NRI

Risks inherent in the rehost or rewrite approach to modernizing legacy systems

Ayumi Naradate / Kazuhiro Torii

10 October 2024

lakyara vol.391

Nomura Research Institute, Ltd.

Executive Summary

Insurers are migrating away from legacy mainframe systems through rehosting or rewriting. However, such modernization projects tend to go awry when undertaken without sufficient preparation or with too much priority placed on minimizing project costs. Such pitfalls, which have in fact cropped up in some high-profile cases, can be avoided by thoroughly analyzing the legacy system before attempting to modernize it.

Ayumi Naradate
General Manager
Sapporo Solution Development Department III

Kazuhiro Torii
General Manager
SA Systems Business Department

NOTE

1) Rehosting means migrating a system to a new OS or host platform while leaving all applications and data unchanged.

2) Rewriting is a system-migration approach whereby applications are converted into another programming language without modifying their specifications.

Companies increasingly opting to rehost or rewrite legacy systems

Migration away from mainframes has been ramping up in response to termination of mainframe support by major hardware vendors and rising maintenance costs. Insurers, many of which still have core systems running on mainframes, are no exception to this accelerating trend.

To safely and securely migrate from a mainframe under tight time constraints, many insurers are opting to modernize their systems by rehosting[1] or rewriting[2]. A growing number of such projects have run into difficulties due to lack of preparation for or attention to (1) differences in architecture between the pre- and post-migration systems, (2) data migration and linkages with peripheral systems, and/or (3) QA testing of the post-migration system as a whole.

Pitfalls to avoid when rehosting or rewriting

Rehosting and rewriting are generally recognized as low-cost modernization strategies but when the decision to rehost or rewrite is made predominantly from a cost perspective without enough attention to other considerations, the following types of problems are prone to ensue.

<Inadequate preparation for changes in architecture>

When a company switches to a relational database without restructuring its programs, system performance may be impaired by differences in architecture between the pre- and post-migration systems. Additionally, when a code base is converted from COBOL to an object-oriented language like Java, the post-migration system may end up memory-constrained and require additional hardware as a result of COBOL COPY statements being converted into verbose Java object structures.

<Inadequate consideration of data migration and linkages with peripheral systems>

When fixed-length data are migrated from an existing host without being converted to the destination encoding, they will not be able to be processed by regular SQL data operations. Such unconverted data can disrupt system operations and/or interoperability with other systems/data. In a worst-case scenario, the modernization project's scope would have to be expanded to additional systems not included in the original plan.

<Inadequate preparation for system-wide QA testing>

While source-code conversion quality is typically vendor-guaranteed, there is no such third-party assurance for overall system quality. The system owner is responsible for system-wide QA inclusive of any tuning and rebuilding. The system owner consequently may end up incurring testing costs, particularly for tests subsequent to integration testing, on a par with a system rebuild. Conversely, if a project skimps on testing, quality deficiencies may result.

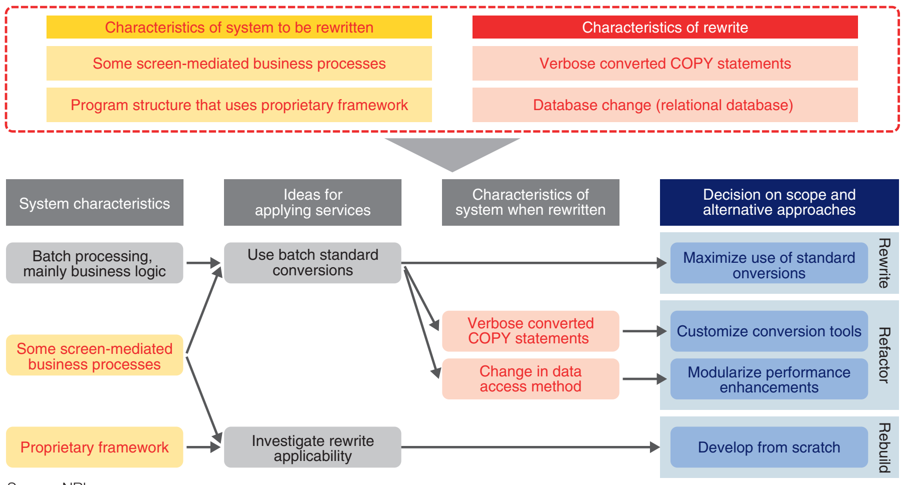
These three inadequacies are resulting in cost and/or time overruns in a growing number of projects.

First step is thorough analysis of legacy system

The key points when opting to rehost or rewrite are to (1) appropriately delineate the scope of the rehosting/rewriting, (2) properly architect the post-migration system as a whole, including peripheral systems, and (3) formulate a post-migration testing plan to ensure overall system quality.

As a prerequisite to performing these three steps, analysis of the existing system is crucial. In the case of rehosting or rewriting, however, the analysis should focus on architectural elements such as data flows and application structure, not business logic. A project plan should then be drawn up based on the analysis. The planning process should include the following steps.

Flow chart of rewrite scope delineation process (example)


Characteristics of system to be rewritten
Some screen-mediated business processes
Program structure that uses proprietary framework
Characteristics of rewrite
Verbose converted COPY statements
Database change (relational database)
System characteristics
Ideas for applying services
Characteristics of system when rewritten
Decision on scope and alternative approaches
Batch processing, mainly business logic
Use batch standard conversions
Maximize use of standard conversions
Rewrite
Some screen-mediated business processes
Verbose converted COPY statements
Change in data access method
Customize conversion tools
Modularize performance enhancements
Refactor
Proprietary framework
Investigate rewrite applicability
Develop from scratch
Rebuild

Source: NRI

(1) Delineation of appropriate scope of rehosting/rewriting

Delineate the project scope by conducting trials in advance to assess in specific terms how application structure would change, but not from the standpoint of the code conversion rate. In terms of data access, implement performance measures premised on the project being a rebuild from the outset and aim to reduce post-migration resource requirements by standardizing conversion of COPY statements (see diagram).

(2) Mapping of overall architecture inclusive of peripheral systems

Formulate implementation policies by mapping the linkages between the pre-migration system and all related systems, identifying data junctures by type and protocol, and taking into account connectivity with external systems (ensure preservation of existing functionality inclusive of external systems).

(3) Testing plan to ensure post-migration system's overall quality

Clarify the existing system's data architecture by mapping data flows using a data flow diagram[3] or other such tool. During integration testing of the post-migration system, identify how the new system compares with its predecessor from an itemized cost-benefit perspective. Key focal points include subsystems' boundary lines (internal interfaces) and interactions with external systems (external interfaces).

3) Data flow diagrams elucidate system functions. They are used mainly as system design documents for upstream processes.

Vendors providing rehosting/rewriting services cannot be counted on to do all of the above well. It is important to build an internal team that includes the legacy system's maintainer(s) to analyze the system's architecture and formulate a project plan appropriately tailored to the system.

about NRI

Founded in 1965, Nomura Research Institute (NRI) is a leading global provider of system solutions and consulting services with annual sales above $4.9 billion. NRI offers clients holistic support of all aspects of operations from back- to front-office, with NRI's research expertise and innovative solutions as well as understanding of operational challenges faced by financial services firms. The clients include broker-dealers, asset managers, banks and insurance providers. NRI has its offices globally including New York, London, Tokyo, Hong Kong and Singapore, and over 17,400 employees.
For more information, visit https://www.nri.com/en

Inquiries to : Financial Market & Digital Business Research Department
Nomura Research Institute, Ltd.
Otemachi Financial City Grand Cube,
1-9-2 Otemachi, Chiyoda-ku, Tokyo 100-0004, Japan
E-mail : kyara@nri.co.jp

https://www.nri.com/en/knowledge/publication/fis/lakyara/