ooRexx 5 Yielding Swiss Army Knife Usability

2019 – International Rexx Symposium

Hursley, September 2019

Rony G. Flatscher (Rony.Flatscher@wu.ac.at, http://www.ronyRexx.net)
Günter Müller (mueller@iig.uni-freiburg.de)
Wirtschaftsuniversität Wien, Austria (http://www.wu.ac.at)

Overview

- Brief history
- Bird-eyes view of ooRexx
- New Features in ooRexx 5
- Roundup

Brief History, ooRexx, 1

- 1979 – Birthday of REXX
  - Developed at Hursley by Mike F. Cowlishaw
- Became of strategic importance to IBM
  - SAA REXX for all IBM operating system platforms
- 1996 – ANSI (INCITS 274-1996) REXX standard!
- Development of Object REXX
  - Original lead: Simon Nash at IBM Hursley
  - Development turned to the US, lead: Rick McGuire
  - 1996 released with OS/2 Warp, versions for Windows, AIX, Linux followed

Brief History, ooRexx, 2

- Fall 2004
  - RexxLA and IBM announce transfer of Object REXX source code to RexxLA
    - Rick McGuire remained active in the opensource project and has been instrumental for the evolution of the programming language
- March 2005
  - "ooRexx 3.0" source and binaries released by RexxLA
  - "ooRexx", acronym for "open object Rexx"

Brief History, ooRexx, 3

- Fall 2009
  - ooRexx 4.0 released
    - New, rewritten kernel
    - New native API comparable in features and power to Java's native interface APIs ("JNI")
      - Fully exploited by the ooRexx function package "BSF4ooRexx" which bridges ooRexx and Java
    - For the first time possible to create 32- and 64-bit binaries from the same source

Brief History, ooRexx, 4

- February 2014
  - ooRexx 4.2 released
- Ever since then work on ooRexx 5.0 has been carried out
- As of the Hursley symposium in 2019, ooRexx 5 is
  - Stabler than 4.2
  - Faster than 4.2 (20% to 2000%)
  - Multithreading support improved considerably
  - Many useful new features over 4.2 making it altogether a "swiss army knife (SAK)" of programming!

Bird-Eyes View of ooRexx, 1

- Compatible to REXX
  - Mandated by IBM customers who did not want to have to rewrite existing REXX code for Object REXX
- Added object-oriented features like
  - Ability to define classes with methods and attributes
  - Multithreaded execution of Object REXX programs
  - Interfacing with other object-oriented technologies in the industry, especially from IBM, e.g.
    - SOM: system object model

Bird-Eyes View of ooRexx, 2

- Still "human-oriented" design like REXX
  - Easy syntax, easy to learn
- Using the message metaphor (cf. Smalltalk)
  - *Everything* is an object
  - An object is like a living thing
  - One interacts with an object by *sending* it *messages*
  - The *object will look for a method* by the name of the received message *and invokes it*
  - Any return value will be returned *by the object*

Bird-Eyes View of ooRexx, 3

- An example

```
test="12345"          /* a string */
say "REXX-style (function invocation): say reverse(test)"
say reverse(test)     /* invoking the built-in string function "reverse" */
say

say "ooRexx-style (sending a message): say test~reverse"
say test~reverse      /* invoking the string method "reverse"          */
```

- Output

```
REXX-style (function invocation): say reverse(test)
54321

ooRexx-style (sending a message): say test~reverse
54321
```

Bird-Eyes View of ooRexx, 4

- At the same time both coding styles possible
- Message paradigm
  - Quite easy to understand (*very easy* for beginners)
  - Decoupling the method invocation
    - Very dynamic solutions at runtime possible
      - E.g. rerouting of messages at runtime
  - Easy to understand inheritance as conceptually
    - The receiving object will look for a method by the same name as the received message by walking up the inheritance tree until it finds one and executes it, otherwise the object raises an error condition

New Features in ooRexx 5, 1

General

- Build-system changed from *autotools* to *CMake*
- Interpreter can be used without administrative rights
  - USB stick solutions become possible
- Significant performance gains (from 20% up to 2000%)
- New package confined local environment
- New *package* scope for methods
- New isNil method for root class Object
- Namespaces introduced (prefix rexx: for ooRexx)

New Features in ooRexx 5, 2
New Array Notation, 1

- An array gets created
  - Explicitly as an instance of the Array class, e.g.

```
arr=.array~new
```

  - *Implicitly* with the new array notation
    - A comma separated list of values in parentheses
    - Parentheses can be omitted if the context expects a single collection object

New Features in ooRexx 5, 3
New Array Notation, 2

- Example

```
arr="one", "two", "three"  -- define an array
do item over arr  -- iterate over items
   say item       -- display item
end
```

- Output

```
one
two
three
```

New Features in ooRexx 5, 4
New Variable Reference Notation, 1

- Variable Reference
  - Can be created with either the new **<** or the **>** prefix
  - Yields an instance of type *VariableReference*
    - Name and value of the variable can be fetched
    - Value of the variable can be set
- Fetching arguments as variable reference
  - USE ARG variables prefixed with either **<** or the **>**
  - Local variables actually refer to the caller's variable reference

New Features in ooRexx 5, 5
New Variable Reference Notation, 2

- Example

```
a="hello!"      -- hello!"
call work >a    -- create and supply a variable reference
say a           -- refers to string "from the work routine"

::routine work
  use arg >tmp -- "tmp" now represents the variable "a"
  tmp="from the work routine"
```

- Output

```
from the work routine
```

New Features in ooRexx 5, 6
New Directives, 1

- ooRexx directives
  - At the end of a program ("package")
  - Led in with two colons "::"
  - Contract with the interpreter
    - Interpreter carries out all directives *before* starting the program with executing the statement starting with line one
    - ::attribute, ::class, ::constant, ::method, ::options, ::requires, ::routine
- New "::ANNOTATE" directive
  - Allows to annotate a package (program), routines, classes, attributes, methods and constants

New Features in ooRexx 5, 7
New Directives, 2

- New "::RESOURCE" directive
  - Allows to store *any* (even multiline) text
    - Binary data could be stored in *base64* encoded form
      - Use the String method encodeBase64 for encoding
      - Use the String method decodeBase64 for decoding
  - "::END" directive serves as the delimiter
  - All resources will be stored in a StringTable
    - Environment symbol .resources returns it
    - Text of each resource will be stored in an array

New Features in ooRexx 5, 8
New Directives, 3

- Example

```
say "greetings:"
say .resources~greetings
-- fetch the string array turn it to a plain string, decode
say .resources~secret~makeString~decodeBase64

::resource greetings -- note the empty lines

  Hello,
  REXX 2019!

::END

::resource secret    -- base64 encoded
b29SZXh4IGlzIGNvb2whIDop
::END
```

- Output

```
greetings:

  Hello,
  REXX 2019!

ooRexx is cool! :)
```

New Features in ooRexx 5, 8 "ADDRESS ... WITH", 1

- ANSI REXX defines the optional WITH subkeyword for the ADDRESS keyword instruction
  - Allows to redirect stdin ("input"), stdout ("output") and stderr ("error") from/to stems
  - ooRexx in addition allows redirections from/to streams and collections
  - On output one can *replace* or *append* the data

New Features in ooRexx 5, 9

"ADDRESS ... WITH", 2

- Example (list environment variables in sorted order)

```
"set | sort"    -- command to environment
say "RC="rc     -- display return code
```

- Output (on Windows, maybe)

```
ACPath=C:\Program Files (x86)\Lenovo\Access Connections\
ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\Administrator\AppData\Roaming
… cut …
RC=0
```

New Features in ooRexx 5, 10 "ADDRESS ... WITH", 3

- Example (list environment variables in sorted order)

```
out=.array~new -- create array to retrieve data
   -- command to environment
address system "set | sort" with output using (out)
do i=1 to 3
   say "out["i"]="out[i]
end
say "RC="rc    -- display return code
```

- Output (on Windows, maybe)

```
out[1]=ACPath=C:\Program Files (x86)\Lenovo\Access Connections\
out[2]=ALLUSERSPROFILE=C:\ProgramData
out[3]=APPDATA=C:\Users\Administrator\AppData\Roaming
RC=0
```

New Features in ooRexx 5, 11

"ADDRESS ... WITH", 4

- Example (use operating system sort command)

```
in ="Tracy","Angie","Berta"                    -- input data
out=.array~new                                 -- output data
   -- command to environment use ooRexx arrays as stdin and stdout
address system "sort" with input using (in) output using (out)
say "RC="rc          -- display return code
do item over out     -- iterate over all items of array
   say item          -- display items
end
```

- Output

```
RC=0
Angie
Berta
Tracy
```

New Features in ooRexx 5, 12
"DO" and "LOOP", 1

- New subkeywords "WITH [INDEX idx] [ITEM val]"
  - Allows to iterate over collections and optionally assign multiple loop variables
    - The *index* value of the collection to a loop variable and
    - The *item* value of the collection to another loop variable

New Features in ooRexx 5, 13
"DO" and "LOOP", 2

- Example

```
ibmers=.stringTable~new      -- faster "Directory" collection class
ibmers["Les"]="Koehler (RIP)"
ibmers["Mike"]="Cowlishaw"
ibmers["Rick"]="McGuire"
ibmers["Simon"]="Nash"
ibmers["Walter"]="Pachl"

say "IBMers who have been closely ;) related to REXX:"
    -- iterate over (unordered) collection, use two loop variables
do with index firstName item lastName over ibmers
   say firstName"," lastName            -- show index and item values
end
```

- Output (random order)

```
IBMers who have been closely ;) related to REXX:
Les, Koehler (RIP)
Rick, McGuire
Simon, Nash
Mike, Cowlishaw
Walter, Pachl
```

New Features in ooRexx 5, 14
"DO" and "LOOP", 3

- New subkeyword "COUNTER c"
  - Allows to supply a counter that starts with "1" and gets increased by 1 at the end of each loop
  - Enables counting in contexts where a numerical loop variable cannot be defined like in "DO … OVER …"

New Features in ooRexx 5, 15
"DO" and "LOOP", 4

- Example

```
   -- new: "of" class method for all kind of collections!
board=.Directory~of(("Chip","Davis"),("Gil","Barmwater"),("Jon","Wolfers"),-
        ("Les","Koehler (RIP)"), ("Mark","Hessling"), ("Mike","Cowlishaw"),-
        ("Pam","Taylor"),("Rene","Jansen"), ("Walter","Pachl"))

say "RexxLA board members:"
   -- also show iteration count while looping over the (unordered) collection
loop counter i with index firstName item lastName over board
   say "#" i":" lastName"," firstName
end
```

- Output (random order)

```
RexxLA board members:
# 1: Wolfers, Jon
# 2: Koehler (RIP), Les
# 3: Hessling, Mark
# 4: Cowlishaw, Mike
# 5: Pachl, Walter
# 6: Barmwater, Gil
# 7: Taylor, Pam
# 8: Davis, Chip
# 9: Jansen, Rene
```

New Features in ooRexx 5, 16
Further Improvements

- SELECT keyword instruction
  - New: accepts an expression
  - WHEN instructions only list the resulting expression values they are intended for
  - Comparable to NetRexx
- USE keyword instruction
  - New subkeyword LOCAL followed by a list of local variables in method routines
    - All other variables in the method routine are defined to be attributes of the class

New Features in ooRexx 5, 17 New Classes, 1

- AlarmNotification (multithreading related)
  - Allows notification when an alarm gets triggered
  - Abstract method triggered must be implemented
- MessageNotification (multithreading related)
  - Allows notification when an asynchroneous message's method completed execution
  - Abstract method messageCompleted must be implemented
- Ticker (multithreading related)
  - Allows notifications to be constantly sent at a given interval

New Features in ooRexx 5, 18
New Classes, 2

- EventSemaphore (multithreading related)
  - Allows to synchronize Rexx threads
  - Once posted all blocked threads resume execution
- MutexSemaphore (multithreading related)
  - Allows to synchronize Rexx threads
  - When a thread completes, one of the blocked threads resumes execution

New Features in ooRexx 5, 19 New Classes, 3

- RexxInfo
  - Its methods return the current settings of ooRexx, e.g
    - date, maxPathLength, platform, revision, version, …
- Validate
  - Eases validating arguments considerably
    - Of a certain class, a certain type (e.g., whole number, logical value), ...
- VariableReference
  - Represents a variable reference (result of applying the new ">", "<" operators to a variable)

New Features in ooRexx 5, 20

- rexxc[.exe]
  - Compiles Rexx source code to binary representation
  - Speeds up loading of Rexx programs, hides source code
  - New undocumented trailing switch "/E"
    - Encode binary representation as *base64*
    - Allows loading and running compiled Rexx programs via scripting frameworks, e.g.,
      - Java scripting framework expects text-only programs
      - Binary data would cause character set translations
        - Would inadvertently destroy the program!
      - *base64* encoded binary data would remain intact

Roundup

- ooRexx 5
  - Since 2014 in the works
    - Great speed improvements
    - Great new functionalities in different areas of the language
    - Still easy to learn and to use
    - Windows, Linux, MacOS
    - USB-stick versions possible, finally!
      - Allows creation of SAK-ooRexx-USB-sticks!
  - Can be compiled for IBM mainframes !
    - Use mainframe ooRexx with BSF4ooRexx on "Linux on Z"!
      - E.g. write ooRexx code to interact directly with DB2 !

URLs

- RexxLA-Homepage (non-profit SIG, owner of ooRexx, BSF4ooRexx)

  <http://www.rexxla.org/>
- ooRexx 5.0 beta on Sourceforge

  <https://sourceforge.net/projects/oorexx/files/oorexx/5.0.0beta/>
- BSF4ooRexx on Sourceforge (ooRexx-Java bridge)

  <https://sourceforge.net/projects/bsf4oorexx/>
- Introduction to ooRexx (254 pages)

  <https://www.facultas.at/Flatscher>
- JetBrains "IntelliJ IDEA", powerful IDE for all operating systems
  - <https://www.jetbrains.com/idea/download>, free "Community-Edition"
  - Alexander Seik's ooRexx-Plugin with readme (as of: 2019-08-27)
    - <https://sourceforge.net/projects/bsf4oorexx/files/Sandbox/aseik/ooRexxIDEA/beta/1.0.5/>