Why is there a need for Mono?

October 3, 2005

Abstract

Mono is an implementation of .NET that differs from Microsoft's .NET in two important aspects; it runs on different platforms and it's released under an open source license.

This article describes the techniques .NET is based on and discusses the importance of Mono's existence.

1 Introduction

The dependency on computer programs increases every day. We use computer programs in more and more different parts of our lives which increases the need for different systems to communicate seamlessly. The computer programs also must fulfill needs that change from day one to day two, which forces the programs to be flexible and easy to extend, change or replace. To allow rapid development of systems of this type, techniques and standards must be development that allows code reuse instead of writing everything from scratch. In the beginning of 2000 Microsoft developed standards that can be used to ease the implementation and shorten development time called .NET (pronounced “dot-Net”).

.NET is associated with Microsoft and Windows only which locks potential users and customers to the Microsoft platform which might create a non competitive market with very few actors where the dependence of one company might become a problem. To change this and open for competition an open source alternative to Microsoft's .NET called Mono exists.

Why is there a need for Mono when Microsoft offers a framework that can be used for developing software?

2 Background

The main idea behind .NET is to create mechanisms that can be used to connect different systems, information, resources and devices in flexible, open and standardized ways.

This standards both, open ways to communicate from and to different computer systems and to communicate between different programming languages. The openness makes it not only easier to build new systems it also makes reuse of existing applications and infrastructure easier since information can be synchronized between many different, already existing, systems without modifying the code.

2.1 Definition

.NET was initiated by Microsoft in the beginning of 2000. Together with Hewlett-Packard and Intel, Microsoft managed to get the standards for the Common Language Interface (CLI) and the programming language C# ECMA standardized ([1] and [2]) in 2001 and later ISO standardized in 2003. Microsoft holds a great amount of patents for techniques used in .NET but these patents are promised to be able to be used royalty free for other implementations since this is the required to get the standards defined by both ECMA and ISO.

Microsoft has created a rich set of programs and libraries to use with .NET. Unfortunately almost all implementations are made for the Windows platform (some limited programs can be used on FreeBSD [3]) and today the Microsoft .NET framework comes bundled with Windows which makes execution of .NET applications on Windows easy.

In the beginning of 2000 a new productivity boosting programming platform for Linux and Gnome was needed by Ximian[1] so they started to make an implementation of .NET called Mono. Since building a complete implementation of . NET was a to big question for Ximian Mono was opened as an open source project in 2001 [4]. Ximian was later bought by Novell that now sponsors the Mono project. Novell's main idea to earn money on Mono is by support and service [5].

One other implementation of .NET, DotGNU Portable.NET [6] that is sponsored by the Free Software Foundation (FSF) also exist. This

1 Ximian was a company focusing on easy to use Gnome products. The company was bought by Novell in 2003.

implementation will not be described in this article.

There are two big differences in Mono and Microsoft's .NET implementations. The first is that Mono runs on several different platforms [7] while Microsoft's .NET only runs on Windows and in some cases FreeBSD. The second is that the source code for Mono is released open source [8] and Microsoft's source code is closed.

Both Microsoft and Mono offers extra features that is not defined in the ECMA definition of . NET. For example does both Microsoft and Mono have implementations of libraries commonly used for websites (ASP.NET) and Microsoft SQL database tools (ADO.NET). Another extra feature from Mono is the extra libraries for a GUI toolkit (GTK+) that is possible to use on several different platforms called GTK#.

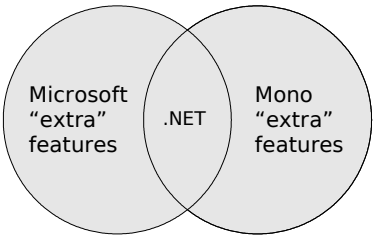

Illustration 1 Defining .NET

To get a common image on what .NET is the following definition will be used from now on:

- At least one compiler that creates data images that conforms the Common Intermediate Specifications (CIL). Most common is the C# compiler.
- A Runtime that conforms the Common Language Specification (CLS)
- A class library that encapsulates useful functionality (for example: various lists, tools to access and create XML, database connectivity, etc.).

2.2 Architecture

Microsoft has been interested in code reuse and software components have been important since the beginning of 1990 when they introduced the Component Object Model (COM). This model was the main cross-software communication and object creating tool until Windows 2000. As . NET is a successor and replacement to COM . NET also is designed in a modular way.

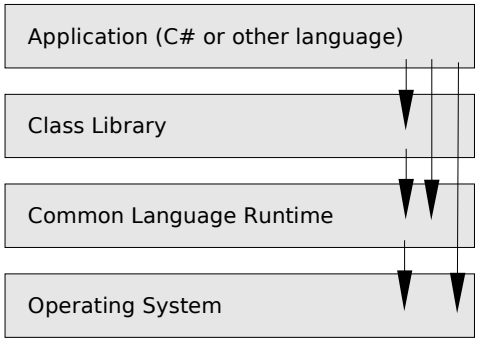

Illustration 2 The different layers in .NET

The architecture is layered where each layer abstracts the underlying implementations through defined interfaces. By using well defined interfaces between the layers it's possible to replace all code inside the layers as long as the facades stay intact without breaking the other parts of the system. The layers also leases the connections between applications and underlying implementations which means that the logic can be changed on future versions as long as the facade stays the same.

Abstracting the underlying operative systems from the programs means that the compiler must compile source code to a machine independent language. This Intermediate Language (IL) is converted at runtime when the Common Language Runtime (CLR) is invoked. This runtime converts the CIL code, on the fly, with a Just In Time compiler (JIT) to machine specific code that runs on the target machine. Compiler for several programming languages exists and hence is it possible to implement .NET applications in several different programming languages ([9]and [10]).

Since several class libraries exists a way of hierarchically structure them is necessary. In . NET the class libraries are logically and hierarchically structured into namespaces that groups several classes with similar functionality. For example is all drawing functionality located in the “*System.Drawing*” namespace. It is possible to create namespaces for example a company can structure their code in a structure that looks something like this: "*CompanyName.ApplicationName. BussinessLogic*".

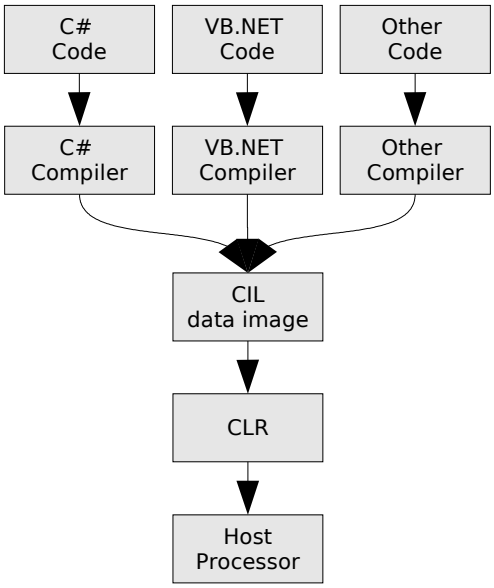

Illustration 3 How code is run on .NET applications

When packing class libraries physical it's called an assemblie. Those assemblies becomes dll-files that can be used in other programs just by loading them into the code. The dll-files should not be confused with the Win32 shared libraries that exists in Windows.

When writing applications in .NET the source code is preferred to be managed, which means that the CLR can guarantee the security of the code since all code is converted to machine code on runtime by the computer that runs the program. This makes it possible to run garbage collecting, exception functionality, type security, etc.

Where the class library fails to give the appropriate access to the needed resources it is possible to write source code that is unmanaged. Here the developer must take care of for example memory management. The biggest advantage of unmanaged code is that it is possible to access memory, as in C, directly via pointers and not via class libraries. This can give performance boosts on CPU intensive calculations that is necessary for the application. Unfortunately does unmanaged code mean that the portability might get lost since one or more step in the layered hierarchy of .NET (see Illustration 2) is lost.

2.3 Web services

In .NET web services is the standard way to do machine to machine interaction via network. The reason for choosing to use web services is that the standards that defines how communication should occur is open and easy to use. Communication is often done by sending the data in formated as XML via HTTP. Since HTTP is used as the standard protocol for sending web pages via Internet much implementations and configurations is possible to reuse which further makes web services simple to implement for an organization. Often the XML data is packed as SOAP packages.

The loose coupling from the programming language used and the web service makes web services a modular way of implementing communication between different systems. It is possible to implement web services in many different programming languages and even change the programming language since the actual data that is send is in standard text format where properties are separated and enclosed by XML tags. Alternatives to using web services also exists but an advantage of using standard web services for communications is the open nature of web services that doesn't depend on certain implementations to send and unpack binary data that can be sent using other techniques.

There are though disadvantages of using the text based XML format that's used in web services. The greatest disadvantage is probably that sending data in clear text is more processor intensive than using some type of binary data.

2.4 ASP.NET

ASP.NET is the part of .NET that makes it possible to generate websites. Even though the name derive from Microsoft's old Active Server Page (ASP) scripting language that also is used to generate websites it should not be confused with it since it is two totally different techniques.

ASP is a script language while it is possible to write code in all languages supported by .NET in ASP.NET and compile the website (just as any other .NET application). ASP.NET is created to work as much as possible as any window application. For example where System.Windows.Forms (WF) is available to create GUIs are WebForms (WeF) available to generate HTML pages where controls that builds the GUI works as much as possible the same way. On a WF application a button know how to draw itself to the screen while a button in WeF knows what HTML code that should be generated to function on the website. Since code is compiled in ASP.NET it also increases

performance and many errors can be caught at development time which gives higher quality on the software.

Another great advantage with ASP.NET compared to scripting languages is that the complete class library that's available in .NET can be used directly on a website that's written in ASP.NET which might shorten development time drastically. Since all existing libraries can be used it's also very likely that code thats being used on an window application can be reused on the website.

2.5 ADO.NET

ADO.NET is the primary relational data access model for .NET developed applications. It is basically an other library that can be used within the .NET application that abstracts the actual databases in favor for accessing data via objects and method calls. Data can be received and edited as DataSet objects, in XML or in some cases directly via method calls (See [11] for more information). Several different databases are supported in both Microsoft's .NET implementation and Mono.

Another important task for ADO.NET is to create functionality that allows creating and interpreting XML files.

2.6 User Interfaces

Different operative systems uses different window controlling methods. For example a radio button don't look the same in Windows as in Gnome as in Mac OS X. In the perfect world GUIs would work and look natively on the host computer without any need for rewrite of the GUI code. Unfortunately this is not the case today, which forces rewrite on at least the GUI. Hopefully this will be solved in future releases on both Microsoft's .NET and Mono.

WF is the standard way in Microsoft's .NET to draw all GUIs. Unfortunately doesn't Microsoft provide an implementation that works on other platforms than Windows. The WF implementation is also tight coupled with the Win32 API ([12]).

Several different approaches have been under developing in Mono to implement the WF functionality on other platforms than Windows. The fist unsuccessful approach was to convert all methods to the native GUI toolkit for Gnome, GTK, but since transforming an API to another was to complicated to be successful this approach was put down. The next approach was to use Wine [13] but since Wine was in to much development and not stable enough this approach was also skipped.

The implementation that's used from Mono version 1.1.4, Managed Window Forms [14] (MWF), works on drawing all GUI controls with the methods from System.Drawing and by that implement the complete methods available in WF. The implementation is not completed yet but it is supposed to be finished in the Mono 1.2 release thats planned for Q1 2006 [15].

While it might be possible to use WF on other platforms that Windows the preferred GUI toolkit for Mono it GTK#. GTK# is the bindings that makes it possible to use GTK in Mono by using library calls in Mono. GTK is open source and ported to several different platforms which means that the same source code can be run on different platforms without editing the source code as long as GTK is installed on the host system.

Mono also supports other GUI toolkits that runs on several different platforms [16].

3 Microsoft .NET vs Mono

A discussion where the most important subjects have been selected to find differences with the . NET implementations follows.

3.1 Licensing and Patents

The source code for Microsoft's implementation is closed which makes redeveloping and customization of the actual framework more or less impossible for third party. This also means that if a platform is unsupported it's impossible to run the framework there. On the other hand is Microsoft the company that initiated .NET and hence holds several patents for the techniques that's used. Even if the patents was opened for use to get the techniques standardized, patents just outside (ASP.NET, ADO.NET and Windows.Forms) might hold advantages for Microsoft (both a business and implementation advantage). Having one company for all support questions might be another advantage as well as a disadvantage.

Mono on the other hand is released open source which means that all source code is open for reading, editing and redistribution. Different parts of Mono is licensed different, but all has the open source idea in common both for program that will

be released as open source and programs that will be released proprietary. One big problem for Mono is the patents that Microsoft holds for ASP.NET, ADO.NET and Windows.Forms alternatives exists in Mono but still it might lock up developers that have been developing for a Microsoft platform to start using Mono instead even though alternatives exist (GTK# can for example be used instead of Windows.Forms). The idea from Mono's side to work around these techniques is to work around the solutions without breaking the interfaces which should make it possible to move applications from Microsoft's .NET platform. If those parts becomes a problem for Mono in the future they will be removed and replaced by other techniques [17].

3.2 Distribution and Platforms

When writing applications for Microsoft .NET one .NET framework must be installed and when writing for Mono, another framework must be installed. This makes distribution of the respective framework a central question from a usability and deployment view.

Microsoft's .NET framework is included on all computers that runs Service Pack 2 (SP2) on Windows XP via Microsoft's update service, Windows Update. Since this often is done automatically, developers that develop for the Microsoft .NET platform can rely on the latest stable release of Microsoft .NET. The main disadvantage with Microsoft's .NET implementation is that is runs only on Windows (and in some limited cases on FreeBSD).

Mono on the other hand have no specific installation and distribution system which of course is a drawback for deploying an application that uses the Mono framework. The reason why Mono doesn't provide such a service is that Mono runs on several different platforms, all with their specific methods for updates and installation. Even though no automatic installation and updating exists Mono offers installable packages for several different operating systems and distributions on their website [18]. The lack of knowing if a client runs the latest release of the Mono framework might of course be a problem when developing for Mono. One note here is that many Linux distributions have Mono included in their package system which means that updates are done in similar ways that with Windows Update.

As a subset of Microsoft's framework does a smaller framework the Microsoft .NET compact framework exist [19]. It's aim at creating a similar platform that the ordinary .NET platform with some restrictions in functionality to make it runnable at smaller computer such as PDAs. The Mono version of the framework runs on PDAs that use the ARM processor [20].

3.3 Releases and Development Platforms

Code for both Microsoft's .NET and Mono can be written in any text editor and compiled for hand. If the need for a complete API is desired Microsoft offers Visual Studio [21] and Mono offers Monodevelop [22]. Both are usable IDEs for writing .NET code with build in debugger, handles project dependencies, projects and instant compiling and running of programs by clicking run in the GUI. Since Visual Studio is a Microsoft developing tools that builds on previous releases that was used for other languages this product is more mature which gives a great advantage. Monodevelop doesn't natively offer a GUI designer even if products like Glade User Interface Builder [23] exist whilst Visual Studio offers a complete GUI designer tool.

Mono 1.0 was released on June 30th, 2004. It is a complete implementation of the .NET components that it specified in the ECMA specifications (See [15]). It's main disadvantage compared to Microsoft's versions is that it lacks the Windows GUI components (WSF) that is widely used even though it's not in the core specifications.

Mono 1.2 is planned to be released in the beginning of 2006 (Quarter 1) where WSF are supposed to be fully implemented. Previews for ASP.NET 2.0, ADO.NET 2.0 will also be included.

Mono version 2.0 is targeted for last quarter 2006. It should implement the new features in the .NET 2.0 specifications.

Microsoft released version 1.0 of their framework in 2002[1] which followed by the current stable 1.1 release in 2003. The current development release is the 2.0 beta 2 that will be followed by a final release some time on the end

1 See .Net on Wikipedia http://en.wikipedia.org/wiki/Microsoft_.NET#.NET_Framework_1.0 (no native information found at Microsofts website)

of this year or next year.

Both Microsoft's .NET and Mono implements the ECMA specifications as a common divider. Both of them have also extra features (such as ASP.NET, WSF and GTK#) in form of class libraries that can be used with the respective release.

3.4 Compatibility

The idea behind .NET is that code shall be possible to reuse as much as possible through the openness of .NET. This idea could however not been considered successful if programs are locked up to the Microsoft platform only. This is why compatibility between the different .NET implementations is important.

Mono's goal is to be as compatible as possible with the Microsoft framework to allow programs to be run on both Microsoft's .NET and Mono and hereby increase the number of developers that can build applications for Mono.

The biggest problem for compatibility today is that the default GUI in Windows (WF) doesn't have an equivalence in Mono. Development on this area is in progress through Managed Window Forms even though it shall be said to be in to much development for production use for the moment. If it is possible it should therefore be recommended that all GUI code in applications are kept away from the logic to allow rewrite of some parts. If it is possible the alternative to use GTK#[1] for GUI could also be considered if it is possible.

4 Conclusion

Since Microsoft's .NET is only available for Windows it means that customers are locked up with Windows. At first thought this might not be a problem but when the software grows into the organization it might force the use of Microsoft products even where it's not desirable. If an organization decides that a change of platforms is necessary, it might be expensive when large parts of the infrastructure is dependent on the developing framework.

Another problem when no alternatives to Microsoft's .NET exist is that Microsoft will be the only player on the field. From a competitive view this is a bad thing since being dependent on **one** company and **one** platform might reduce freedom.

Mono is the framework that changes this by offering similar or the same functionality as Microsoft's .NET framework but with two major differences - Mono is released for several platforms and Mono is released as open source.

1 GUI Toolkit that can be used on several different platforms.

5 References

[1] *Standard ECMA-334C# Language Specification*: http://www.ecma-international.org/publications/standards/Ecma-334.htm (Last accessed on 2005-10-02)

[2] *Standard ECMA-335Common Language Infrastructure (CLI)*: http://www.ecma-international.org/publications/standards/Ecma-335.htm (Last accessed on 2005-10-02)

[3] *Shared Source Common Language Infrastructure 1.0 Release*: http://www.microsoft.com/downloads/details.aspx?familyid=3a1c93fa-7462-47d0-8e56-8dd34c6292f0&displaylang=en (Last accessed on 2005-09-30)

[4] *[Mono-list] Mono early history.*: http://lists.ximian.com/archives/public/mono-list/2003-October/016345.html (Last accessed on 2005-10-02)

[5] *The Novell Role in the Mono Project*: http://www.mono-project.com/FAQ:_General#The_Novell_Role_in_the_Mono_Project (Last accessed on 2005-10-02)

[6] *DotGNU Project*: http://www.gnu.org/projects/dotgnu/ (Last accessed on 2005-10-03)

[7] *Mono Supported Platforms*: http://www.mono-project.com/Supported_Platforms (Last accessed on 2005-02-10)

[8] *Mono FAQ: Licensing*: http://www.mono-project.com/FAQ:_Licensing (Last accessed on)

[9] *Mono Languages*: http://www.mono-project.com/Languages (Last accessed on 2005-10-02)

[10] *Microsoft Programming Languages*: http://msdn.microsoft.com/vstudio/productinfo/whitepapers/default.aspx (Last accessed on 2005-10-02)

[11] *Mono Provider Factory*: http://www.mono-project.com/Provider_Factory#Overview_of_the_ProviderFactory_object_model (Last accessed on 2005-10-02)

[12] *Microsoft Win32 to Microsoft .NET Framework API Map*: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/win32map.asp (Last accessed on 2005-10-02)

[13] *Wine is an Open Source implementation of the Windows API on top of X and Unix.*: http://www.winehq.com/ (Last accessed on 2005-10-03)

[14] *MWF weblog - Tracking progress of Mono's System.Windows.Forms development*: http://svn.myrealbox.com/blog/ (Last accessed on 2005-10-03)

[15] *Mono Project Roadmap*: http://www.mono-project.com/Mono_Project_Roadmap (Last accessed on 2005-10-02)

[16] *Mono GUI Toolkits*: http://www.mono-project.com/Gui_Toolkits (Last accessed on 2005-10-02)

[17] *Mono FAQ: Licensing*: http://www.mono-project.com/FAQ:_Licensing (Last accessed on 2005-10-02)

[18] *Mono Downloads*: http://www.mono-project.com/Downloads (Last accessed on 2005-10-03)

[19] *Fundamentals of Microsoft .NET Compact Framework Development for the Microsoft . NET Framework Developer*: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetcomp/html/net_vs_netcf.asp (Last accessed on 2005-10-03)

[20] *Mono 1.1.9 Release notes*: http://www.go-mono.com/archive/1.1.9/ (Last accessed on 2005-10-03)

[21] *Visual Studio Home*: http://msdn.microsoft.com/vstudio/default.aspx (Last accessed on 2005-10-03)

[22] *MonoDevelop*: http://www.monodevelop.org/ (Last accessed on 2005-10-03)

[23] *Glade - a User Interface Builder for GTK+ and GNOME*: http://glade.gnome.org/ (Last accessed on 2005-10-03)