Architecture-Driven Modernization 101: Concepts, Strategies & Justification

William M. Ulrich –
Tactical Strategy Group, Inc.
www.systemtransformation.com
Seminar Objectives

- Define existing data & application architecture challenges
- Introduce modernization concepts
- Discuss business-driven modernization
- Outline techniques for performing application assessments, integration and transformation
- Describe practical application of these concepts through project-based scenarios
- Introduce enabling tools and technologies
- Help attendees get started
Seminar Outline

I. What is Architecture Driven Modernization?
II. Existing Data & Application Architecture Challenges
III. Attempts to meet BPM Challenge
IV. Modernization Overview
V. Modernization Processes, Techniques & Approaches
VI. Modernization Tools
VII. Modernization Scenarios
VIII. Getting Started
IX. Selling Management & RDI
X. Close
I. What is Architecture-Driven Modernization (ADM)?

Process of understanding and evolving existing software assets for the purpose of

- Software improvement
- Modifications
- Interoperability
- Refactoring
- Restructuring
- Reuse
- Porting
- Migration
- Translation into another language
- Enterprise application integration
- Service-oriented architecture

Modernization starts where existing practices fail to deliver against business objectives

*Source: ADM.OMG.ORG*
II. Existing Data & Application Architecture Challenges

- Existing data & application architectures run in sharp contrast to modern IT disciplines.
- The challenge is how to leverage emerging architectures while adjusting to critical business requirements & changing processes.
What are the major reasons for keeping the legacy systems your company is currently planning to retain?

- Still able to support business processes 54.4%
- Still reliable 49.7
- Staff available to support them 44.3
- Still more cost effective than alternatives 41.6
- We do not have the budget to make any changes 36.9
- Still able to support strategic goals 36.9

Still relevant and reliable

* CIO Insight – December 2002
CIO Legacy Survey

If you are currently migrating or planning to migrate off any of your current legacy systems in the next year, what are the main reasons?

- Meet new strategic goals 65.2%
- Legacy systems can’t support business processes 59.9
- Legacy systems can’t support current strategic goals 56.1
- New systems would be more cost effective 48.5
- Legacy systems provide poor interoperability 41.7

Migration / transformation plans

* CIO Insight – December 2002
Existing Data & Application Architectures

- Numerous languages and platforms
- 200 billion lines of Cobol (60% of total)
- Haphazard design and stovepipe architectures
- Segregated functionality and data structures
- Layers of middleware and data warehouses
- Web-based interfaces
Business processes, applications and data mirror stovepipe information governance structures.
Real World Symptoms of the Legacy Challenge

- Major telecommunications company cancels package implementation effort - 360 person years / no deliverable

- Insurance company cancels 12 year project having spent $199 million - no results / no deliverables

- $10 million banking application development project is canceled - no deliverables

- 3 year old, C-based client/server system must undergo major re-engineering effort - no cohesive design

- Countless other stories
Bottom Line: Aging Architectures Impede Ability to Deliver Business Value

- 85% of IT projects: late or never delivered*
- Only 9% of IS projects come in on time or within budget*
- ERP projects: years to implement, canceled 35% of the time & rarely fully deployed*
- $16.5 billion is spent annually on systems that users never see (Information Week)
- Management believes “quick & easy” solutions can solve complex IT challenges

* Standish Group International
IT & Business Challenges Drive Modernization Requirements

- Time-to-market
- Mergers
- New products & services
- CRM
- Business intelligence
- Process redesign

- The Internet
- Collaborative tools
- Java, XML, HTML
- New platforms
- Integration tools
- Open systems

Changing Business Models
Emerging Information Technologies
Changing Industry Models
New IT Architectures

- E-Business
- Deregulation
- Globalization
- Loss of intermediaries
- Supply chain integration
- Economic weakness

- Model Driven Architecture
- J2EE and .Net
- Component reuse
- Web Services
- Integration architecture
- Agile methods
In a Nutshell…

Current Business Requirements

- Reducing time-to-market for new products and services
- Shifting to a customer-driven philosophy
- Streamlining transaction flow across supply and distribution chains
- Creating flexible information systems to achieve business agility
- Adapting systems to changing business processes
III. Attempts to Meet BPM Challenge

- New tools & technologies
- New development techniques
- But, have these advancements helped?
IT Attempting to Meet Challenges Using Traditional Approaches

- Wholesale Application Replacement
- Non-Invasive Data & Application Integration
- Existing Data & Applications
- Application Packages
A conceptual framework for non-invasive integration allows project teams to incorporate integration with modernization options.
But Integration Technology is One More Layer of Legacy Architecture

- **Message–Oriented Middleware**: Basic middleware that allows an application to send / receive messages to / from other applications.

- **Data Management Middleware**: Facilitates access to data not natively defined to an application.

- **Application Server**: Multipurpose, Web-enabled software that facilitates application-to-application communication across platforms.

- **Adapters**: Custom pieces of code that enable middleware-based communication.
Vendor-driven integration solutions create layers of legacy chaos across information architectures.
But, These Approaches Have Fallen Short

- Wholesale application replacement:
  - Too costly, too time consuming and too risky
- Non-invasive integration:
  - Cannot address underlying architecture limitations
- Package applications* – no panacea:
  - 35% of projects canceled outright
  - Typically deploy only 40% of promised functionality
  - Only 10% of projects are on time and on budget
- Outsourcing: Only moves problem around

*Who Is to Blame for ERP Failure?, Barry Calogero, serverworldmagazine.com, June 2001
Yet Integration is an Issue

Issues reported by IT managers as their main concerns

- Integration with legacies
- Integration with packages
- Skills availability
- Training
- Web development
- Technology standards
- New/emerging technologies
- Outsourcing
- Java adoption
- Platform selection

Source: Gartner Research
Because Today’s Business Processes Cut Across Stovepipe Architectures

Order Entry & Sales  Inventory Control  Shipping & Tracking  Billing  Accounts Receivable

Customer Order Functions

IT must deal with this!
Bottom Line is That...

- Replacement or packages often fail
- No one understands the old systems
- IT is afraid to shut them off
- Time to look at modernization
IV. Modernization Overview

- Bridging the gap
- Standards based
- Modernization infrastructure
These four stages of Architecture Evolution.
Modernization: Bridges Gap Between Existing & Emerging Architectures

Business-Driven Requirements

Architecture Modernization

Existing Architectures

Strategic Architectures
Modernization Alternative – Takes Over Where Existing IT Strategies Fall Short

- Augments or displaces existing IT options
- Enables upgrade, conversion, consolidation, migration and related projects
- Applies a measured, phased approach to meeting IT challenges
- Utilizes scenarios to ensure a business-driven approach
- Builds upon solid foundation of production systems
Modernization Comprised of Three Sets of Tool-Enabled Disciplines

- **Assessment**
  - Analyzes / exposes application artifacts, data and process flows, architecture and behavior

- **Stabilization & Refactoring**
  - Structure, rationalize, realign, modularize and otherwise refactor existing systems

- **Transformation**
  - Extracts logical data definitions and business rules
  - Facilitates reuse of existing extracted artifacts in redesign of target architectures
Modernization is Based on a Set of Emerging OMG Standards

- Object Management Group Task Force* on Architecture-Driven Modernization formed in 2003
- Efforts focused on creating a standard Knowledge Discovery Meta-model (KDM) to facilitate meta-data exchange among vendors
- Secondary goal is to bring vendors into compliance through common processes & definitions
- Roadmap outlines seven stages of standards development*
- Major industry players are participating

*ADM.OMG.ORG
Multi-dimensional integration requires managing business unit, IT and external integration within the transformation repository.
Modernization Infrastructure Requirements

- Recognition of legacy value
- Process for assessing existing architectures and capturing, reusing and migrating existing software artifacts
- Process for analyzing, designing, building and deploying target architectures
- Software that facilitates and integrates modernization and development processes
First, Recognize that Old Systems Contain Business Value

- Old systems do many of the right things (i.e. rules), but not how users want them done or in the time-frames required (i.e. architecture)
- Newly deployed systems and packages omit valuable business rules
- ERP and EAI have not fully addressed these issues
What Organizations Require to Enable Modernization

- Philosophical shift that recognizes:
  - Value of existing systems to a business
  - How phased transformation is more prudent than an all-or-nothing replacement approach
- ROI-driven commitment to incorporate transformation into all system projects
- Modernization support infrastructure that can be deployed across an enterprise
Shift from “From Scratch” Development Philosophy to Phased Reuse

- Replace “throwaway” philosophy with “reuse” philosophy
- Shift from an “all or nothing / go for broke” approach to a phased deployment approach
- Seek lower risks, higher returns and faster delivery through phased delivery strategy
Existing Systems Reuse Should be Built into Application Re-Design Process

Business Process Model

Design Models

Information Architecture

Business models represent as-is & reengineered business processes

Design models depict business processes to be automated

Existing systems: input to redesign process & source of reusable data & business rules

Mapping existing architecture to requirements and reusing existing software artifacts and behaviors in new development projects.
Sample Modernization Methodology

Business Driven Scenarios

Enterprise Redevelopment Planning → Inventory/Analysis → Positioning → Transformation

Systems Transformation Tools

Source: COMSYS / USRM
The Value of a Modernization Framework

- Based on results oriented case studies and success stories
- Each stage contains multiple tasks and each task contains multiple steps
- Supports phased implementation strategy

Source: COMSYS / USRM
Measurement & Metrics

- Metrics support analysis and planning of modernization projects
- Range from simple counts to more complex scores
- **EXAMPLES:**
  - Total # batch jobs in system
  - # Data Elements in current system in common with data elements in ERP system
  - Function points, McCabe, Halstead, etc.
  - Total integration points
ADM Assessment Approach: Decomposition
Scheme Can be Represented in Repository

System contains Application
Application contains Module
Module contains Program
Program uses Interface
Interface is used by Program

System contains Application
Application contains Module
Module contains Program
Program contains Business Rule
Business Rule is defined in Program
Business Process uses Interface
Interface is used by Business Process

Business Unit performs Function
Function is performed by Business Unit

Business Unit
Function
Bus. Rule Query
Bus. Rule
Business Rule
Business Process
Project-level repository facilitates tracking of business data and rules back to physical system artifacts and target requirements.
Example of Repository’s Role in Modernization Scenario

Used for analysis, planning, redundancy identification, consolidation mapping & post-project documentation

Existing Systems

- Payroll
  - 1
  - 3
  - 2
  - 4

- Pension
  - 2
  - 1
  - 3
  - 1

- Insurance
  - 2
  - 3
  - 2
  - 1

Modernization tools

Analyst Interface

Target Systems

- Integrated database
- Integrated application
- Distributed interface

Example of Repository’s Role in Modernization Scenario
Modernization must Address Complex Dimensions of Architecture Challenge

Integration solutions must assess fragmentation impacts between business units and IT while considering the impact of interim integration layers.
## Business Scenario / Modernization Task Mapping

<table>
<thead>
<tr>
<th>Transformation Task</th>
<th>Enterprise Inventory</th>
<th>Application Documentation</th>
<th>Application Cleanup</th>
<th>Data Mining / Cleanup</th>
<th>Data Redesign / Migration</th>
<th>Language Upgrade / Conversion</th>
<th>EAI &amp; User Interface Upgrade</th>
<th>Application Modularization</th>
<th>Architecture Migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strategic IT Planning</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Outsourcing Initiative</td>
<td>X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintenance Upgrades</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BPI / BPA</td>
<td>X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Rehosting</td>
<td>X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package Assessment</td>
<td>X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Package Implementation</td>
<td>X</td>
<td>X</td>
<td>X  X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>HIPAA (data upgrade)</td>
<td>X  X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Web-to-Host</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRM Initiative</td>
<td>X  X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Data Warehouse</td>
<td>X</td>
<td>X</td>
<td>X  X</td>
<td>X  X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application Consolidation</td>
<td>X</td>
<td>X</td>
<td>X  X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Component Development</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Web Services Migration</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Modernization
General Planning Guidelines

- Define business strategy and IT requirements
- Create and / or identify target architecture
- Define project(s) parameters and ROI
- Map existing environment against target requirements and related architecture
- Build plan to assess, capture or migrate, upgrade, reuse and / or deactivate legacy components
V. Modernization Processes, Techniques & Approaches

V. I Application Assessments
V. II Application Stabilization & Refactoring
V. III Application Transformation
V.I Application Assessments

- Enterprise Assessments
- Project-Level Assessments
Enterprise Assessment

Objectives

- Identify strategic information architecture requirements
- Inventory existing application portfolio
- Establish architecture transition strategy
- Link priority application projects with suggested modernization scenarios
Enterprise Assessment Breakdown

- Provides baseline for developing modernization options
- Establish scope and boundaries for cross-application projects
- Key to establishing clear requirements and strategy
## Modernization Planning Deliverable Summary

<table>
<thead>
<tr>
<th>Technical Architecture Assessment</th>
<th>Business Architecture Assessment</th>
<th>Data Architecture Assessment</th>
<th>Modernization Planning</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Current physical systems inventory</td>
<td>• Current systems summary</td>
<td>• Current data stores description*</td>
<td>• Current to target enterprise architecture summary</td>
</tr>
<tr>
<td>• Current physical systems attributes / metrics</td>
<td>• Current physical systems to current functions matrix*</td>
<td>• Current data stores to current systems matrix*</td>
<td>• Architecture transitioning &amp; impact summary</td>
</tr>
<tr>
<td>• Current to target technical architecture variances</td>
<td>• Current physical systems to target functions matrix</td>
<td>• Current data stores to target entity types matrix</td>
<td>• Suggested modernization project options</td>
</tr>
<tr>
<td></td>
<td>• Current systems inter-relationship DFD*</td>
<td>• Data mapping summary</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Functional mapping summary</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Repository View of Systems Decomposition Scheme

Tool captures **Red** information / Analysts add **Blue** information

This information can be stored in open tool vendor repository
• What mix of existing, new and package components will define an organization's application systems under the target architecture?

• How will organizations partition and transition existing systems to achieve strategic targets?
• Strategic Information Plan defines strategic architectures
• Modernization Plan defines how to achieve it
Key Tasks Regarding Modernization & Integration Planning

Modernization Plan identifies path forward

**Information Systems Architecture**
- Business Architecture
- Data Architecture
- Technical Architecture
- Define model driven, implementation independent design

**Architecture Modernization Strategy**
- Assess existing systems
- Map existing to target architecture
- Establish enterprise transition strategy
- Establish maintenance strategy
- Manage shifting target specifications
Evaluating Project Opportunities

- Major Enhancement
- GUI Front-end
- Data Warehouse
- Infrastructure Stabilization
- EAI
- ERP / Total Replacement
- Multi-system Consolidation
- DBMS Migration
- Componentization
- Language / Platform Conversion
Sample Modernization Strategy

**Existing Applications**

- Payroll
- Pension
- Insurance
- Accounts Payable
- Accounts Receivable
- Cash Disbursements
- Invoicing
- Customer Tracking
- Order Processing

**Modernization Options**

- **Integrated Human Resources System**
  - Finalize top-down designs
  - Collapse redundant functions / data
  - Transition functions/data to client/server
  - Phase deactivation of legacy systems

- **Review / Acquire Accounting Package**
  - Assess underlying package functionality
  - Migrate / collapse redundant legacy data
  - Effect shut-down of legacy systems

- **Phased Transition to New Billing System**
  - Stabilize existing applications
  - Develop integrated client/server design
  - Assess current systems
  - Employ phased migration to target architecture

Tasks include consolidation, migration, redesign and redeployment activities.
The Project-Level Assessment

Objectives

- Assess long-term business and information plans
- Identify the best modernization strategy to meet short and long-term information requirements
- Eliminate "guess work" typically involved in IT planning efforts
- Establish phased, cost effective modernization plan based on current position and future needs

Note: Assessments leverage Enterprise Assessment
Three sets of tasks are basis for integration and modernization projects. Relies on enterprise assessment.

- Architectural Assessment
- Existing Architecture Assessment
- Technical Assessment
- Disciplines & Tools
- Functional Assessment
- Meta-Data, Metrics, Documentation & Plan
Technical Assessment

Environmental Analysis

Documentation

Database Definitions

Source Code Reviews

Process Analysis

Data Definition Analysis

Subject Matter Expert Discussions

Metrics Documentation

Technical assessment inventories and documents existing systems environments, data usage and program flow.
- Current system flows, user views and data utilization summary
- Architecture summary metrics
- How systems are constructed and interrelate
Functional Analysis

- Current system, bottom up planning level models
- Multiple current system, integrated models
- Integrated top down / bottom up models
- Functional conformance metrics
Functional Assessment: Gap Analysis

- Assess the potential for data & behavioral reuse
- Determine % of target existing in current systems
- Determine % of target to be developed from scratch
Current-to-target data mapping extracts bottom-up data usage to create or validate target data models.
Program-to-function mapping helps determine level of legacy application conformance to target architecture.

Program / Function GAP Analysis

**Current Programs**
- PROGRAM 1
- PROGRAM 2
- PROGRAM 3
- MANUAL

**Current Functions**
- FUNCTION 1
- FUNCTION 2
- FUNCTION 3
- FUNCTION 4
- FUNCTION 5

**Target Functions**
- FUNCTION A
- FUNCTION B
- FUNCTION C
- FUNCTION D
- FUNCTION E

No Map
Determining Functionality: Reverse Requirements Tracing

Determines where certain functions are performed across a system.
Reverse requirements tracing determines application functionality by tracking user views back into the source code.

Reverse Requirements Tracing of Functions

<table>
<thead>
<tr>
<th>PROGRAM SOURCE</th>
<th>CURRENT PROCESS</th>
<th>CURRENT FUNCTION</th>
<th>TARGET FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Purchasing</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Sending/Receiving</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Inventory</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Manufacturing</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Financial Services</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Etc...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: USRM / Comsys
**Reverse Requirements Tracing of Logical Processes**

Reverse requirements tracing abstracts application functionality by tracking user views back into the source code.

<table>
<thead>
<tr>
<th>PROGRAM SOURCE</th>
<th>CURRENT PROCESS</th>
<th>CURRENT FUNCTION</th>
<th>TARGET FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Select vendor / product</td>
<td>Purchasing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Change vendor / product</td>
<td>Purchasing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Roll off vendor / product</td>
<td>Purchasing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Order supplies</td>
<td>Purchasing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pay vendors</td>
<td>Purchasing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Track vendor shipments</td>
<td>Sending/Receiving</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Disperse supplies</td>
<td>Sending/Receiving</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maintain supply stock</td>
<td>Inventory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Request supplies</td>
<td>Inventory</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Ship corporate products</td>
<td>Sending/Receiving</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Manage materials</td>
<td>Manufacturing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Request supplies</td>
<td>Manufacturing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Create products</td>
<td>Manufacturing</td>
<td></td>
</tr>
</tbody>
</table>

**Source:** USRM / Comsys

---

Vendors/Products Added

Vendors/Products Changed

Vendors/Products Discontinued

---

Reverse requirements tracing abstracts application functionality by tracking user views back into the source code.
Current-to-target functional analysis maps legacy programs, processes and functions to target architecture functional definitions.

<table>
<thead>
<tr>
<th>PROGRAM SOURCE</th>
<th>CURRENT PROCESS</th>
<th>CURRENT FUNCTION</th>
<th>TARGET FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU300200</td>
<td>Select vendor / product</td>
<td>Purchasing</td>
<td></td>
</tr>
<tr>
<td>PU300200</td>
<td>Change vendor / product</td>
<td>Purchasing</td>
<td></td>
</tr>
<tr>
<td>PU300200</td>
<td>Roll off vendor / product</td>
<td>Purchasing</td>
<td></td>
</tr>
<tr>
<td>PU300520</td>
<td>Order supplies</td>
<td>Purchasing</td>
<td></td>
</tr>
<tr>
<td>PU500000</td>
<td>Pay vendors</td>
<td>Purchasing</td>
<td></td>
</tr>
<tr>
<td>SR774200</td>
<td>Track vendor shipments</td>
<td>Sending/Receiving</td>
<td></td>
</tr>
<tr>
<td>SR600350</td>
<td>Disperse supplies</td>
<td>Sending/Receiving</td>
<td></td>
</tr>
<tr>
<td>INV08040</td>
<td>Maintain supply stock</td>
<td>Inventory</td>
<td></td>
</tr>
<tr>
<td>Manual</td>
<td>Maintain supply stock</td>
<td>Inventory</td>
<td></td>
</tr>
<tr>
<td>INV00100</td>
<td>Request supplies</td>
<td>Inventory</td>
<td></td>
</tr>
<tr>
<td>SR400100</td>
<td>Ship corporate products</td>
<td>Sending/Receiving</td>
<td></td>
</tr>
<tr>
<td>SR400200</td>
<td>Ship corporate products</td>
<td>Sending/Receiving</td>
<td></td>
</tr>
<tr>
<td>MA906600</td>
<td>Manage materials</td>
<td>Manufacturing</td>
<td></td>
</tr>
<tr>
<td>Manual</td>
<td>Manage materials</td>
<td>Manufacturing</td>
<td></td>
</tr>
<tr>
<td>MA665000</td>
<td>Request supplies</td>
<td>Manufacturing</td>
<td></td>
</tr>
<tr>
<td>MA240030</td>
<td>Create products</td>
<td>Manufacturing</td>
<td></td>
</tr>
<tr>
<td>MA240050</td>
<td>Create products</td>
<td>Manufacturing</td>
<td></td>
</tr>
</tbody>
</table>
Three-way vs. Two-way Mapping

Current-to-target-to-proposed mapping supports package assessments.
Implementation Project Planning

- Project-level analysis establishes detailed documentation and implementation plan
- Essentially becomes first step in a modernization project
<table>
<thead>
<tr>
<th>FUNCTIONAL CONDITION</th>
<th>ORGANIZATIONAL IMPACT</th>
<th>TECHNICAL QUALITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOOD</td>
<td>LOW</td>
<td>POOR</td>
</tr>
<tr>
<td>MAINTAIN / MIGRATE / INTEGRATE</td>
<td>UPGRADE / CONSOLIDATE</td>
<td>PHASE OUT</td>
</tr>
<tr>
<td>HIGH</td>
<td>LOW</td>
<td>GOOD</td>
</tr>
<tr>
<td>MAINTAIN</td>
<td>INTEGRATE OR ENHANCE</td>
<td>ENHANCE / CONSOLIDATE</td>
</tr>
<tr>
<td>LOW</td>
<td>HIGH</td>
<td></td>
</tr>
<tr>
<td>MAINTAIN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

High impact systems should be enhanced, consolidated, maintained or otherwise modernized or otherwise modernized based on business requirements.
## Software Change Strategy Summary

<table>
<thead>
<tr>
<th>Maintenance</th>
<th>Replacement</th>
<th>Enhancement</th>
<th>Technology Migrations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document &amp; assess existing baseline</td>
<td>Evaluate options</td>
<td>Document &amp; assess existing baseline</td>
<td>Document &amp; assess existing baseline</td>
</tr>
<tr>
<td>Implement stds, methods, procs</td>
<td>- High level architecture</td>
<td>- Apply analyzers</td>
<td>Determine needs / build migration plan</td>
</tr>
<tr>
<td>Position - for maintainability</td>
<td>- Package vs. Rewrite</td>
<td>- Build plan</td>
<td>Implement stds, methods, procs</td>
</tr>
<tr>
<td>Re-document</td>
<td>Rewrite / Redevelop</td>
<td>Implement stds, methods, procs</td>
<td>Positioning techniques</td>
</tr>
<tr>
<td>Personnel reallocation</td>
<td>- Assess existing base</td>
<td>Position systems as appropriate</td>
<td>- Translate</td>
</tr>
<tr>
<td>Port to workstation</td>
<td>- Build strategy</td>
<td></td>
<td>- Restructure</td>
</tr>
<tr>
<td>Refine testing procedures</td>
<td>- Position current system</td>
<td>Refine test procedures</td>
<td>- Rationalize data defs</td>
</tr>
<tr>
<td></td>
<td>- Refine new architecture</td>
<td>Add enhancements</td>
<td>- Re-aggregation</td>
</tr>
<tr>
<td></td>
<td>- Transformation support</td>
<td>Validate &amp; implement</td>
<td>Transformation support</td>
</tr>
<tr>
<td></td>
<td>Positioning as &quot;survival&quot; strategy</td>
<td></td>
<td>Validate &amp; implement</td>
</tr>
</tbody>
</table>
V.II Application Stabilization & Refactoring

- Improve source code quality so that it is easier to analyze, modify, debug and test
- Prepare systems for design level and physical component level reuse within target architectures
Legacy Applications & Data Documentation, improvement and data integrity scenarios streamline management and upgrade tasks.

Data Integrity
- Mining
- Cleanup
- Consolidation

Application Improvement
- Structuring
- Rationalization
- Modularization

Application Management & Application Management & Maintenance Facilitation Scenarios

Legacy Applications & Data

Application Documentation

Documentation, improvement and data integrity scenarios streamline management and upgrade tasks.
Application Staging

- Establish working versions of production source code
- Establish Positioning configuration management procedures
- Facilitate change control procedures to manage and assimilate production upgrades
Validation

- Verifies logical equivalence of retooled source code
- Involves executing comparisons to verify that no functional changes were inadvertently introduced
- It is NOT testing - testing looks for differing results, validation looks for equivalency
- Requires 60 - 70% execution coverage and good test suites
Validation testing attempts to automate as much of the process as possible.
Conversions & Language Level Upgrades

- Focus is on 2nd to 3rd, 3rd to 3rd or 4th to 3rd generation language conversion
- Does not address paradigm shift to objects
Delivers stable code with defective constructs removed
Provides standardized, structured source program logic
Resolves program level design aberrations
Restructuring of COBOL is highly automated.
Data Definition Rationalization

- Simplifies system-wide data tracing and analysis
- Externalizes data usage and representations
- Eliminates definitional redundancy and inconsistency
- Facilitates many strategic migration efforts
Field & Record Size Expansion

- Expands business data and related definitions
- Includes physical data and all definitions of that data
- Examples: area codes, bar codes, dollar amounts, dates
- This is a business–driven initiative
Combined Rationalization / Field Expansion View

- Existing Source
  - Data Definition Analysis
    - Review Analysis
      - Field Expansion Option
  - Build Composite
  - Composite Record
  - Composite Field Expansion
    - Propagate Composite
      - Field Expansion Trace
      - New Source
        - Expand Fields
  - Apply Descriptive Names
  - Secondary Trace & Propagation
Literal Externalization

- Finds embedded business data
- Places it into user – modifiable tables or databases
- Examples: dates, plant-codes, area-codes, interest rates, etc.
Code Slicing & Modularization

- Slicing allows analysts to eliminate or segregate portions of a program
- Modularization realigns application source code / functionality physically
- First step towards Services Oriented Architecture (SOA)
- Techniques maintenance, redesign, migration and package deployment
Code Slicing Techniques

- Isolates logic into new modules
- Supports functional modularity
- Simplifies maintainability for large programs
- Supports redundancy reconciliation and re-aggregation
Functions can be split out and recombined to modularize application functionality.
V.III  Architecture Transformation

- Leverage existing applications to facilitate multiple redevelopment scenarios
- Reuse, where applicable, system and program level artifacts
- Improve the integrity and completeness of new systems
- Shorten / streamline application development process
- Facilitate application integration efforts
Modernization options include data and business rule capture, consolidation, redesign, validation and redeployment.
Redesign & Re-Specification

Business Strategies

Planning

Analysis

Design

Construction

Current Systems

Planning

Analysis

Design

Construction

Target System

Adopted from Al Travis / CSC Index
Building a High Integrity Data Architecture

Deliverables
- Integrated top-down, bottom-up data model
- Gap analysis determining missing data entities & attributes
- Reconciliation of inconsistent data usage
- An understanding of legacy to target data migration requirements

Consolidating top-down and bottom-up data models produces a higher quality data architecture and increases likelihood of project success.
**Logical Functional Mapping Process**

**Top Down Inputs**
- I/A Functional Decomposition
- Logical Data Model
- I/A Functional Mapping

**Function Mapping Process**
- Expand I/A functional decomposition
- Expand I/A current to target mapping
- Complete current to target gap analysis
- Develop integrated functional decomposition & CRUD matrix

**Bottom Up Inputs**
- Source code, screen layouts & JCL
- Screen & report layouts
- I/A assessment results

**Deliverables**
- Detailed mapping of reusable source code components
- Identification of reusable I/O
- Integrated functional structures & data / functional matrix
Functional Derivation

- Functional analysis now extends to the business logic / business rule level
- Requires extraction, filtering, packaging and analysis of business logic
- Services as basis for validation of target functional models and other scenarios
Business Rule Derivation: Extending Functional Decomposition

- **Business Logic**: Bounded set of conditional and imperative statements that relies on and directly or indirectly impacts business data

- **Business Rule (Systems View)**: Description of how a business works

- **The Challenge**: Capturing and filtering business logic to achieve business rule candidates
Business Rule Derivation: Logic Capture, Filtering, Packaging and Analysis

- **Capture**: Process of capturing logic blocks that serve as business rule candidates
- **Filtering**: Eliminating non-business logic
- **Packaging**: Putting logic into a context that makes sense to business and design analysts
- **Analysis**: Categorizing, leveraging and reusing captured business rules
Rules can be reused as-is or as basis for legacy redesign.

Source: Netron, Inc. 2002
## Business Logic Must be Filtered From Non-Business Logic

<table>
<thead>
<tr>
<th>Non-Business Logic</th>
<th>Non-Business Logic Identification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntactically dead code</td>
<td>Logic never executed regardless of data values.</td>
</tr>
<tr>
<td>Semantically dead code</td>
<td>Logic not executed based on setting of data values.</td>
</tr>
<tr>
<td>Program initialization</td>
<td>Logic that initializes element or record area values.</td>
</tr>
<tr>
<td>Input / output logic</td>
<td>Code accessing physical data. Includes call, read, write and other I/O statements.</td>
</tr>
<tr>
<td>Output area build</td>
<td>Logic that moves data to screen and report work areas.</td>
</tr>
<tr>
<td>Status checking</td>
<td>Tests directly after I/O commands checking communication or other status codes.</td>
</tr>
<tr>
<td>Error handling</td>
<td>Imperative logic that invokes exception reporting or module termination based on status code results.</td>
</tr>
<tr>
<td>Data manipulation</td>
<td>Database or file manipulation logic.</td>
</tr>
<tr>
<td>Environmental logic</td>
<td>Manages security, homegrown technology, date handling or similar routines.</td>
</tr>
<tr>
<td>Extraneous logic</td>
<td>Redundant conditionals, mutually exclusive tests or similar routines.</td>
</tr>
</tbody>
</table>

Filtering guidelines help analysts separate business from non-business logic – a key step in rule derivation.
System / Application Name: Accounting / Accounts Receivable

Program Name: PGREG117

Paragraph Name: REGION-VALIDATION

Data Element Search Argument: Element = REGION-CODE

Logic:

\[
\text{IF REGION-CODE NOT } = \text{ ZERO}
\]
\[
\text{AND ACCOUNTING-CLEARANCE NOT } = \text{ ZERO}
\]
\[
\text{ADD WS-VALIDATION-AMT (WRK-SUB)}
\]
\[
\text{TO RGN-TOTAL (IE-REGION).}
\]
Logic Analysis Leads to Reusable Business Rule Candidates

System / Application Name: Accounting / Accounts Receivable

Program Name: PGREG117

Paragraph Name: REGION-VALIDATION

Business Rule: REGION VALIDATION
Precondition: REGION CODE is not a Zero (i.e., a valid REGION)
Trigger: ACCOUNTING CLEARANCE is not a Zero
         (i.e, a valid CLEARANCE)
Postcondition: REGIONAL ACCOUNTING TOTAL
               is increased by VALIDATION AMOUNT
In this example, Account Balance is the query item with rules as follows:
- Find all ACCT-BALANCE & related variables
- Capture rule back to first “IF” & down to end of test.
Redundancy Tracking & Consolidation

- Should address unnecessary data and functional redundancies
- Streamlining / consolidating redundancy requires ROI analysis
- Process is tool supported but must be business-driven
## VI. Modernization Tool Mapping to Tasks

<table>
<thead>
<tr>
<th>Tool Feature</th>
<th>Process Redesign</th>
<th>Asset Mgmt</th>
<th>Development</th>
<th>ReTool/Redeploy</th>
<th>Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>System Wide Analysis</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maintenance environment</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Redevelopment environment</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>System wide summary analysis</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>General purpose repository</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td><strong>Program Level Analysis</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process flow metrics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Program flow diagrams</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>In-line cross reference</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Interactive analysis tools</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td><strong>Code Improvement</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code restructuring</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Data definition rationalization</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Remodularization tools</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Physical data migration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Model Based Tools</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business planning tools</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Analysis tools</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Development tools</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Design tools</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Reverse engineering tools</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Integration Tools</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
VII. Modernization Scenarios

- Scenarios depict practical application of transformation discipline, tools and strategies
- Scenarios have a business-driven project focus and are typically delivered in phases
- Case studies reflect actual project results
12 Modernization Scenarios*

I. Application Portfolio Management
II. Application Improvement
III. Language-to-Language Conversion
IV. Platform Migration
V. Non-Invasive Application Integration
VI. Services Oriented Architecture Transformation
VII. Data Architecture Migration
VIII. Application & Data Architecture Consolidation
IX. Data Warehouse Deployment
X. Application Package Selection & Deployment
XI. Reusable Software Assets / Component Reuse
XII. Model-Driven Architecture Transformation

* Modernization Scenario White Paper -- http://adm.omg.org/adm_info.htm#white%20papers
Objective: Establish multi-dimensional knowledge base for managing & transforming applications.
Application Portfolio Management Scenario

**Issues:**
- Large IT portfolio, difficult to navigate
- Legacy systems poorly documented
- Cross-functional / cross-application project being proposed
- Outsourcing plan proposed

**Strategy:**
- Document cross-functional applications and relation to business functions in relevant business units using repository
Enterprise wide analysis scenario inventories systems portfolio, defines functional / physical boundaries and supports enterprise level transition planning.
Application Portfolio Management: Information Capture & Management

Legacy Applications

- H/R
- A/R
- CLAIMS

Repository
Physical / Logical Mapping & Impact analysis

Parse & Load Facility

Modernization Tools

Access to on-line Documentation
Scenario II. Application Improvement

Objective: Create a stable foundation for managing, enhancing or modernizing systems. Reduce application fragility, increase quality.

Redundant / poorly defined data & process definitions

Structure, rationalize, streamline, stabilize & otherwise refactor

Rationalized, structured streamlined source code
Application Improvement Scenario

Issues:

- Systems and programs hard to change and understand
- Poorly structured, designed source programs
- Data definitions redundant, and hard to understand

Strategy:

- Analyze and cleanup volatile programs and systems
Application Improvement Scenario

Convert, stabilize, structure, and/or slice selected modules in highly modified systems

Rationalize data definitions across systems of interest – determined during assessment

Data Definition Rationalization Process

Interim GUI data design effort

Strategic data modeling effort

Rationalized source code
Scenario III. Language-to-Language Conversion

Objective: Convert applications to new language and/or run time platform. Scenarios can be performed separately or together.

Scenario III:
Converts source code to new language or language level with run time environment

Scenario IV:
Migrates application to new hardware and/or operating system
Rehosting Scenario

Issues:
- Current platform obsolete or inaccessible
- Shared / relational databases require redesign
- Preference is to not rebuild - but to stay in code based environment

Strategy:
- Develop and implement phased migration plan
Rehosting Scenario Process

**Build Client/Server Migration Plan**
- Develop Assessment Proposal
- Assess Migration Requirements
- Finalize Code Based Migration Plan

**Prepare Systems for Code Based Migration**
- Stabilize Application Code
- Standardize System Wide Data Definitions
- Remodularize Selected Source Code

**Finalize Code Based Client/Server System**
- Finalize Client/Server Design
- Redesign & Implement Source Programs

SOURCE: TSRM / JAMES MARTIN & CO.
Rehosting Scenario - Target Architecture

Host based system is redesigned to the extent required to move it to Web-based environment - includes relational redesign of data structures.
Re-hosting Modularization Approach

Re-aggregation slices input/edit/validation logic and creates standalone routines to be ported to workstation environment.

- Host to N-Tier remodularization
- Monolithic Legacy System
- Remodularization

- Isolates I/O functions
- Isolates edit/validation logic
- Eliminates host related logic

Isolated/Consolidated I/O & Routines
Scenario V. Non-Invasive Application Integration

Objective: Create the option of accessing host applications & data via Web-based interfaces*.

*Non-invasive approach only impacts front-ends. Underlying architecture remains intact.
EAI Scenario

Issues:
- Users cannot access back-end application functionality
- Existing systems are not conducive to Web-based access and e-business requirements

Strategy:
- Build front-end with back-end connectivity to address user requirements
Integration via XML Case Study

- **Situation:** Manual reporting from front-end to legacy back-end
  - 25-year old MVS/CICS/390 system
  - Web-based Java Applet interface through Oracle Database
  - Reports generated by front-end Web-based system
  - Data manually re-keyed into MVS Back-end

- **Relativity Solution:**
  - Knowledge Mining of business logic establishing connectivity between mainframe and Websphere front-end
  - Developed “Mediator” to determine 3270 or Web calls
  - “Mediator” enabled XML data stream flow from Web, translated into BMS
  - Transaction message sent back in XML

Source: Relativity
Batch to Online Claims Processing Case Study

- **Situation:** Largest health insurance contract in nation serving 50% of government employees
  - 20-year old IBM S/390 mainframes, VSAM, DB2, 3 million lines of code
  - Claim processing nightly – batch system

- **Relativity Solution:**
  - Links mainframe to thin-client w/ web front end
  - IBM MQ messaging, WebSphere, Java Server Pages
  - Online, real-time access
  - Restructured COBOL, new Java-based programs

Source: Relativity
Re-aggregation slices input/edit/validation logic and creates standalone routines to be ported to workstation environment.

- Isolates I/O functions
- Isolates edit/validation logic
- Creates portable I/O validation routines

Monolithic Legacy System

Distributed Front-End Modularization
Host environment remains basically intact while new front-end routines handle all client interfaces.

Host system retains core processing logic, original TP monitor logic & data access facilities

*Note: Transaction I/O handling is emulated by new client environment*

Workstation handles Web-based I/O, edits, validation & host interfaces

*Note: Uses I/O macro to GUI conversion tool, on-line (CICS) monitor emulation & standard communication software (i.e. Micro Focus)*
Scenario VI. Services Oriented Architecture (SOA) Transformation

**Objective:** Create a framework for constructing and interlinking back-end systems with the goal of making applications more agile.
SOA Transformation Roadmap

- Excavate data meaning & business functionality from systems of interest
- Stabilize, standardize & structure code as required
- Rationalize and (optionally) redesign application data
- Re-aggregate systems to isolate data access, users interface, business logic
- Incorporate results into SOA framework
Redesigned Data & Modularized Systems Serve as SOA Building Blocks

Data & application architecture designed around functional components

- Applications decompose into single function modules (i.e. services)
- Business logic is isolated from I/O and user interface functionality
Scenario VII.
Data Architecture Migration

Objective: Consolidate, cleanse, redesign & migrate existing data structures to relational database. Refactor applications as required.
Data Architecture
Migration Scenario

Issues:
- Data is scattered, redundant
- Data does not conform to business needs
- Data inaccessible to Web applications

Strategy:
- Identify relevant data and approach
- Design new data architecture
- Capture, refine, consolidate and migrate data
- Perform phased migration and adjust applications
• Rationalized views used to create integrated, relational database
• New database may reside on host or in client/server environment
Data Architecture Migration: Functional Modularization

Re-aggregation slices input/edit/validation logic and creates standalone routines

- Isolate I/O functions
- Isolate data access / presentation logic
- Modularize functions based on functional assessment
Objective: Consolidate multiple redundant or related systems into a common data & application architecture.

Consolidation tasks:
- Integrate, automate common processes
- Consolidate & redesign cross-functional data
- Migrate baseline systems to new architecture
- Identify & consolidate redundant logic
- Create & phase in reusable modules
Consolidation Scenario Overview

**Issues:**
- Redundant or stovepipe applications cannot share data
- Segregated functions hindering business agility
- Redundant applications, data, functions inflating operating costs

**Strategy:**
- Design integrated target architecture
- Develop and perform phased migration of relevant applications
Consolidation Approach

"Personnel System Example"

POSITIONING STEPS
Step 1  Code stabilization / restructuring
Step 2  Data name rationalization
Step 3  Redundancy reconciliation & re-aggregation
Step 4  Client/server architecture finalization

TRANSFORMATION STEPS
Step 5  Redevlop person management sub-system
Step 6  Transform base payroll functions
Step 7  Transform base pension functions
Step 8  Transform base insurance functions
Step 9  Finalize target functionality / deploy integrated system
Phased Consolidation: One Application From Many

Integration & Transformation Tasks
- Integrate & automate common processes across business units
- Consolidate & redesign cross-functional data
- Migrate baseline systems to new architecture
- Migrate & consolidate subsequent business unit applications under new architecture
- Web-enable selected user interfaces as required

Target Architecture
Integrated Relational Database

Baseline Applications
- Business Unit 1
  - OE
  - Inv
  - Dist
  - Pro
- Business Unit 2
  - OE
  - Inv
  - Dist
  - Pro
- Business Unit 3
  - OE
  - Inv
  - Dist
  - Pro

Existing Applications
Baseline Applications
- OE
- Inv
- Dist
- Pro
Scenario IX.
Data Warehouse Deployment

Objective: Create non-operational, consolidated view of data & make this data available to business users in ad hoc formats.

Extract, analyze, transform validate & load
Integrated, rationalized, relational view of abstracted data
End user access to cross-functional data
Data Integration / Data Warehouse Scenario

Issues:
- Users find data inaccessible, hard to decipher
- Save data stored across multiple systems, databases

Strategy:
- Extract and consolidate relevant data and make it accessible to end-users, customers
Data Integration / Data Warehouse Scenario

**Build Graphical User Interface Plan**
- Develop Assessment Proposal
- Assess Current System Meta-Data
- Build User Data Query Plan

**Rationalize Cross System Data Representations**
- Rationalize Data Representations

**Develop Integrated Graphical Front-End**
- Finalize Integrated Data Architecture
- Complete Front-end & Server Design
- Deploy User Data Query Capability

*SOURCE: TSRM / JAMES MARTIN & CO.*
Data Warehouse Concept

Selective migration / integration of cross-functional data

End user access to cross-functional data
Data definitions are rationalized across systems of interest
Composite views are used as basis for integrated extract database design
Developing Warehouse

- Rationalized data views used to create integrated, relational extract database
- Extract database may reside on host or be downloaded to client environment
- Extract programs are developed to load & refresh extract data
Extraction & consolidation routines (in-house or commercial tools) must also purify extract data as defined during target attribute model.

**Operational Data:**
*Extract, Transform & Load*

**Warehouse Environment**

**GUI Front-end Routines**
Scenario X. Application Package Replacement Scenario

Issues:

- Current systems not meeting business needs
- Packages under review

Strategy:

- Assess package candidate inability
- Map selected package to legacy environment
- Perform phased package deployment and legacy deactivation
Package selection and replacement options can leverage or ignore legacy architectures. Option 5 uses the legacy architecture as the baseline for the new application as an alternative to licensing and implementing a package.

<table>
<thead>
<tr>
<th>Replacement Option</th>
<th>Utilizes Legacy Analysis Techniques</th>
<th>Reuses Legacy Application in Target Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Standard Package Selection</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>2. Package Selection with Gap Analysis</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>3. Standard Package Implementation</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>4. Package Implementation with Gap Analysis &amp; Legacy Reuse</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>5. Legacy Consolidation &amp; Migration to Target Architecture</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Application Package Selection & Deployment – phase one

Objective: Provide objective analysis of how well various packages meet strategic information requirements.

<table>
<thead>
<tr>
<th>Source Bus Function</th>
<th>REQ</th>
<th>PK1</th>
<th>PK2</th>
<th>PK3</th>
<th>L E G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Func a</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Func b</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Func c</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>--</td>
</tr>
<tr>
<td>Func d</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Func e</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Func f</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Func g</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>.......</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>.......</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>--</td>
</tr>
</tbody>
</table>

Functionality of Proposed Packages

Strategic Requirements

Package Assessment

Legacy Application Functionality

Package Selection Results & Plan
Application Package Selection & Deployment – phase two

Objective: Provide a concrete package deployment roadmap.

- Retain / Integrate with package
- Discard / Deactivate
- Current System Functionality
- Verify integrity of strategic requirements
- Strategic Requirements
- Add new functions to package
- Implement / Integrate
- Package Solution Functionality
- Do not implement
Scenario XI. Reusable Software Assets / Component Reuse

Objective: Create a repository of reusable components.
Component Reuse Scenario

Issues:

- Desire to move legacy functions to Web services
- Need to consolidate and reuse legacy functionality

Strategy:

- Define target architecture
- Perform phased legacy consolidation
- Migrates to component-based environment
- Deploy Web services
Component Reuse Tasks

- Address redundancy and reuse issues
- Perform consolidation as needed
- Select, extract and componentize rules
- Deploy under services-based architecture
Component Reuse Scenario Case Study

- **Situation**: 30-year old inventory management system supporting 91 bases worldwide
  - 6000 people accessing Unisys 2200 mainframe daily
  - 1.6 million lines of code, maintenance became unfeasible
  - After 3 years’ efforts, off-the-shelf software delivered only 27% functionality

- **Relativity Solution**:
  - Analysis and Knowledge Mining used to remove unused code and extract COBOL business rules
  - Java components generated from extracted COBOL business rules

Source: Relativity
Scenario XII. Model-Driven Architecture Transformation

Objective: Migrate existing applications to an environment in which systems are maintained in models.

- Extract business rules & data definitions
- Purge implementation dependent logic
- Rationalize/consolidate data & logic as needed
- Migrate/merge extracted logic & data definitions into applicable models

Existing applications & data definitions

Sample models
VIII. Getting Started: Assessments

- Snapshot the enterprise / develop an architecture transition strategy
- Objectively gather facts and requirements
- Identify hypotheses for various systems
- Build an interim and strategic plan
- Develop a phased migration strategy
Getting Started: Pilots

Develop and implement a pilot project to:

- Test drive and refine a selected strategy
  - Integrate redevelopment task set

- Get the application team on board
  - Verify tool requirements

- Fine tune validation approach

- Adjust application proposal, contract, workplan and cost justification analysis
IX. Selling Management / ROI
Identifying Business-Driven Opportunities

- Maintenance is drawing too many resources
- System reliability is an issue
- Integration requirements exist
- Migration to model driven approach is planned
- Any funded project where existing systems could play a minor or major role

A portfolio list identifying systems, platforms, size, & strategic requirements should be kept as a baseline from which to identify redevelopment candidates
Business Requirements Drive Value-Based ROI

- Reducing time-to-market for new products and services
- Shifting to a customer-driven philosophy
- Streamlining transaction flow across supply and distribution chains
- Creating flexible information systems to achieve business agility
Current State of ROI on IT Projects

- 83% of companies do not track or measure (ROI) on technology projects.*
- Companies that measure ROI track time lines and costs but not value.*
- Value must be determined to assess ROI

* Building and Measuring ROI for Application Integration Projects, Julia King, Computerworld ROI, Brainstorm Business Integration Conference, October 29, 2001
Value-Driven ROI Principles

- Existing data structures and applications are highly valued information assets.
- Application management and maintenance must consider business value - not just IT cost reduction.
- Rewrite, package replacement and integration projects must consider legacy information assets as key ROI factors.
- Projects impacting legacy environments should be cost justified in phases.
Value-Driven ROI Versus Total-Cost-of-Ownership (TCO)

- TCO breaks down IT spending by application area
- One study showed TCO increasing $2,300 per employee after an ERP implementation*
- But if revenue increased $10 million as a result...
- ...the resulting annual cost increase of $1.1 million (based on 500 IT personnel) would still return an ROI of $9 million**

The Wrong Project Approach
Results in Negative Project ROI

- At Hershey, a $112 million ERP implementation resulted in shipping delays and deliveries of incomplete orders*
- This, in turn, caused Hershey’s profits to drop 19% in one quarter*
- Which increases criticality of selecting the right approach and delivering it in phases

*Failed ERP Gamble Haunts Hershey, by Craig Stedman, Computerworld, November 1, 1999
In 2000, top spending priorities at financial service firms: customer relations (37%), competition (18%), cost savings (14%) and efficiency (10%)*

By 2001, priorities shifted to cost savings (32%), efficiency (29%), market share (23%) and customer demands / requirements (15%)*

Customers fell in priority while cost saving priorities increased

*Budget Season is Here: Financial Services IT Spending is in for Big Changes, by John Hagerty, AMR Research, May 2001
Phased ROI: Reduces Project Risk & Helps Manage Shifting Priorities

- “Assessing the business value, related costs and annual projected TCO for a project should be incorporated into a cost model that quantifies ROI by project phase.”*

- “If a project is put on hold or cancelled due to a shift in business priorities, that project will have delivered some value prior to being cancelled.”*

Value-Based ROI for IT Projects*

- Identify various options to meet a given set of business requirements
- Estimate revenue growth and / or cost savings to be gained from each approach
- Calculate cost of delivering each approach as the implementation cost plus annual operating costs
- Determine “net value” for each option: Subtract project costs from projected annual revenue growth and / or cost savings

Information Asset Management & Maintenance ROI

- Application assets must be documented whether maintained in-house or offsite.
- Proactive upgrade activities can streamline delivery of business value or prepare applications for strategic transformation.
- Outsourcing / insourcing agreements should incorporate value-added activities - not just cost reduction.
Transformation Support for Outsourcing ROI

- IT cost cutting is currently the driving force behind outsourcing initiatives
- Value-added outsourcing ROI goes beyond cost cutting and includes:
  - Documenting applications in an open repository
  - Integrating applications using EAI technology
  - Upgrading, migrating and / or consolidating enterprise data and applications
Phased ERP Deployment Leverages Modernization Disciplines

**Phased alternative to standard ERP deployment approach**

- Migrate existing applications to an industry-standard database
- Migrate application front-ends to a client/server environment
- Add core ERP functionality to this environment
- Integrate additional ERP features with best-of-breed solutions

X. Summary / Close / Questions