R (D)COM Server – RExcel – rcom

Embedding R in Applications on Windows

Thomas Baier
Vienna University of Technology
mailto:thomas.baier@ci.tuwien.ac.at

Erich Neuwirth
University of Vienna
mailto:neuwirth@univie.ac.at

Agenda

- Basics
- Excel and VBA
- Visual Basic 6 and Visual Basic .NET
- Configuration and Installation
- Enhanced Topics
- Debugging and Error Handling
- Resources

Agenda: Basics

- Basics
  - COM
  - `IStatConnector`, using `R (D)COM Server` and `rcom`
  - data transfer: `VARIANT`s, special values, COM objects
  - call-back interfaces for text and graphics output
  - using custom interface and dispinterface
- Excel and VBA
- Visual Basic 6 and Visual Basic .NET
- Configuration and Installation
- Enhanced Topics
- Debugging and Error Handling
- Resources

COM (1/3)

- Component Object Model
- defines a standard for interoperability
  - comparable to libraries
  - object-oriented (more specifically: component-oriented)
- basic unit is a “component”
  - encapsulates operations (methods), data (properties) and state
  - life-cycle management by reference counting
  - object creation
  - component is a “COM server”
  - user of component is a “COM client”
- types of components
  - in-process: DLL, lives in the client’s address space
  - out-of-process: EXE, component lives in server’s address space

COM (2/3)

- COM `interface`
  - methods (functions, operations) – visible by client
  - properties (variables) – visible by client, only a special form of methods
  - (internal) state – invisible by client
  - must not be changed (even no new methods)
  - abstract definition
  - identified by its `IID` (interface identifier, 128 bit numeric)
- `coclass`
  - implementation of a COM interface
  - multiple coclasses can implement the same interface
  - identified by its `CLSID` (class identifier, 128 bit numeric) or by the "`progname`" (human-readable name)

COM (3/3)

- COM client
  - is programmed against interface
  - reference to `coclass` is established at run-time
- locally and remote
  - DCOM allows transparent use of remote components
  - application does not have to be changed for remote use
  - configuration on COM client's and on COM server's computer
- synchronization
  - COM object (component) lives in an "apartment" (comparable to a thread context)
  - execution context of a method call is always the component's apartment
  - all method calls are synchronized into this apartment
  - *no parallel computations on a single R "instance"*

Interface `IStatConnector`

- defines methods for access to R

```
interface IStatConnector : IDispatch
{
  HRESULT Init([in] BSTR bstrConnectorName);
  HRESULT Close();
  ...
  HRESULT GetSymbol([in] BSTR bstrSymbolName,
                    [out,retval] VARIANT* pvData);
  HRESULT SetSymbol([in] BSTR bstrSymbolName,
                    [in] VARIANT vData);

  HRESULT Evaluate([in] BSTR bstrExpression,
                   [out,retval] VARIANT* pvData);
  HRESULT EvaluateNoReturn([in] BSTR bstrExpression);
  ...
};
```

Interface IStatConnector

- startup and termination

```
interface IStatConnector : IDispatch
{
  HRESULT Init([in] BSTR bstrConnectorName);
  HRESULT Close();
  ...
  HRESULT GetSymbol([in] BSTR bstrSymbolName,
                    [out,retval] VARIANT* pvData);
  HRESULT SetSymbol([in] BSTR bstrSymbolName,
                    [in] VARIANT vData);

  HRESULT Evaluate([in] BSTR bstrExpression,
                   [out,retval] VARIANT* pvData);
  HRESULT EvaluateNoReturn([in] BSTR bstrExpression);
  ...
};
```

Interface IStatConnector

- data transfer

```
interface IStatConnector : IDispatch
{
  HRESULT Init([in] BSTR bstrConnectorName);
  HRESULT Close();
  ...
  HRESULT GetSymbol([in] BSTR bstrSymbolName,
                    [out,retval] VARIANT* pvData);
  HRESULT SetSymbol([in] BSTR bstrSymbolName,
                    [in] VARIANT vData);

  HRESULT Evaluate([in] BSTR bstrExpression,
                   [out,retval] VARIANT* pvData);
  HRESULT EvaluateNoReturn([in] BSTR bstrExpression);
  ...
};
```

Interface IStatConnector

- evaluating R code

```
interface IStatConnector : IDispatch
{
  HRESULT Init([in] BSTR bstrConnectorName);
  HRESULT Close();
  ...
  HRESULT GetSymbol([in] BSTR bstrSymbolName,
                    [out,retval] VARIANT* pvData);
  HRESULT SetSymbol([in] BSTR bstrSymbolName,
                    [in] VARIANT vData);

  HRESULT Evaluate([in] BSTR bstrExpression,
                   [out,retval] VARIANT* pvData);
  HRESULT EvaluateNoReturn([in] BSTR bstrExpression);
  ...
};
```

Custom and `dispinterface`

- custom interface
  - similar to class interface in C++
  - fast
  - allows all IDL data types
  - cannot be accessed by scripting languages (e.g. VBScript, Perl, Python)
  - type-safe
- `dispinterface`
  - indirect function call (by “name“)
  - dynamic resolution of function address at runtime
  - slower
  - only a subset of types allowed
  - can be accessed by scripting languages and rcom (COM client in R)
- `IStatConnector` provides both

Use of `IStatConnector` (1/2)

1. create COM object
2. init COM object (call `Init`)
3. work with COM object
   - transfer data to R (`SetSymbol`)
   - perform computations (`Evaluate`, `EvaluateNoReturn`)
   - transfer results to application (`GetSymbol`)
4. close R (call `Close`)
5. release COM object

Use of `IStatConnector` (2/2)

- R (D)COM Server
  - R in “background“ (invisible)
  - every objects gets its own process (different symbol spaces)
  - R‘s text and character output may be “redirected“
- rcom
  - R in “foreground“
  - user can use R in parallel to applications
  - every object accesses the same R process (shared symbol space)
  - text and character output in R window
- late binding
  - both `coclasses` implement same COM interface
  - application can use both of them
  - different `CLSID` or `progname` on object creation

Data Transfer: VARIANTs

- `VARIANT`s can contain *any* data
  - numbers
  - strings (Unicode strings)
  - arrays/matrices of numbers and strings
  - objects
- contents are determined by the VARIANT's type (VT)
- most languages allow transparent use (e.g. Visual Basic 6, Visual C#)
- a bit harder to use in C or C++

Data Transfer: Special Values

- R's special values
  - NULL, NaN, -Inf, Inf, NA
- representation in VARIANT
  - VARIANT type `VT_ERROR`
  - compatible with Microsoft Excel's representation

R Value	converts to	back to R
NULL	xlErrNULL	NULL
NaN	xlErrDiv0	Inf
-Inf	xlErrDiv0	Inf
Inf	xlErrDiv0	Inf
NA	xlErrNA	NA

  - `NaN` will become `xlErrNum` in next major release of R

Data Transfer: COM Objects

- rcom is DCOM client and server for R
- allows to access COM objects from R
- only makes use of `dispinterfaces`
- single objects (`IDispatch`) can be sent to R and back
- transferring array of objects at once is not supported
- objects are reference-counted in R
- the R garbage collector will release (NULL) references
  - can be triggered explicitly using `gc()`

Text and Graphics Output

- R (D)COM Server allows redirection of text and graphics output
  - Active X control for capturing text output available
  - Active X control for graphics output available
  - COM object for (programmatically) capturing text output available
- additional functions in `IStatConnector`

```
interface IStatConnector : IDispatch
{
  ...
  HRESULT AddGraphicsDevice([in] BSTR bstrName,
                            [in] ISGFX* pDevice);
  HRESULT RemoveGraphicsDevice([in] BSTR bstrName);
  HRESULT SetCharacterOutputDevice([in]
             IStatConnectorCharacterDevice* pCharDevice);
  ...
}
```

Demo

IDL Files

OLE View

Object Browser

dcomcnfg

Agenda: Excel and VBA

- Basics
- Excel and VBA
  - use of `R (D)COM Server` and `rcom` from Excel
  - data transfer: `R`-specific types
  - prototyping applications hosted by Excel using R in VBA
- Visual Basic 6 and Visual Basic .NET
- Configuration and Installation
- Enhanced Topics
- Debugging and Error Handling
- Resources

R (D)COM Server and rcom

- R(D)COM server:
  - Invisible process, might even live on a different machine
- rcom package (R as COM server)
  - Can be visible
  - Allows simultaneous access to R data space from command line and from client application

R (D)COM Server and rcom

- Three modes for using R from Excel
  - Scratchpad mode
  - Macro programming for simple applications
  - Interactive embedding in spreadsheet recalculation

R and Excel

- **Scratchpad mode: menu operations**
  - **Put**
  - **Get**
  - **Run**

R and Excel

- Macro mode with simple interface
  - Excel ranges correspond to dataframes, vectors, and arrays

```
Sub RegreDemo()
    Call RInterface.StartRServer
    Call RInterface.PutDataframe("mydf", _
          Range("Regression!A1:C26"))
    Call RInterface.RRun("attach(mydf)“)
    Call RInterface.GetArray(,_ _
         "lm(y~x1+x2)$coefficients", _ _ _
         Range("Regression!F2"))
    Call RInterface.StopRServer
End Sub
```

R and Excel

- R embedded in spreadsheet formulas

  REval

  RApply

  RCall

  RPut

Data Transfer: R-specific Types

- Standard types in Excel
  - reals
  - string
- R types
  - date
  - complex
  - factor

Prototyping Applications

- Easily create menus and dialog boxes to control R
- Excel as GUI for canned applications
- Easily formatted tables
- Interactive graphics

Demo

Sratchpad use

Simple macro based application

R embedded in cell formulas

Agenda: VB6 and VB.NET

- Basics
- Excel and VBA
- Visual Basic 6 and Visual Basic .NET
  - using `R (D)COM Server` in Visual Basic 6
  - using `rcom` instead
  - data transfer: scalars, arrays, `VARIANT` arrays
  - data transfer: special values, COM objects
  - capturing text output
  - embedding `R` graphics in your application
  - Visual Basic .NET
- Configuration and Installation
- Enhanced Topics
- Debugging and Error Handling
- Resources

R (D)COM Server in VB 6 (1/3)

- R is invisibly run in the background
- `dispinterface`
  - reference variable is of type `Object`
  - object creation using `CreateObject` function (ProgID is "`StatConnectorSrv.StatConnector`")
  - untyped use shows error when faulty statement is run
  - can also run on machines where component is not available
- custom interface
  - requires a reference to "StatConnectorSrv 1.1 Type Library"
  - reference variable is of type `StatConnector` or `StatConnectorSRVLib.StatConnector`
  - object creation using operator `New`
  - type-safe (shows many errors at compile time)
  - slightly faster
  - better IDE support (shows function names for completion)

R (D)COM Server in VB 6 (2/3)

- **dispinterface**

```
Dim r As Object
Set r =
  CreateObject("StatConnectorSrv.StatConnector")
r.Init "R"
r.EvaluateNoReturn "plot(1:100)"
r.Close
```

- custom interface

```
Dim r As StatConnector
Set r = New StatConnector
r.Init "R"
r.EvaluateNoReturn "plot(1:100)"
r.Close
```

R (D)COM Server in VB 6 (3/3)

- dispinterface

```
Dim r As Object
Set r =
  CreateObject("StatConnectorSrv.StatConnector")
r.Init "R"
r.EvaluateNoReturn "plot(1:100)"
r.Close
```

- custom interface

```
Dim r As StatConnector
Set r = New StatConnector
r.Init "R"
r.EvaluateNoReturn "plot(1:100)"
r.Close
```

`rcom` in Visual Basic 6 (1/3)

- COM object and R window (*rgui* or *rterm*) use same R environment (data, symbols)
- same advantages of dispinterfaces and custom interfaces as with R (D)COM Server
- `dispinterface`
  - ProgID is "`RCOMServerLib.StatConnector`"
- custom interface
  - requires a reference to "RCOM 1.0 Type Library"
  - reference variable is of type `InternalConnector` or `RCOMServerLib.InternalConnector`

rcom in Visual Basic 6 (2/3)

- **dispinterface**

```
Dim r As Object
Set r =
  CreateObject("RCOMServerLib.StatConnector")
r.Init "R"
r.EvaluateNoReturn "plot(1:100)"
r.Close
```

- custom interface

```
Dim r As InternalConnector
Set r = New InternalConnector
r.Init "R"
r.EvaluateNoReturn "plot(1:100)"
r.Close
```

rcom in Visual Basic 6 (3/3)

- dispinterface

```
Dim r As Object
Set r =
  CreateObject("RCOMServerLib.StatConnector")
r.Init "R"
r.EvaluateNoReturn "plot(1:100)"
r.Close
```

- custom interface

```
Dim r As InternalConnector
Set r = New InternalConnector
r.Init "R"
r.EvaluateNoReturn "plot(1:100)"
r.Close
```

Data Transfer: Scalars, Arrays

- scalar values (integer, logical, real, strings)
- multi-dimensional arrays of scalars
- SetSymbol sends data to R

```
r.SetSymbol "v1", 25
r.SetSymbol "v2", 3.1415
r.SetSymbol "v3", "a text value"
Dim a(2) As Double        ' 3-dim array
r.SetSymbol "v4", a
Dim b(1, 2) As Integer    ' 2x3-dim array
r.SetSymbol "v5", b
```

- GetSymbol reads data from R

```
x0 = r.GetSymbol("v1")
x1 = r.GetSymbol("v2")
x2 = r.GetSymbol("v3")
a = r.GetSymbol("v4")    ' 3-dim array
b = r.GetSymbol("v5")    ' 2x3-dim array
```

Data Transfer: VARIANT Arrays

- multi-dimensional arrays of different types of scalars
  - e.g., $\begin{bmatrix} "Age" & "Student" \\ 27 & TRUE \\ 39 & FALSE \end{bmatrix}$
- represented as untyped (VARIANT) array

```
Dim s(2, 1)
s(0, 0) = "Age"
s(0, 1) = "Student"
s(1, 0) = 27
s(1, 1) = True
s(2, 0) = 39
s(2, 1) = False
r.SetSymbol "Students", s
```

Data Transfer: Special Values

- to use xlErr* constants add a reference to “Microsoft Excel x.y Type Library“

xlErr* constant	numeric value
xlErrNA	2042
xlErrDiv0	2007
xlErrNull	2000

- use `CVErr` to create/access special values

```
r.SetSymbol "sv1", CVErr(xlErrNA)
r.SetSymbol "sv2", CVErr(xlErrDiv0)
s1 = r.GetSymbol("sv1")
If s1 = CVErr(xlErrNA) Then
  ...
End If
```

- `xlErrNull` should not be used at the moment

Data Transfer: COM Objects

- `rcom` is a COM client library for R, too
  - COM objects can be accessed from R
  - only dispinterfaces are available to `rcom`
- COM objects can be transferred
  - rcom must be loaded in R
  - only a single object can be tranferred at one (no arrays)
  - use `SetSymbol` and `GetSymbol` for transfer of COM objects
  - the `rcom` function `comThis()` returns a handle to the active COM server in R

Demo

R (D)COM Server
and
rcom
in Visual Basic 6

Text Output: `StringLogDevice` (1/2)

- redirect R's text output in a variable
  - only works with R (D)COM Server (not with `rcom`)
- `coclass` StringLogDevice
  - add a reference to "StatConnTools"
  - use a variable of type `StringLogDevice` or `StatConnTools.StringLogDevice`
  - create object using operator `New`
  - ProgID is (for use with `CreateObject` function)
- bind object to `IStatConnector` object
  - call `BindToServerOutput` after `Init`
  - use the `Text` property to access R's output
  - R will append new text unless `Text` is cleared

Text Output: `StringLogDevice` (2/2)

```
Dim r As New StatConnector
r.Init "R"
Dim s As New StringLogDevice
s.BindToServerOutput r
...
s.Text = ""
r.EvaluateNoReturn "cat(""test""")
MsgBox s.Text
s.Text = ""
...
```

- Gives access to R's messages in case of an error

`StatConnectorCharacterDevice`

- redirect R's text output in a window
  - only works with R (D)COM Server (not with `rcom`)
- ActiveX control `StatConnectorCharacterDevice`
  - add components to the toolbox from "StatConnectorClnt 1.0 Type Library
- put a `StatConnectorCharacterDevice` on a form
- tell the IStatConnector object to use the control
  - `r.SetCharacterOutputDevice myDevice`
- all text output will be appended to the control
- call the control's `Clear` function to clear the text box

Graphics Output (1/2)

- redirect R's graphics output in a window
  - only works with R (D)COM Server (not with `rcom`)
- ActiveX control `StatConnectorGraphicsDevice`
  - add components to the toolbox from "StatConnectorClnt 1.0 Type Library
  - or add components to the toolbox from "StatConnControls"
- StatConnectorClnt 1.0 Type Library
  - put a `StatConnectorCharacterDevice` on a form and set the visual properties
  - call `r.AddGraphicsDevice "gfx", gfxDev.GetGFX`
- StatConnControls
  - put a GraphicsDevice on a form and set the visual properties
  - ```
    gfxDev.DeviceName = "gfx"
    Set gfxDev.Connector = r
    gfxDev.AddToConnector
    ```
    ```

Graphics Output (2/2)

- only one device can be active at the moment
  - remove the device before installing another
- multiple devices can be on the same form

```
' gfxDev1 is a StatConnectorGraphicsDevice
' gfxDev2 is a GraphicsDevice
r.AddGraphicsDevice "gfx", gfxDev1.GetGFX
r.EvaluateNoReturn "plot(sin(1:100))"
r.RemoveGraphicsDevice "gfx"
gfxDev2.DeviceName = "gfx"
Set gfxDev2.Connector = r
gfxDev2.AddToConnector
r.EvaluateNoReturn "plot(1:100)"
gfxDev2.RemoveFromConnector
```

Demo

Text and Graphics Output in Visual Basic 6

Visual Basic .NET: Basics

- VB.NET is very similar to Visual Basic 6 at first glance
- really a new language designed for .NET framework
- using R (D)COM Server and rcom very similar to Visual Basic 6
  - add reference of type COM to the required components
  - use the same code as in VB6
- special value support different
  - use `System.Runtime.InteropServices.ErrorWrapper` as parameter to `SetSymbol`
  - special values cannot be read and interpreted in Visual Basic .NET
  - *we‘re working on this topic and keep you informed!*

Visual Basic .NET: Devices

- devices are used the same way as in VB6
- use “choose items“
  - check “StatconnectorCharacterDevice class“
  - check “StatconnectorGraphicsDevice class“
- install the devices programmatically

```
Dim lConnector As STATCONNECTORSRVLib.IStatConnector
lConnector = New STATCONNECTORSRVLib.StatConnector
lConnector.Init("R")
lConnector.SetCharacterOutputDevice(charDev.GetOcx())
lConnector.AddGraphicsDevice("gfx", gfxDev.GetGFX())
lConnector.EvaluateNoReturn("cat(""Testing...\n"")")
lConnector.EvaluateNoReturn("plot(1:100)")
lConnector.EvaluateNoReturn("cat(""Done\n"")")
lConnector.Close()
```

Demo

Using R (D)COM Server
and
rcom
in
Visual Basic .NET

Configuration and Installation

- Basics
- Excel and VBA
- Visual Basic 6 and Visual Basic .NET
- Configuration and Installation
  - building your own setup
  - configuring R for remote usage
- Enhanced Topics
- Debugging and Error Handling
- Resources

Building Your Own Setup

Deploying and Maintaining DCOM Applications

Balasubramanian Narasimhan
Department of Statistics and Health Research and Policy
Standford University
Stanford, CA 94305

Remoting of R: Basics

- remoting of a COM server
  - COM server machine specified programmatically
  - or by configuration (using `dcomcnfg.exe`)
  - always keep in mind, that client and server are running on different machines (file system, permissions, GUI)
- client machine
  - runs application using the COM server
  - needs configuration information to tell where the COM object should be run
  - remote user name and password can be configured
- server machine
  - requires startup configuration for COM server
  - security setup

Remoting of R: Use Code

- using Windows API (e.g. in C/C++)
  - use CoCreateInstanceEx
  - fill COSERVERINFO structure
- using SCTools.dll
  - function CreateRemoteObject
  - can be used from VB and VBA, too
  - installed with R (D)COM Server setup
- configuration of server machine required
  - security setup
- configuration of client machine not necessary

Client Configuration

- start `dcomcnfg.exe`
- open properties of `StatConnectorSrv` under "`DCOM Config`"
  - on the “Location“ tab check uncheck “Run application on this computer“
  - check “Run application on the following computer“ and add the computer name
- whenever an application creates an instance of StatConnector, it is created on the remote machine
  - it is not necessarily run under the interactive (logged on) user
  - local graphics or text output is not possible
  - access to the file system is on the remote machine
  - access to R and installed packages is on the remote machine

Server Configuration

- add a user account
  - the same name and password as on client machine
- start `dcomcnfg.exe`
- open properties of “My Computer“
  - check “Enable Distributed COM on this computer“
  - set “Default Authentication Level“ to “Connect“
  - set “Default Impersonation Level“ to “Identify“
  - make sure the “Access Permissions“ and “Launch and Acitvation Permissions“ include the correct user account
- open properties of `StatConnectorSrv` under "`DCOM Config`"
  - make sure all options are set to “(Use) Default“
  - Identity should be “The launching user“ or a specified user for R

Some Notes on Security

- same user, same password
  - easy to make and easy to use configuration
  - no authentication issues
  - also works in a domain for domain users
- identitiy
  - can be specified on server machine (user, password)
  - can even run as interactive user
  - may have different access rights than local user
- general configuration issues
  - firewall: its best to disable for first setup, then re-enable when the setup is already running
  - DCOM: make sure DCOM is enabled on the server machine

Demo

Running R remotely

Client and Server Configuration

Agenda: Enhanced Topics

- Basics
- Excel and VBA
- Visual Basic 6 and Visual Basic .NET
- Configuration and Installation
- Enhanced Topics
  - source code walk-through: `RExcel`
  - using Visual C# instead of Visual Basic 6
  - determine COM server at runtime (C#)
  - expose your objects to R: C#
  - expose your objects to R: Visual Basic 6
  - using C and C++ to access R
  - implementation of text output device
- Debugging and Error Handling
- Resources

Source Code: RExcel

- Examples in Demo programs coming with RExcel

Visual C#

- C# is similar to Java in its syntax
- easy to learn
  - with experience in C++ or Java in a few hours
  - full integration into IDE
  - ideal language for advancing Visual Basic developers
- easy to handle
  - more powerful than (old) Visual Basic 6
  - as easy to handle as Visual Basic 6 or .NET
  - compile time checking like C++
  - runtime checking like Java or Visual Basic 6 and .NET
- using R similar to VB.NET
  - uses same class libraries (.NET)
  - same interop mechanisms for COM
  - even ActiveX controls can be used similar to Visual Basic.NET

Demo

Using R (D)COM Server
and
rcom
in
Visual C# .NET

R calls C# (1/5)

- create "Class Library" project
  - in project properties, category "Build" check "Register for COM interop"
- create `interface` and `class`
  - `interface` needs `Guid` and `InterfaceType` attributes
  - `class` needs `Guid` and `ComVisible` attributes, must inherit from `interface`
  - `ProgId` is `ProjectName.ClassName`
- prepare for callbacks to R
  - add a reference to `RCOM 1.0 Type Library`
  - define a public property “`Connector`“ of type `RCOMServerLib.InternalConnector`
- define methods and functions
- compile & deploy

R calls C# (2/5)

- use rcom for access to the object
  - comCreateObject() to create an object
  - comGetObject() to access an existing object
  - comInvoke() to call (invoke) a function
  - comSetProperty() to set a property
  - comGetProperty() to get a property‘s value
  - comThis() to get a handle to the “current“ COM server
  - comGetObjectInfo() return some method and property information about the object
  - comIsValidHandle() check if an object is valid
  - comGetVersion() return rcom major and minor version

R calls C# (3/5)

- Visual C# code: interface

```
[Guid("3ddfd021-a0c6-4218-a253-4fc4328c99a7"),
 InterfaceType(ComInterfaceType.InterfaceIsDual)]
interface IMyComponent
{
  RCOMServerLib.IStatConnector Connector
  {
    set;
  }
  string Text
  {
    set;
  }
  void DoCallback();
}
```

R calls C# (4/5)

- Visual C# code: implementation

```
[Guid("133fee0e-9b31-4429-8a43-6e2a707a9beb"), ComVisible(true)]
public class MyComponent : IMyComponent {
  private string mText;
  private RCOMServerLib.IStatConnector mConnector;

  public RCOMServerLib.IStatConnector Connector {
    set { mConnector = value; }
  }
  public string Text { set { mText = value;}
  }
  public void DoCallback() {
    if (mConnector != null) {
      mConnector.EvaluateNoReturn("cat(\"DoCallback: "
          + mText + "\")\n");
    }
  }
```

R calls C# (5/5)

- R code

```
x<-comCreateObject("CalledByR.MyComponent")
comSetProperty(x,"Text","xxx")
comSetProperty(x,"Connector",comThis())
comInvoke(x,"DoCallback")
```

- generating GUIDs
  - use the Microsoft tools to generated GUIDs
  - `uuidgen.exe` is a console application
  - `guidgen.exe` is a Windows dialog application
  - both tools are part of Visual Studio

R calls Visual Basic 6 (1/2)

- create ActiveX EXE or ActiveX DLL project
  - EXE creates and outproc server, DLL an inproc server
- create a new public class
  - `ProgId` is `ProjectName.ClassName`
- prepare for callbacks to R
  - add a reference to `RCOM 1.0 Type Library`
  - define a public property "`Connector`" of type `InternalConnector`, define an accessor function for the property ("`property let`", not "`property set`")
- define methods and functions
- compile & deploy
  - use `regsvr32.exe` to register for ActiveX DLL projects (`regvr32.exe /u` unregisters)
  - use `mySrv.exe /RegServer` to register ActiveX EXE projects
  - ActiveX DLL (`mySrv.exe /UnregServer` unregisters)

R calls Visual Basic 6 (2/2)

- Visual Basic 6 code

```
Private mConn As InternalConnector
Public Text As String
Property Let Connector(ByVal myconn As Object)
 Set mConn = myconn
End Property
Public Sub DoCallback()
 mConn.EvaluateNoReturn "cat(""DoCallback: " & Text & "\n"")"
End Sub
```

- R code

```
x<-comCreateObject("CalledByR.ClassCalledByR")
comSetProperty(x,"Text","xxx")
comSetProperty(x,"Connector",comThis())
comInvoke(x,"DoCallback")
```

Demo

R

calls

Visual C#

and

Visual Basic 6

C/C++: Basics

- C and C++ should use the custom interface
  - for C++ programmers like a pointer to a C++ object
  - for C programmers like a struct with function pointers
- standard COM mechanisms
  - reference counting (AddRef(), Release())
  - multiple inheritance (QueryInterface())
  - object creation (CoCreateInstance())
  - IDispatch functions (normally not used in C++)
- interface defined in IDL
  - `StatConnectorSrv.idl`
  - `IID`, `CLSID` and `LIBID` in `StatConnectorSrv_i.c`

C/C++: Error Handling

- every function returns `HRESULT`
  - standard COM error handling mechanisms
  - use macros `SUCCEEDED()` and `FAILED()`
  - interface-specific error codes at `0x8004000`
- must check errors on every function call
- interface-specific error codes

Error Codes (1/2)

preprocessor define	hex. value	description
SCN_E_INITIALIZED	0x8004005	Init not called
SCN_E_NOTINITIALIZED	0x8004006	Init already called
SCN_E_INVALIDSYMBOL	0x8004007	symbol not found
SCN_E_PARSE_INVALID	0x8004008	invalid expression
SCN_E_PARSE_INCOMPLETE	0x8004009	incomplete expression
SCN_E_UNSUPPORTEDTYPE	0x800400A	data type not supported
SCN_E_EVALUATE_STOP	0x800400B	evaluation stopped because of an error
SCN_E_INVALIDINTERFACEVERSION	0x8004010	interpreter interface version mismatch (rproxy does not match R (D)COM Server)
SCN_E_INVALIDINTERPRETERVERSION	0x8004011	interpreter version mismatch (reserved for future use)

Error Codes (2/2)

preprocessor define	hex. value	description
SCN_E_INTERFACENOTFOUND	0x8004012	reserved for future use
SCN_E_LIBRARYNOTFOUND	0x8004013	`rproxy.dll` could not be loaded
SCN_E_INVALIDLIBRARY	0x8004014	invalid `rproxy.dll`
SCN_E_INITIALIZATIONFAILED	0x8004015	initialization failed
SCN_E_INVALIDCONNECTORNAME	0x8004016	connector name “R“ expected
SCN_E_INVALIDINTERPRETERSTATE	0x8004016	reserved for future use
SCN_E_FATALBACKEND	0x8004020	access violation in R (interpreter, package code)
SCN_E_INVALIDARG	0x8004001	reserved for future use
SCN_E_INVALIDFORMAT	0x8004002	reserved for future use
SCN_E_NOTIMPL	0x8004003	reserved for future use
SCN_E_UNKNOWN	0x8004004	reserved for future use

C/C++: VARIANTs

- data transfer uses VARIANT data type
  - includes `VT_BYREF` support
- scalar support
  - `VT_BOOL`, `VT_I2`, `VT_I4`, `VT_UI1`, `VT_R4`, `VT_R8`, `VT_BSTR`
  - `VT_EMPTY` (maps to `NULL`)
  - `VT_ERROR` (`xlErrDiv0`, `xlErrNA`, `xlErrNull`)
- arrays of same scalars (any dimensions)
  - optimized data transfer (e.g. 2x5 `VT_R8` are 10 `doubles`)
- arrays of different scalars
  - mixed transfer, not as efficient as shown above
- objects
  - `VT_DISPATCH`
  - `rcom` is required in R

C/C++: Examples

```
#include "StatConnectorSrv_c.i"
{
  IStatConnector lConnector;
  VARIANT lVar;
  CoCreateInstance(CLSID_StatConnector,NULL,CLSCTX_ALL,
       IID_IStatConnector,(LPVOID*) &m_Connector);
  lConnector->Init(L"R");
  VariantInit(&lVar);
  V_VT(&lVar) = VT_R8;
  V_R8(&lVar) = 3.1415;
  lConnector->SetSymbol(L"myPi",lVar);
  lConnector->EvaluateNoReturn(L"plot(myPi*1:100)");
  lConnector->Close();
  lConnector->Release();
}
```

Custom Text Output Device

- implement interface **`IStatConnectorCharacterDevice`**
  - `HRESULT WriteString([in] BSTR bstrLine);`
  - `HRESULT WriteStringLevel(`
    `[in] BSTR bstrLine,[in] LONG lLevel);`
  - `HRESULT Clear();`
- pass interface pointer to **`SetCharacterOutputDevice`** of **`IStatConnector`**

Demo

Using R (D)COM Server
and
rcom
in
C and C++

Debugging and Error Handling

- Basics
- Excel and VBA
- Visual Basic 6 and Visual Basic .NET
- Configuration and Installation
- Enhanced Topics
- Debugging and Error Handling
  - error handling: Visual Basic 6, Visual Basic .NET
  - error handling: C#
  - debugging hints
  - useful tools
- Resources

Error Handling in VB6 (1/2)

- errors are returned by COM object's functions as HRESULTs
- use "On Error Goto" statement

```
...
On Error Goto Handler   ' enable error handler
r.SetSymol "a", 25
b = r.GetSymbol("b")
On Error Goto 0         ' restore default
Exit Sub
:Handler
' perform cleanup code
```

Error Handling in VB6 (2/2)

- use "On Error Resume Next" statement

```
...
On Error Resume Next    ' disable error handler
r.SetSymol "a", 25
If Err.Number <> 0 Then
  ' Error handling code
End If
...
On Error Goto 0         ' restore default
```

- `Err.Number` contains error code (HRESULT)
- call `GetErrorText` for a textual description

Error Handling in VB.NET

- use Try/Catch/Finally statements

```
Dim lConnector As IStatConnector
lConnector = New StatConnector

Try
  lConnector.Init("R2") ' faulty statement
Catch lExc As COMException
  MessageBox.Show("Error: " & _
     lConnector.GetErrorText())
End Try
```

Error Handling in C#

- use try/catch/finally statements

```
try
{
  STATCONNECTORSRVLib.IStatConnector lConnector;
  lConnector = new
  STATCONNECTORSRVLib.StatConnector();

  lConnector.Init("R");
  lConnector.EvaluateNoReturn("plot(1:100)");
  lConnector.Close();
}
catch (COMException lExc)
{
  System.Windows.Forms.MessageBox.Show("Error");
}
```

Debugging: R's Error Text

- check the error code
  - error text as returned by `GetErrorText` might give a hint
  - especially useful for debugging startup problems (installation)
- if `Evaluate` or `EvaluateNoReturn` fails, check R's error output
- install a `StringLogDevice`
- check the `StringLogDevice`
- use `rcom` and check the console window

Useful Tools

- OLEView
  - shows COM interfaces and type libraries
  - gives insight into COM objects used with, e.g. `rcom`
- DebugView
  - download from http://www.sysinternals.com/
  - shows traces made with `OutputDebugString()` API function
  - `rcom` shows debug messages on DebugView
- Error Lookup
  - gives information on error codes (e.g. 0x80000004)
- uuidgen, guidgen
  - generate `GUIDs` (`CLSID`, `IID`, `LIBID`)

Demo

Handling
and
Debugging
Typical Error Situations

Agenda: Resources

- Basics
- Excel and VBA
- Visual Basic 6 and Visual Basic .NET
- Configuration and Installation
- Enhanced Topics
- Debugging and Error Handling
- Resources

Resources (1/2)

- http://sunsite.univie.ac.at/rcom
  - documentation a bit outdated
  - mainly for downloads
- http://learnserver.csd.univie.ac.at/rcomwiki
  - additional documentation
- http://cran.r-project.org/contrib/extra/dcom/
  - R (D)COM Server home on CRAN
- http://cran.r-project.org/src/contrib/Descriptions/rcom.html
  - rcom package home on CRAN
- http://mailman.csd.univie.ac.at/mailman/listinfo/rcom-l
  - mailing list for discussions of R (D)COM Server, rcom and RExcel
  - this is also the “official“ place to ask questions and for support

Resources (2/2)

- Papers and talks
  - http://www.ci.tuwien.ac.at/Conferences/DSC-2001/Proceedings/NeuwirthBaier.pdf
  - http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/Baier.pdf
  - http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/BaierNeuwirth.pdf
  - http://www.ci.tuwien.ac.at/Conferences/useR-2004/abstracts/Baier+Neuwirth_1.pdf
  - http://www.ci.tuwien.ac.at/Conferences/useR-2004/abstracts/Baier+Neuwirth.pdf
  - Baier, T. & Neuwirth, E., *Excel :: COM :: R*, to be published in Computational Statistics