Journal of Engineering and Development, Vol. 16, No.2, June 2012 ISSN 1813- 7822

# *A JavaBeans and DCOM Comparison V.S. Dynamic Link library and it's Role in Distributed Real Time System*

*Assistant lecturer: Mais Abid Khalil*
*Computer and Software Engineering Department*
*Al_Mustansiriay University*

## Abstract

***The main objective of this paper is to investigate the software models like (Javabeans, DCOM, and dynamic link library types) to point the strengthes and weaknesses points of each model, also explain the different types of DLL.s and their impaction on the distributed real time system and network perfomance compared to Javabeans and DCOM models.***

**Key words**: JavaBeans, DCOM, DLL, Distributed real time system

**الخلاصة**

***الهدف الرئيسي من البحث هو فحص خصائص المكونات البرمجية مثل جافابينز والديكوم وشرح انواع مكتبة الارتباط الديناميكي DLL لنثبت نقاط القوة والضعف لكل من هذه المكونات, كذلك توضيح الانواع المختلفة من مكتبات الارتباط الديناميكية وتأثيرها على نظام الزمن الحقيقي الموزع وعلى كفاءة الشبكة مقارنة الى المكونات البرمجية مثل جافابينز والديكوم.***

## Introduction

Different software models like (JavaBeans ,DCOM, DLLs…etc) lead to reduced time to market, improved software quality, and less maintenance effort. The reusability of components has to be increased in order to realize this vision. We need global component models in order to allow the creation of global component markets. Additionally, methods, techniques, tools and process models have to be developed and adapted for the component-based creation of software systems. But before component-based software technologies can successfully contribute to improved software engineering, several questions have yet to be answered. For example:the impaction of each component on distributed real time systems? on network perfomance?.

Distributed object technologies such as Java RMI(remot method invocation) and Microsoft's DCOM allow objects running on one machine to be used by client applications on different computers. Distributed computing allows a business system to be more accessible. Distributed

systems allow parts of the system to be located on separate computers, possibly in many different locations, where they make the most sense. In other words, distributed computing allows business logic and data to be reached from remote locations. Customers, business partners, and other remote parties can use a business system at any time from almost anywhere. The most recent development in distributed computing is *distributed objects*[1]. The JavaBean architecture is based on a component model which enables developers to create software units called *components.*

Some examples of Beans are(GUI (graphical user interface) ,Non-visual beans, such as a spelling checker ,Animation applet ,Spreadsheet application….etc)[2]. DCOM architecture which based on COM technology ties togather the active client and active server,the word active refers to interactive nature objects created for the client and server.

The DLL abbreviation stands for Dynamic Link Library. it is made of functions and/or other resources grouped in a file. opposed to a static library, a DLL allows the programmer to decide on when and how other applications will be linked to this library. For example, a DLL allows difference applications to use its library as they see fit and as necessary. In fact, applications created on different programming environments can use functions or resources stored in one particular DLL. For this reason, an application dynamically links to the library[3].

## JavaBeans

JavaBeans is an object-oriented programming interface from Sun Microsystems that lets you build re-useable applications or program building blocks called components that can be deployed in a network on any major operating system platform. Like Java applets, JavaBeans components (or "Beans") can be used to give World Wide Web pages (or other applications) interactive capabilities such as computing interest rates or varying page content based on user or browser characteristics.

From a user's point-of-view, a component can be a button that you interact with or a small calculating program that gets initiated when you press the button. From a developer's point-of-view, the button component and the calculator component are created separately and can then be used together or in different combinations with other components in different applications or situations.

When the components or Beans are in use, the properties of a Bean (for example, the background color of a window) are visible to other Beans and Beans that haven't "met" before can learn each other's properties dynamically and interact accordingly.

Beans are developed with a Beans Development Kit (BDK) from Sun and can be run on any major operating system platform inside a number of application environments (known as *containers*), including browsers, word processors, and other applications.

To build a component with JavaBeans, you write language statements using Sun's Java programming language and include JavaBeans statements that describe component *properties*

such as user interface characteristics and *events* that trigger a bean to communicate with other beans in the same container or elsewhere in the network.

Beans also have *persistence,* which is a mechanism for storing the state of a component in a safe place. This would allow, for example, a component (bean) to "remember" data that a particular user had already entered in an earlier user session[4].

JavaBeans is based on Java's object model,which allows the usage of classes and objectsin a well-known manner. Interfaces to beans are defined by a set of methods that defineproperties and events of beans. Interfaces of beans may be deducted from interfaces of Java classes,which can be done by means of the reflection and introspection mechanism. However, interfaces of beans have to adhere to certain conventions, called patterns in Java- Beans terminology.For example, properties are defined by a pair of setter- and gettermethods,a property *name* is defined by two methods *setName* and *getName,* presuming they have the right parameters.

A bean can be represented by a single simple Java class. A bean can also comprise many Java classes as well as resources like images and videos.Complex beans are usually stored in and distributed by means of compressed archives, i.e., JAR files.

Java's serialization mechanism is used to make bean instantiations persistent. At run-time, Java's garbage collection is used to get rid of any bean instantiations that are not in use anymore.

JavaBeans had not been designed for distributed systems. However, beans may communicate across process boundaries, e.g., over the Internet,by means of Java's *remote method invocation* (RMI) mechanism.

RMI provides a transparent communication vehicle among objects and, thus, beans, that run on different virtual machines (JVM). Services needed for distributed JavaBeans architectures have been considered in Enterprise JavaBeans (EJB)[5].

## Distributed Component Object Model (DCOM)

The key aspect of COM is that it enables communication between components, between applications, and between clients and servers through clearly defined interfaces. Interfaces provide a way for clients to ask a COM component which features it supports at runtime. To provide additional features for the component, it was shown that it simply add an additional interface for those features. A COM interface is a structure that contains a pointer to a structure containing pointers to functions, while still maintaining the needed high-speed performance of a DLL [6].

The Distributed Component Object Model (DCOM) is a set of extensions to the COM environment and was developed and shipped after the COM system was deployed into the market place [7].

It allow application objects to be distributed over multiple computers, handling the communications among the objects as well as the instantiation and execution of the objects remotely.

Microsoft has taken the Component Object Model, to another level with the introduction of DCOM. DCOM extends COM by enabling application objects to communicate and run across networks, including the Internet, intranets, LANs, and WANs [8].

### Development environment

The acquisition of a proper development environment or builder tool for developing Java-Beans was a time consuming process at the beginning of the implementation phase. The market for JavaBeans development environments was still immature and proper tools with the desired functionality were missing. The market has changed and now various tools object model of Java. Therefore, we only had to implement an additional service for the distribution of events across process boundaries (remote event service)[5].

Although DCOM specification is not geared toward a specific language, The Microsoft Foundation Class Library (MFC) is the easiest and robust choice of all the tools available for DCOM development with C++. MFC does provide a number of features and functions when developing DCOM [8].Résumé: DCOM regarded to be superior to JavaBeans with regard to the development environments.

### Distribution

RMI is one possible communication mechanism for distributed Java programs. It is an *application programming interface* included in the JDK. Remote objects are accessed via their interfaces which allows a homogenous integration of local and remote method invocations. RMI does not offer services like an event distribution service. We implemented such an event distribution service with low effort [5].

DCOM extends applications across networks, including the Internet, allowing components to run on different machines. DCOM allows components to communicate with each other via any network protocol, including TCP/IP and IPX/SPX [8].

Résumé: More complex COM objects can require substantial amounts of system overhead, particularly objects that are distributed among multiple computers. Late binding is an interface technique that increases the component's run-time flexibility but the host application must incur a performance penalty as a result. JavaBeans had not been designed for distributed systems but instead it suitable for internet by RMI mechanism.

### Language Independence

Normally JavaBeans are written in Java and the code is compiled to Java byte code. JavaBeans can, in principal, be written with other languages as long as the compiler compiles the programmed code into Java byte code[5].

Since DCOM is based on COM, the language neutrality of COM is extended to DOM. DCOM components built with different languages can communicate over a network [8].

Résumé: DCOM is superior to Java-Beans with regard to language independence.

### Platform Independence

One important aspect of the popularity of Java is its operating system independence. This independence is still not reached at all points and a final test of software systems on at least Unix and Windows platforms is required [5]. DCOM is designed to run on multiple platforms. Microsoft is openly licensing DCOM to other software companies to run on all major operating systems. Microsoft is also working with the Internet standards committees to make DCOM an Internet standard [8].
Résumé: JavaBeans is superior to DCOM with regard to platform independence.

## Dynamic Link Library (DLL)

If programmer want to write modular software, he'll be very interested in dynamic link libraries (DLLs). he probably thinking that he has been writing modular software all along because C++ classes are modular. But classes are **build-time** modular, and DLLs are **runtime modular**. Instead of programming giant EXEs that programmer must rebuild and test each time he make a change, he can build smaller DLL modules and test them individually. he can, for example, put a C++ class in a DLL, which might be as small as 12 KB after compiling and linking. Client programs can load and link programmer DLL very quickly when they run. DLLs are getting easier to write. there's more and better support from AppWizard and the Microsoft Foundation Class (MFC) library[10]. Dynamic linking differs from static linking in that it allows an executable module (either a DLL or .EXE file) to include only the information needed at run time to locate the executable code for a DLL function. In static linking, the linker gets all of the referenced functions from the static link library and places it with programmer code into the excutable[10] (see Fig 1) .

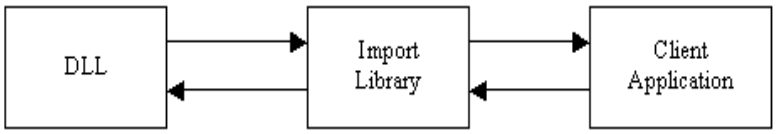


***Figure 1: DLL fundamental concept***

## Create MFC DLL

Microsoft Foundation Class (MFC) library can be used to create simplified DLLs. The MFC supports two types of DLLs. Those are the following.

1. Regular DLL with MFC statically linked
2. Regular DLL using shared MFC Dll
3. MFC Extension DLL ( Using shared MFC DLL)

In the regular DLL with MFC statically linked, the Client may be MFC based application or Non-MFC application. MFC Extension DLL has reusable classes derived from existing Microsoft foundation Classes.

the differences between **MFC library regular DLLs** and **MFC library extension DLLs** are explained the following sections.

Each DLL has some kind of interface. The interface is the set of the variables, pointers, functions or classes provided by the DLL that you can access from the client program. It is the interface that allows the client program to utilize the DLL.

## Regular DLL V.S. Extension DLL

The MFC extension DLL only works with MFC client applications. If you need a DLL that can be loaded and run by a wider range of Win32 programs, you should use a regular DLL. The downside is that your DLL and your client application will not be able to send each other pointers or references to MFC-derived classes and objects. If you export a function from your DLL, that function cannot use MFC data types in any of its parameters or return values. If you export a C++ class from your DLL, it cannot be derived from MFC. You can still use MFC insideyourDLL.

Your regular DLL still needs to have access to the code in the MFC code library DLL. You can dynamically or statically link to this code. If you dynamically link, that means the MFC code your DLL needs in order to function is not built into your DLL. Your DLL will get the code it needs from the MFC code library DLL found on the client application's computer. If the right version of the MFC code library DLL is not there, your DLL won't run. Like the MFC extension DLL, you get a small DLL (because the DLL doesn't include the MFC code), but you can only run if the client computer has the MFC code library DLL.

If you statically link to the MFC code library, your DLL will incorporate within itself all the MFC code it needs. Thus, it will be larger, but it won't be dependent on the client computer having the proper code library. If you can't rely on the host computer having the right version of MFC available, you should use static linking. If you have control over which versions of the MFC DLLs are installed on the client computers, or if

your installation program also loads the right MFC DLL, dynamic linking would be appropriate[10].the following simple example project will illustrate the MFC DLL types , The work space consists of four projects, TheApp, DllA, DllB, and DllC.

TheApp depends on all three DLLs, DllA depends on DllB, DllB has no dependencies, and DllC depends on the other two, DllA and DllB. Each of the DLLs were created with the AppWizard(DLL) with DllA as an Extension DLL using shared MFC DLL, DllB a MFC Extension DLL, and DllC is a Regular DLL. These terms refer to the options available during the AppWizard setup of a DLL. The dependencies are illustrated in Figure 2.

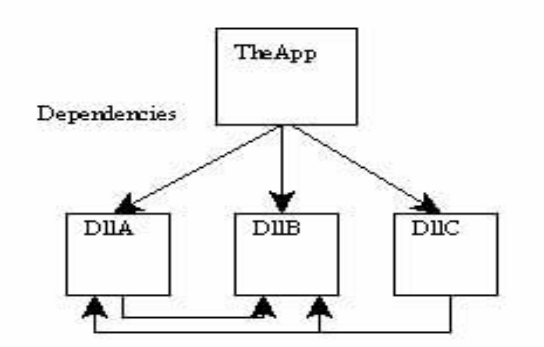


***Figure 2. TheApp Project Dependencies***

## A Real-Time system with DLL

Real time systems operate on strict timing constraints,if the programmer use DLL thread, A real-time thread living inside a native Win32 DLL receives an interrupt from an external source. The thread processes the interrupt and stores relevant information to be presented to the user. On the right side, a separate UI (user interface) thread, written in managed code, reads information that was previously stored by the real-time thread. Given the fact that context switches between processes are expensive, you want the entire system to live within the same process. If you separate real-time functionality from user interface functionality by putting real-time functionality in a DLL and providing an interface between that DLL and the other parts of the system, you have achieved your goal of having one single process dealing with all parts of the system. Communication between the UI thread and the real-time (RT) thread is possible by means of using Platform/Invoke to get into the native Win32 code.

## Conclusion

The basis of the comparison was a qualitative approach and not a quantitative measurement.It is hard to say whether the component is reliable,correct, efficient, secure, etc. So even forhigh quality components it may be a better solution to invest in an own development. In case someone had to choose between the component models, several factors should be taken into account.the strengths of JavaBeans are platform independent and the internet services but it can’t used in distributed systems also it is language dependent.the strengths of DCOM are language independent,allow application objects to be distributed over multiple computers,if DCOM is complex and build with huge MFC Library and remot computer has no MFC,remote computer take long time to load MFC,so the network perfomance degradation will occure,also DCOM are platform dependent.

For DLL component Model it shares the memory. So, the system performance is improving compared to using applications, can build and test separately each DLL, can create DLLs for different languages, load and unload at run time. This helps to improve application

performance, also big software products can be divided into several DLLs, so The developers easily develop their application.The programmer can use deffirent types of MFC DLL.s(Extension and regular). for an MFC extension DLL to work, both the extension DLL and the client program must dynamically link to the same MFC code library DLL, and this must be available on The computer where the application is running.it very small has a size of 10-15 KB.

Regular DLL is stand alone(non-MFC) library,so it can work without MFC to be installed on the client. can dynamically or statically link to MFC ,therefore regular DLL may be large or small.also DLL is relatively suitable solution for real time and distributed systems problems of compiling time overhead , the functionality provided by the common language runtime to enable managed code to call unmanaged native dynamic-link library (DLL) entry points it help to reach the goal of achieve the real time system constraints.

## References

[1]By Richard Monson-Haefel, Enterprise java Beans, Second Edition, Second edition, published March 2000.

[2]A Sun Developer Network Site, JavaBeans Concepts,2010.

[3]Live United, Win32 DLL, FunctionX Press books, 2006.

[4]Susan Adams and Orjan Timan , what is JavaBeans?, TechTarget,2010.

[5]D. Birngruber, W. Kurschl, J. Sametinger, Comparison of JavaBeans and ActiveX, Johannes Kepler Universität Linz, A-4040 Linz, Austria, 2001.

[6]Kasabov N., ”**ActiveX Tutorial**”, TechiWearHouse, 1998.

[7]Dean T., Chris Exton, Leah Garret and A.S.M Sajeev, Monash University,“**Distributed Component Object Model**” Department of Software development, Faculty of computing and information technology, 1997. Mehran S., ”**ActiveX Programming with Visual C++**", Sam Publishing, 1998.

[8]C++, MFC & Dynamic Link Libraries - DLL 1, MochaHost - Professional Hosting, 2010.

[9]Havych, Creating DLL in Visual Studio dot net, Free Website Templates ,2006.