X-Modernize

Fresche’s Application Modernization module automatically converts RPG Applications to Java

What’s Inside

A description of X-Modernize, the heart of the X-Analysis Application Modernization Module. X-Modernize automatically converts your RPG application to Java.
X-Modernize

Design Recovery: The basis for modernization

Whatever the approach to modernization, design recovery is the first step. With this understanding, developers can quickly identify the business rules and reusable designs embedded in core business processes. They can also restructure code, remove dead code and create re-usable components that can be enabled as services within a service-oriented architecture (SOA) or any modern application architecture.

Design Recovery is very valuable for documentation and application support purposes, but the real benefits come when the recovered design can be used to modernize or re-develop a system. Reusing existing designs programmatically can provide a dramatic productivity gain in rebuilding an application.

Some Highlights:
- Creates MVC Design Model from recovered interactive program designs in RPG
- Screens/Functions
- Data Layer
- Business Logic
- Generates MVC code from recovered RPG designs as Java JSF/Angular JS, Hibernate, Spring, DAO
- Works over user-defined application areas or individual programs
- Compares Original and Recovered code
- Transforms RPG batch logic into Java
- Creates Hibernate configuration files and JPA package from recovered relational model
- Fully automates and integrates documentation with X-Analysis
- Creates platform-independent UI/View-Controller meta-data (functions)
X-Modernize

Modernization Process

The modernization process includes two stages: Component Generation and Application Generation.

COMPONENT GENERATION

The Component Generation stage includes: 1. Database Modernization; 2. Business Logic Rebuild; 3. UI/Print Functions. Components are summarized at right, and described below.

1. Database Modernization

The data model for an application as deduced by X-Modernize can be used to modernize the database and database access as well as provide valuable information for analysis and documentation. Once you have a modernized database, you gain a number of advantages:

- Openness/Standards compliance: Using industry-standard SQL means that many different tools and applications on multiple platforms can easily access and use your modernized database.
- Portability: Because the database is defined in purely SQL terms rather than in a proprietary file format, it becomes portable — i.e., it is now viable to consider moving the database to another platform.
- Improved performance: IBM’s data retrieval efforts have been concentrated on SQL access rather than file-based access for many years.
- Ability to use modern Object Relational Mapping (ORM) software such as Hibernate for rapid application development in Java and other modern languages.
- Reduced dependency on System i-specific skills such as DDS, which may lead to cost savings and reduced risk.
- Data Integrity: Journaling is available for SQL access just as it has always been for file-based access. Constraints and referential integrity can be implemented directly at the database level where they are unavoidable rather than at the program level. Database triggers allow code to be run before or after records are added, updated or deleted, providing an easy way of enforcing compliance, audits, validations and applying business rules.
- Database modernization also uses the advanced design extraction directly from the RPG code, making use of long field names, constraint logic and all relevant database abstractions, either kept on IBM i or migrated onto other DBMS systems.
2. Business Logic Rebuild

After recovering the Data Model, the next step is to extract the logic that gives the application its particular characteristics. The generic term for such logic is Business Rules. The challenge is to extract or “harvest” these rules from the code.

Once harvested, these rules need to be narrated and indexed, thus providing critical information for any analyst, architect or developer charged with rebuilding an application. The task of harvesting business rules manually is a highly skilled, labor-intensive, and costly exercise for any organization.

X-Modernize accomplishes this task by automatically scanning the RPG and COBOL programs programmatically. It then separates out rule code from the body of the application and identifies, indexes, narrates, and stores business rule logic code into a structured, usable repository. In the final part of the process, it supplies appropriate textual narratives to describe these harvested rules.

Once the rules are derived, they can be viewed in summary form, as shown above at right.

The business rule repository can then be used programmatically to generate new code (migrated logic - see at right). At the same time, the built-in documentation, which cross references where-used objects and annotates capabilities, may be used by new developers as the necessary input for re-specification exercises, whether for new applications or for modifications to the current system.
3. UI/Print Functions

The screens of an older application are a classic example where the design is useful in a modernization context, and the code is not. The sheer number of screens in a large application presents a logistical problem in recreating them manually. X-Modernize lets you see what the original screen looked like without having to run the application, which is a great time saver for people not involved with the original application.

- View individual screens without running the application
- Design new screens in a GUI designer
- View the code at any time

Example code snippet:

```java
public void validateScreenFields(PanelDataObject zS, EzEditHospitalEditRecord1ScreenExt zStateVariable)
    Integer zmsgidx = 0;
    String zmsgid = "";
    String zmsgsubs[] = initArray(String.class, 21);
    String zmsgfld = "";
    getPunData(zMessages, zmsgidx, zStateVariable);
    stateVariable.setZdefer("N");
    stateVariable.setZtrncode(Ztrncode(Blanks(7)));
    if (equal("RUS", stateVariable.getCountry())) {
        if (compareTo(stateVariable.getAddressPostal2()))
            if (compareTo(stateVariable.getCountry()))
            if (compareTo(stateVariable.getAddressPostal2()))
            else if (equal("UK", stateVariable.getCountry()))
    else {zmsgid = "USR00864"; zmsgsubs[1] = stateVariable.getAddressPostal2(); zmsgfld = "addressPostal2"; zmsgidx = sendErrorMessage(zMessages, z); zmsgidx = sendErrorMessage(zMessages, z);
```
X-Modernize

Modernization Process - Cont’d.

3. UI/Print Functions - Cont’d.

Screen designs of older applications are not just about look and feel; there are attributes and embedded logic that are relevant no matter what technology is used to implement them in a modernized system. Among these attributes are:

- Formats/Layouts - Some screens may benefit from amalgamation or redesign, but table edits and non-transaction type screens will largely remain the same, even if not identical in layout.

- Actions - Whether from sub-file options, command keys, or default enter actions, these often represent an important part of the usefulness of an application design. The mechanisms used to offer or invoke these calls may change, but where they go logically and what parameters they pass will largely remain consistent.

Rebuilding the UI

X-Modernize extracts User Interface design information as described above and stores it as metadata in the X-Analysis repository. This is used as reference documentation for rebuilding UIs manually or for programmatically regenerating new View and Controller artifacts in the new technology. X-Analysis currently generates a JSF / Facelets UI version or a newer AngularJS UI with RESTful Web Services. The design metadata can also be used to generate new interfaces.

Why AngularJS?
The oldest and most mature of all JS MVC frameworks, AngularJS is managed by Google and is best suited for thick client business Web applications. It is built around best design principles and supports scalability, maintainability and modularization.

_X-Modernize_ Extracts User Interface Design Information as Described Above and Stores It as Metadata in the X-Analysis Repository. This is Used as Reference Documentation for Rebuilding UIs Manually or for Programmatically Regenerating New View and Controller Artifacts in the New Technology. X-Analysis Currently Generates a JSF / Facelets UI Version or a Newer AngularJS UI with RESTful Web Services. The Design Metadata Can Also Be Used to Generate New Interfaces._
X-Modernize

Modernization Process - Cont’d.

APPLICATION GENERATION

The modernized Web application based on MVC architecture can be generated after all the components are available. X-Modernize generates the sort of application that would have been written by hand by a professional application architect and team of coders without losing the essence of the original design or functional value. This is all done automatically into the IDEs of the language chosen by the user during generation. The new application is easier to enhance and maintain, and as such naturally attracts a wider, more-readily available resource pool for future development and maintenance.

Rebuilding the UI - Cont’d.

X-Modernize helps you generate screens in the format that fits your workflow.

RPG App

5250 DDS Screens

RPG Business Logic

Data Model

X-Analysis Repository

Extracted Functions

Refactored Business Logic/Rules

Explicit Data Model

View

Controller

Extracted Business Logic Class

ORM

Generic Data Base I/O Classes

Modern Database

Change Hospital Detail

Hospital Code: HOSP2

Hospital Name: Blue Cross Hospital

Street Address: 9th Street

Town: New York

State/Prov/Count: NY

Postal or Zip Code: KT13 9BY

Country: UK - United Kingdom

Telephone Number: 344-444-4444444

Fax Number: 123-456-7890

Email Id: someone@example.com

Edit pop-up window
X-Modernize

Modern and Maintainable Code Base

Because X-Modernize converts code by first converting the RPG to RPG Free Format in an MVC/OO/RESTful pattern, this intermediate code can also be viewed and compared to aid in understanding how the conversion took place. The Java code is then generated from the intermediate code as MVC JEE JSF / AngularJS, Spring/Hibernate application code base using modern coding standards for optimum maintainability.

Characteristics of the converted code

- MVC design pattern – Model-View-Controller, as in a Model layer of components that contains business logic, a View layer that contains the outward presentation, and a Controller layer that handles events in the other layers and directs process flow.

- Object Orientation – organization of code into objects (classes), and those classes containing functions. The functions are either callable from other classes, or protected so they can only be called from within their own class.

- RESTful interaction – the most important point about REST is that server components have no inherent knowledge of session state. Session information that needs to be preserved between work flow activities (screens) is preserved and represented from client-side memory, or via session management functions within the application server software.

Source List of TS/1E1RA in A2ED1M0/QRPGLESRC, Lines: 620, View Level: 5

```
if (equal("USA", stateVariable.getCountry())) {
  if (compareTo(stateVariable.getAddressPostZip(), "1") > 0 && comp)
    {   
      zmmsgid = "USR0034";
      zmmsgb[1] = stateVariable.getAddressPostZip();
      zmFld = "addressPostZip";
      zmldx = sendErrorMessage(zmMessages, zmmsgid, zmFild, zmmsgb[1]);
    }
}
else if (equal("UK", stateVariable.getCountry())) {
  if (compareTo(stateVariable.getAddressPostZip(), "1") > 0 && comp)
  {   
      zmmsgid = "USR0035";
      zmmsgb[1] = stateVariable.getAddressPostZip();
      zmFld = "addressPostZip";
      zmldx = sendErrorMessage(zmMessages, zmmsgid, zmFild, zmmsgb[1]);
    }
}
else if (equal("US", stateVariable.getCountry())) {
  if (compareTo(stateVariable.getAddressPostZip(), "1") > 0 && comp)
  {   
      zmmsgid = "USR0036";
      zmmsgb[1] = stateVariable.getAddressPostZip();
      zmFld = "addressPostZip";
      zmldx = sendErrorMessage(zmMessages, zmmsgid, zmFild, zmmsgb[1]);
    }
}
```
At A Glance...

X-Modernize
The complete, automated solution for conversion to Java.

X-Modernize Summary of Functionality
- Creates MVC Design Model from recovered interactive program designs in RPG
  - Screens/Functions
  - Data Layer
  - Business Logic
- Generates MVC code from recovered RPG designs as Java JSF/Angular JS, Hibernate, Spring, DAO
- Works over user-defined application areas or individual programs
  - Works over user-defined application areas or individual programs

Fresche Solutions — www.freschesolutions.com
Canada/Corporate Office: 995 Wellington, Suite 200 Montreal, CAN, H3C 1V3
North Carolina: 20 Fall Pippin Lane, Suite 202 Asheville, NC, 28803
British Columbia: 101 - 9724 4th St. Sidney, BC CAN, V8L 2Y7
Massachusetts: 124 Grove St., Franklin, MA 02038, USA
North Carolina: 20 Fall Pippin Lane, Suite 202 Asheville, NC, 28803
British Columbia: 101 - 9724 4th St. Sidney, BC CAN, V8L 2Y7
Massachusetts: 124 Grove St., Franklin, MA 02038, USA

- Compares Original and Recovered code
- Transforms RPG batch logic into Java
- Creates Hibernate configuration files and JPA package from recovered relational model
- Fully automates and integrates documentation with X-Analysis
- Creates platform-independent UI/View-Controller meta-data (functions)
- Builds Java data migration package from legacy data model

India: Atrauli, Gaurabagh, P.O. Gudumba, Kursi Road, Lucknow 226026, UP, INDIA
Australia: 9/622 Ferntree Gully Road Wheelers Hill VIC 3150, Australia
Ready to Learn More?

X-Analysis products are available in a variety of configurations. At Fresche Solutions, we work closely with you to assess your needs and recommend the best solutions. To get started, contact us using the information below:

www.freschesolutions.com | info@freschesolutions.com
1.800.361.6782 (Worldwide)

X-Analysis Advisor is the main offering in the X-Analysis suite, which includes productivity and modernization tools for your IBM i applications and databases. Following is a brief description of the solutions in the suite:

Understanding and Impact Analysis: An IBM i analysis tool that provides automated online documentation and powerful impact analysis.

Business Process Mapping and Metrics: A complete IBM i analysis tool, providing automated documentation, impact analysis, business rule extraction and code quality metrics.

Data and Test Management: Analysis of data quality; data archiving, data subsetting and data anonymization. Test data automation and management.

Field Resizing: Automated resizing of all occurrences of a field throughout your entire application environment; includes detailed problem analysis.

CA 2E Analysis: Everything required to analyze and document CA 2E applications. Includes business rule extraction and code quality auditing.

Application Modernization: RPG, COBOL and CA 2E (Synon) automatically converted to Java.

Database Modernization: Automated conversion of DDS to DDL, including creation of constraints, long field names and views.

Open Systems Analysis: Cross-referencing and documentation of Java, C#, PHP, and other languages.

About Fresche Solutions

Fresche is the leading provider of automated digital transformation enablement for companies who rely on IBM i systems. Through its transformation framework (tools, processes and methodologies), Fresche delivers high quality application modernization as a service (MaaS) in addition to a wide range of optimization solutions.

With over 400 digitally connected colleagues around the world and an extensive network of over 200 business partners, Fresche collectively brings clients the best solutions to drive innovation and IT success. For more information about our company, visit us on the Web at www.freschesolutions.com