

# Application Modernization and Rebuilding with X-Analysis

Version 9

## Overview

*A guide to the benefits of using  
X-Analysis Modernization Toolset*

Databorough Limited  
2011



Application Modernization and Rebuilding with X-Analysis

## **Preface**

Developing tools and services for analyzing and reengineering applications for the last 20 years, has given Databorough a unique view of the very large and complex world of legacy applications running on System i. This management overview document focus on the Modernization practices (RPG/COBOL/2E) for System i applications using X-Analysis 8. It discuss new concepts and methods for design recovery and rebuilding of monolithic RPG/COBOL/2E applications into modern application architectures.

Contact [info@databorough.com](mailto:info@databorough.com) for a copy of the trial software.



## Table of Contents

<b>Why Modernize using X-Analysis</b> .....	<b>1</b>
<b>Introducing X-Analysis</b> .....	<b>3</b>
<b>Why Design Recovery is difficult</b> .....	<b>4</b>
<b>Analysis, Documentation &amp; Application Subdivision</b> .....	<b>6</b>
Understanding Design & Function More Easily .....	6
Producing Static Documentation Automatically .....	8
Application Subdivision .....	8
<b>Recovering an Application Design</b> .....	<b>10</b>
Recovering the Data Model .....	10
Recovering the User Interface .....	11
Recovering Business Rule Logic .....	11
UML Diagramming .....	13
<b>Using Design Recovery for Rebuilding</b> .....	<b>15</b>
<b>Database Modernization - using the Data Model Assets</b> .....	<b>16</b>
Rebuilding the View .....	17
Rebuilding the Controller .....	17
Reusing Business Rules .....	17
<b>Rebuilding Example</b> .....	<b>18</b>
Identify what you want to work with .....	18
Application Areas .....	19
Rebuild the Application .....	20
<b>Completing the Modernization Process</b> .....	<b>24</b>
Surrounding Application Framework .....	24
Look & Feel & UI Standards .....	24
<b>Highlights of Modernization Toolset</b> .....	<b>26</b>
<b>Summary</b> .....	<b>27</b>



## Why Modernize using X-Analysis

The knowledge and information contained in an organization's business software is vitally important and very valuable but often this information covering the operation, metrics, and design of the software is tantalizingly out of reach. Without this knowledge, maintenance and changes to the system are not as efficient or effective as they could be, and the risk of failure or problems increases exponentially the larger the enhancement required. This could lead to a paralysis where changes can't be made due to a lack of confidence in the outcome.

Accurate and current information about an entire system can greatly improve the productivity of your IT staff, and reduce maintenance costs by eliminating the need to research, catalog and assemble the information manually for each service request, or modernization project.

Existing System i applications whether they are COBOL, RPG or 2E have some fairly consistent and distinct characteristics that mark them as costly and potentially high-risk:

1. Applications tend to be large and complex
2. Little or no documentation
3. Original Designers no longer available
4. They have been developed over many years
5. Monolithic Programming Model
6. Written in obsolete languages

Points 1 through 3 can largely be managed more effectively by investing in a product like X-Analysis to both recover the design of the application, and provide highly productive analysis tooling to compensate for the complexity of the application.

Inconsistent programming standards and designs, significant amounts of redundant and duplicated code, and an increasingly costly demand for globally diminishing legacy development skills, are the results of points 4, 5 and 6.

This document will illustrate how X-Analysis carries out the Design Recovery process and how it can be used to build re-engineered applications from the Recovered Design. To understand the problem domain just think for a second of two approaches to the above problems namely screen-scrapers and code conversion.

Simply screen scraping the user interface with a GUI or web emulation product does not improve the situation, the application may appear slightly more modern but the cosmetic changes still leave it with all the same maintenance and enhancement issues.

The other common approach code conversion i.e. line by line, syntax conversion of a legacy application will typically transfer the same problems from one environment/language to another. Indeed, it will usually produce source code that is less maintainable, effectively canceling out the benefit of using modern technologies and architectures in the first place.

Removing problems 4 through 6 and thus achieving sustainable and effective application modernization can only truly be achieved with an application rewrite or rebuild. Automated component generation from recovered designs can dramatically reduce the costs of an application rewrite and without inheriting the legacy code's redundancy and complexity.

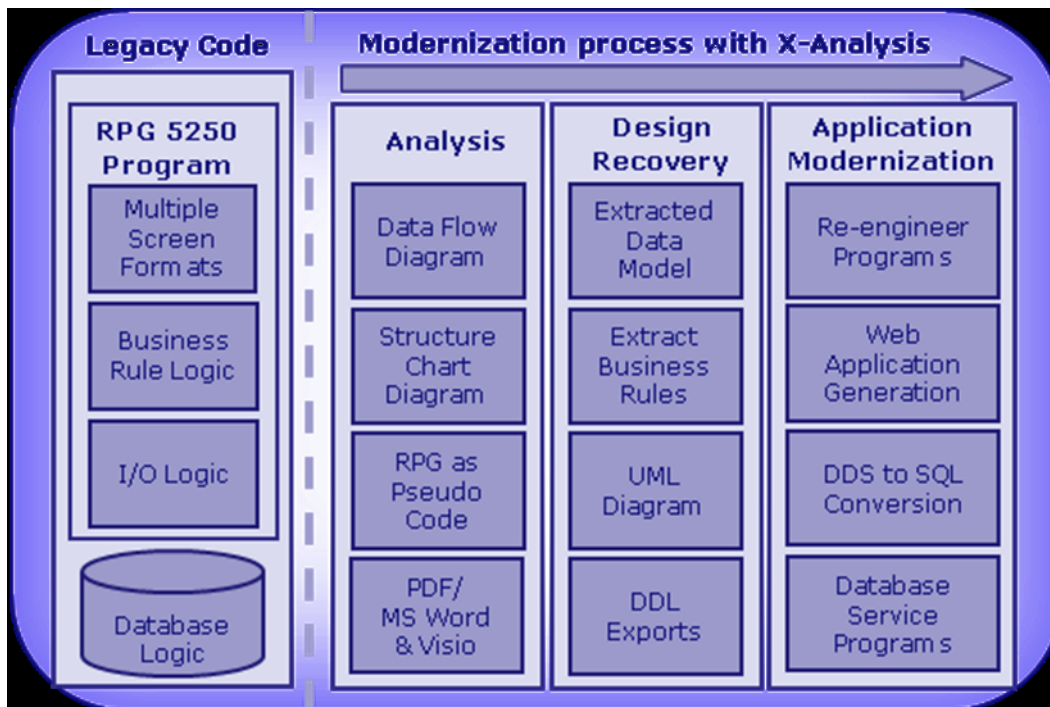


## Application Modernization and Rebuilding with X-Analysis

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 and restructure code, remove dead code, and create reusable components that can be enabled as services within a service-oriented architecture (SOA), or any modern application architecture.

The objective, therefore, is a true modernization exercise 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.

X-Analysis provides analysts, developers, architects and operations teams with detailed analysis and interactive diagrammatic constructs that enable rich understanding of existing applications, whether they were developed yesterday, or 30 years ago. Some companies may have a desire to keep a significant amount of design logic from the existing application design and move that to the modernized version of the application. For those situations, X-Analysis provides design extraction functionality, for automatically creating JEE & RPG industry standard modern components and constructs as exports from the recovered designs themselves. A legacy application component can be rewritten using a combination of the JSF, EGL, Facelets, and persistence frameworks such as Hibernate, all generated by top-down automation from the recovered X-Analysis model. Since each customer situation is potentially different, the X-Analysis suite is deliberately subdivided into modules that suit the appropriate development stage or budget constraints of each company.





## Introducing X-Analysis

As we have seen gathering knowledge about System i applications is not a straightforward task for today's generation of business analysts and developers. To illustrate that point and to fully understand the problem domain we will look at **Why Design Recovery is difficult** by working through the problems that X-Analysis solves in building its repository of design recovery information.

In situations where developers are not familiar with a system or its documentation is inadequate, the system's source code becomes the only reliable source of information. Unfortunately, source code has much more detail than is needed just to understand the system, also it disperses or obscures high-level constructs that would ease the system's understanding. X-Analysis aids system understanding by identifying recurring program features, classifying the system modules based on their purpose and usage patterns, and analyzing dependencies across the modules. This analysis provides detailed design information about the entire system, accessible to non RPG/COBOL experts, and be easily updated to incorporate ongoing changes in the base system.

Whatever the business needs driving companies to modernize their applications, most want to ensure that the business logic and functional design which are core assets to the company, are preserved to varying degrees. Design Recovery of an application can be broken down into a few logical steps or stages that represent a generic adaptable approach to any application modernization project:

**Analysis, Documentation, Application Subdivision** – This type of analysis represents the most common use of the X-Analysis tool across the world. On top of very powerful cross-referencing functionality, graphical, narratives or a combination of both, are used to abstract and describe the system in a simple and intuitive way, even for non-RPG/COBOL experts. The legacy application can be completely documented using modern diagramming standards such as UML, Entity Relationship Diagrams, System Flow Diagrams, and Structure Charts etc.

**Recovering an Application design** – This advanced level of analysis extracts model information from the existing application. X-Analysis uses its own analysis repository, plus pattern searching algorithms, to derive relational data models, extract business rules, build UML Activity/Use/Case Diagrams, and logical screen flows. The structured, repository-based format of these extracted designs, make it possible, to programmatically reuse them for rebuilding the core of a new application.

**Redeveloping Using a Recovered Application Design** – This starts with database modernization using the recovered data model. The designs for the view, controller, and business rule logic are also extracted and reused in modern frameworks such as Hibernate, and with new JSF/Facelets and Java bean components. This option makes it possible to programmatically refactor the existing application into modern, consumable assets and artifacts for developers to use for a system rebuild.



## Why Design Recovery is difficult

From the point of view of the user of X-Analysis, the process of building the cross-reference repository and deriving the models happens automagically! i.e. Its just there and happens typically as part of the installation process - though it can be triggered again later on if required.

If you think for a minute of a typical System i application it is likely to consist of a mix of RPG programs, DDS files and members for display files, database files and logical views, newer systems may have these interspersed with SQL scripts but the sum of knowledge in that system, how it works and interacts amongst its various elements is contained within those source files and compiled objects - the issue is retrieving that knowledge efficiently.

To understand and fully appreciate the problems X-Analysis solves just consider the process you would have to undertake yourself if you wanted to discover how a system operates or make changes to it. As a simple example for part of your application you have a customer details screen with no dedicated place for an email address and mobile phone numbers, the system has adapted itself to the internet age as many System i apps have done by making use of 'extra' and 'notes' ad-hoc fields. The system has coped but it has been time consuming to retrieve these details when required for marketing purposes. But there is now a budget to correct this and start to look at modernizing the application and making the functionality available to more areas of the business.

You would probably first start by looking at the program and display files that handle the display and maintenance of the customer information, from that you would discover the database tables/files involved.

At this point from a simplistic point of view you have the necessary information to make the changes and they are probably not that difficult - add new fields for email and mobile phone to the database tables or rename the existing ones then modify the program and display files accordingly... but you're probably thinking what about the rest of the system? What else uses that table? Is the display file used anywhere else? So the change has more aspects than would first appear these are just a few of the questions we have to answer:

- Scope and impact of the change - how many programs and tables are effected?
- Database changes - do we add new fields or just rename the fields and preserve the status quo? Do we know those fields were only used for email and mobile phone data?
- Database integrity - Fields destined for ad-hoc data like 'extra information' and 'notes' are unlikely to have any validation or to be even required so if migrating the existing values to new fields we can't simply copy it over some cleansing will be required.

The process of gaining the knowledge to answer these questions may not be all that straightforward, particularly if the systems are complex or the people trying to answer them are new to the application, system or platform.

To assess the scope and impact of the change you need to find out which programs use the files/tables affected, this can be very laborious:



## Application Modernization and Rebuilding with X-Analysis

- Go through all source files in PDM
- option 25 to searching
- then F13 to repeat
- press enter
- type in your search term
- review results

... and that's just the first enquiry! Depending on the complexity and history of your systems you may have doubts that you were looking at all of the source or the latest version.

Looking into Database integrity may well throw up items like this screen shot. Where we have a number of different formats of email address and some extraneous text, similarly on the phone number list there is text and a variety of layouts. Finally we have the inevitable result of using ad-hoc fields with no validation or on screen guidance - transposed data mobile in email and vice versa.

Notes Field (used for email address)	Extra Info Field (used for mobile phone number)
<a href="mailto:jbloggs78@aol.com">jbloggs78@aol.com</a>	cell: 7005896236
<a href="mailto:sherylc@msn.com">sherylc@msn.com</a>	07568 456 321
<b>HSMITH at SUN.COM</b>	+447890987852
email: <a href="mailto:jbooth@mac.com">jbooth@mac.com</a>	mobile 0754699888
<b>07678 678912</b>	<a href="mailto:suzip@btconnect.com">suzip@btconnect.com</a>

Hopefully this section raised awareness of the problems around changing and modernizing System i applications, the issues with finding out the necessary information and how seemingly straightforward issues can be time consuming and problematic. X-Analysis is designed and optimized to make the design recovery process as straightforward as possible as the rest of this concepts guide will illustrate.



## Analysis, Documentation & Application Subdivision

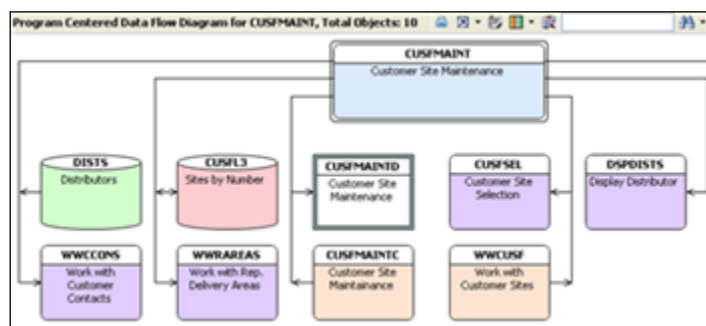
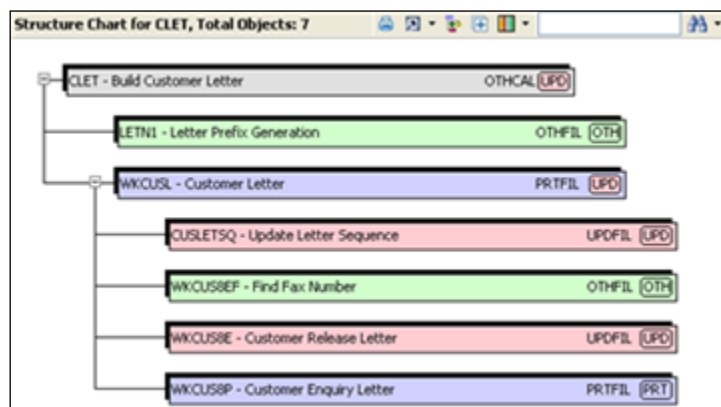
X-Analysis builds a detailed repository over an entire application. The repository maintains all information about application objects, their relationships and all necessary information to obtain detailed information from each object across the entire system. 20 years of ongoing development, over thousands of System i applications written in all variants of RPGII/400/IV, COBOL, 2E and CL, has produced an unmatched capability to extract everything about an application from object right down to individual variables. The repository is built automatically using a single command, and initially collects all object related information, but then parses every source member in the specified system and every source line mapping the contextual information of each variable in the system. A certain amount of logical abstraction processing then takes place while building the repository to account for some of the idiosyncrasies typical in an RPG application. This includes constructs such as variable program calls, file overrides, prefixing and renaming in RPG. The repository thus represents a map of how the entire application functions right down to individual variables.

### Understanding Design & Function More Easily

For efficient familiarization of an application's structure and general function, an abstraction above the source code combined with object-to-object relational information is required. A few simple but rich types of color-coded, graphical diagrams can reveal the data flow and architecture of individual objects or parts of an entire system. This is combined with automatically derived descriptions in the form of Pseudo narratives either in the diagrams or while browsing source code.

Here is a brief description of some of these diagrammatic constructs and views:

**Structure Chart Diagram – A Structure Chart Diagram (SCD) Display** gives a graphic representation of how the control passes from one program to another program within the application. This follows the call structure down the complete stack. The diagram also reveals data input objects and also automatically derives a summarized description of each of the object in the diagram. Color-coding also reveals important functional aspects such as updates, prints, and displays, which help the user to zone in on commonly sought after details.

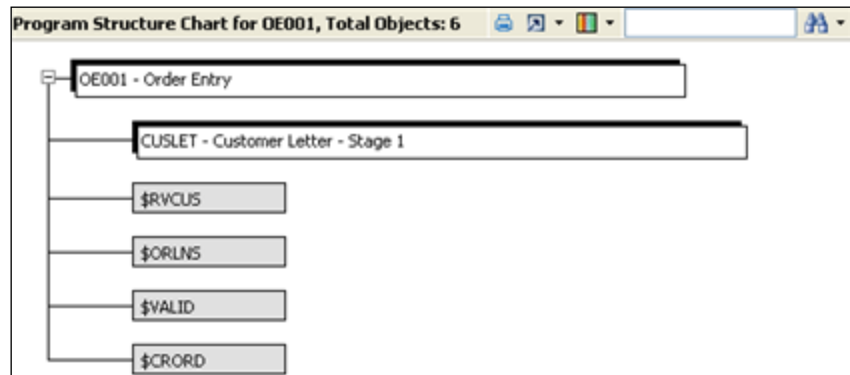




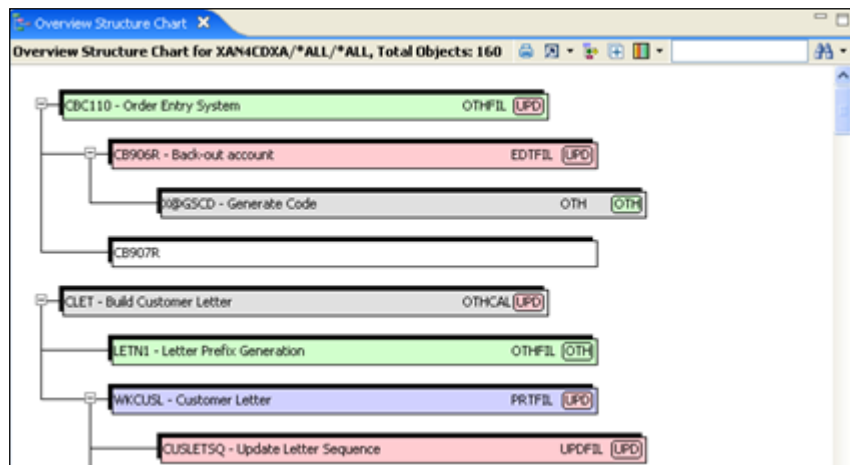
## Application Modernization and Rebuilding with X-Analysis

**Data Flow Diagram** – A Data Flow Diagram (DFD) is a graphical representation of a program/object where used, showing the files and programs accessed by the subject object. It is also color-coded and shows both flow of data at a high object level, and contextual information about the specific variables / parameters passed between objects.

**Program Structure Chart** – A Program Structure Chart graphically displays the sequence of calls in the program. The call could be to execute a Subroutine / Program / Module / Service Program.



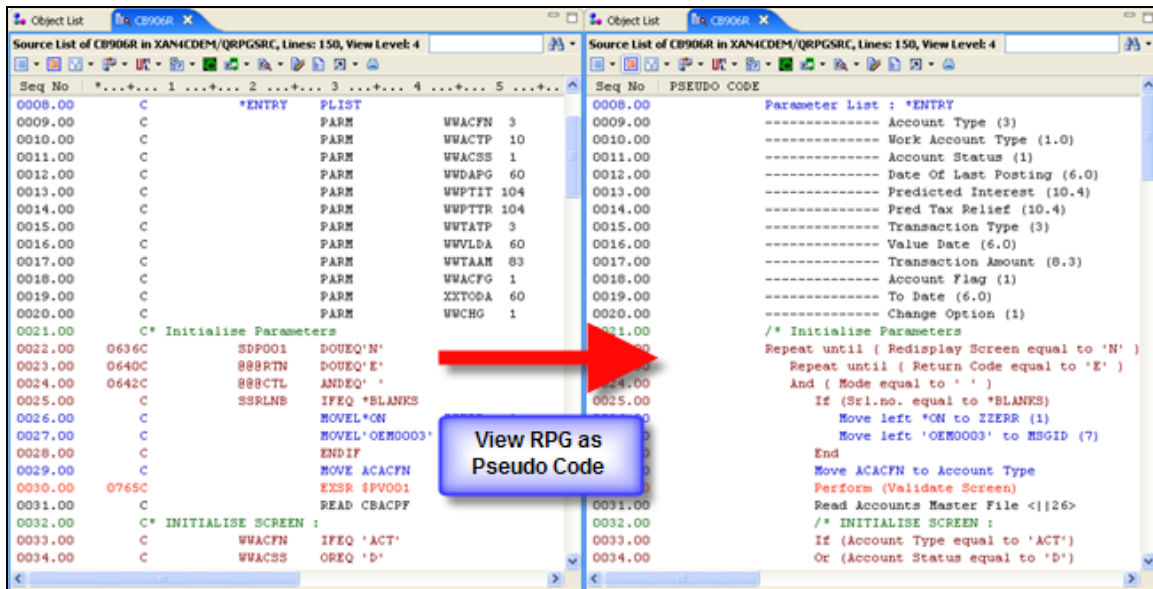
**Overview Structure Chart** – The Overview Structure Chart gives a snapshot of an application. It displays all the entry points to the application, and then the structure chart for each of these entry points.



**RPG as Pseudo Code** – With a single click, RPG can be viewed as a form of structured English or Pseudo code. Mnemonics' are substituted with file/field/variable texts and constants or literals.



## Application Modernization and Rebuilding with X-Analysis



### Producing Static Documentation Automatically

Interactive analysis via a graphical client is generally the most intuitive manner in which to analyze a system, but there is often a requirement for various types of static information in the form of structured documentation. Examples of this are project documentation, auditing information, testing instructions, and customer support documentation (such as with ISV supplied business software). X-Analysis produces a number of these outputs including:

**Data Flow Chart in MS Visio** – Any interactive diagram produced by X-Analysis 8 in the client, can be automatically exported to MS Visio

**Lists and Results sets** – Any source, object, or impact-analysis result list can be directly exported to formatted MS Excel or Word from the client.

**MS Word Project Documentation Wizard** – With the use of a simple wizard, documents that might take weeks to produce manually, allow the user to select any of the graphical diagrams, lists, flowcharts, annotation and business rules summaries generated interactively by the client interface, can be collated into a single document with contents and index. This can be done for a single object, an application area (explained below), a list of objects, or an entire system. Any of these documents can then be edited and distributed as required.

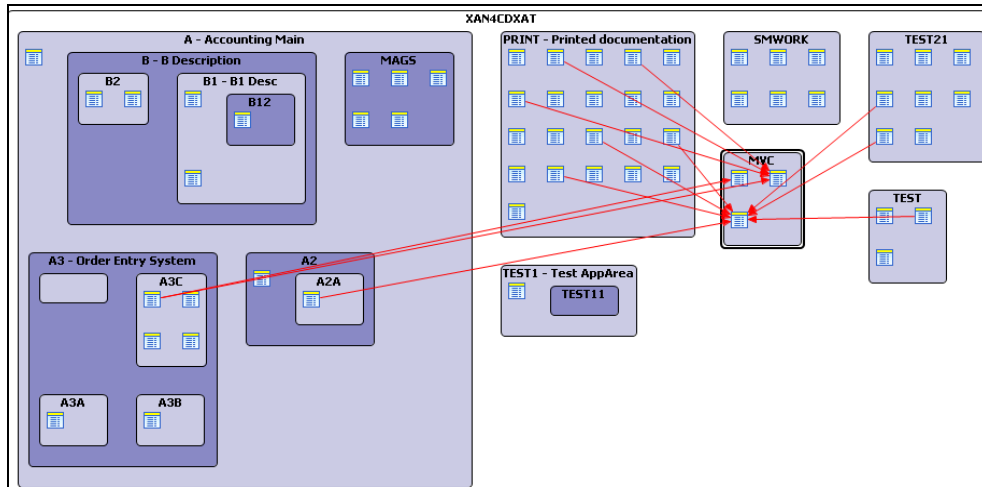
### Application Subdivision

Entire legacy applications are often too large to effectively comprehend or effect wholesale change. For this reason it is often necessary or helpful to sub divide a system into application areas. X-Analysis provides facilities for subdividing an application area into groups of objects that meet user defined selection criteria. Application areas filters can then be used through the X-Analysis Solution Sets to view, document or re-engineer as opposed to individual objects.



## Application Modernization and Rebuilding with X-Analysis

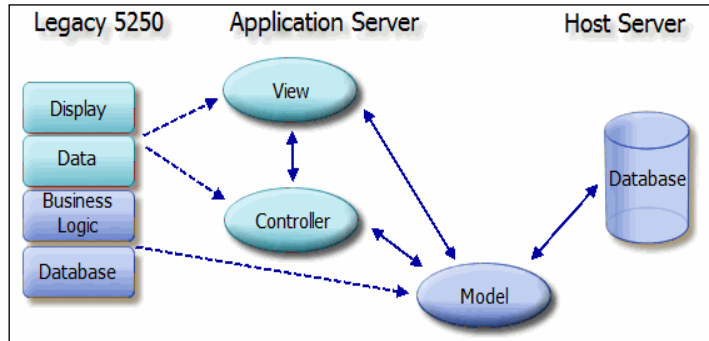
**Application Overview** – The Application Area Diagram displays an overview for the application. It is an interactive diagram and by clicking on different parts of the system you can see the relationships between either all parts or just the area you've clicked on and the areas it relates to.





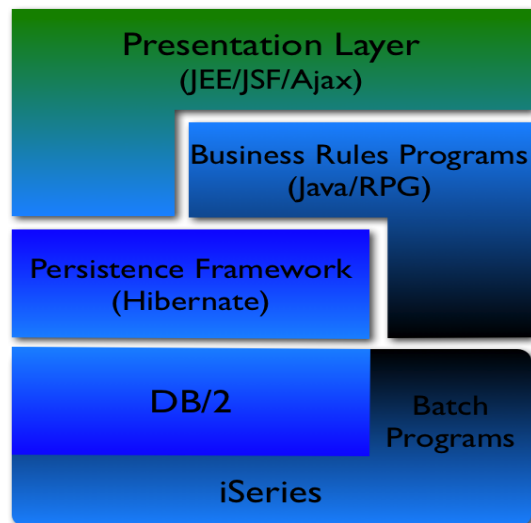
## Recovering an Application Design

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.



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 reusability rarely used in legacy applications. All these characteristics of a modern application radically improve the maintainability and agile nature.

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

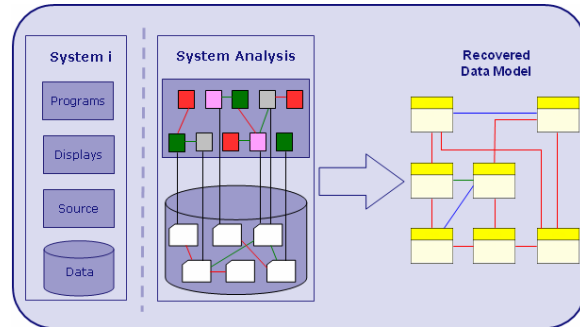


## Application Modernization and Rebuilding with X-Analysis

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

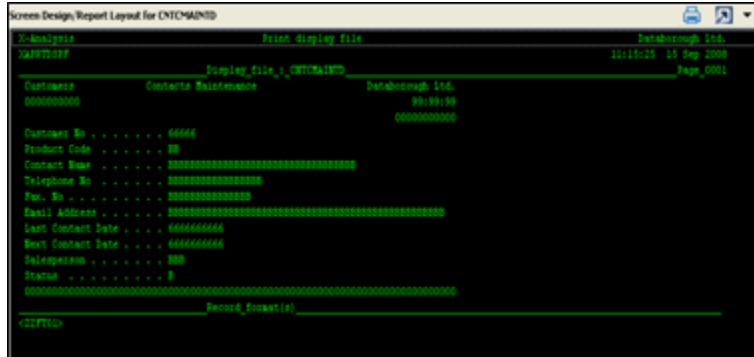
X-Analysis has the unique capability of automatically deriving the explicit system data model from a legacy RPG, COBOL or 2E application. Let us have a look at this and the model reuse capability in a bit more detail.



### 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 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.



## Application Modernization and Rebuilding with X-Analysis

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 Rule Summary):

Source Member	Narration
CUSTOMNT1	Call program 'WCUSTP' passing the field SWBCCD.
CUSTOMNT1	If the field SWBCCD is blank then it is invalid.
CUSTOMNT1	Verify the field SWBCCD against the file "Purchases". If on file then the field "Customer" is invalid.
CUSTOMNT1	If the field SWG4TX is blank then it is invalid.
CUSTOMNT1	Retrieve the record for the field SWBNCD from the file "Customer Groups". If not found then it is invalid.
CUSTOMNT1	Retrieve the record for the field SPERSON from the file "Salespersons". If not found then it is invalid.
CUSTOMNT1	Retrieve the record for the field SDSDCDE from the file "Distributors". If not found then it is invalid.
CUSTOMNT1	If the field "Prospect No" is zero then it is invalid.
CUSTOMNT1	Retrieve the record for the field "Prospect No" from the file "Sites by Number". If not found then it is invalid.
CUSTOMNT1	If the field SWBCCD is blank , set the field *IN96 to on. Otherwise:
CUSTOMNT1	Retrieve the record for the field SWBCCD from the file "Purchases". If not found then it is invalid.
CUSTOMNT1	If the field CUSPRM is not zero , set the field "Prospect No" to the field CUSPRM.
WWCONDET	If *IN99 is '1' then it is invalid , set the field ZMSAGE to the field ERRMSG.
WWCONDET	If *IN99 is '1' then it is invalid , set the field ZMSAGE to the field ERRMSG.

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



## Application Modernization and Rebuilding with X-Analysis

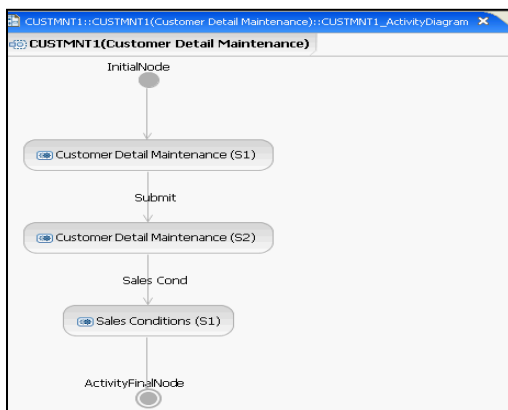
```
0169.00      C* Telephone number
0169.96      !=BRC* If the field ZTELNO is not blank , verify the field ZTELNO against '
0169.97      !=BRC* 0123456789'. If other values found then the field "Phone" is invalid.
0169.98      %!BRC* Business Rule No. XAN4CEM/QRPGLESRC/CNTCMAINT/170 Validation.
0169.99      @!BRC* V00002 CNTACS      TELNO      The telephone no. is invalid.
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:errmsg)
0176.00      ->!BRC                      eval          valid = *off
0177.00      ->!BRC                      leavesr
0178.00      ->!BRC                      endif
0179.00      E>!BRC                      endif
```

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.

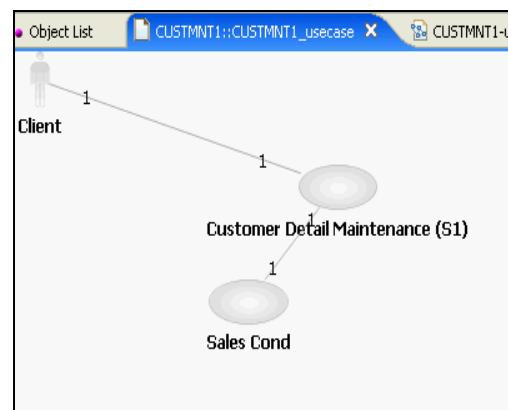
### UML Diagramming

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.

#### Activity Diagram



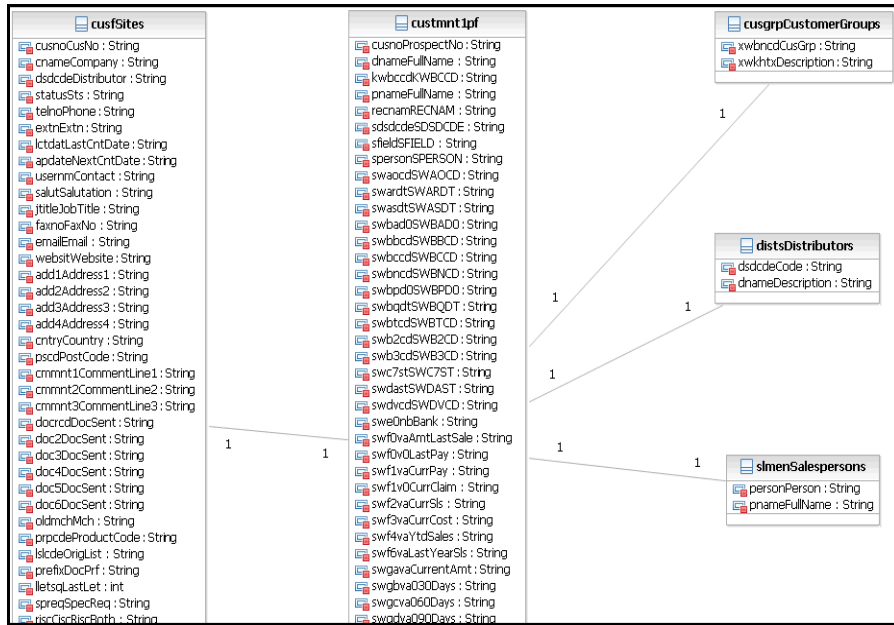
#### Use Case Diagram



#### Class Diagram



## Application Modernization and Rebuilding with X-Analysis



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.



Application Modernization and Rebuilding with X-Analysis

## 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.

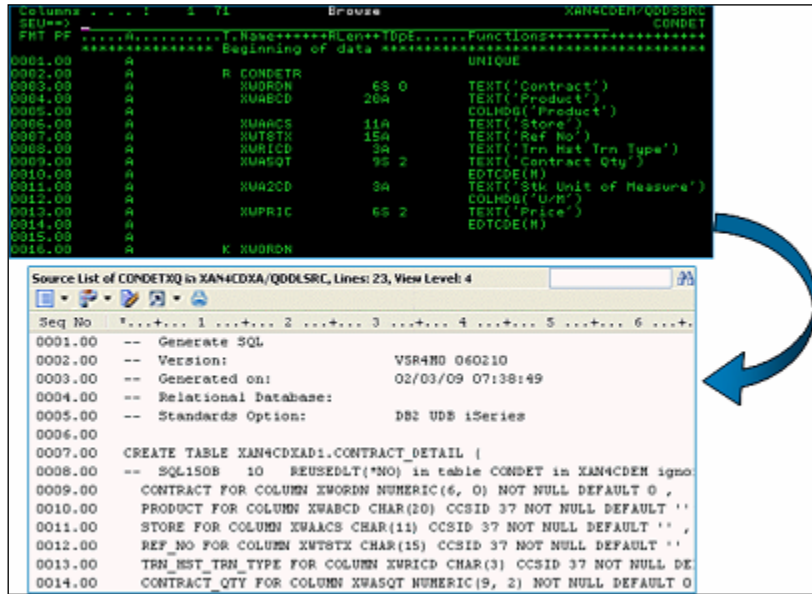


# Database Modernization - using the Data Model Assets

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
- 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.





## Application Modernization and Rebuilding with X-Analysis

### **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 Toolset 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 reengineered 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.



## Rebuilding Example

To illustrate how quickly you can begin the rebuilding process let's have a look at a typical iSeries application and rebuild part of it. The key steps are:

1. Identify the parts of the green screen that you want to work with
2. Use X-Analysis Application Areas to break out this part of the system
3. Rebuild the Application Area into your chosen architecture
4. Run the rebuilt application
5. Refine the rebuilt application , tailor to your requirements

### Identify what you want to work with

Here's some screens from our application first the work with customers

```
Customers                      Work with Customers                      Databorough Ltd.
WWCUSTS                          11:11:16
                                   2009/05/01

Position to: _____
Enter options, press Enter.
2=Change, 4=Delete, 5=Display, 6=Customer Maintenance, 8=Orders, 9=Trans.Hist.

  Customer                      Name
  _____                      _____
  ACC1                          Bertwhistle Ltd1
  ACC10                         Customer ACC10
  ACC2                          BOCK & CO. LTD
  ACC3                          BESSON BROS.
  ACC4                          LEDIA ENTERPRISES LTD
  ACC5                          Bays Engineering Ltd
  ACC6                          Gough Research plc
  ACC7                          Karst plc
  ACC8                          Bays Engineering Ltd
  ACC9                          NEWT FOODS UK LTD.
  ACC99                         Prime Computer Systems
```

We can view Orders



## Application Modernization and Rebuilding with X-Analysis

```
Customers                      Work with Orders                      Dababorough Ltd.
WWCONHDR                                                                11:13:08
                                                                2009/05/01

Customer.....: ACC1          Bertwhistle Ltd1
Representative.....: JKL Johan Klaassen
Telephone Number.....: 04895 580099
Current balance.....:      256254.00  Credit Limit.....:      1001235.00

Position to: _____      Enter options, press Enter.
                               2=Change, 4=Delete, 5=Display, 7=Lines

  Cntrac  Customer Reference  Date  St  Rep  Value
  -----  -
- 000001  PORD01344  2007-02-16  02  STU  25,300.00
- 000002  PORD02134  2007-02-22  03  JKL  1,850.00
- 000153  PORD02478  2007-02-23  01  STU  199.67
```

And maintain or change a customer record with F4 prompting

```
Customers                      Customer Detail Maintenance          Dababorough Ltd.
CUSTMNT1                                                                11:12:08
                                                                2009/05/01

Customer No . . . . ACC1
Customer Name . . . Bertwhistle Ltd1
Statement Account . ACC23
Related Account . . ACC43
Tax Reg . . . . . TAX01
Bank . . . . . 235468
Bank A/c . . . . .
CusGrp . . . . . > PV Private : Please select:
Rep . . . . . > JKL Johan K : _ JKL Johan Klaassen
Distributor . . . > UK only : _ MTT Mark Tregear
Credit Limit . . . 10012 : _ NWD Neil Woodhams
Stl Dsc . . . . . FKL : _ RJD Richard Downey
Cr Guarantee . . . N : _ STU Stuart Milligan
B/O . . . . . Y
Date Loaded . . . . 2003-06-14
Last Sale . . . . . 2003-08-14
```

These screens are fairly typical of thousands of System i applications which exist today.

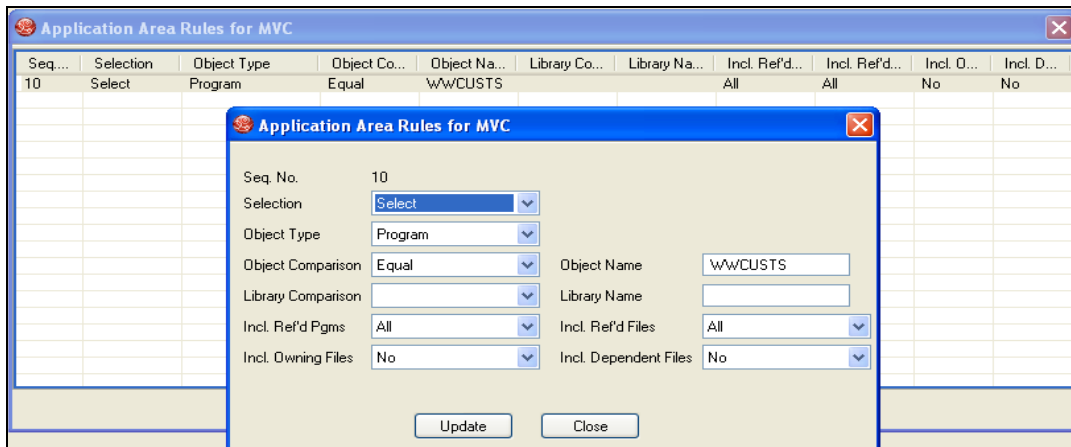
### Application Areas

Having identified that the Work with customers area is what we are interested in the next thing to do is to identify which programs, displays and files are required to make that area work. This would not be a straightforward task if you didn't have good knowledge of the system.

Fortunately X-Analysis makes this process easy for us by allowing us to easily set up multiple application areas which we can base on a program or programs and specify that we want to include all referenced files and programs:



## Application Modernization and Rebuilding with X-Analysis



Once we have our application we can view the objects or programs it contains:

Library	Name	Type	Attribute	Description	Status	Used	Created	Changed	Function	
R1	XANHCDEM	CUSFSEL	*PGM	RPGL	Customer Site Selection	*C	06/03/09	10/25/05	05/26/08	DSPFIL
R1	XANHCDEM	CUSTOMNT1	*PGM	RPGL	Customer Detail Maintenance	*B	06/04/09	03/04/09	03/04/09	EDTRCD
R1	XANHCDEM	CUSTSSEL	*PGM	RPGL	Customer Selection	*C	06/03/09	02/19/07	05/26/08	DSPFIL
R1	XANHCDEM	DISTSSEL	*PGM	RPGL	Distributor Selection	*C	06/03/09	02/19/07	05/26/08	DSPFIL
R1	XANHCDEM	ORDSTSEL	*PGM	RPGL	Order status Selection	*C	06/03/09	02/19/07	05/26/08	DSPFIL
R1	XANHCDEM	STKMASEL	*PGM	RPGL	Product Master Selection	*C	06/03/09	02/19/07	05/26/08	DSPFIL
R1	XANHCDEM	STOMASEL	*PGM	RPGL	Store Master Selection	*C	06/03/09	02/19/07	05/26/08	DSPFIL
R1	XANHCDEM	TRNTPSEL	*PGM	RPGL	Transaction type Selection	*C	06/03/09	02/19/07	05/26/08	DSPFIL
R1	XANHCDEM	WCUSTP	*PGM	RPGL	Print Customer Detail	*C	06/05/09	07/19/07	03/25/09	PRTFIL
C1	XANHCDEM	WKCUSL	*PGM	CLP	Customer Letter	*B	06/03/09	10/25/05	05/26/08	PRTFIL
R1	XANHCDEM	WKCUSLV	*PGM	CLP	Validity Checker for WKCUSL	*D	06/03/09	10/25/05	05/26/08	OTHFIL
R1	XANHCDEM	WWCONDET	*PGM	RPGL	Work with Order Details	*B	06/03/09	01/25/08	05/26/08	EDTFIL
R1	XANHCDEM	WWCONHDR	*PGM	RPGL	Work with Orders	*B	06/03/09	01/25/08	05/26/08	EDTFIL
R1	XANHCDEM	WWCUSTS	*PGM	RPGL	Work with Customers	*A	06/04/09	04/01/09	04/01/09	EDTFIL
R1	XANHCDEM	WWTRNHST	*PGM	RPGL	Work with transaction history	*B	06/03/09	03/11/08	05/26/08	EDTFIL

This is a very powerful technique and allows rapid prototyping and development without having to develop the whole system in one go. The application areas you create can be integrated together into a larger structure and displayed using the application area diagram.

### Rebuild the Application

We can build a web application directly on the application area that we have just created.

We can choose the programs to include, the application architecture we want to use Java or EGL (PHP and .Net under development), whether to include Business Rule logic or not and the type of data access mechanism we wish to use JDBC or ORM (Hibernate).

When we press OK here a new Java Web application is generated for us.

As we can see in the screen shot (given below) the application that is generated comes complete with the necessary script files to build it for different application servers e.g. Apache Tomcat and WebSphere.



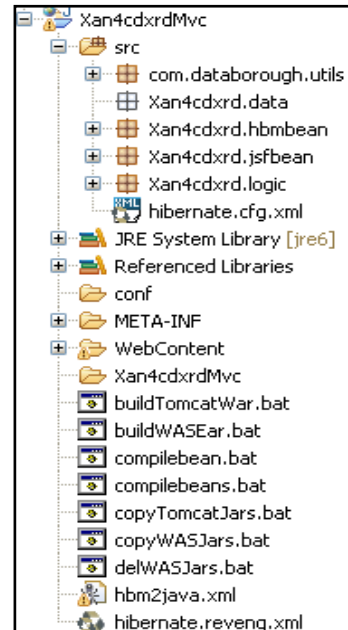
## Application Modernization and Rebuilding with X-Analysis

The code is structured in a logical manner with hibernate beans or data beans according to whether we chose the ORM option or not.

Similarly, if we opened up the classes and the JSF pages produced we would see that they have names which can clearly tie them back to the original application even though this is brand new code.

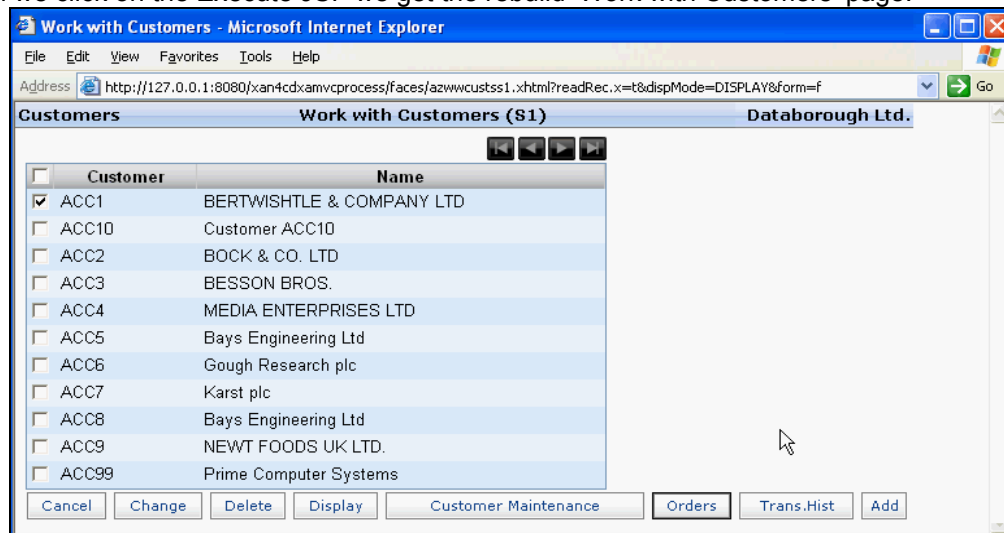
Once this application is built we can deploy it to the application server of our choice and start to use the application.

You can either go straight to a browser and navigate to a page , but an easier way is to use the built in integration with X-Analysis.



If we go back to the X-Analysis view of our application area and bring up the programs , if we select WWCUSTS and right click we have options to allow us to display the corresponding JSF page as can be seen in this screen shot:

When we click on the Execute JSF we get the rebuild 'Work with Customers' page:



The pages that have been generated are fully functional, however, you will probably want to alter the appearance to match your standard pages and perhaps add links and so on to integrate the pages with other systems. The pages use Cascading Style Sheets (CSS) to control their look and feel and positioning thus changes are easy to make.



## Application Modernization and Rebuilding with X-Analysis

Regardless of your opinion on the aesthetics of the generated screens you will agree that the basic pages are quite pleasing for zero coding and just a few clicks away!

The screenshot shows a web browser window titled "Work with Orders - Microsoft Internet Explorer". The address bar contains a URL starting with "http://127.0.0.1:8080/xan4cdxamvcprocess/faces/zzwwconhdrs1.xhtml;jsessionid=48A63C80B936D0CB9477D59CE0C0D9...". The page content includes:

- Page title: **Customers** | **Work with Orders** | **Databorough Ltd.**
- Customer: ACC1 BERTWISHTLE & COMPANY LTD
- Representative: MTT Mark Tregear
- Telephone Number: [input field]
- Current balance: [input field] 0.00
- Credit Limit: [input field] 0.00
- Table with columns: Cntrac, Customer Reference, Date, St, Rep, Value
- Buttons: Cancel, Change, Delete, Display, Lines, Add

<input type="checkbox"/>	Cntrac	Customer Reference	Date	St	Rep	Value
<input type="checkbox"/>	000001	PORD01344	16-02-07	02	STU	25,300.00
<input type="checkbox"/>	000002	PORD02134	22-02-07	03	JKL	1,850.00
<input type="checkbox"/>	000153	PORD02478	23-02-07	01	STU	199.67

Following the screen flow we have on the green screen side, here's the 'Work with Orders' screen.



## Application Modernization and Rebuilding with X-Analysis

Purchases (S1) - Microsoft Internet Explorer

Address <http://127.0.0.1:8080/xan4cdxamvcprocess/faces/zccustmnt1s1.xhtml;jsessionid=9DFD66FD4F92E4641155F7DE8BB43BED?parentScn=AZ> Go

**Customers** **Customer Detail Maintenance** **Daborough Ltd.**

**Customer Ilo**  
ACC1

**Customer Name**  
BERTWISHTLE & COMPANY LTD

**Statement Account**

**Related Account**

**Tax Reg**

**Bank**  
000000000

**Bank A/c**  
0000000000000000

**Forex**

**CusGrp**  
AG Agent  
AG Agent  
CA Cash  
GN General  
GV Government  
PV Private  
RT Retailer

**Terms**

**Stl Dsc**

**Int**

**Cr Guarantee**

**B/O**

**Lang**

**Date Loaded**  
16-02-07

**Chg-Date**  
16-02-07

**Last Sale**  
14-08-03

Cancel Submit

And finally the customer maintenance page with a drop down selected and the calendar prompt on the date field.

**Date Loaded**

16-02-07

< February 2007 >

Su	Mo	Tu	We	Th	Fr	Sa
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	1	2	3
4	5	6	7	8	9	10



## Completing the Modernization Process

The tasks remaining to complete the application rebuild process are listed below:

### Surrounding Application Framework

In most cases, it will be desirable to replace some of the System i application constructs with JEE or other equivalents.

- File Overrides
- LDA type constructs
- Soft Security
- Commitment Control

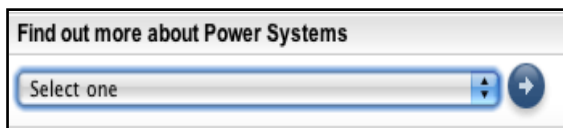
### Look & Feel & UI Standards

Green screen applications have always been optimized for rapid data entry with such helpful features as tabbing between fields and Field exit field completion. It is perfectly possible to make a web application work well for data entry but it requires effort and as was often the case with windows applications it isn't always done well.

The task of designing new interfaces for the modernized applications should not be underestimated you can't simply rely on the 'cool' factor of the web and the colorful and sophisticated look and feel that style sheets enable do not guarantee a usable interface.

When we analyze this scenario we see that in the green screen application we have a system which can be operated quickly by the users using only the keyboard and is very responsive - though it would appear intimidating and strange to new users used only to Windows or web applications - the challenge is to build a new interface which is accessible, intuitive and easy to use for new users whilst allowing the existing users to apply their knowledge of the system to get around it quickly and with good performance and responsiveness.

Take an example, where we wish to produce a new application, where a user enters transactions involving entering codes. In the green screen world this would typically just be an input field with possibly and F4 prompt option to show the allowable values. As discussed above, in the web world the first thought is often to convert this type of construct to a drop down combo box, like this example from the IBM Power site:



This can be cumbersome if there are many values and unless you have only a few codes beginning with each number or letter (in most browsers once the focus is on a drop down box

typing a letter or number will position the cursor at the first matching entry in the list) keyboard navigation is limited and can involve a lot of scrolling.

This example looks quite cumbersome, but imagine if it had several hundred options and you had to repeat the same process many times on each page! So how can we improve the usability and provide help for new users at the same time.



## Application Modernization and Rebuilding with X-Analysis

The best choice is likely to be to combine two approaches with a variable delay on the dynamic list so we don't waste system time generating lists when the user knows the whole code, only prompting when it looks like it would be useful.

Ajax has become widely used since 2005 and a number of frameworks exist to make their use easier and in some cases automatically for example JSON - JavaScript Object Notation and GWT - Google Web Toolkit.

Practical usability and look and feel designs go hand in hand, and require knowledge and information about modern UI controls and legacy UI patterns, which can also be extracted as part of the X-Analysis documentation facilities.



Application Modernization and Rebuilding with X-Analysis

## Highlights of Modernization Toolset

- ✓ Fully automated and integrated documentation with X-Analysis
- ✓ Re-engineered Programs
- ✓ Auto-Generates RPG programs from Business Rule Logic
- ✓ Pseudo code narrative of Business Rule Logic
- ✓ DDS to SQL Database Conversion
- ✓ Service IO modules created per converted SQL table
- ✓ Copies production data into new database
- ✓ Generate Database Service Programs
- ✓ Creates RPG stored procedures and web services
- ✓ Generate Web Applications (JSFs/Facelets and Javabeans/EGL)
- ✓ Can be used with either JEE or .Net frameworks
- ✓ Generates new projects in IDE's such as Eclipse or IBM Rational products
- ✓ Generates UML class diagram for each stored procedure
- ✓ Generates UML Activity diagrams from legacy applications



## Application Modernization and Rebuilding with X-Analysis

### **Summary**

Comprehensive, accurate, and current documentation of a legacy application improves quality, productivity and reduces risk, for any maintenance, modernization or rebuild IT project. The risk associated with maintaining large complex legacy application, with a rapidly diminishing set of legacy skills, can be largely mitigated by access to such documentation.

Understanding and mapping the relevance of existing designs, and quantifying the scope and metrics of an application, is the first step in ANY modernization project, even if the application is to be replaced. Design constructs such as the data model can be used for support, development, and testing tasks such as test data extraction. Source change management can be vastly improved by powerful cross-referencing functionality.

Legacy design constructs can be used passively in the form of information they represent, and programmatically to radically accelerate application rebuilds; a requirement for achieving true long-term application modernization. A combination of both allows optimum use of internal and external resources and existing design assets.

X-Analysis delivers against all of these concepts. 20+ years of development effort, ensures that virtually any legacy application can be automatically reverse engineered onto a high-level design.