JAZYK C# A .NET FRAMEWORK NA LINUXU

Aleš Keprt

Katedra informatiky, FEI, VŠB - Technická Univerzita Ostrava, 17. listopadu 15, 708 33 Ostrava ales.keprt@vsb.cz

Abstrakt
Microsoft .NET Framework je fenomén, který v posledních dvou letech doslova hýbe světem programátorů tvořících software pro Windows. Microsoft, jakožto producent nejrozšířenějšího operačního systému i vývojových nástrojů pro něj nyní prosazuje svůj .NET Framework jako jedinou platformu pro všechny typy aplikací.
Jednou větou se dá .NET popsat jako platforma velmi podobná Javě, kterou však navrhla společnost Microsoft a která má na rozdíl od Javy ve Windows doslova na růžích ustláno. Ačkoliv dnes skutečně všechno nasvědčuje tomu, že .NET *dobude* Windows, existují i další operační systémy, jejichž uživatelé by také rádi využili přednosti .NET Frameworku. Microsoft naštěstí navrhl .NET jako do jisté míry platformově nezávislý produkt a jeho klíčové části standardizoval u ECMA, čímž umožnil, aby se do vývoje alternativních implementací .NET Frameworku zapojili třetí strany. Tento článek popisuje naše zkušenosti s testováním .NETu v Linuxu, jakožto nejběžnějším alternativním operačním systému, zvláště na serverech. Právě na servery je platforma .NET také určena, zejména pro snadné vytváření dynamického webu.

Děkuji Mgr. Vilému Vychodilovi z Katedry informatiky PřF UP Olomouc, bez jehož technické podpory a pomoci by tento článek nemohl vzniknout.

1. Úvod

Jedním z moderních programovacích jazyků, který je díky firmě Microsoft v poslední době stále více používán, je jazyk C#. Uživatelé Windows k mohou k práci s tímto a několika dalšími jazyky pohodlně použít Visual Studio .NET a hotové aplikace díky .NET Frameworku pak spouštějí přímo v systému.

Microsoft navrhl .NET jako platformově nezávislou vrstvu, skutečnost je však taková, že zatímco na jedné straně je vyvíjeno úsilí o portování stále dalších a dalších programovacích jazyků do prostředí .NET, na straně portování .NETu na další operační systémy žádný výsledek ze strany Microsoftu vidět není. Celý .NET je stále k dispozici jen pro Windows.

Standardizace klíčových prvků platformy .NET u ECMA (www.ecma.ch), naštěstí umožnila, aby se o tvorbu .NET Frameworku pro další operační systémy mohly zajímat také třetí strany. Zatímco Microsoft už pracuje na .NET Frameworku verze 2.0 (odložen na rok 2005), alternativní implementace jsou teprve ve vývoji, naštěstí však již v takové fázi, že je již možno začít je pro jednoduché aplikace používat. Proto jsme .NET vyzkoušeli i my, na školním linuxovém serveru.

Druhá kapitola rámcově a velmi stručně seznamuje s hlavními rysy platformy .NET. Třetí kapitola popisuje možné alternativy originálního Microsoft .NET Frameworku, následuje pak popis projektu Mono, který jsme si pro použití zvolili. Závěr článku stručně nastiňuje plány dalšího vývoje Mono v blízké budoucnosti.

2. .NET Framework

2.1 Co je to .NET Framework

Sám Microsoft popisuje .NET Framework jako sadu technologií k přeměně internetu na plně distribuovanou výpočetní platformu, která nabízí nové způsoby vytváření aplikací z kolekcí webových služeb (viz [2]) nebo také jako platformu pro XML webové služby (viz [4]). Cílem Microsoftu je tedy zejména nasazení .NET Frameworku pro snadné vytváření dynamických www stránek s návazností na databáze a snadnou tvorbu distribuovaných aplikací. S tím vším si platforma .NET skutečně hravě poradí, avšak celý .NET má mnohem více schopností.
Celý systém má v následujících letech postupně nahradit stávající vývojové nástroje, jazyky a programovací techniky používané na platformě Windows, jako jsou Visual C++, Visual Basic, ASP, ADO, COM+, JScript, Java nebo VBScript. Platforma .NET nápadně připomíná Javu, avšak neomezuje se na jediný programovací jazyk, naopak poskytuje jednotný rámec pro mnoho jazyků, které navíc mohou být libovolně kombinovány i v rámci jediné aplikace.

2.2 Common Language Runtime

.NET Framework je, podobně jako např. J2EE, rámec, ve kterém teprve běží jednotlivé aplikace a služby. O samotný běh programů, včetně správy paměti a bezpečnosti se stará CLR (common language runtime). Narozdíl od Javy, která je obvykle interpretována ve virtuálním stroji, se v .NETu kód před spuštěním vždy nejprve zkompiluje. Standardním způsobem je JIT (just-in-time), kdy se kompiluje každá metoda při jejím prvním použití. Alternativně je možno přeložit celý kód najednou před jeho spuštěním. Programy v .NET jsou tedy při běhu poměrně rychlé. I když programátoři zvyklí na rychlost C++ se mohou na tento moderní přístup programování dívat s rozpaky, praxe jasně ukazuje, že výhody z rychlejšího vývoje aplikací v moderních objektově orientovaných jazycích s automatickou správou paměti obvykle jasně převáží jakékoliv potíže s rychlostí jejich běhu.

2.3 Bezpečný a řízený kód

Jedna z klíčových vlastností .NETu je označována jako „safe“ nebo „managed code“. Nejde o nic jiného, než klasickou automatickou správu paměti. Ta sice není v .NETu povinná, což umožňuje použití C++ v jeho klasické podobě nebo kombinaci .NET kódu se staršími knihovnami napsanými ještě bez .NETu, aplikace vyvíjené přímo pro .NET však nemají důvod používat neřízený (unmanaged) kód. Kód, který je celý řízený, je označován za bezpečný (safe), není totiž dovoleno používat aritmetiku pointerů a jsou kontrolovány všechny hranice polí, takže se nemůže stát, že dojde k přístupu do špatného místa v paměti (záměrně ani omylem). V dnešní době virů je toto velmi důležité. Řízený a neřízený kód může koexistovat v jedné aplikaci, potom však již nelze hovořit o kódu bezpečném.

2.4 Konec DLL pekla

DLL peklo (DLL Hell) je jedním z již klasických problémů aplikací ve Windows. Sdílené knihovny existují v mnoha různých verzích a instalace nových aplikací může způsobit přepsání knihovny její starší verzí, nebo naopak novější verzí, která však není zpětně kompatibilní. Výsledkem je, že některé programy prostě nemohou společně fungovat. Microsoft nabídl již mnoho návodů, jak řešit tohoto problému (viz [11]), avšak teprve .NET úplně ukončil problémy s DLL soubory tím, že obsahuje inteligentní správu verzí s konfigurovatelnou podporou zpětné i dopředné kompatibility, namísto prostých odkazů na

jména souborů (v případě DLL) nebo potenciálně nekonzistentních zápisů v systémovém registru (v případě COM).

2.5 Metadata

Pojmem často skloňovaným kolem .NETu je slovo „metadata“. Jsou to „data o datech“, neboli informace popisující obsah každého souboru, které jsou uloženy přímo v něm. Metadata mohou být použita i za běhu programu ke zjišťování informací o objektech a jejich typech, o jednotlivých metodách i celých programových celcích. Metadata jsou vždy přibalena přímo k datům, ke kterým patří. Odpadá tedy nebezpečí desynchronizace systémového registru se soubory na disku, apod. Díky metadatům je také možno snadno provádět serializaci objektů (např. pro uložení na disk).

2.6 Intermediate Language

Narozdíl od Javy, .NET není založen na jednom programovacím jazyku. CLR vykonává kód pouze v zásobníkovém jazyku IL (intermediate language), který je spolu s metadaty výstupem překladačů jednotlivých programovacích jazyků. Ačkoliv různé programovací jazyky se mohou velmi lišit, je vyžadována shoda v základních rysech, což umožňuje provázání jazyků na úrovni jednotlivých tříd. Výsledkem je, že můžete např. bázovou třídu z VB.NET zdědit v C# a potom ji instanciovat v NetCobolu. CLS (common language specification) definuje základní vlastnosti, které musí splňovat každý jazyk, aby umožňoval kooperaci s ostatními jazyky. Např. VB.NET nerozlišuje velká a malá písmena, proto abyste mohli zdědit třídu z C++ do VB.NET, nesmíte mít v C++ třídě public nebo protected metody, jejichž názvy se liší jen velikostí písmen.
Výsledkem použití jednotného IL je oddělení programovacího paradigmatu od konkrétního jazyka. Dříve bylo běžné, že programátoři používali určité paradigma (*programovací styl*) podle toho, v jakém jazyku právě pracovali. .NET tento (špatný) zvyk odstraňuje, a tak rozdíl mezi VB.NET a C# nebo C++ je především v použité syntaxi.

2.7 Referenční a hodnotové typy

Narozdíl od pseudo-objektového C++, v .NETu platí základní vlastnost OOP: „Všechno je objekt.“ Datové typy se přitom rozdělují na referenční a hodnotové. Hodnotové jsou základní nestrukturované typy (v C# je to int, char apod.) a také objekty deklarované v C# jako struct. Proměnné hodnotových typů nemohou dědit, ani být děděny, mohou však implementovat rozhraní (interfaces). Jejich instance jsou vytvářeny na zásobníku, což může v některých případech přinést rychlejší běh programu. .NET přitom umožňuje „zabalit“ každou hodnotovou proměnnou do objektu a používat ji jako referenční.

2.8 Základní knihovna funkcí

Jelikož .NET odstiňuje fyzické prostředí (operační systém) od aplikací, je nasnadě, že také obsahuje široké spektrum podpůrných tříd. Ty jsou rozděleny do řady prostorů jmen (namespaces), nejdůležitější z nich jsou:
System – základní datové typy, události, rozhraní, atributy a výjimky
System.Collections – datové kolekce (dynamická pole, seznamy, apod.)
System.Types – třídy ADO.NET pro práci s databázemi, včetně XML databází
System.DirectoryServices – adresářové služby (LDAP apod.)
System.Drawing – kreslení pomocí Windows GDI+

System.Globalization – národní prostředí (abecední řazení, formátování datumu apod.)
System.IO – proudy a soubory
System.Net – síťové protokoly (DNS, HTTP apod.)
System.Net.Sockets – nižší síťové protokoly (TCP/UDP)
System.Reflection – přístup do metadat
System.Runtime.InteropServices – umožňuje používat COM a systémové služby Windows
System.Runtime.Remoting – umožňuje distribuované výpočty
System.Text – podpora kódování textu ASCII, Unicode, UTF-7 a UTF-8
System.Text.RegularExpressions – regulární výrazy pro zpracování textu
System.Threading – podpora vícevláknového programování
System.Timers – časovač
System.Web – client/server komunikace pomocí HTTP, včetně ASP.NET tříd
System.Web.Services – webové služby
System.Web.UI – vytváření ASP.NET stránek a služeb
System.Windows.Forms – třídy pro uživatelské rozhraní okenních aplikací
System.Xml – podpora XML, XPath, XSL
Microsoft.Win32 – systémový registr a systémové události Windows

Uvedený seznam obsahuje jen nejběžněji používané součásti knihovny tříd .NET, přesto je vidět, že možnosti nasazení .NETu jsou mnohem širší, než jen pro avizované internetové aplikace.

2.9 Seskupení neboli assembly

Soubory se spustitelným kódem programů .NETu obsahují namísto přímo spustitelného kódu pouze kód v IL, na první pohled však vypadají jako běžné EXE nebo DLL soubory. Mají totiž formát PE (portable executable), stejně jako skutečné přímo spustitelné soubory ve Windows. Každý takový soubor se v .NETu nazývá assembly, česky pak seskupení, sestavení či balík (závisí na překladateli – bohužel). Seskupení může být složeno i z více souborů, avšak vždy platí, že metadata jsou vždy společně s daty a kódem, ke kterému patří.
Každé seskupení může existovat ve více verzích, .NET podporuje inteligentní správu verzí, umožňuje definovat dopřednou i zpětnou kompatibilitu, soubory lze chránit digitálním podpisem (pomocí privátního a veřejného klíče). V .NETu existuje také podpora pro zpožděné načítání (teprve v okamžiku použití nějaké metody je načten soubor, ve kterém je implementována) a přímé stahování chybějících souborů z internetu.

2.10 Programovací jazyky

.NET je postaven na již zmíněném mezijazyku IL a umožňuje poměrně snadnou implementaci běžných programovacích jazyků. Podpora generování IL kódu je zahrnuta přímo do základní knihovny tříd, takže v literatuře můžete narazit i na ukázkové překladače jednoduchých jazyků (přímo do EXE souborů) v délce několika desítek či stovek řádků (viz. [4]).
Microsoft nabízí překladače několika jazyků přímo v .NET SDK, plus vývojové prostředí Visual Studio .NET. Jedná se především o C#, jazyk vyvinutý přímo pro .NET. Dále Visual Basic.NET, který má nahradit Visual Basic 6, jedná se však spíše o C# se syntaxí Visual Basicu, který zřejmě budou snáze používat programátoři zvyklí na Javu či C++, než programátoři používající Visual Basic 6. Dále je k dispozici C++, které je možno použít pro klasické neřízené aplikace a s jazykovými rozšířeními i pro řízený kód v .NETu. Pro skriptování je JScript.NET a nechybí ani konverze Javy pod jménem J#.NET. Jazyky J# a C# jsou samozřejmě velmi podobné, J# obsahuje navíc knihovnu tříd známou z Javy (bohužel je

kompatibilní pouze s jednou ze starších verzí Javy), zatímco C# je syntakticky bližší C++. Menší programy v Javě je možno překládat a spouštět pod .NETem zcela bez úprav zdrojového kódu.
Třetí strany nabízejí překladače dalších jazyků pro .NET. Jejich použitelnost je někdy spíše otázkou ceny (např. nejobyčejnější verze NetCobolu pro jeden počítač stojí 80000Kč).

3. .NET bez Microsoftu a mimo windows

Předchozí kapitola představila některé základní rysy .NET Frameworku. Jeho nasazení na Windows nic nebrání v cestě, neboť Microsoft nabízí .NET Framework zdarma pro 32bitové, ale i 64bitové a přenosné verze Windows. Pro jiné operační systémy však .NET Framework zatím k dispozici není. Klíčové části celého systému jsou naštěstí standardizovány a tyto dokumenty jsou veřejně přístupné, takže nic nebrání tomu, aby se objevily další implementace .NETu, podobně jako existuje řada implementací C++.

Narozdíl o Javy, kterou Sun poskytuje zdarma pro všechny platformy, ale nikomu dalšímu není umožněno Javu modifikovat, .NET je od Microsoftu k dispozici jen pro Windows, avšak všichni mají volné ruce k tomu, aby vytvářeli další (vlastní) implementace i mutace.

Tak se i stalo, Microsoft .NET Framework není dnes jedinou implementací. Celá situace však má háček v tom, že Microsoft sice část .NETu standardizoval, některé důležité části si však chrání nebo chce chránit patentově, což případnou konkurenci diskvalifikuje. Jedná se především o ASP.NET, ADO.NET a Windows.Forms.

Kromě dalších implementací .NETu se objevily i další zajímavé produkty třetích stran. Jsou to zejména další programovací jazyky, jako již zmíněný Cobol, dále Pascal, Perl, Python, Eiffel, Scheme, SmallTalk aj. Zajímavý je také překladač Javy do IL, který umožní vytvářet ze zdrojového kódu Javy programy v .NETu, slučitelné s programy v jiných jazycích .NETu na úrovni tříd. Zaujmout může i rozšíření System.Remoting o podporu CORBA, což umožňuje distribuované propojení s mnoha jinými aplikacemi.

4. Mono

4.1 Úvod

Má-li být řeč o alternativních implementacích .NETu, je třeba zmínit především Mono (viz [12]). I když i sám Microsoft deklaroval snahu o vytvoření .NETu pro jiné systémy než Windows, v současné době je jedinou použitelnou implementací právě Mono. Jelikož nejrozšířenějším systémem po Windows je jednoznačně Linux, není divu, že právě Linux nás bude zajímat především.

4.2 Co je Mono

Mono je open-source implementace .NET Frameworku, dle oficiálních standardů ECMA. Jeho základem jsou tři prvky:
1. Self-hosting C# compiler, čili překladač C#, který je napsán „sám v sobě“
2. CLR interpreter a JIT compiler, který umožňuje .NETové aplikace spouštět
3. Základní knihovna, obsahující dle možností stejné třídy jako knihovna oficiálního Microsoft .NET Frameworku

4.3 Kde je Mono

Hlavní vývojové platformy, na kterých Mono vzniká, jsou Linux a Windows. (Druhý jmenovaný systém čistě ze snahy o alternativní open-source implementaci .NETu.) Dále je Mono postupně portováno i na další systémy (jako Unix nebo Mac OS), avšak plná funkčnost je zatím jen na procesorech řady Intel x86 (JIT překladač nelze jednoduše portovat). Interpretovaná verze Mono je již k dispozici i pro neintelovské platformy. Zdrojové kódy Mono jsou platformově prakticky nezávislé a jsou k dispozici zdarma pod licencí GNU LGPL, která umožňuje použití i v komerčních aplikacích.
Díky snaze o maximální úroveň kompatibility, Mono skutečně je binárně kompatibilní s Microsoft .NET Frameworkem což umožňuje přímou přenositelnost souborů. Mono tedy umí spouštět programy zkompilované ve Visual Studiu a opačně; soubory vytvořené serializací objektů jsou rovněž přenositelné, jak v binární, tak SODA/XML verzi. Mono zásadně nevytváří žádné nestandardní soubory, čili veškerý jeho programový výstup je PE. Tyto soubory, přesněji aplikace v tomto formátu, se spouštějí příkazem **mono něco.exe**. Ve Windows pak mohou společně koexistovat Microsoft .NET Framework, který je použit při přímém spuštění souboru EXE, a Mono, které se aktivuje výše uvedeným způsobem. V Linuxu nejsou soubory PE standardně použity, je proto možné nastavit jejich automatické spouštění přes Mono.

4.4 Překladač MCS

Microsoft .NET Framework SDK obsahuje překladač C# pod jménem **csc**. Mono má pro tyto účely překladač **mcs**, který je již dnes na velmi dobré úrovni. Má navíc stejné přepínače jako originální csc, takže jeho použití je velmi pohodlné, povoleny jsou přepínače s pomlčkou i s lomítkem. Uživatele Linuxu může sice mást, že jeho výstupem jsou PE soubory (tj. běžné EXE soubory .NETu, jak je známe z Windows), avšak to je právě důsledkem vysoké úrovně kompatibility s originálním Microsoft .NET Frameworkem.

4.5 Podporované funkce

Fakt, že Mono je binárně kompatibilní s Microsoft .NET, umožňuje jednoduše vzít hotový program a spustit ho v Linuxu. Je jistě pěkné vidět, že Linux nyní umožňuje spouštět jistou podmnožinu EXE souborů z Windows, avšak kámen úrazu je v knihovně základních tříd. Mono totiž zatím nemá implementovány všechny třídy běžně používané ve Windows. Cílem však je, aby byly všechny problémy a nedodělky v knihovně základních tříd dořešeny. Podrobněji dále v textu.

4.6 Okenní aplikace

Vytváření klasických okenních aplikací je jedním z největších problémů klasického Visual C++. Ačkoliv MFC nabízí řadu tříd pro zjednodušení běžných operací, programování grafických uživatelských prostředí ve Visual C++ obvykle zabere víc práce, než psaní vlastní logiky aplikace. Microsoft .NET Framework umožňuje vytvářet okenní aplikace v C# mnohem jednodušeji, podobně jako tomu již dříve bylo ve Visual Basicu. Mono zde naráží na nemalý problém, že třídy Windows.Forms jsou ušity na míru Windows a jejich implementace mimo tento systém není snadná.
V současné verzi Mono je použití Windows.Forms obtížné, spíše až nemožné. První možnou variantou je instalace Wine (emulátoru Windows) ve speciálně upravené verzi pro Mono, jehož implementace Win32 API je pak použita pro realizaci Windows.Forms. Cílem tohoto

řešení je dosažení maximální kompatibility s Windows. Instalace Wine však přináší do systému závažné bezpečnostní díry, takže se tato varianta například pro použití na veřejných serverech absolutně nehodí.
Druhou možností je implementace grafického rozhraní pomocí nativních služeb Linuxu. Mono zatím obsahuje knihovnu Gtk#, která zpřístupňuje Gtk+. Gtk+ je jedna z řady knihoven Gnome, které jsou v Mono použity, a může umožnit rychlý vývoj okenních aplikací programátorům zvyklým na Gnome. Na druhou stranu programátoři zvyklí na Win32 API, MFC či Windows.Forms zřejmě do Gtk# tak snadno neproniknou. Možná je i implementace Windows.Forms pomocí Gtk#, avšak Windows.Forms jsou příliš propojené se systémem Windows, takže tato varianta by neumožnila spouštět žádné složitější aplikace.
Poměrně detailní analýzu možností, jak programovat grafická uživatelská rozhraní, je možno nalézt v [5]. Možností je popsáno mnoho, avšak každá z nich buď postrádá 100% kompatibilitu s Windows.Forms, nebo je nesnadno použitelná, nebo vytváří aplikace, jejichž vzhled neodpovídá nativním zvyklostem daného operačního systému. Zdá se, že vývoj přenositelných okenních aplikací skutečně je a i v budoucnu zřejmě bude tou největší slabinou projektu Mono.

4.7 Podporované jazyky

Jazyková nezávislost .NETu teoreticky znamená, že Mono „automaticky“ podporuje všechny .NETové jazyky. Praxe je však trochu složitější, neboť jednotlivé jazyky obvykle nabízejí kromě standardních .NET funkcí i nějaká vlastní rozšíření. Mono pak může tyto jazyky podporovat jen tehdy, když zná a podporuje všechna tato rozšíření. Dalším kamenem úrazu jsou služby, které v Microsoft .NETu umožňují za běhu programu „emitovat“ další kód v jednotlivých jazycích.

Microsoft Visual C# .NET je nejlépe přenositelný jazyk. Existuje v zásadě jen v této verzi, proto bývá obvykle jednoduše označován jako C#. Mono je na něm založeno, má vlastní překladač C#, a proto jeho podpora je vzhledem k ostatním jazykům jednoznačně nejlepší.
Microsoft Visual Basic .NET je v Mono také použitelný, ačkoliv implementace Microsoft.VisualBasic není zatím kompletní. Doporučuje se proto používat **option strict on** (vynutí explicitní deklarace proměnných), což zlepší přenositelnost VB.NET.
Microsoft Visual C++ .NET je zatím nepoužitelný. Mono podporuje jen řízený kód, zatímco C++ umožňuje na úrovni tříd libovolně mixovat řízený i neřízený kód. Navíc CRT knihovna je neřízená, takže například pouhé použití printf nebo fopen automaticky znamená, že C++ program nelze v Mono spustit. Použitelné jsou třídy knihovny STL a některé vestavěné (intrinsic) funkce, v [11] lze nalézt další informace. Přesto však možnost použití C++ v Mono je spíše v rovině teoretické. Zdá se, že Microsoft vůbec neuvažuje o vytvoření kompletního .NET portu C/C++.
Java není přímo podporovaná Microsoftem (ani nemůže, ze známých důvodů), avšak existuje celá řada vývojových nástrojů pro Javu, které by bylo škoda nevyužít. Microsoft nabízí jazyk J#, což je de facto Java.NET, avšak jde o starší verzi jazyka, což samozřejmě není ideální. Z dílen třetích stran je však možno získat překladače skutečné Javy pro .NET, což umožní spouštění Javových programů v .NETu i bez úprav zdrojového kódu. Mono i Microsoft .NET Framework tedy Javu skutečně podporují, i když nepřímo. Tímto způsobem navíc můžete Javu libovolně kombinovat s .NET jazyky na úrovni jednotlivých tříd.

4.8 WebForms a ASP.NET

Jednou ze zajímavých funkcí již podporovaných v současné verzi Mono, je ASP.NET a

WebForms. Nejjednodušší použití je zřejmě v podobě modulu pro Apache 2. Později byl měl být k dispozici i modul pro Apache 1.3 a pro testovací účely je k dispozici také testovací server **xsp**.
Celkově vypadá podpora ASP.NET velmi dobře a umožňuje použití ASP.NET stránek bez instalace systému nebo www serveru Microsoftu. To umožní i kombinaci s mnoha dalšími pluginy pro Apache.

4.9 Databáze a ADO.NET

Podpora databází v podobě ADO.NET není zatím v Mono úplně hotová, ale použitelná již je. Cílem je samozřejmě plná kompatibilita s Microsoft .NET Frameworkem, což společně s ASP.NET vytvoří mocný nástroj pro moderní dynamické www aplikace.

5. DotGNU Portable.NET

Další použitelnou implementací .NETu je DotGNU Portable.NET (viz [10]), pod kterým je podepsána právě skupina DotGNU. Tento produkt je také volně šiřitelný. Nesnaží se o dosažení 100% kompatibility s originálním Microsoft .NET Frameworkem, přesto například v oblasti Windows.Forms v současnosti nabízí lepší funkčnost než Mono.
Hlavní rozdíl mezi Mono a DotGNU je možno vidět přímo na www stránkách obou projektů. Zatímco projekt Mono považuje Microsoft .NET za kvalitní platformu a snaží se o poskytnutí plně funkční a standardům vyhovující implementace pro volné operační systémy (Linux aj.), DotGNU se primárně snaží o boj proti *zlému Microsoftu*. Negativistický přístup DotGNU je důvodem, proč jsme se my rozhodli použít Mono, někdo by však ze stejného důvodu mohl naopak chtít sáhnout po Portable.NET. Vývojáři Mono na svých www stránkách tvrdí, že jejich produkt je v pokročilejším stádiu vývoje, než produkt DotGNU. Vzhledem k rychlému vývoji obou produktů však nelze toto tvrzení jednoznačně potvrdit, ani vyvrátit.

6. Současnost a budoucnost Mono

Funkčnost Mono jsme testovali především na jednoduchých programech. Omezíte-li se na třídy, které současná verze implementuje, spolehlivost je vynikající. Vytknout lze jen třídy, které Mono neimplementuje vůbec - zejména Windows.Forms. Nefungují tedy klasické okenní aplikace.
Podle zveřejněných plánů bude příští verze Visual Studia a .NET Frameworku obsahovat řadu důležitých novinek, avšak nejdříve v roce 2005. Do té doby by mělo být Mono zkompletováno do verze podporující .NET Framework stávajících verzí 1.0 i 1.1, verze odpovídající .NETu 2.0 by měla přijít maximálně jeden rok po oficiálním uvedení .NET 2.0. Smutné je, že z těchto smělých plánů je opět vyjmuto Windows.Forms, které je sice slíbeno, ale s nejméně ročním zpožděním. Uvážíme-li však, že práce na projektu Mono začaly o několik let později, než práce na originálním .NETu, zdá se, že vývoj skutečně jde dostatečně rychle, abychom se kompletní verze (verzí) dočkali dříve, než Microsoft stávající specifikace .NETu opět inovuje.

Literatura:

1. Tom Archer: *Inside C#.* Microsoft Press, Redmond (USA). ISBN 0-7356-1288-9
2. Dan Brown: *.NET Uncovered.* Dunstan Thomas Consulting, 2003
3. Erik Brown: *Windows Forms Programming with C#.* Manning, 2002. ISBN 1-930110-28-6

4. Fergal Grimes: *Microsoft .NET for Programmers.* Manning, 2002. ISBN 1-930110-19-7
5. Miguel de Icaza: *GUI Toolkits Overview for Mono.* http://primates.ximian.com/~miguel/toolkits.html
6. Jesse Liberty: *Programming C#.* 2nd ed. O'Reilly, 2002. ISBN 0-596-00309-9
7. Jeff Prosise. *Programming Microsoft .NET (Core Reference).* Microsoft Press, Redmond, Washington (USA), 2002. ISBN 0-7356-1376-1
8. Michael Stiefer, Robert J. Oberg: Application Development Using C# and .NET. Prentice Hall, 2001. ISBN 0-13-093383-X
9. Thuan L. Thai, Hoang Lam: *.NET Framework Essentials.* 2nd ed. O'Reilly, 2002. ISBN 0-596-00302-1
10. DotGNU Project - *Portable.NET*. http://www.dotgnu.org/
11. *MSDN (Microsoft Developer Network) Library*. http://msdn.microsoft.com/
12. *Mono.* http://www.go-mono.com/