Introducing Rexx2Nrx

© Thomas Schneider, IT-Consultant

www.Rexx2Nrx.com

Rexx LA meeting

Triangle Research Park, NC

May 2002

Rexx2Nrx: the classic Rexx to NetRexx converter

- Transforms classic Rexx programs to NetRexx
- TOP-down or BOTTOM-up
- step-wise transformation possible
- written in IBM CMS compiled REXX
- available in IBM compiled Rexx, NetRexx, & as Java classes

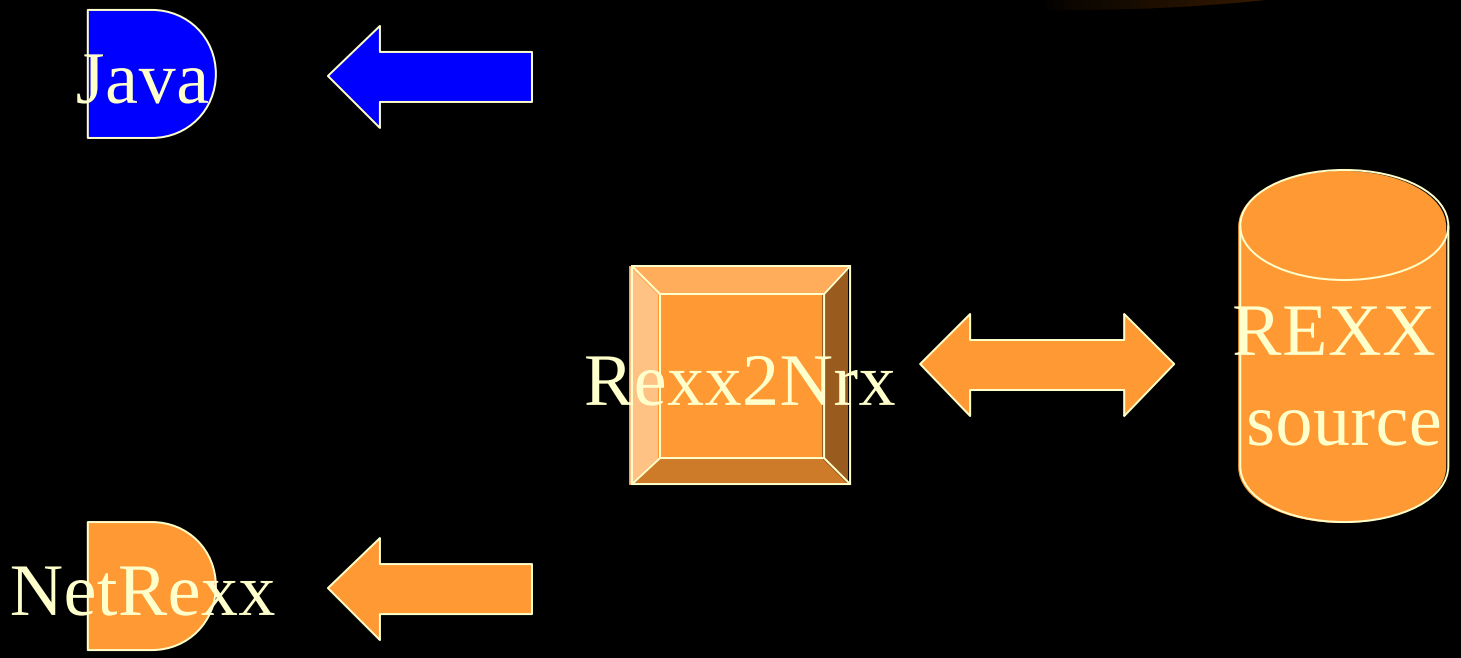
Including the RUN-Time package (Rexx2Nrx.Rexx2RTP)

- RexxTime: Simple functions like date(...) and time(...)
- RexxFile: stream I/O (stream, linein, lineout, lines, charin, charout, chars)
- File I/O utility functions (fileid, parsefid, open, scratch, extend, close, exists, etc)
- RexxMsg (askfor, info, error, warning, abort,...)

... And more

- RexxStk: Stack I/O (pull, push, queue, queued)
- RexxBits: bitand, bitor, bits, ...
- SysCmd: executing System commands
- StrFunc: various useful String Functions
- ... And more

The Transformation Process



Two major steps used

- 1.) The Rexx-Parser (scans and parses Rexx source code)
- ... and ...
- 2.) The actual Rexx to NetRexx translator.

Intermediate results

- The Internal ‚item-declarations‘ are used for storing all the attributes of the various items (multiple languages supported).
- ... and ...
- The Internal Code, again used for various languages of the PreProcessor (‚PP‘)

Objectives of the Parser

- Scanning (tokenizing) of the source code.
- Parsing of the various language statements
- Builds table of all ‚items‘, and a concise, language independent, Code for further analysis
- Items held in Core
- Code currently goes to external file
- Items currently saved at end of Parser

Item-attributes (1)

- Name (case of spelling retained)
- Scope (Global, Local, Parameter, Internal, External, Visible, ...)
- Type (Constant, Id, Var, Stem, Label, Function, Subroutine, Method, Expression, Comparison, Condition,...)
- Class (Number, Text, FileName, Date, Nothing, unknown, ...)

Item-attributes (2)

- initial Value, if available
- Value-Range (domain, list of assigned items)
- Note (short description attached)
- Parent Name (used for packages)
- ... And more when needed for other languages (like COBOL or PL/I)

Item-attributes (3)

- Picture (COBOL/ PL/I)
- Level (COBOL / PL/I)
- Image (e.g. for Dates)
- Header (for Listings)
- Label (for Screens)
- ...

Example : V2PAY.REXX

- Translates old Austrian banking format (V2) to ‚EDIFACT‘ PAYMUL-Transaction
- Just an example of a ‚NOT trivial‘ REXX program
- used to demonstrate the steps of the converter.

Example: what has to be changed (from REXX to NetRexx)

- Strings (backslash used as ESCAPE char in NetRexx/Java, must be duplicated)
- operators („!“ and „!!“ Used instead of „|“ and „||“ in german CMS REXX)
- Stem notation --> subscripted Rexx Strings
- label notation --> method ...
- call abc params --> abc(params)
- etc, etc, ...

Example : what has to be changed (2)

- Public properties have to be declared in front of the program logic
- Typing (at least for Rexx Stems) needed
- default value assignment
  - needed for NOVALUE items (Id‘s)
  - desired for function arguments and variables
  - transport NOTES from first usage to declaration

Critical areas (1)

- Variable, Stem and Function names ***may be*** the same in Rexx (but denote different entities in Rexx)
- Variables (Properties), Qualified Vars (arrays) and methods must all have different names in NetRexx
- ==> renaming algorithm necessary

Critical areas (2)

- Function ‚Parameters‘ (ARGs) in Rexx remain visible outside the function body
- ... even when it is not good practice!
- Parameters in NetRexx are only visible in the body of the same routine (are Local!)

Critical areas (3)

- ‚argument Lists‘ (comma separated) and Parse Instructions might be mixed in one clause.
- Subscripted Strings (Stems) are NOT allowed in a PARSE instruction in NetRexx (ok in Rexx).
- ... Resolved by ‚pending parse‘ instructions code generation!

Critical areas (4)

- Multiple ARG/PARSE ARG instructions allowed in Rexx
- ARG(n) function returns n.th argument
- missing arguments allowed
- ==> implemented by argument-array $arg[n] and multiple method generation when needed.

Critical areas (5)

- All NetRexx-references to ‚builtin‘ Functions have to be object oriented (change in notation necessary!).
- E.g. words(strv) becomes strv.words()
- format(a,b,c) becomes a.format(b,c)

- VERY cumbersome, when done manually!

Critical areas (6)

- Java (and NetRexx) don‘t have a ‚go to‘ statement.
- Resolution:
  - „SIGNAL my_label“ (in Rexx)
  - becomes „my_label();return ...“ (in NetRexx)
- looks quite unusual at the first glance, but it works!

Critical areas (7)

- Program flow, when labels are used
- NetRexx has very rigorous checks (‚statement cannot be reached‘)
- had to implement ‚same‘ checks
- ... Implicit invokations of succeeding (labelled) program parts necessary, if code may be reached!

Automatic Type detection

- Program flow analysed in ‚logical order of execution‘
- assignment statements & comparisons used to detect ‚best type‘ for each item

- becomes effective when option binary is used!

Open issues

- SIGNAL ON ... And CALL ON not yet implemented (ignored with attention msg)
- ‚Symbol‘ and ‚Value‘ function not available
- ‚INTERPRET ...‘ not available
- ‚TRACE‘ only in NetRexx format available.
- ... Might be resolvable, but ‚Rexx2Nrx Class Builder‘ has higher priority!

Upcoming Release 3.01

- BOTTOM-Up translation of ‚external function packages‘ to Java classes
- Formerly „%INCLUDED“ functions may now be „IMPORTED“ and „USED“!
- Public properties of the external function may be accessed by parent (main) program.

Upcoming Release 3.02
The Rexx2Nrx Class Builder

- collections of associated stems (like item_name.ii, item_type.ii, item_class.ii, ...)
- ... Will be translated to public properties of new, generated classes, e.g.:
- class item extends object
- public properties
- name=Rexx „“; type=Rexx „“; ...

Advantages of the Rexx2Nrx Converter

- Equivalent source-code for both platforms available.
- Quick, Rexx-Prototype development with later translation to object-oriented approach
- re-Usage of already existing REXX-Code possible .

Advantages of the Rexx2Nrx Converter (2)

- Nice re-Formatting of the given program source.
- Notes are copied from first reference to declaration
- various OPTIONS to tailor the data-types and format of the generated program.

Moderate Pricing

- Type A: single PC licence € 99
- Type B: PC development licence € 499
- Type C: PC site devel. licence € 4999
- (max. 20 PC‘s)
- Type B & C do contain source of RUN-Time-package.

- Company licences available on request.

Support

- Full support available thru the Internet!
- Thomas.Schneider@Donauland.at
- Own home page: www.Rexx2Nrx.com

- Enjoy the new vision of running classic REXX programs in a Java environment!
- GOOD LUCK!