Invited Paper: Composable Autonomic Offerings for Decentralized Service Provisioning in the Computing Continuum

Ildefons Magrans de Abril*
ildefons.magrans@upf.edu
Universitat Pompeu Fabra
Barcelona, Spain

Boris Sedlak
boris.sedlak@upf.edu
Universitat Pompeu Fabra
Barcelona, Spain

Victor Casamayor Pujol
victor.casamayor@upf.edu
Universitat Pompeu Fabra
Barcelona, Spain

Schahram Dustdar
schahram.dustdar@upf.edu
ICREA, TU Wien
Barcelona, Spain

Abstract

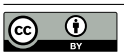
The computing continuum is driving service ecosystems that span cloud, edge, and device-level resources across multiple administrative domains. In such environments, no single entity has complete visibility or control, yet dominant approaches to service composition remain centered on orchestrating functional execution units under centralized assumptions. This creates a fundamental mismatch: existing abstractions do not support composition under decentralized ownership and uncertainty.

This paper proposes a decentralized architectural framework based on *Composable Autonomic Offerings* (CAOs). In this model, the primitive of composition is an offering that encapsulates a compositional structure, a multi-objective SLO vector, a survival model, and a realization envelope capturing admissible deployment conditions. Rather than producing a single configuration, each CAO constructs and maintains a set of candidate offerings through local composition of upstream offerings, and exposes their non-dominated subset as a Pareto Front Offering (PFO).

The architecture is organized as a local, event-driven loop that integrates candidate generation, Pareto-front maintenance, and continuous adaptation of instantiated offerings. This makes trade-offs and uncertainty explicit and composable across providers. In particular, survivability is embedded into the service abstraction itself, allowing resilience to be constructed through composition rather than imposed through external deployment strategies.

This perspective shifts service provisioning from orchestrating execution units to composing autonomous, uncertainty-aware offerings, enabling more flexible, resilient, and accessible service ecosystems in decentralized environments.

*Corresponding author.



ApPLIED'26, Egham, United Kingdom

ACM ISBN 979-8-4007-2462-6/2026/07
https://doi.org/10.1145/3820355.3820395

CCS Concepts

• **Computer systems organization** → *Cloud computing*; *n-tier architectures*; **Self-organizing autonomic computing**; **Heterogeneous (hybrid) systems**; • **Computing methodologies** → **Distributed algorithms**.

Keywords

Computing Continuum, Multi-agent Systems, Service Level Objectives, Multi-Objective optimization, Decentralized Management

ACM Reference Format:
Ildefons Magrans de Abril, Boris Sedlak, Victor Casamayor Pujol, and Schahram Dustdar. 2026. Invited Paper: Composable Autonomic Offerings for Decentralized Service Provisioning in the Computing Continuum. In *Advanced Tools, Programming Languages, and PLatforms for Implementing and Evaluating algorithms for Distributed systems (ApPLIED'26), July 06–10, 2026, Egham, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3820355.3820395

1 Introduction

The vision of autonomic computing originally proposed systems capable of self-management, adapting their behavior in response to changing conditions with minimal human intervention [12]. Importantly, this vision was not limited to a single centralized controller: it also included the idea of autonomic elements or self-managing units, each equipped with local monitoring, analysis, planning, and execution capabilities [12, 13]. However, this vision did not converge on a compositional abstraction through which such autonomic elements could be exposed, combined, and reused across independently managed providers. Moreover, many large-scale practical realizations of infrastructure management evolved toward centralized or tightly coordinated operational models, in which a limited number of control points retained most of the visibility and authority required to manage the system as a whole.

In practice, large-scale distributed systems evolved toward highly centralized operational models, particularly in cloud computing. Hyperscale providers have achieved unprecedented levels of efficiency, performance, and reliability by consolidating infrastructure, control, and service delivery within unified platforms. While this model has proven effective, it has also led to increasing concentration of capabilities and has raised the barrier to entry for smaller actors seeking to offer competitive services with strong guarantees [25].

At the same time, the emergence of the computing continuum, spanning cloud, edge, and device-level resources, has introduced a significantly more challenging environment [5, 10, 23]. This ecosystem is inherently heterogeneous, dynamic, and distributed across multiple administrative domains. Resources appear and disappear, workloads fluctuate, and application deployments continuously evolve. In such settings, no single entity can maintain complete knowledge or control of the system, making global orchestration increasingly difficult and, in many cases, impractical [5, 10].

This evolution exposes a structural mismatch. The decentralized and heterogeneous nature of the computing continuum requires composition mechanisms that account for partial visibility, multiple administrative domains, and uncertainty. Yet, dominant abstractions remain centered on composing functional execution units such as workflow tasks, service endpoints, or microservices [11, 19]. In these approaches, composition primarily defines how components are connected and executed, while robustness and resilience are introduced at a later stage through deployment strategies such as replication, failover, or multi-provider placement [18, 27]. As a result, resilience remains external to the service abstraction, rather than an intrinsic property of the entities being composed.

Consequently, existing approaches lack a unified abstraction that allows services to be constructed, exposed, and composed as autonomous, uncertainty-aware entities. This limitation becomes particularly critical in decentralized environments, where services must be combined across independent providers without relying on central coordination or complete system knowledge.

This paper proposes a decentralized architectural framework for the computing continuum based on the abstraction of *Composable Autonomic Offerings* (CAOs). This model can be interpreted as a step toward extending the autonomic computing vision with an explicit compositional service abstraction. In this approach, the primitive of composition is not a task or service endpoint, but an autonomic offering: a self-managed entity that encapsulates not only functional behavior, but also explicit trade-offs and a characterization of its expected survivability under changing conditions. CAOs interact by composing their offerings with those of other CAOs, enabling complex services to emerge from purely local decisions without requiring central orchestration.

Contributions. This paper makes the following contributions. First, it introduces a decentralized compositional model in which services are constructed from recursively composable autonomic offerings rather than from functional tasks or workflow nodes. Second, it formalizes the notion of service offerings as multi-objective entities exposed through Pareto-front interfaces, allowing trade-offs to be explicitly represented and propagated across compositions. Third, it incorporates survival semantics into the service abstraction, enabling offerings to express their expected persistence under dynamic conditions and making resilience a first-class, composable property rather than an external deployment concern. Fourth, it proposes a uniform abstraction capable of integrating infrastructure resources, software services, and human-provided capabilities within the same compositional framework.

Together, these contributions define a new architectural perspective in which decentralized service ecosystems can be constructed through the composition of autonomic offerings, enabling more flexible, resilient, and accessible forms of service provisioning across the computing continuum.

2 Related work

A first line of work concerns how distributed infrastructures are managed across the computing continuum. The autonomic computing vision introduced the idea of self-managing systems [12], later operationalized in self-adaptive architectures through reference feedback-loop models such as MAPE-K [13]. More recent work has explored decentralized self-adaptive systems [21], as well as fog and edge resource management and orchestration frameworks for heterogeneous cloud–edge environments [5, 10]. These systems distribute decision-making across nodes or administrative domains in order to improve scalability, reduce latency, and avoid single points of failure. However, in most cases the managed entities remain tasks, services, or deployments, while the control logic is responsible for coordinating their placement, execution, and recovery. In contrast, the present work shifts part of this responsibility into the entities themselves: CAOs expose self-managed offerings that encapsulate both functionality and adaptation logic, allowing coordination to emerge from interactions among autonomous providers rather than from an external control layer.

Prior work on DeepSLOs [4] proposed a decentralized conceptual architecture for continuum management centered on hierarchical SLOs, Bayesian-network dependencies, Markov blankets, and active inference. The present work addresses a related architectural space, but shifts the focus from SLO-centric management to the definition of a recursively composable offering primitive that exposes functionality, trade-offs, survivability, and realization constraints.

A second body of work focuses on service composition, workflow orchestration, and application-graph models [11]. Existing approaches compose functionality in the form of workflow nodes, service endpoints, microservices, or dataflow operators [11, 19], often represented as directed acyclic graphs or more expressive workflow structures [24]. These models are widely used in cloud, edge, and fog systems, including simulation frameworks such as YAFS and iFogSim2 [14, 15]. In these settings, composition typically produces a concrete application structure or deployment plan, after which concerns such as placement, scaling, and fault tolerance are addressed by separate mechanisms [14, 15, 19]. In contrast, the proposed architecture treats a composable offering as the primitive unit: each element being composed is itself a self-managed, persistent entity exposing an interface that includes not only functional behavior but also trade-offs and survivability characteristics. This induces a shift from composing workflow steps, service endpoints, and deployment components to composing autonomous service abstractions.

A third line of work addresses how trade-offs are expressed and optimized in distributed systems. Multi-objective optimization has been widely studied in scheduling and resource allocation, where Pareto-based methods are used to balance latency, throughput, cost, and energy consumption [6, 16, 17, 26]. Similarly, probabilistic models have been used to characterize uncertainty in performance and reliability [1, 3]. However, these techniques are typically used within optimization and orchestration processes, where they guide

decision-making, but are not exposed as part of the service interface. In contrast, the present work elevates these elements to the architectural level: CAOs expose Pareto-front offerings augmented with survival semantics, allowing trade-offs and uncertainty to become first-class elements of composition and interaction among providers.

A fourth area of related work concerns resilience and fault tolerance in distributed service infrastructures [27]. Existing approaches rely on replication, failover, migration, multi-cloud deployment, and decentralized coordination mechanisms to improve robustness and availability [7, 18]. While these techniques are effective, resilience is generally treated as a property of the orchestration strategy or the deployment configuration rather than of the service abstraction itself. In contrast, the proposed architecture embeds resilience directly into the offering semantics: each CAO may construct offerings based on alternative or portfolio-based compositions of upstream providers, and expose survival profiles that characterize the probability of maintaining service objectives over time. This enables resilience to be reasoned about and composed explicitly across layers.

Finally, a substantial body of work has explored service ecosystems, contracts, and market mechanisms, including service-oriented architectures, cloud brokerage, and federated provisioning models [2, 20, 22, 25]. These approaches facilitate interoperability and multi-provider integration, but often rely on centralized or semi-centralized intermediaries for discovery, coordination, or brokerage, and assume that providers expose relatively stable service endpoints backed by their own infrastructure and operational capabilities. As a result, smaller actors may face significant barriers to entry when attempting to offer competitive services with strong performance and reliability guarantees [25]. Federated provisioning models are particularly relevant here, since they already support the combination of resources from multiple providers and may therefore resemble a kind of virtual provider [22, 25]. Recent work on IoT–edge–cloud continuum systems highlights the need for federated management layers to coordinate resources across heterogeneous providers [9]. However, they remain primarily resource-centric: what is aggregated is infrastructure capacity, while resilience and service semantics are still largely determined by the orchestration or deployment layer. In contrast, the architecture proposed in this paper shifts part of the burden from ownership of end-to-end infrastructure to composition of upstream offerings. Complex services can be constructed recursively from offerings that already expose trade-offs, constraints, and survivability, allowing providers to participate by offering specialized capabilities while higher-level services emerge through aggregation and local risk management. This opens the possibility of more decentralized and accessible service ecosystems with new roles such as aggregators, resilience providers, and hybrid human-machine service operators. This hybrid direction also connects to Social Compute Units, which model human-provided and software services as programmable composite systems [8].

Taken together, these lines of work address complementary aspects of distributed service systems, but typically treat composition, resilience, optimization, and market interaction as separate layers. The CAO abstraction unifies these concerns by making them intrinsic to the composable service primitive itself.

3 Architectural Requirements

The computing continuum is increasingly heterogeneous, decentralized, and dynamic, yet dominant abstractions for service construction and management remain centered on centralized orchestration and the composition of functional units. As a result, existing models often struggle to support composition across independently managed providers under uncertainty without relying on broad coordination assumptions or levels of visibility that may be difficult to sustain in open continuum environments [5, 9, 21].

Addressing this gap requires a service abstraction that can be exposed, composed, and adapted locally while still enabling coordination across a decentralized environment. This section outlines the key requirements for such an abstraction, focusing on offerings and their composition rather than on specific implementation choices.

Decentralized and autonomic management. The architecture must avoid reliance on global control points or centralized orchestration. Each participating entity should manage its own behavior from local information and objectives, while global behavior emerges from composition of local decisions.

Composable offering abstraction. The primitive of composition must go beyond functional tasks, service endpoints, or workflow nodes. It should expose heterogeneous capabilities as composable offerings that can be combined and recursively re-exposed at higher levels, spanning infrastructure resources, software services, and human-backed operations.

Explicit representation of trade-offs. Offerings must expose not only functionality but also trade-offs across relevant quality dimensions, such as latency, cost, energy, reliability, or accuracy. The architecture should support multiple alternative offerings and propagate these trade-offs through composition rather than burying them entirely inside internal optimization procedures.

Uncertainty-aware offering semantics. In dynamic and partially observable environments, offering behavior cannot be captured by deterministic guarantees alone. The architecture must therefore represent uncertainty, including the expected persistence of service objectives over time, so that offerings can be reasoned about and composed under incomplete knowledge and changing conditions.

Resilience by composition. Resilience should not be treated only as an external deployment strategy applied after service construction. The architecture must support building more robust offerings from less robust ones through alternatives, fallbacks, diversification, or portfolio-based realizations, so that robustness can be created and propagated across layers.

Recursive and open participation. The architecture should support the participation of diverse actors, from infrastructure providers to software and human-backed services. By allowing partial or specialized capabilities to be exposed and composed into higher-level offerings, it lowers the barrier to participation for actors that do not control a complete service stack.

Continuous adaptation. Because the computing continuum is dynamic, offerings must be continuously revised in response to

changing demand, resource availability, and environmental conditions, without requiring centralized coordination or system-wide recomputation.

4 Architecture

This section refines the CAO abstraction by describing the internal objects and processes through which a CAO constructs, evaluates, exposes, and maintains its offerings. Let O_i denote the set of candidate offerings currently maintained by CAO i. Each offering $o \in O_i$ is represented as a tuple $o = (G, s, \sigma, r)$, where G captures composition structure, s is a multi-objective SLO vector, $\sigma(t)$ is the probability that the advertised SLO commitments remain satisfied up to time t, and r is a realization envelope specifying the admissible conditions under which the offering may be instantiated. The resilience example in Figure 2 provides a simple instance of this representation: the compositional structure G encodes a fallback relation between two upstream offerings, while s, σ, and r describe the resulting trade-offs, survival profile, and admissible realization conditions.

These elements form a coupled representation rather than independent descriptors. Together, G and r induce a space of admissible realizations $\mathcal{R}(G, r)$, corresponding to possible instantiations under specific deployment conditions and upstream bindings. The SLO vector s characterizes the commitments induced by those realizations, while the survival model $\sigma(t)$ represents the probability that the advertised SLO commitments remain satisfied up to time t. For non-instantiated offerings, σ is estimated from upstream information, structural assumptions encoded in G, and admissible realizations defined by r, using analytical, simulation-based, or historically calibrated models. For instantiated offerings, the same structure can be progressively refined from runtime observations and agreement outcomes.

Rather than prescribing a specific estimation method, the architecture constrains the role of σ. Any admissible instantiation must: (i) represent a time-indexed probability of maintaining the advertised SLO vector, (ii) remain compatible with the compositional structure G, so that uncertainty can be propagated across sequential, parallel, and alternative realizations, and (iii) support conservative estimation, especially in early stages when empirical evidence is limited.

These requirements admit models inspired by reliability theory, hierarchical or graphical survival models, and simulation-based estimators over the realization space induced by (G, r). The architecture does not commit to one modeling approach but requires consistency with the compositional semantics of the offering and support for incremental refinement as new evidence becomes available.

In practice, σ may combine analytical, simulation-based, and observational components. A baseline estimate $\sigma_{model}(t)$ can be derived from G and upstream offerings, confronted with simulation-based estimates $\sigma_{sim}(t)$ over sampled realizations from $\mathcal{R}(G, r)$, and, when available, observational estimates $\sigma_{obs}(t)$. A conservative choice is to expose

$$\sigma(t) = \min(\sigma_{\text{model}}(t), \sigma_{\text{sim}}(t), \sigma_{\text{obs}}(t)),$$

although other calibration or fusion strategies are also compatible provided they preserve the architectural role of σ as a conservative, revisable estimate.

In dependability terms, $\sigma(t)$ can be interpreted as an SLO-level survivability curve: it estimates the probability that the offering can continue satisfying its advertised SLO vector up to time t, rather than only the probability that an individual component remains available. Classical reliability, availability, or survivability models may therefore be used internally to estimate $\sigma(t)$, while the exposed quantity remains a conservative, revisable estimate at the level of the CAO abstraction.

A similar issue arises for the SLO vector s: although its dimensions are fixed by the CAO interface or domain, the threshold values of a newly generated offering must still be derived from G, r, upstream commitments, and available analytical, simulation-based, or observational evidence. As with σ, these estimates should initially be conservative.

The set O_i therefore represents the CAO's internal candidate offering space, while the interface exposed to other CAOs is its currently relevant non-dominated subset, the Pareto Front Offering (PFO), under the comparison semantics adopted by the implementation. The following subsections refine how offerings are represented, generated, maintained, and adapted over time.

4.1 Offering Representation, Composition Operators, and Realization Constraints

A central requirement of the proposed architecture is that each Composable Autonomic Offering (CAO) maintains an explicit representation of the offerings it may expose. The primary object to be represented is not merely an application, nor a concrete deployment plan, but the offering itself. This is necessary because an offering may depend on linked upstream offerings, inherit constraints from them, combine several of them into a larger capability, and admit multiple possible realizations with different trade-offs and survival properties.

This need for explicit structure is strongest when composition is present. If an offering has no upstream dependencies, its internal structure need not be exposed at the same granularity; the required level of detail is therefore a design variable under the provider's control.

The representation layer must satisfy three requirements. First, it must represent the compositional structure of the offering, including how local capabilities and linked upstream offerings are combined into a higher-level capability, and must support recursive composition so that offerings can themselves appear as constituents of larger offerings. Second, it must support a small family of composition operators through which resilience can be constructed rather than merely assumed, including at least sequential, parallel, fallback, and portfolio-style combinations. These operators are not purely structural, but are associated with selection, switching, or allocation logic that determines how alternative realizations are activated under changing conditions. The architecture does not prescribe the form or complexity of this logic. Instead, it constrains only its role: to select or construct a valid realization from the available alternatives under the current context, in a manner consistent with the compositional structure and realization constraints. As a

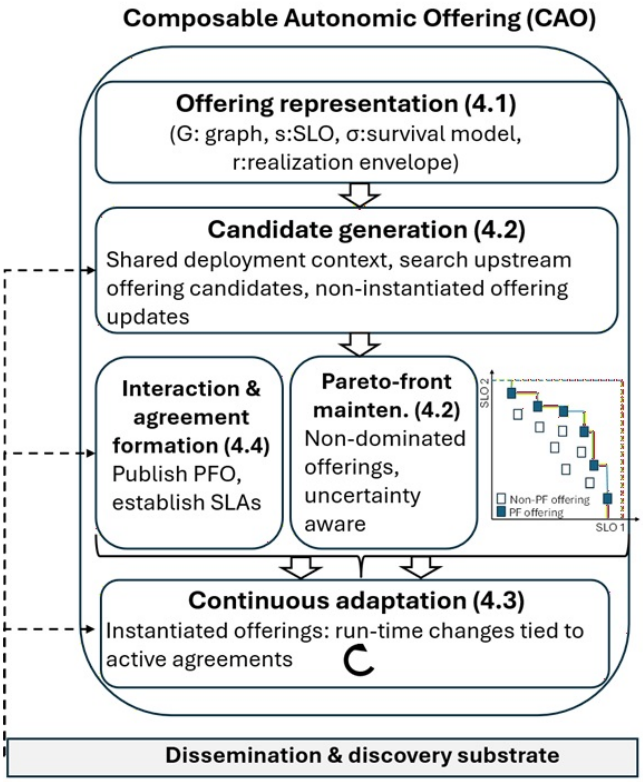


Figure 1: Internal decision and adaptation loop of a composable autonomic offering (CAO). A CAO generates candidate offerings through local and upstream composition, evaluates their multi-objective performance and survival characteristics, and maintains a Pareto-front offering (PFO) that constitutes its exposed interface. This interface represents a set of non-dominated, probabilistic service commitments rather than fixed guarantees. The CAO continuously adapts its candidate set and exposed frontier in response to changes in demand, upstream dependencies, and runtime observations. Interaction with other CAOs occurs through the exchange of PFOs over an external dissemination and discovery substrate, which enables decentralized composition without requiring global coordination.

result, these operators affect not only functionality, but also the trade-offs and survival properties of the resulting offering by introducing alternative realization paths. Third, it must make explicit the realization constraints under which an offering remains valid as advertised, rather than prematurely collapsing it into one fixed realization. In the fallback case illustrated in Figure 2, the operator does not simply denote an edge in a graph: it specifies that an alternative upstream realization may be activated when the currently selected realization no longer satisfies the survival or realization conditions of the offering.

Realization constraints and shared deployment context. The compositional representation G is not sufficient by itself to establish whether an offering is realizable. In a heterogeneous continuum, lower-level offerings must also expose the admissible conditions under which they can participate in a larger composition. The realization envelope r therefore captures the resource, placement, communication, policy, and dependency constraints under which the offering remains valid as advertised. These constraints describe the conditions under which an offering can be validly instantiated, and are distinct from the SLO commitments captured in s, which characterize the performance of the offering once deployed. In general, quantitative and comparable properties (e.g., latency, cost, energy, or even certain capability indicators) are expressed in s, where they can be optimized and propagated through composition. By contrast, the realization envelope r captures admissibility and compatibility constraints that determine whether a realization is feasible at all. These include, for example, shared identity or trust domains, access permissions, protocol or API compatibility, co-location or anti-colocation requirements, shared failure-domain exclusions, regulatory constraints, or other conditions that cannot be naturally expressed as optimizable objectives. In this sense, r defines the admissible realization space of the offering, while s characterizes performance within that space.

More precisely, each realization envelope r_i defines the set of deployment contexts under which the corresponding offering o_i can be validly instantiated. Given a composed offering depending on upstream offerings $o_1, \ldots, o_n$, realizability depends on whether their realization constraints are jointly compatible under the composition structure G. This can be expressed as $\exists x \in \left(\bigcap_{i=1}^{n} r_i\right) \cap C(G)$, where $C(G)$ denotes the additional constraints imposed by the composition itself, such as communication requirements or placement relations between components.

In this sense, the realization envelope of a composed offering is the admissible deployment context induced by the upstream constraints and the additional requirements introduced by the composition structure, so candidate realizations are evaluated for their SLO vector and survival profile only after this compatibility check succeeds.

4.2 Candidate Generation and Pareto-Front Maintenance

This stage describes how a CAO constructs and maintains its local set of candidate offerings from its own capabilities and visible upstream offerings, assigns them provisional multi-objective and survival estimates, and exposes the non-dominated subset as its Pareto Front Offering (PFO). Architecturally, candidate construction and frontier maintenance form a single local decision loop, continuously driven by changes in expected demand, upstream availability, and observed evidence.

The constructive problem faced by a CAO is not merely to enumerate feasible offerings, but to identify and maintain those feasible offerings that may belong to its locally relevant Pareto frontier. Given a target offering specification, the CAO must therefore explore a constrained space of admissible realizations induced by G and r.

The architecture does not require one specific solution procedure. Candidate generation may rely on incremental refinement and pruning, constrained multi-objective optimization, stochastic or simulation-guided search, or hybrid strategies. What matters

architecturally is that generation, feasibility checking, and frontier maintenance remain coupled, since the goal is to maintain relevant non-dominated offerings rather than arbitrary feasible ones.

Once a candidate offering has been specified to the point that its composition structure G and realization envelope r determine an admissible realization space, the CAO derives provisional estimates of its SLO vector s and survival model σ from the relevant upstream offering components (in particular s_i, σ_i, and, where needed, r_i), together with any locally available analytical models, simulation-based estimates, and observational estimates. For instance, a resilience CAO may generate a candidate by selecting two compatible upstream offerings, checking whether their realization envelopes can coexist under the fallback composition, and then deriving a provisional SLO vector and survival profile for the composed offering.

PFO maintenance is event-driven. Reevaluation is triggered by changes affecting the admissible realization space or the estimated characteristics of candidate offerings, including the appearance or disappearance of upstream offerings, updates in their SLO or survival estimates, and demand changes that alter capacity, performance, or reliability assumptions. The CAO then revises affected candidates and refreshes Pareto membership by discarding dominated offerings and promoting newly relevant ones.

4.3 Continuous Adaptation of Instantiated Offerings

Once an offering has been instantiated, the CAO enters a continuous adaptation phase in which it must preserve the validity of the offering under changing conditions. At this stage, the relevant object is an instantiated offering $o = (G, s, \sigma, r)$ with a currently selected realization $\rho \in \mathcal{R}(G, r)$. The purpose of this phase is to connect the offering representation introduced above with runtime adaptation: the same compositional structure and realization constraints used to define the offering also guide how an instantiated offering is re-evaluated and reconfigured under changing conditions.

Adaptation is event-driven and centered on the evolution of the survival model of the instantiated offering. The local control loop of the CAO is triggered when the estimated survival profile $\sigma(t)$ falls below a predefined admissibility threshold, indicating a high probability that the advertised SLO vector s may not be sustained. Such degradation may result from changes in upstream offerings, violations of realization constraints, observed deviations between predicted and actual behavior, or shifts in expected operating conditions. In response, the CAO re-evaluates the instantiated offering through the operator semantics encoded in G, updates the corresponding estimates of (s, σ), and determines whether the current internal configuration should be preserved, switched, rebalanced, or otherwise adapted using the resilience structures already available in the offering representation.

The control problem is therefore to use the operator semantics encoded in G to adapt the internal configuration of the instantiated offering so as to preserve the advertised SLO vector as much as possible while maximizing the achievable survival profile under the current conditions. In this sense, adaptation is not an external mechanism added after composition, but the runtime continuation of the same offering semantics used during candidate generation. The structure encoded in G may include resilience strategies that define alternative admissible configurations under changing conditions.

Fallback, portfolio, and diversification operators are representative rather than exhaustive. In a fallback composition, for instance, G encodes an admissible switch from the selected upstream realization to an alternative one; adaptation is the runtime activation of this switch when the advertised SLO vector and realization envelope can still be preserved under updated survival estimates.

The architectural goal is to absorb variability within the local adaptation loop of each CAO, so that changes are handled locally before propagating to dependent offerings. System-level resilience emerges from local adaptation over composable offerings, rather than from centralized orchestration.

4.4 Interaction and Information Exchange Model

CAOs interact through the continuous exchange of offering information and through the establishment of agreements over selected offerings. Each CAO observes the Pareto-front offerings (PFOs) published by other CAOs and updates its local knowledge state accordingly. Symmetrically, it publishes revisions of its exposed frontier as local conditions evolve, as well as status updates on already instantiated offerings.

The dissemination of offering information is assumed to rely on an external infrastructure, such as a peer-to-peer overlay, gossip-based mechanisms, or distributed registries. This infrastructure is treated as a commodity substrate and is not part of the architectural contribution of this work.

In addition to passive observation, CAOs perform active interactions to establish and maintain agreements corresponding to selected dependencies. Instantiating a local offering often requires a cascade of upstream commitments, since the selected realization may depend on multiple upstream offerings that must themselves be reserved or instantiated consistently. Without transactional guarantees, such cascades may lead to partial commitment, duplicate instantiation, over-allocation of shared upstream resources, or inconsistent dependency state when concurrent decisions occur. For this reason, agreement establishment and subsequent reconfiguration (e.g., fallback activation, provider substitution, or portfolio updates) should be supported by atomic operations over a commodity zero-trust substrate, such as blockchain-backed or otherwise transactional coordination infrastructure.

The content of each agreement is not arbitrary, but is derived from the selected offering under the risk and economic policy of the CAO. In particular, the offering's survival curve at the time of agreement provides the reference for the risk profile that can be exposed contractually, while the precise SLA terms depend on how the CAO's owner chooses to price, bound, or hedge that risk. After agreement formation, the survival profile may evolve under changing conditions, and adaptation may activate fallback or other admissible realizations to preserve the agreed behavior without altering the established terms. The interaction model therefore combines continuous asynchronous information flow with discrete agreement operations that materialize the effective dependency structure of composed offerings.

5 Use Cases

Figure 2 summarizes representative roles that a CAO may play in a decentralized ecosystem. These examples illustrate how the same abstraction supports different forms of value creation.

Entrepreneur CAO. An entrepreneurial CAO constructs a competitive offering by composing upstream offerings without owning a full infrastructure stack. Unlike a traditional broker, it performs the composition itself and exposes the resulting service as its own offering, while a broker only matches consumers to existing providers without constructing a new service. This shifts value creation from infrastructure ownership to composition, allowing small actors to create competitive services by recombining upstream trade-offs, survival profiles, and resilience structures.

Resilience CAO. A resilience-oriented CAO constructs offerings whose primary value is improved robustness under uncertainty rather than new base functionality. It observes multiple upstream offerings providing similar capabilities and composes them using resilience operators such as fallback, diversification, or portfolio-style combinations. As illustrated in Figure 2, a resilience CAO may build an offering with composition structure $G = \text{fallback}(o_1, o_2)$ over two compatible upstream offerings. The resulting offering is not characterized only by its functional behavior, but also by a realization envelope r capturing the conditions under which fallback remains admissible, such as domain, placement, or failure-domain constraints, and by a survival estimate derived from the composition structure, upstream survival information, and available analytical, simulation-based, or observational evidence. In this way, the resilience CAO can expose an offering $o = (G, s, \sigma, r)$ whose value lies in reducing dependence on a single provider or realization path. This may increase cost or other trade-offs relative to an individual dependency, but it allows resilience to be constructed explicitly through composition rather than introduced only as a reactive runtime mechanism.

Edge AI Service CAO. A service-oriented CAO may implement a distributed application, such as an edge AI pipeline spanning devices, edge nodes, and cloud resources. The offering corresponds to a multi-stage processing chain (e.g., data acquisition, preprocessing, inference, and aggregation), naturally represented as a compositional structure G. The realization envelope captures the conditions under which this pipeline can be instantiated across heterogeneous resources, including locality, latency, and compatibility constraints induced by upstream bindings. In this way, the CAO exposes a composite service whose implementation depends on multiple upstream capabilities and admits several valid configurations, leading to a multi-objective Pareto front rather than a single fixed deployment.

Human Labor CAO. The CAO abstraction can be naturally extended beyond software and infrastructure to human-provided capabilities. In this setting, human labor units may expose offerings such as annotation, moderation, validation, or exception handling, and participate in composition alongside automated services within the same decentralized ecosystem. While their internal execution may rely on richer workflow formalisms, the CAO provides a unifying interface for composition and management.

This example also highlights a limitation of the current representation layer. Human-centered processes often require control-flow constructs beyond data dependencies, as captured by workflow models such as BPM or Petri nets, suggesting that a fully unified compositional framework may require richer models than those considered here.

Summary. Taken together, these use cases show that the proposed architecture is not tied to a specific service layer or operational role. A CAO may act as a leaf provider, a composite service provider, a resilience-oriented aggregator, an entrepreneurial market entrant, or a hybrid human–machine service node. This flexibility underpins the central claim of the paper: that decentralized composition of probabilistic multi-objective offerings can enable a more dynamic, resilient, and accessible computing continuum ecosystem.

6 Discussion and Conclusion

Many of the potential economic benefits of the computing continuum remain under-realized because the diversity of capabilities that the continuum can create is not yet matched by equally mature approaches for composing, exposing, and managing them. This is one of the structural motivations behind the present work. While centralized and hyperscale service models have achieved high levels of predictability, efficiency, and operational maturity, they have also concentrated control, raised the barrier to entry for smaller actors, and reduced the pressure for integration technologies to evolve beyond the hyperscaler envelope. As a result, much of the current orchestration and integration machinery remains best suited to relatively unified platform environments, rather than to the broader diversity, heterogeneity, and decentralized ownership structure that the computing continuum is likely to intensify. This structural gap is precisely what the proposed CAO architecture is intended to address. The proposed architecture is not intended to replace all existing service provisioning models: highly centralized platforms may remain preferable in regimes where predictability, tight coordination, and uniform operational control are more important than openness or decentralized participation. The CAO model is instead aimed at settings in which heterogeneity, distributed ownership, and dynamic participation make centralized orchestration increasingly brittle or exclusionary.

A central aspect of the scope of this paper is that it defines the architectural abstraction and the main interfaces of the CAO model, rather than a single fixed implementation. In Sections 3 and 4, the paper specifies the core elements that a CAO must expose and manage: an explicit offering representation centered on the tuple (G, s, σ, r), a notion of admissible realization contexts, local generation of offering candidates, Pareto-front maintenance, interaction through offering exchange and agreement formation, and continuous adaptation under changing conditions. Together, these components define the architectural structure of a CAO and the local decision processes through which higher-level services may emerge from decentralized composition.

At the same time, the proposed architecture intentionally leaves open the precise technological realization of several of these components. The representation layer may be instantiated through

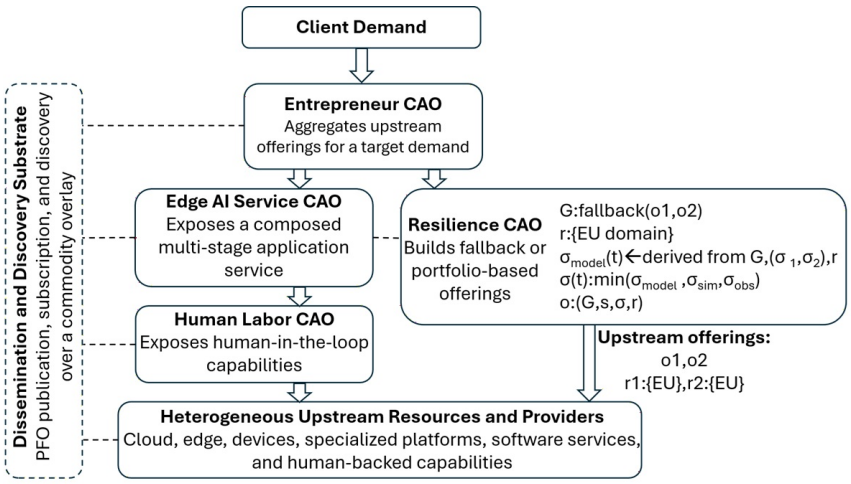


Figure 2: Illustrative CAO ecosystem with a zoomed-in resilience CAO example. An entrepreneur CAO aggregates visible upstream offerings for a target demand. Representative downstream roles include an edge AI service CAO, which exposes a composed multi-stage application service, and a human labor CAO, which extends the abstraction to human-in-the-loop capabilities. The resilience CAO constructs fallback- or portfolio-based offerings from compatible upstream offerings; in the example shown, it builds a fallback composition $G = \text{fallback}(o_1, o_2)$, derives a model-based survival estimate $\sigma_{\text{model}}(t)$ from G, upstream survival information, and the realization envelope r, and exposes a conservative survival estimate $\sigma(t) = \min(\sigma_{\text{model}}(t), \sigma_{\text{sim}}(t), \sigma_{\text{obs}}(t))$. The resulting offering is exposed as $o = (G, s, \sigma, r)$.

dataflow-based models, richer workflow-oriented formalisms, or hybrid approaches that bridge both. Candidate generation and maintenance may rely on analytical methods, heuristics, simulation-based evaluation, learned estimators, or other search and optimization techniques. Likewise, agreement formation and settlement may be supported by different coordination substrates depending on the target environment. This openness is not a weakness of the proposal, but one of its advantages: by focusing on the architectural interface rather than prematurely fixing one detailed implementation, the model remains compatible with multiple technological perspectives and leaves broad room for the community to explore CAO realizations adapted to different domains, levels of the digital-twin spectrum, and target classes of offerings. This makes the proposal easier to tackle incrementally, since different communities may instantiate the same architectural principles using the representations, optimization methods, and runtime substrates most natural to their expertise and target applications.

Resilience itself also deserves discussion. A natural concern is that greater decentralization may protect against failures or attacks on centralized services while simultaneously sacrificing some degree of global coordination efficiency or predictability. The proposed architecture does not eliminate this trade-off. Rather, it shifts the architectural emphasis: instead of relying primarily on centralized coordination to achieve robustness, it enables resilience to be constructed through composition. Some offerings may already carry strong survival characteristics, but downstream CAOs may also build more robust offerings from weaker upstream components by combining alternatives, fallback paths, diversification, or portfolio-based realizations. In this sense, the architecture provides a different route to robustness, one that is compatible with decentralized and multi-provider environments, even if it may not match the global coordination efficiency of tightly controlled centralized systems. The practical balance between these regimes remains an open empirical question.

A further aspect concerns the underlying infrastructure required to support the proposed model. The CAO abstraction assumes the existence of several enabling substrates that are not specified in detail in this work. First, a dissemination and discovery substrate is required to allow offerings to be published, updated, and retrieved across a decentralized environment, ensuring that participants can access relevant and sufficiently up-to-date information. Second, mechanisms for agreement formation, accounting, and settlement are needed to support interactions among autonomous providers, including the ability to prevent conflicting commitments and ensure consistent execution of agreements across multiple parties. Third, the model relies on the availability of observational data to estimate and continuously update the survival characteristics of offerings. The quality of these estimates will depend on access to runtime evidence, monitoring capabilities, and historical data, which may vary across environments.

Interestingly, CAOs are not only autonomic but also sovereignty-preserving: each participant retains control over how its offerings are composed through G, under which deployment, policy, and compatibility conditions they remain admissible through r, which non-dominated alternatives it chooses to expose through its maintained Pareto Front Offering (PFO), and which offerings it instantiates, revises, or withdraws through its local agreement and adaptation processes. This allows providers to encode their own technical and entrepreneurial biases directly into the offerings they publish, rather than delegating these decisions to a central orchestrator.

Finally, the CAO abstraction also points toward a broader cyber-physical management perspective. Because the architecture is defined at the level of offerings rather than at the level of a specific software or infrastructure substrate, it may provide a basis for integrating infrastructure management, software services, human-backed operations, and, eventually, agent societies that both offer and consume services with progressively less human intervention. This broader perspective is still preliminary, but it helps clarify a more general point: the computing continuum requires a new compositional primitive—one that combines functionality, trade-offs, and survivability—if decentralized ecosystems are to become both manageable and economically generative. The CAO model is proposed as a step in that direction.

Acknowledgments

This work and equipment has been supported by CNS2023-144359 financed by MICIU/AEI/10.13039/501100011033 and the European Union NextGeneration EU/PRTR.

References

[1] Omid Alipourfard, Hongqiang Harry Liu, Jianshu Chen, Shivaram Venkataraman, Minlan Yu, and Ming Zhang. 2017. CherryPick: Adaptively unearthing the best cloud configurations for big data analytics. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. 469–482.

[2] Rajkumar Buyya, Satish Narayana Srirama, Giuliano Casale, Rodrigo Calheiros, Yogesh Simmhan, Blesson Varghese, Erol Gelenbe, Bahman Javadi, Luis Miguel Vaquero, Marco AS Netto, et al. 2018. A manifesto for future generation cloud computing: Research directions for the next decade. *ACM computing surveys (CSUR)* 51, 5 (2018), 1–38.

[3] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose, and Rajkumar Buyya. 2011. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience* 41, 1 (2011), 23–50.

[4] Victor Casamayor Pujol, Boris Sedlak, Yanwei Xu, Praveen Kumar Donta, and Schahram Dustdar. 2024. Deepslos for the computing continuum. In *Proceedings of the 2024 Workshop on Advanced Tools, Programming Languages, and PLatforms for Implementing and Evaluating algorithms for Distributed systems*. 1–10.

[5] Breno Costa, Joao Bachiega Jr, Leonardo Rebouças De Carvalho, and Aleteia PF Araujo. 2022. Orchestration in fog computing: A comprehensive survey. *ACM computing surveys (CSUR)* 55, 2 (2022), 1–34.

[6] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.

[7] Robert Dilworth. 2024. Advancements and Challenges in Cloud Computing: Multi-Cloud Management, Security, and AI-Driven Threat Mitigation. In *Proceedings of the 2024 7th Artificial Intelligence and Cloud Computing Conference*. 639–645.

[8] Schahram Dustdar and Kamal Bhattacharya. 2011. The social compute unit. *IEEE Internet Computing* 15, 3 (2011), 64–69.

[9] Panagiotis Gkonis, Anastasios Giannopoulos, Panagiotis Trakadas, Xavi Masip-Bruin, and Francesco D'Andria. 2023. A survey on IoT-edge-cloud continuum systems: Status, challenges, use cases, and open issues. *Future Internet* 15, 12 (2023), 383.

[10] Cheol-Ho Hong and Blesson Varghese. 2019. Resource management in fog/edge computing: a survey on architectures, infrastructure, and algorithms. *ACM computing surveys (csur)* 52, 5 (2019), 1–37.

[11] Ward Jaradat, Alan Dearle, and Adam Barker. 2013. A dataflow language for decentralised orchestration of web service workflows. In *2013 IEEE Ninth World Congress on Services*. IEEE, 13–20.

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

[13] Christian Krupitzer, Felix Maximilian Roth, Sebastian VanSyckel, Gregor Schiele, and Christian Becker. 2015. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing* 17 (2015), 184–206.

[14] Isaac Lera, Carlos Guerrero, and Carlos Juiz. 2019. YAFS: A simulator for IoT scenarios in fog computing. *IEEE Access* 7 (2019), 91745–91758.

[15] Redowan Mahmud, Samodha Pallewatta, Mohammad Goudarzi, and Rajkumar Buyya. 2022. Ifogsim2: An extended ifogsim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *Journal of Systems and Software* 190 (2022), 111351.

[16] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, Zili Meng, and Mohammad Alizadeh. 2019. Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM special interest group on data communication*. 270–288.

[17] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. 2017. A survey on mobile edge computing: The communication perspective. *IEEE communications surveys & tutorials* 19, 4 (2017), 2322–2358.

[18] Rafael Moreno-Vozmediano, Rubén S Montero, Eduardo Huedo, and Ignacio Martín Llorente. 2018. Orchestrating the deployment of high availability services on multi-zone and multi-cloud scenarios. *Journal of Grid Computing* 16, 1 (2018), 39–53.

[19] Samodha Pallewatta, Vassilis Kostakos, and Rajkumar Buyya. 2023. Placement of microservices-based IoT applications in fog computing: A taxonomy and future directions. *Comput. Surveys* 55, 14s (2023), 1–43.

[20] Mike P Papazoglou. 2003. Service-oriented computing: Concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003.* IEEE, 3–12.

[21] Federico Quin, Danny Weyns, and Omid Gheibi. 2021. Decentralized self-adaptive systems: A mapping study. In *2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 18–29.

[22] Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio Martín Llorente, Rubén Montero, Yaron Wolfsthal, Erik Elmroth, Juan Caceres, et al. 2009. The reservoir model and architecture for open federated cloud computing. *IBM Journal of Research and Development* 53, 4 (2009), 4–1.

[23] Boris Sedlak, Víctor Casamayor Pujol, Ildefons Magrans de Abril, Praveen Kumar Donta, Adel N. Toosi, and Schahram Dustdar. 2026. Service Orchestration in the Computing Continuum: Structural Challenges and Vision. doi:10.48550/arXiv.2602.15794 arXiv:2602.15794 [cs].

[24] Boris Sedlak, Víctor Casamayor Pujol, Praveen Kumar Donta, and Schahram Dustdar. 2024. Diffusing High-level SLO in Microservice Pipelines. In *2024 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. Shanghai, China, 11–19. doi:10.1109/SOSE62363.2024.00008

[25] Adel Nadjaran Toosi, Rodrigo N Calheiros, and Rajkumar Buyya. 2014. Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Computing Surveys (CSUR)* 47, 1 (2014), 1–47.

[26] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of the tenth european conference on computer systems*. 1–17.

[27] Thomas Welsh and Elhadj Benkhelifa. 2020. On resilience in cloud computing: A survey of techniques across the cloud domain. *ACM computing surveys (CSUR)* 53, 3 (2020), 1–36.

Received 30 April 2026; revised 30 April 2026; accepted 30 May 2026