. NET

Siddhartha Bhattacharyya
Lecturer, Department of Information Technology

Imagine working as a software developer in a soft ware firm with a billion turnover. You, being adept in Visual Basic, are asked to join a team of VC++ experts to carry out a million dollar project. Simply speaking, you would be left with only two options viz. either to start learning VC++ or quit. Obviously, either way, you are in trouble. This problem is persistent with most of the software engineers all over the world. The day a new language arrives in the market, you have to learn it to steal the show.

The situation has changed all the way round with the advent of Microsoft .NET technology in July 2000. .NET is a multi-language platform (in other words – language independent) that knits various aspects of application development together with the Internet. The framework covers all layers of software development above the operating system. This language independence means that you are free to choose to work in whichever language you (or your company) want.

.NET can be thought of as a layer that exists beneath your programs and provides a set of base services and functions. This layer contains a set of applications and operating systems called the .NET servers; a foundation set of objects called the .NET framework, and a set of services that support all the .NET languages called the Common Language Runtime (CLR). Figure 1 shows the .NET components.

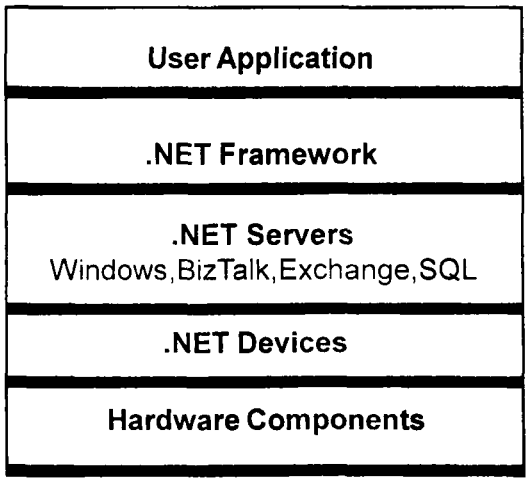

Figure 1 .NET components

- **Common Language Runtime**

Since .NET ensures cross language interoperability, a runtime environment common to all the languages is a prerequisite. The common language runtime (CLR) provides such an environment that all languages share. It manages the execution of code and provides services that make the development process easier. Compilers and tools expose the runtime's functionality and enable you to write code or assembly that benefits from this managed execution environment. The basic components of a CLR are shown in Figure 2. CLR involves a two- stage compilation process.

An assembly is not an application. An assembly is a collection of files that reside in the same directory on the disk and contains source files and resource files. Information about an assembly is contained in the Manifest of the assembly. An application is built from one or more assemblies. A language specific compiler first compiles the source code or an assembly into what is known as Microsoft Intermediate Language (MSIL) code. Microsoft intermediate language (MSIL) is a CPU-independent set of instructions that can be efficiently converted to native code. MSIL includes a wide spectrum of instructions, such as instructions for loading, storing, initializing, and calling methods on objects. It also includes instructions for arithmetic and logical operations, control flow, direct memory access, and exception handling. When a compiler produces MSIL, it also produces metadata by means of a metadata engine. Metadata is a binary information describing your code stored in a .NET Framework portable executable (PE) file or in memory. Every type and member defined and referenced in a file or assembly is described within metadata. The

metadata describes the types in your code, including the definition of each type, the signatures of each type's members, the members that your code references, and other data that the runtime uses at execution time. Metadata provides a common frame of reference that enables communication between the runtime, compilers, debuggers, and code that has been compiled into MSIL. It also helps the runtime and garbage collection keep track of memory that will be released back to the operating system when it is no longer needed. The information stored in metadata also enables the Common Language Runtime to enforce security.

The runtime uses metadata to locate and load classes, lay out instances in memory, resolve method invocations, generate native code, enforce security, and set up run time context boundaries. The MSIL and any other optimized intermediate language code or metadata from any other compiled assembly are linked to form an executable file (.exe) or a dynamic linked library (.dll). This completes the first stage of compilation.

In the second stage of compilation, MSIL is converted to CPU-specific code by a just in time (JIT) compiler. The runtime supplies one or more JIT compilers for each computer architecture, hence the same set of MSIL can be JIT-compiled and executed on any supported architecture. Depending on the degree of optimization and hardware platform, several types of JIT are in existence. They are i) Standard JIT, which produces highly optimized machine code. When an IL function or method is invoked, standard JIT analyzes, compiles, and caches it very fast. All .NET programs using standard JIT become faster as the various branches of execution within them are converted to pure native code. ii) EconoJIT, which targets small hardware platforms that do not have a lot of RAM. It produces machine code that is efficient but not optimal. Because EconoJIT performs fewer optimizations than standard JIT does, it compiles faster. iii) PreJIT, which is not really a JITter in its own right. It is the invocation of the standard JIT at the time when an application is installed on a system. With PreJIT, the entire application is completely converted from IL to machine code.

During the second stage of compilation, any class definitions and type names (or Namespaces) used by an assembly are invoked from the Base class library and verified before being JIT compiled. Thus one gets a fully compiled code rather than one which is interpreted at runtime.

Besides the two-stage compilation procedure, which makes the total process hardware independent, CLR offers the following benefits:

- Interoperability & Scalability
- Stability & Safe Deployment
- Simple Component Replication
- Code Re-use
- Automatic Management
- CLR Debugger

  The CLR supports side-by-side execution. and manages execution to use the benefits provided by the CLR, language compilers such as Visual Basic, C#, Visual C++, or one of many third party compilers such as a Perl or COBOL must target the runtime.

- **.NET Framework**

The essence of .NET lies in a genuine framework, which entails the cross language interoperability feature of the same. It provides two key things: the base runtime environment and a set of foundation classes. The runtime environment is similar to the operating system in that it provides a layer between your program and the complexities of the rest of the system, performing services for your application and simplifying access to the lower layers. The foundation classes provide a large set of functionality, wrapping and abstracting such technologies such as Internet protocols, file system access, XML manipulation, and more. The .NET Framework includes classes, own set of Application Programming Interfaces (APIs) and value types. It also includes types that encapsulate data structures, perform I/O, give you access to information about a loaded class, and provide a way to invoke .NET Framework security checks, data access, server controls and rich GUI

generation. .NET Framework objects automatically communicate and interact with each other, even if they are written in different languages. Objects can call methods on other objects, inherit implementation from other objects, and pass instances of a class to another class's methods. However, for taking advantages of the runtime environment and other functionality of the .NET framework, the compiler must produce a code that adheres to a certain standard. Microsoft provides this standard, the Common Language Specification (CLS) as a way to make a compiler .NET compatible. It includes the basic language features needed by any applications. This specification is also equally applicable for those components built by other languages. If a component is CLS compliant and uses only CLS features in the API, it is guaranteed to be accessible from any object compiled by a language compiler that supports the CLS.

A CLS-compliant code must use only CLS features in the definitions of public classes, public members of public classes, members accessible to subclasses, parameters of public methods of public classes and parameters of methods accessible to subclasses.

- **.NET Security**

Any framework including the .NET framework is prone to be endangered by brute forces. So the security features of .NET framework need special mention. .NET framework security relies on two security mechanisms: i} Code access security and ii} Role-based security. Code access security uses permissions to control the access to protected resources and operations. It helps protect computer systems from malicious mobile code and also provides a way to allow mobile code to run safely. Role-based security provides information needed to make decisions about what a user is allowed to do. These decisions can be based on either the user's identity or role membership or both. .NET framework security policy is the configurable set of rules that the runtime follows when deciding which permissions to grant to code. The runtime determines the access to resources code by examining identifiable characteristics of the code, such as the web site or zone from which the code originated. Based on policy, the runtime grants permissions to both assemblies and application domains. During execution, the runtime ensures that code accesses only the resources that it has been granted permission to access. Some of the .NET framework security tools are:

- Assembly generated utility
- Caspol
- cert2spc
- Certmgr
- Chktrust
- Makecert
- Permview
- Peverify
- Secutil
- Setreg
- Sn
- Storeadm

- **.NET Servers**

A major goal of .the NET concept is to reduce the building of distributed systems, in which the work is done at different locations at the server level. Microsoft provides a set of software products that together are known as the .NET Enterprise Servers. They are designed to supply the back end features needed by a distributed system. These products include:

- The server operating system: MS Windows (Server, Advanced Server, and Datacenter Server)
- Clustering, load balancing, synchronization and deployment software such as MS App Center and MS Cluster Server
- A Database Server: MS SQL Server
- An e-mail, collaboration, and free-form information storage system: MS Exchange Server
- A data-transformation engine based around XML called MS BizTalk Server
- eSecurity and Acceleration Server
- A server for accessing legacy systems such as AS/400, called Host Integration Server
- Mobile Information Server

.NET Services

.NET includes certain concepts that extend beyond the details of programming to describe how systems should be built and how they can interact. One such key concept is the idea of Web Services, functionality delivered in a consistent fashion over the Internet. These services enable a company to supply functionality so that the execution of the functionality is completely contained within their environment. An example is a bill-payment service through which the company can handle bill-payment. The company in addition can provide this bill-payment service to other companies as well through a Web Service. One interesting feature of Web Service is that it can be written on any operating system and in any programming language. Web Services not only allows two programs to communicate over the Internet, they also allow one program to combine information from multiple Web Services into a single application, potentially increasing the value of the data. Web Services communicate with one another and with clients using standard Internet protocols. They can use the common HTTP protocol. Alternatively, a Web Service can use the SOAP (Web Service Wire Format) to communicate with a client or other service. Using SOAP allows the communication to be richer because SOAP allows for objects to be passed between the two applications, whereas HTTP does not. The messages (or methods) supported by a Web Service is described by a Web Service Description Language (WSDL). A particular Web Service is identified by a Web Service Discovery (DISCO) file. It defines the location of the WSDL file, the location of the Web Service and the namespace defining the format of the WSDL file.

- **.NET Devices**

.NET devices refer to a wide range of systems that allows to gain access to the Internet, to a company network or to personal information. These include PCs, TV-based terminals, thin clients, or Personal Digital Assistants (PDAs). These devices can be classified as a combination of hardware and software features designed to work with .NET based services and applications. Currently, computers running Windows (Windows 9x, Millenium Edition, and the Windows 2000 with the .NET framework installed) and devices running Windows CE fall in this category.

- **.NET Components**

A component is a compiled library of classes that can be referenced and used by other programs. It may be of two types: i) Managed and ii) Unmanaged. For proper usage of a managed component, a particular component must be registered in the Windows system registry. However, the .NET components also known as assemblies do not have to be entered into the system registry. This is because of the fact that a .NET assembly files encapsulate their own information in special sections known as manifests. The Common Language Runtime (CLR) allows developers to specify the specific version of a component, thus preventing any version conflicts. Since CLR supports side-by-side execution, it allows for the execution of code from two similar components that only differ in version.

Any code that runs outside of the Common Language Runtime (CLR) is known as unmanaged code. This includes all COM (Component Object Model) components, Active X controls, and any calls to the native Win32 API. The COM components have no concept of metadata associated with them. There is a separate type library file that functions like assembly metadata, but it is part of a separate file and not related in any way to the component itself. To build the metadata for a COM component so that it can be used within the .NET framework a tool called the Type Library Importer (Tlbimp) is used. It uses the component's type library to create the metadata. Once the assembly is created using Tlbimp, the COM component is like any other .NET component. A new instance of it can be created. Its methods and properties can be called. The parameters are converted from COM data types to the CLR data types. When calling the compiler, a */r:* parameter is added to the compiler call that points to the DLL created by Tlbimp. When the application is deployed, you have to take one additional step beyond the *XCOPY* deployment that is supported by the .NET Framework. Just as a

COM component being used in .NET needs to have metadata created for it, a .NET component being used within COM needs to be properly registered in order to work.

- **Memory Management**

Memory management techniques are more versatile in a .NET framework. Allocation and de-allocation of objects have been made simpler. The CLR can allocate nearly 10 million objects per second on a moderately fast machine. The Garbage Collector (GC) is responsible for removing objects from the Managed heap that are no longer referenced. The GC may be automatically invoked by the CLR. The GC helps resolve many of the memory leak problems we are plagued with today. An object will be Garbage Collected automatically once it is loses scope.

- **Hardware Requirements**

For proper functioning of .NET framework, a PC with the following configuration is required.

- Pentium II 450 MHz or equivalent
- 128 MB RAM
- Video card capable of 800x600, 256 colors
- 1 GB hard disk space
- **Software Requirements**

To run Visual Studio .NET, the following basic system specifications need to be met.

- Operating system: Windows XP Professional, Windows 2000 (Datacenter Server, Advanced Server, Server or Professional) or Windows NT 4.0 Server

In addition, the installation package of Visual Studio .NET installs the following software and takes care of any other software upgradation.

- Internet Explorer 6.0
- Updated Data Access Tools

References:

1. Mackenzie D. and Sharkey K., Visual Basic .NET in 21 days, Sams publishing, 1st ed. 2002.
2. http://msdn.microsoft.com/dotnet

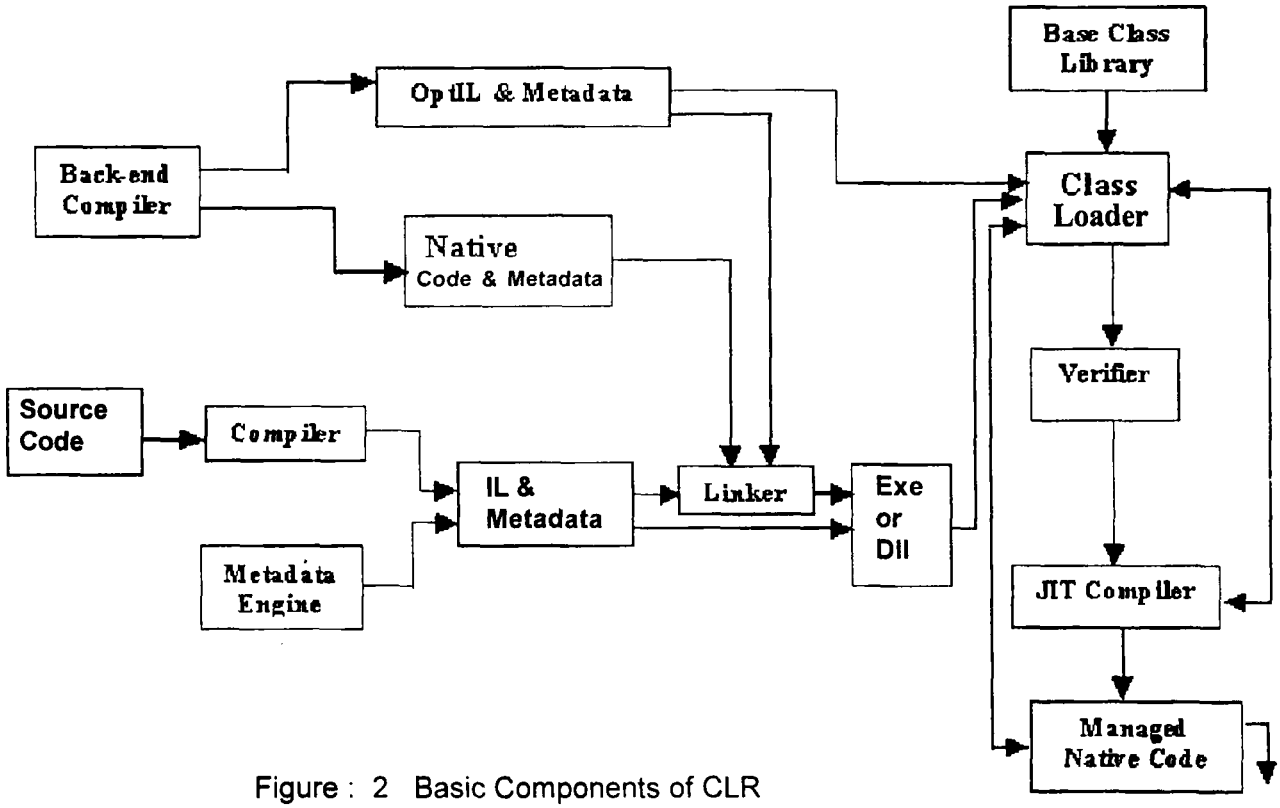

Figure : 2 Basic Components of CLR

Windows Tips & Tricks: You can insert tables in MS WORD even if your mouse is not functioning along with the Alt and the Tab keys. Just type the following sequence on the place where you want to insert the table: +——+——+——+——+ and then press enter. Isn't it simple? Send feedback to siddhartha@kucse.wb.nic.in