MAINFRAME MODERNIZATION

A Practical Decision-Making Workbook

Case study • Modernization trade-offs • AI validation checklist

Estimated time: 45–60 minutes | Audience: mainframe developers, analysts, tech leads and modernization learners

What you will learn

Assess a legacy application, compare modernization approaches, identify trade-offs, and validate AI-generated explanations with evidence and independent tests. This is a practical exercise rather than a course brochure.

1. Case study: The Order Processing System

A fictional retailer runs a critical order-processing application on IBM Z. COBOL implements business rules, DB2 stores customer and order data, CICS supports online transactions, and scheduled JCL jobs handle overnight reconciliation. Dependencies are poorly documented, some interfaces use file transfers, and only a few specialists understand the full system.

Current issue	Business impact
Dependencies are poorly documented.	Changes take longer and can introduce unexpected failures.
Batch jobs have narrow processing windows.	Late completion can delay downstream reporting.
Older file-based interfaces remain.	Newer applications have limited near-real-time integration.
Knowledge is concentrated in a few specialists.	Support continuity and onboarding are difficult.

Your task

Recommend a first phase that improves understanding and integration while protecting order processing, data integrity, service levels and rollback capability. Budget, target platform and downtime constraints are not yet confirmed; identify what must be discovered before making a final architecture decision.

2. Selected approach: incremental, API-led modernization

For this sample, begin with dependency discovery and documentation. Then expose one low-risk, well-understood business capability through a governed API or service while leaving core order-processing logic in place initially. Add automated tests, monitoring and rollback controls before expanding. This is a provisional recommendation; discovery may change the decision.

Pros	Cons / trade-offs
• Smaller changes reduce the blast radius compared with a big-bang rewrite. • Benefits can be delivered and measured in stages. • Core processing can remain stable while a new interface is tested. • Each phase produces evidence about dependencies, performance and readiness. • The roadmap can be adjusted as assumptions are tested.	• A hybrid environment may need support for longer. • API security, versioning, design and ownership need governance. • Interfaces can add latency and operational complexity. • Benefits may remain limited until more components are modernized. • Requires strong regression, integration and end-to-end testing.

Reconsider this approach if discovery shows the component cannot be separated safely, API exposure creates unacceptable risk, or a replacement meets requirements with lower total risk.

© Copyright protected

3. AI prompt for understanding legacy code

Use only an organisation-approved AI tool and code you are permitted to share. Never paste confidential, personal, regulated or production code into an unapproved service.

Prompt template

Review the following sanitised COBOL excerpt. Explain: (1) its apparent purpose, (2) inputs and outputs, (3) data items and business rules, (4) visible calls and dependencies, (5) edge cases and possible failure points, (6) test cases that could validate behaviour, and (7) assumptions that cannot be confirmed from this excerpt. For every important claim, quote the relevant code lines or paragraph names. Clearly separate direct evidence from inference. If the excerpt is insufficient, state which copybooks, called programs, data definitions or business documents are needed. Do not invent missing business rules.

4. How to validate that the AI results are correct

Treat AI output as hypotheses, not an authoritative explanation. Validate important claims against source evidence and independent tests.

Validation step	How to perform it
1. Trace to source	For each claim about a variable, condition, call or update, locate the exact source lines, paragraph, copybook or called program. Mark unsupported claims unverified.
2. Confirm context	Review copybooks, file/database definitions, JCL, CICS/DB2 interfaces, called modules and approved business rules. A short excerpt rarely reveals full behaviour.
3. Independent SME review	Ask an experienced developer or business analyst to check critical rules, exceptions, error handling and side effects.
4. Test predicted behaviour	Create tests for normal cases, boundaries, invalid inputs, error paths, retries/duplicates and data integrity.
5. Compare expected vs actual	Use approved test data in a non-production environment. Compare outputs, database changes, return codes, messages and logs against trusted baseline results.
6. Do not self-validate	Do not rely only on tests generated by the same AI prompt/model. Have an independent tester design or review tests from requirements.
7. Keep an audit trail	Record prompt, code version, AI response, source references, test cases, results, reviewer and unresolved questions. Repeat checks when code changes.

Learning has never been so easy

Acceptance rule

Accept a claim only when supported by source evidence and, for behaviour that matters, verified against requirements or independent test results. If evidence is missing, label it “unverified” and ask for more context rather than guessing.

© Copyright protected

5. Discovery checklist

Area	Questions to answer
Business	Purpose, peak periods, service levels and impact of an outage.
Application	Programs, transactions, shared components, interfaces and active usage.
Data	Data stores, volumes, integrity rules, retention, batch windows and recovery objectives.
Dependencies	CICS, DB2, batch jobs, files, schedulers and downstream systems.
Security	Identities, permissions, encryption, audit and sensitive-data rules.
Operations	Incidents, monitoring, restart procedures, backup and on-call ownership.
Testing	Regression coverage, expected outputs, representative data and sign-off criteria.
Delivery	Skills, tools, environments, change windows and rollback options.

6. Decision worksheet

Decision item	Your response
Business outcome	_____
Three key facts	1. _____ 2. _____ 3. _____
Chosen approach	_____
Pros that matter most	_____
Cons / trade-offs to manage	_____
Key risk and mitigation	_____
Success measures	_____
AI claims and evidence	_____
Rollback trigger and owner	_____

Key takeaway

Incremental modernization can reduce risk and deliver value in stages, but the approach must be supported by discovery evidence. AI can accelerate code understanding; correctness comes from source traceability, domain review, independent testing and documented evidence.

Further reading: IBM, *Overcoming Inherent Challenges in Mainframe Application Modernization* — https://community.ibm.com/community/user/blogs/asit-dan2/2024/12/13/overcoming-inherent-challenges-in-mainframe-applic and IBM Redbooks, *Modernizing IBM i Applications* — https://www.redbooks.ibm.com/docs/MD260020/MD260020-17.html

Educational resource only. Validate technical decisions against your organisation's architecture standards, security policies, approved vendor documentation and production-change procedures.

© Copyright protected