

# X-Analysis Design Recovery Overview

Version 9

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

Databorough Limited  
2011



## X-Analysis Design Recovery Overview

# 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. 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>Introducing X-Analysis .....</b>	<b>1</b>
<b>Why Design Recovery is difficult .....</b>	<b>2</b>
<b>Analysis, Documentation &amp; Application Subdivision.....</b>	<b>4</b>
Understanding Design & Function More Easily .....	4
Producing Static Documentation Automatically .....	7
Application Subdivision .....	7
<b>Recovering an Application Design.....</b>	<b>9</b>
Recovering the Data Model .....	9
Recovering the User Interface .....	10
Recovering Business Rule Logic .....	10
UML Diagramming .....	12
<b>Highlights of Design Recovery Toolset .....</b>	<b>14</b>
<b>Summary.....</b>	<b>15</b>



## 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.



## 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. It's 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:



## X-Analysis Design Recovery Overview

- 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
<a href="tel:07678678912">07678 678912</a>	<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.



## **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.

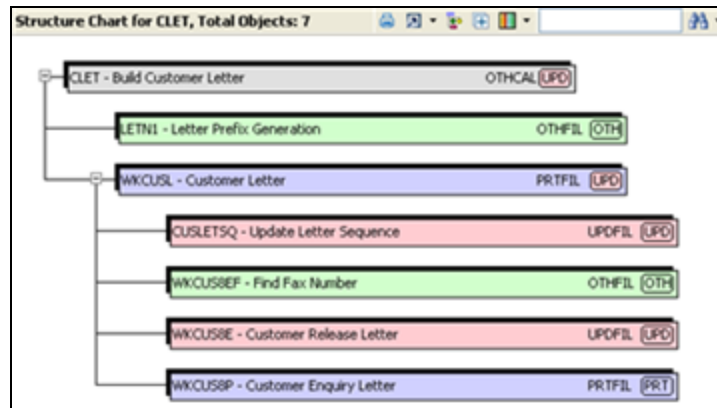
Here is a brief description of some of these diagrammatic constructs and views:

#### **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.

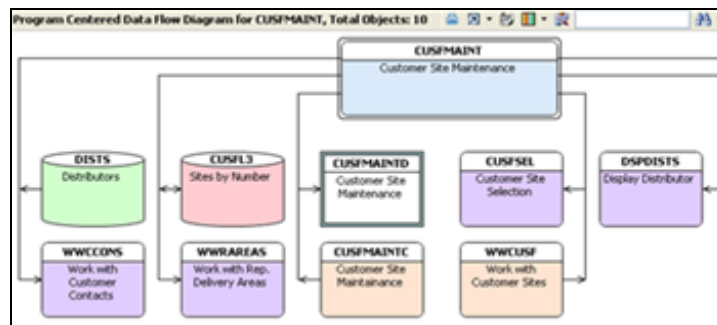


## X-Analysis Design Recovery Overview



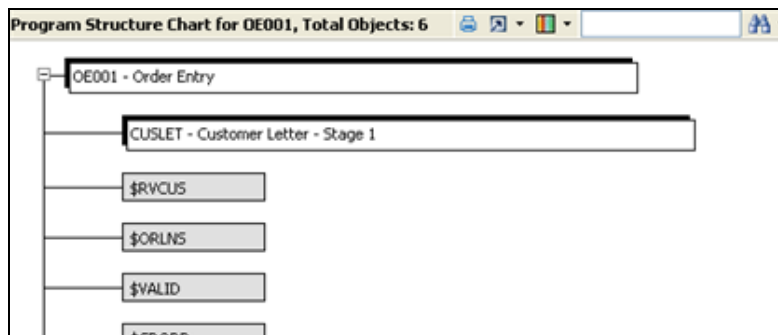
### Data Flow Diagram

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.



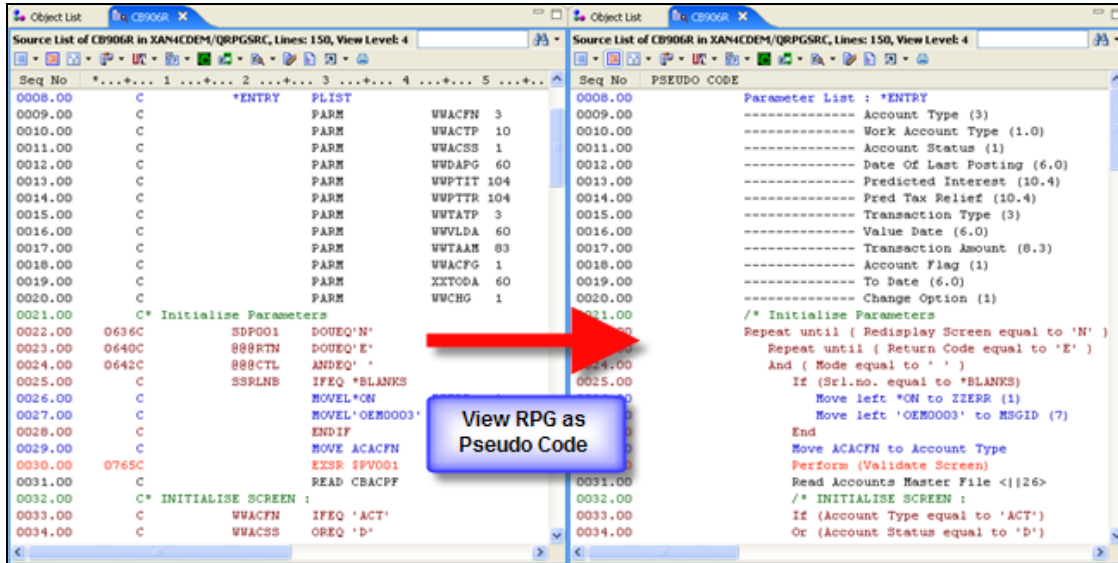




## X-Analysis Design Recovery Overview

### 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.



### 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

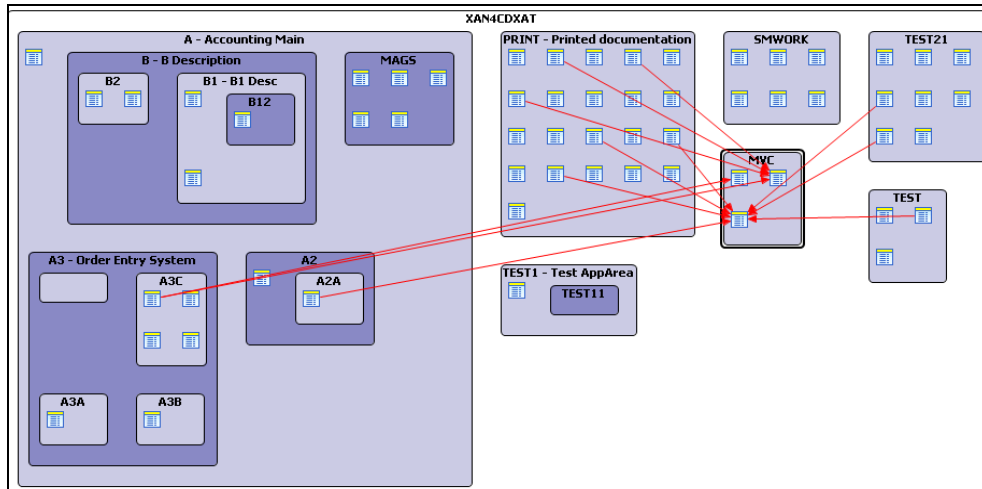
Entire legacy applications are often too large to effectively comprehend or effect wholesale change. For this reason it is often necessary or helpful to sub divide a system into application



## X-Analysis Design Recovery Overview

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 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.

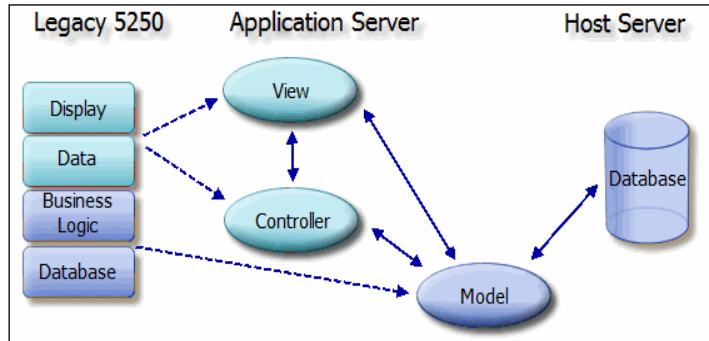




## Recovering an Application Design

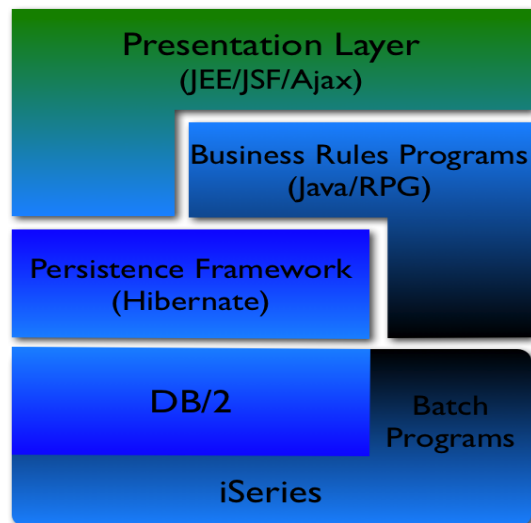
The concept of reusing existing code or logic is not a new one. The challenge has always been to identify, isolate, and reuse only those designs that are relevant in the new context in which they are desirable. The sheer volume of code, its complexity, and the general lack of resources to understand 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

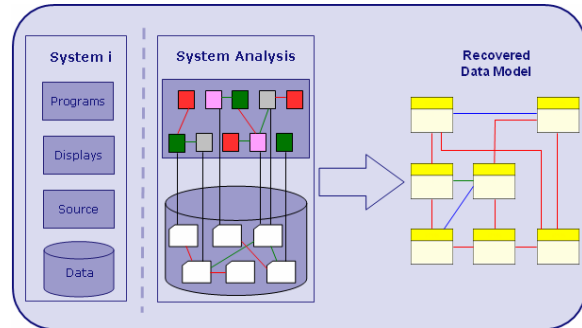


## X-Analysis Design Recovery Overview

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

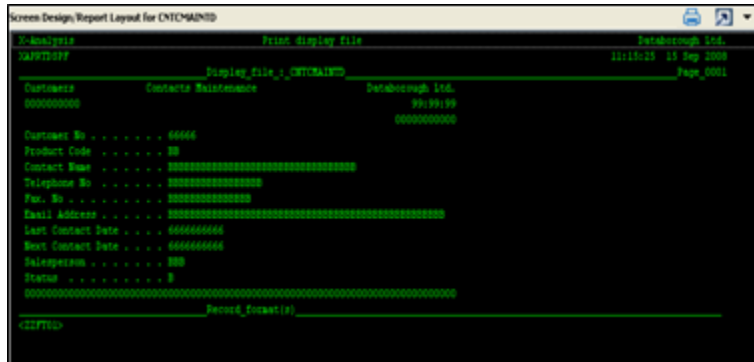
X-Analysis has the unique capability of automatically deriving the explicit system data model from a legacy RPG, COBOL or 2E application. Let us have a look at this and the model reuse capability in a bit more detail.



### 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.



## X-Analysis Design Recovery Overview

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):



## X-Analysis Design Recovery Overview

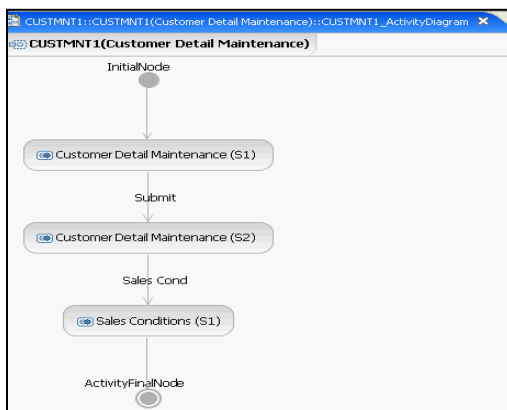
```
0169.00      C* Telephone number
0169.96      !=BRC* If the field ZTELNO is not blank , verify the field ZTELNO against '
0169.97      !=BRC* 0123456789'. If other values found then the field "Phone" is invalid.
0169.98      %!BRC* Business Rule No. XAN4CEM/QRPGLESRC/CNTCMAINT/170 Validation.
0169.99      @!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:errormsg)
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.

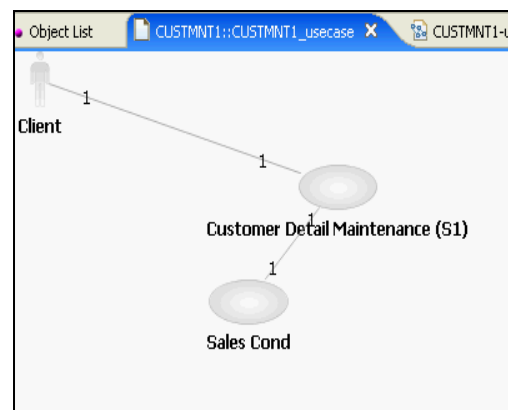
## UML Diagramming

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



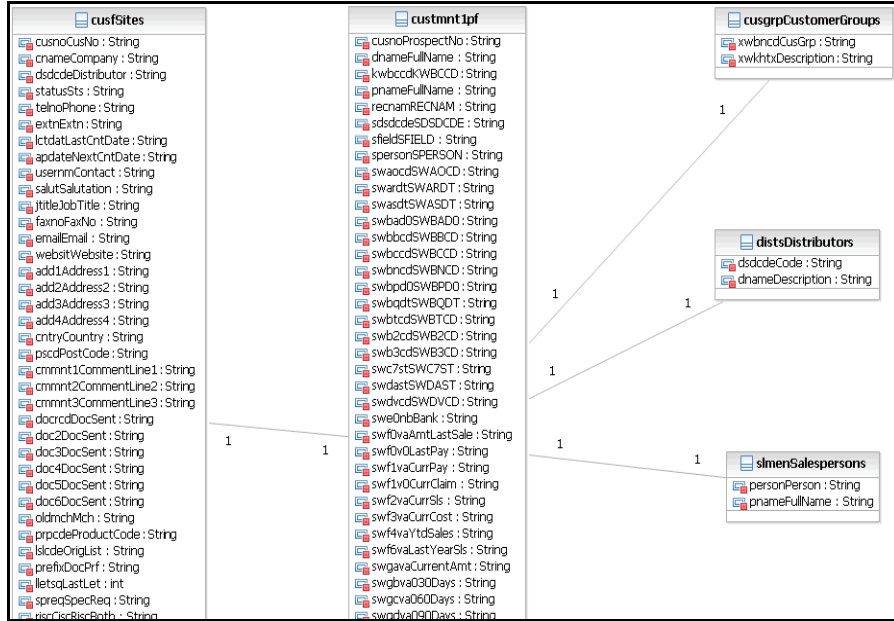
### Use Case Diagram





## X-Analysis Design Recovery Overview

### Class Diagram



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.



## Highlights of Design Recovery Toolset

- ✓ Identifies and documents business rule logic from legacy code
- ✓ Analysis and cross referencing of business rule logic in legacy code
- ✓ Individual Business Rule Annotation
- ✓ Business Rule Where Used & Summaries
- ✓ Pseudo code narrative of Business Rule Logic
- ✓ DDL export from data model
- ✓ Generation of UML Diagrams Activity, Class and Use Case



## X-Analysis Design Recovery Overview

### **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 provides a solid foundation for either ongoing maintenance and development of systems or modernization projects. 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.