Mono's progress and roadmap

Paolo Molaro
lupus@ximian.com

Novell

What is Mono?

An Open Source implementation of .NET

- Cross platform:
  - Unix family: Linux, MacOS X, Solaris, HP-UX.
  - Windows family: 2000, XP.
  - Embedded systems (tiny profile).
- Open Source compilers and tools:
  - Today: C# language, IL, Java, Nemerle.
  - Preview: VB.NET, Jscript, Python.
- Most of the non-windows specific stuff implemented
- Additional features

Motivations

Increase Developer Productivity.

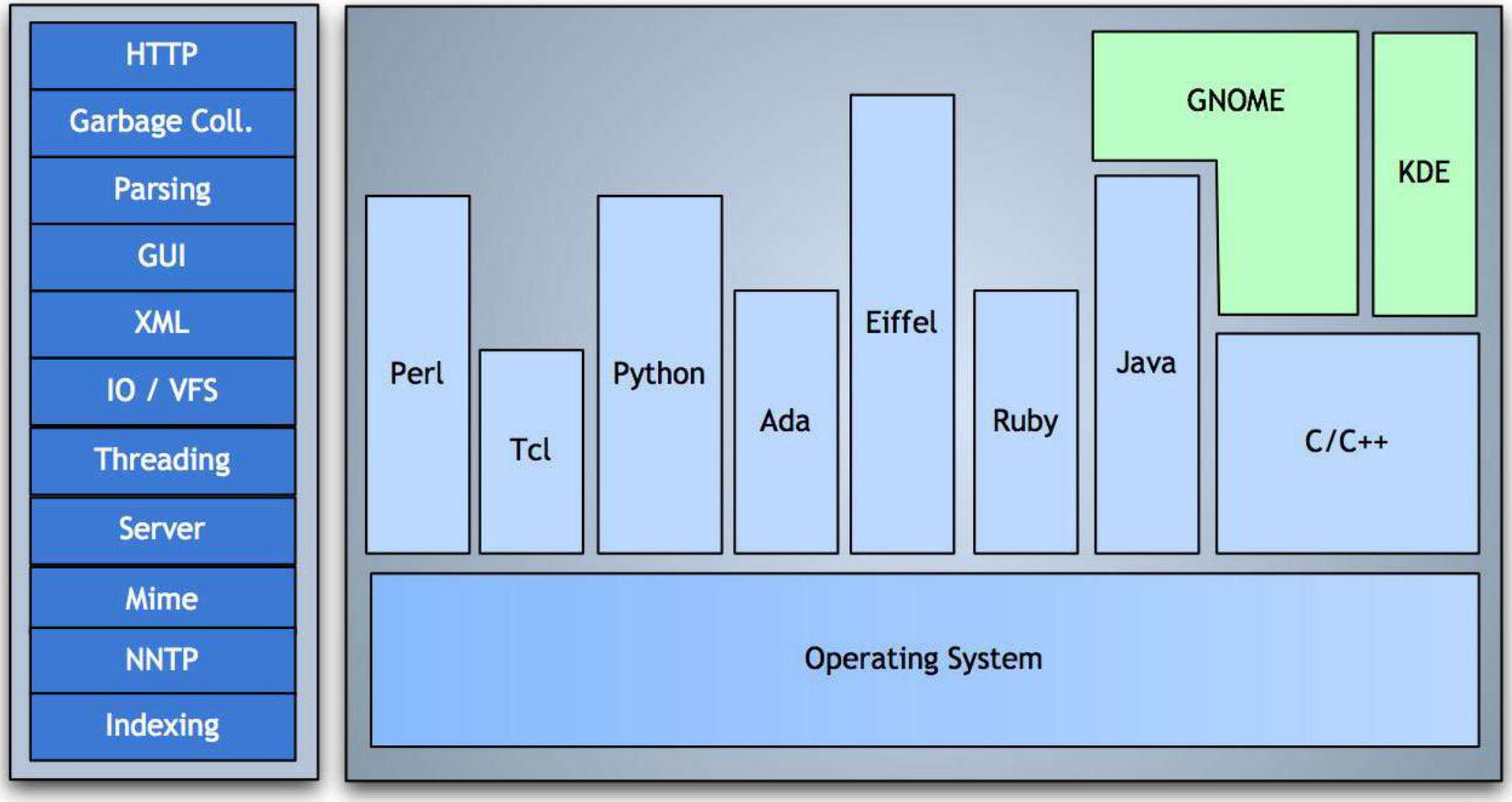
- Modern platform to Unix.
- Managed environment.
- Mainstream langueage with the features most free software langauges had for ages.
- Evolving platform.

Unify API reuse.

- Reduce API-based language fragmentation
- Multiple language support

Side effect: Windows to Linux migration.

Unix:
Everyone Building its Own Platform



Novell

Mono Today.

Mono 1.0 release a few months ago.

- .NET framework 1.1-based.
  - Minus Windows.Forms, EnterpriseServices
- Gtk# and other Un*x integration assemblies
- Extensive third-party database support
- Relax NG extensions.
- Mono.* namespaces.

GUI Tools:

- Monodoc and Monodevelop.

Mono Users Today.

Novell's desktop development platform

- For new software.
- For extending existing software.
- Beagle, Dashboard, F-Spot, Evolution 2.next
- Gtk# community.

ASP.NET applications on Unix

- Voelcker: 400 servers, 150,000 users

Apple/Cocoa#: recent community.

Gnome Project.

Bring Linux to the desktop.

- Started in 1997
- Fill the gaps on the desktop offering.
- Raising the programming level:
  - Component-based software.
  - Raise programming level: language bindings.

Technically:

- C-based APIs, contracts to bind it.
- CORBA used for exposing components APIs.
- Far from perfect solutions.
- Bindings came out of schedule.

The Ximian Background.

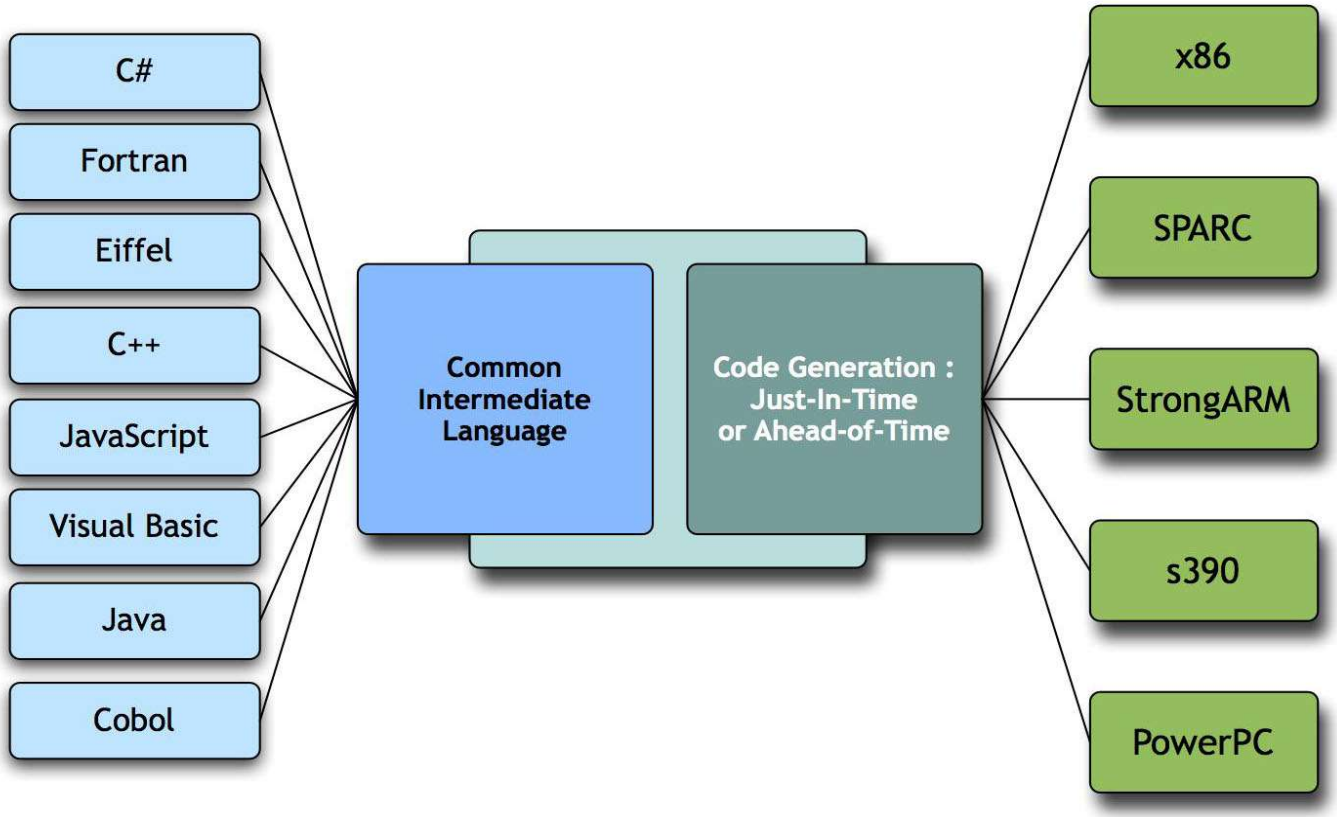
Ximian: focused on making Linux succeed on Desktop.

Evolution: our large application

- Standards compliant, best email/calendar/address.
- Complex: multi-threaded, multi-process
- Too many standards.
- Innovation was hard
- At peak for 1.0: 17 developers, 2.5 years Too Expensive.

We needed better tools.

Multi-language and multi-platform.



Novell

Reuse.

Existing .NET:

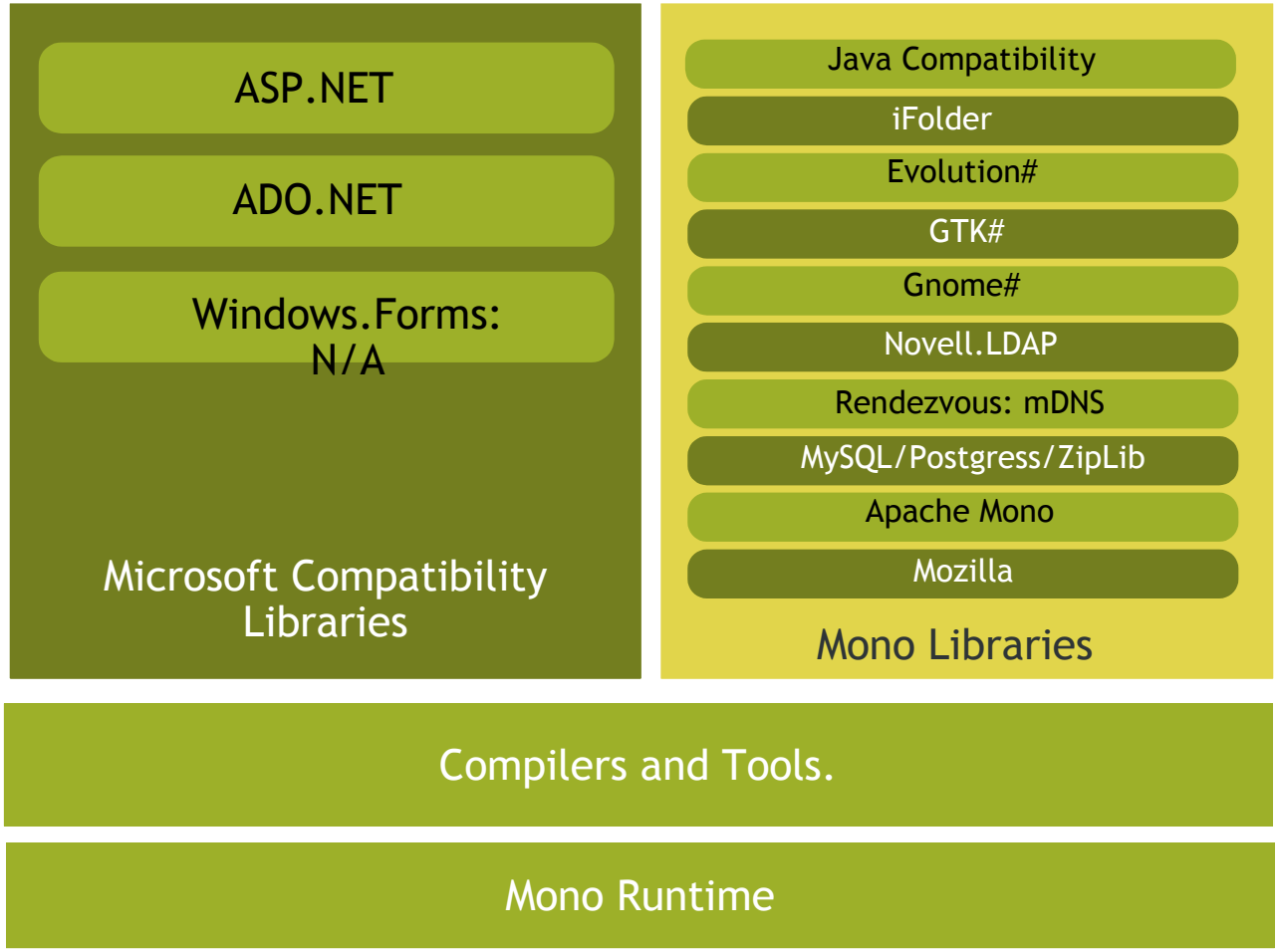
- Third party components written in C#, VB.
- Third party compilers:
  - Eiffel, Ada, Fortran, C/C++, Python.

Java:

- Seemless integration of Java libraries and applications.
- Execute C# and Java code side-by-side
- Share components, expose components.
- The IKVM Virtual Machine.

The Ultimate Open Source Reuse.

The Two Stacks.



Novell

Mono Development.

Mono is an effort lead by Novell.

- 20 Engineers from Novell (5 at the time of Ximian)
- 300 developers from the Open Source community.

Like other open source efforts, other companies help:

- Embedded system vendors.
- Mainsoft
- SourceGear.

Reuse pieces of open source code:

- Intel's ORP research.
- SGI/HP's Boehm GC.

Virtual Execution System.

Two execution systems today:

- Native Code Generator. (ppc, x86, sparc)
- Interpreter (bootstrap helper).

Native Code Generator:

- Just-in-Time compilation mode.
- Ahead-of-Time batch compilation.
  - Avoids startup time
  - Increases shared code across applications.
- Fairly advanced compiler optimization platform.

GC: Boehm conservative GC.

Mini JIT

Two intermediate representations

- IL -> Tree based
  - Most optimizaions done on the trees
  - BURS rules transforms trees into list of quadruples
- quadruples of opcode, dreg, sreg1, sreg2
  - Close to the CPU, but still allow for easy porting and code sharing between multiple targets

Optimizations

- Can be individually enabled
- The quick ones are enabled by default
- SSA framework enabled manually (or during recompilation with profiling info): SCCP, ABC removal

Embeddable VM

The complete VM is on a library.

- Extend existing applications.
- Add scripting.
- Gradual transition to managed code.
- Do not lose investment in current code base.
- Create snippets of code that get jitted.
- Export new facilities to the CLR.
- Plugin interface for multiple languages.
- Easy to use and low overhead.

Where we are going?

Continue improving Unix, Gnome, Cocoa

- iFolder
- Beagle/Novell Dashboard.

Continue .NET/Java compatibility work:

- To allow reuse of third party .NET libraries.
- To allow reuse of Java third party libraries.

Support scripting and new languages:

- IronPython, Boo, Nemerle

Improve our development tools:

- MonoDevelop
- Mono Debugger
- Mono Documentation.

Mono Development.

Three tracks of development.

Mono 1.0: Stable

Mono 1.1: Incremental development

Mono 2.0: .NET 2.0 APIs and bigger runtime changes

Future of Mono: Short Term.

Mono 1.2: incremental update.

- Add Windows.Forms.
- Add Visual Basic compiler.
- ASP.NET, ADO.NET scalability.
- C# 2.0 compiler.

Mono features:

- Debugger.
- Gtk# 1.2, Cocoa# 1.0
- Generics.
- More ports (sparc64, amd64, s390)
- Faster (20-50% faster than 1.0 already in many tasks)
- Stability.

Mono: Medium Term

Mono 2.0:

- Support the .NET 2.0 profile
  - ASP.NET 2.0
  - ADO.NET 2.0
  - System.Xml 2.0
  - Windows.Forms 2.0
- Code Access Security.
- Generational precise garbage collector
- More ports (S/390x, PPC64, ARM, MIPS, Itanium, ...)
- More optimizations in the JIT
  - New register allocator

Ongoing work.

Optimizations

- SSA-Partial Redundancy Elimination.
- New register allocator + scheduler (PPC, SPARC)
- Moving more VM code into managed land
  - Avoid managed<->unmanaged transitions
- Per-architecture rule tuning
- AOT
  - ELF-like relocations (increase page sharing).

Precise Garbage Collector

- Today we use Boehm (using as precise as possible).
- Currently incrementally fixing to introduce a moving GC

Future of Mono as a VM.

More 'scripting' language runtimes and compilers

- Perl, ruby, Tcl ported to run on Mono and other VMs.
- Will have more than one implementation, driving stabilization of interfaces and standardization
- It will push for enhancements of Mono (not just speed, but new features at the runtime and IL level)
  - Dynamic optimizations (self-modifying IL code)
- Creation of a common interface for use when the concepts of types and methods in a language aren't (can't be) mapped to types and methods in the VM

Future of Mono as a VM.

Smaller footprint for embedded systems.

- Tradeoff memory for speed
- Remove some features (Decimal, Re.Emit, locales, verification, Remoting support,...)
- Realtime extensions

Desktop systems

- Reduce GC pause times
- Reduce startup time (fast JIT, AOT)
- Reduce memory footprint

Scalability enhancements for SMP, server loads.

- Threaded GC
- Better performance with shared code

Future of Mono as a VM.

Security and sandboxing

- Statistic-based low-overhead resource accounting

Support from cpu designers and operating systems

- Thread local access (ABI issue): pthread, __thread
  - Needs to be fast for appdomains, code sharing
- VM dirty bits for use in GC
- Virtualized performance counters
- IP-relative addressing (on 64 bit platforms tradeoffs between icache footprint, memory reads)
- Don't pessimize writes to code (callsite patching)
- Standards for controlling a thread (suspend/resume, move to safe points...)

The importance of mistakes.

In the design of a VM

- To get researchers to solve the issues created
- Job security!^W^W

Examples:

- Synchronized methods in Java
  - Thin lock implementations
- Heap allocations
  - Escape analysis
- Imperative langauges
  - SSA form in compilers to convert into functional

Make sure we don't get back to lisp machines in 20 years

More Information.

Mono:

- http://www.mono-project.com

Email:

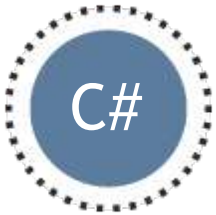
- lupus@ximian.com

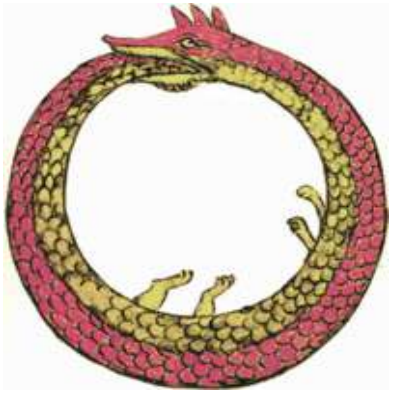
Mailing list:

- http://lists.ximian.com/mailman/listinfo/mono-devel-list

Novell

Mono C# Compiler.





- C# written in C#
  - Fast: 2.5 seconds to rebuild itself.
  - 54,000 lines of code.
  - Includes JIT time.
- Implements C# 1.0
  - 2.0: generics and iterators.
- Self-hosting since Jan 2002.
  - Six months since start.
  - Two developers.
  - AOT:
  - 30% performance increase.

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.

Novell