



MANAGEMENT OVERVIEW

**A GUIDE TO THE BENEFITS OF USING
X-2E ENTERPRISE**

Design Recovery & Rebuild with X-2E Enterprise

Since its introduction the CA 2E has become the most successful 4GL tool on the AS/400 platform. The CA 2E (SYNON) development environment combines a rich and precise model of designs and specifications, with powerful code generation capabilities. Even with the powerful set of features many companies are now facing real pressure to modernize their business applications beyond the scope and capability of what CA 2E can offer. The challenge is to move forward without discarding decades of investment in design, evolution and fine tuning stored in the CA 2E model.

Databorough's X-2E Enterprise is the solution for modernizing the CA 2E applications. The X-2E Enterprise automatically re-factor and generates a MVC web application using OO methods in Java, C#.Net or EGL (Flex & Silverlight UI soon to be added).

Benefits

The X-2E Enterprise offers following benefits:

- Drill down through action diagrams and business rules
- UML Action / Class / Use Case diagrams
- Function definitions – data content, actions / branches / screen designs
- Functions converted into JSF / Flex / Silverlight
- Action Diagrams converted into business logic beans / classes
- Internal functions converted to reusable classes
- Data model converted to long-name DDL and Object Relational Maps in Hibernate & DAO classes
- Complete re-factoring into event driven Model view controller using OO designs
- Exports to DDL / UML / XML

And many more...

The X-2E Enterprise by-passes the generated RPG/COBOL code and uses the CA 2E model as a specification to automatically re-factor and generates a MVC web application. The X-2E Enterprise can be used against an entire system, user defined application areas, or individual functions as part of an ongoing modernization / migration efforts. This provides better flexibility and control to the user, and helps reduce risk and overall cost, while maximizing productivity with user driven automation.

Design Recovery: Modernization Base

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

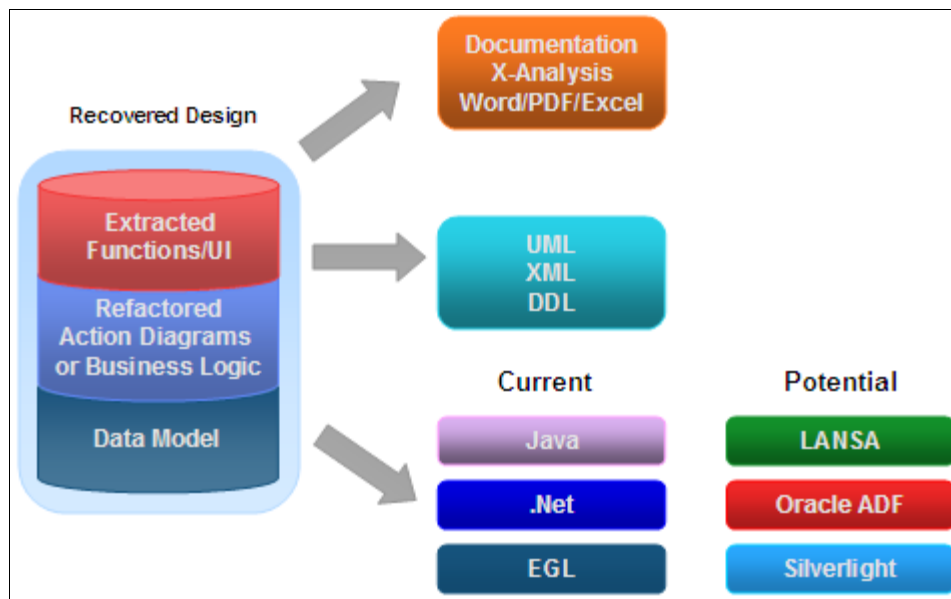


Illustration 1: X-2E Design Reuse Options

Modernization Process

The Modernization process of the X-2E Enterprise consists of the two stages – Component Generation and Application Generation.

Component Generation

The X-2E Enterprise focus on the following three components in the Component Generation stage:

- Database Modernization
- Business Logic Rebuild
- UI/Print Functions

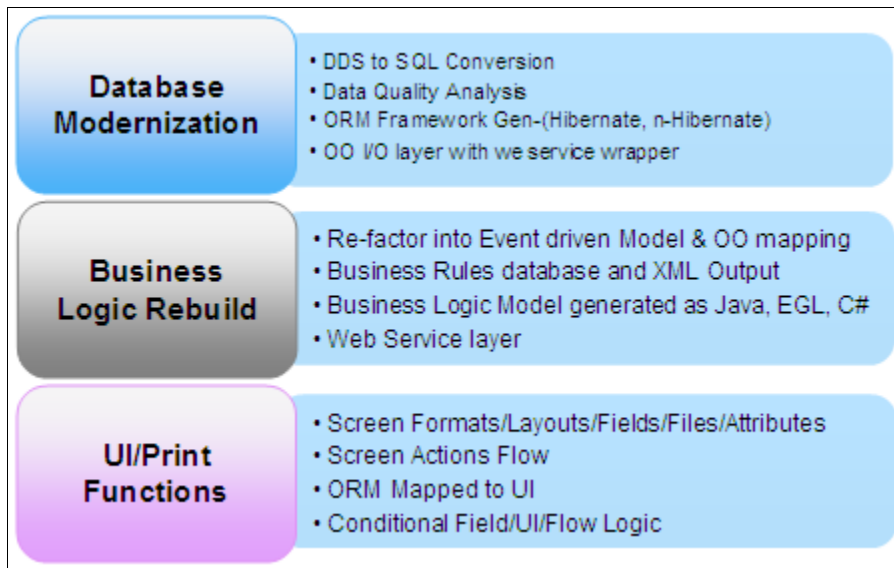


Illustration 2: Component Generation

Database Modernization

It has always been possible to access System i data in a relational database like fashion there was originally but 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-2E 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:

- 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.
- 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.

```

A.....T.Name+++++RLen++TDpB.....Functions+++++-----CUSF
***** 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 3: DDS of a File

```

-- Generate SQL
-- Version:                V6R1M0 080215
-- Generated on:           01/07/10 12:03:51
-- Relational Database:
-- Standards Option:      DB2 15/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 DEFAULT ''
  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 AUPDATE 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 4: 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.
- Ability to use modern Object Relational Mapping (ORM) software such as Hibernate for rapid application development in Java and other modern languages.
- Reduced dependency on System i specific skills such as DDS, which may 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.

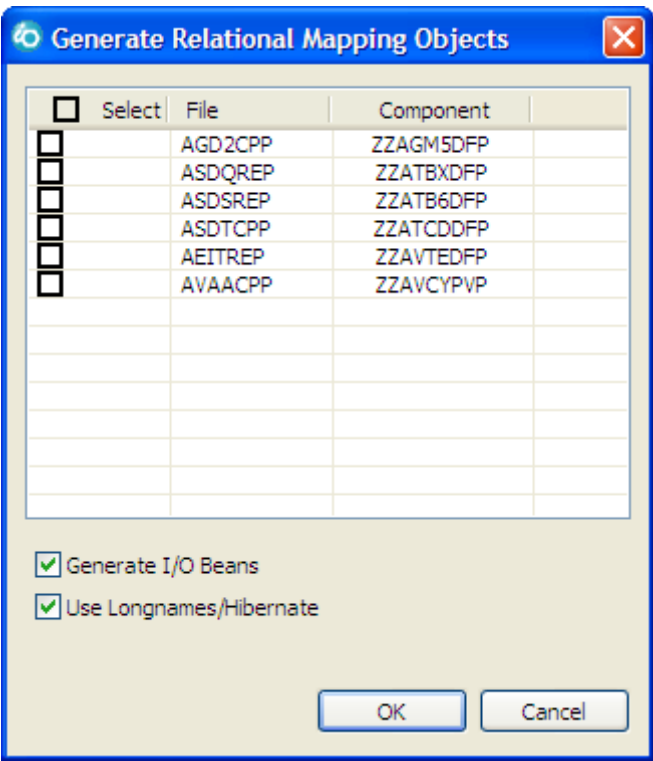


Illustration 5: Generate ORM Objects from X-2E

Database modernization also utilizes the advanced design extraction directly from the CA 2E model, making use of long field names, constraint logic and all relevant database abstractions, either kept on IBM i or migrated onto other DBMS systems.

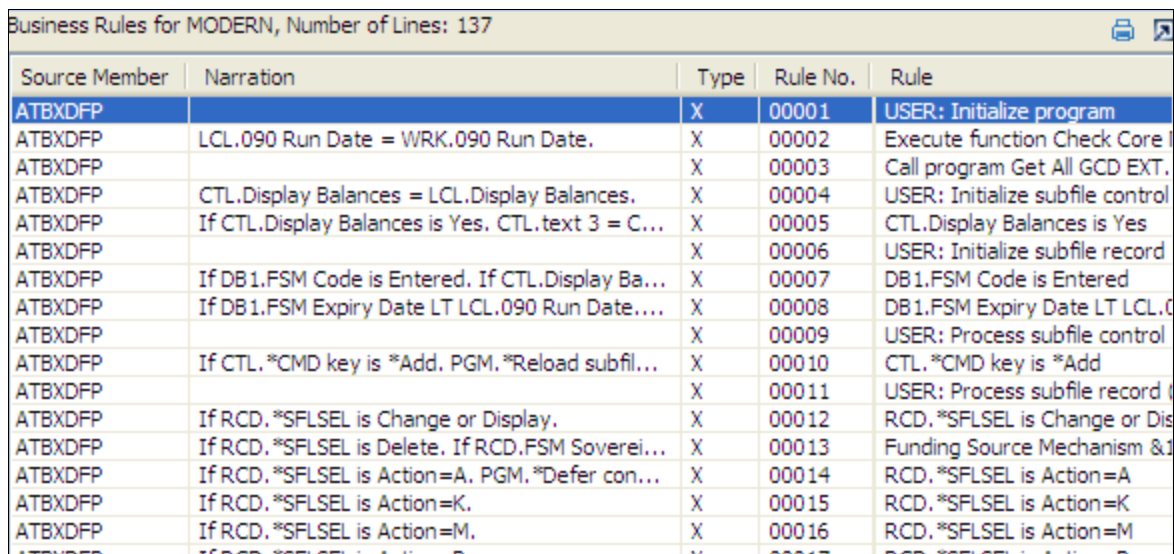
Business Rule Logic Rebuild

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

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.

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



Source Member	Narration	Type	Rule No.	Rule
ATBXDFP		X	00001	USER: Initialize program
ATBXDFP	LCL.090 Run Date = WRK.090 Run Date.	X	00002	Execute function Check Core
ATBXDFP		X	00003	Call program Get All GCD EXT.
ATBXDFP	CTL.Display Balances = LCL.Display Balances.	X	00004	USER: Initialize subfile control
ATBXDFP	If CTL.Display Balances is Yes. CTL.text 3 = C...	X	00005	CTL.Display Balances is Yes
ATBXDFP		X	00006	USER: Initialize subfile record
ATBXDFP	If DB1.FSM Code is Entered. If CTL.Display Ba...	X	00007	DB1.FSM Code is Entered
ATBXDFP	If DB1.FSM Expiry Date LT LCL.090 Run Date....	X	00008	DB1.FSM Expiry Date LT LCL.0
ATBXDFP		X	00009	USER: Process subfile control
ATBXDFP	If CTL.*CMD key is *Add. PGM.*Reload subfil...	X	00010	CTL.*CMD key is *Add
ATBXDFP		X	00011	USER: Process subfile record
ATBXDFP	If RCD.*SFLSEL is Change or Display.	X	00012	RCD.*SFLSEL is Change or Dis
ATBXDFP	If RCD.*SFLSEL is Delete. If RCD.FSM Soverei...	X	00013	Funding Source Mechanism &1
ATBXDFP	If RCD.*SFLSEL is Action=A. PGM.*Defer con...	X	00014	RCD.*SFLSEL is Action=A
ATBXDFP	If RCD.*SFLSEL is Action=K.	X	00015	RCD.*SFLSEL is Action=K
ATBXDFP	If RCD.*SFLSEL is Action=M.	X	00016	RCD.*SFLSEL is Action=M
ATBXDFP	If RCD.*SFLSEL is Action=D.	X	00017	RCD.*SFLSEL is Action=D

Illustration 6: Business Rules Summary

The business rule repository can then either be used programmatically to generate new code (migrated logic -check the Illustration 6 given below), 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.

```

Migrated Logic
+ /* Data Section for ATBXDFP */
  /* Program logic for W W FundSourceMechanisms Display file
- PREENTRY processing
+ Call Check Core Module
  LCL.X090_Run_Date = X090_Run_Date
+ Call Get All GCD EXT
- PREDISPLAY processing for screen format (ZZATBXDFPS1)
  Display_Balances = LCL.Display_Balances
  If Display_Balances = 'Y'
    text_3 = 'Yes'
  Else
    text_3 = 'No'
  End
+ PREDISPLAY processing for screen format (AZATBXDFPS1)
+ VALIDATION processing for screen format (AZATBXDFPS1)

```

Illustration 7: Migrated Logic generated from Business Rules

The generated migrated logic can also be exported to XML.

```

<?xml version="1.0" encoding="UTF-8" ?>
- <BRLOGIC title="" version="XA8.9.48">
- <program-name name="ATBXDFP" text="W/W
  FundSourceMechanisms Display file">
- <data-section>
- <functions>
- <field-type name="Workfield">
- <fields>
  <field decimal="0" name="LCL.Display_Balances"
    size="1" type="A" />
  <field decimal="0" name="LCL.X090_Run_Date"
    size="7" type="P" />
</fields>
</field-type>
</functions>
</data-section>
- <logic-section>
- <functions>
- <function name="" srceen-format="">
- <action-points>
- <action-point>
  <action-name>PREENTRY</action-name>
- <business-rules>
- <business-rule>
  <narration />
- <logic-records>
  <logic-record value="CALL Check

```

Illustration 8: Migrated logic as XML


```

<?xml version="1.0" encoding="UTF-8" ?>
- <FUNCTION>
  <!-- FUN (20A) -->
  <name>ZZATBZPVPS1</name>
  <!-- PFILE (50A) -->
  <file-name>ASDQREP</file-name>
  <!-- TITLE (50A) -->
  <title>Maintain Funding Source Mechanism</title>
  <!-- FUNTYP (1A) -->
  <!-- G=Grid, R=Record(ZA), L=List, D=Drop/Down, U=Update
  (UA) -->
  <type>R</type>
  <!-- ----- -->
  <!-- ----- -->
  <!-- Actions -->
  <!-- ----- -->
  <!-- ----- -->
  <!-- type (DPFTYP) -->
  <!-- C=Call, R=Call (Display), U=Call(Update),
  H=Conditional Call -->
  <!-- I=Conditional Call(Display), S=Server, X=eXit
  Action, W=Write -->
  <!-- Y=Copy, D=Delete, V=Validate, L=Selector Window,
  T=Trailer Format -->
  <!-- F=Dep.Grid, P=Parallel Grid, G=Conditional Grid,
  E=Extended Defn. -->

```

Illustration 11: XML of the Screen Design

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. These are:

- Formats/Layouts - Some screens may benefit from amalgamation or redesign, but table edits, and non-transaction type screens will largely remain the same, if not identical in layout.
- Actions - whether from sub-file options, command keys, or default enter actions, these often represent an important part of the usefulness of an application design. The mechanisms used to offer or invokes these calls may change, but where they go logically and what parameters they pass will largely remain consistent.
- Fields/Files/Attributes - What fields are on what screens, and where the data comes from is a requirement in any system development. Attributes of a field can also help determine what type of controls might be used in a modern UI. For example, a Boolean type might be implemented with a check box, a date with a date prompt. Again, these are simple enough to edit in modern IDE's, but the volume associated with any large legacy application modernization can make this work prohibitive.
- Data Model Mapping - Validations and prompting mechanisms that ensure referential integrity in legacy applications can also be vital to extract. This is both to implement referential integrity and to provide design information for building modern prompt or selection controls such as drop downs or lists.

Naturally, it will be desirable to redesign some UI's completely. For those programs and screens where this is not the case, the design, and mapping information can be used directly in the new version of the application, even though the UI code has been discarded.

X-2E extracts User Interface design information as described above and stores it as meta-data in the X-2E 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.

Application Generation

The modernize web application based on MVC architecture can be generated after all the components are available. The X-2E Enterprise generates the sort of application that would have been written by hand by a professional application architect and team of coders, without the losing the essence of the original design or functional value. This is all done automatically, into the IDE's of the language chosen by the user during generation. The new application is easier to enhance and maintain, and as such naturally attracts a wider more readily available resource pool for future development and maintenance.

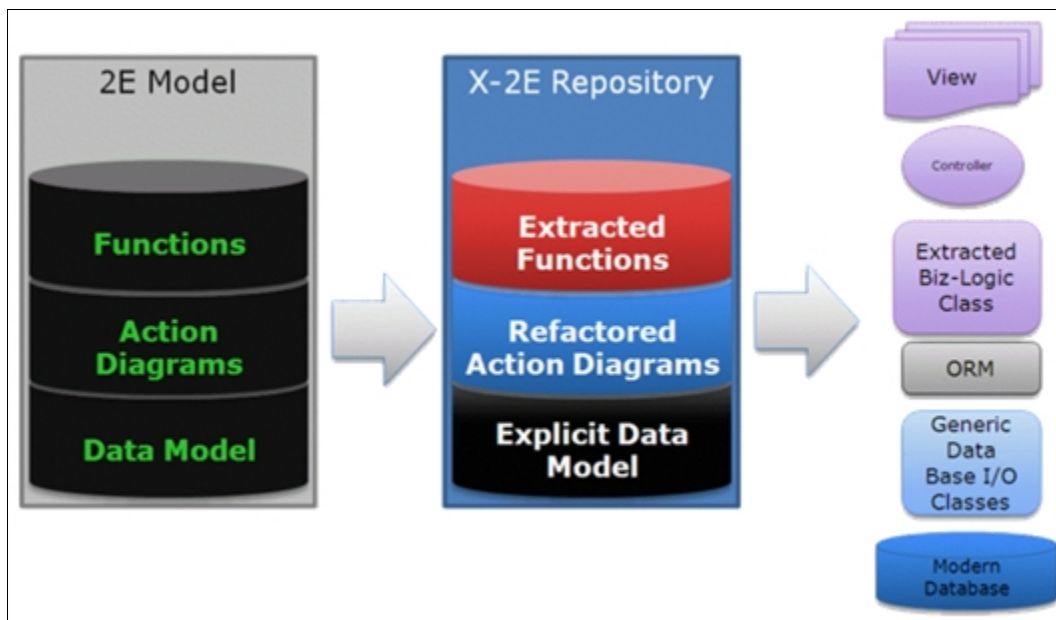


Illustration 12: Application Generation

Generate Web Application

The X-2E Enterprise provides an option to build a web application directly on a individual program or application area or for a complete system. The 'Re-generate Programs' option offers various options related to the new application development. User can choose from Java / EGL / ASP.NET as Language option. Other options related to the use of long names and inclusion of business logic are also available.

After the successful generation of the web application user can execute the generated web application from with the X-2E Enterprise. With just a single click user can view the web application.

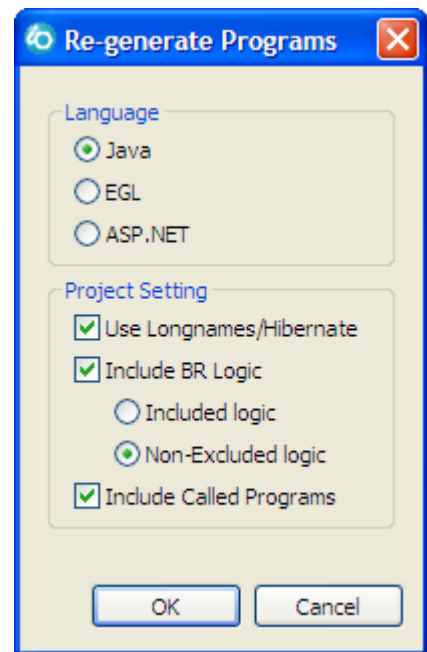


Illustration 13: Generate Web Application Dialog

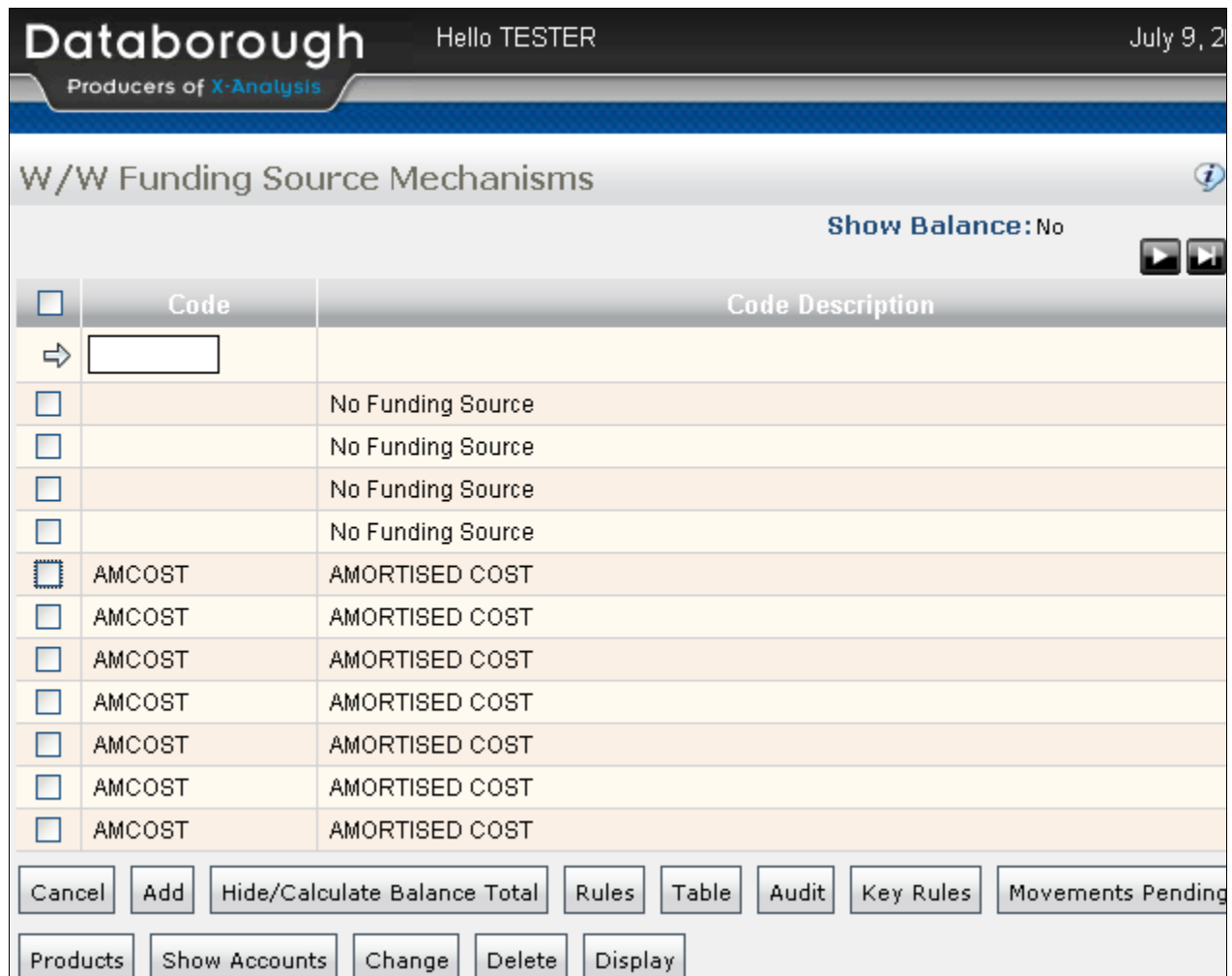


Illustration 14: Web Application

Databorough Hello TESTER J
 Producers of X-Analysis

Maintain Funding Source Mechanism

Code :

Description :

Effective Date : **Expiry Date :**

Client : **Eligibility Pgm :**

Process When Verify : **Next Movement In :**

Selection to FSM : **Selection Report :**

Selection Extract : **Report PGM :**

External Batch :

B	AGC	▼
	AGC	dfdfdf
	AKLG	Auckland Agencies
	AUCK	Warsapperuma Company
	BNEM	B & E Melbourne
	CHGO	Chicago
	CHRI	Canterbury
N	DALZ	Dalzell
	DAV1	Abdullah Hamid Mr
	DAV2	Absolum Walter Raymo
	DAV3	Acme Cars Ltd
	DIL	DIL

Illustration 15: Individual Function with drop down

The pages that have been generated are fully functional. The pages all use Cascading Style Sheets (CSS) to control their look and feel and positioning so changes are easy to make. Regardless of your opinion on the aesthetics of the generated screens you will agree that the basic pages are good for zero coding and just a few clicks!!

The code generated for all the functions can be analyzed and modified as required.

```
WOrWFundsourcemechanismsDisplayFile.java X
+ import java.sql.SQLException;
+ * Program logic for WOrWFundsourcemechanismsDisplayFile (ATBXDFP).
public class WOrWFundsourcemechanismsDisplayFile extends BusinessLo
    // I/O File instance.
    public AsdqrepF asdqrepF = new AsdqrepF();
    // Data File instance.
    public Asdqrep asdqrepR = asdqrepF.asdqrep;
    // Data section.
    public String Display_Balances;
    private int X090_Run_Date;
    public String screenToCall;
    // PREENTRY processing.
    public boolean preEntry() throws SQLException {
        callCheckCoreModule ("C");
        callGetAllGCDEXT ("SECURE", "DSPBALANCE", "DISPBT", Display
        return true;
    }
    // PREDISPLAY processing.
    public boolean preDisplayG1() throws SQLException {
        if (!isBlanks(asdqrepR.FSM_Code)) {
            if ("Y".equals(Display_Balances)) {
                callGetTotalsforSecMechmm (asdqrepR.FSM_Code, asdq
            }
        }
    }
}
```

Illustration 16: Generated Java Code

Highlights

- Entity relationship diagrams
- Graphical object/field/variable where used plus for 2E internal functions
- UML Action/Class/Use Case diagrams
- Drill down through action diagrams and business rules
- Function definitions – data content, actions/branches/screen designs
- Exports to DDL/UML/XML Word, PDF
- Functions converted into JSF/Flex/Silverlight
- Action Diagrams converted into business logic beans/classes
- Internal functions converted to reusable classes
- Data model converted to long-name DDL and Object Relational Maps in Hibernate & DAO classes
- Complete refactoring into event driven Model view controller using OO designs

The X-2E Enterprise can be used by the companies looking to Modernize their CA 2E application. It offers lots of benefits and various useful features which can be used by both highly technical CA 2E programmers and by those with little or no programming knowledge.

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

© copyright Databorough 2010

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