A Large NetRexx Application

René Vincent Jansen, I-Bizz IT Services and Consultancy BV

Böblingen, May 6th 2004, 08.15 h.
Every effort has been undertaken to ascertain this presentation does not contain sensitive material. The data of this application is classified IBM & ABN AMRO Internal and Confidential. The displays of the application that contain this data have been modified for presentation and do not reflect confidential data. Where actual components of the IBM Banking Data Warehouse Model have been shown, care has been taken that only information is been shown that is also publicly available through other sources, in casu the worldwide web or official IBM sales brochures.

The previous notwithstanding, ABN AMRO Bank NV cannot be held accountable in any way, as the work has been carried out as an independent external party for the sole responsibility of I-BIZZ IT Services and Consultancy BV, Amsterdam, The Netherlands, and its intermediates in this assignment.
Goals

- Give an account of the design and implementation of a fairly large Object Oriented, Multi-User Client Server Application
- Show specific advantages
- Present practical experiences in combining NetRexx with generated Gui-builder code
- Show pitfalls and solutions
Non-Goal

- Preach to the converted
How large is Large?

- 52 Dialogs
- 232,427 Records in data base
- 18,834 Object Instances
- 376 NetRexx Classes
- 53 Java - Only Classes
About ABN AMRO

ABN AMRO is a prominent international bank, its origins going back to 1824. ABN AMRO ranks 11th in Europe and 23rd in the world based on tier 1 capital, with over 3,000 branches in more than 60 countries, a staff of over 110,000 full-time equivalents and total assets of EUR 560.4 billion (as of 31 December 2003).
About I-Bizz

I-Bizz IT Services and Consultancy is a privately owned limited company that operates since 1998 and is legally based in Amsterdam, The Netherlands. It has one Full Time Employee. It offers a portfolio of services that ranges from architectural consultancy to technical project management. Its current specialization is in Knowledge Management and Management Information Systems.
Requirements

Business Issue
Deliverables
Package Selection
The Business Issue

- ABN AMRO CFO decision too much was spent decentrally designing MIS Applications - “Data Warehouse Models.”
- A centralized, Head Office department was tasked with developing and scoping an Enterprise Wide MIS Model
- The IBM BDWM was taken as a basis for the AAB MIS Model
Customers

- Our customers are ABN AMRO Business Units that need a data model for MIS Purposes
- We only do word-of-mouth advertising
- We are now 17 people strong but had to turn down some customers
- We only charge late implementations
Deliverables

- The main objective of a logical data model is to serve as a IT/business communication bridge.

- It is delivered in various forms, as a structured MS Word Document and in forms suitable for different data modelling tools like Cool:biz, and ERwin.

- A Prototype Cube proves validity of the model.
Logical Data Model

The logical model is always based on a requirements model. In this, e.g. the need for keeping history on an attribute/relationship level is specified.
Software Selection (August 2000)

- No package fit all requirements
- These are, amongst others
  - Multi User
  - Multi Project
  - Platform Independence
  - Integrated Versioning
- Decision to build; team of 2
Design

Meta Model

UML

Versioning & History

Relational & Object Tech

Persistence Engine
Relational Meta Model
Extended ER Meta Model
Design using UML

Works for Code and Database Design

From UML to NetRexx is totally straightforward

In fact, it is generated now by our system
Package Structure
Dyadic Relationship Modeling
(We say binary)

These relationship instances ...
... end up as the following propositions

\[\begin{align*}
O_{20013284} &= \{T = \text{DomainOnly}, N = \text{Employee}\} \quad (1) \\
O_{10009986} &= \{T = \text{DomainOnly}, N = \text{Project}\} \quad (2) \\
O_{22222222} &= \{T = \text{DomainOnly}, N = \text{Privilege}\} \quad (3) \\
O_{234234} &= \{T_{20013284}, N = \text{R.V.Jansen}\} \quad (4) \\
O_{456456} &= \{T_{22222222}, N = \text{Update}\} \quad (5) \\
O_{567567} &= \{T_{10009986}, N = \text{MetadataV}\} \quad (6) \\
R_{11} &= \{ST_{20013284}, OT_{22222222}\} \quad (7) \\
R_{22} &= \{ST_{11}, OT_{10009986}\} \quad (8) \\
R_{33} &= \{ST_{20013284}, OT_{22222222}, IOT_{10009986}\} \quad (9)
\end{align*}\]

This is the setup needed to assert the following facts:

\[\begin{align*}
R_{999} &= \{R_{11}, s_{234234}, o_{456456}\} \quad (10) \\
R_{1000} &= \{R_{22}, s_{999}, o_{567567}\} \quad (11)
\end{align*}\]
## Scope Specific Properties

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2637</td>
<td>Project explicitly involves Classification</td>
</tr>
<tr>
<td>10034282</td>
<td>Project explicitly includes Classification specifies value for Project Specifiable Relationship Type</td>
</tr>
<tr>
<td>10034283</td>
<td>Object is supplied value for Project Specified Relationship Type</td>
</tr>
<tr>
<td>10050690</td>
<td>Descriptor is supplied value for Project Specified Relationship Type (see above)</td>
</tr>
</tbody>
</table>
Physical Table Design

<table>
<thead>
<tr>
<th>OID</th>
<th>OBJ_TP</th>
<th>PPN_DT</th>
<th>PPN_USR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1166</td>
<td>1</td>
<td>2000-09-01</td>
<td>DI2359</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SBJ_OID</th>
<th>VERB_OID</th>
<th>OBJ_OID</th>
<th>RANK</th>
<th>OID</th>
<th>EFF_DT</th>
<th>END_DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1166</td>
<td>2763</td>
<td>4567</td>
<td>9999</td>
<td>2009765</td>
<td>2000-01-01</td>
<td>9999-12-31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dsctr_x_obj_oid</th>
<th>obj_oid</th>
<th>nm</th>
</tr>
</thead>
<tbody>
<tr>
<td>2710</td>
<td>1166</td>
<td>Involved Party</td>
</tr>
</tbody>
</table>

(Only Three Tables)
Components

The package structure conforms more or less to these components.

Three Tier Architecture with Application Server

Communication to VB & MS Office through SOAP (as a webservice)
Versioning Mechanism

- Relationship instances have effective date and end date
- Every version of Every relationship is saved, by only setting its end date and never physically deleting it
- Every historic deliverable is reproducible
- A project is a subtype of Event; one timestamp suffices to save an entire project version.
Graphical User Interface

Swing
Look & Feel
Netbeans
The Tree
Swing - Some Misconceptions

- Swing is ugly and slow
- Swing is hard to understand if you’re not into Smalltalk Model - View - Controller
- Swing does not look like Windows at all
- Swing can’t use threads
Actually ...

- You have to care for its appearance
- It is not slow at all - slowness is mostly your own doing
- You have to work on application startup to get it faster - multithread and load through introspection
We use the Alloy Look and Feel

- For Cross-Platform consistency
- Frames and Widgets look better than native Swing (ok, not on Macintosh)
- JTree behaviour is more consistent - it seems to be influenced by the look and feel component
All of GUI made with NetBeans

- Paint the screens
- Add the widgets
- Double click
- Add call to NetRexx code
- NetBeans is pure Java
- Works on all platforms
Adding calls to NetRexx methods

```java
private void exitMenuItemActionPerformed(ActionEvent evt) {
    try {
        this.bp.BPDisconnect(System.getProperty("user.name"));
    } catch (Exception s) {
        JOptionPane.showMessageDialog(this, s.getMessage());
    }
    System.exit(0);
}
```

Catch an event from a GUI Widget and call a method in a NetRexx Class
The Famous Tree GUI Design
by Kieran McKeown
# Relationship Types Screen Spec

The famous relationship types screen design by Mike George.
The Tree

- Is a JTree
- Action is in the tree Model
- Our most performance critical component
- Delivers to the user a scoped view to a shared object model
Right Clicking the Tree

- On every object in the repository
- Does scoping of the Common Model and additions and deletions
- Add Relationship Types
Adding a Relationship Type

Partitioning: [ ] (adds a scheme)
Object Type: Involved Party
Subj-Obj Verbphrase: abbreviates Object-Subject Role Name of
Obj-Subj Verbphrase: accrues
Subject Type: [Search ...]
Subj-Obj Cardinality: 0:M
Obj-Subj Cardinality: 1:1
Relationship Type Description:
Generic Object Editor

Edits any object thrown at it

Uses beanpatterns and introspection to determine type of editor
The Editor is a Visitor

do
  /* check whether the getter returns an object instance. if it
does, we pass it an EditorVisitor instance that handles the
editing this of course polymorphically with double dispatch
on the indirect object. */
  invokeResult = this.globalGetter.getMethod().invoke(this.globalObject, null)
  /* if the result from the Getter invocation is null, we
  * instantiate a new object and also have it accept an
  * editorvisitor.
  */
  if invokeResult = null then
    do
      do
        cls = Class.forName(this.globalGetter.getMethod().getReturnType().getName())
        clz = cls.newInstance()
        (Visited clz).accept(edV)
        edV.getEditor.setFont(this.dialogFont)
        ppp.validate()
        this.panel.validate()
      catch Exception
        say "Exception instantiating object-to-be-edited"
      end
    end
    else
      do
        (Visited invokeResult).accept(edV)
        edV.getEditor.setFont(this.dialogFont)
        ppp.validate()
        this.panel.validate()
Graphical Navigator Screen
The Server

Why?

RMI

Caching Objects

Deconstructing Objects
Why an Application Server?

- Needed robust multiuser access to the repository
- Handles transactions
- Caches Objects in storage
The Server Component

The Server is a text mode NetRexx-only Application that can run on any Java 2 VM. At startup the Relational storage is converted to Objects that are indexed and stored on Class and their participation in Relationships.
Option: strictcase
-- we do not usually use STRICTCASE but in this file it is necessary
-- because the Collections class has a static method synchronizedList
-- and also an inner class SynchronizedList etc.

package com.abnamro.midms.RepositoryTool
import com.abnamro.midms.modelbase
import com.abnamro.midms.modelbase.soap
import com.abnamro.midms.util.
import com.abnamro.midms.util.pool.
import com.abnamro.midms.util.exceptions.
import com.abnamro.midms.util.ssl.
import java.rmi.
import java.util.
import java.sql.

/**
 * The Business Process Server is the interface between the application layer
 * and the database, and between the data and the client application. &lt;br&gt;
 * The database is accessible via DataStore, but only the Business Process
 * Server may access it. A DBMS connection pool is set up, and every user
 * (client) of this server uses a separate connection lease to do the work;
 * this work can thus be individually committed or rolled back.&lt;br&gt;
 * SK 13092002: added security manager to authenticate the user connecting to
 * BPServer.
 * @see DataStore
 * @author &lt;a href=http://www.rvjansen.com target=new&gt;R.V. Jansen&lt;/a&gt;
 * @author &lt;a href=http://zaaf.tripod.com target=new&gt;A.J. Bos&lt;/a&gt;
 */

class BPServer extends UnicastRemoteObject implements BPServerInterface, BPClientInterface, ScopeChanging final

properties static inheritable
db = DataStore
oidObjectMap = ObjectMap
oidRelationshipTypeMap = RelationshipTypeMap
Object TreeMaps in Server

```java
public class BPServer extends UnicastRemoteObject implements BPServerInterface {
    properties static inheritable
    logger = Logger.getLogger("BPServer")
    sh = ConsoleHandler()
    db = DataStoreInterface
    oidObjectMap = ObjectMap
    oidRelationshipMap = TreeMap
    oidRelationshipTypeMap = TypeMap
    oidTypeMap = TreeMap
    oidSchemeMap = TreeMap
    oidRlnTpPhlGrpMap = TreeMap
    oidProjectMap = TreeMap
    oidProjectVersionMap = TreeMap
    oidScopeMap = TreeMap
    oidScopeVersionMap = TreeMap
    oidClassificationMap = TreeMap
    oidInvolvedPartyMap = TreeMap
    oidProjectSpecificMap = TreeMap
    verb0idObjRelsCache = TreeMap
    verb0idObjRelsCache = ArrayList
    sessionsMap = boolean
    oidPropertiesMap = LinkedHashSet
    TreeMap
```

![Diagram of Object TreeMaps in Server](image-url)
Update: Hook the local Setters

Meta Model Objects have NetRexx Indirect Properties

Setter Methods are overridden to do remote update
Object Decomposition

For every new domain value, 5 new relationships are added to the system, and as much scoping relationships as there are active scopes.

BPServer
Dissect and Disassemble into Relationships
Scope in Domain Value

New Domain Value

5 new Relationships

N Relationships

Data Base
The *bint* Language

*batch interpreter*

d*daughter* (arabic)

*a strict headmaster*
(in a story by F. Bordewijk)

*lower class girl*
(british slang, usage: outdated)

**Why?**

**Syntax**

**Examples**

**Antlr Grammar**

**Usage in Client**
Next to the Graphical User Interface, we also defined a specialized metadata access language, called bint. Due to the use of the generic universal relation, SQL access to the metadata soon becomes very cumbersome. Bint uses the objects after they are assembled in the Object Factory.
Syntax of **bint**

Syntax is very loosely inspired by CMS Pipelines, APL and Predicate Logic.

Works on stacked maps of **key**:Oid and **value**:Object.

The data appears to flow through pipes with stages:

- Collect
- Subset
- Negate
- Union
Interaction between Antlr and NetRexx in construction of `bint`
Infrastructure and Security

Platform
Development Environment
Middleware Components
RMI & SSL
JNI Based User Authentication
Platform

Due to the Application being all NetRexx, it runs unchanged on Windows NT, MacOS X, Windows XP and Linux.

No testing has been done on z/OS, but we are confident that it will run within a day.

Production Server is a Dual Xeon with 1 GB storage under NT 4.
Development Environment

Emacs, Java, NetRexx, Netbeans, CVS, Make

Switched recently to Subversion

On NT, we used Cygwin (so we have (the taste of) Unix anywhere)

NetRexx editor mode is very important (color, indentation) (Elisp, cross-platform)

The NetRexx Compiler Server
DBMS

DB2 UDB 8.1 is used for production
Development and regular testing takes place on the Open Source DBMS’s ses MySQL and PostgreSQL

The Server Component can run off MS-Access via ODBC-JDBC Bridge (for people who need to travel with copies on generic Windows Laptops)

All DBMS access is done by Server Component using JDBC in NetRexx
Middleware

Tomcat is used as Servlet Container

Axis is used for SOAP access to server

We just migrated these to JBoss
RMI and SSL Encryption

```java
package com.abnamro.midms.util.ssl
import java.io.*
import java.net.*
import java.rmi.server.*
import javax.net.ssl.*
import java.security.KeyStore
import javax.security.cert.X509Certificate

// mind: if this does not compile you probably do not have jsse.jar on your classpath
class RMISSLServerSocketFactory extends RMIServerSocketFactory, Serializable

method createServerSocket(port=int) returns ServerSocket signals IOException

do

    -- set up key manager to do server authentication
    passphrase = char[]
    passphrase = "passphrase".toCharArray()

    ctx = SSLContext.getInstance("TLS")
    kmf = KeyManagerFactory.getInstance("SunX509")
    ks = KeyStore.getInstance("JKS")

    ks.load(ClassLoader.getSystemClassLoader().getResourceAsStream("dmskey"), passphrase)
    kmf.init(ks, passphrase)
    ctx.init(kmf.getKeyManagers(), null, null)
    ssf = ctx.getServerSocketFactory()

    // do stuff
```

```python
```
JNI Based User Authentication

```c
JNIEXPORT jstring JNICALL Java_com_abnamro_midms_platform_NTPlatformSecurity_getUserId(JNIEnv *env, jobject obj)
{
    jchar * buffer[100];
    DWORD length = 100;
    NET_API_STATUS rc = 0;
    rc = GetUserName((LPTSTR)buffer, &length);
}
```

/** This Class is the implementation of the methods of PlatformSecurity for the Windows NT Operating System. Most of the methods are native WIN32 code, implemented in NTPlatformSecurity.c and loaded from NTPlatformSecurity.dll. It is know to run on NT and W2K */

class NTPlatformSecurity extends PlatformSecurity

    /** This constructor loads the NTPlatformSecurity Dynamic Link Library for Windows NT */
method NTPlatformSecurity()

    System.loadLibrary("NTPlatformSecurity")

/** This method overriues PlatformSecurity.getUserId() and is implemented in a native method, which uses the call <code>
    rc = GetUserName((LPTSTR)buffer, &length);
</code>
to get its results. This is deemed more secure than using the Java System Property user.name */

method getUserId() native returns String
### Post Mortem

<table>
<thead>
<tr>
<th>These points were moot since we had a working prototype database in MS-Access</th>
<th>We wasted some time by being not generic enough, for examples in the hooks in the setters - we invented class metadata in BeanInfo objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>That sometimes failed spectacularly, for example when adding RI dropped a random index ...</td>
<td>We should have used more association objects instead of collections in classes - only one place to update and less RMI trouble</td>
</tr>
<tr>
<td></td>
<td>Dyadic relationship modelling is hard</td>
</tr>
</tbody>
</table>

**Our three-tier NetRexx solution never failed yet.**
Questions?

- Ask them now
- By email
  rene.vincent.jansen@nl.abnamro.com
- Or alternatively
  rvjansen@xs4all.nl

Thank you very much for your attention!