

Mono

Miguel de Icaza
miguel@novell.com



Novell.



Agenda.

A brief introduction to Mono.

Most of this talk focuses on:

- The JVM vs CLI differences.
- C# vs Java language differences.

Today's Mono state



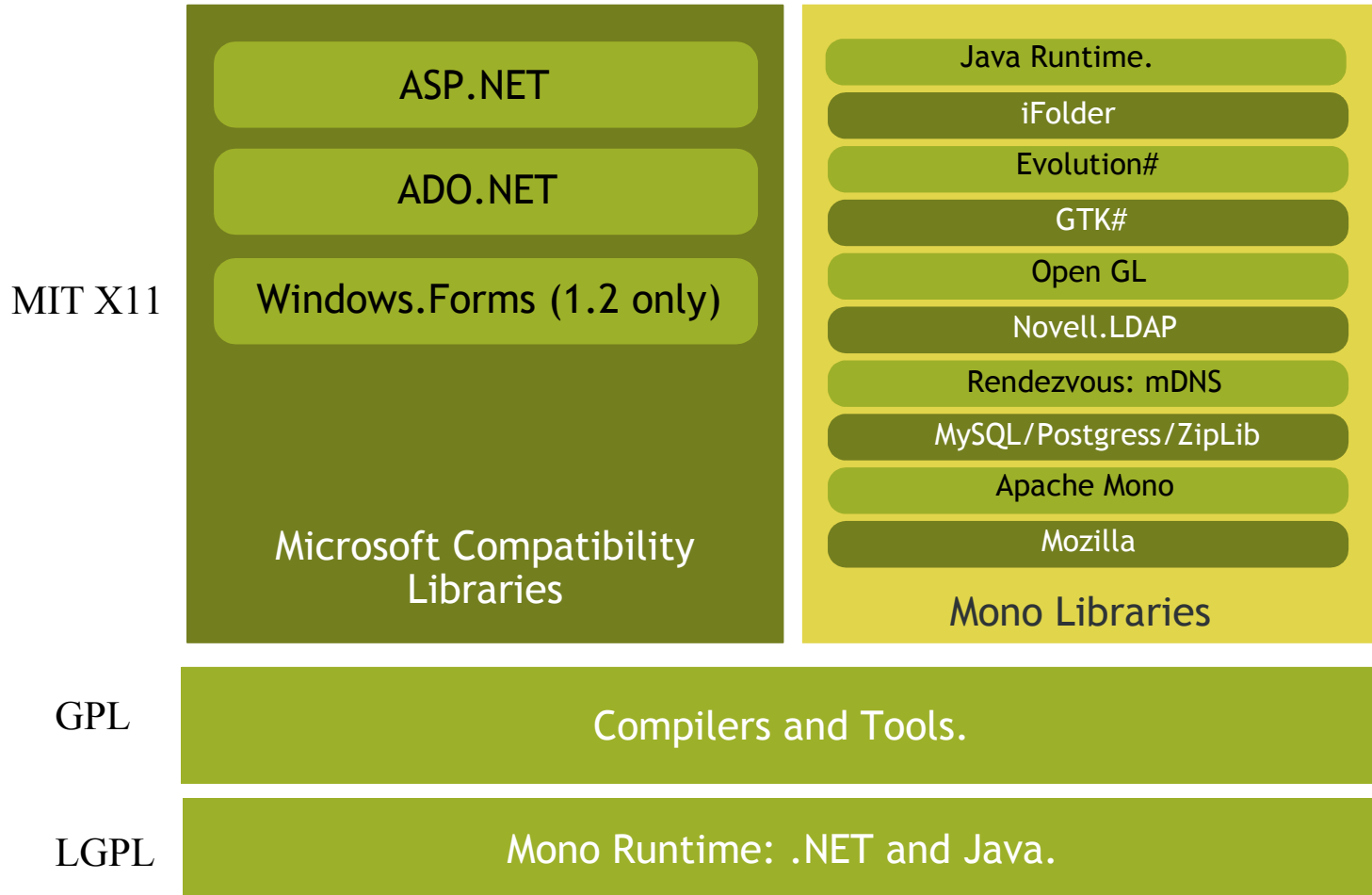
What is Mono?

An Open Source implementation of ECMA CLI and C#

- Cross platform:
 - Unix family: Linux, MacOS X, Solaris, HP-UX, etc.
 - Windows family: 2000, NT, XP.
 - Embedded systems (tiny profile).
- Native CodeGen:
 - 32 bits: x86, sparc, s390
 - 64 bits: x86-64, sparc v9
- Open Source compilers and tools:
 - Today: C# 2.0, Java, Python, Boo, Nemerle.
 - Preview: VB.NET, Jscript.



The Two Stacks





Developers.

Mono is a fairly large open source effort:

- 30+ full time developer from various companies.
- ~330 SVN accounts.
- ~60-70 different people committing per month.

Companies developing Mono:

- Novell.
- Mainsoft.
- Voelcker.
- Embedded system vendor.

Common Language Infrastructure



Common Language Infrastructure

A managed runtime:

- Very similar to Java's VM in spirit.
- Common Intermediate Language (CIL)
- Strong focus on Multi-language support.

Object Model:

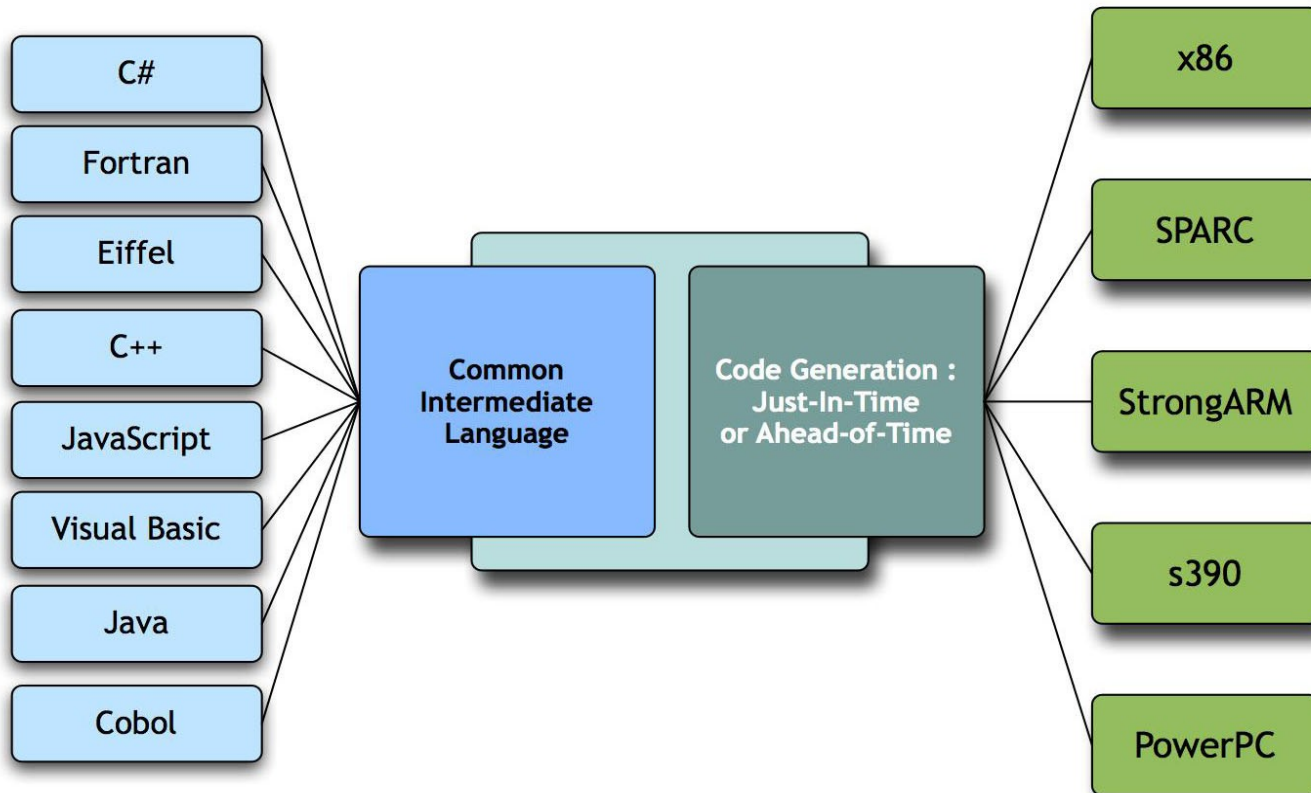
- Single inheritance, multiple-interface implementation.

Standard Services:

- GC, Threading, Reflection.
- JIT/AOT compilation.
- Instruction set with support for generics.



Multi-language and multi-platform.





CLI: Assemblies and Versioning.

Assemblies are the unit of deployment.

- As simple as a single .exe or a single .dll
- Or might be made up of multiple files (held in a directory)
- Or multiple resources in a single file.

Public assemblies are versioned and signed.

- Installed into a system-managed location.
- Strong backwards-compatibility support.
- Versioning of symbols encoded into references.



Sample

```
.assembly extern mscorlib
{
  .ver 1:0:5000:0
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
}
.assembly extern System
{
  .ver 1:0:5000:0
  .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
}

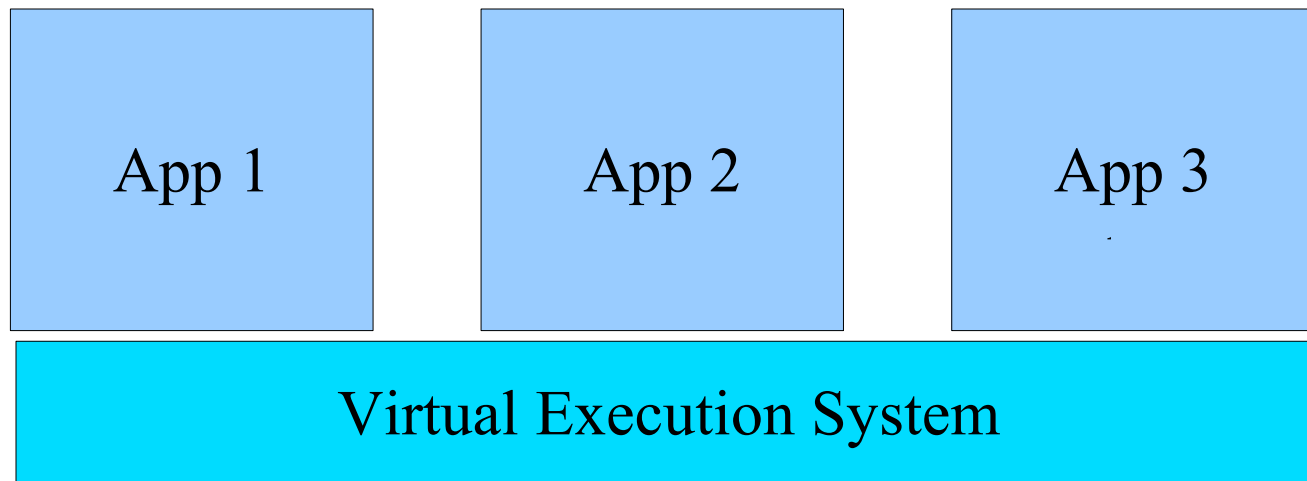
newarr [mscorlib]System.Object
```



CLI: Application Domains.

Similar to processes in Unix.

- Can run more than one application on the same VM.
- Used extensively for hosting multiple applications.
- Isolates failures across applications.





CLI: Reflection.Emit.

An extension to the system reflection.

- Used to produce and execute code at runtime.

Dynamically:

- Used by the IKVM Java VM
 - JITs JVM bytecodes into CIL bytecodes.
- Used by IronPython/JScript:
 - Produces code dynamically.

Statically: Batch compilers

- Mono's C# and VB compilers.
- IKVM .jar to .dll compiler.



CIL: Value Types (structures).

ValueTypes are blobs of memory.

- Similar to C structs.
- No vtable/class pointer at the beginning of each.
- Efficient to allocate
- Minimizes pressure on the GC.
- Can be allocated on the stack.

Sample structs:

- System.Byte, System.Int32
- struct Point { int x, y; }

Limitation: structs can not contain virtual methods.



CLI: Support for other languages.

Project 7

- Commercial and research languages.

Pointer arithmetic opcodes:

- Required for efficient support of C, C++, Fortran, Cobol.
- Only trusted assemblies can use these operations.
- Verifier ensures that code that contains unsafe operations can not execute without the required permissions.

Functional language support:

- The `tail' instruction.
- Extended-IL research project.



CLI: Generics support.

The VM is aware of generic instructions.

- Generics in C# are not only syntactic sugar.
- They actually have a performance benefit.
- Generated code is tuned for datatype (or shared).

Metadata is preserved, not lost.

Works on any datatype

- No limitations on what can be used:
 - `new List<int>`, `new List<Calendar>`

C#



C# basics.

New Microsoft language

- Inspired by Java.
- Incorporates feedback from the community.
- Follows simple recipe for 'A better Java'.

Strong focus on:

- Versioning and evolution of software:
 - Ubiquitous versionin/evolution considerations.
 - Deal gracefully with core upgrades.
- Event-based programming.



C# Source Code and Properties.

Programmer is in control of layout

- No forced directory structure to match class naming.
- Many classes-per file.
- or many files for a class.

Properties: syntactic sugar:

- `b.Text = "hello";`
- ...
- `class Button {`
 - `string Text {`
 - `get { return _text; }`
 - `set { _text = value; ForceRepaint (); }`
 - `}`



C# - delegates.

A delegate points to a method/object.

- Similar to C function pointer + context.
- Avoids creating class with a simple method.

Usage:

```
delegate bool Compare (object a, object b);  
Compare c = MyComparer;
```

...

```
BubbleSort (object [] data, Comparer c)  
{  
}
```



C# - events.

Builds on the delegate concept.

- Similar to “signals” in Gtk+
- Invoke every method.

Declaration:

- event Action Clicked;

Adding/Removing:

- button.Clicked += method;
- button.Clicked -= method;

Invoking:

- button.Clicked ();

Consumed directly by GUI designers



C# - Anonymous Methods.

Allows creation of code-blocks inline:

- `button.Clicked += delegate { print (“clicked”); }`

Unlike Java's Inner classes, it is not macro expansion:

- Field members, parameters, locals can be captured.
- Captured variables retain activation frame semantics:
- `for (int i = 0; i < 10; i++) {`
 - `int j = i;`
 - `b [i].Clicked += delegate { print (“{0} {1}”, i, j);`
- `}`



C# Defaults.

In C# methods are not virtual by default.

- Programmer has to explicitly flag method as virtual.

Contract-wise:

- Explicit contract with consumer, not implicit.
- It is a bad idea to start with.

Side effect:

- JIT engines do not require complex optimization for callvirt inlining to achieve reasonable performance.



C# iterators.

Simplifies implementation of IEnumerable, IEnumerator:

- IEnumerable GetRange (int start, int end)
- {
 - for (int i = start; i < end; i++)
 - yield record [i];
- }
- ...
- foreach (Record r in GetRange (2, 20)){
 - print (r.Name);
- }



Platform Invocation.

Allows simple consumption of native APIs

- No need for separate C/C++ stub file.
- C# source code can be self-contained.

Example:

- [DllImport (“libc”)]
- extern static int system (string s);

Runtime provides extensive marshalling features

- Arrays, string conversions.
- Custom-marshallers for complex data types.
 - To/From methods must be implemented.

Universal Runtime.



A Unifying VM

Mono's VM can be embedded:

- Similar to Tcl, Python, Perl, others.
- Can run side-by-side with other Vms.
- Bridges built to expose various languages to others:
 - MonoConnect (XPCOM)
 - PythonNet (CPython interpreter connector.)
 - Perl.NET (Perl connector).

Mono can produce native code:

- No JIT performance hit.

In addition:

- Languages that target IL can speak to each other freely.



CLS: Common Language Specification

CLS is a set of rules:

- For consumers and producers.
- Establishes a subset for consumers/producers
- Intended to establish a common subset for interop.

Languages can be classified:

- Consumers, Producers, Consumer/Producer.

CLS compliance attribute:

- Attached to assemblies, classes, methods.
- Compiler will error out on infringements.



Importance of CLS

Enables choice of language for domain application.

- Enables consumption of third-party components.
- Bad idea to write a single project in 10 languages.
- A well specified interop subset.

A level playing field for compiler vendors.



Java

Java:

- With IKVM Java and C# code run side-by-side.
- Java support is as good as GNU Classpath libraries are.
- Eclipse, Derby work out of the box.
- Expose .NET to Java.
- Consume Java from .NET
- Native Java:
 - JVM to CIL byte conversion (ikvmc)
 - CIL to native code conversion (mono's Ahead-of-Time).

Downsides:

- Two class libraries in memory at once.



Python.

Jim Hugunin's Python on .NET implementation.

- Started as a proof that CIL was not good enough.
- Impressive results:
 - Dynamic languages can run fast, or faster on .NET
- IronPython is on average faster than CPython on .NET
 - 90% the speed on Mono: we are working on it.

IronPython can consume all classes available on Mono/.NET



C/C++

New Microsoft compiler produces pure CIL

- Available on Whidbey
- No more x86 code embedded.

Managed C++

- Being spec'd out by ECMA.
- ISO chair on the group.
- No plans to support these extensions.

Prototype gcc2whirl, whirl2il can run simple Gtk apps.

- Based on the SGI Whirl work.
- Pathscale compiler maintains the WHIRL backend.
- Today gcc4 might be a better option.



Debugger.

The Desktop.



Mono On The Desktop

Novell's desktop development platform

- For new software.
- For extending existing software.
- Beagle, Dashboard, F-Spot, Debugger, Evolution plugins.
- iFolder: multi-host synchronization (files, contacts).

Gtk# - a binding to the Gtk+ and Gnome APIs.

- MonoDevelop (port of SharpDevelop)
- Tomboy.
- Gnome Fax
- Blam RSS Aggregator
- Muine music player, and many more!



GUI Libraries.

Gtk#

- A binding of the Gtk+/Gnome Libraries.
- Supported and developed by Novell.
- Works on MacOS, Windows and Linux.
 - Uses theme engine on Windows XP.
 - Alien look on MacOS X.

Windows.Forms:

- Under development at Novell
- Will be available with Mono 1.2
- Unlike Gtk#, absolute positioning.
- Uses theme engine on each OS:
 - Win32, Gtk+, CoreGraphics

Cocoa#

- Built by OSX community.
- Only available on OSX.
- Native OSX apps.

wxWidgets:

- Uses native widgets on each platform

Future



Future of Mono: Short Term.

Mono 1.2: incremental update.

- Add Windows.Forms.
- ASP.NET, ADO.NET scalability.
- C# 2.1 language release.
- VB.NET compiler.

Mono features:

- Debugger.
- Gtk# 1.2
- Cocoa# 1.0
- Cairo integrated in Gtk#

Estimated: September 2004



Out-of-Band Development.

Ports:

- Itanium.
- StrongARM

Precise, compacting GC:

- Have an early prototype.
- Post 1.2

Advanced Compiler Optimizations.



Mono: Medium Term

Mono 2.0:

- Stetic: new GUI designer for Gtk#
- Support the .NET 2.0 profile
 - ASP.NET 2.0:
 - Master pages, new controls, web components.
 - Pre-compiled sites
 - ADO.NET 2.0
 - System.Xml 2.0 (done).
 - Windows.Forms 2.0
- Code Access Security debuts.

Release date: September 2006



Avalon, Indigo.

Avalon

- New UI platform from Microsoft.
- To be released in 2006
- Available for Longhorn and XP.

Indigo:

- New stack for RPC applications.
 - Upgrade to web services stack.
 - Makes `web' optional.

No plans at this point to work on these



More Information.

Mono:

- <http://www.mono-project.com>
- <http://www.go-mono.com/monologue> (mono blogs)

Email:

- miguel@novell.com

Blog:

- primates.ximian.com/~miguel/activity-log.php

Novell®

General Disclaimer

This document is not to be construed as a promise by any participating company to develop, deliver, or market a product. Novell, Inc., makes no representations or warranties with respect to the contents of this document, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc., reserves the right to revise this document and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes. All Novell marks referenced in this presentation are trademarks or registered trademarks of Novell, Inc. in the United States and other countries. All third-party trademarks are the property of their respective owners.

No part of this work may be practiced, performed, copied, distributed, revised, modified, translated, abridged, condensed, expanded, collected, or adapted without the prior written consent of Novell, Inc. Any use or exploitation of this work without authorization could subject the perpetrator to criminal and civil liability.



Novell.