathematical models often used for optimization purposes. The main difference is that in nonlinear programming, a change in input is not necessarily proportional to the change in output. Often, nonlinear problems are approximated using linear equations and algorithms to reduce complexity. Linear and nonlinear programming rely on all decisions being in place up front. As a consequence, this family of mechanisms is rarely used to solve complex problems where the answer is inferred from the data, like cluster analysis. A typical example of linear programming is found in [10] where Moser et al. use multiparametric programming for power management of solar harvesting wireless sensor nodes. They present a formal model for solving the optimization problem offline in different environmental conditions and system states, maximizing the utility in a long-term perspective.

2) Probabilistic analysis, Markov-modeling and Bayesian inference: Applying probability theory is a common method for quantitative modeling and analysis of large, stochastic data sets. Probabilistic models use graphs to represent stochastic variables, where edges represent assumptions that are conditionally independent. This means they are well suited to solve problems related to joint probability distributions. Bayesian inference uses the Bayes' theorem to update the probability for a given result as more data becomes available. Thus, by applying structural learning, knowledge can be retained within the model itself. Probabilistic models of both types are often used for dynamic analysis of sequences of data. However, these mechanisms often come with a high computational cost, especially in models with a large number of parameters. Also, the process of choosing the prior distribution is time-consuming and often requires in-depth expertise of the problem to solve. This makes it difficult to apply such methods on constrained devices or in domains characterized by non-stationary properties. An example of a probability-based reasoning mechanism can be seen in [28] where stochastic randomness in available resources and numerous allocation options, in conjunction with reinforcement learning, make a Markov decision process a good fit to solve the problem of computational offloading.

3) Fuzzy logic: Fuzzy logic is used to model logical reasoning on sets that amount to degrees of membership, fuzzy sets. It is based on the idea that a preposition can be partly true and partly false at the same time, with a degree of truth usually defined as a real number in the interval [0,1]. This allows a continuous range of choices [56]. A central aspect of fuzzy logic is the mapping to linguistic variables, like *slow* or *tall*. In a fuzzy expert system, such linguistic variables are used to produce fuzzy rules, which are used to infer a decision from the model. New knowledge can be retained within the model itself, by manipulating fuzzy sets, or by applying new fuzzy rules semantically. Fuzzy logic is suitable for domains where the input is imprecise and the outcome is uncertain. However, fuzzy systems lack the capability of learning from experience or recognizing patterns, and extensive testing is often needed for validation and verification of fuzzy knowledge-based systems. In addition, the iterative process of defining fuzzy rules and membership functions is time consuming and requires expert knowledge [57]. Preden et al. [26] present an example of fuzzy logic, where an application first associates situation parameters like sleeping time, breathing patterns, heart rate and movement with degrees of membership in fuzzy sets before fuzzy rules estimate the quality of sleep.

4) Dynamic programming and recursive optimization: The main principle of dynamic programming is to break a problem down into smaller sub-problems, often to find an optimal score using recursion. Each sub-problem is then solved sequentially, and the result of each iteration is retained in a dynamic programming matrix and used as input to the remaining sub-problems. Finally, the algorithm does a traceback of the matrix to recover the structure of the optimal solution. Although dynamic programming shares properties with semantic and model-driven mechanisms, it is not particularly suited for solving causal inference since it might align unrelated sequences [58]. Thus, dynamic programming is often used as part of, or as a complement to, other adaptation techniques. An example of recurrent dynamic programming can be seen in [30], where Samie et al. use it for energy-aware QoS management of IoT devices under bandwidth, battery, and processing constraints.

5) Rule-based inference, ontologies and knowledge graphs: These methods are all based on semantics, associating meaning to collected data. In systems that rely on semantic adaptation mechanisms, knowledge is usually represented by defining a set of concepts and the relationship between them [21]. In this way data can be interpreted contextually, that is, detached from the syntax or structure, by using annotation techniques to infer knowledge from the interpreted data [52]. Reasoning typically attempts to derive facts that are not explicitly expressed in the ontology or knowledge graphs [59]. New knowledge obtained this way can be retained and stored as a new rule, a new ontology or as an expansion of the knowledge graph. However, since the logic is based on semantics these methods are not particularly suited to solve problems that are based on quantifiable data. An example of ontology-based reasoning can be seen in [41] where knowledge related to a situation is represented in a semantic-based context mode that contains definitions of basic concepts and relations. This provides a common vocabulary that can be used to manage and share context data among users, devices and services.

6) Case-based reasoning: The assumption behind case-based reasoning is that similar problems have similar solutions [60]. Knowledge is usually captured and retained as episodic data in a case base. This means it can be categorized as a hybrid mechanism, part semantic and part data-driven. To solve a problem, a reasoning mechanism typically uses the principle of analogy to retrieve and reuse episodes that match the current situation. Since case-based reasoning considers what happened rather than on how or why it happened, it is well suited for domains where the context is not explicitly defined [61]. This sets it apart from mechanisms based on

semantics. We can see an example of case-based reasoning in [22], where information stored in a knowledge base is used as a case base to find an appropriate action to improve network lifetime and quality of information.

7) Machine learning (ML): In machine learning, data is analyzed statistically using analytical or mathematical models that identify patterns in the data. The ML algorithms are trained using sample data, and can then be used to make predictions or decisions without explicit programming [42]. In *supervised learning* the data is labeled and an output is mapped to an input. Thus, it infers a function from labeled training data. The output can be a continuous numerical variable found through regression, or a discrete or categorical value found through classification. In *unsupervised learning* the data is unlabeled and the algorithms learn to find unknown patterns or structures in the data. It is typically used for clustering, to partition data sets into groups [54]. For both methods, learning happens when previously unseen data is added to the training data and the algorithm is retrained on the new data set. Some limitations of ML origin in its reliance on statistical data. One problem with ML is the need to train each model to fit the particular application. Also, the relationship between output and data is encoded as correlations, but causality or relationships cannot be inferred. In IoT management we can see examples of supervised learning in [19], where Megahead et al. use a support vector machine classifier for autonomous policy determination, and in [40], where Kraemer et al. use random forest regressors and artificial neural networks for energy-aware self-management of solar-powered IoT devices. We did not see a clear example of clustering using unsupervised learning in any of the reviewed cases.

8) Reinforcement learning: Reinforcement learning (RL) is commonly used for control optimization problems with many states and complex stochastic structures. It employs a reward function and learns through interaction of an agent with its environment, with no need for a complete control model or explicit supervision [22]. An RL agent is trained to improve a task by learning from experience, that is, interacting with that particular task in context [62]. The algorithm is trained with the goal to maximize the cumulative reward. The agent thus learns the policy that produces the highest reward while avoiding policies that produce low or negative rewards. We categorize RL as a hybrid of data-driven and model-driven mechanisms, since environments that provide rewards are often based on a mixture of explicit models and data. An example can be seen in [37], where Edalat et al. use reinforcement learning for network lifetime optimization. Challenges with RL are the design of the reward function, as this requires an in-depth knowledge of the domain and the system goals, as well as a potentially high training effort [62].

B. Handling Complexity by Combining Mechanisms

Table III shows that reasoning, learning and planning mechanisms often are combined to solve a particular problem. We also see that some models employ a mix of different categories of adaptation to achieve adaptation. A synthesis of this observation is presented in Table IV. Here we indicate the category of reasoning and learning mechanisms, grouped by the type of operational environment, for each of the 32 cases presented in Table III.

TABLE IV
OVERVIEW OF OBSERVED REASONING AND LEARNING MECHANISMS

	Stationary environment		Non-stationary environment	
	Reasoning	Learning	Reasoning	Learning
Model-driven	4	0	7	3
Data-driven	1	0	4	13
Semantic	5	4	8	2
Mixed	0	1	3	0
Not included	0	5	0	4
Total	10	10	22	22

For systems operating in a stationary environment, we see that only 1 system uses a data-driven mechanism, most likely because behaviors can be adjusted in a deterministic manner. Also, just 5 out of 10 cases include a learning mechanism, 4 of which are purely semantic and 1 has a strong semantic component. For systems that are deployed in a non-stationary environment, we see that there is more variation in which type of reasoning mechanisms is used. Here, 7 cases used a model-driven approach, 4 used data-driven, and 8 used semantics. In addition, 3 cases used a mixed approach to reasoning, with data-driven reasoning being a component in all of them. 18 of 22 systems incorporate learning mechanisms, 13 of which are primarily data-driven.

V. PATTERNS AND MODELS FOR AUTONOMOUS MANAGEMENT IN IOT

We now turn our attention to the combination of the various adaptation mechanisms and how their interactions lead towards a cognitive system. The 32 reviewed cases expose a wide variety in description style and rigor. Most detail only some aspects while neglecting other components or processes that are necessary to understand the bigger picture. 19 models mainly focus on component composition and interactions, while the other 13 models focus on a description of the process. This variety makes it difficult to interpret the adaptation process as a whole in each case. We therefore extract partial models from the reviewed cases and expand them with more general literature and cognitive models.

A. Patterns Observed in Literature

The 32 reviewed architectures typically divide reasoning, learning and planning mechanisms into separate components. The descriptions in 23 papers were so detailed that we could extract the relation between these three mechanisms (marked with *high* in column 11 of Table III). Our study reveals the following patterns:

1) The managing process is centralized, that is, the network is organized in either a star or a cluster topology, as for instance in [26], [16], [35].
2) Sensed events are sent from the device to a reasoning mechanism for analysis, for instance [22], [18], [36].

3) Reasoning processes are initiated either from an observed event, an internal process or a prediction, for instance in [34], [32], [27].
4) Reasoning and learning mechanisms are distributed throughout the architecture, for instance in [21], [19], [29], [41].
5) Learning mechanisms are placed in conjunction with a component responsible for assessing the situation, for instance in [24], [38], [31].
6) There is a separation of concerns between two main processes, namely understanding the current situation and planning a corrective action, for instance in [20], [13], [10].
7) Task allocation, goals and policies are managed by a planning component, e.g., in [33], [37], [39], [40].

Combining multiple adaptation mechanisms is a common strategy. A reason for this is that distributed processes often complement each other, especially in systems that need to solve tasks that require understanding on a higher cognitive level, that is, self-, context- or situation-awareness [55]. In these cases we see that the interaction between the components, and the interplay between the different adaptation mechanisms, define and produce the internal cognition.

B. The MAPE-K Autonomic Control Loop

A general autonomic system architecture is based on sensors and actuators, controlled by a feedback loop [63]. A typical feedback-control-loop involves four steps: 1) Collecting and monitoring sensing and contextual data; 2) Processing and analyzing the collected data, which may trigger a need for adaptation; 3) Making a decision on what to change, based on an adaptation goal; and 4) Executing adaptation through an appropriate mechanism. The new, adapted state of the system is then returned into the feedback loop by a self-reflective mechanism that is used throughout the adaptation cycle. The accumulated data is stored for future reference in a knowledge base, and used to provide a more accurate model of past and future states, in an attempt to identify symptoms and infer trends that go into the decision planning [64].

Many autonomous and self-adaptive systems that make use of sensory input are based on the MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) autonomic feedback loop [65], [66]. A model of this type of control loop is shown in Fig. 2. This architecture allows a device to manage itself and dynamically adapt to changes based on predefined policies and objectives. Some learning is inherent in the model by retaining sensed information and saving the effect of an executed action. Few of the mechanisms identified in Sect. V-A directly refer to MAPE-K, but we observe that the pattern of the four reasoning processes, i.e., monitor, analyze, plan and execute, is present in many of the reviewed papers. It is therefore natural to incorporate this pattern in a generalized model, which in IoT corresponds to a system where devices receive events through sensors or internal processes, and respond to these events through an analysis of the situation and planning of adaptive actions.

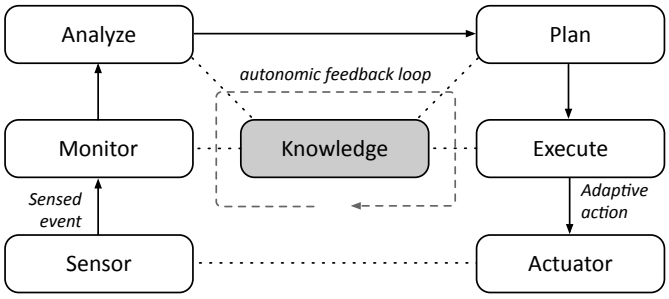

Fig. 2. Basic model of a MAPE-K autonomic control loop. Adapted from [66].

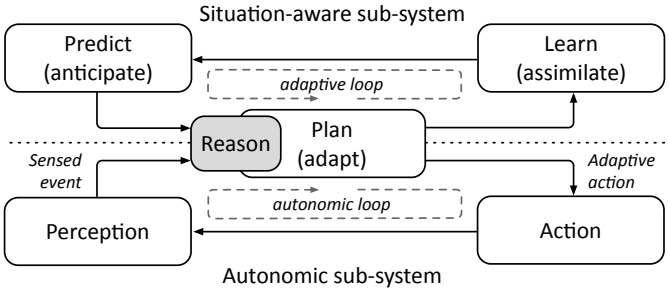

Fig. 3. A model of cognitive planning. Adapted from Vernon's cognitive cycle [53].

C. Cognitive Models

From Tables III and IV we observe that the adaptation mechanisms found in purely autonomic systems tend to be preconfigured, while situation-aware systems more often are able to create their own rules through learning-by-experience. Pramanik et al. redefine the concept of Cognitive IoT found in network management as a process where a stateful and probabilistic system adapt to dynamic changes through situation-awareness and iterative self-learning [50]. This pattern closely resembles Vernon's cognitive cycle, which is based on two independent cycles of (1) perception and action; and (2) anticipation, assimilation and adaptation. In his model, planning is implicit in the process and intelligent behavior thus emerges through circular causality, where global system behavior influences local behavior of system components, while local interactions between components in turn determine global behavior [53]. In contrast to this model, all models in the reviewed cases include planning as a separate component. To reflect this, we created a pattern based on the cognitive cycle that we named *cognitive planning*, shown in Fig. 3.

In the cognitive planning model, the planning process is the central component. The functionality is divided in an autonomic and a situation-aware subsystem, which are executed through two separate control loops. The *autonomic loop* registers an event and initiates a response in accordance to its active policy. The *adaptive loop* analyzes the event and changes the policy if the situation calls for adaptation. This division has two implications: First, the autonomic subsystem can operate without the need for learning, that is, as a simple stimuli-reaction loop. Second, since the situation-aware subsystem receives stimuli from the autonomic subsystem, it

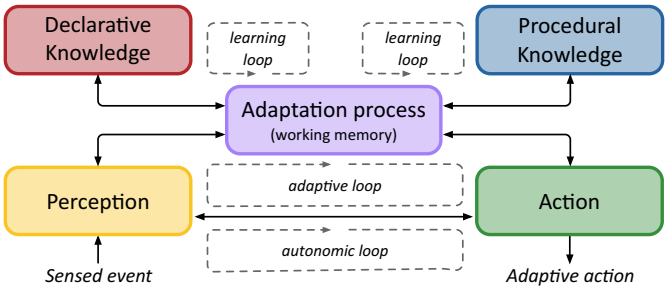

Fig. 4. A standard model of human-like minds. Adapted from [67].

continuously learns from experience. We will later see that this increases autonomy and robustness of the process.

The cognitive reasoning process is initiated either from an observed event or from a prediction. Depending on stimuli received, the reasoning processes can trigger the planning process. This implies either initiating a learning process, performing an adaptive action, or both. To reflect this, we placed reasoning as a sub-process that precedes planning. This pattern matches both the reviewed architectures that actively use learning to adapt to changes in their context, and those that do not include a separate learning process, i.e., purely autonomic systems.

Cognitive architectures are often organized around a working memory that uses a cognitive cycle to collect sensory input, retrieve appropriate declarative or procedural knowledge and initiate adaptive actions. Laird et al. describe this architectural style in a standard model [67], shown in Fig. 4. They state that a key characteristic of intelligent behavior is that changes in a working memory correspond to steps in an abstract reasoning process or internal simulation of an external action. Thus, adaptation emerges from a combination of the implemented architecture, acquired knowledge and learned skills. Implicitly, this corresponds to a system with three different types of control loops. The autonomic and adaptive loops have the same traits and behavior as in the cognitive planning model above. However, in the cognitive architecture the learning processes are outsourced to separate control loops that are responsible for updating declarative and procedural knowledge that is stored within the system, in accordance to dedicated learning policies.

Sifakis et al. concretize the idea of separating knowledge in [49] with a computational model for agents. This model includes declarative knowledge that represents facts about the world, i.e., the entities found in a domain and the relations between them. This knowledge can for instance be stored as system properties, training data, logical formulas or patterns. Procedural knowledge, on the other hand, is represented by executable methods like behavioral descriptions of components, analytical algorithms or prediction techniques like machine learning models.

VI. Best Practices for Autonomous IoT Device Management Systems

Our literature study shows that complexity is a challenge in most large-scale deployments of autonomous IoT device management systems, due to scale, constraints and heterogeneity within the system, and non-stationarity of the environments. Based on our observations, we were able to identify five best practices (BP) that can guide the design and implementation of such systems. The first three BPs are a direct result from our observations, and BP4 and BP5 are derived from an overall analysis of the reviewed papers and the patterns observed in them. In the following paragraphs, we will describe these practices, and give recommendations for when and how to apply them. For clarity, we use imperative language.

BP1: Employ adaptation mechanisms according to environmental stationarity. We observe that the selected adaptation mechanisms correlate with the type of environment: In *stationary environments*, there is usually little need for explicit learning, as the statistical properties of the environment are constant. Adaptive actions can hence be pre-programmed and automated, using semantic methods or purely model-driven mechanisms. Learning processes in such systems are mainly used to identify new rules or policies. In contrast, when operating in *non-stationary environments*, environmental factors often vary from device to device, and they need to adjust their operation to unexpected events. This means that systems must be able to retain and store knowledge of observed events, the action that was taken, and the corresponding effect. They also have to reason about the implication of said action, to make the most suitable corrective action the next time the same situation arise. In other words, they have to employ advanced adaptation mechanisms grounded in previously collected data.

Therefore, assessing the environment of devices is critical: For systems managing devices operating in a stationary environment, the often lower complexity of a purely autonomic architecture may be sufficient. For systems operating in non-stationary environments, one should consider adding a self-aware subsystem to allow explicit learning, since this is often a prerequisite for adaptation under such conditions.

BP2: Select topology according to the inherent systemic constraints and requirements. System topology and locus of computation are influenced by constraints inherent to IoT. Constraints in connectivity favor computation on the device at the edge of the system. We found only one distributed topology [10], where Moser et al. use linear programming for adaptive energy management directly on the IoT device.

The majority of the papers are concerned with constraints inherent in the device rather than the network. This favors cluster or star topologies, as they have the potential for better access to memory, processing power, energy and contextual information. The main argument for employing a star topology is that a fully centralized management improves elasticity [68], allowing easier access to resources-as-a-service, which can add flexibility for systems where the management needs to vary over time. Another argument is that since the managing nodes also need to be managed autonomously, a star topology reduces overall complexity. Regarding the cluster topology, we see four benefits: 1) Clustering can help reducing latency, since processes are placed closer to the sensor devices [69]. 2) A topology that allows responsibility to be shared among the managing nodes in the network can be more suited to handle high variances in the network conditions or frequent changes in application requirements [13]. 3) With cluster organization

we can connect devices that share similar characteristics. This adds flexibility, since it allows that management nodes can use resources on the aspects of management that are most relevant for that particular group [70], [71]. 4) Many IoT systems operate under conditions that may cause instability in the operation. A clustered topology can improve dependability since it reduces the risk that the whole network is shut down in case of internal or external events, like handling too many active connections at once or a power break [25].

We conclude that a star topology fits systems that require variable access to processing power, memory or storage. For systems where low latency or high dependability are key concerns, or where manager nodes are specialized for specific management tasks, a clustered topology may be preferable.

BP3: Separate concerns and reduce complexity with modularization. Modularization is a general mechanism to handle system complexity that also applies for device management systems. Consequently, the majority of the reviewed architectural models divide reasoning, learning and planning mechanisms into separate components. Planning is often done in a component with a central, coordinating role. Systems operating in stationary environments often employ a basic architectural model made up of few components, while higher environmental complexity is often handled by adding more components and using distributed reasoning mechanisms throughout the architecture to control the data flow between components. Such modularization allows specialized subcomponents for a particular task. This enables a better understanding and control of the data flow and of the different states of the system, making it easier to integrate mechanisms into managers and to support black-box designs [72]. Another benefit is that modularity ensures that parts of the system can be replaced or extended independently if the requirements or the understanding of the problem change, which can reduce the risk of project failures due to high complexity.

BP4: Control parallelism and data flow with triggers. Parallelism is a concern in architectures that manage many devices simultaneously. Adaptation processes can be active in several components at the same time. Models need hence to include a detailed description of how these processes work internally, that is, how they are activated and which components they activate in turn. However, few papers mention how and when components are activated.

We suggest to explicitly present triggering mechanisms that describe how knowledge is transferred from an originator to a consumer and how adaptation processes are activated. Such triggers should possess both *push* and *pull* directions, so that both originator and consumer component can trigger the knowledge flow. For instance, an observed event or a predicted future state can trigger the creation of a new plan, corresponding to an originator pushing knowledge towards a consumer. Vice versa, a planning component as a consumer of knowledge may trigger a specific procedure in pull direction if it needs a specific prediction for a future state. This mechanism also works for learning where, for example, a previously unseen event can trigger a command to update the declarative knowledge connected to that event.

BP5: Represent devices by digital twins. A challenge in autonomous management of constrained IoT devices is to keep track of the past, current and future state of each device *individually*. For such tasks, the concept of digital twins is used in industrialized IoT. A digital twin is a virtual representation which reflects a specific physical device. Such complete and holistic representation, as opposed to a more fragmented organization, makes it easier to model the behavior of the devices individually. A central manager can adjust the operation of each device based on their actual experience. The twin can then serve as a platform for simulating behavior and recommend optimal actions in a given situation [73].

Though we have not seen this concept explicitly in IoT device management, virtual representations of devices are central in several of the reviewed cases to allow autonomous self-management and context awareness [74]. One example can be seen in [31], where virtual objects are used to represent both end devices and devices that are responsible for adaptive management. This is challenging because it implies that a device and its twin need to stay synchronized for the representation and its result to be valid.

VII. A Generalized Cognitive Model for Autonomous IoT Device Management

We synthesized a generalized cognitive model for autonomous management of constrained IoT devices, shown in Fig. 5, based on the observations in Table III, the MAPE-K loop in Fig. 2, the adapted model of cognitive planning from Fig. 3, and the standard model of a cognitive architecture in Fig. 4. It provides a blueprint that describes the reasoning, learning and planning processes for autonomous adaptation, the interaction between these processes, and the declarative and procedural knowledge involved in such a system.

A. Component Structure and Digital Twins

Apart from the physical device instances deployed in the field, the model is structured by two types of managers that serve a *system* and a *device* perspective:

- The *System manager* is responsible for assessing the past, present and future states of the system as a whole. It contains knowledge that explains the world, that is, the nature of the environment in which the system is placed, and how the environment influences the operation of the devices. When new data is collected, the system manager applies the necessary filters, before it separates the data into declarative knowledge belonging to either the world or a device, respectively. The main output from the system manager are hence predictions of how external conditions will influence each device in the future.
- The *Device managers* are responsible for assessing and planning the operation of the devices themselves. Following the pattern of digital twins, there is one device manager instance for each physical device. Their main tasks are to monitor the knowledge related to each device and produce plans that reflect the context and environment in which each device is operating.

This separation between situation awareness and planning is prominent in many of the reviewed architectures, for example in [22], where Al-Turjman et al. use context awareness

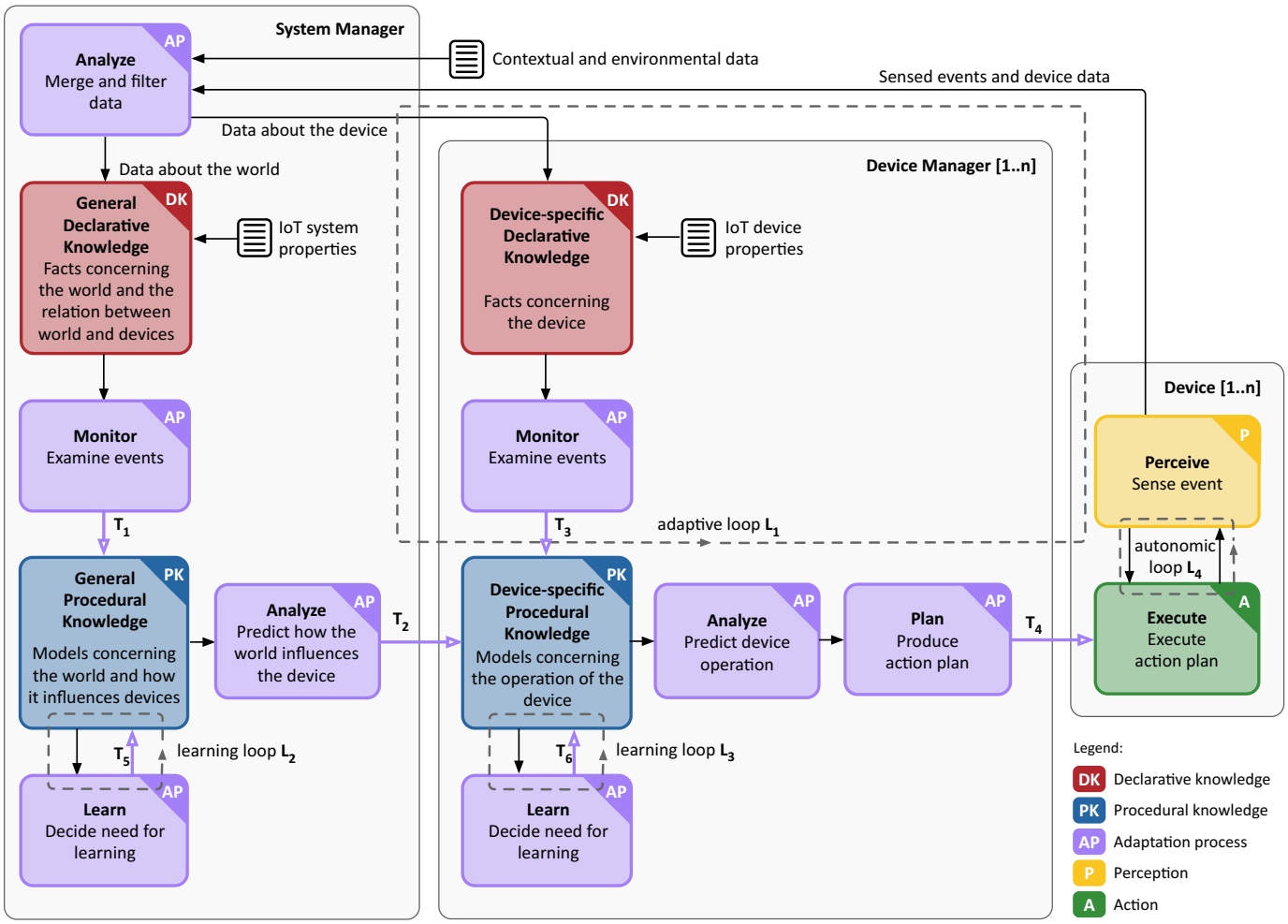

Fig. 5. A generalized cognitive model for autonomous management of constrained IoT devices.

to optimize network lifetime in a wireless sensor network. Other examples are adaptive configuration management [34] and energy-aware self-management [40]. This organization facilitates sharing of information among devices, for instance as value distributions that model the environment in a given context or as generalized episodes that can be used to index a structure for matching and retrieval of similar cases [60]. Sharing of data may also be beneficial for devices with common properties [75], as for example solar energy intake patterns of devices with similar location and orientation [32].

B. Separation of Declarative and Procedural Knowledge

Both top-level components contain subcomponents to handle declarative and procedural knowledge separately. The system manager contains a component to handle declarative knowledge about the world, and the device managers one to handle declarative knowledge about the device. The more interpretive procedural knowledge is encapsulated in the respective modules PK.

This separation is beneficial as these different types of knowledge often use different implementations, as discussed in Sect. IV. For machine learning, for instance, declarative knowledge can be stored in the form of training data, and procedural knowledge is represented by trained machine learning models. Many papers separate these types of knowledge, too. An example can be seen in [13], where Minh et al. separate policies from contextual knowledge. Other examples are dividing knowledge in contextual models, adaptation options, adaptation goals and plans [18], and separating a context repository from logical rules [21].

C. Control Loops for Adaptation

The model contains in total four control loops that handle adaptive behavior in the system.

- L_1 is the main control loop. It follows the MAPE-K pattern from Fig. 2. However, since we have added procedural knowledge components that contain prediction models to the loop, it acts more like the adaptive loops seen in Figs. 3 and 4. L_1 can take two paths through the model, depending on whether a change originated in the environment or in a device. The monitor component in the system manager will detect a change in the environment, which will trigger an analysis of the contextual change. Likewise, the monitor component in the device manager will detect any change related to a single device. This

triggers an analysis that evaluates to which degree the change influences the operation of the device, taking current and future environmental context into account.
- L_2 and L_3 are learning loops that control the incremental learning processes of the system manager and device manager, respectively. They follow the pattern of the situation-aware subsystem (Fig. 3), as they act on incoming events and decide if there is a change that requires learning. Since declarative knowledge is added continuously, the learning loops are only specified for the procedural knowledge components in the model. This is a distinction from Fig. 4.
- L_4 represents the autonomic subsystem from Fig. 3. It controls the device in a short-term perspective based on the last plan sent from the device manager.

D. Explicit Triggers

In line with the best practice identified before, the model identifies explicit triggers that guide the data flow to the correct component and control the behavior of the management system in general, based on local decisions. The triggers are shown as arrows pointing from the originator of knowledge to its consumer, but they can be activated from both directions, as described by BP4 in Sect. VI.

- A change in the general declarative knowledge base may trigger T_1, which in turn activates a model of the world or a model of how the environment influences a device.
- After running a general procedural model, the system manager analyzes the result and triggers T_2 if there is a change in the situation that can influence the operation of a device. This analysis may need to pull specific device data stored in the device manager, to run the analysis. The prediction or result of the analyzed situation is then sent to the device manager for further processing.
- Trigger T_3 monitors device-specific declarative knowledge to see if there is a need for activating a particular behavioral model of that device. After running a device-specific model or procedure, the device manager will produce a plan based on the result from this process. A policy-based planning procedure may for instance analyze the goal of the device, and then select the policy that addresses the present problem in the best way, taking recent results and predictions into account.
- If this new plan deviates from the previous, trigger T_4 informs devices that should adapt to this new plan, as we will outline later.
- In each manager there is a special reasoning process that triggers T_5 resp. T_6 if there is need for learning. Other types of learning triggers are for instance previously unseen data, or the discovery that an executed plan did not have the anticipated effect.

We see examples of such distributed reasoning processes in many of the reviewed cases. For instance, Shah et al. use this pattern to select which task to execute [27]. Other examples are matching similar situations based on previous experience [34], selecting training data based on correlation [32] and context-aware planning of corrective actions [41].

E. Adaptive Instructions vs. Actions

Trigger T_4 is a special adaptation process that serves two purposes: (1) It triggers the creation of an adaptation plan for a device and (2) controls the transfer of that plan to the mirrored device. The latter reflects the *motor action* commonly employed in cognitive architectures [67]. This combined mechanism effectively decouples the adaptive instruction, which is managed by a manager node, from the adaptive action, which is managed by the device. The purpose of this separation is to ensure that the digital twin and the device always are in sync. If they are not synchronized, the manager risks to wrongly interpret the effect of a corrective action, which again will cause it to make incorrect assumptions about the device. In addition, the learning processes may suffer, since unsynchronized states may cause noise or anomalies in the training data. One way to mitigate this problem is to keep the last instruction sent to the device in the working memory of the device manager until it receives an acknowledgment that the instruction was consumed by the device. Another argument in favor of decoupling is that a device capable of some basic reasoning might discover that it has a need for adaptation due to a sudden external event. When this happens, the decoupling allows a device to take an adaptive action on its own, and then send an instruction to the manager that informs it about the action that was made. Again, the device needs to keep a copy of the instruction in its memory until the device manager has acknowledged that the states are synchronized.

None of the reviewed cases explicitly address the difference between an adaptive instruction and an adaptive action. However, Sifakis et al. mention the importance of synchronizing agents and devices in [49].

F. Example Use Case: Solar-Powered IoT Devices

We illustrate the proposed model with a management system for solar-powered air quality sensing devices. It provides feedback to devices about their expected future solar energy intake based on the weather forecast, so that they can adjust their operation to the availability of solar energy and maintain perpetual, energy-neutral operation.

Apart from air quality data, the system manager collects data about the solar energy intake from the devices and the weather forecasts for the devices' locations. Data relevant to the individual device is passed on to the respective device managers. With this data, they can use a prediction model [76] to estimate the energy intake for each device. This estimation is used to plan how many measurements each device can take, also considering the current state-of-charge of the device. To speed up learning, the device manager may also employ training data from other devices [32] provided via the system manager. They may also employ a set of prediction models from which the currently best one is selected [77].

The four control loops guide the information flow through the system. The adaptive control loop L_1 examines the weather forecasts in respect to the device status, and instructs devices to change their operation if the anticipated energy budget changes significantly. L_2 and L_3 guide the learning process. Prediction models for energy intake can for instance be retrained at

midnight, or when the training data significantly changes. The autonomic loop L_4 follows the policy sent by the device manager unless it detects a state-of-charge of the battery that is lower than anticipated by the device management. If that happens, it executes a local policy that restricts the amount of energy that is consumed until it receives updated instructions from the device manager.

VIII. Discussion

We developed the proposed reference model bottom-up, emerging from the systems found in literature. Still, there is no guarantee that it fits every specific device management use case perfectly. A challenge may hence be the application of the model to specific use cases. However, as we managed to unify the emerging model with the principles for autonomic computing (Sect. V), we argue that the resulting model with its loops, main components and responsibilities is likely to be relevant for a wide range of device management tasks. Further, the reference model should not be seen as a rigid design, but rather a blueprint for an architecture that can be further refined following the best practices listed in Sect. VI.

Our survey reveals the wide range of adaptation and learning mechanisms, often more than one within a single system, see Sect. IV. This diversity is good, but developers are often only familiar with a subset of techniques, and not necessarily the most suitable ones. Hence, while our model covers the overall system, the initial selection, application and detailed design of the specific learning and adaptation mechanisms can remain a challenge. Here we expect a maturation of the field, where best practices also regarding the detailed learning and adaptation mechanisms become commonplace and ultimately off-the-shelf components. The current rise in interest and competence within machine learning among developers makes this a likely scenario.

For further research and development, we see the mapping of the reference model to standard components of commercial device management platforms and platforms that automate machine learning tasks. Alongside the expected maturing of the field regarding the adaptation and learning mechanisms for specific concerns, we expect this to be the main driver for consolidation and further progress.

Even though handling security is out of scope for this review, we stress the importance of considering this aspect when designing device management systems. Security should be handled as an integral part of an architecture. Apart from integrating mechanisms like authentication, authorization and certification, security functions are increasingly subject to learning and adaptation, and hence drawing benefit from a better management of them.

IX. Conclusion

We reviewed the state of the art for autonomous device management in IoT. First, we conducted a comprehensive and structured study on the aspects that need to be considered when designing and implementing systems for autonomous management of constrained IoT devices. We further synthesized a taxonomy of the most commonly used adaptation mechanisms in IoT device management and studied how and when they are applied. We combined these findings with general state-of-the-art models of autonomous systems to identify common patterns for autonomous management. From the conducted work, we made two major contributions that help advancing the field: first, we managed to summarize insights of the literature by identifying five best practices for design and implementation; second, we synthesized a generalized cognitive model for autonomous management of constrained IoT devices. This generalized model follows the identified best practices, adheres to the general principles of autonomic computing and connects them with the requirements for the IoT domain. In this way, the proposed model contributes to the vision of efficient and sustainable IoT systems that reduce or prevent expensive downtime or human intervention.

References

[1] E. Jantunen, G. Di Orio, C. Hegedűs, P. Varga, I. Moldován, F. Larrinaga, M. Becker, M. Albano, and P. Maló, *Maintenance 4.0 World of Integrated Information*. Springer International Publishing, 2019. [Online]. Available: http://link.springer.com/10.1007/978-3-030-13693-2_6

[2] D. M. Brock, "Autonomy of individuals and organizations: Towards a strategy research agenda," *International Journal of Business and Economics*, 2003.

[3] T. Lewandowski, D. Henze, M. Sauer, J. Nickles, and B. Bruegge, "A software architecture to enable self-organizing, collaborative iot ressource networks," in *2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*. IEEE, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9144772/

[4] M. Tahir, Q. Mamoon Ashraf, and M. Dabbagh, "Towards enabling autonomic computing in iot ecosystem," in *2019 IEEE Int. Conf. on Dependable, Autonomic and Secure Computing*. IEEE, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8890456

[5] J. Sifakis, "System design in the era of IoT - meeting the autonomy challenge," *Electronic Proceedings in Theoretical Computer Science*, 2018. [Online]. Available: http://arxiv.org/abs/1806.09846v1

[6] S. Sinche, D. Raposo, N. Armando, A. Rodrigues, F. Boavida, V. Pereira, and J. S. Silva, "A survey of IoT management protocols and frameworks," *IEEE Communications Surveys and Tutorials*, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8848791/

[7] A. Chowdhury and S. A. Raut, "A survey study on internet of things resource management," *Journal of Network and Computer Applications*, 2018. [Online]. Available: https://doi.org/10.1016/j.jnca.2018.07.007

[8] A. Čolaković and M. Hadžialić, "Internet of things (IoT): A review of enabling technologies, challenges, and open research issues," *Computer Networks*, 2018. [Online]. Available: https://doi.org/10.1016/j.comnet.2018.07.017

[9] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and Durham University Joint Report, Tech. Rep., 2007.

[10] C. Moser, L. Thiele, D. Brunelli, and L. Benini, "Adaptive power management for environmentally powered systems," *IEEE Transactions on Computers*, 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5291691/

[11] M. Chen, F. Herrera, and K. Hwang, "Cognitive computing: Architecture, technologies and intelligent applications," *IEEE Access*, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2791469

[12] M. Vögler, J. M. Schleicher, C. Inzinger, and S. Dustdar, "A scalable framework for provisioning large-scale IoT deployments," *ACM Transactions on Internet Technology*, 2016. [Online]. Available: https://doi.org/10.1145/2850416

[13] T. C. Minh, B. Bellalta, and M. Oliver, "Dison: A self-organizing network management framework for wireless sensor networks," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*. Springer, 2013. [Online]. Available: https://doi.org/10.1007/978-3-642-36958-2_11

[14] Y. Sahni, J. Cao, S. Zhang, and L. Yang, "Edge Mesh: A new paradigm to enable distributed intelligence in internet of things," *IEEE Access*, 2017. [Online]. Available: https://doi.org/10.1109/ACCESS.2017.2739804

[15] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive configuration of lora networks for dense IoT deployments," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*. IEEE, 2018. [Online]. Available: https://doi.org/10.1109/NOMS.2018.8406255

[16] L. Lan, R. Shi, B. Wang, and L. Zhang, "An IoT unified access platform for heterogeneity sensing devices based on edge computing," *IEEE Access*, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2908684

[17] K. Da, P. Roose, M. Dalmau, J. Nevado, and R. Karchoud, "Kali2Much: a context middleware for autonomic adaptation-driven platform," in *Proceedings of the 1st ACM Workshop on Middleware for Context-Aware Applications in the IoT (M4IOT)*. ACM Press, 2014. [Online]. Available: https://doi.org/10.1145/2676743.2676748

[18] D. Weyns, M. U. Iftikhar, D. Hughes, and N. Matthys, "Applying architecture-based adaptation to automate the management of internet-of-things," in *European Conference on Software Architecture (ECSA)*. Springer, 2018. [Online]. Available: https://doi.org/10.1007/978-3-030-00761-4_4

[19] A. Megahed, S. Tata, and A. Nazeem, "Cognitive determination of policies for data management in IoT systems," in *Service-Oriented Computing, ICSOC 2017 Workshops. Lecture Notes in Computer Science)*. Springer, 2018. [Online]. Available: https://doi.org/10.1007/978-3-319-91764-1_15

[20] J. Zheng, Y. Cai, X. Shen, and Z. Zheng, "Green energy optimization in energy harvesting wireless sensor networks," *IEEE Communications Magazine*, no. 11, 2015. [Online]. Available: https://doi.org/10.1109/MCOM.2015.7321985

[21] K. W. Lee and S. H. Cha, "Ontology-based context-aware management for wireless sensor networks," in *Communications in Computer and Information Science*. Springer, 2011. [Online]. Available: https://doi.org/10.1007/978-3-642-23321-0_55

[22] F. M. Al-Turjman, "Information-centric sensor networks for cognitive IoT: an overview," *Annals of Telecommunications*, 2017. [Online]. Available: https://doi.org/10.1007/s12243-016-0533-8

[23] Q. Wu, G. Ding, Y. Xu, S. Feng, Z. Du, J. Wang, and K. Long, "Cognitive internet of things: A new paradigm beyond connection," *IEEE Internet of Things Journal*, 2014. [Online]. Available: http://ieeexplore.ieee.org/document/6766209/

[24] V. S. Shah, "Multi-agent cognitive architecture-enabled iot applications of mobile edge computing," *Annals of Telecommunications*, 2018. [Online]. Available: http://link.springer.com/10.1007/s12243-018-0648-1

[25] A. P. Athreya, B. DeBruhl, and P. Tague, "Designing for self-configuration and self-adaptation in the internet of things," in *9th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*. IEEE, 2013. [Online]. Available: https://doi.org/10.4108/icst.collaboratecom.2013.254091

[26] J. S. Preden, K. Tammemae, A. Jantsch, M. Leier, A. Riid, and E. Calis, "The benefits of self-awareness and attention in fog and mist computing," *Computer*, 2015. [Online]. Available: https://doi.org/10.1109/MC.2015.207

[27] K. Shah, M. Di Francesco, G. Anastasi, and M. Kumar, "A framework for resource-aware data accumulation in sparse wireless sensor networks," *Computer Communications*, 2011. [Online]. Available: https://doi.org/10.1016/j.comcom.2011.06.010

[28] G. R. Alam, M. M. Hassan, Z. Uddin, A. Almogren, and G. Fortino, "Autonomic computation offloading in mobile edge for iot applications," *Future Generation Computer Systems*, 2019. [Online]. Available: https://doi.org/10.1016/j.future.2018.07.050

[29] A. R. Hilal and O. Basir, "A collaborative energy-aware sensor management system using team theory," *ACM Transactions on Embedded Computing Systems*, 2016. [Online]. Available: https://dl.acm.org/doi/10.1145/2910574

[30] F. Samie, V. Tsoutsouras, S. Xydis, L. Bauer, D. Soudris, and J. Henkel, "Distributed QoS management for internet of things under resource constraints," in *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. ACM Press, 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7750969

[31] S. Sasidharan, A. Somov, A. R. Biswas, and R. Giaffreda, "Cognitive management framework for internet of things: – a prototype implementation," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*. IEEE, 2014. [Online]. Available: https://doi.org/10.1109/WF-IoT.2014.6803225

[32] A. E. Braten, F. A. Kraemer, and D. Palma, "Adaptive, correlation-based training data selection for IoT device management," in *Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*. IEEE, 2019. [Online]. Available: https://doi.org/10.1109/IOTSMS48152.2019.8939220

[33] J. M. T. Portocarrero, F. C. Delicato, P. F. Pires, and T. V. Batista, "Reference architecture for self-adaptive management in wireless sensor networks," in *International Conference on Adaptive and Intelligent Systems*. Springer, 2014. [Online]. Available: https://doi.org/10.1007/978-3-319-11298-5_12

[34] V. Foteinos, D. Kelaidonis, G. Poulios, P. Vlacheas, V. Stavroulaki, and P. Demestichas, "Cognitive management for the internet of things: A framework for enabling autonomous applications," *IEEE Vehicular Technology Magazine*, 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6662384/

[35] N. M. do Nascimento and C. J. P. de Lucena, "FIoT: An agent-based framework for self-adaptive and self-organizing applications based on the internet of things," *Information Sciences*, 2017. [Online]. Available: https://doi.org/10.1016/j.ins.2016.10.031

[36] J.-h. Park, M. M. Salim, J. H. Jo, J. C. S. Sicato, S. Rathore, and J. H. Park, "CIoT-Net: a scalable cognitive IoT based smart city network architecture," *Human-centric Computing and Information Sciences*, 2019. [Online]. Available: https://doi.org/10.1186/s13673-019-0190-9

[37] N. Edalat and M. Motani, "Energy-aware task allocation for energy harvesting sensor networks," *EURASIP Journal on Wireless Communications and Networking*, 2016. [Online]. Available: https://doi.org/10.1186/s13638-015-0490-3

[38] Q. Wei and Z. Jin, "Service discovery for internet of things: a context-awareness perspective," in *Proceedings of the Fourth Asia-Pacific Symposium on Internetware*. ACM Press, 2012. [Online]. Available: https://doi.org/10.1145/2430475.2430500

[39] S. Shresthamali, M. Kondo, and H. Nakamura, "Adaptive power management in solar energy harvesting sensor node using reinforcement learning," *ACM Transactions on Embedded Computing Systems*, 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3126495

[40] F. A. Kraemer, D. Ammar, A. E. Braten, N. Tamkittikhun, and D. Palma, "Solar energy prediction for constrained iot nodes based on public weather forecasts," in *Int. Conf. on the Internet of Things*. ACM Press, 2017. [Online]. Available: https://doi.org/10.1145/3131542.3131544

[41] V. Cristea, C. Dobre, and F. Pop, "Context-aware environments for the internet of things," in *Internet of Things and Inter-cooperative Computational Technologies for Collective Intelligence. Studies in Computational Intelligence*. Springer, 2013. [Online]. Available: https://doi.org/10.1007/978-3-642-34952-2_2

[42] J. M. Tien, "Internet of things, real-time decision making, and artificial intelligence," *Annals of Data Science*, 2017. [Online]. Available: http://link.springer.com/10.1007/s40745-017-0112-5

[43] F. Fleurey, B. Morin, A. Solberg, and O. Barais, "Mde to manage communications with and between resource-constrained systems," in *Model Driven Engineering Languages and Systems. MODELS 2011. Lecture Notes in Computer Science*. Springer, 2011. [Online]. Available: https://doi.org/10.1007/978-3-642-24485-8_25

[44] T. Chen, S. Barbarossa, X. Wang, G. B. Giannakis, and Z.-L. Zhang, "Learning and management for internet of things: Accounting for adaptivity and scalability," *Proceedings of the IEEE*, 2019. [Online]. Available: https://doi.org/10.1109/JPROC.2019.2896243

[45] L. Gürgen and S. Honiden, "Management of networked sensing devices," in *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*. IEEE, 2009. [Online]. Available: https://doi.org/10.1109/MDM.2009.88

[46] M. Meddeb, M. B. Alaya, T. Monteil, A. Dhraief, and K. Drira, "M2M platform with autonomic device management service," in *Procedia Computer Science*, 2014. [Online]. Available: https://doi.org/10.1016/j.procs.2014.05.534

[47] M. Zhang, H. Zhao, and R. Zheng, "Cognitive internet of things: concepts and application example," *IJCSI International Journal of Computer Science Issues*, 2012.

[48] F. C. Delicato, P. F. Pires, and T. Batista, *Resource Management for Internet of Things*, ser. Springer Briefs in Computer Science. Springer, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-54247-8

[49] J. Sifakis, "Autonomous systems – an architectural characterization," in *Models, Languages, and Tools for Concurrent and Distributed Programming. Lecture Notes in Computer Science*. Springer, 2019.

[50] P. K. D. Pramanik, S. Pal, and P. Choudhury, "Beyond automation: The cognitive IoT. artificial intelligence brings sense to the internet of things," in *Cognitive Computing for Big Data Systems Over IoT*. Springer, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-70688-7_1

[51] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, 2003. [Online]. Available: https://doi.org/10.1109/MC.2003.1160055

[52] A. Sheth, "Internet of things to smart iot through semantic, cognitive, and perceptual computing," *IEEE Intelligent Systems*, 2016. [Online]. Available: https://doi.org/10.1109/MIS.2016.34

[53] D. Vernon, *Artificial cognitive systems: A primer*. MIT Press, 2014.

[54] O. B. Sezer, E. Dogdu, and A. M. Ozbayoglu, "Context-aware computing, learning, and big data in internet of things: A survey," *IEEE Internet of Things Journal*, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8110603/

[55] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications Surveys & Tutorials*, 2014. [Online]. Available: https://doi.org/10.1109/SURV.2013.042313.00197

[56] A. Celikyilmaz and I. B. Türksen, *Modeling uncertainty with fuzzy logic*. Springer, 2009. [Online]. Available: https://doi.org/10.1007/978-3-540-89924-2

[57] M. Negnevitsky, *Artificial Intelligence: A Guide to Intelligent Systems*. Pearson Education, 2005.

[58] S. R. Eddy, "What is dynamic programming?" *Nature Biotechnology*, 2004. [Online]. Available: https://doi.org/10.1038/nbt0704-909

[59] G. P. Zarri, "High-level knowledge representation and reasoning in a cognitive IoT/WoT context," in *Cognitive Computing for Big Data Systems Over IoT. Lecture Notes on Data Engineering and Communications Technologies*. Springer, 2017. [Online]. Available: https://doi.org/10.1007/978-3-319-70688-7_10

[60] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI Communications*, 1994.

[61] S. Craw and A. Aamodt, "Case based reasoning as a model for cognitive artificial intelligence," in *Case-Based Reasoning Research and Development. ICCBR 2018*. Springer, 2018.

[62] A. Murad, F. A. Kraemer, K. Bach, and G. Taylor, "Autonomous management of energy-harvesting iot nodes using deep reinforcement learning," in *International Conference on Self-Adaptive and Self-Organizing Systems, SASO*. IEEE, 2019. [Online]. Available: https://doi.org/10.1109/SASO.2019.00015

[63] R. Sterritt, "Autonomic computing," *Innovations in Systems and Software Engineering*, 2005. [Online]. Available: https://doi.org/10.1007/s11334-005-0001-5

[64] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, "A survey of autonomic communications," *ACM Transactions on Autonomous and Adaptive Systems*, 2006. [Online]. Available: https://dl.acm.org/doi/10.1145/1186778.1186782

[65] P. Arcaini, E. Riccobene, and P. Scandurra, "Modeling and analyzing mape-k feedback loops for self-adaptation," in *Int. Sym. on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 2015. [Online]. Available: https://doi.org/10.1109/SEAMS.2015.10

[66] IBM-Group, *An architectural blueprint for autonomic computing*. IBM White paper, 2005.

[67] J. E. Laird, C. Lebiere, and P. S. Rosenbloom, "A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics," *AI Magazine*, 2017. [Online]. Available: https://aaai.org/ojs/index.php/aimagazine/article/view/2744

[68] F. Van den Abeele, J. Hoebeke, G. K. Teklemariam, I. Moerman, and P. Demeester, "Sensor function virtualization to support distributed intelligence in the internet of things," *Wireless Personal Communications*, 2015. [Online]. Available: https://doi.org/10.1007/s11277-015-2481-4

[69] T. Park and W. Saad, "Distributed learning for low latency machine type communication in a massive internet of things," *IEEE Internet of Things Journal*, 2019. [Online]. Available: https://doi.org/10.1109/JIOT.2019.2903832

[70] G. Fortino, A. Guerrieri, W. Russo, and C. Savaglio, "Middlewares for smart objects and smart environments: Overview and comparison," in *Internet of Things Based on Smart Objects*. Springer, 2014. [Online]. Available: https://doi.org/10.1007/978-3-319-00491-4_1

[71] P. Malo, B. Almeida, R. Melo, K. Kalaboukas, and P. Cousin, "Self-organised middleware architecture for the internet-of-things," in *Int. Conf. on Internet of Things (iThings 2013)*. IEEE, 2013. [Online]. Available: https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.92

[72] R. Bianchini, M. Fontoura, E. Cortez, A. Bonde, A. Muzio, A.-M. Constantin, T. Moscibroda, G. Magalhaes, G. Bablani, and M. Russinovich, "Toward ML-centric cloud platforms," *Communications of the ACM*, 2020. [Online]. Available: https://doi.org/10.1145/3364684

[73] S. Mohanty and S. Vyas, "Intelligence of things = IoT + cloud + AI," in *How to Compete in the Age of Artificial Intelligence*. Apress, 2018. [Online]. Available: https://doi.org/10.1007/978-1-4842-3808-0_7

[74] C. Savaglio, M. Ganzha, M. Paprzycki, C. Bădică, M. Ivanović, and G. Fortino, "Agent-based internet of things: State-of-the-art and research challenges," *Future Generation Computer Systems*, 2020. [Online]. Available: https://doi.org/10.1016/j.future.2019.09.016

[75] Y. Wei, Y. Zhang, J. Huang, and Q. Yang, "Transfer learning via learning to transfer," in *International Conference on Machine Learning*, 2018.

[76] F. A. Kraemer, D. Palma, A. E. Braten, and D. Ammar, "Operationalizing solar energy predictions for sustainable, autonomous iot device management," *IEEE Internet of Things Journal*, 2020. [Online]. Available: https://doi.org/10.1109/JIOT.2020.3002330

[77] A. E. Braten and F. A. Kraemer, "Towards cognitive iot: Autonomous prediction model selection for solar-powered nodes," in *2018 IEEE International Congress on Internet of Things (ICIOT)*. IEEE, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8473448/

Anders Eivind Braten Anders Eivind Braten received his degree in computer engineering from Sor-Trondelag University College (HiST), Trondheim, Norway in 2000, and his Cand. scient. degree in Computer Science from the Norwegian University of Science and Technology (NTNU), Trondheim, in 2006. He is currently pursuing the Ph.D. degree within the topic of application development for Internet of Things with the Department of Information Security and Communication Technology, NTNU, where he focuses on autonomous management for constrained IoT devices operating in non-stationary environments. He has a background as a consultant in the IT-industry and has been involved in numerous development projects in Norway.

Frank Alexander Kraemer Frank Alexander Kraemer (Member, IEEE) received the Dipl.Ing. degree in electrical engineering and the M.Sc. degree in information technology from the University of Stuttgart, Germany and holds a Ph.D. degree in model-driven systems development from the Department of Telematics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, since 2008. He is an Associate Professor with the Department of Information Security and Communication Technology at NTNU, and worked previously as Technology Manager at a startup for IoT software that he co-founded. His current research interests include Internet of Things architectures and application development, embedded and autonomous sensor systems, and the application of statistical methods and machine learning in constrained settings.

David Palma David Palma (Member, IEEE) received the B.Sc. degree in informatics engineering in 2007 and a Ph.D. degree in information science and technology in 2013 from the University of Coimbra, Portugal. He is an Associate Professor at the Department of Information Security and Communication Technology, Norwegian University of Science and Technology (NTNU), Trondheim, Norway. He was an H2020 Marie Sklodowska-Curie Postdoctoral fellow at NTNU and has worked in the past as a Researcher and Project Manager at OneSource, Portugal, as well as an invited Assistant Professor at the University of Coimbra. His current research interests are on cognitive IoT, networking in remote areas, routing, cloud-computing and software-defined networks.