T.C.L es el nombre de un proyecto de código abierto iniciado por Ximian y actualmente impulsado por Novell (tras la adquisición de Ximian) para crear un grupo de herramientas libres, basadas en GNU/Linux y compatibles con .NET según lo especificado por el ECMA.

Mono posee importantes componentes útiles para desarrollar software:

- Una máquina virtual de infraestructura de lenguaje común (CLI) que contiene un cargador de clases, un compilador en tiempo de ejecución(JIT), y unas rutinas de recolección de memoria.
- Una biblioteca de clases que puede funcionar en cualquier lenguaje que funcione en el CLR (Common Language Runtime).
- Un compilador para el lenguaje C#, MonoBasic (la versión para mono de Visual Basic), Java y Python.
- El CLR y el Sistema de tipos común (CTS) permite que la aplicación y las bibliotecas sean escritas en una amplia variedad de lenguajes diferentes que compilen para byte code
- Esto significa que si, por ejemplo, se define una clase que realice una manipulación algebraica en C#, ésta pueda ser reutilizada en cualquier lenguaje compatible con CLI. Puede crear una clase en C#, una subclase en C++ e instanciar esa clase en un programa en Eiffel.
- Un sistema de objetos único, sistema de hilos, bibliotecas de clases y sistema recolector de memoria pueden ser compartidos por todos estos lenguajes.
- Es un proyecto independiente de la plataforma. Actualmente Mono funciona en GNU/Linux, OpenBSD, FreeBSD, UNIX, Mac OS X, Solaris y plataformas Windows.

Existe un proyecto similar, llamado Portable.NET, es parte del proyecto dotGNU.

Tan pronto como Microsoft publicó los documentos que especifican la arquitectura .NET en diciembre de 2000, Miguel de Icaza (cofundador de la empresa Ximian y de la GNOME Foundation) comenzó a interesarse en ellos.

GNOME siempre había luchado por proporcionar facilidades al programador, y una de las

características más conocidas es que existen multitud de *bindings* para poder utilizar cualquier lenguaje de programación para desarrollar aplicaciones. Pero la elaboración de dichos *bindings* era tremendamente laboriosa y cada vez que se realizaba un cambio en la interfaz original, era necesario cambiar todos y cada uno de los bindings.

Para intentar mejorar y facilitar la reutilización de código se realizó una implementación de componentes, llamada Bonobo, utilizando CORBA. Pero tampoco ha tenido éxito, ya que era necesario que todo el software utilizase esa característica y eso no fue así. Por tanto, con .NET se abre una nueva puerta para conseguir hacer de GNOME en un futuro un escritorio mejor y más atractivo tanto para usuarios como para programadores. Con esta tecnología por fin se consigue lo que el proyecto GNOME siempre había buscado, independencia del lenguaje para programar en dicho escritorio.

Miguel de Icaza, después de analizar el intérprete del *byte code*, advierte que no existen especificaciones. En febrero de 2001 comienza a indagar por dicha información en las listas de correo de.NET y al mismo tiempo comienza a trabajar en un compilador C# en cooperación con Rhys Weatherley y Jay Freeman, el mismo fue programado en C# como un ejercicio para demostrar su potencia.

En abril de 2001, la ECMA publica el formato de archivos faltante y en GUADEC (6 al 8 de abril de 2001) Icaza demuestra las habilidades de su compilador. Después de un minucioso análisis, donde claramente se concluye que es posible construir esa tecnología, Ximian reasigna recursos humanos de otros proyectos y crea el equipo Mono. Aspirando a tener una herramienta que fuese un sustituto completo de la tecnología .NET, formaron The Mono Open Source Project, el cual fue anunciado en julio de 2001, en la conferencia de O'Reilly.

Pasaron 3 años hasta que el 30 de junio de 2004 Mono 1.0 finalmente fue lanzado.

Bibliotecas de clase

Las bibliotecas de clase proveen un conjunto de facilidades que ayudan al desarrollo de

aplicaciones. Son escritas en primer lugar en C#, pero gracias al lenguaje común de especificación (CLS), las mismas pueden ser invocadas en cualquier otro lenguaje de .NET. Las bibliotecas de clase están estructuradas en espacios de nombres (namespaces) y puestas en producción en bibliotecas compartidas denominadas assemblies o ensamblados. Cuando hablamos del framework de .NET, nos estamos refiriendo en primer lugar a las bibliotecas de clase.[JAC]

Espacios de Nombres (namespaces) y Ensamblados (assemblies)

Los espacios de nombres son un mecanismo que permite agrupar lógicamente clases similares en una estructura jerárquica, evitando así conflictos de nombres. La estructura se implementa utilizando palabras separadas por puntos. Por ejemplo, *System.IO* o *System.Net*agrupan las clases para acceso a ficheros y para comunicaciones de red, respectivamente.

Los ensamblados son el paquete físico de las librerías de clase. Son archivos con extensión.dll, como las librerías de Windows. Ejemplos de librerías son mscorlib.dll, System.dll, System.Data.dll.

Los espacios de nombres, por lo general, están compuestos por muchos ensamblados y un ensamblado puede estar compuesto de varios archivos.

Lenguaje Común de Infraestructura (CLR)

El lenguaje común de infraestructura o más comúnmente llamado Common Language Runtime (CLR) es implementado por el ejecutable de Mono. El runtime es utilizado para correr aplicaciones compiladas en .NET. Este lenguaje común de infraestructura está definido en los estándares ECMA y ECMA-335. Para ejecutar una aplicación se deberá invocar el *runtime* con los parámetros adecuados.

Lenguaje Común de Especificación (CLS)

Se encuentra especificado en el estándar ECMA-335 y define la interfaz con el CLR. Por ejemplo, convenciones sobre el tipo de datos que se utilizará para implementar los enumerados. El compilador Mono genera una imagen que cumple con el CLS, esta imagen está codificada en el denominado Common Intermediate Language (CIL) o Lenguaje Intermedio Común. El*runtime* de Mono toma dicha imagen y la ejecuta.

Mono y las patentes de Microsoft

La implementación de Mono de esos componentes de .NET no sometidos a ECMA para su estandarización ha levantado algunas preocupaciones por la posible violación de patentes de software durante la vida del proyecto. En particular, la discusión se desarrolló por si Microsoft podría o no destruir al proyecto mono mediante demandas sobre las patentes violadas.

En la actualidad existe un vivo debate sobre la (in)conveniencia de aceptar y usar Mono en la comunidad de desarrolladores de GNU/Linux. Los principales argumentos en contra de Mono son:

- No está libre de patentes de software, y existe el riesgo de que Microsoft exija licencias para usar C# / CLI.

Por otra parte, el proyecto Gnome está desarrollando un lenguaje alternativo, Vala, creado específicamente para desarrollar aplicaciones para Gnome, pero libre de las potenciales amenazas de Microsoft.