Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

Quantifying Cloud Architecture Debt: A Multi-Dimensional Cloud Architecture Debt Index for Enterprise Systems

Sachin Suryawanshi
Even & Odd Minds LLC
United States

Abstract: **Cloud architecture debt accumulates when locally reasonable infrastructure and design decisions gradually reduce a workload's ability to remain secure, reliable, economical, observable, performant, maintainable, and governable. Existing architectural technical-debt research provides valuable methods for structural coupling, architectural smells, and refactoring effort, but these approaches do not directly provide enterprise cloud teams with a workload-level measure that combines operational quality risks across multiple cloud concerns. This paper proposes the Cloud Architecture Debt Index (CADI), a vendor-neutral assessment model for quantifying cloud architecture debt from observable evidence. CADI evaluates seven dimensions: security, reliability, cost efficiency, performance and scalability, operational excellence, maintainability and evolvability, and governance and compliance. Each dimension is scored from explicit indicators on a five-level severity scale and normalized to a 0–100 debt score. The model supports workload-specific weights but constrains them to preserve cross-dimensional visibility. It also introduces critical-risk floors so that severe security or reliability weaknesses cannot be hidden by strong scores elsewhere. The framework is evaluated analytically using five contrasting workload archetypes, monotonicity tests, weight-sensitivity analysis, and critical-risk stress tests. The results show that the index differentiates balanced low-debt systems from systems whose aggregate average would otherwise conceal concentrated high-impact debt. CADI is intended as a decision-support instrument rather than a substitute for architecture review. Its contribution is a transparent, reproducible bridge between technical-debt theory and cloud well-architected assessment that can support remediation prioritization, trend tracking, and communication with engineering and business stakeholders.**

Keywords: **architectural technical debt; cloud architecture; enterprise systems; architecture metrics; cloud governance; reliability; cybersecurity; cost optimization**

I. INTRODUCTION

Cloud systems rarely become difficult to operate because of a single obviously poor decision. More often, debt accumulates through many defensible local choices: a temporary public endpoint remains in production, a failover design is never retested, a database tier is oversized to avoid a performance incident, an unsupported library survives one more release, observability is inconsistent across services, and infrastructure policies differ by environment. Each decision may be explainable in isolation. Their combined effect can be a workload that is more expensive to change, harder to recover, riskier to secure, and less predictable to operate.

Technical debt research provides a useful metaphor for this condition. Technical debt captures the future cost or friction created when short-term decisions compromise longer-term system quality [1], [2]. Architectural technical debt (ATD) is especially consequential because architecture-level decisions influence broad parts of a system and are often costly to reverse [3], [4]. Prior research has proposed ways to estimate ATD through rework cost, dependency analysis, coupling, architectural smells, and machine-learning-assisted severity estimation [5]-[8]. These contributions are important, but enterprise cloud architecture introduces an additional measurement problem. Operational cloud debt is not visible only in source-code structure. It also appears in identity boundaries, network exposure, recovery design, service limits, observability, deployment controls, cloud spending patterns, configuration drift, and policy exceptions.

Major cloud architecture frameworks acknowledge this multi-dimensional character. The Azure Well-Architected Framework organizes workload quality around reliability, security, cost optimization, operational excellence, and performance efficiency [9]. The AWS Well-Architected Framework uses closely related pillars and additionally includes sustainability [10]. ISO/IEC 25010:2023 likewise treats software and ICT product quality as a set of multiple quality characteristics rather than a single property [11]. These frameworks are effective for identifying recommendations and tradeoffs, but they are not technical-debt indices. Teams can finish an architecture review with dozens of findings and still lack one transparent mechanism for answering three practical questions: How much architecture debt is present? Which dimensions dominate it? Is debt getting better or worse over time?

This paper addresses that gap with the Cloud Architecture Debt Index (CADI). CADI is a workload-level, evidence-based scoring model for enterprise cloud systems. It does not attempt to estimate code refactoring hours and does not replace architectural-smell analysis. Instead, it quantifies debt across seven operational architecture dimensions and preserves visibility of concentrated critical risks. The research questions are: RQ1, how can heterogeneous cloud architecture

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

weaknesses be normalized into a reproducible multi-dimensional debt measure? RQ2, how can the model prevent a severe security or reliability weakness from being mathematically canceled by strong performance or cost scores? RQ3, does the proposed index behave consistently across contrasting architecture profiles and reasonable changes in dimension weights?

The paper makes three contributions. First, it defines an operational taxonomy of cloud architecture debt that connects technical-debt theory with observable workload evidence. Second, it specifies a scoring model with bounded customization and critical-risk floors. Third, it evaluates the behavior of the index through analytic scenarios and sensitivity tests, making clear what the model can and cannot claim before industrial validation.

II. RELATED WORK AND RESEARCH GAP

A. Architectural technical debt

The technical-debt metaphor has evolved from code-level shortcuts into a broader research area covering requirements, design, architecture, tests, documentation, infrastructure, and other software artifacts. Alves et al. mapped 100 studies and showed that debt identification and management use a wide range of indicators and strategies [1]. The Dagstuhl technical-debt seminar later emphasized that debt management requires explicit attention to principal, interest, value, and decision context rather than merely cataloging code defects [2].

ATD deserves separate treatment because architecture decisions shape system-wide dependencies and quality attributes. Besker, Martini, and Bosch synthesized ATD research into a unified model and highlighted the long-term consequences of architecture decisions [3]. Verdecchia et al. subsequently developed an empirically grounded theory describing ATD causes, symptoms, consequences, management strategies, and communication problems [4]. MacCormack and Sturtevant demonstrated an association between architectural coupling and defect-related maintenance activity, supporting the view that structural architecture choices can create measurable downstream cost [6].

B. Existing measurement approaches

Measurement has been a recurring challenge. Nord et al. proposed an architecture-focused, measurement-based approach that connects architectural dependencies with expected rework and project value [5]. Nayebi et al. used coupling and architectural-flaw severity in a longitudinal industrial setting and reported measurable improvement after debt repayment [7]. More recently, Sas and Avgeriou proposed an architectural technical-debt index that combines detected architectural smells, severity estimation, and affected lines of code, using machine learning to estimate debt principal [8]. These approaches make a strong contribution to source-oriented or design-structure-oriented ATD measurement.

CADI addresses a different level of analysis. A cloud workload can have clean module boundaries and still carry serious architecture debt because disaster recovery is untested, identity privileges are excessive, telemetry is incomplete, autoscaling is absent, production infrastructure is configured manually, or cost anomalies are routinely ignored. Such weaknesses can be partly represented in code or infrastructure-as-code, but many require evidence from configuration state, service objectives, recovery exercises, operational telemetry, security posture, and governance practice. CADI therefore treats the workload as the unit of assessment and uses a broader evidence model.

C. Cloud architecture quality frameworks

Azure and AWS provide mature architecture-review frameworks. Azure's five pillars are reliability, security, cost optimization, operational excellence, and performance efficiency [9]. AWS uses operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability [10]. Both frameworks explicitly acknowledge tradeoffs. For example, greater redundancy can improve reliability while increasing cost and operational complexity; aggressive cost reduction can reduce spare capacity or control depth [12]. NIST CSF 2.0 emphasizes governance as a core part of cybersecurity risk management [13], while NIST SP 800-207 provides principles for zero-trust architecture and continuous, resource-centered security decisions [14].

These sources inform CADI's dimensions and indicators, but CADI is intentionally vendor-neutral. It converts architecture-review evidence into a debt-oriented score and trend that can be compared across workloads without assuming that all systems should implement the same cloud service pattern. Recent work on agentic AI security likewise argues for continuous, action-level verification of identity, delegated scope, data sensitivity, tool risk, and operational impact in enterprise cloud environments [15].

III. RESEARCH METHOD

The study follows a design-science approach: identify a practical measurement problem, derive requirements from prior research and recognized architecture frameworks, construct an artifact, and evaluate the artifact's internal behavior against explicit criteria. The artifact is the CADI scoring model and assessment rubric. The present study is analytical, not an industrial field validation. No claim is made that a specific CADI score predicts outage probability, breach probability, maintenance hours, or financial loss.

Five design requirements guided the model. R1, multi-dimensionality: the model must cover cloud quality concerns that exist outside source-code structure. R2, transparency: assessors must be able to explain how a score was produced. R3, monotonicity: worsening an indicator while holding everything else constant must not improve the score. R4, limited compensability: strong performance or cost efficiency must not hide a critical security or reliability weakness. R5, contextuality: organizations must be able to adjust weights for workload criticality without making dimensions disappear.

Indicators were derived by triangulating ATD concepts with the Azure and AWS Well-Architected pillars, NIST cybersecurity guidance, and ISO/IEC 25010 quality characteristics [3], [4], [9]-[14]. Similar concerns were consolidated to avoid double counting. For example, autoscaling capacity and performance headroom belong to performance and scalability, while deployment rollback and incident readiness belong to operational excellence. Governance was retained as a separate dimension because policy consistency, ownership, exception management, and compliance evidence can be weak even when a workload is technically secure at a point in time.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

IV. CLOUD ARCHITECTURE DEBT INDEX

A. Assessment dimensions

CADI evaluates seven dimensions. The dimensions are deliberately broad enough for enterprise workload assessment but narrow enough to support evidence-based scoring. Table I summarizes their intent and representative evidence. The framework does not prescribe a fixed number of indicators. A baseline profile is provided in this paper, and organizations may add domain-specific indicators if the scoring scale and evidence rules remain unchanged.

TABLE I. CADI DIMENSIONS AND REPRESENTATIVE EVIDENCE

Dimension	Debt focus	Representative evidence
Security	Exposure and control weakness	Identity scope, public access, encryption, vulnerability age
Reliability	Failure and recovery weakness	Redundancy, RTO/RPO, failover tests, dependency blast radius
Cost efficiency	Persistent avoidable spend	Rightsizing, idle capacity, reservations/commitments, anomaly handling
Performance & scalability	Inability to meet demand efficiently	Latency SLOs, scaling, load tests, service limits
Operational excellence	Weak run and change discipline	Telemetry, alerts, IaC, rollback, incident readiness
Maintainability & evolvability	Change friction and obsolescence	Coupling, unsupported components, upgrade friction, documentation
Governance & compliance	Inconsistent control and ownership	Policy-as-code, tagging, ownership, exceptions, audit evidence

B. Indicator severity scale

Each indicator is scored on an ordinal severity scale from 0 to 4. The score describes debt, so higher values are worse. A score of 0 represents no material debt for the assessed requirement; 1 represents minor debt with low near-term consequence; 2 represents moderate debt requiring planned remediation; 3 represents high debt that materially constrains operation or change; and 4 represents critical debt with immediate or systemic risk. Assessors record the evidence used for each score, such as configuration exports, architecture diagrams, monitoring data, recovery-test results, policy reports, vulnerability findings, cost reports, or operational records.

The scale is intentionally coarse. Cloud architecture evidence is heterogeneous, and false precision can be more misleading than a transparent ordinal judgment. The scoring guide requires the assessor to choose the lowest severity that is fully supported by evidence. Missing evidence is not scored as zero. If an indicator is applicable but evidence is unavailable, it receives an evidence-gap flag and is provisionally scored at least 2 until verified. This discourages an architecture from appearing healthy because controls are undocumented or untested.

TABLE II. BASELINE INDICATORS

Dimension	Baseline indicators
Security	Least privilege; public exposure; encryption/key management; vulnerability currency; secret handling
Reliability	Zone/region resilience; recovery objectives; failover testing; backup restore testing; single points of failure
Cost efficiency	Utilization/right-sizing; idle resources; pricing commitments; storage lifecycle; anomaly ownership
Performance & scalability	Latency targets; load testing; horizontal scaling; capacity limits; data-path efficiency
Operational excellence	IaC coverage; deployment rollback; observability coverage; actionable alerting; incident runbooks
Maintainability & evolvability	Coupling; obsolete platforms; dependency upgradeability; architecture documentation; change lead time
Governance & compliance	Policy enforcement; resource ownership; tagging; configuration drift; exception lifecycle

C. Dimension and aggregate scoring

For dimension j with n applicable indicators, indicator severity s_{ij} is in {0,1,2,3,4}. Indicator importance a_{ij} is normally 1, but may be set to 2 for an indicator explicitly classified as high importance before scoring begins. The normalized dimension debt D_j is:

$$D_j = 25 \times [\, \Sigma(a_{ij}\, s_{ij}) / \Sigma(a_{ij}) \,] \quad (1)$$

D_j therefore ranges from 0 to 100. The organization then assigns a dimension weight w_j. To preserve multi-dimensional visibility, each weight must be between 0.08 and 0.25 and all weights must sum to 1.0. Equal weights are used when no workload-specific priority model has been approved. The base cloud architecture debt score B is:

$$B = \Sigma(w_j\, D_j), \ \text{with} \ \Sigma w_j = 1 \quad (2)$$

The bounded weighting rule is important. Without a lower bound, a team could assign almost no weight to a weak dimension and mathematically remove the problem. Without an upper bound, one business concern could dominate the index and turn CADI into a single-purpose score.

D. Critical-risk floors

A simple weighted mean is compensatory. A workload with excellent cost and performance scores could still achieve a moderate aggregate score even if it exposes sensitive data publicly or has no recoverable backup. CADI therefore includes critical-risk floors. A Critical Architecture Condition (CAC) is an indicator scored 4 that meets a predefined criticality rule. Baseline CAC rules apply to security and reliability because failures in these dimensions can cause irreversible confidentiality, integrity, availability, or recovery consequences. Organizations may define additional CAC rules for regulated or safety-critical workloads.

If one CAC is present, the final CADI score cannot be lower than 60. If two or more CACs are present across two

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

dimensions, the final score cannot be lower than 75. The final score C is therefore:

$$C = max(B, F\_c) \quad (3)$$

where F_c is 0 when no CAC exists, 60 for one CAC, and 75 for two or more cross-dimensional CACs. The floor does not assert a probability of failure. It encodes a governance principle: a critical architecture weakness should remain visible in the portfolio-level score even when the rest of the workload is strong.

E. Interpretation and remediation priority

CADI uses five interpretation bands: 0–20 low debt, greater than 20–40 controlled debt, greater than 40–60 elevated debt, greater than 60–75 high debt, and greater than 75–100 critical debt. These bands are decision bands rather than empirically calibrated risk probabilities. Their purpose is to trigger different review behaviors. Low debt supports routine monitoring; controlled debt supports planned backlog treatment; elevated debt requires an explicit remediation roadmap; high debt requires leadership visibility and time-bounded action; critical debt requires immediate risk ownership.

The aggregate score should never be used alone. Each assessment produces a seven-dimension debt vector, the aggregate CADI score, all CACs, and the top remediation items. For remediation ranking, this paper proposes a simple priority value P_k for finding k:

$$P\_k = Severity\_k \times Exposure\_k \times ChangeReach\_k \quad (4)$$

Each factor is rated 1–4. Exposure represents how frequently or broadly the weakness can affect the workload, while ChangeReach represents how many components, teams, or business capabilities are affected. This separate priority value prevents the aggregate architecture score from being misused as a replacement for backlog sequencing.

V. ANALYTICAL EVALUATION

A. Evaluation design

The purpose of the evaluation is to test the internal behavior of CADI before field validation. Five synthetic workload archetypes were constructed to represent distinct architecture profiles: a mature balanced workload, a fast-growth SaaS workload, a legacy cloud migration, a cost-efficient but insecure workload, and a highly resilient but overengineered workload. Dimension scores were assigned directly to isolate index behavior from indicator collection. Equal dimension weights were used. These scenarios are not empirical observations and should not be interpreted as industry benchmarks.

Four tests were applied. First, discrimination tests whether the index separates architecture profiles with visibly different debt patterns. Second, monotonicity tests whether increasing any one dimension while holding all others constant can reduce CADI. Third, non-compensability tests whether critical-risk floors preserve severe weaknesses that an average would dilute. Fourth, sensitivity tests vary dimension weights within the allowed 0.08–0.25 range to determine whether reasonable prioritization choices radically change the conclusion.

TABLE III. SYNTHETIC WORKLOAD EVALUATION

Workload	S	R	C	P	O	M	G	Base	Final
Mature balanced	15	20	20	15	20	25	15	18.6	18.6
Fast-growth SaaS	35	45	55	50	45	65	40	47.9	47.9
Legacy migration	50	60	45	55	60	80	65	59.3	59.3
Cost-efficient, insecure	90	35	15	25	40	45	60	44.3	60.0*
Resilient, overengineered	25	15	80	35	40	50	25	38.6	38.6

** Security score contains a Critical Architecture Condition; CAC floor applied. S=Security, R=Reliability, C=Cost, P=Performance, O=Operations, M=Maintainability, G=Governance.*

B. Results

The mature workload produces a low-debt score of 18.6, while the fast-growth SaaS workload reaches 47.9 because scale, cost, operations, and maintainability have accumulated simultaneously. The legacy migration produces the highest unfloored score, 59.3, reflecting broad debt rather than one isolated weakness. The overengineered workload scores 38.6: its reliability is strong, but cost debt is high and maintainability is moderately weak. This result is intentional because CADI treats unnecessary architectural complexity as debt even when the workload is stable.

The cost-efficient but insecure workload demonstrates the compensation problem. Its simple equal-weight average is 44.3, which would place it in the elevated band. However, the security dimension contains a critical condition, so the final score is floored at 60. This changes the interpretation from an ordinary remediation backlog to a high-debt workload requiring explicit risk treatment. The mechanism therefore satisfies R4 without forcing every security weakness to dominate the entire index.

Monotonicity follows directly from equations (1) and (2) while weights are non-negative: increasing an indicator severity cannot reduce its dimension score or the base aggregate. The critical floor is also monotonic because introducing a CAC can only retain or increase the final score. To examine weight sensitivity, each scenario was recalculated under 500 randomly generated weight vectors satisfying the 0.08–0.25 bounds. The ranking of the mature workload as lowest debt was stable, and the legacy migration remained in the upper portion of the set. Borderline workloads moved within adjacent interpretation bands in a minority of weight configurations, which is expected because weighting represents genuine business context. No valid weight vector reduced a CAC workload below its floor.

C. Worked assessment example

Consider an enterprise customer-facing workload with the following observations: privileged cloud roles are broader than required; all internet traffic passes through a managed application gateway; backups exist but restore testing is irregular; a regional failover runbook exists but has not been

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

exercised in the last year; compute capacity scales horizontally; the primary relational database is manually overprovisioned; infrastructure is mostly deployed through infrastructure-as-code; distributed tracing covers only the customer API tier; two core libraries are near end of support; resource tagging is enforced but policy exceptions do not have expiry dates.

Using the baseline rubric, the architecture team scores the seven dimensions as 45, 55, 40, 25, 35, 50, and 40. Equal weighting produces CADI=41.4, an elevated-debt workload. No indicator is rated critical, so no floor is applied. The debt vector shows that reliability and maintainability deserve more attention than performance. The team can therefore prioritize restore testing, failover exercise, dependency modernization, and privilege reduction instead of launching a broad and expensive architecture rewrite. Six months later, repeating the same evidence-based assessment creates a trend line. If reliability falls from 55 to 30 after successful failover and restore testing while other dimensions remain unchanged, CADI falls to 37.9, moving the workload into controlled debt. The value of the score is not the decimal precision; it is the consistent decision trail connecting findings, remediation, and observed movement.

VI. DISCUSSION

A. What CADI adds

CADI should be viewed as complementary to existing ATD methods. Source-code and architectural-smell approaches can estimate structural principal with a level of automation that CADI does not attempt [8]. Dependency-based measures can reveal coupling and change propagation that a workload review may miss [5]-[7]. CADI contributes a different lens: it combines architectural qualities that are routinely managed by cloud platform, security, reliability, FinOps, operations, and application teams but are often reported in separate dashboards.

This workload-level view is useful because enterprise remediation decisions are rarely made from code structure alone. A modernization program may need to choose between removing a public endpoint, testing disaster recovery, replacing an obsolete runtime, refactoring a tightly coupled service, adding telemetry, or rightsizing a database. These items differ technically but compete for the same engineering capacity. A common debt vocabulary can improve that conversation while the underlying dimension scores preserve technical detail.

B. Tradeoffs and bounded weighting

Cloud architecture is dominated by tradeoffs. Microsoft explicitly documents tradeoffs among reliability, security, cost, performance, and operational excellence [12]. Redundancy can raise cost; additional security controls can add latency or operational complexity; aggressive utilization targets can reduce resilience. A debt index that treats one universal weight vector as objectively correct would ignore these realities.

CADI therefore permits contextual weights but restricts them. A regulated financial workload may give security and governance greater weight, while an internal batch analytics system may emphasize cost and performance. Both remain visible because no dimension can fall below 8 percent. The bounds are policy choices, not empirically optimized constants. Future research should test whether different sectors require different bounds and whether weight elicitation methods such as analytic hierarchy process improve inter-rater consistency.

C. Preventing misuse

Three misuse patterns deserve attention. First, CADI should not be used to compare unrelated organizations as if it were a standardized external rating. Scores depend on workload context, indicator definitions, and evidence quality. Second, leadership should not reward teams for minimizing the score if that creates incentives to downgrade findings. Independent review and evidence retention are preferable for high-criticality systems. Third, teams should not optimize the number while ignoring architecture outcomes. The purpose is to expose debt and guide remediation, not to manufacture a favorable dashboard.

The evidence-gap rule is especially important. Architecture debt often survives because documentation, ownership, or testing is missing. Treating unknown as healthy would systematically bias the score downward. CADI instead makes uncertainty visible and creates an incentive to replace assumptions with evidence.

VII. THREATS TO VALIDITY AND LIMITATIONS

The most important limitation is the absence of industrial validation in the present study. The scenario evaluation establishes mathematical and logical behavior, not predictive validity. It does not demonstrate that a CADI score of 60 causes more incidents than a score of 40, nor that reducing CADI by ten points yields a specific financial return. Longitudinal studies are needed to test relationships with incident rate, recovery performance, cloud spend variance, change failure rate, lead time, security findings, and maintenance effort.

Construct validity is another concern. Seven dimensions simplify a complex architecture and some findings can plausibly belong to more than one dimension. The indicator guide therefore requires each finding to have one primary dimension unless the evidence shows independent consequences. Double counting should be avoided. Weight selection also introduces judgment. The bounded range limits extreme manipulation but does not remove subjectivity.

Inter-rater reliability has not yet been measured. Two experienced architects may assign different severity levels to the same evidence. Future evaluation should use multiple assessors, calculate agreement statistics, refine ambiguous rubric language, and test whether examples improve consistency. External validity is also limited because the framework has not yet been tested across different cloud providers, regulated sectors, system sizes, or hybrid environments.

Finally, CADI is intentionally not a financial debt model. The score does not represent dollars of principal or interest. Financial translation should be performed separately using remediation effort, expected loss, opportunity cost, or delay cost. Keeping those calculations separate avoids presenting an ordinal architecture score as a monetary estimate without evidence.

VIII. PRACTICAL ASSESSMENT PROCEDURE

A repeatable CADI review can be performed in six steps. First, define the workload boundary and business criticality, including data sensitivity, availability objectives, regulatory obligations, and major dependencies. Second, select the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 09 , September - 2026

baseline indicators and add only justified domain-specific indicators. Third, freeze weights and CAC rules before examining detailed findings so that scoring choices are not adjusted to obtain a preferred result. Fourth, collect evidence and score each indicator with a short rationale and owner. Fifth, calculate dimension scores, the base score, applicable floors, and remediation priorities. Sixth, retain the assessment snapshot and repeat it after a meaningful remediation interval, typically quarterly for important workloads or after major architecture change.

The review should be performed by people who understand both architecture and operations. Application architects can assess coupling and maintainability, platform engineers can verify configuration and resilience, security teams can evaluate exposure and identity, and FinOps practitioners can validate cost evidence. The framework does not require every review to become a committee exercise, but high-impact scores should be traceable to evidence rather than one person's memory.

For automation, many indicators can be pre-populated from cloud APIs, configuration policy engines, security posture tools, observability platforms, CI/CD systems, service catalogs, and cost-management exports. Automation should populate evidence, not silently decide all severities. Human judgment remains necessary where architecture intent and business context determine whether a condition is debt.

IX. CONCLUSION

Enterprise cloud architecture debt is broader than structural code debt. It accumulates in security boundaries, recovery design, capacity decisions, operational practices, aging dependencies, governance exceptions, and recurring cloud cost. Existing ATD research provides strong methods for structural measurement, while cloud well-architected frameworks provide broad review guidance. CADI connects these perspectives by expressing workload-level cloud architecture debt as a transparent seven-dimensional score with bounded contextual weighting and critical-risk floors.

The analytical evaluation shows that the proposed model behaves monotonically, distinguishes contrasting debt profiles, and prevents a concentrated critical weakness from being hidden by unrelated strengths. These are necessary properties for a useful decision-support artifact, but they are not sufficient evidence of real-world predictive validity. The next research step is therefore empirical: apply CADI across multiple enterprise workloads, measure assessor agreement, observe debt movement over time, and test relationships with operational and economic outcomes. Until that validation is complete, CADI should be used as a structured architecture-review and prioritization mechanism rather than as an industry benchmark. Its practical value lies in making accumulated cloud architecture compromises visible, comparable within a portfolio, and easier to discuss before they become expensive failures.

REFERENCES

[1] N. S. R. Alves, T. S. Mendes, M. G. de Mendonça Neto, R. O. Spínola, F. Shull, and C. Seaman, “Identification and management of technical debt: A systematic mapping study,” Information and Software Technology, vol. 70, pp. 100–121, 2016, doi: 10.1016/j.infsof.2015.10.008.

[2] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, “Managing technical debt in software engineering,” Dagstuhl Reports, vol. 6, no. 4, pp. 110–138, 2016, doi: 10.4230/DagRep.6.4.110.

[3] T. Besker, A. Martini, and J. Bosch, “Managing architectural technical debt: A unified model and systematic literature review,” Journal of Systems and Software, vol. 135, pp. 1–16, 2018, doi: 10.1016/j.jss.2017.09.025.

[4] R. Verdecchia, P. Kruchten, P. Lago, and I. Malavolta, “Building and evaluating a theory of architectural technical debt in software-intensive systems,” Journal of Systems and Software, vol. 176, 110925, 2021, doi: 10.1016/j.jss.2021.110925.

[5] R. L. Nord, I. Ozkaya, P. Kruchten, and M. Gonzalez-Rojas, “In search of a metric for managing architectural technical debt,” in 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, 2012, pp. 91–100, doi: 10.1109/WICSA-ECSA.212.17.

[6] A. MacCormack and D. J. Sturtevant, “Technical debt and system architecture: The impact of coupling on defect-related activity,” Journal of Systems and Software, vol. 120, pp. 170–182, 2016, doi: 10.1016/j.jss.2016.06.007.

[7] M. Nayebi, Y. Cai, R. Kazman, G. Ruhe, Q. Feng, C. Carlson, and F. Chew, “A longitudinal study of identifying and paying down architectural debt,” in Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, 2019, pp. 171–180.

[8] D. Sas and P. Avgeriou, “An architectural technical debt index based on machine learning and architectural smells,” IEEE Transactions on Software Engineering, vol. 49, no. 8, pp. 4169–4195, 2023, doi: 10.1109/TSE.2023.3286179.

[9] Microsoft, “Azure Well-Architected Framework,” Microsoft Learn, 2026. [Online]. Available: https://learn.microsoft.com/azure/well-architected/. Accessed: Sep. 12, 2026.

[10] Amazon Web Services, “AWS Well-Architected Framework,” AWS Documentation, 2026. [Online]. Available: https://docs.aws.amazon.com/wellarchitected/. Accessed: Sep. 12, 2026.

[11] ISO/IEC 25010:2023, Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Product quality model, International Organization for Standardization, 2023.

[12] Microsoft, “Reliability tradeoffs,” Azure Well-Architected Framework, 2026. [Online]. Available: https://learn.microsoft.com/azure/well-architected/reliability/tradeoffs. Accessed: Sep. 12, 2026.

[13] C. Pascoe, S. Quinn, and K. Scarfone, The NIST Cybersecurity Framework (CSF) 2.0, NIST CSWP 29, National Institute of Standards and Technology, 2024, doi: 10.6028/NIST.CSWP.29.

[14] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, Zero Trust Architecture, NIST Special Publication 800-207, National Institute of Standards and Technology, 2020, doi: 10.6028/NIST.SP.800-207.

[15] S. Suryawanshi, “Security architecture for agentic AI in enterprise cloud environments: A zero-trust framework for secure autonomous systems,” International Journal of Innovative Science and Research Technology, vol. 11, no. 8, 2026, doi: 10.38124/ijisrt/26aug1168.