The Architecture Advantage: Open Systems and Lock-In in Mission-Software Roll-Ups

An acquisition and integration framework for modularity, programme access, technical rights and evidence-led valuation

Authored by Chennakeshav Adya

Independent Researcher

September 2026

Abstract

Mission-software acquirers often describe a target as open, modular, standards-based or interoperable. Those labels do not establish that a buyer can replace components, integrate another product, compete sustainment, deploy across programmes or preserve performance after a transaction. Architecture creates commercial value only when interfaces, technical rights, software delivery, security evidence, customer permissions and operational test results support the claimed freedom of action.

This paper develops an Architecture Advantage Framework for mission-software roll-ups. It separates interface publication from practical substitutability; code possession from executable technical rights; laboratory conformance from operational interoperability; and product revenue from programme access. The framework maps mission threads, modules, interfaces, data models, build environments, infrastructure, security controls, customer dependencies and change authority. It then tests six transaction questions: whether the target can be integrated without breaking accepted capability; whether components can be replaced competitively; whether the acquirer may lawfully use and share the required artefacts; whether the combined platform can enter new programmes; whether software can be delivered and sustained at the claimed cadence; and whether the resulting revenue, margin and cash remain supportable.

A wholly hypothetical case covers four mission-software companies with USD 62 million of combined annual revenue, USD 178 million of management-reported backlog and USD 21.5 million of proposed annual buyer revenue and synergy. Evidence review retains USD 93 million of funded and executable backlog and USD 8.1 million of annual buyer revenue and synergy after adjustments for interface maturity, customer-specific forks, technical rights, accreditation, hosting, integration effort, key-person dependency and cash conversion. Every figure is an illustrative management assumption. It is not observed company data, a market benchmark, a forecast, a military assessment or a valuation opinion.

The evidence base includes current U.S. Department of Defense modular open systems guidance; U.S. Government Accountability Office work on MOSA implementation and data rights; Defense Federal Acquisition Regulation guidance on software and documentation deliverables; NATO interoperability standards and exercises; U.S. Air Force Open Mission Systems material; current export-control sources; NIST secure software guidance; and accounting requirements for business combinations, fair value, revenue and intangible assets. The conclusion is that architecture advantage is an evidenced option set.

A buyer should pay for verified substitution, integration, deployment and sustainment rights, while placing unproven programme access and future interoperability behind milestones.

JEL Classification: G34, H56, L14, L64, O32, O38

Keywords: mission software, modular open systems, software architecture, vendor lock-in, defence M&A, data rights, interoperability, programme access, roll-up integration

This paper is an educational and structural analysis prepared for research purposes. It is not investment advice, an offer, or a solicitation. It contains no client information.

1. DEFINE THE ACQUISITION DECISION

The acquisition decision is whether a group of mission-software businesses can become a coherent, supportable and commercially expandable platform. The buyer needs evidence that applications, data, infrastructure and interfaces can be integrated without degrading accepted mission outcomes or violating customer, security, export or intellectual-property constraints.

An architecture described as open can still depend on a proprietary gateway, an undocumented data model, a customer-owned test environment or engineers who hold essential knowledge outside controlled repositories. A proprietary component can remain valuable when it performs a differentiated function and the buyer has durable rights to operate, modify and support it. The diligence question concerns freedom of action within each mission and contract.

The framework tests architecture at four levels. The technical level asks whether modules and interfaces work as documented. The legal level asks whether the acquirer may use and share the relevant artefacts. The operational level asks whether customers have accepted the configuration. The commercial level asks whether architecture reduces integration time, supports programme access and produces collected cash.

Table 1. Evidence required before architecture receives transaction value

Evidence layer	Core question	Minimum record	Transaction consequence
Mission	Which operational outcome does the software support?	Mission thread, user, environment, safety and failure consequence	Defines required performance
Architecture	Can modules be integrated or replaced without uncontrolled effects?	Interfaces, dependencies, conformance tests and performance results	Defines technical freedom
Rights	May the buyer use, modify, disclose and deploy the required artefacts?	Contracts, licences, markings, assignments and government rights	Defines lawful freedom
Delivery	Can the combined business build, test, accredit and release software?	Repositories, pipeline, test evidence, security approvals and rollback	Defines execution capacity
Programme access	Can the product enter and remain within target programmes?	Standards profile, sponsor, contract route, acceptance and funding	Defines executable expansion
Economics	Does the architecture produce durable revenue, margin and cash?	Cost-to-integrate, support cost, backlog, renewal and cash evidence	Defines price and terms

Each evidence layer should be connected to a defined mission, customer and software configuration.

2. MAP THE MISSION THREAD BEFORE THE SOFTWARE STACK

Mission software exists within an operational thread that can include sensing, communication, identity, data fusion, planning, decision support, command, action and assessment. A software module can perform well in isolation while failing to meet timing, assurance or data requirements across the whole thread. Diligence should therefore begin with the operational outcome and work backward through the technical stack.

The mission map should name the user, decision, latency, availability, data sensitivity, degraded mode and consequence of failure. It should identify which systems provide data, which consume outputs and which authority approves change. The map should distinguish deployed capability from laboratory demonstration, funded development and product roadmap.

This sequence prevents a buyer from valuing interface counts or software releases without understanding their decision relevance. It also exposes duplicated functions across targets. Two products may appear complementary at the marketing level while competing for the same place in the mission thread or depending on incompatible sources of truth.

3. DEFINE OPENNESS AS A MEASURABLE PROPERTY

Openness is a set of attributes rather than a binary label. A buyer should assess modularity, interface specification, implementation access, conformance, substitutability, standards governance, data portability, deployment portability and technical rights. Each attribute can mature independently.

The U.S. Department of Defense describes a modular open systems approach as an integrated business and technical strategy for competitive and affordable acquisition and sustainment over the lifecycle. GAO has reported that key interfaces should be selected where components are likely to change, fail, require replacement or support interoperability. This approach recognises that opening every interface can add cost and complexity without producing useful competition.

A transaction model should score each material interface against a defined use case. Published syntax has little value when semantics remain proprietary. Source access has limited value when the buyer lacks a reproducible build. A standard can support competition only when suppliers can obtain it, implement it and demonstrate conformance in the required environment.

4. BUILD THE ARCHITECTURE AND DEPENDENCY REGISTER

The architecture register should list every material application, service, module, interface, data store, model, library, operating system, cloud service, hardware dependency and external system. It should state owner, version, environment, classification, licence, customer, support status and replacement path.

Dependency discovery should combine design records with repository and runtime evidence. Software bills of materials, package manifests, infrastructure definitions, API catalogues, network flows and deployment inventories can reveal dependencies that diagrams omit. Customer-specific scripts and configuration should be treated as product dependencies when service delivery relies on them.

The register should mark single points of control. These include proprietary gateways, one-person build knowledge, expiring certificates, unavailable test equipment, customer-furnished components and third-party licences that cannot transfer. Each dependency requires an owner and a transaction response.

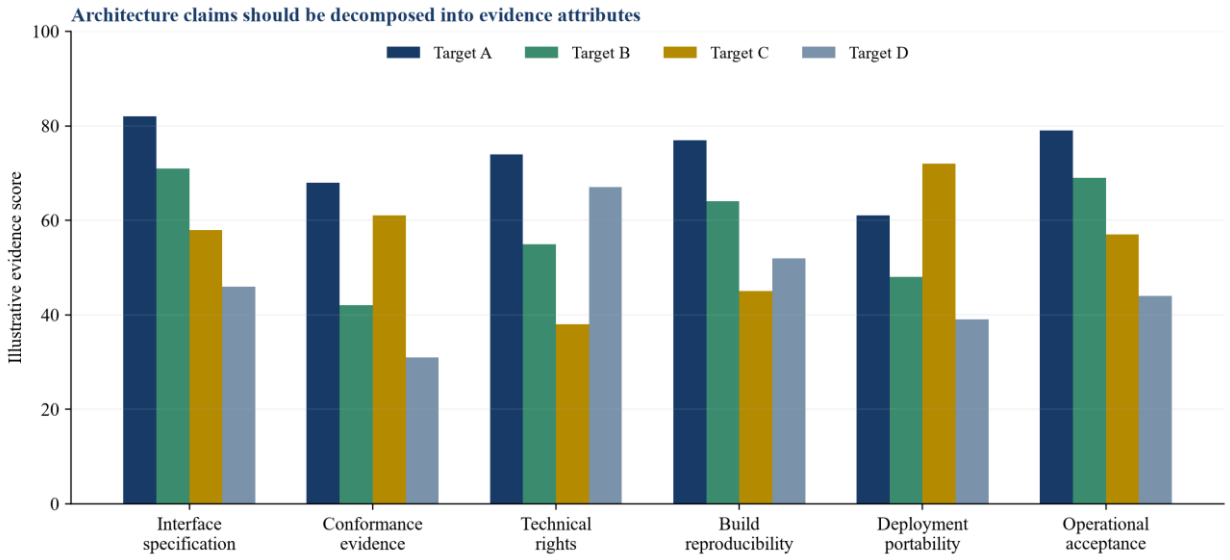

Figure 1. Hypothetical architecture openness profile across four acquisition targets

Scores are illustrative management assumptions, not observations about any company or programme.

5. SELECT THE INTERFACES THAT CONTROL VALUE

Material interfaces sit where change, competition, interoperability or mission performance matter. They can include sensor inputs, track messages, identity services, data buses, command interfaces, model-serving endpoints, map services, audit records and deployment controls. The buyer should distinguish external interfaces from internal implementation seams.

For each interface, diligence should capture syntax, semantics, timing, error handling, security, versioning, test vectors and governance. A document that lists fields without explaining units, coordinate systems, confidence or update rules is incomplete. A versioned specification should connect to a tested implementation and change process.

Interface value also depends on who controls evolution. A consensus standard may move slowly. A government-owned profile may require programme approval. A supplier-controlled API can evolve quickly while increasing dependency. The acquisition case should state how the combined company will participate in governance and fund compatibility.

6. TEST SUBSTITUTABILITY RATHER THAN CONFORMANCE ALONE

Conformance shows that an implementation meets specified requirements. Substitutability asks whether another implementation can replace it within acceptable cost, time and performance. The second test is closer to the buyer's commercial objective.

A substitute may pass message-level tests while failing latency, throughput, security, safety, resource or user-workflow requirements. It may also require customer re-accreditation. Diligence should

perform a controlled replacement or integration trial at a material interface and record engineering effort, defects, retest scope and operational impact.

The buyer should maintain a substitution ledger with baseline component, candidate, interface, test environment, result, unresolved variance and switching cost. Claims about vendor independence should be reduced when no alternate implementation has passed representative testing.

7. EXAMINE TECHNICAL DATA AND SOFTWARE RIGHTS

Architecture cannot provide practical freedom when the buyer lacks the rights or deliverables needed to exercise it. Technical rights can differ across source code, object code, interfaces, documentation, test artefacts, data, models, configuration, training material and deployment scripts. Funding source, contract language, markings and jurisdiction can affect the outcome.

GAO reported in 2025 that selected U.S. defence programmes experienced vendor reliance because of data-rights shortfalls, with potential cost and repair-time effects. Defense acquisition guidance states that lifecycle needs should consider executable code, source code, scripts, build procedures, automation, tools, databases, libraries, test results, datasets, firmware and training material needed to integrate, test, deploy and operate software.

The buyer should trace ownership and licence claims to executed instruments. It should verify assignments from employees and contractors, open-source obligations, third-party terms, government rights and customer restrictions. A transaction schedule should identify which artefacts will transfer, which rights require consent and which capabilities depend on restricted information.

8. REPRODUCE THE BUILD AND DEPLOYMENT CHAIN

A reproducible build converts controlled source and dependencies into the deployed software through documented, repeatable steps. It is a direct test of whether the acquirer can operate the asset without informal knowledge or unavailable services.

Diligence should rebuild a representative release from a clean environment. The test should verify repository access, dependency resolution, compiler and tool versions, secrets handling, signing, test execution, packaging, infrastructure definition, deployment and rollback. Failures should be logged as remediation work rather than repaired invisibly during the demonstration.

The deployment chain should cover disconnected, classified, sovereign, edge and cloud environments where applicable. A product that runs efficiently in one commercial cloud may require major redesign for constrained or air-gapped infrastructure. Portability claims should follow successful deployment evidence.

9. MEASURE SOFTWARE DELIVERY AND ADAPTATION CADENCE

Mission needs and cyber threats can change faster than traditional platform cycles. A buyer should test whether the target can discover a requirement, change software, validate it, obtain approval and deploy it safely. Commit frequency alone does not establish this capability.

The evidence record should include lead time, test automation, release frequency, failure rate, recovery time, vulnerability remediation, accreditation constraints and customer acceptance. These

measures should be segmented by product and environment. A laboratory service may release daily while an accepted mission configuration changes quarterly.

The combined company needs a release architecture that supports shared components without forcing every programme onto one schedule. Versioned interfaces and backward compatibility can permit local delivery. Uncontrolled forks can create the opposite result: duplicate engineering, inconsistent security and rising support cost.

10. QUANTIFY CUSTOMER-SPECIFIC FORKS

Customer adaptation can be commercially necessary. It becomes architectural debt when changes cannot rejoin the main product, require separate testing or depend on unique infrastructure. The diligence team should inventory forks by customer, code difference, release age, revenue, gross margin and support burden.

The buyer should determine whether a fork reflects a reusable capability, configuration, contractual restriction or historical expedient. Configuration and extension points can absorb some variation. Other differences may require a maintained branch because of classification, sovereignty or mission assurance.

The integration plan should avoid a forced consolidation that breaks accepted capability. Each fork needs a disposition: preserve, parameterise, modularise, migrate or retire. The cost and customer approval for that disposition belong in the transaction model.

11. VALIDATE STANDARDS AND PROFILE COMPLIANCE

Standards can improve interoperability when a programme selects a profile, verifies implementation and manages change. Compliance with a broad family of standards does not establish compatibility with the exact version, optional fields and security profile used by a customer.

NATO's interoperability standards and Federated Mission Networking profiles place standards within defined mission-network contexts. NATO exercises test systems, interfaces, data and operating procedures together. The U.S. Air Force describes Open Mission Systems as an industry-consensus, non-proprietary architectural standard intended to support technical refresh, reuse, interoperability and lifecycle competition.

The target should provide conformance statements, profiles, test reports, waivers and defect records. The buyer should reproduce material exchanges against the customer's implementation. Standards claims should be tied to dates and versions because specifications and profiles evolve.

12. TEST INTEROPERABILITY IN REPRESENTATIVE CONDITIONS

Interoperability means that systems exchange information and use it correctly within the mission. It includes technical, semantic, procedural and security dimensions. Passing a laboratory message test establishes only part of the requirement.

Representative testing should cover latency, load, degraded networks, identity, access control, data marking, time synchronisation, error recovery and operator workflow. It should include old and new

versions where customers upgrade at different times. The test record should preserve the exact configurations.

NATO's current interoperability activity emphasises testing data standards, interfaces, software, command-and-control systems and procedures. This breadth is relevant to M&A. A roll-up that integrates code while leaving identity, policy and operational processes unresolved may create a larger portfolio without a usable combined mission system.

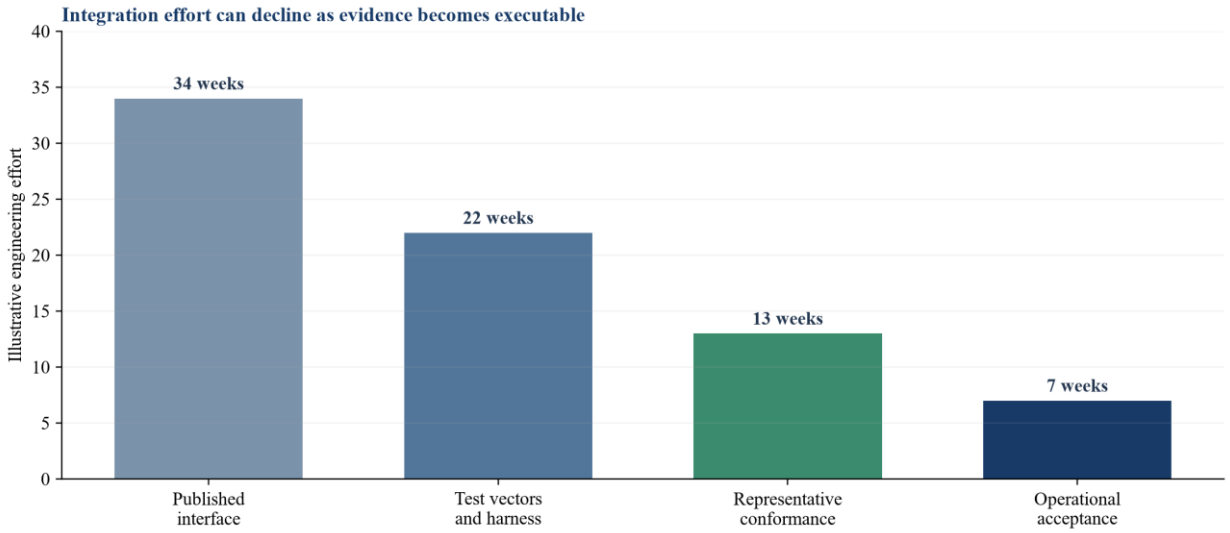

Figure 2. Hypothetical integration effort by evidence maturity

Engineering weeks are illustrative management assumptions and exclude customer approval time.

13. ESTABLISH DATA ARCHITECTURE AND SEMANTIC CONTROL

Mission integration often fails at the data layer. Two systems can exchange a field while assigning different meaning, confidence, time basis or ownership. The buyer should map authoritative sources, transformations, schemas, metadata, quality controls, retention and dissemination rules.

Data lineage should connect source event to user-facing decision. It should identify derived and model-generated fields. The architecture should preserve markings and access conditions through transformation. Cross-domain transfer and multinational sharing can impose additional design and approval requirements.

Commercial value depends on rights as well as technical access. Customer and government data may support service delivery without becoming an acquirer-owned asset. Training, benchmarking and product improvement require separate analysis. The valuation model should exclude data uses that contracts or law do not permit.

14. EVALUATE ARTIFICIAL INTELLIGENCE COMPONENTS

Mission software may use machine learning for detection, classification, prediction, planning, prioritisation or interface automation. Each model should be connected to a decision, user and failure consequence. Aggregate model accuracy cannot substitute for operational performance by relevant class and condition.

Diligence should inspect data provenance, labels, train-test separation, synthetic data, calibration, drift, adversarial exposure, human review, update authority and rollback. The buyer should determine whether models can be moved between customer environments and whether training data or weights are restricted.

An AI component can increase integration cost when inputs, computing requirements or decision thresholds differ across targets. It can create value when common data contracts and evaluation harnesses permit reuse. The roll-up case should value demonstrated transfer and keep speculative reuse outside the signing case.

15. TEST CYBERSECURITY AND SOFTWARE SUPPLY-CHAIN CONTROLS

Open interfaces can expand participation and attack surface. Proprietary integration can also conceal vulnerable dependencies. The relevant question is whether the combined business can identify, test, authorise and monitor software across its supply chain.

The buyer should inspect secure development practices, software bills of materials, dependency governance, code review, test coverage, signing, provenance, vulnerability management, secrets, access controls and incident response. NIST's Secure Software Development Framework provides a general reference for integrating secure practices into development lifecycles.

Security evidence should follow each deployed configuration. A platform-level accreditation may not transfer to an acquired module or new hosting environment. Integration should sequence identities, repositories, build services and deployment privileges so that the acquisition does not weaken controlled delivery.

16. MAP INFRASTRUCTURE AND HOSTING DEPENDENCIES

Mission applications can operate on tactical hardware, embedded computers, private data centres, sovereign cloud, commercial cloud or disconnected environments. Infrastructure affects performance, cost, accreditation, export and customer control.

The target should provide resource profiles, infrastructure definitions, supported platforms, licences, data flows, backup, recovery and capacity evidence. Cloud spend should be attributed to products and customers. Hardware acceleration or proprietary services should be identified as portability constraints.

The acquirer should test the target's claimed deployment model in the intended environment. Hosting consolidation can create scale benefits, while re-accreditation and data-residency requirements can delay migration. Synergy should follow an approved and costed path.

17. PROTECT SAFETY AND MISSION ASSURANCE

Software changes can affect safety, command authority and operational outcomes. The buyer should identify safety cases, assurance levels, independent verification, hazard records, test evidence and change-control obligations. These artefacts can define the pace and scope of integration.

An open interface does not remove assurance responsibility. Replacing a component can require regression testing across the mission thread. The architecture should isolate change where possible and provide evidence that faults remain contained. Claims about modularity should be reduced when every change triggers system-wide retest.

The transaction agreement should preserve controlled configurations and evidence. Day-one integration should avoid merging repositories, pipelines or runtime services before authority and regression requirements are understood.

18. DETERMINE PROGRAMME ACCESS

Programme access is the practical ability to compete for, enter, integrate with and remain within a funded customer programme. It depends on mission need, sponsor, procurement route, security eligibility, standards profile, incumbency, test evidence, rights, funding and customer acceptance.

The target should distinguish executed contracts, funded options, framework eligibility, trials, memoranda and pipeline. A standards-compliant product does not automatically gain programme access. It may still need a sponsor, integration partner, test event and contract vehicle.

The roll-up case should identify which acquired capabilities improve an existing position and which depend on a new programme. Cross-sell value should be adjusted for customer permission, procurement timing, integration capacity and competitive response.

19. ANALYSE PRIME CONTRACTOR AND ECOSYSTEM POSITION

Mission-software companies can sell directly, through prime contractors, as subcontractors or through government integration organisations. Each route affects margin, customer access, data rights and control of the interface.

Diligence should map partner agreements, exclusivity, flow-down terms, customer ownership, bid rights, workshare and conflict restrictions. A target's access may depend on a founder relationship or a specific prime. The buyer should test whether that access transfers after change of control.

Open architecture can increase the number of potential partners while exposing the product to substitute suppliers. The combined company needs a clear position: platform controller, module provider, integrator or specialist service. Economics and responsibility differ across these roles.

20. REVIEW EXPORT CONTROLS AND FOREIGN INVESTMENT

Mission software, source code, technical data, encryption, models, services and demonstrations can be subject to export controls. U.S. ITAR and EAR rules and UK strategic export controls require transaction-specific classification. National-security foreign-investment review can also affect ownership and information access.

The target should maintain classifications, licences, provisos, nationality controls, technology-control plans and retransfer records. The acquirer should test whether planned integration would expose controlled technology to new people, systems or locations. Remote repositories and support access belong in this review.

Revenue and synergy should be segmented into currently executable, approval-dependent and restricted categories. The transaction timetable should allow for required filings and consents. Current specialist advice is necessary for each deal.

21. RECONCILE PRODUCT REVENUE WITH ENGINEERING SERVICES

Mission-software revenue can combine licences, subscriptions, implementation, integration, engineering, hosting, support and hardware. A product label does not establish scalable economics. The buyer should allocate revenue and direct cost by obligation and customer.

Recurring revenue should be supported by enforceable service, licence or support commitments and customer acceptance. Engineering required to maintain customer forks should be treated as service cost. Capitalised development and contract assets should be reconciled with delivery evidence.

IFRS 15 provides the accounting framework for revenue from customer contracts, while transaction diligence should also examine billing, milestones, acceptance, service credits, working capital and collections. Sustainable margin follows repeatable delivery rather than presentation categories.

22. TEST BACKLOG AND OPTION QUALITY

Backlog should be traced to executed instruments, funding, scope, configuration, acceptance, schedule, cancellation, security, export and cash milestones. Framework ceilings, unfunded options and public announcements should remain separate from contracted backlog.

The buyer should sample the largest, oldest, lowest-margin and most conditional awards. It should identify customer dependencies such as access to test ranges, networks, data and government-furnished equipment. Delays can consume engineering capacity before revenue recognition.

Architecture affects backlog quality when a contract assumes reuse or interface access that has not been established. Integration estimates should be compared with prior actuals. Backlog receives value after technical and commercial executability are reconciled.

23. BUILD THE INTEGRATION COST LEDGER

The integration ledger should include code and repository migration, identity, interfaces, data models, test harnesses, cybersecurity, accreditation, hosting, licences, customer changes, programme management and support. It should separate one-time work from continuing duplicated cost.

Common-platform savings can be offset by the cost of maintaining multiple accepted configurations. The buyer should model a base case that preserves customer service and a consolidation case that requires approvals. The difference shows which synergy depends on external decisions.

Engineering estimates should include contingency for undocumented dependencies and failed tests. Milestones should be tied to reproduced builds, validated interfaces and customer-approved releases rather than completion percentages reported by the integration team.

Table 2. Hypothetical integration cost ledger for a four-company mission-software roll-up

Workstream	Initial estimate	Evidence-adjusted estimate	Primary driver
Repositories, build and release	1.2	2.4	Non-reproducible builds and unique toolchains
Interfaces and data	1.8	4.1	Semantic differences and proprietary gateways
Security and accreditation	1.5	3.6	Separate environments and customer approvals
Hosting and deployment	1.0	2.2	Edge, sovereign and disconnected variants
Customer migration and support	1.4	3.3	Accepted forks and contractual constraints
Total	6.9	15.6	Evidence-based integration scope

Values are illustrative management assumptions in USD millions.

24. BUILD THE CUSTOMER-VALUE LEDGER

Customer value can arise from faster capability insertion, lower integration cost, improved interoperability, supplier competition, reduced downtime and more reliable software delivery. Each component requires a baseline, time period and attribution method.

The hypothetical management case proposes USD 17.8 million of annual customer value across the combined portfolio. Evidence review retains USD 6.9 million after adjustments for customer-specific architectures, approval times, incomplete rights, duplicated benefits and uncertain adoption. These numbers are illustrative management assumptions.

The ledger should avoid counting the same integration saving as both customer value and buyer cost synergy. It should also separate a customer's avoided cost from revenue available to the supplier. A benefit supports valuation when contracts, renewals or credible willingness-to-pay evidence connect it to cash.

Table 3. Hypothetical annual customer-value ledger

Proposed value component	Management case	Evidence retained	Principal adjustment
Faster capability insertion	USD 5.6m	USD 2.2m	Approval and test cycle
Lower integration and switching cost	USD 4.8m	USD 1.9m	Interface and rights maturity
Improved interoperability	USD 3.7m	USD 1.4m	Representative acceptance evidence
Delivery and support reliability	USD 3.7m	USD 1.4m	Fork and infrastructure burden
Total annual customer value	USD 17.8m	USD 6.9m	Governed evidence gates

Every number is an illustrative management assumption, not observed customer or programme data.

25. BUILD THE TRANSACTION VALUATION BRIDGE

The hypothetical four-company group has USD 62 million of combined annual revenue, USD 178 million of management-reported backlog and USD 21.5 million of proposed annual buyer revenue and synergy. These figures are illustrative management assumptions only.

Evidence review retains USD 93 million of funded and executable backlog and USD 8.1 million of annual buyer revenue and synergy. Adjustments reflect interface maturity, technical rights, customer forks, accreditation, programme access, support cost, key-person dependency and cash conversion.

The bridge does not produce a valuation opinion. It produces evidence-adjusted inputs for the buyer's selected cash-flow, market or transaction method. IFRS 3, IFRS 13 and IAS 38 provide relevant accounting context for business combinations, fair value and identifiable intangible assets; transaction-specific judgement remains necessary.

Table 4. Hypothetical annual buyer revenue and synergy bridge

Component	Management case	Evidence-adjusted case	Evidence gate
Revenue retained	8.4	4.0	Contract, acceptance, support and cash
Cross-sell and programme access	5.1	1.4	Sponsor, procurement route and approval
Engineering and hosting savings	4.6	1.8	Reproducible build and migration plan
Data and product reuse	3.4	0.9	Rights, interfaces and customer permission
Total	21.5	8.1	No duplication with customer value

Values are illustrative management assumptions in USD millions.

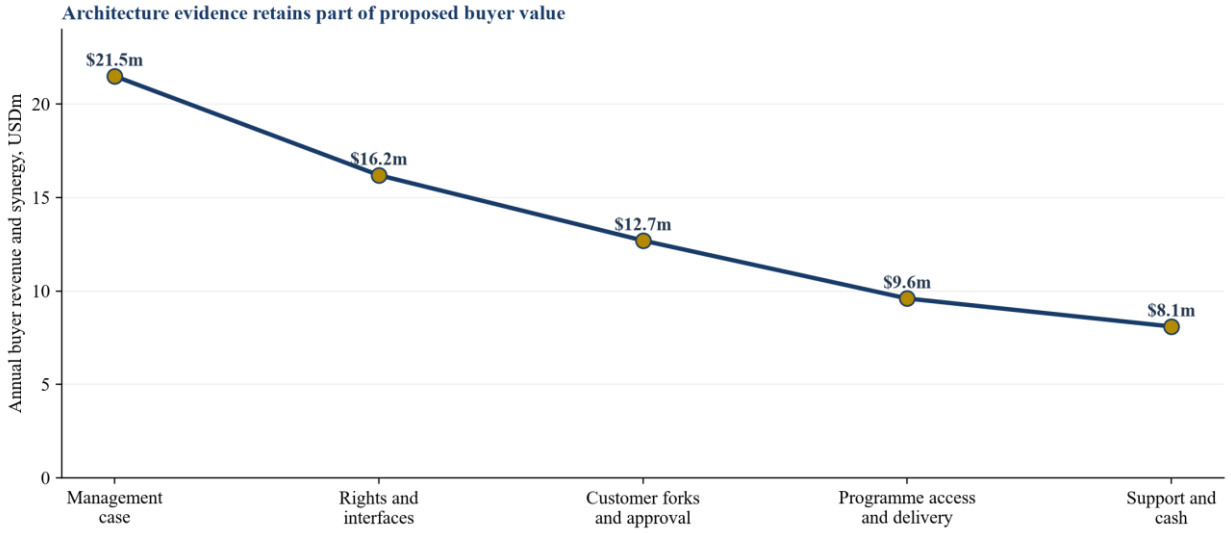

Figure 3. Hypothetical buyer revenue and synergy validation bridge

Values in USD millions are illustrative management assumptions.

26. TRANSLATE EVIDENCE INTO PRICE AND TRANSACTION TERMS

Verified contracted revenue, accepted configurations, reproducible builds, controlled rights and successful substitution tests can support value at signing. Unproven programme access, future standards compliance and customer migration should support contingent value only when the milestone is observable.

Representations should cover ownership, licences, government rights, source and artefact delivery, open-source compliance, cybersecurity, export controls, customer contracts, architecture claims, test evidence and incidents. Specific indemnities or holdbacks may address defined exposures. Conditions precedent can cover consents, filings and controlled-access arrangements.

Earn-outs should define revenue, hardware pass-through, engineering services, options, delayed acceptance, integration cost and buyer-led cross-sell. A milestone tied to a customer-approved integrated release is stronger than one tied to an internal product launch. Collected cash remains the most reliable commercial endpoint.

27. EXECUTE THE DILIGENCE WORKPLAN

The request list should include architecture records, interface specifications, data models, repositories, branches, dependency manifests, software bills of materials, build and deployment procedures, test harnesses, accreditation artefacts, incident records, rights schedules, export files, customer contracts, backlog and financial models.

Sampling should cover the largest programmes, newest and oldest releases, customer forks, failed builds, interface defects, security exceptions, unsupported dependencies and material third parties. The diligence team should reproduce one representative build, one deployment and one substitution test for each major product family where access permits.

Technical findings should reconcile with contracts, invoices and cash. A product described as deployed may remain in trial. A standard-compliant module may depend on a proprietary adaptor. A recurring licence may require substantial engineering to remain operable. These distinctions affect valuation and integration design.

28. GOVERN THE FIRST ONE HUNDRED DAYS

The first thirty days should freeze controlled inventories, secure repositories, confirm identities and reconcile products, contracts, versions, environments and rights. Days thirty-one to sixty should reproduce material builds, establish common evidence formats and close priority access and security gaps. Days sixty-one to one hundred should validate the first shared interface, align release governance and confirm customer-specific integration plans.

The buyer should preserve accepted configurations while building common engineering services around them. Shared source control, testing, signing and vulnerability management can be introduced through governed migration. Product consolidation should follow evidence and customer approval.

Board reporting should include reproducible-build coverage, interface conformance, customer forks, release lead time, failed changes, critical vulnerabilities, accreditation status, rights gaps, funded backlog, programme milestones, integration spend, support margin and collected cash.

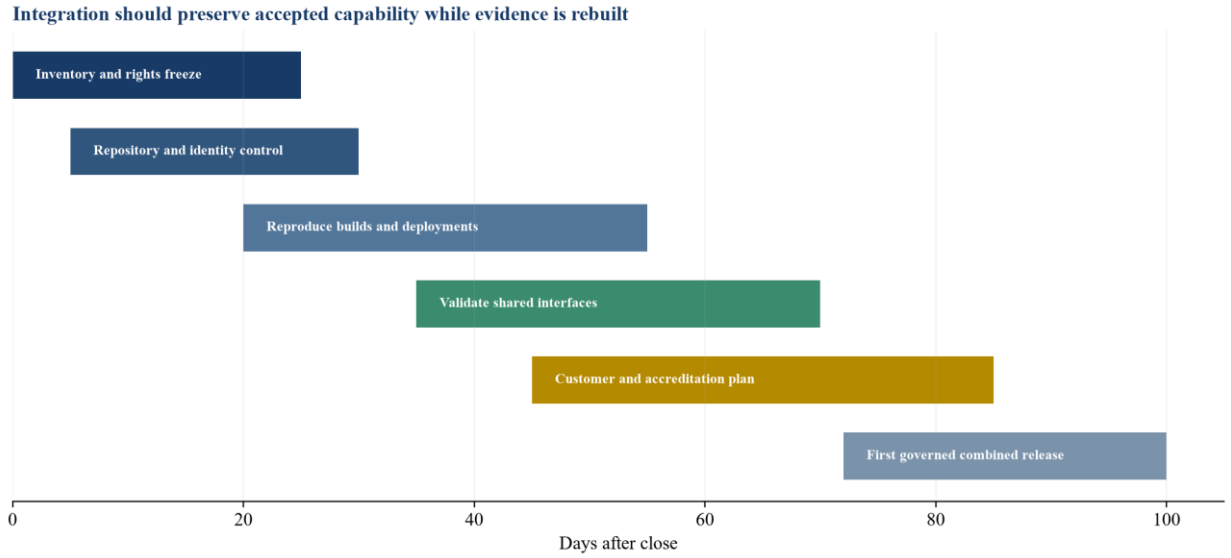

Figure 4. Hypothetical one-hundred-day architecture integration sequence

Timing is illustrative and should be adapted to customer, security and programme requirements.

29. ESTABLISH ARCHITECTURE GOVERNANCE AFTER CLOSE

The combined company needs one accountable architecture forum with authority over material interfaces, data contracts, shared services and deprecation. The forum should include product, engineering, security, delivery, commercial and customer-account owners. Its decisions should remain traceable to mission needs and contractual obligations.

Governance should classify interfaces by their commercial and operational importance. A strategic external interface requires a named owner, compatibility policy, test harness, security review, version roadmap and customer communication process. An internal implementation detail can change more freely. This distinction prevents bureaucracy from spreading across every technical decision while protecting the parts of the architecture that enable integration and competition.

The board should approve an architecture investment plan that connects expenditure to observable outcomes. Examples include a reproducible-build target, reduction in unsupported branches, completion of a substitution test, adoption of a common identity service or customer acceptance of a shared data profile. Each initiative should state the affected programmes, expected engineering effort, approval dependency and financial consequence.

Technical debt should be recorded as a costed obligation rather than a general score. A debt item should identify the component, failure mode, affected customers, remediation, owner and deadline. Items that can interrupt a mission, prevent accreditation or block a contract should receive priority over aesthetic code cleanup. This register also improves acquisition sequencing because the buyer can decide which weaknesses to remedy before combining products.

Architecture governance should preserve credible product independence where customers or missions require it. A common engineering platform can support distinct applications without forcing a single user interface or operating concept. The evidence goal is controlled reuse: shared components should have known consumers, tests, security properties and support owners.

30. RECOGNISE LIMITATIONS AND CONCLUDE

This framework does not validate a product, architecture, standard, programme, security posture, export classification, licence or transaction. Requirements vary by mission, customer, jurisdiction and configuration. Current legal, technical, security, export, accounting and procurement advice is required.

The hypothetical case does not estimate market demand or transaction value. Public policies and interoperability exercises establish context. They do not prove a private company's capability or commercial access. Protected evidence should remain within authorised review channels.

Architecture advantage is an evidenced option set. A buyer gains value when it can replace a component, integrate an acquired product, reproduce a build, deploy to the required environment, pass representative tests, obtain customer acceptance and exercise the required rights. Each option should be connected to a cost, timeframe and programme.

The strongest mission-software roll-up case joins technical architecture with commercial execution. Interfaces need verified semantics. Rights need usable artefacts. Standards need operational testing. Delivery needs secure and reproducible pipelines. Programme access needs funding, sponsorship and acceptance. Valuation should follow the part of that chain that evidence supports.

REFERENCES

[1] U.S. Department of Defense, Office of the Under Secretary of Defense for Research and Engineering. Modular Open Systems Approach. https://www.cto.mil/sea/mosa/

[2] U.S. Department of Defense. Implementing a Modular Open Systems Approach in Department of Defense Programs, MOSA Implementation Guidebook. https://www.cto.mil/wp-content/uploads/2023/06/MOSA-Implementation-Guidebook-2023.pdf

[3] U.S. Government Accountability Office. Weapon Systems Acquisition: DOD Needs Better Planning to Attain Benefits of Modular Open Systems, GAO-25-106931, January 2025. https://www.gao.gov/products/gao-25-106931

[4] U.S. Government Accountability Office. Weapon System Sustainment: DOD Can Improve Planning and Management of Data Rights, GAO-25-107468, September 2025. https://www.gao.gov/products/gao-25-107468

[5] U.S. Government Accountability Office. Weapon Systems Annual Assessment: DOD Leaders Should Ensure That Newer Programs Are Structured for Speed and Innovation, GAO-25-107569, June 2025. https://www.gao.gov/products/gao-25-107569

[6] U.S. Government Accountability Office. Defense Software Acquisitions: Changes to Requirements, Oversight, and Tools Needed for Weapon Programs, GAO-23-105867, June 2023. https://www.gao.gov/products/gao-23-105867

[7] U.S. Government Accountability Office. Weapon System Sustainment: DOD Needs to Better Capture and Report Software Sustainment Costs, GAO-19-173, February 2019. https://www.gao.gov/products/gao-19-173

[8] U.S. Government Accountability Office. Defense Acquisitions: Review of Private Industry and Department of Defense Open Systems Experiences, GAO-14-617R, June 2014. https://www.gao.gov/products/gao-14-617r

[9] U.S. Government Accountability Office. Defense Acquisitions: DOD Efforts to Adopt Open Systems for Its Unmanned Aircraft Systems Have Progressed Slowly, GAO-13-651, July 2013. https://www.gao.gov/products/gao-13-651

[10] Acquisition.gov. PGI 227.72, Computer Software, Computer Software Documentation, and Associated Rights. https://www.acquisition.gov/dfarspgi/pgi-227.72-computer-software-computer-software-documentation-and-associated-rights

[11] U.S. Department of Defense. DoD Instruction 5000.87, Operation of the Software Acquisition Pathway. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/500087p.pdf

[12] U.S. Department of Defense. DoD Instruction 5010.44, Intellectual Property Acquisition and Licensing. https://www.esd.whs.mil/Portals/54/Documents/DD/issuances/dodi/501044p.pdf

[13] U.S. Air Force Research Laboratory. Open Mission Systems. https://www.vdl.afrl.af.mil/programs/oam/oms.php

[14] NATO Allied Command Transformation. Federated Mission Networking. https://www.act.nato.int/activities/federated-mission-networking/

[15] NATO Allied Command Transformation. Federated Interoperability. https://www.act.nato.int/activities/federated-interoperability/

[16] NATO Allied Command Transformation. CWIX 26 Strengthens NATO's Digital Interoperability and Operational Readiness, June 2026. https://www.act.nato.int/article/cwix-2026-concludes/

[17] NATO. NATO's Digital Transformation Implementation Strategy, October 2024. https://www.nato.int/cps/en/natohq/official_texts_229801.htm

[18] NATO. NATO Interoperability Standards and Profiles, Edition N Version 2. https://nhqc3s.hq.nato.int/Apps/Architecture/NISP/pdf/NISP-Vol2-v15-release.pdf

[19] National Institute of Standards and Technology. Secure Software Development Framework, SP 800-218. https://csrc.nist.gov/pubs/sp/800/218/final

[20] U.S. Department of State, Directorate of Defense Trade Controls. International Traffic in Arms Regulations. https://www.pmddtc.state.gov/ddtc_public/ddtc_public?id=ddtc_kb_article_page&sys_id=24d528fddbfc930044f9ff621f961987

[21] U.S. Department of Commerce, Bureau of Industry and Security. Export Administration Regulations. https://www.bis.gov/ear

[22] UK Government. UK Strategic Export Controls. https://www.gov.uk/guidance/uk-strategic-export-controls

[23] IFRS Foundation. IFRS 3 Business Combinations. https://www.ifrs.org/issued-standards/list-of-standards/ifrs-3-business-combinations/

[24] IFRS Foundation. IFRS 13 Fair Value Measurement. https://www.ifrs.org/issued-standards/list-of-standards/ifrs-13-fair-value-measurement/

[25] IFRS Foundation. IFRS 15 Revenue from Contracts with Customers. https://www.ifrs.org/issued-standards/list-of-standards/ifrs-15-revenue-from-contracts-with-customers/

[26] IFRS Foundation. IAS 38 Intangible Assets. https://www.ifrs.org/issued-standards/list-of-standards/ias-38-intangible-assets/

ABOUT THE AUTHOR

Chennakeshav (CK) is a corporate finance and investment banking executive with 25+ years of global experience in deal origination, structuring and execution across M&A, growth capital and corporate strategy. He has led value-creation mandates for founders, corporates and funds — bridging the boardroom view to hands-on execution and close.

His career spans Morgan Stanley, HSBC, Lloyds Banking Group, EWEC, ADQ portfolio companies and Emirates Growth Fund, across TMT, real estate, fintech, deeptech, cleantech, infrastructure and energy. He has partnered with C-suite leaders, private equity and venture funds, sovereign wealth funds and family offices to finance complex fund raises and scale-up ventures, and has led M&A due diligence, post-merger integration and business-transformation initiatives to create value.

At Matchpoint Partners he is Managing Partner, leading the firm's corporate finance, M&A and capital-raising practice. He holds an MBA from London Business School, an engineering degree from VTU and a Master of Laws (LLM, in progress) from UCL London.

An active start-up mentor, CK mentors at Techstars, DIFC FinTech Hive, Startup Grind, Founder Institute and IN5, serves as Entrepreneur Mentor in Residence (EMiR) at London Business School, and judges the Entrepreneurship World Cup.

https://www.linkedin.com/in/ckadya/

https://www.matchpoint-partners.com/team/ck-adya.html

This paper is part of a continuing series on the structure of private and alternative markets. The views expressed are the author's own. The paper is for information only, describes market structure in general terms, and does not constitute investment, legal, tax or regulatory advice or a recommendation in respect of any security, vehicle or counterparty.