***10 Jan. 2006***

# Architecture-Driven Modernization Scenarios

## Abstract

*This white paper discusses various modernization scenarios that can and are being applied to existing computer software systems. The specific focus of this paper is to summarize these scenarios and discuss how each scenario leverages the computing standards being developed by the OMG, Architecture-Driven Modernization Task Force.*

## Introduction

The Object Management Group (OMG) Architecture-Driven Modernization (ADM) Task Force (ADMTF) is creating a set of standards to facilitate the interoperability of modernization tools. These interoperability standards are being established in a series of meta-models that facilitate the collection, analysis, refactoring and transformation of existing systems. Under these standards, modernization vendors, each of which may support different aspects of the modernization process, can seamlessly interchange application meta-data across platforms, languages and stages of the modernization process.

The ADMTF standards that are either in place or under development are discussed in the ADMTF Roadmap White Paper[1]. In order to provide a context and purpose for creating and deploying each of these ADM standards, the ADMTF determined the need to identify a set of scenarios under which one or some combination of these standards might be applied. The ADMTF Roadmap includes the following standards.

- *Knowledge Discovery Meta-Model (KDM) Package*
  KDM provides a foundation for all other ADM standards by establishing a meta-model that represents a cross-section of system artifacts and relationships among those artifacts.
- *Abstract Syntax Tree Meta-Model (ASTM) Package*
  ASTM complements KDM by fully representing system data usage and related statements and actions to the lowest level possible across a variety of environments and languages.
- *Analysis Package*
  The Analysis Package extends the structural representations in the KDM and ASTM by representing additional levels of analytical or behavioral representations of systems in certain meta-models.
- *Metrics Package*
  The Metrics Package represents metric information within meta-models about various aspects of existing systems.
- *Visualization Package*
  The Visualization Package provides meta-model(s) to streamline the ability to create various visualizations of existing system structures, data usage and behavior.

[1] Architecture-Driven Modernization Roadmap, http://adm.omg.org/ADMTF%20Roadmap.pdf, 2004

- *Refactoring Package*
  The Refactoring Package provides for meta-model based refactoring to enable phased restructuring, rationalization and other improvements to systems.
- *Target Mapping & Transformation Package*
  The Target Mapping Transformation (Transformation) Package enables meta-model based transformations of existing systems to target architectures, such as MDA.

**Why Modernization Scenarios are Important**

A modernization scenario can be defined as an initiative, such as the ongoing task of portfolio management, a project, such as migrating from one platform to another, or a series of project phases (i.e., a super project) as would be the case when consolidating, redesigning and redeploying an application in a model-driven architecture. This white paper discusses the scenario concept and introduces a set of basic modernization scenarios.

The use of scenarios in discussing modernization has several benefits. First, it is difficult for someone unfamiliar with modernization to envision all of its potential applications. For example, it may not be readily apparent that modernization can help leverage the refactoring of several applications to facilitate migration to services oriented architecture (SOA). An SOA migration scenario helps a user determine the tasks, tools and use of modernization and related standards within an SOA project.

Second, scenarios provide templates for crafting project objectives, plans and related deliverables. This includes defining tasks, the role of ADM packages and personnel requirements. Task identification is required because modernization encompasses many types of initiatives involving existing systems and the entire universe of analysis, refactoring and transformation tasks is rarely applied within a given scenario.

For example, a language-to-language conversion or a platform migration would not involve the redesign of application data structures. Similarly, a data architecture migration effort, in which a flat file or hierarchical database is redesigned and redeployed as a relational database, would not employ tasks needed to convert from one language to another. Scenarios, therefore, define tasks necessary to complete a given modernization initiative and omits unnecessary tasks that would not apply to such a scenario.

Finally, carving out specific tasks that apply uniquely to a given scenario allows a user to pinpoint the types of tools necessary to perform these tasks. For example, a language-to-language conversion would not require a data redesign tool. Similarly, a platform migration does not require a business rule extraction tool. Therefore, task identification, based on a given scenario, is essential to identifying the necessary tools for a given project.

**Modernization Scenarios & ADM Standards**

Knowing which modernization tasks and tools are required for a given scenario helps identify the specific role of certain ADM packages within that scenario. It follows that identifying the universe of modernization scenarios and tasks provides a guide as to the role of ADM packages within modernization in general. The remainder of this white paper outlines these scenarios and their relationship with ADM packages.

Each of the following sections outlines a modernization scenario, why an organization would pursue such a scenario and some of the ways that ADM packages apply to such a scenario. The points highlighting the role of modernization within a given scenario are not meant to be exhaustive but merely exemplify how modernization packages facilitate a given scenario.

These scenarios address various systems analysis, refactoring, consolidation, migration and redesign initiatives that an organization may pursue. Many of theses scenarios can be mixed and matched based on an organization's goals. For example, a language-to-language conversion may be coupled with a platform migration. Other scenarios may be added to this list from time to time.

## I. Application Portfolio Management

Organizations must manage application systems as business assets and this requires portfolio analysis and management. Whether driven by internal audit requirements or government regulations, accounting of information assets is essential. This scenario captures and exposes technical and functional meta-data on an organization's systems. Many old systems are poorly documented and this scenario addresses this shortcoming. The need for such a scenario is characterized as follows.

- There is no documentation depicting information flows, system-wide data usage or overall systems architecture.
- Management requires detailed accounting of systems to fulfill audit or regulatory requirements (e.g., Sarbanes-Oxley, Basil Accord).
- Systems will undergo major changes or are to be outsourced.
- Modernization projects require baseline information to segment a portfolio into functional work units and provide input to an overall strategy.

This scenario entails running systems analysis tools and augmenting the resulting meta-data with analyst expertise to create a reliable knowledge base on existing systems. Meta-data would be stored and exchanged using the KDM, which is central to the effective and ongoing ability to manage information systems as organizational assets. Specifically, ADM standards facilitate the following portfolio management activities.

- Provide a repository of cross-systems and cross-platform application meta-data.
- Enable analysts to visualize the relationship among business processes and related business artifacts with application artifacts.
- Enable analysts to incorporate evolving application and related meta-data as it changes over time.
- Allow analysts to expose systems meta-data through visualization tools.
- Support distributed or centralized meta-data viewing, including the ability to scale up or down based on organizational size, structure or reporting requirements.

- Support the federation of systems meta-data, based on organizational structure, size or reporting requirements.

## II. Application Improvement

The application improvement scenario is a “super scenario” comprised of several sub-scenarios. The goal of this scenario is to improve the robustness, integrity, quality, consistency and/or performance of applications. Activities include the correction of program or system flaws (e.g., recursion), source code restructuring, data definition and source code rationalization, field size standardization or other system refactoring tasks. This scenario involves no architecture transformation tasks and is based on the following requirements.

- A growing upgrade request backlog cannot be met due to system quality.
- The system is experiencing high failure rates or reliability problems.
- There is a long learning curve, poor IT responsiveness and user dissatisfaction.
- System upgrade costs are not proportionate to business returns.
- One or more systems are being prepared to be outsourced or are being brought back in-house.
- Management has given up even trying to change the applications.
- Field expansion is required based on specific business requirements (e.g., uniform bar code, phone number or revenue growth).

The portfolio analysis and asset management scenario may precede this scenario, but this is not essential. ADM standards support this scenario as follows.

- Store and associate systems and business meta-data associated with the systems of interest for the purposes of planning, management and execution.
- Expose system and program weaknesses discovered by system analysis tools across multiple environments and languages.
- Provide planning input to improvement tasks by linking system weaknesses with business needs, particularly across systems, platforms and languages.
- Assist analysts with the process of rationalizing system-wide data names and source code, attributes, records, segments and tables.
- Assist with the change management throughout the improvement process.
- Facilitate the tracking of system meta-data as it changes through a systems refactoring initiative.
- Streamline the planning and execution of the validation and verification stage of the project.

## III. Language-to-Language Conversion

This scenario involves converting one or more information systems from one language to another language. The language-to-language conversion scenario addresses the physical need to move from one language to another. This may be driven by a variety of factors, but does not involve a redesign of the application functionality beyond that which is required by the language change itself. This scenario is driven by the following needs.

- A language has become obsolete, is no longer vendor supported, is no longer understood by available programming talent or is just too hard to change.
- There is a business requirement to enhance the functionality of the current system but the language is no longer supported or readily adapted to change.
- Systems must move to a new platform and the new platform does not run the existing language or particular version of that language.
- A baseline system must be established from which current systems may be migrated to a strategic architecture

ADM standards play the following role in the language-to-language conversion scenario.

- Tracks system artifacts through the planning, phasing and staging of a conversion effort.
- Facilitates discovery of high risk issues such as the use of runtime libraries or language constructs not available in the target environment.
- Assists with the change management process as the conversion proceeds.
- Streamline the planning and execution of the validation and verification stage of the project.
- Allows multiple languages to be represented in a common meta-model and for that meta-data to be converted to a common language.

## IV. Platform Migration

Moving systems from one platform to another is driven by platform obsolescence or the desire to standardize applications to an organizational standard. This scenario does not involve any functional or data redesign beyond that which is essential to the platform migration. This scenario may also be combined with a language-to-language conversion, although language conversion is not always required (e.g., UNIX to LINUX). The following situations typically drive a platform migration.

- The hardware and/or operating system is no longer supported or viable.
- Management has decided to “right size” a system by moving it to a distributed environment or back to a mainframe.
- The current platform does not support the accepted operating system standard.
- Management has mandated a platform change.

ADM standards play the following role in a platform migration project.

- Supports tracking of system artifacts through the planning, phasing and staging of the migration, particularly if it has to be phased in over time.
- Facilitates discovery of high risk issues such as the use of runtime libraries or language constructs not available in the target environment.
- Assists with the change management process as the migration proceeds.
- Streamlines the planning and execution of the validation and verification stage of the migration.
- Allows multiple tools to be applied across the various stages of migration.

## V. Non-Invasive Application Integration

Organizations with an immediate need to bring a graphically oriented look and feel to end users can utilize middleware technology to replace existing front-ends with Web-based front-ends. While a non-invasive integration project qualifies as a modernization scenario only at a user interface level, this scenario is still supported by the ADM standards. The integration scenario is characterized by the following requirements.

- Business users want to replace aging front-ends with Web-based front-ends.
- Users gain value from replacing older front-ends while leaving core system functionality, data structures and interfaces essentially unchanged.
- A front-end integration project is seen as an interim stepping stone to subsequent modernization objectives such as an SOA migration.

In spite of the fact that this is a non-invasive approach (i.e., does not change an underlying system) to providing new user front-ends to business users, ADM standards play a role in the planning, execution and post-project documentation of new user front-ends. In general, KDM aids in the discovery and adaptation of existing user interfaces. KDM also supports the post-implementation phase because middleware environments are becoming so complex. Because larger organizations are losing track of which interfaces and systems are connected to other systems, documentation is a key aspect of integration deployment. The following points exemplify the role of ADM standards in a non-invasive integration scenario.

- Aid in determining how to create a new interface to existing business logic because it has high-level and granular information about application data and processing logic.
- Allow analysts to pinpoint all user interface candidates targeted for migration to Web-based front-ends.
- Help identify existing front-ends that may be redundant with other front-ends across application environments.
- Highlight front-end consolidation opportunities for redundant back-ends.
- Facilitate the tracking of distributed user interfaces and middleware on an ongoing basis as new front-ends are deployed.
- Ensure alignment with the OMG Enterprise Application Integration (EAI) standard.

## VI. Services Oriented Architecture (SOA) Transformation

The transformation to services oriented architecture involves more than just attaching new front-ends to existing system user interfaces as some may mistakenly believe. Because most existing application functionality is embedded in monolithic, functionally and architecturally dated systems, application and data architectures cannot be segregated into services in any useful or meaningful way. In addition, business logic is typically intertwined with user interface and data access logic. In these situations, a true SOA cannot be created without retroactively applying modular design principles to existing, back-end systems. The SOA scenario is applicable in the following situations.

- Business functions embedded in monolithic batch or online applications need to be accessed in a modular, services oriented fashion.
- System functionality is locked into backend batch processing systems.
- Complex user interface and data access logic complicates the isolation of business logic that may be deemed a service.
- Online applications do not update data in real time, which results in a service relying on back-end batch update systems.
- Existing front-ends rely on segmented, inconsistent and redundant functionality in back-end systems, which is not conducive to forming well-bounded services.

ADM standards play a key role in an SOA scenario by helping identify and track relationships between the physical system, program functionality, data usage and user interfaces. Such a project requires the componentization of existing applications to facilitate the reuse of business logic contained within them. Based on this requirement, ADM standards help as follows.

- Facilitates the front-end planning necessary to identify redundant, inconsistent and segregated functionality that needs to be refactored to create services.
- Identifies the front-ends of interest that could serve as prototypes for creating a service that hooks into an existing system.
- Allows analysts to determine the need to consolidate and reconcile redundant and inconsistent user interfaces and program functionality into reusable services.
- Aids in discovery and extraction of the business logic as service candidates.
- Helps with the discovery and adaptation of new, component level interfaces.
- Streamlines efforts to track consolidated user interfaces and program functions throughout the service creation and deployment process.
- Facilitates the tracking of new user interfaces into back-end applications as a way to document post-project SOA architecture.
- Allows for necessary refactoring across multiple platforms and languages.
- Simplifies the planning and execution of the validation and verification stage of the project.

## VII. Data Architecture Migration

A data architecture migration moves one or more data structures from a non-relational file or database to relational data architecture. Many times this is done using a "quick and dirty" approach, leaving users with performance, reliability and data accessibility problems. Pitfalls include ignoring business requirements, sidestepping relational design techniques, not incorporating related or redundant data in the project, not utilizing qualified data analysts or treating the project as a straight conversion effort. This scenario shuns the quick and dirty approach. Typical requirements are as follows.

- Users are experiencing data consistency, accessibility, redundancy and integrity problems.
- The business is experiencing an inability to get at the same types of data defined

or calculated differently across different systems.
- Existing flat file, hierarchical or networked structures are not readily accessible to distributed or new technologies.
- Users require more flexible views of business data.
- Older file or database structures are obsolete and being eliminated.

ADM standards facilitate data architecture migration because the various meta-models can assist with tracking and refactoring artifacts impacted by such a project including program-based data definitions, data access logic, database definition language and the data itself across multiple platforms. ADM standards assist as follows.

- Enable analysts to define the scope of the project based on the artifacts impacted by existing data structures.
- Allow analysts to identify all relevant and impacted artifacts based on the nature and scope of the data migration effort.
- Help determine if a common basis exists, within underlying computation and data models, to identify changes that would allow different systems to produce compatible answers for the same data.
- Facilitate the isolation, consolidation and reconciliation of data access logic as a basic step in simplifying programmatic access to the newly redesigned database.
- Streamline the planning and execution of the validation and verification stage of the project.
- Support these tasks across platforms and languages.

## VIII. System & Data Architecture Consolidation

Many organizations have multiple systems that perform the same basic functions. For example, a merger or an acquisition may have resulted in three separate billing systems. System cloning also contributes to these redundancies. Redundant systems and data structures contribute to usage inconsistencies, redundant business processes, customer frustration, integration problems and excessive maintenance workloads. Another factor driving this scenario is the need to construct a single application from multiple stand alone systems. For example, analysts may want to create a single human resources system from separate payroll, insurance and pension systems. The following factors drive such a scenario.

- An organization has recently undergone a merger or acquisition or has not yet fully streamlined applications from a past merger or acquisition.
- Management wishes to consolidate redundant business areas into a single functional unit.
- High business and IT costs require the consolidation of redundant business processes and related systems.
- Currently running several redundant systems that essentially perform the same or similar functions.
- Several similar systems contain large segments of overlapping functionality.
- Related, stand alone systems process similar data redundantly and inconsistently.

- Inadequate or nonexistent sharing of data between systems is severely limiting user service levels.
- Downstream systems have evolved to handle much of the functionality that should be defined in core systems.
- Business users get different answers to the same questions from different systems.

Systems and data architecture consolidation scenario is far reaching because it involves major retooling multiple applications. This scenario does not involve model-driven transformation, language change or platform migration. It can, however, be combined with these scenarios. Care must be taken to make a business case for this scenario. This may not be difficult when millions of dollars are spent on redundant business units that cannot be combined due to information systems redundancies. ADM standards support this scenario as follows.

- Help pinpoint which applications and data structures are within the project scope.
- Assist in determining the business process, user interface, application logic, data definition and related redundancies as candidates for consolidation.
- Enable the tracking of application artifacts as consolidation efforts proceed and are phased into production.
- Streamline the planning and execution of the validation and verification stage of the project.
- Support these consolidation activities across disparate environments.

## IX. Data Warehouse Deployment

This scenario builds a data warehouse of business data and creates ways to access this data. The warehouse contains data that has been extracted, analyzed and transformed into a common repository that users can access, but not update, as required. This is common when organizations have the following requirements.

- Business functions require consolidated access to certain data (e.g., customer information) to streamline user tasks.
- Users require access to data that crosses organizational and application boundaries.
- Related data is defined across multiple systems, making it difficult to access user summary information.
- There is not enough time or budget to integrate or modify core applications to address these data requirements.

As discussed in the data architecture migration scenario, KDM facilitates data analysis and capture from existing systems. Further, the KDM is aligned with the OMG Common Warehouse Model (CWM), which is a standard for representing data from disparate sources for the purposes of building and maintaining a data warehouse. ADM standards help facilitate this scenario as follows.

- Allow analysts to identify relevant data and related data definitions that need to be analyzed, reconciled, validated and loaded into the warehouse.

- Facilitate the detailed analysis needed to identify data definition discrepancies across systems or business units.
- Facilitate the tracking of data models and the physical data from which these models are derived.
- Allow analysts to continue to track multiple data sources on an ongoing basis as the data warehouse is used.
- Enable these activities across multiple platforms and environments.

## X. Application Package Selection & Deployment

This scenario defines how to analyze, select and deploy third party application packages. If management is weighing one or more packages against in-house options, this scenario assists with comparing functional requirements. It assists with the deployment of the package by helping analysts determine which portions of the package need to be implemented, integrated, discarded or updated. It also outlines how existing systems are to be retired, integrated or retooled to work with a package. The following requirements drive the application package scenario.

- A decision has been made to investigate a third party application software package options.
- An application package has already been acquired and needs to be implemented.
- Documentation of the package is required.
- A roadmap is needed to determine how a package can interact, interface or integrate with existing systems or packages for implementation purposes.

Many times, package selection does not consider whether a package meets functional requirements. This scenario, however, ensures that a functional analysis is performed on candidate packages and that an objective approach is used to select the appropriate package. It also creates a deployment roadmap. ADM standards play a role in both of these efforts by mapping requirements to candidate packages and to in-house applications. This mapping process may even lead analysts to determine that modernizing in-house applications may be a better approach than employing a package option. This is summarized below.

- Assist with mapping strategic requirements to the data and functional capabilities of various application packages.
- Allow analysts to compare the functionality of various packages with the functionality running in the existing application environment.
- Upon package selection, determine which portions of the package would be deployed and integrated into the current environment.
- Determine which portions of the existing application environment need to be retained, retired or integrated into the package environment.
- Provide a data mapping, integration and migration plan based on current data structures and those used by the package.
- Offer insights into functions that need to be added to the newly deployed application based on functions the package does not perform.
- Streamline the planning and execution of the acceptance testing process.

- Support these tasks across a variety of environments.

## XI. Reusable Software Assets / Component Reuse

Reuse is one of the critical ways in which an IT organization can improve productivity and ensure consistency of business functionality across an enterprise. Consider the massive redundancies hidden in application and data structures across software portfolios and the savings become apparent. Organizations could save significant time and money by leveraging previously implemented functionality in new development projects requiring. Further, businesses that are now running duplicate customer management, payment, claims, inventory and other systems have a wealth of untapped information building blocks. Modernization helps identify, capture, streamline and prepare information assets for reuse. This scenario applies in the following situations.

- An organization understands and has bought into the value of reuse and component-based development.
- There is a significant installed base of application systems that contain functionality that IT wishes to turn into reusable assets or components.

ADM standards support the reuse scenario as follows.

- Facilitate the identification of certain software assets based on data utilization, transaction access or other criteria.
- Identify related or duplicate functionality based on common data usage, structural considerations or other cross-reference criteria.
- Assist with tracking reusable assets as they are consolidated and populated into reuse libraries across environments.

## XII. Model Driven Architecture (MDA) Transformation

Transforming a non-model-driven environment to a model-driven environment requires a series of phased tasks over a window of time. Moving to MDA, given that rewriting all of the existing applications is not an option in most cases, requires transformation of existing, hand-crafted applications into models that can be used to generate replacement applications. The following are common requirements driving this concept.

- An organization has adopted and is committed to an IT environment in which applications are built and maintained in models.
- MDA concepts and tools are accepted within the organization.
- The current data and application architecture dated or obsolete.
- The business has changed to the degree that existing systems and data structures no longer support the organization in their current form.
- Users require functional upgrades that are difficult to add to the existing architecture.

ADM standards can assist with the phased transition to such an environment as follows.

- New business requirements and processes can be defined within the KDM as a

vehicle for defining target requirements.
- Current data and application artifacts can be mapped to these new requirements in the KDM to assist with defining a transformation plan.
- Transformation projects can be defined across enterprise data and applications of interest based on KDM mappings.
- As transformation proceeds, KDM serves as the vehicle for mapping current-to-target functionality, along with existing artifacts, for transformation purposes.
- Streamline the planning and execution testing process.
- Post-transformation documentation ensures that emerging MDA models and system artifacts are documented during and after the transformation process.
- Transformation tools can be applied across a variety of platforms, languages and project phases.

## XIII. Software Assurance

This justifiable trustworthiness against established business and security objectives is called Software Assurance. As complications to assessing trustworthiness continue to evolve, formalized and standardized mechanisms and approaches must be developed that increase Software Assurance. The main benefits to be achieved as a part of Software Assurance efforts include:

- Determine that software is secure and reliable in its environment for the user's intended use.
- Ensure that users and consumers are using secure software assets and systems.
- Facilitate the development of tools to help build more secure and reliable software.
- Provide for better trained and educated software developers that utilize community-approved processes and tools to produce secure software.

ADM standards benefit Software Assurance as follows.

- Facilitate ability to verify and authenticate software attributes across a variety of platforms and languages.
- Allow for the visualization of software attributes.
- Provide for the metrical analysis of systems based on Software Assurance requirements.
- Enable the behavioral analysis of systems and the ability of multiple tools to share and compare this analytical and behavioral information.

## Summary

In summary, the above scenarios create a baseline for visualizing how ADM standards can help facilitate various types of modernization scenarios. It is also meant to guide the developers of ADM packages to ensure that those packages address industry requirements. As stated earlier, this is not an exhaustive list and the bullet points contained herein do not fully explain the myriad processes underlying such projects. The ADM Task Force will continue to expand upon these concepts.