DATA INTEROPERABILITY IN A EMC SIMULATION SOFTWARE TOOL

D. Cheremisinov [1] and I. Charamisinau [2]

[1] - Institute of Engineering Cybernetics of National Academy of Sciences of Belarus, Surganov str., 6, Minsk, 220012, BELARUS, cher@newman.bas-net.by
[2] - South Methodist University, 3160 SMU Blvd, Dallas, TX, 75275, USA charamisinau@yahoo.com

ABSTRACT

We present a domain-independent model of software system design and construction that is based on interchangeable software components and large-scale reuse. The model is a successful example of software component/building-block technologies. Building-block technologies exploit large-scale reuse, rely on open architecture software, and elevate the granularity of programming to the subsystem level. Considerable work is being done in this area, but the trend is toward object-oriented integration facilities, spearheaded by the Object Management Group's CORBA[1] standard and Microsoft's COM[2] dispatch and automation facilities. One (namely COM) of these current, highly visible, well-publicized efforts is discussed in this paper.

1. INTRODUCTION

Taking into account that the cost of avionics approaching 40 per cent of the total fly-away costs of aircrafts or helicopters, and the need to support, cost-effectively, a mission system update cycle through the expected 30+ years life cycle, it is clear that the research into the prediction and control of undesirable interactions between on-board and external electronic systems caused by environmental electromagnetic effects are really important factors in the development of aircraft electronic equipment. This research is laborious and can't be done without the usage of some software tools, which apply the computational electromagnetic, equipment knowledge, aircraft dynamics, and flight paths. To determine the antenna placement on an aircraft we must use all these software tools in design process. The data interoperability of software tools is vital in this design process.

In [3] were presented a software tool developed for electromagnetic compatibility analysis. The software product is called EMC-Analyzer. It is designed for the analysis and prediction of intrasystem electromagnetic compatibility for board as well as ground radio-electronic systems. The EMC-Analyzer can be used for EMC analysis and prediction in a variety of systems: avionics, space/missile systems, communication systems, radar systems, navigation systems and others. The EMC-Analyzer is a tool researchers would like to use because it provides ready to use features and have the build in integration capability.

The EMC-Analyzer performs three main types of analysis: intrasystem EMC linear analysis, aircraft surface geometry calculation and nonlinear receiver simulation.

2. INTEGRATION

The days of solving problems with monolithic applications are gone. Almost all real-world systems today are built by harnessing individual applications in ways dictated by a design process. This means integration.

Application integration can be defined as combining multiple standalone applications, or parts thereof, into a new or greater "distributed" application. This can be done in many ways, such as by having one application to use the functionality of another application, or by having applications running on heterogeneous systems to communicate by passing messages using a common syntax. Either way, application integration allows applications to work together to achieve a greater performance than either application could achieve alone.

This "working together" requires some level of interoperability—the ability to work together—between systems. Building applications that share information with each other is not an easy task. This is especially difficult, when the applications in question were not built at the same time, or even by the same group in the company. Difficulties are almost guaranteed to arise when your application needs some services or functionality already provided by an existing application.

We have always faced problems getting heterogeneous applications to talk to each other. Even in the very best cases the scope of this problem is huge, from native messaging protocols to incompatible operating systems to desktop applications that need to exchange data.

Today, we have several options for performing distributed communication. The first is DCOM, it rules the Microsoft world. While it's easy to use in client/server applications where both the client and server run a Microsoft operating system, DCOM becomes more difficult or impossible to use when trying to achieve interoperability between multiple environments.

CORBA and EJB[4] have been building support in non-Microsoft development groups. But like DCOM, CORBA and EJB are often tied to proprietary hardware or software systems.

Another approach is IP messaging (for example, open a socket to port 9981 on server 127.0.0.1). While this approach lets the client and server portions of your application communicate directly, problems can still arise. For example, when an application needs to communicate through a proxy server or a firewall, it must now open ports on a firewall. Many system administrators frown on opening,additional ports to the Internet. Even more problematic with the direct IP approach is the fact that an application using pure socket connections is tied to a specific IP—so scalability quickly becomes an issue.

Component-based software development focuses on building large software systems by integrating previously existing software components. By enhancing the *flexibility* and *maintainability* of systems, this approach can potentially be used to reduce software development costs, assemble systems rapidly, and reduce the spiraling maintenance burden associated with the support and upgrade of large systems. At the foundation of this approach is the assumption that certain parts of large software systems reappear with sufficient regularity that common parts should be written once, rather than many times, and that common systems should be assembled through reuse rather than rewritten over and over.

In contrast to traditional development, where system integration is often the tail end of an implementation effort, component integration is the centerpiece of the approach; thus, implementation has given way to integration as the focus of system construction. Because of this, integrability is a key consideration in the decision whether to acquire, reuse, or build the components.

2.1. COM Automation and Scripting

COM automation and Scripting is an environment for integrating applications and data sources to control projects involving a variety of applications and data sources.

COM automation allows applications to communicate, exchange data, and control one another. Specifically, automation allows a client application to create and control an object, using the exposed object's interface. Automation is powerful because programmers can use this technology to access the functionality of any application that supports Automation. For example, if you are the need in a calculation that don't realized in EMC-Analyzer, you could write the calculation engine yourself, but Automation gives you another option. Using Automation, you can access the calculation engine of Microsoft Excel and get Excel to do the calculations for you. In this way, developers can use the functionality of existing applications without replicating the same functionality in their own applications.

COM is a protocol that enables software components to communicate directly. COM is a binary standard. A fully compliant COM object can be written in any language that can produce binary compatible code. So you can write COM objects using C, C++, Java, J++ or Visual Basic. All of the Windows NT shell has been written using COM.

EMC-Analyzer is developed as part of a distributed application and hence it is a candidate for future reuse. Organizing the development process around the component paradigm lets designer continuously raise the level of functionality in new applications and reduce time-to-market by building on previous work. COM in EMC-Analyzer is the glue that makes component integration work.

The scripting is a capability allows designer to use VBScript to automate operations throughout EMC-Analyzer. *VBScript* (Visual Basic Scripting Edition), a subset of the Microsoft Visual Basic programming language, is a fast, portable, lightweight interpreter for use in applications that use Microsoft Automation servers. . With it, designer can glue together and drive the various objects of EMC-Analyzer or any other available ActiveX-compatible server, because EMC-Analyzer is scripting host[5]. The script language is a flexible and powerful tool to manipulate objects from the programming model of EMC-

Analyzer and any automation object installed on your machine.

The script is the program written on script language. Any COM Server object that is installed on your machine is eligible for use by the scripting host. The programming model of EMC-Analyzer is not comprehensive, but designer by script can access everything that exposes a COM-compliant interface.

2.2. Data Interoperability by COM Automation and Scripting

COM automation and scripting can be used to design a lot of different systems involving different applications and data sources. For example, you can use COM automation and scripting to integrate EMC-Analyzer with other applications such as Excel, AutoCAD and Mathcad.

Data can be exchanged between EMC-Analyzer and any object that supports COM Automation. You use the Scripting host component of EMCA to write a custom script for an object so that it behaves following manner:

- Accepts values from EMC-Analyzer.
- Activates the server application or control to manipulate the data when the EMC-Analyzer is running.
- Sends values to EMC-Analyzer.

For example, you can design a project for EMC-Analyzer to send spectra to a MS Excel worksheet, where the data are transformed and output back to EMC-Analyzer.

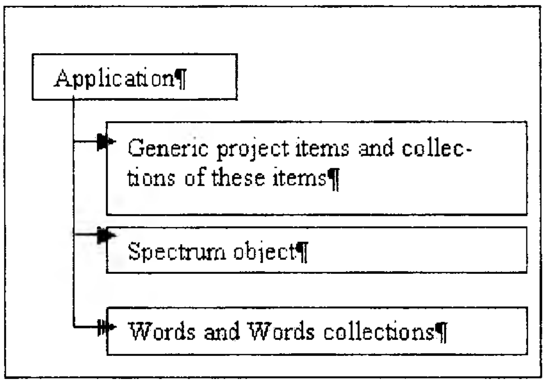
EMC-Analyzer can be used as a component in other application. These systems, in which EMC-Analyzer will be used as component, must be used as container for COM objects. For example, you can design a worksheet for MS Excel in which EMC-Analyzer project run and the data from EMC-Analyzer project are transformed and output back to EMC-Analyzer.

3. Programming Model of EMC-Analyzer

The programming model of EMC-Analyzer is the set of interfaces, properties, methods, and events exposed from a program EMCA that allows a developer to write another programs to manipulate it. The programming model presents a *logical* view of the functionality as opposed to the *physical* reality of that functionality. It expresses things in a way that matches how the intended user thinks and not necessarily how the system actually works. The programming model of EMC-Analyzer expresses its functionality in a way that matches how the developer wants to think about it. The programming model of EMCA doesn't just expose internal structures—it exposes its functionality at a higher level of abstraction so that the customer (the developer) can concentrate on what the customer *wants* to do and not on *how* one has to accomplish a simple task.

Now the programming model of EMC-Analyzer consists from following objects: Application, Spectrum, Generic project item, Word and Words collection. Each object is realized as COM interface. Software objects play an important role in Automation. Like objects in Microsoft Visual Basic, these components contain properties and methods. In fact, objects that are accessible through Automation behave just like the tools in the Visual Basic toolbox—if you know the properties and methods, you can easily use the component.

An object model of EMCA is a hierarchical representation of the objects contained in an application and their interrelationships.



Application¶
Generic project items and collections of these items¶
Spectrum object¶
Words and Words collections¶

Figure 1. The EMC-Analizer object model

4. Visualisation of Ground Complex

COM automation and Scripting offer a new capability for advanced users of EMC-Analyzer. They can apply the EMC-Analyzer to aims don't was foreseen initially.

Now EMC-Analyzer doesn't have ability to visualize a location of antennas of ground complex. But the advanced user can do this work by script. In this script he can collect information about coordinates of antennas in all ground complexes of the radio-electronic systems. The text of this piece of script is shown on fig. 2.

```
Set Collect =
EMCA.FindTag(TAG_GROUND_COMPLEX)
i=Collect.Count
Mn=0
Mw=0
Mh=C
For Each s In Collect
    Set Child=s.Childrens
    For Each a In Child
        If a.Tag=TAG_ANTENNA Then
        MarkUp=a.CoordMarkUp
        MarkUp1=a.ParamMarkUp
        h=a.Number(MarkUp, 11)
        If Mh<h then
   Mh=h
   end if
        n=a.Number(MarkUp, 15)
        If Mn<n then
   Mn=n
   end if
        w=a.Number(MarkUp, 16)
        If Mw<w then
   Mw=w
   end if
        End If
    Next
Next
```

Figure 2. The script to collect information about coordinates of antennas in all ground complexes of the radio-electronic systems.

In this script the *EMCA* is predefined name of EMC-Analyzer application object. The variable *Collect* contains a collection of objects. The value of this variable is a result of EMC-Analyzer application object method call. Each member of this collection is project item. Each project item object has a property *Tag* to represent a purpose of this object. *Tag* values are predefined constants. The call of the method

FindTag(TAG_GROUND_COMPLEX)

returns the collection of objects represented ground complexes of the system. Each project item object has a property *Childrens* to represent a collection of objects, which are contained in this object. The values of variables *Mn, Mw, Mh* are the volume of a system.

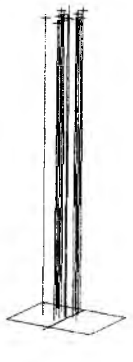


Figure 3. The fake aircraft to visualize coordinates of antennas in ground complexes.

Then designer can create the fake board complex of aircraft with very thin fuselage. The wings of this aircraft are the polygon with the size of group of ground complexes. To visualize the locations of antennas of ground complex group he calls the visualizer of aircraft on EMC-Analyzer. The text of this piece of script is not shown.

In this example EMC-Analyzer control itself. This example shows the enhancing the *flexibility* of EMC-Analyzer by proposed integration technique.

5. A Control of EMC-Analyzer

We consider situation when designer have a device description in a literature and wish to add this device to the EMC-Analyzer project. In this example another application control EMC-Analyzer.

The designer can scan this literature text and convert data into form of MS Excel spreadsheet (see fig. 4). This is a diagram of horizontal polarisation of antenna. Then he connect *type library* of EMC-Analyzer and write a macro on Visual Basic.

If user runs this macro MS Excel open EMC-Analyzer project and find device (receiver *R*) in this project. Then it adds a new port in this device, create a new device, which type is antenna, and connect new antenna to new port in receiver *R*.

0	-5
20	-10
40	-15
60	-30
80	-40
100	-20
110	-30
120	-45
140	-50
160	-65
180	-70
200	-65
220	-50
240	-45
260	-30
280	-20
300	-40
320	-30
340	-15
350	-10

Figure 4. Diagram of horizontal polarisation of antenna as MS Excel spreadsheet.

The first piece of code in this script is a creation of EMC-Analyzer object. If property *Visible* is set to TRUE, the application window is visible on desktop of computer. By call of a method *OpenProject* the project is opened.

In the next fragment the new port in receiver *R* is created. Initially the blank project item is created, and this item is cast to port type connected to receiver *R* and is initialised by setting the appropriately mark-up string. The new antenna item in the next fragment is created in the same way. This antenna included into a ground complex *R_22_92*.

The next piece of code is a creation of mark-up string to set the parameters of new antenna. Looping cells of Excel spreadsheet makes this string.

Figure 5. Macro of MS Excel to move spreadsheet table to EMC-Analyzer project.

The purpose of last fragment is a creation of a new link to connect the port with a new antenna.

This example shows the enhancing the *maintainability* of EMC-Analyzer by proposed integration technique.

```
Sub addAntenna()
Set emca=New AuEmcaLib.Application
emca.Visible=True
emca.OpenProject
("e:\examples\Ant_Table\Ant_table_DN1.prj ")

Set p=New AuEmcaLib.GItem
p.Init
Set r=emca.Find(TAG_RECEIVER, "R")
res=p.NewDevice(TAG_PORT, "qq", r.Item)
par="@NUMBERMINAF@1M@UNITHz@NUMESF1@1@
SUBTYPE@501"
p.ParamMarkUp=par

Set a=New AuEmcaLib.GItem
a.Init
res=a.NewDevice(TAG_ANTENNA, "Ant",
emca.Find(TAG_BOARD_COMPLEX, "R_22_92").Item)
BStr="@CHOICEPT@vertical@NUMBERAL@1@UNITm
@NUMBERGain@6@UNITdB@TABLEAC_HRP@"

j=1
While Selection.Cells(j, 1).Value <> ""
BStr=BStr & Selection.Cells(j, 1).Value & ","
BStr=BStr & Selection.Cells(j, 2).Value & ","
j=j+1
Wend
BStr=BStr & "@UNITdeg,dB  @TABLEAC_VRP@"
BStr=BStr & "-55,-80,-45,-60,-10,-60,0,0,10,-60,45,-60,55,-
80,@UNITdeg,dB @SUBTYPE@110"
a.ParamMarkUp=BStr

Set ws=New AuEmcaLib.Words
Set w1=New AuEmcaLib.Word
w1.Tag=p.Item
w1.Text="to"
ws.Add(w1)
ret=a.NewLink(ws)
Set ws=Nothing
Set w1=Nothing
```

6. CONCLUSION

COM Automation and scripting makes it easy to write a distributed application that

- Scales from the smallest single computer environment to the biggest pool of server machines.
- Provides rich, symmetric communication between components.
- Takes advantage of existing custom and off-the-shelf components.
- Integrates teams proficient in any programming language and development tool.
- Uses network bandwidth carefully, while providing fast response times for end-users.
- Can be used with any network protocol and integrated into any hardware platform.
- Can for sure take advantage of other Internet standards and protocols.

The most powerful and advanced part of the EMC-Analyzer makes it possible to carry out comprehensive and sophisticated simulation of receivers taking into account nonlinear effects (intermodulation, cross-modulation, desensitisation, conversion at the local oscillator noise and harmonics and so on) with very precise spectra representation: up to one million sample frequencies can be simulated in dozens of minutes on PCs with Pentium or compatible processors.

If we use EMC-Analyzer as component, possibility to simulate nonlinear effects is available to EMC engineers because it provides ready to use this feature.

REFERENCES

[1] OMG. *The Common Object Request Broker: Architecture and Specification V2.0.* Object Management Group, Inc., Formal/97-02-25 edition, July 1996.

[2] Microsoft Corporation. The Component Object Model Specification (MSDN Library, Specifications). Available at http://msdn.microsoft.com/isapi/gomscom.asp?TARGET=/com/comintro.htm

[3] Charamisinau I.D., Mordachev V.I. Intrasystem EMC simulation software tool based on expert system approach, *3rd Int.Conf. New Information Technologies*, 2,1998, p.33-35.

[4] Roman E. Mastering Enterprise JavaBeans and the Java 2 Platform, Enterprise Edition John Wiley & Sons 1999. 752 p.

[5] Esposito D. Scripting in Windows 95, 98 and NT Operating Systems from Microsoft Interactive Developer