Databorough

A Visual Guide to Automated MVC Reengineering

Steve Kilner

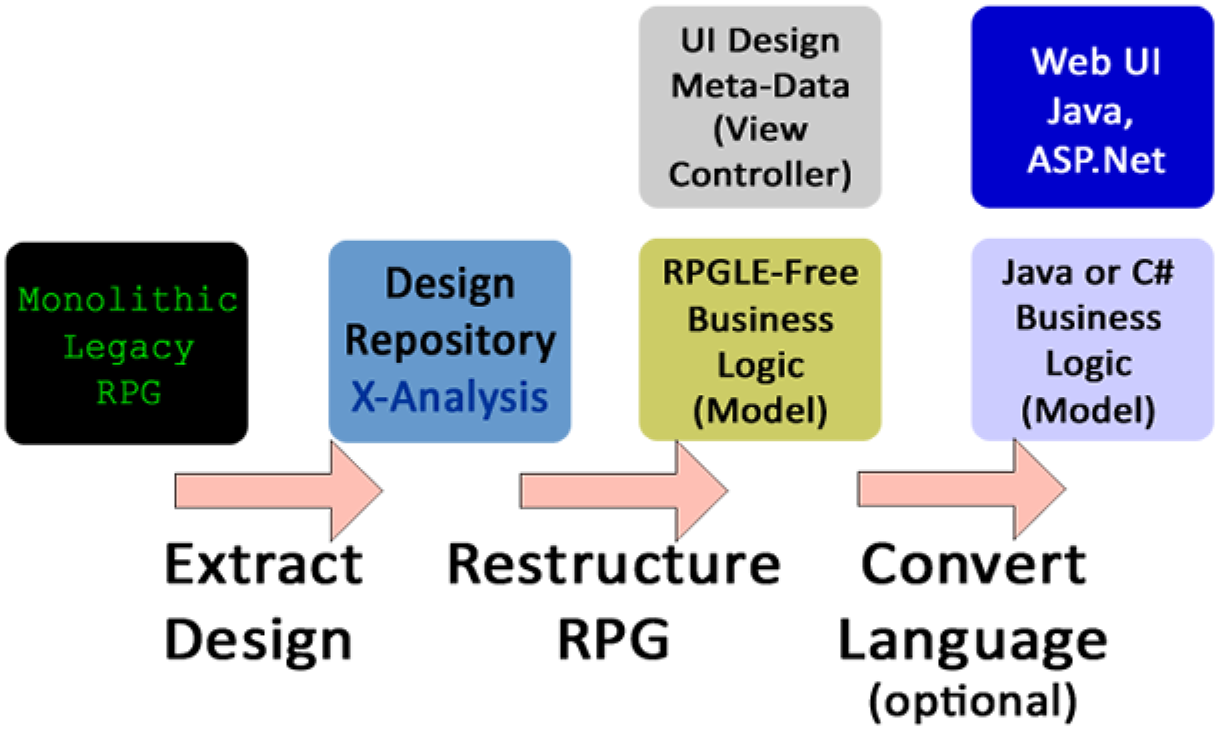
This guide has the following sections:

While this page explains automated reengineering in the context of Databorough products, it also presents an essential strategy for IT organizations with legacy RPG applications

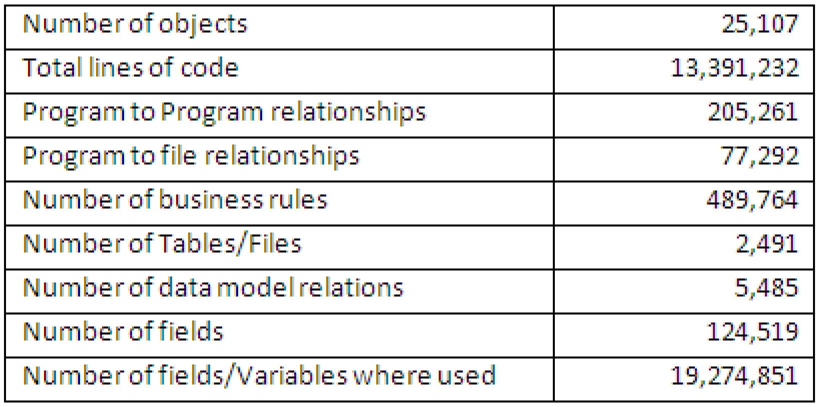
By first restructuring legacy RPG programs into a modern architecture, **new capabilities emerge to**

- extend the life of RPG application - or -
- migrate to a new language and platform

Three Steps (or maybe just TWO) to New & Improved System


UI Design Meta-Data (View Controller)
Web UI Java, ASP.Net
Monolithic Legacy RPG
Design Repository X-Analysis
RPGLE-Free Business Logic (Model)
Java or C# Business Logic (Model)
Extract Design
Restructure RPG
Convert Language (optional)

Extract Design – The first step in any reengineering project is to recover as much intelligence from the existing application as possible. X-Analysis has an amazing amount of information it recovers about applications. For example, X-Analysis generates the following numbers for one large scale ERP application:



Number of objects	25,107
Total lines of code	13,391,232
Program to Program relationships	205,261
Program to file relationships	77,292
Number of business rules	489,764
Number of Tables/Files	2,491
Number of data model relations	5,485
Number of fields	124,519
Number of fields/Variables where used	19,274,851

Restructure RPG – Restructuring monolithic RPG code into an architecture of OO/MVC/REST brings numerous benefits, both for ongoing maintenance, if the system is to be retained, and for complete reengineering if migrating to another language, such as Java or C#.

Using Databorough's X-Modernize tool, legacy, monolithic RPG programs can be restructured into a modern architecture. Such a restructuring results in the business logic residing in exportable RPGLE/Free procedures. These programs represent the model portion of the architecture. X-Modernize also provides the meta data for you to construct your own view and controller components, which may be in RPG, Java or C# for the controller layer and DDS, JSF or .Net for the view layer.

Convert Language (optional) – For organizations who have made a strategic decision to migrate to a new language or platform, Databorough's X-Migrate product (formerly X-Redo) completes the transition from restructured RPG into another language, such as Java or C#.

An important benefit of using the restructured RPG for the language conversion is that it facilitates bridging the knowledge gap between the RPG application and the new language application. The new language application retains the same essential components as the restructured RPG application thus enabling both RPG developers and new language developers to easily compare code results.
X-Migrate also provides the extra step of creating the view and controller components in the target languages, Java, C#, JSF, .Net.

Why Reengineer?

Typical RPG applications represent millions of dollars of investment over the years of their existence.

Such applications typically contain many thousands of business rules , data modeling rules and user interaction rules. To attempt to rewrite so much functionality from scratch would be a huge undertaking and can be greatly expedited by making use of information from the existing application.

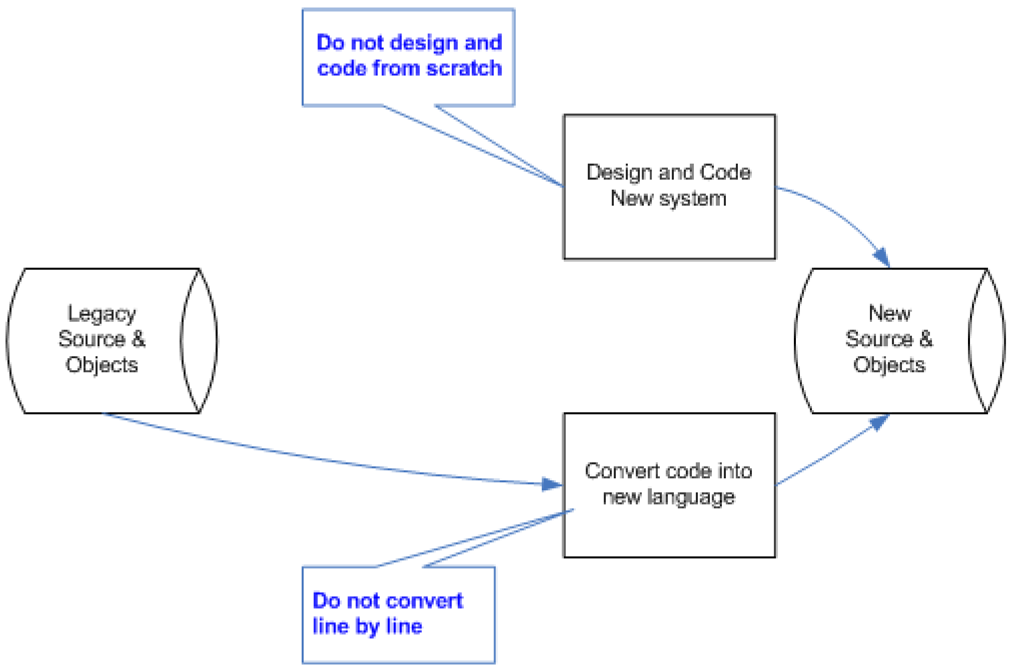
Some of the Goals of Reengineering can Include:

- Obtaining the benefits of a more modern architecture, such as:
    - Modularity.
    - Loose coupling .
    - Reusability
    - Accessibility
    - Distributability
    - Scalability

- Improving the application's maintainability and responsiveness to business needs. By untangling the accumulation of many years of modifications and restructuring and refactoring overly complex, monolithic code, maintainability can be significantly improved
- Exposing functionality embedded within the application so that a web services, or SOA, architecture can be realized
- Moving the application to another language or platform more in line with future directions of IT.

There is no single correct approach to reengineering – there are many options, and each organization must decide which option best fits its strategy and needs.

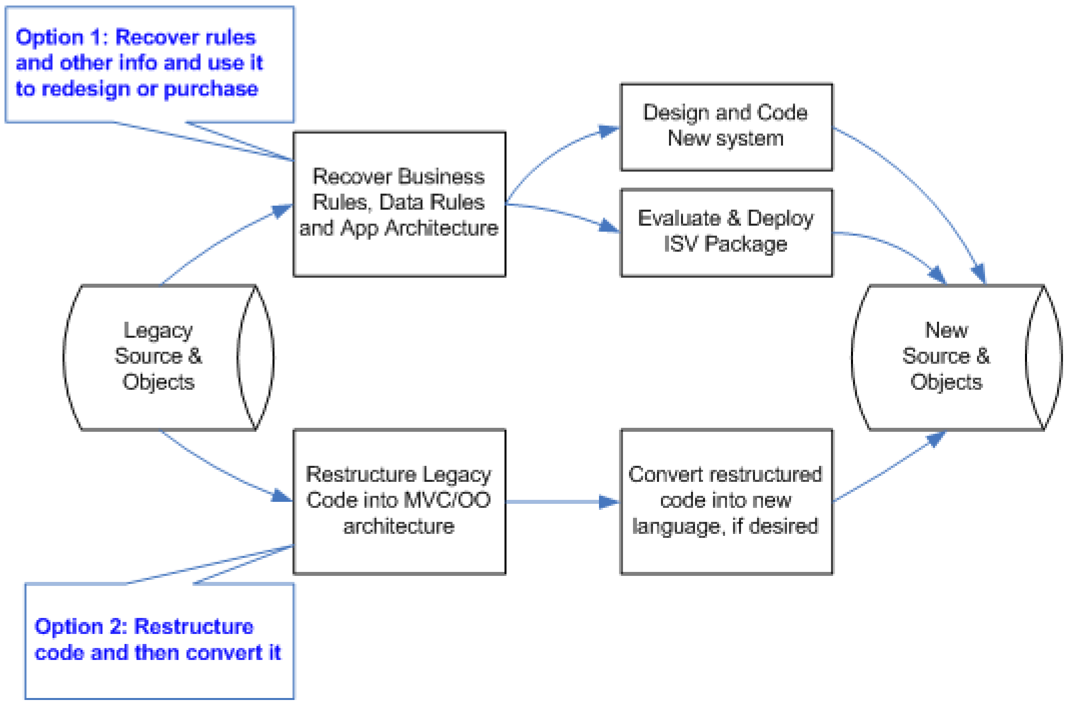
Bad Practices of Reengineering- What Not to Do


Do not design and code from scratch
Design and Code New system
Legacy Source & Objects
New Source & Objects
Convert code into new language
Do not convert line by line

Do not design from scratch – You will add huge amounts of risk, cost and time to your project. Your legacy system has vast amounts of proven, recoverable business rules, data rules and application definitions – use them and save on money, time and risk.

Do not convert line by line – You think your current system's maintenance is slow, expensive and risky? Wait till there is no individual, let alone an entire team, who knows both your business and the new programming language.

Bad Practices of Reengineering- Two Good Options


Option 1: Recover rules and other info and use it to redesign or purchase
Recover Business Rules, Data Rules and App Architecture
Design and Code New system
Evaluate & Deploy ISV Package
Legacy Source & Objects
New Source & Objects
Restructure Legacy Code into MVC/OO architecture
Convert restructured code into new language, if desired
Option 2: Restructure code and then convert it

Option 1: Recover rules to rewrite or purchase – Your legacy application may represent as much as millions of dollars of investment. It's *full of time-tested business rules, data relationships, application boundaries* and so on. You may well want to improve all those things, but chances are that the vast majority of this design is still good – use it!

The recovered business rules and other information can be fed into:

- The requirements and design processes of a new system, or
- The functionality and gap analyses of packages being considered for purchase

Option 2: Restructure code first, and then convert it – Before converting your RPG application to another language, first restructure it into a modern architecture with characteristics such as:

- Object oriented structure
- Model-View-Controller, MVC, architecture
- Stateless, or REST, session management
- Free formatted coding, such as RPGLE/Free

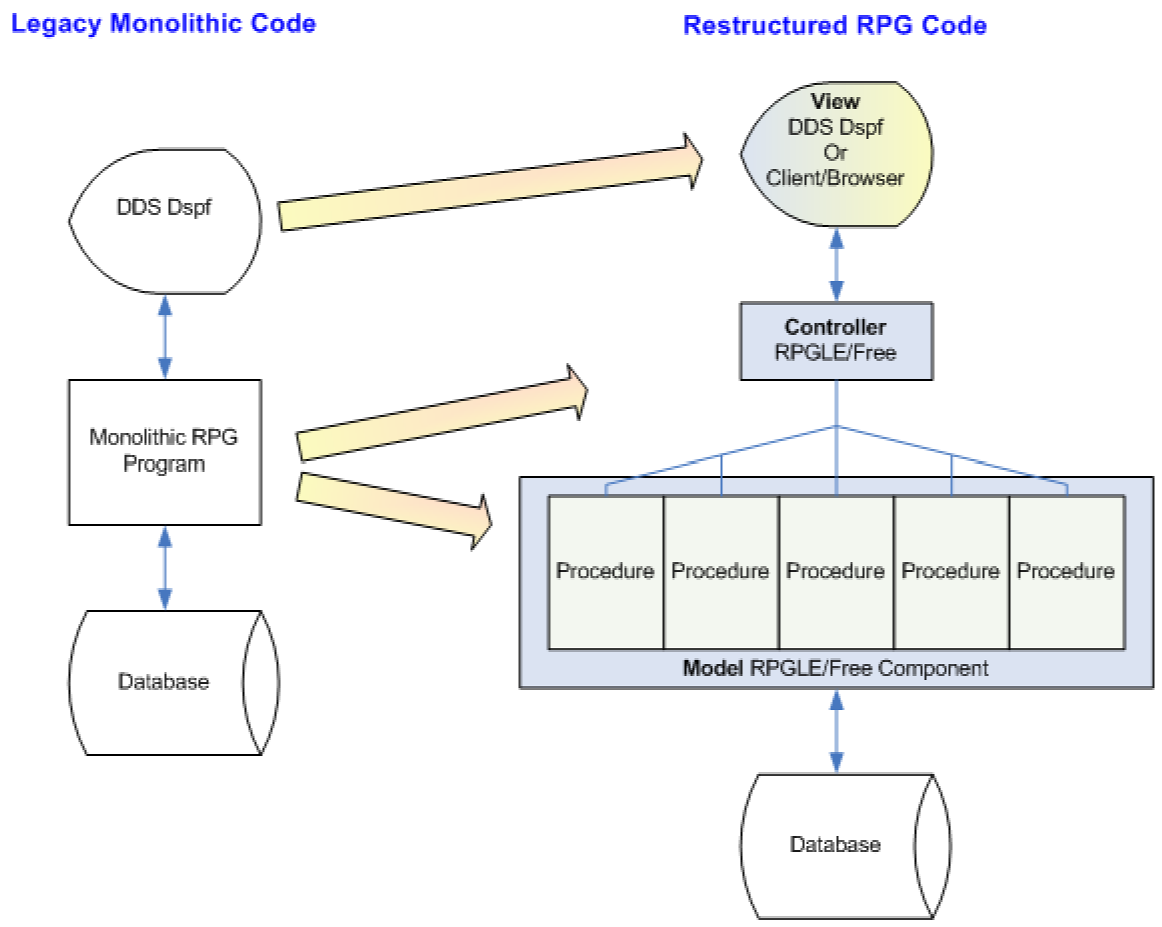
By first restructuring RPG into a modern architecture it enables the existing team with their existing skills to understand and work with a more modern architecture. It also has the benefit of improving the maintainability and openness of the application.

Any subsequent conversion to another language, if desired, results in a new application with the same component architecture as the original, restructured application, thus greatly improving understanding and manageability.

Automated Restructuring and Modernization of Monolithic Legacy RPG Code

Overview

Once X-Analysis has been used to recover design information, X-Modernize can build the restructured components in RPGLE/Free, and provide view and controller meta data. The meta data can be used to write the view and controller components in the language of your choice, as shown below (RPGLE/Free shown as example in diagram):



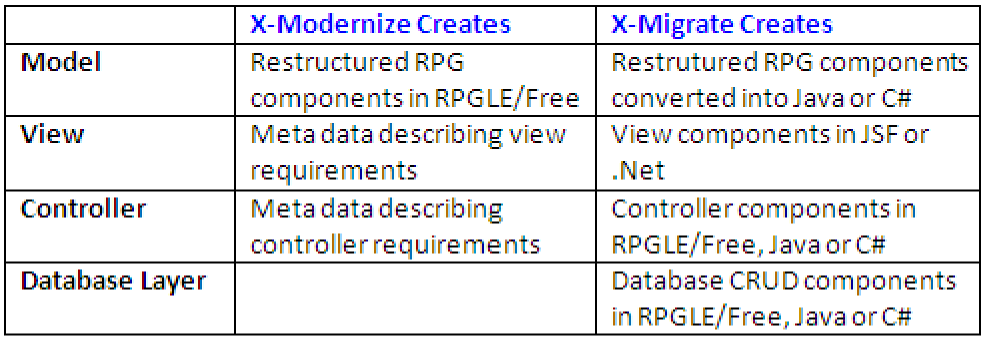
In this diagram, the monolithic RPG program and its display file are restructured into three components:

Model – this component is comprised of RPGLE/Free with all code having been reorganized into procedures that are exportable and may be called by other components, specifically the controller component.

View – the initial Databorough restructuring process, as executed by X-Modernize, provides the meta data necessary to manually code the view component. The view component may be coded as existing display file DDS, or recoded into JSF or .Net.

Controller – like the view component, the initial restructuring process provides the meta data enabling the development of a controller component in RPG , Java or C#. This component controls the interaction between user (the view) and the functionality (the model).

Understanding what is generated by different Databorough products related to reengineering-



	X-Modernize Creates	X-Migrate Creates
Model	Restructured RPG components in RPGLE/Free	Restrutured RPG components converted into Java or C#
View	Meta data describing view requirements	View components in JSF or .Net
Controller	Meta data describing controller requirements	Controller components in RPGLE/Free, Java or C#
Database Layer		Database CRUD components in RPGLE/Free, Java or C#

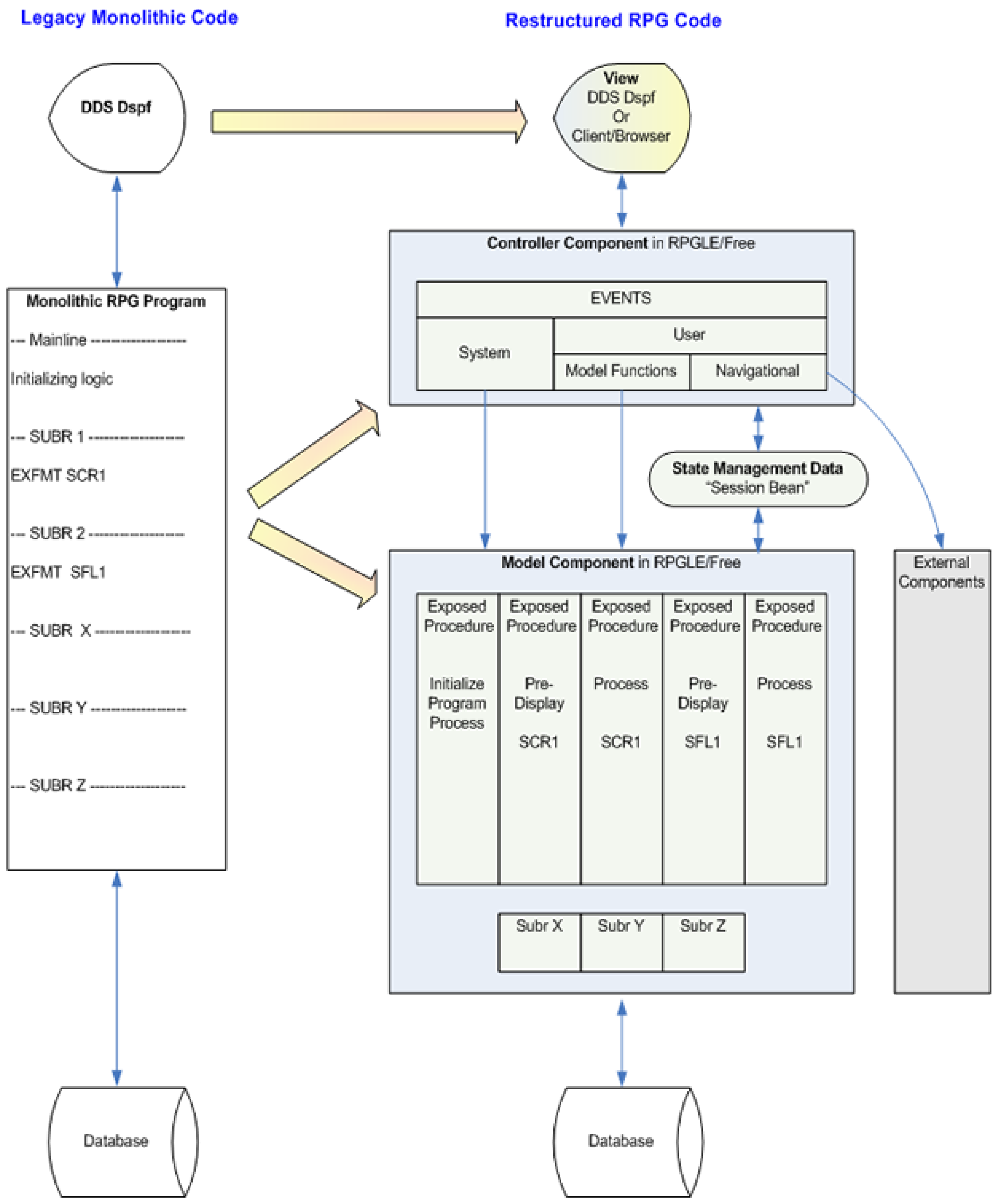
Note: The X-Analysis module is also required inorder to build the repository needed to drive these functions.

Restructuring RPG – The Details

By leveraging the intelligence gathered in the X-Analysis repository, X-Modernize can restructure monolithic, procedural RPG code into more manageable and useful procedures.

The overall architecture being created is:

- Object oriented, in so far as is practical in RPGLE
- MVC architected
- Stateless, making use of a state data management object, also known as a session bean


Legacy Monolithic Code
Restructured RPG Code
DDS Dspf
View
DDS Dspf
Or
Client/Browser
Controller Component in RPGLE/Free
EVENTS
System
User
Model Functions
Navigational
State Management Data
"Session Bean"
Monolithic RPG Program
--- Mainline ------------------
Initializing logic
--- SUBR 1 ------------------
EXFMT SCR1
--- SUBR 2 ------------------
EXFMT SFL1
--- SUBR X ------------------
--- SUBR Y ------------------
--- SUBR Z ------------------
Model Component in RPGLE/Free
Exposed Procedure
Initialize Program Process
Exposed Procedure
Pre-Display
SCR1
Exposed Procedure
Process
SCR1
Exposed Procedure
Pre-Display
SFL1
Exposed Procedure
Process
SFL1
Subr X
Subr Y
Subr Z
External Components
Database
Database

View component – By design, the view component handles the UI data definitions, layout and user event definitions. In the first pass of restructuring, X-Modernize provides meta data that can be used as design specifications for developing the view component in JSF or .Net.

Controller component – By design, the controller component receives all events, whether system generated or user generated, such as function keys, selecting subfile options, etc, and invokes the appropriate procedure in the model component. The controller component also receives and retransmits the state data management object, ensuring that all data needed to maintain state is preserved in each user session. In the first pass of restructuring, X-Modernize provides meta data that can be used as design specifications for developing the controller component in RPG, Java or C#.

Model component – The model component contains the essence of the functionality of the original RPG program. All business logic, such as that used for loading data from files, preparing data for display, validating data, preparing data for output and other miscellaneous calculations are contained in the model component.

All such functionality is organized into exportable procedures, which are called by the controller component, or anything else, if desired. The procedures are organized around the display of screen formats, such that for any given screen format that is displayed in a user session, there is one procedure for preparing the format for display, and another one for post-display processing, such as validations or data output.

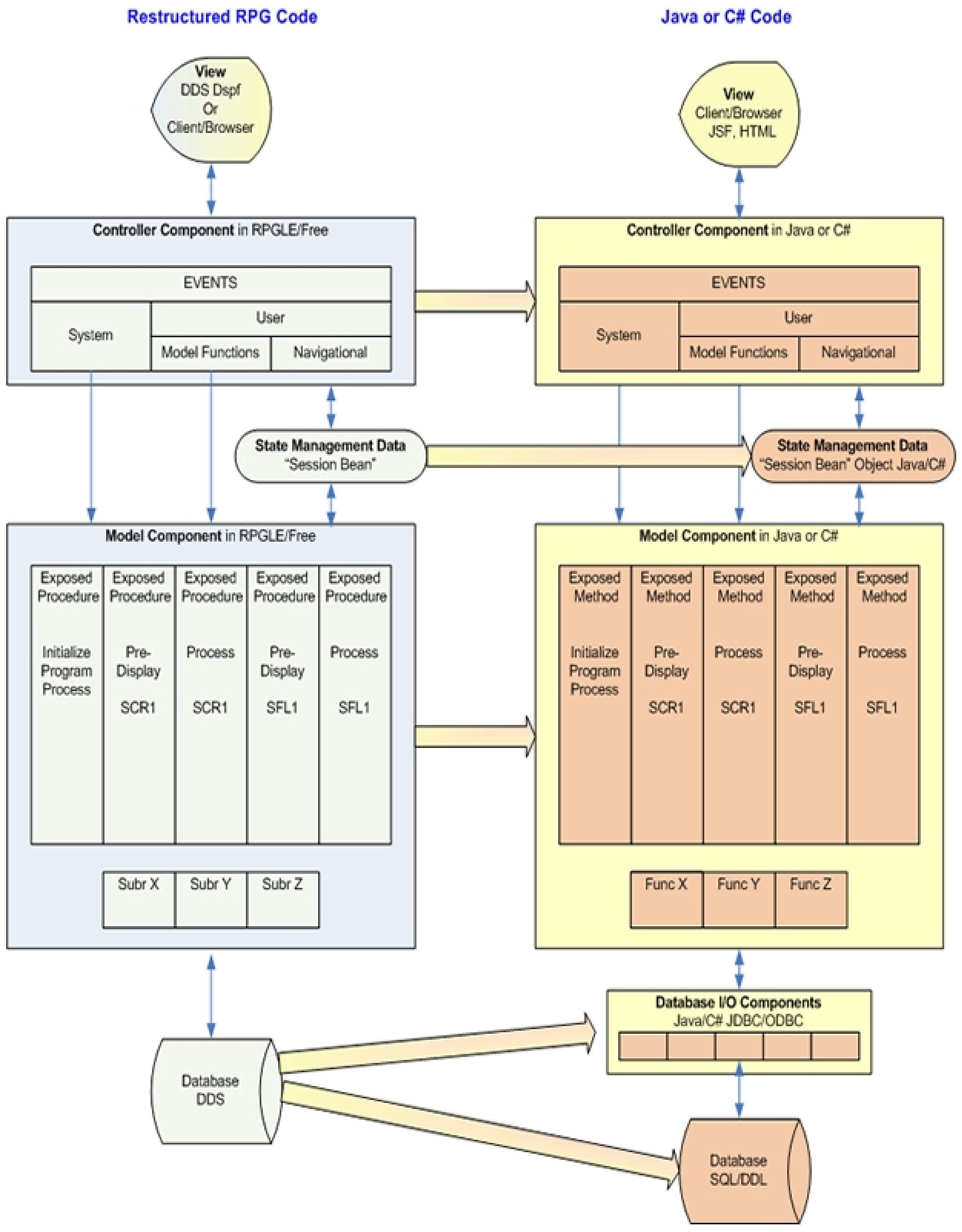
State management data – In order to create a stateless architecture, a data object is created which contains all data necessary to maintain the state of the user's session between screens. This data object is passed back and forth between the controller component and the model component.

Converting Restructured RPG to Another Language

Once an RPG program or application has been restructured, it is optimally positioned for conversion to a more modern language, if that is the direction of IT.

The point should be made, however, that restructuring RPG may be an end strategy in itself, as the application's maintainability and openness to SOA and other interfaces is greatly improved.

If desired, however, the restructured RPG may undergo a further step of being converted to Java or C#.


Restructured RPG Code
Java or C# Code
View
DDS Dspf
Or
Client/Browser
View
Client/Browser
JSF, HTML
Controller Component in RPGLE/Free
EVENTS
System
User
Model Functions
Navigational
Controller Component in Java or C#
EVENTS
System
User
Model Functions
Navigational
State Management Data
"Session Bean"
State Management Data
"Session Bean" Object Java/C#
Model Component in RPGLE/Free
Exposed Procedure
Initialize Program Process
Exposed Procedure
Pre-Display
SCR1
Exposed Procedure
Process
SCR1
Exposed Procedure
Pre-Display
SFL1
Exposed Procedure
Process
SFL1
Subr X
Subr Y
Subr Z
Model Component in Java or C#
Exposed Method
Initialize Program Process
Exposed Method
Pre-Display
SCR1
Exposed Method
Process
SCR1
Exposed Method
Pre-Display
SFL1
Exposed Method
Process
SFL1
Func X
Func Y
Func Z
Database I/O Components
Java/C# JDBC/ODBC
Database
DDS
Database
SQL/DDL

By using X-Migrate (formerly X-Redo) as a final processing step, the restructured RPG code may be converted to Java or C#. Additionally, the database may be converted from a DDS-based definition to a SQL-based definition, so that it may be migrated to another platform, if that is IT's chosen direction.

By following this path of language conversion, several important advantages are realized:

- The resulting code is organically suited to the new language; both Java and C# are object-oriented and most commonly used in an MVC design pattern.

- Developers on either side of the conversion, the RPG developers, or the Java/C# developers, are able to make sense of the conversion when comparing functionality and code specifics, when questions or problems arise.
- A bridge between existing staff skills and the old application, and new staff skills and the new application is built into the conversion process. Because the before and after components are both symmetrical and natural to the new language, existing application domain knowledge is easily applied to the new code.