調査報告書

「.NET 互換環境のオープンソースについて」

2004 年 3 月

財団法人 情報処理相互運用技術協会

目　次

1. 調査目的

マイクロソフト社から Windows 向けの新しいソフトウェア環境として.NET 環境がリリースされている。.NET 環境は Java を研究して開発されており、仕組みの上からは Windows 上で動作するだけでなく、クロスプラットフォームで動作するアプリケーションを開発することができる。マイクロソフト社の開発ツールである Visual Studio はすでに.NET 環境に対応しており、Windows 向けアプリケーション開発者は.NET 環境に移行していくと考えられる。

以前のマイクロソフト社のソフトウェア開発環境では、Windows 自体の機能（Windows API）を直接利用するアプリケーションが作り出され、そのようなアプリケーションを他の OS 上で動作させることは継続的に試みられているものの、未だに実用のレベルに達しているとは言い難い。

一方、マイクロソフト社自身が、学術研究用という位置づけではあるが、Unix 系 OS である FreeBSD 向けの.NET 環境を公開している。.NET 環境の仕様のいくつかは公開されており、それに基づき互換性のある開発環境を作るオープンソースプロジェクトがすでに開始されている。Linux や他の OS 向けの.NET 環境が用意されれば、今後増加すると考えられる.NET 対応アプリケーションが、それらの OS 上で動作する可能性がある。

同様のことは Java ですでにほぼ実現されており、クロスプラットフォーム性が必要なアプリケーションは現時点では Java により開発されることが多くなっている。しかし、Windows 向けアプリケーション開発者の数は膨大であり、.NET 環境のクロスプラットフォーム性が実現されるとこの流れが変わることも考えられ、Linux 等の OS における.NET 環境の動向把握は非常に重要と考えられる。

そこで、オープンソース実装による.NET 互換環境の中で最も有力と考えられる mono プロジェクトを中心に、プロジェクト内容とその状況、商用利用・想定されるビジネスモデル、課題について記述する。

2. .NET 概要

Microsoft .NET 構想と、それを実現するための.NET プラットフォームについて概説する。また、.NET プラットフォームを構成する 3 要素である.NET Enterprise Server、.NET MyServices、.NET フレームワークについて記述する。.NET フレームワークは、次章でもう少し詳しく記述する。

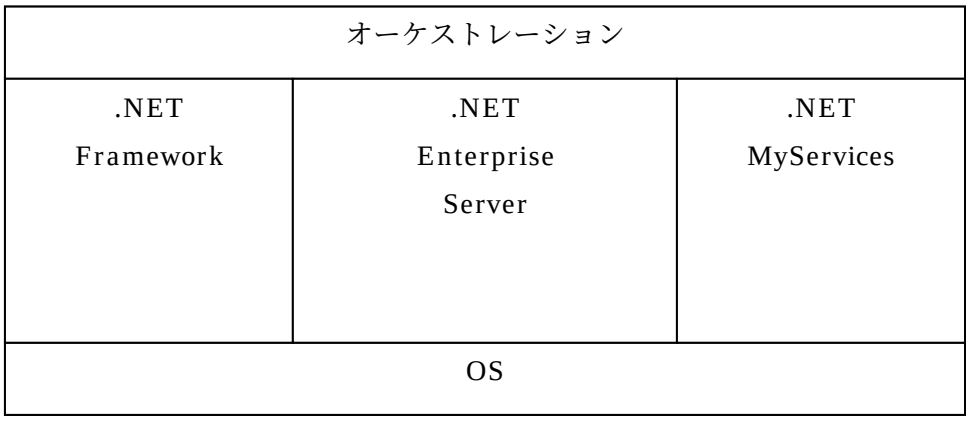
2.1. .NET 構想と.NET プラットフォーム

.NET は、次世代アプリケーション実行・開発プラットフォームのあるべき姿として、マイクロソフト社が提示した構想である。.NET の背景には、Web サービスの登場がある。

Web サービスが普及すれば、従来は閉じた独自の世界で提供されていたサービスが、インターネットを介して標準的な方法で提供され、さらにそれを組み合わせてより高度なサービスも提供されることが考えられる。その時、サービスを利用するシステムの開発者から見ると、個々のサービスを提供する OS を気にする必要はなく、Web サービスの規約にのみ着目すればよい。また、サービスの利用者も、サービスを提供する OS には全く関係がない。

そのような状況になった時、サービスを提供するシステムを容易に構築できる、あるいは高度なサービスを提供するシステムを構築できる基盤があれば、その基盤が普及する可能性は高いと考えられる。.NET は、そのような基盤の構想である。

この.NET 構想を実現するためのサービス指向のプラットフォームが、.NET プラットフォームである。.NET プラットフォームは、個々の Web サービスの提供、Web サービスが必要とする機能の実現、Web サービス間の連携等を行うものである。.NET プラットフォームはユーザ本位のサービスを提供するものであり、特定の OS やアーキテクチャに関係なく、ユーザは透過的にサービスを利用できるようになる。 .NET Platform は特定の実装手段に依存しない、インターネットを基盤としたオープンなプラットフォームである。.NET プラットフォームは、大まかには次の図のような構成となる。



.NET プラットフォーム

マイクロソフト社の構想では、.NET プラットフォームのベースとなる OS として Windows を用いている。ただし、技術的には Windows 以外の OS を用いることもできるはずである。

OS の上には、アプリケーション開発・実行のフレームワークである.NET Framework、様々なサービスのバックエンドで動作するサーバである.NET エンタープライズサーバ、個々のユーザの情報を管理し、それに基づくサービスを提供する.NET MyServices がある。

.NET プラットフォームでは、.NET Framework や.NET エンタープライズサーバ、インターネット上に公開される様々なサービスや既存のコンポーネントを統合して、１つにまとめた付加価値のあるサービスを提供できる。このような統合をオーケストレーションと呼ぶ。統合するサービスは、必要に応じて新しく開発することもあれば、既存のものを使う場合も考えられる。

2.2. .NET Enterprise Server

.NET Enterprise Server は、.NET の中核となるサーバーソフトの総称で、従来 BackOffice と呼ばれてきた製品群である。データーベースソフトの Microsoft SQL Server、メッセージングソフトの Microsoft Exchange Server などの製品から構成されている。

製品名	概要
Windows 2000 Server	Internet Information Server、Message Queue Server、Transaction Server などを内蔵。BizTalk Server や SQL Server と連携した、XML 文書や SOAP の HTTP などによる通信
BizTalk Server 2000	XML 文書を取引先とやりとりするためのフローの定義などを行う「BizTalk オーケストレーション・サービス」。XML 文書へのデータの変換、フォーマットのチェックなどを行う「BizTalk メッセージング・サービス」
SQL Server 2000	検索結果を XML 文書化する「FOR XML」。データにスキーマ定義を当てはめて、XML 文書化する「XML ビュー」。XML 文書をテーブルに直接格納する「OpenXML」など
Exchange 2000 Server	BizTalk Server と連携した、SMTP 経由でのメッセージの受信。Web Storage System のプロパティ情報
Commerce Server 2000	XML 形式によるカタログ情報の交換
Host Integration Server 2000	BizTalk Server と連携した「XML-to-Host 統合」
Internet Security & Acceleration Server 2000	データの送受信にセキュアな環境を提供
Application Center 2000	内部で、ロードバランス用の情報交換に XML 文書形式を採用

2.3. .NET My Services

.NET My Services (開発コード名: "Hailstorm") は、ユーザ指向の Web サービスセットである。開発者は Web サービスのデータ中心的な性質を利用して、Web サイト、Web サービス、アプリケーション、およびデバイスのためのユーザ指向のインターネット アプリケーションを構築することができる。

.NET My Services の認証は .NET Passport によって提供されます。この サービスは、Kerberos 分散セキュリティプロトコルを実装している。Kerberos は集中化されたセキュリティ情報をベースにしてクライアント要求を認証し、クライアントが特定のサービスにアクセスするために使用する一時的な暗号キーである「チケット」を配布する。

.NET My Services は、ユーザの具体的なニーズを中心として設計されています。これらのニーズを満たすために、以下のサービスが考えられた。

ただし、.NET が提唱された時分に比べると、現在では名前を聞くことは少なくなっている。

サービス名	概要
.NET Presence	このサービスは、ユーザがアラートを受け取る場所を示す、ユーザの電子的プレゼンス情報を含んでいる。たとえば、ユーザは自分の「プレゼンス」を Online/Offline/Busy/Away 、On-Phone/Off-Phone などに設定することができる。
.NET Location	このサービスは、ユーザの物理的なプレゼンス情報を含んでいます。ユーザは自分の「ロケーション」を "At Home" や "At Work" などに設定し、自分に連絡する方法を他人に知らせることができる。
.NET Services	ユーザが利用するサービスを追跡管理する。
.NET Alerts	アプリケーション、Web サイト、および Web サービスが、任意のデバイス上で、いつでもどこでも重要なイベントに関する「通知」を送信できるようにする。ユーザは、自ら選択して、これらの通知を受け取るための .NET Presence 設定を指定する。
.NET Calendar	ユーザのすべてのカレンダー情報を 1 つの場所に格納します。これにより、ユーザの仕事、

	家族、および個人に関する情報を、いつでもどのデバイスからでもアクセスできるようになる。ユーザは、カレンダーの一部またはすべてを他人と共有することができる。
.NET Contacts	ユーザは自分の個人的および仕事上の連絡先に関する情報を格納し、その情報にいつでもどこからでもアクセスできるようになる。また、この情報を共有することも可能である。
.NET Inbox	ユーザは任意のデバイスから電子メールにアクセスすることができます。ユーザーは電子メールを.NET My Services にサインインできる任意のコンピュータまたはデバイスからチェックすることができる。
.NET Documents	ユーザに個人的および仕事上のドキュメントの安全な格納場所を提供し、任意のコンピュータまたはデバイスからこれらのファイルにアクセスできるようにする。
.NET Wallet	ユーザはオンラインで商品を購入する際に使用する情報を格納しておくことができる。このようなデータが必要になるたびに、同じクレジットカードと配達先住所の情報を何度も再入力する必要がなくなる。
.NET Application Settings	ツールバー、アイコン、およびスクリーン セーバーなどのユーザ情報を格納し、ユーザがサインオンした任意のデバイスが、これらの設定を自動的に使用するようにする。
.NET Profile	ユーザは住所や誕生日などの個人情報を格納することができる。
.NET Favorite Web Sites	ユーザは、任意の場所にある任意のデバイスから、お気に入りの Web サイトへのリンクにアクセスすることができる。
.NET Lists	ユーザは任意の種類のリストを格納することができる。たとえば、買い物リスト、ウィッシュ リスト、TO-DO リストなどが考えられる。
.NET Categories	すべてのユーザサービスで使用できる、標準

	化されたカテゴリリスト。カテゴリは、ドキュメントをグループ化するために使用される。

2.4. .NET フレームワーク

.NET フレームワークは、従来の Windows ベースのアプリケーション開発のためのプラットフォームに対して、Web サービスをコアとして提供される次世代のアプリケーションプラットフォームである。開発者にとっては、従来の Windows アプリケーションで懸案となっていた開発生産性、メンテナンス性、セキュリティ、コードの安全性といった問題に対応した、新しい開発プラットフォームであるといえる。

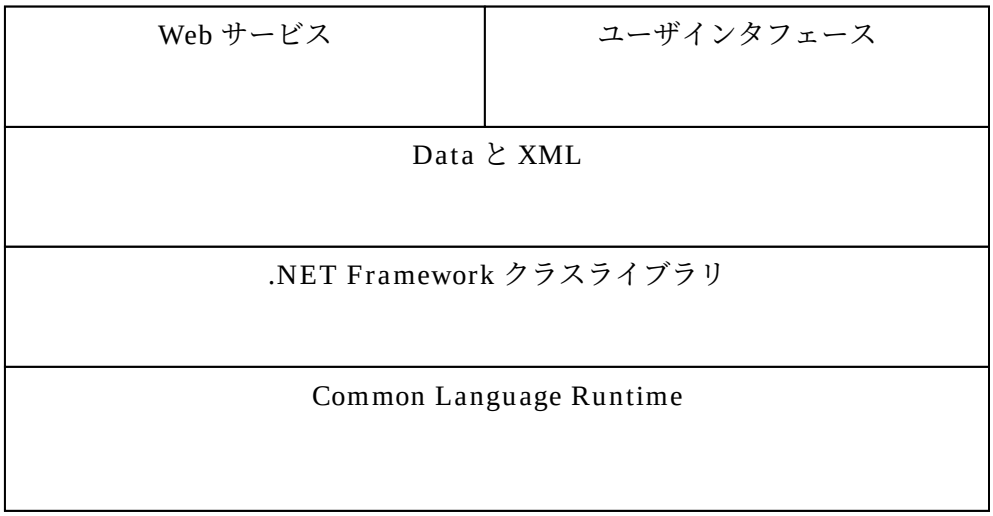
.NET フレームワークは、Web アプリケーション開発を効率よく行うためのベースとして用いることもできる。また、Web サービスを簡単に利用することができるので、企業内または企業間の異なるシステムを簡単に連携させることができる。また、サービス試行のコンポーネントを.NET フレームワーク上に開発することで、実装言語やプロトコルに依存しないソフトウェアサービスインフラストラクチャを構築できる。

マイクロソフト社は、Windows の将来のバージョンにおいて.NET ベースのアーキテクチャを採用することを表明しており、開発者はその時点において.NET ベースのアプリケーション開発に移行する必要がある。

マイクロソフト MSDN .NET Framework ホームページ
（http://msdn.microsoft.com/netframework/）

3. .NET Framework 概要

.NET Framework は、下図に示すように CLR（Common Language Runtime）、.NET Framework クラスライブラリ、Data と XML、Web サービス、ユーザインタフェースといった要素から構成される。



.NET Framework の構成

CLR はメモリ管理、セキュリティ管理等を行う実行環境である。CLR 上で実行するプログラムは、Visual Studio などにより MSIL と呼ばれる中間言語にコンパイルされる。このように中間言語にコンパイルされ、CLR により実行されるプログラムは、マネージドコードと呼ばれる。

CLR 上で実行するプログラムが必要とする基本的なクラスライブラリが、.NET Framework クラスライブラリである。

Data と XML は、.NET Framework でサービス間のデータのやりとりを行うものであり、XML 操作、メッセージング等の処理を行う。

ユーザインタフェースはユーザとのインタラクションのための処理と、本来の処理であるビジネスロジックの切り分けが難しい部分であるが、.NET Framework ではその分離を支援する機能を持つ。また、一つのビジネスロジックに対して複数のユーザインタフェースを用意するといったことも可能になる。ユーザインタフェースと Web サービスは、従来の ASP の拡張である ASP.NET 等により提供される。

3.1. Common Language Runtime (CLR)

CLR（Common Language Runtime）は、.NET Framework 上で動作するプログラムの実行環境である。.NET Framework 上で動作するプログラムは、Visual Studio などによりソースコードから MSIL 形式の中間言語にコンパイルされ、CLR によってその中間言語が実行される。その点においては、CLR は Java VM に似ている。しかし、Java VM ではプログラミング言語として Java だけが用いられるのに対して、CLR は名前が示すように、複数のプログラミング言語の共通実行環境である。利用できるプログラミング言語として、マイクロソフト社が処理系を提供する C#、Visual Basic、J#等の他、様々なプログラミング言語の処理系が用意されつつある。

CLR により実行されるソフトウェアは、マネージドコードと呼ばれる。このコードの生成・実行をサポートするために、CLR では CTS（Common Type System）と呼ばれる CLR 上で実行される言語間で共通に使用される型や、CLS（Common Language Specification）と呼ばれる言語間での相互運用性を確保するための仕様、ソフトウェアに関する情報を記述したメタデータの標準形式などを定めている。

これらにより、異なる言語を用いて一つのソフトウェアを作成することも容易に行える。例えば C#によりベースとなるクラスを作成し、Visual Basic を用いてそのクラスを継承し、カスタマイズしたクラスを実装することも可能である。またこのようなソフトウェアをデバッグする場合、Visual Studio では Visual Basic のソースコードからクラス階層を上位にたどっていき、C#のソースコードにたどり着く、といったことも行える。

CLR は次図のように、クラスローダ、ガベージコレクタ、メモリ管理、CLS や CTS、セキュリティ、JIT コンパイラによって構成される。

CLR の構成

クラスローダは、CLS 仕様に基づくプログラムをメモリ上にロードする。実行時には必要に応じて JIT コンパイラが動作し、中間言語形式から CPU が直接実行可能なネイティブコードへの変換が行われる。また実行時には、ガベージコレクタ、メモリ管理、セキュリティといった各機能が、プログラムの安全で堅牢な実行を支援する。

3.2. .NET Framework クラスライブラリ

.NET Framework クラスライブラリは、CLR 上で動作するプログラムに提供される共通機能である。.NET Framework クラスライブラリは膨大な機能からなっており、その機能は名前空間によって分類されている。ここでは、厳密な意味でのクラスとインタフェースをまとめてクラスと呼ぶことにする。以下に、.NET Framework クラスライブラリの主要な名前空間を示す。

.NET Framework クラスライブラリの主要な名前空間

System		
	Collections	
		Specialized
	ComponentModel	
		Design
	Configuration	
		Assemblies
		Install
	Data	
		OleDb
		SqlClient
	Diagnostics	
	DirectoryServices	
	Drawing	
		Drawing2D
		Imaging
		Printing
		Text
	EnterpriseServices	
	Globalization	
	IO	
	Management	
	Messaging	
	Net	
		Sockets
	Reflection	
		Emit
	Resources	

- Runtime
  - InteropServices
  - Remoting
  - Serialization
- Security
  - Cryptography
    - X509Certificates
    - Xml
  - Permissions
  - Policy
  - Principal
- ServiceProcess
- Text
  - RegularExpressions
- Threading
- Timers
- Web
  - Services
  - UI
- Windows
  - Forms
- Xml
  - Schema
  - Serialization
  - XPath
  - Xsl

ルート名前空間である System の直下に定義されているクラスには、まず CTS によって定義される型である Int32, Char, Boolean, Array, Object などがある。それ以外にも、Console, Environment, GC, Math などの重要なクラスがある。

System 以下の名前空間のうち、主要なものを以下に簡単に説明する。

System.Collections：各種コレクションを作成、操作するためのクラスを含む名前空間である。コレクションには、配列、キュー、スタックなどがある。

System.ComponentModel：様々なコンポーネントを作るためのクラスを含む名前空間である。.NET Framework クラスライブラリの多くの基盤である Component クラスもこの名前空間に含まれる。

System.Configuration：アプリケーションの構成情報にアクセスするクラスを含む名前空間である。また、カスタムインストーラを作成する名前空間であるSystem.Configuration.Installも、ここに属する。

System.Data：データアクセスに関連するクラスを含む名前空間であり、ADO.NETを実装するクラスを含んでいる。この名前空間に含まれるクラスには、.NET Frameworkで動作するアプリケーションがデータにアクセスするための標準的な手段があり、各種SQLデータベースへのアクセス、XMLデータへのアクセスなどを透過的に実行することができる。

System.Diagnostics：.NET Frameworkで動作するアプリケーションのデバッグを支援するクラスを含む名前空間である。

System.DirectoryServices：ActiveDirectoryなどのディレクトリサービスにアクセスするためのクラスを含む名前空間である。

System.Drawing：描画インタフェースであるGDI+の機能を利用するためのクラスを含む名前空間である。イメージデータ、フォント、印刷などの機能を提供するクラスが含まれている。

System.EnterpriseServices：分散トランザクション、ロールによる認証などの機能を提供するクラスが含まれる名前空間である。

System.Globalization：ソフトウェアの国際化に必要な機能を提供するクラスが含まれる名前空間である。

System.IO：ファイル、ディレクトリ、ストリームなどのアクセスを行うクラスが含まれる名前空間である。

System.Management：WMI（Windows Management Instrumentation）機能にアクセスするためのクラスが含まれる名前空間である。

System.Messaging：メッセージキューサービスにアクセスするためのクラスが含まれる名前空間である。

System.Net：Socketや上位プロトコルを利用するためのクラスが含まれる名前空間である。

System.Reflection：プログラムのメタデータを調べるためのクラスが含まれる名前空間である。

System.Resources：メッセージ、アイコンなどのリソースを操作するためのクラスが含まれる名前空間である。

System.Runtime：以下のような名前空間を含む、親名前空間である。

System.Runtime.InteropServices：CLR上で実行するプログラムと、そうでないプログラムの間で相互運用を可能にするためのクラスを含む名前空間である。

System.Runtime.Remoting：リモートオブジェクトへのアクセスを行うためのクラスを含む名前空間である。

System.Runtime.Serialization：オブジェクトのシリアライゼーションを行うためのクラスを含む名前空間である。

System.Security：セキュリティ関連機能を提供するクラスを含む名前空間である。

System.ServiceProcess：Windows におけるサービスと同様の動作を行うプログラムを作成するためのクラスを含む名前空間である。

System.Text：文字列を操作するためのクラスを含む名前空間である。またこの下位には、正規表現のためのクラスを含む名前空間である System.Text.RegularExpressions も含まれる。

System.Threading：スレッド操作を行うためのクラスを含む名前空間である。

System.Timers：タイマ操作やイベント処理を行うためのクラスを含む名前空間である。

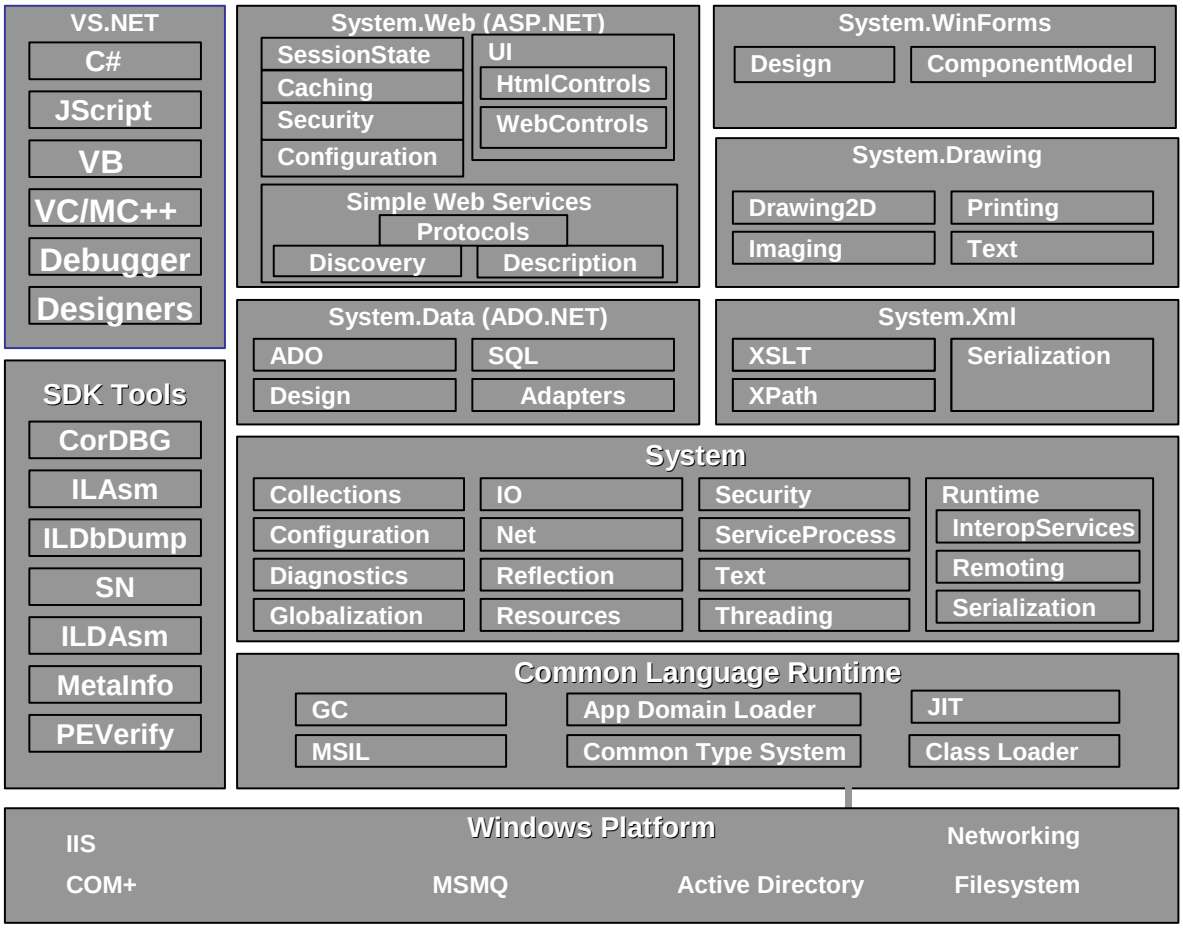
System.Web：Web アプリケーションや Web サービスを実装するために使用するクラスを含む名前空間であり、ASP.NET の実装に必要とされる。この下位の名前空間には、ブラウザを用いたユーザインタフェースを実現するためのクラスが含まれる System.Web.UI 名前空間や、Web サービスの実装に必要なクラスが含まれる System.Web.Services 名前空間が含まれる。

System.Windows.Forms：従来からの Windows GUI やその拡張された機能を用いる GUI を作成するためのクラスを含む名前空間である。

System.Xml：XML データを操作するためのクラスを含む名前空間である。

以上で説明したクラスライブラリを含む、.NET Framework 全体の概要（一部、開発環境等を含む）を次の図に示す。

.NET Framework 全体図
（http://research.microsoft.com/programs/europe/rotor/workshop.aspx）

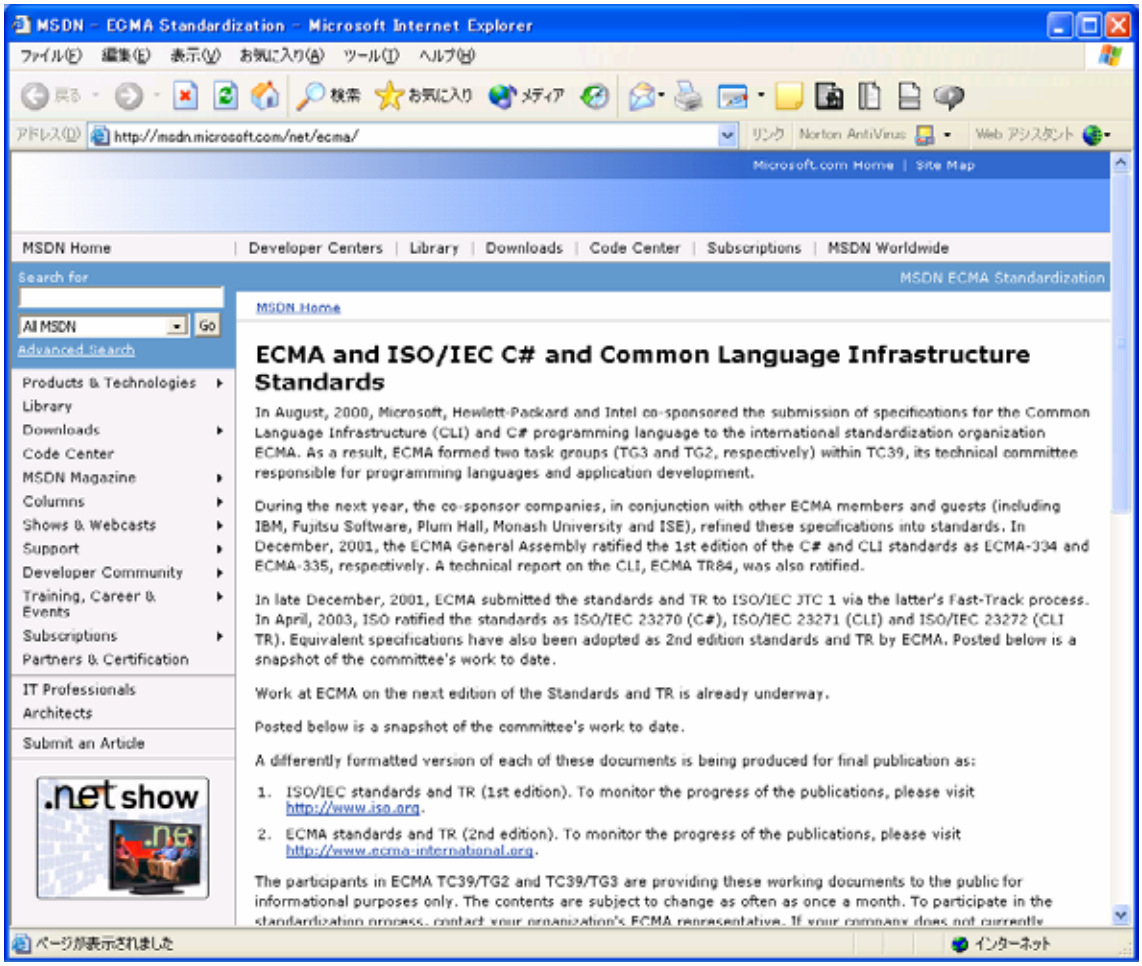


4. .NET の公開された仕様について

.NET に関する仕様の一部は標準化団体 ECMA, ISO に提出され、すでに標準になっている。これについて簡単にまとめる。

マイクロソフト社による ECMA 標準化に関する情報のページ

http://msdn.microsoft.com/net/ecma/



4.1. 標準化された仕様

プログラミング言語 C#、CLR のサブセットである CLI（Common Language Infrastructure）の仕様は、2000 年 8 月に国際標準化団体 ECMA に提出された。これを受け、プログラミング言語とアプリケーション開発に関する技術委員会である TC39 に、タスクグループ TG2、TG3 がそれぞれ作られた。

2001 年 12 月に、C#と CLI の仕様の初版が、それぞれ ECMA-334, ECMA-335 標準として批准された。また、CLI のテクニカルレポート ECMA TR84 も批准された。

ECMA はこれらを ISO/IEC JTC 1 に提出し、2003 年 4 月に ISO は C#を ISO/IEC 23270、CLI を ISO/IEC 23271、CLI TR を ISO/IEC 23272 として批准した。また、ここで行われた変更は、ECMA 標準第 2 版として反映された。

C#

- http://www.ecma-international.org/publications/standards/Ecma-334.htm
- http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=36768&ICS1=35&ICS2=60&ICS3=

CLI

- http://www.ecma-international.org/publications/standards/Ecma-335.htm
- http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=36769&ICS1=35&ICS2=60&ICS3=

CLI TR

- http://www.ecma-international.org/publications/techreports/E-TR-084.htm
- http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=36770&ICS1=35&ICS2=60&ICS3=

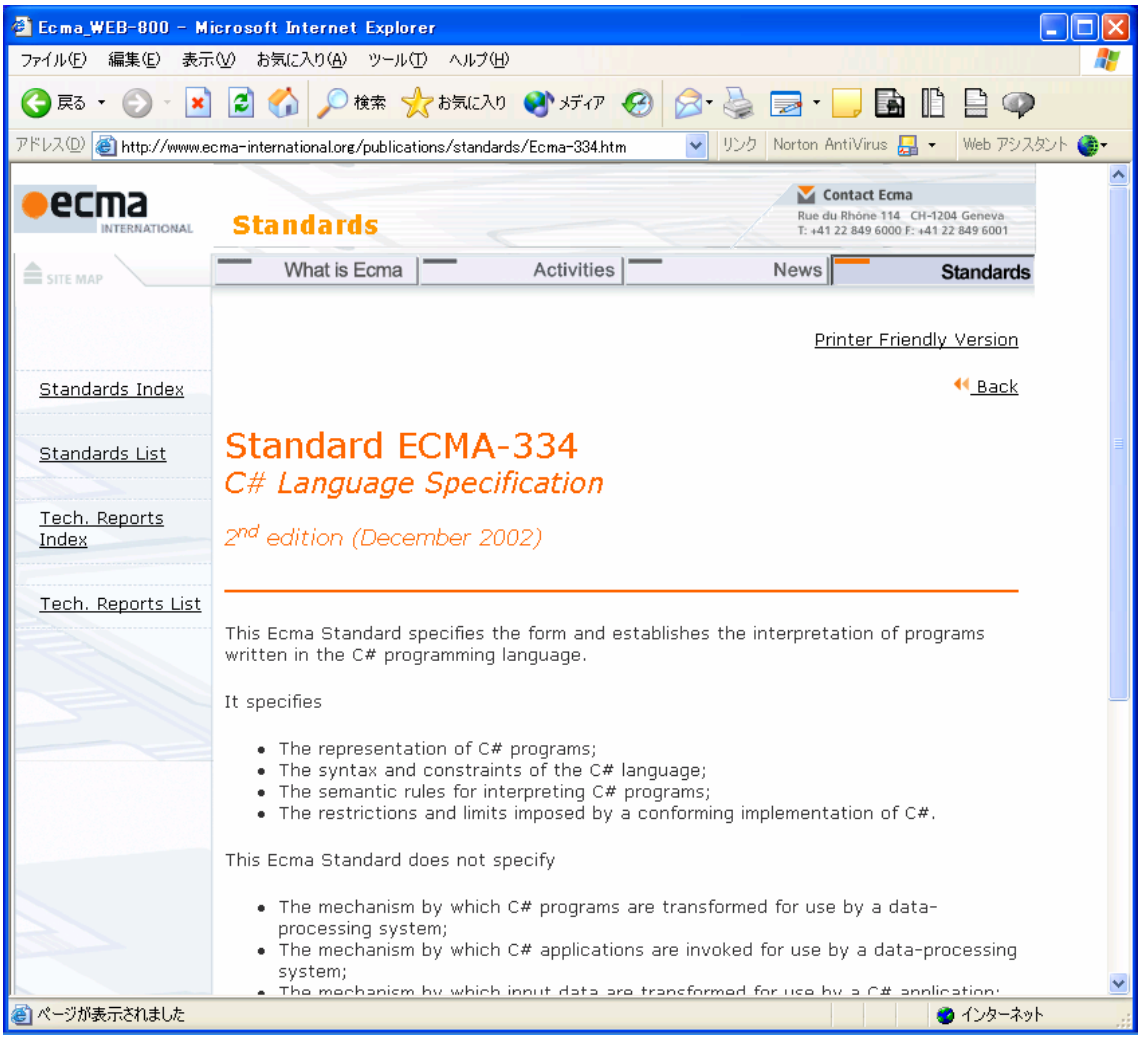
4.1.1. C#

ECMA-334 では、C#プログラミング言語で書かれたプログラムの形式とその解釈を記述している。この仕様では、以下を規定している。

- C#プログラムの表現
- C#言語の文法と制約
- C#プログラムの解釈に関する意味論的規則
- C#の準拠した実装に課せられる制限

逆に、この仕様では以下を規定していない。

- C#プログラムが変換されるメカニズム
- C#アプリケーションが呼び出されるメカニズム
- C#アプリケーションへの入出力データが与えられるメカニズム
- 特定の処理系の容量を超えるプログラムやデータのサイズや複雑さ
- 準拠した実装をサポートできる極小の処理系の要件



4.1.2. CLI

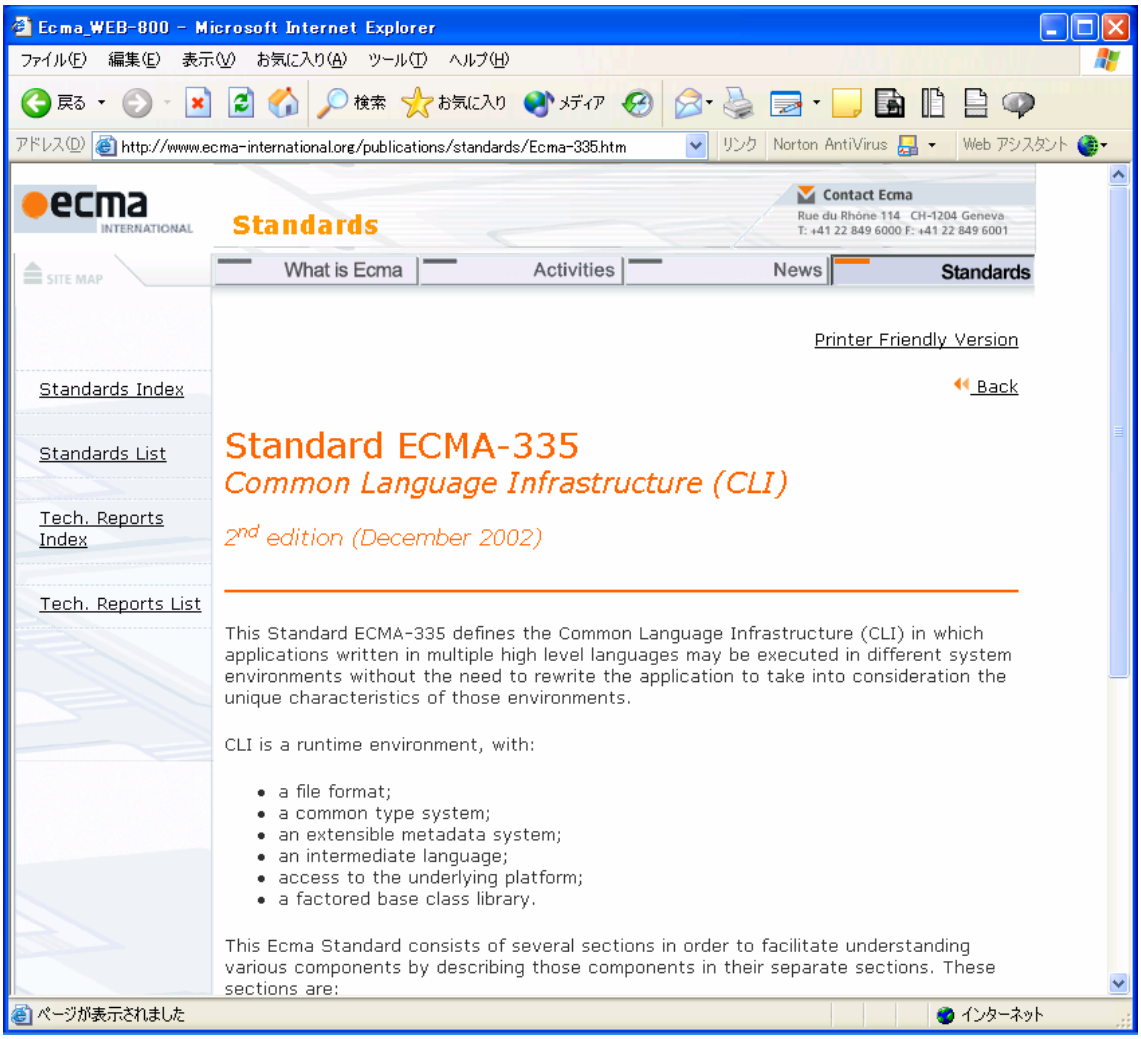
ECMA-335 では、複数の高水準言語によって書かれたアプリケーションを、異なるシステム環境上で環境の違いを考慮した書き直しを行わずに実行できるインフラである CLI を定義している。CLI は以下の要素を持つ実行環境である。

- 単一ファイルフォーマット
- 共通型システム
- 拡張可能なメタデータシステム

- 単一中間言語
- 基底プラットフォームへのアクセス
- ベースクラスライブラリ

この ECMA 標準は、様々な要素の理解を容易にするために、以下のように要素を独立したセクションに分けて記述している。

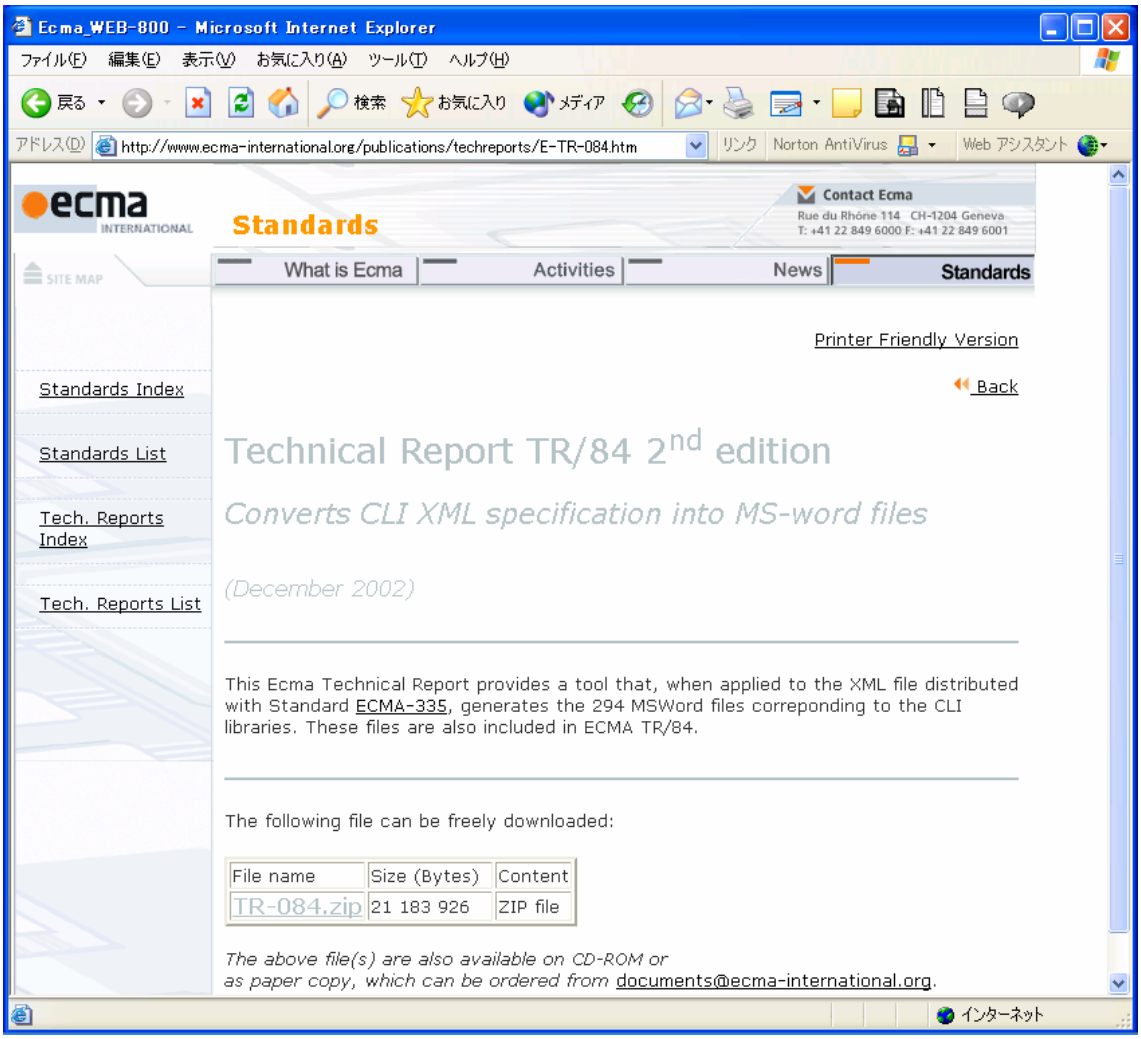
1. アーキテクチャ
2. メタデータ定義と意味
3. 中間言語命令セット
4. プロファイルとライブラリ
5. 付録



4.1.3. CLI TR

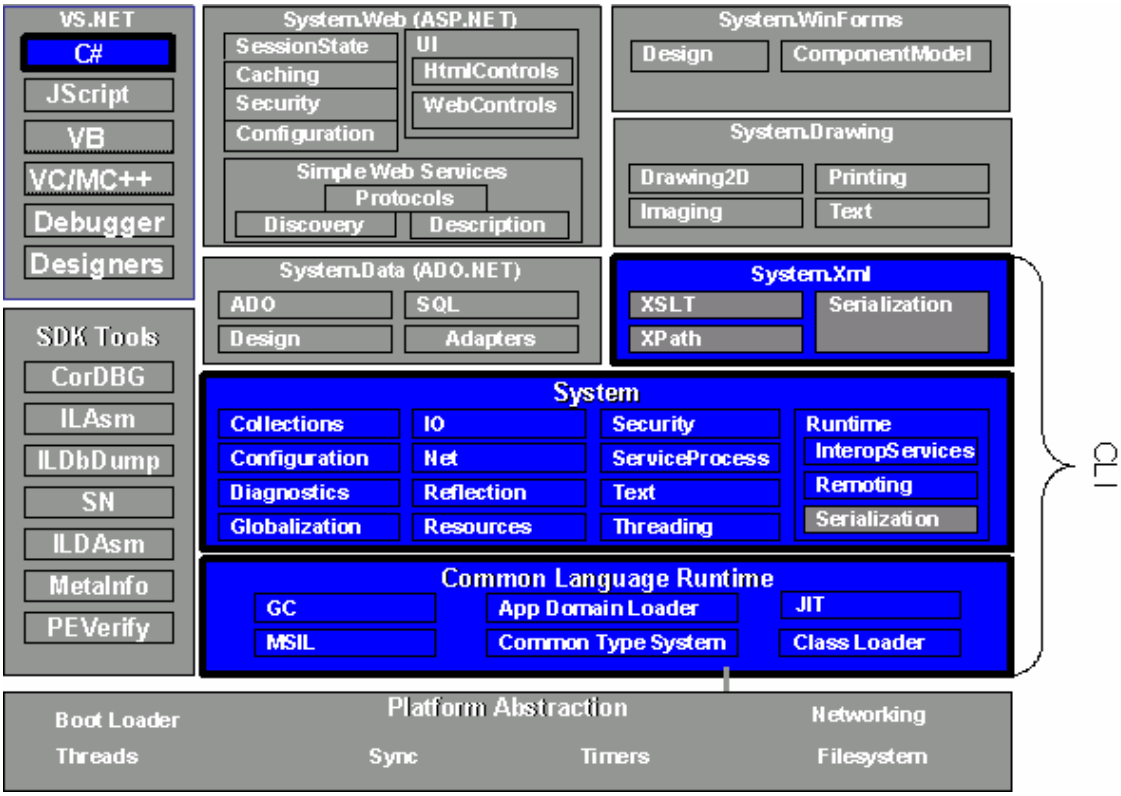
ECMA-335 には、クラスライブラリについて機械可読な XML 形式の記述が含まれる。

この ECMA TR84 には、それを変換するソースコードと、変換結果のドキュメントが含まれている。



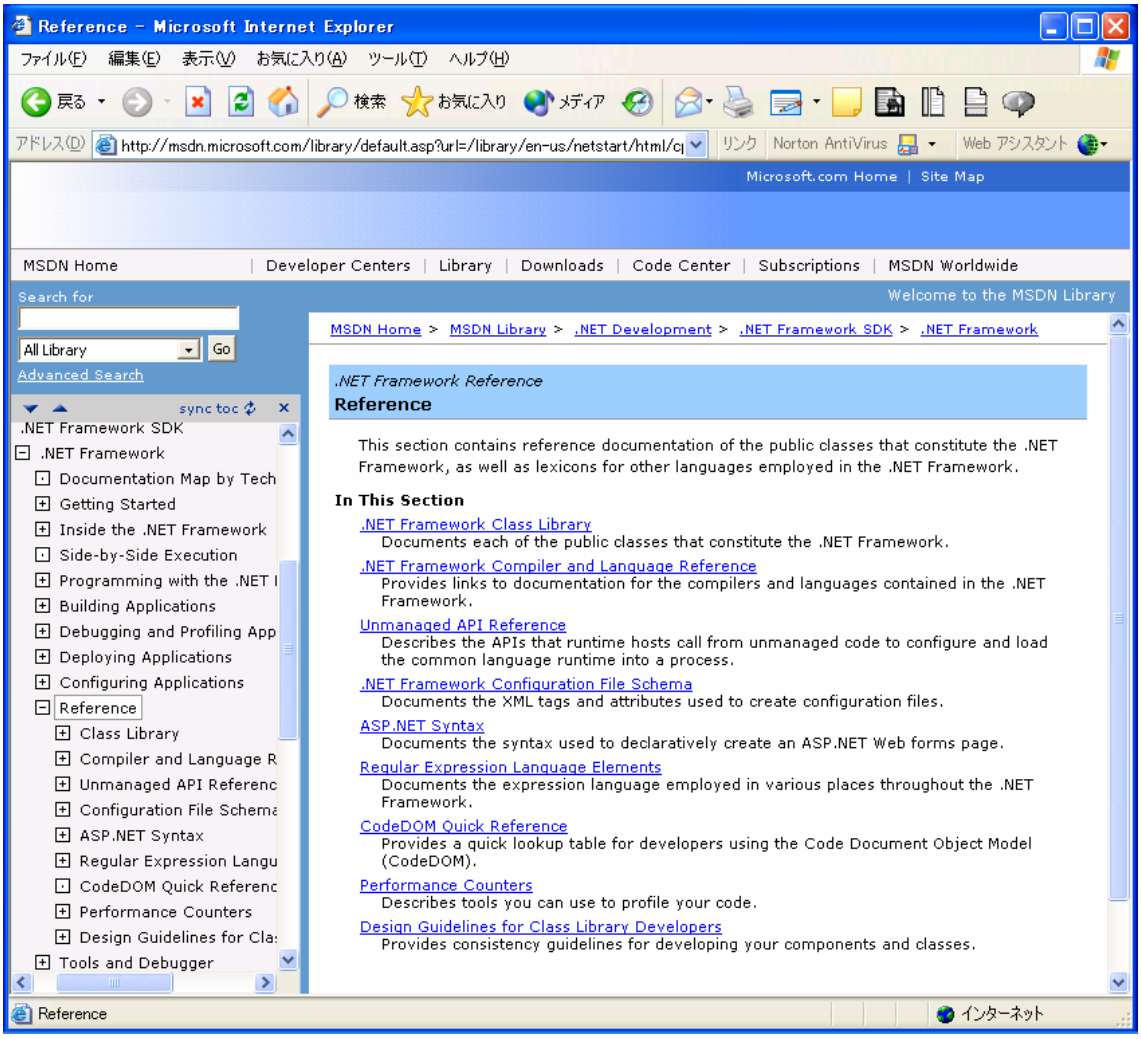
.NET Framework 全体の概略のうち、ECMA 標準になっているものを、次図に太線で示す。

ECMA 標準で記述された範囲
（http://research.microsoft.com/programs/europe/rotor/workshop.aspx）



4.2. 公開されているその他の仕様

マイクロソフト社の Web サイトでは、MSDN Library として様々なドキュメントが公開されている（http://msdn.microsoft.com/library/default.asp）。.NET Framework に関するドキュメントは、.NET Framework Reference として公開されている（http://msdn.microsoft.com/library/default.asp?url=/library/en-us/netstart/html/cpframeworkref_start.asp）。



5. .NET 互換オープンソースプロジェクトについて

.NET の一部の仕様が ECMA 標準になったことを受け、その仕様に基づくオープンソースプロジェクトである mono と DotGNU が開始された。また、マイクロソフト社自体もオープンソースではなく Shared Source ではあるが、FreeBSD と MacOS X 上で動作する Rotor プロジェクトの成果を公開している。

また、.NET そのものではないが、.NET 開発環境の開発がオープンソースプロジェクトで開始されている。

この章では、このようなオープンソースプロジェクトについて解説する。

5.1. mono プロジェクト

mono プロジェクトは 2001 年から開始された.NET フレームワークの Linux, Unix 版を開発するオープンソースプロジェクトであり、Ximian 社によりサポートされている。このプロジェクトは、Linux, Unix 開発者がクロスプラットフォームの.NET アプリケーションを開発できるようにすることを指している。現在では、Linux, Unix だけでなく、様々な OS 上で動作するようになってきている。

mono プロジェクトホームページ
http://www.go-mono.com/

すでに、mono プロジェクトの成果を用いてソフトウェアの開発を行うための解説書も出版されている。

- Mono Kick Start (KICK START) Hans-Jurgen Schonig, Ewald Geschwinde (2003/09/15) Macmillan Computer Pub
- Developing on Linux With C# and .Net: The Mono Project Daniel Solin (2003/10/15) Springer-Verlag New York Inc

5.1.1. mono プロジェクト開発物の構成

当初の mono プロジェクトによる開発物は、大きく分けて次の 3 つからなる。

- C#コンパイラ（ECMA-334）
- CLI 準拠のクラスライブラリ（ECMA-335）
- CLI 準拠のインタープリタと JIT（Just In Time コンパイラ）（ECMA-335）

また、ECMA-335 で規定されているクラスライブラリは.NET フレームワークで実装

されるクラスライブラリに比べてかなり小規模であるため、ECMA-335 で規定されていないがマイクロソフト社により実装されているクラスライブラリも、すべてではないが mono プロジェクトによる開発対象になっている。

さらに、以下の機能の実装が追加されている。

- ASP.NET
- ADO.NET
- VB コンパイラ

monoプロジェクトはECMA標準に基づいたソフトウェアの開発からスタートしたが、現在では単に.NET フレームワークの実装にとどまらず、その他のコンポーネントも含むようになってきている。そのようなコンポーネントには、以下のものがある。これらは、mono 開発チームにより開発されたものもあれば、他のオープンソースプロジェクトの成果が取り入れられたものもある。

- Remoting.CORBA
- Ginzu
- Gtk#
- #ZipLib
- GlGen
- Mono.LDAP
- Mono.Data
- Mono.Cairo
- Mono.Posix
- Mono.Http

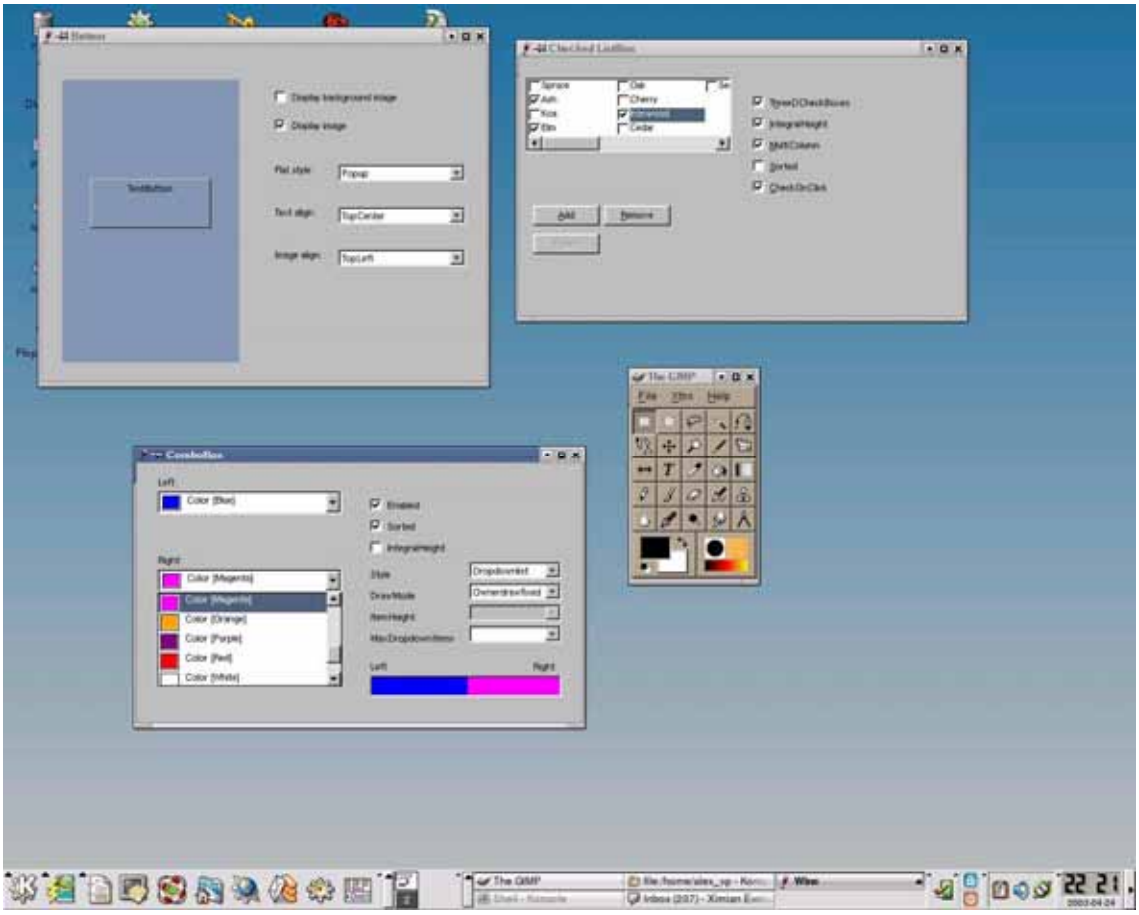
5.1.2. mono プロジェクトにおける GUI クラスライブラリ

mono プロジェクトは、全体としてはマイクロソフト.NET との互換性を重視しているが、GUI クラスライブラリに関しては異なるアプローチをとっている。

mono プロジェクトでは、GNOME ツールキットである Gtk+をベースに、その C#言語バインディングである Gtk#の開発を先行して行っている。これを用いて GUI を開発した場合、.NET Framework とは互換性のない Gtk+を用いて画面を作成することになる。Gtk+自体は Windows にも移植されているので、クロスプラットフォーム性が必要なプロジェクトでは Gtk+を使うことができれば問題は少ないと思われる。しかしマイクロソフト.NET で先行して開発を行い、それを mono プロジェクト側に移植するような場合には不都合である。

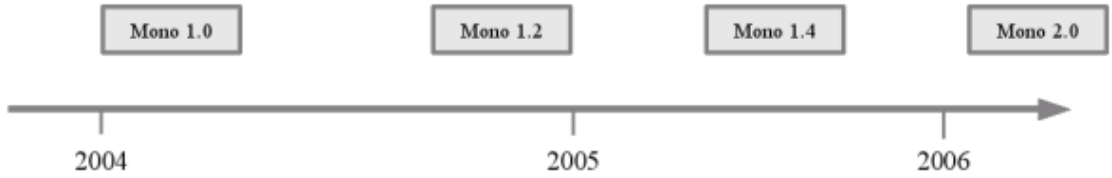
Gtk#の開発からは遅れているものの、並行して.NET の標準的な GUI クラスライブラリである System.Windows.Forms の開発も行われている。本報告書の執筆時点では、そ

の進捗状況は 69%になっている。System.Windows.Forms について公開されているスクリーンショットを以下に示す。



5.1.3. プロジェクトロードマップ

mono プロジェクトは 2001 年に開始され、2004 年第二四半期にバージョン 1.0 がリリースされる予定である。その後、以下のようなタイムスケジュールでリリースが行われる予定である。



個々のバージョンの概要は、以下の通りである。

方針：

mono を構成するコンポーネントの完成度は、その開発に投入した工数や使用頻度により大きく異なる。例えば仮想マシンや C#コンパイラは非常に完成度が高いと言えるが、Windows.Forms や VB.NET は開発中の状態である。

mono のリリースでは、完成度の高いコンポーネントをバージョン 1.0 としてリリースし、その後のリリースで順次機能を付け加えていく。

mono 1.0：

mono 1.0 は、2004 年第二四半期をリリース目標としている。mono 1.0 のリリースには、以下のコンポーネントが含まれる。

- C#コンパイラ
- 仮想マシンと JIT コンパイラ、プリコンパイラ
- IL（中間言語）アセンブラ、逆アセンブラ
- 開発ツール、セキュリティツール
- Core ライブラリ：mscorlib, System, System.XML
- System.Data ライブラリと mono データベースプロバイダ
- System.Web ライブラリ：Web アプリケーションプラットフォームと Apache 統合モジュール
- System.Web.Services ライブラリ：Web サービスのクライアント・サーバサポート
- System.Drawing ライブラリ
- System.DirectoryServices ライブラリ
- JIT コンパイラ：x86 と PPC アーキテクチャ（その他のアーキテクチャはインタープリタを用意する）
- ECMA プロファイル:様々な ECMA プロファイルの実装として mono をビルド可能にするために必要なビルドオプション
- IKVM による Java との統合
- 実行環境のための埋め込みインタフェース

パッケージ化：

- mono：.NET API 1.1 に基づき、上記機能を実装したもの。
- mono-1.0-compat：.NET API 1.0 ライブラリを含むパッケージであり、.NET 1.0 アプリケーションを実行する場合のための互換パッケージ。
- mono-unstable：開発者の便宜を図るため、開発中だがその時点では未サポートの機能のスナップショットを含むパッケージである。これには、C#コンパイラの Generic 版を含む。
- mono-ecma：ECMA コンポーネントだけを含むパッケージ。

mono 1.2：

mono 1.0 以後については、2004 年にリリースされるマイクロソフト社の Whidbey 製品を視野に入れる必要がある。Whidbey により.NET フレームワークに導入される新機能には以下のものがある。

- Generic 型：これにより、コンパイラ、実行環境、クラスライブラリの変更が必要になる。
- ASP.NET 2：Web アプリケーション開発を簡単にするための以下のような多くのツールが用意される：マスターページ、共通の操作、パーソナライゼーションとテーマのための新コントロール。
- Remoting:新しいセキュリティチャネルとバージョン耐性のあるリモーティング。
- XML：XQuery と、XPath に基づく XmlDocument システムの改良である XPathDocument が導入される。
- ネットワーク：FTP クライアント、SSL ストリーム
- コンソールとシリアルポート：コンソール端末の入出力とシリアルポートは同様の扱いが可能になる。
- Windows.Forms：レイアウトコンテナと、様々な新コントロールが導入される。
- オブジェクト空間：より単純なデータベースアクセスのための API。

バージョン 1.0 リリースには安定版が間に合わない機能について、mono チームは並行して開発を進めている。これが、バージョン 1.2 の基礎になる。mono バージョン 1.2 の焦点は、.NET フレームワーク 1.2 のコア API の行方を追うことであるが、フレームワークのサブセットの提供になる見込みである。
mono バージョン 1.2 は概ね、mono バージョン 1.0 では十分には安定していなかったコンポーネントのうち 1.2 で完成度の高まったものからなるが、さらに Whidbey から新機能をいくつか導入する予定である。mono バージョン 1.0 のコンポーネントに加え、バージョン 1.2 では以下を追加する予定である。

- Generic 型のサポート：C#コンパイラ、実行系、コアクラスライブラリ
- ASP.NET2.0 での改善
- Whidbey における Remoting の改善
- コンソールとシリアルポートのサポート
- 新しいコンパイラ：VB.NET と JScript のサポート
- WSE1/WSE2（.NET Framework Web Service Enhancements の version1 と 2）の実装

- .NET 1.0API レベルでの System.Windows.Forms。.NET 1.2 レベルは、安定ではない追加機能として提供する

このリリースでは、デフォルトで.NET 1.2 API を提供するが、1.0 と 1.1 の互換ライブラリも mono-compat パッケージの形式で提供する。mono の安定ではないコンポーネントは、mono-unstable パッケージの形式で提供し、そのライブラリはバージョン 1.2 のリリースではサポートされない。
リリースの目標は、2004 年の第 4 四半期である。

mono 1.4：
このバージョンは、mono 1.2 のリフレッシュアップデートであり、欠けていたコンポーネントの追加や不完全な箇所の改訂を行うものである。System.Xml、ASP.NET、Windows.Forms が.NET 1.2 API に合うように更新する。
リリース目標は、2005 年第 2 四半期である。

連携するプロジェクト：
デバッガやドキュメンテーションブラウザ、IKVM による Java との統合、Gtk#といったその他のプロジェクトは、（mono プロジェクトとの連携を行うが、mono プロジェクトに含まれないので）それぞれのプロジェクトのスケジュールに基づいて作業が進められる。

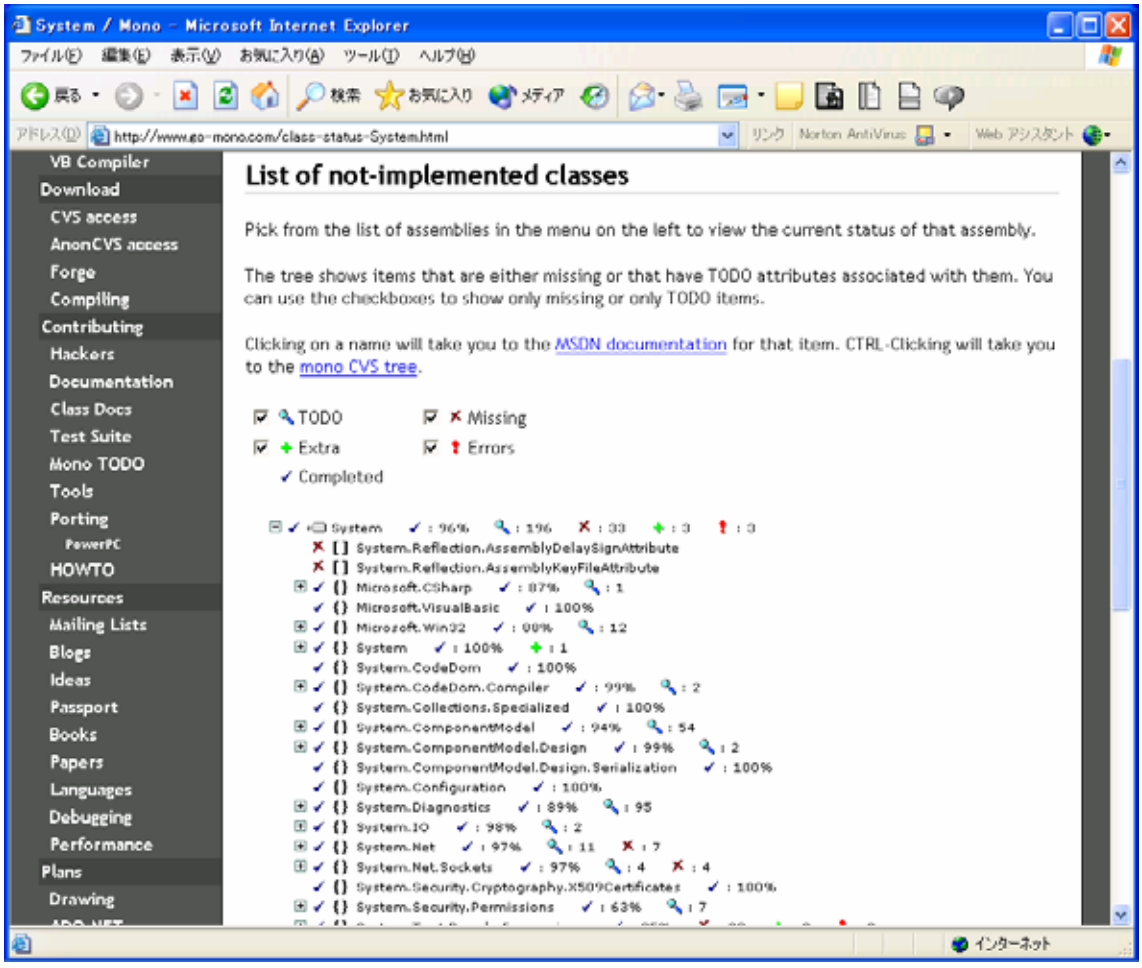
スケジュールされていないアクティビティ：
mono に欠けているコンポーネントは CAS（Code Access Security）である。この機能は、現時点での mono ではアプリケーションは完全に信頼されたアプリケーションとして実行されるので、必要とされない。この機能を実現するには多くの作業が必要となるが、現時点ではその実装を予定していない。

mono と WinFX：
WinFX は、.NET を新しい OS である Longhorn で作り上げるための新しいライブラリセットの名前であり、既存のクラスライブラリにこの OS で新たに利用可能になる新機能を加えたものである。
WinFXでは、ストレージファシリティ（WinFS）、新しい多目的通信スタック（Indigo）、新しい GUI プログラミングシステム（Avalon）のような機能が追加される。しかし現時点ではこれらの機能の詳細が不明であり、どの時点で mono に追加されるかは不明である。

現時点では、mono2.0 の詳細は明確化されていない。

5.1.4. mono プロジェクトのステータス

本報告書の執筆時点（2004 年 3 月）では、mono バージョン 1.0 はまだリリースされておらず、バージョン 0.30.1 である。mono プロジェクトでは、クラスライブラリの実装状況について随時状態を公表している。本報告書の執筆時点で基本的なクラスライブラリである System 直下のいくつかのクラスライブラリについてみると、完了しているものは 96%になっている。



5.2. DotGNU プロジェクト

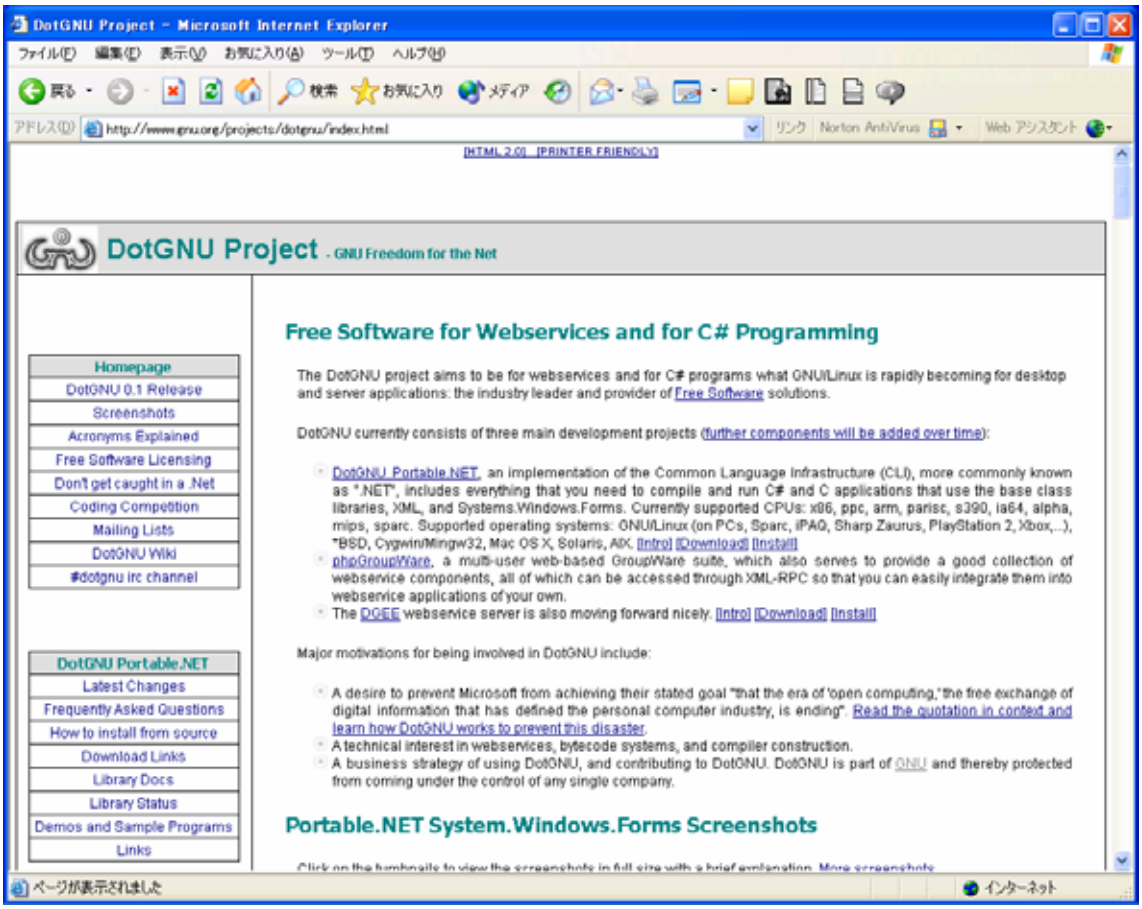
DotGNU プロジェクトは、Web サービスと C#プログラミングのためのフリーソフトウェアの開発を主たる目的として、Free Software Foundation により行われているプロジェクトである。

DotGNU プロジェクトは、現時点では次の 3 つの開発プロジェクトからなる。

- DotGNU Portable.NET：CLI の実装であり、C#と C 言語で書かれ、ベースクラスライブラリを用いるアプリケーションのコンパイルと実行に必要なものすべてを含む。現時点でサポートしている CPU は、x86, ppc, arm, parisc, s390, ia64, alpha, mips, sparc である。サポートしている OS は、Linux, *BSD, Cygwin/Mingw32, Mac OS X, Solaris, AIX である。
- phpGroupWare：マルチユーザ対応の Web ベースのグループウェアであり、また Web サービスアプリケーションからアクセス可能な Web サービスコンポーネントのコレクションでもある。
- DGEE：Web サービスサーバである。

DotGNU プロジェクトホームページ

http://www.gnu.org/projects/dotgnu/index.html



これらのサブプロジェクトのうち、.NET 開発環境に関連の強い DotGNU Portable.NET プロジェクトについて見てみる。

5.2.1. DotGNU Portable.NET

DotGNU Portable.NET の目標は、CLI 上で動作するアプリケーションのコンパイルと実行を行うフリーソフトウェアツールの構築である。最初のターゲットプラットフォームは Linux である。その他に動作するプラットフォームとして Windows, NetBSD, FreeBSD, Solaris, MacOS X などがあり、様々な CPU でも動作する。

DotGNU Portable.NET は、ECMA-334 と ECMA-335 の C#と CLI 仕様との互換性を目指すとともに、マイクロソフト社の商用 CLI 実装との互換性もめざしている。また、DotGNU Portable.NET とマイクロソフト.NET プラットフォームの両方で動作するアプリケーションを容易に開発できるようにすることを目指している。

さらに、マイクロソフト.NET プラットフォーム向けに移植性を考慮せずに開発されたアプリケーションが、DotGNU Portable.NET 上でも動作するようにすることもねらっている。

5.2.2. DotGNU **における** System.Windows.Forms

DotGNU Portable.NET では、System.Windows.Forms の実装に特徴がある。DotGNU Portable.NET では、Gtk や Qt など他のウィジェットをラップするのではなく、基本的な描画レイヤを用意してそれを用いてコントロールの描画を独自に行っている。同様の方法を用いる例として、Java の Swing がある。

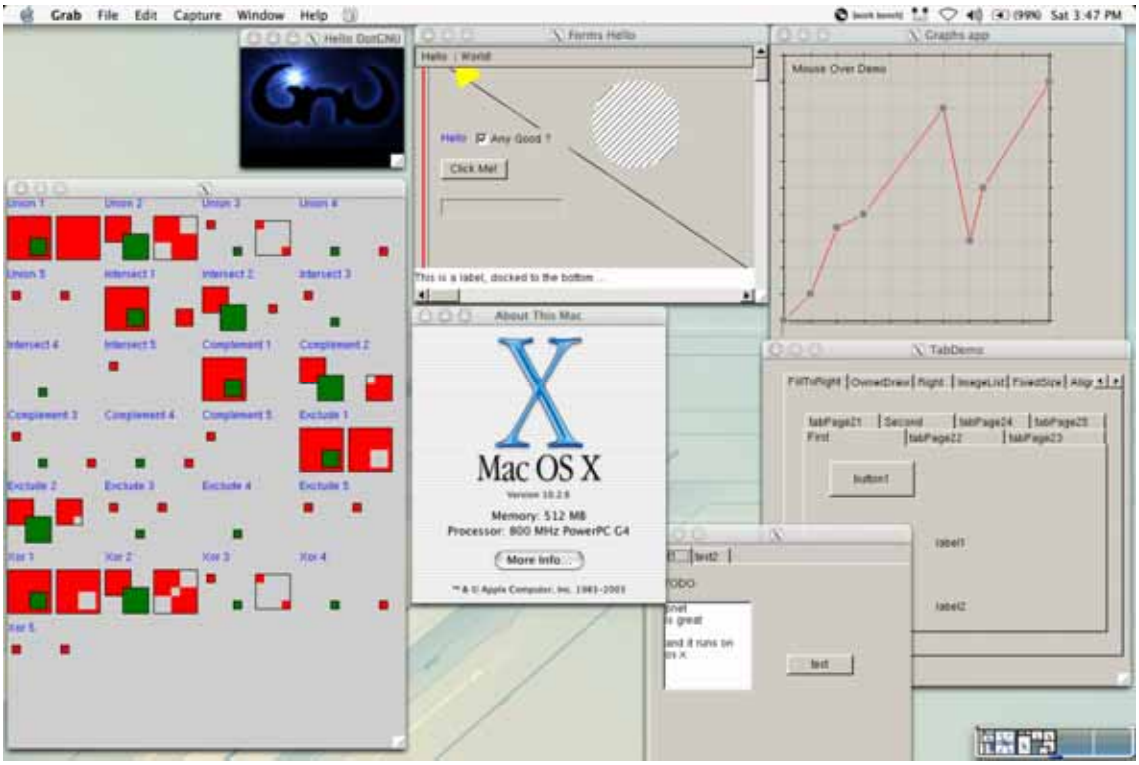
このアプローチは、他のウィジェットを使う方法に比べて、Windows の表示や振る舞いをよりよく高い移植性で、エミュレートできる。この実装は、すでに Linux や Mac OS X 上で動作している。

この実装は次の 3 つのレイヤからなる。

- System.Drawing：基本的な描画機能を提供し、Windows の GDI を可能な限りエミュレートする。コントロールへの描画は、System.Drawing.Graphics を用いる。
- System.Drawing/Toolkit：プリミティブな描画ツールキットへのインタフェースを定義する。ツールキットは単純な線やテキストなどの描画に加えて、簡単なウィンドウのメカニズムも定義する。
- System.Windows.Forms：上記 2 つのレイヤである、System.Drawing と System.Drawing/Toolkit を用いて、Forms API で定義されている様々なコントロールやフォーム、ダイアログなどを実装する。

System.Windows.Forms クラスライブラリを使ったサンプルアプリケーションを、

Mac 上の DotGNU で実行した場合のスクリーンショットを以下に示す。

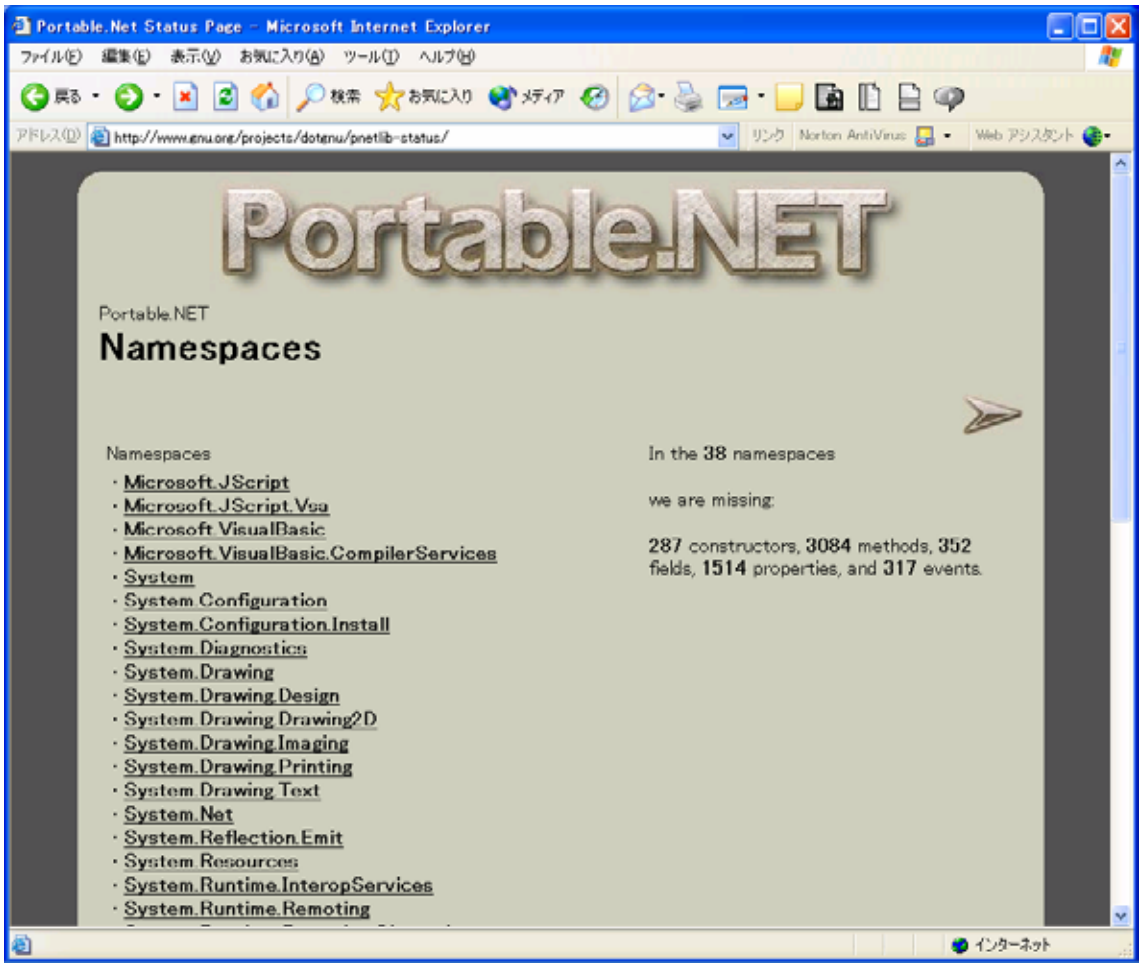


5.2.3. DotGNU プロジェクトのステータス

DotGNU プロジェクト全体としては、mono プロジェクトのようなロードマップは公開されていない DotGNU プロジェクト全体は、2003 年 11 月にバージョン 0.1 がリリースされ、現在も同じの状態である。

DotGNU プロジェクトのサブプロジェクトである Portable.NET については、2004 年 3 月に 0.6.4 がリリースされている。

クラスライブラリのステータスは、次のような Web により公開されている。



5.3. Rotor プロジェクト

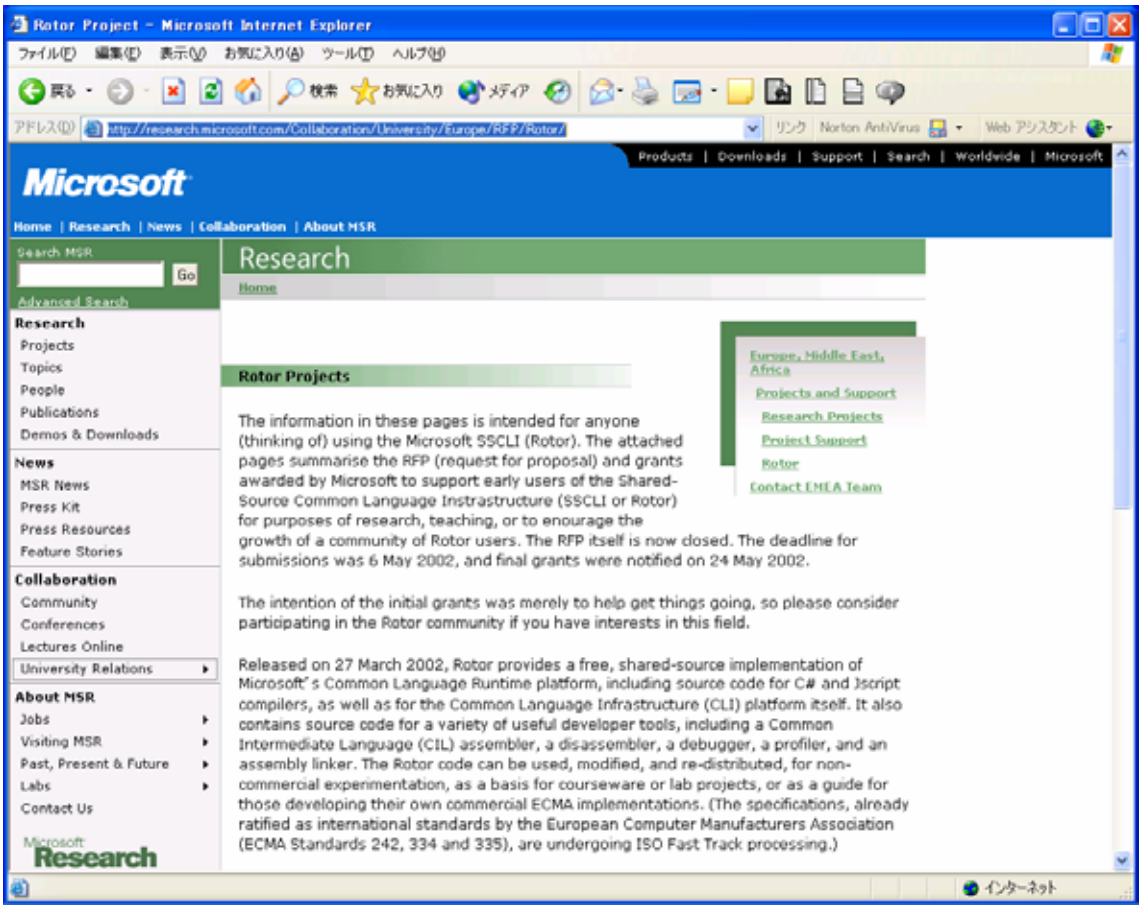
Rotor は、シェアードソースライセンスに基づきマイクロソフト社から提供されている、C#コンパイラ、Jscript コンパイラ、CLI プラットフォーム等からなるプロジェクトであり、2002 年 3 月にβリリースされ、11 月にバージョン 1.0 がリリースされた。また、アセンブラ、逆アセンブラ、デバッガ、プロファイラ、リンカといった、開発者のための様々なツールも提供されている。

このプロジェクトの成果物は、Windows XP と Free BSD、Mac OS X 上で動作する。

Rotor プロジェクトは、シェアードソースライセンスに基づき公開されており、オープンソースソフトウェアではない。

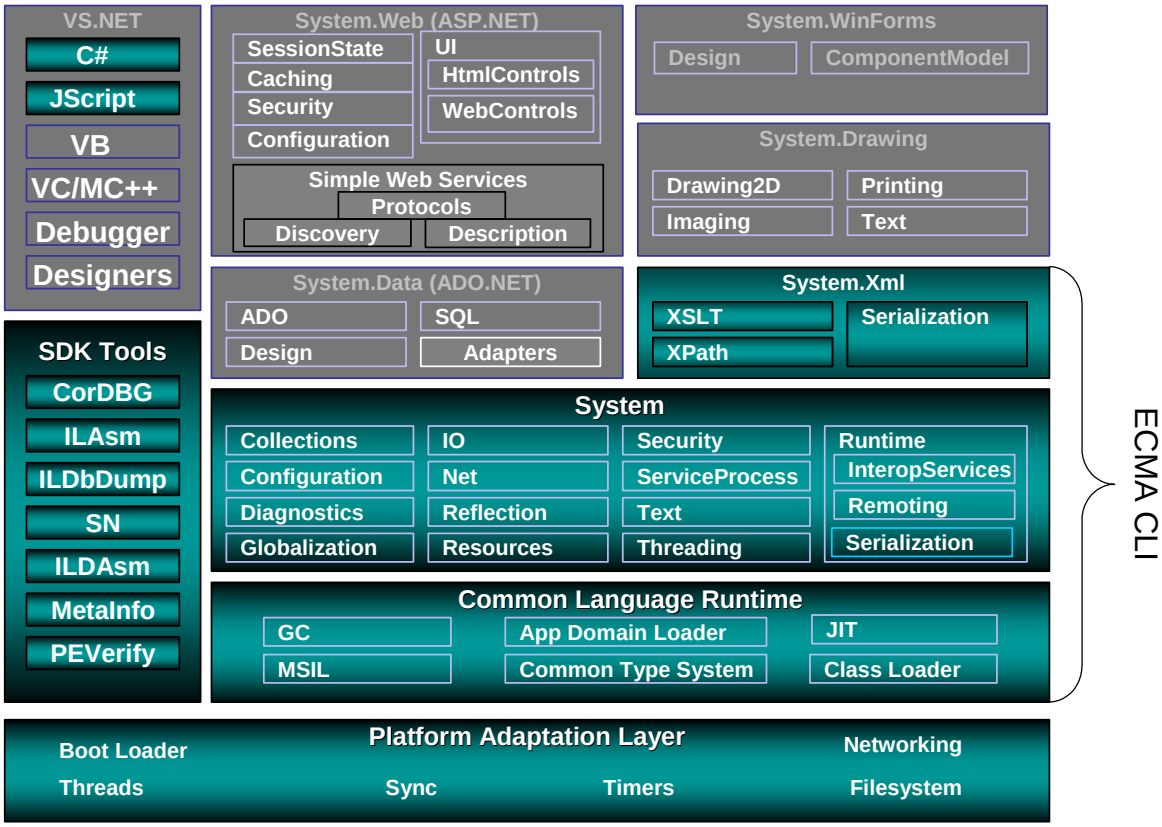
プロジェクトのホームページ：

http://research.microsoft.com/Collaboration/University/Europe/RFP/Rotor/



Rotor プロジェクトの範囲を、下図に示す。

Rotor プロジェクトの範囲
（http://research.microsoft.com/programs/europe/rotor/workshop.aspx）



5.4. その他のプロジェクト

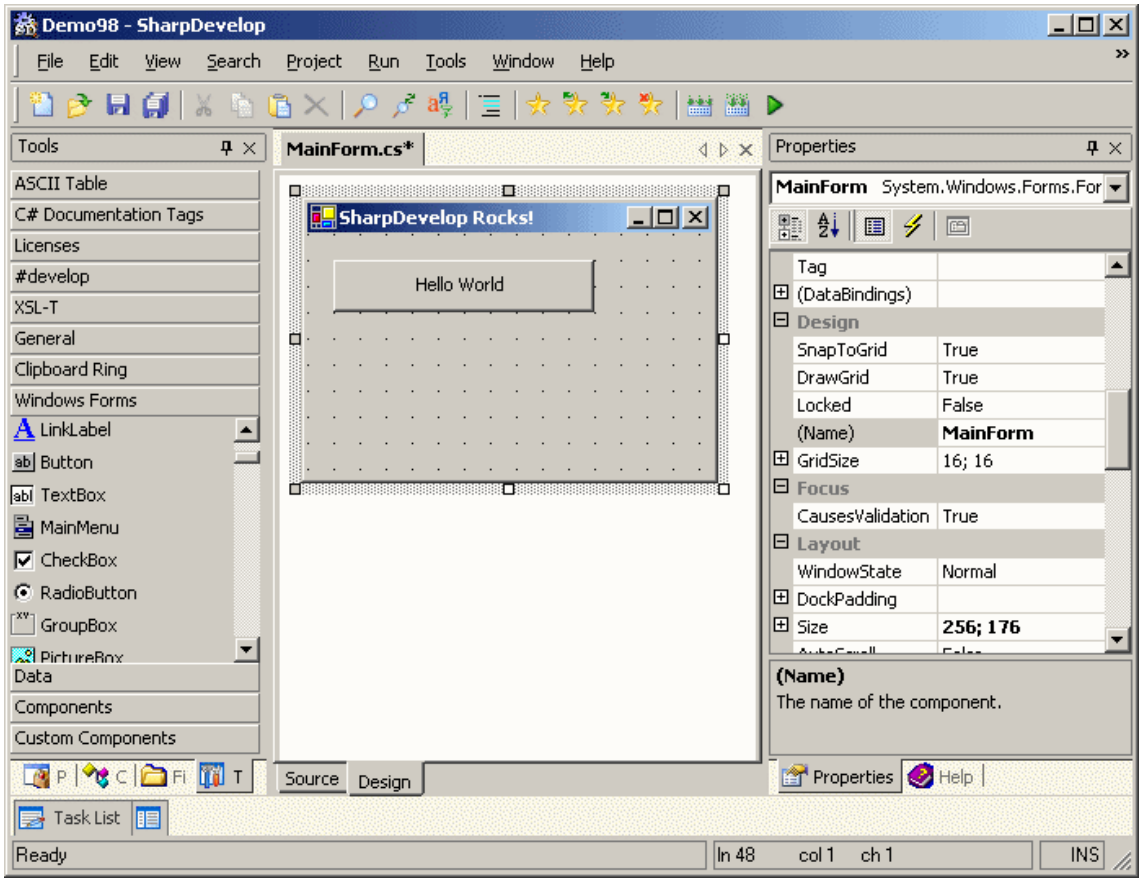
.NET Framework の他にも関連するオープンソースソフトウエアが開発されている。ここではその中から、次の 3 つを解説する。

5.4.1. IDE SharpDevelop

IDE SharpDevelop は、Visual Studio のような開発環境を構築するオープンソースプロジェクトである。このプロジェクトでは、プログラミング言語として C#と VB.NET を対象とする統合開発環境（IDE）を開発している。この成果物は、.NET 上で動作する。

このプロジェクトは、オープンソースのライセンスとして GPL を用いている。

この IDE の外観を以下に示す。



この IDE の特徴および機能概要を以下に示す。

- C#と VB.NET のフォームデザイナ
- C#と VB.NET のコンパイラ
- C#から VB.NET へのコンバータ
- 多言語に翻訳されたユーザインタフェース

- この IDE 自体は C#で記述されており、軽量である。
- C#や VB.NET の他に、ASP.NET、ADO.NET、XML、HTML が記述できる。

プロジェクトのホームページ：http://www.icsharpcode.net/OpenSource/SD/Default.aspx

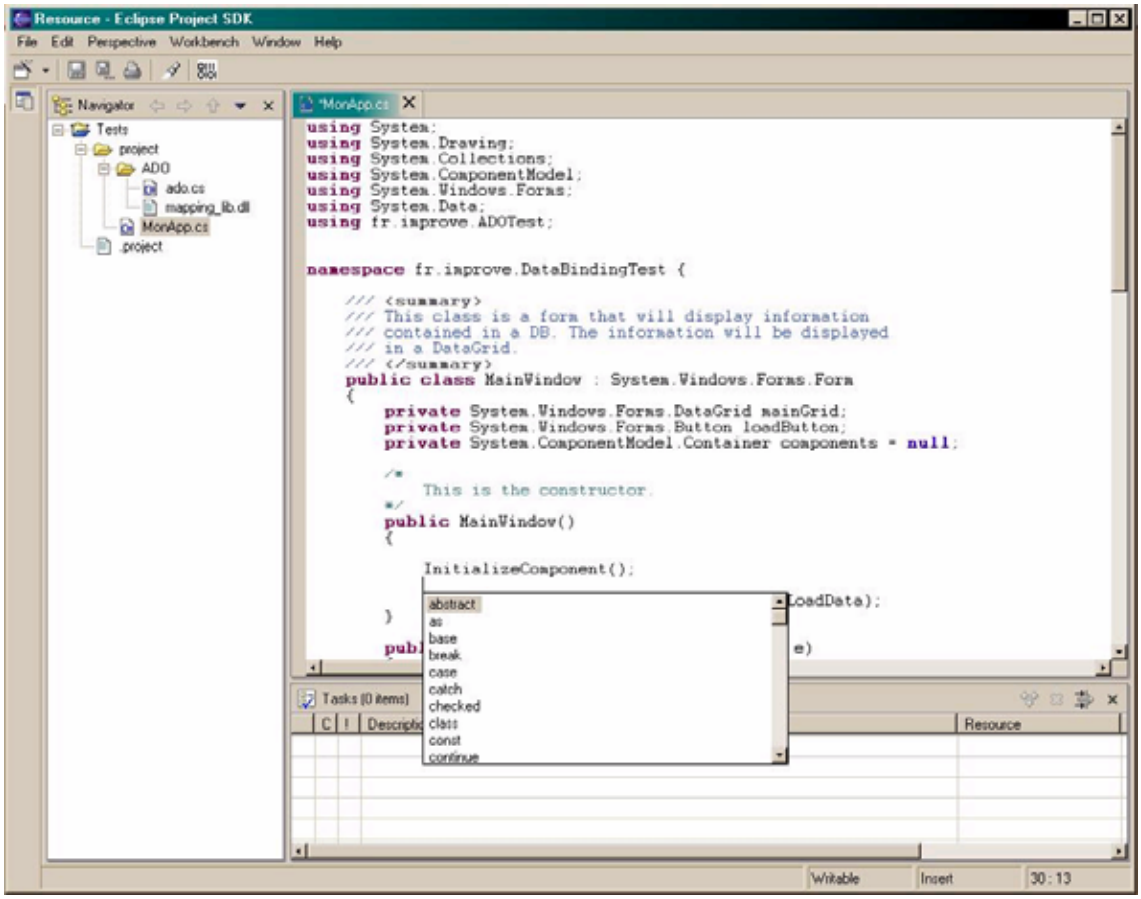
5.4.2. Improve C# Plug-in for Eclipse

このプロジェクトでは、Java 言語などのプログラミングにおけるデファクトスタンダードの開発環境である Eclipse を用いて C#のプログラム開発を可能にするためのプラグインを開発している。

前述の IDE SharpDevelop では統合開発環境全体の作成しているが、このプロジェクトは既存の開発環境である Eclipse を用い、プラグインにより C#への対応を行うものである。

このプロジェクトはオープンソースプロジェクトであり、用いているライセンスは CPL v0.5 である。

このプラグインを適用した Eclipse の外観を次図に示す。

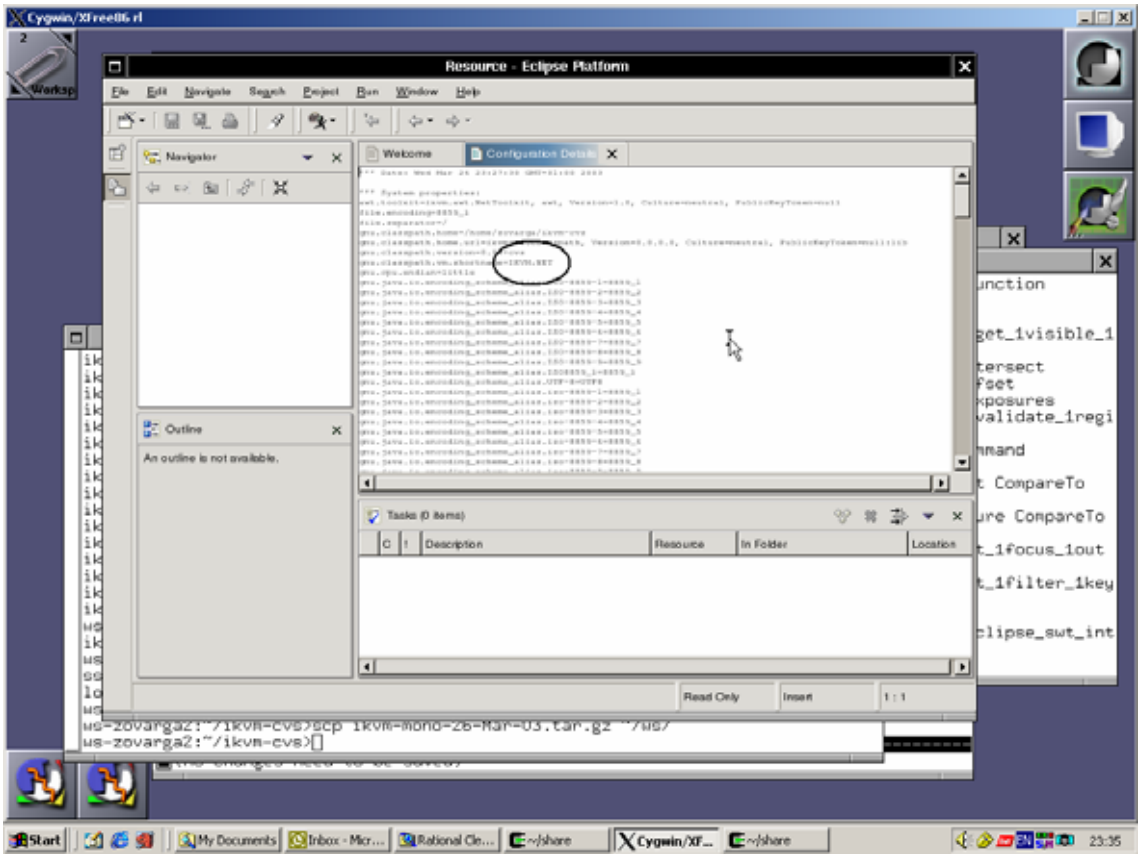


プロジェクトのホームページ：http://www.improve-technologies.com/alpha/esharp/

5.4.3. IKVM.NET

このプロジェクトは、.NET 環境上で Java プログラムを実行可能にするための Java Virtual Machine を開発するものである。オープンソースソフトウェアであり、そのライセンスは独自である。

実際に Java プログラムを実行するには、それが依存する Java のクラスライブラリが必要になる。このプロジェクトはクラスライブラリの開発は目的としておらず、GNU Classpath プロジェクトの成果に依存している。



プロジェクトのホームページ：http://sourceforge.net/projects/ikvm/

6. .NET 互換オープンソースの商用利用

前章で述べた.NET 互換オープンソースプロジェクトの商用利用について、各プロジェクトのライセンスを調査すると共に、すでに商用利用することが公表されている製品の概要と利用される技術について調査し、今後考えられるビジネスモデルについて記述する。

6.1. ライセンス

6.1.1. mono のライセンス

mono はそれを構成するコンポーネントによって、以下のように異なるライセンスが用いられている。

- C#コンパイラには GNU GPL が適用されている。
- ランタイムライブラリには、GNU Library GPL が適用されている。
- クラスライブラリには MIT X11 ライセンスが適用されている。

また GPL や LGPL を用いられない場合、C#コンパイラとランタイムライブラリにはそれ以外の独自のライセンスを適用することもできる。

mono 上で実行される独自のアプリケーションについては、独自のライセンスを用いられるようになる見込みである。

ライセンスのページ：http://www.go-mono.com/faq.html#licensing

6.1.2. DotGNU のライセンス

DotGNU のプロジェクト全体のポリシとしては、GNU GPL と互換なライセンスを用いることになっている。このプロジェクトは、それを構成するコンポーネントによって、以下のように異なるライセンスが用いられている。

- Dot GNU アプリケーションとサンプルは、主に GNU GPL が適用されている。
- Dot GNU ライブラリは、主に GNU Lesser GPL か、以下の例外のついた GNU GPL が適用されている。

  As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice, provided that you also meet, for each linked independent module, the terms and conditions of the license of that module. An independent module is a module which is not derived from or based on this library. If you modify this library, you may extend

this exception to your version of the library, but you are not obligated to do so. If you do not wish to do so, delete this exception statement from your version.

- Dot GNU ドキュメントは、主に GNU FDL（Free Documentation License）が適用されている。

ライセンスのページ：http://www.gnu.org/projects/dotgnu/free-software.html

6.1.3. Rotor のライセンス

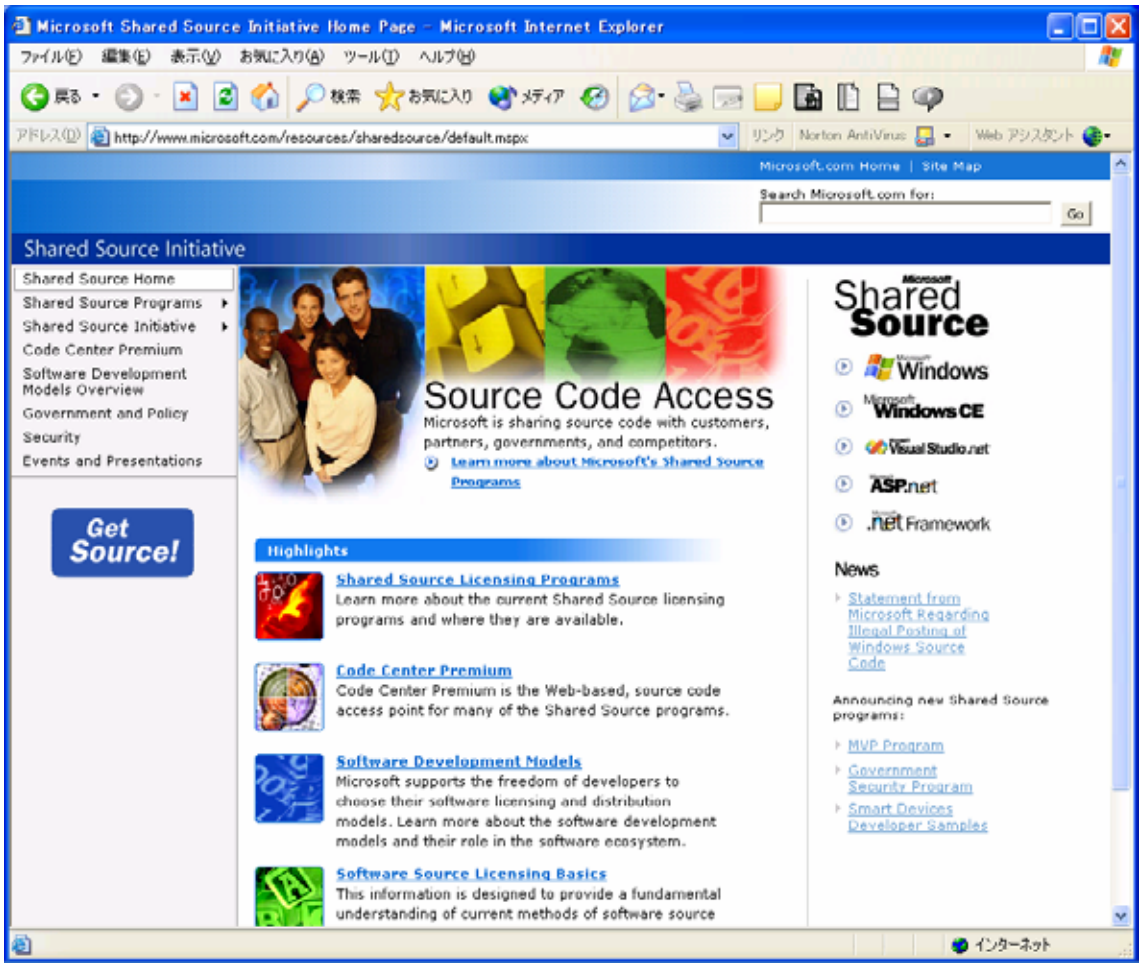
Rotor プロジェクトは、シェアードソースライセンスが用いられている。このライセンスの基では、Rotor プロジェクトの成果を非商用の実験や、教育のための教材などに用いることに限定されており、この成果物を用いた商品を開発することはできない。

Rotor のライセンスは以下の URL で公開されている。また、付録として添付した。

Rotor のライセンスのページ：
http://msdn.microsoft.com/MSDN-FILES/027/002/097/ShSourceCLILicense.htm

シェアードソースライセンスのページ

http://www.microsoft.com/resources/sharedsource/default.mspx



6.2. .NET 互換オープンソースを利用した商用アプリケーション

本報告の作成時点では、DotGNU プロジェクトの成果を用いた商用アプリケーションの存在は確認できなかった。また Rotor プロジェクトは、シェアードソースライセンスであることから、プロジェクトの成果物を用いた商用アプリケーションを開発することはできない。

mono プロジェクトの成果物を用いた商用アプリケーションとしては、以下のものが販売中あるいは開発（計画）中であることが知られている。

6.2.1. Winfessor

インスタントメッセージング(IM)ソリューションプロバイダの Winfessor 社は、mono プロジェクトの成果を、Winfessor SoapBox Framework Mono Edition に適用した。Winfessor SoapBox Framework Mono Edition は、Unix や Linux などの mono をサポートするプラットフォーム上で、.NET ベースの IM ソリューションを開発者が構築することを可能にする。

Winfessor 社は、.NET を用いることで、IM 開発プラットフォームとしての柔軟性と高い生産性を得ると共に、mono を用いることでそれを Windows 以外のプラットフォームでも実行可能になると見込んでいる。

製品概要：

SoapBox Framework は、開発者が独自の IM アプリケーションを開発する場合や、既存のアプリケーションに IM 機能を組み込む場合に用いるフレームワークである。このフレームワークは IM 分野における標準的な仕様である XMPP に準拠しており、フレームワーク自体は次のソフトウェアと相互運用可能であることを確認している。

- SoapBox Server 2004
- Jive Messenger
- Jabber Software Foundation's JabberD
- The Jabber Inc. Platform

SoapBox Framework には、次の特徴がある。

開発期間の短縮	SoapBox Framework を用いれば、開発者が XMPP プロトコルなどの詳細に立ち入る必要がない。また XMPP についての知識がない開発者でも、容易に短期間でデスクトップ、あるいはモバイル向けの IM アプリケーションを開発できる。

統合デバッガ	拡張された.NET トレースにより、開発者は内部で起こっている問題について詳しい情報を知ることができる。.NET のトレースインフラに完全に統合しているので、Winfessor のトレースツールは実行中のアプリケーションについても調べることができる。
パフォーマンスカウンタ	Windows のパフォーマンスカウンタに統合することで、開発者はアプリケーションのスケーラビリティなどについて確認できる。またそのアプリケーションを運用開始後は、アプリケーションの動作状況についてリモートモニタできる。
セキュリティ	セキュリティは XMPP プロトコルの主要な部分である。XMPP を用いることにより、ストリーム暗号と認証を実現できる。

SoapBox Framework には、現在 Desktop, Mobile, Mono, Web Service の 4 種類がある。

Desktop：Windows の.NET 上で動作するアプリケーションの開発用

Mobile：PocketPC 等の.NET Compact 上で動作するアプリケーションの開発用

Mono：Linux を含む mono 上で動作するアプリケーションの開発用

Web Service：Web サービスを提供するアプリケーションの開発用

SoapBox Framework ホームページ

Mono Edition ホームページ

Winfessor 社：http://www.winfessor.com/

Winfessor SoapBox Framework Mono Edition：
http://www.winfessor.com/portal/DesktopDefault.aspx?tabid=57

6.2.2. Tipic

Winfessor 社と同様に、.NET ベースの IM ソフトである Tipic Instant Messaging Platform（TIMP）を、mono プロジェクトを用いることにより Unix や Linux 上でも動作させようとしている。本報告の執筆時点では、同社のホームページには mono プロジェクトを使用しているという記述はない。

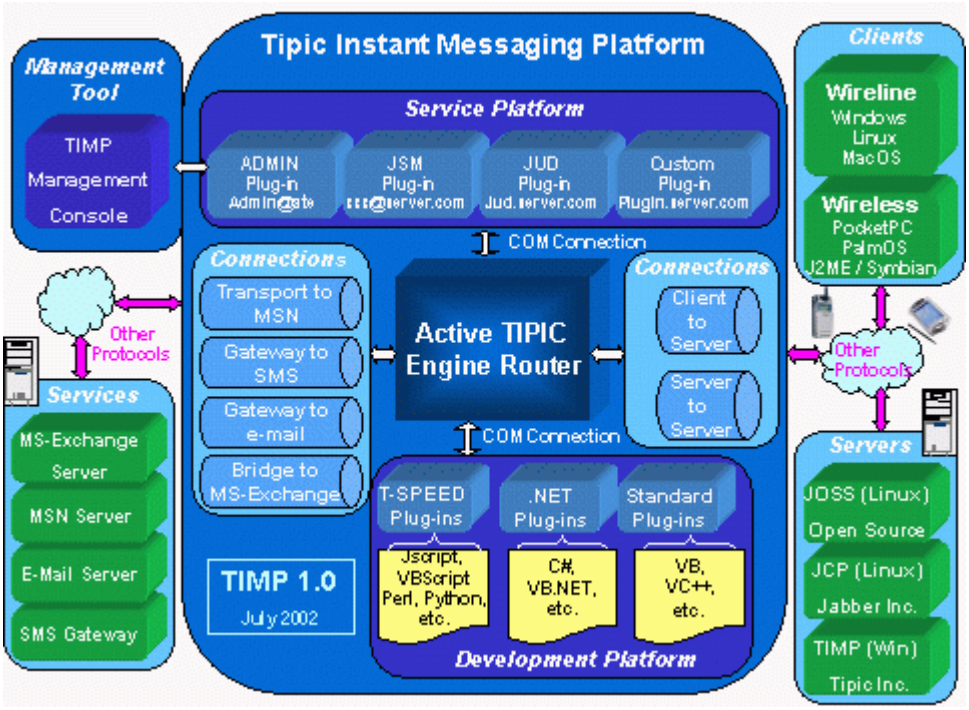
製品概要：

Winfessor Soapbox Framework の製品と同様、.NET ベースの IM ソフトであるが、Soapbox Framework が基本的にクライアントサイドのアプリケーション開発ソフトで

あったのに対して、TIMP はサーバ製品である。

TIMP を用いたシステムの構成

TIMP の内部構成



http://www.tipic.com/

6.2.3. OpenLink Software（Virtuoso Universal Server）

OpenLink Virtuoso は、仮想データベース、XML データベース、SQL データベース、Web アプリケーションサーバの機能を一つのサーバで実現する、クロスプラットフォームユニバーサルサーバである。

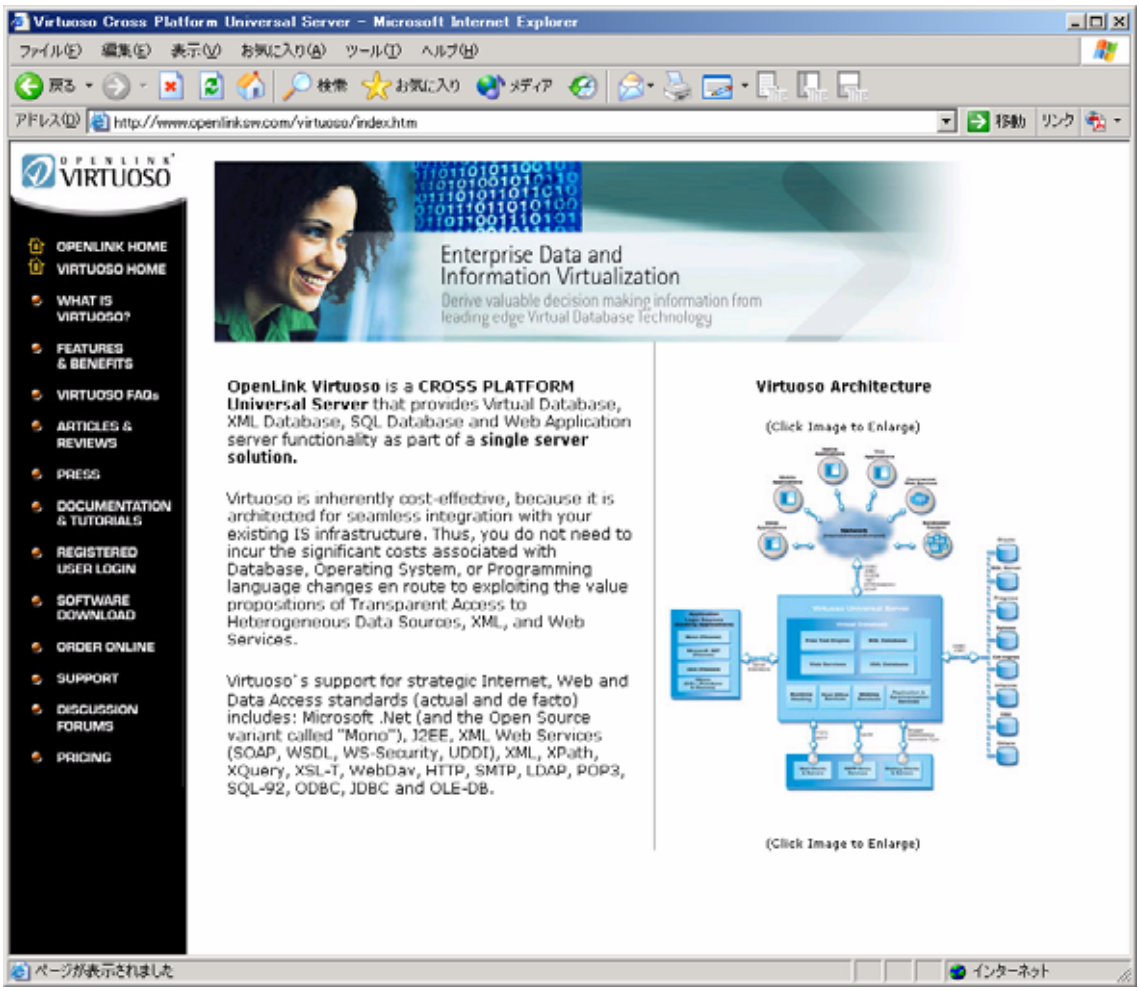
Virtuoso は、次のような様々な標準（事実上の標準も含む）をサポートしている。

- .NET（および mono）
- J2EE
- Web サービス（SOAP, WSDL, WS-Security, UDDI）
- XML
- XPath
- XQuery
- XSLT
- WebDAV
- LDAP
- HTTP, SMTP, POP3 など

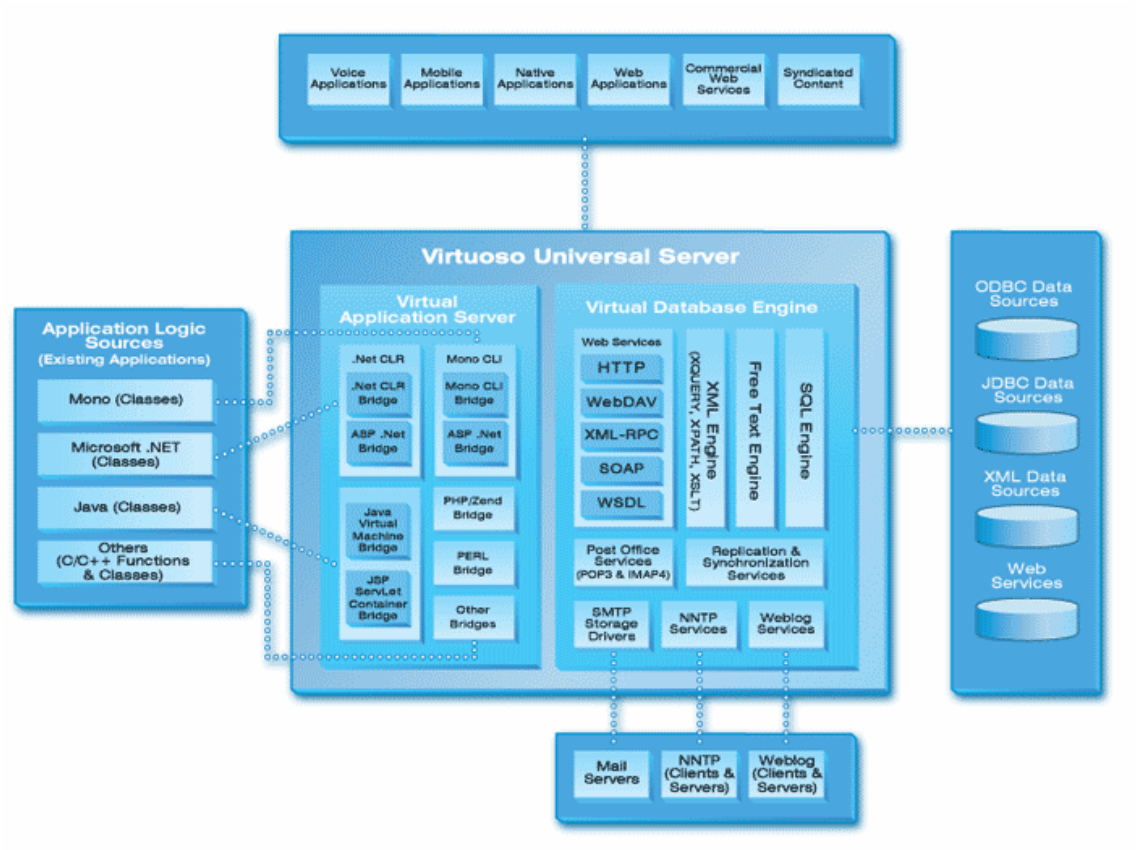
Virtuoso は、Web アプリケーションサーバの言語として、ASP.NET を利用しており、

これをクロスプラットフォームで利用可能にするためにも mono が用いられている。

Virtuoso ホームページ：



Virtuoso のアーキテクチャ



http://www.openlinksw.com/

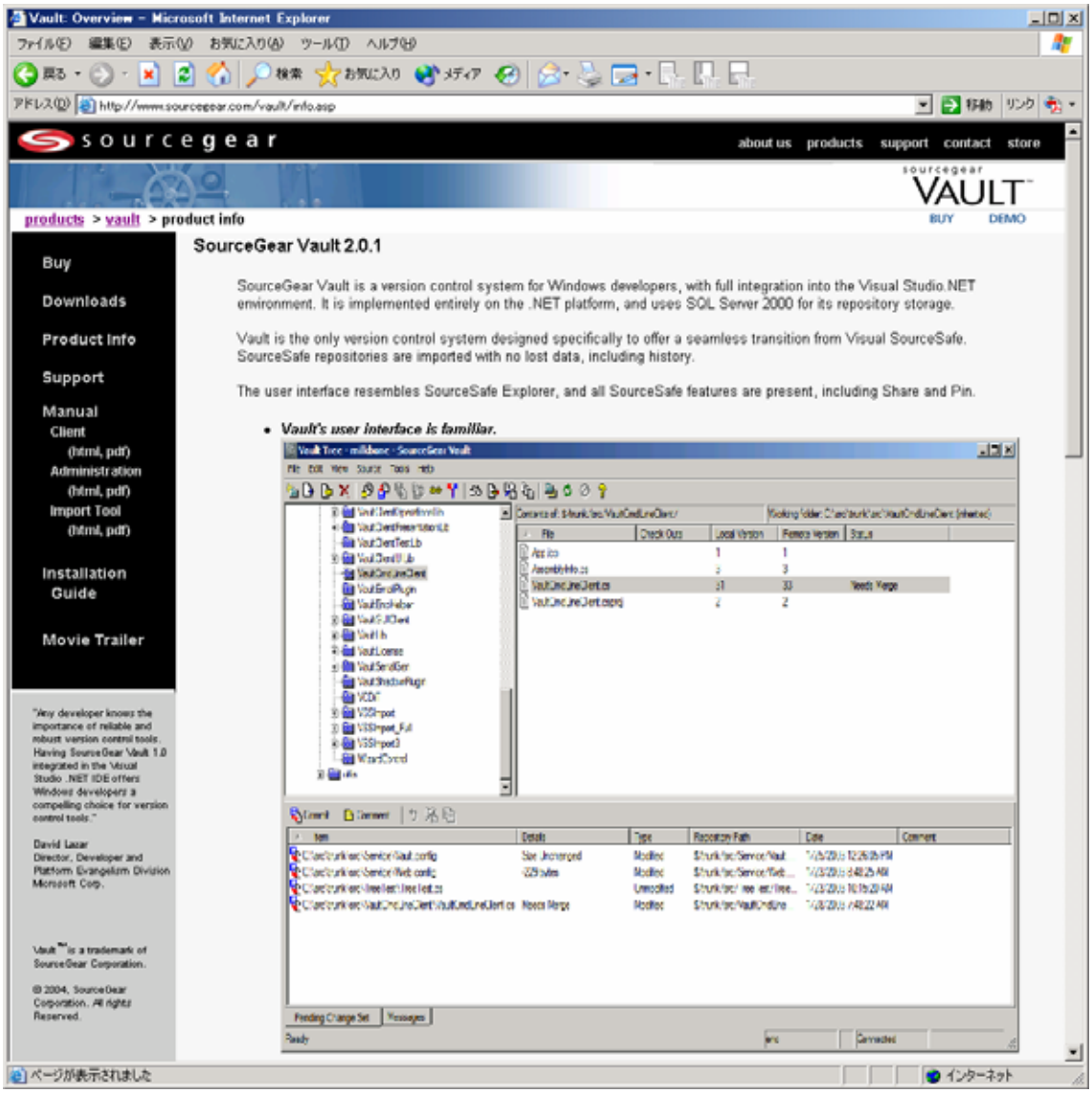
Virtuoso の開発者が mono の利用についてインタビューを受けた記事を、付録として添付する。

6.2.4. SourceGear Corporation

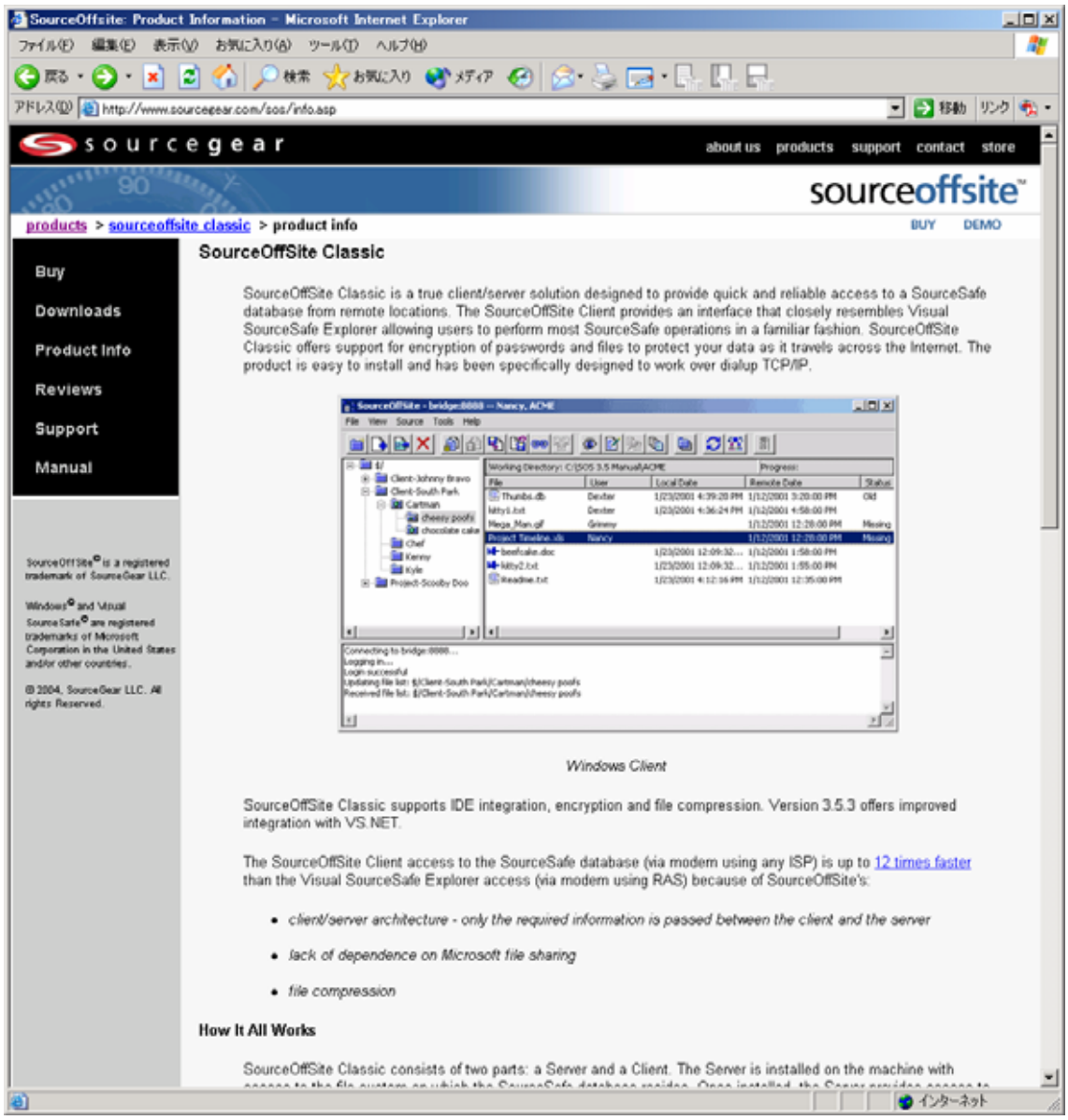
SourceGear Vault ソースコード管理システムのクライアントとして、Unix や Linux 上で動作する製品を開発する予定である。

同社の Web サイトを検索したが、mono を利用しているという記述はなく、また SourceGear Vault のクライアントとしては、現時点では Windows 版しかない。それとは別に、SourceOffSite Classic という製品はクロスプラットフォームをうたっており、こちらに mono が使われている可能性はあるが、記述がないため詳細は不明である。

SourceGear Vault 製品紹介ページ：



SourceOffSite Classic 製品紹介ページ：



http://www.sourcegear.com/

6.3. 想定されるビジネスモデル

オープンソースの.NET 互換環境を用いたビジネスモデルとして、想定されるものを以下にあげる。

- クライアントサイドアプリケーションの開発
- サーバサイドアプリケーションの開発
- アプリケーションサーバの開発
- ホスティングサービス

6.3.1. クライアントサイドアプリケーションの開発

現在クライアント PC の OS としては Windows が圧倒的に多く、クライアントサイドのアプリケーションを開発する場合、まず Windows を対象と考えるのが一般的である。これは.NET になっても急激には変化しないと考えられるが、一方でオープンソースの普及も徐々に進んでおり、Linux 等のクライアントも無視できない数になっていく可能性がある。また、特定の分野では Mac OS や Unix クライアントがある程度の割合で用いられている。

.NET 以前であれば、エミュレーションの試みは続けられているものの実用レベルに到達しているとは言えず、実際にはこれらの OS は全く別 OS として、それぞれに別々にアプリケーションを開発する必要があった。

.NET になると、CLR によって実行環境自体が抽象化されるため、mono などのプロジェクトを適用することにより移植性の高いアプリケーションを開発できる可能性がある。前述の Winfessor 社の例は、このモデルにあたると考えられる。

このモデルを用いる場合、開発するアプリケーションが GUI を持たないとすれば、例えばクライアント PC 上においてバックグラウンドで動作するアプリケーションや、コマンドラインのインタフェースを持つアプリケーションなどは、比較的容易に移植性の高いアプリケーションを開発できると考えられる。

GUI は細かい違いでも見た目にはっきりわかるため、なかなか互換環境での動作を一致させることが難しい。例えば表示するフォントに同じものがなければ、サイズが微妙に異なる場合があり、画面に表示しきれないような場合が一例として考えられる。mono プロジェクトでも DotGNU プロジェクトでも、GUI の互換環境開発に力を入れており、このような問題は徐々に改善されていくことを期待したい。また、アプリケーション開発時に、クロスプラットフォーム性を検証しながら、問題の起こらない機能を用いて開発することも考えられる。

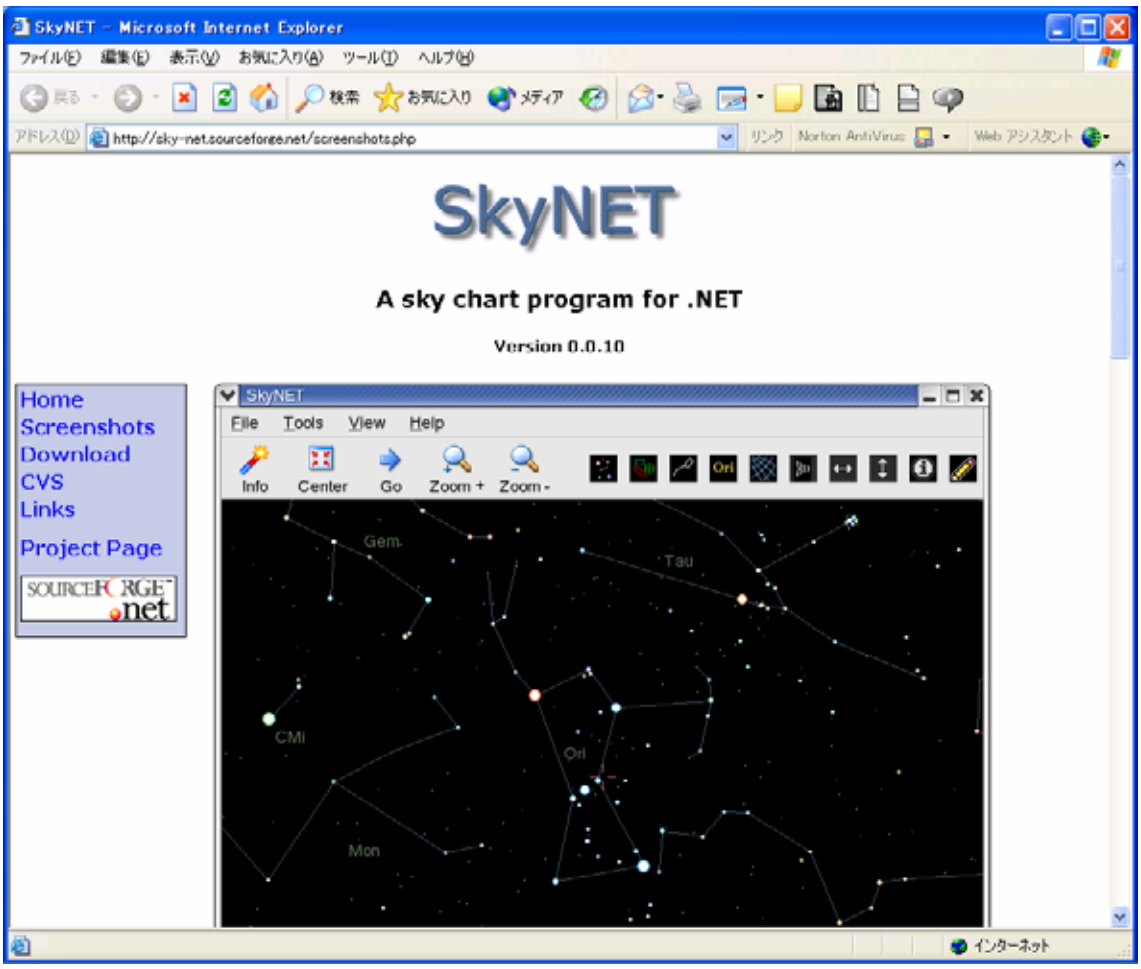
クロスプラットフォーム性を謳う Java においても、すでに 5 年以上にわたって積極的な活動が行われているものの、未だにクライアントサイドアプリケーションで成功しているとは言えない状況と思われる。これには様々な原因が考えられるが、その一つとして、クライアント PC の多数を占める Windows にも Java 実行環境をインストールする必要があ

ることがあげられると思われる。その点では、.NET 実行環境も現時点では同様と思われるが、今後クライアント PC の老朽化等により新しい PC に徐々に移行して行くにつれて.NET 実行環境がインストールされる方向に改善すると考えられる。

逆に、Windows プラットフォームにも実行環境をインストールするのであれば、mono プロジェクトが採用しているような Gtk#を使うというのも一つの選択肢になる可能性もある。このような例として、商用ではないがオープンソースプロジェクトである Sky.NET がある。これは、＃WT という Eclipse で用いている GUI クラスライブラリである SWT を C＃に移植したものを用いており、どのプラットフォームで実行する場合にも＃WT を必要とする。このプロジェクトは最初から.NET 互換プロジェクトをターゲットに設定しているので、このように独自の描画クラスライブラリを用いることになったと思われる。

SkyNET のホームページ（次図はスクリーンショットのページ）

http://sky-net.sourceforge.net/index.php



一方、.NET 互換環境を開発プラットフォームとしてみると、言語の生産性の高さ、メモリリークを起こしにくい実行環境、充実したクラスライブラリなど、既存の開発プラット

フォームに比べて優れた点が多いと思われる。その点だけに着目して.NET との互換性を問わず、単に生産性の高い開発プラットフォームとして用いるという利用方法も考えられる。

これをふまえて、クライアントサイドアプリケーション開発のビジネスモデルとして次のようなものが考えられる。

- 互換プロジェクトとマイクロソフト.NET の両方で動作するクライアントサイドアプリケーションを開発する。GUI のあるアプリケーションでは容易ではない場合があり、GUI がないアプリケーションか、問題の起こらない程度の簡単な GUI を持つアプリケーションが当面の対象になると思われる。
- 現在の Windows でも、サードパーティから様々な高度な GUI 部品が販売されている。このような会社から、互換プロジェクトとマイクロソフト.NET の両方で同じ見栄えの GUI 部品が販売されることが考えられる。このような部品が販売されると、互換プロジェクトとマイクロソフト.NET の両方で動作するクライアントサイドアプリケーションの開発は容易になって広く用いられるようになり、それがさらに他の会社の参入を促す、という正のフィードバックを期待したい。
- 特定の用途で Mac OS や Linux などのクライアントのみを対象とする場合でも、開発プラットフォームとしての生産性の高さから互換プロジェクトを用いる場合が考えられる。
- 互換プロジェクトとマイクロソフト.NET の両方で動作するクライアントサイドアプリケーションの開発をスムーズに行うには、ある程度の経験が必要になると思われる。新規にこのようなアプリケーションを開発する開発者向けに、コンサルティングを行うことができれば、その需要はあると考えられる。

6.3.2. サーバサイドアプリケーションの開発

サーバサイドアプリケーションはクライアントサイドアプリケーションに比べると、クロスプラットフォーム性の実現が一般的には容易である。これは、Java においても、サーバサイドにおいて活用されている状況からも見て取れる。この理由には、実行する環境のバリエーションが少ないこと、GUI が不要であること、サーバサイドアプリケーションにアクセスするためのインタフェースが定められており、クライアントサイドアプリケーションと明確に分離されていることなどが理由として考えられる。また、Java の場合は、Java Virtual Machine 上でプログラムを実行することでメモリリークが起こりにくいという点も理由として考えられる。

これらの理由は.NET にもそのまま適用でき、.NET においてもサーバサイドアプリケーションに利用される傾向は同様と思われる。特に.NET は Web サービスの優れた実行環境であることを目標の一つとしており、Web サービスのインタフェースによってクライアン

トサイドと明確に分離できる。

前述の Tipic 社や OpenLink 社の例は、これに該当すると思われる。

また、Windows はサーバ OS としての信頼感に欠けるという印象を持つ運用管理者も少なくないと思われるが、Unix や Linux 上で動作する.NET 互換環境を用いることによりこのような懸念が改善されることも考えられる。さらに、例えば mono プロジェクトは IBM のメインフレームにも移植されつつあり、現状ではインタープリタによる実行のため動作速度が遅いものの、このようなプロジェクトが進めばさらに状況は改善することが考えられる。

これをふまえて、サーバサイドアプリケーション開発のビジネスモデルとして次のようなものが考えられる。

- 互換プロジェクトとマイクロソフト.NET の両方で動作するアプリケーションを開発する。
- mono などの互換プロジェクト上だけで動作するアプリケーションを開発する。サーバサイドであれば、アプリケーションを実行するプラットフォームを選択できる場合が多いと思われるので、例えばサーバに Linux を用いることが決まっているようなプロジェクトでマイクロソフト.NET 上での動作を保証する必要がない場合も出てくる可能性がある。
- サーバサイドアプリケーションを開発するときの利便性を高めるツールや、互換プロジェクトとマイクロソフト.NET の差を埋めるような製品が出てくる
- サーバサイドアプリケーションは、複数のサーバソフトウェアの組み合わせにより実現することが多いと思われ、その組み合わせ方や設定などは経験により明らかになっていくと考えられる。そこでその知見をもとにコンサルティングを行うビジネスは成立するのではないかと思われる。

6.3.3. アプリケーションサーバの開発

Java を例に取ると、サーバサイドアプリケーションを実行するための環境として、いろいろなアプリケーションサーバが提供されている。.NET の場合も、同様の状況になると考えられる。

Windows では、第 2 章で記述した.NET Enterprise Server などマイクロソフト社が提供する各種サーバソフトウェアを組み合わせることにより、アプリケーションサーバが構成される可能性がある。しかし mono のような互換プロジェクトは、.NET Framework という基本となる実行環境の機能だけからなり、アプリケーションサーバとして構成するために必要な.NET Enterprise Server に相当するものや、それと連携するためのソフトウェアを別に用意する必要がある。その例として、例えばデータベースサーバとの接続を行う場合、アプリケーションからデータベースの操作には ADO.NET という抽象的なインタフェ

ースを用い、データベース自体はPostgreSQL等を利用できるが、ADO.NETとPostgreSQLの間をつなぐためのソフトウェア（ADO.NET 対応の PostgreSQL ドライバ）が必要になるが、それは個々に開発する必要がある。

これをふまえて、アプリケーションサーバ開発のビジネスモデルとして次のようなものが考えられる。

- Java における BEA WebLogic や IBM WebSphere のような、.NET 互換プロジェクトの成果を用いて.NETにおけるWebアプリケーションサーバを開発する。Windows以外のプラットフォームでは、これが重要になると考えられる。
- 上記の ADO.NET 対応の PostgreSQL ドライバのように、.NET 互換環境と共に用いるソフトウェアにはドライバ等の開発が必要になる場合が多いと思われる。高性能なドライバの開発や、各種サーバ間を連携するためのソフトウェア等の開発については、普及のために重要でありその需要があると考えられる。

6.3.4. ホスティングサービス

サーバサイドアプリケーションを開発し、それを社内のような閉じた世界ではなく、インターネットから利用可能にする方法としてホスティングサービスがある。ここで言うホスティングサービスでは、ホスティングサービス業者（以下、ホスティング業者と省略）はハードウェアとOSなどの基本的なソフトウェアからなるサーバを運用し、ユーザへのサービス提供者（以下、サービス提供者と省略）はそのサーバ上でアプリケーションを運用し、ユーザはインターネットを経由してそのサービスを利用することを想定している。

Webホスティングサービスの分野において、当初は静的なコンテンツや、CGIやPHP等のスクリプトが主流であったが、最近はJavaが利用可能なサーバを用意したホスティング業者が増えてきている。このサーバを用いると、サービス提供者は Java サーブレットや JSP などを用いてアプリケーションを構築し、それをサーバで運用することになる。

同様の形態は.NET でもすでに始まっている。つまり、サービス提供者が.NET または互換プロジェクトを用いてサービス提供者がアプリケーションを開発し、ホスティング業者はそれをサーバ上で稼働させる形態である。マイクロソフト社から提供されているホスティング業者情報を以下に示す。

マイクロソフト社による ASP.NET ホスティング サービス情報
（http://www.microsoft.com/japan/msdn/asp.net/hosting/default.aspx）

有限会社アースリンクネットワークのホスティングサービスのページ
（http://www.eln.ne.jp/rental_server/hosting/hosting_top.aspx）

また、前述の OpenLink 社の例は、それ自身がホスティングサービスを行うわけではないが、そのようなサービス業者向けのソフトウェアとしての利用形態もあり得る。

ホスティングサービスのビジネスモデルとして次のようなものが考えられる。

- 上記と同様に、ホスティングサービスを行うが、その時に使うのは mono などの互換プロジェクトである。.NET を利用したいがサーバには Windows を使うのをためらうサービス提供者に対して、有効である可能性がある。

- ホスティングサービス業者向けにソフトウェアを開発する。ホスティングサービス業者ごとに得意とする OS が異なることも考えら、Unix を得意とする業者向けに.NET に関係するサービス提供を可能にするようなシステムを提供することはあり得るのではないか。

7. 将来の課題

マイクロソフト社が.NET に力を入れて普及させていく考えであることは明らかであり、また開発者から見ても C#のようにすっきりした言語仕様のプログラミング言語、メモリリークやメモリ破壊を起こしにくい実行環境、充実したクラスライブラリなど、魅力的な点が多く、Windows においてマイクロソフト.NET が普及していく方向には間違いないと思われる。

.NET 互換オープンソースプロジェクトによって、その他の OS においても開発者は上記の利点を享受できるなど、メリットがあることははっきりしている。しかし、実際のプロジェクトに.NET 互換オープンソースプロジェクトの成果を利用するかどうかを判断するためには、その将来性について検討することも非常に重要である。ここでは、.NET 互換オープンソースプロジェクトの課題として以下の点について考察した。

- .NET との互換性
- 仕様への追随
- 特許・著作権等
- 開発の継続性

7.1. .NET との互換性

.NET 互換オープンソースプロジェクトは、もちろんマイクロソフト.NET と互換であることを目指して開発を進められている。しかし、以下のような点で、互換性に問題が生じる可能性があると思われる。

(a) 機能が欠落している場合

.NET 互換オープンソースプロジェクトは.NET の機能を順次開発しているが、いくつかの機能は開発しない可能性がある。例えば mono プロジェクトのロードマップには次のような記載がある。

> **Unsupported technologies**
>
> Some technologies are very hard to implement or are being phased out by components in the Longhorn time frame. In some cases, we feel that they are not crucial to the future of the open source desktop.
>
> System.EnterpriseServices and System.Management come to mind, and we are unlikely to put any resources into the task. We would gladly host the code if someone cares to implement it, but they would likely remain unsupported features of Mono.

この記述からわかるように、mono プロジェクトでは System.EnterpriseServices と System.Management の名前空間にあるクラスライブラリは開発しない方針である。こ

の機能を利用するアプリケーションがあれば、mono プロジェクトは.NET 互換ではない。

ただし、このような隙間を埋める他のオープンソースプロジェクトが立ち上がる可能性や製品が出る可能性がないとは言えない。

(b) 機能はあるがその動作に互換性がない場合

.NET 互換オープンソースプロジェクトの開発は、ECMA 標準やマイクロソフト社が公開しているドキュメントをもとに行われると思われる。しかし、このようなドキュメントだけでは、代表的な場合の動作はわかっても、例外的な場合やエラー発生時の振る舞いについてはわからない可能性がある。そのため、.NET 互換オープンソースプロジェクトを用いてアプリケーションを開発しても、マイクロソフト.NET 上で実行した場合とだいたい同じ振る舞いをするが、時々振る舞いが違う、という非常にやっかいな問題が発生する可能性がある。

Java のように、最初から複数のベンダにより開発環境が実装されることを前提としている場合は、仕様書の明確化と共に仕様への互換性を検証するツールが用意されることも考えられるが、.NET の場合はマイクロソフト社がすべてをコントロールしているため、検証ツールは期待できない。

このような問題は、.NET 互換オープンソースプロジェクトが使われるようになれば、具体的な問題点の発見とその対応が行われるようになると思われる。特に動作に互換性がないような問題であれば、オープンソースプロジェクトの多くに見られる特徴である、素早いフィードバックによって解決されることが多いのではないかと思われる。

また、.NET 互換オープンソースプロジェクトの使いこなしに関する知見が蓄積されていけば、あらかじめ問題を回避することもできると考えられる。すでに現時点でも、次のような書籍の出版が予定されており、詳細は不明だがタイトルから判断する限り、クロスプラットフォーム性を考慮したソフトウェア開発のための手引きとして利用できるであろう。

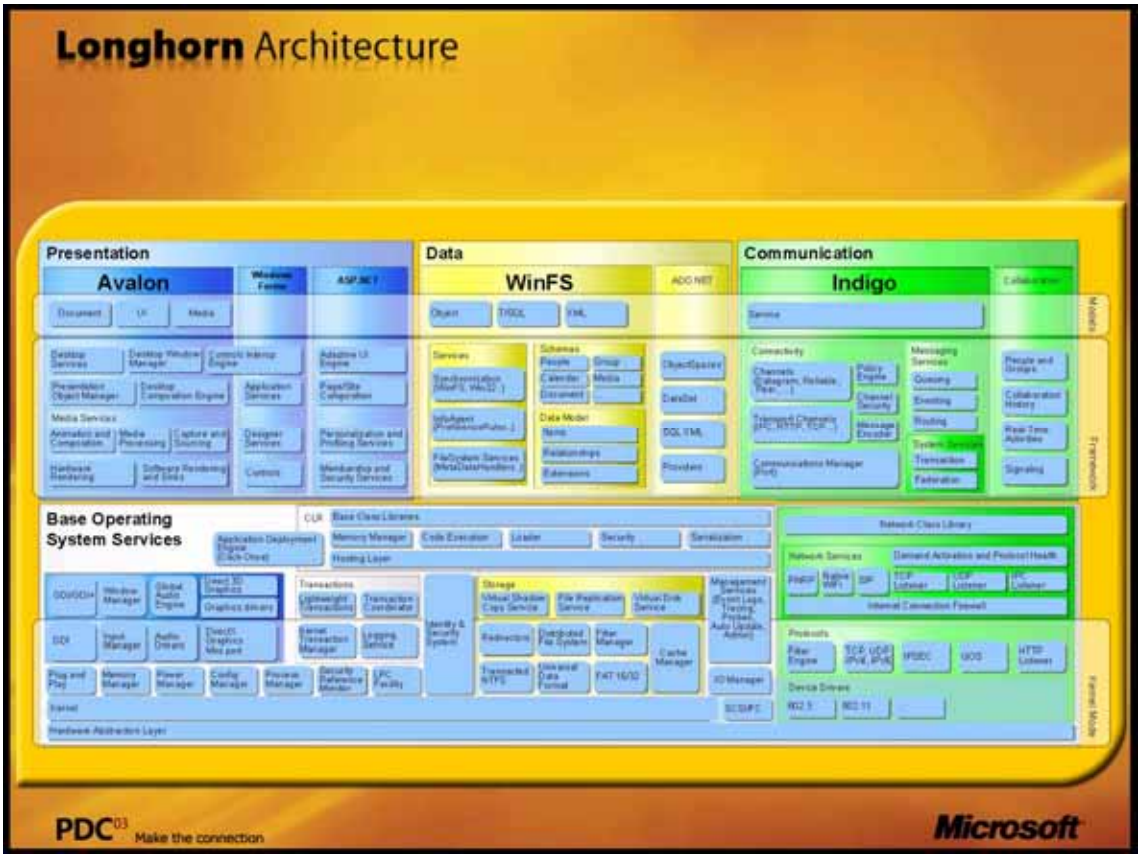
Mono, Portable.Net, and .Net: Cross-Platform .Net Coding Mark Easton (2004/06/01) Springer-Verlag New York Inc

7.2. 仕様への追随

マイクロソフト社は次世代 OS である Longhorn に向け全力で開発を進めており、その過程で.NET クラスライブラリも大幅に拡張される。次図に示す Longhorn アーキテクチャで、下半分が既存のフレームワークであり、その上にある Avalon, WinFS, Indigo が新規開発中の機能である。また、これらの機能を新規開発するにあたって、既存の機能の拡張・変更が必要になることも想像できる。

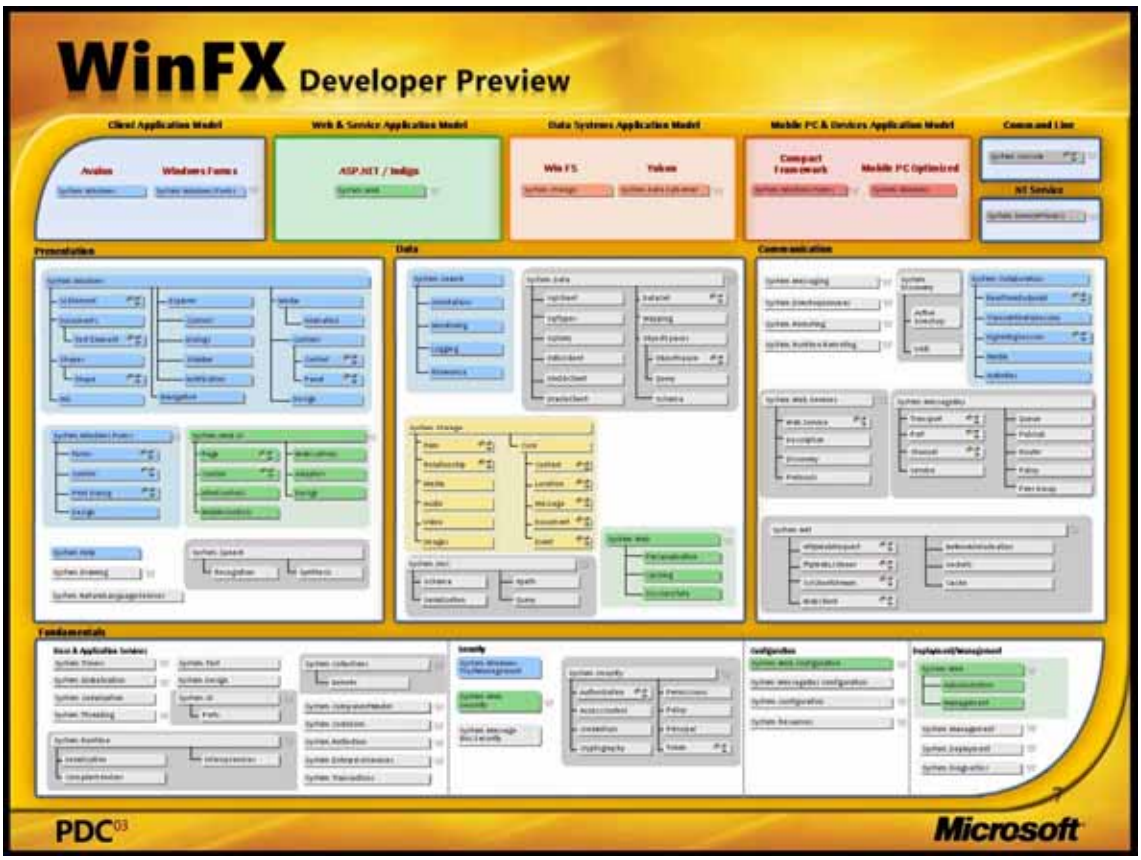
Longhorn アーキテクチャ

（Professional Developers Conference 2003 のプレゼンテーション資料より）



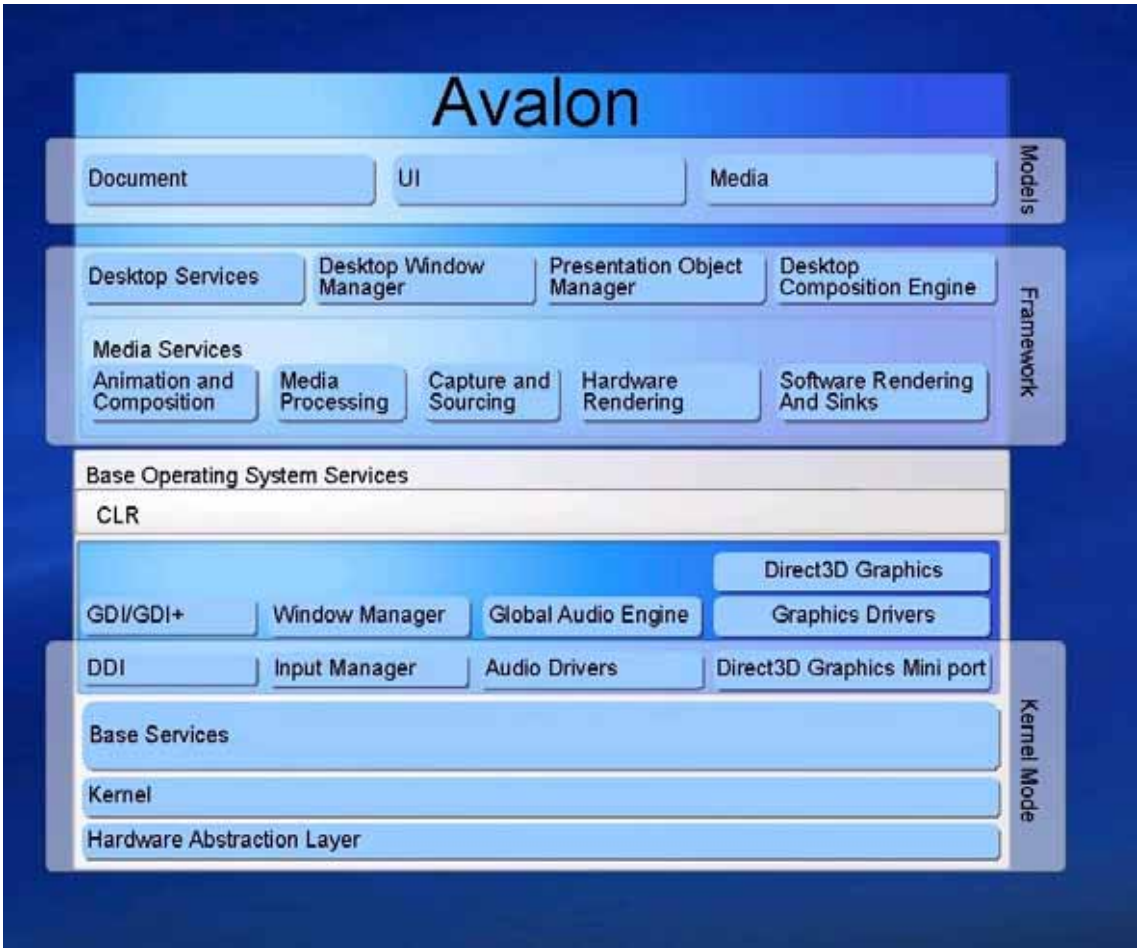
また、Longhorn 開発の中で WinFX という名前を出している。これは Win32 API の Longhorn 版と思われるが、それに対応してアプリケーション開発方法が変更される可能性もある。

WinFX
Professional Developers Conference 2003 のプレゼンテーション資料より



Longhorn の新機能の一つである Avalon を見ると、次のようになっており、大きく拡張されることが見て取れる。また、Avalon は画面描画に関してハードウェアの要求スペックがあがるという情報もある。

Avalon
Professional Developers Conference 2003 のプレゼンテーション資料より



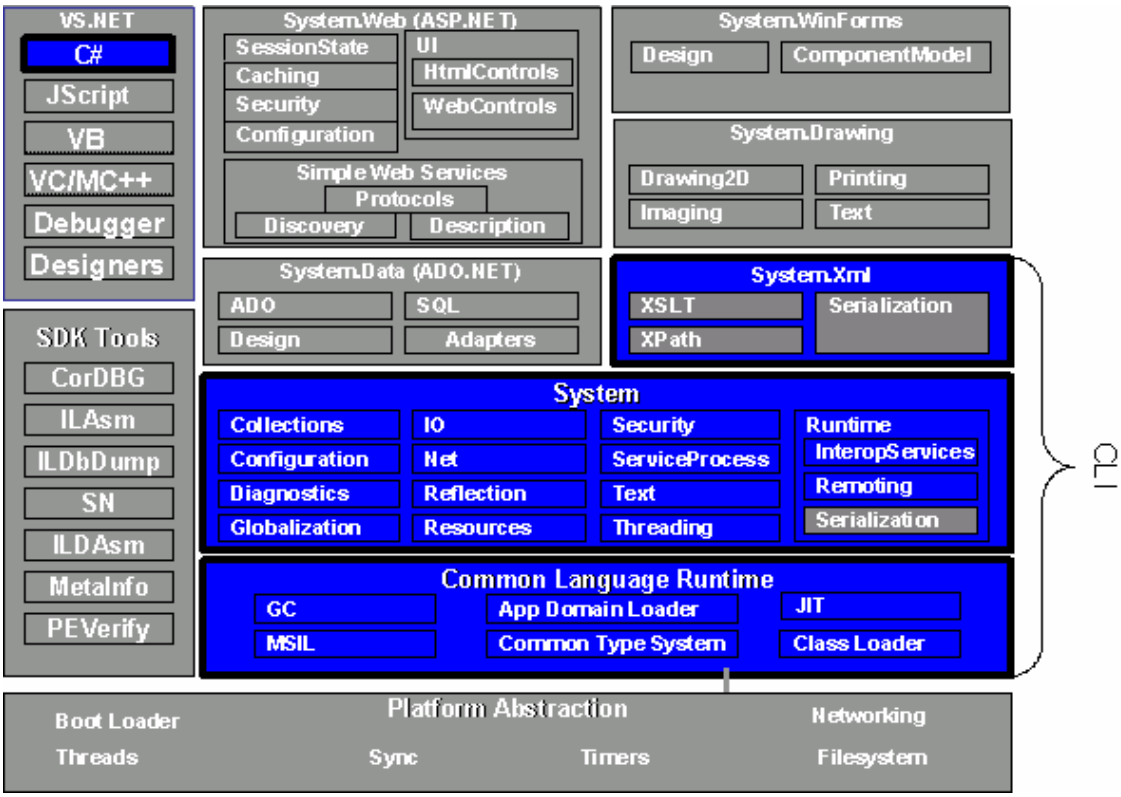
このように、.NET Framework をターゲットとして開発を開始した.NET 互換オープンソースプロジェクトではあるが、そのターゲットが大きく前進しようとしている。また、必要とするハードウェアに制限がある場合、そのハードウェアが例えば Linux でサポートされているかどうかといった、.NET 互換オープンソースプロジェクト自体では解決できない課題も出てくる可能性がある。

7.3. 特許・著作権等

.NET 互換オープンソースプロジェクトは、基本的に ECMA 標準に基づき開発を行っているが、しかし、ECMA 標準だけでは現在のマイクロソフト.NET の範囲をカバーできない。

下図に再掲するように、ECMA 標準（太線内）だけでは、データベースアクセスを行う ADO.NET、Web アプリケーションを開発するための ASP.NET、GUI を開発するための System.WinForms など、重要な機能が欠けている。

ECMA 標準で記述された範囲
（http://research.microsoft.com/programs/europe/rotor/workshop.aspx）



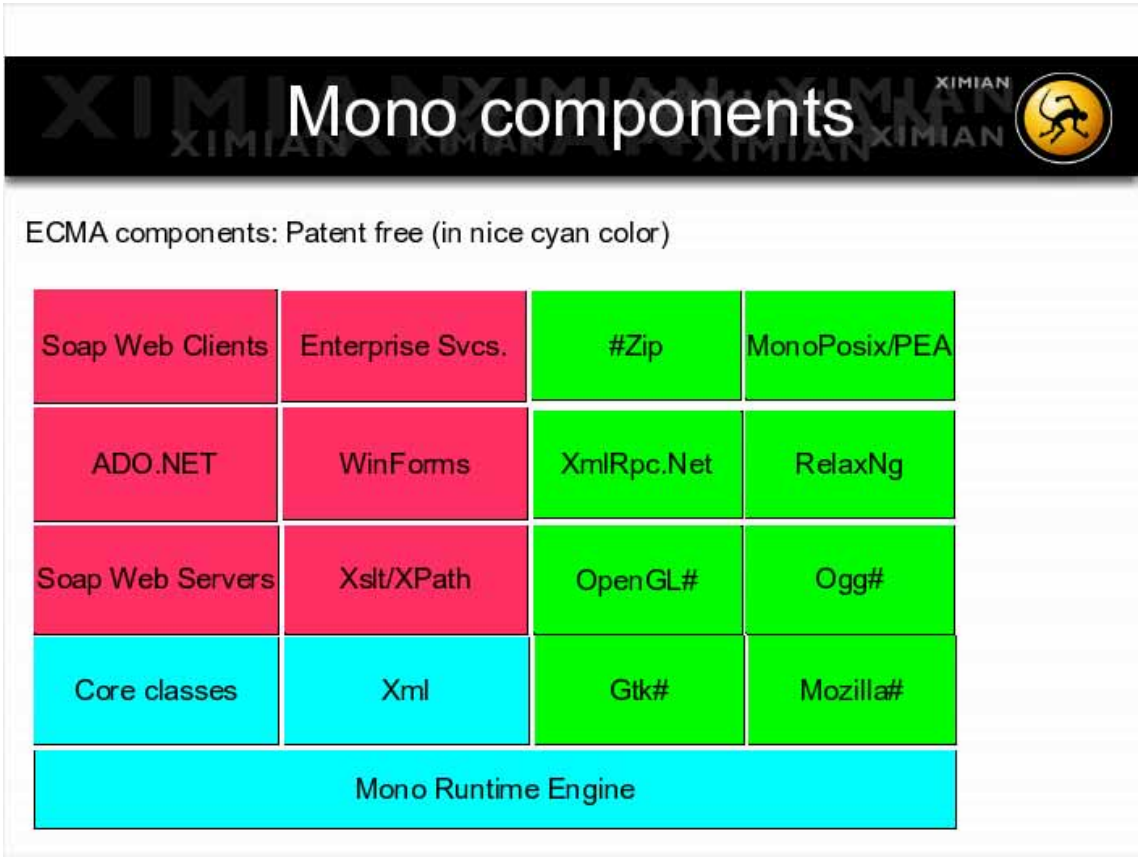
そのため、マイクロソフト社が公開したドキュメントなどを参照しながら開発することになると思われるが、それでも不明な場合、例えば Rotor プロジェクトのソースコードを参照する可能性があるかもしれない。もしそうした場合は、Rotor プロジェクト自体はマイクロソフト社が行っているものであり、著作権の問題が発生する可能性もある。

また、マイクロソフト社が.NET Framework 内の機能に関して特許を取得している可能性があり、それに抵触する場合にどのようにするかといった問題もある。

次図に示す mono プロジェクトのドキュメントに、特許に関してわかっていることがまとめられている。この図は mono プロジェクトのコンポーネントについてマイクロソ

フト.NET の特許の問題が起こる可能性を示している。緑色の部分（#Zip, MonoPosix/PEA, XmlRpc.Net, RelaxNg, OpenGL#, Ogg#, Gtk#, Mozilla#）はマイクロソフト.NET とは直接関係がないコンポーネントである。水色の部分（Core classes, Xml, Mono Runtime Engine）は ECMA 標準に基づき実装している部分であり、ECMA に提出したものについて特許権を行使しないとすれば、特許の問題は起こらない。残りの赤色の部分（Soap Web Clients, Enterprise Svcs., ADO.NET, WinForms, Soap Web Servers, Xslt/XPath）については、特許の問題が起こり得る。

mono プロジェクト FAQ より
（http://www.go-mono.com/faq.html）



7.4. 開発の継続性

.NET 互換オープンソースプロジェクトに限らず、オープンソースプロジェクト全般において、プロジェクト自体の継続性を疑う意見がある。これは要するに、.NET 互換オープンソースプロジェクトの開発者が興味を持って行っていることなので、興味が無くなった時や、他に興味深い対象が出てきたときには開発が継続されないのではないか、という懸念である。

これに対する一般的な回答は、もしそのような事態になったとしても、ソースコードは公開されているので、他の開発者が引き継いで開発すればよいというものである。

.NET 互換オープンソースプロジェクトのうち、mono プロジェクトについて言うと、開発リーダが属する会社である Ximian（現在は、Novell に買収された）が、mono プロジェクトの成果を用いてビジネスを行うと言っているということもあり、オープンソースプロジェクトだからという理由は特にないと思われる。

8. 参考・引用文献、URL

1. Microsoft.NET 入門―これだけ知っておけばなんとかなる.NET の基礎知識 .NET プログラミングシリーズ アスキー書籍編集部
2. よくわかる.NET テクノロジーのすべて デビッド チャペル(2002/06) インプレス
3. Microsoft .NET テクノロジガイド マイクロソフト公式解説書 David S. Platt, (2001/09) 日経 BP ソフトプレス
4. Mono Kick Start (KICK START) Hans-Jurgen Schonig, Ewald Geschwinde (2003/09/15) Macmillan Computer Pub
5. Developing on Linux With C# and .Net: The Mono Project Daniel Solin (2003/10/15) Springer-Verlag New York Inc
6. .NET Framework ホームページ
   http://msdn.microsoft.com/netframework/
7. Rotor プロジェクト解説
   http://research.microsoft.com/programs/europe/rotor/workshop.aspx
8. マイクロソフト社による ECMA 標準化のページ
   http://msdn.microsoft.com/net/ecma/
9. Standard ECMA-334 C# Language Specification 2nd edition (December 2002)
   http://www.ecma-international.org/publications/standards/Ecma-334.htm
10. ISO/IEC 23270:2003 Information technology -- C# Language Specification
    http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=36768&ICS1=35&ICS2=60&ICS3=
11. Standard ECMA-335 Common Language Infrastructure (CLI) 2nd edition (December 2002)
    http://www.ecma-international.org/publications/standards/Ecma-335.htm
12. ISO/IEC 23271:2003 Information technology -- Common Language Infrastructure
    http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=36769&ICS1=35&ICS2=60&ICS3=
13. Technical Report TR/84 2nd edition Converts CLI XML specification into MS-word files
    http://www.ecma-international.org/publications/techreports/E-TR-084.htm
14. ISO/IEC TR 23272:2003 Information technology -- Common Language Infrastructure -- Profiles and Libraries
    http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=36770&ICS1=35&ICS2=60&ICS3=

15. .NET Framework 公開されている仕様のリファレンス
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/netstart/html/cpframeworkref_start.asp
16. mono プロジェクト ホームページ
http://www.go-mono.com/
17. mono プロジェクト FAQ
http://www.go-mono.com/faq.html
18. mono プロジェクト ロードマップ
http://www.go-mono.com/mono-roadmap.html
19. mono プロジェクト ステータス
http://www.go-mono.com/class-status.html
20. DotGNU プロジェクト ホームページ
http://www.gnu.org/projects/dotgnu/index.html
21. DotGNU Portable.NET の概要
http://www.gnu.org/projects/dotgnu/pnet.html
22. DotGNU Portable.NET ステータス
http://www.gnu.org/projects/dotgnu/pnetlib-status/
23. Rotor プロジェクト ホームページ
http://research.microsoft.com/Collaboration/University/Europe/RFP/Rotor/
24. IDE SharpDevelop プロジェクト ホームページ
http://www.icsharpcode.net/OpenSource/SD/Default.aspx
25. Improve C# Plugin for Eclipse プロジェクト ホームページ
http://www.improve-technologies.com/alpha/esharp/
26. IKVM.NET プロジェクト ホームページ
http://sourceforge.net/projects/ikvm/
27. Winfessor 社 Soapbox Framework
http://www.winfessor.com/portal/DesktopDefault.aspx?tabid=28
28. Tipic 社
http://www.tipic.com/
29. OpenLink Software 社 Virtuoso Crossplatform Universal Server
http://www.openlinksw.com/virtuoso/index.htm
30. Source Gear 社
http://www.sourcegear.com/
31. SkyNET プロジェクト
http://sky-net.sourceforge.net/index.php

32. ASP.NET ホスティングサービス業者一覧
http://www.microsoft.com/japan/msdn/asp.net/hosting/default.aspx
33. 有限会社アースリンクネットワークのホスティングサービスページ
http://www.eln.ne.jp/rental_server/hosting/hosting_top.aspx
34. PDC2003 レポート
http://www.atmarkit.co.jp/fwin2k/insiderseye/indexpage/insiderseye_index.html
35. .NET 概要
http://msdn.microsoft.com/netframework/technologyinfo/overview/
36. Rotor プロジェクト リリース概要
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/mssharsourcecli2.asp
37. Rotor プロジェクト ライセンス
http://msdn.microsoft.com/msdn-files/027/001/901/ShSourceCLIbetaLicense.htm
38. Virtuoso 開発者インタビュー
http://idevnews.com/IntegrationNews.asp?ID=45

9. 付録

9.1. .NET 概要

.NET 概要としてマイクロソフト社が公開している情報を以下に添付する。

http://msdn.microsoft.com/netframework/technologyinfo/overview/

What Is Microsoft .NET?

Microsoft .NET is software that connects information, people, systems, and devices. It spans clients, servers, and developer tools, and consists of:

- The .NET Framework 1.1, used for building and running all kinds of software, including Web-based applications, smart client applications, and XML Web services—components that facilitate integration by sharing data and functionality over a network through standard, platform-independent protocols such as XML (Extensible Markup Language), SOAP, and HTTP.
- Developer tools, such as Microsoft Visual Studio® .NET 2003 which provides an integrated development environment (IDE) for maximizing developer productivity with the .NET Framework.
- A set of servers, including Microsoft Windows® Server 2003, Microsoft SQL Server™, and Microsoft BizTalk® Server, that integrates, runs, operates, and manages Web services and Web-based applications.
- Client software, such as Windows XP, Windows CE, and Microsoft Office XP, that helps developers deliver a deep and compelling user experience across a family of devices and existing products.

What Is the .NET Framework?

The .NET Framework is an integral Windows component for building and running the next generation of software applications and Web services. The .NET Framework:

- Supports over 20 different programming languages.
- Manages much of the plumbing involved in developing software, enabling developers to focus on the core business logic code.
- Makes it easier than ever before to build, deploy, and administer secure, robust, and high-performing applications.

The .NET Framework is composed of the common language runtime and a unified set of class libraries.

Common Language Runtime

The common language runtime (CLR) is responsible for run-time services such as language integration, security enforcement, and memory, process, and thread management. In addition, the CLR has a role at development time when features such as life-cycle management, strong type naming, cross-language exception handling, and dynamic binding reduce the amount of code that a developer must write to turn business logic into a reusable component.

Class Libraries

Base classes provide standard functionality such as input/output, string manipulation, security management, network communications, thread management, text management, and user interface design features.

The ADO.NET classes enable developers to interact with data accessed in the form of XML through the OLE DB, ODBC, Oracle, and SQL Server interfaces. XML classes enable XML manipulation, searching, and translations. The ASP.NET classes support the development of Web-based applications and Web services. The Windows Forms classes support the development of desktop-based smart client applications.

Together, the class libraries provide a common, consistent development interface across all languages supported by the .NET Framework.

Note: The .NET Compact Framework does not ship natively with the .NET Framework. Developers may access the .NET Compact Framework using Visual Studio .NET 2003.

Adoption Momentum

The .NET Framework has been live since version 1.0 was released in January 2002. It has achieved numerous adoption milestones:

- Compilers for over 20 programming languages are available for use with the .NET Framework.
- Over 350 tools are available from third-party vendors to aid in .NET Framework development, including approximately 250 add-ins for Visual Studio .NET, as well as IDEs from Borland and Macromedia.
- Over 350 books have been published or soon will be published discussing software development with the .NET Framework.
- Over 750 .NET Framework user groups exist worldwide.
- Millions of users every month visit the .NET Code Wise Community Web sites.
- Over one million developers are using Visual Studio .NET.

- Thousands of leading companies, from Autodesk to Credit Suisse First Boston to Honeywell to Xerox, are realizing tremendous cost savings, new opportunities for integration, and improved time-to-market by developing and deploying their applications with the .NET Framework.
- Microsoft is aggressively deploying applications built using the .NET Framework. MSN®, Microsoft CRM, Windows XP Media Center Edition, and the Microsoft.com Smart 404 are just a few of the many Microsoft applications already built using the .NET Framework.

Rapid Development

The multiple-language capability of the .NET Framework enables developers to use the programming language that is most appropriate for a given task and to combine languages within a single application. Components written in different languages can consume functionality from each other transparently, without any extra work required from the developer. Support for the .NET Framework has been announced for over 20 commercial and academic programming languages.

The component-based, plumbing-free design of the .NET Framework minimizes the amount of code developers have to rewrite and maximizes potential for code reuse.

In the Industry

"With the capabilities provided by [the .NET Framework], we'll be 25- to 50-percent more productive in delivering new solutions. Every aspect of the software development life cycle is made easier, from the creation of user interfaces to debugging and deploying the solution." *Brandie Lerner, team leader/manager, Pfizer, Inc.*

Improved Operations

The .NET Framework improves the performance of typical Web applications.

- The Middleware Company, founders of the leading J2EE developer forum TheServerSide.com, have conducted a benchmark of the .NET Framework and J2EE, finding the .NET Framework to significantly outperform J2EE for Web application hosting, Web services, and distributed transactions, as shown in the graphs below.
- The .NET Framework also offers significant performance and scalability benefits over the previous Active Server Pages (ASP) technology, thanks to its just-in-time (JIT) compilation and caching technologies.

The results below were achieved with both Windows 2000 Advanced Server and Windows Server 2003 running versions 1.0 and 1.1 of the .NET Framework, respectively. Both setups used a SQL Server 2000 database.

In the Industry

"We get subsecond page loads while handling millions of page views a day. We deployed on December 23, 2000 and haven't had a minute of downtime as of October 3, 2001, and we saved $1.3 million (US) over a Java 2 Enterprise Edition solution." *Stephen Forte, chief technology officer, Zagat Survey*

"Compared with similar projects in the past, we're measuring deployment time in hours instead of weeks." *Ferdy Khater, director of application development, Continental Airlines*

Agile Architecture

Companies worldwide are using the XML Web services communication mechanism that is native to the .NET Framework to integrate quickly and easily with suppliers and customers.

In the Industry

"From our partners' perspective, accessing our content via XML Web services will be far easier than what they've had to go through in the past. They will no longer need to build the infrastructure to import, store, and manage it. When combined with our new flexibility in licensing options, this means we'll have a far more attractive package to offer to prospective partners." *Stephen Forte, chief technology officer, Zagat Survey*

"This makes it easier for us to inform portals and enterprises about how our code handles user data, security concerns, and integration with existing databases. Particularly handy are the automatically generated documentation and test Web pages, which enable our partners to integrate their systems with ours using minimal assistance." *Tore Lode, senior developer, CyberWatcher*

Vibrant User Community

Numerous user groups and discussion lists exist around the world on a variety of topics and in a multitude of languages, including English, Japanese, German, and Spanish. More information can be found on the MSDN developer community pages and the .NET Code Wise Community site, which provide access to newsgroups, chats, user groups, and other opportunities to interact with developers who are interested in Microsoft products and technologies.

More than 350 publications covering the .NET Framework and programming languages for the .NET Framework are either currently available or will soon be released. Some highlights include:

- *.NET Framework Essentials*, O'Reilly Press
- *Professional ASP.NET*, Wrox Press Ltd.

- *Visual Basic to Visual Basic .NET*, Sams Publishing

For more books and articles, visit the MSDN Developer Bookstore, or go to the .NET Books site.

For training and events, visit the .NET Framework Training and Events page.

References, Links, Sources, and More Information

To obtain the latest version of the .NET Framework, visit the Downloads for the .NET Framework page.

To learn about new features in version 1.1 of the .NET Framework, check out What's New in the .NET Framework 1.1.

Language compilers that support the .NET Framework have been announced for the following programming languages:

Supported programming languages		
APL	Fortran	Pascal
C++	Haskell	Perl
C#	Java Language	Python
COBOL	Microsoft JScript®	RPG
Component Pascal	Mercury	Scheme
Curriculum	Mondrian	SmallTalk
Eiffel	Oberon	Standard ML
Forth	Oz	Microsoft Visual Basic®

Follow the links below for more information on key .NET Framework topics:

- **Microsoft .NET Language Partners**

  Get information on compilers, controls, and components for use with the .NET Framework.

- **Microsoft .NET Pet Shop 2.0**

  Compare the performance of .NET and J2EE implementations of a J2EE reference application for building highly scalable Web applications.

- **XML Web Services**

  Visit the XML Web services developer center on MSDN®.

Customer Solutions Built on the .NET Framework

Honeywell: **Using XML Web Services, Honeywell Seamlessly Integrates E-Commerce Sites**

The current pace of business, with growing mergers and reorganizations, creates both opportunity and challenges. Honeywell International is no exception. Honeywell's Automation and Control Solutions (ACS) division needed to provide a central authentication service that could integrate the legacy systems of the seven e-commerce sites under its responsibility. Using Visual Studio .NET and the .NET Framework, ACS developed a best-of-breed, business-to-everyone portal providing comprehensive access to profiles for employees, customers, partners, and affiliates. The solution, which took just four weeks to code, provides users with access to back-end systems through an integrated Web solution.

USATODAY.com: **USATODAY.com Moves Toward Dynamic Publishing Model and Saves $400,000 US Annually with .NET**

Using Visual Studio .NET and the .NET Framework, USATODAY.com produced its Automated Fronts application. ASP.NET made the development of the application easy and quick, and staffers were able to use the time that they saved to add more features to the application. The use of ASP.NET allowed USATODAY.com to introduce dynamic Web pages without having to scale its hosting infrastructure, while the .NET Framework COM interop technology simplified integration with the legacy system and allowed USATODAY.com to proceed with the migration to the .NET Framework at its own pace. Automated Fronts represents an initial step toward a fully dynamic publishing model and is expected to save 7,800 hours annually (valued at an estimated $400,000 US per year)—a labor savings that can be reinvested into the overall quality of USATODAY.com.

Dresdner Kleinwort Wasserstein: **Investment Banker Achieves Performance and Uptime, Retains Clients with Microsoft .NET**

Dresdner Kleinwort Wasserstein (DrKW) provides a wide range of services and information to its international clientele. Offering timely market data and analysis is key to retaining corporate clients for whom such information can be worth millions and even billions of dollars. To provide an easy way for its clients to securely access valuable data, DrKW developed its BrokerPulse application using Visual Studio .NET and the .NET Framework, all in just three months. With its excellent performance, BrokerPulse lets DrKW clients access valuable data with just a Web browser, anytime and from just about anywhere.

Ingram Micro: **Leading Global Wholesaler Builds Next-Generation E-Commerce Application in Record Time**

Ingram Micro developed the latest version of its IMPipeline business-to-business e-commerce storefront using Visual Studio .NET and the .NET Framework. The company estimates that it was able to build and deploy the application in half the time that it would have taken before the move to .NET. Featuring a data management solution built on SQL Server 2000, IMPipeline uses Web services to make valuable product,

customer, and order information readily available to reseller partners in a format that is easy to integrate into their own storefronts. Ingram Micro anticipates continued growth and is confident that Microsoft and .NET-connected technologies offer the best strategy to meet the company's global expansion requirements and scalability issues.

Bank of New York: **The Bank of New York European Fund Services Group Switches Mission-Critical Fund Servicing Solution to Microsoft .NET, Cuts Time-to-Market by 40 Percent**

Switching to Visual Studio .NET and the .NET Framework for the next major version of its fund servicing solution is delivering several strong benefits for The Bank of New York European Fund Services Group, including a 40-percent decrease in time-to-market, significant development cost savings, and a richer, more responsive user experience through the smart client capabilities of Windows Forms.

9.2. mono プロジェクトロードマップ

mono プロジェクトのロードマップを以下に添付する。

http://www.go-mono.com/mono-roadmap.html

Mono Project Roadmap

Miguel de Icaza (miguel@ximian.com)
Last update: Jan 18th, 2004

Introduction

This document describes the high-level roadmap for Mono.

The Mono project started in 2001 as an effort to implement the .NET Framework to Unix. To bring both the new programming model based on the Common Language Infrastructure and C# as well as helping people migrate their existing knowledge and applications to Unix. Mono today supports a wide variety of operating systems, CPUs and a large chunk of the functionality available in the .NET Framework.

At the October 2003 PDC conference a number of new technologies were announced. From the Mono release schedule perspective, we should think about these technologies from their release time standpoint, and the features that must be supported.

This document outlines the roadmap for the Mono project from my perspective: what we can effectively deliver on the dates outlined. Since Mono is a large open source project, things might change and new features can be incorporated into the plan if external sources devote enough attention to those problems.

This is the timeline:

Background

So far Microsoft has published two versions of the .NET Framework: 1.0 and 1.1, the later with incremental updates to the changes in 1.0

The Mono project has been tracking some of the improvements available in those two releases, some of the highlights of our work so far are:

- Core: mscorlib, System, System.Security and System.XML assemblies.
- ADO.NET: System.Data and various other database providers.
- ASP.NET: WebForms and Web Services are supported. Work on WSE1/WSE2 has also started.
- Compilers: C#, VB.NET and various command line tools that are part of the SDK.
- Open Source, Unix and Gnome specific libraries.

Other components like Windows.Forms, Directory.Services, Enterprise Services and JScript are being developed but are not as mature as the other components but are under development by various people.

Some other smaller and less used components do not have yet a Mono equivalent (System.Management, System.Drawing.Design).

Mono release strategy

The levels of maturity of Mono fluctuate depending on the development effort we have put into it, and the use we have given to them. For example, the virtual machine and the C# compiler very mature, while less commonly used functionality in Mono like Windows.Forms or VB.NET are still under heavy development.

Our strategy is to release the mature components as Mono 1.0, and have upcoming versions of Mono add extra functionality.

Mono 1.0 goals

The Mono 1.0 release would include the following components:

- C# compiler.

- VM, with JIT and pre-compiler.
- IL assembler, disassembler.
- Development and security tools.
- Core libraries: mscorlib, System, System.XML.
- System.Data and Mono database providers.
- System.Web: Web applications platform and Apache integration module.
- System.Web.Services: client and server support.
- System.Drawing.
- System.DirectoryServices
- JIT support: x86, SPARC and PPC architectures (interpreter available for other architectures).
- ECMA profiles: special build options to build Mono as an implementation of the various ECMA profiles will be available.
- Java integration through IKVM.
- Embedding interface for the runtime.

Packaging:

- mono: will contain the above features implementing the .NET 1.1 API.
- mono-1.0-compat: Will include a build of the libraries with the .NET 1.0 API, this is a compatibility build for people running .NET 1.0 applications.
- mono-unstable: Will contain a snapshot of the other technologies under development for developer's convenience, but will be unsupported at this time. These include the Generics edition of the C# compiler.
- mono-ecma: A build that only includes the ECMA components.

Release target: Q2/2004.

Bug fix releases would be done on a monthly basis.

For a detailed list, see the mono-1.0 feature list.

Microsoft's Whidbey

To understand post 1.0 editions of Mono, it is important to put it into perspective with the Microsoft Whidbey product, to be released in 2004.

The new features in the Whidbey release of the .NET Framework include:

- **Generic types**
  These introduce changes to the compiler, runtime and class libraries.
- **ASP.NET 2**
  Many tools to simplify web application development: Master pages, new controls for common operations, personalization and themes.
- **Remoting**
  New security channels and version-resistant remoting (good news in the interop department).
- **XML**
  XQuery debuts on this release as well as an improved XmlDocument system based on XPath: XPathDocument.
- **Networking**
  FTP client, Ssl streams.
- **Console and Serial ports:**
  Console terminal input/output is available as well as serial port handling.
- **Windows.Forms**
  Layout containers finally appeared on Windows.Forms as well as various new controls.
- **ObjectSpaces**
  An API for simpler data-base access.

Mono 1.2

The Mono team is developing in parallel some features that wont make it to the 1.0 release in stable form. These will be the foundation for the 1.2 release. The focus of this release is to track the core API for the .NET Framework 1.2, but again, only a subset of the total framework will be available.

Mostly, Mono 1.2 consists of components that were not stable enough for Mono 1.0, but that would be mature at this point, plus the incorporation of some new features from Whidbey. In addition to the Mono 1.0 components, this release will add:

- Generic types support: C# compiler, execution system and core class libraries.
- ASP.NET 2.0 improvements.
- Remoting improvements from Whidbey.
- System.XML: simpler improvements from Whidbey, lacking the large additions (XQuery for example).
- Console and Serial ports support.
- New compilers: VB.NET and JScript support.
- WSE1/WSE2 implementations.
- System.Windows.Forms officially debuts with .NET 1.0 API; 1.2 API available as an unstable addition.

This release will by default provide .NET 1.2 APIs, but compatibility libraries for 1.0 and 1.1 will be distributed in the mono-compat package. The unstable components of Mono will be distributed on the `mono-unstable' package, the libraries in this release will be unsupported.

Release target: Q4/2004.

Mono 1.4

A refresh update on the Mono 1.2 release containing the missing components from the previous release and complete any under performing pieces. Updates to System.Xml, ASP.NET and Windows.Forms to match the .NET 1.2 API.

Release target: Q2/2005.

Peer projects

Other projects like the debugger, the documentation browser, Java integration through IKVM and Gtk# will remain on their own schedules.

This page will be updated to contain that information when it becomes available.

Unscheduled activities

A missing component of Mono is the Code Access Security (CAS). This functionality is not needed in today's Mono as currently Mono is being used to run fully trusted applications, and we are not using it on embedded scenarios where assemblies would have different trust levels.

This is an important component, but requires three major pieces of work:

- Runtime support for implementing the security demands.
- A guidelines document outlining what and where must have security demands in place.
- A full audit of our class libraries: method by method

All of these are major pieces of work, and we currently have no plans to implement any of those. A volunteer effort might be able to help with the runtime requirements and the document, but until those are done, we are unlikely to start doing any work on the actual class library audit and instrumentation.

Mono and WinFX: 2006

WinFX is the name given to the new set of libraries that makes up .NET in the Longhorn operating system: the existing .NET set of class libraries, plus the new functionality available in the OS.

WinFX adds things like storage facilities (WinFS), a new versatile communications stack (Indigo) and a new eye-candy packed GUI programming system (Avalon).

Although WinFS, Avalon and Indigo are very exciting components, at this time it is too early to tell when those components will be available for Mono. Open source developers will very likely start work on these,

but since they are still far from being officially released, they are not in the radar at this point.

Unsupported technologies

Some technologies are very hard to implement or are being phased out by components in the Longhorn time frame. In some cases, we feel that they are not crucial to the future of the open source desktop.

System.EnterpriseServices and System.Management come to mind, and we are unlikely to put any resources into the task. We would gladly host the code if someone cares to implement it, but they would likely remain unsupported features of Mono.

Mono Developer Strategy

Mono Developers should read the Mono Hacking Roadmap

Comments

Feel free to send your comments or questions the roadmap to miguel@ximian.com

Last Updated: Nov 1st, 2003

9.3. DotGNU（Portable.NET）概要

DotGNUプロジェクトにおける.NET互換環境開発プロジェクトであるPortable.NETについて、その概要を添付する。

http://www.gnu.org/projects/dotgnu/pnet.html

DotGNU Portable.NET

The goal of DotGNU Portable.NET is to build a suite of Free Software tools to compile and execute applications for the Common Language Infrastructure (CLI), which is often referred to as ".NET".

The initial target platform was GNU/Linux, but DotGNU Portable.NET has been known to work under Windows, NetBSD, FreeBSD, Solaris, and MacOS X, amongst others. It also runs on a variety of CPUs including x86, PPC, ARM, Sparc, s390, Alpha, ia-64, and PARISC.

Focus on Compatibility

DotGNU Portable.NET is focused on compatibility with the ECMA-334 and ECMA-335 specifications for C# and CLI, and with Microsoft's commercial CLI implementation. Our main goal is to make it easy to write portable application programs which work well both on DotGNU Portable.NET and on Microsoft's .NET platform.

In addition, we want to make sure that many application programs which were written for Microsoft's .NET platform (with no consideration for portability) will work well with DotGNU on many operating systems.

Installing

DotGNU Portable.NET can be installed either from a binary package or from source code (recommended).

Once you have installed DotGNU Portable.NET, you may want to install DGEE. Also, check out the demo and sample programs.

Runtime engine

The runtime engine in DotGNU Portable.NET, called "ilrun", is used to interpret programs in

the Common Intermediate Language (CIL) bytecode format, described in the ECMA-335 specifications.

Because interpreting CIL bytecode directly is quite inefficient, we take a different approach. We first convert the CIL bytecode into a simpler instruction set for what we call the Converted Virtual Machine (CVM). The simpler CVM instructions are then executed using a high-performance interpreter. A description of the CVM instruction set can be found here.

The CVM approach gives us many of the benefits of a Just-In-Time compiler (JIT), in that the opcodes can be tailored to handle system differences (e.g. 32-bit vs 64-bit CPU's). At the same time, the engine's source code is highly portable to new platforms.

As another optimization on some CPU's (currently x86 and ARM), we further translate CVM opcodes into native machine code for direct execution. This gives an additional performance improvement with only a small coding effort required.

We will eventually write a full JIT for DotGNU Portable.NET, but we are currently focused on completeness and stability.

See the file "**pnet/engine/HACKING**" in the source code for further information on the structure of the runtime engine.

Compiler

The DotGNU Portable.NET compiler, cscc, is a modular compiler system with good support for the C# and C programming languages. Work on support for some other languages (Java and VB.NET) has been started. The C# front-end implements the ECMA C# Language Specification ECMA-334. The C language front-end implements ANSI C.

Implementing additional languages is aided significantly by the innovative "Tree Compiler Compiler" (treecc) program, which uses aspect-oriented programming techniques to manage the complexity of compiler construction. See the treecc documentation for more information on this tool.

The compiler is designed to support bytecode generation for multiple bytecode systems. The current bytecode back-end supports the Common Language Infrastructure (CLI) defined byECMA C# Language Specification ECMA-335. Other back-ends to support the

Java Virtual Machine (JVM) and the Perl 6 Parrot engine are in progress.

System.Windows.Forms

An advantage of our implementation of System.Windows.Forms is that we don't try to wrap up third party widget sets like Gtk, Qt, Wine, etc. Instead, we provide a basic drawing layer and then render the controls ourselves. The approach is similar to Java Swing, in that all controls are implemented in pure C#.

This approach should allow us to emulate the Windows visual appearance and behavior more closely and portably than other approaches because we don't need to work around the quirks in foreign toolkits. Our implementation has been known to run on x86, PPC, and ARM based GNU/Linux systems, as well as Mac OS X. See the DotGNU Web site for current screenshots.

The DotGNU Portable.NET Forms implementation is structured into three layers, which are found in the source directories **System.Drawing**, **System.Drawing/Toolkit**, and **System.Windows.Forms**.

System.Drawing provides the basic drawing functionality, emulating the Windows GDI layer as faithfully as possible. When drawing to a control, use the definitions in **System.Drawing.Graphics**.

System.Drawing/Toolkit defines an interface to primitive drawing toolkits. Toolkits provide simple line/text/etc drawing (**IToolkitGraphics**), plus a simple window mechanism (**IToolkitWindow**).

There may be multiple toolkits in the system, each providing drawing functionality in a different manner. The current toolkits include **System.Drawing.Xsharp**, which wraps around the DotGNU Portable.NET **Xsharp** library; and **System.Drawing.Win32**, which wraps up the native Win32 API under Windows.

System.Windows.Forms builds upon the primitive drawing and window facilities from **System.Drawing** and **System.Drawing/Toolkit** to implement the various controls, forms, dialogs, etc, that are defined by the Forms API.

The file "**pnetlib/System.Windows.Forms/HACKING**" in the source code contains more

information on developing for our implementation of System.Windows.Forms.

We are looking for volunteers to help us finish the **System.Windows.Forms** implementation, and are offering cash prizes. See the competition page for more details.

Further information

For further information on DotGNU Portable.NET, please refer to the FAQ of the Portable.Net development project. "Latest Changes" information with information on source code packages released by the project is available here.

9.4. Rotor プロジェクト リリース概要

マイクロソフト Rotor プロジェクト（Shared Source CLI とも呼ばれる）のリリースの概要を以下に添付する。

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dndotnet/html/msshar sourcecli2.asp

The Microsoft Shared Source CLI Implementation Refresh

Stephen R. Walli
Microsoft Corporation

June 2002

Applies to:

- Microsoft® .NET Framework
- Microsoft® Windows® XP
- Microsoft® JScript®

Summary: The refreshed Microsoft Shared Source CLI Implementation source archive unpacks and builds on both the Microsoft Windows XP and FreeBSD operating systems and provides improvements to the Rotor experience, including better debugging, documentation and samples, build environment and tools enhancements, and bug fixes. (3 printed pages)

Download the Shared Source CLI (sscli_20020619.tgz) from the MSDN Download Center.

The team delivering the Microsoft® Shared Source CLI implementation has just released a refresh of the beta code base on MSDN.

On March 26, 2002 at a conference in Cambridge, England, Microsoft released the source code to a complete working implementation of the ECMA C# and Common Language Infrastructure (CLI) standards. These standards specify the core of the Microsoft .NET Framework. At this event, Microsoft also issued a research request for proposal. Since then, more than 30,000 people have downloaded the shared source CLI (code-named "Rotor") and a lively community has developed around the code base. (You can find links to the discussion groups on the download page listed above). In addition, we received more than 100 requests for proposal and

accepted more than 30 of them. The Microsoft Research group in Cambridge, UK is facilitating this work, and will be hosting a workshop this summer to bring together the various Rotor projects to share information.

Like the previous source archive, the refreshed source archive unpacks and builds on both the Microsoft Windows XP and FreeBSD operating systems. This refresh also provides lots of extras to improve the Rotor experience. There are changes in more than 500 files, as well as another 1000 new files in the archive bringing the archive total to 8722 files. (This doesn't include all the extra class reference documentation in a separate archive.)

The changes to this source archive refresh fall into four broad categories:

- Debugger improvements
- Improved documentation and samples
- Enhancements to the build environment and tools
- General clean-up and bug fixing

Several things have been improved on the debugging front. The managed debugger (cordbg) now works on FreeBSD (and is improved on Windows XP as well). The SOS command plug-in for gdb that allows class and stack structures to be dumped now takes case-insensitive commands. A new tool (ildbdump) that dumps Rotor .ildb files has been added to the tool chain (clr/src/tools/ildbdump). Microsoft JScript support to emit symbolid debug information has been added.

Several new samples have been added to the samples directory. Look for the xsd schema conversion tool sample and the dnprofiler managed profiler sample in the samples directory.

A big change happened in the documentation space. Along with new documents matching the changes, this source refresh delivers class reference documentation as part of the distribution. This documentation set is large enough that it has been added in its own compressed archive for separate download.

One of the strengths of the Shared Source CLI is its build and test infrastructure. The Beta refresh includes a number of improvements in the build system:

- The build tool (build.exe) parsing of csc.exe and gcc's output has been improved so the build*.err log file is more informative about errors.
- Two new tools, permview and genpalunitable, are now available. (These are SDK tools. The genpalunitable tool is used to generate part of the Unicode support for the FreeBSD Platform Adaptation Layer. The output from this tool was shipped in the March beta release, but the tool itself is now shipped in this refresh.)
- The test suite driver rrun.pl now conforms to the different types of build environments (checked, fast checked, free) and test executables are built into appropriately named directory hierarchies as well as the test logs.
- A considerable number of new test cases have been added to the test suites.

Finally, a lot of code clean-up and bug fixing has gone on in this release. The JIT has been re-factored and builds much faster. Code pitching in the JIT has also been improved. Bugs across the entire code base including cordbg, the base class library and frameworks, the FreeBSD PAL, execution engine, JIT, and so on, have been fixed.

We continue our commitment to our growing community with this refresh of the Shared Source CLI implementation and we are very excited on the Rotor team to watch the community develop and the source base evolve in new directions. I encourage you to download the latest Shared Source CLI archive and explore the possibilities it presents to experiment with and learn about the internals of the Common Language Infrastructure. Happy hacking!

You can read more about the Microsoft Shared Source CLI in MSDN Magazine

9.5. Rotor プロジェクト ライセンス

mono プロジェクトや DotGNU プロジェクトは知名度の高い GPL、LPGL、X11 ライセンスなどを使っているので省略し、Rotor のライセンスのみ添付する。原本は、以下の URL を参照。

http://msdn.microsoft.com/msdn-files/027/001/901/ShSourceCLIbetaLicense.htm

MICROSOFT SHARED SOURCE CLI, C#, AND JSCRIPT LICENSE

This License governs use of the accompanying Software, and your use of the Software constitutes acceptance of this license.

You may use this Software for any non-commercial purpose, subject to the restrictions in this license. Some purposes which can be non-commercial are teaching, academic research, and personal experimentation. You may also distribute this Software with books or other teaching materials, or publish the Software on websites, that are intended to teach the use of the Software.

You may not use or distribute this Software or any derivative works in any form for commercial purposes. Examples of commercial purposes would be running business operations, licensing, leasing, or selling the Software, or distributing the Software for use with commercial products.

You may modify this Software and distribute the modified Software for non-commercial purposes, however, you may not grant rights to the Software or derivative works that are broader than those provided by this License. For example, you may not distribute modifications of the Software under terms that would permit commercial use, or under terms that purport to require the Software or derivative works to be sublicensed to others.

You may use any information in intangible form that you remember after accessing the Software. However, this right does not grant you a license to any of Microsoft's copyrights or patents for anything you might create using such information.

In return, we simply require that you agree:

1. **Not to remove any copyright or other notices from the Software.**

2. That if you distribute the Software in source or object form, you will include a verbatim copy of this license.

3. That if you distribute derivative works of the Software in source code form you do so only under a license that includes all of the provisions of this License, and if you distribute derivative works of the Software solely in object form you do so only under a license that complies with this License.

4. That if you have modified the Software or created derivative works, and distribute such modifications or derivative works, you will cause the modified files to carry prominent notices so that recipients know that they are not receiving the original Software. Such notices must state: (i) that you have changed the Software; and (ii) the date of any changes.

5. THAT THE SOFTWARE COMES "AS IS", WITH NO WARRANTIES. THIS MEANS NO EXPRESS, IMPLIED OR STATUTORY WARRANTY, INCLUDING WITHOUT LIMITATION, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OR ANY WARRANTY OF TITLE OR NON-INFRINGEMENT. ALSO, YOU MUST PASS THIS DISCLAIMER ON WHENEVER YOU DISTRIBUTE THE SOFTWARE OR DERIVATIVE WORKS.

6. THAT MICROSOFT WILL NOT BE LIABLE FOR ANY DAMAGES RELATED TO THE SOFTWARE OR THIS LICENSE, INCLUDING DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL OR INCIDENTAL DAMAGES, TO THE MAXIMUM EXTENT THE LAW PERMITS, NO MATTER WHAT LEGAL THEORY IT IS BASED ON. ALSO, YOU MUST PASS THIS LIMITATION OF LIABILITY ON WHENEVER YOU DISTRIBUTE THE SOFTWARE OR DERIVATIVE WORKS.

7. That if you sue anyone over patents that you think may apply to the Software or anyone's use of the Software, your license to the Software ends automatically.

8. That your rights under the License end automatically if you breach it in any way.

9. Microsoft reserves all rights not expressly granted to you in this license.

9.6. Virtuoso 開発者インタビュー

mono プロジェクトを組み込んだサーバである Virtuoso の開発者のインタビューを以下に添付する。

http://idevnews.com/IntegrationNews.asp?ID=45

Virtuoso Expands UDA Features with ASP.NET

12/30/2002 | Print Version

by **Vance McCarthy**

OpenLink Software, Inc., a provider of UDA (Universal Data Access) and web services middleware, will include support for ASP.NET in its Virtuoso 3.0 Universal Server, slated to ship in Q1 2003.

OpenLink has a more than decade-long heritage of helping developers tie into legacy data, providing both JDBC and ODBC tools with Virtuoso, their multipurpose, multiprotocol server virtual database for SQL and non-SQL data. More than a simple staging server, Virtuoso offers developers and users links between different data stores. With the coming release of Virtuoso 3.0, the company will push into the realm of data integration and web services, bringing support for ASP.NET and SQL-to-XML data communications and conversions.

Integration Developer News spoke with Kingsley Idehen, OpenLink's president/CEO, to discuss the features and strategy behind Virtuoso 3.0, which will enable developers and DBAs to create XML web services, user-defined types, stored procedures, functions and triggers. With Virtuoso 3.0's support for ASP.NET and Mono (the Open Source project to bring Microsoft's.NET framework to Linux and Unix), developers will be able to use any .NET-bound language on Windows, Linux, Mac OS X and a variety of UNIX platforms, Idehen said.

To learn more about how OpenLink will use existing and new technologies to deliver on this vision of the "developer-as-integrator," check out the full interview.

An *Integration Developer News* Interview with Kingsley Idehen, CEO and President, OpenLink

IDN: Before we discuss your support for ASP.NET and Mono, let's get some

perspective on your company. What's the target for Virtuoso?

Idehen: Virtuoso is a multipurpose, multiprotocol server virtual database for SQL and non-SQL data. With Virtuoso, we look to provide developers and users a single connection to provide links to different databases. We also provide ability to convert SQL –to XML.

IDN: So, web services is a natural expansion of that focus?

Idehen: We're adding a web services element using SOAP as a standard invocation method, to transparently make SQL stored procedures invocable as SOAP services and describable by WSDL. With our Virtuoso 3.0, we also wanted to extend the platform to work with other platforms, and that's where Mono comes in.

IDN: And what shape does your support of .NET and Mono take?

Idehen: We host Mono's .NET CLR on Windows to homogenize the heterogeneous applications logic. Using Mono helps us make that logic invocable by SOAP. Also, as increasing number of technology providers provide .NET bindings for such legacy systems as COBOL and other languages, the developer will get more flexibility. All Virtuoso will have to do is glue our functionality to those [.NET resources], and developers won't have to learn a lot of new things about new environments. We have been in the connectivity space for years, supporting many Microsoft ODBC and Java JDBC drivers.

IDN: Could you explain precisely how the CLR will help you deliver on your vision?

Idehen: With ASP.NET and Mono [combined], that gives us the ability to let developers' .aspx applications and code be platform independent, and work with many varieties of Unix and Windows.aspx.

IDN: How important would you say Mono and ASP.NET are to that vision for Virtuoso?

Idehen: It's a very, very strategic piece of our road map going forward.

IDN: Is it so strategic that Virtuoso will be committed only to .NET as an enabling

technology?

Idehen: With Virtuoso, we want to give developers an option. Freedom is an important element for our value proposition, so you have an option to say, 'I want it with CLR hosting (.NET with Windows and Mono for non-Windows). At the same time, I can have Virtuoso hosting with binaries for .NET or Java.'

IDN: At this point in technology development, do you have a preference between .NET or Java in conjunction with Open Source, cross-platform options such as Mono?

Idehen: .NET is the one with the most solid road map. I see .NET over the long haul adversely affecting Java; I fundamentally believe the technology is superior. The existence of Mono as a project speaks volumes of the technical road map of .NET versus Java.

You're taking a lot of functionality that already exists today in the Visual Basic world and C++ world, and all of sudden this will compete with the Java world. The only problem with all these developer technologies is that you could use it in a Windows world. Now, you can use it all, through Mono, in a non-Windows world. The ability to take a bindery from Windows and make it work somewhere else is where Mono is so strong.

IDN: What are some key benefits that Mono and ASP.NET bring Virtuoso?

Idehen: With Mono's support for ASP.NET, all you do, with .aspx files, is load them into Virtuoso files. Without ASP.NET support, you would have to write the engine, write the renderer -- you would have to be able to render those .aspx files. And then you would have to have a C# compiler. And, you'd have to have a mechanism for parsing the .aspx pages. You can also support ASP code, as well, because ASP.NET supports ASP.

IDN: What are some specific advances you see from ASP.NET, compared with options, like PHP, for instance?

Idehen: We've been doing database integration support for years, and if developers

want dynamic database-driven pages, and they want a flushed-out IDE, that's where Visual Studio has some strong advantages over PHP.

IDN: Can you elaborate on that?

Idehen: There are a lot of PHP-driven sites out there, but 90 percent of those sites are talking to MySQL, or aren't really doing any heavy database work. When you get to the ASP.NET world, there's a much richer underlying database-driven model, so with today's ASP or .aspx page, when it comes to database centricity, the .aspx page is doing much more sophisticated work than the PHP page.

As the need to develop more sophisticated database-driven web pages increases, especially with data accessing and the ability to talk to different databases or to communicate with web services in the background, an IDE that can bring that together coherently will increase developer productivity. And I think those kind of tools, available cross-platform because of Mono, will create a groundswell because the conventional Linux or Unix developer now realizes they are no longer confined.

IDN: So, do you think Mono's bundling of ASP.NET, and implementations like Virtuoso will affect how developers think about building dynamic web pages or web services?

Idehen: It will have a ripple effect, I think. But there's another development from Microsoft coming next year that will add to that.

IDN: What's that?

Idehen: Microsoft has come up with something they call the Web Matrix. It's essentially a stripped-down version of Visual Studio .NET, and it's written in C#, so it's all managed classes. It's the equivalent of a Java IDE built 100% in Java, so there are no non-Java components. All I need is to take to my classes to wherever there is a JVM, and my IDE will manifest itself.

A typical Linux developer may not touch Visual Studio.NET, let's say, because he doesn't touch Windows -- period. But with Web Matrix, which is a VS.NET replica with managed classes, you now have an IDE for both the Windows and the non-Windows

world. All that will be needed is the completion of support for Windows Forms, which will come from WINE, another Open Source project.

IDN: Let's talk specifically about Virtuoso 3.0. Will Virtuoso 3.0 have Mono bundled with it from the outset? And will it be branded as "Mono" so developers will know you have Open Source with your product?

Idehen: Virtuosos 3.0 will have Mono in it, and that's how it will be branded. Our initial beta release is for Linux, and then we'll have the Windows bindings and then Mac OS X. After that, we'll then take it to other platforms we've worked on for years, such as Solaris and HP Itanium.

Virtuoso 3.0 with Mono/ASP.NET will be available in commercial beta in January 2003 from OpenLink Software.

Inside Mono's ASP.NET Support

"Although we had ASP.NET in the prototype version, we have now integrated the System.Web classes," Mono creator Miguel de Icaza told *IDN*. "Now, we've done it the way Microsoft has designed it."

In part, the new upgrade, called Mono 0.17, comes with "many new System.Data providers and a more mature System.Web (ASP.NET), which can now be hosted in any web server," according to de Icaza's release notes. A simple test web server to host ASP.NET has been released, as well. The Mono version 0.17, which is the first Open Source support for Microsoft's ASP.NET development environment, can be downloaded here.

この事業は、競輪の補助金を受けて実施したものです。