Quantum Software M&A: Valuing Algorithms without Hardware Independence

An Evidence Framework for Portability, Benchmark Quality, Compiler Dependence, Customer Retention and Integration Value

Authored by Chennakeshav Adya

Independent Researcher

September 2026

Abstract

Quantum software companies are often described as hardware agnostic. The economic meaning of that claim is narrower than the marketing phrase. A source-level algorithm may be portable while its performance depends on a compiler, intermediate representation, device topology, native gate set, noise model, calibration state, control features, cloud interface and classical workflow. An acquirer therefore needs to determine which capabilities survive a change of hardware, which require costly retargeting and which customer outcomes depend on a vendor roadmap outside the target's control.

This paper develops an M&A framework for valuing quantum algorithms, compilers, orchestration tools and application software. It separates source-code portability, semantic portability, compilability, executable portability, performance portability and commercial portability. It connects each layer to an algorithm evidence ladder, a dependency graph, a customer cohort analysis, a replacement-cost model and a probability-weighted valuation scorecard.

The analysis draws on open standards and primary documentation. Quantum Intermediate Representation provides a language- and hardware-agnostic interface while leaving target capabilities to the execution environment. OpenQASM 3 defines a broad language, yet implementations may support different runtime subsets. Amazon Braket documents device-specific OpenQASM support. Azure Quantum documents target profiles that differ in support for mid-circuit measurement, classical arithmetic and loops. QED-C application-oriented benchmarks measure result quality, execution time and resource consumption. These sources show why syntactic conversion alone does not establish equivalent output, cost or customer value.

Disclosed transactions provide additional evidence. Honeywell Quantum Solutions and Cambridge Quantum combined in 2021 to create Quantinuum, joining hardware and software capabilities. Keysight acquired Quantum Benchmark to add error diagnostics, suppression and validation software. Quantum Machines acquired QDevil to extend quantum-control capability. SandboxAQ acquired Good Chemistry to add computational-chemistry technology, customers and talent. These examples illustrate several acquisition theses; they do not provide directly comparable standalone algorithm values.

A wholly hypothetical target illustrates the method. It reports USD 18 million of annual revenue, USD 11 million of recurring revenue, USD 4 million of services revenue, USD 3 million of grants and other income, USD 96 million of cash and USD 38 million of annual cash use. The central enterprise value is USD 420 million, allocated among validated algorithms, compiler and orchestration assets, customer relationships, proprietary data and workflows, team and know-how, and probability-weighted roadmap options. Every amount, benchmark, probability and scenario in the worked example is a management assumption created solely to explain the framework.

The paper concludes that algorithm value should follow demonstrated transferability and accepted customer outcomes. An acquirer should reproduce builds, execute held-out workloads across defined

targets, measure solution quality and time to solution, reconcile customer cash, map third-party rights and price the cost of retargeting. Consideration can then separate achieved capability from conditional hardware access, future performance and customer conversion.

JEL Classification: G12, G24, G34, L86, O32, O33

Keywords: quantum software, mergers and acquisitions, algorithm valuation, hardware portability, compiler dependence, quantum benchmarks, intellectual property, technology diligence

Introduction

Quantum software spans programming languages, compilers, circuit optimisation, error mitigation, control systems, simulators, workflow orchestration, application libraries, resource estimation and domain solutions. Some products sit close to a quantum processor. Others coordinate quantum and classical computation, manage experiments, or package scientific methods for chemistry, finance, optimisation and machine learning. The transaction perimeter can therefore contain code, data, patents, trade secrets, employees, customer contracts, cloud relationships and future access to hardware.

The buyer's central question is whether the target owns a durable capability or a temporary implementation tuned to a particular device. A circuit that executes on two platforms may produce different accuracy, depth, cost and latency. A compiler can reduce gates on one topology while increasing routing overhead on another. An application may rely on a provider's proprietary pulse access, error-mitigation service or queue priority. A customer contract may fund research rather than a repeatable product.

Valuation becomes difficult when portability is treated as a yes-or-no attribute. Software can be portable at source level and dependent at performance level. It can compile to a common representation and still require target-specific lowering. It can reproduce a mathematical result while losing the speed, cost or accuracy that supported the commercial claim. It can run on alternative hardware while the customer relationship remains tied to a preferred provider.

This paper replaces the broad claim of hardware independence with an evidence hierarchy. It asks what moves, what must be rebuilt, what performs, who pays and which rights transfer. The output is a transaction framework for corporate-development teams, investors, founders and advisers assessing quantum software acquisitions, combinations and strategic investments.

1 Define the acquisition thesis

The transaction committee should state why ownership is required. A buyer may seek an algorithm portfolio, a compiler or control layer, an application team, customer access, proprietary data, patents, a developer community, or a route to integrate hardware and software. Each thesis requires different diligence and produces a different valuation bridge.

A hardware buyer may value software because it improves utilisation, exposes device capabilities and attracts workloads. A cloud or platform buyer may value abstraction, orchestration and distribution. An industrial buyer may value a domain workflow and the scientists who understand it. A financial investor may value optionality across several hardware providers. The committee should identify the revenue, cost, time or strategic dependency that the acquisition changes.

The thesis should also state the counterfactual. Alternatives can include licensing, partnership, internal build, open-source adoption, acqui-hiring or waiting. Replacement time and execution risk often matter more than historical development spend. An acquisition premium is difficult to justify when a credible alternative reaches the same customer outcome before quantum hardware becomes commercially relevant.

The board paper should specify the valuation date, security, consideration, assumed funding and integration budget. It should identify which value is present at closing and which depends on technical, customer or hardware milestones. That separation allows price and protection to follow evidence.

2 Decompose hardware independence

Source portability means that code or a high-level algorithm can be moved or translated. Semantic portability means that the intended mathematical operation remains equivalent. Compilation portability means that the programme can be lowered into an accepted intermediate representation and target instruction set. Executable portability means that the complete workflow runs on a target under supported conditions.

Performance portability adds solution quality, resource use, latency, throughput and cost. Commercial portability asks whether the customer accepts the output, the support model remains viable and the economics survive. These layers should be tested separately. Passing an earlier layer does not establish the next.

An intermediate representation can reduce language and compiler coupling. Microsoft describes Quantum Intermediate Representation as language and hardware agnostic, using LLVM infrastructure as a common interface. The target environment still determines available gates, control flow, measurement, timing and runtime services. Azure Quantum target profiles document different capabilities for conditional branching, arithmetic and loops. The buyer should test the profile required by each valuable workflow.

OpenQASM 3 similarly provides a broad language for quantum programmes and real-time classical control. Its specification permits implementations to restrict runtime processing to operations that hardware can perform efficiently. Amazon Braket documents supported statements, operations and device capabilities. The existence of a standard therefore improves interoperability while leaving material implementation differences.

3 Build the algorithm evidence ladder

The first level is a mathematical specification with a defined problem, inputs, outputs and correctness criterion. The second is a reproducible classical simulation. The third is compilation for a named target. The fourth is execution on hardware with complete resource and error records. The fifth is repeated performance across dates, devices and problem instances. The sixth is an accepted customer outcome.

Each level should have a frozen evidence package. It should contain source code, dependencies, environment files, test cases, data, seeds, compiler settings, target identifiers, calibration records, queue time, execution time, shots, post-processing, cost and result-quality measures. The acquirer should be able to reproduce the claimed output from a clean environment.

Algorithmic novelty does not automatically create commercial value. A method can be scientifically interesting while classical alternatives remain faster, cheaper or more accurate. A valuable application should define the decision improved, the relevant baseline and the conditions under which quantum or quantum-inspired computation changes economics.

The ladder should identify the highest evidence level achieved for each product. A portfolio can contain mature error-diagnostics software, experimental optimisation routines and unfunded research. Applying one revenue multiple to all three hides the difference. The valuation should allocate achieved value and conditional option value separately.

4 Measure benchmark quality

Benchmarks should answer a transaction question. QED-C application-oriented benchmarks evaluate result fidelity, execution time and resource consumption across algorithms and problem sizes. They are

useful because they move beyond a single hardware metric. The buyer should still check whether the benchmark represents the target's customer workloads and whether the implementation choices favour a platform.

The benchmark plan should include held-out instances, classical baselines and multiple target configurations. It should record compilation time, circuit width and depth, two-qubit operations, shots, queue time, execution time, classical post-processing, retries and total cost. Solution quality should be defined before results are known.

Hardware and compiler versions matter. A result can improve because of calibration, transpilation or error mitigation rather than the target's proprietary algorithm. The buyer should rerun a baseline implementation through the same stack and test the target's contribution through ablation. Proprietary value is supported when the improvement persists after controlling for access and configuration.

Benchmark governance should prevent selective reporting. All attempted instances, exclusions and failed runs should be retained. The diligence team should understand parameter tuning and whether customer deployments require similar specialist intervention. A result that depends on founder-led manual tuning may represent services or talent value rather than a scalable software product.

5 Map compiler and intermediate-representation dependence

Quantum compilation includes decomposition, mapping, routing, scheduling, optimisation, pulse lowering and runtime integration. An application may call a high-level library while value-moving performance sits in third-party passes or provider services. The buyer should trace each transformation from source to executed job.

The dependency graph should identify languages, libraries, intermediate representations, compilers, plugins, target backends, simulators, cloud APIs and classical services. For each component, diligence should record ownership, licence, version, maintenance, replacement effort and operational importance. Open-source availability reduces some acquisition risk while creating governance and compatibility obligations.

Compiler performance should be tested by target. A pass that reduces depth on one connectivity graph may provide limited benefit elsewhere. Dynamic circuits, mid-circuit measurement, reset, pulse access and error-mitigation features can change the available algorithm. QIR target profiles and provider documentation should be reconciled with the target's actual requirements.

The acquirer should reproduce a clean build without founder credentials, local files or undisclosed services. It should regenerate intermediate artefacts and execute an agreed benchmark. Build reproducibility supports transferability. It does not by itself establish freedom to operate, scale or customer value.

6 Test portability across hardware modalities

Hardware modalities differ in connectivity, native operations, measurement, reset, coherence, gate duration, control and queue economics. Superconducting, trapped-ion, neutral-atom, photonic and annealing systems can require different formulations or compilation choices. A claim of portability should specify the supported class of problem and target.

The test matrix should include at least one primary target, one credible alternative and one simulator or emulator. The same workload should use defined input data and acceptance criteria. Differences in solution quality, depth, runtime, retries and cost should be measured. When the algorithm changes materially by target, the buyer should treat each implementation as a separate maintained product.

Provider access can be a hidden asset. Priority queues, reserved capacity, calibration information, engineering support and non-public features may support results that ordinary customers cannot reproduce. Contracts should be reviewed for assignment, change of control, pricing, data and continuity. The valuation should separate the software from privileged access.

Portability can also weaken differentiation. If a standard representation allows competitors to move equivalent algorithms easily, the moat may sit in data, optimisation, workflow integration or customer relationships. The diligence report should state which layer remains proprietary after the programme is expressed through open interfaces.

7 Diligence intellectual property and open source

The buyer should map patents, copyright, trade secrets, data rights, licences and contributor agreements to each product. Source repositories should show authorship, commit history, third-party components and releases. Employee and contractor assignments should be complete. University or government-funded work may carry additional rights and obligations.

Open-source software can accelerate adoption and create a developer community. It can also allow competitors to use core capability. The buyer should understand which repositories are open, which components remain proprietary and whether a change in licensing is legally and commercially practical. Copyleft, attribution, notice, patent and redistribution obligations should be reviewed by qualified counsel.

Training data, molecular data, customer datasets and benchmark corpora can be material. The target should demonstrate provenance, consent, contractual use rights, retention and transfer rights. Data obtained for a specific project may not be reusable after an acquisition. Synthetic or public data may be replaceable and should not receive unsupported scarcity value.

Trade secrets require operational controls. The buyer should examine access, documentation, encryption, offboarding and knowledge concentration. A method known only to one researcher should be valued with retention and transfer risk. Patents should be mapped to the actual product rather than counted.

8 Separate product revenue from research funding

Quantum software revenue can include subscriptions, licences, professional services, government awards, research collaborations, milestone payments and cloud usage. These categories have different repeatability and gross margins. The buyer should reconcile contracts, invoices, acceptance and cash by customer and product.

Recurring revenue should require a continuing obligation and a credible renewal basis. A multi-year research agreement may provide contracted cash while funding bespoke work. A pilot may demonstrate budget and engagement without proving product-market fit. Grants can finance valuable development while carrying restrictions and limited commercial recurrence.

The customer cohort should show opening revenue, renewals, expansion, contraction, churn, services, deferred revenue and cash collection. It should identify hardware provider, workload, product version and specialist support. Concentration can be high because early customers are strategic partners. The buyer should test whether those relationships survive a change of control or hardware strategy.

Revenue forecasts should not assume that broader hardware availability automatically creates demand. The model should link each customer segment to a defined capability, adoption event and selling cost. Unsupported market forecasts should remain outside the achieved-value bridge.

9 Diligence customer outcomes

Customer references should focus on decisions and accepted outputs. The buyer should ask what problem was addressed, what classical baseline existed, which target was used, how results were validated, what resources were consumed and whether the customer changed an operational decision. A published collaboration can be strategically meaningful without establishing recurring economic value.

Acceptance criteria should be contractual where possible. Chemistry software may be judged by predictive accuracy and experimental validation. Optimisation may be judged by objective value, feasibility, runtime and repeatability. Error diagnostics may be judged by measured reduction or validation quality. The metric should match the customer's use.

Portability should be tested commercially. If the original provider is unavailable or the target is acquired by a competing hardware company, can the customer continue? The buyer should examine termination, exclusivity, data, output ownership, audit and support obligations.

The strongest evidence is repeated paid use with stable delivery economics. The weakest is an unsigned expression of interest. The valuation should use a customer evidence ladder rather than placing all logos into pipeline value.

10 Value talent and scientific know-how

Quantum software teams may combine theoretical physicists, applied mathematicians, compiler engineers, domain scientists, software engineers, product leaders and customer scientists. The buyer should map each critical capability to products and milestones. Organisational titles alone do not reveal technical dependency.

The target may rely on founders for algorithm design, customer credibility and manual tuning. Key engineers may own the compiler or deployment system. Domain scientists may translate customer problems into solvable formulations. The diligence team should identify undocumented knowledge and realistic replacement time.

Retention should reflect post-close work. Service-based awards can support continuity. Milestone awards should use reproducible outputs and accepted customer outcomes. Compensation should avoid rewarding benchmark selection or unsupported performance claims.

Integration should preserve scientific challenge and software delivery discipline. The buyer should define repository access, release governance, security, architecture authority and product ownership. A hardware acquirer should avoid forcing every application onto its own platform before portability and customer value are tested.

11 Assess cybersecurity and software supply-chain risk

Quantum software products inherit classical software risk. The buyer should review identity, privileged access, secrets, dependencies, build pipelines, package signing, vulnerability management, logging, incident response and recovery. Cloud tokens and hardware credentials require particular control.

Software bills of materials should identify packages, versions, licences and known vulnerabilities. Reproducible builds and protected release pipelines reduce the risk that the acquired product cannot be reconstructed or trusted. The buyer should test backup restoration and the ability to rotate all credentials at closing.

Customer and experimental data can be sensitive. Contracts and architecture should show where inputs, circuits, calibration data and outputs are stored. Cross-border processing, export controls and sector requirements may affect integration. Security representations should be tested against logs and incidents.

Remediation cost belongs in the valuation and integration plan. A valuable algorithm with weak software controls may require repository, cloud and identity work before it can be deployed to regulated customers. That spending should not be hidden in general synergy.

12 Model replacement cost and avoided time

Replacement cost should estimate the team, data, experimentation, software and elapsed time required to reproduce the transferable capability. Historical spend is evidence but includes failed paths and sunk cost. The buyer should value the current system, documentation and learning that shorten a credible alternative.

The model should separate code volume from difficulty. A small compiler pass can encode rare expertise. A large application repository can contain replaceable integration code. Expert interviews, repository history and benchmark reproduction help identify the bottleneck.

Avoided time can be valuable when a hardware or industrial roadmap has a fixed window. The acquirer should compare purchase and integration time with an internal build. The benefit should be reduced when key rights, customers or staff may not transfer.

Open standards and open-source frameworks can reduce replacement cost. They can also expand the market and increase the value of complementary proprietary layers. The valuation should identify whether the target's moat strengthens or weakens as interoperability improves.

13 Build the income model

An income model should use customer evidence rather than a distant quantum market forecast. Revenue should be segmented by product, services, grants and other sources. Gross margin should include cloud execution, hardware access, specialist delivery, support and third-party licences. Research payroll and product development should remain visible.

The forecast should model conversion events. A pilot converts when acceptance, procurement and budget are satisfied. A subscription renews when the product remains useful across hardware and versions. A services project becomes product revenue only when delivery is repeatable with less bespoke effort.

The model should include hardware dependencies. If a valuable workflow requires a provider capability expected in two years, revenue should be probability weighted and capitalised cautiously. If the target earns revenue today from diagnostics or simulation, that achieved cash flow can support conventional analysis.

Terminal value should reflect technology change and competition. A long growth period unsupported by customer cohorts creates false precision. The transaction committee should compare the income approach with replacement cost and scenario value.

14 Use a portability-adjusted scorecard

The scorecard should cover algorithm evidence, portability, benchmark quality, intellectual property, data, customer outcomes, revenue quality, team, security and runway. Each score should link to evidence and a valuation implication. A high scientific score cannot repair missing ownership or weak customer acceptance.

Portability should receive separate sub-scores for source, semantics, compilation, execution, performance and commercial continuity. The buyer should identify the minimum level required by the acquisition thesis. A hardware buyer may accept dependence that strengthens its platform. A neutral cloud buyer may require broad performance portability.

The scorecard should show concentration. A target can receive strong average marks while depending on one provider, scientist or customer. Gating failures should therefore be reported separately from weighted scores. The board should know which issue can invalidate the thesis.

Scores should change after confirmatory tests. The model should preserve the evidence and rationale for each update. This supports price negotiation and post-close accountability.

15 Learn from disclosed combinations

The formation of Quantinuum combined Honeywell Quantum Solutions with Cambridge Quantum. Public announcements described an integrated hardware and software company and long-term manufacturing support. The combination illustrates a vertical-integration thesis, including the ability to develop software across platforms while controlling a hardware roadmap.

Keysight's acquisition of Quantum Benchmark added error diagnostics, error suppression and performance-validation software to a broader quantum portfolio. The disclosed rationale illustrates the value of software that measures and improves hardware. It does not disclose a standalone algorithm valuation.

Quantum Machines acquired QDevil to extend quantum-control capability from gate level to qubit. SandboxAQ acquired Good Chemistry to add computational-chemistry technology, customers and talent. These transactions illustrate control-stack and domain-application theses.

The examples should not be treated as direct comparables without price, revenue, rights and evidence alignment. They are useful for identifying strategic value routes: vertical integration, validation, control, domain workflow, talent and customer access. The target should be mapped to the route supported by its evidence.

16 Construct the hypothetical valuation

The worked example uses a wholly hypothetical quantum software target. It reports USD 18 million of annual revenue: USD 11 million recurring product revenue, USD 4 million services and USD 3 million grants and other income. It has USD 96 million of unrestricted cash and USD 38 million of annual cash use. These amounts do not represent an observed company.

The central enterprise value is USD 420 million. It allocates USD 105 million to validated algorithms and applications, USD 90 million to compiler and orchestration assets, USD 60 million to customer relationships and contracts, USD 45 million to proprietary data and workflows, USD 70 million to team and know-how, and USD 50 million to roadmap options. Each value is a management assumption.

Four scenarios test portability. A constrained case in which the principal algorithm remains tied to one provider has a value of USD 140 million and a probability of 25 percent. A source-portable but performance-variable case has a value of USD 360 million and a probability of 40 percent. A performance-portable product with repeat customers has a value of USD 820 million and a probability of 27 percent. A category platform with broad hardware access has a value of USD 1.80 billion and a probability of 8 percent. The illustrative probability-weighted value is USD 508.4 million before integration, financing and dilution.

The difference between central allocated value and probability-weighted scenario value exposes assumptions. A buyer should adjust for integration cost, retention, customer consent, hardware contracts and funding. Contingent consideration can align payment with portability tests and customer retention.

17 Structure consideration around evidence

Upfront consideration can pay for controlled code, rights, cash flow and transferable capability. Deferred payments can depend on a clean build, held-out multi-target benchmark, customer retention or accepted product release. Retention awards should reward service and knowledge transfer.

Milestones should specify targets, versions, workloads, baselines, metrics and independent review. A general requirement that software remain hardware agnostic invites dispute. A better milestone states that defined workloads compile and execute on agreed targets while meeting solution-quality, time and cost thresholds.

The agreement should accommodate technical change. New hardware may replace an original target. Substitution rules should preserve economic equivalence and prevent either party from choosing an artificially easy or impossible test. Qualified advisers should address accounting, tax, employment and securities consequences.

The buyer should also protect data and customer continuity. Consents, transition services, cloud access and security remediation can be closing conditions or covenants. An escrow can address identified rights gaps. Each protection should map to a value-moving risk.

18 Plan the first one hundred days

The first thirty days should secure repositories, credentials, cloud accounts, customer data and release pipelines. The buyer should confirm critical staff, freeze evidence baselines and preserve provider access. Customer communications should explain continuity without making unsupported performance promises.

Days thirty to sixty should reproduce builds, rerun key benchmarks, map product architecture and validate customer obligations. The integration team should separate shared services from protected scientific work. A combined roadmap should identify which platforms remain supported and why.

Days sixty to one hundred should rationalise overlapping tools, approve the product and hardware matrix, remediate security, and test knowledge transfer. Commercial leaders should review pricing, support and renewal plans. Finance should reconcile integration spend and milestone progress with the investment case.

The board should receive a value-realisation report. It should show achieved synergies, preserved customers, portability evidence, cash use, risks and next decisions. Technical learning should update valuation assumptions rather than being reported only as activity.

Conclusion

Quantum software value is not established by a hardware-agnostic label. It depends on the extent to which algorithms, compilers, workflows and customer outcomes survive a change in execution environment. Source translation, successful compilation and equivalent commercial performance are different evidence states.

An acquirer should build an algorithm evidence ladder, dependency graph, portability matrix, benchmark plan and customer cohort. It should reproduce builds, control for provider features, measure quality and time to solution, confirm rights and reconcile customer cash. Achieved assets should be separated from conditional roadmap value.

Open representations and standards can reduce switching cost while exposing target-specific capability limits. Strategic combinations show that software can create value through vertical integration, diagnostics, control and domain applications. The value of a specific target still requires transaction-specific evidence.

Boards can use this framework to decide whether to acquire, invest, licence, partner or build. Every material value component should point to controlled rights, reproducible performance, accepted customer outcomes or an explicitly stated management assumption. Consideration should move when that evidence changes.

Appendix A Portability test protocol

The protocol should freeze source, dependencies, compiler versions, target profiles, data, seeds and acceptance criteria. It should include a primary target, a credible alternative and a simulator or emulator. The target should build from a clean environment using documented credentials and infrastructure.

Each run should record compilation time, circuit width and depth, native operations, shots, queue and execution time, classical post-processing, retries, total cost and solution quality. Exclusions and failed runs should remain in the record. A baseline implementation should use the same access and configuration.

The report should classify source, semantic, compilation, execution, performance and commercial portability. It should state which changes were required and whether they form maintained product branches. Independent reviewers should have enough access to reproduce the conclusion.

Appendix B Customer and revenue evidence

The customer schedule should identify legal counterparty, product, workload, hardware provider, term, committed value, services, acceptance, invoicing, cash, renewal and change of control. Research funding and grants should be separated from recurring product revenue.

The cohort analysis should show opening recurring revenue, new customers, expansion, contraction, churn and closing recurring revenue. It should reconcile to accounting records. Pipeline should remain outside contracted revenue and should identify procurement, technical and budget gates.

Customer references should confirm the decision supported, baseline, accepted output, specialist effort and future intention. The buyer should test whether the relationship survives a hardware or ownership change.

Appendix C Valuation data room

The technical data room should contain repositories, build instructions, dependency locks, software bills of materials, licences, contributor agreements, patents, data rights, benchmarks, raw execution records, target configurations, security records and incident history. It should include negative and failed experiments.

Commercial records should include contracts, statements of work, grants, invoices, cash receipts, support logs and usage. Financial records should reconcile revenue categories, gross margin, research spending, cash, commitments and monthly cash use.

The diligence report should identify what was reproduced, sampled, unavailable or disputed. Each unresolved gap should appear in price, structure, integration budget or approval conditions.

Appendix D Integration value ledger

The integration value ledger should distinguish revenue, cost, capital and strategic effects. Revenue effects can include customer retention, cross-sell, new platform access and faster conversion from pilots. Cost effects can include removal of duplicated infrastructure, procurement leverage and lower external licence spend. Capital effects include avoided internal development and reduced time to the next product release. Strategic effects include control of a critical compiler, workflow, data asset or customer interface.

Each item should have a baseline, accountable owner, timing, required spending and evidence source. A gross synergy estimate should be reduced for customer attrition, product disruption, retention awards, cloud migration, security remediation, duplicated platform support and tax. Benefits that depend on future hardware should be probability weighted and shown separately from near-term actions.

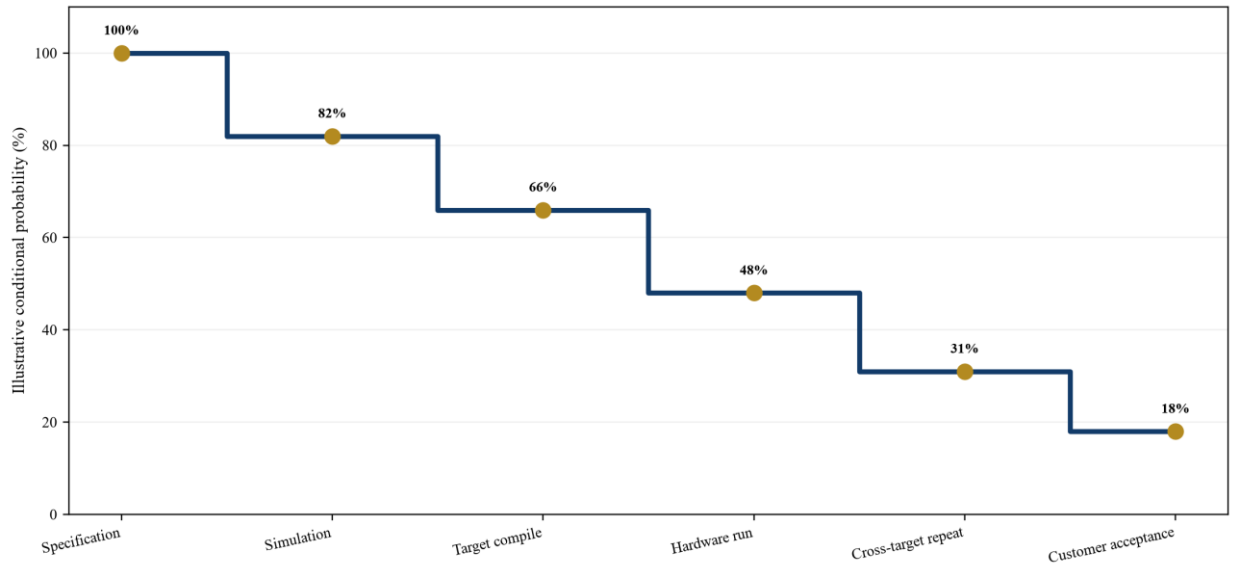
The ledger should prevent double counting. Algorithm portability can support customer retention and reduce rebuilding cost, yet the same benefit should not appear in both lines without a reconciliation. The buyer should also distinguish value transferred from another business unit from value created for the combined group. Moving cloud or engineering cost into a central budget does not itself create enterprise value.

Post-close reporting should compare realised value with the approved case. Technical metrics should connect to commercial outcomes. A successful multi-target benchmark has limited transaction value until it reduces support cost, retains a customer, expands distribution or unlocks a product. A customer renewal should be attributed carefully when it also depends on sales, hardware progress or pricing.

The board should review the ledger at defined intervals and revise the investment case when evidence changes. Unachieved value should trigger a product, capital or integration decision. The purpose is to preserve a traceable connection among the acquisition thesis, technical evidence, cash and accountable execution.

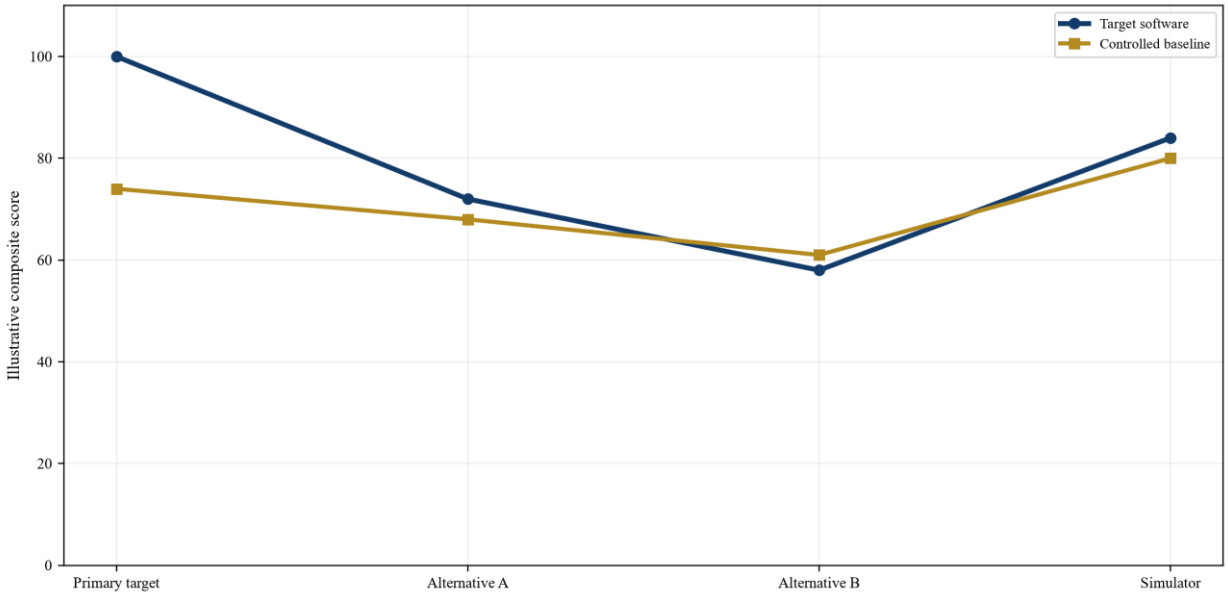
Appendix E Decision figures and tables

Figure 1. Algorithm evidence ladder from mathematical claim to accepted customer outcome



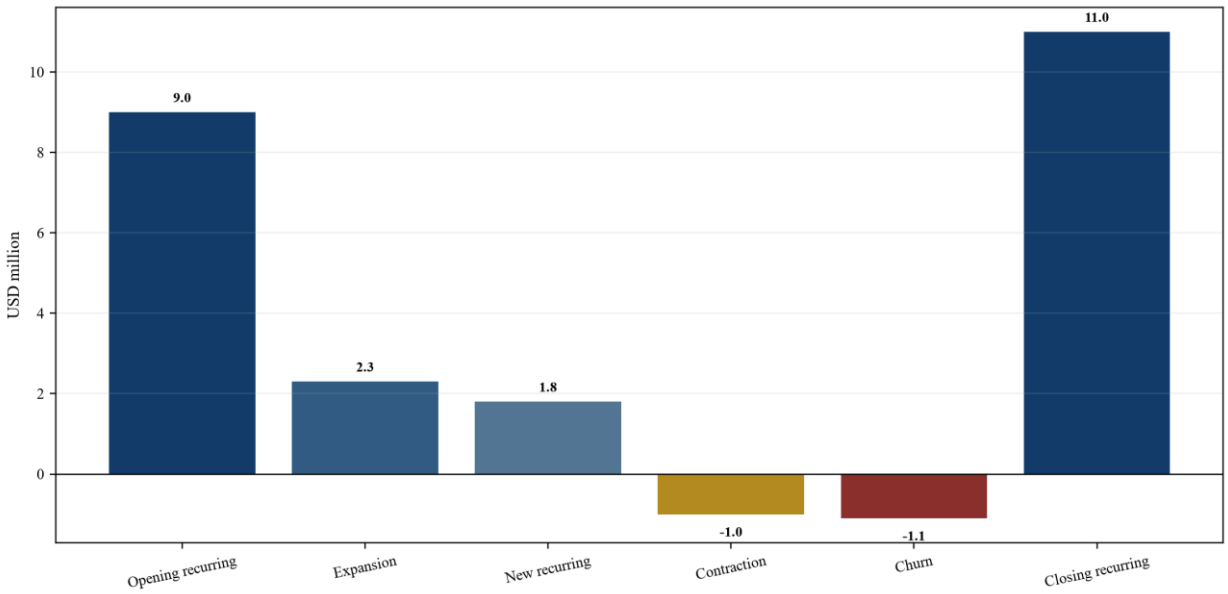
Proposed sequence; each stage requires a reproducible record and defined acceptance test.

Figure 2. Hypothetical performance portability across execution targets



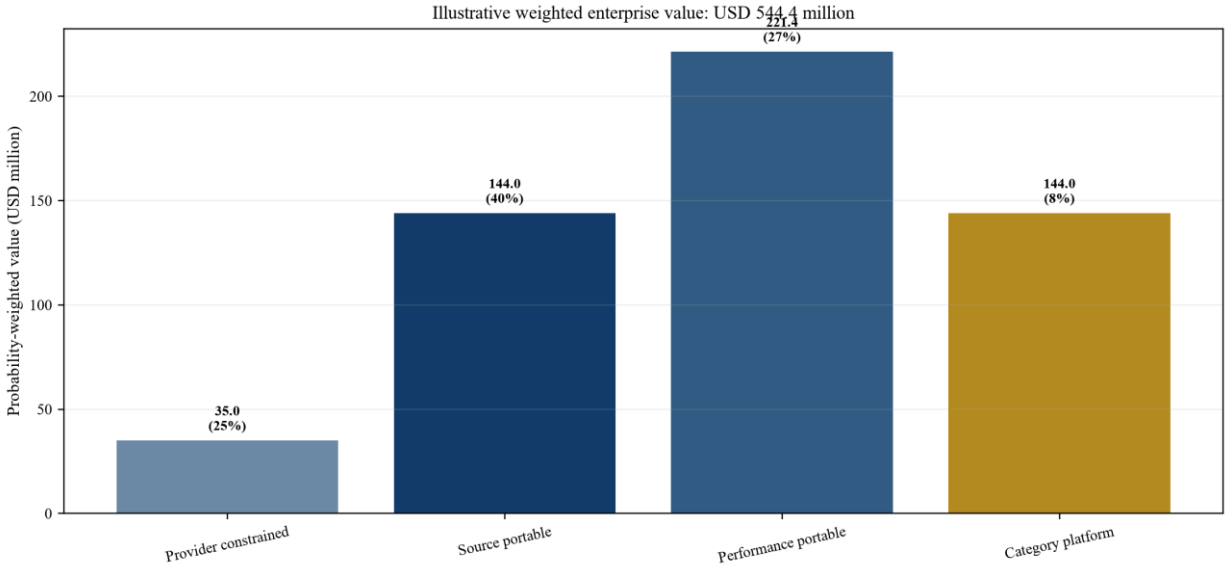
Wholly hypothetical management assumptions; score combines normalised solution quality, time and cost.

Figure 3. Hypothetical customer revenue cohort



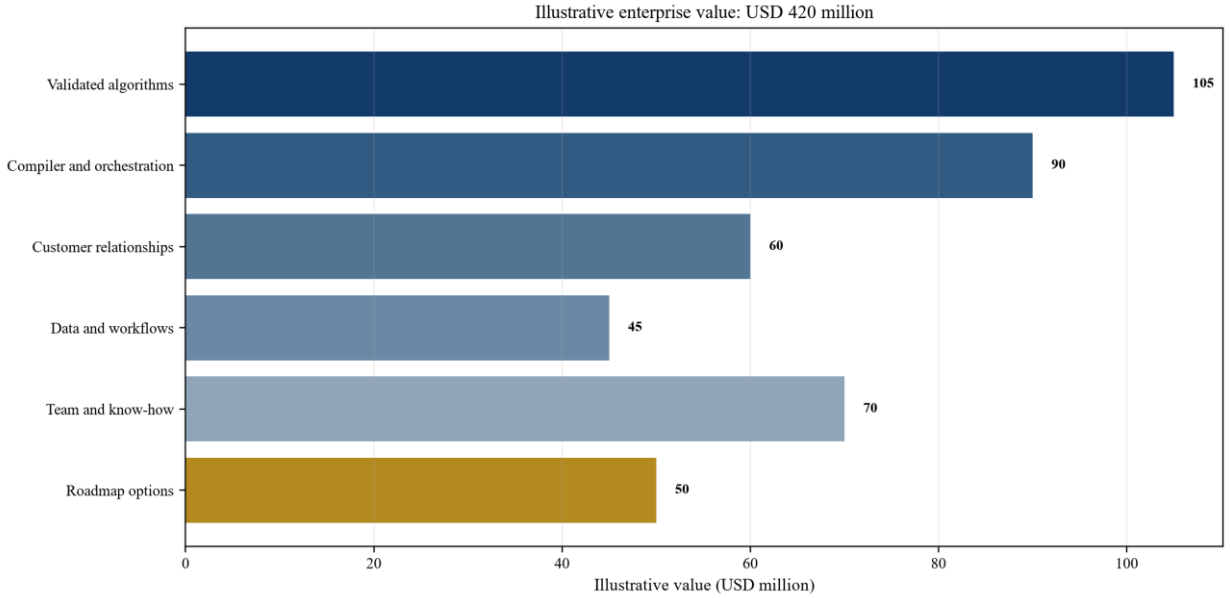
Wholly hypothetical management assumptions; USD million.

Figure 4. Hypothetical probability-weighted enterprise value



Wholly hypothetical management assumptions; USD million.

Figure 5. Hypothetical evidence-based valuation bridge



Wholly hypothetical management assumptions; USD million.

Table 1. Portability hierarchy for quantum software

Layer	Test	Common failure	Valuation use
Source	Code translates or is expressed through a shared language	Syntax moves while dependencies remain	Limited transferability evidence
Semantic	Mathematical intent remains equivalent	Target restrictions change the method	Algorithm evidence adjustment
Compilation	Programme lowers into the target stack	Unsupported control, gates or runtime features	Retargeting cost
Execution	Complete workflow runs under supported conditions	Hidden provider service or manual intervention	Achieved capability test

Layer	Test	Common failure	Valuation use
Performance	Quality, time and cost remain within threshold	Equivalent output becomes uneconomic	Product and option value
Commercial	Customer accepts, pays and renews	Hardware or owner change breaks adoption	Relationship and income value

Proposed hierarchy; each layer requires its own evidence.

Table 2. Algorithm evidence ladder

Stage	Required record	Independent check	Valuation treatment
Mathematical claim	Problem, assumptions and correctness criterion	Expert review	Research option only
Reproducible simulation	Code, data, seeds and classical baseline	Clean-environment rerun	Know-how and code value
Target compilation	Compiler, profile, circuit and resource records	Repeat build	Conditional implementation value
Hardware execution	Jobs, calibration, shots, time, cost and output	Held-out run	Achieved technical evidence
Cross-target performance	Comparable targets and acceptance thresholds	Independent benchmark	Portability uplift
Customer acceptance	Contract, delivery, invoice and cash	Customer and accounting reconciliation	Commercial value

Proposed transaction-diligence sequence.

Table 3. Dependency map

Layer	Evidence	Dependency risk	Value implication
Languages and libraries	Repositories, versions, licences and tests	Breaking changes and unowned components	Maintenance and remediation cost
Intermediate representation	Profiles, transformations and validation	Unsupported semantics or target features	Portability adjustment
Compiler and passes	Build, benchmarks, ownership and documentation	Target-specific optimisation	Achieved asset or retargeting cost
Hardware and cloud	Contracts, access, queues, pricing and features	Provider concentration and change of control	Performance and customer adjustment
Classical workflow	Data, HPC, AI, post-processing and orchestration	Quantum claim depends on classical assets	Allocate value to complete system

Proposed minimum review of the software execution chain.

Table 4. Hypothetical valuation allocation

Value component	Illustrative amount	Evidence gate	Downside treatment
Validated algorithms and applications	USD 105 million	Reproducible benchmarks and rights	Portability holdback
Compiler and orchestration	USD 90 million	Clean build, target evidence and ownership	Remediation reserve
Customer relationships and contracts	USD 60 million	Acceptance, renewal, cash and consent	Retention adjustment

Value component	Illustrative amount	Evidence gate	Downside treatment
Proprietary data and workflows	USD 45 million	Provenance, transfer rights and contribution	Rights deduction
Team and know-how	USD 70 million	Critical-role retention and knowledge transfer	Service-based retention
Roadmap options	USD 50 million	Funded portability and customer milestones	Contingent consideration

Wholly hypothetical management assumptions; not observed company or transaction data.

Table 5. Customer evidence quality ladder

Evidence	What it supports	Limitation	Valuation use
Research collaboration	Access and technical engagement	May lack recurring budget or acceptance	Relationship evidence
Grant or award	Funded scope and policy support	Restricted use and limited recurrence	Contracted cash with conditions
Paid pilot	Customer budget and defined experiment	Bespoke work may not scale	Probability-adjusted conversion
Accepted product delivery	Performance against agreed criteria	Does not establish renewal	Contract and delivery value
Repeat paid use	Continuing budget and operational relevance	Can remain provider dependent	Stronger income evidence

Proposed classification for revenue diligence.

Table 6. Hypothetical revenue and runway profile

Measure	Illustrative amount	Diligence question	Valuation consequence
Recurring product revenue	USD 11 million	Renewal, margin and portability	Income support
Services revenue	USD 4 million	Founder effort and repeatability	Services adjustment
Grants and other income	USD 3 million	Restrictions and recurrence	Separate from product multiple
Unrestricted cash	USD 96 million	Availability and commitments	Equity-value reconciliation
Annual cash use	USD 38 million	Next evidence gate and financing lead time	Runway and dilution adjustment

Wholly hypothetical management assumptions; USD million.

Table 7. Board approval thresholds

Decision area	Green evidence	Amber condition	Red condition
Portability	Held-out workloads meet quality, time and cost thresholds across agreed targets	Source and execution portability with performance variation	Marketing claim without reproducible target evidence
Rights	Code, data, licences and contributor rights confirmed	Bounded remediation with cost and date	Critical capability unowned or non-transferable
Customers	Paid acceptance, renewal and cash reconciled	Pilot or funded research with conversion plan	Logos or pipeline treated as recurring revenue

Decision area	Green evidence	Amber condition	Red condition
Team and security	Critical roles retained; clean build and access transfer passed	Documented remediation and retention plan	Founder credentials or insecure supply chain required
Valuation	Achieved assets and conditional options separated	Wide but explicit scenario range	Market forecast substitutes for evidence

Proposed decision framework.

Frequently asked questions

Q: Does support for an open quantum language make software hardware independent? A: It can improve source and compilation portability. The target environment still determines gates, control flow, measurement, timing, error behaviour, cost and runtime services. Performance and commercial portability require separate tests.

Q: What should an acquirer benchmark? A: Use held-out customer-relevant workloads with defined classical baselines. Record solution quality, compilation, circuit resources, queue and execution time, post-processing, retries, total cost and specialist effort across agreed targets.

Q: How should open-source quantum software be valued? A: Identify the proprietary complement: data, optimisation, workflow integration, support, customer relationships, hosted service or specialist know-how. Review licences, contributor rights, community health and the cost of maintaining forks.

Q: What customer evidence supports value? A: Executed contracts, accepted deliveries, invoices, cash, repeat use and renewal provide progressively stronger evidence. Collaborations, grants and pilots should be classified separately according to obligations and recurrence.

Q: How should a buyer treat hardware-provider access? A: Review contracts, assignment, pricing, queues, reserved capacity, support, data and change of control. Separate software value from privileged access and test a credible alternative target.

Q: Why separate services revenue? A: Services can demonstrate expertise and customer demand while depending on scarce staff and bespoke work. The buyer should test delivery effort, gross margin, repeatability and conversion into a maintained product.

Q: Which transaction structures fit portability risk? A: Upfront consideration can pay for controlled assets and achieved cash flow. Holdbacks, contingent consideration and staged investment can depend on clean builds, multi-target benchmarks, customer retention and accepted product releases.

Q: What should a board require before approval? A: Require reproducible builds, rights mapping, a portability matrix, held-out benchmarks, customer and cash reconciliation, critical-role retention, security remediation, integration cost, runway and a valuation bridge separating achieved capability from conditional options.

References

[1] Quantum Economic Development Consortium, Application-oriented performance benchmarks for quantum computing. https://quantumconsortium.org/publication/application-oriented-performance-benchmarks-for-quantum-computing/

[2] Quantum Economic Development Consortium, Quantum algorithm exploration using application-oriented performance benchmarks, 2024. https://quantumconsortium.org/publication/quantum-algorithm-exploration-using-application-oriented-performance-benchmarks/

[3] Quantum Economic Development Consortium, Standards and Performance Metrics Technical Advisory Committee. https://quantumconsortium.org/tac/standards/

[4] Microsoft Azure Quantum, Quantum Intermediate Representation. https://learn.microsoft.com/en-us/azure/quantum/concepts-qir

[5] Microsoft Azure Quantum, QIR target profiles in the Quantum Development Kit, 2026. https://learn.microsoft.com/en-us/azure/quantum/quantum-computing-target-profiles

[6] Microsoft Azure Quantum, Backend quantum simulators from quantum providers, 2026. https://learn.microsoft.com/en-us/azure/quantum/backend-simulators

[7] OpenQASM, OpenQASM 3 live specification. https://openqasm.com/

[8] Cross and collaborators, OpenQASM 3: A broader and deeper quantum assembly language, ACM Transactions on Quantum Computing, 2022. https://doi.org/10.1145/3505636

[9] Amazon Web Services, Support for OpenQASM on different Amazon Braket devices. https://docs.aws.amazon.com/braket/latest/developerguide/braket-openqasm-device-support.html

[10] Amazon Web Services, OpenQASM features supported by Amazon Braket. https://docs.aws.amazon.com/braket/latest/developerguide/braket-openqasm-supported-features.html

[11] IBM Quantum, Qiskit transpiler documentation. https://docs.quantum.ibm.com/guides/transpile

[12] IBM Quantum, Backend and target documentation. https://docs.quantum.ibm.com/api/qiskit/qiskit.transpiler.Target

[13] Google Quantum AI, Cirq documentation. https://quantumai.google/cirq

[14] NVIDIA, CUDA-Q documentation. https://nvidia.github.io/cuda-quantum/latest/

[15] NVIDIA, cuQuantum SDK documentation, 2026. https://docs.nvidia.com/cuda/cuquantum/26.01.0/index.html

[16] Xanadu, PennyLane documentation. https://docs.pennylane.ai/

[17] Linux Foundation, QIR Alliance. https://www.qir-alliance.org/

[18] National Institute of Standards and Technology, Quantum information science. https://www.nist.gov/topics/quantum-information-science

[19] ISO, ISO/IEC 4879:2024 Quantum technologies; Vocabulary. https://www.iso.org/standard/80432.html

[20] Honeywell, Honeywell Quantum Solutions and Cambridge Quantum complete business combination, 2021. https://www.honeywell.com/us/en/news/press-releases/2021/11/honeywell-quantum-solutions-and-cambridge-quantum-complete-business-combination-to-form-world-s-largest-most-advanced-standalone-quantum-company

[21] Quantinuum, Introducing Quantinuum, 2021. https://www.quantinuum.com/press-releases/introducing-quantinuum

[22] Quantinuum, Registration statement filed with the United States Securities and Exchange Commission, 2026. https://www.sec.gov/Archives/edgar/data/2110105/000162828026032836/quantinuum-sx1.htm

[23] Keysight Technologies, Keysight Technologies acquires Quantum Benchmark, 2021. https://investor.keysight.com/investor-news-and-events/financial-press-releases/press-release-details/2021/Keysight-Technologies-Acquires-Quantum-Benchmark/default.aspx

[24] Quantum Machines, Quantum Machines acquires QDevil, 2022. https://www.quantum-machines.co/press-release/quantum-machines-acquires-qdevil-to-provide-full-stack-orchestration-platform/

[25] SandboxAQ, SandboxAQ acquires Good Chemistry, 2024. https://www.sandboxaq.com/press/sandboxaq-buys-good-chemistry-melding-ai-and-quantum-tech-for-drug-discovery

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

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

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

[29] International Valuation Standards Council, International Valuation Standards. https://www.ivsc.org/standards/

[30] United States Securities and Exchange Commission, Financial Reporting Manual. https://www.sec.gov/corpfin/cf-manual

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

[32] National Institute of Standards and Technology, Cybersecurity Framework 2.0. https://www.nist.gov/cyberframework

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.