"A Syllabus for Introducing MBA Students to Procedural and Object-oriented Programming (Object Rexx)"

Rony G. Flatscher (Rony.Flatscher@wu-wien.ac.at)

Vienna University of Economics and Business Administration (http://www.wu-wien.ac.at)

MIS Department (http://www.wu-wien.ac.at/wi)

Foils: http://wwwi.wu-wien.ac.at/rgf/conf/amcis/1999/T09-003-foils.pdf

Abteilung für Wirtschaftsinformatik

Overview

- Vienna University of Economics and Business Administration and the MIS department
  - "Special field of Business Administration": MIS curriculum
- The advent of OO in MIS, problems for MBA students
  - OO-development trends in business applications
    - *ERP software*
    - *Individual software*
- Teaching MBA students the OO paradigm
  - Syllabus for teaching the mandatory concepts
- Discussion
  - Experiences with the concepts, the Object Rexx language
  - Lessons learned

WU-Wien (1) (Wirtschaftsuniversität Wien)

- WU-Wien - `http://www.wu-wien.ac.at`
  - "Wirtschaftsuniversität Wien"
  - English: "Vienna University of Economics and Business Administration"
- Over 20,000 MBA students
  - Master degree in one of the four fields of studies: Business Administration (51%), Commerce (36%), Economics (8%), Business Education (5%)
  - Miscellaneous
    - *Free form studies (no class system!)*
    - *Duration in effect 6-7 years in the average (4 years minimum)*
    - *Drop-out rate more than 60%*

WU-Wien (2) (Wirtschaftsuniversität Wien)

- 4 years studies in two parts, e.g. BA studies
  - *Part I studies encompasses*
    - *Business Administration (15-16hrs), Economics (13hrs), Private Law (8-12hrs), Mathematics and Statistics (8-12hrs), Foreign Language (8hrs), Foreign Language II (12hrs) and Sociology (8-12hrs)*
  - *Part II studies encompasses*
    - *Business Administration (13hrs), First Special Field of Business Administration (12hrs), Second Special Field of Business Administration (12hrs), Elective (8hrs), Economics (10hrs), Public Law (8hrs)*
  - *Master Thesis (6 months to a couple of years)*
- Management Information Systems (MIS)
  - *Special field of Business Administration in Part II studies*
  - *Elective in Part II studies*

WU-Wien (3)
Special BA "MIS" (3 Semesters)

Semester 5
(6-8hrs)

Electronic Commerce (lecture, 2 hours)	Electronic Finance (lecture, 2 hours)
Business Process Modelling (lecture, 2 hours)	
Choice of	
„ISD“: Information Systems Development with CASE (optional lecture, 2 hours)	„SPET“: Solving Problems with Enduser Tools (optional lecture, 2 hours)

Semester 6
(6hrs)

ISD – Application (mandatory proseminar, 2 hours)	SPET - Application (mandatory proseminar, 2 hours)

at least two (à 2 hours) lectures („Electives“) out of:

IS in Finance and Accounting; IS in Marketing; IS in Commerce; Computer Law; IT Market and Information Management; Electronic Money, Payment Systems and Security; *Introduction to Procedural and Object-oriented Programming (Object Rexx)*

Semester 7
(2hrs)

MIS Seminar (mandatory seminar, 2 hours)

Introducing OO-Concepts into Business Applications (1)

- OO-development trends relevant to business applications
  - Enterprise Resource Programs (ERP) start to employ terms (and introduce concepts) like
    - *"Business objects"*
    - *"Business components"*
    - *"Patterns"*
    - *"Frameworks"*
  - Object Management Group (OMG)
    - *Business Object Domain Task Force (BODT)*
      - *Produced OO-standard for Workflow Management Systems*
      - *Devised a "Business Object Component Architecture"*
        - *Split up (March 1999) to three new RFP's (August 1999)*
        - *Genuine OMG "Component Architecture Standard"*

Introducing OO-Concepts into Business Applications (2)

- Problems with MBA students
  - No *working* knowledge of the fundamentals of OO
    - *Classes, objects, messages, methods, ...*
    - *Inheritance (attributes and methods)*
  - As a result
    - *No ability to fully evaluate and assess OO-based business applications*
    - *No working knowledge for analyzing and devising OO-models*
      - *MBA students with a working knowledge in EERM (Extended Entity-Relationship-Modelling) think they have no knowledge whatsoever with respect to OOM (Object-oriented Modelling)*
        - *True for OMG's UML (Unified Modelling Language, a meta-model)*
        - *True for OMG's MOF (Meta Object Facility, a meta-meta-model)*

Mandatory OO-Concepts (1)

- "Class"
  - Specification (and in the context of an OO-language an implementation) of an abstract data type (ADT)
    - *Properties, e.g.*
      - *Attributes*
        - *Determining data structure*
      - *Functions/Procedures (Methods)*
        - *Methods are invoked by sending messages to the instances (objects)*
        - *Flow of messages can be seen as **behaviour***
  - Classification **tree**
    - *Generalization/Specialization*
      - *There is a **root class***
    - *Inheritance, Multiple Inheritance*
      - *Taking advantage of pre-defined **and** pre-tested properties of superclasses up to and including the root class*

Mandatory OO-Concepts (2)

- Instance (an "object") of a class
    - *Creation (Initializing)*
    - *Destroying (Uninitializing)*
- Sending of messages
    - Resolution of methods
    - Unknown messages
- "Instance" methods versus "Class" methods
    - Metaclasses
- Concurrency
    - Execution of methods in parallel, differentiated between:
        - *Inter-object*
        - *Intra-object*

Teaching MBA Students the OO-paradigm with Object Rexx

- Pre-requisites
  - Only 2 (two!) hours available due to the MIS curriculum
    - *Students, who mostlikely have no knowledge of OO-concepts*
    - *Students, who possibly have no prior experience with programming languages at all*
- Conclusions
  - Teaching OO-concepts should be supported with examples in order to ease understanding
    - *Programming language for experimenting with the examples*
    - ***Must have***
      - *Easy syntax (in order to save time), preferably pseudo-code like*
      - *Powerful OO-model (in order to experiment with all OO-concepts)*
    - *Examples for demonstrating OO-concepts need to be carefully chosen and worked out*

Syllabus
Procedural Concepts (1)

- **Class 1** (2hrs, i.e. 90 minutes)
  - Overview of the lecture, history of Rexx, new developments: ANSI Rexx, Object Rexx, NetRexx
- **Class 2** (2hrs, i.e. 90 minutes)
  - Minimal Rexx program, "Rexxtry.cmd" resp. "Rexxtry.rex", variables, constants, comments
  - Statement, block, conditional branch, iteration
- **Class 3** (2hrs, i.e. 90 minutes)
  - Labels, procedures/functions, resolution of function calls, Scopes
- **Class 4** (2hrs, i.e. 90 minutes)
  - Rexx builtin functions
  - Stems (associative arrays), RexxUtility functions

Syllabus
Procedural Concepts (2)

- **Class 5** (2hrs, i.e. 90 minutes)
  - Exceptions (SIGNAL, RAISE) and their handling
  - Object Rexx extensions
    - *Routines*
      - *Public or private depending on the keyword PUBLIC at the end of a ::ROUTINE-directive*
      - *All public routines can be called from other programs*
    - *References to arguments which allows stems to be passed by reference*
    - *USER-definable exceptions*
- **Class 6** (2hrs, i.e. 90 minutes)
  - Examples and possible solutions

Syllabus
Object-oriented Concepts (1)

- **Class 7** (2hrs, i.e. 90 minutes)
  - Abstract data type (ADT)
  - Implementing an ADT with Object Rexx
    - *Class, methods, attributes*
  - Messages (message operator "~")
    - *Cascading messages "~~"*
  - Scopes
  - Creating an instance (an object) of a class
    - *Initializing (INIT)*
  - Destroying an instance (an object) of a class
    - *DROP*
    - *Garbage collector*
    - *Uninitializing (UNINIT)*

Syllabus
Object-oriented Concepts (2)

- Class 8 (2hrs, i.e. 90 minutes)
  - Reiterating ADT, class, method, attribute, message, INIT and UNINIT
  - Specializing, inheritance
  - Multi-threading
  - Scopes
- Class 9+10+11 (2hrs, i.e. 90 minutes)
  - Method resolution
    - *Special variables supplied by the run-time and available within methods only*
      - *self and super*
    - *UNKNOWN Method*
    - *Effects of multiple inheritance on the method resolution*

Syllabus
Object-oriented Concepts (3)

- **Class 9+10+11**continued (2hrs, i.e. 90 minutes)
  - Object Rexx classification tree
    - *Introduction class by class*
      - *Fundamental classes: Object, Class, Method, Message*
      - *Alarm class Alarm and monitor class Monitor*
      - *Classic Rexx classes: String, Stem, Stream*
      - *Collection classes*
        - *System (external) supplied indices: Array, List, Queue*
        - *User (programmer) supplied indices: Directory, Relation, Bag, Table, Set*
        - *Iterating over all collected objects with DO...OVER or with the help of a Supplier object*

Syllabus
Object-oriented Concepts (4)

- Class 12 (2hrs, i.e. 90 minutes)
  - Class methods
  - Metaclasses
    - *Taking advantage of metaclass programming, e.g.*
      - *Singleton pattern*
      - *Manager pattern*
  - Defining classes and methods at run-time
  - "One-off objects" and creating them
  - "The Big Picture"
    - *Starting and instantiating the Object Rexx run-time environment*

Syllabus
Object-oriented Concepts (5)

- **Class 13** (2hrs, i.e. 90 minutes)
  - Coupling of Object Rexx programs with the available environments (interpreter supplied directory objects)
    - *.local*
    - *.environment*
- ~~**Class 14** (2hrs, i.e. 90 minutes)~~
  - ~~Introduction to Object Rexx utilities~~
    - ~~*ORX7 (from the 7th International Rexx symposium)*~~
      - ~~*Object Rexx program for analyzing (Object) Rexx programs and rendering them into ASCII or HTML*~~
      - ~~*Articles on explaining the "environment", "classes" and "metaclasses" and documenting the analysis tool*~~
  - ~~URLs~~
    - ~~`ftp://hobbes.nmsu.edu/pub/os2/dev/orexx/orx7.zip`~~
    - ~~`ftp://hobbes.nmsu.edu/pub/os2/dev/orexx/orx7doc.zip`~~

Syllabus

Object-oriented Concepts (6)

- ~~Class 14$_{\text{continued}}$ (2hrs, i.e. 90 minutes)~~
  - ~~*ORX8 (from the 8th International Rexx symposium)*~~
    - ~~*Utility classes e.g.*~~
      - ~~*Classes for managing anchors and references*~~
      - ~~*Classes for implementing a NLS version of the class Directory, etc.*~~
    - ~~*Utility routines e.g.*~~
      - ~~*Routines for sorting any collection in a versatile manner*~~
      - ~~*Routines for supporting national languages*~~
      - ~~*Routines for determining whether an object is of a given type or whether an object is a class object, etc.*~~
  - ~~URLs~~
    - ~~`ftp://hobbes.nmsu.edu/pub/os2/dev/orexx/orx8.zip`~~
    - ~~`ftp://hobbes.nmsu.edu/pub/os2/dev/orexx/orx8doc.zip`~~

Syllabus
Object-oriented Concepts (7)

- Class 15 (2hrs, i.e. 90 minutes)
  - Concurrency
    - *Inter-Object*
    - *Intra-Object*
    - *GUARD and REPLY*
    - *Object Rexx classes Message and Alarm*
- Class 16 (2hrs, i.e. 90 minutes)
  - Overview of the Object Rexx "Security Manager":
    - *Tasks, Implementation*
    - *Example of implementing a sandbox*
  - FORWARD statement
  - Direct D/SOM support
  - Direct OLE-/ActiveX-support

Roundup
Teaching MBA students ... (1)

- Experiences
  - Understood all taught OO-concepts
    - E.g., the concept of multiple inheritance has not posed any conceptual problems
    - Yet, problems with metaclasses
  - Students have *no* problems whatsoever understanding examples presented in the Object Rexx syntax
    - The examples can be read as if they were pseudo code
  - Preparing the classes was *extremely* time consuming
    - Defining the sequence of OO-concepts to be introduced
    - Devising examples highlighting the freshly introduced OO-concepts such that the OO-concepts become perfectly clear
    - "Inventing" excercises which need the taught concepts only

Roundup
Teaching MBA students ... (2)

- Conclusions
  - It *is* possible to teach the fundamental procedural and object-oriented concepts in a lecture of 2 hours
  - Object Rexx seems to be an ideal language for crafting example code and have the students experiment with it
    - Simple Syntax
    - Powerful OO-model
    - Masterable with respect to the built-in classes
  - Learned concepts directly applicable to real-world problems
    - CGI-Scripting
    - Scripting of applications, components
      - Taking advantage of OLE-/ActiveX-automation, D/SOM
    - Stand-alone applications

Related University URL's

- WU-Wien ("Vienna University of Economics and Business Administration")
  - *Home:* `http://www.wu-wien.ac.at`
  - *English:* `http://www.wu-wien.ac.at/englhome.html`
  - *Key data (English)* `http://www.wu-wien.ac.at/rektorat/KeyData.html`
- MIS Department at the WU-Wien
  - *Home:* `http://www.wu-wien.ac.at/wi`
- PDF-foils for the lecture *"Einführung in die Prozedurale und objekt-orientierte Programmierung (Object Rexx)"* (in German)

  `http://wwwi.wu-wien.ac.at/Studium/LVA-Unterlagen/poolv/1999s/`

Rexx-Related URL's

- "Rexx Language Association" homepage

  `http://www.RexxLA.org/`
- Object Rexx homepage

  `http://www2.hursley.ibm.com/orexx/`

  `http://www.software.ibm.com/ad/obj-rexx/`
- Rexx homepage

  `http://www2.hursley.ibm.com/rexx/`
- NetRexx homepage

  `http://www2.hursley.ibm.com/netrexx/`

Addendum - Object Rexx (1) Available OO-Features

- Object Rexx
  - Backward compatible with "classic" Rexx
  - Internally totally OO
    - *"Classic" Rexx statements transformed internally to their OO equivalents*
- ✓ Abstract data type (ADT)
    - *::CLASS- and ::METHOD-directives allow for fully implementing ADT's including attributes*
- ✓ Classification tree available
  - Object, Class, Method, Message
  - Alarm, Monitor
  - String, Stem, Stream

Addendum - Object Rexx (2) Available OO-Features

- ✓ Classification tree available continued
  - – Collection classes
    - *Array*, *List*, *Queue*, *Directory*, *Relation*, *Bag*, *Table*, *Set*
- ✓ Multiple Inheritance
- ✓ Instantiation/destruction (creating/destructing objects)
  - – Initializing (method INIT)
  - – Uninitializing (method UNINIT)
- ✓ Resolution of messages
  - – Handling of unknown messages (method UNKNOWN)
- ✓ "Instance" methods versus "Class" methods
  - – Metaclasses

Addendum - Object Rexx (3) Available OO-Features

- ✓ Concurrency
  - Execution of methods in parallel, differentiated between:
    - *Inter-object*
      - *By default available*
    - *Intra-object*
      - *Individual objects are sheltered by default from having more than one method activated from the same class (possible for programmers to change this behaviour)*
      - *By default available if the methods running in parallel for individual objects stem from different classes*
  - Pre- and Postconditions (GUARD)

Addendum - Object Rexx (4) Miscellaneous Aspects

- Syntax
  - Same intention as with classic Rexx
    - *Keeps the* ***syntax*** *and* ***built-in functionality*** *"****user friendly****", i.e.* ***as simple as possible*** *("pseudo-code like")*
- Versatility
  - Underpinned with a powerful OO-model
  - Classes, methods, messages can be generated/inspected at runtime
  - "One-off objects"
- Direct support of OO-infrastructure in OS
  - OS/2: direct SOM- and DSOM-support
    - *Object Rexx classes specializing D/SOM classes*
    - *Instantiating D/SOM classes from Object Rexx, sending D/SOM messages as if they were Object Rexx messages*

Addendum - Object Rexx (5) Miscellaneous Aspects

- Direct support of OO-infrastructure in OS$_{\textbf{continued}}$
  - Windows 95/98/NT/2000: direct OLE-/ActiveX-automation support in beta test
    - *Instantiating OLE-/ActiveX-classes from Object Rexx, sending OLE-/ActiveX-messages as if they were Object Rexx messages*

      `http://www.software.ibm.com/ad/obj-rexx/download.html`
      - *URL to download "OLE/ActiveX extension" for Object Rexx*
- Multiplatform availability
  - AIX (since 1999)
  - Linux (since 1998, freely available)
  - OS/2 (since 1997 part of Warp4, freely available for Warp3)
  - Windows 95/98/NT/2000 (since 1998)

Addendum - ::CLASS-Directive Example: ADT "Vehicle" (1)

```
/* single inheritance */
.land_vehicle~new("Truck")~Drive
.watercraft~new("Boat")~Swim
```

```
::CLASS   Vehicle
::METHOD Type             ATTRIBUTE
::METHOD INIT
  self~Type = ARG(1)
```

```
::CLASS   Land_Vehicle   SUBCLASS Vehicle
::METHOD Drive
  SAY self~Type": 'Now, I am driving ...'"
```

```
::CLASS   Watercraft     SUBCLASS Vehicle
::METHOD Swim
  SAY self~Type": 'Now, I am swimming ...'"
```

Output:

```
  Truck: 'Now, I am driving ...'
  Boat: 'Now, I am swimming ...'
```

Addendum - ::CLASS-Directive Example: ADT "Vehicle" (2)

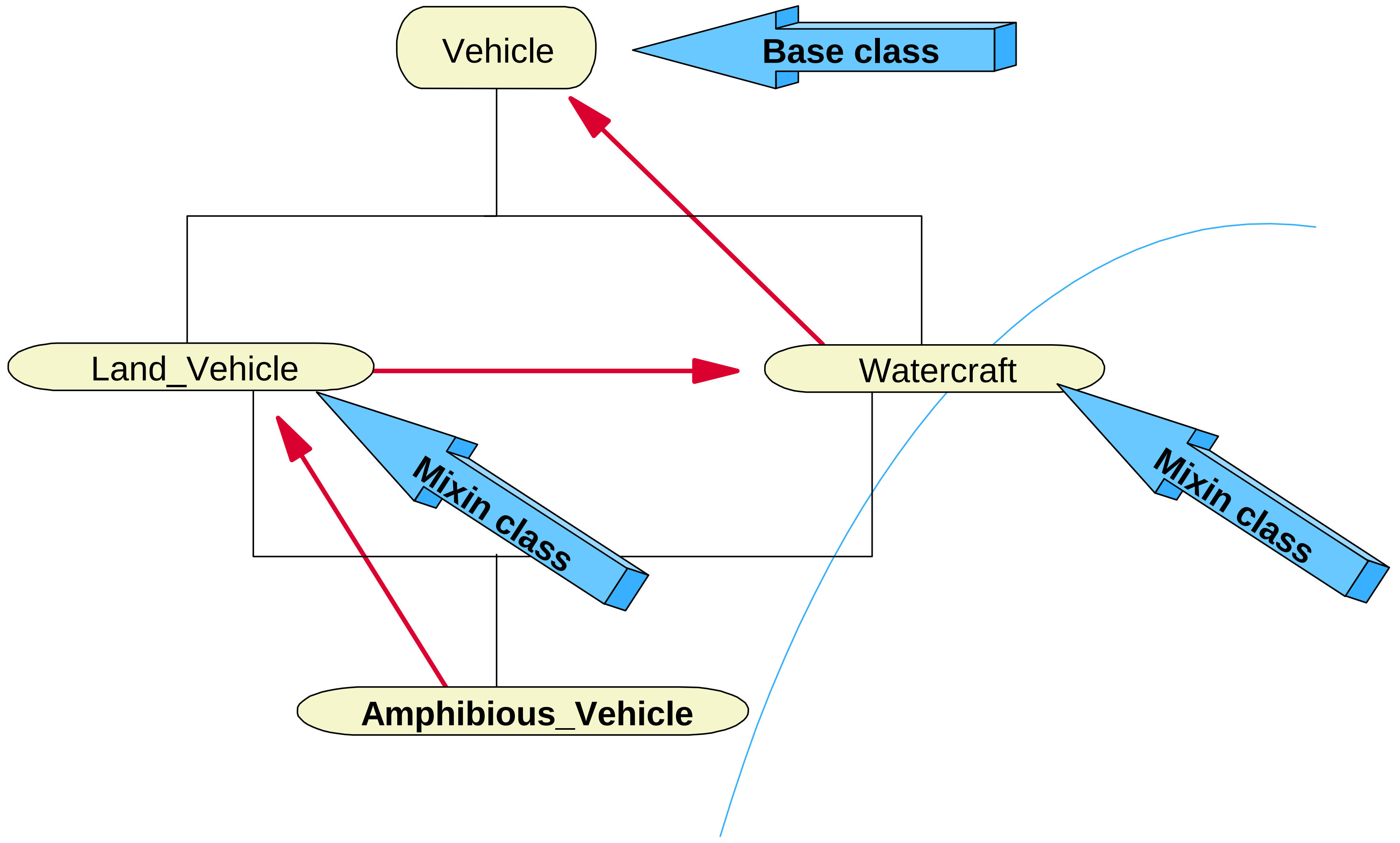
```
 /* Multiple Inheritance */
.land_vehicle~new("Truck")~Drive
.watercraft~new("Boat")~Swim
.Amphibious_Vehicle ~new("Floatable_Car") ~ Show_What_You_Can

::CLASS   Vehicle
::METHOD Type            ATTRIBUTE
::METHOD INIT
  self~Type = ARG(1)
::CLASS   Land_Vehicle   MIXINCLASS Vehicle
::METHOD Drive
  SAY self~Type": 'Now, I am driving ...'"
::CLASS   Watercraft     MIXINCLASS Vehicle
::METHOD Swim
  SAY self~Type": 'Now, I am swimming ...'"
::CLASS Amphibious_Vehicle SUBCLASS Land_Vehicle INHERIT Watercraft
::METHOD Show_What_You_Can
  self~~Drive~~Swim
```

Output:

```
  Truck: 'Now, I am driving ...'
  Boat: 'Now, I am swimming ...'
  Floatable_Car: 'Now, I am driving ...'
  Floatable_Car: 'Now, I am swimming ...'
```

Addendum - ::CLASS-Direktive
Example: ADT "Vehicle" (3)


Vehicle
Base class
Land_Vehicle
Watercraft
Mixin class
Mixin class
Amphibious_Vehicle