2

Understanding .NET: The C# Environment

2.1 THE .NET STRATEGY

We briefly discussed in the previous chapter how C# was created as a language for developing systems for the .NET platform. As pointed out, Microsoft wanted to make the World Wide Web (WWW) more vibrant by enabling individual devices, computers and web services to work together intelligently to provide richer solutions to users. By the intelligent integration of web sites on the Internet, users can create a wide variety of value-based applications such as unified banking services, electronic bill payment, stock trading, insurance services and comprehensive supply chain management. Microsoft calls this 'Web services' and the software strategy for implementing and delivering these services is '.NET'.

.NET is a software framework that includes everything required for developing software for web services. It integrates pre-sentation technologies, component tech-nologies and data technologies on a single platform so as to enable users to develop Internet applications as easily as they do on desktop systems. Microsoft took many of the best ideas in the industry, added their own creativity and innovations and produced a coherent systems solution popularly known as *Microsoft .NET*. The *Microsoft .NET* software solution strategy includes three key approaches as shown in Fig. 2.1.

Microsoft .NET platform includes the following components that would help develop a new generation of smart Internet services:

- .NET infrastructure and tools
- .NET user experience
- .NET building block
- .NET device software

Microsoft .NET products and services consist of the following:

- Windows .NET
- MSN .NET
- Office .NET
- Visual Studio .NET
- Personal subscription services
- bCentral for .NET

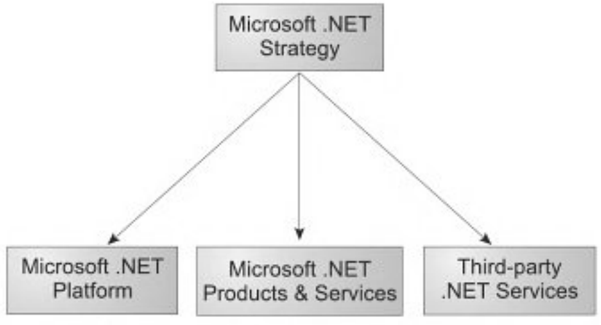

Fig. 2.1 *Microsoft .NET strategy*

Third-party .NET services will provide opportunities to a vast range of developers and users to produce corporate and vertical services using .NET platform.

2.2 THE ORIGINS OF .NET TECHNOLOGY

Before going into the details of .NET features further, we shall see how the concept of .NET was evolved over the past ten years. The current technology of .NET has gone through three significant phases of development:

- OLE technology
- COM technology
- .NET technology

These developments that took place during the 1990s are illustrated in Fig. 2.2.

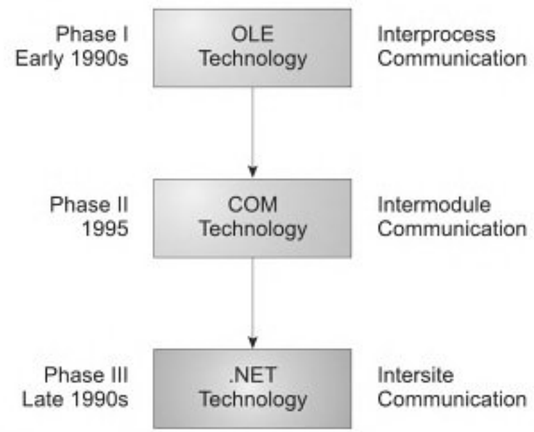

Fig. 2.2 *Three generations of component model*

2.2.1 OLE Technology

OLE (Object Linking and Embedding) technology was developed by Microsoft in the early 1990s to enable easy interprocess communications. OLE provided support to achieve the following:

- To embed documents from one application into another application
- To enable one application to manipulate objects located in another application

This enabled users to develop applications which required inter-operability between various products such as MS Word and MS Excel.

2.2.2 COM Technology

Till the advent of COM technology, the *monolithic* approach had been used for developing software. But when programs become too large and complex, the monolithic approach leads to a number of problems in terms of maintainability and testing of software. To overcome these problems, Microsoft introduced the component-based model for developing software programs. In the component-based approach, a program is broken into a number of independent components where each one offers a particular service. Each component can be developed and tested independently and then integrated it into the main system. This technology is known as the *Component Object Model* (COM) and the software built using COM is referred to as *componentware*.

COM technology offers a number of benefits to developers and users. It

- reduces the overall complexity of software
- enables distributed development across multiple organizations or departments and
- enhances software maintainability

2.2.3 .NET Technology

.NET technology is a third-generation component model. This provides a new level of inter-operability compared to COM technology. COM provides a standard binary mechanism for inter-module communication. This mechanism is replaced by an intermediate language called *Microsoft Intermedia Language* (MSIL) or simply IL in the .NET technology. Various .NET-language compilers

enforce inter-operability by compiling code into IL, which is automatically compatible with other IL modules. An inherent characteristic of IL code is *metadata*. Metadata is data about data and describes its characteristics, including data types and locations. IL allows for true cross-language integration. In addition to IL, .NET includes a host of other technologies and tools that will enable us develop and implement Web-based applications easily.

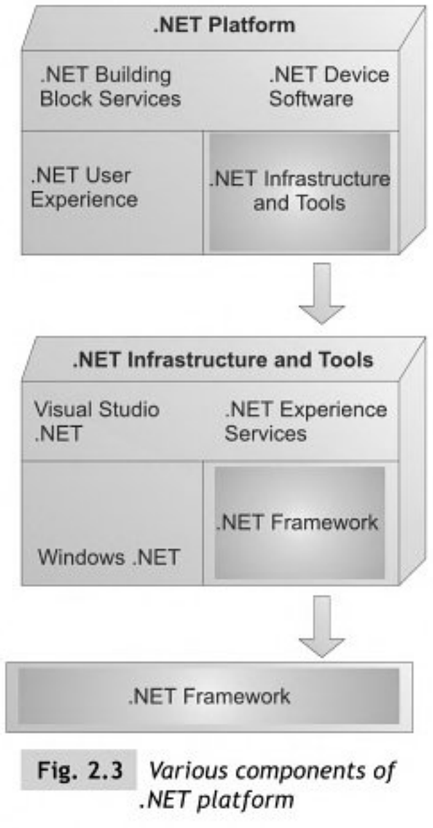

Fig. 2.3 *Various components of .NET platform*

2.3 THE .NET FRAMEWORK

The .NET Framework is one of the tools provided by the .NET infrastructure and tools component of the .NET platform as shown in Fig. 2.3. As pointed out earlier, the .NET platform provides a new environment for creating and running robust, scalable and distributed applications over the Web. A detailed description of either the .NET strategy or the .NET platform is beyond the scope of this book. Since C# derives much of its power from the .NET Framework on which it runs, we shall briefly discuss the key features of the .NET Framework.

The .NET Framework provides an environment for building, deploying and running web services and other applications. It consists of three distinct technologies as shown in Fig. 2.4.

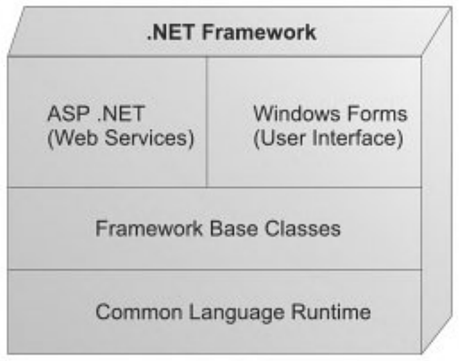

Fig. 2.4 *Architecture of .NET framework*

- Common Language Runtime (CLR)
- Framework Base Classes
- User and program interfaces (ASP .NET and Winforms)

The CLR is the core of the .NET Framework and is responsible for loading and running C# programs. Base classes provide basic data types, collection classes and other general classes for use by C# and other .NET languages. The top layer contains a set of classes for developing web services and to deal with the user interface.

2.4 THE COMMON LANGUAGE RUNTIME

The Common Language Runtime, popularly known as CLR is the heart and soul of the .NET Framework. As the name suggests, CLR is a runtime environment in which programs written in C# and other .NET languages are executed. It also supports *cross-language interoperability*. Figure 2.5 gives a diagrammatic summary of the major components of the CLR.

The CLR provides a number of services that include:

- Loading and execution of programs
- Memory isolation for applications
- Verification of type safety

- Compilation of IL into native executable code
- Providing metadata
- Memory management (automatic garbage collection)
- Enforcement of security
- Interoperability with other systems
- Managing exceptions and errors
- Support for tasks such as debugging and profiling

Figure 2.6 shows a flow chart of CLR activities that go on when an application is executed. The source code is compiled to IL while the metadata engine creates metadata information. IL and metadata are linked with other native code if required and the resultant IL code is saved. During execution, the IL code and any requirement from the base class library are brought together by the class loader. The combined code is tested for type-safety and then compiled by the JIT compiler to produce native machine code, which is sent to the runtime manager for execution.

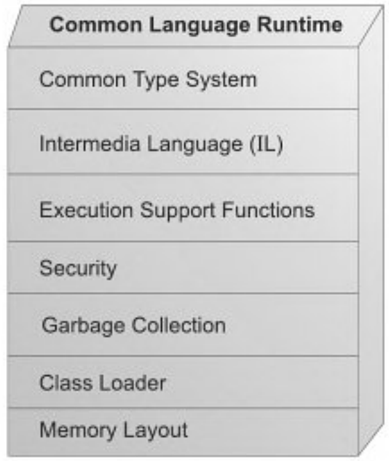

Fig. 2.5 *Component of CLR*

2.4.1 Common Type System (CTS)

The .NET Framework provides multiple language support using the feature known as *Common Type System* that is built into the CLR. The CTS supports a variety of types and operations found in most programming languages and therefore calling one language from another does not require type conversions. Although C# is specially designed for the .NET platform, we can build .NET programs in a number of other languages including C++ and Visual Basic.

2.4.2 Common Language Specification (CLS)

The Common Language Specification defines a set of rules that enables interoperability on the .NET platform. These rules serve as a guide to third-party compiler designers and library builders. The CLS is a subset of CTS and therefore the languages supporting the CLS can use each others' class libraries as if they are their own. Application Program Interfaces (APIs) that are designed following the rules of CLS can easily be used by all the .NET languages.

2.4.3 Microsoft Intermediate Language (MSIL)

MSIL, or simply IL, is an instruction set into which all the .NET programs are compiled. It is akin to assembly language and contains instructions for loading, storing, initializing and calling methods. When we compile a C# program or any program written in a CLS-compliant language, the source code is compiled into MSIL.

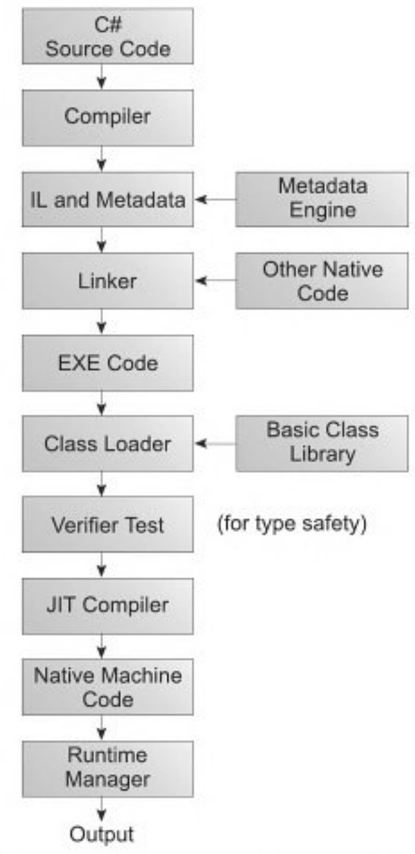

Fig. 2.6 *Flowchart of CLR activities for executing a program*

2.4.4 Managed Code

As we know, the CLR is responsible for managing the execution of code compiled for the .NET platform. The code that satisfies the CLR at runtime in order to execute is referred to as *managed code*. Compilers that are compatible to the .NET platform generate managed code. For example, the C# compiler generates managed code. The managed code generated by C# (and other compilers capable of generating managed code) is IL code. The IL code is then converted to *native machine code* by the JIT compilers.

2.5 FRAMEWORK BASE CLASSES

.NET supplies a library of base classes that we can use to implement applications quickly. We can use them by simply instantiating them and invoking their methods or by inheriting them through derived classes, thus extending their functionality.

Much of the functionality in the base framework classes resides in the vast namespace called **System**. We can use the base classes in the system namespace for many different tasks including:

- Input/Output Operations
- String handling
- Managing arrays, lists, maps, etc
- Accessing files and file systems
- Accessing the registry
- Security
- Windowing
- Windows messages
- Database management
- Evaluation of mathematical functions
- Drawing
- Managing errors and exceptions
- Connecting to the Internet
- And many more

2.6 USER AND PROGRAM INTERFACES

The .NET Framework provides the following tools for managing user- and application interfaces:

- Windows forms
- Web forms
- Console Applications
- Web services

These tools enable users to develop user-friendly desktop-based as well as web-based applications using a wide variety of languages on the .NET platform.

2.7 VISUAL STUDIO .NET

Visual Studio .NET (VS .NET) supports an Integrated Development Environment (IDE) with a rich set of features and productivity tools. These features and tools allow developers to build web applications faster and easier. Using web services and XML regardless of the language chosen for development, there is now one environment to learn, configure and use. We need not have to switch back and forth between environments to build, debug and deploy our code. VS .NET provides tools that extends support to the development lifecycle. As shown in Fig. 2.7, it acts as a foundation for the lifecycle platform. A detailed discussion on these tools is beyond the scope of this book.

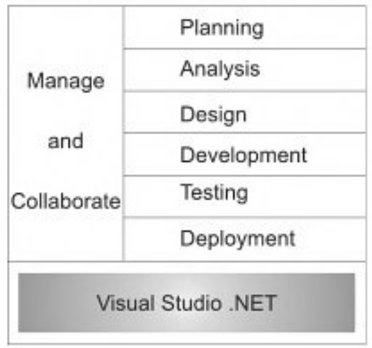

Fig. 2.7 *Developing lifecycle*

2.8 .NET LANGUAGES

The .NET Framework is language neutral. Currently, we can use a number of languages for developing .NET applications. They include:

2.8.1 Native to .NET

- C# (Specially created for .NET)
- C++
- Visual Basic
- JScript

2.8.2 Third-party Languages

- COBOL
- Eiffel
- Perl
- Python
- SmallTalk
- Mercury
- Scheme

All .NET languages are not created equal. Some can use the components of other languages, some can use the classes produced in other languages to create objects, and some languages can extend the classes of other languages using the inheritance features of .NET.

2.9 BENEFITS OF THE .NET APPROACH

Microsoft has advanced the .NET strategy in order to provide a number of benefits to developers and users. Some of the major benefits envisaged are:

- Simple and faster systems development
- Rich object model
- Enhanced built-in functionality
- Many different ways to communicate with the outside world
- Integration of different languages into one platform
- Easy deployment and execution
- Wide range of scalability
- Interoperability with existing applications
- Simple and easy-to-build sophisticated development tools
- Fewer bugs
- Potentially better performance

2.10 C# AND THE .NET

C# is a new programming language introduced with .NET. It is a concise, elegant .NET language. In many respects it is a version of the .NET object model. With C#, developers can quickly implement applications and components using the built-in capabilities of the .NET Framework. Since the C# code is managed by the CLR (of the .NET Framework) it becomes leaner and safer than C++.

The CLR extends a number of benefits to C# when it is implemented on the .NET platform. These include:

- Interoperability with other languages
- Enhanced security
- Versioning support
- Debugging support
- Automatic garbage collection
- XML support for web-based applications

Case Study

Problem Statement ABC Finance Solutions is a finance company involved in providing financial solutions to different individuals. The main task of ABC Finance Solutions is to provide suggestions to individuals on how to invest money. The IT Research and Development department in ABC Finance Solutions is currently involved in developing a software for financial planning to automate the task of creating financial plans, which can be provided to individuals so that they can invest money according to that plan. The development of software for financial planning is a difficult task as it involves doing a lot of complex calculations. How can the programmers involved in developing the financial planning software make their task easy?

Solution Since the development of financial planning software involves performing a lot of complex calculations, the team of programmers at ABC Finance Solutions must use C# to create the financial planning software. In the financial planning software, the functionality for Monte Carlo modelling has to be provided, which is a very complex task. A large number of objects need to be created for different classes in the financial planning software. The objects in the financial planning software need to interact with each other through sending and receiving of messages. If the programmers at ABC Finance Solutions use C++ programming language and create Component Object Model (COM) and ALT objects in the financial planning software then the financial planning software will become complex and time consuming. With the use of COM and ALT objects in a C++ application, it will take the programmers of ABC Finance Solutions a large amount of time to develop the financial planning software. Debugging and writing code for financial planning software will become easy with the use of C# as most of the functionality required for financial planning software is already available in C#. As the development time for creating the financial planning software reduces with the use of C# programming language, the cost of developing the financial planning software also decreases. This results in financial benefits to the ABC Finance Solutions company.

Review Questions

2.1 What is .NET?
2.2 What is .NET Framework?
2.3 What is .NET technology? Describe briefly its origins.
2.4 List the tools provided by .NET Framework for managing the user- and application interfaces.
2.5 What is Web service? How is it achieved using the .NET strategy?
2.6 What is the Microsoft Intermediate Language?
2.7 What is the Common Language Runtime (CLR)?
2.8 Enumerate major services provided by the CLR.
2.9 What are the additional benefits provided by CLR to programs developed using C#?
2.10 How does the CLR implements a C# program?
2.11 What is the Common Type System?
2.12 What is the Common Language Specification?
2.13 What is managed code?
2.14 List some of the important services the Framework Base Classes can offer to the users.
2.15 Describe the application of Visual Studio .NET?
2.16 List the languages supported by the .NET Framework.
2.17 What are the benefits of .NET strategy advanced by Microsoft?