MANAGEMENT OVERVIEW

A GUIDE TO THE BENEFITS OF USING DESIGN RECOVERY & REBUILD

Design Recovery & Rebuild with X-Analysis

Databorough Ltd.
66 York Road • Weybridge • UK
info@databorough.com
Phone +44 (0)1932 848564 • Fax +44 (0)1932 859211

The objective behind the true modernization exercise is to extract the essence or design of the legacy application and reuse these designs as appropriate in rebuilding the application, using modern languages, development tools, and techniques, tapping into more widely available skills and resources. In this Overview document we shall see how the Design Recovery & Rebuild tool set helps in rebuilding an application.

Benefits

The Design Recovery & Rebuild tool set offers following benefits:

- Automatically construct cleanly designed new applications from the extracted UML & Business Logic
- Generates new projects in IDE's such as MyEclipseBlue or Rational / WDSc
- MVC generation from recovered designs as JSF/Java or JSF/EGL
- Generates source objects from recovered Business Rule Logic & integrates with JSF's
- Reuses existing stored procedures & assists in creating new ones
- Creates Hibernate configuration files from recovered relational model
- Maps Hibernate configuration to new UI's
- Web Service Generation
- Compare Original and Recovered code

And many more...

Recovering an Application Design

The concept of reusing existing code or logic is not a new one. The challenge has always been to identify, isolate, and reuse only those designs that are relevant in the new context in which they are desirable. The sheer volume of code, its complexity, and the general lack of resources to understand legacy languages, specifically RPG, represents a tragic potential waste of valuable business assets. In many cases, these expensive and well-established legacy designs have little chance of even having their relevance assessed, let alone being reused. The Design Recovery Solution Set of X-Analysis addresses this problem more directly, by isolating, indexing, and documenting those design elements that could be relevant in a modern version of the application being assessed.

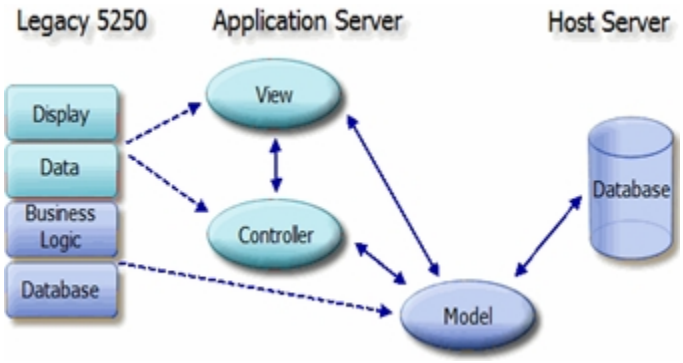


Illustration 1: Legacy versus modern architecture

Modern applications are implemented with distributed architecture. A popular standard used for this architecture is MVC or Model-View Controller. Figure given below display a typical legacy and MVC architecture side by side. MVC allows for independent implementation and development of each layer, and facilitates OO techniques and code re usability rarely used in legacy applications. All these characteristics of a modern application radically improve the maintainability and agile nature.

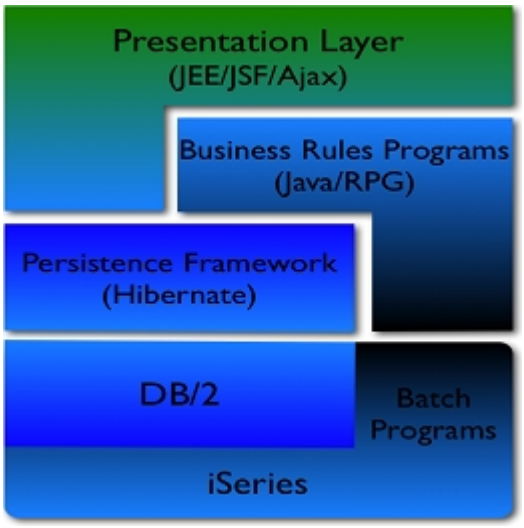


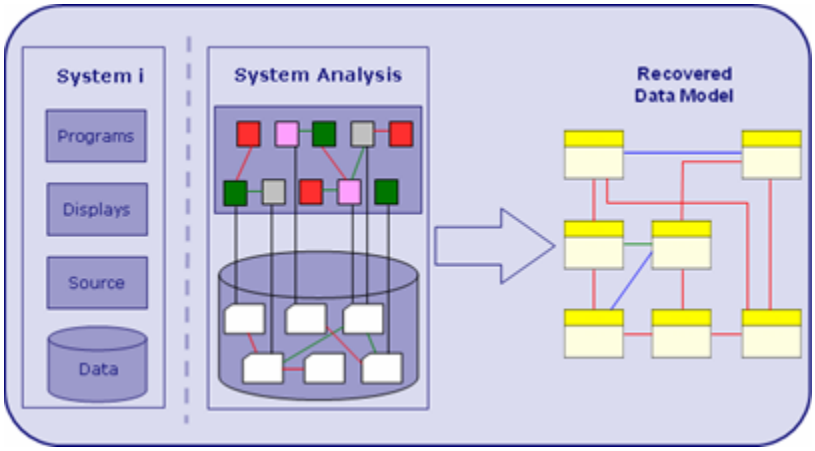
Illustration 2: Modernized Architecture with X-Analysis

Legacy applications do have these same elements, but they tend to be embedded in and mixed up in large monolithic programs, with vast amounts of redundancy and duplication throughout. Implementing an RPG application using MVC requires that the business logic be separate from the user interface and controller logic. This can be implemented using a web interface for the view, with the controller logic written in a modern language that supports web interfaces such as Java, EGL or C#. The optimum modernization result is to reduce dependency on legacy languages as much as is possible, if not altogether. To achieve this recovered design assets are reused as input to redevelop the appropriate layer.

Recovering the Data Model

The relational model of an enterprise application is an extremely powerful piece of information and potentially valuable asset to the organization. Unlike 2E systems for almost all RPG or COBOL applications running on System i, there is no explicit data model or schema defined. By the term model, we are referring to the foreign key or relational model, not just the physical model of the database. The relational model or architecture of the database can be reused in a number of scenarios including:

- Understanding application architecture
- Data quality analysis - referential integrity testing
- Automated test data extraction, scrambling and aging
- Building BI applications or Data warehouses



The X-Analysis has unique capability of automatically deriving the explicit system data model from a legacy RPG, COBOL or 2E application. Let us have a look at this and the model reuse capability in a bit more detail.

Recovering the User Interface

The screens of a legacy application are a classic example where the design is useful in a modernization context, and the code is not. All modern IDE's provide powerful UI development tools. Modern UI standards and preferences for style and technology also vary from project to project. The sheer number of screens in a legacy application presents a logistical problem in recreating them manually. X-Analysis lets

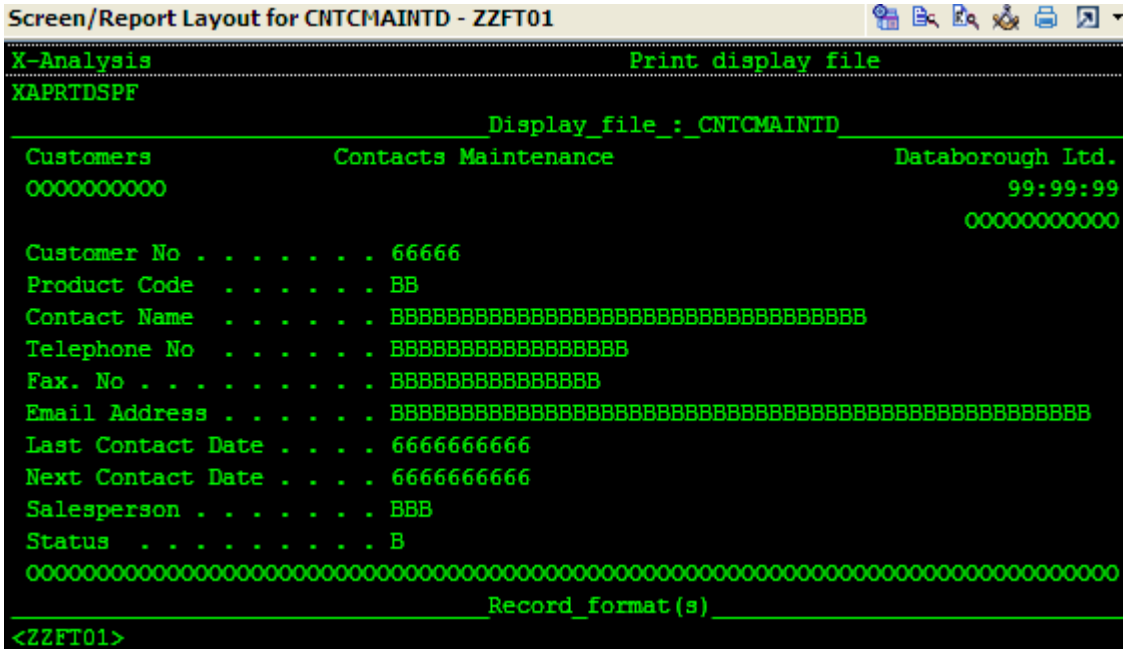


Illustration 3: Screen Design Layout in X-Analysis

you see what the legacy screen looked like without having to run the application which is a great time saver for people who haven't been involved with the original application:

Screen designs of legacy applications are not just about look and feel, there are attributes, and logic embedded which from a design point of view is relevant, no matter what technology being used to implement them.

X-Analysis extracts User Interface design information and stores it as meta-data in the X-Analysis repository. This is used as reference documentation for rebuilding UI's manually, or for programmatically regenerating new View and Controller artifacts in the chosen new technology. X-Analysis currently generates a JSF/Facelets UI version. The design meta-data can also naturally be used to generate new interfaces using any technology such as EGL, Ajax, RCP, C#, VB or even RPG.

Recovering Business Rule Logic

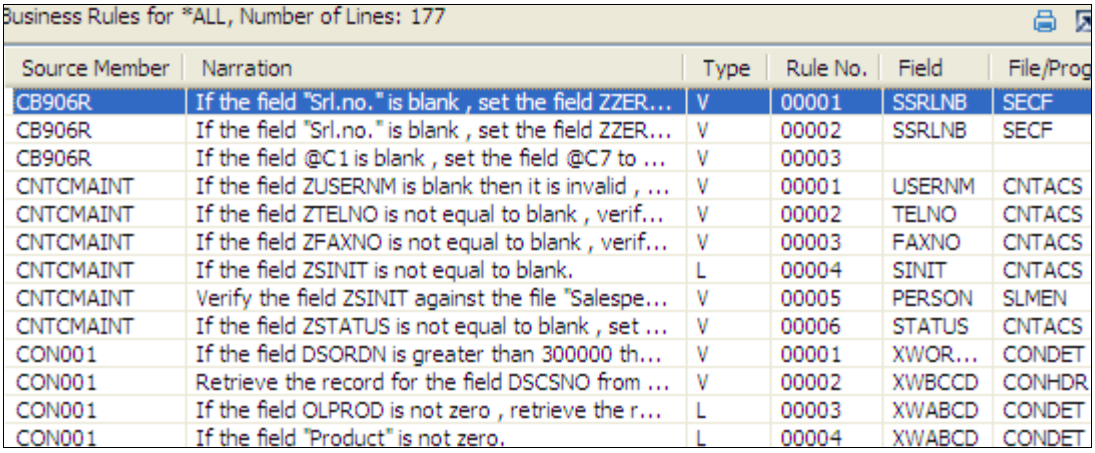
Once the system UI, data access & data model has been recovered & the application has been rebuilt or rewritten from this design, it is then necessary to extract the logic that gives the application its particular characteristics. The generic term for such logic is Business Rules. The challenge is to extract or "harvest" these rules from the legacy code.

The problem is that in the vast majority of legacy RPG and COBOL programs, the business rule logic is mixed in with screen handling, database I/O, and flow control. So harvesting these business rules from legacy applications requires knowledge of the application and the language used to implement it, both of which are steadily diminishing resource.

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

X-Analysis accomplishes this task by automatically scanning the RPG and COBOL programs and 2E model 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:


Business Rules for *ALL, Number of Lines: 177

Source Member	Narration	Type	Rule No.	Field	File/Prog
CB906R	If the field "Srl.no." is blank , set the field ZZER...	V	00001	SSRLNB	SECF
CB906R	If the field "Srl.no." is blank , set the field ZZER...	V	00002	SSRLNB	SECF
CB906R	If the field @C1 is blank , set the field @C7 to ...	V	00003		
CNTCMAINT	If the field ZUSERNM is blank then it is invalid , ...	V	00001	USERNM	CNTACS
CNTCMAINT	If the field ZTELNO is not equal to blank , verif...	V	00002	TELNO	CNTACS
CNTCMAINT	If the field ZFAXNO is not equal to blank , verif...	V	00003	FAXNO	CNTACS
CNTCMAINT	If the field ZSINIT is not equal to blank.	L	00004	SINIT	CNTACS
CNTCMAINT	Verify the field ZSINIT against the file "Salespe...	V	00005	PERSON	SLMEN
CNTCMAINT	If the field ZSTATUS is not equal to blank , set ...	V	00006	STATUS	CNTACS
CON001	If the field DSORDN is greater than 300000 th...	V	00001	XWOR...	CONDET
CON001	Retrieve the record for the field DSCSNO from ...	V	00002	XWBCCD	CONHDR
CON001	If the field OLPROD is not zero , retrieve the r...	L	00003	XWABCD	CONDET
CON001	If the field "Product" is not zero.	L	00004	XWABCD	CONDET

Illustration 4: Business Rule Summary

Or embedded in the code from where they are derived (Embedded Rules):

```
Seq No  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+...
0169.00        C* Telephone number
0169.95     =!BRC*  If the field ZTELNO is not equal to blank , verif
0169.96     =!BRC*  against ' 0123456789'. If FOUND then the field "B
0169.97     =!BRC*  the field VALID to off.
0169.98     %!BRC*  Business Rule No. XAN4CDEM/QRPGLESRC/CNTCMAINT/17
0169.99     @!BRC*  V00002  CNTACS      TELNO       The telephone no.
0170.00    B>!BRC                   if        ztelno <> *blanks
0171.00    ->!BRC     ' 0123456789' check     ztelno        z1
0172.00    ->!BRC                   if        %found
0173.00    ->!BRC                   eval      *in34 = *on
0174.00    ->!BRC                   eval      msgid = 'OEM0014'
0175.00    ->!BRC                   callp(e)  rtnmsgtext(msgid:errorm
0176.00    ->!BRC                   eval      valid = *off
0177.00    ->!BRC                   leavesr
0178.00    ->!BRC                   endif
0179.00    E>!BRC                   endif
```

Illustration 5: Embedded Rules Program

The business rule repository can then either be used programmatically to generate new code, or the built-in documentation, cross referencing where-used and annotation 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.

UML Diagramming

The objective of UML diagrams in this context is to help sketch application designs and to make such sketches portable and reusable in other IDE's such as Rational, Borland, MyEclipse, etc. The three diagrams automatically generated by X-Analysis are: Activity diagram, Use Case diagram and Class diagram.

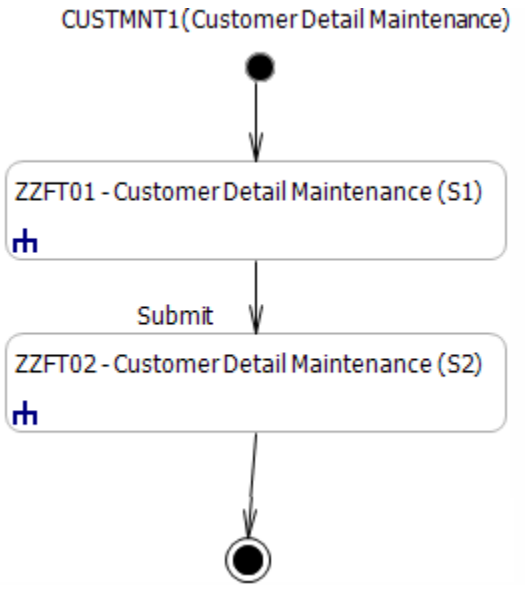


Illustration 6: Activity Diagram for a Program

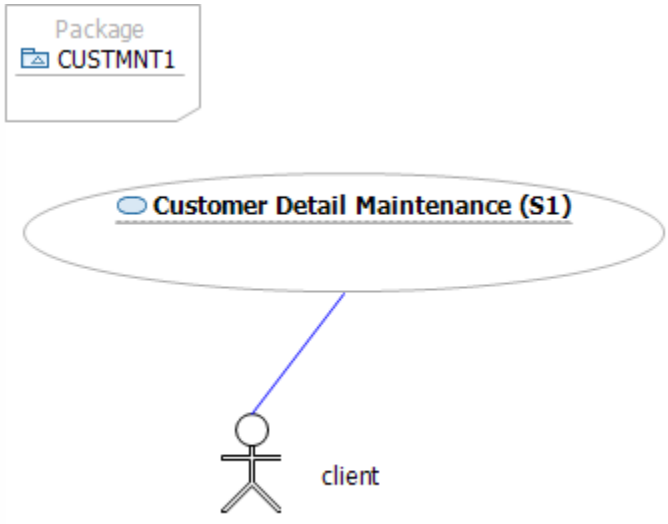


Illustration 7: Use Case Diagram for a Program

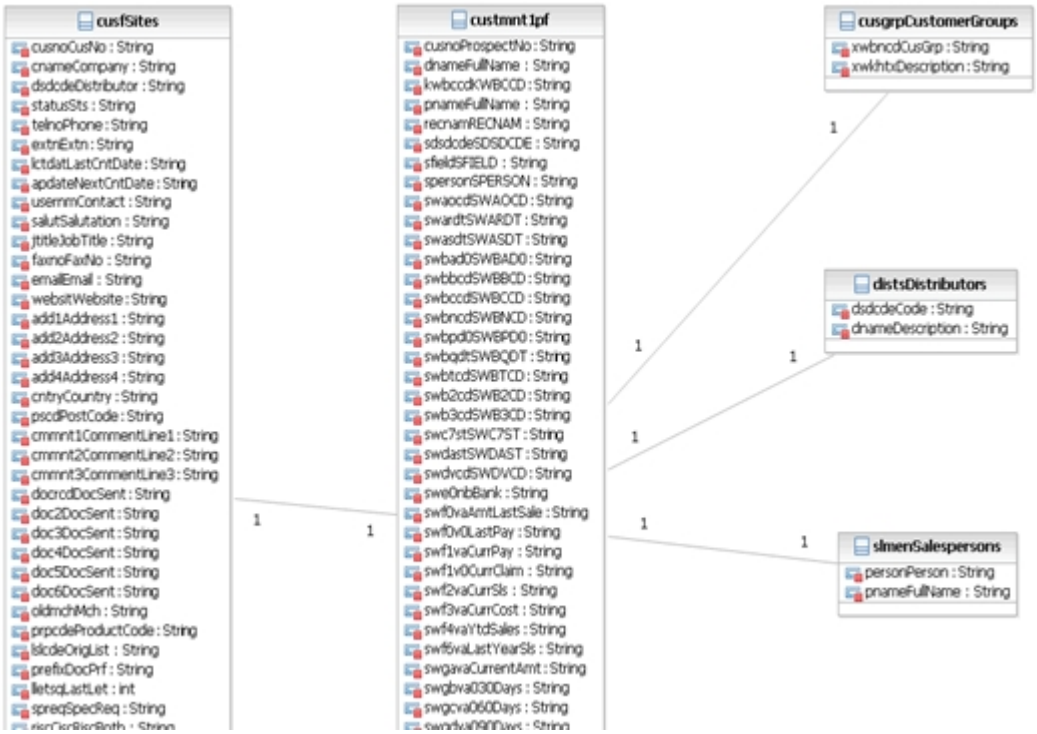


Illustration 8: Class Diagram for a Program

Producing any of these diagrams from within X-Analysis is as simple as right-clicking on an object and selecting the appropriate option from the menu.

Using Design Recovery for Rebuilding

Whilst Design Recovery is very valuable for documentation and application support purposes 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. While legacy application designs in their entirety might not suit a modern application implementation, design components are often suitable, as long they can be re-used at a sufficiently high level without introducing complexity to the redeveloped application. Therefore, being able to select and enhance, or ignore these at a granular level, removes the inheritance of irrelevant or legacy-specific code constructs, and allows direct access to elements that might have otherwise been deemed unusable in their current form.

Another important factor in this scenario is the ability to choose an implementation technology or language that suits the technology constraints or resources specific to a region or organization. The next sections cover how we can use the recovered application design in different ways to effect varying levels of modernization and re-development.

Database Modernization - using the Data Model Assets

Whilst it has always been possible to access System i data in a relational database like fashion there was originally no way of defining your database in the traditional relational database form with a schema or model. This has meant that most System i applications don't have an explicitly defined relational database

schema or model.

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

- Ability to use modern Object Relational Mapping (ORM) software such as Hibernate for rapid application development in Java and other modern languages.

- Because the database is defined in purely SQL terms rather than in a proprietary file format it becomes portable i.e. it is now an option to consider moving the database to another platform.

- Openness and Standards compliance using Industry standard SQL means that many different tools and applications on multiple platforms can easily access and use your modernized database.

```
                                                                  CUSF
A..........T.Name++++++RLen++TDpB......Functions++++++++++++++++++
********** Beginning of data *************************************
A           R RCUSF
A             CNAME          34A         TEXT('Company')
A             DSDCDE          2A         TEXT('Distributor')
A             STATUS          1A         TEXT('Status')
A                                        COLHDG('Sts')
A             TELNO          17A         TEXT('Phone')
A             EXTN            6A         TEXT('Extn.')
A             LCTDAT          6S 0       TEXT('Last Contact Date')
A                                        COLHDG('Last Cnt' 'Date')
A                                        EDTCDE(Y)
A             APDATE          6S 0       TEXT('Next Contact Date')
A                                        COLHDG('Next Cnt' 'Date')
A                                        EDTCDE(Y)
A             USERNM         34A         TEXT('Contact')
A             SALUT          34A         TEXT('Salutation')
A             JTITLE         34A         TEXT('Job Title')
```

Illustration 9: DDS of a File

```
--  Generate SQL
--  Version:                    V6R1M0 080215
--  Generated on:               01/07/10 12:03:51
--  Relational Database:
--  Standards Option:           DB2 i5/OS

CREATE TABLE XAN4CDXAD1.CONTACTS_CNTACS (
  CUS_NO_ FOR COLUMN CUSNO DECIMAL(5, 0) NOT NULL DEFAULT 0 ,
  CONTACT FOR COLUMN USERNM CHAR(34) CCSID 37 NOT NULL DEFAULT ''
  PRODUCT_CODE FOR COLUMN PRPCDE CHAR(2) CCSID 37 NOT NULL DEFAUL
  PHONE FOR COLUMN TELNO CHAR(17) CCSID 37 NOT NULL DEFAULT '' ,
  FAX_NO_ FOR COLUMN FAXNO CHAR(15) CCSID 37 NOT NULL DEFAULT ''
  EMAIL CHAR(40) CCSID 37 NOT NULL DEFAULT '' ,
  LAST_DATE FOR COLUMN LCTDAT NUMERIC(6, 0) NOT NULL DEFAULT 0 ,
  NEXT_DATE FOR COLUMN APDATE NUMERIC(6, 0) NOT NULL DEFAULT 0 ,
  SALESPERSON FOR COLUMN SINIT CHAR(3) CCSID 37 NOT NULL DEFAULT
  STATUS CHAR(1) CCSID 37 NOT NULL DEFAULT '' )
  RCDFMT RCNTAC     ;

LABEL ON TABLE XAN4CDXAD1.CONTACTS_CNTACS
```

Illustration 10: SQL converted from DDS

- Improved performance as IBM's data retrieval efforts have been concentrated on SQL access rather than file based access for many years now

- Reduced dependency on System i specific skills such as DDS, which may led 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. Databases 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.

Rebuilding the View

We described how useful screen design information is extracted into Function Definitions in the X-Analysis repository. These function definitions are effectively input specifications for generating new UI's or Views. The X-Analysis Modernization Tool set actually uses the Function Definitions to automatically, generate JSF/Facelets and Java bean source for each recovered screen design.
EGL versions are also available for View/Controller generation, with future generation options for PHP, C#, and XAML, becoming available from Databorough and other Business Partners.

Actions from the function definitions translate effectively into links on the generated JSF/Facelets, and these can be implemented with tab, buttons, or any appropriate UI standard demanded by a project. The required logic to invoke these actions is placed in the appropriate methods of the generated Java Bean as described below.

Rebuilding the Controller

The Java bean that drives the JSF/Facelet has standard methods for Data I/O, navigational actions, and for using any residual services that might remain on the legacy server in RPG, COBOL or 2E. This JSF bean has standardized exit points and a set of standard parameters making maintenance and development more efficient and consistent.

A separate call bean is implemented for each transaction group or legacy service program. This call bean provides a standard interface to these re-engineered legacy RPG services, and therefore greatly simplifies the controller or JSF bean as it is often referred to. In the case of a set multiple JSF's that make up a transaction, the call bean also acts as a persistence manager for the transaction.

Reusing Business Rules

The optimum design objective is to move as much of the business rule logic into the Java as possible, thus reducing the dependency on legacy languages. The monolithic architecture of legacy applications produces significant amounts of redundant and duplicate validation and field or calculation logic type business rules. These need to be re-factored if the maintainability of the application maintenance is to be improved - a primary objective of modernization in the first place.

This means that code duplication & redundancy can be almost completely avoided in the modern application. Typically, the only logic recreated in UI specific classes or beans will be context specific calculations field logic such as calculating the value of the order line being captured, along with conditional display or navigation logic for some UI's These new beans/classes should therefore be fairly simple and easy to maintain by comparison to their legacy counterparts.

The fact that we can easily verify that business rules from the legacy system have been built into the new system provides a high degree of confidence in the new system and is important from a compliance and audit standpoint.

Highlights

- Fully automated and integrated documentation with X-Analysis
- Automatically construct cleanly designed new applications
- Generation of UML Activity, Use Case and Class Diagrams
- Creates XML Business Rule Model from recovered business rule logic
- Creates Transaction Model from screens, I/O, program actions/options, DB Model
- Can be used over user defined application areas or individual programs
- MVC generation from recovered designs as JSF/Java or JSF/EGL
- Generates source objects from recovered Business Rule Logic & integrates with JSF's
- Reuses existing stored procedures & assists in creating new ones
- Maps Hibernate configuration to new UI's
- Compare Original and Recovered code

The objective behind the true rebuilding exercise is to extract the essence or design of the legacy application and reuse these designs as appropriate in rebuilding the application, using modern languages, development tools, and techniques, tapping into more widely available skills and resources. The Design Recovery & Rebuild tool set offers simple way of rebuilding a complex legacy application.

Experience the fully loaded X-Analysis with 30 days trial copy of the software.
For any information regarding the X-Analysis please visit our web site:

www.databorough.com
or write e-mail to us at:
info@databorough.com

Databorough

Corporate Headquarters >

Databorough Ltd.
Weybridge Business Centre,
66 York Road,
Weybridge,
KT 129DY
United Kingdom
044-1932-848564
044-1932-859211
info@databorough.com
www.databorough.com

International Office >

Databorough Services
Suit# / Box# 504,
92 Caplan Avenue,
Barrie,
Ontario,
L4N 9J2
Canada
01705-458-8672
1800-605-5023 Toll Free
info@databorough.com
www.databorough.com