COMponent-based Statistical Computing

A B S C H L U S S A R B E I T

zur Erlangung des akademischen Grades
Master of Science
(m. sc.)
im Masterstudiengang Statistik eingereicht an der

Wirtschaftswissenschaftlichen Fakultät
Humboldt-Universität zu Berlin

von
Herr Dipl.-Volksw. Gökhan Aydınlı
geboren am 15.10.1974 in Berlin

Gutachter:

1. Prof. Dr. Wolfgang Härdle
2. Prof. Dr. Bernd Rönz

eingereicht am: 15. Juli 2004

Abstract

Modern statistical analysis requires standardization, transparency, interactivity, and reproducibility. This thesis presents an add-in based solution building on Microsoft's COM technology which aims to fulfil these requirements. We will argue in favor of open and flexible environments within a distributed, i.e. client/server framework. Our emphasize lies on spreadsheets as suitable frontends for add-in based statistical systems.

Keywords:
Component Architectures for Computational Statistcs, Add-In solutions, Spreadsheets, Applications in Client/Server Systems

Zusammenfassung

Standardisierung und Transparenz sind Grundvoraussetzungen für moderne statistische Datenanalyse. Darüberhinaus sind Interaktivität und Reproduzierbarkeit wünschenswert. Im Rahmen dieser Arbeit wird eine Add-in basierte Lösung vorgestellt, die auf Microsofts COM Technologie beruht und versucht die genannten Ziele zu ermöglichen. Tabellenkalkulationen können dabei ein nützliches Werkzeug sein, wenn sie im Zusammenspiel mit statistischen Spezialpaketen zum Einsatz kommen.

Schlagwörter:
Komponenten Architekturen für Computergestützte Statistik, Add-In Lösungen, Tabellenkalkulationen, Praktischer Einsatz in Client/Server Systemen

Dedicated

to my Parents

Acknowledgements

I would like to thank the Deutsche Forschungsgemeinschaft SFB 373 'Simulation and Quantification of Economic Processes' and all members of the Center for Applied Statistics and Economics (CASE).

Preface

This thesis presents a high-level approach for client/server based statistical computing. The underlying technology is a component based architecture, which allows for flexible integration, reusability and expansion of existing systems

The aim is to provide statistical intelligence in a distributed manner on add-in level. The discussed tool is developed with the idea in mind to combine a well-known GUI spreadsheet with a procedural statistical language exploiting Microsoft's COM technology.

The accompanying CD-ROM contains this thesis in portable format along with the latest version of the add-in application.

This document was typeset in LaTeX according to the EDOC requirements of the Humboldt-Universität zu Berlin.

Please refer to http://edoc.hu-berlin.de/epdiss/latex/latex.html for further information.

Contents

List of Figures

List of Tables

Chapter 1

Motivation

> *Let's not kid ourselves: the most widely used piece of software for statistics is Excel.*
>
> [38]

In our understanding modern statistical analysis and efficient method proliferation require standardization, interactivity, transparency, and reproducibility. The used terminology will be explained throughout the thesis, whenever it occurs.

This chapter will discuss our motivation for statistical computing utilizing spreadsheets. We also will explain why add-in based solutions might help to overcome some of the deficiencies of spreadsheet applications.

1.1 Spreadsheets

Our modern data-oriented and computer-centric society heavily uses one category of software application: Spreadsheets. The manipulation of figures and functional relations and their conversion respectively representation in charts is the main objective behind the philosophy of spreadsheets, i.e. organizing information into machine readable columns and rows [3].

Put differently [36]:

> "[...] spreadsheet programs are the paradigm for numerical software for most users of desktop PCs."

The value of the spreadsheet lies in its flexibility. It allows one to interactively manipulate data and obtain corresponding graphical representations. In other words spreadsheets offer an interaction model radically different from an 'enriched' statistical language like `XploRe` or `R` [3].

And since a ubiquitous software vendor has had enough market power to promote his 'own' solution, we can literally recognize a *standard* for this type of software. When one talks about spreadsheets, the first think which comes to mind is probably Microsoft's Excel. Evidently Excel is not the only available spreadsheet and one might identify many reasons (compare section 1.1.3) to assume that Excel is not even the most appropriate product for statistical analysis.

But the mentioned circumstances made Excel to a widely used software suite for data analysis. As it is mainly bundled with new PCs and the according operating system, Excel became a quasi standard for working with data in professional, scientific, and educational settings [2].

1.1.1 Excel and Statistical Data Analysis

Of course Microsoft recognized the value of Excel for statistical data analysis and equipped the application with a variety of additional analysis tools. With add-on modules as the *Scenario-Manager*, the *Solver*, the *Analysis Tool-Pack* and 81 built-in statistical functions Microsoft enhanced Excel for statistical analysis (see table 1.1).

Thus Excel seems to be well suited to accomplish the usual tasks of statistical analysis given that the basic operations of Excel are known to the user. In particular these are according to [31]:

- Data input and storage

1	one/two way ANOVA
2	Covariance
3	Correlation
4	Exponential Smoothing
5	Fourier Anaylsis
6	Two-Sample F-Test
7	Histogram
8	Moving Average
9	various Two-Sample T-Tests
10	Random Number Generation
11	Rank and Percentile
12	Regression
13	Sampling

Table 1.1: Microsoft Excel statistical tools

- Data correction
- Tabular and graphical representation
- Statistical calculation
- Usage of Excel's statistical functions

Hence it may be not farfetched to assume that anyone who has worked with Windows PCs is capable of using Excel and its basic features in a short amount of time. Especially in a teaching environment it should be expectable that students are familiar with Excel. Therefore we might conclude that our first recommendation for efficient statistics is fulfilled: standardization. Of course only if one is willing to accept a proprietary quasi-standard which Microsoft gained through market power.

However since market frictions and anti-trust issues are not of concern within this research, it is enough for us to state that there is a sufficient large amount of installations and users of Excel.

1.1.2 Some Remarks on Excel's Graphical Capabilities

From personal communications with statisticians and academics I assume that Excel's graphical engine is not regarded as, say, suitable for data representation. It is often supposed that an 'ordinary' office application cannot cope with the state-of-the-art graphics produced by e.g. `SAS` or `S-Plus`.

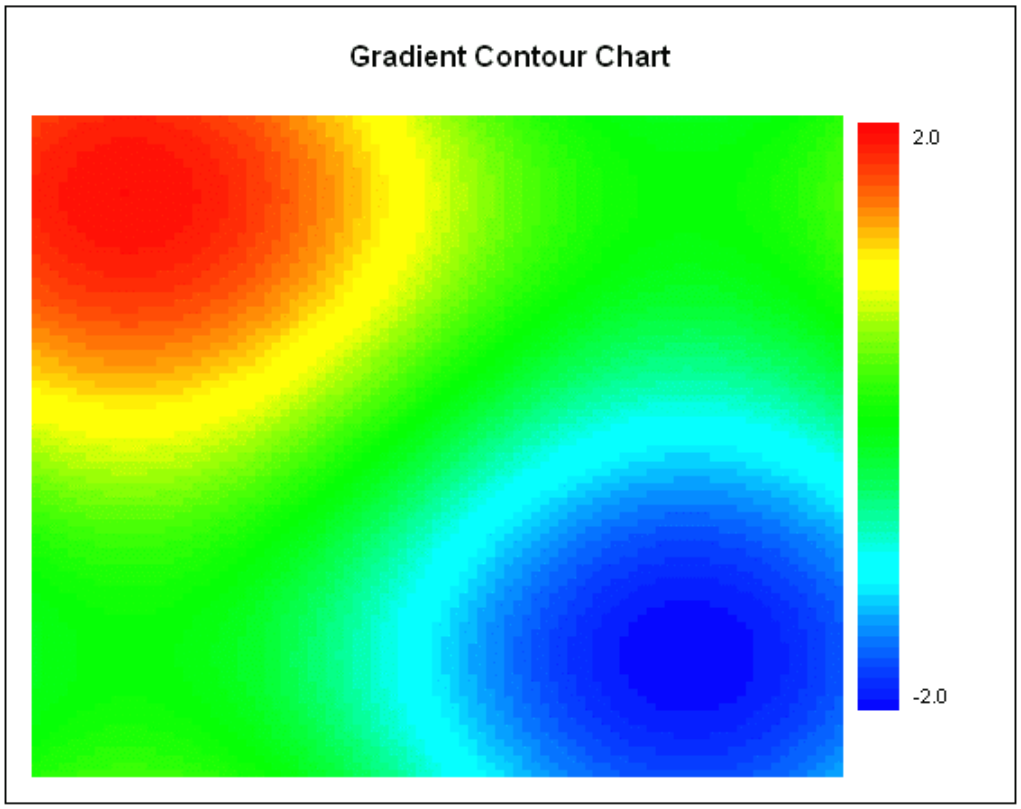


Figure 1.1: Examples of Excel Charts taken from [47]

I would like to use this opportunity to state that this is a somewhat biased attitude. With only a few mouse clicks or lines of **VBA** code the charting abilities of Excel can be expanded to generate sophisticated statistical graphics, like histograms, box-plots, scatter- or three dimensional plots. Excellent references on Excel and how to customize it via its accompanying macro language can be found in [41], [45] and [46].

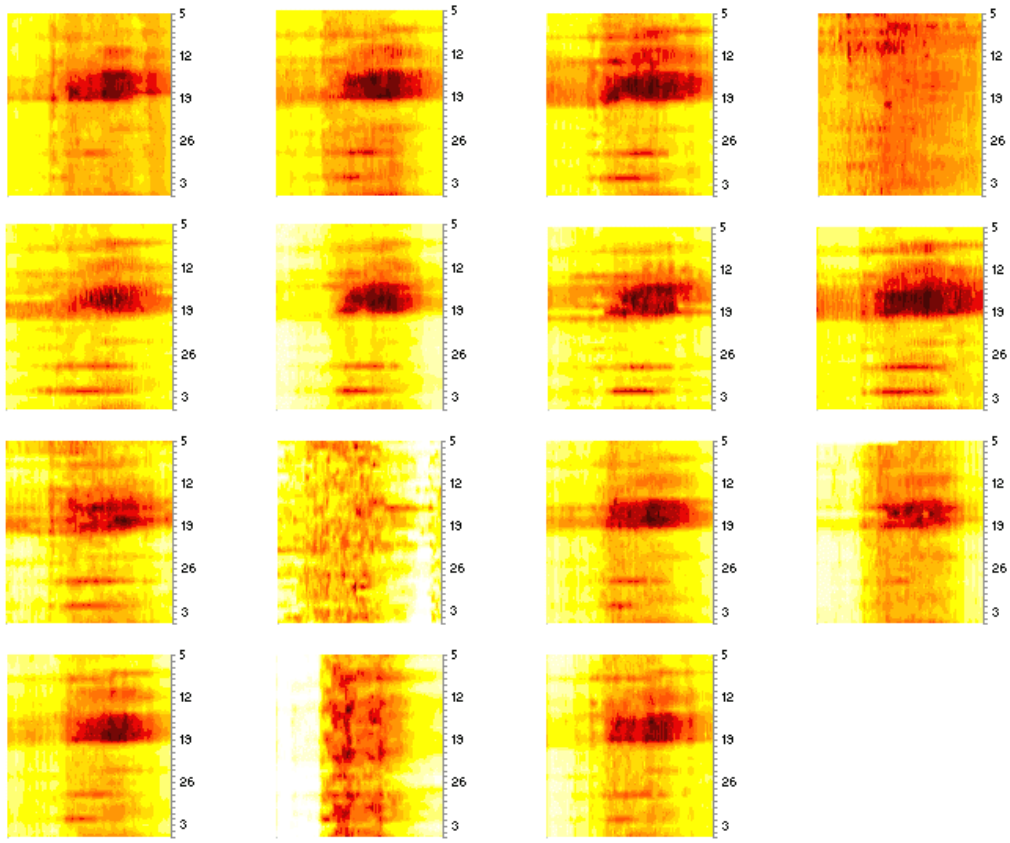


Figure 1.2: Examples of Excel Charts taken from [47]

These can furthermore equipped with sliders, buttons, and other objects for interactive exploration of (e.g. multivariate) data as seen in [36]. Slicing and brushing techniques can be implemented quite easily as well. Excel also supports the export to portable graphic formats to use for example in web publishing. We will see some examples while discussing MD*ReX in section 3.3.

Graphical representations are a central theme for statistical research and thus a huge literature exists on this topic. A general purpose overview is given in [13]. Exploratory techniques are described e.g. in [22]. Applications in multivariate statistics can be found in [20].

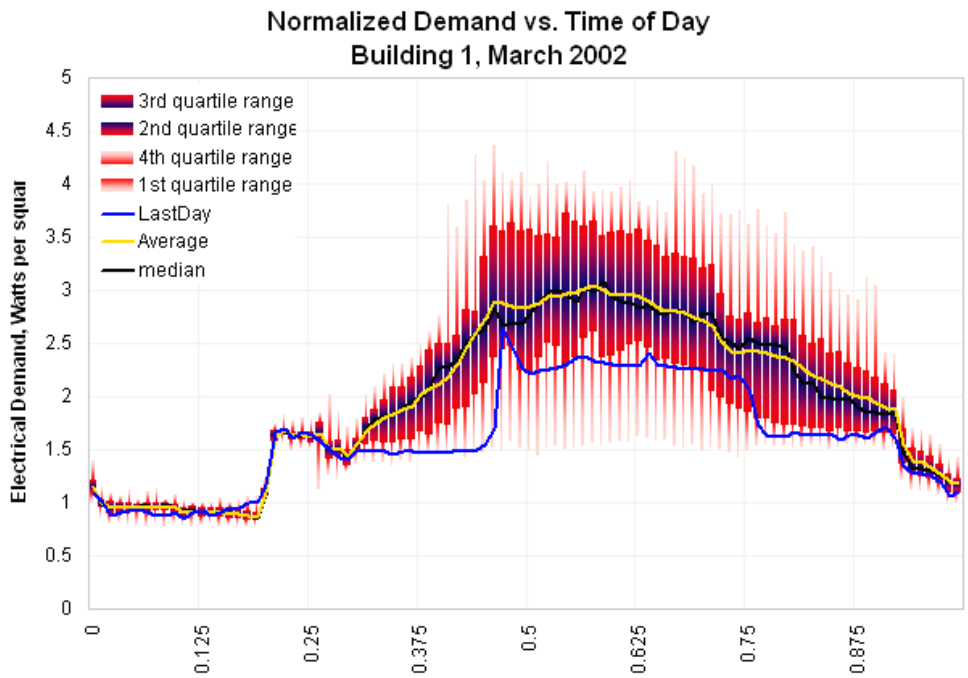


Figure 1.3: Examples of Excel Charts taken from [47]

In the context of reproducible respectively replicable, see [29], research we are encountered with further requirements. [42] proposes a 'living document' approach building upon *component technology*. The `SWEAVE` and `MD*Book` projects are LaTeX based approaches for documenting statistical research using `R` respectively `XploRe` as backend services. While the former is aimed towards generating integrated documents, i.e. LaTeX documents with `R/S` source and objects as described in [28], the latter is a system to generate various output formats ranging from PDF to XML based e-stat modules, where statistical methods (*Quantlets*) are incorporated as e.g. executable hyperlinks, see [48].

The importance of making (statistical) research replicable has been recognized by the academic community and the above mentioned approaches are only a few of the efforts to achieve this. In the `Yxilon` project this issue is a

central theme, thus pushing the `XploRe` environment into a further *componentized* architecture.

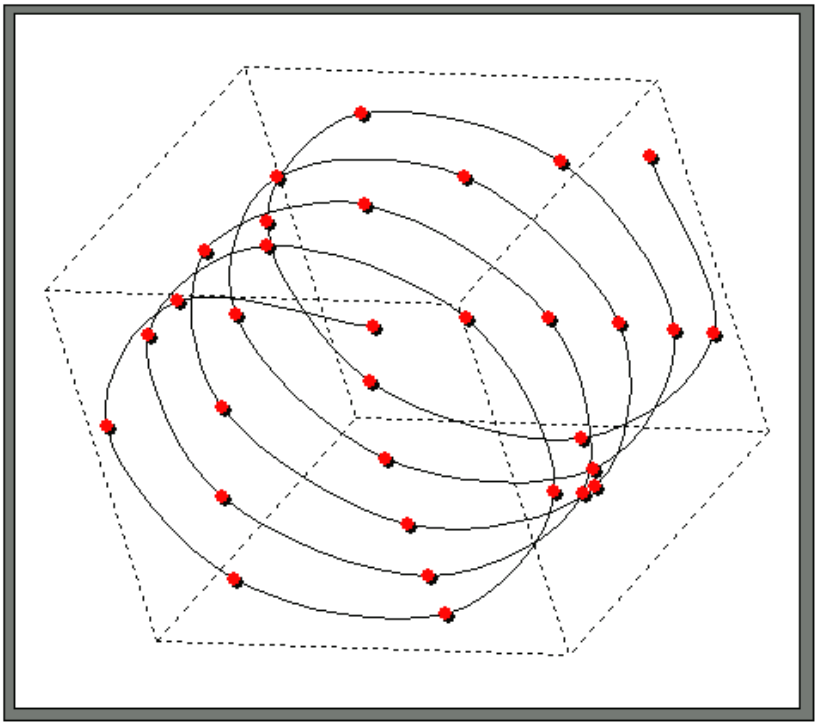


Figure 1.4: Examples of Excel Charts taken from [47]

Now, what has this to do with Excel. Excel is only one part of a whole suite of applications. The paradigm of 'living objects' has long been addressed by Microsoft and other industry players.

The features we like to play with on modern Windows operating systems like *Drag and Drop* from one document to another or *embedding* a complex Excel worksheet into a Word document are the result of a long lasting technology research beginning in the early 90s at Microsoft formerly known as **OLE**. And it was exactly this pursuit which was the driving force behind the now ubiquitous **COM** / **ActiveX** technology, see [7] and [39].

Why should one reinvent the wheel when the technologies are already there? This is one possible motivation of exploiting existing component technolo-

gies like **COM**, **CORBA** and to some extent **Web Services** in statistical computing, compare e.g. [6] and [8]. Another one might be increased performance efficiency through binary communication in distributed setups, see [5].

So wouldn't it be desirable to have modern statistical methods and documentation technologies under one common (standardized) roof? It might sound strange to statisticians, but maybe a completely component oriented, [14], office environment (e.g. Microsoft Office, SUN StarOffice or OpenOffice) is a readily available solution.

To get back to Excel's graphic engine, an excellent reference to start with is [47] which has been the source of the sample graphics in figure 1.1 to figure 1.4 which I have arbitrarily chosen for illustrative purposes.

Furthermore we might also conclude that our second requirement is fulfilled: interactivity since the direct manipulation and interaction philosophy is inherent to the spreadsheet paradigm. Moreover interaction can even be accomplished within the graphical representation of statistical analyses as shown.

1.1.3 The Risk of using Excel for Statistics

Nevertheless statisticians should be careful in exploiting the statistical features of Excel. Excel has never been designed to be a full blown statistical package. Therefore we cannot expect functionality similar to professional statistical programs. There is a lack of advanced statistical methods like seasonal time series analysis or neural networks [3].

Despite of this, there is still a growing literature, which promotes Excel itself as a tool for computational statistics. E.g. [31] remark that Excel is an 'excellent application' for statistical analysis and classify it as tool to avoid 'calculation errors'. In view of the literature on the deficiency of Excel for statistical analysis such statements should be handled carefully.

Features within Excel which should be used with due care in statistical analysis are according to [11]:

- Computing algorithms

- (prefab) Graphics
- Treatment of missing data
- Random number generators
- Regression
- Help screens

Because of the known deficiencies of Excel in the field of numerical accuracy and statistics, the literature even suggests not to use Excel at all, see e.g. [23], [10], [30], [24] or [43].

We would not go that far: Since numerical and methodological impreciseness can be circumvent by redirecting the numerical computations to a statistical backend. We will discuss involved technical aspects in the course of this thesis, see section 3.2.

Of course this immediately raises the question whether the statistical backend is reliable. In this context [29] recommends cross checking with at least two statistical packages and the rigorous implementation of benchmarks.

Anyhow another reason not to use Excel for statistical analysis is the violation of the transparency requirement. Like in any other proprietary statistical package we do not how Microsoft developers implemented, e.g. the Two-Sample F-Test or the random number generator in Excel. But there is also a solution to this issue. The statistical backend, which is in charge of the numerical computations, should provide its method implementations in open (human readable) format. This is guaranteed for example within the `XploRe` environment, see 2.1.

Combining the beneficial features of spreadsheets, namely the direct manipulation and graphical interaction abilities with powerful statistical methods, might help to promote this class of applications to well known and convenient frontends to modern statistical engines. For a discussion of the benefits of using spreadsheets to convey mathematical and statistical concepts see [33], [34], and [35].

Now how can such an integration be accomplished? As can be expected there does not exist *the* only right method or approach. We rather have the freedom to choose from various possible integration architectures. This question has to be addressed by examining the given conditions and the aimed goals of such a combination.

As mentioned we will concentrate on Excel and its facilities on the frontend side and `XploRe` on the statistical backend side. This alone seems to bear the notion of a possible client / server architecture. But even within such client / server architectures we have to cope with subtle differences and technological paradigms varying from language, platform and vendor. Since my thesis is concerned with high-level approaches, we will examine the possible integrative handles the frontend has to offer.

1.2 Add-Ins

One reason for the popularity of Excel is its strict component oriented architecture which allows the user to:

customize it either via the GUI, or e.g. the built-in macro language VBA,

automate it, i.e. repetitive tasks can be solved via batch processing or scripting,

expand its functionality through (third-party) software.

Especially the last two points created a whole industry of special purpose software / component vendors for the Microsoft Office platform. Some of them specialized for example in statistical tools for Excel, to address those issues mentioned in section 1.1.3. Other larger software manufacturers saw further market potential for own products by offering their functionality within the Microsoft Office suite. Or simply would like to provide some value added for their existing customers.

Well known examples are the Adobe Acrobat PDF converter for Office or the Bloomberg trader tools for Excel. Figure 1.5 shows both tools in Excel.

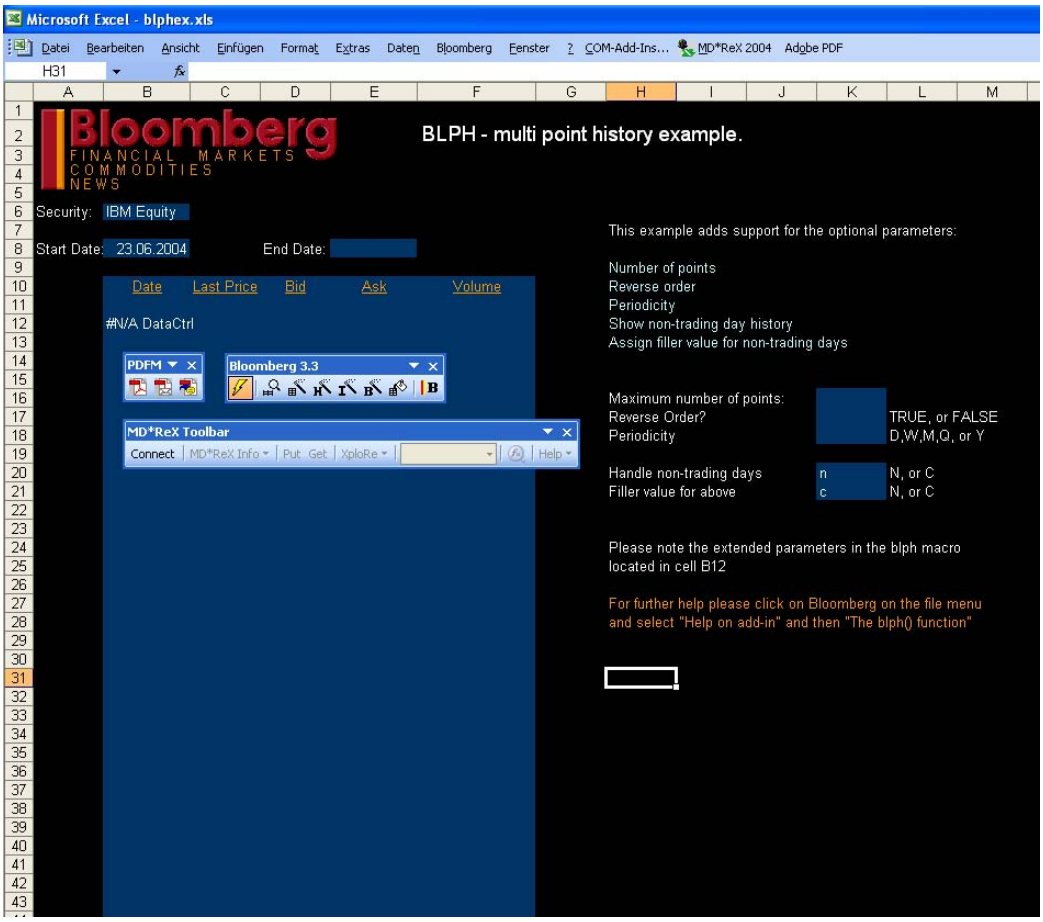


Figure 1.5: Adobe and Bloomberg Excel add-ins

What all these solutions have in common is that they are add-in based approaches. Add-ins are software applications which solely live in the execution or process runtime of another hosting application. In terms of Microsoft Office add-ins, [40] gives the following definition:

> "Speaking in the most general terms, an add-in can be thought of as any software component that is used to add functionality to another application. [...] We will refer to a "document"(Word document, Excel workbook, Access database, etc.) within the

> target application as a client for the add-in. [...] This general definition would even include a dynamic link library (DLL), whose exported functions are called from the target application. However, the term add-in generally applies to a more restricted type of software component, that is, an ActiveX server component (DLL or EXE) that is designed specifically to provide additional functionality to a particular type of application, such as one of the major Office applications."

However the presented add-ins in figure 1.5 are not COM servers in the just mentioned fashion, they are rather special Excel workbooks. What makes them special despite the file extension, `.xla` instead of `.xls`, is the way Excel (and the other Office products) handle such add-ins. Connectivity to the hosting application is achieved via the *Add-in Manager* within Excel. And once activated an Excel add-in maintains this state even when Excel is shutdown and restarted in the meanwhile.

1.3 COM Add-ins

The COM add-in model was introduced with the advent of Microsoft Office 2000.

Again, speaking with [40]:

> "[...] a COM add-in is an ActiveX server component (DLL or EXE) that implements a specific COM interface called *IDTExtensibility*. [...] An interface is just a collection of functions that are designed for a specific purpose. [...] The main purpose of the IDTExtensibility interface is simply to provide [...] feature accessibility (to the user) and access to the client's object model."

The *IDTExtensibility2* COM interface provides five events the add-in developer can utilize to manipulate her add-in and the hosting, i.e. calling application. The members are depicted in figure 1.6. The complete reference is available from [32].

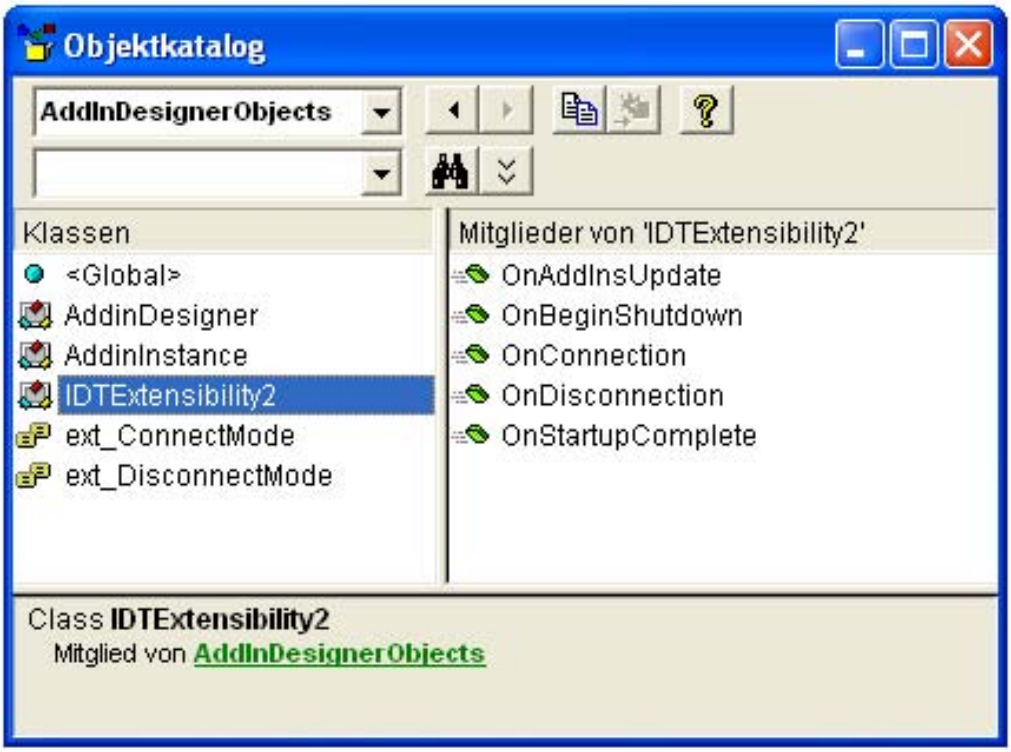


Figure 1.6: The *IDTExtensibility2* interface

The major improvement in contrast to former legacy add-ins is that a single COM add-in is callable from any application which supports COM add-ins. In other words if a developer wanted her add-in application to be callable across different Microsoft Office applications, she had to program every single application-specific portion in a programming language unique to the according Office application, see [37]. Admittedly a time consuming and tedious task.

Instead now the only thing to do is to add an application specific version of the *IDTExtensibility2* interface to the add-in. The benefits are obvious: the developer has to maintain only one source code base for a whole family of add-ins which can be used in different applications. I implemented a prototype version of the MD*ReX add-in for Microsoft Word, see figure 1.7, many

more e.g. for the database application Access or the presentation software Powerpoint could be added.

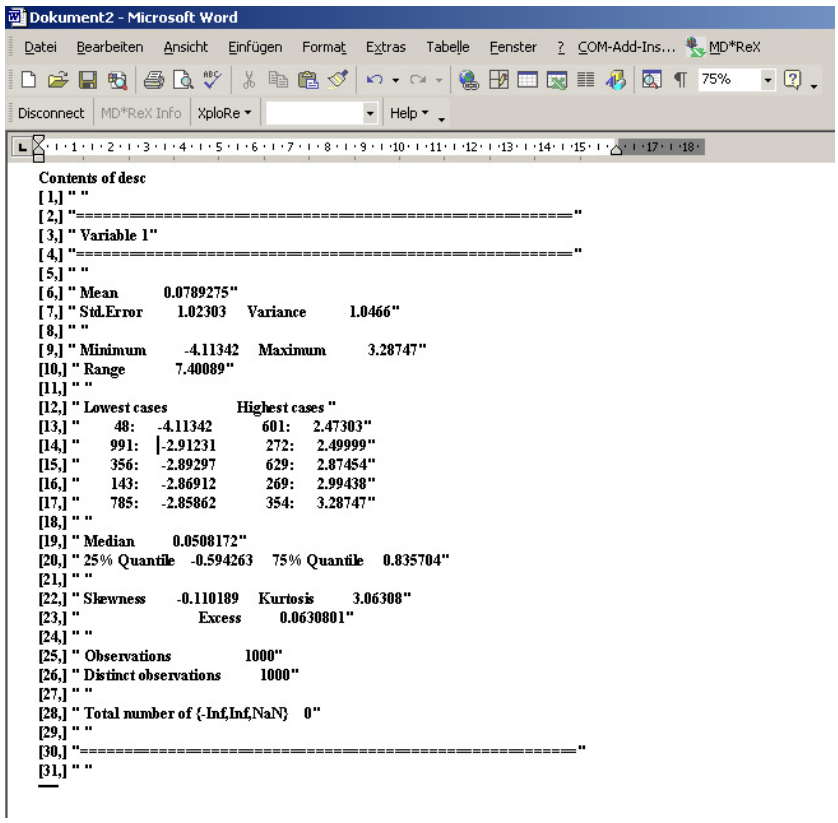


Figure 1.7: MD*ReX within Microsoft Word

Thus on a high-level view the COM technology and the provided *IDTExtensibility2* interface allow developers to rapidly implement add-in versions with suitable functionality within a variety of COM enabled host applications, a majority on the Windows platform.

A huge improvement for statisticians who want to provide methods to a broad audience of users of e.g. standard office applications, especially spreadsheets.

Chapter 2

Client / Server based Statistical Computing

The potential applicability of statistical software is growing steadily. On the one hand we have efficient and cheap data collection methods available. Relational database applications and modern data analysis paradigms as e.g. data mining, see [25], provide statisticians with very large data sets. Computational statistics can hence be more generally described as *computing with data*, see e.g. [9].

On the other hand the Internet overwhelmingly influenced our perception of distributed computational statistics. Reading and writing data from network applications or code execution over the network are of utmost importance in the pursuit for *distributed computing with data*, see [8].

And of course technical progress resulting in hardware which is getting relatively cheaper and relatively more powerful with each product cycle.

Those factors govern, among others mentioned in the beginning, the design paradigms of statistical software. For educational purposes other issues might be of concern too, see [4].

2.1 XploRe

`XploRe` is a high level object-oriented programming language, i.e. the user writes procedures or functions, such as in Pascal or C/C++. In contrast to these languages the declaration of variables is not necessary in order to preserve the character of an interpreter. Furthermore, variables can be collected in list structures, so that it is possible to hold common information of a data set in a single data object.

Features of an high-level language like recursion, local variables, loops, and conditional execution are available. The building blocks of the `XploRe` language like language elements, data types, grammar and flow control are discussed in [26]. The first WWW and Java interfaces are also described there. [22] describes data structures (for among others graphical and data objects) and their implementations within `XploRe`.

Statistical methods (called *Quantlets* in `XploRe`) are provided as plain-text ASCII files and collected into libraries (*Quantlibs* in `XploRe` jargon), covering modern statistical methods for time series, panel data, neural networks and financial engineering, etc. Dynamic link calls are possible, so one can incorporate own methods in `XploRe`, written in the language of his or her choice. An automatic HTML converter ensures integration of *Quantlets* and *Quantlibs* into the help system.

A basic introduction into using XploRe is available in [18]. More refined methods are explained in [15]. Methods with a statistical finance view are discussed in [16].

2.2 XQS/XQC

The `XploRe Quantlet` client / server system of `XploRe` has been described in [21], [17] and to its full extent in [27]. The `MD*Crypt` protocol stack is illustrated in [12]. So I will not go into much detail on this architecture here.

Nevertheless it should be mentioned, that I had to apply some slight changes to the middleware application `MD*Serv` in order to make it a native Windows

executable (MDCOM.exe) instead of a Java executable archive (MDServ.jar) and to slightly improve performance while working with MD*ReX (figures 2.1 and B.2).

Furthermore [12] describes the relevant modification applied to the MD*Crypt protocol stack in order to make it implementable within a COM based environment, namely the MD*ReX client, and discusses performance issues of socket based communications.

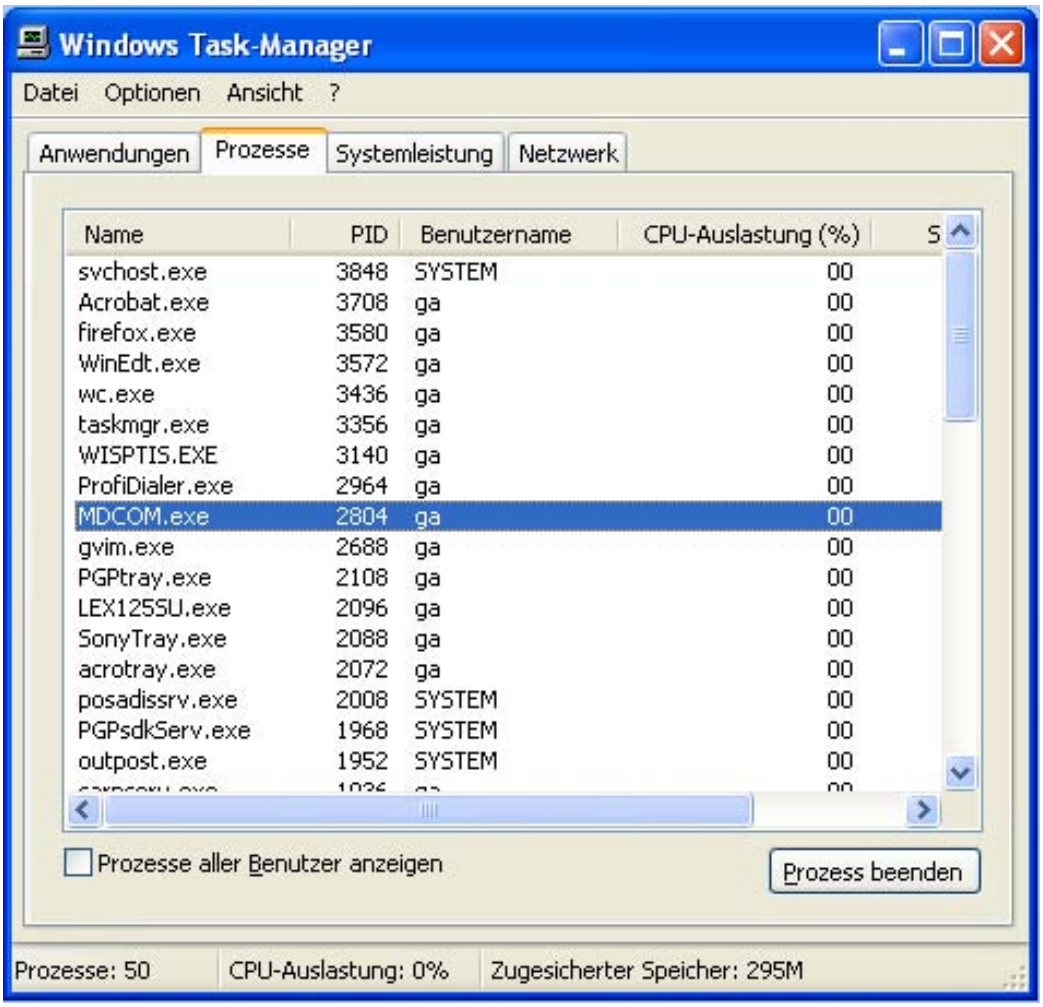


Figure 2.1: MD*Serv as native Win32 executable (MDCOM.exe)

Chapter 3

MD*ReX

MD*ReX is an add-in solution for client/server based statistical computing (compare sections 1.2 and 2). The MD*ReX client is hosted within a COM enabled application, in this particular case Microsoft's Excel spreadsheet. Its server counterpart is the `XploRe Quantlet Server` (XQS). The communication broker between both is the middleware solution `MD*Serv`. The language in which the communication is realized is the **TCP/IP** based `MD*Crypt` protocol.

3.1 Evolution of the Excel Client

The story of MD*ReX begun with a prototype add-in version developed by Erich Neuwirth in 1999/2000, using raw Windows Socket connections to talk to the `XploRe` server application.

The next incremental step was then introduced by [1], implementing a custom COM enabled `MD*Crypt` version developed in Microsoft Visual Java.

Both versions were pure Excel add-ins, i.e. implemented as special Excel workbooks with `.xla` extension, as explained in section 1.2. The development environment for both has been the integrated development environment (IDE) of Microsoft Office for its own macro language VBA.

With the progressing spread of the Microsoft Office 2000 package I seized to support pure Excel add-ins and thus stopped developing with VBA. This conversely had the effect that I also had to stop the support for Excel versions prior to Excel 2000, which has the internal version number 9. Thus Excel 95 and Excel 97 were cut-off from the potential user list. As of this writing MD*ReX has full support only for Excel 2003 (version 11). I assume this loss was outweighed by the increased flexibility gained through the COM add-in technology as described in section 1.3.

3.2 The MD*ReX Architecture

With the implementation of the *IDTExtensibility2* COM interface I started to develop MD*ReX in Visual Basic 6 (**VB**). This had first of all the advantage of using a full featured language for Windows application development rather than a macro language for Office development and secondly the positive side effect that I now could compile the add-in to a DLL, promising slightly higher performance, instead of distributing a blown-up Excel workbook.

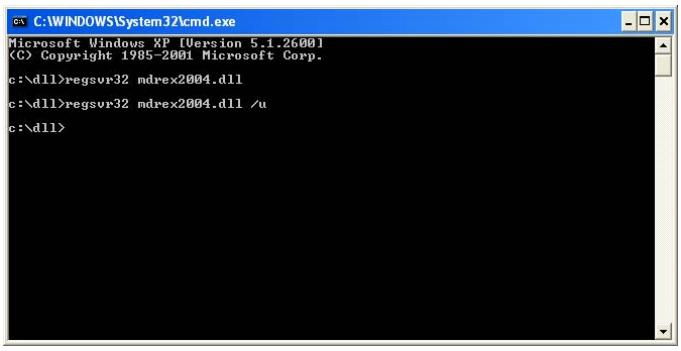


Figure 3.1: Command line (un)registration of MD*ReX

In technical terms MD*ReX is an In-Process COM server, [39], which is linked against the regarding Office and Excel object libraries and which references the mentioned `MD*Crypt` DLL.

To be able to work with the COM add-in, MD*ReX has to be introduced to the user's operating system. This is achieved via registering the DLL in

Windows' application repository also known as the registry. This is automatically done during setup. But can also be accomplished manually via the Windows command line utility `regsvr32.exe` (figure 3.1).

If the switch `/u` is supplied then the utility will unregister (figure 3.3) the DLL from the system otherwise it will register the DLL, (figure 3.2).

Figure 3.2: MD*ReX registered

Figure 3.3: MD*ReX unregistered

Of course the described command line registration only *installs* the COM DLL. The middleware applications, the `XQS`, and any other supplied files are unaffected by this.

The registration is needed for the COM runtime to find the specific programmatic identifier of the COM application and to load the appropriate DLL or EXE. Once registered the add-in is started as soon as Excel is triggered by the user. This is the default behavior of the client. The *IDTExtensibility2* interface allows for other start-up modes like load on demand or load once. But in my opinion we do not need to take care of these options, since the user can control whether the COM add-in is loaded or unloaded via a menu bar entry in the regarding Office application.

This control is somewhat hidden, but via right-clicking in the main menu

Figure 3.4: COM Add-in dialogue in Office

bar of the Office application and choosing customize, a window opens which allows for various customizations of the Office menu.

Under the *Extras* entry, a command button called *"COM Add-ins"* can be found (figure 3.4). Dragging and dropping this somewhere on the menu bar gives the user full control of every Office COM add-in on the System.

In the current version MD*ReX consists of a total of five classes, six modules and 12 GUI elements (figure B.1). The functionality of the client is discussed in the following section.

An important class is `clsMDCrpt.cls` which implements the `MD*Crypt` protocol. This class is designed to encapsulate the communication details of the client and could also be used by other **VB** developers. This class currently consists of 17 properties, ten public members and 14 functions.

Another important class is `clsMDCOMMenu.cls`, which controls the main user interface of the add-in. Both classes are generic and can be used for creating COM add-ins for various Office applications. The complete source listing of the client can be found in the appendix of this thesis.

3.2.1 Design Issues for Add-in based Solutions

As outlined before, Excel might be a suitable tool for data storage, manipulation and analysis. A success factor for a statistical add-in is the functionality it offers and the design of its user interface. MD*ReX has been designed to seamless fit into the well known user interface of the hosting Office application without burdening the user to learn a complex and overloaded GUI.

Another design issue was to address a broad audience of users. As pointed out in [3] and [4] we can at least identify three potential user categories: method developers, advanced consumers of methods, and naïve users. Another classification might be teachers, graduate, and undergraduate students.

These user groups have different claims while using a statistical software. Accordingly the environment should account for these profiles. A method developer needing direct access to the statistical engine. In MD*ReX this is achieved through a command line utility in the toolbar (figure 3.13). The sophisticated methods user seeking for a macro editor to develop his own functions. In MD*ReX this is the *XploRe Direct* utility (figure 3.15). And of course the naïve user who is accustomed to or needs a menu driven interface with dialogues and menu options. These can be accomplished via custom add-in workbooks also known as `.xla` add-ins (figure 3.21).

3.2.2 Customizing the Add-in Environment

This last feature is extremely important if one needs to customize the COM add-in environment itself. For the Excel client I decided to provide this customization ability via user provided workbooks and a well-defined entry point for custom VBA macros to be inline with the Excel object and user model. The directory structure of the client is as follows:

```
/root-directory
                /mdserv
                /mdrexxla
                /debug
```

where the root directory contains the MD*ReX COM DLL itself and one default Excel `.xla` add-in. This default add-in, called *Xploregetresult.xla*,

is automatically loaded when the client connects to a server and serves as a generic excel worksheet function. It contains the VBA function *XPLEval* which is a reference implementation of Excel worksheet functions and can be seen in action in figure 3.19. The source is given below.

```
Option Explicit

Public Function XPLEval(XPLExpression As Variant, ParamArray XploReArgs() As Variant) As Variant
    Dim i%
    Dim oAdd As Object
    Dim tArgs As Variant

    If UBound(XploReArgs) >= 0 Then
        For i = 0 To UBound(XploReArgs)
            tArgs = XploReArgs(i)
        Next i
    End If
    On Error GoTo Catch_Err
    Set oAdd = Application.COMAddIns.Item("mdrex2004.dsrExcel11").Object
    XPLEval = oAdd.XPLEval(XPLExpression, tArgs)
    Exit Function

Catch_Err:
    XPLEval = "#XPLError"
End Function

Sub AddDescription(Name, DescText, Optional Category = 4)
    Application.MacroOptions Macro:=Name, Description:=DescText End
Sub

Private Sub Workbook_Open()
    Call AddDescription(XPLEval, "Evaluates XploRe Quantlets")
End Sub
```

The *mdserv* directory contains the complete middleware and the `XQS` along with all needed libraries, `Quantlets` and DLLs. The *debug* directory is the place where MD*ReX writes its log file. The directory where MD*ReX looks for user supplied add-ins is *mdrexxla*. By default this directory contains one workbook add-in called `oneVarSummary.xla`. The functionality of this add-in is described below.

As stated previously the Office application functionality can be enhanced by problem specific add-in packages. In this setup of spreadsheet and statistical engine we differentiate two user interaction models: **in-sheet** functions (also referred to as worksheet functions) and customizations via **menu based** functionality. We will try to make this approaches clear considering the examples in the following section.

3.3 How to work with MD*ReX

Before explaining how MD*ReX can be used for statistical data analysis, some remarks on the variable size restrictions might be sensible. MD*ReX can handle as much data as Excel can, and this is in its most current version up to 255 variables (in columns), containing up to 65,535 cases per variable (in rows) on workbook level. The amount of workbooks Excel can handle is theoretically only limited to the amount of memory on the PC.

After installing MD*ReX, which is usually done via the provided setup application (but which can also be done manually as described in 3.2) the user can start working with the add-in simply by starting the hosting office application. In this examples the caller is always Excel and the callee is MD*ReX for Excel.

Figure 3.5 shows Excel right after the user started it. As can be seen, the user is confronted with the usual Excel GUI. Despite the fact that an additional menu item appears on the top menu bar of Excel, also called *Worksheet Menu Bar*.

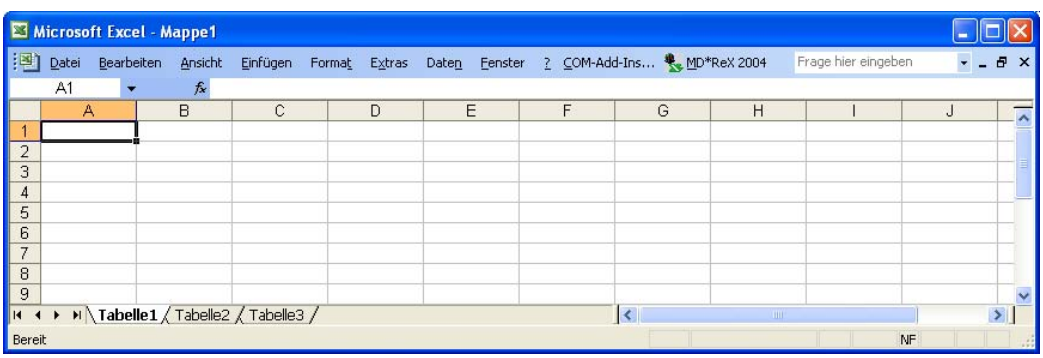


Figure 3.5: Excel's start up view

Clicking the new menu item *MD*ReX 2004* some background activity is recognizable which results in a splash screen with some information regarding the MD*ReX COM add-in and a new menu bar appearing on the *Worksheet Menu Bar*. Figure 3.6 shows this.

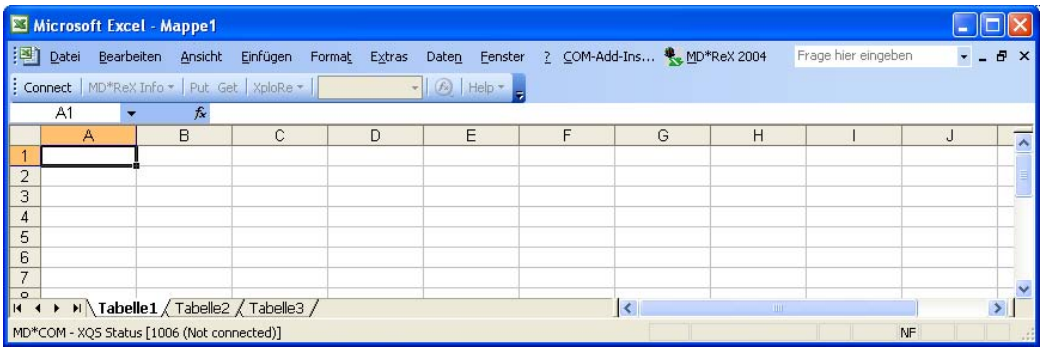


Figure 3.6: MD*ReX after initialization.

A status message is displayed to the user on the default status bar location in the lower part of Excel's window. As can be seen the new menu bar item consists of a couple of buttons and other items which are not accessible to the user except for one item labeled *Connect*. Clicking on the only enabled *Connect* button brings up a small window where the user has the opportunity to select an `XQS` from a dropdown list. The choices are either a local connection or a remote connection to other `XQS`.

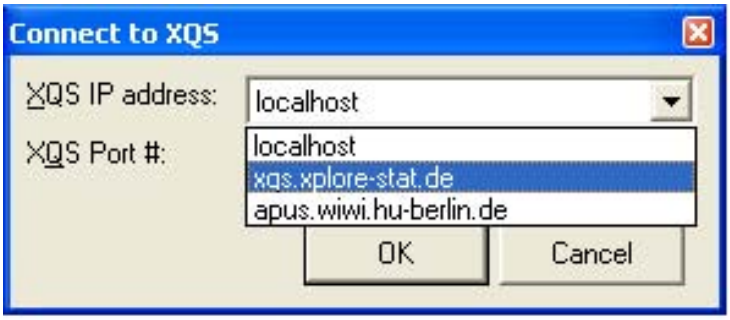


Figure 3.7: Connect dialogue

While this is happening MD*ReX starts the middleware application MDCOM.exe (formerly known as MDServ.jar) in the background. This process is invisible to the user and can only be seen if the Windows process manager is started as shown in figure 2.1. If the user now selects a local connection the background process MDCOM.exe, which is listening to incoming connection

requests, triggers a new instance of the `XQS`. This process is also hidden from the user and can be watched either in the process manager or in the Excel status bar. Evidently the user will remark a difference on the new menu bar, since the previously disabled items are now activated and can be used, see figure 3.8.

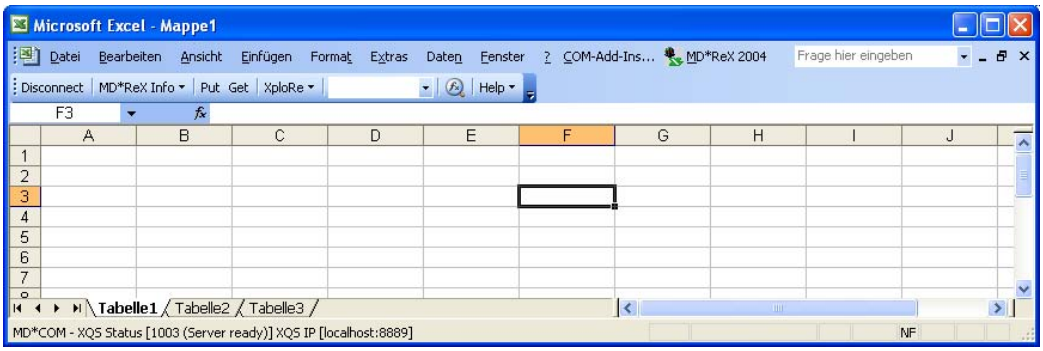


Figure 3.8: MD*ReX after connection

Having gone so far the client is now ready to start some statistical analysis. As a simple example I will show some techniques to work with MD*ReX. A basic requirement is naturally to work with data which is already contained within an Excel spreadsheet. For illustration figure 3.9 shows a workbook which contains some time series of the MSCI World stock index (Morgan Stanley Capital International) obtained from Datastream.

As can be seen from figure 3.9, standard Excel interface features like a context menu which opens up when right-clicking with the mouse anywhere in the currently open workbook, can provide an intuitive way for the user to communicate with a statistical backend software. In our implementation MD*ReX offers a *put* and *get* method to post and retrieve data to and from the `XQS`. The necessary action is now to mark an appropriate range of worksheet cells which contain the desired data and activate the *put* method. This can be done either by the context menu or the *Put*-button in the menu bar. Both are consistent with the Excel user interaction model.

MD*ReX' *put* method utilizes another window based control to facilitate the data exchange. Both the context menu entry as well as the menu bar entry

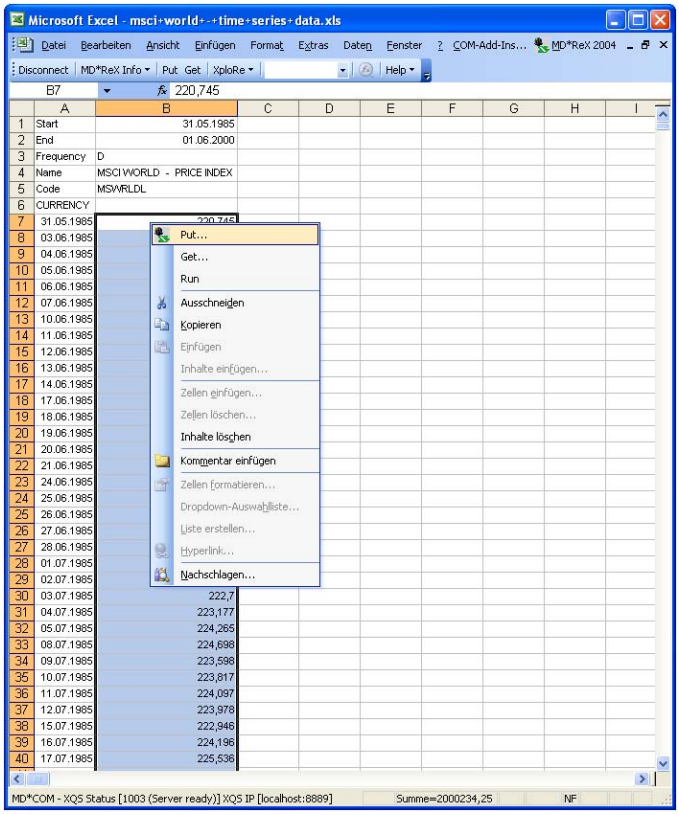


Figure 3.9: Excel workbook with time series data and MD*ReX context menu entries

open up a window asking the user to provide an appropriate name for the data object to be sent to the `XQS`, see figure 3.10. After the user has given the object a name, MD*ReX uploads the data onto the `XQS`. While doing this it stores the name the user supplied and creates a mapping table which contains all objects the user sent or retrieved from the `XQS`. This mapping table furthermore creates so called *named ranges* which facilitate the use of worksheet functions within Excel.

The mapping is crucial for obtaining an overview of data objects and for facilitating the navigation within the client environment. The mapping consists of the name supplied by the user (this is the first entry in the line and also corresponds to the Excel named range), the worksheet name the data

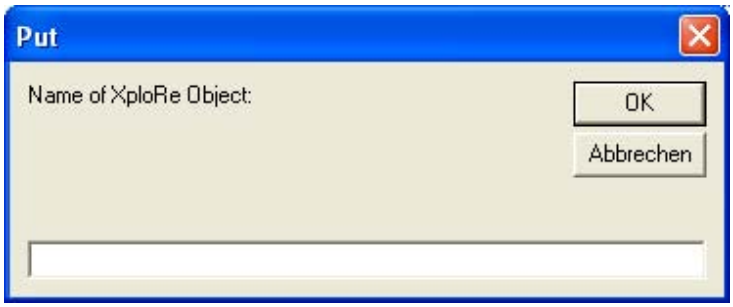


Figure 3.10: MD*ReX Put dialogue

object resides in, the exact address of the object in standard Excel notation and finally the name of the data object on the server marked with the string *XPLORE:* followed by its name. The mapping window is accessible via the menu tool bar as seen in figure 3.12, the mapping table itself is depicted in figure 3.11

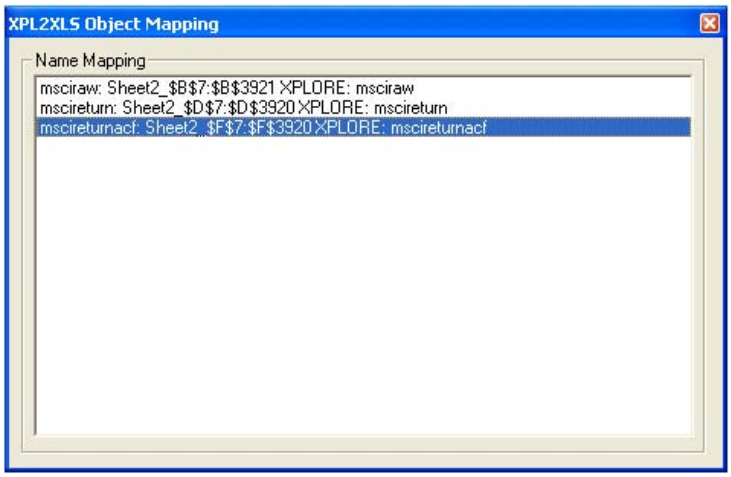


Figure 3.11: MD*ReX mapped object table

As stated above, addressing various user needs to interact with a statistical environment, I implemented two GUI elements to account for different needs. The first one is a command line menu element which allows the user to directly issue `XploRe` commands. Each command is evaluated instantaneously.

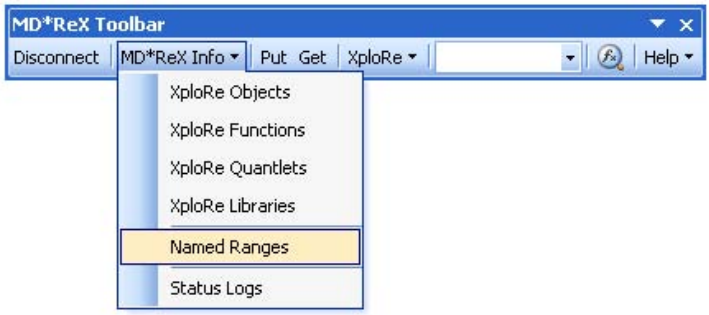


Figure 3.12: MD*ReX *Named Ranges* dialogue

If the evaluated command has a return value it will be displayed in an output window, similar to 3.14 otherwise the command is silently processed. The command line menu item also provides a command history for convenience.

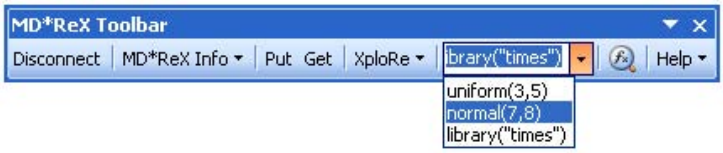


Figure 3.13: MD*ReX command line interface

The other GUI element is an editor with built in result window. Both items are necessary for providing a flexible and user friendly environment and are standard in the `XploRe` GUI version and hence are expected by users of the add-in version.

The editor can be reached via the XploRe Direct menu entry as shown in figure 3.16.

The editor can be used to run custom or existing *Quantlets*. In our running example we first load some libraries and then calculate log-returns and the autocorrelation of the index series. Finally we print the according values in the result window of the editor as depicted in figure 3.15.

But having results only in an editor window is not satisfying. We want the resulting data objects back in our Excel workbook. As mentioned MD*ReX

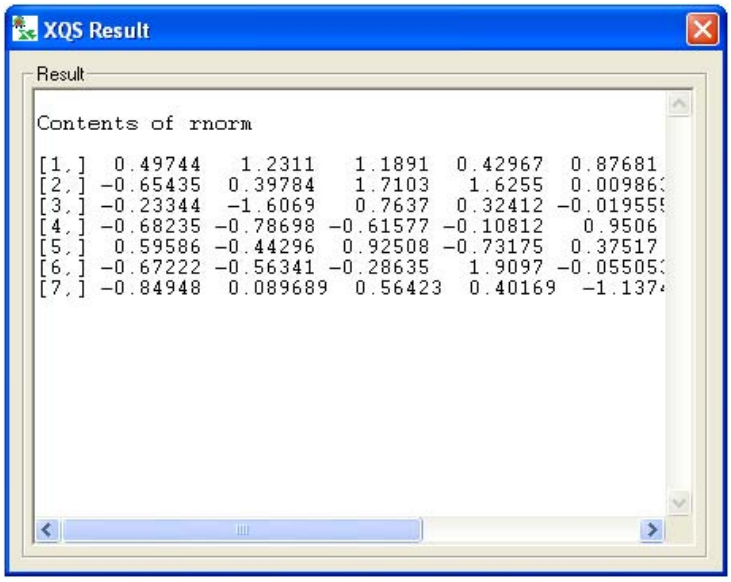


Figure 3.14: MD*ReX result window with evaluated command

provides a *get* method. In analogy to the *put* method this will get any existing `XploRe` data object into Excel. The name mapping is also done in this direction. After clicking *back* in the context menu or the menu bar, the according object is written into the cell range of the current mouse position. To be more precise the put and get methods read or write into the cell range with the current Excel focus. Hence a picture similar to figure 3.17 might be obtained. Note the open drop down field showing current Excel named ranges.

We mentioned two user interaction models: **in-sheet** functions and **menu based** functionality. Figure 3.18 shows the first kind: in-sheet functions. Via the *XplEval* worksheet function we are able to evaluate `XploRe` commands as if they were standard Excel functions, see also figure 3.19.

What remains is the customization respectively expansion of the client environment via user supplied *.xla* workbook add-ins. A prototype workbook add-in which makes use of a **menu-based** functionality is supplied during setup of the COM add-in. What this add-in does, is simply wrapping the `XploRe` *descriptive()* command into a GUI based Excel function. To acti-

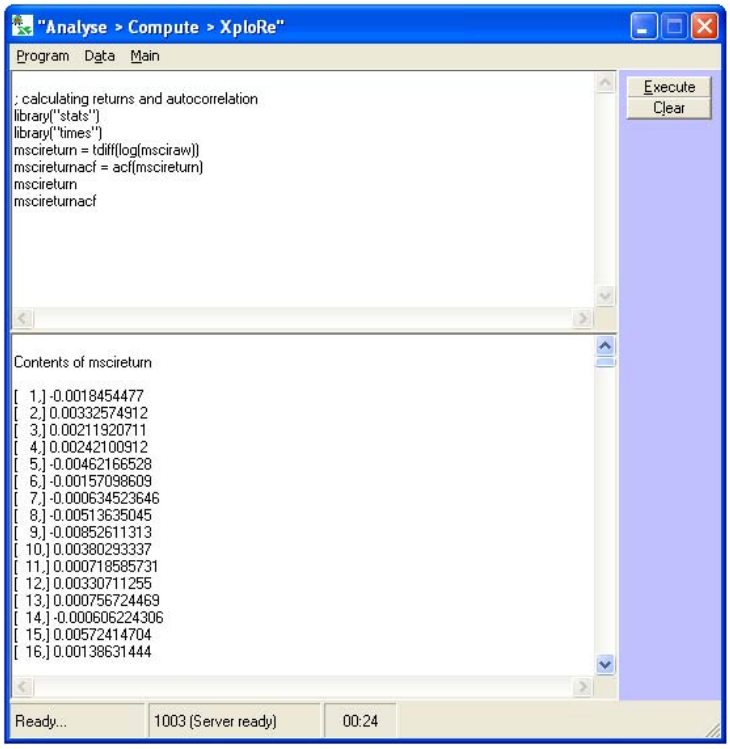


Figure 3.15: *XploRe Direct* editor

vate this add-in, the menu item *MD*ReX Add-Ins*, see figure 3.16, has to be clicked. This instructs MD*ReX to scan the *mdrexxla* directory for workbook add-ins. If according add-ins were found they are displayed in the *MD*Rex Excel Add-Ins* window as shown in 3.20.

The user now can select the desired add-in which is loaded into Excel. This kind of add-ins can contain GUI elements but do not have to. The supplied example add-in contains a small menu bar which opens another control when clicked. With the so called Excel *Refedit* control the user can select arbitrary Excel ranges, see figure 3.21.

After selecting a range the add-in asks for a **XQS** object name and then sends the data to the server. The sent range is evaluated and in this case the XploRe command *descriptive()*, which returns some descriptive statistics, is executed on that range. The result of this command is then sent back to the

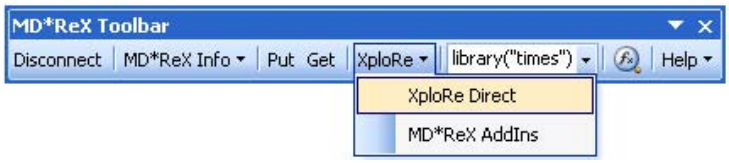


Figure 3.16: MD*ReX *XploRe Direct* menu entry

MD*ReX result window as shown in figure 3.22.

When the user is finished with the analysis and wants to quit working with MD*ReX, the recommend procedure is to click on *Disconnect*. This terminates the current session with the `XQS` and unloads any custom workbook add-in. If one wants to get rid of the additional menu bar a click on the *MD*ReX 2004* icon in the Worksheet menu bar is sufficient to close it. This also unloads default add-ins. To completely unload the COM add-in itself either Excel's *COM Add-ins* menu can be used or the DLL can be unregistered via the

```
regsrv32 mdrex2004.dll /u
```

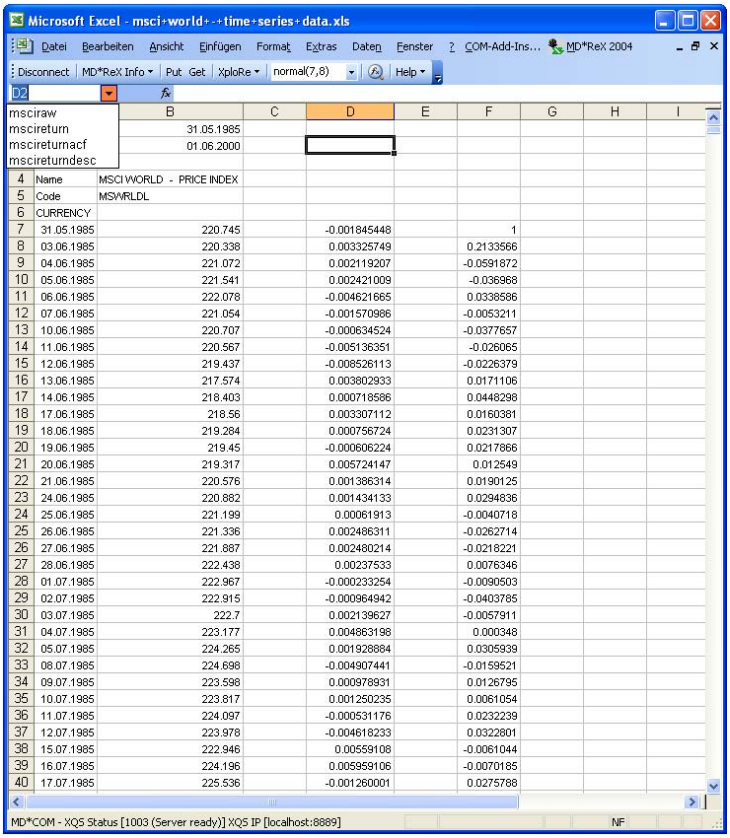
command issued at the windows command prompt. This works only in the directory where the COM add-in is installed, by default this resembles to:

```
%program_files_folder%\MDTECH\MDREX\
```

3.4 Future Work

3.4.1 Graphics

With the presented features MD*ReX has quite a lot to offer for statistical data analysis. The immediate execution and representation of any `XploRe` method which does not reply on graphical output is an advantage appreciated in a spreadsheet environment. Its ability to connect to local `XQS` instances as



7	31.05.1985	220.745	-0.001845448	1
8	03.06.1985	220.338	0.003325749	0.2133566
9	04.06.1985	221.072	0.002119207	-0.0591872
10	05.06.1985	221.541	0.002421009	-0.036968
11	06.06.1985	222.078	-0.004621665	0.0338586
12	07.06.1985	221.054	-0.001570986	-0.0053211
13	10.06.1985	220.707	-0.000634524	-0.0377657
14	11.06.1985	220.567	-0.005136351	-0.026065
15	12.06.1985	219.437	-0.008526113	-0.0226379
16	13.06.1985	217.574	0.003802933	0.0171106
17	14.06.1985	218.403	0.000718586	0.0448298
18	17.06.1985	218.56	0.003307112	0.0160381
19	18.06.1985	219.284	0.000756724	0.0231307
20	19.06.1985	219.45	-0.000606224	0.0217866
21	20.06.1985	219.317	0.005724147	0.012549
22	21.06.1985	220.576	0.001386314	0.0190125
23	24.06.1985	220.882	0.001434133	0.0294836
24	25.06.1985	221.199	0.00061913	-0.0040718
25	26.06.1985	221.336	0.002486311	-0.0262714
26	27.06.1985	221.887	0.002480214	-0.0218221
27	28.06.1985	222.438	0.00237533	0.0076346
28	01.07.1985	222.967	-0.000233254	-0.0090503
29	02.07.1985	222.915	-0.000964942	-0.0403785
30	03.07.1985	222.7	0.002139627	-0.0057911
31	04.07.1985	223.177	0.004863198	0.000348
32	05.07.1985	224.265	0.001928884	0.0305939
33	08.07.1985	224.698	-0.004907441	-0.0159521
34	09.07.1985	223.598	0.000978931	0.0126795
35	10.07.1985	223.817	0.001250235	0.0061054
36	11.07.1985	224.097	-0.000531176	0.0232239
37	12.07.1985	223.978	-0.004618233	0.0322801
38	15.07.1985	222.946	0.00559108	-0.0061044
39	16.07.1985	224.196	0.005959106	-0.0070185
40	17.07.1985	225.536	-0.001260001	0.0275788

Figure 3.17: MD*ReX after receiving data from `XQS`

well as those on remote machines is another one which offers flexibility when computing power is needed, especially in heterogeneous environments.

However there is still room for improvement. First of all MD*ReX does not exploit the graphical facilities provided by `MD*Crypt` but rather relies on a manual treatment of `XploRe` data objects.

Hence the next incremental step would be to render `XploRe` graphical objects into Excel chart objects. The interactivity features of `XploRe` graphics are well suited to be mapped into interactive Excel charts, maybe via employing additional slider objects within Excel. An alternative approach could be achieved by creating a graphics device as ActiveX object in Visual Basic,

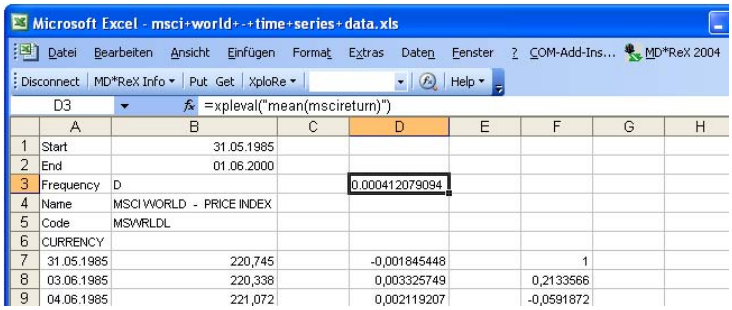


Figure 3.18: MD*ReX worksheet function evaluating the mean of the series

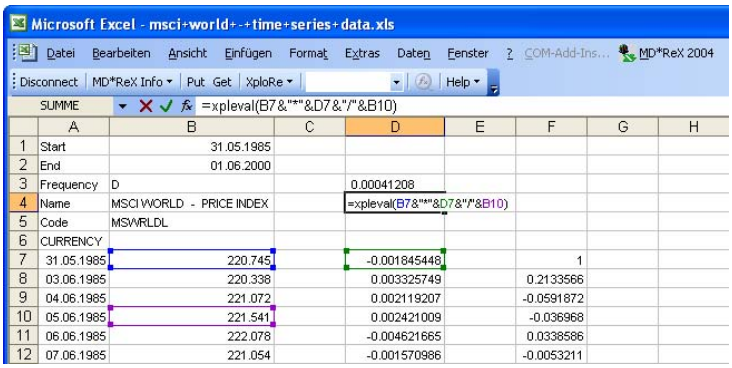


Figure 3.19: *XPLEval* worksheet function

Visual C++ or Visual Java and to display `XploRe` graphics in this device, when MD*ReX encounters an `XploRe` graphics object.

Nevertheless I would favor the former approach since this would represent a solution which is more compatible with Excel's object world and would not disturb the Excel user experience with an additional graphics component.

3.4.2 User Customization

The outlined proposal to account for user added methods via Excel workbook add-ins is in a preliminary stage yet. A more sophisticated handling of this feature is desirable. Therefore the published application programming

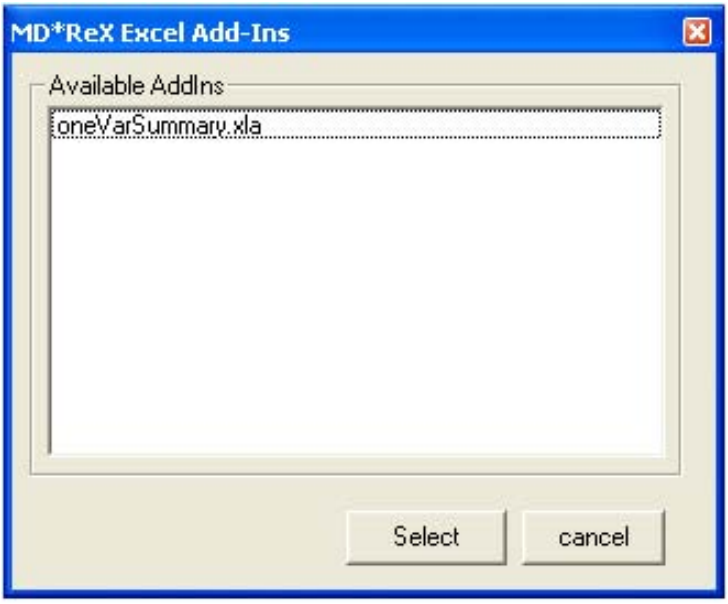


Figure 3.20: MD*ReX worksheet

interface (API) of MD*ReX needs further refinement, in order to allow the user to pass arbitrary (user defined) data objects to MD*ReX for further evaluation by the `XploRe` runtime environment. The current entry point is quite restrictive and only allows for passing Excel range objects and character strings.

3.4.3 Performance

The major drawback in socket based communication is a latency in sending requests and receiving corresponding answers. While the `MD*Crypt`/`MD*Serv` middleware approach is well dimensioned for light-weight applications like browser based applet or e-book solutions a local communication via a socket based unstructured byte stream is a restraint in circumstances where very fast response cycles are required.

Unfortunately this holds true for the recalculation paradigm of spreadsheet applications. Every time when Excel enters a recalculation cycle, e.g. because

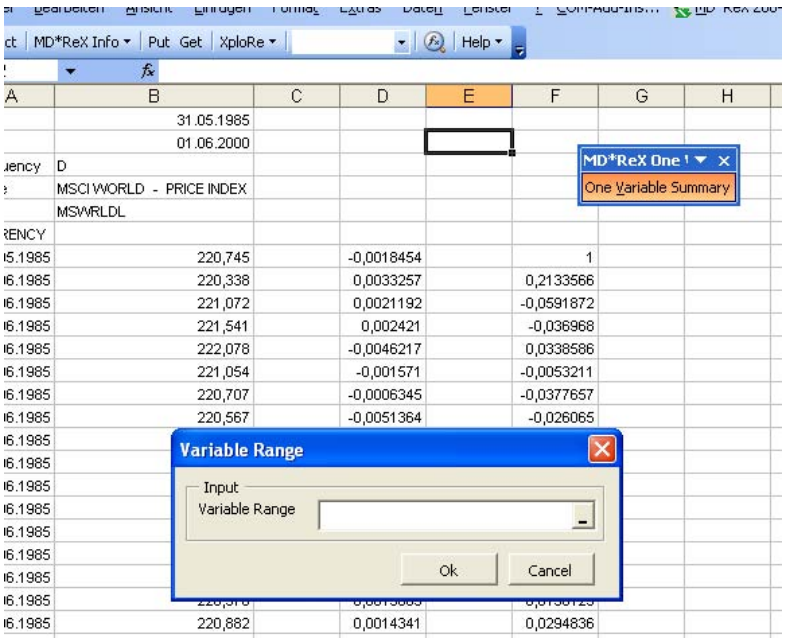


Figure 3.21: Example of a custom add-in

the user changed a cell value which was calculated by a worksheet function, the application has to wait until the middleware sends and receives data between client and server. For menu based applications, which as shown might also be realized in spreadsheets, this is not much of a problem. But things can get really slow, if the user makes use of worksheet functions. Since for every cell this extra middleware cycle has to be undergone.

3.4.4 Outlook

The statistical package `XploRe` emphasized a flexible, transparent, and interactive environment in its earliest versions. It was clear that distributed working in heterogeneous environments was crucial for scientific research and education in statistics.

The client/server concept which was then derived from this findings, still underline the topicality and necessity of such an approach. As discussed the

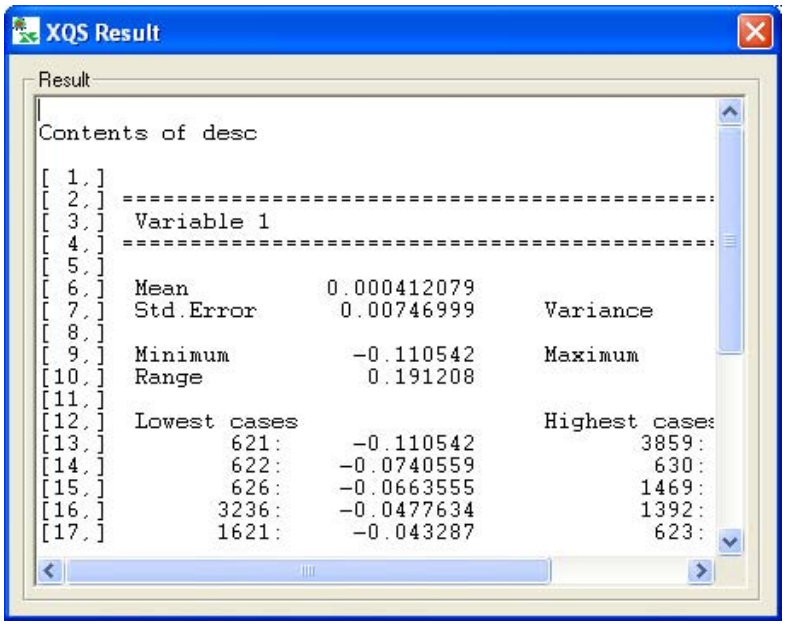


Figure 3.22: Result of a custom add-in

current Java based middleware architecture is formidable in applications like browser based services or e-books. In volume and time critical circumstances however the communication overhead can create a bottleneck. To sum up we can identify following concerns:

The communication is low-level: a special purpose format is used for interchange and hence there is hardly potential for reusability of the communication structure and only little support for universal data formats. It follows that the developer of an client application has to have enough knowledge of the underlying system, i.e. `XploRe` **and** its communication structure to start an analysis and evaluate the result. Low level communication details, like restart of an calculation, identification of communication errors, assurance of the transmitted data are cumbersome details a developer has to take care of rather on concentrating on the big picture of his application. A possible remedy might be to shift communication from sockets to a binary level e.g. by exploiting a component interaction architecture like COM, at least in local environments.

Especially within the MD*ReX project these problems are of concern, though there is a high demand for such an spreadsheet based application as shown e.g. in [44] and suggested by the download figures among the various clients the `XploRe` project has to offer.

A major concern in application development is reusability, customization, and rapid application composition out of pre fabric components, see e.g. [14]. In order to be able to compose applications using components they should meet the following requirements, see e.g. [39]:

dynamic linking

implementation encapsulation

language independence

A component technology like COM is aimed towards standardization of how components expose their functionality using interfaces. Software component architectures break the existing barriers between different programs by defining a framework where different components can interact which each other in a seamless manner, see e.g. [5]

Thus one aim of a component oriented approach can be identified as to be the seamless integration of the statistical environment in whichsoever component environment and ideally a complete encapsulation of the communication and, if feasible, of those provided statistical methods.

This consequently would allow users and developers to access statistical knowledge exposed by e.g. the `XploRe` system as if it were the environment (language, application, etc.) they are so familiar with, e.g Excel.

A bunch of alternatives (COM, CORBA, XML based Web Services) are available and the decision in favor of one or against the other has to be made in light of various influencing factors: the targeted audience, i.e. which platforms have to be supported, what kind of applications one wants to address and which development horizon is acceptable, just to name a few.

The newly started `Yxilon` project might help in fostering a strictly component oriented architecture. Then even the support of different component paradigms could be achieved, as a developer could exploit the paradigm which best suits her requirements.

3.5 Some Graphical Examples

With the described tool set and command functionality it is possible to conduct sophisticated statistical analyses with MD*ReX, like the following charting examples show. These are examples of existing *Quantlets* which are reproduced with MD*ReX and presented here as a closing feature of this thesis.

3.5.1 Implied Volatility

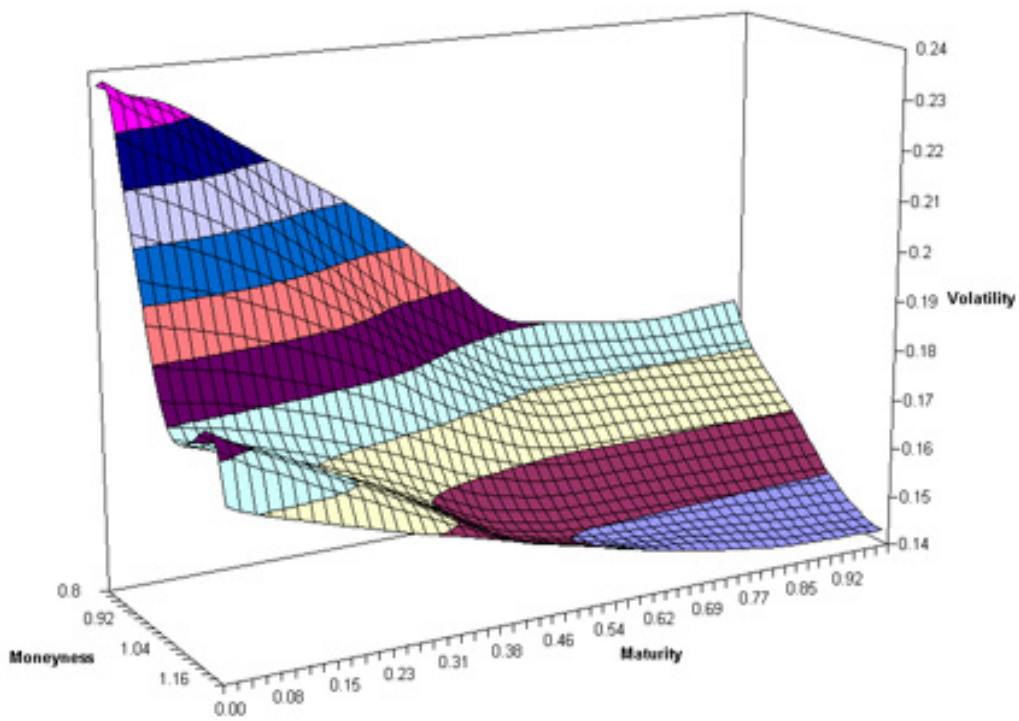


Figure 3.23: MD*ReX Implied Volatility Illustration

3.5.2 DAX30 Time Series Analysis

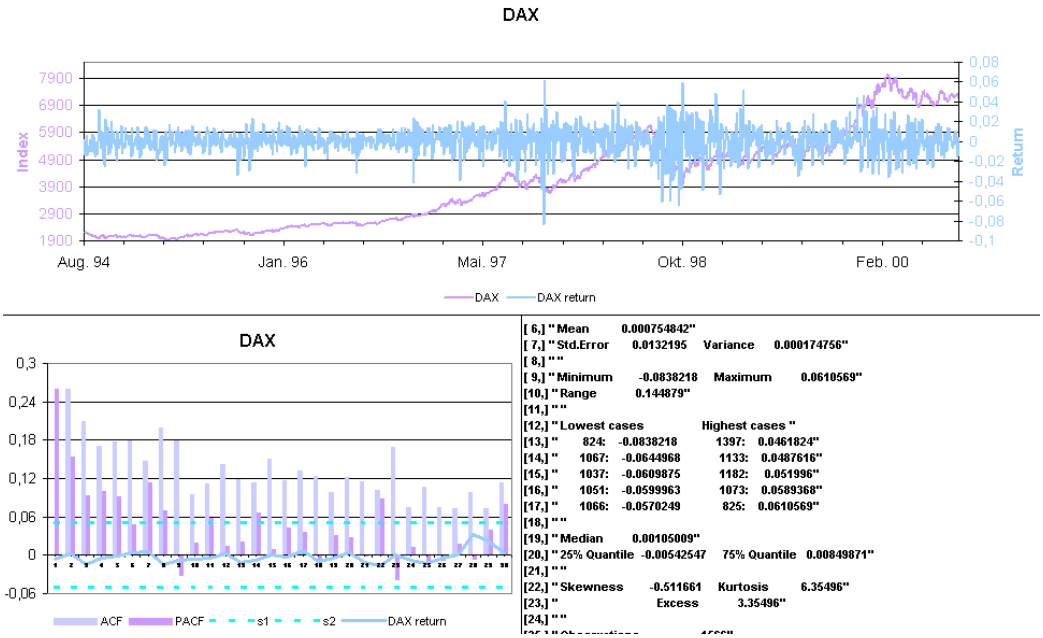


Figure 3.24: MD*ReX Time Series Analysis for DAX30

3.5.3 SARIMA Time Series Analysis

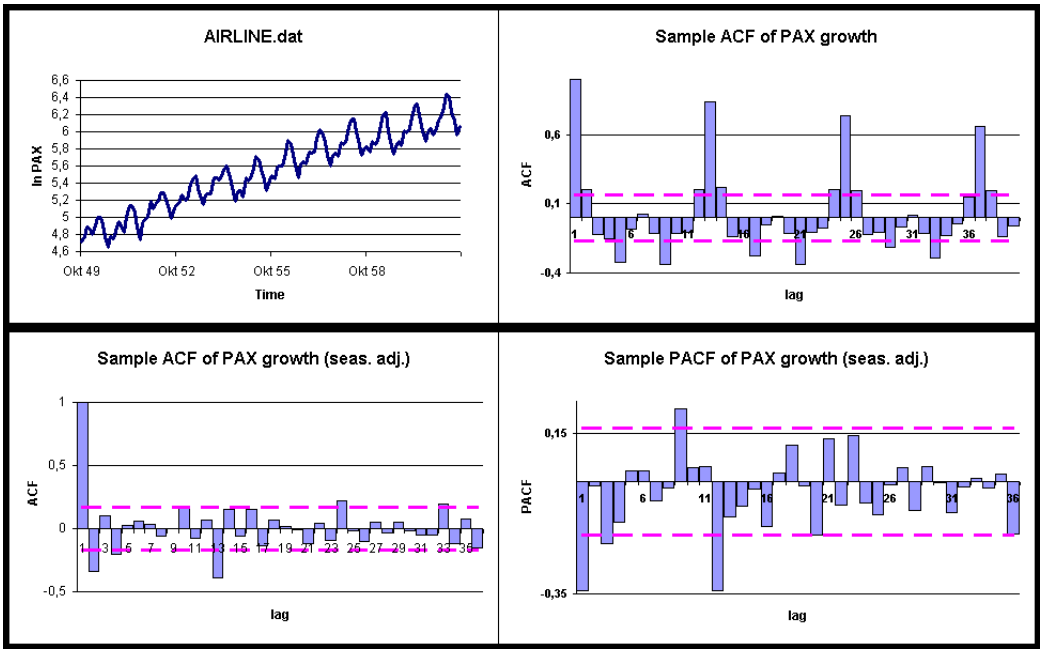


Figure 3.25: MD*ReX SARIMA Analysis for Airline Data

3.5.4 Spline Smoothing

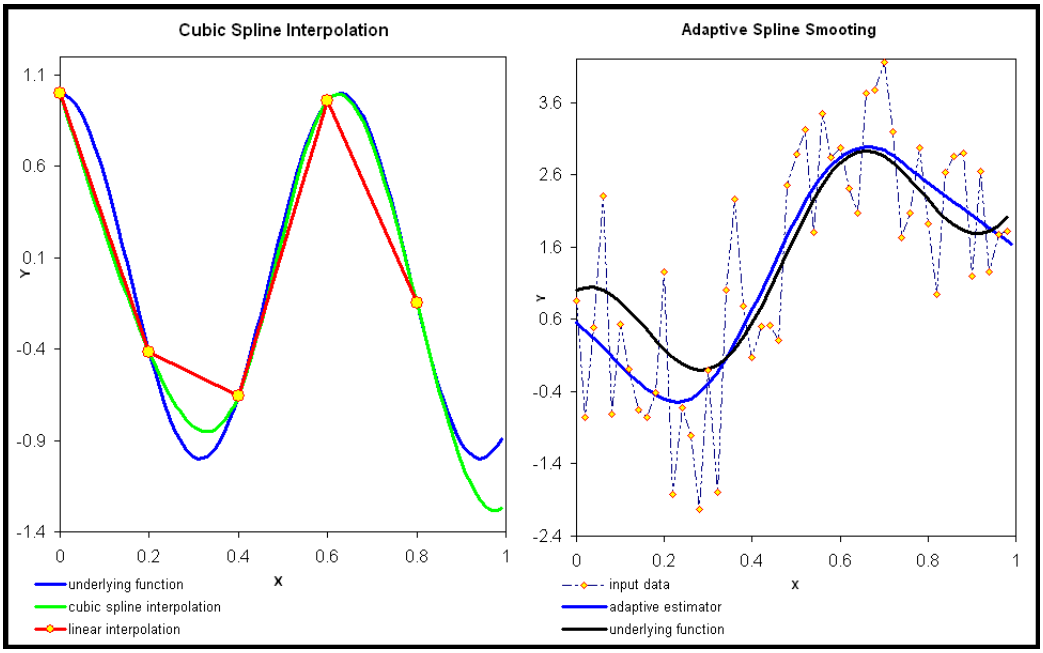


Figure 3.26: Cubic and Adaptive Spline smoothing

3.5.5 Kernel Regression

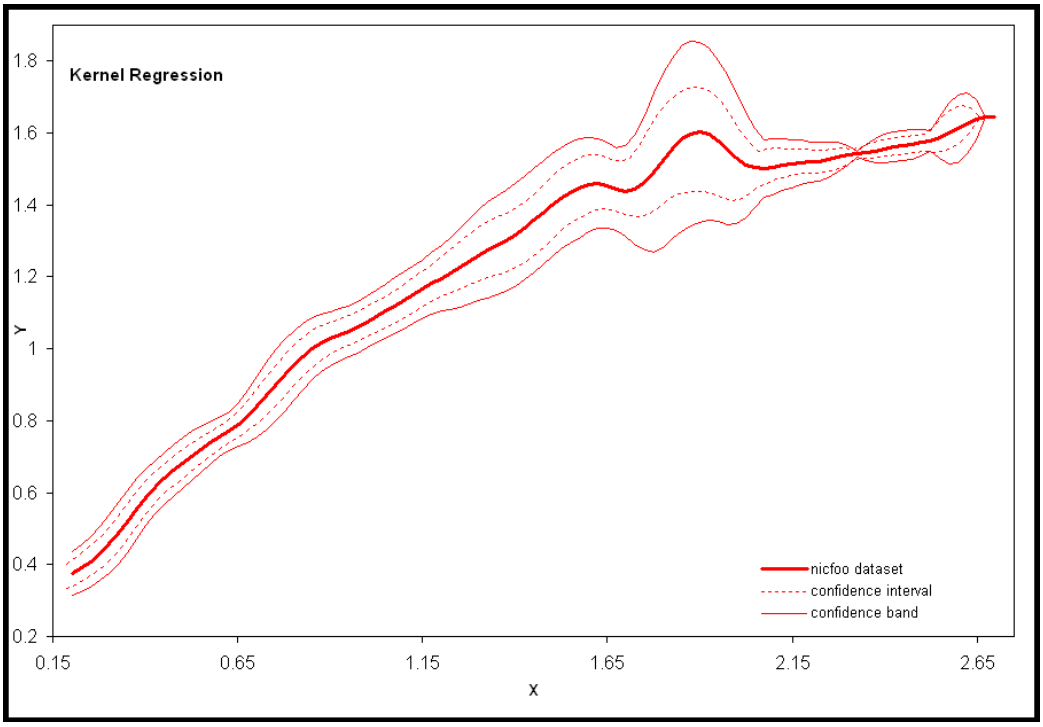


Figure 3.27: Kernel Regression

3.5.6 Kernel Densities

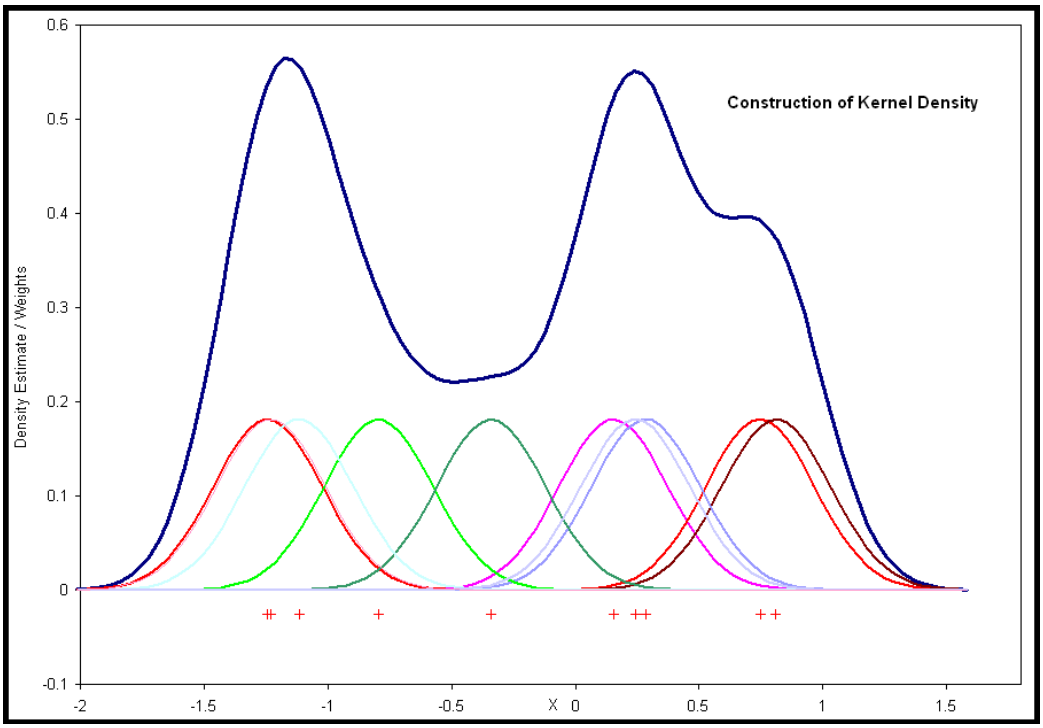


Figure 3.28: Construction of Kernel Densities

Bibliography

[1] G. Aydınlı, *Web Based Finance Tools, EGARCH and the ReX Client*, Master's thesis, Wirtschaftswissenschaftliche Fakultät, Humboldt-Universität zu Berlin, 2000.

[2] _____, *in Härdle, W. and Kleinow, T. and Stahl, G. (eds.): Applied Quantitative Finance, Theory and Computational Tools*, ch. Net Based Spreadsheets in Quantitative Finance, Springer-Verlag, 2002.

[3] G. Aydınlı, W. Härdle, and E. Neuwirth, *Efficient and Secure Statistics in Office Applications*, Proc. of the 35th Symposium on the Interface "Security and Infrastructure Protection", 2003.

[4] G. Aydınlı, W. Härdle, and B. Rönz, *E-Learning / E-Teaching of Statistics: A New Challange*, Proc. of the IASE satellite conference on "Statistics Education and the Internet", 2003.

[5] T. Baier, *R: Windows Component Services, Integrating R and Excel on the COM layer*, DSC 2003 Working Paper, available online at http://www.ci.tuwien.ac.at/Conferences/DSC-2003/, 2003.

[6] T. Baier and E. Neuwirth, *High-Level Interface between R and Excel*, DSC 2003 Working Paper, available online at http://www.ci.tuwien.ac.at/Conferences/DSC-2003/, 2003.

[7] K. Brockschmidt, *Inside Ole*, 2 ed., Microsoft Press, 1995.

[8] J. Chambers and D.T. Lang, $\hat{\Omega}$ *A Component-based Statistical Computing Environment*, Statistics and Data Mining Re-

search, Bell Labs, Technical Reports, available at http://cm.bell-labs.com/cm/ms/departments/sia/doc/comp.html, 1999.

[9] J.M. Chambers, M.H. Hansen, D.A. James, and D.T. Lang, *Distributed Computing with Data: A CORBA-Based Approach*, Statistics and Data Mining Research, Bell Labs, Technical Reports, available at http://cm.bell-labs.com/cm/ms/departments/sia/doc/comp.html, 1998.

[10] H.R. Cook, M.G. Cox, M.P. Dainton, and P.M. Harris, *Testing spreadsheets and other packages used in metrology, testing the instrinsic functions of excel*, Tech. Report NPL Report CISE 27/99, National Physical Laboratory, Queens Road, Teddington, Middlesex, TW11 0LW, September 1999, ISSN 1361-407X.

[11] J.D. Cryer, *Problems with using Microsoft Excel for Statistics*, Joint Statistical Meetings, August 2001, Altanta, GA, 2001.

[12] J. Feuerhake, *XQS/MD*Crypt as a Means of Education and Computation*, in *COMPSTAT 2002, Proceedings in Computational Statistics* [19], pp. 635–640.

[13] J. R. Geßler, *Statistische Graphik*, Birkhäuser, 1993.

[14] F. Griffel, *Componentware*, dpunkt-Verlag, 1998.

[15] W. Härdle, Z. Hlávka, and S. Klinke, *XploRe - Application Guide*, Springer-Verlag, 2000.

[16] W. Härdle, T. Kleinow, and G. Stahl (eds.), *Applied Quantitative Finance, Theory and Computational Tools*, Springer-Verlag, 2002.

[17] W. Härdle, T. Kleinow, and R. Tschernig, *Web Quantlets for Time Series Analysis*, The Annals of Mathematical Statistics **53** (2001), no. 1, 179–188.

[18] W. Härdle, S. Klinke, and M. Müller, *XploRe - Learning Guide*, Springer-Verlag, 2000.

[19] W. Härdle and B. Rönz (eds.), *COMPSTAT 20002, Proceedings in Computational Statistics*, Physica-Verlag, 2002.

[20] W. Härdle and L. Simar, *Applied Multivariate Statistical Analysis*, Springer-Verlag, 2003.

[21] T. Kleinow and T. Lehmann, *Computational Statistics*, vol. 17:3, ch. Client/Server based Statistical Computing, Physica-Verlag, 2002.

[22] S. Klinke, *Data Structures for Computational Statistics*, Contributions to Statistics, Physica-Verlag, 1997.

[23] L. Knüsel, *On the Accuracy of Statistical Distributions in Microsoft Excel 97*, Accuracy of Statistical Packages, available online at http://www.stat.uni-muenchen.de/∼knuesel/elv/accuracy.html, 1998.

[24] ———, *On the Reliability of Microsoft Excel XP for Statistical Purposes*, Accuracy of Statistical Packages, available online at http://www.stat.uni-muenchen.de/∼knuesel/elv/accuracy.html, 2002.

[25] D. Krahl, U. Windhäuser, and F.-K. Zick, *Data Mining - Einsatz in der Praxis*, Addison-Wesley, 1998.

[26] T. Kötter, *Entwicklung Statistischer Software*, Wirtschaftswissenschaftliche Beiträge, Physica-Verlag, 1998.

[27] H. Lehmann, *Client/Server Based Statistical Computing*, Ph.D. thesis, Wirtschaftswissenschafliche Fakultät, Humboldt-Uniervistät zu Berlin, 2004.

[28] F. Leisch, *Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis*, in *COMPSTAT 2002, Proceedings in Computational Statistics* [19], pp. 575–580.

[29] B.D. McCullough, *Is it safe to assume that software is accurate?*, International Journal of Forecasting **16** (2000), 349–357.

[30] B.D. McCullough and B. Wilson, *On the Accuracy of Statistical Procedures in Microsoft Excel*, Computational Statistics & Data Analysis **31** (1999), 27–37.

[31] M. Monka and W. Voß, *Statistik am PC*, Hanser, 2002.

[32] MSDN, *Microsoft Developer Network Library*, Microsoft online references available at http://msdn.microsoft.com/library/default.asp, 2004.

[33] E. Neuwirth, *in DiSessa, A. and Hoyles, C. (eds.): The Design of Computational Media to Support Exploratory Learning*, ch. Visualizing structural and formal relationships with spreadsheets, Springer-Verlag, 1996.

[34] _____, *in Filby, G. (ed.): Spreadsheets in Science and Engineering*, ch. Spreadsheets as Tools in Mathematical Modeling and Numerical Mathematics, Springer-Verlag, 1997.

[35] _____, *in Tinsley, D. and Johnson, D. (eds.): Information and Communications Technology in School Mathematics*, ch. Spreadsheets: just smart calculators or a new paradigm for thinking about mathematics structure?, Chapman-Hall, 1998.

[36] _____, *Spreadsheets as Tools for Statistical Computing and Statistics Education*, COMPSTAT 2000, Proceedings in Computational Statistics, 2000, pp. 131–138.

[37] F.C. Rice, *Building a COM Add-in for Microsoft Office XP*, Tech. report, Microsoft Corporation, 2002.

[38] B.D. Ripley, *Statistical Methods Need Software: A View of Statiscal Computing*, Opening Lecture RSS 2002, presentation available online at http://www.stats.ox.ac.uk/∼ripley/RSS2002.pdf, 2002.

[39] D. Rogerson, *Inside COM*, Microsoft Press, 1997.

[40] S. Roman, *What is an Add-in?*, Articles on VB/VBA/Office Programming, available online at http://www.romanpress.com/Articles/Articles.htm, 2001.

[41] _____, *Writing Excel Macros with VBA, 2nd Edition*, O'Reilly, 2002.

[42] G. Sawitzki, *Keeping Statistics Alive in Documents*, Computational Statistics **17** (2002), no. 1, 65–88.

[43] J.S. Simonoff, *Statistical Analysis using Microsoft Excel*, Statistics and Data Analysis Handout, available online at http://pages.stern.nyu.edu/∼jsimonof/classes/1305/pdf/excelreg.pdf, 2002.

[44] H. Sofyan and A. Werwatz, *Analysing XploRe Download Profiles with Intelligent Miner*, Computational Statistics **16** (2001), 465–479.

[45] J. Walkenbach, *Excel 2002 Formulas*, M & T Books, 2002.

[46] ———, *Excel 2002 Power Programming with VBA*, M & T Books, 2002.

[47] ———, *Excel Charts*, Wiley Publishing, 2002.

[48] R. Witzel and S. Klinke, *MD*Book online & e-stat: Generating e-stat Modules from LaTeX*, in *COMPSTAT 2002, Proceedings in Computational Statistics* [19], pp. 449–454.

Appendix A

Glossary

Abbreviations

COM	Component Object Model
CORBA	Common Object Request Broker
CUI	Commandline User Interface
DLL	Dynamic Link Library
GUI	Graphical User Interface
OLE	Object Linking and Embedding
PDF	Portable Document Format
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol
TCP/IP	Transport Control Protocol / Internet Protocol
VB	Visual Basic
VBA	Visual Basic for Applications
XML	Extensible Markup Language

Appendix B

Program Source

B.1 MD*ReX Source Tree

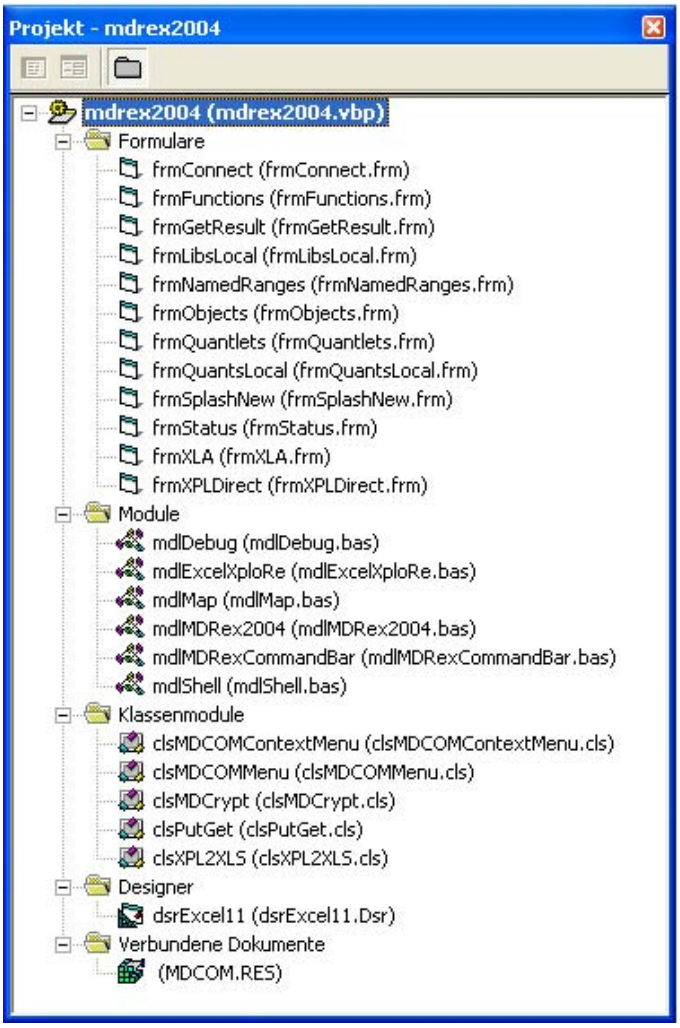


Figure B.1: MD*ReX source tree

B.2 MD*Serv Source Tree

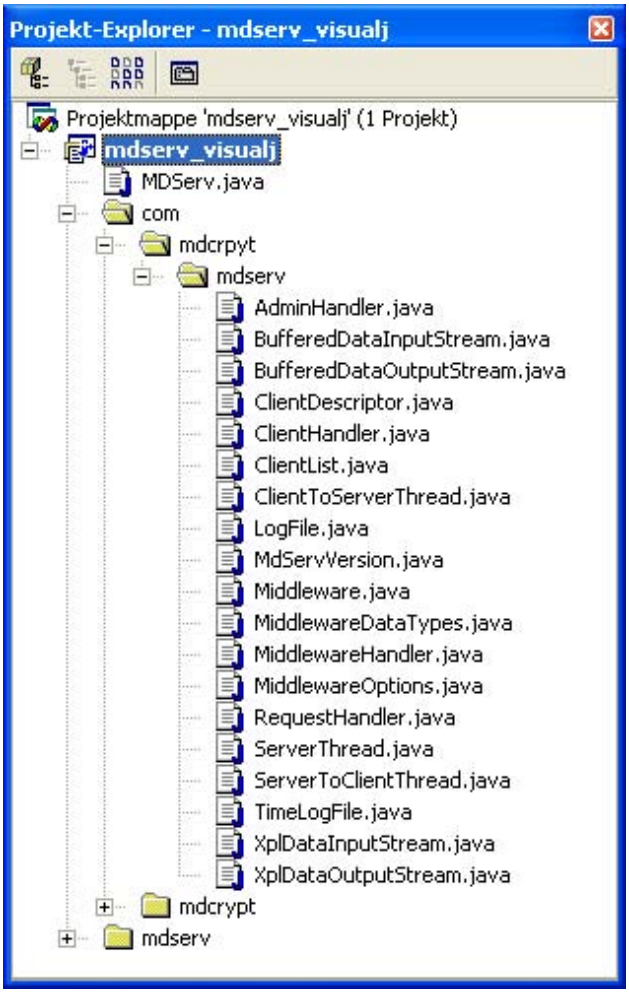


Figure B.2: MD*Serv source tree

B.3 Visual Basic Source

This section contains the complete source code of the discussed MD*ReX client.

B.4 mdlDebug.bas

```
Attribute VB_Name = "mdlDebug"
Option Explicit

Public Const dbgstar = "***" & vbTab
Public myFile As FileSystemObject

Sub ExeErr(errX As ErrObject)
    ' Zeigt ein Meldungsfeld mit Fehlerinformationen an.
    Dim strMsg As String
    strMsg = App.Title & "caused an error." & vbCrLf & "Error:" & errX.Number _
        & vbCrLf & errX.Description
    MsgBox strMsg, , App.EXEName
End Sub

Sub CreateDBGFolder()
    On Error GoTo Debug_Err:

    If Not (myFile.FolderExists(App.Path & "\debug")) Then
        myFile.CreateFolder (App.Path & "\debug")
        Call WriteDBGString("Creating debug folder in: " & App.Path & "\debug\" & vbCrLf, App.
             EXEName & ".log")
    End If

Exit Sub

Debug_Err:
    Call ExeErr(Err)
End Sub

Sub WriteDBGString(DBGString As String, DBGFileName As String)
    Dim tmpFile As Object
    On Error GoTo Debug_Err

    If Not (myFile.FileExists(App.Path & "\debug\" & DBGFileName)) Then
        myFile.CreateTextFile App.Path & "\debug\" & DBGFileName, True
        Call WriteDBGString("Creating debug file in: " & App.Path & "\debug\" & DBGFileName &
             vbCrLf, App.EXEName & ".log")
    End If

    Set tmpFile = myFile.OpenTextFile(App.Path & "\debug\" & DBGFileName, ForAppending, ,
         TristateUseDefault)
    tmpFile.write (dbgstar & DBGString)
    tmpFile.Close

Exit Sub

Debug_Err:
    Call ExeErr(Err)
End Sub
```

B.5 mdlExcelXploRe.bas

```
Attribute VB_Name = "mdlExcelXploRe"
Option Explicit
Public struFunc As String
Public strFunc As String
Public strPathTmp As String
Public qNameArray() As String
Public libNameArray() As String
Public xlaNameArray() As String
Public myQuantLibMapper As LibQuantMapper

Sub SendFunctionToVBA(FunctionToSend As String)
On Error GoTo SendFunctionErr:
    oHostInst.VBE.ActiveVBProject.VBComponents.Import (FunctionToSend)

Exit Sub

SendFunctionErr:
Select Case Err.Number
    Case Is = 0
        Exit Sub
    Case Else
        MsgBox "You must turn on access to VBA projects!" & vbCrLf & "To do this, point to Macro
             on the Tools menu" & vbCrLf & _
        "and then click Security, then click trusted sources tab, then click to select the TRUST
             access to Visual Basic Project check box.", vbOKOnly, Err.Number & " " & Err.
             Description
        Exit Sub
    End Select
End Sub

Function LoadVBAFunction() As String
Dim i As Integer
On Error Resume Next

Select Case (oHostInst.Version)
    Case "10.0"
        'struFunc = LoadResData(101, "CUSTOM")
        struFunc = LoadResData(103, "CUSTOM")
        For i = 1 To LenB(struFunc)
            strFunc = strFunc & Chr(AscB((MidB(struFunc, i, 1))))
        Next i
        LoadVBAFunction = strFunc
    Case Else
        struFunc = LoadResData(102, "CUSTOM")
        For i = 1 To LenB(struFunc)
            strFunc = strFunc & Chr(AscB((MidB(struFunc, i, 1))))
        Next i
        LoadVBAFunction = strFunc
    End Select

End Function

Sub CreateVBATxt()
Dim tmpFSO As FileSystemObject
Dim a As Variant

On Error Resume Next
    strPathTmp = App.Path & "\VBA.tmp"
    Set tmpFSO = New FileSystemObject

    Set a = tmpFSO.OpenTextFile(strPathTmp, ForWriting, True)
    Debug.Print strFunc
    a.write (strFunc)
    a.Close
    Set tmpFSO = Nothing
End Sub

Function GetLocalAddins(FileSpec As String) As Variant
    Dim FileCount As Integer
    Dim FileName As String

    On Error GoTo NoFilesFound

    FileCount = 0
    FileName = Dir(FileSpec)
    If FileName = "" Then GoTo NoFilesFound

'   Loop until no more matching files are found
    Do While FileName <> ""
        FileCount = FileCount + 1
        ReDim Preserve xlaNameArray(0 To FileCount)
        xlaNameArray(FileCount) = FileName
        FileName = Dir()
    Loop
    GetLocalAddins = xlaNameArray
```

```
    Exit Function

'   Error handler
NoFilesFound:
    MsgBox "No Addins found!"
    GetLocalAddins = False

End Function

Function GetLocalQList(FileSpec As String) As Variant
'   Returns an array of filenames that match FileSpec
'   If no matching files are found, it returns False

    Dim FileCount As Integer
    Dim FileName As String

    On Error GoTo NoFilesFound

    FileCount = 0
    FileName = Dir(FileSpec)
    If FileName = "" Then GoTo NoFilesFound

'   Loop until no more matching files are found
    Do While FileName <> ""
        FileCount = FileCount + 1
        ReDim Preserve qNameArray(1 To FileCount)
        qNameArray(FileCount) = FileName
        FileName = Dir()
    Loop
    GetLocalQList = qNameArray
    Exit Function

'   Error handler
NoFilesFound:
    GetLocalQList = False
End Function

Function GetLocalLibList(FileSpec As String) As Variant
'   Returns an array of filenames that match FileSpec
'   If no matching files are found, it returns False

    Dim FileCount As Integer
    Dim FileName As String

    On Error GoTo NoFilesFound

    FileCount = 0
    FileName = Dir(FileSpec)
    If FileName = "" Then GoTo NoFilesFound

'   Loop until no more matching files are found
    Do While FileName <> ""
        FileCount = FileCount + 1
        ReDim Preserve libNameArray(1 To FileCount)
        libNameArray(FileCount) = FileName
        FileName = Dir()
    Loop
    GetLocalLibList = libNameArray
    Exit Function

'   Error handler
NoFilesFound:
    GetLocalLibList = False
End Function

Function ReadQuantletFile(QFileName As String) As String()
Dim a
Dim i As Long
Dim temp
Set myFile = New FileSystemObject
Set a = myFile.OpenTextFile(QFileName, ForReading, False, TristateFalse)

For i = 1 To a.Line
temp = a.readLine(i)
Next i

End Function

Function IsRangeEmpty(RangeToCheck As Range) As Boolean
Dim rngTemp As Range
    For Each rngTemp In RangeToCheck
        If IsEmpty(rngTemp) Then
            IsRangeEmpty = True
        Else
            IsRangeEmpty = False
        End If
    Next
End Function
```

B.6 mdlMap.bas

```
Attribute VB_Name = "mdlMap"
Option Explicit
Public myMapper As XPL2XLSMapper
Public CounterRuns As Long

Sub Map(ByVal XLS As Range, ByVal XPL As String, Runs As Long)
    On Error GoTo MapErr
    Dim tString As String
        With myMapper
            ReDim Preserve .XLSObject(Runs - 1)
            ReDim Preserve .XPLObject(Runs - 1)
            ReDim Preserve .XLSObjectName(Runs - 1)
            .XLSObject(Runs - 1) = CStr(XLS.Worksheet.Name) & "_" & CStr(XLS.Address)
            Debug.Print XLS.Worksheet.Name
            XLS.Name = XPL
            .XLSObjectName(Runs - 1) = CStr(XPL)
            .XPLObject(Runs - 1) = XPL
            .XPLObjectCount = .XPLObjectCount + Runs
        End With
        Call WriteDBGString("mapping" & CStr(XPL) & vbCrLf, App.EXEName & ".log")
    Exit Sub

MapErr:
    Call ExeErr(Err)
    Exit Sub
End Sub

Function CountRuns() As Long
Dim j As Long
    CountRuns = j + 1
    CounterRuns = CounterRuns + CountRuns
End Function

Sub WriteName(ByVal sName As String, ByVal XLS As Range)
'
End Sub

Sub WriteMapper(Row As Long, Col As Long)
'//deprecated
Dim i%
    On Error GoTo Mapper_Err
    If SheetExists("XploRe 2 Excel Mapping Table") Then
        With oHostInst.Sheets("XploRe 2 Excel Mapping Table")
            .Unprotect
            .Cells(Row, 1).Value = "XploRe: " & myMapper.XPLObject(CounterRuns - 1)
            .Cells(Row, 1).Interior.ColorIndex = 3
            .Cells(Row, 2).Value = "Excel: " & myMapper.XLSObject(CounterRuns - 1)
            .Cells(Row, 2).Interior.ColorIndex = 32
            .Rows("1:" & Row).EntireRow.Hidden = False
            .Rows(Row + 1 & ":65536").EntireRow.Hidden = True
            .Columns(1).EntireColumn.Hidden = False
            .Columns(2).EntireColumn.Hidden = False
            .Columns(1).EntireColumn.AutoFit
            .Columns(2).EntireColumn.AutoFit
                For i = 3 To 256
                    .Columns(i).EntireColumn.Hidden = True
                Next
            .Protect
        End With
    Else
        AddMappingSheet ("XploRe 2 Excel Mapping Table")
        With oHostInst.Sheets("XploRe 2 Excel Mapping Table")
            .Unprotect
            .Cells(Row, 1).Value = "XploRe: " & myMapper.XPLObject(CounterRuns - 1)
            .Cells(Row, 1).ColorIndex = 3
            .Cells(Row, 1).Font.ColorIndex = 3
            .Cells(Row, 2).Value = "Excel: " & myMapper.XLSObject(CounterRuns - 1)
            .Cells(Row, 1).ColorIndex = 32
            .Cells(Row, 1).Font.ColorIndex = 2
            .Rows("1:" & Row).EntireRow.Hidden = False
            .Rows(Row + 1 & ":65536").EntireRow.Hidden = True
            .Columns(1).EntireColumn.Hidden = False
            .Columns(2).EntireColumn.Hidden = False
            .Columns(1).EntireColumn.AutoFit
            .Columns(2).EntireColumn.AutoFit
                For i = 3 To 256
                    .Columns(i).EntireColumn.Hidden = True
                Next
            .Protect
        End With
    End If
Exit Sub

Mapper_End:
    Exit Sub
```

```
Mapper_Err:
    Call ExeErr(Err)
    Resume Mapper_End
End Sub
```

B.7 mdlMDRex2004.bas

```
Attribute VB_Name = "mdlMDRex2004"
Option Explicit

Public oHostInst            As Object '//forward declare hosting Office App
Public oAddinInst           As Object '//forward declare hosted (COM)AddIn App
Public p_AddinInst          As Object
Public MenuItem             As Office.CommandBarButton '//forward declare container for MenuBar,
     needs Reference to MSOffice v.11 ObjectLib
Public PopupItem            As Office.CommandBarPopup '//forward declare container for PopUp
Public ContextMenuItem      As Office.CommandBarButton

Public p_OfficeCrypt        As clsMDCrypt '//forward declare MDCrypt Protocol implemented by MDRex
Public str_InfoMDCOM        As String '//container for MDCrypt messages
Public str_InfoMDCRYPT      As String '//container for MDCrypt messages
Public OfficeInstances      As Integer '//counter for calling Office Instances
Public clsResult()          As String '//container for Results

Public Const PROG_ID_START  As String = "!<" '//const for ProgIDs of MDRex COM-Addin
Public Const PROG_ID_END    As String = ">"

Public Const CBR_NAME       As String = "Worksheet Menu Bar" '//const for menu items in MSExcel
Public Const CTL_CAPTION    As String = "&MD*ReX 2004"
Public Const CTL_KEY        As String = "MDCOMAddIn"
Public Const CTL_NAME       As String = "MD*ReX 2004"

Public myPutGet             As clsPutGet
Public myMDCOMMenu          As clsMDCOMMenu '//pointer to clsMDCOMMenu, main UserInterface class
Public myMDCOMContextMenu   As clsMDCOMContextMenu '//pointer to clsMDCOMContextMenu

Public myFrmSplash          As frmSplashNew '//prefetch forms and keep them globally available
Public myFrmGetResult       As frmGetResult
Public myFrmFunctions       As frmFunctions
Public myFrmLibsLocal       As frmLibsLocal
Public myFrmNamedRanges     As frmNamedRanges
Public myFrmObjects         As frmObjects
Public myFrmQuantlets       As frmQuantlets
Public myFrmQuantsLocal     As frmQuantsLocal
Public myFrmStatus          As frmStatus
Public myFrmXLA             As frmXLA

Public blnRexToggled        As Boolean
Public LocalMDServ          As Boolean
Public myAddin1             As Excel.AddIn

Public ObjectString         As String '//global container for Objects
Public FunctionString       As String '//global container for Functions
Public QuantletString       As String '//global container for Quantlets
Public InfoString       As String '//global container for Quantlets

Public Function SaveHostApp(ByRef oHost As Object, ByRef oAddin As Object)
    Set oHostInst = oHost
    Set oAddinInst = oAddin
    If (myMDCOMMenu Is Nothing) Then
        Set myMDCOMMenu = New clsMDCOMMenu
        Set myMDCOMContextMenu = New clsMDCOMContextMenu
    End If
End Function

Public Function UpdateViewsMenu()
    With oHostInst
        If p_OfficeCrypt.ConnectedToServer = True Then
            .StatusBar = "MD*COM - XQS Status " & "[" & p_OfficeCrypt.GetServerStatus() & "]" & "
                 XQS IP " & "[" & p_OfficeCrypt.propServerIP & ":" & p_OfficeCrypt.propServerPort
                 & "]"
        Else
            .StatusBar = "MD*COM - XQS Status " & "[" & p_OfficeCrypt.GetServerStatus() & "]"
        End If
    End With
    Call UpdateMenuItems(myMDCOMMenu.cbMDCOM)
End Function

Public Function UpdateViewsContext()
    With oHostInst
        If p_OfficeCrypt.ConnectedToServer = True Then
            .StatusBar = "MD*COM - XQS Status " & "[" & p_OfficeCrypt.GetServerStatus() & "]" & "
                 XQS IP " & "[" & p_OfficeCrypt.propServerIP & ":" & p_OfficeCrypt.propServerPort
                 & "]"
        Else
            .StatusBar = "MD*COM - XQS Status " & "[" & p_OfficeCrypt.GetServerStatus() & "]"
        End If
    End With
    Call UpdateContextMenuItems(myMDCOMContextMenu.cmPut)
    Call UpdateContextMenuItems(myMDCOMContextMenu.cmGet)
    Call UpdateContextMenuItems(myMDCOMContextMenu.cmRun)
End Function
```

```
Public Sub UpdateMenuItems(ByRef cmdBar As Office.CommandBar)
    On Error GoTo UpdateMenuItemsErr
    Dim i%
    Set MenuItem = cmdBar.Controls.Item(1)
    With MenuItem
        If Not p_OfficeCrypt.ConnectedToServer Then
          .Style = msoButtonCaption
          .Enabled = True
          .Caption = "Connect"
          .Width = 50
        Else
          .Style = msoButtonCaption
          .Enabled = True
          .Caption = "Disconnect"
          .Width = 50
        End If
    End With

    For i = 2 To 8
       cmdBar.Controls(i).Enabled = p_OfficeCrypt.ConnectedToServer
    Next i

Exit Sub

UpdateMenuItemsErr:
    Call ExeErr(Err)
    Exit Sub

End Sub

Public Sub UpdateContextMenuItems(ByRef cmdBtn As Office.CommandBarButton)
    On Error GoTo UpdateContextMenuItemsErr
    Dim i%
    Set ContextMenuItem = cmdBtn '.Controls.Item(1)
    With ContextMenuItem
        If Not p_OfficeCrypt.ConnectedToServer Then
          .Enabled = False
        Else
          .Enabled = True
        End If
    End With

Exit Sub

UpdateContextMenuItemsErr:
    Call ExeErr(Err)
    Exit Sub

End Sub

Public Function LoadAddin(bool As Boolean, whichAddin As String, Optional str As String) As
      Boolean
    On Error GoTo LoadAddin_err

    If str <> "" Then
        Set myAddin1 = oHostInst.AddIns.Add(App.Path & "\" & whichAddin) ', True)
        Call WriteDBGString("trying to load external Excel addin (" & myAddin1.Name & ")" & vbCrLf
              , App.EXEName & ".log")
    End If

    If myAddin1 Is Nothing Then
        Exit Function
        Call WriteDBGString("no AddIn (" & myAddin1.Name & ") found." & vbCrLf, App.EXEName & ".
              log")
    End If

    If bool Then
        myAddin1.Installed = True
        Call WriteDBGString("loading (" & myAddin1.Name & ") succeeded." & vbCrLf, App.EXEName &
              ".log")
    Else
        myAddin1.Installed = False
        Call WriteDBGString("unloading (" & myAddin1.Name & ") succeeded." & vbCrLf, App.EXEName &
               ".log")
    End If

    Exit Function

LoadAddin_err:
    Call ExeErr(Err)
    Exit Function

End Function

Public Function FileToTextBox(TextBox As TextBox, Path As String)
Dim tmpFSO As FileSystemObject
Dim a As Variant
```

```
    On Error Resume Next
    strPathTmp = Path 'App.Path & "\VBA.tmp"
    Set tmpFSO = New FileSystemObject

    Set a = tmpFSO.OpenTextFile(strPathTmp, ForReading, False)
    TextBox.Text = a.ReadAll
    a.Close
    Set tmpFSO = Nothing
    TextBox.Text = Replace(TextBox.Text, Chr$(10), Chr$(13) & Chr$(10))
    TextBox.Text = Replace(TextBox.Text, Chr$(13) & Chr$(13), Chr$(13))
    Debug.Print TextBox.Text
End Function

Function SheetExists(NewSheetName) As Boolean
    'Returns TRUE if sheet exists in the active workbook
    Dim x As Object
    On Error Resume Next
    Set x = oHostInst.ActiveWorkbook.Sheets(NewSheetName)
    If Err = 0 Then SheetExists = True _
        Else: SheetExists = False
End Function

Sub AddMappingSheet(SheetName As String)
On Error Resume Next
If Not SheetExists(SheetName) Then
  oHostInst.Worksheets.Add.Move After:=oHostInst.Worksheets(oHostInst.Worksheets.Count)
  oHostInst.Worksheets(oHostInst.Worksheets.Count).Name = SheetName
  oHostInst.Sheets(SheetName).Protect
Else: Exit Sub
End If
End Sub

Sub DeleteAddIns(AddInName As String)
Dim tmpStr As String
tmpStr = AddInName
tmpStr = Replace(tmpStr, ".xla", "")
    On Error Resume Next
    'oHostInst.AddIns("Onevarsummary").Installed = False
    oHostInst.AddIns(tmpStr).Installed = False '//to assure that addins are correctly un/loaded
End Sub
```

B.8 mdlMDRexCommandBar.bas

```
Attribute VB_Name = "mdlMDRexCommandBar"
Option Explicit

Function CreateAddInCommandBarButton(ByVal Application As Object, _
    ByVal ConnectMode As AddInDesignerObjects.ext_ConnectMode, _
    ByVal AddInInst As Object) As Office.CommandBarButton

    ' Diese Prozedur ordnet einen Verweis auf das Application-Objekt, das
    ' an das OnConnection-Ereignis übergeben wurde, einer globalen
    ' Objektvariablen zu. Dann erstellt sie eine neue
    ' Befehlsleisten-Schaltfläche und gibt einen Verweis auf die
    ' Schaltfläche an die OnConnection-Ereignisprozedur zurück. Der
    ' Vorteil gegenüber dem Einfügen dieser Codeanweisung in ein
    ' öffentliches Modul besteht darin, dass Sie diese Prozedur bei einem
    ' Projekt mit mehreren Add-In-Designern aus den einzelnen Designern
    ' aufrufen können, anstatt die Codeanweisung duplizieren zu müssen.

    Dim cbrMenu                     As Office.CommandBar
    Dim ctlBtnAddIn                 As Office.CommandBarButton
    Dim myIcon                      As Variant
    Dim myClip                      As Clipboard
    Dim cbrProtection               As Long

    'MsgBox "Creating CommandBar!", vbInformation, "CreateAddInCommandBar"
    On Error GoTo CreateAddInCommandBarButton_Err

    ' Zurückgeben eines Verweises auf das Application-Objekt und Speichern
    ' dieses Objekts in einer öffentlichen Variablen, so dass andere Prozeduren
    ' im Add-In es verwenden können.
    Set oHostInst = Application

    ' Verweis auf Befehlsleiste zurückgeben.

    If (oHostInst <> "Microsoft Word") Then
        Set cbrMenu = oHostInst.CommandBars(CBR_NAME)
        cbrProtection = cbrMenu.Protection
        cbrMenu.Protection = msoBarNoProtection
    Else
        Set cbrMenu = oHostInst.CommandBars("Menu Bar")
        cbrProtection = cbrMenu.Protection
        cbrMenu.Protection = msoBarNoProtection
    End If

    ' Schaltfläche hinzufügen, um das Add-In aus der Befehlsleiste
    ' aufzurufen, wenn nicht bereits vorhanden.
    ' Konstanten werden auf Modulebene deklariert.
    ' Suchen nach der Schaltfläche auf der Befehlsleiste.
    Set ctlBtnAddIn = cbrMenu.FindControl(Tag:=CTL_KEY)
    If ctlBtnAddIn Is Nothing Then
        ' Neue Schaltfläche hinzufügen.
        Set ctlBtnAddIn = cbrMenu.Controls.Add(Type:=msoControlButton, _
            Parameter:=CTL_KEY)
        ' Schaltflächeneigenschaften Caption, Tag, Style und OnAction festlegen.
        With ctlBtnAddIn
            .Caption = CTL_CAPTION
            .Tag = CTL_KEY
            .Style = msoButtonIconAndCaption
            Clipboard.SetData LoadResPicture(104, vbResBitmap)
            .PasteFace
            Clipboard.Clear
            .OnAction = PROG_ID_START & AddInInst.ProgId & PROG_ID_END
        End With
    End If
    ' Verweis auf neue Befehlsleisten-Schaltfläche zurückgeben.
    Set CreateAddInCommandBarButton = ctlBtnAddIn

Exit Function

CreateAddInCommandBarButton_End:
    Exit Function

CreateAddInCommandBarButton_Err:
    ' Aufrufen einer generischen Fehlerbehandlungsroutine für das
    ' Add-In.
    Call ExeErr(Err)
    Resume CreateAddInCommandBarButton_End

End Function

Function CreateAddInComboBox(ByVal Application As Object, _
    ByVal AddInInst As Object) As Office.CommandBarComboBox

    ' Diese Prozedur ordnet einen Verweis auf das Application-Objekt, das
    ' an das OnConnection-Ereignis übergeben wurde, einer globalen
    ' Objektvariablen zu. Dann erstellt sie eine neue
```

```
    ' Befehlsleisten-Schaltfläche und gibt einen Verweis auf die
    ' Schaltfläche an die OnConnection-Ereignisprozedur zurück. Der
    ' Vorteil gegenüber dem Einfügen dieser Codeanweisung in ein
    ' öffentliches Modul besteht darin, dass Sie diese Prozedur bei einem
    ' Projekt mit mehreren Add-In-Designern aus den einzelnen Designern
    ' aufrufen können, anstatt die Codeanweisung duplizieren zu müssen.

    Dim cbrMenu                     As Office.CommandBar
    Dim ctlComboAddIn               As Office.CommandBarComboBox

    'MsgBox "Creating CommandBar!", vbInformation, "CreateAddInCommandBar"
    On Error GoTo CreateAddInComboBox_Err

    ' Zurückgeben eines Verweises auf das Application-Objekt und Speichern
    ' dieses Objekts in einer öffentlichen Variablen, so dass andere Prozeduren
    ' im Add-In es verwenden können.
    Set oHostInst = Application

    ' Verweis auf Befehlsleiste zurückgeben.
    Set cbrMenu = oHostInst.CommandBars("MD*ReX Toolbar")

    ' Schaltfläche hinzufügen, um das Add-In aus der Befehlsleiste
    ' aufzurufen, wenn nicht bereits vorhanden.
    ' Konstanten werden auf Modulebene deklariert.
    ' Suchen nach der Schaltfläche auf der Befehlsleiste.
    Set ctlComboAddIn = cbrMenu.FindControl(Tag:="MDCOMCMDLINE")
    ' Verweis auf neue Befehlsleisten-Schaltfläche zurückgeben.
    Set CreateAddInComboBox = ctlComboAddIn

Exit Function

CreateAddInComboBox_End:
    Exit Function

CreateAddInComboBox_Err:
    ' Aufrufen einer generischen Fehlerbehandlungsroutine für das
    ' Add-In.
    Call ExeErr(Err)
    Resume CreateAddInComboBox_End

End Function

Function RemoveAddInCommandBarButton(ByVal _
    RemoveMode As AddInDesignerObjects.ext_DisconnectMode)
    Dim cbrMenu                     As Office.CommandBar
    Dim ctlBtnAddIn                 As Office.CommandBarButton
    Dim temp As Integer
    Dim i As Integer

    ' Dieses Verfahren entfernt die Befehlsleisten-Schaltfläche für das
    ' Add-In, wenn der Benutzer die Verbindung getrennt hat.

    On Error GoTo RemoveAddInCommandBarButton_Err

    ' Wenn der Benutzer das Add-In aus dem Speicher entfernt,
    ' Schaltflächen entfernen. Wenn nicht, Add-In aus dem
    ' Speicher entfernen, da die Anwendung schließt. In diesem
    ' Fall muss die Schaltfläche nicht entfernt werden.

    If (p_OfficeCrypt.ConnectedToServer = True) Then
        temp = MsgBox("You are still connected to: " & vbCrLf & p_OfficeCrypt.propServerIP &
              vbCrLf & ". You will now be disconected." & vbCrLf & "WARNING: all data on the server
               will be lost!" & vbCrLf, vbOKOnly, "Disconnect " & p_OfficeCrypt.propServerIP)
        p_OfficeCrypt.clsTerminate
    End If

    Select Case RemoveMode
        Case ext_dm_HostShutdown
            Call myMDCOMMenu.MDCOMDisconnect
            Call WriteDBGString("MD*ReX Host shut down" & vbCrLf & "RemoveMode: Host Shutdown" &
                  vbCrLf & "Will Remove AddInCommandbarButton", App.EXEName & ".log")
            oHostInst.CommandBars(CBR_NAME).Controls(CTL_NAME).Delete
        Case ext_dm_UserClosed
            On Error Resume Next
            Call myMDCOMMenu.MDCOMDisconnect
            ' Benutzerdefinierte Befehlsleisten-Schaltfläche löschen.
            oHostInst.CommandBars(CBR_NAME).Controls(CTL_NAME).Delete
            Call WriteDBGString("MD*ReX Host shut down" & vbCrLf & "RemoveMode: User Closed" &
                  vbCrLf & "Will Remove AddInCommandbarButton", App.EXEName & ".log")
            On Error GoTo RemoveAddInCommandBarButton_Err
    End Select

RemoveAddInCommandBarButton_End:
    Exit Function

RemoveAddInCommandBarButton_Err:
    Call ExeErr(Err)
    Resume RemoveAddInCommandBarButton_End
End Function
```

B.9 mdlShell.bas

```
Attribute VB_Name = "mdlShell"
Option Explicit

      Private Type PROCESS_INFORMATION
         hProcess As Long
         hThread As Long
         dwProcessId As Long
         dwThreadId As Long
      End Type

      Private Type STARTUPINFO
         cb As Long
         lpReserved As String
         lpDesktop As String
         lpTitle As String
         dwX As Long
         dwY As Long
         dwXSize As Long
         dwYSize As Long
         dwXCountChars As Long
         dwYCountChars As Long
         dwFillAttribute As Long
         dwFlags As Long
         wShowWindow As Integer
         cbReserved2 As Integer
         lpReserved2 As Long
         hStdInput As Long
         hStdOutput As Long
         hStdError As Long
      End Type

Public hProcID As Long

Public Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA" (ByVal hWnd As Long,
      ByVal lpszOp As String, ByVal lpszFile As String, ByVal lpszParams As String, ByVal lpszDir
      As String, ByVal FsShowCmd As Long) As Long

Public Declare Function WaitForSingleObject Lib "kernel32" (ByVal hHandle As Long, ByVal
      dwMilliseconds As Long) As Long

'gets hObject as dwProcessId
Private Declare Function OpenProcess Lib "kernel32" (ByVal dwDesiredAccess As Long, ByVal
      bInheritHandle As Long, ByVal dwProcessId As Long) As Long

Private Declare Function PostMessage Lib "user32" _
         Alias "PostMessageA" _
         (ByVal hWnd As Long, _
         ByVal wMsg As Long, _
         ByVal wParam As Long, _
         ByVal lParam As Long) As Long

Private Declare Function IsWindow Lib "user32" _
         (ByVal hWnd As Long) As Long

Private Declare Function FindWindow Lib "user32" _
         Alias "FindWindowA" _
         (ByVal lpClassName As String, _
         ByVal lpWindowName As String) As Long

Private Declare Function CreateProcess Lib "kernel32" _
   Alias "CreateProcessA" _
   (ByVal lpApplicationName As String, _
   ByVal lpCommandLine As String, _
   lpProcessAttributes As Any, _
   lpThreadAttributes As Any, _
   ByVal bInheritHandles As Long, _
   ByVal dwCreationFlags As Long, _
   lpEnvironment As Any, _
   ByVal lpCurrentDriectory As String, _
   lpStartupInfo As STARTUPINFO, _
   lpProcessInformation As PROCESS_INFORMATION) As Long


Private Declare Function TerminateProcess Lib "kernel32" (ByVal hProcess As Long, ByVal uExitCode
      As Long) As Long

Private Declare Function GetExitCodeProcess Lib "kernel32" (ByVal hProcess As Long, lpExitCode As
      Long) As Long

Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long

Private Declare Function GetWindow Lib "user32" (ByVal hWnd As Long, ByVal wCmd As Long) As Long

Private Declare Function GetDesktopWindow Lib "user32" () As Long
```

```
Private Declare Function BringWindowToTop Lib "user32" (ByVal hWnd As Long) As Long

Private Declare Function GetWindowThreadProcessId Lib "user32" (ByVal hWnd As Long, lpdwProcessId
      As Long) As Long

Private Declare Function SetWindowText Lib "user32" Alias "SetWindowTextA" (ByVal hWnd As Long,
      ByVal lpString As String) As Long

Private Declare Function GetClassName Lib "user32" Alias "GetClassNameA" (ByVal hWnd As Long,
      ByVal lpClassName As String, ByVal nMaxCount As Long) As Long

Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd As Long, ByVal
      wMsg As Long, ByVal wParam As Long, lParam As Any) As Long

Public Declare Function FindWindowEx Lib "user32" Alias "FindWindowExA" (ByVal hWnd1 As Long,
      ByVal hWnd2 As Long, ByVal lpsz1 As String, ByVal lpsz2 As String) As Long

Const GW_HWNDFIRST = 0
Const GW_HWNDNEXT = 2
Const GW_CHILD = 5
Const WM_SETTEXT = &HC
Const SYNCHRONIZE = 1048576
Const NORMAL_PRIORITY_CLASS = &H20&

'Constants used by the API functions
      Const WM_CLOSE = &H10
      Const INFINITE = &HFFFFFFFF


Public Const SW_NOSHOW = 0
Public Const SE_ERR_FNF = 2&
Public Const SE_ERR_PNF = 3&
Public Const SE_ERR_ACCESSDENIED = 5&
Public Const SE_ERR_OOM = 8&
Public Const SE_ERR_DLLNOTFOUND = 32&
Public Const SE_ERR_SHARE = 26&
Public Const SE_ERR_ASSOCINCOMPLETE = 27&
Public Const SE_ERR_DDETIMEOUT = 28&
Public Const SE_ERR_DDEFAIL = 29&
Public Const SE_ERR_DDEBUSY = 30&
Public Const SE_ERR_NOASSOC = 31&
Public Const ERROR_BAD_FORMAT = 11&
Public AddInFSO As FileSystemObject

Dim pInfo As PROCESS_INFORMATION
Dim sInfo As STARTUPINFO
Dim sNull As String
Dim lSuccess As Long
Dim lRetValue As Long

Function GethWndFromProcID(hProcIDToFind As Long) As Long
Dim hWndDesktop As Long
Dim hWndChild As Long
Dim hWndChildProcID As Long

On Local Error GoTo GethWndFromProcID_Error
hWndDesktop = GetDesktopWindow()
hWndChild = GetWindow(hWndDesktop, GW_CHILD)

Do While hWndChild <> 0
    Call GetWindowThreadProcessId(hWndChild, hWndChildProcID)
        If hWndChildProcID = hProcIDToFind Then
            GethWndFromProcID = hWndChild
            Exit Do
        End If
    hWndChild = GetWindow(hWndDesktop, GW_HWNDNEXT)
Loop

Exit Function

GethWndFromProcID_Error:
    GethWndFromProcID = 0
    Exit Function

End Function

Public Function TriggerMDServ() As Boolean
Dim r                           As Long
Dim msg                         As String
Dim ShellString As String

    ShellString = App.Path & "\mdserv\MDCOM.exe"
    r = StartMDServ(ShellString)
          If r <= 32 Then
            Select Case r
                Case SE_ERR_FNF
                    msg = "File runme.bat not found. It should be located under: " & App.Path & "\
                          mdserv"
                Case SE_ERR_PNF
```

```
                    msg = "Path to mdserv not found. Path should be: " & App.Path & "\mdserv"
                Case SE_ERR_ACCESSDENIED
                    msg = "Access denied"
                Case SE_ERR_OOM
                    msg = "Out of memory"
                Case SE_ERR_DLLNOTFOUND
                    msg = "DLL not found"
                Case SE_ERR_SHARE
                    msg = "A sharing violation occurred"
                Case SE_ERR_ASSOCINCOMPLETE
                    msg = "Incomplete or invalid file association"
                Case SE_ERR_DDETIMEOUT
                    msg = "DDE Time out"
                Case SE_ERR_DDEFAIL
                    msg = "DDE transaction failed"
                Case SE_ERR_DDEBUSY
                    msg = "DDE busy"
                Case SE_ERR_NOASSOC
                    msg = "No association for file extension"
                Case ERROR_BAD_FORMAT
                    msg = "Invalid EXE file or error in EXE image"
                Case Else
                    msg = "Unknown error"
            End Select
            Call WriteDBGString(msg & vbCrLf, App.EXEName & ".log")
            TriggerMDServ = False
            Exit Function
        Else
            TriggerMDServ = True
        End If
End Function

Public Function StartMDServ(DocName As String) As Long
'trusts in a well behaved IS installation of MDReX where mdserv is installed
Dim Scr_hDC As Long
Scr_hDC = GetDesktopWindow()
hProcID = ShellExecute(Scr_hDC, "Open", DocName, "", App.Path & "\mdserv", 0) '1 visible 0
     invisible
StartMDServ = hProcID
Call WriteDBGString("MDCOM.exe middleware has been launched! ProcID = " & hProcID & "" & vbCrLf,
     App.EXEName & ".log")
End Function

Public Sub CloseMDCOM()
        'Closes the MDCOM console app
        Dim hWindow As Long
        Dim lngResult As Long
        Dim lngReturnValue As Long
        Dim retval As Long
        Call WriteDBGString("Sending message WM_CLOSE to MDCOM.exe" & vbCrLf, App.EXEName & ".log
             ")
        hWindow = FindWindow(vbNullString, App.Path & "\MDServ\MDCOM.exe")
        lngReturnValue = PostMessage(hWindow, WM_CLOSE, vbNull, vbNull)
        lngResult = WaitForSingleObject(hWindow, INFINITE)
        retval = TerminateProcess(hWindow, 0&)
        Call WriteDBGString("MDCOM.exe almost surely closed" & vbCrLf, App.EXEName & ".log")
End Sub
```

B.10 clsMDCOMContextMenu.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "clsMDCOMContextMenu"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Public WithEvents cmPut                     As Office.CommandBarButton
Attribute cmPut.VB_VarHelpID = -1
Public WithEvents cmGet                     As Office.CommandBarButton
Attribute cmGet.VB_VarHelpID = -1
Public WithEvents cmRun                     As Office.CommandBarButton
Attribute cmRun.VB_VarHelpID = -1

Dim cMenuExists                             As Boolean
Dim dbgStr                                  As String

Public Sub MDCOMConnectContextMenu(ByRef oAddin As Object, ByRef oApp As Object, cmdBar As Office.
      CommandBar)
    Dim i As Integer
    Dim j%
    Dim str As String
    Dim ContextItem
    Dim cbar As CommandBar
    dbgStr = dbgstar & "Entering GUI setup MDCOMConnectContextMenu" & vbCrLf
    On Error GoTo ContextMenu_Err
    Call DestroyContextMenu(oApp)  '//obtain clean environment
    str = "Cell" '// for now only in cells, later we can traverse through
                 '// various context menus as follows:
                 '//for i=1 to 3
                 '//Array("Cell", "Column", "Row")(i - 1)

    Set cbar = oApp.CommandBars(str)
    Set cmPut = cbar.Controls.Add(Type:=msoControlButton, Before:=1, temporary:=True)
        With cmPut
            .BeginGroup = True
            .Style = msoButtonIconAndCaption
            .Caption = "Put..."
            .ToolTipText = "Upload Excel Range as XploRe Object"
            .OnAction = "!<" & oAddin.ProgId & ">"
            .Tag = "MDCOMCMPUT"
            .Visible = True
            Clipboard.SetData LoadResPicture(104, vbResBitmap)
            .PasteFace
            Clipboard.Clear
        End With
    Set cmGet = cbar.Controls.Add(Type:=msoControlButton, Before:=2, temporary:=True)
        With cmGet
            .Style = msoButtonCaption
            .Caption = "Get..."
            .ToolTipText = "Get XploRe Object as Excel Range"
            .OnAction = "!<" & oAddin.ProgId & ">"
            .Tag = "MDCOMCMGET"
            .Visible = True
        End With
    Set cmRun = cbar.Controls.Add(Type:=msoControlButton, Before:=3, temporary:=True)
        With cmRun
            .Style = msoButtonCaption
            .Caption = "Run"
            .ToolTipText = "Execute XploRe command from Excel cells"
            .OnAction = "!<" & oAddin.ProgId & ">"
            .Tag = "MDCOMCMRUN"
            .Visible = True
        End With

    dbgStr = dbgstar & dbgStr & "creating menu bar succeeded" & vbCrLf
    Call WriteDBGString(dbgStr, App.EXEName & ".log")
    dbgStr = ""
    UpdateViewsContext

Exit Sub

ContextMenu_Err:
    Call ExeErr(Err)
    Exit Sub

End Sub
```

```
Public Sub MDCOMContextMenuDisconnect()
    On Error GoTo MDCOMContextMenuDisconnectErr
    dbgStr = dbgstar & "Entering GUI setup MDCOMContextMenuDisconnect" & vbCrLf
    Call DestroyContextMenu(oHostInst)
    'Set cmMDCOM = Nothing
    dbgStr = dbgstar & "Deleting GUI context menu finished" & vbCrLf
    Exit Sub

MDCOMContextMenuDisconnectErr_exit:
        Exit Sub

MDCOMContextMenuDisconnectErr:
        Select Case Err.Number
            Case Err.Number = 0
                GoTo MDCOMContextMenuDisconnectErr_exit:
            Case Err.Number <> 0
                MsgBox Err.Number & " " & Err.Description, vbCritical, "DestroyContextMenu"
                GoTo MDCOMContextMenuDisconnectErr_exit:
        End Select
End Sub

Public Sub DestroyContextMenu(ByRef oApp As Object)
    Dim cbar As CommandBar
    On Error GoTo DestroyContextMenuErr

    Set cbar = oApp.CommandBars("Cell")
    Set cmPut = cbar.FindControl(Tag:="MDCOMCMPUT")
    Set cmGet = cbar.FindControl(Tag:="MDCOMCMGET")
    Set cmRun = cbar.FindControl(Tag:="MDCOMCMRUN")
    cmPut.Delete
    cmGet.Delete
    cmRun.Delete
Exit Sub

DestroyContextMenu_exit:
Exit Sub

DestroyContextMenuErr:
    Select Case Err.Number
        Case Err.Number = 0
            GoTo DestroyContextMenu_exit:
        Case Err.Number <> 0
            MsgBox Err.Number & " " & Err.Description, vbCritical, "DestroyContextMenu"
            GoTo DestroyContextMenu_exit:
    End Select
End Sub

Private Sub cmGet_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    myPutGet.GetXPL
End Sub

Private Sub cmPut_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    myPutGet.PutXpl
End Sub

Private Sub cmRun_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
MsgBox "RUN"
End Sub
```

B.11 clsMDCOMMenu.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "clsMDCOMMenu"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
Option Explicit

Public WithEvents oHostXLS                  As Excel.Application
Attribute oHostXLS.VB_VarHelpID = -1
Public WithEvents oHostDOC                  As Word.Application
Attribute oHostDOC.VB_VarHelpID = -1

Dim oHostApp                                As Object
Dim oAddin                                  As Object
Dim oApp                                    As Object
Public cbMDCOM                              As Office.CommandBar

Dim btnInfo                                 As Office.CommandBarPopup
Attribute btnInfo.VB_VarHelpID = -1
Dim WithEvents btnXPLObjects                As Office.CommandBarButton
Attribute btnXPLObjects.VB_VarHelpID = -1
Dim WithEvents btnXPLFunctions              As Office.CommandBarButton
Attribute btnXPLFunctions.VB_VarHelpID = -1
Dim WithEvents btnXPLQuantlets              As Office.CommandBarButton
Attribute btnXPLQuantlets.VB_VarHelpID = -1
Dim WithEvents btnXPLLibraries              As Office.CommandBarButton
Attribute btnXPLLibraries.VB_VarHelpID = -1
Dim WithEvents btnXPLNamedRanges            As Office.CommandBarButton
Attribute btnXPLNamedRanges.VB_VarHelpID = -1
Dim WithEvents btnXPLLogs                   As Office.CommandBarButton
Attribute btnXPLLogs.VB_VarHelpID = -1

Dim WithEvents btnXQScrap                   As Office.CommandBarButton
Attribute btnXQScrap.VB_VarHelpID = -1
Dim WithEvents btnPut                       As Office.CommandBarButton
Attribute btnPut.VB_VarHelpID = -1
Dim WithEvents btnRun                       As Office.CommandBarButton
Attribute btnRun.VB_VarHelpID = -1
Dim WithEvents btnGet                       As Office.CommandBarButton
Attribute btnGet.VB_VarHelpID = -1

Dim btnConsole                              As Office.CommandBarPopup
Dim WithEvents btnCMDLine                   As Office.CommandBarComboBox
Attribute btnCMDLine.VB_VarHelpID = -1
Dim WithEvents btnXPLDirect                 As Office.CommandBarButton
Attribute btnXPLDirect.VB_VarHelpID = -1
Dim WithEvents btnXPL_xla                   As Office.CommandBarButton
Attribute btnXPL_xla.VB_VarHelpID = -1
'Dim WithEvents btnXPL_adstat                 As Office.CommandBarButton
'Dim WithEvents btnXPL_anddar                 As Office.CommandBarButton
'Dim WithEvents btnXPL_anova                  As Office.CommandBarButton
'Dim WithEvents btnXPL_bootstrap              As Office.CommandBarButton
'Dim WithEvents btnXPL_factoranalysis         As Office.CommandBarButton
'Dim WithEvents btnXPL_hazard                 As Office.CommandBarButton
'Dim WithEvents btnXPL_linreg                 As Office.CommandBarButton
'Dim WithEvents btnXPL_lorenz                 As Office.CommandBarButton
'Dim WithEvents btnXPL_meancl                 As Office.CommandBarButton
'Dim WithEvents btnXPL_mediancl               As Office.CommandBarButton
'Dim WithEvents btnXPL_moduscl                As Office.CommandBarButton
'Dim WithEvents btnXPL_rankcorr               As Office.CommandBarButton
'Dim WithEvents btnXPL_summarize              As Office.CommandBarButton
'Dim WithEvents btnXPL_varcl                  As Office.CommandBarButton
'Dim WithEvents btnXPL_covcl                  As Office.CommandBarButton
'Dim WithEvents btnXPL_ttest                  As Office.CommandBarButton
'Dim WithEvents btnXPL_ftest                  As Office.CommandBarButton

Dim WithEvents btnFormulaWatch              As Office.CommandBarButton
Attribute btnFormulaWatch.VB_VarHelpID = -1
Dim btnHelp                                 As Office.CommandBarPopup
Attribute btnHelp.VB_VarHelpID = -1
Dim WithEvents btnMDCOMHelp                 As Office.CommandBarButton
Attribute btnMDCOMHelp.VB_VarHelpID = -1
Dim WithEvents btnXPLHelp                   As Office.CommandBarButton
Attribute btnXPLHelp.VB_VarHelpID = -1
Dim WithEvents btnWWWAPSS                   As Office.CommandBarButton
Attribute btnWWWAPSS.VB_VarHelpID = -1
Dim WithEvents btnWWWMDCOM                  As Office.CommandBarButton
```

```
Attribute btnWWWMDCOM.VB_VarHelpID = -1
Dim WithEvents btnWWWXPL                    As Office.CommandBarButton
Attribute btnWWWXPL.VB_VarHelpID = -1
Dim btnAbout                                As Office.CommandBarPopup
Dim WithEvents btnAboutMDCOM                As Office.CommandBarButton
Attribute btnAboutMDCOM.VB_VarHelpID = -1
Dim WithEvents btnAboutXPL                  As Office.CommandBarButton
Attribute btnAboutXPL.VB_VarHelpID = -1

Dim WithEvents MenuItem                     As Office.CommandBarButton
Attribute MenuItem.VB_VarHelpID = -1
Dim bBarExists                              As Boolean

Dim myFrmConnect                            As frmConnect
Dim myFrmXPLDirect                          As frmXPLDirect
Dim strResults                              As String
Dim dbgStr                                  As String

Public Sub MDCOMConnect(ByRef oAddin As Object, ByRef oApp As Object, cmdBar As Office.CommandBar)
    Dim i As Integer
    Dim j%
    Dim MenuItem

    dbgStr = dbgstar & "Entering GUI setup MDCOMConnect" & vbCrLf

    On Error Resume Next
    Set myFrmConnect = New frmConnect '//forward references
    Set myFrmXPLDirect = New frmXPLDirect

    DestroyToolBars '//obtain clean environment

    If oApp = "Microsoft Excel" Then '//switch to control for hosting Office app
        Set oHostXLS = oApp

            Set cbMDCOM = oApp.CommandBars.FindControl("MD*ReX Toolbar")
            i = oHostXLS.CommandBars.FindControl(Tag:="MDCOMPUT").Index
            'i = oHostXLS.Toolbars("MD*ReX Toolbar").Index
            'If (cbMDCOM Is Nothing) And (IsEmpty(i) = True) Then bBarExists = False
            'If (i <> 0) Then bBarExists = True
            If (cbMDCOM Is Nothing) Then bBarExists = False
            On Error GoTo 0
            If bBarExists Then '//toolbar already exists
                dbgStr = dbgstar & dbgStr & "bBarExists returned true" & vbCrLf
                Set cbMDCOM = oHostXLS.CommandBars("MD*ReX Toolbar")
                Set btnInfo = cbMDCOM.FindControl(Tag:="MDCOMNetEnvInfo", recursive:=msoTrue)
                Set btnXQScrap = cbMDCOM.FindControl(Tag:="MDCOMXQS", recursive:=msoTrue)
                Set btnPut = cbMDCOM.FindControl(Tag:="MDCOMPUT", recursive:=msoTrue)
                Set btnGet = cbMDCOM.FindControl(Tag:="MDCOMGet", recursive:=msoTrue)
                Set btnConsole = cbMDCOM.FindControl(Tag:="MDCOMSHELL", recursive:=msoTrue)
                Set btnCMDLine = cbMDCOM.FindControl(Tag:="MDCOMCMDLINE", recursive:=msoTrue)
                Set btnXPLDirect = cbMDCOM.FindControl(Tag:="MDCOMXPLDIRECT", recursive:=msoTrue)
                Set btnHelp = cbMDCOM.FindControl(Tag:="MDHELP", recursive:=msoTrue)
                Set btnMDCOMHelp = cbMDCOM.FindControl(Tag:="MDCOMHELP", recursive:=msoTrue)
                Set btnXPLHelp = cbMDCOM.FindControl(Tag:="MDCOMXPLHELP", recursive:=msoTrue)
                Set btnAboutMDCOM = cbMDCOM.FindControl(Tag:="MDCOMABOUT1", recursive:=msoTrue)
                Set btnAboutXPL = cbMDCOM.FindControl(Tag:="MDCOMABOUT2", recursive:=msoTrue)
                dbgStr = dbgstar & dbgStr & "recursively activated menu items " & vbCrLf
            Else '//now create the toolbar
                dbgStr = dbgstar & dbgStr & "bBarExists returned false"
                Set cbMDCOM = oHostXLS.CommandBars.Add("MD*ReX Toolbar", msoBarTop, , msoFalse)
                With cbMDCOM
                .Protection = msoBarNoCustomize '//don't allow changes on toolbar
                .Visible = True
                dbgStr = dbgstar & dbgStr & "creating menu bar from scratch" & vbCrLf
                Set btnXQScrap = .Controls.Add(1) '//1st toolbar menu
                With btnXQScrap
                    .Style = msoButtonCaption
                    .Caption = ""
                    .ToolTipText = ""
                    .DescriptionText = ""
                    .OnAction = "" '// empty because handled by dsr
                    .Tag = "MDCOMXQS"
                    .Width = 40
                    .Visible = True
                    .BeginGroup = True
                End With

                Set btnInfo = .Controls.Add(msoControlPopup) '//this is a popup hosting further
                     menus
                With btnInfo
                    .BeginGroup = True
                    .Caption = "MD*ReX Info"
                    .DescriptionText = "XPL Shell"
                    .Tag = "MDCOMNetEnvInfo"
                    .ToolTipText = "Provides Current Networking and Environment info."""
                    .Visible = True

                        Set btnXPLObjects = .Controls.Add(1)
                        With btnXPLObjects
```

```
                    .Style = msoButtonCaption
                    .Caption = "XploRe Objects"
                    .ToolTipText = "Objects known on XQS"
                    .DescriptionText = "XPLObjects"
                    .OnAction = "!<" & oAddin.ProgId & ">"
                    .Tag = "MDCOMXPLOBJECTS"
                    .Visible = True
                End With

                Set btnXPLFunctions = .Controls.Add(1)
                With btnXPLFunctions
                    .Style = msoButtonCaption
                    .Caption = "XploRe Functions"
                    .ToolTipText = "Functions known on XQS"
                    .DescriptionText = "XPLFunctions"
                    .OnAction = "!<" & oAddin.ProgId & ">"
                    .Tag = "MDCOMXPLFUNCTIONS"
                    .Visible = True
                End With

                Set btnXPLQuantlets = .Controls.Add(1)
                With btnXPLQuantlets
                    .Style = msoButtonCaption
                    .Caption = "XploRe Quantlets"
                    .ToolTipText = "Quantlets known on XQS"
                    .DescriptionText = "XPLQuantlets"
                    .OnAction = "!<" & oAddin.ProgId & ">"
                    .Tag = "MDCOMXPLQUANTLETS"
                    .Visible = True
                End With

                Set btnXPLLibraries = .Controls.Add(1)
                With btnXPLLibraries
                    .Style = msoButtonCaption
                    .Caption = "XploRe Libraries"
                    .ToolTipText = "Libraries known on XQS"
                    .DescriptionText = "XPLLibraries"
                    .OnAction = "!<" & oAddin.ProgId & ">"
                    .Tag = "MDCOMXPLLIBRARIES"
                    .Visible = True
                End With

                Set btnXPLNamedRanges = .Controls.Add(1)
                With btnXPLNamedRanges
                    .BeginGroup = True
                    .Style = msoButtonCaption
                    .Caption = "Named Ranges"
                    .ToolTipText = "Mapping between XPL Objects and XLS Ranges"
                    .DescriptionText = "NamedRanges"
                    .OnAction = "!<" & oAddin.ProgId & ">"
                    .Tag = "MDCOMXPLNAMEDRANGES"
                    .Visible = True
                End With

                Set btnXPLLogs = .Controls.Add(1)
                With btnXPLLogs
                    .BeginGroup = True
                    .Style = msoButtonCaption
                    .Caption = "Status Logs"
                    .ToolTipText = "View Log files"
                    .DescriptionText = "XPLLogs"
                    .OnAction = "!<" & oAddin.ProgId & ">"
                    .Tag = "MDCOMXPLLOGS"
                    .Visible = True
                End With

            End With

            Set btnPut = .Controls.Add(1)
            With btnPut
                .Style = msoButtonCaption
                .Caption = "Put"
                .ToolTipText = "Put"
                .DescriptionText = "Put"
                .OnAction = "!<" & oAddin.ProgId & ">"
                .Tag = "MDCOMPUT"
                .Visible = True
                .BeginGroup = True
            End With

            Set btnGet = .Controls.Add(1)
            With btnGet
                .Style = msoButtonCaption
                .Caption = "Get"
                .ToolTipText = "Get"
                .DescriptionText = "Get"
                .OnAction = "!<" & oAddin.ProgId & ">"
                .Tag = "MDCOMGet"
                .Visible = True
```

```
End With

Set btnConsole = .Controls.Add(msoControlPopup) '//this is a popup hosting further
      menus
With btnConsole
    .BeginGroup = True
    .Caption = "XploRe"
    .DescriptionText = "XPL Shell"
    .Tag = "MDCOMSHELL"
    .ToolTipText = "XploRe"
    .Visible = True

        Set btnXPLDirect = .Controls.Add(1)
        With btnXPLDirect
            .Style = msoButtonCaption
            .Caption = "XploRe Direct"
            .ToolTipText = "XploRe Direct"
            .DescriptionText = "XPLDirect"
            .OnAction = "!<" & oAddin.ProgId & ">"
            .Tag = "MDCOMXPLDIRECT"
            .Visible = True
        End With

        Set btnXPL_xla = .Controls.Add(1)
        With btnXPL_xla
            .Style = msoButtonCaption
            .Caption = "MD*ReX AddIns"
            .ToolTipText = "Load MD*ReX Excel AddIns (.xla files)"
            .DescriptionText = "XPLXLA"
            .OnAction = "!<" & oAddin.ProgId & ">"
            .Tag = "MDCOMXPLXLA"
            .Visible = True
        End With
End With

Set btnCMDLine = .Controls.Add(msoControlComboBox) 'Add '(2)
With btnCMDLine
    .BeginGroup = True
    .DropDownLines = 3
    .DropDownWidth = 75
    .Tag = "MDCOMCMDLINE"
End With

    If oApp.Version <> "9.0" Then
        Set btnFormulaWatch = .Controls.Add(msoControlButton, 5687)
                With btnFormulaWatch
                    .BeginGroup = True
                End With
    End If

Set btnHelp = .Controls.Add(msoControlPopup)
With btnHelp
    .BeginGroup = True
    .Caption = "Help"
    .DescriptionText = "Help"
    .Tag = "MDHELP"
    .ToolTipText = "Get Help."
    .Visible = True

        Set btnMDCOMHelp = .Controls.Add(1)
        With btnMDCOMHelp
            .Caption = "MD*ReX Help"
            .ToolTipText = "Online Help"
            .DescriptionText = "MDReXHelp"
            .OnAction = "!<" & oAddin.ProgId & ">"
            .Tag = "MDCOMHELP"
            .Visible = True
        End With

        Set btnXPLHelp = .Controls.Add(1)
        With btnXPLHelp
            .Caption = "XploRe APSS"
            .ToolTipText = "Auto Pilot Support System (APSS)"
            .DescriptionText = "XPLHelp"
            .OnAction = "!<" & oAddin.ProgId & ">"
            .Tag = "MDCOMXPLHELP"
            .Visible = True
        End With

        Set btnAbout = .Controls.Add(msoControlPopup)
        With btnAbout
            .BeginGroup = True
            .Caption = "About"
            .DescriptionText = "About"
            .Tag = "MDAbout"
            .ToolTipText = "About MD*ReX & XploRe"
            .Visible = True

                Set btnAboutMDCOM = .Controls.Add(1)
```

```
                                With btnAboutMDCOM
                                    .Caption = "MD*ReX"
                                    .ToolTipText = "MD*ReX"
                                    .DescriptionText = "AboutMDREX"
                                    .OnAction = "!<" & oAddin.ProgId & ">"
                                    .Tag = "MDCOMABOUT1"
                                    .Visible = True
                                End With

                                Set btnAboutXPL = .Controls.Add(1)
                                With btnAboutXPL
                                    .Caption = "XploRe"
                                    .ToolTipText = "XploRe"
                                    .DescriptionText = "AboutMDXPL"
                                    .OnAction = "!<" & oAddin.ProgId & ">"
                                    .Tag = "MDCOMABOUT2"
                                .Visible = True
                                End With
                            End With
                        End With
                    End With
                    dbgStr = dbgstar & dbgStr & "creating menu bar succeeded" & vbCrLf
                End If
    End If
Call WriteDBGString(dbgStr, App.EXEName & ".log")
dbgStr = ""
UpdateViewsMenu
End Sub

Public Sub MDCOMDisconnect()
    On Error GoTo MDCOMDisconnectErr
    dbgStr = dbgstar & "Entering GUI setup MDCOMDisconnect" & vbCrLf
    DestroyToolBars
    Set myFrmXPLDirect = Nothing
    Set cbMDCOM = Nothing
    Set oHostApp = Nothing
    dbgStr = dbgstar & "Deleting GUI finished" & vbCrLf
    Exit Sub

MDCOMDisconnect_exit:
        Exit Sub

MDCOMDisconnectErr:
        Select Case Err.Number
            Case Err.Number = 0
                GoTo MDCOMDisconnect_exit:
            Case Err.Number <> 0
                MsgBox Err.Number & " " & Err.Description, vbCritical, "DestroyToolBar"
                GoTo MDCOMDisconnect_exit:
        End Select
End Sub

Private Sub DestroyToolBars()
    On Error GoTo DestroyToolBarsErr:
    dbgStr = dbgstar & dbgStr & "Will destroy ToolBar now" & vbCrLf
    cbMDCOM.Delete

DestroyToolBars_exit:
        dbgStr = dbgstar & dbgStr & "Successfully destroyed ToolBar" & vbCrLf
        Exit Sub

DestroyToolBarsErr:
        Select Case Err.Number
            Case Err.Number = 0
                GoTo DestroyToolBars_exit:
            Case Err.Number <> 0
                MsgBox Err.Number & " " & Err.Description, vbCritical, "DestroyContextMenu"
                GoTo DestroyToolBars_exit:
        End Select
End Sub

Private Sub btnAboutMDCOM_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)

    If oHostInst = "Microsoft Word" Then
    With myFrmSplash
        .Caption = "MD*ReX"
        .lblCompanyProduct.Caption = "MD*ReX Microsoft Office COM Add-in"
        .lblVersion = "MD*ReX Version " & Format(App.Major, "#0") & "." & Format(App.Minor, "#00")
              & "." & Format(App.Revision, "#0000")
        .lblCopyright = App.LegalCopyright
        .txtSplash.ToolTipText = "MD*ReX"
        .txtSplash.Text = "MD*COM guid: " & oAddinInst.Guid & vbCrLf & "MD*COM says..." & vbCrLf &
              "Path to MD*SERV: " & App.Path & "\mdserv" & vbCrLf & "Operating System: " &
             oHostDOC.System.OperatingSystem & _
              vbCrLf & "Host: " & oHostDOC.Name _
            & " " & oHostDOC.Version & " Build: " & oHostDOC.Build & vbCrLf & "MD*Crypt says... "
                & _
            vbCrLf & "Server Status" & p_OfficeCrypt.propServerStatus & vbCrLf & "MD*Crypt version
                " _
```

```
            & p_OfficeCrypt.MDReXInfoServer.getMdCryptVersion & vbCrLf & "MD*Crypt Java " _
            & p_OfficeCrypt.MDReXInfoServer.getMdCryptJava & vbCrLf & "MD*XQS build date: " _
            & p_OfficeCrypt.MDReXInfoServer.serverBuildDate
        .txtSplash.Visible = True
        .txtSplash.Locked = True
        .Show
    End With
    Else
        With myFrmSplash
            strResults = p_OfficeCrypt.MDReXInfoServer.serverBuildDate
            .lblCopyright = App.LegalCopyright
            .lblCompanyProduct.Caption = "MD*ReX Microsoft Office COM Add-in"
            .lblVersion = "MD*ReX " & Format(App.Major, "#0") & "." & Format(App.Minor, "#00") &
                 "." & Format(App.Revision, "#0000")
            .txtSplash2.ToolTipText = "MD*ReX"
            .txtSplash2.Text = "MD*COM guid: " & oAddinInst.Guid & vbCrLf & "MD*COM says..." &
                 vbCrLf & "Path to MD*SERV: " & App.Path & "\mdserv" & vbCrLf & "Operating System:
                  " & _
            oHostInst.OperatingSystem & vbCrLf & "Host: " & oHostInst.Value & " " _
            & oHostInst.Version & " Build: " & oHostInst.Build & vbCrLf & "MD*Crypt says... " & _
            vbCrLf & "Server Status: " & p_OfficeCrypt.propServerStatus & vbCrLf & "MD*Crypt
                 version " _
            & p_OfficeCrypt.MDReXInfoServer.getMdCryptVersion & vbCrLf & "MD*Crypt Java: " _
            & p_OfficeCrypt.MDReXInfoServer.getMdCryptJava & vbCrLf & "MD*Serv build: " _
            & p_OfficeCrypt.MDReXInfoServer.getMdServVersion & vbCrLf _
            & "MD*XQS Server ID: " _
            & p_OfficeCrypt.MDReXInfoServer.getMyMdServId & vbCrLf _
            & "MD*XQS Server System: " _
            & p_OfficeCrypt.MDReXInfoServer.getServerSystem & vbCrLf _
            & "MD*XQS Server Build: " _
            & p_OfficeCrypt.MDReXInfoServer.getServerBuild & vbCrLf _
            & "Java SDK Version: " _
            & p_OfficeCrypt.MDReXInfoServer.getMdServJava & vbCrLf _
            & "MD*XQS Server Build Date: " _
            & strResults
            .txtSplash2.Visible = True
            .txtSplash2.Locked = True
            .Command3.Caption = "Ok"
            .Command4.Caption = "www.md-rex.com"
            .Show
        End With
    End If
End Sub

Private Sub btnAboutXPL_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    Dim Browser As Object
    On Error Resume Next
    Set Browser = CreateObject("internetexplorer.application")
    Browser.navigate "http://www.xplore-stat.de/"
    With Browser
        .StatusBar = False
        .MenuBar = False
        .Toolbar = 1
        .Visible = True
    End With
End Sub

Private Sub btnGet_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    On Error Resume Next
    myPutGet.GetXPL
End Sub

Private Sub btnMDCOMHelp_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    MsgBox "FUNCTION MD*ReXHELP"
End Sub

Private Sub btnPut_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    On Error Resume Next
    myPutGet.PutXpl
End Sub

Private Sub btnWWWAPSS_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
'
End Sub

Private Sub btnWWWMDCOM_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
'
End Sub

Private Sub btnWWWXPL_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
'
End Sub

Private Sub btnXPL_xla_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    Dim i%
    Dim xlaString As String
    On Error GoTo XLAErr
        GetLocalAddins (App.Path & "\mdrexxla\*.xla")
```

```
        xlaString = "Available MD*ReX Excel AddIns (total " & UBound(xlaNameArray) & "): " &
             vbCrLf
        With myFrmXLA
            .List1.Clear
            For i = 1 To UBound(xlaNameArray)
                .List1.AddItem (xlaNameArray(i))
            Next i
            .Show vbModal
        End With

Exit Sub

XLAErr:
    Call ExeErr(Err)
    Exit Sub

End Sub

Private Sub btnXPLDirect_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    On Error GoTo btnXPLDirect_Click_Err
    myFrmXPLDirect.Show
    Exit Sub

btnXPLDirect_Click_Err:
    Exit Sub
End Sub

Private Sub btnXPLHelp_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    Dim Browser As Object
    On Error Resume Next
    Set Browser = CreateObject("internetexplorer.application")
    Browser.navigate ("http://www.xplore-stat.de/help/_Xpl_Start.html")
    With Browser
        .StatusBar = True
        .MenuBar = False
        .Toolbar = 1
        .Visible = True
    End With
End Sub

Private Sub btnXPLFunctions_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    On Error Resume Next
    With myFrmFunctions
        .txtFunctions = FunctionString
        .Show vbModal
    End With
End Sub

Private Sub btnXPLLibraries_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    Dim i%
    Dim libString As String
    On Error Resume Next
    With myFrmLibsLocal
        If (p_OfficeCrypt.propServerIP = "localhost") Then
            GetLocalLibList (App.Path & "\mdserv\lib\*.lib")
            libString = "Available Libraries (total " & UBound(libNameArray) & "): " & vbCrLf
            For i = 1 To UBound(libNameArray)
                libString = libString & libNameArray(i) & vbCrLf
            Next i
        Else
            MsgBox "Only supported when connected locally"
        End If
    .txtLibsLocal = libString
    .Show vbModal
    End With
End Sub

Private Sub btnXPLLogs_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    On Error Resume Next
    With myFrmStatus
        Call FileToTextBox(.txtProtLog, "C:\" & "prot.log")
        Call FileToTextBox(.txtMDRexLog, App.Path & "\debug\" & App.EXEName & ".log")
        .Show vbModal
    End With
End Sub

Private Sub btnXPLNamedRanges_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As
     Boolean)
Dim i%
    On Error Resume Next
    With myFrmNamedRanges
        If (CounterRuns > 0) Then
                For i = 0 To UBound(myMapper.XLSObject)
                    .listNamedRanges.AddItem (myMapper.XLSObjectName(i) & ": " & myMapper.
                         XLSObject(i) & " XPLORE: " & myMapper.XPLObject(i))
                Next i
        End If
    .Show vbModal
    End With
```

```
End Sub

Private Sub btnXPLObjects_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    On Error Resume Next
    With myFrmObjects
        .txtObjects = ObjectString
        .Show vbModal
    End With
End Sub

Private Sub btnXPLQuantlets_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    Dim i%
    Dim qString As String
    On Error Resume Next
    With myFrmQuantlets
        If (p_OfficeCrypt.propServerIP = "localhost") Then
            GetLocalQList (App.Path & "\mdserv\lib\*.xpl")
            qString = "Available Quantlets (total " & UBound(qNameArray) & "): " & vbCrLf
            For i = 1 To UBound(qNameArray)
                qString = qString & qNameArray(i) & vbCrLf
            Next i
            .txtQuantlets = qString
            .Show vbModal
        Else
            .txtQuantlets = QuantletString
            .Show vbModal
        End If
    End With
End Sub

Private Sub btnXQScrap_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
Dim i%
On Error GoTo CatchErr
    If btnXQScrap.Caption = "Connect" Then
        Call LoadAddin(True, "XploReGetResult.xla", "go") '//default add_in, always loaded
        myFrmConnect.Show vbModal
    Else
        p_OfficeCrypt.clsTerminate
        UpdateViewsMenu
        UpdateViewsContext
        If (oHostInst <> "Microsoft Word") Then
        With oHostInst
        End With
        ReDim myMapper.XLSObject(0)
        ReDim myMapper.XPLObject(0)
        myMapper.XPLObjectCount = 0
        GetLocalAddins (App.Path & "\mdrexxla\*.xla")
        If xlaNameArray(1) <> "" Then
            If (UBound(xlaNameArray) > 0) Then
                For i = 1 To UBound(xlaNameArray)
                    Call LoadAddin(False, "\mdrexxla\" & xlaNameArray(i)) '//unload custom addins
                Next i
            End If
        End If
        Call LoadAddin(False, "XploReGetResult.xla") '//unload default addin
        End If
        ReDim xlaNameArray(0)
    End If

Exit Sub

CatchErr:
    Call ExeErr(Err)
    Exit Sub

End Sub

Private Sub Class_Terminate()
    If Not oHostApp Is Nothing Then MDCOMDisconnect
End Sub

Private Sub cmMDCOM1_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
'
End Sub

Private Sub cmMDCOM2_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    MsgBox "FUNCTION RUN"
End Sub

Private Sub cmMDCOM3_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    MsgBox "FUNCTION GET"
End Sub

Public Sub SyncBox(box As Office.CommandBarComboBox)
    Set btnCMDLine = box
End Sub
```

B.12 clsMDCryt.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "clsMDCrypt"
Attribute VB_GlobalNameSpace = True
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Public MDReXServer          As mdcrypt.RexServer
Public MDReXInfoServer      As mdcrypt.XQSInfoObject
Public MDReXCryptVersion    As mdcrypt.MdCryptVersion
Public ConnectedToServer    As Boolean
Public tempResult           As Variant
Public finalResult          As String
Public xlPath               As String
Public ReXDisplay           As Object
Public ReXGraphic           As Object


Private Sub Class_Initialize()
    On Error GoTo Class_Ini_Err:
    Set MDReXServer = New mdcrypt.RexServer
    Set MDReXInfoServer = New mdcrypt.XQSInfoObject
    Set MDReXCryptVersion = New mdcrypt.MdCryptVersion
Exit Sub

Class_Ini_Err:
    Select Case Err.Number
    Case 0
        GoTo 0
    Case Else
        MsgBox Err.Number & " " & Err.Description, vbCritical, "MDCrypt Class_Initialize() Error"
        Exit Sub
    End Select
0: End Sub

Private Sub Class_Terminate()
    Set MDReXServer = Nothing
    Set MDReXInfoServer = Nothing
    Set MDReXCryptVersion = Nothing
End Sub

Public Property Get propReXServer() As String
propReXServer = MDReXServer
End Property

Public Property Get propClientReady() As String
propClientReady = MDReXServer.clientReady
End Property

Public Property Get propdblMatrix() As Object
Set propdblMatrix = MDReXServer.dblMatrix
End Property

Public Property Get propInfo() As String
propInfo = MDReXServer.info
End Property

Public Property Get propmatrixCols() As Long
propmatrixCols = MDReXServer.matrixCols
End Property

Public Property Get propmatrixRows() As Long
propmatrixRows = MDReXServer.matrixRows
End Property

Public Property Get propmatrixName() As String
propmatrixName = MDReXServer.Matrixname
End Property

Public Property Get propResult() As String
propResult = MDReXServer.result
End Property

Public Property Get propServerIP() As String
propServerIP = MDReXServer.serverIP
End Property
```

```
Public Property Get propServerPort() As Long
propServerPort = MDReXServer.serverPort
End Property

Public Property Get propServerStatus() As Long
propServerStatus = MDReXServer.serverStatus
End Property

Public Property Get propMDCryptVersion() As String
propMDCryptVersion = MDReXInfoServer.MdCryptVersion
End Property

Public Property Get propMDCryptJavaVersion() As String
propMDCryptJavaVersion = MDReXInfoServer.mdCryptJava
End Property

Public Property Get propMDCryptConectionError() As String
propMDCryptConectionError = MDReXServer.connectionError
End Property

Public Property Get propMDCryptGraphic(Index As Long) As Object
propMDCryptGraphic = MDReXServer.graphic(Index)
End Property

Public Property Get propMDCryptNumberOfGraphics() As Object
propMDCryptNumberOfGraphics = MDReXServer.numberOfGraphics
End Property

Public Property Get propMDCryptNumberOfDisplays() As Object
propMDCryptNumberOfDisplays = MDReXServer.numberOfDisplays
End Property

Public Sub SetServerIP(sip As String)
    MDReXServer.SetServerIP (sip)
End Sub

Public Sub SetServerPort(spt As Long)
    MDReXServer.SetServerPort (spt)
End Sub

Public Function clsConnect() As Boolean
    Dim i%

    While propServerStatus = 1006
        clsConnect = MDReXServer.Connect
        If Len(propMDCryptConectionError) <> 0 Then
                MsgBox "A timeout occurred. Trying to reconnect!" & vbCrLf & "MD*Crypt: " &
                     propMDCryptConectionError
                For i = 0 To 1
                MDReXServer.Connect
                Next i
        End If
        If clsConnect = False Then
                Exit Function
            End If
    Wend

    If clsConnect = True And propServerStatus = 1003 Then
        ConnectedToServer = clsConnect
    Else: If ConnectedToServer = False And propServerStatus <> 1003 Then ConnectedToServer =
         clsConnect
        MsgBox Err.Number
    End If

End Function

Public Sub clsTerminate()
    While propServerStatus <> 1006
        MDReXServer.Terminate
    Wend

    If propServerStatus = 1006 Then
        ConnectedToServer = False
    Else
        ConnectedToServer = True
        Debug.Print Err.Number
    End If
    'to be sure
    MDReXServer.Terminate
End Sub

Public Function clsBruteTerminate() As Boolean
    While propServerStatus <> 1006
        MDReXServer.Terminate
    Wend

    If propServerStatus = 1006 Then
        ConnectedToServer = False
    Else
```

```
        ConnectedToServer = True
        Debug.Print Err.Number
    End If

    If propServerStatus = 1006 Then
        clsBruteTerminate = True
    Else
        clsBruteTerminate = False
    End If
End Function

Public Function clsSendQuantlet(ByVal Quantlet As String) As Boolean
clsSendQuantlet = MDReXServer.sendQuantlet(Quantlet)
RenderGraphic
End Function

Public Function clsGetResult() As String
tempResult = MDReXServer.getResultEncoded("")
finalResult = Replace(tempResult, Chr$(10), Chr$(13) & Chr$(10))
finalResult = Replace(finalResult, Chr$(13) & Chr$(13), Chr$(13))
clsGetResult = finalResult
End Function

Public Function GetServerStatus() As String
GetServerStatus = MDReXServer.GetServerStatus()
Select Case MDReXServer.GetServerStatus
  Case 1000
  GetServerStatus = GetServerStatus & Chr(32) & "(Socket initialized)"
  Case 1001
  GetServerStatus = GetServerStatus & Chr(32) & "(Handshake done)"
  Case 1002
  GetServerStatus = GetServerStatus & Chr(32) & "(Connection accepted)"
  Case 1003
  GetServerStatus = GetServerStatus & Chr(32) & "(Server ready)"
  Case 1004
  GetServerStatus = GetServerStatus & Chr(32) & "(Server busy)"
  Case 1005
  GetServerStatus = GetServerStatus & Chr(32) & "(Server waiting)"
  Case 1006
  GetServerStatus = GetServerStatus & Chr(32) & "(Not connected)"
  ConnectedToServer = False
End Select
End Function

Public Function SendDoubleMatrix(Matrixname As String, Matrix, ByVal Cols As Long, ByVal Rows As
     Long) As Boolean
Dim Dims
Dim done As Boolean
Dim iCols As Long
Dim iRows As Long
Dim tempArray
    Call MDReXServer.wipeMatrix
    Dims = MDReXServer.setDims(Rows, Cols)
    MDReXServer.setMatrixName (Matrixname)

    On Error GoTo SendDoubleMatrixErr:

    Select Case (IsArray(Matrix))
        Case False
            If IsNumeric(Matrix) = False Or IsEmpty(Matrix) = True Then
                Call MDReXServer.setdblNaN(Rows - 1, Cols - 1)
            Else
                Call MDReXServer.setDblElement(CDbl(Matrix), 0, 0)
            End If
        Case Else
            For iCols = 0 To Cols - 1
                For iRows = 0 To Rows - 1
                oHostInst.StatusBar = "Parsing cell value: " & Matrix(iRows + 1, iCols + 1) & " @
                     position: " & iRows & "/" & iCols
                    If IsNumeric(Matrix(iRows + 1, iCols + 1)) = False Or IsEmpty(Matrix(iRows +
                         1, iCols + 1)) = True Then
                        Call MDReXServer.setdblNaN(iRows, iCols)
                    Else
                        Call MDReXServer.setDblElement(CDbl(Matrix(iRows + 1, iCols + 1)), iRows,
                             iCols)
                    End If
                Next iRows
            Next iCols
        End Select

    SendDoubleMatrix = MDReXServer.sendMatrix
    Call WriteDBGString("SendDoubleMatrix with params: " & Matrixname & " (matrixname) " & Cols &
         " (cols) " & Rows & " (rows) " & "succeeded." & vbCrLf, App.EXEName & ".log")
    oHostInst.StatusBar = "Put finished successfully!"
    Call MDReXServer.wipeMatrix

Exit Function

SendDoubleMatrixErr:
```

```
    Select Case Err.Number
    Case 13
        MsgBox Err.Number & " " & Err.Description & vbCrLf & "Single Cells not allowed!" & vbCrLf
             & "Use XploRe Direct instead!", vbCritical, "SendDoubleMatrixError"
        Exit Function
    Case 0
        Call MDReXServer.wipeMatrix
        GoTo 0
    Case Else
        MsgBox Err.Number & " " & Err.Description, vbCritical, "SendDoubleMatrixError"
        Call WriteDBGString("SendDoubleMatrix with params: " & Matrixname & " (matrixname) " &
             Cols & " (cols) " & Rows & " (rows) " & "failed." & vbCrLf, App.EXEName & ".log")
        Exit Function
    End Select
0: End Function

Public Function GetDoubleMatrix(DblMatrixName As String) As Variant
Dim tmpArray() As Double
Dim Klaus As Boolean
On Error GoTo ErrHndl:

    Call MDReXServer.wipeMatrix
    oHostInst.StatusBar = "Getting double element " & DblMatrixName & "! Dimensions of " &
         DblMatrixName & ": " & MDReXServer.GetDoubleMatrix(DblMatrixName).Cols & "x" &
         MDReXServer.GetDoubleMatrix(DblMatrixName).Rows
    Klaus = MDReXServer.getMatrix(DblMatrixName)
    Set GetDoubleMatrix = MDReXServer.getDblMatrix

    If Klaus = False Then
    GoTo ErrHndl
    End If

Exit Function

ErrHndl: MsgBox "MD*COM GetDoubleMatrix(): " & vbCrLf & "Could not get " & DblMatrixName & " " &
     vbCrLf & Err.Description & vbCrLf & "[" & Err.Number & "]"
    Set GetDoubleMatrix = Err
    Exit Function

End Function

Public Function Matrixname()
Matrixname = MDReXServer.getMatrixName
End Function

Public Function RenderGraphic() As Object
Dim i As Long
On Error GoTo ErrHndl:

Select Case (MDReXServer.getNumberOfDisplays)
    Case (0)
        Debug.Print "RenderGraphic: # of Displays: "; MDReXServer.getNumberOfDisplays
        GoTo SelectHndl:
    Case Else
        Debug.Print "RenderGraphic: # of Displays: "; MDReXServer.getNumberOfDisplays
        For i = 0 To (MDReXServer.getNumberOfDisplays - 1)
            If ReXDisplay Is Nothing Then
                Set ReXDisplay = MDReXServer.getDisplay(i)
                Debug.Print "RenderGraphic_ReXDisplay Cols (" & ReXDisplay.Cols & "), Rows (" &
                     ReXDisplay.Rows & "), ID (" & ReXDisplay.id & "), Type (" & ReXDisplay.Type &
                      ")" & vbCrLf
            End If
        Next i
End Select

Select Case (MDReXServer.getNumberOfGraphics)
    Case (0)
        Debug.Print "RenderGraphic: # of Graphics: "; MDReXServer.getNumberOfGraphics
        GoTo SelectHndl:
    Case Else
        For i = 0 To (MDReXServer.getNumberOfGraphics - 1)
            Set ReXGraphic = MDReXServer.getGraphic(i)
            Debug.Print "Attributes of Graphic " & ReXGraphic.displayID & " (displayID)." & vbCrLf
                  & "Rows: " & ReXGraphic.Row & vbCrLf & "Cols: " & ReXGraphic.Col & vbCrLf & "Dim
                 : " & ReXGraphic.dim
        Next i
End Select

SelectHndl:
    Set RenderGraphic = Nothing
    Exit Function

ErrHndl:
        Debug.Print "MD*COM getGraphic(): " & vbCrLf & "Error occured! " & vbCrLf & Err.
             Description & vbCrLf & "[" & Err.Number & "]"
        Set RenderGraphic = Err
        Exit Function
End Function
```

B.13 clsPutGet.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "clsPutGet"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Public Sub GetXPL()
Dim Varname As String
Dim PutRange As Range, r1 As Range, r2 As Range
Dim TempVariantArray()
Dim myCols As Long, myRows As Long
Dim Klaus As Boolean
Dim rCount As Long
Dim cCount As Long
Dim tempRows As Long
Dim tempCols As Long
Dim i As Long, j As Long, k As Long, l As Long
Dim dummy As String

Dim rngTemp         As Range
Dim strValue        As String
Dim strRangeName    As String
Dim strAddress      As String
Dim strFormula      As String
Dim TmpComment As Excel.Comment

    dummy = "Sorry object doesn't exsist at XQS"
    Varname = InputBox("Name of XploRe Object: ", "Get")

    If Varname = "" Then
        MsgBox "You Specified an empty Object name!"
        Exit Sub
    End If

    On Error GoTo GetXPLErr:
    Call WriteDBGString("Going to get " & Varname & " from XQS." & vbCrLf, App.EXEName & ".log")
    TempVariantArray() = p_OfficeCrypt.GetDoubleMatrix(Varname).elements
    tempRows = UBound(TempVariantArray, 1)
    tempCols = UBound(TempVariantArray, 2)
    Call WriteDBGString("#ROWS: " & vbTab & tempRows & vbCrLf, App.EXEName & ".log")
    Call WriteDBGString("#COLS: " & vbTab & tempCols & vbCrLf, App.EXEName & ".log")

    If tempCols > 255 Then
        MsgBox "Sorry your matrix has more than 255 Columns!" & vbCrLf _
        & "MS Excel does not support more than 255 Columns in a single worksheet." & vbCrLf _
        & "Your matrix has " & tempCols & " columns."
        UpdateViews
        Exit Sub
    End If

    If tempCols = 0 Then
        Set PutRange = oHostInst.Selection
        Set PutRange = PutRange.Cells(1, 1)
        Set PutRange = oHostInst.Range(PutRange, PutRange.Offset(tempRows, tempCols))
        Call WriteDBGString("just got" & Varname & " from XQS." & vbCrLf, App.EXEName & ".log")
            If (IsRangeEmpty(PutRange) = True) Then
                PutRange.Value = TempVariantArray
                strAddress = PutRange.Address & vbCrLf & PutRange.AddressLocal
                Call WriteDBGString("placeing " & Varname & " to " & PutRange.AddressLocal &
                     vbCrLf, App.EXEName & ".log")
            Else
                MsgBox "The Range you selected is not empty. Select another range"
                Call WriteDBGString("placeing " & Varname & " not done. Range not empty!" & vbCrLf
                     , App.EXEName & ".log")
                Exit Sub
            End If
    Else
        Set PutRange = oHostInst.Selection
        Set PutRange = PutRange.Cells(1, 1)
        Set PutRange = oHostInst.Range(PutRange, PutRange.Offset(tempRows, tempCols))
        Call WriteDBGString("just got" & Varname & " from XQS." & vbCrLf, App.EXEName & ".log")
            If (IsRangeEmpty(PutRange) = True) Then
                PutRange.Value = TempVariantArray
                strAddress = PutRange.Address & vbCrLf & PutRange.AddressLocal
                Call WriteDBGString("placeing " & Varname & " to " & PutRange.AddressLocal &
                     vbCrLf, App.EXEName & ".log")
```

```
            Else
                MsgBox "The Range you selected is not empty. Select another range"
                Call WriteDBGString("placeing " & Varname & " not done. Range not empty!" & vbCrLf
                        , App.EXEName & ".log")
                Exit Sub
            End If
    End If
    CountRuns
    Call Map(PutRange, Varname, CounterRuns)
    Set PutRange = Nothing
Exit Sub

GetXPLErr:
    Select Case Err.Number
        Case -2147467259
            MsgBox "The object name you specified [" & Varname & "] seems not to exist at the XQS
                .", vbCritical, dummy
        Case 0
            GoTo 0
        Case Else
            MsgBox Err.Number & " " & Err.Description, vbCritical
    End Select
    Exit Sub

0: End Sub

Public Sub PutXpl()
Dim Varname As String
Dim Matrix
Dim ValRange As Range
Dim tempVal As Object
Dim i As Long, j As Long
Dim RowNum As Long, ColNum As Long
    On Error GoTo PutXpl_Err
1:     Varname = InputBox("Name of XploRe Object: ", "Put")

    If Varname = "" Then
        MsgBox "You Specified an empty Object name!"
        Exit Sub
    End If

    Select Case CounterRuns
        Case 0
            GoTo 2:
        Case Else
            For i = 0 To UBound(myMapper.XPLObject)
                If Varname = myMapper.XPLObject(i) Then
                    If MsgBox(Varname & " already exists at server! Overwrite?", vbOKCancel) =
                          vbCancel Then GoTo 1:
                End If
            Next i
    End Select

2:     Set ValRange = oHostInst.Selection

    If IsEmpty(ValRange) Then
            MsgBox "You haven't selected a cell range for the object" & vbCrLf & "or the selected
                range is empty!"
        Exit Sub
    End If
    RowNum = ValRange.Rows.Count
    ColNum = ValRange.Columns.Count
    Matrix = ValRange
    Call WriteDBGString("Going to put " & Varname & " to XQS." & vbCrLf, App.EXEName & ".log")
    Call WriteDBGString("#ROWS: " & vbTab & RowNum & vbCrLf, App.EXEName & ".log")
    Call WriteDBGString("#COLS: " & vbTab & ColNum & vbCrLf, App.EXEName & ".log")
    Call p_OfficeCrypt.SendDoubleMatrix(Varname, Matrix, ColNum, RowNum)
    UpdateViews
    CountRuns
    Call Map(ValRange, Varname, CounterRuns)
Exit Sub

PutXpl_Err:
    Call ExeErr(Err)
    Exit Sub
End Sub
```

B.14 clsXPL2XLS.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "clsXPL2XLS"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit

Public Type XPL2XLSMapper
    XPLObject() As String
    XLSObject() As String
    XLSObjectName() As String
    XPLObjectCount As Long
End Type

Public Type LibQuantMapper
    QuantletName() As String
    QuantletLibrary() As String
    InputArgs() As String
    OutputArgs() As String
    OptionalArgs() As String
End Type
```

B.15 dsrExcel11.Dsr

```
VERSION 5.00
Begin {AC0714F6-3D04-11D1-AE7D-00A0C90F26F4} dsrExcel11
   ClientHeight     =   12315
   ClientLeft       =   0
   ClientTop        =   0
   ClientWidth      =   19140
   _ExtentX         =   33761
   _ExtentY         =   21722
   _Version         =   393216
   Description      =   "XploRe COM client"
   DisplayName      =   "MD*ReX 2004"
   AppName          =   "Microsoft Excel"
   AppVer           =   "Microsoft Excel 11.0"
   LoadName         =   "Startup"
   LoadBehavior     =   3
   RegLocation      =   "HKEY_CURRENT_USER\Software\Microsoft\Office\Excel"
   RegInfoCount     =   1
   RegType0         =   1
   RegKeyName0      =   "Author"
   RegSData0        =   ""
End
Attribute VB_Name = "dsrExcel11"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = True
Option Explicit
'//public members declared in mdlMDRex2004.bas

Private WithEvents p_mctlBtnEvents                          As Office.CommandBarButton
Attribute p_mctlBtnEvents.VB_VarHelpID = -1
Private WithEvents p_mcomboEvents                           As Office.CommandBarComboBox
Attribute p_mcomboEvents.VB_VarHelpID = -1

Private Declare Function GetForegroundWindow Lib "user32" () As Long
Dim hwndXLS                                                 As Long
Dim sProgID                                                 As String
Dim myCommandBar                                            As Office.CommandBar

Private Sub AddinInstance_Initialize()
    On Error GoTo DSR_Err
    Set myPutGet = New clsPutGet
    Set myFile = New FileSystemObject
    Call CreateDBGFolder

    blnRexToggled = False
    Set myFrmSplash = New frmSplashNew
    Set myFrmGetResult = New frmGetResult
    Set myFrmFunctions = New frmFunctions
    Set myFrmLibsLocal = New frmLibsLocal
    Set myFrmNamedRanges = New frmNamedRanges
    Set myFrmObjects = New frmObjects
    Set myFrmQuantlets = New frmQuantlets
    Set myFrmQuantsLocal = New frmQuantsLocal
    Set myFrmStatus = New frmStatus
    Set myFrmXLA = New frmXLA

    Set p_OfficeCrypt = New clsMDCrypt
    With p_OfficeCrypt
        str_InfoMDCRYPT = .MDReXServer
        str_InfoMDCRYPT = "MD*Crypt instance: " & str_InfoMDCRYPT & vbCrLf & "XQS Status: " & .
            propServerStatus
    End With
    Call WriteDBGString(App.EXEName & " Initialize() @ " & Date & " " & Time & vbCrLf, App.EXEName
        & ".log")
    Call WriteDBGString("XploRe Middleware " & str_InfoMDCRYPT & vbCrLf, App.EXEName & ".log")

    TriggerMDServ
Exit Sub

DSR_Err:
    Call ExeErr(Err)
End Sub

Private Sub AddinInstance_OnAddInsUpdate(custom() As Variant)
    On Error GoTo DSR_Err
    If p_OfficeCrypt Is Nothing Then
        Set p_OfficeCrypt = New clsMDCrypt
    End If
    Call WriteDBGString(App.EXEName & " OnAddInsUpdate() @ " & Date & " " & Time & vbCrLf, App.
        EXEName & ".log")
Exit Sub

DSR_Err:
    Call ExeErr(Err)
```

```
End Sub

Private Sub AddinInstance_OnBeginShutdown(custom() As Variant)
    On Error GoTo DSR_Err
    Call WriteDBGString(App.EXEName & " OnBeginShutdown() @ " & Date & " " & Time & vbCrLf, App.
         EXEName & ".log")
Exit Sub

DSR_Err:
    Call ExeErr(Err)
End Sub

Private Sub AddinInstance_OnConnection(ByVal Application As Object, ByVal ConnectMode As
     AddInDesignerObjects.ext_ConnectMode, ByVal AddInInst As Object, custom() As Variant)
    On Error GoTo DSR_Err
    Call WriteDBGString(App.EXEName & " OnConnection() (caller: " & Application & ") @ " & Date &
         " " & Time & vbCrLf, App.EXEName & ".log")

    Set oHostInst = Application
    Set p_AddinInst = AddInInst
    If (oHostInst.Version <> "11.0") Then MsgBox "This COM AddIn has only been tested with MSExcel
          v.11 (Excel2003)." & vbCrLf & "Though it should be downward compatible" & vbCrLf & "it
         is not guaranteed that this AddIn " & vbCrLf & "will work as intended with earlier
         versions." & vbCrLf & "Please update Excel.", vbInformation, App.EXEName
    Call WriteDBGString(App.EXEName & " called from: " & oHostInst & " v." & oHostInst.Version &
         vbCrLf, App.EXEName & ".log")
    str_InfoMDCOM = p_AddinInst.Description & " On Connection" & vbCrLf & "GUID: " & p_AddinInst.
         Guid & vbCrLf & "ProgID: " & p_AddinInst.ProgId
    Select Case ConnectMode
        Case ext_cm_AfterStartup
            Call WriteDBGString(str_InfoMDCOM & vbCrLf & str_InfoMDCRYPT & App.EXEName & "
                 initialized! Connect mode: " & ConnectMode & "(ext_cm_AfterStartup)" & vbCrLf,
                 App.EXEName & ".log")
        Case ext_cm_CommandLine
            Call WriteDBGString(str_InfoMDCOM & vbCrLf & str_InfoMDCRYPT & App.EXEName & "
                 initialized! Connect mode: " & ConnectMode & "(ext_cm_CommandLine)" & vbCrLf, App
                 .EXEName & ".log")
        Case ext_cm_External
            Call WriteDBGString(str_InfoMDCOM & vbCrLf & str_InfoMDCRYPT & App.EXEName & "
                 initialized! Connect mode: " & ConnectMode & "(ext_cm_External)" & vbCrLf, App.
                 EXEName & ".log")
        Case ext_cm_Startup
            Call WriteDBGString(str_InfoMDCOM & vbCrLf & str_InfoMDCRYPT & App.EXEName & "
                 initialized! Connect mode: " & ConnectMode & "(ext_cm_Startup)" & vbCrLf, App.
                 EXEName & ".log")
    End Select

    AddInInst.object = Me '//obtain reference to this == Me
    'obtain reference to Excel, MD*Crypt and window handle for Excel
    'Excel is assumed to be in the foreground
    Call SaveHostApp(Application, AddInInst)
    hwndXLS = GetForegroundWindow()
    On Error Resume Next
    sProgID = AddInInst.ProgId
    Set p_mctlBtnEvents = CreateAddInCommandBarButton(Application, ConnectMode, AddInInst)
    'DeleteAddIns
    Call LoadAddin(False, "XploReGetResult.xla") '//unload default addin

Exit Sub

DSR_Err:
    Call ExeErr(Err)
End Sub

Private Sub AddinInstance_OnDisconnection(ByVal RemoveMode As AddInDesignerObjects.
     ext_DisconnectMode, custom() As Variant)
Dim i%
    On Error GoTo DSR_Err
    Call WriteDBGString(App.EXEName & " OnDisconnection() @ " & Date & " " & Time & vbCrLf, App.
         EXEName & ".log")
    RemoveAddInCommandBarButton RemoveMode

    GetLocalAddins (App.Path & "\mdrexxla\*.xla")
    If xlaNameArray(1) <> "" Then
        If (UBound(xlaNameArray) > 0) Then
            For i = 1 To UBound(xlaNameArray)
                DeleteAddIns (xlaNameArray(i)) '//unload custom addins
            Next i
        End If
    End If
    DeleteAddIns ("Xploregetresult")
    ReDim xlaNameArray(0)

    If p_OfficeCrypt.ConnectedToServer Then Call WriteDBGString(App.EXEName & " is still connected
         during OnDisconnection()" & vbCrLf, App.EXEName & ".log")
    p_OfficeCrypt.clsTerminate

Exit Sub
```

```
DSR_Err:
    Call ExeErr(Err)
End Sub

Private Sub AddinInstance_OnStartupComplete(custom() As Variant)
    On Error GoTo DSR_Err
    Call WriteDBGString(App.EXEName & " OnStartupComplete() @ " & Date & " " & Time & vbCrLf, App.
         EXEName & ".log")

Exit Sub

DSR_Err:
    Call ExeErr(Err)
End Sub

Private Sub AddinInstance_Terminate()
    Dim killer As Boolean
    On Error GoTo DSR_Err
    Call WriteDBGString(App.EXEName & " OnAddInsUpdate @ " & Date & " " & Time & vbCrLf, App.
         EXEName & ".log")
    Set p_AddinInst = Nothing
    If p_OfficeCrypt.ConnectedToServer Then Call WriteDBGString(App.EXEName & " is still connected
          during Terminate()" & vbCrLf & "Will force termination now!", App.EXEName & ".log")

    While Not killer
        killer = p_OfficeCrypt.clsBruteTerminate
    Wend

    Set p_OfficeCrypt = Nothing
    Set oHostInst = Nothing
    Set myFrmSplash = Nothing
    Set myFrmFunctions = Nothing
    Set myFrmLibsLocal = Nothing
    Set myFrmNamedRanges = Nothing
    Set myFrmObjects = Nothing
    Set myFrmQuantlets = Nothing
    Set myFrmQuantsLocal = Nothing
    Set myFrmStatus = Nothing
    Set myFrmXLA = New frmXLA

    CloseMDCOM
    Call WriteDBGString(App.EXEName & " finished Terminate() successfully!" & vbCrLf & "Object
         Cleanup done!" & vbCrLf, App.EXEName & ".log")
    Call WriteDBGString("Have a nice day..." & vbCrLf & vbCrLf, App.EXEName & ".log")
Exit Sub

DSR_Err:
    Call ExeErr(Err)
End Sub

Private Sub p_mcomboEvents_Change(ByVal Ctrl As Office.CommandBarComboBox)
    Dim i%
    On Error GoTo DSR_Err
    With p_mcomboEvents
        ReDim str_ComboText(.ListCount)
            If Ctrl.Text <> "" Then
                str_ComboText(.ListCount) = Ctrl.Text
                .AddItem (str_ComboText(.ListCount))
            End If
        .DropDownLines = .ListCount
    End With

    If Ctrl.Text <> "" Then
        p_OfficeCrypt.clsSendQuantlet (Ctrl.Text)
        myFrmGetResult.txtGetResult.Text = p_OfficeCrypt.clsGetResult
        If myFrmGetResult.txtGetResult.Text <> "" Then myFrmGetResult.Show
    End If
    Exit Sub

DSR_Err:
    Call ExeErr(Err)
End Sub

Private Sub p_mctlBtnEvents_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    Dim strResults As String
    Dim temp As Integer
    On Error GoTo Event_Err
    ' get instance on splash screen.
    If myFrmSplash Is Nothing Then Set myFrmSplash = New frmSplashNew
    If blnRexToggled = False Then
        strResults = p_OfficeCrypt.MDReXInfoServer.serverBuildDate
        With myFrmSplash
            .Caption = "MD*ReX 2004"
            .lblCopyright = App.LegalCopyright
            .lblCompanyProduct.Caption = "MD*ReX Microsoft Office COM Add-in"
            .lblVersion = "MD*ReX " & Format(App.Major, "#0") & "." & Format(App.Minor, "#00") &
                 "." & Format(App.Revision, "#0000")
            .txtSplash2.ToolTipText = "MD*ReX"
```

```
            .txtSplash2.Text = "MD*COM guid: " & oAddinInst.Guid & vbCrLf & "MD*COM says..." &
                 vbCrLf & "Path to MD*SERV: " & App.Path & "\mdserv" & vbCrLf & "Operating System:
                  " & _
            oHostInst.OperatingSystem & vbCrLf & "Host: " & oHostInst.Value & " " _
            & oHostInst.Version & " Build: " & oHostInst.Build & vbCrLf & "MD*Crypt says... " & _
            vbCrLf & "Server Status: " & p_OfficeCrypt.propServerStatus & vbCrLf & "MD*Crypt
                 version " _
            & p_OfficeCrypt.MDReXInfoServer.getMdCryptVersion & vbCrLf & "MD*Crypt Java: " _
            & p_OfficeCrypt.MDReXInfoServer.getMdCryptJava & vbCrLf & "MD*Serv build: " _
            & p_OfficeCrypt.MDReXInfoServer.getMdServVersion & vbCrLf _
            & "MD*XQS Server ID: " _
            & p_OfficeCrypt.MDReXInfoServer.getMyMdServId & vbCrLf _
            & "MD*XQS Server System: " _
            & p_OfficeCrypt.MDReXInfoServer.getServerSystem & vbCrLf _
            & "MD*XQS Server Build: " _
            & p_OfficeCrypt.MDReXInfoServer.getServerBuild & vbCrLf _
            & "Java SDK Version: " _
            & p_OfficeCrypt.MDReXInfoServer.getMdServJava & vbCrLf _
            & "MD*XQS Server Build Date: " _
            & strResults
            .txtSplash2.Visible = True
            .txtSplash2.Locked = True
            .Command3.Caption = "Ok"
            .Command4.Caption = "www.md-rex.com"
            .Show
        End With
        Set myCommandBar = oHostInst.CommandBars("Worksheet Menu Bar") '//handle on MSExcel Main
             Commandbar
        Call myMDCOMMenu.MDCOMConnect(p_AddinInst, oHostInst, myCommandBar) '//entry point for
             creating MDRex Toolbar
        Call myMDCOMContextMenu.MDCOMConnectContextMenu(p_AddinInst, oHostInst, myCommandBar) '//
             entry point for creating MDRex Context Menu
        Set p_mcomboEvents = CreateAddInComboBox(oHostInst, p_AddinInst) '//handle on MDReX
             commandline button
        blnRexToggled = True
    Else
        If p_OfficeCrypt.ConnectedToServer = True Then
            temp = MsgBox("You are still connected to: " & vbCrLf & p_OfficeCrypt.propServerIP &
                 vbCrLf & "Pressing yes will disconnect you." & vbCrLf & "WARNING: all data on the
                  server will be lost!" & vbCrLf & "Do you want to disconnect?", 4 + 32, "
                 Disconnect " & p_OfficeCrypt.propServerIP)
            If temp = 6 Then
                p_OfficeCrypt.clsTerminate
            Else
                Exit Sub
            End If
            oHostInst.StatusBar = ""
            Call myMDCOMMenu.MDCOMDisconnect
            Call myMDCOMContextMenu.MDCOMContextMenuDisconnect
        Else
            oHostInst.StatusBar = ""
            Call myMDCOMMenu.MDCOMDisconnect
            Call myMDCOMContextMenu.MDCOMContextMenuDisconnect
        End If
        blnRexToggled = False
    End If

Exit Sub

Event_End:
    Exit Sub

Event_Err:
    Call ExeErr(Err)
    Resume Event_End
End Sub

Public Function OneVar(x As Excel.Range) As Excel.Range

Dim Varname As String
Dim Matrix As Range
Dim ValRange As Range
Dim tempVal As Object
Dim i As Long, j As Long
Dim RowNum As Long, ColNum As Long

    If Not ActiveworkbookIsValid(False, "rexone") Then

1:          Varname = InputBox("Name of XploRe Object: ", "Put")

        If Varname = "" Then
            MsgBox "You Specified an empty Object name!"
            Exit Function
        End If

        Select Case CounterRuns
        Case 0
            GoTo 2:
        Case Else
```

```
            For i = 0 To UBound(myMapper.XPLObject)
                If Varname = myMapper.XPLObject(i) Then
                    If MsgBox(Varname & " already exists at server! Overwrite", vbOKCancel) =
                        vbCancel Then GoTo 1:
                End If
            Next i
        End Select

2:          Set ValRange = x

        If IsEmpty(ValRange) Then
            MsgBox "You haven't selected a cell range for the object" & vbCrLf & "or the selected
                range is empty!"
            Exit Function
        End If
        RowNum = ValRange.Rows.Count
        ColNum = ValRange.Columns.Count
        Set Matrix = ValRange

        Call p_OfficeCrypt.SendDoubleMatrix(Varname, Matrix, ColNum, RowNum)
        UpdateViewsMenu
        UpdateViewsContext
        CountRuns
        Call Map(ValRange, Varname, CounterRuns)

        p_OfficeCrypt.clsSendQuantlet ("library (""stats"")")
        p_OfficeCrypt.clsSendQuantlet ("setenv(""outputstringformat"", ""%s"")")
        p_OfficeCrypt.clsSendQuantlet ("descriptive(" & Varname & ")")
        myFrmGetResult.txtGetResult.Text = p_OfficeCrypt.clsGetResult
        If myFrmGetResult.txtGetResult.Text <> "" Then myFrmGetResult.Show
        'OpenTemplate ("onevar")
End If
End Function

Public Function ActiveworkbookIsValid(bThrowMsg As Boolean, CustDocProp As String) As Boolean

    'Returns True/False indicating whether or not the activeworkbook
    'is created from our template.  If bThrowMsg is true, it also displays
    'a warning to the user when the workbook is not "valid".
    '
    '** Note: The workbook template contains a custom document property named
    '   "GanttChart". If this property appears in the activeworkbook, then the
    '   workbook is assumed to be "valid".

    ActiveworkbookIsValid = False
    If Not (oHostInst.ActiveWorkbook Is Nothing) Then
        Dim b As Boolean
        Dim PropString As Object
        Dim wb As Workbook
        On Error Resume Next
        Set wb = oHostInst.ActiveWorkbook
        b = oHostInst.wb.CustomDocumentProperties(CustDocProp).Value
        PropString = wb.CustomDocumentProperties
        PropString = wb.CustomDocumentProperties(CustDocProp).Value
        If Err = 0 Then ActiveworkbookIsValid = True
    End If

    If Not (ActiveworkbookIsValid) And bThrowMsg Then
        MsgBox "The active workbook is not a valid workbook for use with " & _
               "the MD*ReX Add-in.", , "MD*ReX"
    End If

End Function

Public Function XPLEval(XPLExpression As Variant, ParamArray XPLArgs() As Variant) As Variant

    Dim i%
    Dim tExpr$
    Dim tArgs           As Variant

    If UBound(XPLArgs) >= 0 Then
        tArgs = XPLArgs(i)
    End If

    If Not ActiveworkbookIsValid(False, "rexeval") Then


    tExpr = XPLExpression
    XPLEval = XPLEvalReturn(tExpr)

    End If

End Function

Function XPLEvalReturn(EvalString As String) As Variant

Dim tStr            As String
Dim errStr          As String
```

```
tStr = EvalString

p_OfficeCrypt.clsSendQuantlet ("setenv(""outheadline"", """")")
'p_OfficeCrypt.clsSendQuantlet ("setenv(""outlayerline"", """")")
p_OfficeCrypt.clsSendQuantlet ("setenv(""outlineno"", """")")
p_OfficeCrypt.clsSendQuantlet ("setenv(""outputstringformat"", ""%s"")")
p_OfficeCrypt.clsSendQuantlet (tStr)
tStr = p_OfficeCrypt.clsGetResult
tStr = Replace(tStr, Chr(13), "")
tStr = Replace(tStr, Chr(10), "")
XPLEvalReturn = tStr
If Not (Err.Number <> 0) Then
    Exit Function
Else
XPLEvalReturn = "#XPLError"
End If
End Function
```

B.16 frmConnect.frm

```
VERSION 5.00
Begin VB.Form frmConnect
   BorderStyle     =   4  'Festes Werkzeugfenster
   Caption         =   "Connect to XQS"
   ClientHeight    =   1545
   ClientLeft      =   2835
   ClientTop       =   3435
   ClientWidth     =   4350
   Icon            =   "frmConnect.frx":0000
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   912.837
   ScaleMode       =   0  'Benutzerdefiniert
   ScaleWidth      =   4084.415
   ShowInTaskbar   =   0   'False
   StartUpPosition =   2  'Bildschirmmitte
   Begin VB.ComboBox Combo2
      Height          =   315
      Left            =   1440
      Style           =   2  'Dropdown-Liste
      TabIndex        =   5
      Top             =   480
      Width           =   2775
   End
   Begin VB.ComboBox Combo1
      Height          =   315
      ItemData        =   "frmConnect.frx":08CA
      Left            =   1440
      List            =   "frmConnect.frx":08CC
      Style           =   2  'Dropdown-Liste
      TabIndex        =   4
      Top             =   120
      Width           =   2775
   End
   Begin VB.CommandButton cmdOK
      Caption         =   "OK"
      CausesValidation=   0   'False
      Default         =   -1  'True
      Height          =   390
      Left            =   1800
      TabIndex        =   2
      Top             =   1020
      Width           =   1140
   End
   Begin VB.CommandButton cmdCancel
      Cancel          =   -1  'True
      Caption         =   "Cancel"
      CausesValidation=   0   'False
      Height          =   390
      Left            =   3000
      TabIndex        =   3
      Top             =   1020
      Width           =   1140
   End
   Begin VB.Label lblLabels
      Caption         =   "&XQS IP address:"
      Height          =   270
      Index           =   0
      Left            =   105
      TabIndex        =   0
      Top             =   142
      Width           =   1320
   End
   Begin VB.Label lblLabels
      Caption         =   "X&QS Port #:"
      Height          =   270
      Index           =   1
      Left            =   105
      TabIndex        =   1
      Top             =   502
      Width           =   1080
   End
End
Attribute VB_Name = "frmConnect"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

Public ConnectSucceeded As Boolean
Dim IPString As String
Dim PortLong As Long
Dim myMDCOMMenu As clsMDCOMMenu
```

```
Private Sub cmdCancel_Click()
    'Globale Variable auf False setzen,
    'um eine fehlgeschlagene Anmeldung zu kennzeichnen.
    ConnectSucceeded = False
    Me.Hide
End Sub

Private Sub cmdOk_Click()
Me.MousePointer = vbHourglass

p_OfficeCrypt.SetServerIP (Me.Combo1.Text)
p_OfficeCrypt.SetServerPort (Me.Combo2.Text)

If p_OfficeCrypt.clsConnect = True Then

        UpdateViewsMenu
        UpdateViewsContext
        Me.MousePointer = 0
        Me.Hide

' REMOVED from project since we use excel names
'    If (oHostInst <> "Microsoft Word") Then
'        AddMappingSheet ("XploRe 2 Excel Mapping Table")
'        oHostInst.Worksheets(1).Activate
'    End If

p_OfficeCrypt.clsSendQuantlet ("mdrexinfo=info()")
InfoString = p_OfficeCrypt.clsGetResult
p_OfficeCrypt.clsSendQuantlet ("mdrexinfo.objects")
ObjectString = p_OfficeCrypt.clsGetResult
p_OfficeCrypt.clsSendQuantlet ("mdrexinfo.functions")
FunctionString = p_OfficeCrypt.clsGetResult
p_OfficeCrypt.clsSendQuantlet ("mdrexinfo.quantlets")
QuantletString = p_OfficeCrypt.clsGetResult

Else
    MsgBox "Could not connect!" & vbCrLf & "This might be due to a running xqs.exe process." &
          vbCrLf & "Switch to process manager and kill any xqs... processes." & vbCrLf & "Then
          restart Excel and try again."
    Exit Sub
End If
Unload Me
End Sub

Private Sub Combo1_Change()
IPString = Combo1.SelText
End Sub

Private Sub Combo2_Change()
If IsNumeric(Me.Combo2.Text) = True Then
PortLong = Combo2.SelText
End If
End Sub

Private Sub Form_Load()
If (myMDCOMMenu Is Nothing) Then
Set myMDCOMMenu = New clsMDCOMMenu
End If
With Me.Combo1
    .AddItem ("localhost")
    .AddItem ("xqs.xplore-stat.de")
    .AddItem ("apus.wiwi.hu-berlin.de")
'    .AddItem ("amadeus.statistik.uni-dortmund.de")
'    .AddItem ("helena.stat.uni-muenchen.de")
'    .AddItem ("mid.ism.ac.jp")
'    .AddItem ("corona.utstat.utoronto.ca")
'    .AddItem ("xplore.math.usu.edu")
'    .AddItem ("stat.wharton.upenn.edu")
'    .AddItem ("pulsar.galaxy.gmu.edu")
'    .AddItem ("euterpe.ensae.fr")
'    .AddItem ("stat2.wu-wien.ac.at")
'    .AddItem ("stat4ux.stat.ucl.ac.be")
'    .AddItem ("sunsite.univie.ac.at")
    .Enabled = True
    .ListIndex = 0
    .Locked = False
End With
With Me.Combo2
    .AddItem ("8889")
    .AddItem ("8890")
    .AddItem ("8891")
    .Enabled = True
    .ListIndex = 0
    .Locked = False
End With
End Sub
```

B.17 frmFunctions.frm

```
VERSION 5.00
Begin VB.Form frmFunctions
   BorderStyle     =   4  'Festes Werkzeugfenster
   Caption         =   "Functions"
   ClientHeight    =   7500
   ClientLeft      =   45
   ClientTop       =   315
   ClientWidth     =   7725
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   7500
   ScaleWidth      =   7725
   ShowInTaskbar   =   0   'False
   StartUpPosition =   3  'Windows-Standard
   Begin VB.Frame Frame1
      Caption         =   "Frame1"
      Height          =   7215
      Left            =   120
      TabIndex        =   0
      Top             =   120
      Width           =   7455
      Begin VB.TextBox txtFunctions
         BackColor       =   &H80000004&
         Height          =   6765
         Left            =   120
         Locked          =   -1  'True
         MultiLine       =   -1  'True
         ScrollBars      =   3  'Beides
         TabIndex        =   1
         Top             =   240
         Width           =   7125
      End
   End
End
Attribute VB_Name = "frmFunctions"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit
```

B.18 frmGetResult.frm

```
VERSION 5.00
Begin VB.Form frmGetResult
   BorderStyle     =   3  'Fester Dialog
   Caption         =   "XQS Result"
   ClientHeight    =   4935
   ClientLeft      =   45
   ClientTop       =   405
   ClientWidth     =   6750
   Icon            =   "frmGetResult.frx":0000
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   4935
   ScaleWidth      =   6750
   ShowInTaskbar   =   0   'False
   Begin VB.Frame Frame1
      Caption         =   "Result"
      Height          =   4695
      Left            =   120
      TabIndex        =   0
      Top             =   120
      Width           =   6495
      Begin VB.TextBox txtGetResult
         BeginProperty Font
            Name            =   "Courier"
            Size            =   9.75
            Charset         =   0
            Weight          =   400
            Underline       =   0   'False
            Italic          =   0   'False
            Strikethrough   =   0   'False
         EndProperty
         Height          =   4350
         Left            =   90
         Locked          =   -1  'True
         MultiLine       =   -1  'True
         ScrollBars      =   3  'Beides
         TabIndex        =   1
         Text            =   "frmGetResult.frx":08CA
         Top             =   225
         Width           =   6285
      End
   End
End
Attribute VB_Name = "frmGetResult"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
    Option Explicit
```

B.19 frmLibsLocal.frm

```
VERSION 5.00
Begin VB.Form frmLibsLocal
   BorderStyle     =   4  'Festes Werkzeugfenster
   Caption         =   "Local Libraries"
   ClientHeight    =   5235
   ClientLeft      =   45
   ClientTop       =   315
   ClientWidth     =   7740
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   5235
   ScaleWidth      =   7740
   ShowInTaskbar   =   0   'False
   StartUpPosition =   3  'Windows-Standard
   Begin VB.TextBox txtLibsLocal
      BackColor       =   &H80000004&
      Height          =   4500
      Left            =   240
      Locked          =   -1  'True
      MultiLine       =   -1  'True
      ScrollBars      =   3  'Beides
      TabIndex        =   0
      Top             =   360
      Width           =   7155
   End
   Begin VB.Frame Frame1
      Caption         =   "Libraries"
      Height          =   4935
      Left            =   120
      TabIndex        =   1
      Top             =   120
      Width           =   7455
   End
End
Attribute VB_Name = "frmLibsLocal"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit
```

B.20 frmNamedRanges.frm

```
VERSION 5.00
Begin VB.Form frmNamedRanges
   BorderStyle     =   4  'Festes Werkzeugfenster
   Caption         =   "XPL2XLS Object Mapping"
   ClientHeight    =   4230
   ClientLeft      =   45
   ClientTop       =   315
   ClientWidth     =   7020
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   4230
   ScaleWidth      =   7020
   ShowInTaskbar   =   0   'False
   StartUpPosition =   3  'Windows-Standard
   Begin VB.Frame fraListNames
      Caption         =   "Name Mapping"
      Height          =   3975
      Left            =   120
      TabIndex        =   0
      Top             =   120
      Width           =   6735
      Begin VB.ListBox listNamedRanges
         Height          =   3570
         Left            =   120
         TabIndex        =   1
         Top             =   240
         Width           =   6495
      End
   End
End
Attribute VB_Name = "frmNamedRanges"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit
```

B.21 frmObjects.frm

```
VERSION 5.00
Begin VB.Form frmObjects
   BorderStyle     =   4  'Festes Werkzeugfenster
   Caption         =   "Objects"
   ClientHeight    =   7440
   ClientLeft      =   45
   ClientTop       =   315
   ClientWidth     =   7710
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   7440
   ScaleWidth      =   7710
   ShowInTaskbar   =   0   'False
   StartUpPosition =   3  'Windows-Standard
   Begin VB.Frame Frame1
      Caption         =   "Objects"
      Height          =   7215
      Left            =   120
      TabIndex        =   0
      Top             =   120
      Width           =   7455
      Begin VB.TextBox txtObjects
         BackColor       =   &H80000004&
         Height          =   6780
         Left            =   120
         Locked          =   -1  'True
         MultiLine       =   -1  'True
         ScrollBars      =   3  'Beides
         TabIndex        =   1
         Top             =   240
         Width           =   7125
      End
   End
End
Attribute VB_Name = "frmObjects"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit
```

B.22 frmQuantlets.frm

```
VERSION 5.00
Begin VB.Form frmQuantlets
   BorderStyle     =   4  'Festes Werkzeugfenster
   Caption         =   "Quantlets"
   ClientHeight    =   7350
   ClientLeft      =   45
   ClientTop       =   315
   ClientWidth     =   7545
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   7350
   ScaleWidth      =   7545
   ShowInTaskbar   =   0   'False
   StartUpPosition =   3  'Windows-Standard
   Begin VB.TextBox txtQuantlets
      BackColor       =   &H80000004&
      Height          =   6735
      Left            =   240
      Locked          =   -1  'True
      MultiLine       =   -1  'True
      ScrollBars      =   3  'Beides
      TabIndex        =   0
      Top             =   360
      Width           =   7095
   End
   Begin VB.Frame Frame1
      Caption         =   "Quantlets"
      Height          =   7095
      Left            =   120
      TabIndex        =   1
      Top             =   120
      Width           =   7335
   End
End
Attribute VB_Name = "frmQuantlets"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit
```

B.23 frmQuantsLocal.frm

```
VERSION 5.00
Begin VB.Form frmQuantsLocal
   BorderStyle     =   4  'Festes Werkzeugfenster
   Caption         =   "Quantlets (local)"
   ClientHeight    =   5730
   ClientLeft      =   45
   ClientTop       =   315
   ClientWidth     =   8190
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   5730
   ScaleWidth      =   8190
   ShowInTaskbar   =   0   'False
   StartUpPosition =   3  'Windows-Standard
   Begin VB.TextBox txtQuantsLocal
      BackColor       =   &H80000004&
      Height          =   4980
      Left            =   240
      Locked          =   -1  'True
      MultiLine       =   -1  'True
      ScrollBars      =   3  'Beides
      TabIndex        =   0
      Top             =   360
      Width           =   7605
   End
   Begin VB.Frame Frame1
      Caption         =   "Quantlets (local)"
      Height          =   5415
      Left            =   120
      TabIndex        =   1
      Top             =   120
      Width           =   7935
   End
End
Attribute VB_Name = "frmQuantsLocal"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit
```

B.24 frmSplashNew.frm

```
VERSION 5.00
Begin VB.Form frmSplashNew
   BackColor       =   &H00800000&
   BorderStyle     =   1  'Fest Einfach
   Caption         =   "Form1"
   ClientHeight    =   8460
   ClientLeft      =   45
   ClientTop       =   435
   ClientWidth     =   6210
   Icon            =   "frmSplashNew.frx":0000
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   Picture         =   "frmSplashNew.frx":08CA
   ScaleHeight     =   8460
   ScaleWidth      =   6210
   StartUpPosition =   2  'Bildschirmmitte
   Begin VB.TextBox txtSplash2
      BackColor       =   &H80000009&
      Height          =   4335
      Left            =   0
      MultiLine       =   -1  'True
      ScrollBars      =   3  'Beides
      TabIndex        =   9
      Text            =   "frmSplashNew.frx":5136A
      ToolTipText     =   "Status Info"
      Top             =   4080
      Width           =   6255
   End
   Begin VB.CommandButton Command4
      Caption         =   "Command4"
      Height          =   375
      Left            =   4680
      TabIndex        =   8
      Top             =   2880
      Width           =   1455
   End
   Begin VB.CommandButton Command3
      Caption         =   "Command3"
      Height          =   375
      Left            =   3960
      TabIndex        =   7
      Top             =   2880
      Width           =   615
   End
   Begin VB.CommandButton Command2
      Caption         =   "md-rex.com"
      Height          =   30
      Left            =   14070
      TabIndex        =   6
      Top             =   6300
      Width           =   0
   End
   Begin VB.CommandButton Command1
      Caption         =   "ok"
      Height          =   660
      Left            =   28140
      TabIndex        =   5
      Top             =   7050
      Width           =   0
   End
   Begin VB.TextBox txtSplash
      BackColor       =   &H00FFC0C0&
      Height          =   720
      Left            =   42210
      Locked          =   -1  'True
      MultiLine       =   -1  'True
      ScrollBars      =   2  'Vertikal
      TabIndex        =   0
      Top             =   8040
      Width           =   0
   End
   Begin VB.Label lblCopyright
      Alignment       =   1  'Rechts
      BackColor       =   &H00000000&
      Caption         =   "Copyright"
      BeginProperty Font
         Name            =   "Arial"
         Size            =   8.25
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   -1  'True
         Strikethrough   =   0   'False
      EndProperty
```

```
      ForeColor       =   &H80000005&
      Height          =   255
      Left            =   2520
      TabIndex        =   4
      Top             =   3735
      Width           =   3600
   End
   Begin VB.Label lblVersion
      Alignment       =   1  'Rechts
      BackColor       =   &H00000000&
      Caption         =   "Version"
      BeginProperty Font
         Name            =   "Arial"
         Size            =   8.25
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000005&
      Height          =   255
      Left            =   2520
      TabIndex        =   3
      Top             =   3540
      Width           =   3600
   End
   Begin VB.Label lblCompanyProduct
      Alignment       =   1  'Rechts
      BackColor       =   &H00000000&
      Caption         =   "Unternehmen/Produkt"
      BeginProperty Font
         Name            =   "Arial"
         Size            =   8.25
         Charset         =   0
         Weight          =   700
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000005&
      Height          =   255
      Left            =   2520
      TabIndex        =   2
      Top             =   3330
      Width           =   3600
   End
   Begin VB.Label lblLicenseTo
      Appearance      =   0  '2D
      BackColor       =   &H80000006&
      Caption         =   "      Member of XploRe Quantlet Client Family "
      BeginProperty Font
         Name            =   "Arial"
         Size            =   8.25
         Charset         =   0
         Weight          =   400
         Underline       =   0   'False
         Italic          =   0   'False
         Strikethrough   =   0   'False
      EndProperty
      ForeColor       =   &H80000005&
      Height          =   255
      Left            =   0
      TabIndex        =   1
      Top             =   0
      Width           =   3255
   End
End
Attribute VB_Name = "frmSplashNew"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

Private Sub Command3_Click()
    Unload Me
End Sub

Private Sub Command4_Click()
    Dim Browser As Object
    On Error Resume Next
    Set Browser = CreateObject("internetexplorer.application")
    Browser.navigate ("http://www.md-rex.com/")
    With Browser
        .StatusBar = True
        .MenuBar = False
        .Toolbar = 1
        .Visible = True
```

```
    End With
End Sub

Private Sub Form_Click()
    Unload Me
End Sub

Private Sub Form_Load()
'
End Sub
```

B.25 frmStatus.frm

```
VERSION 5.00
Begin VB.Form frmStatus
   BorderStyle     =   4  'Festes Werkzeugfenster
   Caption         =   "Log Files"
   ClientHeight    =   7485
   ClientLeft      =   45
   ClientTop       =   315
   ClientWidth     =   8430
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   7485
   ScaleWidth      =   8430
   ShowInTaskbar   =   0   'False
   StartUpPosition =   3  'Windows-Standard
   Begin VB.Frame Frame2
      Caption         =   "mdrex.log"
      Height          =   3255
      Left            =   240
      TabIndex        =   1
      Top             =   4080
      Width           =   7815
      Begin VB.TextBox txtMDRexLog
         Height          =   2655
         Left            =   360
         MultiLine       =   -1  'True
         ScrollBars      =   3  'Beides
         TabIndex        =   3
         Text            =   "frmStatus.frx":0000
         Top             =   360
         Width           =   7335
      End
   End
   Begin VB.Frame Frame1
      Caption         =   "prot.log"
      Height          =   3735
      Left            =   240
      TabIndex        =   0
      Top             =   120
      Width           =   7815
      Begin VB.TextBox txtProtLog
         Height          =   3135
         Left            =   240
         MultiLine       =   -1  'True
         ScrollBars      =   3  'Beides
         TabIndex        =   2
         Text            =   "frmStatus.frx":0006
         Top             =   360
         Width           =   7335
      End
   End
End
Attribute VB_Name = "frmStatus"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit
```

B.26 frmXLA.frm

```
VERSION 5.00
Begin VB.Form frmXLA
   BorderStyle     =   5  'Änderbares Werkzeugfenster
   Caption         =   "MD*ReX Excel Add-Ins"
   ClientHeight    =   3660
   ClientLeft      =   60
   ClientTop       =   330
   ClientWidth     =   4800
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   3660
   ScaleWidth      =   4800
   ShowInTaskbar   =   0   'False
   StartUpPosition =   3  'Windows-Standard
   Begin VB.CommandButton cmdXLAcancel
      Caption         =   "cancel"
      Height          =   375
      Left            =   3480
      TabIndex        =   3
      Top             =   3120
      Width           =   975
   End
   Begin VB.CommandButton cmdXLASelect
      Caption         =   "Select"
      Height          =   375
      Left            =   2280
      TabIndex        =   2
      Top             =   3120
      Width           =   1095
   End
   Begin VB.Frame Frame1
      Caption         =   "Available AddIns"
      Height          =   2775
      Left            =   120
      TabIndex        =   0
      Top             =   120
      Width           =   4455
      Begin VB.ListBox List1
         Height          =   2400
         Left            =   120
         TabIndex        =   1
         Top             =   240
         Width           =   4215
      End
   End
End
Attribute VB_Name = "frmXLA"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Option Explicit

Private Sub cmdXLAcancel_Click()
    Unload Me
End Sub

Private Sub cmdXLASelect_Click()
With Me
    Call LoadAddin(True, .List1.Text, "go")
    .Hide
End With
End Sub

Private Sub List1_Click()
With Me
    Call LoadAddin(True, .List1.Text, "go")
    .Hide
End With
End Sub


Private Sub List1_DblClick()
With Me
    Call LoadAddin(True, .List1.Text, "go")
    .Hide
End With
End Sub
```

B.27 frmXPLDirect.frm

```
VERSION 5.00
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
Begin VB.Form frmXPLDirect
   BackColor       =   &H00FFC0C0&
   Caption         =   """Analyse > Compute > XploRe"""
   ClientHeight    =   7650
   ClientLeft      =   165
   ClientTop       =   855
   ClientWidth     =   8145
   Icon            =   "frmXPLDirect.frx":0000
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   ScaleHeight     =   7650
   ScaleWidth      =   8145
   StartUpPosition =   3  'Windows-Standard
   Begin MSComDlg.CommonDialog cmDlg
      Left            =   7305
      Top             =   4080
      _ExtentX        =   847
      _ExtentY        =   847
      _Version        =   393216
      CancelError     =   -1  'True
   End
   Begin MSComctlLib.StatusBar statB
      Align           =   2  'Unten ausrichten
      Height          =   435
      Left            =   0
      TabIndex        =   4
      Top             =   7215
      Width           =   8145
      _ExtentX        =   14367
      _ExtentY        =   767
      SimpleText      =   "TEST"
      _Version        =   393216
      BeginProperty Panels {8E3867A5-8586-11D1-B16A-00C0F0283628}
         NumPanels       =   3
         BeginProperty Panel1 {8E3867AB-8586-11D1-B16A-00C0F0283628}
            Object.Width           =   2734
            MinWidth        =   2734
         EndProperty
         BeginProperty Panel2 {8E3867AB-8586-11D1-B16A-00C0F0283628}
            Object.Width           =   3528
            MinWidth        =   3528
         EndProperty
         BeginProperty Panel3 {8E3867AB-8586-11D1-B16A-00C0F0283628}
            Style           =   5
            Alignment       =   1
            AutoSize        =   2
            Object.Width           =   1508
            MinWidth        =   1499
            TextSave        =   "01:00"
         EndProperty
      EndProperty
   End
   Begin VB.TextBox txtXPLOutput
      BackColor       =   &H80000004&
      Height          =   4215
      Index           =   1
      Left            =   0
      Locked          =   -1  'True
      MultiLine       =   -1  'True
      ScrollBars      =   3  'Beides
      TabIndex        =   3
      Text            =   "frmXPLDirect.frx":08CA
      Top             =   3000
      Width           =   6975
   End
   Begin VB.CommandButton cmdClear
      Caption         =   "C&lear"
      Height          =   255
      Left            =   7065
      TabIndex        =   2
      Top             =   315
      Width           =   975
   End
   Begin VB.CommandButton cmdRun
      Caption         =   "&Execute"
      Height          =   255
      Left            =   7065
      TabIndex        =   1
      Top             =   75
      Width           =   975
   End
   Begin VB.TextBox txtXPLInput
```

```
      BackColor       =   &H00FFFFFF&
      Height          =   3015
      Index           =   0
      Left            =   0
      MultiLine       =   -1  'True
      ScrollBars      =   3  'Beides
      TabIndex        =   0
      Text            =   "frmXPLDirect.frx":08DB
      ToolTipText     =   "only Quantlets go here..."
      Top             =   -15
      Width           =   6975
   End
   Begin VB.Menu mnuProgram
      Caption         =   "&Program"
      Begin VB.Menu mnuNew
         Caption         =   "New"
         Shortcut        =   ^N
      End
      Begin VB.Menu mnuOpen
         Caption         =   "Open"
         Shortcut        =   ^O
      End
      Begin VB.Menu mnuExit
         Caption         =   "Exit"
      End
   End
   Begin VB.Menu mnuData
      Caption         =   "D&ata"
      Begin VB.Menu mnuDataOpen
         Caption         =   "Open"
      End
   End
   Begin VB.Menu mnuMain
      Caption         =   "&Main"
      Begin VB.Menu mnuObjects
         Caption         =   "Objects"
      End
      Begin VB.Menu mnuFunc
         Caption         =   "Functions"
      End
      Begin VB.Menu mnuQuantlets
         Caption         =   "Quantlets"
      End
   End
End
Attribute VB_Name = "frmXPLDirect"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Sub cmdClear_Click()
Me.txtXPLInput(0).Text = ""
Me.txtXPLOutput(1).Text = ""
End Sub

Private Sub cmdRun_Click()
Dim tempQuantlet As String
Dim sQuantlet As String
Dim done As Boolean

Me.MousePointer = vbHourglass
Me.statB.Panels.Item(1).Text = "Executing..."
Me.statB.Panels.Item(2).Text = p_OfficeCrypt.GetServerStatus

If Me.txtXPLInput.Item(0).Text = "" Then
    MsgBox "NULL STRINGS NOT ALLOWED!"
    Exit Sub
End If

Call p_OfficeCrypt.clsSendQuantlet("setenv(" + Chr(34) + "outputformat" + Chr(34) + "," + Chr(34)
     + "%8.9g" + Chr(34) + ")")
Call p_OfficeCrypt.clsSendQuantlet("setenv(" + Chr(34) + "outmaxdata" + Chr(34) + "," + Chr(34) +
     "10000" + Chr(34) + ")")

tempQuantlet = Me.txtXPLInput(0).Text
sQuantlet = Replace(tempQuantlet, Chr$(13) & Chr$(10), Chr$(10))

If (p_OfficeCrypt.ConnectedToServer = False) Or (p_OfficeCrypt Is Nothing) Then
    MsgBox "CANNOT SEND! PLEASE CONNECT FIRST!"
    Exit Sub
End If


Me.statB.Panels.Item(2).Text = p_OfficeCrypt.GetServerStatus
done = p_OfficeCrypt.clsSendQuantlet(sQuantlet)
Me.statB.Panels.Item(2).Text = p_OfficeCrypt.GetServerStatus

'Call p_OfficeCrypt.clsGetResult
```

```
'Me.txtXPLOutput(1).Text = clsresult(0)

Me.txtXPLOutput(1).Text = p_OfficeCrypt.clsGetResult
Me.MousePointer = 0
Me.statB.Panels.Item(1).Text = "Ready..."
Me.statB.Panels.Item(2).Text = p_OfficeCrypt.GetServerStatus
End Sub

Private Sub Form_Load()
Me.statB.Panels.Item(1).Text = "Ready..."
Me.statB.Panels.Item(2).Text = p_OfficeCrypt.GetServerStatus
End Sub

Private Sub mnuDataOpen_Click()
On Error GoTo ErrHnd
With Me.cmDlg
    .DialogTitle = "Open Data file"
    .Filter = "XploRe Data Files (*.dat) |*.dat|Text files (*.txt)|(*.txt)|All files (*.*)|(*.*)
        """
    .FilterIndex = 1
    .InitDir = App.Path & "\mdserv\data"
    .ShowOpen
End With
Call FileToTextBox(Me.txtXPLInput.Item(0), Me.cmDlg.FileName)

Exit Sub

ErrHnd:
    Exit Sub
End Sub

Private Sub mnuExit_Click()
Unload Me
End Sub

Private Sub mnuFunc_Click()
MsgBox "FUNCTIONS"
End Sub

Private Sub mnuNew_Click()
Me.txtXPLInput.Item(0).Text = ""
End Sub

Private Sub mnuObjects_Click()
MsgBox "OBJECTS"
End Sub

Private Sub mnuOpen_Click()
Dim tmp As String
On Error GoTo ErrHnd
With Me.cmDlg
    .DialogTitle = "Open Quantlet"
    .Filter = "Quantlets (*.xpl) |*.xpl|Text files (*.txt)|(*.txt)|All files (*.*)|(*.*)"
    .FilterIndex = 1
    .InitDir = App.Path & "\mdserv\lib"
    .ShowOpen
End With
Call FileToTextBox(Me.txtXPLInput.Item(0), Me.cmDlg.FileName)

Exit Sub

ErrHnd:
    Exit Sub
End Sub

Private Sub mnuQuantlets_Click()
MsgBox "OMLETTES"
End Sub
```

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständig ohne fremde Hilfe verfaßt und nur die angegebene Literatur und Hilfsmittel verwendet zu haben.

Gökhan Aydınlı
15. Juli 2004