International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(21s) 2026 pp. 1757-1764
https://doi.org/10.70917/ijcisim-2026-5937
Article

Unified Architecture For Cross-Layer Application

Rohit Chudasama

Independent Researcher,India, Email:rohitgchudasama@yahoo.in

Abstract: Modern Internet of Things (IoT) platforms are typically built as a chain of loosely coupled layers embedded hardware, device firmware, cloud-hosted software services, field-facing mobile applications, and, increasingly, an AI-driven visual analytics layer each engineered, released, and secured on its own schedule. This independence is manageable for a single application pattern, but it becomes a liability once a platform must support several distinct cross-layer application patterns for example, real-time visibility versus predictive condition monitoring each with different reporting cadence, retention, and reasoning requirements, and once an agentic visual-intelligence layer is added on top and needs trustworthy, cross-layer context to be useful rather than cosmetic. This paper proposes a unified cross-layer reference architecture for IoT platforms. We define five engineering layers: hardware, firmware, software services, mobile application, and agentic visual intelligence and three cross-cutting contracts that bind them: identity and tenancy, a unified telemetry/digital-twin data model, and security and observability. We instantiate the architecture against three representative cross-layer application patterns real-time asset visibility, predictive condition monitoring, and compliance/custody-chain tracking and discuss the engineering trade-offs power and compute asymmetry, end-to-end latency budgeting, and the autonomy-versus-governance tension introduced by agentic components that the unified design surfaces. The contribution is a reference architecture and a vocabulary for reasoning about cross-layer trade-offs, not a benchmarked implementation.

Keywords: Internet of Things, cross-layer architecture, reference architecture, multi-tenancy, embedded firmware, edge computing, digital twin, agentic artificial intelligence, identity and access management, asset tracking

1. INTRODUCTION

The number of connected IoT endpoints has moved from research curiosity to ordinary infrastructure: recent surveys put the worldwide device count above 15 billion and rising [1]. A large and growing share of that population exists to answer one question on behalf of a business: where is this physical asset, and what condition is it in? Tools in a maintenance facility, pallets and containers moving through a warehouse, instrumented equipment in the field, and reusable packaging in a retail supply chain are all instances of the same underlying problem attach a low-power sensor to a physical object, get its state into a backend reliably, and put that state in front of the person or system that needs to act on it.

Engineering organizations that build these platforms tend to organize around the same five disciplines, usually as five separate teams: a hardware team that selects radios, sensors, and system-on-modules; a firmware team that implements the on-device state machine and power budget; a backend/platform team that operates the software services that ingest and serve telemetry; a mobile team that builds the field-technician and installer applications; and, increasingly, a data/AI team that builds dashboards, anomaly detectors, and most recently agentic copilots that reason over telemetry and take or recommend action. Each discipline optimizes its own layer well. The weak point is the seam between layers: a firmware change to a reporting interval, a backend schema migration, or a new mobile onboarding flow routinely breaks an assumption that a different team made about the same data, and an agentic layer placed on top of these seams inherits every inconsistency underneath it.


Copyright: © 2026 by the authors

The problem compounds in two ways that are common in IoT platforms specifically. First, the same stack is frequently asked to support more than one cross-layer application pattern for instance, real-time visibility versus predictive condition monitoring that share most of the technology but differ in duty-cycle and connectivity assumptions, reporting cadence, data-retention requirements, and the kind of reasoning the top layer must perform. Treating each application pattern as a separate platform duplicates infrastructure; treating them as one undifferentiated deployment ignores real differences. Second, the recent arrival of agentic AI systems that perceive, reason, plan, and act with only loose human supervision [2] adds a layer that is only as trustworthy as the cross-layer data it reasons over. An agent that recommends re-routing a shipment or flags as a non-compliant asset is making a claim about hardware-layer sensor readings that have already passed through firmware filtering, backend aggregation, and mobile-reported correction; if the contract between those layers is informal, the agent inherits that informality as silent risk.

This paper treats "cross-layer" in a broader sense than is customary in the networking literature, where the term usually refers to optimizing interactions between OSI-style protocol layers MAC, routing, and physical layers for energy or latency [3], [4]. We instead use cross-layer in the systems-architecture sense common to industrial reference models [5]–[7]: the full path from silicon to user-facing intelligence, co-designed as one stack rather than integrated after the fact.

Contributions. This paper makes four contributions: (1) a five-layer reference architecture for IoT platforms hardware, firmware, software services, mobile application, and agentic visual intelligence defined by the responsibilities and interfaces of each layer; (2) three cross-cutting contracts (identity and tenancy, a unified telemetry/digital-twin data model, and security and observability) that bind the five layers without requiring point-to-point integration between every pair of teams; (3) an instantiation of the architecture against three representative cross-layer application patterns real-time asset visibility, predictive condition monitoring, and compliance/custody-chain tracking showing which layers bear the most engineering weight under each pattern and which remain shared; and (4) a discussion of the engineering trade-offs that the unified view surfaces, including power and compute asymmetry across layers, end-to-end latency budgeting, and the autonomy-versus-governance tension that agentic components introduce.

Paper organization. Section II positions this work against layered IoT reference architectures, classical cross-layer network design, multi-tenant platform and identity literature, digital-twin architectures, and recent work on agentic AI in IoT. Section III presents the proposed architecture, layer by layer, and the cross-cutting contracts that bind them. Section IV instantiates the architecture against three cross-layer application patterns. Section V discusses the resulting engineering trade-offs, and Section VI concludes.

2. RELATED WORK

A. Layered IoT Reference Architectures

The simplest and most widely taught IoT model is the three-layer perception–network–application model formalized by the ITU-T in Recommendation Y.2060 [6]. The European Union–funded IoT Architectural Reference Model (IoT-A) extends this into a full reference architecture spanning device, communication, IoT service, virtual-entity, and business-process layers, with security and service management treated as orthogonal concerns rather than a sixth layer [5] an early acknowledgment that some properties must cut across layers rather than live inside one. In manufacturing contexts, the Reference Architecture Model Industrie 4.0 (RAMI 4.0) takes a complementary three-dimensional view, crossing a hierarchy axis against a life-cycle axis and a set of IT layers [7]. Surveys of contemporary IoT application architectures further split each of these conceptual layers by deployment model cloud, edge, and fog each trading scalability against latency and bandwidth differently [1], [8]. What these models share is a description of layers; what they generally leave informal is the contract between layers when a single deployment must serve multiple tenants and support multiple cross-layer application patterns at once, which is the gap this paper targets.

B. Cross-Layer Design in Wireless and IoT Networks

A separate and older literature uses “cross-layer design” to describe joint optimization of the MAC, routing, and physical layers of the wireless protocol stack, typically to trade energy consumption against latency or reliability in resource-constrained sensor networks [3], [4]. This is valuable, mature work, but it operates one level below the concern of this paper: it improves how a device talks to a gateway, not how a hardware-to-AI engineering organization keeps five disciplines mutually consistent. We adopt the term “cross-layer” deliberately to signal a parallel concern at the systems-architecture level rather than the protocol level, and we are explicit about that distinction, so the two literatures are not conflated.

C. Multi-Tenant Platforms and Identity

Multi-tenancy a single platform instance serving multiple isolated customers, each with strict data isolation is a well-established cloud-architecture pattern that IoT platforms have inherited along with its trade-offs between shared-schema efficiency and per-tenant isolation. WSO2's IoT reference architecture is presented as “inherently multi-tenant” from the device layer upward rather than as a backend-only concern [9], which is the posture this paper also takes tenancy needs to be visible to firmware-facing provisioning and mobile-facing onboarding, not only to the database schema. The federation substrate for multi-tenant identity is almost universally OAuth 2.0 [10] and its OpenID Connect identity layer [11], and Microsoft's guidance on multitenant identity architecture catalogs the resulting design questions which directory of record, whether tenants federate their own identity provider, and how authorization is scoped per tenant that any cross-layer architecture must answer once, rather than once per layer [12].

D. Digital Twins and the Edge-Cloud Continuum

A digital twin a live, addressable virtual representation of a physical asset is a natural unifying data abstraction for a cross-layer architecture, since it gives every layer the same handle on “this asset” regardless of whether the layer is reading a hardware sensor value or an agentic layer's inferred risk score. Fett et al. propose a medium-level digital-twin architecture that deliberately sits between abstract reference models such as RAMI 4.0 and concrete IT deployment, supporting local, cloud, and hybrid edge-cloud placements of the same twin [13]. We borrow this medium-level posture for the telemetry/digital-twin contract in Section III: abstract enough to survive a change of cloud vendor, concrete enough that a firmware engineer and a dashboard engineer can implement against it without a meeting.

E. Agentic AI in IoT

Large language models and the agentic systems built around them components that perceive, plan, use tools, and act with limited supervision are moving from a cybersecurity research topic into a general IoT operations concern. Nageshwaran and Ezekiel's systematic survey of agentic AI for autonomous IoT cybersecurity organizes the field along agent architecture (single- versus multi-agent), reasoning strategy, action scope, and deployment topology (edge, fog, or cloud) [2]. Although that survey's action scopes are security-specific (detection, response, threat hunting), the same four-pillar lens and in particular its finding that hybrid edge/cloud placement, where lightweight on-device models pre-filter events for a cloud-hosted reasoning agent, is the most plausible deployment pattern for production environments applies directly to the operational and visual agents this paper places at Layer 5: an agent that explains an anomalous temperature reading or recommends a corrective action is performing the same perceive-reason-act loop, under the same deployment-topology trade-offs, with a different action scope.

F. Positioning

In summary, prior work either describes a layered IoT model in the abstract [1], [5]–[7], optimizes a narrower cross-layer protocol concern [3], [4], or treats multi-tenancy [9], digital twins [13], or agentic AI [2] as a standalone topic. This paper's contribution is to integrate these into a single reference architecture that is concrete enough to instantiate against multiple cross-layer application patterns, which to our knowledge has not been done for the specific hardware-to-agentic-AI span addressed here.

3. PROPOSED UNIFIED CROSS-LAYER ARCHITECTURE

A. Design Principles

The architecture rests on four principles. First, contracts, not point integrations: any two layers interact only through one of the three cross-cutting contracts defined in Section III-G, never through a bilateral API negotiated outside that contract. Second, tenant and identity context are propagated, not re-derived: a device, mobile session, or agent call carries its tenant and identity context from the layer that established it rather than having each downstream layer re-infer it. Third, one data model from sensor to agent: the unit of telemetry that leaves Layer 1 and the unit of context an agent reasons over at Layer 5 are the same digital-twin object at different levels of enrichment, not two independently designed schemas connected by a translation script. Fourth, layers degrade independently: a Layer 5 outage must not affect Layer 1–3 data collection, and a connectivity loss at Layer 1 must be representable as a known device state rather than silence

Fig. 1 summarizes the resulting stack.

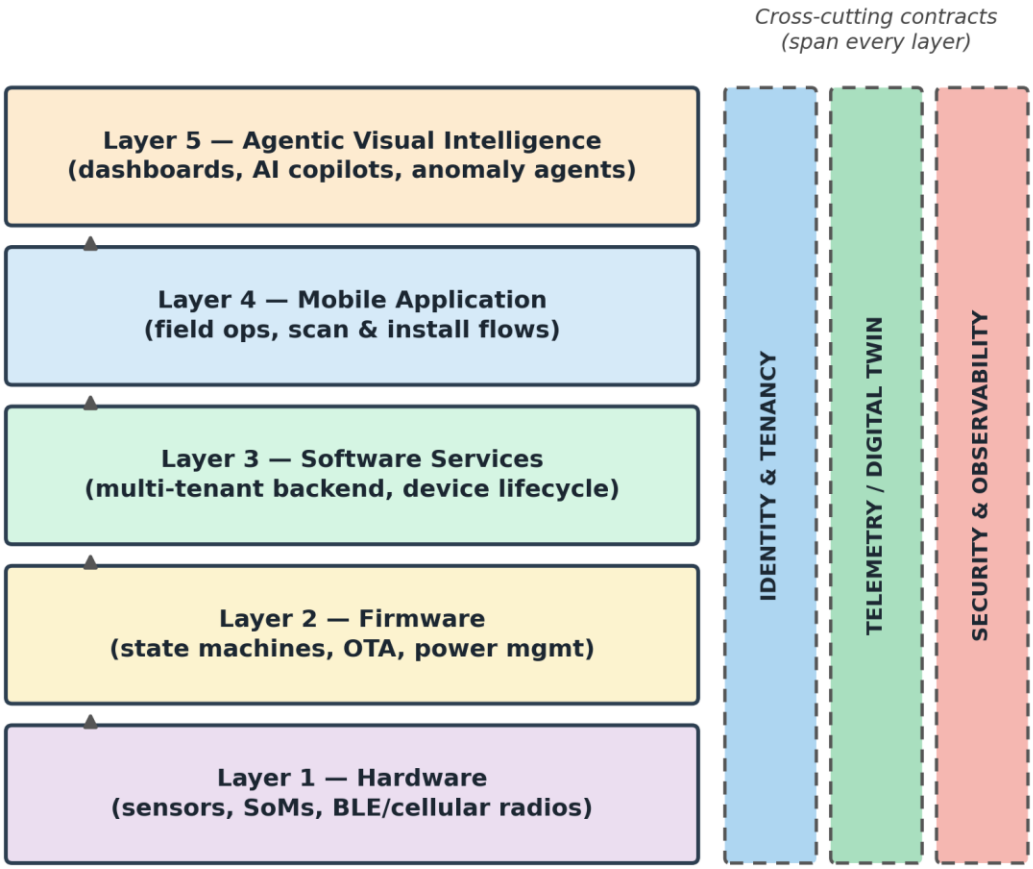


Fig. 1. Unified cross-layer architecture: five engineering layers bound by three cross-cutting contracts.

B. Layer 1 — Hardware

The hardware layer comprises the sensing and radio components: low-power microcontrollers or systems-on-module, inertial and environmental sensors, and one or more radios chosen for the deployment's indoor/outdoor and range profile typically Bluetooth Low Energy (BLE) for zone-level indoor positioning, cellular (LTE-M/NB-IoT) or GNSS for wide-area outdoor tracking, and increasingly a heterogeneous multi-core system-on-module that runs a real-time core alongside a Linux-capable application core for sites that need on-device aggregation. This layer's contract upward is narrow and deliberately so: a signed, versioned telemetry frame and a device identity credential. Everything

else calibration, filtering, scheduling is Layer 2's responsibility, which keeps Layer 1 replaceable as radio and silicon vendors change without destabilizing the layers above it.

C. Layer 2 — Firmware

Firmware owns the on-device state machine: movement and user-mode detection, duty-cycled scan/upload scheduling (commonly pseudorandomized against the device's own MAC address to avoid fleet-wide synchronized radio contention), local buffering during connectivity loss, and over-the-air update handling. The cross-layer obligation this paper places on firmware is that every state transition is reported, not inferred after the fact a backend that has to guess whether a multi-hour data gap was a sleep state, a dead battery, or a lost asset is a backend that cannot give Layer 5 a trustworthy signal to reason over. Firmware is therefore the layer most directly responsible for the quality of the “known device state” principle from Section III-A.

D. Layer 3 — Software Services

The software-services layer is the multi-tenant backend: telemetry ingestion, device lifecycle and provisioning, tenant- and role-scoped identity (Section III-G), per-tenant feature configuration, and the persistent store for both raw telemetry and the enriched digital-twin representation that Layer 5 consumes. This is also the layer that resolves the tenancy model: a shared-schema design (lower operating cost, common when tenants share most configuration) or a shared-infrastructure, isolated-schema design (stronger isolation, more appropriate where regulatory or contractual separation between tenants is required) [9]. Because this layer sits between two very different consumers low-bandwidth, intermittently connected devices below it and high-expectation human and agentic consumers above it where backpressure and rate-limiting for fleet-wide events (e.g., a connectivity outage that causes thousands of devices to reconnect simultaneously) must be absorbed.

E. Layer 4 — Mobile Application

The mobile layer is the human-in-the-loop complement to the autonomous layers above and below it: field-technician installation and decommissioning flows, manual asset-state corrections, and increasingly the surface through which a human accepts, rejects, or overrides a Layer 5 agent's recommendation. Because mobile sessions are themselves a tenant- and identity-scoped client of Layer 3, this layer adds no new identity model of its own; its distinguishing engineering concern is graceful operation in the same connectivity-poor environments (hangars, ports, warehouses) that motivate the hardware layer's radio choices, including offline-first local caching and conflict resolution when a technician's manual correction and a device's own report disagree.

F. Layer 5 — Agentic Visual Intelligence

This is the newest layer and the one whose trustworthiness depends most directly on the contracts below it. We define it as a visual analytics surface backed by one or more agentic components software that perceives state via the Layer 3 data plane, reasons over it (with strategies ranging from simple threshold rules to large-language-model-based chain-of-thought or tool-using reasoning loops), and either recommends or, within explicit guardrails, executes an action [2]. Consistent with the deployment-topology finding in Section II-E, we recommend a hybrid placement: lightweight, possibly on-device or fog-tier filtering and anomaly scoring close to the data, with heavier reasoning natural-language explanation, multi-step investigation, cross-asset correlation placed in the cloud tier where latency budgets are looser. Two properties are specific to this layer's cross-layer obligations: every recommendation must cite the digital-twin state (and its provenance through Layers 1–3) that produced it, so a human reviewing the recommendation at Layer 4 can audit it; and the layer must declare its action scope explicitly detection-only, human-approved action, or autonomous action within bounds because, as Section V-C discusses, these carry very different governance requirements.

G. Cross-Cutting Contracts

1) Identity and Tenancy: A single OAuth 2.0/OpenID Connect–based identity plane [10], [11] issues tokens carrying both an identity and a tenant claim, consumed identically by Layer 3 APIs, Layer 4 mobile sessions, and Layer 5 agent tool calls; the tenant claim, not a per-layer configuration file, is what determines which tenant's data a given request is authorized to see. This mirrors current multitenant-identity guidance for choosing a directory of record and federation model [12] and the "inherently multi-tenant" posture of comparable IoT reference architectures [9], applied consistently from the mobile session down to the agent tool call.

2) Unified Telemetry / Digital-Twin Data Model: Every physical asset is represented by one digital-twin object whose schema is versioned once, centrally, and referenced not copied by firmware encoders, backend services, mobile clients, and Layer 5 agents, in the medium-level-architecture sense of sitting between an abstract reference model and a concrete deployment [13]. Enrichment (e.g., a Layer 5 anomaly score) is added as new, clearly provenanced fields on the same object rather than a parallel schema, which is what lets a Layer 5 recommendation in Section III-F point all the way back to the originating sensor reading.

3) Security and Observability: Transport security, key management, and audit logging are implemented once, as shared infrastructure invoked by every layer, rather than re-implemented per layer; this is also the layer at which agent-specific risks flagged in the agentic-AI literature prompt injection via untrusted telemetry content, and tool-call overreach are mitigated, since both are, structurally, an authorization and audit-logging problem before they are an AI-safety problem.

4. CROSS-LAYER APPLICATION PATTERNS

The same five-layer stack supports several distinct classes of cross-layer application, distinguished less by the layers present which stay fixed than by which layer carries the most engineering weight and what Layer 5 is asked to reason about. We describe three representative patterns below; Section IV-D summarizes how they map onto the architecture.

A. Real-Time Asset Visibility

The most common pattern asks a simple question continuously: where is this asset, and is it where it should be? Layers 1–2 dominate the engineering effort duty-cycled scanning, zone-level positioning, and reliable state reporting matter far more here than downstream reasoning and Layer 5 logic is correspondingly simple: rule-based presence and dwell-time checks rather than multi-step inference. This pattern stresses the architecture's lower layers and the unified telemetry contract (Section III-G-2) more than its agentic one.

B. Predictive Condition Monitoring

A second pattern reasons about trend rather than position: is this asset's measured condition (temperature, vibration, battery health) drifting toward a failure state? Here Layer 2's sampling strategy and Layer 3's time-series retention matter as much as Layer 1's sensor choice, and Layer 5 shifts from threshold rules toward the chain-of-thought and tool-using reasoning loops surveyed in Section II-E [2], since a meaningful prediction usually requires correlating several individually unremarkable signals over time rather than reacting to a single threshold breach.

C. Compliance and Custody-Chain Tracking

A third pattern is less about the asset's physical state than about an auditable record of who held it and when. This pattern leans hardest on the identity-and-tenancy and security-and-observability contracts (Sections III-G-1 and III-G-3): Layer 3 must retain a tamper-evident handoff history, Layer 4 becomes the primary point of human-asserted state change (e.g., a technician confirming a handoff), and Layer 5's role is reconciliation flagging a missing or out-of-order handoff event rather than prediction.

D. What Changes and What Doesn't

Table I summarizes the comparison. Across all three patterns, Layer 1's hardware and Layer 2's firmware obligations are nearly identical at the protocol level the same signed telemetry frame and reported-state-transition contract from Sections III-B and III-C apply regardless of which pattern consumes the data. What changes are concentrated in Layer 3's retention policy, Layer 4's role as a passive sensor client versus an active state-correction surface, and, most of all, Layer 5's reasoning strategy and the cross-cutting contract each pattern leans on hardest. This is the practical payoff of treating the five layers and three contracts as fixed infrastructure: a new application pattern is, in the common case, a Layer 5 and configuration change, not a new platform.

TABLE I

How Each Layer's Role Shifts Across Cross-Layer Application Patterns

Layer	**Asset Visibility**	**Predictive Monitoring**	**Custody Tracking**
Hardware	Position sensing	Condition sensors (temp., vibration)	Standard sensing + tamper seal
Firmware	Frequent position reporting	Continuous trend sampling	Event-triggered handoff reporting
Software Services	Live state store	Time-series retention	Tamper-evident audit log
Mobile App	Passive client	Passive client	Active handoff confirmation
Agentic Visual	Rule-based presence checks	Trend / anomaly reasoning	Reconciliation & gap detection

5. DISCUSSION: ENGINEERING TRADE-OFFS

A. Power and Compute Asymmetry Across Layers

The five layers span roughly six orders of magnitude of available power and compute, from a coin-cell-powered microcontroller at Layer 1 to a cloud GPU cluster reasoning at Layer 5. A unified architecture must make this asymmetry explicit rather than let it surface as an emergent bug: filtering and data-reduction decisions made cheaply at Layers 1–2 directly set the achievable latency and cost envelope for Layer 5 reasoning, and a common failure mode is specifying an agentic use case (e.g., sub-minute anomaly response) that is unreachable given the duty cycle a battery-powered Layer 1 device was provisioned with.

B. Latency Budgets Across the Stack

Related to the above, end-to-end latency in this stack is usually dominated by Layer 1's reporting interval rather than any layer above it; this argues for budgeting latency top-down from the business requirement (e.g., “flag a custody gap within five minutes”) back to a Layer 1/2 duty cycle, rather than bottom-up from whatever duty cycle a hardware team already shipped. As Section II-E notes, the agentic-AI literature's hybrid edge/cloud placement pattern [2] is the relevant precedent: cheap, fast filtering near the device and expensive, slower reasoning in the cloud, composed so that the cloud tier's latency is hidden behind, not added to, the device tier's own reporting interval.

C. Autonomy versus Governance in the Agentic Layer

The action-scope distinction from Section II-E detection-only, human-approved action, and autonomous action maps directly onto increasing governance burden, and we recommend that Layer 5 components default to the most conservative scope (detection-only, citing provenance per Section III-F) and graduate to human-approved or autonomous action only where the cost of a false action is bounded and recoverable. This follows directly from treating the agentic layer as inheriting risk from every layer below it: an agent cannot be more trustworthy than the firmware-reported state it reasons over, and over-automating before that trust is established converts an occasional bad sensor reading into an occasional bad autonomous action.

D. Standardization and Interoperability Gaps

Two gaps stood out while instantiating this architecture against multiple cross-layer application patterns. First, while device- and transport-level standards are comparatively mature (e.g., Bluetooth SIG and ASTM/ISO guidance relevant to BLE and RFID asset tracking), there is no equally mature standard for the Layer 3/Layer 5 boundary the schema an agent should expect a digital twin to expose which is exactly the gap the unified telemetry contract in Section III-G is designed to fill locally, pending broader standardization. Second, multi-agent interoperability standards (for an agent serving one application pattern to safely call an agent serving another) are still emergent outside IoT entirely and have not yet been evaluated for IoT deployment topologies at all, which we flag as a direction for future work rather than a solved cross-cutting contract.

6. CONCLUSION AND FUTURE WORK

This paper proposed a unified, five-layer cross-layer architecture for multi-tenant IoT platforms hardware, firmware, software services, mobile application, and agentic visual intelligence bound by three cross-cutting contracts for identity/tenancy, telemetry/digital-twin data, and security/observability, and instantiated it against three cross-layer application patterns spanning real-time visibility, predictive condition monitoring, and compliance tracking. The central claim is architectural rather than empirical: engineering these five disciplines against shared contracts, rather than as independently integrated layers, is what makes an agentic visual layer trustworthy rather than cosmetic, and what lets a platform support multiple cross-layer application patterns without duplicating its lower layers. Future work includes a quantitative evaluation of the architecture against a reference implementation onboarding time for a new tenant or application pattern, end-to-end latency under the budgeting approach of Section V-B, and agent recommendation precision conditioned on provenance completeness and tracking the multi-agent interoperability standards flagged in Section V-D as they mature.

REFERENCES

1. A. Dauda, O. Flauzac, and F. Nolot, “A Survey on IoT Application Architectures,” Sensors, vol. 24, no. 16, Art. no. 5320, Aug. 2024.
2. V. Nageshwaran and S. Ezekiel, “Agentic AI and Large Language Models for Autonomous IoT Cybersecurity: A Systematic Survey, Taxonomy, and Research Roadmap,” Electronics, vol. 15, no. 12, Art. no. 2740, Jun. 2026.
3. B. Fu, Y. Xiao, H. J. Deng, and H. Zeng, “A Survey of Cross-Layer Designs in Wireless Networks,” IEEE Commun. Surveys Tuts., vol. 16, no. 1, pp. 110–126, 2014.
4. L. D. Mendes and J. J. P. C. Rodrigues, “A Survey on Cross-Layer Solutions for Wireless Sensor Networks,” J. Netw. Comput. Appl., vol. 34, no. 2, pp. 523–534, Mar. 2011.
5. M. Bauer et al., “IoT Reference Architecture,” in Enabling Things to Talk: Designing IoT Solutions with the IoT Architectural Reference Model. Berlin, Germany: Springer, 2013, pp. 163–211.
6. ITU-T, Overview of the Internet of Things, ITU-T Recommendation Y.2060, Geneva, Switzerland, Jun. 2012.
7. DIN SPEC 91345:2016-04, Referenzarchitekturmodell Industrie 4.0 (RAMI4.0), Beuth Verlag, Berlin, Germany, 2016.
8. S. Hamdan, M. Ayyash, and S. Almajali, “Edge-Computing Architectures for Internet of Things Applications: A Survey,” Sensors, vol. 20, no. 22, Art. no. 6441, Nov. 2020.
9. WSO2 Inc., “A Reference Architecture for the Internet of Things,” WSO2 White Paper. [Online]. Available: https://wso2.com/whitepapers/a-reference-architecture-for-the-internet-of-things/
10. D. Hardt, Ed., “The OAuth 2.0 Authorization Framework,” IETF, RFC 6749, Oct. 2012.
11. N. Sakimura, J. Bradley, M. B. Jones, B. de Medeiros, and C. Mortimore, “OpenID Connect Core 1.0,” OpenID Foundation, Nov. 2014.
12. Microsoft, “Architectural Considerations for Identity in a Multitenant Solution,” Azure Architecture Center. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/guide/multitenant/considerations/identity
13. M. Fett, M. Kraft, F. Wilking, S. Goetz, S. Wartzack, and E. Kirchner, “Medium-Level Architectures for Digital Twins: Bridging Conceptual Reference Architectures to Practical Implementation in Cloud, Edge and Cloud–Edge Deployments,” Electronics, vol. 13, no. 7, Art. no. 1373, Apr. 2024.