CenterPiece and Object Oriented REXX

Sandy Syx
Mantissa Corporation

Pages 150-173

Mantissa Corporation

CenterPiece and Object Oriented REXX

Sandy Syx - ssyx@mantissa.com
205-945-8930

1

Introduction

- ➢ *Mantissa Corporation*
- ➢ *Data Center Automation software products since 1981.*

- ➢ *RMS "The Report Management System"*
- ➢ *OPS "Operations Productivity System"*
- ➢ *FYI "Windows/LAN-based Document/Image Management and more"*

2

Agenda

- *CenterPiece Architectural Overview*
- *CenterPiece Built-in Classes*
- *CenterPiece Object-Oriented REXX*
- *REXX Improvements for Complex Problems*
- *Developing CenterPiece Classes*

3

CenterPiece Architectural Overview

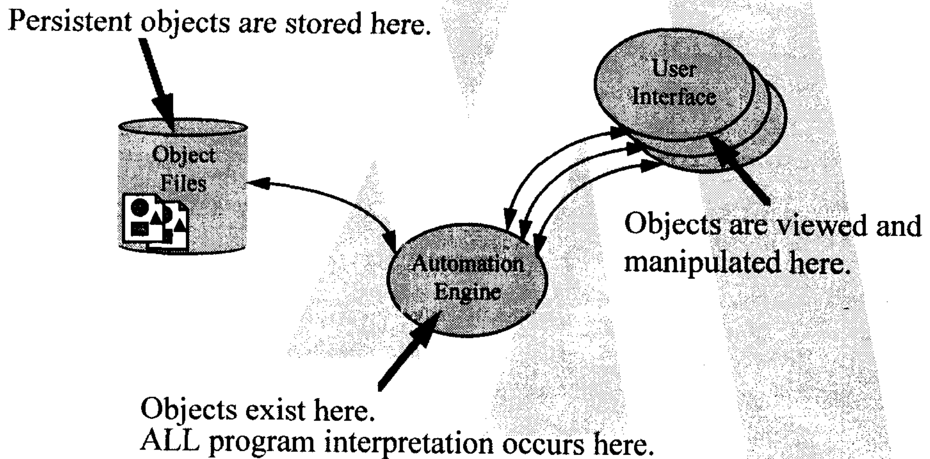
4

What is CenterPiece?

- ***CenterPiece is a Distributed, Multi-platform, Object-Oriented, Interpretive, Development and Runtime Environment.***
- ***Two main Executables:***
  - ◆The Engine - a multi-threaded interpreter that serves objects to multiple simultaneous clients in psuedo realtime.
  - ◆The User Interface - A graphical application that allows one to view and manipulate objects that exist in an "engine".

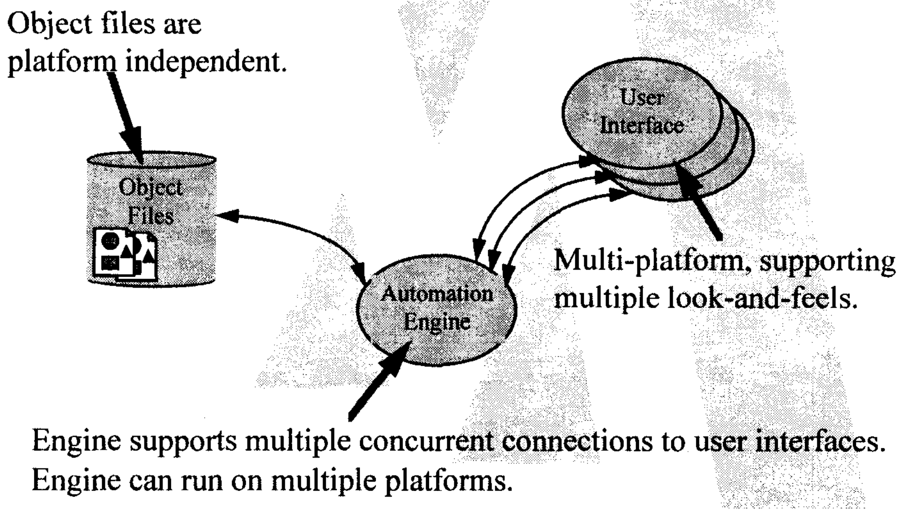
5

CenterPiece Architecture


Persistent objects are stored here.
Object Files
User Interface
Automation Engine
Objects are viewed and manipulated here.
Objects exist here.
ALL program interpretation occurs here.

6

CenterPiece is Multi-platform


Object files are platform independent.
Object Files
User Interface
Automation Engine
Multi-platform, supporting multiple look-and-feels.
Engine supports multiple concurrent connections to user interfaces.
Engine can run on multiple platforms.

7

The Engine Is...

- *The heart of the system.*
- *An object server.*
- *A multi-threaded object oriented REXX executor.*
- *Basically event driven.*
- *Responsible for reading and writing object files.*
- *Not visual.*

8

The User Interface

- ➢ *Graphical User Interface*
- ➢ *Runs on multiple platforms and window systems (X-Motif, OS/2 Presentation Manager, MsWindows)*
- ➢ *Supports multiple look-and-feels (Motif, CUA, Windows)*
- ➢ *Very interactive allowing direct manipulation (object menus and drag-and-drop).*

CenterPiece Built-in Classes

Fundamental Built-in Classes

- ***Workspace***
  $2^{1/2}$ Dimensional Visual Container of WorkspaceObjects.
- ***WorkspaceObject***
  Gives objects the ability to be on a workspace. (Name,X,Y,Layer, Icon,Workspace, etc...)
- ***Class***
  Allows one to create new classes.

11

Programmer's Helper Classes

- ***Program***
  Allow interpretation and execution of REXX logic.
- ***Thread***
  Instance of executing program.
- ***List - Ordered collection of items.***
- ***Dictionary - Unordered collection of key/data pairs.***
- ***Semaphore - Resource Arbitor***
- ***Queue - Object version of REXX queues***

12

Simple Visual Objects

- ➢ *Text*
  Floating text. (font, color, angle)
- ➢ *Line*
  Line segments. (X2,Y2,width,color)
- ➢ *Rectangle*
  Hollow or filled rectangles (width, height,fillcolor)
- ➢ *Image*
  2D color images. Can be large and deep.

13

Dialog Objects

- ➢ ***Button - Action button that runs a "Click" method when pressed.***
- ➢ ***Checkbox - State selector runs a "Click" method when pressed.***
- ➢ ***TextEntry - Text entry field, allows multi-line, scrollbars, etc.. Runs a "Changed" method when the text is entered.***

14

More Dialog Objects

- ***ListBox** - Combination of a List and a ListView. Visual list, allows images and text. Items can be dragged from the list.*
- ***Slider** - Allows a value to be selected within some range. Runs a "Slide" method when the slider is slid.*
- ***RadioGroup** - Mutually exclusive group. Runs a "Click" method when the selection changes.*
- ***Spinner** - Allows spinning or typing in a number from a specified range of values.*

15

Communication Classes

- ***MTAServer***

  Message Transport Agent - allows one to create a "server" that will listen for connections from "clients" at any number of access points (tranport,port). Allows telneting into the server if tcp is used.
- ***MTAClient***

  Allows one to connect to a server to exchange messages.

16

Object Storage to Disk

➢ *ObjectFile*
Saves all owned objects to a disk file.

17

Application Delivery

➢ ***UserProfile - This class allows one to secure access to a CenterPiece engine by defining exactly who can connect, and how they connect. Users can be "Developers", "EndUsers" or both. An "EndUser" has a "Connect" method that can be overridden to show the appropriate application dialogs for the user on connection to an engine.***

18

CenterPiece Object-Oriented Extensions to REXX

19

Objects

- ➢*Objects are instances of some Class.*
- ➢*Objects have any number of attributes.*
- ➢*Objects are globally visible.*
- ➢***Every object has a universally unique immutable identifier.***
- ➢*Any object can be made persistent.*

20

Object Ownership

- *Objects can own any number of other objects.*
- *An object can have at most one owner.*
- *When an object is destroyed, all of its owned objects are also destroyed.*
- *When an object is saved, all of its owned objects are saved.*

Attributes

- *Attributes act much like REXX variables.*
- *They can be simple or compound.*
- *Object attributes must be defined in some superior class.*
- *Attributes names are case and space preserving, but case and space insensitive.*

Referencing Objects

- ***Objects have global visibility.***
- ***Each object is unique not because of its name, class, nor attribute values, but because of its universally unique immutable identifier (UUID). These are normally just called <u>object identifiers</u> or <u>object-ids</u>.***
- ***Objects are referenced by REXX variables that have an object-id as their value.***

Attribute Access

Object Attributes are selected with a double-dot (..).

object..attribute

$$\begin{tikzcd} \textbf{object identifier or, classname} & \arrow[l] \arrow[r] & \textbf{attribute specification} \end{tikzcd}$$

The symbol to the left of the double-dot is translated into a value. The translated value must be an object-id or a class name.

The symbol to the right (up to the next double-dot) is treated exactly like a variable symbol and must reference an object or class member.

Attribute Access Examples

Simple Attribute Access
Assume b is an object of the Button class.

```
b..Name = "Press Me"
b..BackgroundColor = "maroon"
```

Multiple Indirections
Assume that b is a Button, and assume that the button has an attribute "Workspace" that references an object of the Workspace class that the button is on. The name of the workspace could be accessed by:

```
b..Workspace..Name
|└─┘the button   |
└─ the button's workspace
```

Object Creation/Destruction

Accomplished with two new REXX built-in functions:

object_id = ObjectCreate(<classname>)

rc = ObjectDestroy(<object_id>)

For Example,

```
    aLine = ObjectCreate( "Line" )
    aLine..x = 100
    aLine..y = 100
    aLine..x2 = 200
    aLine..y2 = 200
    ...
    call ObjectDestroy aLine
```

Classes

- ➢*Classes define attributes that each instance of the class will have.*
- ➢*CenterPiece allows multiple inheritance.*
- ➢*Classes are objects and are instances of the "Class" class.*
- ➢*Classes are typically used by their name.*

27

Inheritance Model

- ➢*Attributes are inherited dynamically.*
- ➢*A class can be modified "on the fly" with existing instances.*
- ➢*Attribute lookup precedence:*
  1. Local Object Override
  2. Object's Class
  3. Primary Superclass--->Root Class
  4.Secondary Superclass--->Root Class
  In other words: "A Depth first, breadth next search up the class hierarchy".

28

Dropping Attributes

The REXX - DROP instruction can be used to cause an attribute to revert to its class default.

For example, assume that a class "Author" exists which has an attribute named "Name" that has a class default value of "anonymous".

```
anAuthor = ObjectCreate( "Author" )
anAuthor..Name = "Fred Brooks"
say  anAuthor..Name   ==> would print "Fred Brooks"
drop anAuthor..Name
say  anAuthor..Name   ==> would print "anonymous"
```

Object Related Built-in Functions

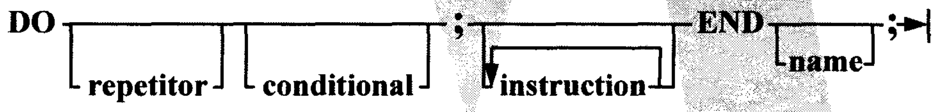
- ***ObjectCreate***
- ***ObjectDestroy***
- ***ObjectClone***
- ***ObjectFindOfClass***

- ***IsObject***
- ***IsObjectOfClass***
- ***ClassOfObject***
- ***ClassIsSubclassOf***
- ***ClassIsDirectSubclassOf***

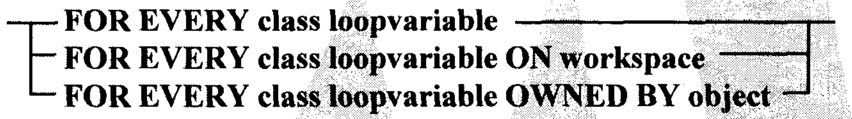
- ***ObjectOpen***
- ***ObjectOpenAsDialog***
- ***ObjectClose***
- ***ObjectGoto***
- ***ObjectGetOwner***
- ***ObjectSetOwner***

- ***ObjectFileOpen***
- ***ObjectFileSave***
- ***ObjectFileClose***

Iterating Over Objects



repetitor (extensions):


FOR EVERY class loopvariable
FOR EVERY class loopvariable ON workspace
FOR EVERY class loopvariable OWNED BY object

31

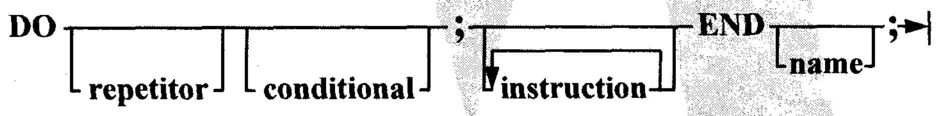
Object Iteration Example

Iterating Over All Objects of a Given Class

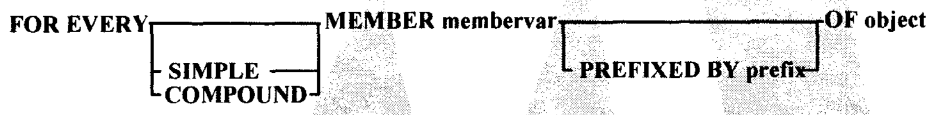
```
num_employees = 0

DO FOR EVERY Employee e
   SAY e..Name
   num_employees = num_employees + 1
END
```

32

Object Member Iteration



repetitor (extensions):



Composite Objects

- ***"An object by itself is intensely uninteresting". - Grady Booch***
- ***Object Identifiers behave much like pointers to structures in 'C' or 'C++'.***
- ***Any object attribute can contain an object identifier of another object.***
- ***Composite objects can be made in which one object references and owns any number of other objects.***

Embedded Objects

- ***It is possible to embed objects within other objects.***
- ***This must be done by adding a class member that references an object of a specified class.***
- ***The embedded object will be cloned for each instance of the class.***
- ***The embedded object may not be destroyed independently of its owner.***

Methods

- ***Methods are simply objects of the Program class that are referenced by some attribute of an object.***
- ***Method invocation is no different than calling any other REXX function or subroutine. The method is addressed just like any other object attribute, except that it is used where a function or subroutine name would normally be used.***

Self Reference In Methods

The double-dot with no prefix is an object self reference inside an object method.

For example, imagine a user interface Button method that runs when the button is clicked.

```
/* begin Button..Click */
   ..Name = "Hello"
return 0
```

In this example the double-dot with no prefix means "this" button.

REXX Extensions for Complex Problems

Multi-Threaded REXX

- ***An additional built-in function, Start(), is provided to allow one thread to start another.***
- ***Each thread executes concurrently.***
- ***Threads are re-dispatched, basically, after each source instruction.***

39

Unwinding the stack on a Raised Condition

Normal REXX, strangely, doesn't unwind the call stack when a condition (exception) is raised.

We extended the CALL ON and SIGNAL ON instruction to allow them to be prefixed with the keyword UNWIND.

For example,

```
UNWIND CALL ON syntax NAME mysyntaxtrap
...
mysyntaxtrap:
  say "Tarfu"
return
```

40

Developing CenterPiece Classes

41

Modularity

- *Instances do not have to be saved in the same ObjectFile as their classes.*
- *Classes do not have to be saved in the same ObjectFile as their superclasses.*

42

Constructors/Destructors

- ➢*Any class can have a "Create" method. Simply add an attribute named Create and make it a sub-object that is of the Program class.*
- ➢*The method will automatically be run when an instance of the class is created.*
- ➢*Ditto for "Destroy" and "Load" which will be run when the object is destroyed or loaded from an object file, respectively.*

User Events

- ➢*Many classes have methods that are run in response to user actions.*
- ➢*These methods are optional, and if not provided, a default built-in action occurs in response to the user event.*
- ➢*Some examples are:*
  WorkspaceObject..Drop or DroppedUpon
  Button..Click
  TextEntry..Changed

User Events - Continued

- ***The first argument to a user event method is always a Dictionary object that contains entries that indicate what happened.***
- ***The attributes present in the context dictionary depend on the event.***
  For example, a Drop event would have the new X and Y locations of the object dropped.