EDITOR: Schahram Dustdar, dustdar@infosys.tuwien.ac.at

Edge Intelligence—Research Opportunities for Distributed Computing Continuum Systems

Victor Casamayor Pujol, Praveen Kumar Donta, Andrea Morichetta, Ilir Murturi, and Schahram Dustdar, *Technische Universität Wien, 1040, Vienna, Austria*

Edge intelligence and, by extension, any distributed computing continuum system will bring to our future society a plethora of new and useful applications, which will certainly revolutionize our way of living. Nevertheless, managing these systems challenges all previously developed technologies for Internet-distributed systems. In this regard, this article presents a set of techniques and concepts that can help manage these systems; these are framed in the main paradigm for autonomic computing, the well-known monitor–analyze–plan–execute over shared knowledge, or MAPE-K. All in all, this article aims at unveiling research opportunities for these new systems, encouraging the community to work together toward new technologies to make edge intelligence a reality.

Internet-distributed systems are undergoing transcendent changes. More than a decade ago, the emergence of cloud computing was seen as the architecture and future for all Internet-distributed systems. In general, it has produced excellent solutions and substantial business opportunities. However, this hegemony is now changing. Several reasons are driving this paradigm shift; some clear examples follow. First, cloud computing's distance to data generation [the Internet of Things (IoT) or the edge of the network] heavily impacts applications' latency and network congestion. Second, cloud clusters are single points of failure for applications, given their monolithic and centralized architecture. And third, there is an increasing concern for data privacy, which is stored in data centers far from the control of their owners.

To solve the shortcomings of cloud computing, the current emerging paradigm is integrating edge computing and cloud computing. Furthermore, it is merging all computational tiers to develop a computing fabric that gets the best out of each tier. This promised paradigm, named the distributed computing continuum, brings many new opportunities and challenges. In brief, it promises to improve all of the shortcomings of cloud computing by letting applications use their most suited computational resources. However, this implies that this new computational milieu, or computing fabric, can adapt and reconfigure itself to provide such a service.

In parallel, cloud computing is embracing a new trend, named serverless computing.[1] This alleviates all infrastructure management from application developers. Hence, all responsibility is on the infrastructure side, which is managed transparently for the application. In such situations, the infrastructure is adaptive to the dynamic needs of applications. This trend also needs to be embraced at the edge; alleviating the infrastructure management from the application is essential to enable the distributed computing continuum. Otherwise, the complexity that the application developer needs to tackle is massive. The characteristics of the edge tier give this complexity; nevertheless, the combination of the different tiers amplifies its scale and significance. Further, using machine learning (ML) and/or AI components at the edge hardens application requirements in all terms: the amount of data to manage is drastically increased, the communication latency allowed is minimal, or the hardware requirements are

1089-7801 © 2023 IEEE
Digital Object Identifier 10.1109/MIC.2023.3284693
Date of current version 10 July 2023.

much more specific. Simply put, AI/ML at the edge boosts the overall system complexity.

In the following sections, we describe the characteristics of edge computing that pose challenges in the development of a self-adaptive distributed computing continuum paradigm. We differentiate three types of characteristics: core characteristics, which stem from the definition of distributed computing continuum systems (DCCSs); specific characteristics, which further describe the specific behavior of a core characteristic; and derived characteristics, which arise from the aggregation of the two previous ones. Figure 1 shows a schema of the discussed types with the specific instantiation for DCCSs.

CORE CHARACTERISTICS

- *Geographic distribution*: This is a core characteristic of DCCSs. The computational milieu is physically distributed across the space.[2] In other words, any computational resource that can eventually reach the network is part of it. This heterogeneity typically leads to an uneven distribution of resources, which can hinder performance, availability, or maintenance.[3,4] Last, the geographic distribution also imposes different network characteristics across the computational resources, including security/privacy aspects and latency constraints.
- *Large scale*: This is a core characteristic of these systems. They comprise many resources and provide service to many users.[2,5] In this regard, the scale must be considered in the design, and only methodologies that can scale accordingly to the application needs should be applied.
- *Multitenant and multiproprietary*: These are core characteristics of these systems, as computational resources will need to be shared in this emerging paradigm.[6] Otherwise, each application would require its infrastructure, and this is entirely unsustainable. Further, each part of the computational fabric can have different owners, which brings heterogeneity at the business level as well as possible vendor locks and incompatibilities between resources. Hence, this requires open methodologies that allow all involved stakeholders to cooperate and attend.
- *Open system*: Open system is a core characteristic of DCCSs, which are affected by external agents and the state of the external environment.[2,5] Imagine other systems competing or blocking resources or network connections, being jammed due to special events. Hence, they also demand high resilience and adaptive capacity against external and unexpected situations,

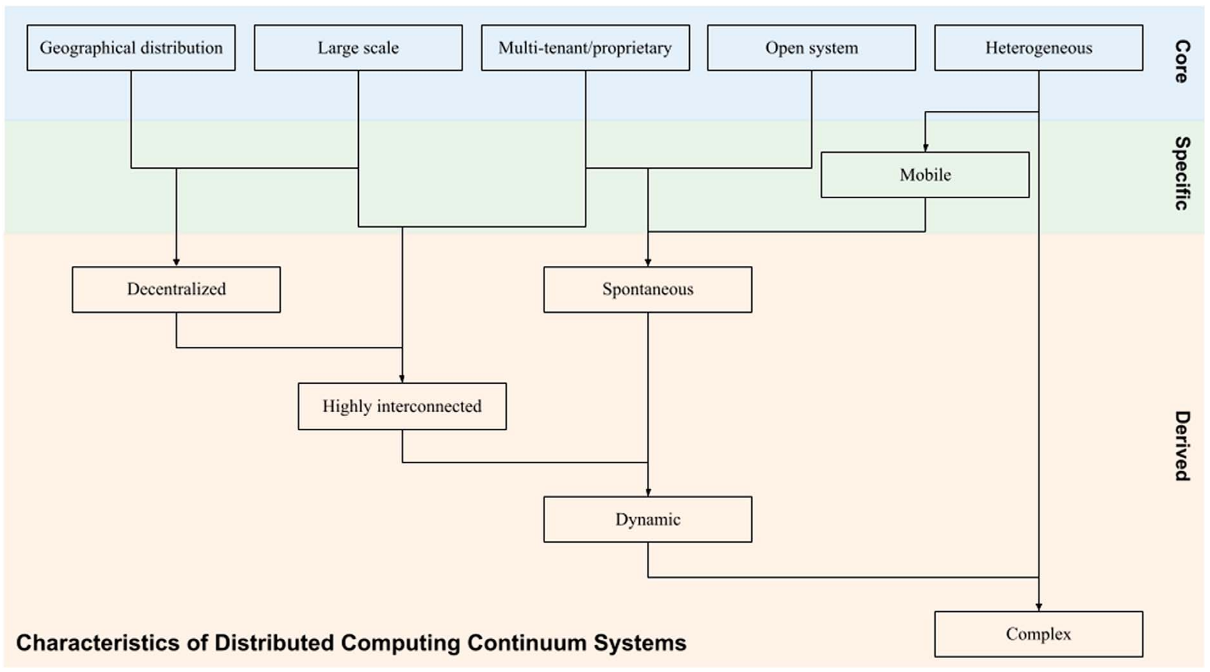


FIGURE 1. Classification and relation of edge computing characteristics.

making security and privacy first-class aspects of these systems.

- *Heterogeneity*: This is the last identified core characteristic. The range of different computing resources on the distributed computing continuum is immense: from single-board computers or small controllers to high-performance computers or quantum computers. Heterogeneity poses great challenges in distributing tasks over resources.[7,8] However, it is also an asset for the efficient usage of infrastructure. Heterogeneity is not only in terms of capabilities but also in terms of limitations and constraints. Hence, each specific resource needs to be managed considering its specific constraints, such as a limited energy budget or intermittent connectivity. Further, it challenges the privacy and security aspects of the systems. However, heterogeneity has another facet: there is heterogeneity in data types, frameworks, and middlewares across the computational fabric.

SPECIFIC CHARACTERISTICS

- *Mobile*: This characteristic specifies that, within the heterogeneity of devices, some operate in mobility, which changes the standard view on computing infrastructures. As a consequence, these devices might become unavailable due to network connectivity problems or battery exhaustion.[4,9,10] On the positive side, mobility enables the relocation of a computing resource to gain network access or to provide computing resources in an isolated area. In any case, mobility requires dynamic topologies and architectures, as resources that were in one place with specific features can move and offer a dynamic quality of service (QoS).

DERIVED CHARACTERISTICS

- *Decentralization*: The previous characteristics (geographic distribution and large scale) preclude centralized architectures. Further, decentralization is needed to benefit from the advantages of edge computing; otherwise, the same issues found in cloud computing will eventually arise. Decentralization poses challenges to resource orchestration and communication,[2,11] given that decisions and messages derived from these processes need to be efficiently spread and managed across the components without a central entity.
- *Spontaneity*: Spontaneity comes from the combination of an open system with a multitenant/multiproprietary organization, and it is further accentuated by the mobility of some infrastructure components. In general, the components of the computing fabric can connect and disconnect unexpectedly.[9] There can be many reasons for this spontaneous behavior; in any case, it hinders the optimal usage of resources, and, further, it risks the general performance of applications as well as the overall security of the system. This brings another dimension to the system's dynamism: beyond the application, users, and environment, now the infrastructure is also dynamic. Therefore, resilient mechanisms to address spontaneity are of utmost importance.
- *Highly interconnected*: This characteristic is derived from the geographical distribution, decentralization, large scale, and multitenant/multiproprietary characteristics.[7] Devices are entangled to provide a specific service; hence, their interconnected dependencies complicate the ability to understand influences between them. This can result in unexpected behavior and a cascade of failures if it is not managed correctly. However, if well managed, this feature can provide robustness to the system, as it can be a source of redundancy.
- *Dynamic*: Several of the previous characteristics, such as spontaneity, have explicitly mentioned that the distributed computing continuum is dynamic. Nevertheless, due to its significant difference from cloud computing systems, which are mostly static, we have also included this characteristic separately. This dynamic infrastructure behavior is novel for Internet-distributed systems, and it requires a new set of methods and techniques to enable and embrace this dynamism.
- *Complexity*: This last characteristic of the distributed computing continuum can be considered a consequence of all of the previous ones. However, its connotations on the system can produce severe phenomena. Three main issues need to be highlighted: (1) Small differences in the system's features can lead to very different scenarios, which means that making decisions without properly considering the entire system can be dangerous. (2) Failures can propagate in a cascade mode, making it costly to solve them reactively once they start propagating. (3) Understanding the system's internal logic and relations can be sophisticated and require specific tools and methods.

This discussion shows, from a high-level perspective, the complexity that the distributed computing continuum and, consequently, the edge need to manage to enable applications that fulfill their expected requirements. Furthermore, threats exist inside and outside traditional network boundaries, and establishing trustworthiness in the distributed computing continuum is among the core challenges that should be addressed. Finally, we aim to highlight that the challenge is massive, and we only foresee one way to achieve it: bringing self-adaptive intelligence to the computing fabric. This is the only way to enable general edge intelligence.

In the "Vision" section, the overall vision for DCCSs is presented. Then, in the "Techniques" section, the candidate techniques to be part of the presented vision are explained and discussed. Finally, the last section presents this work's conclusions.

VISION

The challenges to benefiting from DCCSs are massive. Fortunately, we do not need to start from scratch. This work takes as a baseline the current autonomic cloud computing and aims to show our vision on how to build a framework to manage DCCSs. In this regard, this work leverages the *monitor–analyze–plan–execute* over shared knowledge (MAPE-K) cycle as a basis for structuring the relations between the different pieces that will enable the transformation from autonomic cloud computing to self-adaptive DCCSs.

Self-adaptive intelligence refers to those intelligence traits that allow adaptive and efficient behavior in systems inhabiting the distributed computing continuum. Hence, we are looking at self-adaptive systems—in this context, their management typically follows the MAPE-K schema.[12] In its basic form, it consists of five blocks:

› *Monitoring* is in charge of interacting with the sensors and processing the obtained data.
› *Analyze* aims at understanding the current situation and needs of the system being managed.
› *Plan* develops the required actions to be performed to keep all requirements satisfied.
› *Execute* interacts with the actuators of the managed system to modify those necessary aspects of the system.
› *Knowledge*, which is represented as a shared block, provides to the others the insights required to complete their tasks.

This schema, compared to the classic control loop, provides a clear separation between the managing system and the managed system and better modularity capabilities due to its block definition. Further, *monitor-analyze-plan-execute* are connected, although not always sequentially related, which means that monitor could send data to plan without the intervention of analyze. The original MAPE-K schema can be observed in Figure 2.

However, we argue here the need to build on top of this schema to make it suitable for DCCSs. First, the environment, as one of the primary sources of uncertainty,[13] needs to be explicitly managed. This requirement is already highlighted in this survey by Krupitzer et al.[14] from a general perspective. Hence, we add to the managing system the capability to use sensors and actuators from the managed system to influence the environment, as depicted in Figure 2.

Second, systems are large scale and complex; hence, we must highlight the need to learn from runtime experience explicitly. In general, the learning feature is assumed to be within the *analyze* block; however, given the advances in ML techniques and their suitability,[15] we argue that it is important to incorporate a *learn* block. In general, the distributed learning technique selected will set requirements for the managing system that need to be carefully incorporated. Hence, adding this block sets them up front and ensures that the learning is feasible.

Finally, there are several architectural solutions for distributed systems following the MAPE-K schema,[16] and we are not yet in the position of providing a final architecture, or representation, for DCCSs. Further, we question if the concept of architecture, which relates to something static, properly fits the computing fabric or if we need a new type of abstraction to represent these emerging systems. Regardless of how they are represented, we are convinced that any system will be at least partially distributed. Hence, due to the massive amount of data and their heterogeneity, a *communicate* block needs to be added to the managing system to ensure that the techniques and methods used for communication are in line with the system's capabilities. Within this context, *communicate* is foreseen as a vital block to enforce security and privacy policies in a distributed, large-scale, and heterogeneous system. Furthermore, one should notice that neither the *communicate* nor *learn* blocks are explicitly connected to any of the original MAPE-K blocks. They are conceptualized as the knowledge block, which any of the others can leverage. This allows any block to use learning techniques to improve its performance, reliability, etc. Similarly, communication can be used by any block to communicate with another system.

The MAPE-K schema, even with the proposed conceptual modifications, is very general and can be

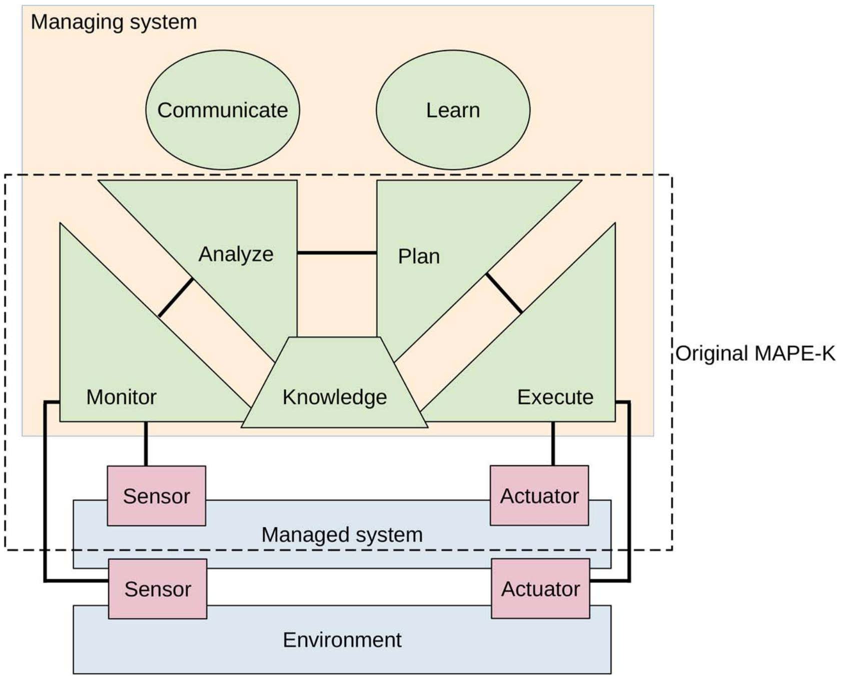


FIGURE 2. *Monitor–analyze–plan–execute* over shared knowledge (MAPE-K) schema with two additional blocks: communicate and learn. It also shows the relation of the managing system to the environment.

applied to any system. Hence, the key is not solely on the general schema but on the conceptualization and development of each block to match the needs of the distributed computing continuum. In this regard, we can take an evolutionary perspective to understand how systems are conditioned to the characteristics of their environment. In general, systems inhabiting a specific environment are local models of it, which means that, by observing the system, one can understand its environment's characteristics. Further, systems that persist are the ones optimally modeling their local environment.

We have already started to provide some intuitions on how these systems could behave, as introduced in Dustdar et al.[17] However, considering that it is in the hands of our community to develop these new DCCSs, it is of utmost importance to develop them and, consequently, each block of the managing system by understanding and taking into consideration all environment characteristics.

TECHNIQUES

In this section, we analyze promising techniques for developing DCCSs, providing their description, their relations with the challenges described, and their mapping to the MAPE-K cycle as well as, finally, showcasing an example of their applicability. More importantly, the techniques are organized incrementally from those that would provide an evolution from the cloud computing model to the simplest cloud–edge model, until a complete aggregation provides a fully autonomous and self-adaptive DCCS. Figure 3 presents how the phases and associated candidate techniques incrementally help solve our ambitions for these systems.

Incorporate Edge Computing

Our first ambition is to incorporate edge computing into the baseline. This does not aim to solve all challenges presented before but to set the basics of the system we are building, i.e., cloud computing, edge computing, and the MAPE-K cycle. The three following techniques (Figure 4) are candidates to enable the transition from an autonomic cloud system to a system that jointly leverages cloud and edge resources. Consider that, in this case, the techniques presented are complementary.

Deep Service-Level Objectives (DeepSLOs)

SLOs are commonly used in cloud computing to develop agreements between users and cloud service providers, named service-level agreements. In general, SLOs target low-level metrics of the cloud infrastructure, and they specify a range of values or a set of thresholds in which the requirements are considered satisfied. Recently, we

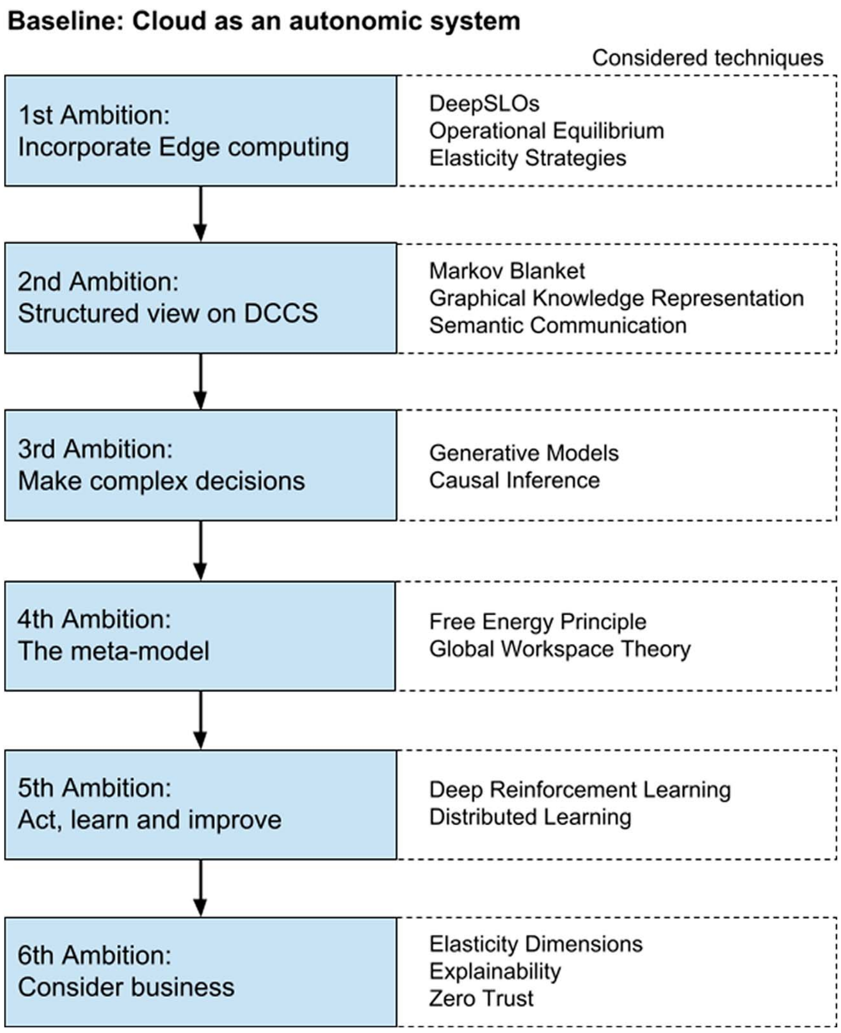

Baseline: Cloud as an autonomic system
Considered techniques
1st Ambition: Incorporate Edge computing
DeepSLOs
Operational Equilibrium
Elasticity Strategies
2nd Ambition: Structured view on DCCS
Markov Blanket
Graphical Knowledge Representation
Semantic Communication
3rd Ambition: Make complex decisions
Generative Models
Causal Inference
4th Ambition: The meta-model
Free Energy Principle
Global Workspace Theory
5th Ambition: Act, learn and improve
Deep Reinforcement Learning
Distributed Learning
6th Ambition: Consider business
Elasticity Dimensions
Explainability
Zero Trust

FIGURE 3. Schema of ambitions to develop our vision for distributed computing continuum systems (DCCSs) from the current autonomic cloud computing paradigm. DeepSLO: deep service-level objective.

have been working on the conceptualization[18] and use of high-level metrics for cloud SLOs[19,20] to better align cloud users and providers and to bring informed and precise elasticity strategies to the cloud.

From these concepts, we foresee an extension of SLOs toward a hierarchically structured set of SLOs, named DeepSLOs, where the lower layers of a DeepSLO consist of SLOs mapping low-level requirements, such as CPU or memory usage, and, as we move to higher layers of the DeepSLO, the abstraction level of the requirements is equally increased, such as cost efficiency. These bring several features to DCCSs: 1) A single DeepSLO can be in charge of an autonomic component of the system, providing objectives and elasticity strategies at different levels of abstraction. 2) There exist internal relations between the SLOs at the same level of abstraction (horizontal) and also between different levels (vertical). In general, relations aim at describing causal behaviors. However, vertical relations also incorporate purpose due to the increase in abstraction. 3) A complete DCCS can be described with a set of DeepSLOs connected at the highest level of abstraction. This way, they can share the main objectives, and, as information goes down toward lower abstraction levels, each component can address them according to its intrinsic needs and capabilities.

› *Challenges addressed*: DeepSLOs are conceptualized to handle several characteristics of the distributed computing continuum, such as its geographical distribution, decentralization, large scale, and heterogeneity. The hierarchical

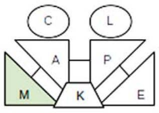
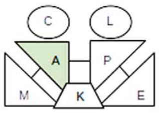
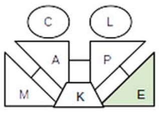
Method	MAPE-K+CL	Edge-Cloud continuum
DeepSLOs		• Geographical distribution • Decentralization • Large scale • Heterogeneity
Operational Equilibrium		• Dynamic • Heterogeneous
Elasticity Strategies		• Spontaneous • Decentralization • Open

FIGURE 4. Techniques for the *incorporate edge computing* ambition with their relation to MAPE-K and the characteristics of the edge. CL: communicate and learn.

structure of DeepSLOs enforces an aggregation of data toward higher level system metrics, providing a structured schema to deal with heterogeneous data at lower levels and ensuring homogeneity at top levels. Further, several DeepSLOs can be connected only through their top-level abstraction to manage an entire application. Both their structure and capacity to connect top-level metrics provide means to tackle the decentralization and the geographical distribution of DCCSs.

› *Components of MAPE-K*: Regarding its relation with the MAPE-K schema (see Figure 4), DeepSLOs are focused on the monitor block, given their ability to keep track of system metrics at all levels. Further, they also provide the required knowledge regarding the aggregation and abstraction of the metrics needed to reach all of the necessary abstraction levels. DeepSLOs are also linked with the communication block, as they define the data that have to be communicated among different entities to ensure the proper management of the entire system. Finally, in cloud computing, each SLO is usually associated with an elasticity strategy; given an SLO violation, the elasticity strategy is triggered. However, in this new paradigm that we are discussing, several SLOs might be simultaneously violated, and the possible elasticity strategies and their consequences belong to a much larger space. Therefore, we assume the need for specific techniques in the other blocks to deal with this second feature of typical cloud SLOs. In other words, we decouple the elasticity strategies of the system from the SLOs, given that it might not be a one-to-one relation as it is now.

Operational Equilibrium

This is a novel concept forced by the inclusion of cloud systems in edge computing. SLOs and, by extension, DeepSLOs are driven through thresholds that generally are fixed and are only changed if the application being managed undergoes important changes. However, in the distributed computing continuum paradigm, where the underlying infrastructure is dynamic, we claim that these thresholds also depend on its current configuration. Simply put, the thresholds managing the elasticity strategy of an inference function in the cloud have to be different from those for the same inference function at the edge. Hence, the operational equilibrium allows us to describe and adjust a set of thresholds managing the SLOs' behavior according to the infrastructure configuration. Now, the infrastructure's behavior depends not only on the application requirements but also on the same infrastructure where it is deployed.

› *Challenges addressed*: The operation equilibrium is a needed concept to ease the management of dynamic and heterogeneous systems. Further, the operational equilibrium can be leveraged at any scale or abstraction level of the system. The expected behavior of the system, which is embedded in the operational equilibrium, can be defined through SLOs, and these can have different levels of abstraction. Hence, the operational equilibrium can aggregate lower abstractions, such as CPU usage, and higher level abstractions. The key is its capacity to tailor the SLOs to the dynamism and heterogeneity of the underlying infrastructure.
› *Components of MAPE-K*: Concerning the MAPE-K schema, this concept falls into the analyze block,

as it can evaluate the state of the system given its behavior and current configuration. Intuitively, the operational equilibrium can be imagined as a composed signal that summarizes the system status but that takes into account its current configuration.

Elasticity Strategies

The capacity of cloud computing to adjust the resources provisioned to each specific workload accordingly to its specific needs is called elasticity. Elasticity (together with its business model) has been the enabler of the cloud computing success. There are many works on cloud computing and elasticity.[21,22,23] Although there has been some work aiming at modeling elasticity[24] to provide a more formal basis, current solutions are generally practical.[25,26] Hence, they are able to elastically manage the cloud without the need for a model. In a previous work,[27] we developed a language to connect the three elasticity axes[28] (resources, quality, and cost) with their associated elasticity strategies.

All of this falls short for the distributed computing continuum. In cloud computing, there are mainly three elasticity strategies—scaling vertically or horizontally or migrating a task—but the complexity of the distributed computing continuum calls for a more fine-grained definition of elasticity strategies, which are able to actually change the behavior and architecture of the system. Simply put, some tasks and responsibilities can be moved from the cloud to the edge or vice versa, implying a modification of network connections and other components of the system to accommodate such change. Further, the complexity of the distributed computing continuum requires a precise model of the system able to predict the system's change caused by an elasticity strategy. In this regard, there is work on proactively applying elasticity strategies based on ML models.[29,30] Otherwise, a reactive and nonmodeled action can produce unexpected and unwanted consequences in the system. In this regard, we see the need for modeling elasticity strategies as functions that modify the structure and configuration of the system, providing predictions about the system's behavior.

- *Challenges addressed*: Model-based and proactive elasticity strategies are required to cope with distributed computing continuum complexity. Developing this technique for a decentralized, spontaneous, and open system is challenging but crucial to achieving similar success to cloud computing.
- *Components of MAPE-K*: The elasticity strategies for distributed computing continuum are in the execute block (see Figure 4), given that they are in charge of applying the selected elasticity strategy.

Summary—Techniques for a Cloud Evolution

DeepSLOs, operational equilibrium, and elasticity are inherited from cloud computing and are envisioned as the first required evolution for DCCSs. These three techniques have clear interfaces between them. DeepSLOs are in charge of the structure and relations between the system components, setting the right hooks for the application and its underlying infrastructure. The operational equilibrium is simply a conceptual twist on the SLO definition, which enables different SLO descriptions according to the relation between the infrastructure and the deployed application component. Finally, the elasticity strategies for DCCSs are more complex and diverse than in cloud computing, and, hence, they required a model and proactive behavior.

Structured View on DCCSs

Our second ambition is to find an adequate structure for edge–cloud autonomic systems. A proper system (or ecosystem) organization is needed to benefit from all components and their relations. Hence, once we have added the cloud and edge components, our second ambition is to deal with some of the challenges by benefiting from a meaningful system structure. In this regard, the following techniques (Figure 5) intend to overcome some of the aforementioned challenges; e.g., the Markov blanket provides separation capabilities to limit the system scale. Two of them, the Markov blanket and graphical knowledge representation (GKR), focus on the actual representation and structure of the system, while semantic communication aims at making these structures viable without the overhead of a communication overlay. These techniques can integrate with each other to provide a structured model for DCCSs.

Markov Blanket

In Bayesian statistics, a Markov blanket allows the value of a variable to be inferred.[31] Hence, it consists of a set of variables that provide enough information to infer the value of another one. When the variable to be inferred is represented in a Bayesian network as a directed acyclic graph, its Markov blanket is formed by its parents, children, and the other parents of its children. Then, the variable to be inferred is statistically dependent only on its Markov blanket and is statistically independent of all the rest.

An ontological perspective is given to the Markov blanket in the context of the free energy principle (FEP).[32] Given the Markov blanket of "anything," Hipolito et al.[32]

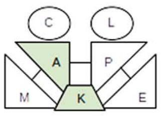
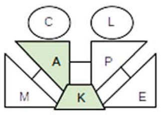
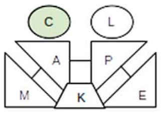
Method	MAPE-K+CL	Edge-Cloud continuum
Markov Blanket		• Open System • Decentralization • Large scale
Graphical Knowledge Representation		• Decentralization • Geographically distribution • Complex • Large scale
Semantic Communication		• Geographic distribution • Decentralization • Open

FIGURE 5. Techniques for the *structured view on DCCS* ambition with their relation to MAPE-K and the characteristics of the edge.

use the Markov blanket's property of the statistical independence of a variable with respect to all others to mathematically develop the concept of the separation of a thing with respect to its environment. From this perspective, Hipolito et al.,[32] define four types of variables: those internal to the Markov blanket, which establish the system state; those nodes that belong to the Markov blanket influenced by external variables and affecting the inner states, which are the sensory states; those nodes also belonging to the Markov blanket influenced by the internal states and that can influence the external states, which are the action states; and, finally, the external variables or environment.

› *Challenges addressed*: This latter view on the Markov blanket provides a taxonomy of components for any adaptive system enabling modularity and a formal separation of components through conditional independence. This allows us to formally model the interactions with the environment, enhancing the tools to deal with open systems. Further, a Markov blanket is not linked to a precise granularity, which means that the Markov blanket can be built over different scales of the system, providing the same structure and allowing the usage of the same techniques. Both modularity and this capacity to adjust to the required scale are aligned with the large scale and decentralization characteristics of the distributed computing continuum, providing different lenses with which to analyze the system at the most convenient scale. Further, considering the conditional independence between system components brings a filtering capacity to the system. Simply put, it is possible to only focus on those components that are influencing the part of the system under analysis, greatly alleviating the system's scale issue.

› *Components of MAPE-K*: The relation of the Markov blanket with the MAPE-K loop of Figure 2 is twofold. On the one side, it is part of the knowledge, as it embeds a system's representation. Hence, all other blocks can use such knowledge to find relations and dependencies between components of the system. On the other side, it can be part of the analyze block. The Markov blanket probabilistic concept enables one to obtain state variables without the need for a direct observation, which is fundamental when dealing with higher level abstractions and decentralized systems.

GKR

GKR is a visualized representation of knowledge extracted from raw data and structured as a graph with their entities and relations.[33] In general, the quality of the representation is based on the ability to generalize the knowledge.[34] Most of the traditional approaches use GRK for data analytics in cloud computing, such as knowledge representation and ontology engineering[35,36] or monitoring system performance.[37,38] In contrast, it can also help control the systems and their resources,[39,40] given that GKR simplifies the decision making of systems and provides a fast response. Further, it can also inspect the uncovered constraints in the systems.[41]

› *Challenges addressed*: The ability of GKR, as a tool, to inspect faults, gaps, and flows in systems[39] helps address decentralized, complex, and distributed systems. Further, it can also be used in common tasks for DCCSs, such as scheduling, using the functional dependencies based on the local and global geometry in networks. Further, the knowledge extracted from the GKR also identifies the resources' availability and improves its usability efficiency.[40] Hence, it can enhance the predictability of the system and provide tools to deal with uncertainty. Last, GKR is sensitive to rare faults, which can be helpful when managing large-scale systems.

› *Components of MAPE-K*: The GKR is envisioned as part of the knowledge and analysis blocks in the MAPE-K schema (Figure 2). On the knowledge block, due to its ability to keep track of functional dependencies and the availability of resources, information can be helpful for any other block. GKR's capabilities to detect faults and inspect flows are part of the needs within the analysis block.

Semantic Communication

This technique is seen as a required evolution from Shannon's communication ideas, where the focus is no longer on accurately reproducing the transmitter's data in the receiver, which is currently taken for granted, but on enriching the data with meaningful information for the receiver to better process and understand the information received.[42,43] The simplest example of semantic communication is adding a timestamp to the data, allowing the receiver to decide how to process it according to its freshness.[44] Semantic communication requires incorporating encoders and decoders to extract the information embedded inside the message. Our intuition sets this technique as a way to embed system status information in the messages sent by components, preventing the need to develop an entire managing network overlay for DCCSs.

› *Challenges addressed*: Semantic communication is a technique that can provide solutions to the geographic distribution and decentralization of these systems, enabling an enriched communication capacity that prevents the need to develop an overlay communication for the system. Further, it also fits with the system's dynamic behavior, as the enriched information can reach the complete topology of the system even if it is not fixed, as it leverages the application messages to convey the information.

› *Components of MAPE-K*: Regarding Figure 2, where the MAPE-K schema is represented, semantic communication relates to the communication block. This capacity of enriching the communication's information requires the development of specific components able to encode and decode that information.

Summary—Structured View on DCCS

The techniques selected to structure the DCCS have different perspectives. The Markov blanket technique aims at providing a fixed ontological structure to any component of the system, regardless of its granularity. Further, it offers causal filtering to the system; i.e., it identifies the influencing components enabling a precise focus on the needed ones. GKR is a set of tools that can be used to obtain the Markov blankets describing the system, and also, through its analysis, they can help detect rare faults and gaps in the system. Finally, semantic communication adds the capability to all of these components to communicate state information, creating a virtual overlay of communication that can be used by techniques, such as the DeepSLO, to extract the required data to evaluate the state of the system.

Make Complex Decisions

This third ambition provides the organized DCCS with the capability of making complex decisions. With the techniques introduced here (Figure 6), we expect that the system will be able to predict, to a certain extent, the behavior of its components and relations, including the environment. Hence, we are going beyond basic thresholds from SLO based decision making. As an example, the last technique presented is causal inference, which is mostly focused on the relations between system components and can be used to forecast interventions and counterfactuals.

Generative Models

A generative model is a statistical model of a joint distribution of variables. Simply put, it is a model for the $P(X, Y)$, where X is an observable variable, and Y is an objective variable. Hence, generative models can infer possible future states of an objective variable together with its expected observation. There are different ways to develop generative models.

The central topic of this article is DCCSs; hence, we are interested in these methods having a tool to model the environment or complex components of the system, which can be only partially observable. Generative adversarial networks (GANs)[45,46]; variational autoencoders[47]; or even energy-based approaches, such as the one introduced by Zhang et al.[48] can produce generative models but require a training period. Further, in the case of GANs, their training also requires

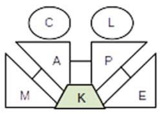
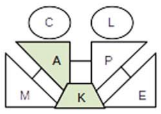
Method	MAPE-K+CL	Edge-Cloud continuum
Generative Models		• Interconnected • Dynamic • Complex
Causal Inference		• Interconnected • Dynamic • Complex

FIGURE 6. Techniques for the *make complex decisions* ambition with their relation to MAPE-K and the characteristics of the edge.

developing another deep network model able to discriminate whether its input is from a GAN or is real. Hence, in general, the training of these models is complex and unstable.[49] Nevertheless, they are achieving great success in image generation and other fields, such as learning from imitation.[50]

Another type of generative model is based on Markov chains, and they have been successfully used in the cloud environment. For instance, they can be used to model cloud workloads through a Markovian arrival process,[51] which is an extension of continuous-time Markov chains (CTMCs). Further, several works use Markov decision processes (MDPs) to model the elasticity characteristics of cloud computing. Naskos et al.[52] model horizontal elasticity through MDPs to obtain a quantitative runtime verification of the elasticity decisions. Similarly, Ai et al.[24] develop a CTMC model to evaluate the elasticity behavior of cloud platforms. Also, in the context of self-adaptive systems, Moreno et al.[53] develop a model for an intensive computing system, such as cloud computing, where adaptation decisions, which can be understood as elastic strategies, are analyzed and decided using an MDP that models the adaptation process.

› *Challenges addressed*: DCCSs are open, interconnected, complex, and dynamic. Hence, developing models to grasp their behavior is difficult. Further, the impossibility of observing *all* variables due to decentralization, geographical distribution, or just because the environment cannot be completely observed limits the capacity of developing models. Nevertheless, generative models are a unique opportunity to predict the system components' behavior through the viable observations and take actions consequently.
› *Components of MAPE-K*: Generative models produce knowledge for the managing system. Hence, their relation with the MAPE-K schema (Figure 2) is straightforward, as they belong to the knowledge block.

Causal Inference

Causal inference is devoted to describing the causal relations between a set of variables. Discovering causal relations or discovering why certain events occur has always been part of the core of human nature. Thanks to the work of Pearl and Mackenzie,[54] causal inference has a mathematical framework that allows the unveiling of causal relations between variables from data. However, observational data are not usually enough to provide causal knowledge. Hence, experiments are required to extract complete causal knowledge. To be precise, in Pearl and Mackenzie's book,[54] three causal rungs are described, from observational to counterfactual, which present different types of causal queries. In this regard, the higher the rung is, the more complex the required data are.

In general, this knowledge is represented as a causal graph. These are an evolution from Bayesian probabilistic representations,[31] in which the relation between two nodes models cause and effect. Further, deep learning approaches are also being developed to make use of causal relations to help to model situations.[56]

In the distributed computing continuum, causal inference can help solve several issues: 1) Monitoring can be driven by causal inference, ensuring that only the relevant aspects of the system are tracked. 2) Anomalies can be tracked to their causal roots by exploiting a causal graph. 3) Elasticity strategies can be analyzed before being used due to the capability of causal inference to analyze how an action can affect the system.

› *Challenges addressed*: Causal inference is a technique that will allow understanding of the reasons for events happening in the system, even if the system is complex and highly interconnected.

Further, it is able to anticipate how the system or the environment will respond to certain actions or events, providing a causal rationale for existing models.

- *Components of MAPE-K*: In the MAPE-K schema, from Figure 2, causal inference is part of the knowledge and analysis blocks, given that, from one side, causal graphs store information about the system relations, and causal inference can analyze runtime situations and actions.

Summary—Make Complex Decisions

For this ambition, we have selected two techniques. We highlighted generative models as techniques that are able to model the behavior of the system's components. Most of the techniques take probabilistic approaches, including the usage of deep neural networks, such as GANs, or different types of Markov chains. Causal inference also takes a probabilistic approach, focusing more on the relations between system components. Hence, by combining both and leveraging the developed structure of the system, we can have an optimized organization with prediction capabilities for all components and relations.

THE METAMODEL

The fourth ambition aims at abstracting the DCCSs beyond their components and relations, developing the capacity to consider the system as an entity with overarching goals beyond optimizing the performance of a specific component (Figure 7). This brings the opportunity to make decisions and policies systemwide, which, in turn, will be applied accordingly by the system components. Hence, we envision here the opportunity to challenge key issues such as system efficiency and sustainability.

FEP

From cognitive science, while studying the brain as a complex adaptive system, it was found that the system followed the FEP.[57] Further studies have shown the applicability of the FEP spread to any complex adaptive system.[58,59] During the last few years, the momentum achieved by this research has been huge—and not free of critique,[60,61] due to claims about generalization and the lack of precise applicability. In any case, it is a research line that is relevant to DCCSs.

In brief, the FEP states that an adaptive system always minimizes the difference between the expected observation of its environment and the obtained observation. Hence, a system that behaves as stated in the FEP will improve its knowledge of the environment, reducing the gap between its internal model of the environment and the actual environment by comparing an expected observation to the obtained one. This implies that the system has a generative model of the environment and its components, given that it has to predict future observations to compare them. Therefore, it is possible to embed systemwide policies and analyze them through the FEP framework. Interestingly, the FEP comes with a corollary named active inference, where the system has to perform actions to adapt while minimizing the *surprise*.[62] This technique has some similarities with reinforcement learning (RL),[63,64] mainly considering that an agent is learning from its actions in an open and complex setting. However, the main goal of active inference is not to maximize a reward function but to optimize an internal environment model to which the agent has to adapt.

- *Challenges addressed*: The FEP provides an interesting perspective for DCCSs, as it focuses on the expected behavior of a complex system aiming to persist in a dynamic environment. Hence, the FEP

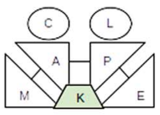
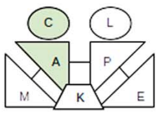
Method	MAPE-K+CL	Edge-Cloud continuum
Free Energy Principle		• Interconnected • Dynamic • Complex • Multi-tenant • Large scale
Global Workspace Theory		• Geographic distribution • Heterogeneity • Complex

FIGURE 7. Techniques for the *metamodel* ambition with their relation to MAPE-K and the characteristics of the edge.

can help us understand the needs of a system to adapt to a dynamic environment continuously. Further, if systems behave following the FEP, this allows us to understand and predict future behaviors of other systems that are sharing resources, enabling smoother relationships. In this regard, it can be argued that the FEP is less greedy than other goal-oriented behaviors, as it aims at improving the system behavior with respect to the environment instead of optimizing any other external metric, which may require competition with other components or systems. Simply put, given a fixed network connection capacity, latency optimization of competing applications can lead to unsatisfying results, while a better knowledge of the environment, including other applications and the network capacity, could lead all components to an overall optimum. Nevertheless, it is still to be investigated if such adaptive behaviors will lead to this overall optimum or will lack the capacity to address the application requirements properly. Last but not least, it has been shown by Palacios et al.[65] that the same behavior described by the FEP is followed by systems with different scales, which is a very convenient feature for DCCSs.

› *Components of MAPE-K*: The relation of the FEP and MAPE-K is with the knowledge block. There, the managing system can leverage the FEP to know its most convenient behavior or to help determine the expected behavior of other systems inhabiting the distributed computing continuum.

Global Workspace Theory (GWT)

The *GWT*[66,67] is a cognitive neuroscience theory that proposes that the mind is a computational model where different sources of information come together to create an internal representation of the environment. According to GWT, the mind processes all relevant memories and perceptions in a "global neuronal workspace" and broadcasts conclusions globally to make the system "conscious." This theory is relevant for understanding self-reference, where an observer can create a learning loop that builds causal models. This approach helps humans to generalize from and modify their perceptions. Recently, the deep learning community[68] has shown interest in this theory as a potential influence on decision making and generalization in AI. Some researchers[69] have suggested that deep learning techniques could represent GWT, creating a "global latent workspace." Others have proposed using a shared global workspace for coordinating neural network modules in multiagent systems.[70] These approaches typically use the concept of attention[71] as a building block, allowing the system to highlight essential input components dynamically. Focusing on DCCSs, this theory presents a framework to combine inputs from different system components and is capable, through these attention mechanisms, of selecting and *summarizing* the most valuable information. Hence, it can opt for systemwide decisions by considering componentwise models.

› *Challenges addressed*: As introduced by Morichetta et al.,[72] GWT supports the DCCS decision-making phase, creating a higher level representation of a system's internal components and aggregating and filtering relevant information. It also contributes to dealing with DCCSs' geographic distribution and heterogeneity. The GWT builds on top of "inductive biases"; i.e., it starts from the assumption of knowing what modules are involved in the system's decision for the particular goal.

› *Components of MAPE-K*: Finally, GWT would be present in the communicate and analysis blocks in the MAPE-K schema. The role of GWT in communication is to let components coordinate decision making through higher abstraction information. GWT would also help get a global and informed understanding of the environment, providing valuable functionalities for the analysis phase.

Summary—The Metamodel

Two complementary approaches to build a metamodel for DCCSs have been proposed. On the one side, the FEP aims at leveraging the generative models of the system and the environment to optimize the entire system's interaction with its environment. On the other side, the GWT presents a systematic theory to analyze the different viewpoints of the system components to build the systemwide perspective.

Act, Learn, and Improve

Now that we have the DCCS completed, our ambition is to provide a lifelong learning strategy (Figure 8). Consider that the acting capacities of the system are considered from the beginning, when elasticity strategies are extended to the edge. In brief, it consists of using its (meta-) models to take actions, learn over the outcomes, and improve the models.

Deep RL

Deep RL (DRL) is an ML branch that combines RL with neural networks. Hence, the system or agent learns

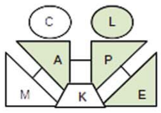
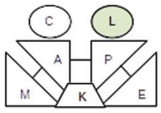
Method	MAPE-K+CL	Edge-Cloud continuum
Deep Reinforcement Learning		• Spontaneous • Dynamic • Complex
Distributed Learning		• Interconnected • Dynamic • Complex

FIGURE 8. Techniques for the *act, learn, and improve* ambition with their relation to MAPE-K and the characteristics of the edge.

through experiences by interacting with the environment.[73] DRL is used in various fields, including natural language processing, vehicular communication, robotics, health care, computer vision, etc. The success of these applications has motivated researchers to choose DRL to address different issues of edge computing and make it intelligent.[74] The significant benefits identified from the DRL are as follows. 1) The DRL does not require any predetermined datasets to train the system, unlike other ML approaches, such as supervised or unsupervised. 2) It provides the best decisions based on the trial-and-error method by considering the exploitation or exploration algorithms with the previous optimal decisions. 3) Unlike RL, DRL does not require additional space to maintain a Q-table, and, thus, it is also not necessary to compute all of the Q-values associated with each state. It uses an experience replay buffer instead of a Q-table.

In the literature, DRL is used to address various challenges of the components of DCCSs, mainly on the edge tier. These include resource management,[75,76,77] computational/service offloading,[78] task migration,[79,80] caching,[81] trajectory control for mobile devices,[82,83] performance optimization,[84] etc. Further, it can be helpful for developing scalable and elastic libraries for DCCSs from the learnings on the edge tier.

› *Challenges addressed*: DRL is useful to DCCSs due to its capacity to deal with uncertain and complex environments and provides a framework to make decisions on the fly. Nevertheless, some issues from DRL need to be addressed to become feasible. The complexity of the neural network, defining the best reward function, and how to involve multiagent systems are still open challenges. Nevertheless, embracing such technology can bring new strategies to solve these problems.
› *Components of MAPE-K*: Regarding the MAPE-K schema, Figure 2, DRL involves an entire decision loop. Hence, it influences the analysis, plan, and execution blocks.

Distributed Learning

In the context of distributed learning, federated learning (FL) is currently the most significant technique. FL[85] proposes new ways to develop learning methods for large-scale ML, focusing on distributed optimization and privacy-preserving mechanisms. At its core, FL aims at learning a single networkwide model, training it locally; maintaining privacy within certain boundaries guaranteed by the federations; and only exchanging intermediate, periodic, and higher abstract updates with a central server.[86] This approach guarantees the distribution of the algorithm computation and lets systems develop potentially more reliable models, performing a knowledge combination from multiple data sources that would not have been possible otherwise due to law or privacy constraints.[87] However, efficiently achieving this solution in real scenarios is still challenging.

In the context of DCCSs, distributed learning techniques are crucial to improving the models that describe the system and the environment while keeping data on premises to minimize network congestion and to have privacy as a cornerstone for these systems.

› *Challenges addressed*: Distributed learning techniques are needed in DCCSs to deal with their dynamic, complex behavior given the scale, distribution, and decentralization of the system. Otherwise, learning and the improvements to the system that it can bring will not be available.
› *Components of MAPE-K*: Regarding the MAPE-K schema, distributed learning techniques will influence the learning block as well as the communication block, given that these methods usually require specific communication settings between their learning components. Finally, this needs to be embedded in the

knowledge block, given that the obtained learning will update all models in a DCCS.

Summary—Act, Learn, and Improve

We have presented two techniques for this ambition. Distributed learning is complementary to RL, as it has to provide the techniques to enable the other techniques in a distributed and decentralized manner. RL is an approach that integrates the entire act–learn–improve cycle.

Consider Business

Last but not least, this ambition considers the need for DCCSs to be embraced by all concerned stakeholders to make them viable. All previous ambitions have achieved a functional and sustainable system. However, there are aspects that need to be considered to make these systems a reality. Hence, the following techniques (Figure 9) provide three required aspects for all stakeholders.

Elastic Dimensions: Resources, Quality, and Cost

Resources, quality, and cost are three abstractions that allow grouping all cloud computing system requirements. The resources dimension explicitly references the required infrastructure to host an application; in cloud computing, resources are usually seen as homogeneous and unlimited, given their large-scale and virtualization capabilities. In DCCSs, some resources can be scarce or very specific for an application, which needs special consideration and is critical given the characteristics of the infrastructure resources. The quality dimension measures the output given by the application; it is not QoS or quality of experience (QoE) but an abstraction that defines the performance by combining both QoS and QoE. An illustrative example can be the resolution of a streaming video, which can vary to adjust to the current state of the environment. Hence, the tradeoff can be done as follows: instead of adding more resources, the quality—i.e., the resolution of the video—can be lowered to keep the performance in the current situation. Finally, the cost dimension provides a clear business perspective of the system state, allowing the development of tradeoffs with respect to the other variables.

Using these three elastic dimensions to represent DCCSs stems from previous work on cloud computing and elasticity strategies,[27,28,88,89] where cloud computing systems are elastically managed using these higher level abstractions. This allows for several benefits, such as a business-oriented perspective on system management or the development of holistic elastic strategies that can consider these three dimensions simultaneously. The research work of Murturi and Dustdar[90] considers three-dimensional elasticity strategies for adapting applications deployed on edge resources.

- *Challenges addressed*: Using similar high-level abstractions for the distributed computing continuum can be beneficial, considering the large-scale, heterogeneous, and multitenant/multiproprietary characteristics. Simply put, large-scale and heterogeneous resources demand abstractions that the different components of the system can understand and share. Similarly but at another level, multitenant and multiproprietary characteristics imply that there is a manifold of

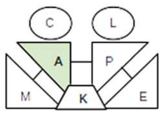
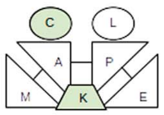
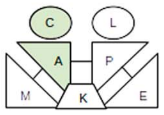
Method	MAPE-K+CL	Edge-Cloud continuum
Elasticity Dimensions		• Large scale • Heterogeneous • Multi-tenant/multi-proprietary
Explainability		• Multi-tenant/multi-proprietary • Highly interconnected • Complex
Zero Trust		• Geographically distributed • Heterogeneous • Spontaneous

FIGURE 9. Techniques for the *consider business* ambition with their relation to MAPE-K and the characteristics of the edge.

stakeholders who can leverage a shared understanding by using high-level abstractions of the requirements.

- *Components of MAPE-K*: The usage of these concepts within the presented MAPE-K schema falls on the analyze block, given that these will be in charge of understanding the system state with respect to these high-level requirements. Further, the MAPE-K schema can be applied at different system levels, meaning that it can be used over the entire system or over the smallest autonomic component of the system, being a scale-free concept. From this perspective, the use of high-level abstractions, such as the resources, quality, and cost, aims at being applicable at any scale.

Explainability

The omnipresence and complexity reached by ML algorithms, especially neural networks, calls for ways to explore the results and how the model took certain decisions. In this context, *explainability* plays an important role, mainly through explainable AI (XAI). XAI refers to methods and techniques in applying ML such that human experts can understand the models' results.[91,92,93,94] The *explanation* results from inspecting which of the models' inputs lead to a specific output. XAI methods can belong to various classes.[95] They can be intrinsic in the model, i.e., the model itself is interpretable, or post hoc, i.e., the interpretation methods provide clarification on top of the trained model. XAI can have interpretation methods with different objectives, like feature summary statistics or visualization, showing the model internals or looking for representative points. Furthermore, XAI methods can be model specific or model agnostic and provide local or global interpretations—i.e., if it can explain the entire model behavior or provide an individual prediction. It is essential to build systems and processes where accountability, transparency, and integrity are first-class citizens.[96] In the last years, researchers have been inspecting ways to address this.[97,98]

In the scenario of the self-adaptive management of DCCSs, explainability, transparency, and accountability can allow understanding of how components—or the system itself—make decisions and produce predictions, bringing a higher level knowledge. In addition, this result can have positive implications for better communicating the system behavior, increasing trust, and supporting users to evaluate the decisions,[99] characteristics that are needed when dealing with real-time or critical systems.

- *Challenges addressed*: Explainability is key in dealing with multitenant and multiproprietary systems. It can provide a framework where liability and responsibility can be obtained and tracked, enabling trusted communication between all stakeholders. Further, it can help understand the system's state by alleviating characteristics such as complexity or intricate interconnections between components.
- *Components of MAPE-K*: In the MAPE-K loop, explainability is part of two components. On the one hand, explainability has to be considered in the communication block as it influences how and what is communicated between DCCS stakeholders. On the other hand, explainability is part of the system's knowledge, given that it can answer queries that aim to answer why the system acts in a certain way.

Zero Trust

Conventional security approaches focus on guarding sensitive resources by applying cryptography or perimeter-based security methods to ensure that only trusted entities can access a secured domain.[100] The perimeter-based security approach via control access points grants or denies access to requestors (i.e., users, devices, software components, etc.) based on credentials or certificate-based authentication. These control points check and verify the authenticity of requestors before they can access any internal digital asset (i.e., resources, services, data, etc.). Once the requestors are verified, they can access internal resources and then are always considered *trusted peers* since they are within the network perimeter. Consequently, perimeter-based security, known as the impenetrable fortress, largely ignores threats that may derive from trusted peers in the network (i.e., an attacker may have compromised the user credentials or device of the trusted peer). In the context of DCCSs, this involves partially giving access to digital assets; regulating the amount of traffic; and regularly verifying resource access help minimize cyberattack surfaces, detect and prevent the lateral movement of attackers, and increase the trustworthiness within the systems.

Zero trust[55] is an innovative approach that comes into play where an associated set of mechanisms can be applied to secure and increase trustworthiness in the distributed computing continuum. The principles of zero trust emphasize protecting individual digital assets rather than the entire network or domain. Therefore, no implicit trust is assumed, and peers are always considered untrustworthy (i.e., no matter if they are external or internal entities). More specifically, every access to a digital asset is always verified, and the granted access is always as minimal as possible.

- *Challenges addressed*: Zero trust techniques enable the enforcement of security and privacy policies in distributed, heterogeneous, and spontaneous systems, such as the distributed computing continuum.
- *Components of MAPE-K*: Regarding the MAPE-K schema, zero trust concepts will influence the communication and analysis block. Regarding communication, these methods require secure and trusted communication between heterogeneous devices, system components, or stakeholders. Zero trust will require performing analysis on components and user behavior; hence, it will also be there.

Summary—Consider Business

We have provided three techniques to deal with the business-oriented needs of DCCSs. The elasticity dimensions provide a common ground for analysis and agreement of the system state when multiple stakeholders are involved. Explainability and auditing capacities over these large and complex systems are also a *must-have* when different stakeholders are involved; otherwise, accountability would be impossible. Finally, any cyberphysical system requires the strong consideration of security and privacy; due to the characteristics of DCCSs, zero trust is the best candidate.

CONCLUSION

Edge intelligence is an emerging computational paradigm that uses ML/AI techniques at the edge. In this article, we first present the distributed computing continuum paradigm as a needed extension of all emerging computational paradigms and current paradigms, such as cloud computing. Enabling edge intelligence and, by extension, the distributed computing continuum paradigm still requires addressing many challenges.

In this work, we have highlighted the characteristics of edge systems that depart from any previous Internet-distributed system. Hence, new perspectives are needed to face these challenges. We have used the autonomic cloud computing paradigm together with the MAPE-K framework as a basis to build DCCSs. We have introduced new components that we foresee as fundamental to shaping the managing framework and aligning it with the needs brought by the unique characteristics of DCCSs.

Further, we propose several techniques and concepts, some already existing in other fields and others new, to cope with all of the challenges brought by DCCSs. Further, we organize the techniques through development ambitions, sketching a road map to enable DCCSs. We have provided a short description of each technique and critical references for the interested reader, and we have shared some clues regarding how these techniques and concepts relate to the challenges of the DCCSs and their mapping to a MAPE-K schema. Edge intelligence and DCCSs will bring a manifold of new opportunities to Internet-distributed systems that will shape our future. Hence, we highlight the main research opportunities to encourage researchers to investigate and develop new solutions to make these new paradigms a reality.

REFERENCES

1. J. Schleier-Smith et al., “What serverless computing is and should become,” *Commun. ACM*, vol. 64, no. 5, pp. 76–84, May 2021, doi: 10.1145/3406011.
2. D. Kumar, G. Baranwal, Y. Shankar, and D. P. Vidyarthi, “A survey on nature-inspired techniques for computation offloading and service placement in emerging edge technologies,” *World Wide Web*, vol. 25, no. 5, pp. 2049–2107, Sep. 2022, doi: 10.1007/s11280-022-01053-y.
3. F. Wang, X. Huang, H. Nian, Q. He, Y. Yang, and C. Zhang, “Cost-effective edge server placement in edge computing,” in *Proc. 5th Int. Conf. Syst., Contr. Commun.*, New York, NY, USA: Association for Computing Machinery, Mar. 2020, pp. 6–10, doi: 10.1145/3377458.3377461.
4. K. Cao, L. Li, Y. Cui, T. Wei, and S. Hu, “Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing,” *IEEE Trans. Ind. Informat.*, vol. 17, no. 1, pp. 494–503, Jan. 2021, doi: 10.1109/TII.2020.2975897.
5. T. L. Duc, R. G. Leiva, P. Casari, and P.-O. Östberg, “Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey,” *ACM Comput. Surv.*, vol. 52, no. 5, pp. 94:1–94:39, Sep. 2019, doi: 10.1145/3341145.
6. S. Zobaed, A. Mokhtari, J. P. Champati, M. Kourouma, and M. A. Salehi, “Edge-multiAI: Multi-tenancy of latency-sensitive deep learning applications on edge,” in *Proc. IEEE/ACM 15th Int. Conf. Utility Cloud Comput. (UCC)*, Dec. 2022, pp. 11–20, doi: 10.1109/UCC56403.2022.00012.
7. A. Ullah, H. Dagdeviren, R. C. Ariyattu, J. DesLauriers, T. Kiss, and J. Bowden, “MiCADO-edge: Towards an application-level orchestrator for the cloud-to-edge computing continuum,” *J Grid Comput.*, vol. 19, no. 4, Nov. 2021, Art. no. 47, doi: 10.1007/s10723-021-09589-5.
8. S. Dustdar and I. Murturi, “Towards IoT processes on the edge,” in *Next-Gen Digital Services. A Retrospective and Roadmap for Service Computing*

of the Future: Essays Dedicated to Michael Papazoglou on the Occasion of His 65th Birthday and His Retirement, M. Aiello, A. Bouguettaya, D. A. Tamburri, and W. J. van den Heuvel, Eds., Cham, Switzerland: Springer, 2021, pp. 167–178.

9. M. Habib Ur Rehman, S. L. Chee, T. Y. Wah, A. Iqbal, and P. P. Jayaraman, "Opportunistic computation offloading in mobile edge cloud computing environments," in *Proc. 17th IEEE Int. Conf. Mobile Data Manage. (MDM)*, Jun. 2016, vol. 1, pp. 208–213, doi: 10.1109/MDM.2016.40.
10. I. Murturi and S. Dustdar, "A decentralized approach for resource discovery using metadata replication in edge networks," *IEEE Trans. Services Comput.*, vol. 15, no. 5, pp. 2526–2537, Sep./Oct. 2021, doi: 10.1109/TSC.2021.3082305.
11. S. Dustdar and I. Murturi, "Towards distributed edge-based systems," in *Proc. IEEE 2nd Int. Conf. Cogn. Mach. Intell. (CogMI)*, 2020, pp. 1–9, doi: 10.1109/CogMI50398.2020.00021.
12. J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003, doi: 10.1109/MC.2003.1160055.
13. S. M. Hezavehi, D. Weyns, P. Avgeriou, R. Calinescu, R. Mirandola, and D. Perez-Palacin, "Uncertainty in self-adaptive systems: A research community perspective," *ACM Trans. Auton. Adaptive Syst.*, vol. 15, no. 4, pp. 1–36, Dec. 2021, doi: 10.1145/3487921.
14. C. Krupitzer, F. M. Roth, S. Vansyckel, G. Schiele, and C. Becker, "A survey on engineering approaches for self-adaptive systems," *Pervasive Mobile Comput.*, vol. 17, no. Part B, pp. 184–206, Feb. 2015, doi: 10.1016/j.pmcj.2014.09.009.
15. D. Weyns et al., "Towards better adaptive systems by combining MAPE, control theory, and machine learning," in *Proc. Int. Symp. Softw. Eng. Adaptive Self-Manag. Syst. (SEAMS)*, May 2021, pp. 217–223, doi: 10.1109/SEAMS51251.2021.00036.
16. D. Weyns et al., "On patterns for decentralized control in self-adaptive systems," in *Software Engineering for Self-Adaptive Systems II*, vol. 7475, R. de Lemos, H. Giese, H. A. Müller, and M. Shaw, Eds., Berlin, Heidelberg: Springer, 2013, pp. 76–107, doi: 10.1007/978-3-642-35813-5.
17. S. Dustdar, V. C. Pujol, and P. K. Donta, "On distributed computing continuum systems," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 4092–4105, Apr. 2023, doi: 10.1109/TKDE.2022.3142856.
18. S. Nastic et al., "SLOC: Service level objectives for next generation cloud computing," *IEEE Internet Comput.*, vol. 24, no. 3, pp. 39–50, May/Jun. 2020, doi: 10.1109/MIC.2020.2987739.
19. T. Pusztai et al., "SLO script: A novel language for implementing complex cloud-native elasticity-driven SLOs," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, 2021, pp. 21–31, doi: 10.1109/ICWS53863.2021.00017.
20. T. Pusztai et al., "A novel middleware for efficiently implementing complex cloud-native SLOs," in *Proc. IEEE 14th Int. Conf. Cloud Comput. (CLOUD)*, 2021, pp. 410–420, doi: 10.1109/CLOUD53861.2021.00055.
21. E. F. Coutinho, F. R. de Carvalho Sousa, P. A. L. Rego, D. G. Gomes, and J. N. de Souza, "Elasticity in cloud computing: A survey," *Ann. Telecommun.*, vol. 70, nos. 7–8, pp. 289–309, Aug. 2015, doi: 10.1007/s12243-014-0450-7.
22. Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Elasticity in cloud computing: State of the art and research challenges," *IEEE Trans. Services Comput.*, vol. 11, no. 2, pp. 430–447, Mar. 2018, doi: 10.1109/TSC.2017.2711009.
23. A. Barnawi, S. Sakr, W. Xiao, and A. Al-Barakati, "The views, measurements and challenges of elasticity in the cloud: A review," *Comput. Commun.*, vol. 154, pp. 111–117, Mar. 2020, doi: 10.1016/j.comcom.2020.02.010.
24. W. Ai et al., "On elasticity measurement in cloud computing," *Scientific Program.*, vol. 2016, Jun. 2016, Art. no. 7519507, doi: 10.1155/2016/7519507.
25. Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Autonomic vertical elasticity of docker containers with ELASTICDOCKER," in *Proc. IEEE Int. Conf. Cloud Comput. (CLOUD)*, Sep. 2017, pp. 472–479, doi: 10.1109/CLOUD.2017.67.
26. L. Baresi, D. Y. X. Hu, G. Quattrocchi, and L. Terracciano, "KOSMOS: Vertical and horizontal resource autoscaling for Kubernetes," in *Service-Oriented Computing*, vol. 13121, H. Hacid, O. Kao, M. Mecella, N. Moha, and H. Paik, Eds., Cham, Switzerland: Springer, 2021, pp. 821–829.
27. G. Copil, D. Moldovan, H. L. Truong, and S. Dustdar, "SYBL: An extensible language for controlling elasticity in cloud applications," in *Proc. 13th IEEE/ACM Int. Symp. Cluster, Cloud, Grid Comput. (CCGrid)*, 2013, pp. 112–119, doi: 10.1109/CCGrid.2013.42.
28. S. Dustdar, Y. Guo, B. Satzger, and H. L. Truong, "Principles of elastic processes," *IEEE Internet Comput.*, vol. 15, no. 5, pp. 66–71, Sep. 2011, doi: 10.1109/MIC.2011.121.
29. L. Toka, G. Dobreff, B. Fodor, and B. Sonkoly, "Adaptive AI-based auto-scaling for Kubernetes," in *Proc. 20th IEEE/ACM Int. Symp. Cluster, Cloud

Internet Comput. (CCGRID), May 2020, pp. 599–608, doi: 10.1109/CCGrid49817.2020.00-33.

30. N.-M. Dang-Quang, M. Yoo, J. F. De, and P. Santana, “Deep learning-based autoscaling using bidirectional long short-term memory for Kubernetes,” *Appl. Sci.*, vol. 11, Apr. 2021, Art. no. 3835, doi: 10.3390/app11093835.
31. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann, 1988.
32. I. Hipolito, M. Ramstead, L. Convertino, A. Bhat, K. Friston, and T. Parr, “Markov blankets in the brain,” *Neuroscience Biobehavioral Rev.*, vol. 125, pp. 88–97, Jun. 2020, doi: 10.1016/j.neubiorev.2021.02.003.
33. A.-L. Kalouli and R. Crouch, “GKR: The graphical knowledge representation for semantic parsing,” in *Proc. Workshop Comput. Semantics Events Roles (SemBEaR)*, 2018, pp. 27–37.
34. B. P. Majumder, N. Potti, S. Tata, J. B. Wendt, Q. Zhao, and M. Najork, “Representation learning for information extraction from form-like documents,” in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, Jul. 2020, pp. 6495–6504, doi: 10.18653/v1/2020.acl-main.580. [Online]. Available: https://aclanthology.org/2020.acl-main.580
35. Z. S. Ageed, R. K. Ibrahim, and M. Sadeeq, “Unified ontology implementation of cloud computing for distributed systems,” *Current J. Appl. Sci. Technol.*, vol. 39, pp. 82–97, Nov. 2020, doi: 10.9734/cjast/2020/v39i3431039.
36. W. Wang, S. De, G. Cassar, and K. Moessner, “Knowledge representation in the Internet of Things: Semantic modelling and its applications,” *Automatika*, vol. 54, no. 4, pp. 388–400, Oct. 2013, doi: 10.7305/automatika.54-4.414.
37. D. Gürdür et al., “Knowledge representation of cyber-physical systems for monitoring purpose,” *Procedia CIRP*, vol. 72, pp. 468–473, Jun. 2018, doi: 10.1016/j.procir.2018.03.018.
38. P. O’Donovan, C. Gallagher, K. Leahy, and D. T. O’Sullivan, “A comparison of fog and cloud computing cyber-physical interfaces for industry 4.0 real-time embedded machine learning engineering applications,” *Comput. Industry*, vol. 110, no. 1, pp. 12–35, May 2019, doi: 10.1016/j.compind.2019.04.016.
39. P. D. Maio, “System level knowledge representation for edge intelligence,” in *Proc. Artif. Intell. Cloud Edge Comput.*, Cham, Switzerland: Springer, 2022, pp. 255–275, doi: 10.1007/978-3-030-80821-1_12.
40. C. Sanin, Z. Haoxi, I. Shafiq, M. M. Waris, C. S. de Oliveira, and E. Szczerbicki, “Experience based knowledge representation for Internet of Things and cyber physical systems with case studies,” *Future Gener. Comput. Syst.*, vol. 92, pp. 604–616, Mar. 2019, doi: 10.1016/j.future.2018.01.062.
41. P. K. Donta and S. Dustdar, “The promising role of representation learning for distributed computing continuum systems,” in *Proc. IEEE Int. Congr. Intell. Service-Oriented Syst. Eng. (IEEE SOSE)*, 2022, pp. 126–132, doi: 10.1109/SOSE55356.2022.00021.
42. J. Bao et al., “Towards a theory of semantic communication,” in *Proc. IEEE 1st Int. Netw. Sci. Workshop*, 2011, pp. 110–117, doi: 10.1109/NSW.2011.6004632.
43. Q. Lan et al., “What is semantic communication? A view on conveying meaning in the era of machine intelligence,” *J. Commun. Inf. Netw.*, vol. 6, no. 4, pp. 336–371, Dec. 2021, doi: 10.23919/JCIN.2021.9663101.
44. E. Uysal et al., “Semantic communications in networked systems: A data significance perspective,” Mar. 2021. [Online]. Available: https://arxiv.org/abs/2103.05391v3
45. I. Goodfellow et al., “Generative adversarial nets,” in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
46. I. Goodfellow et al., “Generative adversarial networks,” *Commun. ACM*, vol. 63, no. 11, pp. 139–144, Nov. 2020, doi: 10.1145/3422622.
47. D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” Dec. 2013. [Online]. Available: https://arxiv.org/abs/1312.6114v10
48. D. Zhang, N. Malkin, Z. Liu, A. Volokhova, A. Courville, and Y. Bengio, “Generative flow networks for discrete probabilistic modeling,” Feb. 2022. [Online]. Available: https://arxiv.org/abs/2202.01361v1
49. T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, “Improved techniques for training GANs,” in *Proc. 30th Int. Conf. Adv. Neural Inf. Process. Syst.*, Jun. 2016, pp. 2234–2242.
50. J. Ho and S. Ermon, “Generative adversarial imitation learning,” in *Proc. 30th Int. Conf. Adv. Neural Inf. Process. Syst.*, Dec. 2016, pp. 4572–4580.
51. S. Pacheco-Sanchez, G. Casale, B. Scotney, S. McClean, G. Parr, and S. Dawson, “Markovian workload characterization for QoS prediction in the cloud,” in *Proc. IEEE 4th Int. Conf. Cloud Comput.*, 2011, pp. 147–154, doi: 10.1109/CLOUD.2011.100.
52. A. Naskos et al., “Dependable horizontal scaling based on probabilistic model checking,” in *Proc. IEEE/ACM 15th Int. Symp. Cluster, Cloud, Grid*

Comput. (CCGrid), Aug. 2015, pp. 31–40, doi: 10.1109/CCGrid.2015.91.

53. G. A. Moreno, J. Cámara, D. Garlan, and B. Schmerl, "Proactive self-adaptation under uncertainty: A probabilistic model checking approach," in *Proc. 10th Joint Meeting Eur. Softw. Eng. Conf. ACM SIGSOFT Symp. Found. Softw. Eng. (ESEC/FSE)*, Aug. 2015, pp. 1–12, doi: 10.1145/2786805.2786853.
54. J. Pearl and D. Mackenzie, *The Book of Why: The New Science of Cause and Effect*, 1st ed. New York, NY, USA: Basic Books, 2018.
55. S. Rose, O. Borchert, S. Mitchell, and S. Connelly, "Zero trust architecture," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. Special Publication (NIST SP) 800-207, 2020.
56. K. Javed, M. White, and Y. Bengio, "Learning causal models online," Jun. 2020, *arXiv:2006.07461*.
57. K. Friston, J. Kilner, and L. Harrison, "A free energy principle for the brain," *J. Physiol. Paris*, vol. 100, nos. 1–3, pp. 70–87, Jul. 2006, doi: 10.1016/j.jphysparis.2006.10.001.
58. M. Kirchhoff, T. Parr, E. Palacios, K. Friston, and J. Kiverstein, "The Markov blankets of life: Autonomy, active inference and the free energy principle," *J. Roy. Soc. Interface*, vol. 15, no. 138, Jan. 2018, Art. no. 20170792, doi: 10.1098/rsif.2017.0792.
59. K. Friston et al., "The free energy principle made simpler but not too simple," Jan. 2022. [Online]. Available: https://arxiv.org/abs/2201.06387v2
60. M. Aguilera, B. Millidge, A. Tschantz, and C. L. Buckley, "How particular is the physics of the free energy principle?" *Phys. Life Rev.*, vol. 40, pp. 24–50, Mar. 2022, doi: 10.1016/j.plrev.2021.11.001.
61. V. Raja, D. Valluri, E. Baggs, A. Chemero, and M. L. Anderson, "The Markov blanket trick: On the scope of the free energy principle and active inference," *Phys. Life Rev.*, vol. 39, pp. 49–72, Dec. 2021, doi: 10.1016/j.plrev.2021.09.001.
62. K. Friston, J. Mattout, and J. Kilner, "Action understanding and active inference," *Biol. Cybern.*, vol. 104, nos. 1–2, pp. 137–160, Feb. 2011, doi: 10.1007/s00422-011-0424-z.
63. A. Tschantz, B. Millidge, A. K. Seth, and C. L. Buckley, "Reinforcement learning through active inference," Feb. 2020. [Online]. Available: https://arxiv.org/abs/2002.12636v1
64. N. Sajid, P. J. Ball, T. Parr, and K. J. Friston, "Active inference: Demystified and compared," *Neural Comput.*, vol. 33, no. 3, pp. 674–712, Jan. 2021, doi: 10.1162/neco_a_01357.
65. E. R. Palacios, A. Razi, T. Parr, M. Kirchhoff, and K. Friston, "On Markov blankets and hierarchical self-organisation," *J. Theor. Biol.*, vol. 486, Feb. 2020, Art. no. 110089, doi: 10.1016/j.jtbi.2019.110089.
66. B. J. Baars, *A Cognitive Theory of Consciousness*. New York, NY, USA: Cambridge Univ. Press, 1993.
67. S. Dehaene and L. Naccache, "Towards a cognitive neuroscience of consciousness: Basic evidence and a workspace framework," *Cognition*, vol. 79, nos. 1–2, pp. 1–37, Apr. 2001, doi: 10.1016/S0010-0277(00)00123-2.
68. Y. Bengio, Y. LeCun, and G. E. Hinton, "Deep learning for AI," *Commun. ACM*, vol. 64, no. 7, pp. 58–65, Jul. 2021, doi: 10.1145/3448250.
69. R. Van Rullen and R. Kanai, "Deep learning and the global workspace theory," *Trends Neurosci.*, vol. 44, no. 9, pp. 692–704, Sep. 2021, doi: 10.1016/j.tins.2021.04.005.
70. A. Goyal et al., "Coordination among neural modules through a shared global workspace," 2021, *arXiv:2103.01197*.
71. D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2015, *arXiv:1409.0473*.
72. A. Morichetta, V. C. Pujol, and S. Dustdar, "A roadmap on learning and reasoning for distributed computing continuum ecosystems," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, 2021, pp. 25–31, doi: 10.1109/EDGE53862.2021.00021.
73. K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017, doi: 10.1109/MSP.2017.2743240.
74. L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, and X. Shen, "Deep reinforcement learning for autonomous Internet of Things: Model, applications and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1722–1760, Thirdquarter 2020, doi: 10.1109/COMST.2020.2988367.
75. X. Xiong, K. Zheng, L. Lei, and L. Hou, "Resource allocation based on deep reinforcement learning in IoT edge computing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1133–1146, Jun. 2020, doi: 10.1109/JSAC.2020.2986615.
76. Y. He, Y. Wang, C. Qiu, Q. Lin, J. Li, and Z. Ming, "Blockchain-based edge computing resource allocation in IoT: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2226–2237, Feb. 2021, doi: 10.1109/JIOT.2020.3035437.
77. H. Peng and X. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Trans. Netw.*

Sci. Eng., vol. 7, no. 4, pp. 2416–2428, Oct./Dec. 2020, doi: 10.1109/TNSE.2020.2978856.

78. J. Chen, H. Xing, Z. Xiao, L. Xu, and T. Tao, "A DRL agent for jointly optimizing computation offloading and resource allocation in MEC," *IEEE Internet Things J.*, vol. 8, no. 24, pp. 17,508–17,524, Dec. 2021, doi: 10.1109/JIOT.2021.3081694.
79. C. Liu, F. Tang, Y. Hu, K. Li, Z. Tang, and K. Li, "Distributed task migration optimization in MEC by extending multi-agent deep reinforcement learning approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1603–1614, Jul. 2021, doi: 10.1109/TPDS.2020.3046737.
80. Y. Chen, Y. Sun, C. Wang, and T. Taleb, "Dynamic task allocation and service migration in edge-cloud IoT system based on deep reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 16,742–16,757, Sep. 2022, doi: 10.1109/JIOT.2022.3164441.
81. G. Qiao, S. Leng, S. Maharjan, Y. Zhang, and N. Ansari, "Deep reinforcement learning for cooperative content caching in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 247–257, Oct. 2019, doi: 10.1109/JIOT.2019.2945640.
82. L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning-based trajectory planning for multi-UAV assisted mobile edge computing," *IEEE Trans. Cogn. Commun. Netw.*, vol. 7, no. 1, pp. 73–84, Mar. 2021, doi: 10.1109/TCCN.2020.3027695.
83. Q. Liu, L. Shi, L. Sun, J. Li, M. Ding, and F. Shu, "Path planning for UAV-mounted mobile edge computing with deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5723–5728, May 2020, doi: 10.1109/TVT.2020.2982508.
84. X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Performance optimization in mobile-edge computing via deep reinforcement learning," in *Proc. IEEE 88th Veh. Technol. Conf. (VTC-Fall)*, 2018, pp. 1–6, doi: 10.1109/VTCFall.2018.8690980.
85. H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, 2017, pp. 1273–1282.
86. T. Li, A. K. Sahu, A. S. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020, doi: 10.1109/MSP.2020.2975749.
87. P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, Jun. 2021, doi: 10.1561/2200000083.
88. G. Copil, D. Moldovan, H.-L. Truong, and S. Dustdar, "Multi-level elasticity control of cloud services," in *Service-Oriented Computing*, vol. 8274, S. Basu, C. Pautasso, L. Zhang, and X. Fu, Eds., Berlin, Heidelberg: Springer, 2013, pp. 429–436.
89. H. L. Truong, S. Dustdar, and F. Leymann, "Towards the realization of multi-dimensional elasticity for distributed cloud systems," *Procedia Comput. Sci.*, vol. 97, pp. 14–23, Jan. 2016, doi: 10.1016/j.procs.2016.08.276.
90. I. Murturi and S. Dustdar, "DECENT: A decentralized configurator for controlling elasticity in dynamic edge networks," *ACM Trans. Internet Technol.*, vol. 22, no. 3, pp. 1–21, Apr. 2022, doi: 10.1145/3530692.
91. D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, and G. Z. Yang, "XAI—Explainable artificial intelligence," *Sci. Robot.*, vol. 4, no. 37, Dec. 2019, Art. no. eaay7120, doi: 10.1126/scirobotics.aay7120.
92. T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artif. Intell.*, vol. 267, pp. 1–38, Feb. 2019, doi: 10.1016/j.artint.2018.07.007.
93. C. J. Hazard, C. Fusting, M. Resnick, M. Auerbach, M. Meehan, and V. Korobov, "Natively interpretable machine learning and artificial intelligence: Preliminary results and future directions," 2019. [Online]. Available: http://arxiv.org/abs/1901.00246
94. M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?': Explaining the predictions of any classifier," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.
95. C. Molnar, *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Independently Published, Mar. 2023. [Online]. Available: https://christophm.github.io/interpretable-ml-book/
96. M. Brundage et al., "Toward trustworthy AI development: Mechanisms for supporting verifiable claims," 2020, *arXiv:2004.07213*.
97. I. D. Raji et al., "Closing the AI accountability gap: Defining an end-to-end framework for internal algorithmic auditing," in *Proc. Conf. Fairness, Accountability, Transparency*, 2020, pp. 33–44.
98. S. Oppold and M. Herschel, "A system framework for personalized and transparent data-driven decisions," in *Proc. Int. Conf. Adv. Inf. Syst. Eng.*, Cham, Switzerland: Springer, 2020, pp. 153–168, doi: 10.1007/978-3-030-49435-3_10.

99. D. D. Shin, "The effects of explainability and causability on perception, trust, and acceptance: Implications for explainable AI," *Int. J. Hum. Comput. Stud.*, vol. 146, Feb. 2021, Art. no. 102551, doi: 10.1016/j.ijhcs.2020.102551.
100. N. F. Syed, S. W. Shah, A. Shaghaghi, A. Anwar, Z. Baig, and R. Doss, "Zero trust architecture (ZTA): A comprehensive survey," *IEEE Access*, vol. 10, pp. 57,143–57,179, May 2022, doi: 10.1109/ACCESS.2022.3174679.

VICTOR CASAMAYOR PUJOL is a project assistant in the Distributed Systems Group at Technische Universität (TU) Wien, 1040, Vienna, Austria. His research interests include self-adaptive methodologies for computing continuum systems. Pujol received his Ph.D. degree in information and communication technologies from Universitat Pompeu Fabra, Barcelona, Spain. He is a Member of IEEE. Contact him at v.casamayor@dsg.tuwien.ac.at.

PRAVEEN KUMAR DONTA is a university assistant in the Distributed Systems Group, TU Wien, 1040, Vienna, Austria. His research interests include machine learning for wireless sensor networks, the Internet of Things, and fog/edge computing. Donta received his Ph.D. degree from the Department of Computer Science and Engineering at the Indian Institute of Technology (Indian School of Mines), Dhanbad, India. He is a Member of IEEE. Contact him at p.donta@dsg.tuwien.ac.at.

ANDREA MORICHETTA is a university assistant with the Distributed Systems Group, TU Wien, 1040, Vienna, Austria. His research interests include the intersection of complex distributed systems and machine learning, network monitoring, and security. Morichetta received his Ph.D. degree in machine learning and big data approaches for automatic Internet monitoring from the Telecommunication Network Group, Politecnico di Torino, Turin, Italy. He is a Member of IEEE. Contact him at a.morichetta@dsg.tuwien.ac.at.

ILIR MURTURI is a postdoctoral researcher in the Distributed Systems Group, TU Wien, 1040, Vienna, Austria. His research interests include DCCSs, edge AI, and cyberphysical systems. Murturi received his Ph.D. degree in resource management and elasticity control in edge networks from the Distributed Systems Group, TU Wien, Vienna, Austria. He is a Member of IEEE. Contact him at i.murturi@dsg.tuwien.ac.at.

SCHAHRAM DUSTDAR is a full professor of computer science, heading the Research Division of Distributed Systems at TU Wien, 1040, Vienna, Austria. He is a Fellow of IEEE. Contact him at dustdar@dsg.tuwien.ac.at.