



Enterprise Evolution

The Remediation Process

Enterprise Evolution, a discipline in the legacy modernization domain, is a collection of tool-enabled disciplines that facilitates the understanding, improvement, migration, reuse and/or transformation of existing software systems. This is the second in a 3-part series of white papers that discusses the remediation options in an Enterprise Evolution modernization initiative, which may be applied to a variety of IT project initiatives.

Remediation, as a modernization discipline, is defined as a collective set of tool-enabled disciplines that support or provide for the re-factoring, stabilization and improvement in the packaging and design of existing software systems. Remediation, unlike the concept of architecture transformation, delivers application improvements and upgrades within the context of the existing architecture. ¹

This last aspect of remediation is an important consideration because remediation does not “transform” existing functional, data or technical architectures. Rather, remediation is based on the concept of significantly improving and evolving programs or systems within the context of existing software architectures.

This article discusses remediation benefits, disciplines and options available to organizations seeking to make existing systems more agile and adaptable to changing business requirements.

Remediation Benefits

There are short-term and long-term benefits of remediating a given program or system. Remediation tasks can be incorporated into larger projects or be applied stand alone as a way to evolve applications and increase their functional life. Remediation benefits can be summarized as follows.

- Enable analysts to identify, apply and test ongoing enhancements and related modifications more efficiency and effectively.
- Allow for the deactivation and modularization of systems to facilitate the integration of those systems and related data into a package-based environment.
- Provide rationalized data representations as input to data architecture redesign and migration, thereby ensuring the integrity and quality of the resulting architecture.
- Modularize applications into componentized non-redundant, discreet routines to expedite the deployment of services oriented architectures.
- Isolate and consolidate program logic as a foundation for language, platform or other types of migrations.

Remediation can target any aspect of an application environment including user interfaces, program logic, cross-system data definitions and data usage, system modularity and functional redundancies.

¹ Note that the Object Management Group’s Modernization Task Force uses the terms “Refactoring and Stabilization” to define this same set of remediation disciplines.



Each of these remediation aspects is discussed in the following sections.

User Interface Retooling

While applying new front-ends to older applications and related interfaces is not a new or unique concept, remediation of front-ends may also be viewed as a first in a series of incremental steps in a modernization initiative. This involves working with frontline users to automate and streamline certain processes, create front-end systems that connect to multiple back-end applications and replace old style interfaces with modern interfaces.

Interface retooling projects may also consolidate multiple interfaces and incorporate logic in front-end applications that utilize extracted back-end logic. This approach may be coupled with the modularization and reuse of back-end systems and serves as a driver of subsequent modernization activities while evolving a business process-driven front-end architecture.

Program-Level Remediation

Program-level remediation involves resolving programmatic flaws, structuring poorly modularized source code and applying program-level design improvements. Program flaws represent structural and design anomalies in source code and include program-level recursion, runaway logic, premature procedure exits and syntactically and semantically dead code. These programming anomalies can stymie projects ranging from simple enhancements to rule extractions. Remediation tools and techniques facilitate the identification and resolution of these problematic constructs.

Structuring source code creates a series of modular, single entry / single exit procedures that can be invoked and relied upon to return control to the point of invocation. A structured module creates source code that can be changed more quickly and adapted to business requirements more reliably and more efficiently. Structuring also creates a modularized baseline that more readily supports various migration and transformation projects.

Program-level design review and improvement resolves poor programming techniques such as the non-use of tables or arrays, reliance on spurious switches, poorly constructed decision logic and other issues. Remediation through this stage is recommended for any programs or systems that will continue to serve as viable software assets over the long-term. Design improvements also ensure that problematic designs are not migrated into target architectures.

Data Rationalization & Standardization

Data definitions, which cross program and system boundaries, can be a limiting factor in applying enhancements to existing systems as well as in the migration and transformation of application and data architectures. Consider a single physical file structure defined by hundreds of related physical data layouts that contain discrepancies across data definitions, attributes and naming conventions.

No analyst or project team can reliably depend on a given view to fully represent that data structure. This problem may be replicated across hundreds of data structures and systems, creating confusion in projects relying on or impacting those structures. Efforts to understand, change or reuse these data structures, including file update, field expansion, data migration, platform migration and rule extraction projects, are put at risk from widespread redundancies and inconsistencies.



Data definition rationalization and standardization projects involve tool-enabled cross-systems analysis, business definition mapping and the reconciliation and re-factoring of layouts, names and attributes. Data definition remediation activities are analyst-assisted, tool-enabled processes that involve subject matter review and input as well as fine tuning based on the particular scenarios or initiatives that an organization wishes to pursue.

Modularization

Modularizing or componentizing existing systems creates more adaptable, reusable applications. Modularization is the process of selectively slicing, isolating, consolidating or otherwise realigning application functionality across a system. Key objectives include streamlining module size; creating reusable routines; eliminating inconsistencies across modules; isolating business, user access and data access logic; and reducing systems to a subset of essential functionality.

Creating modular, reusable components from existing systems involves various approaches based on the intended end-state results. At the most simplistic level of modularization is the process of slicing large, unwieldy programs into more streamlined programs. More sophisticated options involve isolating and consolidating business, user access and data access logic, which prepares systems for data, platform and language migrations.

Reconciling and consolidating redundancy also creates a baseline for rule extraction and architecture transformation. Consider that when inconsistent, redundant functionality infects target architectures, whether those architectures involve language-based services or functional models, those architectures will be corrupted along with the business areas they support.

Mapping Remediation Tasks to Project Initiatives

Remediation work is typically driven by one or more scenarios. These may be general improvement projects geared at streamlining ongoing maintenance or enhancements or be tied to a given migration or transformation initiative. Certain project initiatives can be mapped to remediation activities as shown below:

- Ongoing Enhancements: Elimination of program flaws, structuring, program-level design improvement and selected rationalization of data definitions.
- Data Enhancement / Migration: Data definition rationalization and standardization, selected modularization and program re-factoring as needed.
- Platform Migration: Selective program re-factoring, modularization of I/O logic, selected data usage rationalization and user interface retooling.
- Transformation to Target Architecture: Selected data definition rationalization, consolidation and isolation of business, user access and data access logic, user interface retooling and program-level re-factoring based on assessment findings.

Note that the above bullets refer only to remediation tasks and do not represent the entire set of tasks for a given project. Phased deployment of individual steps in various remediation roadmaps ensures that ROI can be accrued on an incremental basis. This approach also allows application support teams to apply ongoing maintenance work to systems without freezing software changes for long periods of time.



In addition, depending on the type of remediation work planned, project teams will need to complete certain assessment activities. For example, a data enhancement project would require data definition analysis as a prerequisite to any data definition remediation work. Similarly, program remediation or modularization requires process flow analysis.

The bottom line is that application remediation activities can benefit a wide range of projects. Evolving applications through continuous improvement allows application project teams to deliver more streamlined and robust enhancements as well as other project-driven changes to systems. Remediation also plays a key role in longer term initiatives including language and platform migration, data architecture migration or transformation to various target architectures. While the first two white papers in this series discussed the assessment and remediation of existing systems, this paper discusses the third leg of modernization – transformation. While remediation can be thought of as “modernizing systems in place”, the concept of transformation goes further by transforming existing data and application architectures into target architectures and environments.

Visit ecubesystems.com to download **Part 1 (Assessment)** and **Part 3 (Modernization)** of this paper

William Ulrich is President of Tactical Strategy Group, Inc. and a strategic IT planning consultant. He is Co-Chair of the OMG Architecture-Driven Modernization Task Force and his latest book is entitled “Legacy Systems: Transformation Strategies”.