Closed-Loop Network Automation: Architectural Principles from Catalyst Center-Scale Systems

Manevannan Ramasamy
Software Engineering Senior Technical Leader, Milpitas, California, United States of America
manevannan3@gmail.com

Abstract

Enterprise network controllers must manage large, heterogeneous environments of devices, clients, locations, applications, and services while preserving explainability, reliability, and change safety. This paper proposes a vendor-neutral architecture informed by public descriptions of Cisco DNA Center and established work on intent-based networking, telemetry, assurance, and autonomic networking. The architecture organizes closed-loop operation into six interacting layers: data acquisition and context, correlation and diagnosis, intent and policy, workflow orchestration, outcome verification, and external integration. It also defines hierarchical control scopes, evidence aggregation, action classes, and control budgets to limit concurrent remediation and reduce operational risk. Cisco DNA Center provides product context; this paper does not claim access to proprietary implementation details. The paper discusses proposed assessment dimensions, including outcome coverage, telemetry freshness, diagnostic accuracy, control-loop convergence, failover resilience, and operator understanding. The framework offers a product-aware, vendor-neutral basis for analyzing scalable enterprise controller automation and assurance.

Keywords: Cisco DNA Center, Closed-Loop Automation, Enterprise Network Controller, Assurance, Intent-Based Networking, Scale

1. Introduction

Enterprise networks combine switches, routers, wireless infrastructure, clients, applications, services, and multiple management domains. As these environments grow and become more heterogeneous, it becomes harder to configure devices safely while meeting demands for rapid service delivery, reliability, and assurance. Network management therefore increasingly uses automation to monitor network state, infer conditions, coordinate actions, and verify outcomes. These needs motivate controller architectures that support closed-loop operation beyond remote configuration alone.

Autonomic networking introduced principles such as self-configuration, self-optimization, self-healing, and self-protection [3]. In an autonomic system, management functions respond to changes in network conditions rather than relying only on fixed procedures. Intent-based approaches add a high-level description of desired network behavior, separating operational goals from low-level configuration [3]. A controller may achieve the same goal through different actions depending on topology, device capabilities, workload, and current network state.

Zero-touch network and service management extends these ideas through service-oriented management. The ETSI Zero-touch Network and Service Management (ZSM) reference architecture describes functional components and end-to-end management interfaces for networks and services [1]. Its modular design supports scalability, interoperability, and coordination across domains and technologies. For large enterprise controllers, however,

functional decomposition alone does not determine what evidence is reliable, how observations should be correlated, or when an automated action is permitted.

Telemetry is a central input to a closed-loop controller. It provides information about network activity that can support diagnosis and operational decisions [4]. Its value depends not only on the volume of measurements but also on freshness, provenance, identity, topology, and context. Large quantities of disconnected telemetry may provide little diagnostic value. Enterprise automation must therefore associate observations with relevant evidence and context before using them to diagnose conditions or select actions.

Intent-based networking describes desired network behavior at a higher level of abstraction. It allows operators or external systems to express goals without specifying every device-level implementation detail [5]. A controller can map those goals to policies, workflows, and configuration actions suited to the environment. Intent can also serve as a desired-state reference throughout a continuous assurance loop, rather than only during initial configuration.

Autonomic networking, zero-touch management, telemetry, and intent provide concepts for closed-loop enterprise control. A generalized loop observes the system, adds context, diagnoses conditions, selects an authorized action, executes it, and verifies the result. Observation records the current state; contextualization associates facts with relevant objects and relationships; diagnosis identifies deviations; decision-making applies policy; execution performs approved actions; and verification checks whether the intended state was achieved. This model extends event-triggered automation toward continuous, evidence-based operations.

Large environments may contain many control loops operating at device, site, service, application, and enterprise levels. An observation or action in one loop can affect another, creating conflicting or excessive remediation. Controllers must therefore account for loop hierarchy, action scope, concurrency, evidence quality, and change risk—not only individual alerts. Operators also need to understand why an action was proposed or executed, what evidence supported it, and whether the resulting state meets the original objective.

This article proposes a vendor-neutral architecture informed by publicly documented controller capabilities and established networking concepts. The framework connects data collection and context, correlation and diagnosis, intent and policy, workflow execution, outcome verification, and external integration. It also proposes hierarchical control scopes, evidence aggregation, action classes, and control budgets to limit concurrent remediation. These design concepts aim to support safer and more responsive network changes.

The framework treats an enterprise controller as a closed-loop system in which evidence, intent, execution, and verification are connected. It proposes assessment dimensions that include outcome coverage, telemetry freshness, diagnostic quality, convergence, failover behavior, and operator understanding. The goal is to provide a basis for analyzing scalable and manageable enterprise network automation.

2. Related Work

Enterprise network automation draws on research and standards concerning autonomic networking, intent-based networking (IBN), telemetry, assurance, service orchestration, and

zero-touch management. These areas provide conceptual foundations for the architecture proposed here.

ETSI ZSM 002 describes management domains, policy-bounded closed-loop operation, and conditions for human oversight, escalation, delegation, and coordination [1]. Its 2019 version explicitly leaves the mechanism by which closed loops coordinate with one another outside its scope. Building on that foundation, this article proposes an enterprise-controller arrangement that links contextual evidence, action classes, control budgets for each scope, and explicit outcome verification. The proposed contribution is this specific integration and its assessment criteria; closed-loop management and policy governance are established foundations, not introduced by this article.

Cisco DNA Center documentation describes APIs for accessing controller and network-management capabilities [2]. These APIs illustrate how an integration layer can let applications and orchestration systems obtain network information or request management processes. API exposure alone does not constitute a complete closed-loop control system. Reliable automation also requires policy interpretation, contextual information, action selection, execution safeguards, and follow-up verification.

Autonomic networking research describes systems that monitor their own state, analyze collected data, generate responses, and carry out some responses without human intervention [3]. These principles align with the feedback loops required for enterprise network automation. The general autonomic model does not, by itself, address the varied business objectives, telemetry, management systems, and remediation constraints found in heterogeneous enterprise environments.

Network telemetry provides mechanisms and architectural considerations for collecting and moving operational data [4]. Timely telemetry can help controllers maintain a current view of the environment and assess whether desired conditions are being met. Stale or incomplete context can lead to incorrect diagnoses. The framework proposed here treats telemetry as evidence that must be associated with topology, identity, policy, and operational context before it informs automated action.

Intent-based networking provides a semantic description of desired network behavior. The concepts in the September 2021 Internet-Draft [5] distinguish intent from the mechanisms used to achieve the desired result. This abstraction lets operators state outcomes without listing every low-level configuration action. The architecture proposed here builds on that distinction by connecting intent with policy, workflow execution, evidence, and outcome verification. Practical controllers must also account for dependencies, conflicts, authorization, and operational risk.

Network-assurance research examines how telemetry and historical data can reveal deviations from expected network behavior, including through machine-learning approaches [6]. This work supports treating monitoring and diagnosis as connected functions. The proposed architecture builds on those ideas by including evidence aggregation and explicit outcome verification. This article describes those as design principles; it does not report a completed empirical evaluation of them.

Research on multi-tenant intent-based networking has examined how high-level requirements can influence virtual-network placement and resource allocation [7]. Other work has studied translating high-level specifications into network configurations for scientific networks [8].

These studies show how intent can guide resource-sensitive decisions in particular settings. The architecture proposed here considers how related mechanisms could be organized across multiple operational loops in an enterprise controller.

Research on end-to-end service orchestration has examined how service requirements can be mapped to coordinated actions across heterogeneous platforms [9]. Work on intent-based network slicing has considered declarative requirements in software-defined networks [10]. These studies inform the proposed workflow-orchestration and hierarchical-control concepts for environments with multiple domains, controllers, and policies.

Research on intent refinement addresses how high-level goals can be translated into operationally relevant network requirements [11]. Intent, telemetry, assurance, orchestration, and execution are often studied separately. This article proposes connecting them in a vendor-neutral architecture that links contextual evidence, diagnosis, intent, policy, controlled execution, verification, and external integration across hierarchical loops. It further proposes evidence aggregation, action classes, and control budgets as ways to manage concurrent remediation while supporting explainability and change safety.

3. Architecture: Multi-Scale Evidence-to-Action Control Fabric

The proposed architecture is intended for enterprise controller platforms that manage heterogeneous devices, clients, sites, applications, services, policies, and administrative domains. It treats scale as multidimensional rather than reducing it to device count. Control responsibilities are distributed across layers and nested loops. This structure connects intent, diagnosis, action, and outcome verification while defining boundaries for authority, concurrency, and failure recovery.

3.1. Multidimensional Scale Model

Controller workload depends on several interacting dimensions. Device count affects inventory size, configuration fan-out, command volume, and state reconciliation. Client populations affect event volume, identity changes, mobility, and authentication observations. Site count affects geographic partitioning, delegated administration, latency, and failure domains. Telemetry rate affects ingestion, buffering, storage, and analytics. Applications, services, and policies add dependencies and relationships. Concurrent workflows and hardware, software, protocol, and configuration diversity also affect planning and execution capacity.

Controller scale cannot be represented by device count alone. A large client population or a burst of events may create high operational load even when the device inventory is small. A policy change may also trigger broad configuration fan-out. Capacity measurements should therefore include telemetry-ingestion rate, event backlog, graph-processing latency, workflow concurrency, and verification delay. Resource budgets at each loop can help prevent one condition from consuming excessive platform capacity.

3.2. Layered Closed-Loop Architecture

The architecture is organized into six functional layers that collectively implement an evidence-to-action feedback cycle.

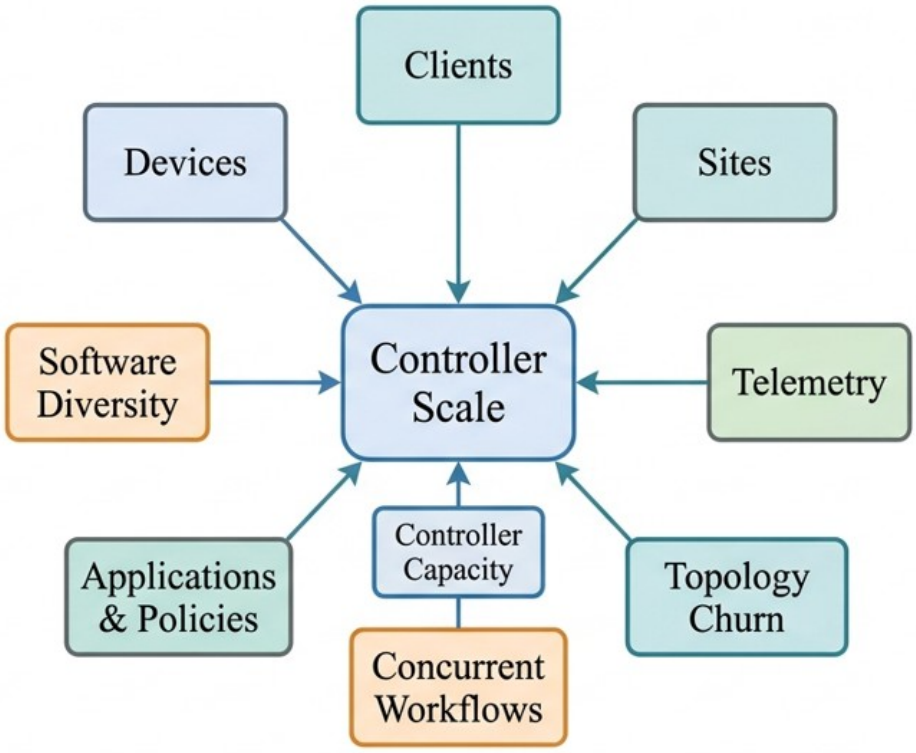


Figure 1 – Multidimensional enterprise controller scale model

The collection and context layer accepts telemetry, logs, flow records, synthetic-test results, inventory and identity data, addressing information, topology state, and controller task events. It normalizes observations and records their identity, time, source, and event context so that they can be analyzed. Collection detail can vary by operational need: economical sampling may support baseline monitoring, targeted diagnostics may increase resolution, and forensic collection may temporarily preserve more detailed evidence. This approach aims to provide useful detail while managing resource use.

The correlation and diagnosis layer converts raw observations into operational findings. It correlates events associated with users, applications, or services and may use a temporal dependency graph to connect clients, access infrastructure, controllers, policies, sites, and applications. Deterministic rules can be combined with statistical baselines, anomaly models, and topology reasoning. The goal is to distinguish high-confidence findings from ambiguous conditions rather than relying on a single analysis method.

The intent and policy layer represents desired outcomes, operational scope, licensing constraints, and service objectives. Intent provides a desired-state reference against which the operational state can be assessed. The September 2, 2021 Internet-Draft version -05 describes intent-based networking concepts [5]. The proposed architecture extends this foundation with explicit ownership, scope, authorization, and action constraints. The layer identifies intent drift and recommends responses within applicable policy and authority.

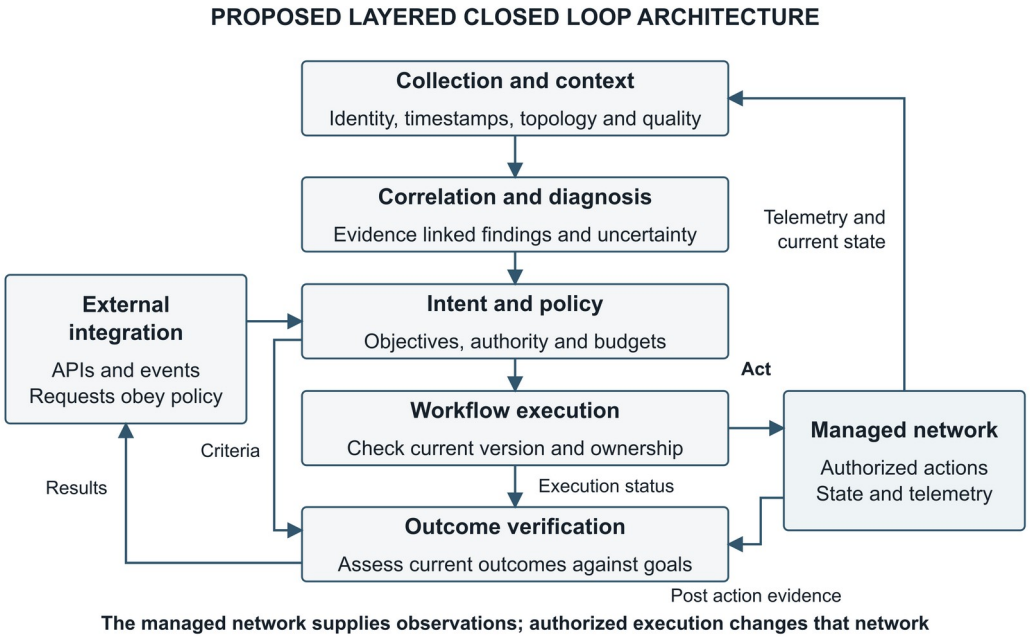


Figure 2 – Proposed layered closed-loop controller architecture. Authorized workflow execution acts on the managed network. Network observations return to collection and provide post-action evidence for outcome verification; execution status alone does not establish recovery. Current objectives supply verification criteria, and external requests remain subject to policy.

The workflow-execution layer translates approved plans into ordered, durable, and preferably idempotent actions. Device and service adapters abstract differences among underlying technologies. The workflow engine tracks checkpoints, dependencies, retries, and partial completion. Reversible actions may use compensating steps. Workloads can be separated by site, resource ownership, or failure domain. Maintenance, repair, and higher-impact responses can have distinct concurrency limits so routine automation does not consume capacity needed for more urgent work.

The verification layer determines whether the original operational objective has been restored. After execution, the controller observes a settling period and checks the primary outcome and supporting objectives. It records the result as resolved, changed, or inconclusive. A command completing successfully does not necessarily mean the service has recovered. If the desired service state was not achieved, the workflow should remain unresolved and the control loop should stay open.

The integration layer provides findings, objective indicators, workflow states, and selected operational information to authorized external systems through versioned APIs and event interfaces. External systems may subscribe to results, generate events, submit requests, or provide context. Integration must preserve the same authorization, action-classification, and control-budget requirements that apply to internal workflows; an external request must not bypass them.

3.3. Hierarchical Control Scopes

Enterprise decisions do not all belong to one centralized loop. The architecture organizes control into scopes. Device-level loops address local process, interface, or connectivity conditions. Site-level loops coordinate access, wireless, and local service behavior. Domain-

level loops address fabric, identity, software, or assurance policies. A global loop verifies results that cross site or domain boundaries.

Each loop sends summarized state to its parent or communicates constraints on higher-level goals. A local loop acts within its assigned authority, while higher-level loops coordinate resource use and inter-domain objectives. Control loops may also operate on different timescales: faster loops can handle protection and collection issues, while slower loops can manage optimization, planning, and policy. Version labels alone do not prevent stale changes. Executors should enforce current state-version and ownership preconditions at the relevant execution boundary, rejecting or replanning obsolete decisions rather than overwriting newer local corrections.

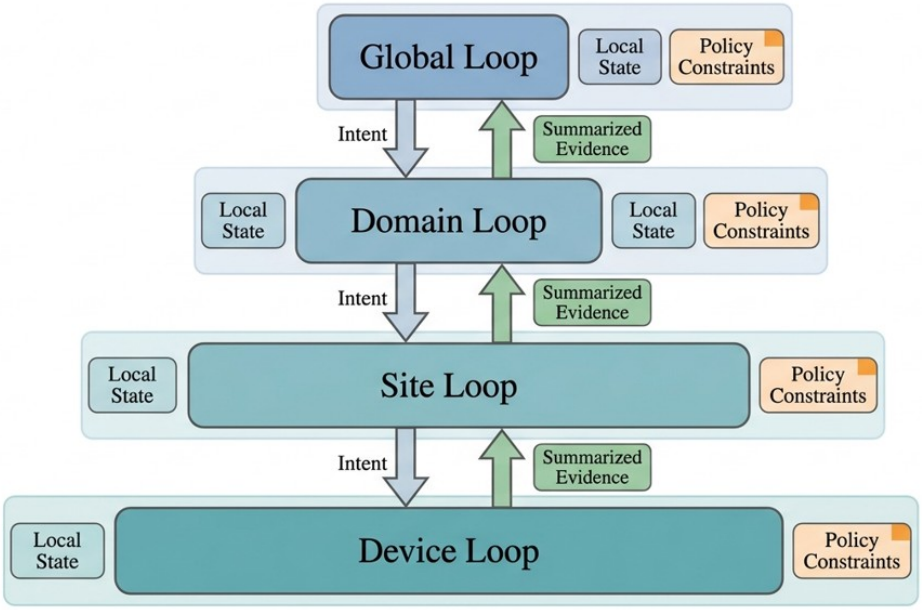


Figure 3 – Hierarchical multi-loop control architecture

Evidence aggregation can reduce repeated central processing. High-rate telemetry may be processed in partitions, while higher-level controllers receive findings, goal indicators, topology summaries, and exceptions. More detailed evidence can be retrieved when a condition crosses domain boundaries or cannot be diagnosed with sufficient confidence locally.

3.4. Control Budgets and Action Classes

Enterprise automation needs explicit limits on concurrent changes. The proposed control budget constrains concurrent remediation by factors such as site, device group, protocol, resource availability, and operational risk. These limits can reduce the impact of correlated findings and preserve capacity for higher-priority cases.

Actions can be classified by reversibility, scope, confidence requirements, and verification strength. Evidence collection and diagnostic tests are generally lower-risk than service restarts, configuration changes, or software changes, which require progressively stronger authorization and validation. Prior operational experience may inform the limits assigned to well-understood actions; unknown or high-impact actions should remain subject to stricter controls.

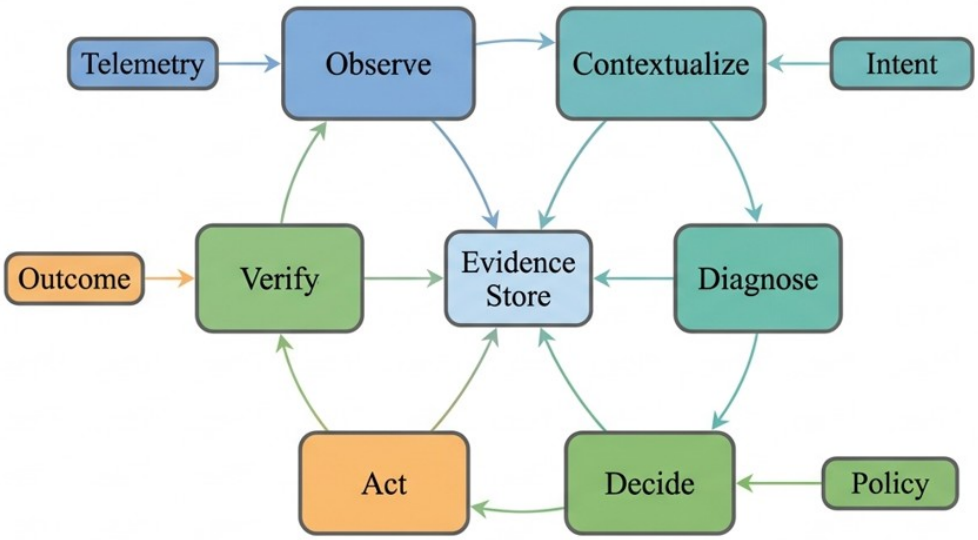


Figure 4 – Evidence-to-action closed loop

Circuit breakers provide another safety mechanism. If repeated remediation does not improve the target outcome, the loop can stop further attempts, preserve diagnostic evidence, and escalate the condition. This can prevent duplicate actions from consuming the control budget or further affecting the infrastructure.

3.5. High Availability and Loop Continuity

Closed-loop availability means that control functions continue safely through component failures while preserving operational authority and state. Intent, workflow checkpoints, ownership information, evidence, and execution state should remain available. In the proposed quorum-based deployment, only a quorum-authorized controller may issue state-changing actions. Controllers that lose that authority, including an isolated minority, must stop issuing those actions. If a quorum cannot be established, state-changing automation remains suspended. The selected protocol and effective ownership or fencing checks must enforce this authority; replica count or process status alone does not establish it.

After a failure, reconciliation logic should compare recorded workflow state with the actual external state before retrying an action. This helps avoid duplicate execution when a previous controller may have acted just before failure. The controller should also monitor evidence freshness during degraded operation. If collection or analysis is delayed, it may preserve read access or allow restricted configuration while temporarily disabling autonomous remediation. This prevents actions based on stale network state.

The proposed architecture links scale, hierarchy, evidence, intent, execution, verification, and failure handling in a multi-scale control fabric. It offers a conceptual basis for dependable closed-loop enterprise operations without requiring access to proprietary controller internals.

4. Framework Assessment: Stress, Resilience, and Operator-Centric Validation

4.1. Multidimensional Scale Assessment

The proposed framework should be assessed across independent workload dimensions, not only by the number of devices managed. Tests could vary device, client, and site counts;

telemetry rates; topology changes; co-located findings; and workflow fan-out. Varying these dimensions separately can reveal bottlenecks that may be hidden when all workload dimensions increase at the same rate.

An assessment should examine the complete control loop. Ingestion loss and event freshness measure whether the collection layer supplies timely evidence. Correlation measures may include diagnosis latency and root-cause ranking quality. Intent coverage is the fraction of operational goals that the architecture can represent and analyze. Queue age and workflow throughput indicate execution load; convergence time measures how long the loop takes to reach a verified outcome after detecting a deviation. False-remediation rate measures incorrect or unnecessary automated responses. Control-budget saturation indicates how close concurrent remediation is to its safety limits. These are proposed measures, not results reported in this article.

4.2. Resilience and Failure Assessment

Resilience testing could examine whether the architecture remains safe when components or information streams fail. Scenarios may include controller-node removal, service partition, delayed topology updates, duplicate telemetry or events, and termination of a workflow after an external action is submitted. These conditions can cause conventional automation to repeat actions, rely on stale state, or lose track of workflow ownership.

The assessment should examine four properties. First, single ownership should prevent multiple controllers from issuing conflicting remediation for the same workflow. Second, state preservation should keep committed intent, workflow checkpoints, ownership, and relevant evidence available after a failure. Third, failover testing should measure the time required to restore authoritative control. Fourth, before retrying an uncertain operation, the controller should observe the external state because the action may already have been delivered. These tests could assess availability and change safety; this article does not report that they were performed.

4.3. Closed-Loop Convergence Assessment

Convergence time should be measured from deviation detection to the first verification that the intended outcome is satisfied, consistent with Section 4.1. Observation-to-detection delay should be measured separately. Assessment should track diagnosis, approved action, execution, and verification within that interval. Results should be considered separately across loop scopes and action classes because local corrective actions and cross-domain processes may have different durations. The assessment should also record cases in which verification is inconclusive, distinguishing actual recovery from successful command execution.

4.4. Human-Centric Explainability Assessment

Automation quality should also be assessed from the operator’s perspective. Evaluators could provide operators with system records and ask them to identify the affected outcome, likely cause, supporting evidence, actions taken or planned, and remaining uncertainty. Identification accuracy, interpretation time, and consistency across operators are possible measures. An automated response that does not explain what it did or show evidence of the result gives operators limited insight into system behavior. Human-factors testing can complement technical measures under normal, degraded, and multi-domain conditions.

5. Discussion: From Product Observability to Generalizable Control Principles

Publicly documented product architectures can inform general architectural analysis without turning the article into a product critique or marketing piece. An enterprise controller provides a concrete context for examining large-scale network-management challenges. The aim is not to reproduce proprietary implementation details, but to identify design challenges that may apply across controller platforms and deployment contexts. As controller environments grow, they must associate telemetry with identity, topology, inventory, applications, and policies that may come from different sources. High-cardinality telemetry increases ingestion, processing, storage, and freshness demands. Cross-domain findings create dependencies that may not be resolved by one local controller. Long-running workflows require persistent state, ownership, idempotency, partial-completion handling, and later verification. External APIs and event interfaces should support extensibility without bypassing authorization or execution policies.

Quorum-based continuity illustrates why high availability requires more than redundant controller instances. Accepted intent, workflow state, ownership, and evidence must remain consistent enough to prevent conflicting actions during shutdowns or network partitions. The proposed architecture translates these challenges into design principles that can be examined and tested. It treats scale as multidimensional, uses hierarchical loops to limit decision scope, aggregates evidence, classifies actions, and uses budgets to constrain concurrent remediation. Verification distinguishes successful command execution from successful recovery. Together, these principles provide a vendor-neutral basis for assessing scalability, resilience, explainability, and operational safety without treating proprietary features as the standard.

6. Conclusion

Enterprise-scale closed-loop automation requires persistent state and actions that are limited, authorized, and verifiable. A centralized control model may encounter capacity, coordination, and failure-isolation limits as the populations of devices, clients, and sites grow, along with applications, policies, telemetry flows, and active workflows. The proposed architecture addresses these challenges through hierarchical control scopes, evidence aggregation, action classes, control budgets, durable workflows, and outcome-focused verification. Hierarchical loops assign decisions to the appropriate scope, allowing local conditions to be addressed near their source while preserving enterprise-wide intent and policy. Evidence aggregation can reduce the volume of information passed to higher-level controllers while retaining detail for cross-domain diagnosis. Action classes distinguish lower-risk diagnostic operations from changes that require stronger authorization and verification. Control budgets limit concurrent remediation and may reduce the impact of erroneous or correlated actions. Durable workflows support continuity through partial execution failures. Post-action verification distinguishes completion of a configuration command from recovery of the intended service. High-availability systems should preserve ownership and accepted state to reduce uncertain or duplicate actions during failover. This article proposes a vendor-neutral model for closed-loop automation in enterprise networks. It treats autonomy as bounded by authority, evidence, resources, and verification. These principles offer a basis for analyzing scalable, explainable, and resilient network operations.

References

[1]. ETSI, Zero-touch network and Service Management (ZSM); Reference Architecture, ETSI GS ZSM 002 V1.1.1, August 2019. Available:

https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/002/01.01.01_60/gs_zsm002v010101p.pdf

[2]. Cisco Systems, “Developer Toolkit GUI,” in Cisco DNA Center Platform User Guide, Release 2.2.1, pp. 1–3. Available: https://www.cisco.com/c/en/us/td/docs/cloud-systems-management/network-automation-and-management/dna-center-platform/2-2-1-0/user_guide/b_dnac_platform_ug_2_2_1/b_dnac_platform_ug_2_2_1_chapter_0111.pdf. Accessed October 27, 2026.

[3]. M. Behringer, M. Pritikin, S. Bjarnason, A. Clemm, B. Carpenter, S. Jiang, and L. Ciavaglia, “Autonomic Networking: Definitions and Design Goals,” RFC 7575, Internet Research Task Force, Informational, June 2015. doi: 10.17487/RFC7575. https://www.rfc-editor.org/rfc/rfc7575.html

[4]. H. Song, F. Qin, P. Martinez-Julia, L. Ciavaglia, and A. Wang, “Network Telemetry Framework,” RFC 9232, Internet Engineering Task Force, Informational, May 2022. doi: 10.17487/RFC9232. https://www.rfc-editor.org/rfc/rfc9232.html

[5]. A. Clemm, L. Ciavaglia, L. Z. Granville, and J. Tantsura, “Intent-Based Networking —Concepts and Definitions,” Internet-Draft draft-irtf-nmrg-ibn-concepts-definitions-05, Work in Progress, September 2, 2021. Available: https://datatracker.ietf.org/doc/draft-irtf-nmrg-ibn-concepts-definitions/05/

[6]. X. Zheng and A. Leivadeas, “Network Assurance in Intent-Based Networking Data Centers with Machine Learning Techniques,” in 1st Joint International Workshop on Network Programmability & Automation (NetPA 2021), held with the 17th International Conference on Network and Service Management (CNSM 2021), 2021, pp. 14–20. doi: 10.23919/CNSM52442.2021.9615580.

[7]. A. Leivadeas and M. Falkner, “VNF Placement Problem: A Multi-Tenant Intent-Based Networking Approach,” in 2021 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), 2021, pp. 143–150. doi: 10.1109/ICIN51074.2021.9385553.

[8]. M. Kiran, E. Pouyoul, A. Mercian, B. Tierney, C. Guok, and I. Monga, “Enabling intent to configure scientific networks for high performance demands,” Future Generation Computer Systems, vol. 79, part 1, pp. 205–214, February 2018. doi: 10.1016/j.future.2017.04.020.

[9]. A. Rafiq, A. Mehmood, T. A. Khan, K. Abbas, M. Afaq, and W.-C. Song, “Intent-Based End-to-End Network Service Orchestration System for Multi-Platforms,” Sustainability, vol. 12, no. 7, article 2782, 2020. doi: 10.3390/su12072782.

[10]. A. Rafiq, A. Mehmood, and W.-C. Song, “Intent-Based Slicing between Containers in SDN Overlay Network,” Journal of Communications, vol. 15, no. 3, pp. 237–244, March 2020. doi: 10.12720/jcm.15.3.237-244.

[11]. A. S. Jacobs, R. J. Pfitscher, R. A. Ferreira, and L. Z. Granville, “Refining Network Intents for Self-Driving Networks,” in Proceedings of the Afternoon Workshop on Self-Driving Networks (SelfDN ’18), 2018, pp. 15–21. doi: 10.1145/3229584.3229590.