A PRACTICAL OPERATING GUIDE

The Modern Enterprise Architect's Handbook

A Strategic Framework for 2026 and Beyond

JUNIOR WILLIAMS

Contents

Executive summary

Enterprise architecture (EA) is not a catalogue of diagrams, a standards committee, or a final review before delivery. It is a federated decision system and operating discipline that connects strategy, capabilities, investment, information, applications, integration, platforms, security, resilience, and operating ownership to a coherent path of change.

The modern enterprise architect works horizontally across the enterprise while engineering, product, operations, security, data, finance, and risk teams bring depth within their domains. Architecture and engineering are complementary. Engineering turns decisions into working systems. Architecture improves the quality, coherence, and economic value of the decisions that shape those systems.

The 2026 operating environment raises the standard:

- Artificial intelligence (AI) systems increasingly act through tools, workflows, data, and enterprise identities, so governance must work at runtime rather than exist only in policy.
- Technology economics span cloud, software as a service, data centres, data platforms, AI, licensing, and shared services. Cost, usage, and value must be designed together.
- Resilience depends on service relationships, third parties, recoverability, observability, and practiced response, not only redundancy.
- Delivery teams need autonomy, but the enterprise still needs coherent platforms, shared controls, explicit decision rights, and credible exit options.
- Leaders expect architecture to show realized value, reduced risk, faster decisions, and improved delivery flow rather than activity counts.

This handbook provides a practical way to meet that standard. Its central method is **SCOPE**:

1. **Strategy:** define the outcome, constraints, and material risks.
2. **Current state:** understand capabilities, dependencies, evidence, economics, and operational reality.
3. **Objective target:** describe the capabilities and operating model required.
4. **Prioritized transition:** sequence decisions and change through bounded, fundable states.
5. **Evidence:** prove what was decided, delivered, operated, and measured.

The handbook then turns SCOPE into an operating model: federated roles, decision thresholds, architecture services, governance cadences, principles, transition roadmaps, agentic-AI controls, value metrics, executive communication, and a 90-day adoption plan.

What good enterprise architecture produces

Good enterprise architecture should create four results:

1. **Clarity:** leaders can see capabilities, dependencies, choices, costs, risks, and accountable owners.
2. **Coherence:** teams can make local decisions without creating avoidable enterprise fragmentation.
3. **Executability:** target states are translated into transition states, paved roads, funded work, and explicit decision gates.
4. **Evidence:** the organization can show what changed, what value was realized, what risk remains, and what should happen next.

How to use this handbook

This is an operating guide, not a compliance checklist. Use it selectively:

- Use Chapters 1 through 3 to establish a shared definition and method.

- Use Chapters 4 through 8 to build or reset an architecture practice.
- Use Chapter 9 when governing AI systems and autonomous agents.
- Use Chapters 10 and 11 to prove value and communicate decisions.
- Use Chapters 12 and 13 to develop the profession and launch the practice.
- Copy the templates in Chapter 14 into working documents and repositories.

The methods in this handbook are informed by established architecture, risk, security, software-delivery, and technology-economics frameworks and are intended to be adapted alongside them. They are practical guidance, not a replacement for any standard, contractual obligation, regulation, or professional advice.

1. The modern enterprise architect's mandate

Architecture is a decision discipline

An enterprise architect is accountable for improving decisions that cross products, domains, business units, or planning horizons. The architect does not need to own every decision. The architect makes the relevant outcomes, dependencies, options, trade-offs, risks, economics, and evidence visible to the person who does.

The practical distinction is not "architects think and engineers build." Both groups think and build. The distinction is the decision horizon.

Perspective	Primary horizon	Core question
Engineering	A service, product, platform, or implementation	How can we make this work safely and well?
Solution architecture	A defined business change or solution	How should this solution fit its context and constraints?
Domain architecture	A capability area such as data, security, integration, or applications	How should this domain evolve and enable multiple initiatives?
Enterprise architecture	Cross-domain capability, portfolio, and operating-model change	Which enterprise choices create the most coherent path to strategy?

Architecture becomes necessary when a choice affects several teams, creates a long-lived dependency, changes material risk, commits significant capital, introduces a strategic vendor, or is difficult to reverse.

The horizontal view

A modern architecture description should be able to trace one line from outcome to operation:

```
Strategy and measurable outcomes
            |
Business capabilities and value streams
            |
Information, decisions, and data ownership
            |
Applications, services, and integration
            |
Platforms, infrastructure, and engineering paths
            |
Identity, security, privacy, safety, and resilience
            |
Operations, service ownership, economics, and evidence
            |
Transition roadmap and investment decisions
```

If a component cannot be connected to a capability and outcome, its purpose should be challenged. If an outcome has no enabling capability, owner, measure, or transition path, it is still an aspiration rather than an architecture.

Seven architecture domains

The traditional business, application, data, and technology lenses remain useful. Modern practice makes integration, security, and operations explicit because many enterprise failures occur between systems, organizations, and accountabilities.

Domain	What the architect must make clear
Business and operating model	Outcomes, capabilities, value streams, policies, accountabilities, and organizational constraints
Information and data	Meaning, ownership, quality, classification, lineage, retention, access, and decision use
Applications and services	Business fit, lifecycle, duplication, coupling, product ownership, and service boundaries
Integration and interaction	APIs, events, workflows, contracts, channels, dependencies, and failure behaviour
Platforms and infrastructure	Placement, shared capabilities, engineering paths, capacity, observability, and economics
Security, identity, privacy, and safety	Trust boundaries, authority, control objectives, abuse cases, obligations, and risk ownership
Resilience and operations	Critical services, recoverability, service ownership, support, change, incident response, and evidence

Four promises of a modern EA practice

1. **Outcome before technology:** begin with the result, constraint, and material risk.
2. **Modernization without destabilization:** preserve what must remain dependable, create stable seams, and change in bounded increments.
3. **Governance that enables delivery:** decentralize reversible decisions and encode reusable controls into working paths.
4. **Evidence over assertion:** baseline, instrument, measure, and revise.

The mandate boundary

Enterprise architecture should:

- Connect strategy to capability and investment choices.
- Define cross-domain principles, target patterns, and transition directions.
- Clarify decision rights and escalation thresholds.
- Improve reuse, interoperability, resilience, security, and economic transparency.
- Help teams move faster through paved roads and reusable controls.
- Maintain a coherent view of technology lifecycle, technical debt, and exceptions.
- Show whether architecture-enabled change produced value.

Enterprise architecture should not:

- Replace accountable business, product, engineering, security, risk, or operations owners.
- Require central approval for every local design decision.
- Use framework compliance as a substitute for judgment.

- Freeze a target state while the operating environment changes.
- Claim value from diagrams, meetings, or standards alone.

2. The SCOPE method

SCOPE is a repeatable method for designing, explaining, and governing architecture. It prevents a common failure: jumping from a strategic aspiration directly to a target-state diagram without understanding current reality or the transition required.

```
STRATEGY
Outcome | Constraint | Material risk
        |
CURRENT STATE
Capabilities | Dependencies | Evidence | Economics
        |
OBJECTIVE TARGET
Required capabilities | Operating model | Guardrails
        |
PRIORITIZED TRANSITION
Sequence | Trade-offs | Decision gates | Funding
        |
EVIDENCE
Decided | Delivered | Operated | Measured | Learned
```

S - Strategy

Start with the reason the architecture exists.

Ask:

- Which business outcome must improve?
- Which customer, colleague, partner, citizen, or operational journey is affected?
- Which constraint cannot be negotiated?
- Which material risk must be reduced, accepted, transferred, or monitored?
- What is the cost of delay or the cost of doing nothing?
- What decision must leadership make?

Minimum output:

- One outcome statement
- One to three measurable indicators
- Named accountable owner
- Non-negotiable constraints
- Material risks and risk owners

A useful outcome statement is:

> *Improve [business or service outcome] from [baseline] toward [target] by [time horizon], while respecting [constraints] and keeping [material risks] within [tolerance].*

C - Current state

Current-state work is not an inventory exercise. It is evidence-based discovery of what enables or obstructs the outcome.

Ask:

- Which capabilities and value streams create the outcome?
- Which applications, data, integrations, platforms, controls, and suppliers support them?
- Where are the single points of dependency or accountability gaps?
- What is working and must be protected?
- Which assumptions are unverified?
- What does the estate cost to operate and change?
- Which risks, debt items, and exceptions have material business consequences?

Minimum output:

- Capability and service heatmap
- Dependency and information-flow view
- Application and platform lifecycle view
- Risk, debt, and exception register
- Economic and operational baseline
- Facts, assumptions, decisions, and unknowns log

O - Objective target

The target state describes required capabilities and operating conditions, not a shopping list.

Ask:

- Which capabilities must be added, improved, combined, or retired?
- How will decisions, ownership, and funding work?
- Which information must be trusted, shared, protected, or retained?
- Which services and interfaces should become reusable?
- Which controls should be built into platforms and workflows?
- How will the target be operated, observed, recovered, and exited?
- What will remain intentionally unchanged?

Minimum output:

- Target capability and operating-model view
- Domain-level target architecture
- Principles and decision criteria
- Reference patterns and paved-road needs
- Target measures and acceptance conditions

P - Prioritized transition

The transition architecture is where strategy becomes executable. It describes how the organization moves through safe, valuable states while old and new capabilities coexist.

Ask:

- Which change unlocks the most downstream value?
- Which risks or deadlines make timing material?
- Which dependencies must move first?
- Which decisions are reversible?
- Where should the organization preserve, extend, transform, or retire?
- How will data be reconciled and services cut over?
- What are the rollback, coexistence, and exit conditions?
- Which decision gates control further funding or autonomy?

Minimum output:

- Sequenced transition states
- Dependency and decision map
- Portfolio priority and funding view
- Risks, controls, and acceptance criteria by stage
- Benefits-realization plan

E - Evidence

Evidence prevents architecture from becoming a collection of intentions. Track maturity explicitly:

```
Proposed -> Approved -> Funded -> Built -> Operated -> Measured -> Improved
```

Do not report a proposed target as delivered, a built capability as adopted, or an operational service as valuable until the relevant evidence exists.

Ask:

- Which decisions were approved and by whom?
- What was built and accepted?
- Is it used in normal operations?
- Did the expected outcome improve?
- Which risks and costs moved?
- What did production teach us?
- Should the roadmap continue, change, scale, or stop?

The one-page SCOPE canvas

Field	Working answer
Strategy	Outcome, constraints, material risks, accountable owner

Field	Working answer
Current state	Critical capabilities, dependencies, evidence, costs, and problems
Objective target	Required capabilities, ownership, controls, and operating model
Prioritized transition	First move, sequence, dependencies, decision gates, and rollback
Evidence	Baseline, acceptance criteria, operational signals, and value measures

How to describe an architecture clearly

Use this sequence when writing or presenting:

1. State the outcome and constraint.
2. Describe the connected current environment.
3. Name the target capabilities and operating model.
4. Explain one material trade-off.
5. Walk the transition states.
6. State the measures and decision gates.
7. Distinguish what is proposed, approved, built, operated, and measured.

The short form is:

> *Because of this outcome and these constraints, the current environment cannot reliably provide these capabilities. I recommend this target operating model, reached through these transition states, with these owners, controls, decision gates, and measures.*

3. Seeing the whole enterprise

Begin with capabilities, not systems

A capability describes what the enterprise must be able to do. It is more stable than an organizational chart, project, or technology product. Capability maps create a shared language for strategy, investment, risk, and technology.

For each priority capability, record:

- Business outcome and value stream supported
- Accountable business owner
- Critical information and decisions
- Applications, integrations, and platforms
- Criticality and resilience expectations
- Security, privacy, safety, and regulatory obligations
- Current performance, cost, risk, and technical debt
- Planned investments and dependencies

Connect capabilities to critical services

A capability may be broad. A critical service is an end-to-end service whose disruption causes material harm. Architecture should trace the people, process, information, applications, infrastructure, facilities, suppliers, and controls that make the service work.

This view changes resilience discussions. The question is not "Is the system highly available?" It is "Can the enterprise continue or restore the service within an acceptable impact boundary?"

Current-state discovery in six passes

Pass 1: Outcomes and capability

- Confirm strategy, value streams, service outcomes, and accountable owners.
- Identify where performance, cost, risk, or change friction is material.
- Separate executive intent from current operational evidence.

Pass 2: Information and decisions

- Identify critical data, records, models, and decision points.
- Record ownership, quality, lineage, access, retention, residency, and sharing constraints.
- Find manual reconciliation, duplicate truth, and unowned information flows.

Pass 3: Applications and integration

- Map systems to capabilities and journeys.
- Identify lifecycle, duplication, coupling, interfaces, batch dependencies, and unsupported components.
- Record which systems are products, utilities, records, engagement channels, or temporary bridges.

Pass 4: Platforms and engineering paths

- Assess deployment paths, environments, observability, identity, secrets, networking, capacity, and support.
- Measure the effort required to provision a compliant production path.
- Identify repeated local solutions that should become platform capabilities.

Pass 5: Security, resilience, and operations

- Identify trust boundaries, privileged paths, material abuse cases, and control ownership.
- Map recovery objectives, dependencies, failure modes, incident paths, and operational handoffs.
- Verify whether recovery, rollback, and incident controls are tested.

Pass 6: Economics, suppliers, and change

- Allocate run and change costs to meaningful business scopes.
- Assess licences, consumption, contracts, concentration, skills, and exit costs.
- Map funded work, decision bottlenecks, delivery capacity, and benefits commitments.

Facts, assumptions, decisions, and unknowns

Every discovery should maintain four distinct records:

Record	Meaning
Fact	Supported by observable evidence

Record	Meaning
Assumption	Believed for planning but not yet verified
Decision	Approved choice with owner, rationale, and date
Unknown	Material question requiring discovery or risk treatment

Blurring these categories creates false confidence. Architecture quality improves when uncertainty is visible early.

Capability heatmap

Score heatmaps for decision support, not decoration. A simple scale can cover:

Dimension	Question
Strategic importance	How strongly does this capability affect stated priorities?
Performance gap	How far is current performance from the required outcome?
Risk exposure	What material harm could occur?
Cost and friction	What does the capability cost to run and change?
Change demand	How much funded or expected change depends on it?
Enablement value	How many other priorities improve if this capability improves?

Use a three-level signal such as stable, constrained, and critical. False precision is less useful than a clear evidence trail and agreed action.

Target-state design across the seven domains

A target state should answer:

1. Which business capability and service outcomes will improve?
2. Which information will be authoritative and how will it flow?
3. Which applications and services will own which responsibilities?
4. Which integrations and contracts will decouple change?
5. Which platform capabilities will become reusable?
6. Which security, identity, privacy, and safety controls will be inherent?
7. How will the environment be operated, observed, recovered, and economically managed?

The target must also show ownership. A capability without an operating owner, funding model, support path, or measure is not ready to become a target commitment.

4. The modern EA operating model

Purpose

The EA operating model turns strategy into executable technology choices while reducing avoidable delivery friction, risk, duplication, and cost.

It should provide services, not merely oversight:

- Strategy-to-capability translation
- Current-state intelligence
- Target and transition architecture
- Portfolio option analysis
- Reference architectures and paved roads
- Architecture decisions and exceptions
- Technology lifecycle and vendor strategy
- Risk, resilience, and economic traceability
- Benefits and outcome measurement

Federated structure

Role	Primary responsibility
Enterprise architecture leadership	Cross-domain direction, principles, portfolio coherence, strategic trade-offs, and executive decisions
Business and domain architects	Capability and transition roadmaps with business, product, data, security, and operations leaders
Platform architects and platform teams	Paved roads, reusable controls, reference implementations, operability, and developer experience
Product and solution teams	Local and reversible decisions within agreed guardrails
Security, privacy, risk, resilience, and data leaders	Risk thresholds, obligations, control objectives, evidence, and acceptance
Finance, procurement, and supplier management	Investment economics, commercial options, concentration, obligations, and exit
Architecture forum	Cross-domain, hard-to-reverse, material-risk, or exception decisions only

Decision rights

Governance should be proportional to reach, reversibility, consequence, and uncertainty.

Decision type	Default owner	Governance path
Local, reversible, and compliant	Product or solution team	Decide, record, and proceed
Multi-product within one domain	Domain leadership	Domain review and dependency check

Decision type	Default owner	Governance path
Cross-domain platform or shared contract	Enterprise leadership	Enterprise decision with option analysis
Strategic supplier or hard-to-exit commitment	Accountable executive	Enterprise, commercial, risk, and exit review
Material risk acceptance	Accountable business or risk executive	Formal risk decision
Exception to an agreed standard	Named exception owner	Rationale, compensating control, expiry, and review

Review thresholds

An enterprise review is warranted when one or more conditions apply:

- The decision affects several domains, products, or business units.
- The commitment is expensive or difficult to reverse.
- The decision creates a new system of record or shared enterprise capability.
- Material security, privacy, safety, resilience, or regulatory risk changes.
- A strategic supplier, concentration risk, or significant exit cost is introduced.
- An agreed standard or target direction requires an exception.
- The uncertainty is high enough that leadership must explicitly accept it.

Everything else should remain with the team closest to the work.

Operating cadence

Cadence	Purpose	Typical output
Continuous	Record local decisions, exceptions, evidence, and technology signals	ADRs, repository updates, automated evidence
Weekly	Resolve active cross-team design questions and unblock delivery	Decisions, owners, short actions
Monthly	Review capability hotspots, exceptions, technical debt, and emerging options	Priority updates, expired exceptions, escalations
Quarterly	Reconcile strategy, portfolio, target states, economics, and benefits	Transition roadmap and executive decision brief
Semi-annual	Reassess principles, standards, supplier concentration, and technology lifecycle	Updated standards and investment direction
Event-driven	Respond to incidents, acquisitions, regulatory change, major supplier change, or material new technology	Focused architecture and risk decision

The architecture forum charter

The forum exists to make decisions, not hear presentations.

Every item should arrive with:

- Decision requested
- Accountable decision owner
- Outcome and constraints
- Options and trade-offs
- Recommendation
- Material risks and economic implications
- Evidence available and uncertainty remaining
- Deadline and consequence of delay

If no decision is required, use a working session, repository review, or information update instead.

Standards and paved roads

A standard states an agreed constraint. A paved road makes the compliant path easier to use.

A strong paved road combines:

- Reference architecture
- Working implementation
- Automated provisioning
- Identity and access pattern
- Security and policy controls
- Observability and cost allocation
- Documentation and support
- Versioning, lifecycle, and feedback

Measure paved-road adoption, time to first compliant production deployment, exception demand, and user feedback. A standard with no usable path often creates shadow architecture.

Exception lifecycle

Every exception should contain:

- Standard or principle affected
- Business reason
- Scope and owner
- Material risk
- Compensating controls
- Approval authority
- Start and expiry dates
- Remediation or retirement action
- Review trigger

Expired exceptions are decisions waiting to be made. Track count, age, exposure, and overdue rate.

5. Principles and decision tests

Principles should change decisions. If a principle cannot help distinguish one option from another, it is only a slogan.

Twelve principles of enterprise viability

Principle	Decision test	Common failure
1. Outcome traceability	Can every material component be traced to a capability and measurable outcome?	Technology selected before the problem is defined
2. Simplification	Does the choice reduce unnecessary variation, duplication, and operational burden?	Adding a new platform without retiring or consolidating anything
3. Modularity and reuse	Are responsibilities bounded and reusable through stable contracts?	Shared code or data without clear ownership and versioning
4. Interoperability and exit	Can information and services move through documented interfaces, and is exit credible?	Calling a design vendor-neutral while accepting hidden proprietary dependencies
5. Security, privacy, and safety by design	Are trust, authority, sensitive data, abuse cases, and consequence addressed from the start?	Controls added after architecture and procurement decisions
6. Data and AI stewardship	Is information fit for purpose, governed, traceable, and used within authorized context?	Treating data access as a technical entitlement rather than a business responsibility
7. Resilience and operability	Can the service be observed, supported, recovered, and changed safely?	Designing for launch while ignoring normal operations and failure
8. Economic accountability	Are total cost, unit economics, consumption, commitments, and value visible?	Approving on acquisition cost while ignoring run, integration, and exit cost
9. Incremental and reversible change	Can value be delivered in bounded steps with learning and rollback?	One irreversible programme carrying all risk to the final cutover
10. Govern through evidence	Are decisions, controls, exceptions, and outcomes observable and reviewable?	Policy documents with no enforcement or operational signals
11. Purposeful innovation	Does novelty solve a material problem better than established options?	Pilots selected for attention rather than enterprise value
12. Named accountability	Is there an owner for the outcome, service, data, control, risk, and lifecycle?	Shared responsibility becoming unowned responsibility

Principle hierarchy

Use three levels:

1. **Enterprise principles:** durable decision intent.
2. **Standards:** required constraints for a defined scope and period.

3. **Patterns and paved roads:** practical ways to comply.

Principles should change slowly. Standards should be versioned and reassessed. Patterns should evolve through production learning.

Managing principle conflicts

Principles can conflict. Simplification may compete with local differentiation. Resilience may increase cost. Portability may reduce access to a specialized capability.

Resolve conflicts by making these items explicit:

- Outcome priority
- Risk tolerance
- Time horizon
- Reversibility
- Total economic effect
- Affected stakeholders
- Decision owner

The architect's role is not to pretend the conflict does not exist. It is to make the trade-off decision-ready.

6. From strategy to an investable roadmap

Convert strategy into capability outcomes

For each strategic priority:

1. Identify the capabilities that create the outcome.
2. Baseline current performance, cost, risk, and change demand.
3. Define the target capability and operating conditions.
4. Identify gaps and dependencies.
5. Generate options.
6. Compare value, risk, cost, timing, reversibility, and capacity.
7. Sequence transition states.
8. Assign owners, funding, measures, and decision gates.

Prioritization model

A practical scorecard can use a one-to-five scale across:

- Strategic alignment
- Customer or operational value
- Financial effect
- Risk and time criticality
- Enablement of other priorities

- Reversibility and learning value
- Delivery feasibility and capacity
- Ongoing operating cost

Weights should reflect current strategy. Do not hide judgment behind a formula. Use scoring to make assumptions and disagreement visible.

Portfolio balance

Maintain a deliberate balance among:

Portfolio category	Purpose
Strategic growth and differentiation	Create new or improved business capability
Customer and colleague experience	Improve journeys, service, and productivity
Operational efficiency	Reduce cost, friction, delay, and manual work
Risk and resilience	Reduce material exposure and improve recoverability
Foundational platforms	Create reusable capabilities and paved roads
Simplification and retirement	Remove duplication, unsupported technology, and avoidable cost
Discovery and options	Reduce uncertainty before large commitment

Foundational work should name the downstream outcomes it enables. Innovation should name the evidence required to scale. Risk work should describe business consequence, not only control gaps.

Preserve, extend, transform, or retire

Use four explicit dispositions:

- **Preserve:** keep a capability stable because it remains fit and dependable.
- **Extend:** add seams, interfaces, controls, or adjacent services without replacing the core.
- **Transform:** materially change the capability, architecture, or operating model.
- **Retire:** remove the capability or technology and its cost, risk, and dependencies.

"Modernize" without a disposition is ambiguous.

Transition-state design

Every transition state should be:

- Valuable enough to justify funding
- Safe enough to operate
- Coherent with the target direction
- Reversible or recoverable within agreed limits
- Measurable
- Explicit about old-and-new coexistence

For each state, document:

Field	Required content
Capability delivered	What becomes possible or improves
Scope	Products, services, users, data, and locations affected
Dependencies	Decisions, platforms, contracts, data, and skills required
Controls	Security, privacy, safety, resilience, and economic guardrails
Cutover and coexistence	How old and new operate together
Acceptance evidence	What proves the state is ready
Decision gate	Continue, change, scale, pause, or retire

Material roadmap trade-offs

- **Deadline versus completeness:** put critical capabilities first without pretending everything can change at once.
- **Automation versus consequence:** increase autonomy only as evidence and control improve.
- **Volume versus value:** collect and retain information because it supports decisions, obligations, or operations.
- **Ambition versus capacity:** expose staffing, skills, supplier, and dependency assumptions before commitment.
- **Modernization versus continuity:** protect critical services while changing their supporting estate.
- **Optionality versus optimization:** decide where portability matters and where specialization creates justified value.

7. Architecture artifacts that change decisions

An architecture portfolio is evidence of thinking and governance. The value of an artifact is the decision it improves.

Core artifact catalogue

Artifact	Decision served	Minimum content
Capability map and heatmap	Where should leadership focus?	Outcomes, owners, performance, risk, cost, and demand
Current-state architecture	What exists and what constrains change?	Connected domains, dependencies, evidence, and uncertainty
Target-state architecture	Which capabilities and operating conditions are required?	Domain responsibilities, principles, ownership, and measures
Transition architecture	How can change happen safely and incrementally?	States, dependencies, coexistence, controls, and gates
Architecture decision record	Why was this option selected?	Context, options, trade-offs, decision, owner, and consequences
Reference architecture	How should a recurring problem be solved?	Scope, pattern, interfaces, controls, operability, and variation points
Technology lifecycle view	Where are adoption, concentration, support, and retirement decisions needed?	Status, owner, supported uses, obligations, and exit
Standard and exception register	Which constraints apply and where are deviations accepted?	Scope, version, evidence, exceptions, owners, and expiry

Artifact	Decision served	Minimum content
Outcome scorecard	Did architecture-enabled change create value?	Baseline, target, data source, owner, cadence, and actual
Executive decision brief	What must leadership decide now?	Outcome, constraint, options, recommendation, trade-offs, ask, and measure

Architecture decision record template

```
Decision:
Status:
Date:
Owner:

Context and outcome:
Constraints:
Options considered:
Decision criteria:
Recommendation:
Trade-offs accepted:
Consequences:
Risks and controls:
Revisit trigger:
Evidence links:
```

Reference architecture quality test

A reference architecture should specify:

- Problem and intended outcomes
- Scope and non-scope
- Logical responsibilities and trust boundaries
- Required interfaces and contracts
- Security, privacy, resilience, and operational controls
- Cost and capacity considerations
- Variation points
- Working implementation or paved road
- Conformance evidence
- Version, owner, and retirement path

If teams cannot use it, test it, or ask for support, it is reference material rather than a reference architecture.

Evidence maturity

Use precise status language:

Stage	Evidence
Proposed	Option or target documented
Approved	Accountable owner made the decision
Funded	Resources and commercial authority committed
Built	Capability implemented and technically accepted

Stage	Evidence
Operated	Capability used and supported in normal operations
Measured	Outcome, cost, risk, or service effect observed
Improved	Evidence changed the next decision or design

Artifact hygiene

- Store artifacts near the decisions and work they govern.
- Prefer living views generated from authoritative data where practical.
- Separate current, target, and transition states.
- Put dates, owners, scope, and status on every material artifact.
- Link every artifact to a decision, outcome, or obligation.
- Retire obsolete diagrams and standards.
- Keep executive views simple and supporting evidence deep.

8. Modernization without destabilization

Start with capability and coupling

Legacy is not an age. A legacy constraint is a capability that is expensive, risky, or slow to change relative to business need.

Before selecting a modernization path:

- Identify the business capabilities and critical services involved.
- Map data, interfaces, batch flows, identities, operational procedures, and supplier obligations.
- Determine which behaviour must remain stable.
- Quantify run cost, change cost, incident exposure, and opportunity cost.
- Identify where a stable seam can reduce coupling.

Bounded modernization pattern

```
Observe -> Stabilize -> Create seam -> Move bounded capability
        -> Reconcile -> Prove resilience -> Scale or stop -> Retire
```

Common seams include APIs, events, replication, workflow orchestration, identity brokering, and channel abstraction. A seam is valuable only when it has ownership, service expectations, versioning, security, observability, and a retirement plan.

Build versus buy

Decide layer by layer.

Buy when:

- The capability is common and non-differentiating.
- A supplier can provide proven scale, support, security, and economics.
- Integration and exit are manageable.

Build when:

- The capability creates material differentiation.
- Specialized workflow or intellectual property matters.
- Control, latency, data, safety, or integration needs justify ownership.

Evaluate:

- Time to value
- Total cost to acquire, integrate, operate, change, and exit
- Data ownership and portability
- Resilience and service obligations
- Security and regulatory responsibility
- Skills and operating capacity
- Supplier concentration and roadmap influence

Workload placement

Cloud is a placement and operating-model choice, not a destination. Compare placement options using:

- Business criticality
- Data and sovereignty requirements
- Latency and connectivity
- Elasticity and utilization
- Availability and recovery
- Security and isolation
- Engineering experience
- Existing commitments and skills
- Unit economics and forecastability
- Exit and concentration risk
- Sustainability requirements

The FinOps Framework's 2026 evolution reinforces a broader view of technology value across cloud, software, data centres, AI, and other technology categories. Enterprise architecture should connect workload choices to business value, usage, cost, and accountability.

Data and integration modernization

Prioritize:

- Authoritative ownership and shared meaning
- Data quality at the point of creation
- Contracted APIs and events

- Lineage and provenance
- Purpose-based access
- Retention and deletion
- Reconciliation during transition
- Observability for data products and flows

Avoid replacing point-to-point integration with an unmanaged central bottleneck. Integration architecture should distribute ownership while preserving shared contracts, security, discoverability, and operational evidence.

Platform engineering and paved roads

Platform teams should provide reusable capabilities as products. Enterprise architecture should help choose which repeated needs justify a platform and which variation should remain local.

A platform is earning its place when it improves:

- Time to a compliant production path
- Reliability and recoverability
- Security and evidence automation
- Developer and operator experience
- Reuse and consistency
- Unit economics

Adoption should be earned through value, not forced only through policy.

Resilience and rollback

Every modernization should answer:

- Which critical services depend on this change?
- What can fail independently?
- What is the acceptable impact boundary?
- Are recovery time and recovery point objectives defined and tested?
- Can the change be rolled back or isolated?
- Are suppliers and operational owners included in exercises?
- What evidence proves recoverability?

9. Agentic AI governance reference architecture

Why agentic systems change the architecture

Traditional applications execute predefined logic. Agentic systems can interpret context, plan steps, select tools, use memory, and take actions through enterprise systems. The risk is shaped not only by the model but by the combination of identity, purpose, data, tools, autonomy, and consequence.

Governance must therefore apply at three levels:

1. **Management system:** ownership, policy, risk tolerance, inventory, lifecycle, and accountability.
2. **Design and release:** threat modelling, data controls, evaluation, approval, and change control.
3. **Runtime:** identity, authorization, tool policy, limits, audit, monitoring, intervention, and rollback.

Logical reference architecture

```
People / services / external events
                |
       Identity + purpose + consent
                |
             Agent gateway
                |
   Workflow / orchestration / memory
          /          |           \
 Model gateway  Context gateway  Action gateway
      |              |                |
 Approved        Governed data    Allow-listed tools
 models          and knowledge    and enterprise services

Governance control plane across every layer:
registry | owner | risk tier | policy | evaluation | audit | monitoring
human override | incident response | rollback | retirement
```

Required control plane

- Agent and use-case registry with named business and technical owners
- Purpose, intended users, affected parties, and prohibited uses
- Risk tier based on autonomy, data sensitivity, tool reach, reversibility, and consequence
- Separate user and workload identities with least privilege
- Approved model and model-gateway policy
- Governed context sources with provenance, classification, freshness, and retention
- Allow-listed tools and enterprise actions
- Prompt, policy, memory, model, and tool versioning
- Evaluation, adversarial testing, release approval, and change control
- Runtime authorization, rate limits, spend limits, and segregation of duties
- Proportionate, tamper-evident authorization and action records with data minimization, access control, redaction, and retention
- Monitoring for quality, drift, unsafe action, control bypass, and anomalous cost
- Human override, kill switch, incident response, rollback, and retirement

Audit evidence should capture inputs, policy decisions, authorizations, tool calls, actions, outcomes, and exceptions needed for accountability. It should not require hidden chain-of-thought or indiscriminate retention of sensitive prompts, personal data, or credentials.

Autonomy ladder

Level	Behaviour	Default control
1. Assist	Read-only retrieval, summarization, or drafting	Identity, data access, disclosure, and logging

Level	Behaviour	Default control
2. Recommend	Produces a recommendation; an accountable person decides	Evidence, uncertainty, review, and recorded decision
3. Act within bounds	Performs reversible, pre-authorized actions	Runtime policy, limits, audit, and exception handling
4. High-impact action	Creates material financial, customer, employment, safety, or regulatory effect	Approval by an accountable human, or by a separately governed deterministic control explicitly approved by the risk owner; no agent self-approval
5. Prohibited	Uses unacceptable authority, data, or tools	Block, alert, investigate, and redesign

Autonomy is earned through evidence. Moving to a higher level should require stronger evaluation, monitoring, authorization, intervention, and business accountability.

Lifecycle

```
Discover -> Classify -> Design -> Evaluate -> Approve
         -> Deploy -> Monitor -> Intervene -> Reassess -> Retire
```

Discover and classify

- Define intended outcome and users.
- Name accountable owners.
- Identify affected parties and material consequences.
- Map data, models, memory, tools, suppliers, and jurisdictions.
- Assign risk and autonomy tiers.

Design

- Minimize data and tool access.
- Separate model reasoning from authorization.
- Treat retrieved content and tool output as untrusted inputs.
- Define human authority and fallback.
- Design logging, cost controls, rollback, and retirement.

Evaluate

- Test task quality and failure behaviour.
- Test prompt injection, data leakage, unsafe tool use, privilege escalation, and control bypass.
- Test fairness, privacy, safety, security, reliability, and explainability as relevant to context.
- Test with realistic workflows, data boundaries, and users.
- Record known limitations and non-measurable risks.

Approve and deploy

- Confirm that residual risk is within tolerance.
- Bind identity, purpose, model, policy, tools, and data to a release.

- Require approval by an accountable human, or a separately governed deterministic control explicitly approved by the risk owner, for material actions. An agent must not approve itself or another agent within the same trust boundary.
- Make runtime evidence available to operations, security, risk, and business owners.

Monitor and intervene

- Monitor quality, drift, cost, latency, policy decisions, tool actions, overrides, and incidents.
- Re-evaluate after model, prompt, data, tool, policy, or context changes.
- Revoke identity or tool access quickly.
- Maintain kill switch, safe fallback, and rollback procedures.

Retire

- Disable identities, tools, schedules, and integrations.
- Preserve required evidence.
- Delete data and memory according to policy.
- Reassign or terminate ownership.
- Confirm that dependent workflows have a safe alternative.

Prompt injection as a business-process compromise

This handbook treats prompt injection as more than a model-quality issue. When a malicious or untrusted instruction changes an agent's use of data or tools, it can override business logic, approval boundaries, or operating policy. That is a potential business-process compromise.

The design response is layered:

- Treat external content as untrusted.
- Keep authorization outside model output.
- Limit tool and data permissions.
- Validate structured inputs and outputs.
- Apply runtime policy to every action.
- Require approval for material consequences.
- Monitor and rehearse incident response.

Runtime authorization decision

```
Allow action only when:
user identity + agent identity + approved purpose
+ data classification + requested tool + action scope
+ current risk state + consequence + approval condition
all satisfy runtime policy.
```

A policy document alone cannot govern an autonomous action.

Framework alignment

Organizations can adapt this reference architecture alongside the NIST AI Risk Management Framework functions of Govern, Map, Measure, and Manage; the NIST Generative AI Profile; ISO/IEC 42001's AI management-system requirements; OWASP guidance for LLM and agentic applications; and MITRE ATLAS threat knowledge. It is not a formal crosswalk or claim of conformance. Apply the controls appropriate to the use case, sector, jurisdiction, and risk. NIST states that AI RMF 1.0 is under revision at publication.

10. Measuring enterprise architecture value

Measure outcomes, not architecture activity

Do not use diagrams produced, standards published, reviews held, or projects touched as primary proof of value. Those measures may describe workload, but not enterprise effect.

Use a balanced scorecard with six families.

Business value

Metric	Why it matters
Realized benefit versus committed	Tests whether architecture-enabled investment produced value
Customer, colleague, or service outcome	Connects change to the people and operations affected
Unit cost per product, service, case, or transaction	Makes economics visible
Revenue protected or enabled	Connects capability and resilience to enterprise value
Loss avoided with documented basis	Makes risk treatment economically legible without inventing certainty

Delivery flow and decision quality

Metric	Why it matters
Median lead time for material architecture decisions	Reveals governance friction
Concept-to-production lead time	Measures the end-to-end delivery system
Time to provision a compliant production path	Tests paved-road effectiveness
Decision reversal caused by missing evidence	Exposes discovery and governance quality
Age of unresolved cross-domain decisions	Identifies portfolio delay

Coherence and simplification

Metric	Why it matters
Reuse of APIs, services, data products, and controls	Reduces repeated effort and inconsistency

Metric	Why it matters
Duplicate applications or capabilities retired	Converts architecture into simplification
Paved-road adoption and satisfaction	Tests whether standards are usable
Unsupported technology exposure	Makes lifecycle risk visible
Integration contracts with named owners and service levels	Measures operational coherence

Risk and resilience

Metric	Why it matters
Critical services meeting tested recovery objectives	Measures recoverability rather than documentation
Change failure rate and time to restore	Connects delivery speed to operational confidence
Material architecture risk and debt exposure	Prioritizes investment by consequence
Exception count, age, and overdue-expiry rate	Prevents permanent temporary decisions
Controls producing automated evidence	Reduces assurance effort and improves freshness

Technology economics

Metric	Why it matters
Cost allocated to accountable products or services	Enables ownership and comparison
Forecast variance	Tests economic control
Idle, duplicate, or underused spend	Identifies avoidable cost
Unit economics by meaningful business scope	Connects consumption to value
Exit cost and concentration exposure	Makes strategic dependency visible

Agentic AI

Metric	Why it matters
Agents with owner, purpose, identity, and risk tier	Establishes accountability
Evaluation pass rate by use case and version	Tracks quality and safety
Material actions covered by runtime policy and audit	Measures enforceable governance
Unauthorized or approval-bypass attempts	Surfaces control pressure
Time to suspend an agent or revoke a tool	Tests operational control
Quality, intervention, incident, and unit-cost rates by autonomy level	Shows whether autonomy is earning trust

Metric contract

Every metric should have:

- Decision it supports
- Clear definition and calculation
- Baseline and target
- Data source
- Accountable owner
- Reporting cadence
- Segmentation required
- Known limitations
- Gaming or unintended-consequence risk
- Trigger for action

Leading and lagging indicators

Use both.

- Leading indicators: decision lead time, paved-road adoption, exception age, evaluation coverage, recovery-test completion.
- Lagging indicators: realized benefit, incident impact, customer outcome, cost per transaction, recovery performance.

Leading indicators show whether the system is changing. Lagging indicators show whether the enterprise benefited.

11. Executive communication and influence

The five-item executive page

Put five things on the main page:

1. Outcome sought
2. Material current-state constraint
3. Target capability
4. Decision or investment required
5. Measure and accountable owner

Keep detailed diagrams, standards, control mappings, and vendor comparisons in supporting material.

Seven-step decision narrative

1. **Business problem:** what is not working or cannot scale?
2. **Urgency:** what is the cost, risk, or lost option if nothing changes?
3. **Target outcome:** what will be measurably different?
4. **Architecture capability:** which connected capabilities create the result?
5. **Options:** which credible paths were considered?

6. **Trade-offs:** what cost, risk, constraint, or dependency is accepted?
7. **Recommendation and ask:** which decision is required, from whom, and by when?

Executive decision brief

```
Decision required:
Decision owner:
Decision deadline:

Outcome:
Current constraint:
Options:
Recommendation:
Trade-offs:
Investment and economic effect:
Material risks and controls:
Measure and owner:
Consequence of delay:
```

Communicate at the audience's decision altitude

The objective is not to remove technical substance. It is to match detail to the decision.

Audience	Lead with	Supporting depth
Board or executive committee	Outcome, exposure, investment, options, accountability	Business capability, material dependencies, assurance
Portfolio and business leadership	Capability, roadmap, economics, dependencies, benefits	Target and transition architecture
Engineering and operations	Contracts, constraints, quality attributes, operability, evidence	Reference patterns and implementation detail
Security, privacy, risk, and audit	Threats, obligations, control objectives, residual risk, evidence	Control design and test results
Finance and procurement	Unit economics, commitments, supplier risk, concentration, exit	Cost model and commercial options

Influence without formal authority

Architecture earns influence by:

- Clarifying decisions before meetings
- Bringing credible options rather than one predetermined answer
- Disclosing trade-offs and uncertainty
- Making constraints practical for delivery teams
- Helping leaders see second-order effects
- Following decisions through delivery and operations
- Changing guidance when evidence changes

When stakeholders disagree

1. Restate the shared outcome.
2. Separate facts, assumptions, preferences, and constraints.

3. Compare options against agreed criteria.
4. Identify whether the decision is reversible.
5. Name the accountable owner.
6. Record the decision and revisit trigger.
7. Escalate only when a defined threshold is crossed.

Consensus is useful, but accountability cannot be replaced by endless alignment.

12. The architect's competency model

Technical knowledge is the foundation, not the destination. Modern enterprise architects need seven connected competencies.

Competency	What mastery looks like
Technical breadth	Understands how business, data, applications, integration, platforms, security, resilience, and AI interact
Business acumen	Connects capabilities and technology choices to strategy, operating models, economics, and market or service outcomes
Executive presence	Communicates with clarity, calm, relevance, and sound judgment
Influence and investment storytelling	Builds a credible case for funding, sequencing, and change without exaggeration
Leadership	Creates direction and alignment across teams without relying only on hierarchy
Communication	Makes complex decisions understandable at several levels of depth
Stakeholder management	Understands incentives, constraints, decision rights, and what success means to affected groups

Development practices

- Study business models, financial statements, operating measures, and customer journeys.
- Rotate through delivery, operations, security, data, and commercial decisions.
- Write architecture decision records that expose trade-offs.
- Practice one-page executive briefs.
- Review production incidents and benefits outcomes, not only project designs.
- Maintain a portfolio showing problem, options, decision, transition, and measured result.
- Teach patterns and coach teams.
- Ask for feedback on clarity, usefulness, and decision impact.

The architect's portfolio

A strong portfolio demonstrates:

- A business problem and operating context
- Current-state evidence
- Options and trade-offs

- Target and transition architecture
- Decisions and ownership
- Delivery and operational evidence
- Measured outcomes and learning

Screenshots of tools and lists of technologies may support the story, but they do not replace evidence of decision quality.

13. Establishing or resetting EA in 90 days

Days 1-30: listen, map, and baseline

Actions:

- Complete structured listening across business, product, engineering, data, security, operations, risk, finance, and procurement.
- Clarify mandate, decision rights, success measures, and escalation thresholds.
- Map strategic outcomes, critical capabilities, services, dependencies, investments, risks, and decision bottlenecks.
- Baseline a small architecture value scorecard.
- Establish facts, assumptions, decisions, and unknowns.

Outputs:

- Stakeholder and decision map
- Capability and service heatmap
- Current-state hypothesis
- Baseline scorecard
- Open-decision register
- Initial technology lifecycle and exception view

Days 31-60: align choices and operating model

Actions:

- Confirm the capability hotspots that matter most.
- Agree federated roles and review thresholds.
- Define the minimum architecture services.
- Identify two or three paved roads that could reduce immediate friction.
- Build a transition roadmap showing preserve, extend, transform, and retire decisions.
- Select one bounded proof point.

Outputs:

- Decision-rights model
- Architecture forum charter and cadence
- Transition roadmap

- Initial principles and standards
- Paved-road backlog
- Bounded proof-point charter

Days 61-90: prove the model

Actions:

- Apply SCOPE to the bounded priority.
- Make one material decision faster and with better evidence.
- Instrument delivery and production signals.
- Demonstrate one reusable architecture path or control.
- Review the result with accountable leaders and delivery teams.
- Publish the next two quarters of priorities and decisions.

Outputs:

- First proven pattern
- Decision and evidence pack
- Updated scorecard
- Two-quarter roadmap
- Executive decision brief
- Improvement backlog

The 90-day standard

By day 90, the practice should have:

- A trusted baseline
- Explicit decision rights
- A measurable transition roadmap
- One proven architecture path
- Evidence that the practice improved a real decision or delivery outcome

It should not claim to have redesigned the enterprise.

Maturity progression

Stage	Characteristics
0. Reactive	Architecture appears late and depends on individual intervention
1. Visible	Decisions, owners, current-state evidence, and priorities are recorded
2. Repeatable	SCOPE, decision rights, standards, exceptions, and cadences are consistently used
3. Enabled	Paved roads, automated controls, and platform capabilities reduce delivery friction
4. Measured	Business, delivery, risk, resilience, and economic outcomes influence roadmaps
5. Adaptive	Production evidence and strategic change continuously reshape the architecture

Common failure modes

- Starting with an enterprise-wide inventory that never becomes a decision.
- Centralizing every review.
- Creating principles without tests or paved roads.
- Publishing target states without transition economics.
- Measuring architecture workload instead of outcomes.
- Treating platform adoption as a mandate rather than a product problem.
- Ignoring operations, recovery, and retirement.
- Governing AI models while leaving agent identities, tools, and runtime actions uncontrolled.
- Hiding uncertainty to create an appearance of completeness.

14. Field toolkit

A. SCOPE canvas

```
Initiative or capability:
Accountable owner:
Architecture owner:
Date and status:

STRATEGY
Outcome:
Baseline and target:
Constraints:
Material risks:
Decision required:

CURRENT STATE
Capabilities:
Information and decisions:
Applications and integrations:
Platforms:
Security and identity:
Resilience and operations:
Economics and suppliers:
Facts, assumptions, and unknowns:

OBJECTIVE TARGET
Required capabilities:
Operating model and ownership:
Target measures:
Principles and guardrails:
What remains unchanged:

PRIORITIZED TRANSITION
First bounded move:
Transition states:
Dependencies:
Trade-offs:
Coexistence and rollback:
Decision gates:

EVIDENCE
Acceptance criteria:
Operational signals:
Value measures:
Review cadence:
```

B. Capability heatmap

Capability	Owner	Strategic importance	Performance gap	Risk	Cost/friction	Change demand	Action
[Capability]	[Owner]	[L/M/H]	[L/M/H]	[L/M/H]	[L/M/H]	[L/M/H]	Preserve / Extend / Transform / Retire

C. Architecture decision record

```
ADR number and title:
Status:
Date:
Decision owner:

Context:
Outcome:
Constraints:
Options:
Decision criteria:
Recommendation:
Trade-offs:
Consequences:
Risks and controls:
Revisit trigger:
Evidence:
```

D. Transition-roadmap record

Transition state	Capability delivered	Dependencies	Controls	Acceptance evidence	Decision gate
Now	[Value available now]	[Dependencies]	[Controls]	[Evidence]	Continue / Change / Stop
Next	[Next bounded value]	[Dependencies]	[Controls]	[Evidence]	Scale / Pause
Later	[Target capability]	[Dependencies]	[Controls]	[Evidence]	Operate / Retire old state

E. Exception record

```
Exception:
Standard or principle:
Business reason:
Scope:
Owner:
Risk:
Compensating controls:
Approver:
Start date:
Expiry date:
Remediation or retirement:
Review trigger:
Evidence:
```

F. Agent risk-tier worksheet

Factor	Questions
Purpose	Is the use authorized, necessary, and clearly bounded?
Affected parties	Who could benefit or be harmed?
Data	What can be read, inferred, retained, or disclosed?
Tools	Which systems and actions are reachable?
Autonomy	Does the agent assist, recommend, or act?
Consequence	Could action create material financial, customer, employment, safety, legal, or operational effect?
Reversibility	Can the action be undone quickly and completely?
Oversight	Which approval, monitoring, and intervention are required?
Suppliers	Which model, tool, data, and service providers are involved?
Evidence	Which evaluations and runtime signals prove control?

The autonomy ladder does not replace risk classification. Use the organization's existing risk method or define a separate tiering model with these outputs.

```
Risk-tier decision:
Tier:
Rationale:
Mandatory controls:
Approval authority:
Prohibited actions:
Review or re-tier trigger:
Evidence required:
```

G. Executive decision brief

```
Decision required:
Owner and deadline:
Outcome:
Current constraint:
Options:
Recommendation:
Trade-offs:
Investment:
Material risks and controls:
Measure:
Consequence of delay:
```

H. EA scorecard starter

Family	Metric	Baseline	Target	Data source	Owner	Cadence
Business value	[Metric]	[Value]	[Value]	[Source]	[Owner]	[Cadence]
Delivery flow	[Metric]	[Value]	[Value]	[Source]	[Owner]	[Cadence]
Coherence	[Metric]	[Value]	[Value]	[Source]	[Owner]	[Cadence]
Risk and resilience	[Metric]	[Value]	[Value]	[Source]	[Owner]	[Cadence]
Economics	[Metric]	[Value]	[Value]	[Source]	[Owner]	[Cadence]

Family	Metric	Baseline	Target	Data source	Owner	Cadence
Agentic AI	[Metric]	[Value]	[Value]	[Source]	[Owner]	[Cadence]

I. Architecture review checklist

Outcome and ownership

- Is the outcome measurable?
- Is there an accountable business owner?
- Are constraints and material risks explicit?
- Is a decision actually required?

Current state

- Are facts separated from assumptions?
- Are capabilities, information, applications, integrations, platforms, controls, operations, suppliers, and economics connected?
- Is what must be preserved identified?

Target state

- Are required capabilities and ownership clear?
- Are security, privacy, safety, resilience, operations, and economics designed in?
- Are interoperability and exit considered?

Transition

- Are transition states valuable and operable?
- Are dependencies, coexistence, reconciliation, and rollback explicit?
- Are decision gates and funding conditions clear?

Evidence

- Are baseline, acceptance, operational, and value measures defined?
- Are evidence sources and owners named?
- Is the delivery status stated precisely?

15. The modern architect's operating doctrine

1. Start with the outcome, not the technology.
2. See the enterprise horizontally while respecting domain depth.
3. Make capabilities, dependencies, economics, and ownership visible.
4. Preserve what must remain dependable.
5. Build target states with transition states.
6. Decentralize reversible decisions.
7. Escalate cross-domain, hard-to-reverse, and material-risk choices.
8. Turn standards into paved roads.

9. Treat exceptions as owned, time-bound decisions.
10. Design security, resilience, operations, and exit from the start.
11. Govern AI through identity, purpose, data, tools, consequence, and runtime policy.
12. Increase autonomy only as evidence and control improve.
13. Measure realized outcomes, not architecture activity.
14. Communicate options and trade-offs with clarity.
15. Change the architecture when evidence changes.

> *The goal is not a perfectly documented enterprise. The goal is an enterprise that can make high-quality technology decisions, execute change safely, and prove the result.*

Framework alignment and sources

This handbook is an author-developed operating guide informed by established public frameworks and standards. It does not reproduce or replace them.

- The Open Group - TOGAF Standard, 10th Edition - enterprise-architecture method and configurable practice guidance.
- NIST Cybersecurity Framework 2.0 - cybersecurity outcomes and the Govern, Identify, Protect, Detect, Respond, and Recover functions.
- NIST AI Risk Management Framework Playbook - suggested actions aligned to Govern, Map, Measure, and Manage. NIST states that AI RMF 1.0 is under revision at publication.
- NIST Generative AI Profile, NIST AI 600-1 - cross-sector generative-AI risk guidance.
- ISO/IEC 42001:2023 - requirements for an AI management system.
- NIST Secure Software Development Framework 1.1 - high-level secure software-development practices.
- CISA Zero Trust Maturity Model 2.0 - zero-trust maturity guidance.
- DORA metrics guidance - five software-delivery metrics grouped under throughput and instability, with guidance to apply them in context.
- FinOps Framework and 2026 Framework update - technology-value, economic-accountability, and cross-technology operating guidance.
- OWASP Top 10 for LLM Applications and Agentic AI threats and mitigations - application and agentic-AI security risks.
- MITRE ATLAS - adversary tactics and techniques for AI-enabled systems.
- European Union Artificial Intelligence Act - jurisdiction-specific legal requirements with phased application.

All links were reviewed on 27 July 2026.

About Junior Williams

Junior Williams is an enterprise architect and technology advisor focused on turning complex technology, cybersecurity, and AI decisions into coherent architecture, executable roadmaps, and measurable outcomes.

Areas of focus: Systems and Enterprise Architecture | Cyber and AI Risk **Book an intro call** **Web:** trustcyber.ca

This handbook provides general professional guidance. It is not legal, regulatory, financial, or sector-specific advice. Organizations should adapt the methods to their operating context, risk tolerance, obligations, and applicable standards.