

- Promise

Using a re-architecture approach, businesses can transform legacy applications in order to keep current and responsive to market demands.

- Reality

Businesses and IT departments with significant investment in existing systems must both respond quickly to business mandates and intense competition while minimizing the loss in early investments.

- Problem

Wholesale replacement of outdated systems to stay current is not necessarily the best approach.

- Solution

The re-architecture of legacy applications allows existing systems to be updated in the most seamless manner.

## Introduction

Intense competition, deregulation, mergers/acquisitions and compliance are forcing IT departments to react speedily to changing business demands. These organizations in turn find it difficult to respond in a timely manner to business mandates because inflexible and complex outdated systems preclude timely response.

There are four primary approaches to the modernization of a legacy application. These include:

- *Replace*: With packaged software
- *Re-host*: Recompile on to new hardware
- *Re-write*: Custom develop the application
- *Re-architect*: Transform the legacy application into an n-tiered application

The selection criteria for which approach to take depend on several factors. The key drivers include:

- Need for scalable and flexible functionality and technology
- Avoidance of technology lock-in

- Implementation time, costs, and risks
- Ability to leverage investment made and existing IT knowledge
- Ability to minimize re-training and retain operational continuity
- Compliance to industry and organizational standards
- Ability to integrate with existing IT infrastructure

While there are many approaches to mechanical conversion of code from one language to another, the best results can be produced if the information for the entire program can be referenced in order to generate the new code. Frequently, some information about the entire system and hardware on which it runs must also be taken into consideration.

### Understanding the Re-Architecture Approach

The Re-architecture approach is the efficient transformation of existing legacy systems to open, multi-tier environments, preserving core business functionality of the legacy system, while shedding any constraints of legacy applications.

The approach comprises:

- Comprehensive understanding of the features and functions of the source architecture
- Defining the new target architecture
- Reverse-engineering the problem to provide valuable features and functions.

Both the source and target architectures will typically involve the following elements:

- Presentation/Front-end
- Business Functionality
- Database
- Transaction/System, and
- Hardware (HW) and Operating System (OS)

### Enter: *Toolkit*

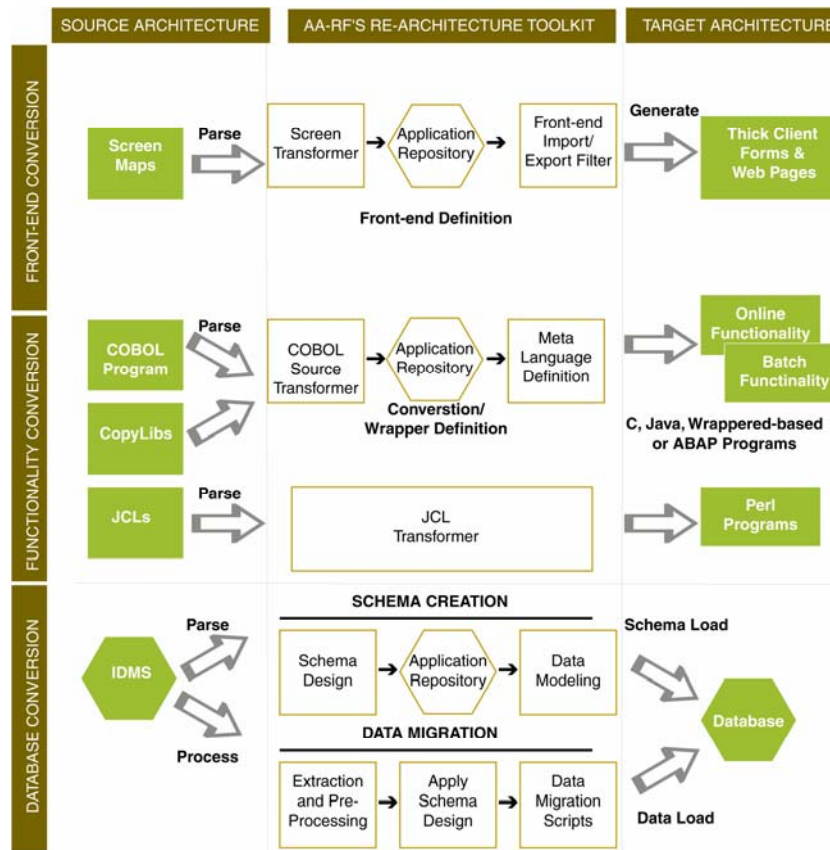
Through the use of a *toolkit*, unused (“orphan”) code and sub-optimal logic can be identified and either enhanced or eliminated paving the way for more reliable, modernized code. The *toolkit* is an expandable system that allows parsing, analyzing and converting computer programs and systems from one language and architecture into another. It provides the framework for implementation of analytical and code conversion modules. The principal idea behind the *toolkit* is to enable easy development of specialized processing components for distinct tasks. *Toolkits* make sense because:

- The conversion of code from one architecture or language to another is primarily a mechanical process
- The rules for conversion can be described as a sequence of manipulations on the original code that produces new code. (Any step that can be described in this way can be implemented as an automated solution.)
- Any systematic changes to an existing system may be applied through an automated system

## The Toolkit Framework

The *toolkit* processing unit is a complete set of files to run on a source language compiler. This would be a COBOL program with all the copybooks.

The following diagram shows an overview of the transformation organization *toolkit*.



There are three primary areas of conversion:

- Front-end
- Functionality
- Database access and data

Discrete parts of the process are split into several distinct categories:

- *Parsing tools*—that read the input code and generate Meta language information,
- *Data manipulations tools*—that manipulate the data inside the toolkit, and
- *Output modules*—that generate either output code or analysis information, such as variable or file usage.

The modular design and unified interface allow for the addition or removal of processing steps and ease of debugging for the new modules.

The first step toward conversion is to parse the COBOL code to generate a generic format.

### **Front-end Conversion**

The front-end conversion tools translate the Screen Section within the COBOL code to web pages or the client server screen, based on the front-end definition. This front-end definition is based upon the target front-end requirements and is stored in the application repository.

### **Functionality Conversion**

There are two alternatives to migrating the functionality or business logic to a target architecture; converting to target architecture (like Java Beans, .Net Components, C) or wrapping the COBOL based business logic with target architecture code.

The second alternative is often preferable as it retains the business logic within the COBOL code.

Business functionality conversion tools translate the COBOL code to service-based business components using Meta language definition in the application repository to provide the conversion logic. Meta language definition consists of database access methods and rules that translate COBOL statements like MOVE and GOTO.

### **Database Conversion**

Database conversion tools translate the COBOL Copylib code to database schema and data migration scripts based on ANSI SQL standards. Database conversion tools maintain data objects and data access mechanisms within the application repository which is used only during the conversion process.

Some of these conversion tools may require enhancements to support a particular target architecture, the organization's technology standards as well as its integration requirements.

## **The Benefits of a Re-Architecture Approach**

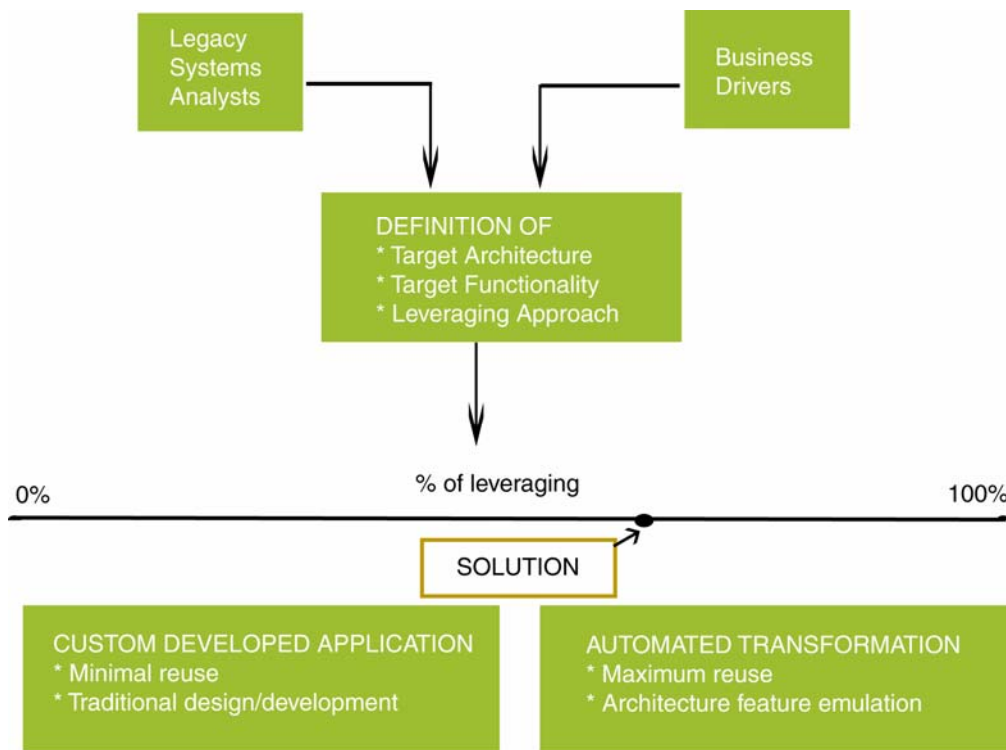
Some of the benefits of re-architecture include:

- Leveraging business processes, technical infrastructure and support personnel.
- Developing a custom architecture based on business and technical drivers.
- Fundamental change in system development and maintenance processes through implementation and customized training.
- Increase in flexibility for future growth.
- Reduce enhancement turn-around time.
- Facilitate easier maintenance - client/server architecture.
- Lower operating costs - cheaper hardware & software.
- Lower development costs - faster system development cycle.
- Increase in user productivity - GUI & relational database features.
- Improve competitiveness - faster response to market demands.
- Low risk alternative.
- Shorter development time (faster than custom development).
- Leverage existing hardware.
- Increase in revenues - new services & strategic business applications.

The following table provides a comparison between a complete Re-write and the Re-architecture solution.

Rewrite (Custom Development) (0%)	Re-architecture (Automated Transformation (100%))
<ul style="list-style-type: none"> <li>• High risk in terms of existing business process capture</li> <li>• Maximum flexibility for system enhancement</li> <li>• Increase in ability to maintain the application</li> <li>• Relatively longer due to addition of process design and test phases</li> </ul>	<ul style="list-style-type: none"> <li>• Minimum risk in terms of business process preservation</li> <li>• Reduced flexibility due to source architecture emulation</li> <li>• System maintenance requires dual skills – source and target architectures</li> <li>• Relatively shorter development timeframe due to high degree of automation</li> </ul>

The diagram below offers a “Continuum of Leveraging Options” view of the Re-write and Re-architecture approach.



The method to determine a suitable approach for an organization can be determined through a series of solution workshops that addresses:

- Analysis of the legacy applications,
- Understanding business drivers for modernization,
- Developing organizational baseline of technical architecture and systems development,
- Understanding technology critical success factors,
- Developing target enterprise architecture components, and
- Creating implementation roadmap for leveraged/custom development.

While the trade-offs may vary by application, a combination of Rewrite and Re-architecture yields optimal results.

### Case Study

Avenue A/Razorfish re-architected Alliant Utilities (Wisconsin Power & Light) Customer Information System (CIS legacy) in less than a year, and total transition in 18 months.

Challenge	Solution
<p><i>Business Problem</i></p> <ul style="list-style-type: none"> <li>• Deregulation forcing WPL to compete on customer service; four-way merger into Alliant Utilities</li> <li>• Need flexible customer information system (CIS) to survive and deal with exponential increase in numbers of customers</li> </ul> <p><i>Source Environment</i></p> <ul style="list-style-type: none"> <li>• Character 3270; BMS Screens</li> <li>• COBOL / CICS &amp; Batch</li> <li>• IMS / VSAM</li> <li>• 3.5 Million lines of code</li> <li>• 700+ Users; 300 Screens</li> <li>• 400+ IMS / VSAM entities: 100 GB Data</li> <li>• 12hr-72hr batch cycle; 150 batch programs</li> </ul>	<p><i>Re-architect Open 3-tier C/S</i></p> <ul style="list-style-type: none"> <li>• Rapid enhancement Implementation</li> </ul> <p><i>Application – Target</i></p> <ul style="list-style-type: none"> <li>• Client – PowerBuilder, Java</li> <li>• Functionality – ANSIC</li> <li>• Data – Oracle</li> <li>• Environment – HP-UX</li> </ul> <p><i>Benefits</i></p> <ul style="list-style-type: none"> <li>• Faster time to market (3 – 4 months)</li> <li>• Customer Internet access</li> <li>• Customer-centric Relational database</li> <li>• 24x7 operation</li> <li>• Non-utility billing feature</li> </ul>

## About Avenue A | Razorfish

Avenue A | Razorfish ([www.avenuea-razorfish.com](http://www.avenuea-razorfish.com)) is the largest independent interactive services firm and an operating unit of Seattle-based aQuantive, Inc. (NASDAQ: AQNT), a digital marketing services and technology company. Avenue A | Razorfish solutions are entrenched in deep technology, rigorous analytics and a rich understanding of customer needs, including award-winning web media & creative, search marketing services, email marketing/eCRM, and world-class creative, design and implementation of customer websites and intranets/extranets. Avenue A | Razorfish has offices in 12 US cities and is headquartered in Seattle, Washington. Clients include AstraZeneca, Best Buy, Kraft, Microsoft/MSN, WeightWatchers.com, and Wells Fargo.

Avenue A | Razorfish  
821 2<sup>nd</sup> Avenue, Suite 1800  
Seattle, WA 98104  
Voice: 206.816.8800  
Fax: 206.816.8808  
[www.avenuea-razorfish.com](http://www.avenuea-razorfish.com)