Technical Research Paper

Engineering Interoperable, Secure and Evolvable Digital Platforms for Complex Enterprise and Government Ecosystems

A study of digital platform architecture, integration, governance and resilience

This paper examines the architectural and engineering principles required to build digital platforms that remain interoperable, secure and evolvable as they grow. It reviews international and Saudi national standards, derives eight architectural principles, and proposes a six-dimension engineering framework with a four-level assessment rubric.

Authors	Ehab Attar
Company	Al Moammar Information Systems Co. (MIS), Kingdom of Saudi Arabia
Research domain	Digital Platforms
Document type	Technical Research Paper / White Paper
Version	1.1
Publication date	12 January 2026

Published by Al Moammar Information Systems Co. (MIS)

Contents

Abstract

Digital platforms have become a foundational component of modern enterprise and government technology ecosystems. Their role has expanded beyond providing isolated digital services to enabling interaction among users, applications, data sources, infrastructure, partners and external services through shared architectural capabilities.

This evolution introduces a central engineering challenge: digital platforms must support interoperability and rapid service evolution while maintaining security, resilience, governance, and manageable dependencies between components.

This paper investigates the architectural characteristics required to build sustainable digital platforms in complex enterprise and government environments. It uses a structured qualitative research methodology based on comparative analysis of recognised architecture, cybersecurity, cloud-computing, interoperability and digital-government frameworks.

The study examines six dimensions: modular platform architecture, interoperability and API-based integration, data and service architecture, cybersecurity and identity, operational resilience, and platform governance.

The findings indicate that sustainable digital platforms should not be designed as monolithic applications, nor as collections of disconnected systems. They should instead be treated as governed digital ecosystems composed of loosely coupled services, standardised interfaces, reusable capabilities, controlled data exchange, identity-aware security and observable operational components.

The paper proposes an integrated Digital Platform Engineering Framework that can be used to assess or guide the design of enterprise and government digital platforms. The framework is intended to support technical decision-making rather than to prescribe a particular commercial technology stack, and it is extended with a four-level assessment rubric so that each dimension can be positioned on a graduated scale rather than recorded as present or absent.

Keywords: *digital platforms; enterprise architecture; interoperability; APIs; platform architecture; digital government; zero trust; resilience; integration; data architecture.*

1. Introduction

Digital transformation has changed the role of information systems within organisations. Historically, many systems were developed around individual business functions and operated with limited interaction with other applications. Modern digital services increasingly require a different model.

A digital platform may need to integrate identity services, applications, data stores, external APIs, analytics capabilities, business processes, infrastructure services, partner systems and multiple user channels.

The Digital Government Authority (DGA) in Saudi Arabia emphasises integration between business, applications, data and technology through national enterprise architecture practices. Its National Enterprise Architecture programme describes enterprise architecture as a mechanism for aligning business processes and services with applications, data and technology in support of integrated digital transformation [3]. DGA guidance on application programming interfaces similarly identifies APIs as an important enabler of digital transformation and of increased integration between digital services [4].

The engineering problem therefore extends beyond building a digital interface. The platform must remain capable of accommodating new services and integrations without creating excessive dependency, security exposure, architectural fragmentation or operational instability. This study focuses on that problem.

2. Research Problem

Digital platforms often evolve incrementally. New services are added, external systems are connected, new data sources emerge, and additional users and organisations become part of the ecosystem. Without a coherent architectural model, this growth can lead to:

- Tightly coupled systems;
- Duplicated functionality;
- Inconsistent interfaces;
- Fragmented identity mechanisms;
- Incompatible data models;
- Security exposure;
- Difficult upgrades;
- Limited observability;
- Operational dependencies that are difficult to manage.

The research problem can therefore be expressed as follows: how can digital platforms be architected so that they remain interoperable, secure, scalable, resilient and adaptable as their ecosystem expands?

3. Research Question

The principal research question is:

> *What architectural and engineering principles enable digital platforms to support secure interoperability and continuous evolution in complex enterprise and government ecosystems?*

Supporting questions are:

- How should digital platform components be separated in order to reduce dependency?
- What role do APIs and standardised interfaces play in interoperability?
- How should security be integrated into platform architecture?
- How can data and services be shared without creating uncontrolled coupling?
- What architectural characteristics improve resilience and future adaptability?
- How should platform governance influence technical architecture?

4. Research Objectives

The study has six objectives:

1. Identify the principal architectural characteristics of sustainable digital platforms.
2. Analyse the relationship between modularity and interoperability.
3. Examine APIs as a mechanism for controlled interaction between platform components and external systems.
4. Integrate cybersecurity and identity requirements into the platform architecture model.
5. Examine the role of resilience, observability and governance.
6. Develop a practical conceptual framework for evaluating or designing digital platforms.

5. Background

5.1 From applications to platforms

An application primarily delivers a defined set of functions. A platform has a broader role: it provides shared capabilities upon which multiple services, applications, organisations or user groups can operate.

ISO/IEC TS 38508:2024 describes a shared digital service platform as an environment through which ecosystem organisations can share resources, processes and capabilities in order to provide digital services. The same specification introduces the concept of a plug-and-play architecture intended to minimise dependencies between the platform core and the applications built on it, so that change can occur without requiring corresponding change throughout the ecosystem [2].

This distinction matters. The architectural value of a platform is not determined solely by the number of features it contains, but by its ability to allow capabilities to interact and evolve under controlled architectural rules.

6. Literature Review

6.1 Digital platform reference architecture

ISO/IEC 24039:2022 defines a digital-platform reference architecture focused on enabling applications to access data and services through a platform architecture. Although its stated context is smart-city platforms, its architectural treatment of data and services provides useful reference principles for broader digital-platform design [1]. Two fundamental concepts follow: data must be accessible through controlled architectural mechanisms, and services must be exposed in a reusable and structured manner.

6.2 Enterprise architecture and digital platforms

The Saudi national enterprise architecture approach emphasises alignment between business, data, applications and technology. DGA describes its national enterprise architecture framework as a common national reference supporting integrated digital transformation and consistent architecture practices across government entities [3].

The significance for digital-platform engineering is that a platform cannot be considered only from the application layer. Its design should address business architecture, application architecture, data architecture and technology architecture together, because weakness in one layer can constrain the entire platform.

6.3 Platform interoperability

Interoperability refers to the ability of independently developed systems or components to exchange information and interact meaningfully. In platform environments, interoperability occurs at several levels: technical connectivity, APIs and service interfaces, data structures, semantic interpretation, identity, and business processes.

The plug-and-play principle in ISO/IEC TS 38508:2024 addresses the same concern from a governance perspective, requiring that dependencies between the platform core and its applications be minimised so that interoperability is preserved as components evolve [2]. The implication is that interoperability should be designed deliberately rather than achieved through repeated custom connections.

6.4 API-based integration

DGA's API guideline identifies APIs as an important mechanism for accelerating digital transformation and integration [4]. An API provides a controlled contract between systems. A mature API architecture should define service ownership, authentication, authorisation, request and response structure, versioning, error handling, rate control, logging and lifecycle management.

Without governance, an API ecosystem can reproduce the very complexity it was intended to solve. API availability alone does not equal interoperability maturity; API governance is equally important.

6.5 Security architecture

Modern digital platforms frequently cross conventional organisational and network boundaries. NIST SP 800-207 recommends a Zero Trust approach in which access decisions focus on users, devices, assets and resources rather than assuming trust simply because an entity is located within a particular network boundary [5]. NIST SP 800-207A extends the same concept toward application and service identities in cloud-native and multi-cloud environments [6].

For platform architecture, this means identity and authorisation should be integral to service interaction. A platform should not rely exclusively on perimeter protection; service-to-service and user-to-service interaction should be subject to explicit authentication and authorisation.

In the Saudi context, platform security architecture also operates within the national cybersecurity control environment established by the National Cybersecurity Authority, including the essential cybersecurity controls and the specific control sets addressing data and critical systems [8]–[11]. Platform design should therefore treat these controls as architectural requirements rather than as post-implementation compliance activities.

6.6 Data protection and privacy

Platform data architecture is also shaped by data-protection obligations. The Personal Data Protection Law and its implementing regulations establish requirements relating to lawful processing, purpose limitation, data-subject rights, transfer conditions and accountability [12]. Because a platform typically consolidates data from several sources and exposes it to several consumers, these obligations affect interface design, data classification, retention and access control directly, and are most economically addressed while the architecture is being defined.

6.7 Cloud and distributed platform architecture

Cloud architectures increase the importance of clearly defined responsibilities and components. The NIST Cloud Computing Reference Architecture uses a vendor-neutral model to define cloud actors, roles and architectural relationships without prescribing a particular technical implementation [7]. This vendor-neutral principle is useful for digital platforms because it reduces architectural dependency on individual products: an architectural model should describe capabilities and relationships first, and technology products second.

6.8 Architectural evaluation and maturity models

The references examined above describe architectural principles, controls and governance expectations, but they are written primarily as normative guidance rather than as assessment instruments. ISO/IEC 24039:2022 specifies reference architecture content [1]; ISO/IEC TS 38508:2024 specifies governance implications [2]; the DGA frameworks specify national architecture and API practice [3], [4]; and the NIST and NCA publications specify security architecture and control expectations [5]–[11]. None of them provides a graduated scale against which an existing platform can be positioned.

This distinction matters in practice. An organisation that has adopted an API gateway, a central identity provider and a monitoring platform can demonstrate compliance with individual controls while still operating a tightly coupled architecture in which a single schema change propagates across unrelated services. Compliance with a control set and architectural sustainability are therefore related but not equivalent properties.

The absence of a graduated instrument is the specific gap this paper addresses. Section 9.1 accordingly extends the proposed framework with a four-level assessment rubric, so that each dimension can be positioned on a scale rather than recorded as present or absent.

7. Research Methodology

This study uses a structured qualitative comparative research method. It does not claim experimental validation or statistical testing. The methodology consists of five stages.

Stage 1 — Source selection

Authoritative technical sources were selected from the Digital Government Authority, the International Organization for Standardization, the National Institute of Standards and Technology, the National Cybersecurity Authority, and the national data-protection framework. Sources were selected on the basis of relevance to digital platforms, architecture, interoperability, APIs, cloud systems, cybersecurity and data protection.

Stage 2 — Architectural theme extraction

The selected references were reviewed in order to identify recurring architectural themes. Six themes were identified: modularity, interoperability, data and services, security, resilience, and governance.

Stage 3 — Cross-framework comparison

The themes were compared across the selected frameworks in order to identify principles that are technology-neutral and that recur across multiple architecture domains.

Stage 4 — Framework construction

The recurring principles were consolidated into the conceptual Digital Platform Engineering Framework presented in Section 9.

Stage 5 — Internal consistency review

The consolidated framework was then reviewed for internal consistency. Each dimension was checked against three criteria: whether it was supported by more than one of the selected sources, whether it could be expressed without reference to a specific product or vendor, and whether it could be observed in an existing platform through evidence such as interface definitions, dependency records, identity configuration or operational telemetry.

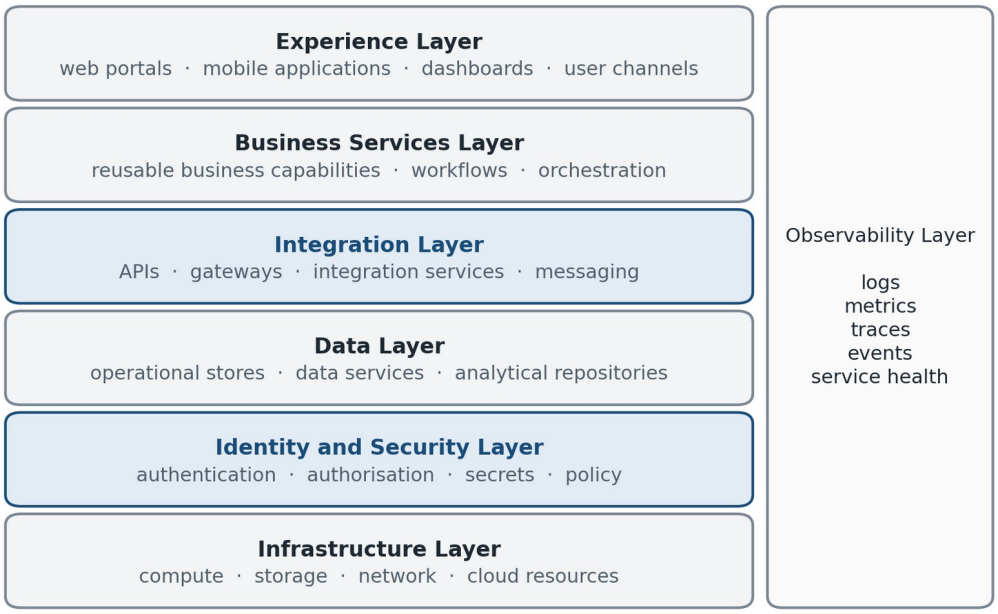
Dimensions that satisfied all three criteria were retained. This review is an analytical check on the coherence of the framework rather than empirical validation; the distinction is stated explicitly in Section 14.

8. Technical Analysis: Eight Architectural Principles

8.1 Principle 1 — Modular architecture

A platform should separate major architectural responsibilities. Figure 1 presents a conceptual layered structure in which each layer carries a distinct responsibility and is bounded by an explicit interface.

Figure 1 — Layered reference structure of a governed digital platform



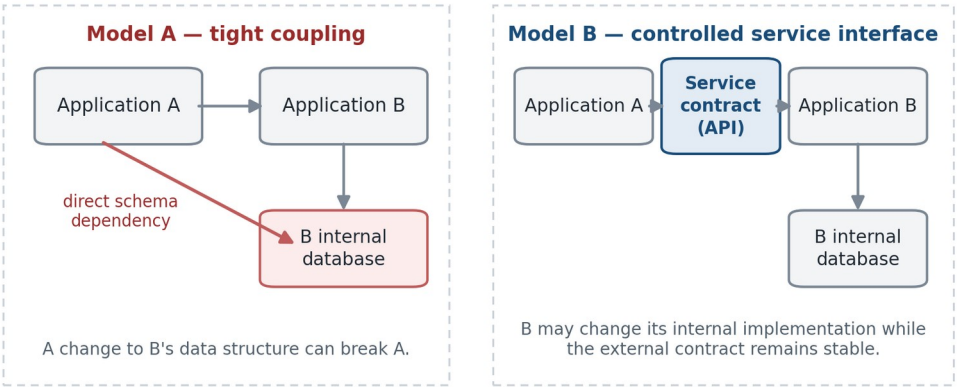
Each layer is bounded by an explicit interface, so change in one layer should not propagate through the others. Observability spans every layer rather than sitting beside them.

This separation limits the effect of change. If a user interface changes, the entire data architecture should not require redesign; likewise, if an infrastructure technology changes, business services should ideally remain stable. Observability is shown as spanning all layers, because visibility that stops at a layer boundary cannot explain a failure that crosses one.

8.2 Principle 2 — Loose coupling

A critical property of an evolvable platform is low dependency between components. Figure 2 contrasts two dependency models.

Figure 2 — Dependency models: direct data access versus a controlled service contract



In Model A, one application reads another application's internal database directly; a change to the internal data structure can therefore break an unrelated application. In Model B, the same interaction passes through a defined service contract, so the provider may change its internal implementation as long as the external contract remains stable. The second architecture is materially more adaptable, and aligns with the plug-and-play concept

described in ISO/IEC TS 38508:2024, where dependencies between the platform core and its applications should be minimised in order to preserve interoperability as components evolve [2].

8.3 Principle 3 — API-first interoperability

Interoperability should begin during architecture design. The preferable sequence is capability, then service contract, then API, then consumer — rather than system, then custom integration, then system. The first model encourages reusable services; the second tends to create point-to-point dependencies. A platform API architecture should therefore include an API catalogue and explicit lifecycle governance [4].

8.4 Principle 4 — Data as a shared but governed capability

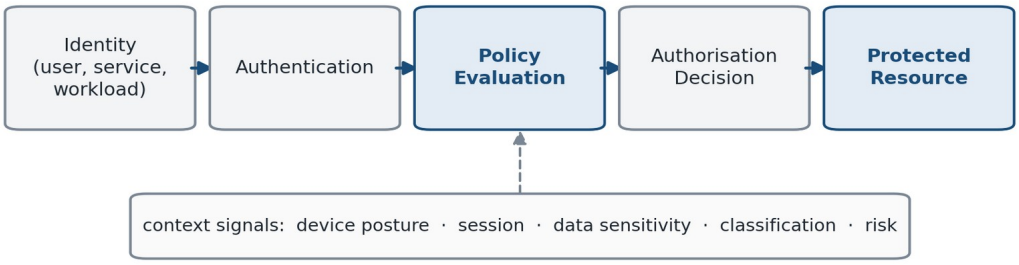
Digital platforms frequently fail when data sharing is treated as unrestricted database access. A more sustainable model distinguishes between data ownership and data consumption: one domain may own a dataset while other applications consume approved representations of that data.

Governance should address ownership, classification, quality, lineage, access rights, retention, privacy and interface definitions [8], [12]. This allows reuse without eliminating accountability.

8.5 Principle 5 — Identity-centric security

Platform security should be attached to identities and resources rather than to network location. Figure 3 shows the authorisation path for a single platform request.

Figure 3 — Identity-centric authorisation path for a single platform request



No request is trusted on the basis of network location alone. Every service-to-service and user-to-service interaction is evaluated explicitly against identity and policy.

Every service request is evaluated according to the relevant trust and access context. This model is aligned with the Zero Trust concepts promoted by NIST, which reject implicit trust based only on network position [5], and with their extension to application and service identities in cloud-native and multi-cloud environments [6].

8.6 Principle 6 — Resilience by design

High availability is not created solely by deploying redundant hardware; platform resilience depends on identifying failure domains. Potential failure domains include identity services, network connectivity, integration gateways, databases, third-party services, cloud regions and external APIs.

For each dependency, the architecture should address failure detection, containment, recovery and continuity. A platform that cannot isolate a failing component may transform a local failure into a platform-wide incident.

8.7 Principle 7 — Observability

An integrated platform requires integrated visibility. Operational teams should be able to determine which service failed, which dependent service caused the failure, which users were affected, which transaction path was involved, whether performance has degraded, and whether the failure is infrastructure-related or application-related.

This requires correlation between logs, metrics, traces, events and service-health information. Observability therefore becomes an architectural capability rather than merely an operations tool.

8.8 Principle 8 — Governance

Digital-platform governance should define both decision rights and technical boundaries. The governance model should establish platform ownership, service ownership, API ownership, data ownership, architecture standards, security responsibilities, lifecycle responsibilities and change-approval rules.

DGA's National Enterprise Architecture framework similarly positions governance and architecture practices as mechanisms for aligning digital transformation with business and technology [3], while ISO/IEC TS 38508:2024 addresses the governance implications that arise specifically when a platform is shared among ecosystem organisations [2].

9. Proposed Digital Platform Engineering Framework

Based on the comparative analysis, this paper proposes the following framework.

Table 1. The proposed Digital Platform Engineering Framework: six dimensions, guiding questions and desired architectural characteristics.

Dimension	Core question	Desired characteristic
Architecture	Can components evolve independently?	Modularity
Integration	Can systems interact through controlled interfaces?	Interoperability
Data	Is data reusable without losing governance?	Controlled data sharing
Security	Is access based on explicit identity and policy?	Zero-Trust-aligned control
Operations	Can failures be detected and isolated?	Observability and resilience
Governance	Are ownership and architecture rules defined?	Accountable evolution

The model can be summarised as: a sustainable digital platform is the combination of modular architecture, interoperability, governed data, identity-centric security, resilience and governance. This is a conceptual model derived from the literature reviewed in this paper; it is not presented as a formal DGA, ISO or NIST formula.

9.1 Applying the framework: a four-level assessment rubric

To make the framework usable as an assessment instrument, each dimension can be positioned on a four-level scale. Level 1 describes an ad hoc condition in which the property is absent or incidental. Level 2 describes a condition in which the property exists in isolated parts of the platform but is not applied consistently. Level 3 describes a condition in which the property is defined, documented and applied across the platform. Level 4 describes a condition in which the property is additionally measured, enforced and used as an input to architectural decision-making.

Table 2 applies this scale to the six dimensions. The rubric is deliberately evidence-based: each level should be assessed against observable artefacts rather than stated intent. Modularity, for example, is assessed from dependency records and interface definitions rather than from an architecture diagram, because a diagram may describe the intended structure while the deployed dependencies differ.

Table 2. Four-level assessment rubric for the six framework dimensions.

Dimension	Level 1 — Ad hoc	Level 2 — Partial	Level 3 — Defined	Level 4 — Managed
Architecture	Monolithic;	Some services	Layered boundaries	Dependencies

Dimension	Level 1 — Ad hoc	Level 2 — Partial	Level 3 — Defined	Level 4 — Managed
	components share internal state	separated; boundaries inconsistent	defined and documented	measured; violations caught in review
Integration	Point-to-point custom links	Some APIs exist; no catalogue or standards	API catalogue, standards and versioning applied	Contract testing and lifecycle rules enforced
Data	Applications read each other's databases directly	Extracts and copies circulate; ownership unclear	Ownership, classification and approved interfaces defined	Lineage and access measured; retention enforced
Security	Perimeter trust; local accounts per system	Central identity for users only	Identity-based control for both users and services	Policy evaluated per request and continuously monitored
Operations	Failures reported by users	Component-level monitoring only	Correlated logs, metrics and traces; failure domains isolated	Recovery objectives defined and tested
Governance	No defined ownership	Ownership informal or project-based	Platform, service, API and data ownership defined	Architecture conformance reviewed and enforced

An assessment produces a profile rather than a single score. A platform may reach Level 3 for security while remaining at Level 1 for data governance, and that asymmetry is itself the useful finding: it identifies which dimension currently constrains the platform's ability to evolve. Aggregating the six dimensions into one figure would conceal exactly the information the assessment is intended to reveal.

10. Findings

The study produces six principal findings.

Finding 1 — Interoperability is primarily an architecture problem. Adding integration products does not by itself create an interoperable platform. Interfaces, dependencies, ownership and data models must be designed coherently.

Finding 2 — Modularity reduces the cost of change. Low dependency between components improves the platform's ability to accommodate new applications and services.

Finding 3 — APIs require governance. Uncontrolled APIs can create new fragmentation. API ownership, standards, security and lifecycle management are required.

Finding 4 — Identity is becoming a fundamental architectural boundary. Distributed platforms make traditional perimeter-only security insufficient.

Finding 5 — Resilience requires architecture-level planning. Redundancy without failure isolation and recovery design does not ensure continuity.

Finding 6 — Digital platforms require governance beyond technology selection. The effectiveness of the platform depends on clear ownership of services, data, interfaces, security and lifecycle decisions.

11. Discussion

The research suggests that digital platforms should be understood as managed ecosystems rather than as large applications, and this distinction changes engineering priorities. In a traditional application-centric model, teams may optimise each system independently. In a platform model, the principal concern becomes the quality of interaction between capabilities.

The platform therefore derives value from reuse rather than duplication; from interfaces rather than direct dependency; from identity rather than location-based trust; from shared standards rather than project-specific conventions; and from observable services rather than opaque integrations.

This also explains why enterprise architecture and platform architecture are closely related. Enterprise architecture defines broader organisational alignment, while platform architecture converts many of those principles into technical mechanisms.

11.1 Architectural trade-offs and counter-arguments

The principles set out in Section 8 are not free of cost, and a research paper that presents them as universally beneficial would overstate the evidence. Each principle transfers effort from one part of the delivery process to another.

Modularity and loose coupling increase the number of interfaces that must be defined, versioned, tested and operated. For a platform with a small number of stable services and a single delivery team, the coordination cost of many service contracts may exceed the change-cost saving they produce. The architectural argument for modularity strengthens as the number of independent consumers, delivery teams and integration partners grows; it is weakest in small, stable, single-team environments.

API-first design carries a similar trade-off. Defining a contract before implementation slows the first delivery, and the benefit appears only when the second and third consumers arrive. Where a service will demonstrably have one consumer for its lifetime, a lighter integration may be the proportionate choice.

Identity-centric security increases the number of authorisation decisions in a request path, which introduces latency and makes the identity provider a critical dependency. This is precisely why Section 8.6 treats identity services as a failure domain: a platform that authenticates every service call has concentrated risk in the component that performs that authentication, and must design for its unavailability.

Observability introduces telemetry volume, storage cost and, where traces carry request content, its own data-protection exposure under the obligations discussed in Section 6.6 [12]. The architectural response is selective instrumentation with defined retention, not maximal collection.

These trade-offs do not weaken the framework; they define its scope. The framework describes the properties a platform requires in order to remain adaptable at ecosystem scale, and the assessment rubric in Section 9.1 should be applied with the platform's actual scale and rate of change in view.

11.2 Relationship to national digital-government objectives

The architectural properties examined in this paper also align with the direction of national digital-government practice in the Kingdom of Saudi Arabia. The DGA's national enterprise architecture programme positions architecture as a mechanism for aligning business services with applications, data and technology across government entities [3], and its API guidance positions interfaces as an enabler of integration between digital services [4].

Both positions imply architectural properties beyond individual system delivery. Cross-entity service integration presumes that participating platforms expose stable interfaces, maintain identifiable data ownership and can be held to consistent security expectations — which are, in the terms of this paper, the interoperability, data and security dimensions of the framework.

The cybersecurity control environment established by the National Cybersecurity Authority [8]–[11] and the data-protection obligations under the PDPL and its implementing regulations [12] operate in the same direction. Both are more economically satisfied by architectural design than by subsequent remediation, because both constrain interface design, data classification, access control and retention — decisions that are inexpensive while the architecture is being defined and expensive once services are in production.

The framework is therefore not an alternative to these national instruments. It is a technical lens through which their architectural implications can be assessed consistently across platforms.

12. Practical Applications

The framework proposed in this paper can be applied during new digital-platform design, modernisation of legacy platforms, integration of government or enterprise services, API-management initiatives, cloud migration, platform-security modernisation, data-platform integration and architecture assessments.

For an existing platform, the framework can also be used as a diagnostic model. High coupling combined with weak API governance may indicate integration risk; multiple parallel identity mechanisms may indicate security inconsistency; databases shared across unrelated applications may indicate data coupling; and the absence of distributed tracing may indicate limited observability.

Where an assessment identifies weaknesses across several dimensions, the sequence of remediation matters. Two dimensions are best treated as prerequisites: governance, because ownership must be established before interfaces or data can be assigned to anyone, and identity, because service-level authorisation cannot be applied consistently while multiple parallel authentication mechanisms remain in use.

Modularity and interface work can then proceed incrementally, typically by isolating the highest-traffic integration first and replacing direct data access with a defined contract. Data governance follows the same path, beginning with the datasets that already have multiple consumers. Observability is most valuable when established before, rather than during, this work, because dependency and failure evidence is required in order to verify that decoupling has actually taken effect.

This sequence is a practical ordering rather than a rule. Its purpose is to avoid the common pattern in which interface modernisation proceeds while ownership remains undefined, with the result that new interfaces reproduce the dependency structure they were intended to replace.

13. Applied Context

Al Moammar Information Systems operates within complex enterprise and government technology environments in the Kingdom of Saudi Arabia. That environment provides the applied context in which the issues examined in this paper — integration, platform architecture, data exchange, security and operational resilience — arise in practice.

This paper does not present any specific delivered project as a research experiment, and does not claim that the proposed framework was derived from a particular engagement. Maintaining that separation is deliberate: it preserves the distinction between a research publication and a corporate project profile.

14. Limitations

This paper has several limitations. First, it is a qualitative architecture study and does not include controlled experimental testing. Second, it compares architecture and governance references whose original scopes differ, including government architecture, digital platforms, cloud computing, interoperability, cybersecurity and data protection. Third, the proposed framework has not yet been quantitatively validated across a large sample of digital platforms. Fourth, no confidential customer architecture was analysed for this publication.

These limitations are deliberate: the paper is designed as a public technical publication that does not disclose client-sensitive information.

15. Future Research

Future research could extend this work toward quantitative platform interoperability maturity, API dependency metrics, resilience testing for distributed platforms, automated architecture conformance checking, artificial-

intelligence-assisted platform operations, digital-platform observability, semantic interoperability, sovereign cloud and national digital platforms, and platform security in multi-cloud environments.

A particularly useful next study would be to develop a digital platform interoperability maturity model and to test it against real enterprise platform environments.

16. Conclusion

Digital platforms are becoming central to modern enterprise and government service delivery, but their sustainability depends on architecture rather than on functionality alone. The research indicates that six characteristics are particularly important: modularity, interoperability, governed data, identity-centric security, resilience and governance.

A platform that performs well today but cannot accommodate new services, changing technologies, additional partners or evolving security requirements cannot be considered architecturally sustainable. The proposed Digital Platform Engineering Framework provides a technology-neutral approach for evaluating these characteristics, and its four-level rubric allows an existing platform to be positioned dimension by dimension, producing a profile that identifies which characteristic currently constrains the platform's ability to evolve.

For organisations operating complex digital ecosystems, the most important architectural shift is therefore from asking which systems the platform should contain, to asking how the platform can allow systems, services, data, users and organisations to interact and evolve safely over time. That question is central to the engineering of sustainable digital platforms.

References

[1] International Organization for Standardization, ISO/IEC 24039:2022 — Information technology — Smart city digital platform reference architecture — Data and service. Geneva: ISO, 2022.

[2] International Organization for Standardization, ISO/IEC TS 38508:2024 — Information technology — Governance of IT — Governance implications of the use of a shared digital service platform among ecosystem organizations. Geneva: ISO, 2024.

[3] Digital Government Authority, Kingdom of Saudi Arabia, National Enterprise Architecture (NEA). Riyadh: DGA. [Online]. Available: https://dga.gov.sa/en/programs/nea

[4] Digital Government Authority, Kingdom of Saudi Arabia, Guideline of Application Programming Interfaces (APIs). Riyadh: DGA, 2024. [Online]. Available: https://dga.gov.sa

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

[6] R. Chandramouli and Z. Butcher, A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Cloud Environments, NIST Special Publication 800-207A. Gaithersburg, MD: NIST, 2023.

[7] F. Liu, J. Tong, J. Mao, R. Bohn, J. Messina, M. Badger and D. Leaf, NIST Cloud Computing Reference Architecture, NIST Special Publication 500-292. Gaithersburg, MD: NIST, 2011. DOI: 10.6028/NIST.SP.500-292.

[8] National Cybersecurity Authority, Kingdom of Saudi Arabia, Data Cybersecurity Controls. Riyadh: NCA, 2022.

[9] National Cybersecurity Authority, Kingdom of Saudi Arabia, Critical Systems Cybersecurity Controls. Riyadh: NCA, 2022.

[10] National Cybersecurity Authority, Kingdom of Saudi Arabia, Essential Cybersecurity Controls (ECC). Riyadh: NCA. [edition and year of issue to be confirmed against the NCA portal before publication]

[11] National Cybersecurity Authority, Kingdom of Saudi Arabia, Cloud Cybersecurity Controls (CCC). Riyadh: NCA. [edition and year of issue to be confirmed against the NCA portal before publication]

[12] Personal Data Protection Law (PDPL), Kingdom of Saudi Arabia, and its Implementing Regulations issued by the Saudi Data and Artificial Intelligence Authority (SDAIA). [Royal Decree number and dates of issue and amendment to be confirmed before publication]