2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)

Towards Resilient Internet of Things: Vision, Challenges, and Research Roadmap

Christos Tsigkanos, Stefan Nastic and Schahram Dustdar
Distributed Systems Group
TU Wien
Austria

***Abstract*—Internet of Things (IoT) systems open up massive versatility and opportunity to our world. Providing solutions for smart cities, healthcare, energy, and mobility, such systems increasingly permeate critical aspects of human activity. In a flourish of growth, these complex systems run software, are dynamic, without stable spatial and temporal boundaries, and involve mostly independent software components with different lifespans and evolution models. IoT systems provide data-centric, device-centric and service-centric functionalities that are subject to continuous disruption, under limitations such as resource-constrained devices, platforms heterogeneity, deployment in adverse environments and administrative domains. As these systems evolve and gain complexity, resilience becomes a crucial system property. Bolstering resilience entails understanding and systematically managing dynamic behavior and decentralizing operations. We advocate that to systematically engineer resilience in IoT systems, a complete rethink is necessary regarding their design and operation. In this paradigm shift, systems demand conceptual frameworks, techniques, and mathematically-backed formalisms to treat change and achieve decentralization. We outline a vision for addressing fundamental challenges that software engineering and distributed systems research encounters when building resilient IoT systems. Within a roadmap, we identify techniques and methods that can be leveraged to maintain resilience in the face of disruption, especially in the absence of central control and persistently at the system's runtime.**

I. Introduction

The recent evolution towards an increasingly integrated world has at its basis novel types of large-scale distributed systems achieved through new technologies and paradigms such as the Internet of Things (IoT), inducing systems composed of heterogeneous devices, computing infrastructures and cloud services. With these novel types of distributed systems also come along new types of requirements and an increasing need of satisfying them in a dependable manner [1], as systems often address societal challenges such as solutions for smart cities, healthcare, energy, and mobility, permeating more and more critical aspects of human activity.

IoT already bridges a number of research communities – from sensing, networking, and cyber-physical systems to control theory. IoT is increasingly made up of software, as devices now increasingly host (or are near) software stacks which manage data, interact with users, or implement control facilities. Complex software IoT systems [2] often (i) lack clearly stable spatial and temporal boundaries, (ii) are dynamic, and (iii) entail mostly independent software components with different lifespans and evolution models. Despite these common traits, these systems are quite different regarding system structure, environments they operate in, execution and coordination among elements within them.

IoT provides data-centric, device-centric and service-centric functionalities that are subject to continuous disruption, under limitations such as resource-constrained devices, platform heterogeneity, deployment in adverse environments or unknown administrative domains. This raises an abundance of issues related to reliable system requirements satisfaction. In an attempt to mitigate such deficiencies, often data or control in IoT systems is architecturally offloaded to the cloud. This however is not a panacea, as novel functional and non-functional requirements dictate data, computation or control to be situated locally near devices and not to the cloud. Such requirements may capture diverse system concerns ranging from reliability to performance or privacy, whose satisfaction suggests placing computational, control and data resources and facilities close to IoT end-devices [3].

As a class of distributed systems on its own, IoT is quite different. Connected devices are often distributed in space and their environment context is dynamic and composite [4]; it embeds both physical and computational aspects, as devices interact with the physical world. Similarly, software components hosted on IoT devices may belong in different administrative domains or legal jurisdictions. Thus, locality emerges as a key contextual characteristic. Distribution in space raises execution aspects as well, and communication becomes an issue when internet-connected devices rely on remote cloud facilities. From an engineering perspective, the way applications are built differs from traditional systems. Firstly, devices are heterogeneous and typically host software stacks of varying complexity. Their software is also developed and maintained by different teams, by communities which may originate in other engineering disciplines and dominated by their practices, tools and domain knowledge. Finally, despite all such differences, IoT as any other complex software system is increasingly sought to be dependable [1], as applications increasingly require guarantees of meeting their design goals. This poses challenges, as IoT software components are faced with constant disruption – e.g., internal faults may lead to service unavailability, connectivity to cloud control structures may not be persistent, transfer of administrative domains may occur, or the current circumstances a device is found in may be untrusted.

2575-8411/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDCS.2019.00173

The IoT ecosystem is made up of different devices and scales and accordingly is often considered from different viewpoints – from sensor networks to software engineering. We consider IoT devices that range from internet-connected sensors and actuators to powerful cloudlets and gateways deployed close to end-devices. Notice that IoT devices – from microcontrollers to mobile phones and micro-clouds – are made up of software; of course, in various degrees and with varying abilities. Even microscopic sensors expose microservices [5]–[7], while complex computational processing may be performed on mobile devices such as phones or cars [8]. We collectively refer to IoT entities able to host computational, control and data facilities as *edge* components [9]. Edge components may be themselves resource-constrained, low-powered, but able to run software – they are typically deployed at the network boundary and are heavily interacting with other end-devices.

As IoT infrastructures are often deployed in unknown or adverse environments, understanding and systematically managing disruption to the system is key. Disruption is an adverse change to system stability, which fundamentally affects system requirements. An adverse change refers to an external to the system (i.e. due to the environment) or internal to the system (i.e. due to a fault) event, that may lead to violation of its design goal witnessed especially in the heterogeneous, possibly untrusted, highly distributed IoT systems under investigation. In such systems, adverse changes constitute disruptions that may affect resilience. We argue that the key to resilient IoT systems lies in providing engineering support for the paradigm shift from a traditional centralized system design, configuration and operation of coupled cloud-IoT, to decentralized IoT systems. This requires novel methods spanning design and operation of IoT systems. Given the abundance of taxonomies in the wider engineering field [10], our working definition of resilience in IoT systems is drawn from ecological systems [11] – we treat resilience as "the persistence of reliable requirements satisfaction when facing change" [12].

In this paper, we outline a vision for addressing the fundamental challenges that software engineering and distributed systems research encounters when building resilient IoT systems. To this end, we propose a roadmap, mapping key research challenges to bodies of work that have the potential to realize large-scale, resilient IoT. First, from a requirements engineering perspective, we need to characterize resilience; identify its components and constituent properties, and represent them as well as the IoT systems under investigation accordingly in analyzable models. Subsequently, we seek to leverage mechanisms and techniques – particularly from distributed systems and formal software engineering research – of how they can maintain resilience in the face of disruption, especially in the absence of a central control. The role of data is certainly an integral part that must be investigated – particularly how data flows between components that comprise the IoT system. Finally, when the system is operational, resilience must persist – thus, monitoring and validation must occur at runtime, and possible counteractions that the system can perform to mitigate issues must be devised. In essence, we advocate that to achieve resilience, we have to equip edge devices with higher-order reasoning regarding coordination, data management, and runtime adaptive reasoning.

The rest of the paper is structured as follows. Section II outlines the current landscape of software IoT. Section III outlines challenges for resilient IoT, and provides a roadmap. Subsequently, Section IV describes modeling and foundations; Section V illustrates control and coordination as necessary for resilient IoT; Section VI discusses data, and Section VII describes runtime aspects. Section VIII concludes the paper.

II. Software-Defined IoT Systems

The recent evolution toward an increasingly integrated world has at its basis, novel types of large-scale distributed systems achieved via new technologies and paradigms that blend IoT, heterogeneous computing and communication infrastructures, mobile devices, and cloud services. Figure 1 provides a birds-eye view on contemporary software-defined IoT systems; their elements are software components, and each component has its own (possibly heterogeneous) software stack. Often components are in different locations and administrative domains, and their functionality is typically exposed through software services. Devices may range from computationally powerful mobile devices to microcontrollers responsible for sensing or actuation, having minimal software. Typical deployments may involve microscopic sensors as part of a wireless infrastructure feeding data to network gateways, forwarding it for cloud processing, or a computing infrastructure controlling processes through actuators operating upon the physical environment in remote locations. The overall systems are further characterized by mobility, unpredictable human activity, and emergent behaviors. Edge entities able to host computational, control and data facilities run software, are typically situated at the network boundary and heavily interact with other end-devices and resources [7].

To achieve the system's high-level goals, coordination of components lets their functionalities be composed. They may also exchange data to do so, either with centralized cloud facilities or among them. System modeling and validation can enable formal and systematic reasoning of the requirements that the overall system should satisfy. A key concept permeating the IoT ecosystem is that dynamism – of the system's context as well as its software configuration – is omnipresent.

IoT systems provide data, device, and service functionalities that are subject to continuous disruption [13]. Disruption can be external or internal to system stimuli, something witnessed especially in the heterogeneous, possibly untrusted, and highly distributed IoT systems under investigation. In such systems, it is important to factor in potential changes such as transfer of administrative domains, internal faults, connectivity changes, and non-persistent control structures. Specifically, the system should change form to accommodate the external or internal forces while continuing to satisfy its design goals. This raises multiple issues related to reliable system requirements satisfaction.

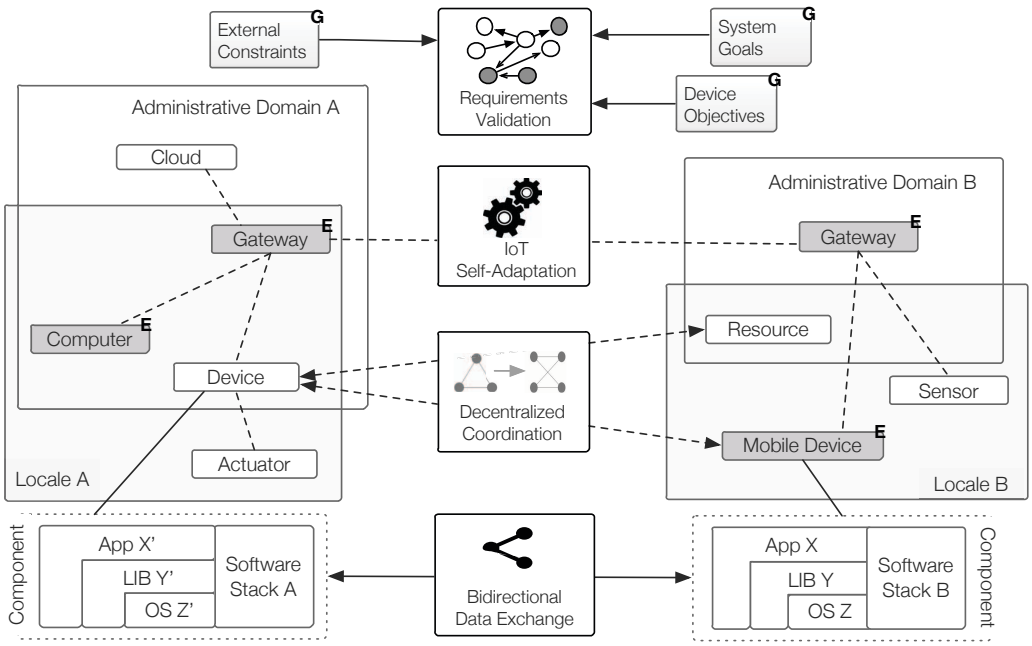

Fig. 1. The current landscape of IoT, comprising of cloud, device and edge entities and IoT resources. To achieve resilience, software components situated on heterogeneous devices, themselves located in different administrative domains or locales, must coordinate in a decentralized manner and exchange data. Requirements reasoning facilities take into account design goals, while self-adaptation at runtime can ensure persistent requirements satisfaction.

III. Resilient IoT: Evolution and Roadmap

As we observed, software-intensive IoT systems provide data, device, and service functionalities that are subject to continuous disruption. Functionality should be persistent under limitations such as resource-constrained nodes, platform heterogeneity, and deployment in adverse environments or unknown administrative domains. In the following, we outline specific challenges for engineering resilience in IoT, as well as illustrate a roadmap.

A. Challenges for Engineering Resilience in IoT

IoT systems' development context extends beyond writing custom business logic components (e.g., services). When developing these systems, researchers must consider (1) the IoT devices' technical specification and configuration details (e.g., their capabilities), and (2) deployment and provisioning of such services across geographically dispersed, heterogeneous IoT infrastructures. The reasons for this include the business logic's complex and strong interdependence on underlying devices (and their specific capabilities), and the novel (resource) features that must be considered, such as device location and IoT cloud resources' heterogeneity.

Satisfying these new requirements in a dependable and resilient manner remains largely under-explored in contemporary scientific literature. This becomes of paramount importance when contemplating systems' surge in development and operations complexity, as we move from simple devices and services to satisfying higher-level design goals. A current setback is that we lack suitable abstractions, analysis techniques, runtime mechanisms, and software engineering best practices to support development and operation.

To overcome this, resilience should become a first-class, native concept in systems throughout the entire IoT software stack. Inevitably, resilience is fundamental in building and operating such systems, as it allows supporting dynamic demands and governing system change, such as varying performance and cost, as well as dealing with system disruption such as failures, governance assurances, and software configuration changes. Although studies have outlined general issues related to a composite system's reliable requirements satisfaction, achieving resilience as a first-class, native "citizen" in IoT presents significant challenges:

1) Addressing the inherent heterogeneity in software stacks, languages, and architecture;
2) Eliminating central points of failure by component coordination, autonomous control and data exchange;
3) Obtaining assurances on reliable requirements satisfaction in an environment that may adversely change at the system's runtime operation; and
4) Addressing mobility and distribution of software components between diverse administrative domains and locales.

B. Resilient IoT Roadmap

In this section, we present our research roadmap towards resilient IoT, which aims at achieving resilience in IoT systems. The main high-level principles which underpin our roadmap include:

- First, we recognize that with the inherently different nature of IoT systems, it is infeasible to adopt traditional resilience mechanisms (e.g., as for fault tolerance) and tailor them by making small and incremental changes.
- Second, we recognize that the required paradigm shift cannot happen in a one-dimensional world view–instead, it requires disrupting multiple relevant research fields along several so-called disruption vectors.
- Third, we argue that resilience must be built into core IoT component mechanisms natively, as opposed to the traditional view of adding (self-contained) resilience mechanisms (e.g., circuit breakers) to applications.
- Finally, we submit that the most groundbreaking results will emerge as a combined effect of individual advancements along the aforementioned disruption vectors.

Table 1 compares state-of-the-art research (e.g., in fog [14] and edge [9] computing) and our vision of resilient IoT. There is currently a wide gap between the current IoT systems and resilient IoT systems of the future, considering our disruption vectors. For each disruption vector, we identify four main evolutionary steps, i.e., maturity levels (MLs) towards resilient IoT systems:

(ML1) Traditional vertically coupled IoT systems;
(ML2) Hybrid IoT-Cloud systems;
(ML3) Edge-centric systems; and
(ML4) Resilient IoT systems.

Next, we briefly discuss the main disruption vectors. From a pervasiveness perspective, to facilitate openness in IoT infrastructure and enable applications to consume IoT resources uniformly as a full-fledged utility we need novel abstractions and infrastructure virtualization approaches rooted in a formal representation and treatment of resource capabilities [15], [16]. To reduce the coupling between applications and infrastructure capabilities and to eliminate the need for manual service management, we advocate a deviceless paradigm [17], [18]. This entails engineering a new set of techniques for service orchestration and scheduling, as well as a set of resilience mechanisms intrinsically built into IoT systems' cores. For validation of requirements of both infrastructure and application logic, we advocate that IoT systems need formally analyzable and verifiable models to enable reasoning, starting from the early stages of design to models@runtime. Regarding automation in operations techniques research in self-adaptive systems must be brought in this new domain. In particular operations and management processes [19], by employing novel management and control techniques. Finally, enabling unconstrained inter-IoT communication and data flows is critical; this can be achieved by developing fundamental mechanisms and methodologies for data governance (on both the data and control planes) among administrative domains and different levels of trust [20], [21].

Based on this roadmap, we identify four main research directions that we urge the community to investigate – those are crucial in achieving the highest maturity levels of our disruption vectors (Table 2). We consider latency, heterogeneity, and locality as particularly pertinent to resilience in IoT. Achieving resilience entails system awareness through monitoring and analysis, along with operating controllable components. A system must have provisions to deal with change in those key factors–otherwise, its requirements satisfaction in the face of change may not persist. To realize this approach, we advocate research directions that represent distinct scientific research topics, yet whose combination could maximize impact in achieving resilient systems:

1) Conceptual foundations and representations to address the need for defining modeling and formal aspects related to systematically engineering resilient IoT, as well as relevant requirements concepts such as domain knowledge, goals, context and scope (Section IV);
2) Coordination between system components is crucial, leveraging current distributed systems research for software engineering and addressing the non-central decision-making theme that emerges as a key resilience attribute (Section V);
3) Data as it is stored, processed, and transmitted in and between components (Section VI); and
4) Operationalization, where runtime aspects and unforeseen as well as emergent system behaviors that might hinder resilience must be monitored, and potential counteractions must be devised in a self-adaptive manner (Section VII).

Such concepts manifest themselves in different scientific communities within computer science and engineering, working in diverse subfields such as software engineering or distributed systems. However, the common denominator is that the paradigm shift to resilient IoT is theoretically and methodologically under-investigated. The following sections discuss the fundamental topics identified by our roadmap.

IV. Modeling and Foundations for Resilient IoT

Reasoning about resilience requires first its precise characterization, representation, and appropriate abstractions that can enable systematic reasoning. After contextualizing resilience as persistence of an IoT system's requirements satisfaction when faced with change, we argue that analyzable representations of certain system aspects are necessary to enable systematic reasoning.

A. Persistence of Requirements Satisfaction in IoT

Resilience in computing is the persistence of dependability when facing change [10], and is understood as the ability of a system to handle disruptions and variations that fall outside the defined base mechanisms for being adaptive as defined in that system [22]. Generally, in computer science, it has been used in lieu of, and related to dependability, fault tolerance, or robustness, especially within critical systems engineering [12]. Resilience concepts have been investigated for decades in fields ranging from critical and dependable systems [23], embedded and cyber-physical systems [24], [25], networks [26]–[28], security [29], [30], and cloud computing [31]. Engineering support similarly ranges from theoretical foundations [32], operations and process [33], [34], formally backed system validation [26], [35], quality control [36], and fault management [37]–[39].

Disruption	Infrastructure (degree of openness)	App. execution models (degree of coupling & automated management)	Dependable engineering (degree of requirements assurance)	Operations techniques (degree of automation)	Data flows (degree of governed data permeation)
ML1	IoT silos – vertically closed and task-specific IoT infrastructure	Business logic bundled and shipped with IoT devices	Ad hoc requirements with little to no validation	Exclusively manual interactions with on-site presence	Proprietary and task-specific communication protocols. Isolated data flows
ML2	Cloud-based platforms for brokering IoT data	Services are decoupled, with a hard line between the IoT and cloud responsibilities (business logic)	Limited verification. Parts of the system offer service-level agreements	Partly automated operations processes, mainly on the Cloud side	Unidirectional data flows, with no explicit support for data governance
ML3	Common access to specific types of resources (gateways, cloudlets, and microclouds)	Some shared services exist. Services are partly managed	Task-specific formal verification possible	Full automation of specific tasks. Manual interactions still needed, but mainly handled remotely	Bidirectional (Edge-Cloud) data flows. Data governance limited to specific domains

Table 1: Current state of the art within engineering IoT systems.

ML4	Edge infrastructure consumed as a full-fledged utility	Deviceless – business logic fully managed and abstracted from the infrastructure capabilities	Formally verifiable requirements of both infrastructure and application logic	Autonomous control, coordination and self-healing	Unconstrained data flows. Governance among administrative domains & trust levels

Table 2: Future directions for engineering resilient IoT systems.

The challenge here is that the key taxonomical subconcepts of resilience must be understood in terms of the IoT – such concepts are required to be brought into the way we design and operate contemporary IoT systems. We consider the factors of locality (e.g., in physical or computational domains), latency (e.g., in network connectivity), and heterogeneity (e.g. in device or software stacks) as highly pertinent to reliable requirements satisfaction in IoT. As such, the development of usable abstractions and analyzable models should be geared to reason on those factors.

B. Analyzable Models for Resilient IoT

This future challenge expresses the need for defining modeling and representation aspects of IoT systems, which demand conceptual frameworks and definitions to capture key concerns; the environment and its uncertainty, software configurations, and system objectives and overall design goals. Then, mathematically backed formal methods can enable systematic engineering. Modeling the environment and software configurations of IoT systems as well as their dynamics (to enable formal reasoning about system properties) is a crucial prerequisite for engineering systems that reliably satisfy requirements and render them resilient. Thus, modeling is not merely a representation, but a foundation for both design-time analysis of resilience factors and resilient system operationalization. In essence, the factors that make IoT different must be understood, captured and analyzed in a systematic way.

As we observed within IoT, system-wide requirements may state desired collective behavior – however, devices themselves, as they operate within different administrative domains and dynamic environments may have possibly conflicting goals [40]. Domain knowledge may include environment uncertainty – hindering precise reasoning – while users introduce variability. Thus, requirements methods (e.g. goal modeling and validation [41], [42]) can be applied in novel ways. Similarly, the organization of components into software architectures must not be understood as static, as devices may be updated by vendors, their interfaces changed and thus their overall software configuration altered. Subsequently, such model representations [43] –e.g. of environments, configurations and requirements – can be analyzed. What we advocate here, is that for IoT to become resilient, such aspects must be systematically treated, especially in the ways that traditional, established methods did not foresee.

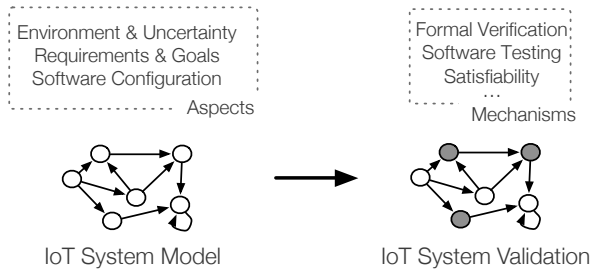

Fig. 2. IoT models as foundations for formal reasoning.

As a general methodology from a requirements engineering perspective, the initial steps are to (i) characterize resilience, identify its components and constituent properties and represent them; and (ii) capture representations of the IoT systems under investigation in analyzable models. Representations, in the form of *views*, can enable the assessment of a system's resilience attributes [44], [45]. Then, (iii) based on the de-

fined resilience properties and system model representation, this is a classical system validation and verification problem (Figure IV). The verification process checks whether a given system (i.e., a facet of an IoT system model) satisfies a given correctness specification (i.e., resilience properties). The challenge here is to leverage formal methods and verification research from traditional software engineering and theoretical computer science to this new domain, developing a modeling discipline and methodology that captures and analyses resilience (i.e. through its various attributes). We foresee formal aspects of software engineering to be leveraged, including formal logics, computational models, and stochastic processes or uncertainty quantification techniques.

In the engineering sense, this paves the way to adopt reasoning tools developed by the software engineering community, such as requirements reasoning with qualitative and quantitative model checking techniques including statistical, systematic testing or satisfiability. This future challenge is the foundational precursor of the others, as representation and requirements validation are critical for engineering resilience in every aspect. For example, validating distributed protocols is fundamental for coordination and composition, and it requires appropriate system representations; timeliness, availability and privacy data characteristics needed for reliable inter-IoT data exchanges can be expressed as quantitative logical properties; verification and planning facilities needed to assess and ensure resilience at runtime in a changing environment are naturally a port to runtime of design time representations, enriched with validation techniques suitable for system operation.

V. Decentralized Coordination

IoT applications can be of various types, software stacks, and complexities, with multiple system components deployed in diverse domains and contexts. Those, however, do not live in isolation and must be able to coordinate to fulfill application requirements [46]. In the following, we seek to leverage mechanisms and techniques – particularly from distributed systems and formal software engineering research – of how systems can resiliently coordinate in the face of disruption, especially in the absence of a central control.

A. Control and Coordination

Recall that IoT applications need to operate on diverse infrastructures and integrate heterogeneous components from various providers in a long-running system, with possibly conflicting goals between components. Software-based control here entails actively setting in motion configuration changes to satisfy system objectives. Centralizing control – typically in the cloud and evident in today's IoT-cloud architectures – partly mitigates such problems, but requires cloud control structures to be always available, secure, and fault tolerant (including other aspects such as within low latency). This has been driven by necessity, as IoT may be comprised of sensor devices whose operational program structures are minimal.

Coordination and control play a key role in several current research streams, where various research communities are attempting to provide theoretical foundations and practical frameworks for the development of dynamically adaptive systems. From a networking perspective, for instance, networked systems capable of autonomous changes in topology, load, and physical and logical network characteristics have been developed [47]. The intelligent agent, machine learning, and planning communities have also had an abiding interest in autonomous systems. Self-adaptive systems are capturing interest in the wider software engineering community; initial research in self-adaptive software architectures focused on replaceable components and connectors [48]. Other approaches consider all possible levels of abstraction: from the architectural level down to language primitives, and from requirements analysis and validation to runtime system operation. On a higher level of abstraction, many techniques support adaptation or self-healing at the component or system level [49], [50].

For resilient IoT, coordination presupposes a general absence of centralized control [51], instead leveraging cooperation between software components, in a peer-to-peer fashion. Generally, the state of the art in IoT systems usually adopts centralized coordination techniques [52]–[54], adhering to the device-cloud archetype. We advocate both distributed computing infrastructures and decentralized management policies. The attributes of locality, variability in the environment, and distributed computation in IoT systems (manifested as uncertainty at runtime) hinder coordination and control. Similar attributes have been investigated as uncertainties [55], [56] and can provide the basis to analyze them for resilience in highly distributed, widely deployed IoT systems. For example, one taxonomy [55] classifies types of uncertainties by the place where they manifest, their uncertainty level, and their nature – i.e., whether the uncertainty is because of imperfect knowledge or variability [51]. Research on self-adaptive systems has tackled such attributes, albeit in a different context, by managing uncertainty at runtime and considering both functional and non-functional requirements [49]. Information sharing patterns where each entity self-adapts locally by implementing its own MAPE-K loop – using information from other entities in the system [57] – is a characteristic self-adaptive view. Moreover, requirements engineering proposed approaches that use requirements management to handle uncertainty at runtime, which may be used for characterizing and modeling resilience factors.

B. Decentralization for Resilience

Going forward, resilient IoT systems must correctly fulfill requirements in changing, unpredictable, and potentially adversarial environments where availability of a central point of control cannot be guaranteed. We argue that because IoT distributed systems are made up of software components, facilities providing control and coordination must be performed at the software components' level. It is therefore imperative that the control facilities of such systems become grounded in distributed systems research, ensuring that IoT systems can autonomously react to changes in different contexts derived from changes in environment, software configuration, execution infrastructure, or other unforeseen issues in a resilient manner. Essentially, our community needs to address the

problems created from the move of IoT software systems to decentralization, something imposed by (i) new requirements emerging from the increasing need of distributing computation to heterogeneous software components, and (ii) the prevalence of internet-enabled devices, which call for additional management and control closer to their operating architectural layer, so that central points of failure are eliminated. We view decentralization as the key to achieving resilience in the face of the uncertainty and variability that IoT systems are exposed to.

Going from centralized, traditional systems design, configuration, and operation of IoT-Cloud systems to decentralized IoT systems is a paradigm shift – to another technology maturity level (ML4 in Table 2). Parts of the control logic situated on the Cloud now must move closer to end-devices, situated at the network edge. This requires edge devices to decentralize operations, but raises complex issues associated with distributed systems [58]. To this end, understanding and systematically managing change in is key: change may refer to software configuration, environment, execution context or resources.

Certainly, control in the large-scale IoT systems we investigate aims to achieve requirements satisfaction – autonomously – in a changing environment. Thus, research on formal aspects of self-adaptive systems can be leveraged, especially as relating to model-based planning and self-healing on a system level using contextual information. However, this can also diverge from major research directions in self-adaptive systems, because planning may be required to be performed in a distributed fashion. To this end, distributed systems mechanisms relevant to process coordination and control can be adopted. System models, kept at runtime, can facilitate coordinating and determining how control actions affect system resiliency.

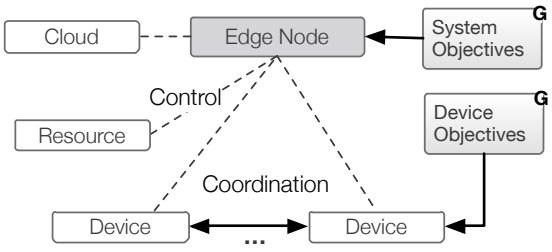

Fig. 3. Decentralized coordination and control performed at the IoT's edge.

Regarding architectural deployment, we envision the edge acting as a manifestation of a control agent responsible for observing and evaluating contextual information, and inducing appropriate actions for its controlled subsystem. In Figure 3, an edge entity takes into account device objectives as well as ones of the overall system, providing control and coordination facilities to participating devices within its scope. Coordination may extend to cloud entities or other edge nodes, in a decentralized manner. The attainability of the edge paradigm has been shown in several examples emerged over the past decade; "cloudlets" that offer locally cloud services, "crowdsensing" as collaborative devices sensing the environment, "edge analytics" leveraging stream operations before reaching remote storage, or "edge networking" for overlay control of wireless networks. In a broader sense, coordination is the underlying theme, and distributed systems research is required to support such applications. Situating coordination facilities on edge components eliminates central points of failure and leads to decentralization.

VI. Inter-IoT Data Flows

A system of connected devices that collect, send, and act upon information from various surrounding sources certainly makes data management an integral part of the IoT paradigm. Composite applications such as data analytics are built on IoT data foundations, and data provenance has been widely researched within the IoT at various abstraction levels, ranging from wireless sensor networks to databases, covering aspects of the input sources, programs, and humans involved.

A. Data Management within IoT

However, research has predominantly focused on tightly coupled IoT-Cloud systems; information such as sensor readings flow from devices to cloud storage and processing. As IoT becomes increasingly made up of software and decentralization emerges as a crucial property, IoT software components not only produce, but transmit, share, and act upon data. Now data flows from the device to device in a bidirectional manner, and among different data consumers and producers [59]. A further challenge is that data often traverses through computational resources of diverse administrative domains and different levels of trust, raising issues in reliable satisfaction of privacy, timeliness, and availability requirements.

The sheer number of heterogeneous devices, software stacks, and processes involved in IoT make it challenging to achieve applications' data goals [60]. Moreover, novel requirements dictate data to reside and be processed close to where it is produced, for several emerging reasons. Traditional tightly coupled IoT-Cloud computing cannot meet especially stringent requirements about latency [23], privacy [61], or geographical restrictions of data. Privacy [62], [63] for instance, may require not only data within a IoT application to remain locally close to where it is sourced, but also for all mechanisms managing it to respect different legal or administrative frameworks (e.g., the EU General Data Protection Regulation [64] versus the California Consumer Privacy Act [65]) and user preferences. Enabling the application to operate within diverse domains points to another facet of resilience, where data producers and consumers require control over data exchanges. Several research directions in the distributed systems community have advocated that because the edge is closer to data sources and users, there is not only obvious latency advantages but also an opportunity for stronger privacy and operation within administrative domains by building appropriate data handling logic inside software components.

B. Data Flows among IoT Software Components

Engineering decentralized, data-driven applications where information is managed and exchanged by software components raises issues of availability, timeliness, and privacy. Such issues as they pertain to IoT have not been previously investigated by the community, because (i) IoT-Cloud coupling was considered a basic assumption, (ii) IoT devices were largely considered producers of data (i.e., in sensing applications), and (iii) data processing within IoT entities was rarely an option (because of minimal storage or limited computational facilities). However, devices become increasingly able to host software stacks and emerging requirements restrict centralized cloud storage and processing. As a community, we need to leverage research from overlapping areas of distributed systems and databases for the IoT paradigm. The main idea is that instead of arbitrary networked processes, the particularities of IoT software components require novel applications of data synchronization, network storage, messaging [58] and their supporting distributed protocols in a decentralized manner.

One way to achieve this, is to methodologically follow the data lineage within IoT– data's origins, what happens to it and where it moves over time, and providing mechanisms for resilient data governance. Furthermore, how availability and timeliness of data relate to privacy and decision-making goals must be addressed, as those are particularly pertinent to the applications under investigation; we note an absence of a systematic treatment within the requirements engineering of IoT data goals, especially regarding privacy. This paves the way for employing distributed systems techniques to implement support in software IoT components, based on metrics and evaluating mechanisms for data management that the database community actively investigates.

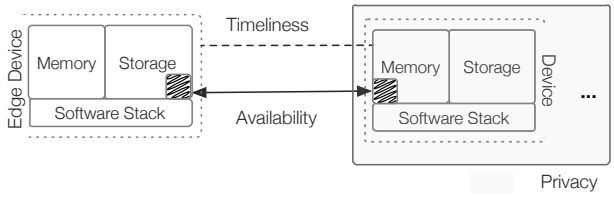

Fig. 4. Inter-IoT data flows highlighting privacy, timeliness and availability.

Figure 4 illustrates data-handling software components; data may need to be kept synchronized or transfered among them. This must occur in a timely manner and with certain availability requirements. Moreover, privacy needs to be respected, as certain data can be sensitive. Sensitive data-producing devices can be in privacy *scopes*, defined by particular legal jurisdictions (e.g. EU GDPR) or end-user privacy preferences. Privacy requirements in this case, dictate what data should leave (or enter, e.g. to modify) a component, and each component must have control of its own data out- or in-flow privacy policies (e.g. that govern data synchronizations) [66]. Note that in the case of resource-constrained IoT deployments, the edge paradigm similarly applies. Situating data storage, management and computational facilities in edge devices entails making those available to resource-constrained devices that may lack them, while the edge is often located within the privacy domain of its local IoT devices. Thus, the edge can manage a local privacy scope, by ensuring privacy requirements. For example, a user's mobile phone as an edge device, can enforce privacy preferences on data generated from her wearable IoT devices.

VII. Runtime IoT Self-Adaptation

As we observed, unforeseen changes in the software configuration or the environment within an IoT system operates, may hinder overall system resilience. This future challenge addresses (i) continuous monitoring of IoT systems for checking the conformance of their behavior with respect to requirements and for discovering unforeseen or emergent behaviors that might hinder their resilience, as well as (ii) planning appropriate counteractions that can maintain requirements satisfaction. The system must continuously work with little or no disruption, despite exogenous changes in component context. The challenge arises due to dynamic behavior – design-time assumptions about the environment or configuration might not hold at runtime, and behaviors (unforeseen during design) might arise.

A. IoT as a Self-adaptive System

Maintaining a software system that operates in a dynamic environment faces the manifold challenges of software system evolution [67], and demands operational management to observe a constantly changing context and potentially react to changes. Addressing the problem of continuously monitoring [68] IoT systems requires extending (i.e., from foundations as per Section IV) modeling and validation to runtime and the further definition of novel techniques to maintain system knowledge and its contextual characteristics. If the system is found to violate its resilience objectives, counteractions must be devised and actuated; to this end, techniques from self-adaptive systems engineering can be incorporated [69]. Typically, this can be achieved through an autonomic, self-adaptive approach – e.g., a MAPE loop [70]: (M)onitoring the environment for changes which are reflected in a model, (A)nalyzing the model for possible requirements violations, (P)lanning required countermeasures and then (E)xecuting the appropriate actions and updating the model for the next loop. Research in self-adaptive systems has long considered such issues, however they must be brought into the IoT context [71].

IoT encompasses computing and communication capabilities embedded into multiple environments – a device may be found in a physical space, its software component part of an overall infrastructure and logically connected to others. Hence, locality emerges as a key contextual characteristic, and a view of the system's environment [4] as a composite model can be the foundation for model-based analysis and planning. The system model can encompass descriptions of the environment, the software configuration and the requirements (Section IV). To enable analysis, the model can then be translated into formal abstractions enjoying well-defined semantics. Formalization aims to encode the model into a form that facilitates automated reasoning; different analyzable

models may be automatically generated to support different kinds of analyses [72], [73], including quantitative or statistical validation [74]–[76].

B. Runtime Validation and Planning at the Edge

As we observed in Section IV, the extent of assurances obtained from analyses of system requirements depends on whether they address concerns that arise at design time or runtime. When contemplating design time analysis, the objective is that the IoT system must satisfy requirements "by design"; disruption that may occur when the system is deployed cannot harm resilience, given that certain assumptions are met. However, certain resilience aspects may not be guaranteed at design time. Instead, resilience must be achieved at runtime by generating adaptive actions that can prevent the IoT system from violating stated resilience goals. What we advocate, is a self-adaptive systems view; a composite model of the environment must be kept alive at runtime [77] and populated with information as they become available. This model is then analyzed with respect to factors critical for system resilience [78], and if it is found to violate them, counteractions are devised [79], [80].

Bringing self-* properties to the IoT involves an increased focus on their operational aspects. As we observed in Section II, IoT has distinctive characteristics. Software stacks are heterogeneous but generally more limited than traditional software systems, as IoT is comprised of resource-constrained devices. As such, computationally-intensive analyses cannot be performed there. Moreover, devices are often spatially distributed [81] in a composite environment that changes [4]. Analysis must also take into account the rate of change of the environment. Similarly, actuation of countermeasures [82] to satisfy requirements must be performed in accordance to constraints imposed by the application domain and the composite cyber-physical environment.

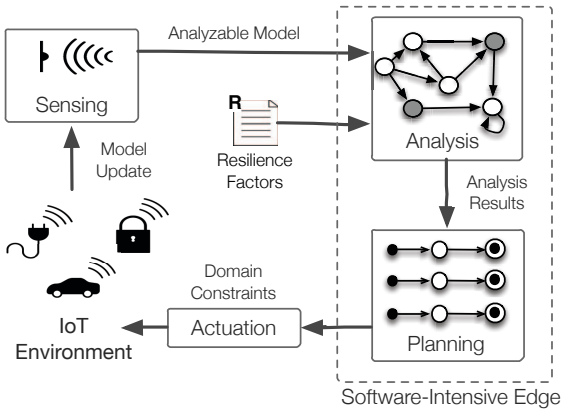

Fig. 5. Analysis and planning situated at IoT edge components at runtime.

Figure 5 captures activities within a MAPE loop for IoT systems, where monitoring and execution may be referred to as sensing and actuation, as they are dominant in the IoT end-devices. The above distinctive IoT characteristics suggest placing analysis and planning activities on *edge* components – close to end-devices and responsible for their management within a certain local scope. Limiting scope to a certain extent is in accordance to the domain – deploying an edge entity within a local wireless network or administrative domain can render its resources available to local resource-constrained devices and be responsible for evaluating resilience factors (i.e. analysis) and for their overall coordination and counteraction construction (i.e. planning). Note that this adheres to the decentralization and coordination theme discussed in Section V as well. IoT's consideration as a runtime, self-adaptive system must also address facets that often do not appear in traditional adaptive software systems. As devices are often deployed in wide physical spaces, the spatial aspect (and how locality affects the system) is significant [4]. Similarly, their IoT's highly distributed nature allows emergent, unforeseen behaviors to occur. Centralized control is often unfeasible in practice, not only because it presents a central point of failure, but also because network connections may not be persistent. This points to analysis and planning to be performed close to the end-device level, and perhaps on edge entities which may not enjoy the vast resources of cloud computing.

VIII. An Emerging Research Agenda

The novel types of large-scale distributed systems that have emerged with the prevalence of the IoT, bring together heterogeneous computing and communication infrastructures, mobile devices, and cloud services. IoT systems provide data-centric, device-centric and service-centric functionalities that are subject to continuous disruption, under limitations such as resource-constrained devices, platforms heterogeneity, deployment in adverse environments and administrative domains. As these systems evolve and gain complexity, resilience becomes crucial; to achieve it, understanding and systematically managing dynamic behavior and decentralizing operations are key.

We advocated that to systematically engineer resilience in IoT systems, a complete rethink is needed regarding design and operation. We outlined a vision for addressing fundamental challenges for building resilient IoT, and presented a roadmap where we identified techniques and methods from formal aspects of software engineering as well as distributed systems that can be leveraged to foster resilience in the face of disruption, especially in the absence of centralization and persistently at the system's runtime.

Acknowledgment

Research partially supported by the Research Cluster Smart Communities and Technologies (SmartCT) at TU Wien.

References

[1] Algirdas Avizienis, J-C Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[2] Maarten van Steen, Guillaume Pierre, and Spyros Voulgaris. Challenges in very large distributed systems. *Journal of Internet Services and Applications*, 3(1):59–66, 2012.

[3] Ju Ren, Hui Guo, Chugui Xu, and Yaoxue Zhang. Serving at the edge: A scalable iot architecture based on transparent computing. *IEEE Network*, 31(5):96–105, 2017.

[4] Christos Tsigkanos, Timo Kehrer, and Carlo Ghezzi. Modeling and verification of evolving cyber-physical spaces. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017, 2017*, pages 38–48, 2017.

[5] Athman Bouguettaya, Munindar Singh, Michael Huhns, Quan Z Sheng, Hai Dong, Qi Yu, Azadeh Ghari Neiat, Sajib Mistry, Boualem Benatallah, Brahim Medjahed, et al. A service computing manifesto: the next 10 years. *Communications of the ACM*, 60(4):64–72, 2017.

[6] Dominique Guinard, Vlad Trifa, Stamatis Karnouskos, Patrik Spiess, and Domnic Savio. Interacting with the soa-based internet of things: Discovery, query, selection, and on-demand provisioning of web services. *IEEE transactions on Services Computing*, (3):223–235, 2010.

[7] Xiongnan Jin, Sejin Chun, Jooik Jung, and Kyong-Ho Lee. Iot service selection based on physical service model and absolute dominance relationship. In *Service-Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on*, pages 65–72. IEEE, 2014.

[8] Maria Laura Stefanizzi, Luca Mottola, Luca Mainetti, and Luigi Patrono. COIN: opening the internet of things to people's mobile devices. *IEEE Communications Magazine*, 55(2):20–26, 2017.

[9] Weisong Shi and Schahram Dustdar. The promise of edge computing. *IEEE Computer*, 49(5):78–81, 2016.

[10] Jean-Claude Laprie. *From dependability to resilience*. In 38th IEEE/IFIP Int. Conf. On Dependable Systems and Networks, pages G8–G9. Citeseer, 2008.

[11] Crawford S Holling. Resilience and stability of ecological systems. *Annual review of ecology and systematics*, 4(1):1–23, 1973.

[12] Algirdas Avižienis. A visit to the jungle of terminology. *In Dependable Systems and Networks Workshop (DSN-W)*, 2017(47):149–152, 2017.

[13] Miruna Stoicescu, Jean-Charles Fabre, and Matthieu Roy. Architecting resilient computing systems: overall approach and open issues. In *International Workshop on Software Engineering for Resilient Systems*, pages 48–62. Springer, 2011.

[14] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[15] Stefan Nastic, Sanjin Sehic, Duc-Hung Le, Hong-Linh Truong, and Schahram Dustdar. Provisioning software-defined iot cloud systems. In *2014 2nd International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 288–295. IEEE, 2014.

[16] Stefan Nastic, Hong-Linh Truong, and Schahram Dustdar. Sdg-pro: a programming framework for software-defined iot cloud gateways. *Journal of Internet Services and Applications*, 6(1):21, 2015.

[17] Alex Glikson, Stefan Nastic, and Schahram Dustdar. Deviceless edge computing: extending serverless computing to the edge of the network. In *Proceedings of the 10th ACM International Systems and Storage Conference*, page 28. ACM, 2017.

[18] Stefan Nastic, Thomas Rausch, Ognjen Scekic, Schahram Dustdar, Marjan Gusev, Bojana Koteska, Magdalena Kostoska, Boro Jakimovski, Sasko Ristov, and Radu Prodan. A serverless real-time data analytics platform for edge computing. *IEEE Internet Computing*, 21(4):64–71, 2017.

[19] Stefan Nastic, Hong-Linh Truong, and Schahram Dustdar. A middleware infrastructure for utility-based provisioning of iot cloud systems. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 28–40. IEEE, 2016.

[20] Stefan Nastic, Christian Inzinger, Hong-Linh Truong, and Schahram Dustdar. Govops: The missing link for governance in software-defined iot cloud systems. In *Service-Oriented Computing-ICSOC 2014 Workshops*, pages 20–31. Springer, 2015.

[21] Stefan Nastic, Michael Vögler, Christian Inzinger, Hong-Linh Truong, and Schahram Dustdar. rtgovops: A runtime framework for governance in large-scale software-defined iot cloud systems. In *Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2015 3rd IEEE International Conference on*, pages 24–33. IEEE, 2015.

[22] Erik Hollnagel, David D. Woods, and Nancy Leveson. *Resilience engineering: Concepts and precepts*. Ashgate Publishing, Ltd, 2007.

[23] Philipp Schulz, Maximilian Matthe, Henrik Klessig, Meryem Simsek, Gerhard Fettweis, Junaid Ansari, Shehzad Ali Ashraf, Bjoern Almeroth, Jens Voigt, Ines Riedel, et al. Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture. *IEEE Communications Magazine*, 55(2):70–78, 2017.

[24] Grit Denker, Nikil Dutt, Sharad Mehrotra, Mark-Oliver Stehr, Carolyn Talcott, and Nalini Venkatasubramanian. Resilient dependable cyber-physical systems: a middleware perspective. *Journal of Internet Services and Applications*, 3(1):41–49, 2012.

[25] Jean Arlat, Michel Diaz, and Mohamed Kaâniche. Towards resilient cyber-physical systems: The adream project. *In Design and Technology of Integrated Systems In Nanoscale Era (DTIS)*, 2014(9):1–5, 2014.

[26] John Rushby. Formal methods and their role in the certification of critical systems. In *Safety and reliability of software based systems*, pages 1–42. 1997.

[27] Nalini Venkatasubramanian, Carolyn Talcott, and Gul A. Agha. A formal model for reasoning about adaptive qos-enabled middleware. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 13(1):86–147, 2004.

[28] Marcello Cinque, Antonio Coronato, Alessandro Testa, and Catello Di Martino. A survey on resiliency assessment techniques for wireless sensor networks. In *Proceedings of the 11th ACM international symposium on Mobility management and wireless access*, pages 73–80. ACM, 2013.

[29] Yanchao Zhang and Yuguang Fang. Arsa: An attack-resilient security architecture for multihop wireless mesh networks. *IEEE Journal on Selected areas in communications*, 24(10):1916–1928, 2006.

[30] Debra S. Herrmann. *Complete guide to security and privacy metrics: measuring regulatory compliance, operational resilience, and ROI*. Auerbach Publications, 2007.

[31] Rahul Ghosh, Francesco Longo, Vijay K. Naik, and Kishor S. Trivedi. Quantifying resiliency of iaas cloud. *In Reliable Distributed Systems*, 2010(29):343–347, 2010.

[32] Miruna Stoicescu, Jean-Charles Fabre, and Matthieu Roy. Architecting resilient computing systems: overall approach and open issues. In *International Workshop on Software Engineering for Resilient Systems*, pages 48–62. 2011.

[33] Simona Bernardi, José Merseguer, and Dorina C. Petriu. Dependability modeling and analysis of software systems specified with uml. *ACM Computing Surveys (CSUR)*, 45:1, 2012.

[34] John F. Meyer. Model-based evaluation of system resilience. *In Dependable Systems and Networks Workshop (DSN-W)*, 2013(43):1–7, 2013.

[35] Christel Baier, Clemens Dubslaff, Sascha Kl"uppelholz, and Linda Leuschner. Energy-utility analysis for resilient systems using probabilistic model checking. In *International Conference on Applications and Theory of Petri Nets and Concurrency*, pages 20–39, 2014.

[36] Vincenzo De Florio. Quality indicators for collective systems resilience. 2014. arxiv. preprint.

[37] Dimiter Avresky, Jean Arlat, J c Laprie, and Yves Crouzet. Fault injection for formal testing of fault tolerance. *IEEE Transactions on Reliability*, 45(3):443–455, 1996.

[38] Flavin Cristian. Understanding fault-tolerant distributed systems. *Communications of the ACM*, 34(2):56–78, 1991.

[39] Anjali Bhargava and Bharat K. Bhargava. Applying fault-tolerance principles to security research. In *20th Symposium on Reliable Distributed Systems (SRDS 2001)*, pages 28–31, New Orleans, LA, USA, pages 68–69, 2001. October 2001.

[40] Antoine Cailliau and Axel van Lamsweerde. Runtime monitoring and resolution of probabilistic obstacles to system goals. In *Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2017 IEEE/ACM 12th International Symposium on*, pages 1–11. IEEE, 2017.

[41] A. van Lamsweerde. *Requirements engineering - from system goals to UML models to software specification*. Wiley, 2009.

[42] B. Cheng, P. Sawyer, N. Bencomo, and J. Whittle. A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In *Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems (MoDELS'09)*, pages 468–483, 2009.

[43] Xing Chen, Aipeng Li, Xue'e Zeng, Wenzhong Guo, and Gang Huang. Runtime model based approach to iot application development. *Frontiers of Computer Science*, 9(4):540–553, 2015.

[44] Bashar Nuseibeh, Jeff Kramer, and Anthony Finkelstein. A framework for expressing the relationships between multiple views in requirements specification. *IEEE Trans. Software Eng.*, 20(10):760–773, 1994.

[45] Anthony Finkelstein, Jeff Kramer, Bashar Nuseibeh, Ludwik Finkelstein, and Michael Goedicke. Viewpoints: A framework for integrating multiple perspectives in system development. *International Journal of Software Engineering and Knowledge Engineering*, 2(01):31–57, 1992.

[46] Victor R Lesser and Daniel D Corkill. Functionally accurate, cooperative distributed systems. In *Readings in Distributed Artificial Intelligence*, pages 295–310. Elsevier, 1988.

[47] Michael Stein, Alexander Fr"ommgen, Roland Kluge, Frank L"offler, Andy Sch"urr, Alejandro Buchmann, and Max M"uhlh"auser. Tarl: modeling topology adaptations for networking applications. In *16). ACM, New York, NY, USA*, pages 57–63. Proceedings of the 11th

International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS, 2016.

[48] G. Coulson, G. S. Blair, P. Grace, A. Joolia, K. Lee, J. Ueyama, and T.v Sivaharan. *A generic component model for building systems software*. ACM Transactions on Computer Systems, 2008.

[49] B. Cheng, R. De Lemos, H. Giese, et al. Software engineering for self-adaptive systems: A research roadmap. In *Software engineering for self-adaptive systems*, pages 1–26. Springer, 2009.

[50] Jeff Kramer and Jeff Magee. Self-managed systems: an architectural challenge. *In FOSE '0*, 7(2007):259–268, 2007.

[51] Mirko D'Angelo. Decentralized self-adaptive computing at the edge. In *18). ACM, New York, NY, USA*, pages 144–148. Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems (SEAMS, 2018.

[52] D. P. Anderson. Boinc: A system for public-resource computing and storage. In *04, , Washington, DC, USA IEEE Computer Society*, pages 4–10. Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, GRID, 2004.

[53] P. Mayer, A. Klarl, R. Hennicker, F. Tiezzi M92Puviani, R. Pugliese, J. Keznikl, and T. Bure. The autonomic cloud: A vision of voluntary. In *Peer-2-Peer Cloud Computing*, pages 89–94. In 2013 IEEE 7th International Conference on Self-Adaptation and Self-Organizing Systems Workshops, 2013.

[54] Fahed Alkhabbas, Romina Spalazzese, and Paul Davidsson. Eco-iot: an architectural approach for realizing emergent configurations in the internet of things. In *European Conference on Software Architecture*, pages 86–102. Springer, 2018.

[55] D. Perez-Palacin and R. Mirandola. Uncertainties in the modeling of self-adaptive systems: A taxonomy and an example of availability evaluation. In *14, , New York, NY, USA ACM*, pages 3–14, ICPE, 2014. Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering.

[56] N. Esfahani and S. Malek. Uncertainty in self-adaptive software systems. In Heidelberg Berlin, editor, . *Springer Berlin Heidelberg*, pages 214–238. 2013.

[57] D. Weyns et al. On patterns for decentralized control in self-adaptive systems. *Lecture Notes in Computer Science*, 7475:76–107, 2013.

[58] Thomas Rausch, Schahram Dustdar, and Rajiv Ranjan. Osmotic message-oriented middleware for the internet of things. *IEEE Cloud Computing*, 5(2):17–25, 2018.

[59] Keiichi Yasumoto, Hirozumi Yamaguchi, and Hiroshi Shigeno. Survey of real-time processing technologies of iot data streams. *Journal of Information Processing*, 24(2):195–202, 2016.

[60] Nam Ky Giang, Michael Blackstock, Rodger Lea, and Victor C. M. Leung. Developing iot applications in the fog: a distributed dataflow approach. In *Internet of Things (IOT)*, pages 155–162, IEEE, 2015. 2015 5th International Conference on the.

[61] Andrew Raij, Animikh Ghosh, Santosh Kumar, and Mani Srivastava. Privacy Risks Emerging from the Adoption of Innocuous Wearable Sensors in the Mobile Environment. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, pages 11–20, 2011.

[62] Vanessa Ayala-Rivera and Liliana Pasquale. The grace period has ended: An approach to operationalize GDPR requirements. In *26th IEEE International Requirements Engineering Conference, RE 2018, Banff, AB, Canada, August 20-24, 2018*, pages 136–146, 2018.

[63] Michele Guerriero, Damian Andrew Tamburri, and Elisabetta Di Nitto. Defining, enforcing and checking privacy policies in data-intensive applications. In *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2018, Gothenburg, Sweden, May 28-29, 2018*, pages 172–182, 2018.

[64] Regulation (EU) 2016/679 of the European Parliament, of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data, on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). 2016.

[65] California Senate Judiciary Committee et al. California consumer privacy act: Ab 375 legislative history. 2018.

[66] Nianyu Li, Christos Tsigkanos, Zhi Jin, Schahram Dustdar, Zhenjiang Hu, and Carlo Ghezzi. Poet: Privacy on the edge with bidirectional data transformations. In *2019 IEEE International Conference on Pervasive Computing and Communications, PerCom 2019, Kyoto, Japan, March 11-15, 2019*. IEEE, 2019.

[67] Meir M Lehman. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9):1060–1076, 1980.

[68] Martin Leucker and Christian Schallhart. A brief account of runtime verification. *The Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009.

[69] Jeff Kramer and Jeff Magee. Self-managed systems: an architectural challenge. In *2007 Future of Software Engineering*, pages 259–268. IEEE Computer Society, 2007.

[70] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[71] Mikhail Afanasov, Luca Mottola, and Carlo Ghezzi. Software adaptation in wireless sensor networks. *TAAS*, 12(4):18:1–18:29, 2018.

[72] Radu Calinescu, Danny Weyns, Simos Gerasimou, Muhammad Usman Iftikhar, Ibrahim Habli, and Tim Kelly. Engineering trustworthy self-adaptive software with dynamic assurance cases. *IEEE Transactions on Software Engineering*, 44(11):1039–1069, 2018.

[73] Antonio Filieri, Carlo Ghezzi, Alberto Leva, and Martina Maggio. Self-adaptive software meets control theory: A preliminary approach supporting reliability requirements. In *26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, KS, USA, November 6-10, 2011*, pages 283–292, 2011.

[74] Radu Calinescu, Carlo Ghezzi, Marta Kwiatkowska, and Raffaela Mirandola. Self-adaptive software needs quantitative verification at runtime. *Communications of the ACM*, 55(9):69–77, 2012.

[75] Antonio Filieri, Giordano Tamburrelli, and Carlo Ghezzi. Supporting self-adaptation via quantitative verification and sensitivity analysis at run time. *IEEE Trans. Software Eng.*, 42(1):75–99, 2016.

[76] Christos Tsigkanos, Nianyu Li, Zhi Jin, Zhenjiang Hu, and Carlo Ghezzi. On early statistical requirements validation of cyber-physical space systems. In *Proceedings of the 4th International Workshop on Software Engineering for Smart Cyber-Physical Systems, ICSE 2018, Gothenburg, Sweden, May 27, 2018*, pages 13–18, 2018.

[77] Gordon Blair, Nelly Bencomo, and Robert B France. Models@ run. time. *Computer*, 42(10), 2009.

[78] Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Karl Aberer. Semantic trajectories: Mobility data computation and annotation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(3):49, 2013.

[79] Danny Weyns, M Usman Iftikhar, Danny Hughes, and Nelson Matthys. Applying architecture-based adaptation to automate the management of internet-of-things. In *European Conference on Software Architecture*, pages 49–67. Springer, 2018.

[80] Christos Tsigkanos, Liliana Pasquale, Claudio Menghi, Carlo Ghezzi, and Bashar Nuseibeh. Engineering Topology Aware Adaptive Security: Preventing Requirements Violations at Runtime. In *Proc. of the 22nd Int. Requirements Engineering Conf.*, pages 203–212, 2014.

[81] Christos Tsigkanos, Laura Nenzi, Michele Loreti, Martin Garriga, Schahram Dustdar, and Carlo Ghezzi. Inferring analyzable models from trajectories of spatially-distributed internet-of-things. In *1th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2019, Montreal, Canada, May 25-26, 2019*. IEEE Computer Society, 2019.

[82] Christos Tsigkanos, Liliana Pasquale, Carlo Ghezzi, and Bashar Nuseibeh. On the interplay between cyber and physical spaces for adaptive security. *IEEE Trans. Dependable Sec. Comput.*, 15(3):466–480, 2018.