**TEAM ASSIGNMENT 3**

**Huy Son.**

**Version 0.2**

Contents

[**I.** **Purpose of document.** 2](#_Toc324713168)

[**II.** **Part 1. Introduction.** 2](#_Toc324713169)

[**2.1.** **Introduction of .NET framework.** 2](#_Toc324713170)

[**2.2.** **Structure of .NET framework.** 6](#_Toc324713171)

[**2.3.** **Comparison between .NET framework architect and Java architect**. 8](#_Toc324713172)

[**III.** **Part 2. Description of standard.** 9](#_Toc324713173)

[**3.1.** **.NET framework architect description.** 9](#_Toc324713174)

[**3.2.** **Style and pattern.** 12](#_Toc324713175)

[**1.** **multi-tier architecture (often referred to as n-tier architecture) .** 12](#_Toc324713176)

[**2.** **Service-oriented architecture.** 13](#_Toc324713177)

[**3.** **Component-based software engineering.** 13](#_Toc324713178)

[**3.3.** **Semantics and variability mechanisms.** 15](#_Toc324713179)

[**IV.** **Part 3. Analysis of Properties, Tradeoffs, and Applicability** 15](#_Toc324713180)

[**3.1.** **Architecture the driver.** 15](#_Toc324713181)

[**V.** **Reference.** 16](#_Toc324713182)

1. **Purpose of document.**

* Document will provide an overview and detail about .NET framework, how to it work and applicability.

1. **Part 1. Introduction.**
   1. **Introduction of .NET framework.**

The **.NET Framework** (pronounced *dot net*) is a [software framework](http://en.wikipedia.org/wiki/Software_framework) developed by [Microsoft](http://en.wikipedia.org/wiki/Microsoft) that runs primarily on [Microsoft Windows](http://en.wikipedia.org/wiki/Microsoft_Windows).

*Software framework is a reusable set of libraries or classes for a software system (or subsystem).*

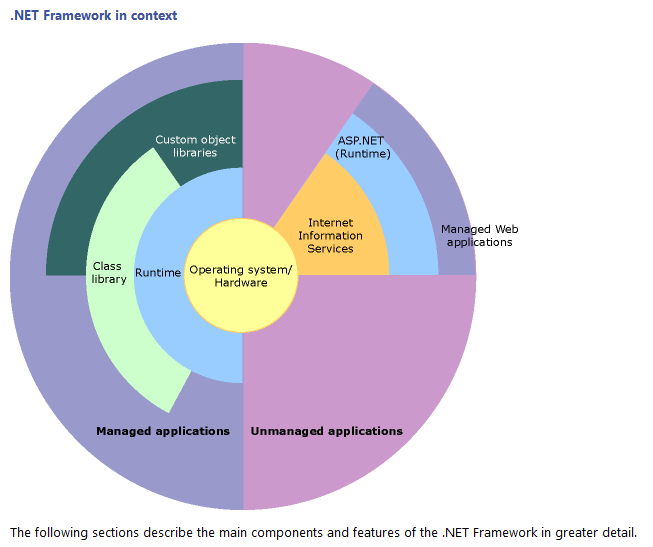
It includes a large [library](http://en.wikipedia.org/wiki/Base_Class_Library) and provides language interoperability (each language can use code written in other languages) across several [programming languages](http://en.wikipedia.org/wiki/Programming_language).

Programs written for the .NET Framework execute in a [software](http://en.wikipedia.org/wiki/Software) environment (as contrasted to [hardware](http://en.wikipedia.org/wiki/Hardware) environment), known as the [Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime) (CLR), an [application virtual machine](http://en.wikipedia.org/wiki/Process_virtual_machine) that provides important services such as security, [memory management](http://en.wikipedia.org/wiki/Memory_management), and [exception handling](http://en.wikipedia.org/wiki/Exception_handling). The class library and the CLR together constitute the .NET Framework.

The .NET Framework's [Base Class Library](http://en.wikipedia.org/wiki/Base_Class_Library) provides [user interface](http://en.wikipedia.org/wiki/User_interface), [data access](http://en.wikipedia.org/wiki/Data_access), [database connectivity](http://en.wikipedia.org/wiki/Database_connection), [cryptography](http://en.wikipedia.org/wiki/Cryptography), [web application](http://en.wikipedia.org/wiki/Web_application) development, numeric [algorithms](http://en.wikipedia.org/wiki/Algorithm), and [network communications](http://en.wikipedia.org/wiki/Computer_networking). Programmers produce software by combining their own [source code](http://en.wikipedia.org/wiki/Source_code) with the .NET Framework and other libraries. The .NET Framework is intended to be used by most new applications created for the Windows platform. Microsoft also produces a popular [integrated development environment](http://en.wikipedia.org/wiki/Integrated_development_environment) largely for .NET software called [**Visual Studio**](http://en.wikipedia.org/wiki/Microsoft_Visual_Studio).

As you would expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use the .NET Framework to develop the following types of applications and services:

Console applications, Windows GUI application, Windows Presentation Foundation (WPF) applications, ASP.NET applications, web services, Windows service and Workflow-enabled applications using Windows Workflow Foundation (WF).



.NET framework support two type of user.

**For developer.**

The .NET Framework provides the following services for application developers:

**Memory management**. In many programming languages, programmers are responsible for allocating and releasing memory and for handling object lifetimes. In .NET Framework applications, the CLR provides these services on behalf of the application.

**A common type system.** In traditional programming languages, basic types are defined by the compiler, which complicates cross-language interoperability. In the .NET Framework, basic types are defined by the .NET Framework type system and are common to all languages that target the .NET Framework.

**An extensive class library.** Instead of having to write vast amounts of code to handle common low-level programming operations, programmers can use a readily accessible library of types and their members from the .NET Framework Class Library.

**Development frameworks and technologies.** The .NET Framework includes libraries for specific areas of application development, such as ASP.NET for web applications, ADO.NET for data access, and Windows Communication Foundation for service-oriented applications.

**Language interoperability**. Language compilers that target the .NET Framework emit an intermediate code named Common Intermediate Language (CIL), which, in turn, is compiled at run time by the common language runtime. With this feature, routines written in one language are accessible to other languages, and programmers can focus on creating applications in their preferred language or languages.

**Version compatibility.** With rare exceptions, applications that are developed by using a particular version of the .NET Framework can run without modification on a later version.

**Side-by-side execution.** The .NET Framework helps resolve version conflicts by allowing multiple versions of the common language runtime to exist on the same computer. This means that multiple versions of applications can also coexist, and that an application can run on the version of the .NET Framework with which it was built.

**Multi-targeting**. By targeting the .NET Framework Portable Class Library, developers can create assemblies that work on multiple .NET Framework platforms, such as the .NET Framework, Silverlight, Windows Phone 7, or Xbox 360.

You can choose any programming language that supports the .NET Framework to create your application. Because the .NET Framework provides language independence and interoperability, you can interact with other .NET Framework applications and components regardless of the language with which they were developed.

**And for the others user.**

History.

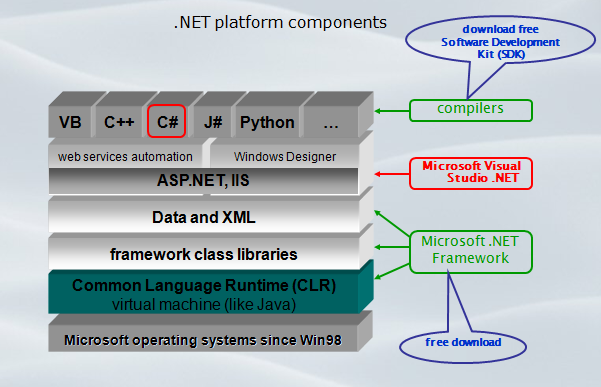
Microsoft started the development on the .NET Framework in the late 1990s originally under the name of Next Generation Windows Services (NGWS). By late 2000 the first beta versions of .NET 1.0 were released.

Version 3.0 of the .NET Framework is included with [Windows Server 2008](http://en.wikipedia.org/wiki/Windows_Server_2008) and [Windows Vista](http://en.wikipedia.org/wiki/Windows_Vista). Version 3.5 is included with [Windows 7](http://en.wikipedia.org/wiki/Windows_7), and can also be installed on [Windows XP](http://en.wikipedia.org/wiki/Windows_XP) and the [Windows Server 2003](http://en.wikipedia.org/wiki/Windows_Server_2003) family of operating systems. On 12 April 2010, .NET Framework 4 was released alongside [Visual Studio 2010](http://en.wikipedia.org/wiki/Visual_Studio_2010).

The .NET Framework family also includes two versions for [mobile](http://en.wikipedia.org/wiki/Mobile_computing) or [embedded](http://en.wikipedia.org/wiki/Embedded_system) device use. A reduced version of the framework, the [.NET Compact Framework](http://en.wikipedia.org/wiki/.NET_Compact_Framework), is available on [Windows CE](http://en.wikipedia.org/wiki/Microsoft_Windows_CE) platforms, including [Windows Mobile](http://en.wikipedia.org/wiki/Windows_Mobile) devices such as [smart phones](http://en.wikipedia.org/wiki/Smartphones). Additionally, the [.NET Micro Framework](http://en.wikipedia.org/wiki/.NET_Micro_Framework) is targeted at severely resource-constrained devices.

* 1. **Structure of .NET framework.**

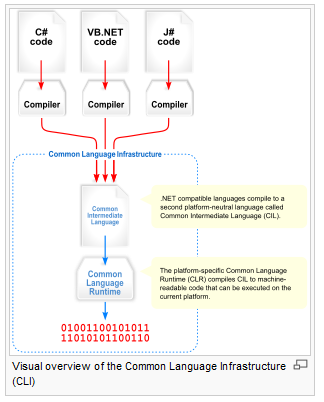
the next section will introduce .NET framework architecture.



Above picture show the .NET framework components in overview.

Common Language Infrastructure (CLI).

CLI is an open [specification](http://en.wikipedia.org/wiki/Specification) developed by [Microsoft](http://en.wikipedia.org/wiki/Microsoft) and standardized by ISO and ECMA that describes the executable code and runtime environment that form the core of the Microsoft [.NET Framework](http://en.wikipedia.org/wiki/.NET_Framework) and the [free and open source](http://en.wikipedia.org/wiki/Free_and_open_source_software) implementations [Mono](http://en.wikipedia.org/wiki/Mono_(software)) and [Portable.NET](http://en.wikipedia.org/wiki/Portable.NET). The specification defines an environment that allows multiple high-level languages to be used on different computer platforms without being rewritten for specific architectures.

**

*Above picture present utility of CLI.*

**The purpose of the Common Language Infrastructure (CL)** is to provide a language-neutral platform for application development and execution, including functions for [Exception handling](http://en.wikipedia.org/wiki/Exception_handling), [Garbage Collection](http://en.wikipedia.org/wiki/Garbage_collection_(computer_science)), security, and interoperability. By implementing the core aspects of the .NET Framework within the scope of the CL, this functionality will not be tied to a single language but will be available across the many languages supported by the framework. Microsoft's implementation of the CLI is called the [Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime), or CL.

*.****NET assembly****is an assembly is a*[*compiled*](http://en.wikipedia.org/wiki/Compiler)*code library used for deployment, versioning, and security. There are two types: process assemblies (*[*EXE*](http://en.wikipedia.org/wiki/EXE)*) and library assemblies (*[*DLL*](http://en.wikipedia.org/wiki/Dynamic-link_library)*). A process assembly represents a process that will use*[*classes*](http://en.wikipedia.org/wiki/Class_(computer_science))*defined in library assemblies. .NET assemblies contain code in*[*CIL*](http://en.wikipedia.org/wiki/Common_Intermediate_Language)*, which is usually generated from a*[*CLI language*](http://en.wikipedia.org/wiki/List_of_CLI_languages)*, and then compiled into*[*machine language*](http://en.wikipedia.org/wiki/Machine_language)*at*[*run time*](http://en.wikipedia.org/wiki/Run_time_(program_lifecycle_phase))*by the*[*CLR*](http://en.wikipedia.org/wiki/Common_Language_Runtime)[*just-in-time compiler*](http://en.wikipedia.org/wiki/Just-in-time_compiler)*.*

**Memory management.** The .NET Framework CL frees the developer from the burden of managing memory (allocating and freeing up when done); it handles memory management itself by detecting when memory can be safely freed. Memory is allocated to instantiations of .NET types (objects) from the managed heap, a pool of memory managed by the CL. As long as there exists a reference to an object, which might be either a direct reference to an object or via a [graph](http://en.wikipedia.org/wiki/Graph_(data_structure)) of objects, the object is considered to be in use. When there is no reference to an object, and it cannot be reached or used, it becomes garbage, eligible for collection. NET Framework includes a [garbage collector](http://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) which runs periodically, on a separate [thread](http://en.wikipedia.org/wiki/Thread_(computing)) from the application's thread, that enumerates all the unusable objects and reclaims the memory allocated to them.

**The .NET**[**Garbage Collector**](http://en.wikipedia.org/wiki/Garbage_collection_(computer_science))**(GC)** is a non-deterministic, compacting, [mark-and-sweep](http://en.wikipedia.org/wiki/Garbage_collection_(computer_science)#Copying_vs._mark-and-sweep_vs._mark-and-don.27t-sweep) garbage collector. The GC runs only when a certain amount of memory has been used or there is enough pressure for memory on the system. Since it is not guaranteed when the conditions to reclaim memory are reached, the GC runs are non-deterministic.

The GC used by .NET Framework is actually [*generational*](http://en.wikipedia.org/wiki/Garbage_collection_(computer_science)#Generational_GC_.28ephemeral_GC.29). Objects are assigned a *generation*; newly created objects belong to *Generation 0*. The objects that survive a garbage collection are tagged as *Generation 1*, and the Generation 1 objects that survive another collection are *Generation 2* objects. The .NET Framework uses up to Generation 2 objects. Higher generation objects are garbage collected less frequently than lower generation objects. This helps increase the efficiency of garbage collection, as older objects tend to have a larger lifetime than newer objects. Thus, by removing older (and thus more likely to survive a collection) objects from the scope of a collection run, fewer objects need to be checked and compacted.

* 1. **Comparison between .NET framework architect and Java architect**.

*Java EE is Oracle's enterprise java computing platform. The platform provides an API and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multi-tiered, scalable, reliable, and secure network applications*.

*.NET is Microsoft's platform for XML Web services. XML Web services allow applications to communicate and share data over the Internet, regardless of operating system or programming language*

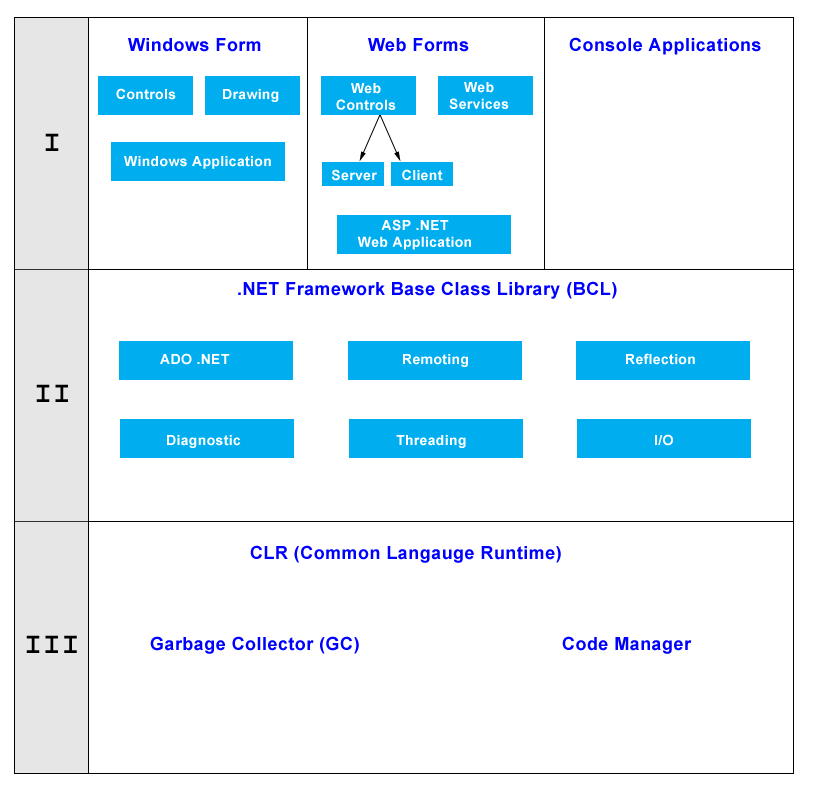
*Both Java EE and .NET framework have the same architecture design. Have the same component and functionality but different name.*

|  |  |  |
| --- | --- | --- |
| **Component** | The Common Language Runtime (CLR), an application virtual machine that provides important services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework.  Base Class Library provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. Programmers produce software by combining their own source code with the .NET Framework and other libraries. | A J2EE component is a self-contained functional software unit that is assembled into a J2EE application with its related classes and files and communicates with other components. The J2EE specification defines the following J2EE components:   * Application clients and applets are client components. Java Servlet and Java Server Pages technology components are web components. * Enterprise JavaBeans components (enterprise beans) are business components. * Resource adapter components provided by * EIS and tool vendors |

|  |  |  |
| --- | --- | --- |
| **Architecture** | Designs support multiple difference programming language. Currently, 30 languages support the .net framework. Runs primarily on Microsoft Windows.  The .NET CLR contains an Intermediate Language (IL) engine. Code and objects written in a language can be compiled into the IL runtime, once an IL compiler is developed for the language  CRL implement algorithm continuous distribution  Compiling works faster | Though other language’s code can be converted run under JVM they don’t acquire true. Cross language capability. Run in multiple platforms.  Compiling source code into the Java "byte code.", as long as platforms have a Java Virtual Machine to execute byte code.  JVM implement algorithm non -continuous distribution |

1. **Part 2. Description of standard.**
   1. **.NET framework architect description.**

.Net framework architect has 3 main layer.



1st layer is .NET application (.NET win form, web service, web application .. )

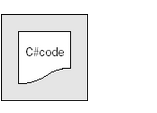
2nd layer is .NET framework base class library (BCL)

3rd layer is CLR (common langue runtime)

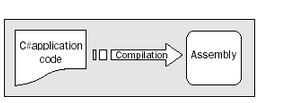
With above architect, .NET framework can compile .NET apps run on any supported platform: Win 64, win 32(seven, vista, XP, 2k, 98), win server, etc … and support multi language VB, C# (C-sharp), C++, J# (Java 1.2).

*In next session we will try to explain the processing in terms of C# code which is written using .NET Framework.*

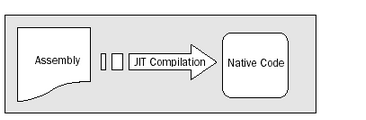
**Step 1**: Application code is written using a .NET - compatible language C#.



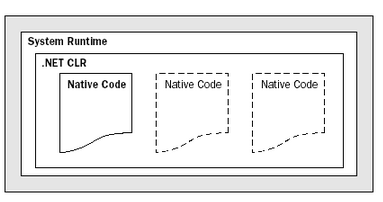
**Step 2**: Code is compiled into MSIL, which is stored in an assembly.



**Step 3 -** When this code is executed (either in its own right if it is an executable or when it is used from other code), it must first be compiled into native code using a JIT compiler.



**Step 4 -** The native code is executed in the context of the managed CLR, along with any other running applications or processes.



**Note:**One additional point concerning this process. The C# code that compiles into MSIL in step 2 needn't be contained in a single file. It's possible to split application code across multiple source code files, which are then compiled together into a single assembly. This extremely useful process is known as linking.

This is because it is far easier to work with several smaller files than one enormous one. You can separate out logically related code into an individual file so that it can be worked on independently and then practically forgotten about when completed.

This also makes it easy to locate specific pieces of code when you need them and enables teams of developers to divide up the programming burden into manageable chunks, whereby individuals can check out pieces of code to work on without risking damage to otherwise satisfactory sections or sections other people are working on.

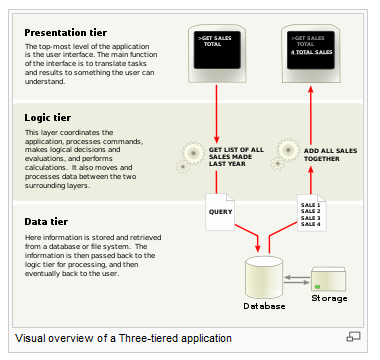
* 1. **Style and pattern.**

In this session we will show some styles and patterns are utilized by .NET framework.

1. **multi-tier architecture (often referred to as n-tier architecture) .**

*Segregates functionality into separate segments in much the same way as the layered style, but with each segment being a tier located on a physically separate computer.*

In software engineering, multi-tier architecture (often referred to as n-tier architecture) is a client–server architecture in which the presentation, the application processing, and the data management are logically separate processes. For example, an application that uses middleware to service data requests between a user and a database employs multi-tier architecture.



1. **Service-oriented architecture.**

*Refers to applications that expose and consume functionality as a service using contracts and messages.*

A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.

Service-oriented architectures are not a new thing. The first service-oriented architecture for many people in the past was with the use DCOM or Object Request Brokers (ORBs) based on the CORBA specification. For more on DCOM and CORBA.

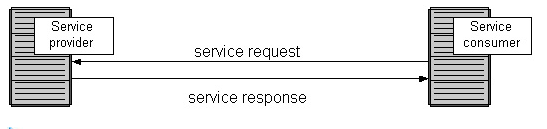
**Services**

If a service-oriented architecture is to be effective, we need a clear understanding of the term service. A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services.

**Connections**

The technology of Web services (new window) is the most likely connection technology of service-oriented architectures. Web services essentially use XML (new window) to create a robust connection.

The following figure illustrates a basic service-oriented architecture. It shows a service consumer at the right sending a service request message to a service provider at the left. The service provider returns a response message to the service consumer. The request and subsequent response connections are defined in some way that is understandable to both the service consumer and service provider. How those connections are defined is explained in Web Services explained (new window). A service provider can also be a service consumer.

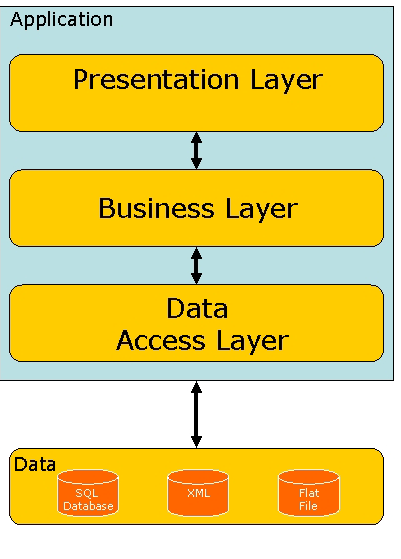


1. **Component-based software engineering.**

*Decomposes application design into reusable functional or logical components that expose well-defined communication interfaces.*

Component-based architecture describes a software engineering approach to system design and development. It focuses on the decomposition of the design into individual functional or logical components that expose well-defined communication interfaces containing methods, events, and properties. This provides a higher level of abstraction than object-oriented design principles, and does not focus on issues such as communication protocols and shared state.

.NET Framework provides support for building applications using such a component based approach.



Picture provide the structure of **Component-based software engineering.**

**Architecture (basic):** A computer running several software components is often called an application server. Using this combination of application servers and software components is usually called distributed computing. The usual real-world application of this is in e.g. financial applications or business software.

* 1. **Semantics and variability mechanisms.**

**Semantics.**

1. .NET is not available in an old Windows (95, 98 or before). User should install .NET framework before using, if not, user cannot use the programs which run on .NET framework.
2. Only use languages supported by .NET framework (C#, J#, VB …).
3. Microsoft has many operating systems and version of .NET framework. So, many programs were written on different platforms. User cannot run a program was on .NET framework version 4 on a lower system with .NET framework 3.5.
4. .NET framework only run on Windows system.

**Variability mechanisms.**

User can write many application and service with many language (C#, VB, J#) which were supported by .NET framework. Customers are free to write console applications, win form applications, Windows Presentation Foundation (WPF), ASP.NET applications, web and windows services … with strongly support from .NET class library.

*A language is complied with the Common Language Runtime standard to become a .NET language. In .NET, code is compiled to Microsoft Intermediate Language (MSIL for short). This is called as Managed Code. This Managed code is run in .NET environment. So after compilation to this IL the language is not a barrier. A code can call or use a function written in another language*.

1. **Part 3. Analysis of Properties, Tradeoffs, and Applicability**
   1. **Architecture the driver.**

**Constraints: (technical constraints)**

Run on Windows platform, can comply multi languages (C#, VB, J#).

**Quality attributes.**

Scalability. NET framework support multi-tiered architecture and component-based (user can develop system with many tiers and layer as they want) and .NET framework architecture has built-in mechanisms for expanding the number of servers available in a configuration and to load balance among server

**Availability.** NET Framework provide ready-to-use transaction services that enhance availability and reliability of the application by providing built-in failure recovery mechanisms (help system has no downtime).

**Usability.** .NET Framework provide common run time language, User and developer can choose many other language C#, Visual basic, Sliver Light,… to access different content in different forms

**Portability**. While Microsoft has never implemented the full framework on any system except Microsoft Windows, the framework is engineered to be platform agnostic, and cross-platform implementations are available for other operating systems. Microsoft submitted the specifications for the Common Language Infrastructure (which includes the core class libraries, Common Type System, and the Common Intermediate Language), the C# language, and the C++/CLI language to both ECMA and the ISO, making them available as official standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

**Security.** .NET has its own security mechanism with two general features: Code Access Security (CA), and validation and verification

1. **Reference.**

|  |  |
| --- | --- |
| **Source** | **content** |
| **.NET framework Wikipedia in English** (<http://en.wikipedia.org/wiki/.NET_Framework>) | Support information about overview of .NET framework |
| **Platform Comparison Java and .NET (Pat Palmer University of Pennsylvania)** | Compare .NET and J2EE |
| **Microsoft MSDN - .NET Framework conceptual**  (http://msdn.microsoft.com/library/zw4w  595w.aspx/) | Support information about overview of .NET framework |
| **Article .NET framework and architect**  (http://www.c-sharpcorner.com/uploadfile/puranindia/n  et-framework-and-architecture/) | Provide detail information about .NET framework architecture |
| **Architecture of .NET framework**  (PDF document – unknown source) | Provide detail information about .NET framework architecture |
| **Service oriented architecture**  (http://en.wikipedia.org/wiki/Service-oriented\_architecture) | Provide detail information about **Service oriented architecture.** |
| **Component-based software engineering**  (http://en.wikipedia.org/wiki/Component-based\_software\_engineering) |  |
| **Multitier architecture**  (http://en.wikipedia.org/wiki/Multitier\_architecture) | Provide detail information about **Multitier architecture.** |
| **service-oriented architecture**  (http://www.service-architecture.com/web-services/articles/service-oriented\_architecture\_soa\_definition.html) | Provide detail information about **service-oriented architecture.** |
| Architectural Patterns and Styles  (http://msdn.microsoft.com/en-us/library/ee658117.aspx) | Help to analyze pattern and styles in .NET framework. |
| SOA vs. Component-Based Architecture  (http://geekswithblogs.net/chrisfalter/archive/2010/09/05/soa-vs.-component-based-architecture.aspx) | Help to analyze semantics and quality attribute of .NET framework. |
| .NET Interview Questions and Answers  **(http://dev.fyicenter.com/)** | Provide fully information about .NET framework |
| J2EE over view SE452 (Enterprise Application Development -  Matthew Wright)  **(http://condor.depaul.edu/mwright1/se452/lectures/class-01.html#slide020)** | Provide overview information about .J2EE |

**End of document.**