Automatisierung mit ooRexx und BSF4ooRexx

Rony G. Flatscher

Institut für BWL und Wirtschaftsinformatik
Wirtschaftsuniversität Wien (WU)
Augasse 2-6
A-1090 Wien
rony.flatscher@wu.ac.at

Abstract: Diese Arbeit führt in die quelloffene und freie Skriptsprache ooRexx und das gleichermaßen quelloffene und freie Funktionspaket BSF4ooRexx zur Anbindung an Java-Klassenbibliotheken ein. Ein Anwendungsschwerpunkt liegt hierbei in der Automatisierung von wiederkehrenden, geschäftlichen Abläufen in betrieblichen Fachabteilungen durch „Endbenutzerprogrammierer“ („Business Programmers“). Zur Illustration der Ausführungen werden kurze („Nutshell“-) Programmbeispiele dargestellt, die aufgrund der einfachen Syntax von ooRexx oft wie Pseudocode aussehen.

1 Einleitung

Absolventen von sozial- und wirtschaftswissenschaftlichen Studien arbeiten typischerweise in ihren Ausbildungsgebieten wie Marketing, Controlling, Finance. An vielen deutschsprachigen Universitäten, so auch an der Wirtschaftsuniversität Wien, haben Studierende die Möglichkeit, sich auch auf das Fach Wirtschaftsinformatik als spezielle Betriebswirtschaftslehre oder als Studienzweig zu spezialisieren. Im Rahmen der Spezialisierung „Wirtschaftsinformatik“ im Bachelorstudium können Studierende zwei Lehrveranstaltungen belegen, die in die Grundlagen der objektorientierten Programmierung einführen und diese Kenntnisse unmittelbar auf einfache, betriebswirtschaftliche Problemstellungen anwendbar machen. Dem Zeitgeist geschuldet, lauten die Bezeichnungen dieser konsekutiv zu besuchenden 2-stündigen (4 ECTS, European Credit Transfer System) Lehrveranstaltungen „Business Programming 1“ [W3a] und „Business Programming 2“ [W3b], womit auch zum Ausdruck kommen soll, dass ausschließlich die Erstellung von (kurzen) Programmen im Kontext von betrieblichen Geschäftsabläufen damit ermöglicht werden soll.

Die grundlegende Überlegung dabei ist, dass jene Studierende, die nach ihrem Studium in Fachabteilungen von Unternehmen arbeiten, ihre erworbenen grundlegenden wirtschaftsinformatorischen Kenntnisse dazu einsetzen, wiederholende Geschäftsabläufe durch selbst erstellte, einfache Programme zu automatisieren. Dadurch sollen sie im Vergleich zu ihren Kollegen über eine für alle erkennbar höhere Problemlösungskompetenz verfügen, die es ihnen erlaubt, die modernen,

informationsbezogenen Betriebsmittel effizienter als nicht in Wirtschaftsinformatik Ausgebildete einzusetzen.

Aufgrund von regelmäßig stattfindenden, jährlichen Treffen der Absolventen der Spezialisierung „Wirtschaftsinformatik“, erhalten wir immer wieder Rückkoppelungen, die darauf hinweisen, dass die Zielsetzungen dieser Ausbildung tatsächlich zu erhöhten Rationalisierungseffekten in den Abteilungen und zu Wettbewerbsvorteilen der wirtschaftsinformatorisch Ausgebildeten gegenüber ihren Mitkollegen führen.

Nachdem aufgrund von jahrelangen Experimenten des Autors mit verschiedenen Programmiersprachen[1], „Open Object Rexx (ooRexx)“ sich als die am schnellsten zu erlernende Sprache für die wirtschaftswissenschaftlichen Studierenden an der WU[2] erwiesen hat, wird zunächst diese quelloffene und freie Programmiersprache kurz vorgestellt und anschließend anhand einfacher, kurzer Beispiele („Nutshell“-Beispiele) demonstriert, wie derartige ooRexx-Programme aussehen, um z.B. Microsoft Excel anzusteuern.

Daran anschließend wird das quelloffene und freie ooRexx-Funktionspaket „BSF4ooRexx“ vorgestellt, das im übrigen in Deutschland und Österreich entwickelt wurde, das Java-Klassenbibliotheken in Form von ooRexx-Klassenbibliotheken kaschiert und zur Verfügung stellt. Damit wird es möglich, ooRexx-Programme zu erstellen, die betriebssystemunabhängig ohne Änderungen ablauffähig sind und damit unter anderem auch die Möglichkeit eröffnen, systematisch aus Betriebssystem-Lockins zu entkommen, indem neue Automatisierungsprogramme – aus strategischen Überlegungen – bewusst mit BSF4ooRexx erstellt werden. Auch hier soll anhand von „Nutshell“-Beispielen gezeigt werden, wie derartige ooRexx-Programme aussehen, um z.B. betriebssystemunabhängig das Tabellenkalkulationsprogramm „scalc“ von OpenOffice anzusteuern.

An dieser Stelle sollte vielleicht auch darauf hingewiesen werden, dass es Betriebe gibt, die in durchaus großflächigem Maßstab „Open Object Rexx (ooRexx)“ und BSF4ooRexx für die Lösung ihrer Probleme einsetzen, beispielsweise das Europäische Patentamt (EPA)[3]. Insoferne sollen sowohl die (interdisziplinäre) Forschungsgemeinschaft als auch Betriebe, die mit dem Einsatz von quelloffenen, freien

[1] Unter anderem hat der Autor in diesem Kontext die Programmiersprachen Visual Basic, VBScript und JavaScript/ECMAScript unterrichtet und analysiert. Unabhängig davon gingen in die über viele Jahre hinweg erfolgende Konzeption entsprechender Lehrveranstaltungen auch die Kenntnisse und Erfahrungen zahlreicher weiterer Programmiersprachen ein, wie beispielsweise Assembler, C, C++, C#, COBOL, Java, NetRexx, Pascal, Perl, PHP, PROLOG, Python, REXX, RPG, Smalltalk.

[2] In [F12] gibt der Autor einen konzeptionellen Überblick über die zwei aufeinander aufbauenden, selbst entwickelten Lehrveranstaltungen, „Business Programming 1“ (Grundlagen der objektorientierten Programmierung mit ooRexx und Anwendung unter Windows) und „Business Programming 2“ (BSF4ooRexx als Brücke zu Java, um „lock-in-freie“, also betriebssystemunabhängige, Programme zu erstellen lernen), die wirtschaftswissenschaftliche Studierende zu erfolgreichen „Endbenutzer-Programmierern“ ausbilden. Vgl. in diesem Zusammenhang auf diesen erworbenen Kenntnissen aufbauende Studentenarbeiten unter [W3i].

[3] Das Europäische Patentamt setzt einen Korpus von ca. 200.000 Rexx-Programmen ein, wobei die Top 10-Programme nach Auskunft der EPA pro Jahr etwa vier Millionen Mal aufgerufen werden. Diese vielleicht überraschenden Zahlen deuten unter anderem auch darauf hin, dass diese Infrastruktur stabil im industriellen Maßstab eingesetzt wird. Folien zu einem Übersichtsvortrag zum Einsatz von ooRexx und BSF4ooRexx bei der EPA finden sich in [F11].

Infrastrukturen liebäugeln, auf ooRexx und BSF4ooRexx aufmerksam gemacht werden, da sich damit interessante Forschungs- und Problemlösungsmöglichkeiten ergeben, die es wert sind, systematisch erforscht, hinterfragt und diskutiert zu werden. Eine derartige Diskussion kann aber erst dann beginnen, wenn grundlegende Informationen zu ooRexx und BSF4ooRexx vermittelt wurden.

2 Open Object Rexx (ooRexx)

Open Object Rexx (ooRexx) ist die quelloffene, freie Version von "Object REXX", das ursprünglich von IBM als proprietärer, objektorientierter Nachfolger der Skriptsprache REXX erstellt wurde. Die Skriptsprache REXX [C90] löste vor über 30 Jahren auf IBM Mainframes die kryptisch gewordene Sprache Exec-2 ab und wurde im Vergleich dazu bewusst als „menschenzentriert" konzipiert, woraus unter anderem die einfache und daher leicht erlernbare Syntax folgt. REXX wurde durch IBM's System Application Architecture (SAA)-Strategie auf allen IBM-Plattformen als Skript- und Batchsprache verfügbar gemacht. REXX erfreute sich in den 80-er Jahren auch außerhalb der Firma IBM großer Anziehungskraft, was unter anderem dazu geführt hatte, dass mehrere Firmen eigene kommerzielle REXX-Interpreter erstellten, dass REXX sogar in Nicht-IBM-Betriebssystemen wie Amiga OS Eingang gefunden und letztlich dazu geführt hat, dass diese Sprache von INCITS (früher: ANSI) 1996 standardisiert und 2006 für weitere zehn Jahre als aktiver Standard verlängert wurde [I06].

Object REXX [V96] wurde 1996 von IBM als Produkt mit seinem PC-Betriebssystem OS/2 Warp 4 ausgeliefert und separat als Produkt für AIX und Windows verkauft. Nachdem die Migration von IBM Großkunden auf Windows auch dazu geführt hatte, dass Object REXX für Windows eine Zeitlang gut verkauft wurde, folgte auch die Notwendigkeit, dieses Produkt für alle Betriebssysteme von IBM zu warten. Versuche, es erfolgreich auf Dauer zu verkaufen, schlugen fehl, sodass 2004 Gespräche zwischen der gemeinnützigen Rexx Language Association (RexxLA, [W3c]) und IBM aufgenommen wurden, um den Quellcode von Object REXX über die RexxLA zu veröffentlichen und weiterzuentwickeln.[4] Die erste quelloffene, freie Version wurde unter der Bezeichnung „Open Object Rexx (ooRexx)" im Frühjahr 2005 von der RexxLA als Version 3.0 veröffentlicht und wird seitdem aktiv weiterentwickelt. Mittlerweile steht ooRexx auf Dauer für die wichtigsten Betriebssysteme vorkompiliert in 32- und 64-Bit zur Verfügung: Linux, MacOSX und Windows. [F05] beschreibt eine Reihe von Rexx-Interpretern und erstmals wird darin auch die quelloffene Version ooRexx beschrieben.

2.1 Kurzcharakterisierung von ooRexx

ooRexx ist eine dynamisch typisierte, interpretierte Sprache, deren Syntax der REXX-Philosophie entsprechend sehr einfach in englischer Sprache gehalten ist, sodass ooRexx-Programme häufig wie Pseudocode aussehen (vgl. Abbildung 1, das Programm

[4] Der Autor war einer der vier Verhandler auf Seiten der RexxLA.

erzeugt als Ausgabe „`Durchlauf # 1`“, „`Durchlauf # 2`“, „`Durchlauf # 3`“) und damit gut lesbar sind. Unter anderem wird die Groß- und Kleinschreibung nicht unterschieden, es gibt keine reservierten Schlüsselwörter, Variablen werden nicht ausdrücklich vereinbart, sondern bei Bedarf eingeführt.

```
do i=1 to 3
   say "Durchlauf #" i
end
```

Abbildung 1: Schleife mit einer Laufvariable

Das objektorientierte Modell ist zu einem großen Teil von Smalltalk-Konzepten beeinflusst, allerdings in der Syntax vereinfacht, beispielsweise dadurch, dass ein expliziter, sichtbarer Nachrichtenoperator (~, Tilde) für das Versenden von Nachrichten definiert ist. Hierbei ist links vom Nachrichtenoperator das Empfängerobjekt angegeben, rechts davon der Namen der Nachricht, optional gefolgt von Argumenten in runden Klammern, die der Nachricht mitgegeben werden sollen. Abbildung 2 zeigt ein Programm, das die Ausgabe „`9. GI-Workshop ISOS 2012`“ erzeugt.

```
say "2102 SOSI pohskroW-IG .9"~reverse
```

Abbildung 2: Ausgabeanweisung mit Nachricht `reverse` an eine Zeichenkette

2.2 Automatisierung mit ooRexx

„Automatisierung“ in diesem Kontext beschreibt Abläufe in Fachabteilungen, die sich wiederholen und mit Hilfe von Programmen automatisiert werden können. Beispielsweise kann es sein, dass Mitarbeiter immer wieder händisch temporäre Sicherungen von Excel-Analysedaten vornehmen müssen. Abbildung 3 zeigt ein ooRexx-Programm, das dies automatisiert, indem der entsprechende Befehl erzeugt und mit Hilfe von ooRexx als Kommando[5] abgesetzt wird. Darüber hinaus wird unter Programmkontrolle mit einer entsprechenden Ausgabeanweisung festgestellt, ob der Befehl erfolgreich war oder nicht.

```
quelle   = "C:\report\*"
ziel     = "D:\bkp\report"
schalter = "/E /A /V"
befehl   = "xcopy" quelle ziel schalter
say "Befehl:" befehl
befehl
if rc<>0 then say "rc="rc "Befehl NICHT erfolgreich!"
         else say "Befehl erfolgreich!"
```

Abbildung 3: Kommando an Kommandozeileninterpreter

[5] Ein Kommando ist eine Zeichenkette (Literal oder Variablenwert), die der Shell zur Ausführung übergeben wird, wobei der Rückgabecode anschließend in der Rexx-Variable „rc“ abgerufen werden kann. Ein Rückgabewert von „0“ zeigt üblicherweise an, dass das Kommando erfolgreich – also ohne Fehler – ausgeführt wurde.

Die Windows-Version von ooRexx erlaubt es über die Proxy-Klasse „`.OleObject`“ sämtliche Windows-COM/OLE-Programme anzusteuern und damit zu interagieren. Daher ist es mit ooRexx unter Windows möglich, sowohl die Windows-Shell als auch die Informationssysteme anzusteuern („zu automatisieren“), die in Abteilungen für die Abarbeitung von Geschäftsprozessen eingesetzt werden. Darüber hinaus stehen ooRexx-Klassen zur Verfügung, mit deren Hilfe man auch Tastatur- und Mauseingaben unter Programmkontrolle einfach an beliebige Fenster schicken kann, sodass jede Form von Informationssystem mit Benutzerschnittstelle automatisiert werden kann.

Abbildung 4[6] zeigt beispielhaft, wie man mit der Windows-Shell interagieren kann. Hier wird auf der Windows-Benutzeroberfläche des Mitarbeiters direkt ein Verweis auf das Windows-Notepad-Programm angelegt und damit zur Verfügung gestellt.

```
shell    = .OLEObject~new("WScript.Shell")
desktop  = shell~specialFolders("Desktop")
shortcut = shell~createShortcut(Desktop || "\Link zu Notepad.lnk")
shortCut~targetPath = "%WINDIR%\notepad.exe"
shortCut~save
```

Abbildung 4: Verweis (Link) auf `notepad.exe` auf Benutzeroberfläche anlegen

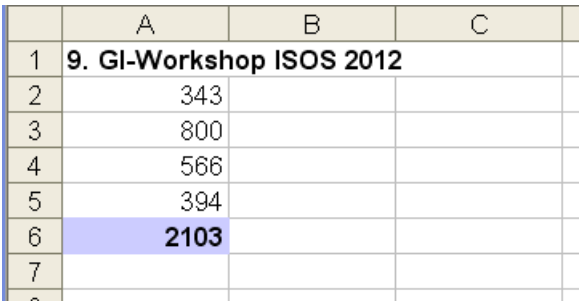
Nachdem in Abteilungen häufig Endbenutzerwerkzeuge wie Textverarbeitungs-, Tabellenkalkulations- oder Präsentationsprogramme eingesetzt werden, können häufig nach demselben Muster wiederkehrende, manuelle Arbeiten automatisiert werden, sofern die Ansteuerung dieser Werkzeuge von Programmen aus möglich ist. Abbildung 5 zeigt ein einfaches ooRexx-Programm, das eine Instanz von Microsoft Excel erzeugt und darin Einträge vornimmt.

```
excelObjekt         = .OLEObject~new("Excel.Application")
excelObjekt~visible = .true
tabellenBlatt       = excelObjekt~workbooks~add~worksheets[1]

titel           = tabellenBlatt~range("A1")
titel~font~bold = .true
titel~value     = "9. GI-Workshop ISOS 2012"

do i=2 to 5
   tabellenBlatt~range("A"i)~value = random()
end

summe                     = tabellenBlatt~range("A6")
summe~font~bold           = .true
summe~interior~colorIndex = 24
summe~formula             = "=summe(A2:A5)"

tabellenBlatt~saveAs(directory()"\demo.xls")
```

Abbildung 5: Ansteuerung von MS-Excel

[6] Die Windows-Version von ooRexx installiert im Verzeichnis „ooRexx\samples\ole“ eine Reihe von ooRexx-Nutshell-Programmen, die die Ansteuerung von Windows und Windows-Anwendungen demonstrieren. Dieses und das folgende Beispiel basieren darauf.

Abbildung 6 zeigt das resultierende Tabellenkalkulationsblatt. Es ist relativ einfach für entsprechend wirtschaftsinformatorisch Ausgebildete, dieses Nutshell-Beispiel für fast alle Anwendungsbedürfnisse einer Fachabteilung zu adaptieren.



	A	B	C
1	**9. GI-Workshop ISOS 2012**		
2	343		
3	800		
4	566		
5	394		
6	**2103**		
7			

Abbildung 6: Ergebnis der Ansteuerung von MS-Excel aus Abbildung 5 oben

3 Bean Scripting Framework for ooRexx (BSF4ooRexx)

Als Ende der 90er Jahre klar wurde, dass IBM OS/2 nicht mehr weitergeführt werden wird, war es dem Autor ein Bedürfnis, den unter OS/2 weit verbreiteten REXX- und Object REXX-Programmierern eine Brücke in die Welt von Linux und Windows zu bauen. Die ursprüngliche Absicht lag darin, mit Hilfe eines externen REXX-Funktionspakets den Zugang zu Java zu ermöglichen, mit dem Hintergedanken, dass derartige Programme unverändert auf all diesen Plattformen ablauffähig wären und somit die Investitionskosten nicht verloren sind. Java war bereits damals auf allen wichtigeren Betriebssystemen, insbesondere OS/2 und Windows, in Form der Java Runtime Envrionment (JRE) auf den meisten Personal Computern installiert.

Ausgehend von einer erfolgreichen proof-of-concept-Arbeit an der Universität Essen im Jahre 2000 wurde über die folgenden Jahre ein externes REXX-Funktionspaket erarbeitet, das in der Tat diesen Brückenschlag ermöglichte. Hierzu wurde die quelloffene und freie Java-Klassenbibliothek Bean Scripting Framework (BSF) der Apache Software Foundation (ASF) eingesetzt, die ursprünglich als quelloffene und freie Klassenbibliothek von Mitarbeitern der Firma IBM konzipiert wurde, um in Java Server Pages (JSP) auch Programme in populären Skriptsprachen einbetten zu können.

Die Anbindung von REXX, das über C-Programmierschnittstellen verfügt, an BSF wurde mit Hilfe der Java Native Interface (JNI) Programmierschnittstellen realisiert und das resultierende Funktionspaket als „Essener-Version“ von BSF4Rexx bezeichnet.[7]

[7] Nachdem der Autor bei den Implementierungsarbeiten auf Fehler in ASF BSF gestoßen war, hatte er in weiterer Folge entsprechende Fehlerbehebungs-Patches, aber auch Anregungen in das ASF BSF-Projekt eingebracht. In weiterer Folge erhielt der Autor zunächst Committer-Status und wurde später als Mitglied der ASF eingeladen. Die Erfahrungen und die gewonnene Expertise im Zusammenhang mit BSF führte später dazu, dass der Autor als Experte im Java Community Process (JCP) für den Java Specification Request (JSR) 223 fungierte, der das Rahmenwerk für die Java-Programmierschnittstellen für Skriptsprachen definierte. Die Arbeitsergebnisse wurden in Java 1.6/6 im Paket „javax.script“ in Java eingeführt. In weiterer Folge wurde im

Nachdem Java eine strikt typisierte Programmiersprache ist, wurden die Programmierschnittstellen zu Java für REXX-Programme gleichermaßen strikt typisiert, was der grundlegenden REXX-Philosophie (dynamisch typisiert, Programmierer sollen nicht mit strikter Typisierung „belastet“ werden) entgegenstand.

Während des Aufenthalts an der Universität Augsburg wurde eine dynamisch typisierte Schnittstelle realisiert, sodass die Interaktion mit Java nicht mehr strikt typisiert erfolgen muss, was die Nutzung von Java für REXX-Programmierer erheblich erleichtert hat. Die resultierende Version wurde in der Rexx-Community als „Augsburger-Version“ von BSF4Rexx bekannt.

Mit der Version 4.0 von ooRexx wurde ein neuer Kern mit objektorientierten C++-Schnittstellen zu ooRexx verfügbar gemacht, wobei sich die Architektur an die von JNI orientierte. Damit wurde es u.a. möglich, Java-Callbacks in ooRexx zu implementieren, beispielsweise, indem abstrakte bzw. Interface-Methoden in ooRexx ausprogrammierbar wurden. Diese Arbeiten erfolgten nach dem Wechsel an die Wirtschaftsuniversität Wien, die resultierende „Wiener-Version“ des externen Funktionspakets wurde zudem in „BSF4ooRexx“ umbenannt, um im Namen die Tatsache zum Ausdruck zu bringen, dass diese Version nur mehr zusammen mit ooRexx einsetzbar ist.

3.1 Kurzcharakterisierung von BSF4ooRexx

BSF4ooRexx [W3f] ist ein externes Rexx-Funktionspaket, was bedeutet, dass seine Funktionalität von Rexx-Programmen aus angesprochen werden kann, obwohl die externen Funktionen in C++ und nicht in Rexx geschrieben sind. Um für ooRexx-Programmierer, die Java überhaupt nicht beherrschen müssen, das Interagieren mit Java möglichst drastisch zu vereinfachen, wurde mit Hilfe des ooRexx-Pakets, „`BSF.CLS`“[8], die gesamte Java-Klassenbibiliothek und die Interaktion mit Java-Objekten so maskiert, dass beinahe der Eindruck entsteht, dass es sich dabei in Wirklichkeit um ooRexx-Klassenbibliotheken und ooRexx-Objekte handelt, denen man lediglich ooRexx-Nachrichten schicken müsse.

Für ooRexx-Programmierer ergeben sich mehrere Vorteile aus dieser Anbindung an Java, unter anderem:

- Es bedarf keiner weiteren externen Funktionspakete für Rexx, wenn in der Java-Laufzeitumgebung (Java Runtime Environment, JRE) die benötigte Funktionalität bereits (plattformunabhängig!) verfügbar ist. Beispiel: ssl-

Rahmen des ASF BSF Projektes eine Version 3.0 kreiert, die von sri-lankesischen Studierenden unter der Projektleitung von Sanjeeva Weerewarana, einer der ursprünglichen BSF-Autoren bei IBM und Gründer einer sri-lankesischen Opensource-Firma, eine „clean-room“-Implementierung vornahmen, sodass das JSR-223 „javax.script“ auch in früheren Versionen von Java damit verfügbar gemacht worden ist.

[8] In diesem Paket wird unter anderem die ooRexx-Proxy-Klasse „`.BSF`“ und eine Reihe öffentlicher Routinen definiert. Nachdem in Rexx – und damit auch in ooRexx – Bezeichner auch Punkte als normale Zeichen beinhalten dürfen, wurde bei manchen Routinen- und Methodenbezeichnern bewusst ein Punkt eingefügt. Einerseits erleichtern die Punkte das Lesen von Bestandteilen in den Bezeichnern, andererseits wird damit in Methodennamen sichergestellt, dass diese nicht mit Java-Methodenbezeichner verwechselt werden können, die ihrer Spezifikation wegen über keine Punkte in ihren Namen aufweisen dürfen.

Programmierung ist über BSF4ooRexx möglich, sodass kein eigenständiges, externes ssl-Rexx-Funktionspaket benötigt wird.

- Es können alle JRE-Java-Klassen verwendet werden, beispielsweise um graphische Benutzerschnittstellen-Programme (GUI) in ooRexx zu erstellen, die auf allen von Java unterstützten Betriebssystemen unverändert ablauffähig sind. Beispiel: mit BSF4ooRexx kommt mit „ooRexxTry.rxj“ ein ooRexx-GUI-Programm, das das experimentelle Ausführen von Rexx-Anweisungen ermöglicht. Es läuft unverändert und in vergleichbarer Darstellung unter Linux, MacOSX und Windows.

- Es können beliebige Java-Klassenbibliotheken direkt verwendet werden. Beispiel: JFreeChart [W3g] für die Erstellung von Business-Grafiken.

- Es können auch Java-Klassen verwendet werden, die über abstrakte Methoden verfügen oder Argumente von bestimmten Interface-Klassen erwarten, wobei die Implementierung der abstrakten Methoden in ooRexx erfolgt. Somit wird damit ein Callback-Mechanismus von Java nach ooRexx realisiert.

- Es können alle Informationssysteme über ooRexx angesteuert werden, die über Java-Programmierschnittstellen verfügen, unabhängig vom installierten Betriebssystem. Beispiele: OpenOffice.org/Apache OpenOffice [W3h], LibreOffice, Lotus Notes.

- Es können Java-Anwendungen in einer einfachen Form Rexx-Skripte unter eigener Kontrolle ausführen, wobei es den Java-Anwendungen möglich ist, beliebig viele ooRexx-Interpreterinstanzen, im selben oder in verschiedenen Java-Threads, dafür anzulegen. In jeder derartigen ooRexx-Interpreterinstanz, die individuell konfigurierbar ist, können ooRexx-Programme nebenläufig, also im Multithreading-Modus ausgeführt werden.

Abbildung 7 stellt ein BSF4ooRexx-Programm dar, das die aktuelle Version von Java abfrägt und darstellt (z.B. „`1.6.0_31`“). In diesem Programm wird der Direktivenmechanismus von ooRexx eingesetzt: eine Direktive wird mit zwei Doppelpunkten eingeleitet und wird vom Interpreter ausgeführt, ehe das Programm selbst mit der Anweisung in der ersten Zeile abgearbeitet wird. Die „`::requires BSF.CLS`“-Direktive führt dazu, dass der ooRexx-Interpreter als erstes das Rexx-Paket „`BSF.CLS`“ wie ein Unterprogramm aufruft, sodass anschließend alle darin als öffentlich definierte ooRexx-Klassen (z.B. die Proxy-Klasse `.BSF`) und Routinen (z.B. die Routine `box()`) verfügbar sind. Anschließend beginnt das Interpretieren des Programmes mit der Anweisung in der ersten Zeile, in der daher diese öffentlichen Klassen und Routinen nunmehr sichtbar sind.

```
clz=.bsf~bsf.import("java.lang.System")
say clz~getProperty("java.version")

::requires BSF.CLS
```

Abbildung 7: Ausgabe der auf dem Rechner installierten Java-Version

3.2 Automatisierung mit BSF4ooRexx

In diesem Abschnitt wird in Form von einem Nutshell-Beispiel demonstriert, wie BSF4ooRexx benutzt werden kann, um das Endbenutzerwerkzeug OpenOffice `scalc` (Tabellenkalkulation) mit Hilfe von ooRexx anzusteuern. Die Ansteuerung über BSF4ooRexx ist deshalb möglich, da OpenOffice über Java-Programmierschnittstellen zu seinem UNO (Universal Network Object) Rahmenwerk verfügt. Die abgebildeten Beispielsprogramme laufen unverändert unter Linux, MacOSX und Windows.

Die gezeigten Beispiele benutzen das ooRexx-Paket „`UNO.CLS`“, das selbst über die `::requires`-Direktive das ooRexx-Paket „`BSF.CLS`“ zur Anbindung an Java einbindet. „`UNO.CLS`“ vereinfacht die Ansteuerung von OpenOffice im Vergleich zu C++ oder Java spürbar, indem es öffentliche Proxy-Klassen und Routinen dafür definiert. Eine besonders vereinfachende Unterstützung ergibt sich daraus, dass man benötigte Interface-Objekte von UNO-Objekten einfach erhält, indem man den unqualifizierten Interface-Namen[9] als ooRexx-Nachricht dem UNO-Objekt sendet.

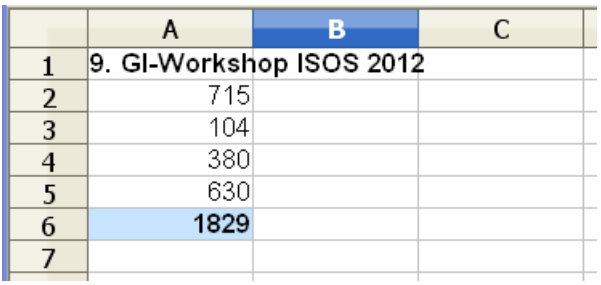
Abbildung 8 zeigt das ooRexx-Programm, das dem aus Abbildung 5 oben entspricht. Im Gegensatz dazu läuft es aber unverändert unter Linux, MacOSX und Windows.

```
Loader = uno.createDesktop()~XComponentLoader
url    = "private:factory/scalc"
calc   = loader~loadComponentFromURL(url,"_blank",0,.uno~noProps)
sheets = calc~XSpreadSheetDocument~getSheets
xSheet = sheets~XIndexAccess~getByIndex(0)~XSpreadSheet

call uno.setCell xSheet,"A1","9. GI-Workshop ISOS 2012"
bold  = bsf.getConstant("com.sun.star.awt.FontWeight","BOLD")
title = uno.getCell(xSheet,"A1")
title~XPropertySet~setPropertyValue("CharWeight",box("float",bold))

do i=2 to 5
   call uno.setCell xSheet,"A"i,random()
end

call uno.setCell xSheet,"A6","=summe(A2:A5)"
lBlue = box("int", "c6e4ff"x ~c2d)
summe = uno.getCell(xSheet,"A6")
summe~XPropertySet~setPropertyValue("CellBackColor",lBlue)
summe~XPropertySet~setPropertyValue("CharWeight",bold)
url   = uno.convertToUrl(directory()"/demo.ods")
calc~XStorable~storeAsURL(url,.uno~noProps)

::requires UNO.CLS
```

Abbildung 8: Ansteuerung von OpenOffice calc (Tabellenkalkulation)

Abbildung 9 zeigt das resultierende Tabellenkalkulationsblatt. Wie im MS-Excel-Beispiel (vgl. Abbildung 5 oben) ist es relativ einfach für entsprechend wirtschaftsinformatoisch Ausgebildete, dieses Nutshell-Beispiel für fast alle Anwendungsbedürfnisse einer Fachabteilung zu adaptieren.

[9] In OpenOffice beginnen aus einer Konvention heraus die unqualifizierten Interface-Namen mit einem „X“.



	A	B	C
1	9. GI-Workshop ISOS 2012		
2	715		
3	104		
4	380		
5	630		
6	1829		
7			

Abbildung 9: Ergebnis der Ansteuerung von OpenOffice calc aus Abbildung 8 oben

4 Zusammenfassung

Dieser Beitrag stellt die quelloffene und freie Programmiersprache ooRexx sowie das externe Funktionspaket BSF4ooRexx vor. ooRexx ist aufgrund der Syntax und den in englischer Sprache verfassten Schlüsselwortanweisungen eine einfach zu erlernende und einfach zu verstehende Programmiersprache. Sie geht zurück auf die IBM-Produkte REXX auf allen IBM-Systemplattformen und IBM's Produkt Object REXX, das 2004 im Quellcode der Rexx Language Association zur Veröffentlichung und weiteren Entwicklung übergeben wurde. Seit Mai 2005 ist Open Object Rexx (ooRexx) frei und quelloffen auf Dauer verfügbar und ist mittlerweile in der Version 4.1.1 in 32- und 64-Bit-Versionen vorkompiliert für Linux, MacOSX und Windows verfügbar.

ooRexx wird an der Wirtschaftsuniversität Wien für das Unterrichten von wirtschaftswissenschaftlichen Studierenden im objektorientierten Programmieren seit mehr als zehn Jahren erfolgreich eingesetzt.

BSF4ooRexx ist eine weitere, auf ooRexx abgestimmte, quelloffene und freie Funktionsbibliothek, die in Deutschland und Österreich seit dem Ende der 90er Jahre entwickelt wird. Ihr Zweck ist es, die „menschenzentrierten" Konzepte von ooRexx auf Java-Klassenbibliotheken in einer Form auszuweiten, die das Erlernen von Java überflüssig macht.

BSF4ooRexx wird an der Wirtschaftsuniversität Wien für das Unterrichten von wirtschaftswissenschaftlichen Studierenden im Interagieren mit Java-Klassen und Java-Objekten gleichermaßen seit Jahren erfolgreich eingesetzt [F12]. Hierbei wird der Schwerpunkt auf die Entwicklung portabler Anwendungen gelegt, sodass die resultierenden Programme unverändert auf Linux, MacOSX und Windows ablauffähig sind.

Unter anderem sollen die Studierenden damit die Kenntnisse und Erfahrungen gewinnen, die es ihnen ermöglichen, den Betrieben, in denen sie arbeiten werden, aus Betriebssystem-Lockins entkommen zu helfen. Betriebswirtschaftlich ist es grundsätzlich nachteilig, wenn bei gegebenen Alternativen lediglich Lockin-Lösungen

betrachtet werden, da damit die Entscheidungsspielräume von Betrieben – oft massiv – beschränkt werden.

Unter [W3i] finden sich zahlreiche, ausgewählte Studentenarbeiten, die einerseits veranschaulichen, was wirtschaftswissenschaftliche Studierende mit diesen Kenntnissen an Problemlösungskapazitäten gewonnen und erfolgreich angewendet haben. Andererseits finden sich dort zahlreiche ooRexx- und BSF4ooRexx-bezogene Beispiele, die zum großen Teil unmittelbar in Betrieben produktiv eingesetzt werden können, insbesondere alle OpenOffice-bezogenen Arbeiten. Diese Beispiele wurden von wirtschaftswissenschaftlichen Studierenden in selbständiger Arbeit erstellt und sollen daher auch die Hinweise exemplarisch untermauern, dass es für einschlägig wirtschaftsinformatorisch Gebildete es möglich wird, betriebswirtschaftliche Probleme mit Hilfe von ooRexx- bzw. BSF4ooRexx-Programmen selbständig zu lösen – oft lediglich durch Abänderung beziehungsweise durch Weiterentwicklung bestehender Problemlösungen in Form von Nutshell-Beispielen oder Lösungen von anderen Studierenden.

Sowohl ooRexx als auch BSF4ooRexx werden unter der Ägide der Rexx Language Association gepflegt und weiterentwickelt.

Mit der Konzeption der Programmiersprache REXX hat [C90] den Anspruch erhoben, eine Sprache für Menschen („human centric“) zu konzipieren, die leicht verständlich und erlernbar ist. ooRexx versucht diesem Prinzip zu folgen und aufgrund der Erfahrungen, die der Autor damit über die Jahre in der Lehre gewonnen hat, scheint dies zu einem großen Teil gelungen.[10] Es wäre grundsätzlich wichtig, würden Programmiersprachen unter dem Aspekt systematisch analysiert, charakterisiert und verglichen werden, inwieweit sie möglichst einfach (insbesondere von Nichtinformatikern) erlernt und auf Problemstellungen im entsprechenden Arbeitsumfeld erfolgreich angewandt werden können. Hier gäbe es ein großes Experimentierfeld mit vielfältigen Herausforderungen und wahrscheinlich auch interessanten Einsichten, die man unter anderem auch für die Gestaltung und Überarbeitung von Programmiersprachen einsetzen könnte.

Literaturverzeichnis

[C90] Cowlishaw, M.: The REXX Language – A Practical Approach to Programming”, 2. Auflage, Prentice-Hall, Englewood Cliffs, 1990.

[F05] Fosdick, H.: Rexx Programmer’s Reference, Wiley Publishing, Indianapolis, 2005.

[F11] Flatscher, R.G.: ooRexx at the European Patent Office, Vortrag auf der Tagung “The 2011 International Rexx Symposium”, 4.-7. Dezember 2011, Aruba. Folien (abgerufen: 2012-06-15): http://wi.wu-wien.ac.at/rgf/rexx/orx22/201112-EPO_ooRexx.pdf.

[F12] Flatscher, R.G.: Breeding “Business Programmers”, in: Proceedings von der Konferenz “7ª Conferencia Ibérica de Sistemas y Tecnologias de Información (CISTI’2012)”, 20.-23. Juni 2012, Madrid, Europa.

[10] BSF4ooRexx versucht gleichermaßen dem „human centric“-Prinzip von Rexx [C90] zu folgen: während der jahrelangen Entwicklungsarbeiten wurden grundsätzlich jene Programmierschnittstellen zu Java vereinfacht, die den Studierenden Verständnis- und Handhabungsprobleme bereitet hatten.

[I06] INCITS (InterNational Committee for Information Technology Standards; früher ANSI: American National Standard for Information Systems) 274: Programming Language Rexx, 1996/2006.

[V96] Veneskey, G.L.; Trosky, W.; Urbaniak, J.J.: Object Rexx by Example, Aviar Inc., Pittsburgh, 1996.

[W3a] Syllabus “Business Programming 1” (abgerufen: 2012-05-01): http://wi.wu.ac.at/rgf/wu/lehre/autowin/2012sBP1/BP1-autowin-2012s-uebersicht.pdf

[W3b] Syllabus “Business Programming 2” (abgerufen: 2012-05-01): http://wi.wu.ac.at/rgf/wu/lehre/autojava/2012sBP2/BP2-autojava-2012s-uebersicht.pdf

[W3c] Homepage der gemeinnützigen „Rexx Language Association (RexxLA)“ (abgerufen: 2012-05-01): http://www.RexxLA.org

[W3d] Homepage von ooRexx (abgerufen: 2012-05-01): http://www.ooRexx.org

[W3e] Homepage von „Bean Scripting Framework (BSF)“ (abgerufen: 2012-05-01): http://commons.apache.org/bsf/index.html

[W3f] Homepage von „Bean Scripting Framework for ooRexx (BSF4ooRexx)” (abgerufen: 2012-05-01): http://sourceforge.net/projects/bsf4oorexx/

[W3g] Homepage von „“JFreeChart“ (abgerufen: 2012-05-01): http://www.jfree.org/jfreechart/

[W3h] Homepage von “Apache OpenOffice” (abgerufen: 2012-05-01): http://www.openoffice.org/

[W3i] Ausgewählt, vom Autor betreute, Studentenarbeiten mit ooRexx- und BSF4ooRexx-Bezug (abgerufen: 2012-05-01): http://wi.wu.ac.at/rgf/diplomarbeiten/