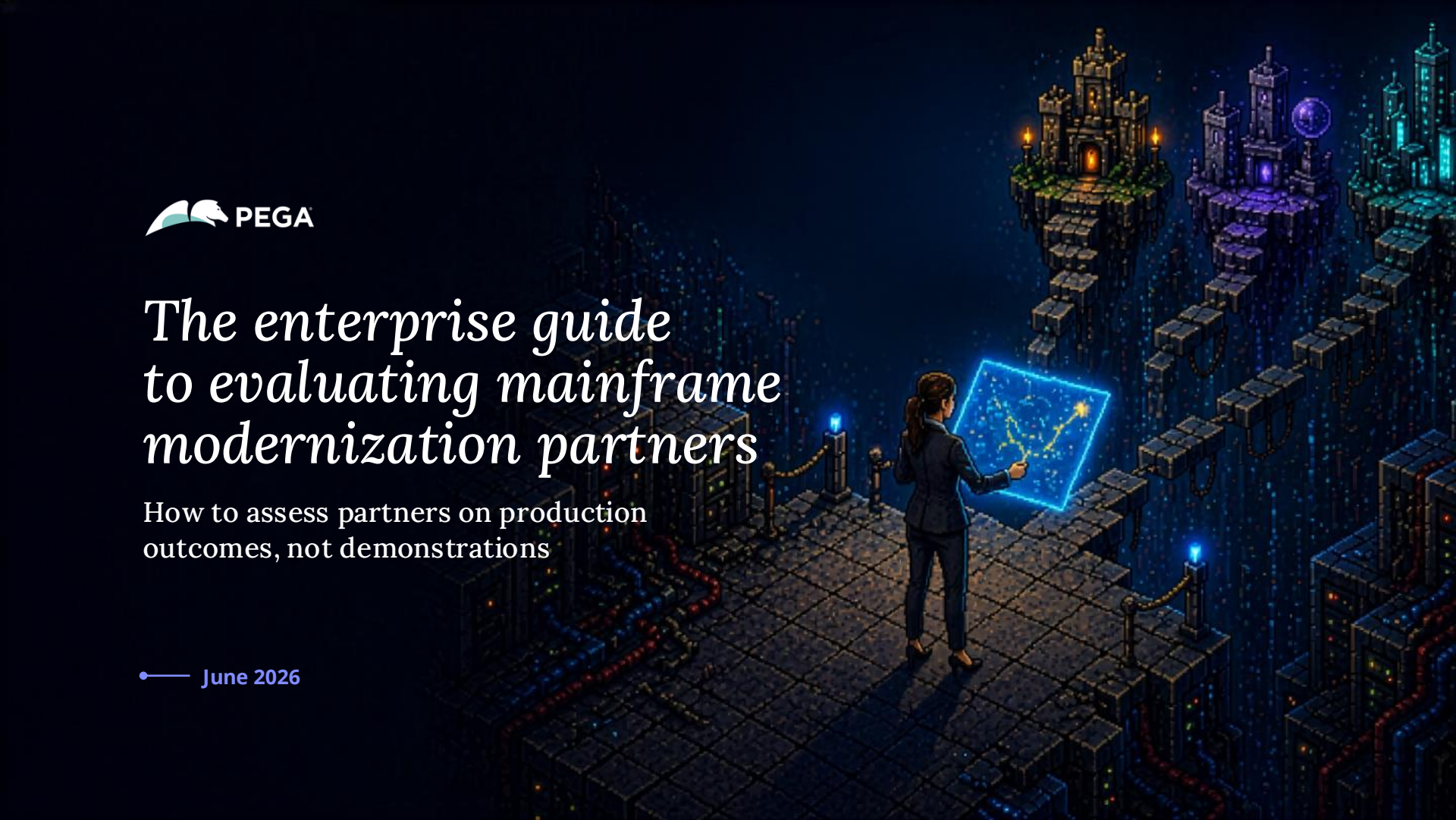


The enterprise guide to evaluating mainframe modernization partners

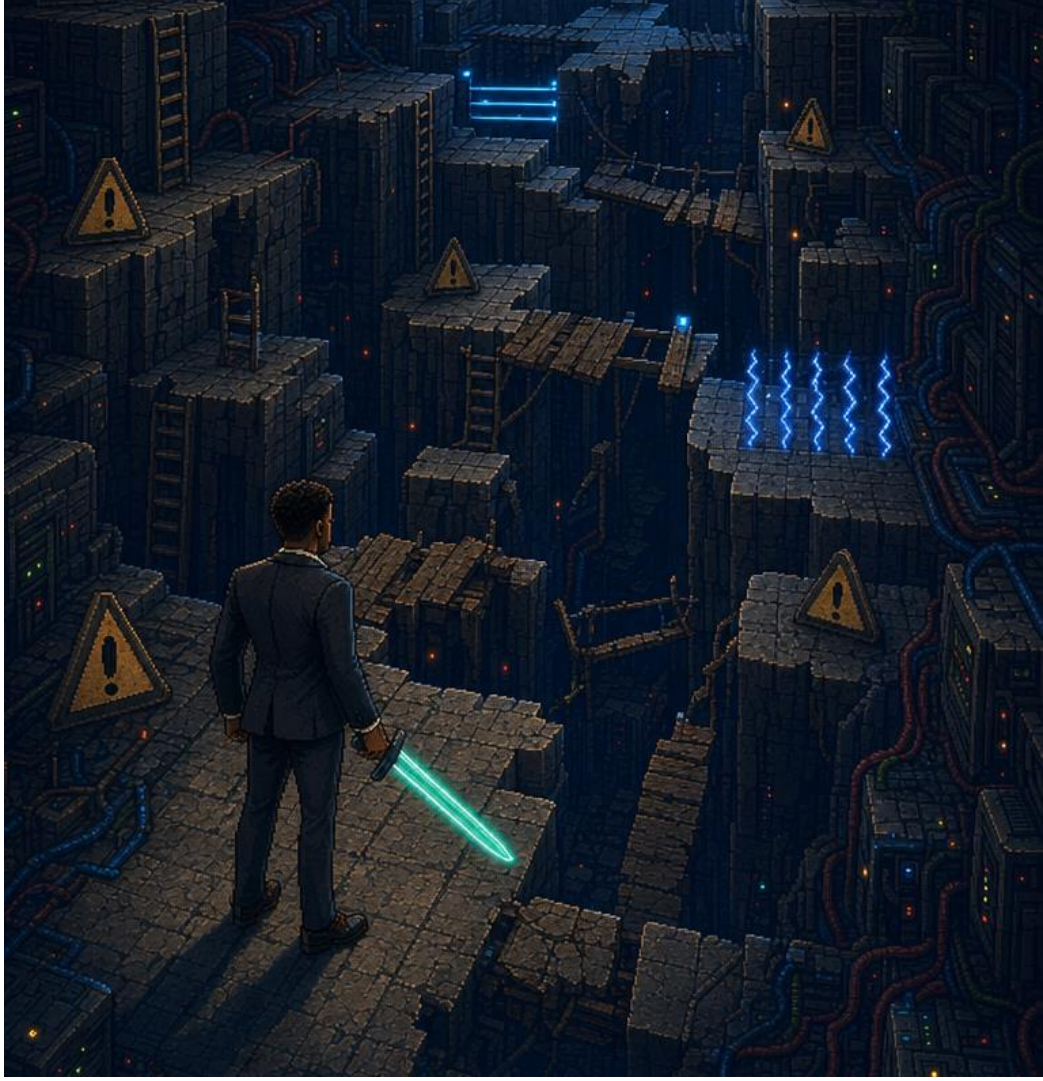
How to assess partners on production outcomes, not demonstrations

June 2026

Most large enterprises have started the modernization quest – yet few move mission-critical workloads safely into production.

That gap is not caused by tools or strategy. It shows up between the process steps– starting with discovery that does not flow into design, business rules that cannot be traced to legacy behavior, generated artifacts that need rework, and cutover planned too late.

This guide is designed to help you evaluate modernization partners on one question: Can they move a critical workload from legacy understanding to validated production through a connected, governed execution chain? It assumes you already know the strategies, and focuses on choosing partners, patterns, and pilots with confidence.



Why modernization stalls

Modernization typically stalls in predictable places – and these are the traps in the dungeon:

DEMONSTRATIONS AVOID THE HARD PARTS.

A demo works on a self-contained application or a clean code sample. The real test is the mission-critical components: embedded business rules, batch dependencies, data access patterns, integrations, and exception paths. A credible partner will work with a representative workload, not just a sample.

OUTPUT MUST BE TRUSTED BEFORE IT SHIPS.

Generated rules, tests, or code are not production-ready because they were produced quickly. They must trace back to legacy behavior, pass business and technical validation, and prove functional equivalence against a baseline. Output that cannot be traced and audited cannot support a production decision.

ARTIFACTS MUST CONNECT ACROSS PHASES.

Value leaks when discovery produces documents that people re-key into design tools, test plans, and cutover workstreams. Strong partners show artifacts flowing from discovery through design, build, test, and operational readiness without repeated manual translation.

COEXISTENCE IS ENGINEERED, NOT ASSUMED.

AI can redesign an application, but it does not migrate the data, hold interface continuity, protect transaction integrity, or run a hybrid estate during transition. Most programs need a coexistence and retirement model planned early, before build decisions lock.

ONE APPLICATION IS NOT A FACTORY.

A pilot proves feasibility; a program needs repeatable patterns, reusable assets, governance, and a wave model that improves over time. The objective is a repeatable capability for reducing risk and debt across the portfolio, and a target state the client can maintain after the partner exits.

OWNERSHIP SITS WITH A PROGRAM, NOT A LAB.

Experiments stall when they lack an executive sponsor, funding authority, and named owners across architecture, data, security, and operations. Production modernization is an accountable program, not an innovation showcase.

AI is a capability inside this chain. Where it accelerates discovery or generation, the output must be validated, traceable, and connected to the build, or it adds risk rather than removing it.

Why the partner decision is hard

Mainframe modernization is a program decision, not a standard software or solution purchase. It affects business continuity, operating model, data strategy, and long-term delivery velocity. Choosing a partner is a critical program decision – with boss-level stakes.

THE SYSTEMS ARE MISSION CRITICAL.

Mainframes often hold the core execution path for transaction processing, policy enforcement, claims, billing, or settlement, even when the customer experience has moved to modern channels. Modernizing a core system of record is not the same as modernizing a secondary application.

THE DECISION IS HARD TO UNWIND.

The chosen path shapes how logic data is governed, how integrations are rebuilt, and how future change is delivered. A poor choice can create a second legacy estate rather than reduce the first.

EXECUTION SPANS MANY TEAMS.

Value leaks when discovery produces documents that people re-key into design tools, test plans, and cutover workstreams. Strong partners show artifacts flowing from discovery through design, build, test, and operational readiness without repeated manual translation.

What to evaluate

Partners should be assessed against six key areas – your pre-boss checklist:

EVALUATION AREA	WHAT GOOD LOOKS LIKE
Discovery methodology and tooling	Produces validated artifacts describing applications, dependencies, rules, data flows, batch behavior, and integrations, connecting them directly to design, test, and cutover planning.
Transformation philosophy	Distinguishes migration from modernization and defines the future state in business terms before mapping to architecture and platform. Lift-and-shift is not a level-up.
End-to-end toolchain integration	Connects discovery, design, build, test, deployment, and operational readiness, reducing manual re-entry and preserving traceability across phases.
Mainframe and delivery expertise	Covers COBOL, PL/I, CICS, JCL, VSAM, DB2, and batch operations, with a clear plan for higher-risk technologies, such as Assembler, IMS, IDMS, and proprietary utilities. If they haven't fought this boss before, now is a bad time to find out.
Risk, data, and cutover management	Defines coexistence, data migration, reconciliation, parallel run, rollback triggers, performance parity, security controls, and acceptance criteria before transition.
Pilot structure and scale path	Starts with a bounded workload and clear success measures, then shows how the approach industrializes across waves through reusable patterns and governance. Level 1 first – then the factory.

Illustrative strategy scorecard

A model for ranking for modernization paths (1 = weak fit, 5 = strong fit). Pick your character wisely – choosing the wrong one for the boss fight is a wipe.

CRITERION	LIFT & SHIFT	WRAP & EXTEND	REPLATFORM	REFACTOR	REIMAGINE
Business agility gained	1	3	2	3	5
Legacy debt reduction	1	2	2	4	5
Speed & near-term affordability	5	4	3	2	4
Cutover & delivery risk control	4	5	3	2	3
Long-term TCO & maintainability	2	2	3	4	5
Change management	5	4	4	3	2
Total (equal weight)	**18**	**20**	**17**	**18**	**24**

Read the strategies by intent. Lift & shift fits a data-center exit or platform-risk play but does little for agility or debt. Wrap & extend buys time and exposes legacy capabilities through modern interfaces yet preserves the core unless paired with a retirement plan. Replatform and refactor improve manageability or technical architecture but can shift debt rather than remove it. Reimagine scores highest where the goal is to redesign workflows and shed legacy constraints, though only with disciplined scope, strong discovery, and explicit coexistence planning. None of these is automatically right for every workload.

Questions that matter

Use these to test whether a partner can support production modernization, not just a convincing demonstration.

DISCOVERY AND BASELINE

1. Describe your discovery methodology and the artifacts it produces, and how those artifacts feed design, build, test, and cutover.
2. Provide a validated artifact set from a comparable engagement, and explain how each was validated by business, technical, and operations stakeholders.
3. How do you establish a behavioral baseline, including batch dependencies, interfaces, data ownership, and exception paths, to prove future equivalence?

TRANSFORMATION AND AI TRUST

4. Give an example of how a workload modernized with your approach differs from the same workload replatformed.
5. Where is AI used, what context does it rely on, and how is its output validated before it informs design or build?
6. On a representative workload, what share of AI-generated output typically needs human rework? How are errors caught before build?
7. How do you keep the target state from becoming a new black box your teams cannot maintain?

TOOLCHAIN AND TRACEABILITY

8. Which artifacts move programmatically from discovery to design, test, and deployment, and where is manual re-entry still required?
9. Show traceability from legacy behavior to requirement, future workflow, test case, and production validation.
10. How are changes governed after go-live, and what is exportable if we change platform, partner, or cloud provider?

MAINFRAME SCOPE

11. Which mainframe technologies do you support directly (COBOL, PL/I, CICS, JCL, VSAM, DB2, IMS, IDMS, Assembler, schedulers), and how do unsupported ones affect risk, cost, and timeline?
12. How do you decide whether a workload should be rehosted, replatformed, refactored, wrapped, reimagined, or retired, and which workloads are a poor fit for your approach?

DATA, SECURITY, AND CUTOVER

13. What is the system-of-record strategy during coexistence, and when does data ownership move to the target state?
14. How are sensitive data flows protected during replication, testing, coexistence, and cutover?
15. How do you validate data integrity, functional equivalence, and performance parity during parallel run? What triggers rollback?
16. What monitoring, runbooks, and incident processes are in place before go-live? How do our teams operate the target state after you exit?

PILOT, SCALE, AND COMMERCIAL

17. What workload suits the first pilot, what outcomes define success, and what assumptions must hold?
18. What patterns, assets, and governance carry into wave two? What specifically makes it faster and lower risk?
19. Who owns generated artifacts, rules, configuration, and accelerators? What is portable in open formats?
20. How do you model full transition economics (parallel run, retained mainframe capacity, license overlap, migration, decommissioning), not just your delivery scope?

What strong programs look like

Strong programs aren't one-off runs – they're repeatable delivery capabilities. Check out your six-phase quest map:

PHASE 0

Executive alignment. One accountable sponsor; named owners across architecture, data, security, and operations; a portfolio segmented into retain, retire, modernize, and reimagine candidates; and funding tied to outcomes per wave. A decision cadence keeps it out of committee paralysis.

PHASE 1

Discovery that produces usable artifacts. Application inventory, dependency and data-flow maps, a business-rule inventory, and a behavioral baseline, in a form stakeholders validate and that feeds design and test directly rather than sitting as a static report.

PHASE 2

Future-state design. Target workflow, data ownership, integration and batch strategy, coexistence model, security controls, and acceptance criteria defined and validated before build scales.

PHASE 3

Build, test, and production readiness. Automated tests tied to the baseline; functional, data, performance, and security validation; monitoring, runbooks, and an incident model; and cutover, rollback, and parallel-run plans. The program can state what "ready" means, and cutover is engineered rather than improvised.

PHASE 4

Cutover and stabilization. Parallel run with reconciliation checkpoints, rehearsed rollback triggers, production monitoring, and clear ownership through stabilization, closing with business sign-off and operational handoff. The program exits stabilization on evidence.

PHASE 5

Scale the factory. Reusable patterns, standardized discovery assets, shared governance, and delivery metrics turn the first wave into a repeatable capability. This is where a pilot becomes an enterprise program.

Business case, cost, and lock-in

Inaction has a price tag – and legacy drag costs more than most enterprises admit.

BUILD THE CASE AGAINST THE COST OF INACTION.

Compare modernization with the current trajectory, not with zero. Account for maintenance and license cost, delivery delays, scarce legacy skills, operational concentration risk, and the inability to support new digital and AI-enabled models. Fund in waves so the program can adjust without a large sunk commitment.

MODEL THE WHOLE TRANSITION.

Ask partners to model cost discovery, build, test, data migration, parallel run, retained mainframe capacity, license overlap, security validation, operational readiness, decommissioning, and target-state run cost. A partner that models only its own delivery scope is not showing you the real economics.

TREAT LOCK-IN AS A MANAGED DECISION.

Every path creates dependence: platform, cloud, SI, tooling, AI-generated artifacts, data models, and runtime. The test is whether it is understood, acceptable, and reversible at reasonable cost. Settle artifact ownership, open-format export, and exit paths up front. Reluctance to discuss exit is itself a signal.

A practical evaluation sequence

Structure the evaluation as a sequence, each step gated by evidence rather than assertion.

STEP	DECISION QUESTION	EVIDENCE REQUIRED
1. Scope the workload	Is this workload representative of real complexity?	Inventory, dependencies, batch scope, data flows, interfaces, business criticality
2. Validate the baseline	Do we understand current behavior well enough to test the future?	Behavioral baseline, rule inventory, test evidence, SME validation
3. Select the pattern	Which path fits business goals, risk, and constraints?	Pattern rationale, tradeoff analysis, target-state architecture
4. Prove the chain	Can artifacts flow from discovery to design, build, test, and cutover?	Toolchain walkthrough using representative artifacts
5. Prove production readiness	Can the partner define and meet go-live acceptance criteria?	Test, reconciliation, rollback, and operational-readiness plans
6. Prove scale	Can wave two be faster and lower risk than wave one?	Reusable assets, governance model, metrics, staffing model

The strongest partner is not the one with the best demonstration. It is the one that can show a mission-critical workload moving through this chain with evidence at each gate.

Press start on the right partner. The dungeon isn't getting any smaller.

Pega provides the leading AI-powered platform for enterprise transformation. The world's most influential organizations trust our technology to reimagine how work gets done by automating workflows, personalizing customer experiences, and modernizing legacy systems. Since 1983, our scalable, flexible architecture has fueled continuous innovation, helping clients accelerate their path to the autonomous enterprise.

Sources: aws.amazon.com/experience-based-acceleration, aws.amazon.com/mainframe-modernization.