MONO, a free software alternative to .NET

Alberto Moreno Jiménez, Elena Villalba Mora, M.T. Arredondo
Life Supporting Technologies
ETSI Telecomunicación
Universidad Politécnica de Madrid
amoreno@lst.tfo.upm.es

Abstract

In the year 2002 Microsoft released the version 1.0 of the .NET Framework] as a new tool for developing professional software applications. The aim of this environment was to fill the need for interoperable systems through platforms and programming languages, where other alternatives like Java and CORBA did not succeed. The Java solution lacks feasible interoperability with non-Java systems, and CORBA is complex to deploy. The .NET platform offers an interoperable solution based on standards like XML, HTTP and Web Services and it works though a wide variety of languages and platforms. These characteristics have motivated the creation of MONO, a free software implementation of the .NET Framework.

This paper presents the MONO framework, remarking its achievements and its faults during the development of an interoperable set of libraries for a Heart Failure Management System.

1. Introduction

e-Health applications manage important and personal information about people and diseases. This kind of information requires important security conditions and restrictions, since data confidentiality have to be guaranteed. Also, these systems are commonly oriented towards people with no technical background, who expect a correct behavior from the system, even in complex low level error situations. These people require a high level of reliability; otherwise they will not use the system whatsoever. Application programmers cannot deal with all this complexity and high level of expectation. In order to develop usable e-Health systems, it is necessary a mature framework to build on, which can assure the proper level of security and reliability. Moreover, these systems often need to be executed on different operating systems and different hardware, so portability is also an important issue to take into account.

In order to resolve this kind of problems, common to many software developers, Microsoft proposes the .NET Framework [1], a general purpose application development environment. The framework consists of a wide platform that includes several possible programming languages, tools for agile product development and different information servers. It also defines the new C# programming language, as reference for the .NET Framework. According to the last Open Source and standardization initiatives in the software community, both the Framework and the C# language are part of ECMA standards. Therefore, third parties can implement versions of these products maintaining compatibility. Also, free software groups have been sponsored by Microsoft to build general purpose .NET libraries.

.NET Framework relies on three main components to provide its features [4]: the Common Language Runtime (CLR), a base class library and a strong security structure. The CLR is the .NET virtual machine that runs the applications code. The different .NET programming languages, compile to platform independent bytecodes. These bytecodes define a low level virtual code known as MSIL (Microsoft Intermediate Language). The first time the CRL runs a .NET code in a system the JIT (Just In Time Compiler) performs the compilation to the specific platform machine code, thus providing the appropriate performance.

The following figure sketches the layers of .NET Framework.

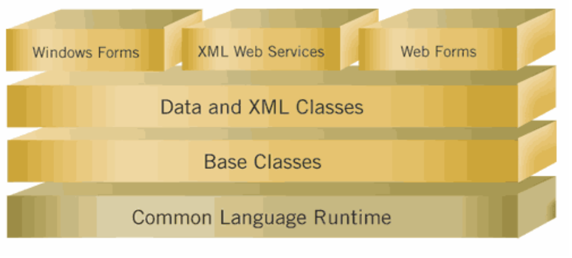


Figure 1: the .NET Framework

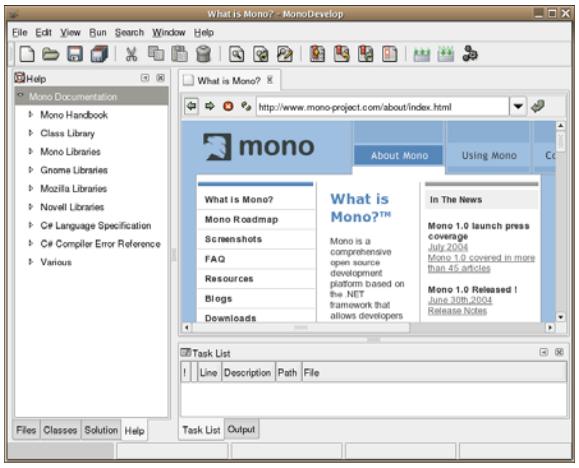
The base class library (BCL) provides tools to facilitate and strengthen software development. It includes utilities to build visual event-based Windows formularies. The BCL also incorporates connectivity mechanisms for databases and Web technologies. .NET focuses on new techniques, such as XML and Web Services [8], to intercommunicate distributed systems on a network. The assemblies that constitute the BCL are part of the ECMA standard and are available in all compliant .NET implementations. However, optional libraries provided from Microsoft are not part of the standard and remain subject to software patents. Examples of these optional libraries include the Windows Forms namespaces and abstract database access, among other important utilities.
Security is a critical requirement, especially in distributed network applications. .NET Framework provides code access security that grants permissions depending on the code origin. This way, .NET applications coming from the desktop, the local network or the Internet, will run on a system with different security restrictions. Also .NET provides code validation and verification, in order to guarantee the internal consistency of an assembly. Before running any code, the Common Language Routine (CLR) checks that the assembly only contains valid metadata and that code does not perform unsafe operations. This security approach protects users against potential attacks to their systems. It focuses on the most common security holes found in Internet applications.
MONO [3] is an Open Source implementation of the .NET Framework. MONO started in December of the year 2000 with the creation of a C# compiler by Miguel de Icaza. On July 10th, 2001, Novell announces at an O'Reilly conference the creation of the MONO project. Three years later, on June 30, 2004 MONO 1.0 was released.

2. Methodology

The MONO project is divided in three main research lines:

- The Core group implements the framework common products. This includes the MONO virtual machine, the C# compiler and the Base Class Libraries. All of these components are based on the Ecma-334 and Ecma-335 standards, allowing Mono to provide standards compliant, free and open source virtual machine.
- The MONO/Linux/GNOME development stack focuses on applications under GNOME, a common windows manager used in the Open Source Community. The main developments include Gtk# for GUI development, integration libraries for Unix and Mozilla and a XML schema language called RelaxNG. Gtk# allows MONO applications to integrate with GNOME desktops with the same facility as native applications [5]. This group also develops database connectivity drivers for most common servers, such as MySQL, SQLite, PostgreSQL, Firebird, Open Database Connectivity (ODBC), Microsoft SQL Server and Oracle, among many others.
- The Microsoft compatibility stack works on porting MS applications to MONO. The main objectives are the porting of Windows Forms, Web Forms and ADO .NET but, since these components are not covered by ECMA standards, they remain subject to patent issues [6].

MONO environment provides several tools to develop applications. The MONODevelop is a programming environment for Gnome and Windows that allows programmers to build visual MONO applications. Other general purpose tools and applications have been built for MONO. Some examples are Monoppix: a Live Linux Distribution that runs MONO applications or some Open Source Libraries such as nHibernate, used for abstract database access, and log4net for debugging purposes.



Mono is a comprehensive open source development platform based on the .NET framework that allows developers

Figure 2: application development under Monodevelop

MONO has been tested in MyHeart[7] project aiming at creating shared libraries between two products. The first product was a tool for people who need to look after their health. The concept consisted of providing a Linux based embedded device running a MONO application that controls certain life habits. The second product was design for old people with heart failure diseases. The product was provided running on a Windows PocketPC PDA with Compact Framework [8]. Since products shared certain similarities, as well as physical measurement devices, there was an interest to share libraries between them. To test this possibility the following steps were followed:

1. Study of the state of the art in Open Source Software .NET drivers for different technologies, such as Bluetooth connectivity, pocket database access, Web Service coupling and user formularies implementation.
2. A small application was created in Visual Studio. This application consisted of a user interface with some labels and buttons, which retrieved the information from a local file and sent it to a Web Service for processing. Also, an extended version of this application was created, with access to a pocket SQLite database.
3. These applications were compiled using Microsoft .NET framework for Desktop and Compact Framework environments.

3. Results

The results of the state of the art in MONO technologies are summarized in the table 1. This corresponds to the latest version of the product, MONO 1.1.10. According to Novell, the major feature missing before Mono 1.2 is the completion of a Windows Forms implementation.

Windows Forms	The Windows Forms assemblies provide the means to build visual applications in .NET. The major problem found in the MONO development group, was that the implementation was based on the Microsoft Windows application interface (API). The differences between the Windows API and other interfaces like GNOME or KDE is an obstacle for the implementation of this functionality. Windows Forms applications running under MONO lack some of the most complex components like grid tables among others. According to the MONO official schedule, the .NET version 1.1 of Windows Forms will be available by MONO 1.2.
.NET 2.0 Support	Support for .NET 2.0 ECMA standards is considered complete. Non-standard assemblies will still need more time to be fully developed by the MONO team.
Code Access Security	MONO provides code security, depending on the source origin. Nevertheless, still there is no code verifying, so changes to the MSIL metadata are not detected.
Visual Basic	Visual Basic support is in beta version.
ASP .NET	Support for ASP .NET 2.0 is available in MONO 1.1.10. A module for the Web Server Apache is provided to run ASP scripts.
C# Compiler	A complete C# compiler is provided since the first versions of MONO.
GTK#	GTK# is a set of .NET libraries to program visual forms user applications in MONO. These are provided as an alternative to the Windows Forms namespace. Since it is based on a working toolkit, the status is more advanced and mature than the Windows Forms compatibility.

Table 1: MONO Status

As far as now, MONO has a partial implementation of the .NET platform, which can compile and run .NET applications. It can run over several software platforms like Linux, Mac OS, BSD, Solaris and, of course, Microsoft Windows. There are particular MONO versions for most common hardware platforms, such as x86, IA64, ARM and PowerPC, and there are groups working for other platforms.

As an independent framework, MONO provides a complete environment for applications development. Available assemblies provide database access, Web Services, and GUI formularies through GTK# and partial Windows Forms. Since MONO runs on different platforms, it provides a real interoperable development framework. Although not all necessary requirements are implemented yet, the system is stable and can be used in non critical applications. The main

drawback in MONO environment is the fair Microsoft implementation compatibility. Although simple applications can run straight forward from the Windows Implementation to MONO, most complicated systems with database access and complex visual formularies are not supported yet.
The tests done in MyHeart project tried to measure the compatibility level between the Windows .NET Desktop Framework, and MONO. The state of the art showed that assemblies for database management and Bluetooth connectivity were different from one framework to the other. The Web Service application created in Visual Studio ran over MONO as expected, getting the file from the local file system, and sending it to a remote server. However, the extended version with light database support could not run directly.
The compatibility problems grow when the environment is mobile and the applications run on Microsoft Compact Framework. Microsoft creates Compact Framework as a .NET subset, and provides different reduced assemblies to build applications. However, these applications can still run on the .NET desktop framework, because these reduced assemblies export a retargetable flag, which allows it to run on the full desktop platform. As far as now, MONO does not provide any Compact Framework support. Moreover, since most Linux based embedded devices run a full Linux distribution, there is no need for a reduced version at the moment. The problem arises when MONO cannot support retargetable assemblies at all and, according to the last news from Novell, this is not bound to change in the near future. For this reason, Compact Framework applications will not run on MONO for now unless compiled versions of the assemblies are provided.

4. Conclusions

The number of systems that use the .NET Framework is growing day by day, and MONO aims at creating a software platform that can let the Free Software take advantage of it. This is the reason why there are many enterprises that look forward MONO, and support it financially.
As far as now, MONO provides a stable framework for software development based on the Microsoft .NET ECMA standards. Even though MONO supplies a consistent framework to build applications, certain compatibility issues still isolate one environment with the other.
The maturity level of MONO platform does not reach the Microsoft implementation. However, the interest around MONO is growing among the Free Software community and it is achieving a lot of support. Eventually, MONO will bring up the necessary functionalities to provide a real alternative to Microsoft .NET Framework.

5. Acknowledgment

The MONO Framework has been studied within the MyHeart project, “Fighting Cardiovascular Diseases by prevention and early diagnosis” (IST-2002-507816). MyHeart is a 6th Framework Project of the IST Programme, partly funded by the European Commission.

6. Bibliography

[1] Thuan Thai, Hoang Q. Lam, ".NET Framework Essentials". Publisher: O'Reilly. ISBN: 0596005059.
[2] George M. Doss. "Corba Networking with Java". Publisher: Wordware Publishing Inc.,U.S. ISBN: 1556226543.
[3] Dumbill, Ed. "MONO: A developer's notebook". Publisher: O'Reilly. ISBN: 0596007922
[4] Matt Reynolds, Karli Watson. ".NET Enterprise Development in C#: From Design to Deployment". Publisher: Wrox Press Ltd. ISBN: 1861005911.
[5] Pennington, H. "GTK+/GNOME Development". Publisher: O'Reilly. ISBN: 0735700788
[6] MyHeart Description of Work, February 2006. IST-2002-507816 MYHEART IP
[7] Peter Stanski, Craig Morris, Srinivasa Sivakumar, Andrew Polshaw, ".NET Compact Framework". Publisher: Wrox Press Ltd. ISBN: 1861007000
[8] Web Services Essentials. By Ethan Cerami. Publisher: O'Reilly. Pub Date: February 2002. ISBN: 0-596-00224-6.

Address for correspondence

Alberto Moreno Jiménez
Life Supporting Technologies
ETSI Telecomunicación. D-204
Ciudad Universitaria s/n.
Madrid 28040.
e-mail: amoreno@lst.tfo.upm.es
Tel. +34913366834 ext 3407 / Fax: +34913366828