PhUSE 2008

Paper AD05

Decimaker: A statistical software using R, Microsoft .NET, R (D)COM Server and graphical libraries

Julien Vanwinsberghe, ClinBAY, France
Francois Vandenhende, ClinBAY, Belgium

ABSTRACT

Most statistical packages in R are command-based and do not involve user-friendly graphical interfaces. This limits their use by people who are not familiar with the R language. In this paper, we present a framework and methods used to build a complete software package using R as statistical engine. The user interface was developed in C# (Microsoft programming language from .NET framework), and connection to R used the R (D)COM Server.

The resulting product: Decimaker (www.decimaker.com) is an innovative statistical software for the adaptive design and the Bayesian decision analysis of clinical trials.

In this paper, we detail the components that were used to build this GUI, including : R(D)COM Server software to communicate with R, *Zedgraph* library to create interactive pretty-looking graphics, *FamFamFam silk* icon library to make your interface looking as nice as you dream, and many others.

INTRODUCTION

CONTEXT

R (1) is a powerful command-line statistical software widely used by researchers. But its console interface restricts its use to people who are familiar with the R language. The diffusion of R methods to a large number of users may be limited by the lack of an intuitive graphical interface for novices. That is why it may sometimes be necessary to develop an interface easily understandable and readily available around the R programs to make them usable by a wider number of users.

In this paper, we present a graphical software that uses R and OpenBUGS programs for statistical analysis. Decimaker is a software used for the adaptive design and the Bayesian decision analysis of clinical trials. It is enhanced by a graphical interface to data processing, modeling, data analysis and trial simulations.

WHY USING R?

A lot of statistical methods have been developed in R and, as R is open-source, and the number of contributed packages steadily increasing. R is also fast, robust and validated.

WHY BUILDING A GRAPHICAL INTERFACE?

As said before, building a graphical interface around a set of R programs can be useful for specific applications, when the user community doesn't know R. With only few controls on a windows form, a user not familiar to R can easily and rapidly specify inputs for analyses and review output interactively.

DECIMAKER, A SOFTWARE FOR ADAPTIVE DESIGN AND BAYESIAN DECISION ANALYSIS

Our motivation for developing Decimaker was driven by the FDA critical path initiative that is pushing pharmaceutical, biotechnology and medical device companies to adopt new strategies to bring innovative therapies to the market. By bringing a software suite using adaptive designs and Bayesian decision analyses to the pharmaceutical desktop, we felt that the return on clinical research investments may be much improved, leading to better go/no go decisions, faster dose selections and less failed studies.

Decimaker was designed to assist clinical development teams during protocol design, trial simulations and analysis. It handles innovative adaptive designs, complex Bayesian analysis and decision-making solutions with a user-friendly graphical interface, as an optimized and comprehensive solution to drug development.

More information about the product is available on our website: www.decimaker.com. In this article, we use Decimaker as a case example and discuss various components used for the development of this software. These include the R kernel and R (D)COM Server to perform all the data processing calculations, and C# with .NET Framework 2.0 for programming, in Visual Studio 2005 .

PART I - GENERAL WORKFLOW TO BUILD A SOFTWARE USING R AS STATISTICAL KERNEL.

COMPONENTS

Schematic diagram of components

Decimaker is built as following architecture: the application kernel creates instances of database connections and R connection objects, and transmits it to user input and output interfaces, as references.

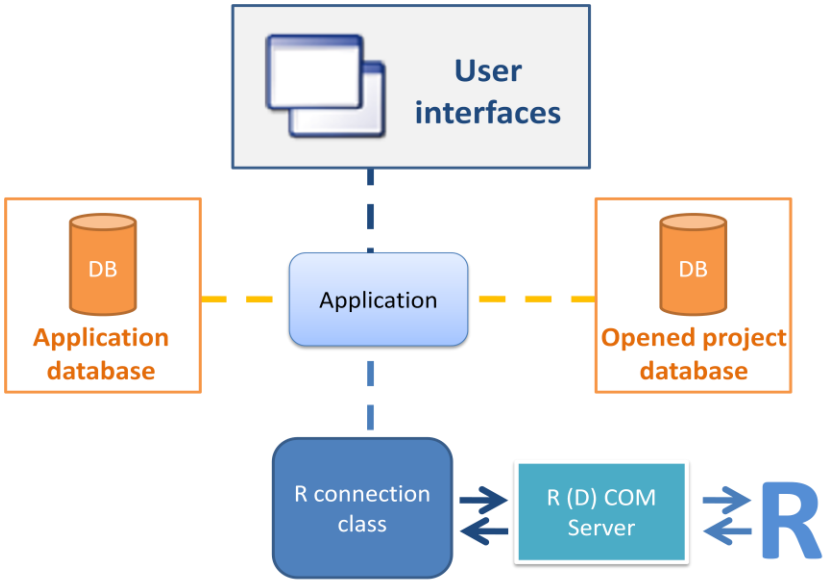

Figure 1. Schematic diagram of Decimaker components

R (D) COM Server

Description

R (D) COM Server (2) is a package that contains a COM server used to connect a client application (e.g. Microsoft Excel) with R. It runs only under the Windows environment, and provides methods to transfer data to/from R, and Active X controls for text and graphics output.

How to use it?

After installation, the programmer adds the COM references to his C# .NET project, as for any COM object. Then, he uses the methods of the *StatConnector* object to evaluate R commands, with (*Evaluate* method) or without (*EvaluateNoReturn* method) return values.

In the following example, we set a R variable *a* to a value of *12*, and read it back in the C# variable *ret*:

```
STATCONNECTORSRVLib.StatConnector R = new STATCONNECTORSRVLib.StatConnector();
R.Init("R");
R.EvaluateNoReturn("a<-12");
object ret = R.Evaluate("a");
```

Code 1. Example of use of the R (D) COM object in C#

The management of R-related classes in C#

Description

The methods provided by the *StatConnector* class are very useful but may be limited in some cases. For example, when one wants to evaluate a large number of commands at the same time, he must execute them one after the other. Moreover, the *Evaluate* method returns a result of object type, which must be casted manually and is sometimes altered in special cases such as values Na, NaN, or even simply lists.

That is why it can be worthwhile to write an additional class in order to avoid these errors, and add more functionality (like a log in our case).

Example

The following code represents a part of a possible R connection class in C#. It provides a method *EvaluateByFile* that permits to evaluate a complete R script at one time, instead of evaluating it line by line.

```
public class R
{
 // R (D)COM StatConnector class used to access data to/from R
 private StatConnectorClass Robj;
 // temp file used to write R scripts before loading it into R
 private string tempFilePath;

      […]

 /// <summary>
 /// Evaluate a R script by writing it into a file, and make R load it
 /// </summary>
 /// <param name="R_SCRIPT">The R script to evaluate</param>
 public void EvaluateByFile(string R_SCRIPT)
 {
   System.IO.StreamWriter sw = CreateTempFileForWrite();
   sw.Write(R_SCRIPT);
   sw.Close();

   try
   {
    Robj.EvaluateNoReturn("source('" + tempFilePath.Replace("\\", "\\\\") +
"')",true,true);
   }
   catch (System.Runtime.InteropServices.COMException ex)
   {
    DeleteTempFile();
    throw ex;
   }
   DeleteTempFile();
 }
 /// <summary>
 /// Open a stream writer on a temp file
 /// </summary>
 private System.IO.StreamWriter CreateTempFileForWrite()
 {
  tempFilePath = System.IO.Path.GetTempFileName();
  System.IO.FileStream file = new
System.IO.FileStream(tempFilePath,FileMode.Create,FileAccess.Write);
   return new System.IO.StreamWriter(file);
 }

 /// <summary>
 /// Delete the temp file
 /// </summary>
 private void DeleteTempFile()
 {
  System.IO.File.Delete(tempFilePath);
 }
}
```

Code 2. Part of a possible R-related class using R(D)COM Server

This class can also integrate some methods that look on the registry and on the computer disks (see part II for details) to determinate if the required components are installed before launching the installation (see following figure).

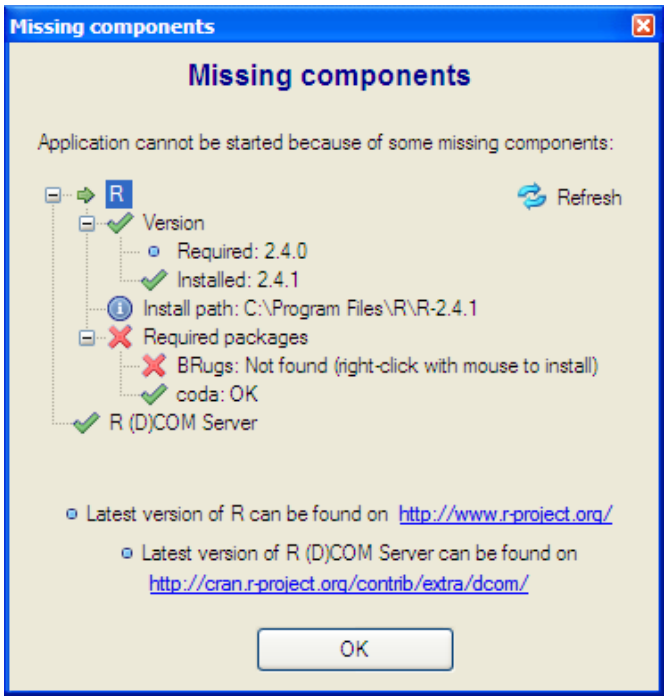

Figure 2. An interface to test if required components are installed

The databases

Database format

As our purpose was to develop Microsoft -integrated software, we used an MS Access database in Decimaker. Queries are executed via Microsoft Jet OleDb Provider, which can execute queries on the MS Access database even if the MS Office Suite is not installed on the computer. The advantage of this solution is that the client doesn't have to install any particular software, as for MS Sql Server for example, and it is designed for end user, without need of any server.

The application database

One database, in the application startup directory, contains all generic software information, including projects history, global information, and default values for input fields.

A database for each project

In the application, the user can create, save, export and import projects, containing data, parameters, reports and graphical results. Each project file is a database, on which we open an OleDb connection to execute queries.

User interfaces

The interest of building a software using R as a statistical kernel, instead of only developing a R package, resides in the graphical interface for users. As show in Figure 3, one can easily create or use existing controls to graphically show results (see part II for more details on some existing components). Data grids, and common input controls, with easily integration of copy-paste and drag-drop functionality can also be used for input data and report edition.

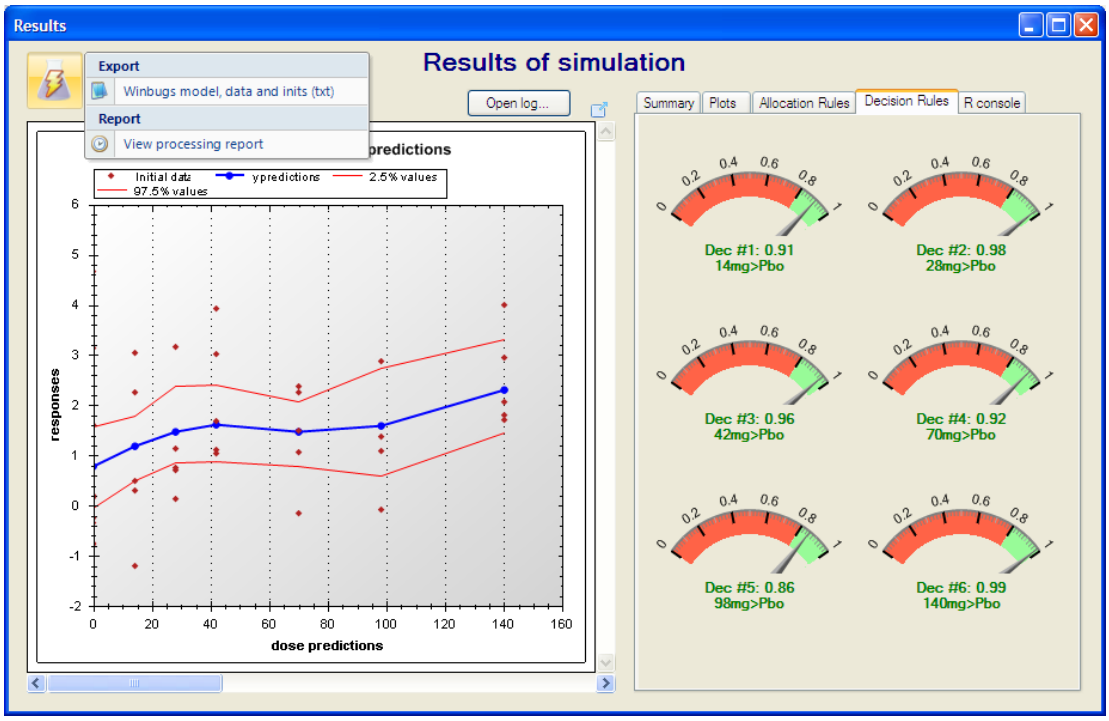

Figure 3. Screenshot of a graphical interface showing results of an analysis with R in Decimaker

DEVELOPMENT METHODOLOGY

Requirements and design

Before starting to write code when developing software, it is advisable to prepare in advance documents containing the requirements and design of the application. The **requirements** define the functions of the software, what it needs to do. The **design** activity consists in designing how the software is built and how its components interact.

These documents are essential to validate the software, to conduct tests of the software before distribution**,** to ensure that it corresponds to the expectations and contains no errors.

Build and application with .NET C# and Visual Studio

Decimaker has been developed with all the power of the .NET framework 2.0, which provides easy-to-use libraries for database connection, data encryption and form design. We used MS Visual Studio 2005 as development IDE, but one can use other free products, like Eclipse with C# plug-in, or Visual Studio 2008 Express, as necessary.

Some good practices for C# developers

Each programming language has its own conventions and best practices, which make the difference between "working code" and "efficient code" (3)**.** It is important for an application whose code will be reviewed to have been built so clean, structured and documented that it remains comprehensive in the long-run. In C#, there are, for example, naming conventions (first character in upper case for class and method names, and in lower case for variables and method parameters), and good programming practices (build short methods with obvious name, avoid using member variables, use public properties with private variables instead of public variables, …) that it is advised to apply for writing robust and easily maintainable code.

Validation

Overview

Validation is an integrated component of software development and it is a required step prior to release. When it comes time to distribute software to potential users, validation is a quality insurance certification that insures programs will behave as expected and accurately.

Software testing is the process of checking software, to verify that it satisfies its requirements and to detect errors (4). Create tests tailored to the requirements of the software, and integrate them throughout the development cycle requires time and patience.

Verification and validation tests

Software verification consists to verify that "the design outputs of a particular phase of the software development life cycle meet all of the specified requirements for that phase" (5). This includes verifying the requirements set on the behavior of the software, but also a review of the code and documentation.

Software validation is where software testing occurs. It includes a test plan of what and how it will be tested, with individual test cases, map directly back to the requirements (6). These tests include verification of math, valid/invalid inputs, power failure, structural and functional tests.

Bug tracking and testing tools

Software testing is rapidly evolving as a critical sub-discipline of software engineering, and it is a full-fledged job. Fortunately, there are free software to help in various stages of the validation. We have used four of these during the development life-cycle of Decimaker:

- **Subversion** (7) is a version control system used on many open source projects (Apache, KDE, Pyton, …) that manages current and historical versions of files. Subversion options can be integrated in Windows Explorer context menu (8) and even in Visual Studio (9).
- **BugTracker .NET** (10) is a free, open-source, web-based bug tracker particularly easy to handle. It permits to keep track of reported software bugs, define them a priority and status, attach screenshots, managing reports with charts, and more. It also supports integration with Subversion, to link code version files to fixed bugs.
- **FxCop** (11) is a Microsoft application that analyzes managed code assemblies to find possible improvements that can be made, such as related to performance, security, and violations of the programming and design rules.
- **Testlink** (12) is tool with web based interface that enables to create, manage and plan tests. One can generate reports, trace software requirements, prioritize and assign tasks. It seems to be the most popular open-source testing tool (13).

PART II – A CLOSER LOOK AT SPECIFIC SOFTWARE COMPONENTS

OPEN-SOURCE GRAPHICAL COMPONENTS

Adding a professional and pleasant interface to an application can be the most time-consuming part of the development effort. Fortunately, many open-source and very efficient components exist on the Internet, that the hardest is not to find them, but having the time to test them and choose the best suited to its needs.

Input controls

Decimaker contains references to the *Component Factory Toolkit* library (14), which provides improved versions of common used form controls, like *textbox*, *listbox* or *datagridview*, as pretty design controls with more functionality (see figure 4).

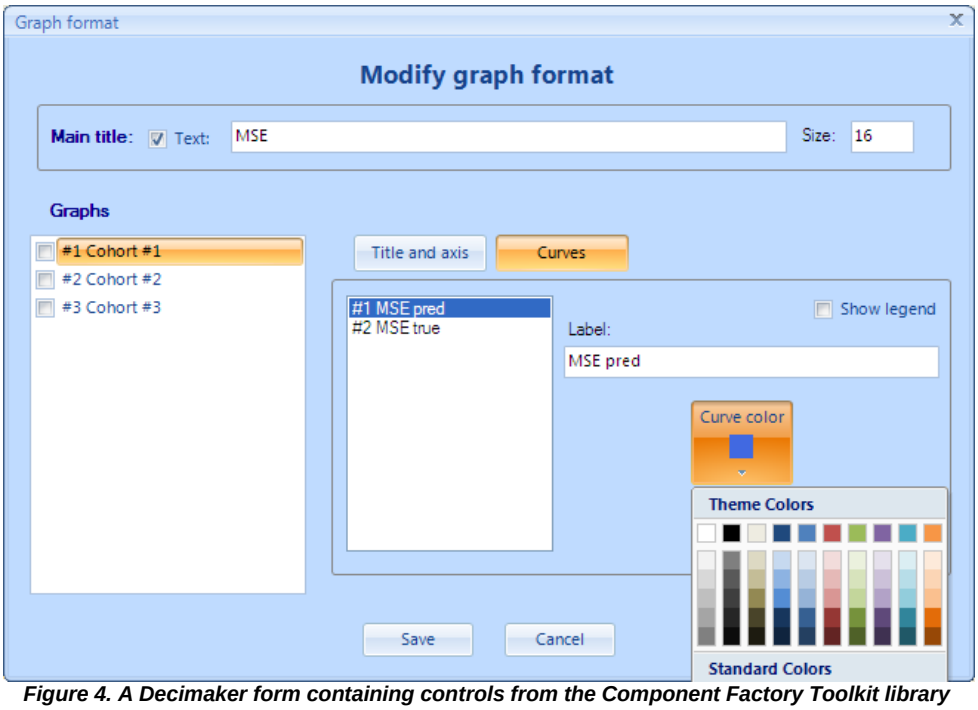

Figure 4. A Decimaker form containing controls from the Component Factory Toolkit library

Output controls

R Server ActiveX integrated components

R (D)COM Server includes two ActiveX controls (see following figure) that can be used on a form to show R plots (*StatConnectorGraphicsDevice*) and console (*StatConnectorCharacterDevice*) outputs.

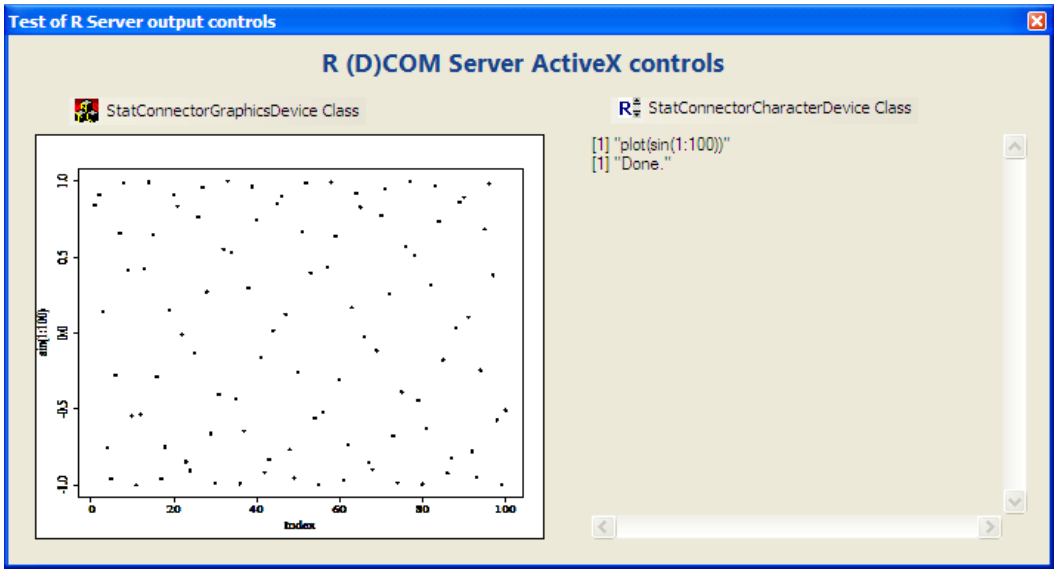

Figure 5. Example of use of the R Server ActiveX controls

Next piece of code shows how to use these objects to obtain the outputs as shown in Figure 5.

```
        public void TestAxStatConnectors()
        {
            // Create an instance of the StatConnector object
            STATCONNECTORSRVLib.StatConnector R = new
STATCONNECTORSRVLib.StatConnector();
            // Initialize R
            R.Init("R");
            // Connect graphical and text devices
            R.AddGraphicsDevice("gfx", axStatConnectorGraphicsDevice.GetGFX());

R.SetCharacterOutputDevice((StatConnectorCommonLib.IStatConnectorCharacterDevice)ax
StatConnectorCharacterDevice.GetOcx());
            // Test with R commands
            R.EvaluateNoReturn("print(\"plot(sin(1:100))\")");
            R.EvaluateNoReturn("plot(sin(1:100))");
            R.EvaluateNoReturn("print(\"Done.\")");
        }
```

Code 3. C# function used to test R Server ActiveX components

Open source gauge and charting controls

The ActiveX components included in R Server are very practical and easy to use, but many .NET open-source controls permit to display graphics on a more pleasant and professional way. For example, next figure shows an interface from Decimaker that displays charts with the *Zedgraph* (15) charting control, and gauges with the A.J. Bauer gauge control (16). Using these controls necessitates more code, to get data from R, send it to the control, and parameterize colors and styles, but the result is worth the work, as the quality of the graphical outputs is significantly improved.

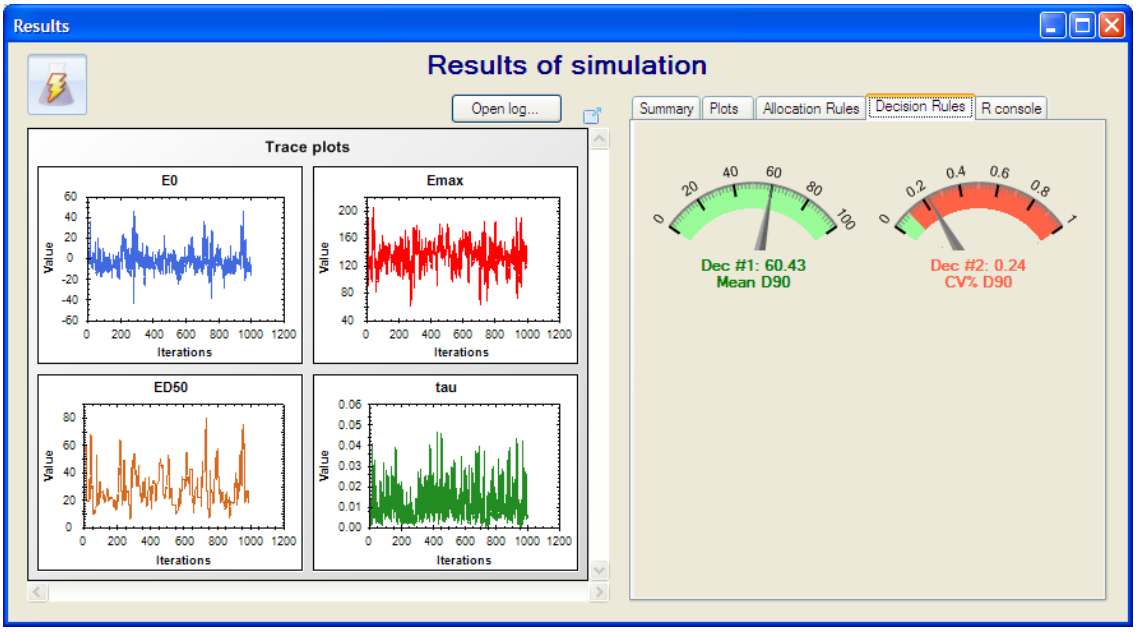

Figure 6. Example of use of the charting and gauge controls Zedgraph and AGauge

Icons

A subset of the icons used in Decimaker are built with The GIMP (17), a free image editor, but most are coming from the FamFamFam Silk free icon set (18), which provides more than 700 16-by-16 pixel icons in PNG format. A set of pretty and varied icons is very useful to build an intuitive interface.

CHECKING REQUIRED R COMPONENTS BY IO AND REGISTRY ACCESS

When you install a software using R on a computer, it adds value to develop an installation module that would also test the proper installation of required external components: such as R and R (D)COM software, as well as packages from R. This feature exists in Decimaker. Checks were made by looking into registry and user hard disks (see following figure).

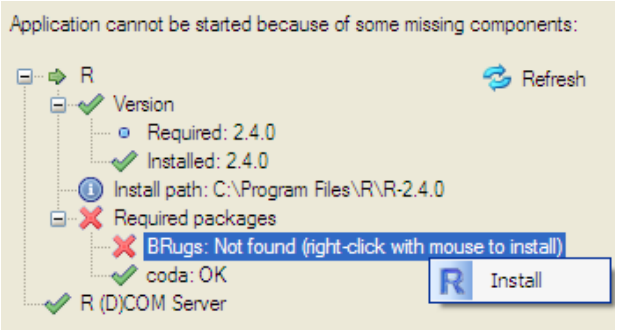

Figure 7. Tree view that displays required components

The following code searches install location of R Server in two possible registry keys:

```
    string RServerLoc; // Variable that will contain the R Server location
    object location; // Object used to get key values from registry
    location =
Microsoft.Win32.Registry.GetValue(@"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\C
urrentVersion\Uninstall\R/Scilab (D)COM Server and RExcel_is1", "InstallLocation",
"");
    if (location != null && location.ToString().Length > 0) RServerLoc =
location.ToString();
    else
    {
        location =
Microsoft.Win32.Registry.GetValue(@"HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\C
urrentVersion\Uninstall\R (D)COM Server_is1", "InstallLocation", "");
        if (location != null) RServerLoc == location.ToString();
    }
```

Code 4. Code looking into registry to find location of R Server program

In the same way, we can find the installation path of R, but it could be in the DOS format (e.g. “C:\PROGRA~1\ R\R-2.4.1\bin\R.EXE” for “C:\Program Files\R\R-2.4.1\bin\R.exe”), so it must be transformed as in the following code:

```
    string Rpath; // Variable that will contain R install path
    Microsoft.Win32.RegistryKey key =
Microsoft.Win32.Registry.LocalMachine.OpenSubKey(@"SOFTWARE\R-core\R");
    if (key != null)
    {
        Rpath = key.GetValue("InstallPath", "").ToString(); // Can be in DOS format
        if (System.IO.Directory.Exists(Rpath)) // Transform to complete path
            Rpath = (new System.IO.DirectoryInfo(installedPath)).FullName;
    }
```

Code 5. Look into registry to find R location, and transform from DOS to standard format

CONCLUSION

In this paper we have presented key components that were used to develop a statistical software application that uses R for calculations. The resulting product, called Decimaker, is used within the pharmaceutical, biotechnology and medical device industry to design and simulate adaptive trials and to perform Bayesian decision analyses.

During software development, significant time and resources were spent on searching, assessing and in integrating existing components into the final product. The resulting product is a true software, producing accurate results, fast computations and very nice and usable graphical outputs. This couldn’t be accomplished without the help of the open source community that developed key products. We hope by our sharing our learning and know-how from this project, we may also contribute slightly to other future developments.

Our reliance on R for the calculations provided an undeniable advantage to insure quality of results, speed up development and produce a quite versatile end product. As a result, Decimaker is not a black-box. It allows to import statistical models from R, define R formulas and queries, and export analysis and simulation results to R objects. A downside is that users are bound to install all required R components: R, R Server, R packages in addition to our software. This process was greatly streamlined using the installation checker and assistant tool that was described.

REFERENCES

1. *The R Project for Statistical Computing*. http://www.r-project.org/.

2. *R COM Server: Index.* http://sunsite.univie.ac.at/rcom/server/index.html.

3. **Manjaly, T.** *C# Coding Standards and Best Programming Practices.* [http://www.codeproject.com/KB/cs/c__coding_standards.aspx]

4. *Software testing - Wikipedia, the freee encyclopedia.* http://en.wikipedia.org/wiki/Software_testing.

5. **Administration, U.S. Department Of Health and Human Services Food and Drug.** *General Principles of Software Validation; Final Guidance for Industry and FDA Staff.* [http://www.fda.gov/cdrh/comp/guidance/938.html]

6. **Gogates, Gregory D.** *Software Validation in Accredited Laboratories A Pratical Guide.* [http://a2la.org/guidance/adequate_for_use.pdf]

7. *subversion.tigris.org.* http://subversion.tigris.org/.

8. *tortoisesvn.tigris.org.* http://tortoisesvn.tigris.org/.

9. **Bodsworth, Garry.** *Garry's Bit Patterns: TortoiseSVN and Visual Studio Integration.* http://garrys-brain.blogspot.com/2007/07/tortoisesvn-and-visual-studio.html.

10. **Trager, Corey.** *BugTracker.NET Home - Free Bug Tracking.* http://ifdefined.com/bugtrackernet.html.

11. *FxCop*. http://msdn.microsoft.com/en-us/library/bb429476(VS.80).aspx.

12. *Testlink.* http://testlink.org.

13. *Open Source Testing Survey.* http://www.opensourcetesting.org/survey.php.

14. *Component Factory - User Interface Controls and Componants for .NET Windows Forms (WinForms)*. http://www.componentfactory.com.

15. *Main Page - ZedGraphWiki.* http://zedgraph.org.

16. **Bauer, A.J.** *CodeProject: A fast and performing gauge. Free source code and programming help.* http://www.codeproject.com/KB/miscctrl/A_Gauge.aspx.

17. *GIMP - The GNU Image Manipulation Program*. http://www.gimp.org/.

18. **James, Mark.** *famfamfam.com: Silk Icons.* http://www.famfamfam.com/lab/icons/silk/.

RECOMMENDED READING

WEBSITES

- Visit http://www.opensourcetesting.com/ if you are looking for open source software testing tools
- http://www.codeproject.com/ contains gigantic resources for developers, including free controls and articles
- http://msdn.microsoft.com/en-us/library/czefa0ke.aspx presents Microsoft's Design Guidelines for class Library Developers for writing robust and easily maintainable code by using the .NET Framework

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Julien Vanwinsberghe
ClinBAY
Rue Banterley 80
1471 LOUPOIGNE
Belgium

julien@clinbay.com
http://www.decimaker.com
http://www.clinbay.com