.NET なら

～Java との比較～

平成１６年２月１０日

大分県情報サービス産業協会
技術研究会 ドットネット部会

目次

1. はじめに

現在の企業の IT 戦略には、より効率的な情報活用のためにさらなる変化が求められているが、その中で少なからずとも Web サービスへとパラダイムの移行が行われている。それは「いつでも・どこでも」情報をコンピュータで扱うことを求めて、より柔軟により高度なサービスを提供する仕組みを必要としているからである。

そこで、本部会では Web サービスが構築できる.NET について、長所を探り出す研究を行った。

.NET とは、Microsoft 社が進める次世代 Windows およびアプリケーションやシステムといった Microsoft 社プラットフォームをベースとするアーキテクチャの総称であり、Web サービスを実現するための仕組みの中心である。

また、Sun Microsystems 社が進める Java は Web サービスを構築する上で、「.NET」と同じような効果をもたらすことはすでに知られている。

我々は、同じような特徴を持つといわれる Java との比較を、機能や特徴、動向などの多方面から行った。さらには、実際に.NET 上で作成された Web アプリを Linux 上の.NET フレームワークに移植することで、.NET におけるアプリ開発の実態調査を行った。

本論文は、比較や調査の結果を元に.NET で Web サービスを構築する場合の長所について述べる。

2.　.NET の目指すところ

2.1　.NET と Java のアーキテクチャ

2.1.1　.NET Framework の構成について

.NET Framework は CLR (Common Language Runtime)とクラスライブラリで構成されるソフトウェア実行環境のフレームワークで、Java における VM(Virtual Machine)と Java クラスライブラリの組み合わせのようなものである。

.NET Framework の内部は、図のように大きく 3 つの要素から構成されている。最下層にある CLR は、アプリケーションやコンポーネントを実行するためのエンジンである。そしてアプリケーションのシステム・インターフェイスとなるクラスライブラリ群がその上にある。ASP.NET は、Web サービスと Web アプリケーションを実装するためのクラスライブラリである。

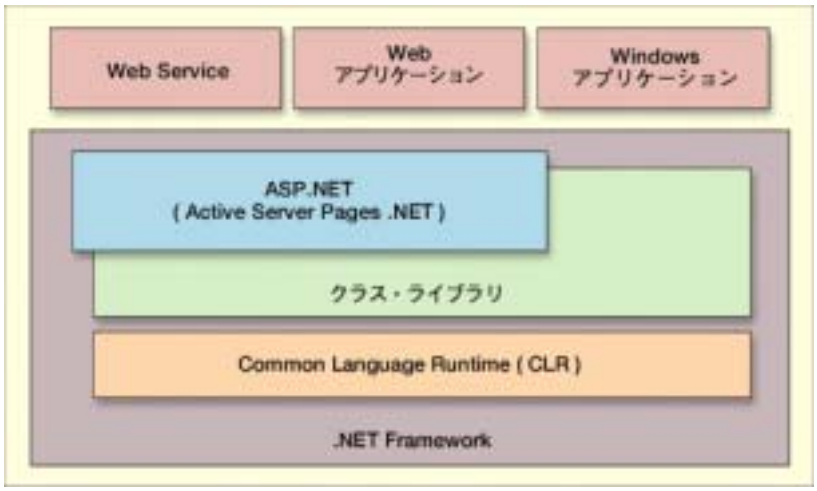

図２.１　　.NET Framework の構成

2.1.2　CLR について

2.1.2.1　CLR の構成について

CLR は.NET Framework 上のプログラムの実行管理を行い、プログラミングを補助する様々な機能を提供する。CLR 上で動作するコードは、マネージコードと呼ばれ、CLR が提供する異言語間のオブジェクトの互換性、例外処理、セキュリティー機能、バージョン管理といった機能の上で構築されている。

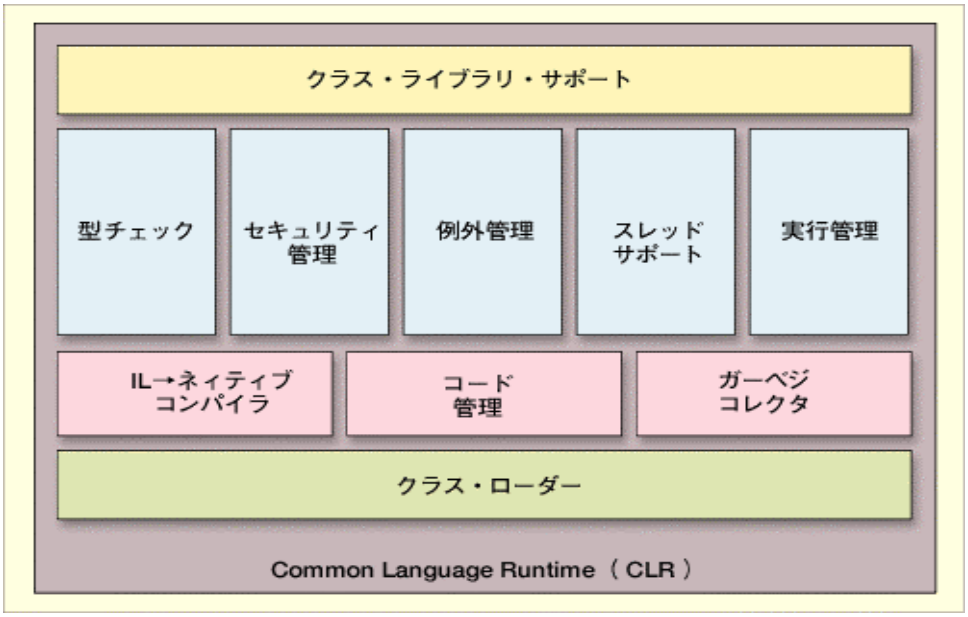

図２.２　CLR の構成

以下に CLR の主要な特徴の一部を示す。

（１）　複数言語への対応
　　プログラム開発に複数の言語の中から最適な言語を選択できる。

（２）　メタデータの採用
　　プログラム自身がメタデータを持ち、自己記述的なオブジェクトとなることで、プラットフォームからの独立が高まり、そのコードの再利用性も高まる。

（３）　ガーベージコレクションとメモリ管理
　　ガーベージコレクション、メモリ管理といった機能を CLR が提供することで、ポインタにまつわる様々な、エラーの発生が無くなる。

（４） DLL 地獄の解消

従来の DLL 問題は、バージョンが異なる、同一名称の DLL が複数要求された場合、DLL の互換性が維持できず、場合によっては、動かないケースもある。CLR ではこれを解消するために、バージョンの異なるアセンブリのサイドバイサイド実行がサポートされた。これにより、バージョンが異なるアセンブリが必要となる場面でも、それぞれのプログラムは自分が必要とする適切なバージョンのアセンブリを使用することができる。

上記の中でも最も注目すべき点は複数言語が使用できる機能である。CLR が言語に依存しない共通のランタイムライブラリとして存在するため、.NET Framework が対応するプログラミング言語であれば、相互に呼び出しが可能となる。IL コードの実行までの流れとその仕様である CLS についての説明を後述する。

2.1.2.2 JIT コンパイラと実行手順

既に Java で取り入れられている技術であり、Java の実行方式は、コンパイルにより中間コードであるクラスファイルを生成し、JVM（Java Virtual Machine）が解釈しながらプログラムを実行する。一方、.NET では、各プログラム言語に対応したコンパイラにより中間コードが生成された後、JIT コンパイラにより CPU に対応したネイティブコードにコンパイルされる。Java での実行方式のように、全てのコードを一気にコンパイルするのではなく、呼び出すメソッドごとにコンパイルして、CPU 資源の浪費を防ぐといった機能も提供している。

CLR 上でのソースコードのコンパイルから、生成された MSIL(Microsoft Intermediate Language)コードの実行までの流れを図示すると次のようになる。

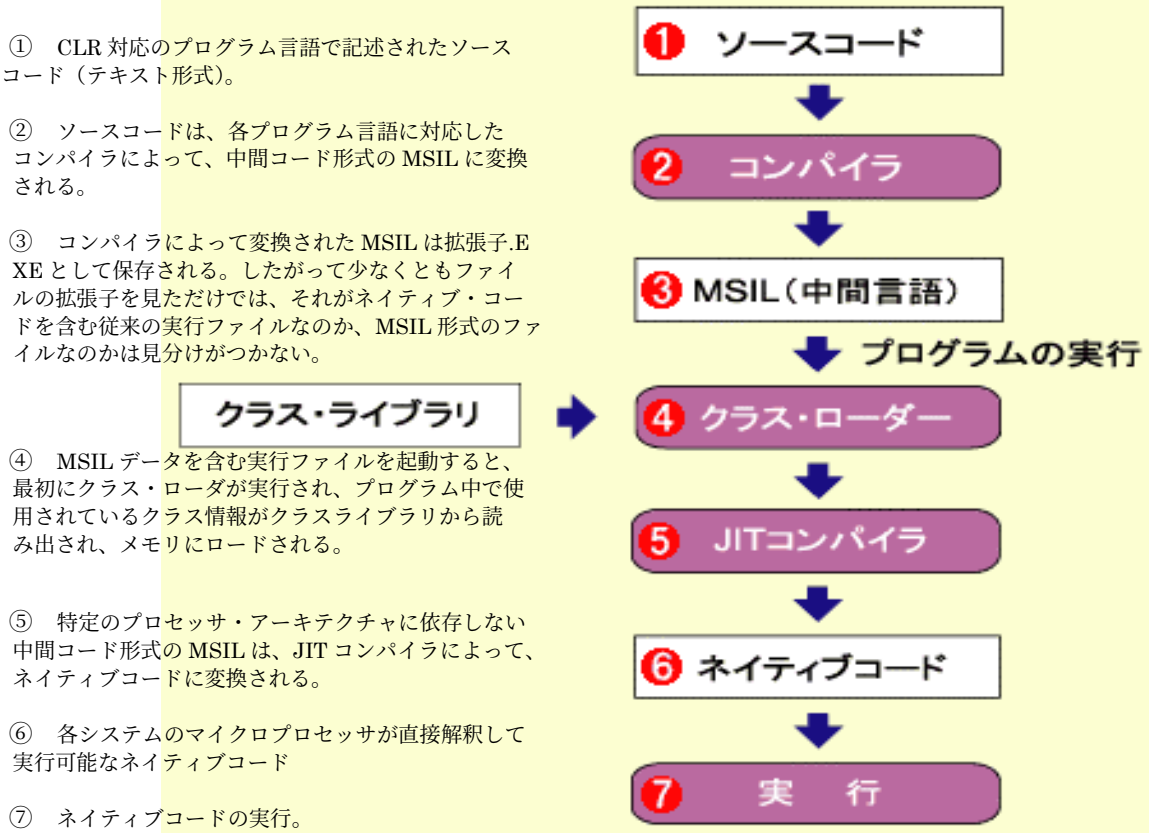

図２.３　アプリケーションの開発から実行まで

2.1.2.3　CLS

CLS（Common Language Specification）は共通言語仕様と言われるもので、異なるプログラミング言語で作成されたプログラム間の相互運用性をサポートするために定められた規約である。以下、この CLS を説明していく。

2.1.2.3.1　CLR を実現させる規約

CLS を説明するにあたり、CLR の言語仕様と CTS について触れる必要がある。まず、CLR の言語仕様について説明していく。以下の３点は CLR の主な言語仕様であり、またそれに伴い定義すべき規約について列挙している。

（１）CLR にネイティブな言語は IL である

.NET では、その上で動作するプログラムは基本的に IL コードの形で、CLR が解釈、実行する。そのため、CLR と IL 間で整数や、浮動少数点数などについてデータ型を共通に規定する必要がある。

（２）CLR は複数言語に対応した実行環境である

CLR では、複数の言語を用いてプログラミングすることが可能である。しかし CLR にネイティブな言語は（１）の通り、IL のみであり、その上の層に位置する各プログラミング言語では、IL でサポートされるデータ型、インストラクションなどの要素と、自らの言語が持つデータ型、文法などとの間でのすり合わせが必要となる。

これらの要素が共通化していれば、異なるプログラミング言語で作成したプログラム間でのデータの受け渡しや、メソッドの呼び出しが可能となる。

（３）CLR はオブジェクトプログラミングをサポートする

.NET ではオブジェクト指向により、プログラムの再利用性を高めている。そのために CLR で、再利用可能なデータ構造の定義、実行時にバインディングされるメソッドの呼び出し、例外処理などの機能を提供する必要がある。これらの機能をその上で動作するプログラム言語が使用するには、CLR との間にやはりすり合わせの必要が発生してくる。

CLS は異なる言語間での相互運用性をサポートする上での規約を定めている。

各プログラムはメタデータという形で、そのプログラムが外部に対して公開するインターフェイスを自己記述する。またデータ型やメソッド呼出形式などは CTS に準拠したサブセットを定め、これを遵守させることで、異なるプログラミング言語間での互換性をもたせている。

CLS が規定するのは、あくまでプログラムが外部に公開するインターフェイス部のみであって、その実際の中身は CTS のサブセットとプログラム間の互換性についての規約である。

つまり、CLS により.NET Framework 上で動作する各プログラミング言語同士の相互運用の実現が可能となっている。

上記に述べたとおり.NET では、様々な言語において、同一の実行環境が保証されるため、作成されたプログラムはプログラミング言語の違いによるクオリティの差が無くなる。

.NET プラットフォームでは、このように数多くのプログラミング言語の中から、プログラマが自分の最も得意な言語を選択してもよいし、開発対象

から最適な機能性を持った言語を選択してもよい。また単一のシステムであっても、場面に応じて異なる言語処理系を使用し、それらを組み合わせることも可能である。具体的には、異なる言語間でのクラスの継承が可能である。これにより、例えば、C#で記述したクラスを VisualBasic.NET で継承してサブクラス化するといった使い方ができる。また Visual Studio.NET を使えば、このように複数言語を組み合わせたソフトウェアでも、ソース・コードレベル・デバッグが可能になる。

これはある意味で Java とは対極的な考え方である。Java では、Java 言語でコードを記述しておけば、Java 向けの仮想マシン環境が用意された任意の環境でプログラムを実行できることを特徴としている。しかし逆に言えば、Java でコードを記述できなければ、その恩恵に浴することができない。Java の言語仕様自体はシンプルなものだが、複雑なクラスライブラリを使いこなすのは簡単ではないし、オブジェクト指向設計に不慣れなプログラマも少なくない。このため Java を活用した情報システムのニーズは非常に高いが、それに応えられる SE の絶対数が不足しているのが現実である。この点.NET プラットフォームなら、クラスライブラリに対する理解などが必要となるものの、従来から使い慣れたプログラミング言語を使いながら、最新の環境に対応したソフトウェアを開発できる。

上記の点から、CLR の中核となる機能を実現するためにはこのような規約が必要となることが分かる。この細かい規約の役割を担っているのが次に述べる CTS（Common Type System）である。

2.1.2.3.2 CTS

CLR はプログラム言語に関して規約を必要とする。CTS はこの規約の役割を担っており、プログラム言語間で共通の基本データ型、データ型定義、命名規約などを定めている。つまり、CTS は CLR をサポートするプログラミング言語がどういうタイプシステムを採用するべきか記述するものである。

では.NET 上で動作するすべてのプログラミング言語はこの CTS に完全に準拠しなければならないのか、となるとそうではない。もちろん、先にのべた三点の要件を満たすためには完全な準拠が必要であるが、その中で例えば IL のみをターゲットとした言語であるならば、それは必要ではない。

しかし、CLR が異なる言語で作成されたプログラム間（CTS 完全準拠のものとそうではないもの）での相互運用をサポートするためには、さらなる規約が必要となる。それが CLS である。

2.1.3 .NET と Java のアーキテクチャの違いについて

.NET と Java のアーキテクチャは、非常によく似ているが以下の 2 点が大きな違いと思われる。

⑴ 開発言語の種類

Java は文字通り、開発言語が Java に限られる。しかし、.NET は、CLR が言語の違いを吸収してくれるため、Visual Basic.NET や C#・Cobol など様々な言語に対応している。

⑵ 使用可能な OS の種類

Java は Java 仮想マシンにより、OS の違いを吸収し、UNIX や Linux・Windows など異なる OS に対応している。一方、.NET は、OS が Windows に限定されていた。しかし、昨今 Mono の出現により、Linux 上でも使用可能となっている。

2.2 .NET の目指すところ

.NET とはコンピューティングとインターネットを融合することで、次世代のインターネット環境を構築するというコンセプトである。従来のインターネットの問題を根本から解決し、ユーザーと開発者の両方に大きな恩恵をもたらすものと提唱されている。

2.2.1 現在の企業連携の限界とインターネットが抱える諸問題の解決

ビジネスにおいてインターネットの活用が急拡大している。しかし現在のインターネットにはまだ数々の問題があり、ビジネスインフラとしての重責を負えるものにはなっていない。電子的な企業間取引も、現在ではまだ EDI （Electronic Data Interchange ）が主流となっている。このような状況を打破して n 対 n の企業間連携を行える環境を確立していくことが必要である。

この問題を解決するには企業の情報化戦略はさらに一歩進む必要がある。これまでの企業情報戦略といえば、社内の情報化によって経営スピードを高めていくことに主眼が置かれていた。しかしこれからは、インターネットによって顧客やパートナー企業との関係をより緊密にすることが重要になる。そのためにはこれまで情報システムが提供してきた様々な機能を、社外からも利用できる“サービス”へと進化させる必要がある。

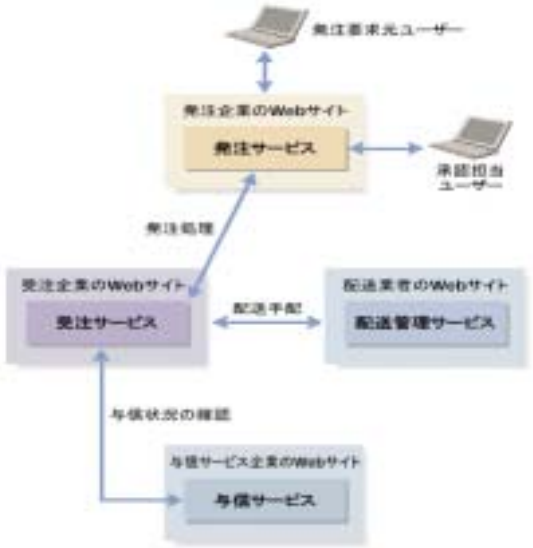

図２.４　受発注業務における Web サービス連携の例

このような企業間のシステム連携は、様々なビジネス領域で大きな効果を発揮する。

2.2.2 Web サービス連携の実現

インターネットが本当の意味でのビジネスインフラになるには、Web サービス間での連携が必要にある。しかし、Web サービスを実現するにはインターネットが現在抱えている諸問題をすべて解消する必要がある。その一つが現在の分散オブジェクト技術の問題である。分散オブジェクト技術は、複数のコンピュータ上で稼動するオブジェクト同士を動的に連携させ、複数コンピュータによる協調処理を実現する。しかし現在の技術では、ファイアウォールの存在によりインターネットを介した協調処理ができない。さらには、システムの一部に過剰な負荷がかかるような構造になっていることも、現在の Web 環境の重大な問題点である。Web ブラウザの登場によって、クライアント側は確かにシンプルになった。しかしその分、Web サーバにかかる負担が大きくなっている。その他にも、開発環境の未整備など、問題は決して少なくない。これらの諸問題を解決する鍵を握っているのが、XML と SOAP と呼ばれる標準技術である。.NET はこの標準技術をベースにしている事により容易に Web サービスの開発を行うことができる。

また、Web サービス間連携の実現だけではなく、他にも新しいプログラミングモデルの提供や多様なデバイスの提供、アプリケーションサービスの提供など、様々な要素が含まれている。これらはすべて現在のインターネットの問題を打破するために欠かせないものである。

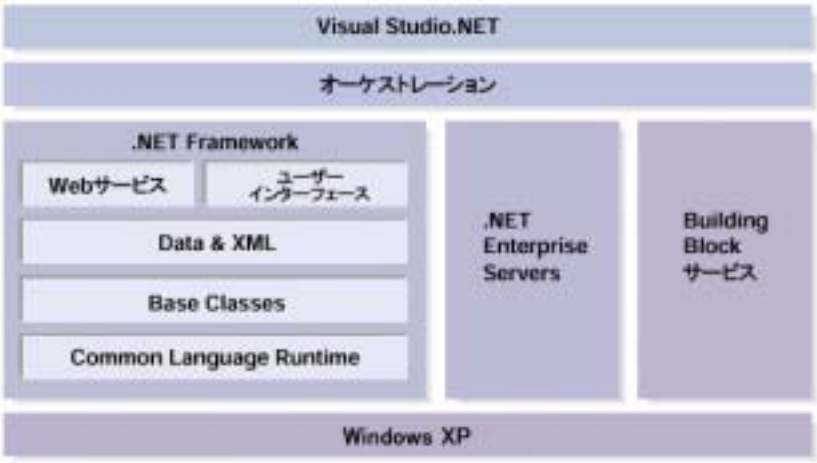

図２.５　.NET の構成要素

XML と SOAP をベースに様々な技術が体系化されている。

2.2.3 Web サービスの容易な作成

Visual Studio.NET では Web Forms によって、従来のデスクトップアプリケーション開発と同じ手法(IDE(Integrated Development Environment)を使った RAD(Rapid Application Development)スタイル)を使って Web アプリケーションを作成することができる。これは従来型の Web アプリケーションを開発している開発者にとっても、生産性の向上という大きなメリットをもたらすだろう。しかしその最大の特長はなんといっても、XML や SOAP に対応した Web サービスを簡単に作成できる点にある。

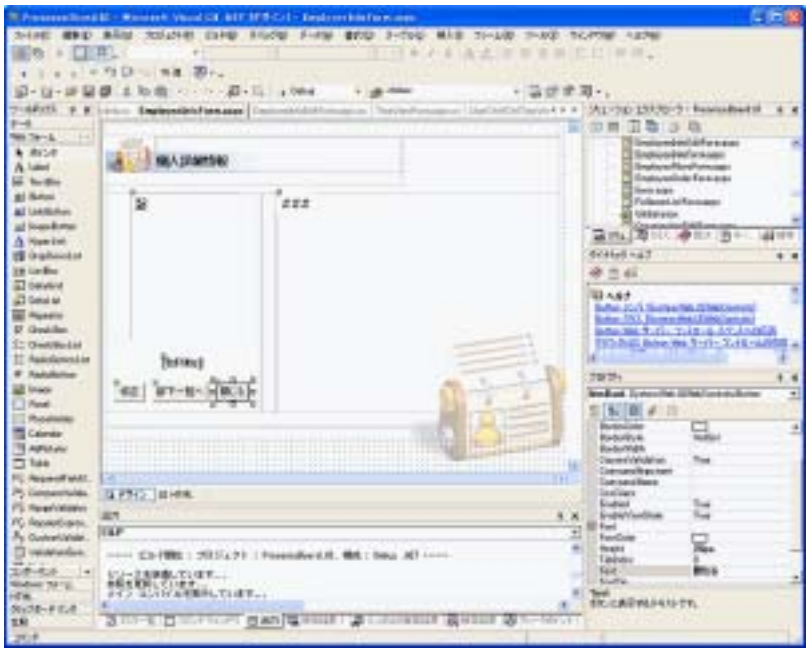

図２.６　Visual Studio.NET による Web アプリケーションの作成

従来の Visual Basic と同様に、必要なコントロールをフォーム上に貼り付けるだけで UI の設計を行うことができる。

2.2.4 .NET 端末の普及

Microsoft 社は 2002 年 2 月 17 日､同社主催の携帯通信向け開発者会議｢Microsoft Mobility Developer Conference」において、携帯端末向けのプラットフォーム「.NET Compact Framework」ベータ版を発表した。同社の「.NET」戦略を携帯端末に拡大するもので、あらゆる機器にリッチなコンピューティング体験を提供するものである。

また、Microsoft 社は併せて、統合開発環境「Visual Studio .NET」を「.NET Compact Framework」に対応させるための拡張ツール「SDE (Smart Device Extensions)」ベータ版を発表した。SDE には携帯用アプリケーションの開発を容易にする、新しいプロジェクトタイプや端末エミュレータ機能、リモートデバッグ機能などを備えている。

現在「Visual Studio .NET」及び「.NET Compact Framework」対応の.NET 搭載の端末としてはカーナビゲーション、Pocket PC などがすでに開発されている。

カーナビゲーションは、2003 年 5 月 2 日 Microsoft 社は車載情報端末向けのソフトウェアプラットフォームの最新版「Windows Automotive」の出荷に伴い、.NET Framework、.NET Compact Framework、XML Web サービスの組み合わせにより、サーバーと車載情報端末のデータ交換および更新が行われるサービスの開発が可能となると発表している。例えば、車載情報端末からレストランの予約だけでなく、同時に割引サービスの適用や予約通知を自動的に受け取ることが可能としている。

実際、2003 年 4 月 24 日に行われた「Microsoft Automotive & Telemetric Conference in Japan」においては.NET Framework を利用したデモとして地図上に表示されたレストランからの予約や、リアルタイムメッセンジャーを利用したメッセージの交換のデモなども行われ、実用化への道を着実に歩んでいる。

また携帯端末では 2002 年ラスベガスで開催された CES2002 において.NET 対応の「次期 Pocket PC」の試作機を発表している。

この Pocket PC では電話アプリケーションが内蔵され携帯電話としても使用できる機能や、Internet Explorer でログインし、録画したいチャネルをサーチなどの機能を用いて効率的に探し録画の設定を行うビデオ録画機能も搭載している。試作機としての発表で利用ユーザーのニーズに若干の乖離があるものの、今後の Microsoft 社の展開において、.NET Framework 1.1 での新機能として、ASP.NET モバイルコントロールによる各種携帯端末や PDA、及び IPv6 対応機能を設けた点や、開発ツールとしてエミュレータ機能を搭載し、モバイル Web 用アプリケーションとして.NET Compact Framework をサブセットとして提供していることから、更なる携帯デバイス需要の獲得を伺っていることが垣間見られる。

そして、PC、ラップトップ、ワークステーション、電話機、ハンドヘルド コン

ピュータ、タブレット PC、Microsoft® XBox™ ゲーム コンソールといったスマートデバイスを XML Web サービスの利用や、テキストの音声変換、手書き文字の認識といった情報提示、収集能力を発展させることによってさらなる.NET 搭載端末の普及を図っていっている。

2.3 .NET と Java の目指すところの違いについて

Sun Microsystems 社は 2002 年 3 月 25 日、Web サービス標準技術を、携帯電話や PDA、セットトップボックス(STB)、車載システムといった携帯端末や軽量端末に拡大する計画を明らかにした。Sun Microsystems 社は、軽量端末向けの Web サービスパッケージを用意し、開発者が統一アーキテクチャによって、サーバーから各種の消費者向け機器に Web サービスを配信できるようにする。

Sun Microsystems 社はこの計画を推進するため、無線向け統合開発環境「Forte for Java wireless toolkit」、ガイドラインとなる「Java blueprint」、開発者コミュニティ「Java Community Process」(JCP)が承認した新しい提案仕様「Java Specification Request」(JSR)をベースとする API をバンドルして出荷する。これにより、開発者が軽量端末向けの Web サービスを容易に開発できるようにする。

また、軽量端末向けの Web サービス仕様「JSR＃172」を提出し、JCP の管理委員会に承認されている。

「JSR #172」は、「XML」や「SOAP」「UDDI」「ebXML」といった Web サービスの標準技術を、インターネット対応の携帯端末や家電機器に拡大するものとなる。これにより、ユーザーが携帯端末向けの Java「J2ME」を搭載する携帯電話や PDA などを用いて、Web サービスと連携できるようになる。

.NET と Java を比較した結果、詳細な部分では細かく異なる箇所が見受けられるが、最終的に目指すところには相違がないように思われる。

なぜなら、様々な機器で Web サービスを利用できる事を目指し、それぞれの動作できる環境を拡大している事に違いはないからである。

そうした中で、.NET を Linux 上で動作させることの出来る Mono の出現は.NET の様々な機器で利用可能となる可能性を示したと言える。

我々は、実際に Mono を使用して Linux 上で.NET を動作させる事を試みた。

詳細については次章で述べることとする。

3. Web サービスの試作

3.1 .NET と Mono について

.NET に対する Java の優位な点としてマルチプラットフォーム対応が挙げられるが、.NET を Windows 以外のプラットフォームで動作させることはできないのだろうか？

現在、Microsoft 社から出ている.NET Framework（CLI 以外の非公開となっている Windows フォームや ASP.NET、ADO.NET を含む）は Windows 上でしか動作させることができない。

しかし、こうした非公開部分も含めた.NET Framework 互換環境の全体を独自に開発している団体がいくつかある。

その中のひとつが、今回取り上げる Mono である。

3.1.1 Mono とは？

Mono は Novell 社の「Mono Project」（※１）で主に Linux 向けに開発されているオープンソース･プロジェクトである。

元々は Ximian 社が手がけていたが、2003 年 8 月に Novell 社によって買収された。Ximian 社はオープンソース Linux デスクトップの「GNOME」の開発企業。

Mono Project では、仕様が公開されていない部分については、振る舞いやクラスに実装されているメンバーを調べて、互換の環境を開発しているようである。

Mono Project は 2001 年 7 月にスタートし、最近はほぼ 2 ヶ月に一度のペースでバージョンが 0.01 ずつ上がっている。2003 年 12 月末の最新バージョンは 0.29 となっている。

2004 年の第 2 四半期頃に「Mono 1.0」がリリースされる予定になっており、これが初めての正式リリースとなる。

Mono 1.0 は、.NET Framework 1.0 や 1.1 との互換性を持つ。さらにこのリリースでは JIT（just-in-time）モードや AOT（ahead-of-time）モードで動作するコードジェネレータを持ち、x86 と PowerPC アーキテクチャをサポートする。そのほかの ARM、SPARC、HPPA、s390 アーキテクチャ向けにはインタープリタが提供される。

ただし、GUI ベースのアプリケーションを開発できるようになるのは Mono 1.2 からとなっている。（※２）

3.1.2 Mono Project 誕生の理由

旧 Ximian 社の CTO（最高技術責任者），Miguel de Icaza 氏によると，このプロジェクトはもともと，GNOME コミュニティと Ximian 社で開発ツールを改善する必要性が高まったことから生まれたものだという。

「このような取り組みを進めているのは，当社のデスクトップアプリケーション，Evolution の開発プラットフォームを“アップグレード”するためだ」と同氏。Evolution は Ximian 社の電子メールクライアントと個人／ワークグループ情報管理機能を備えたアプリケーション。

同氏は「.NET フレームワークを見て，次世代の製品を構築するために利用したいと話していたんだ」と語っている。（※３）

3.1.3 Mono の課題

順調に開発が進んでいる Mono ではあるが、いくつかの問題点も抱えている。

.NET には Microsoft 社の特許が多数絡んでおり、もし Mono が軌道に乗ったとしても Microsoft 社がライセンス料や特許料の支払いを求めるのではないかという懸念がある。これに対して、Novell では.NET 関連で存在する特許の調査のために弁護士を雇い，特許付きの.NET 技術を，既に公開され，特許で保護されていない技術に置き換えるつもりのようだ。

また、デベロッパーコミュニティを広く取り込めるかどうかも課題となっている。例えば、Sun Microsystems 社や IBM は Java にかなりの額を投資しているため、競合しそうなプロジェクトを支持するのはかなり難しいだろう。（※４）

3.1.4 Windows プラットフォーム以外で.NET を動かす事の利点

Windows 上で.NET を使用する利点として、製品間の親和性が高い、アプリケーションサーバーが不要などと言った点が上げられる。また、一般的に Unix サーバーよりも Windows サーバーの方が安価なため、構築費用を安く済ますことができるという利点もある。

反面、Windows プラットフォームに固定されると言うのが大きなデメリットとなる。システム構成の選択肢の幅が狭く、また、Windows プラットフォームが使用できない案件では.NET そのものが使用できない、という事態になりかねない。

しかし、Mono が登場し、.NET が Windows プラットフォーム以外でも動作するようになったことによって、プラットフォーム選択の自由度が格段に広がった。無料の Linux を使用して.NET 環境を構築することができるようになり、初期費用を低くしたい場合には、これに無料のソフトウェアを組み合わせることによって、ハードウェアコストのみでシステムを構築することも可能である。逆に、

UNIX レベルの堅牢性が必要な場合には、高価になってしまうが UNIX をプラットフォームとして選択することもできる。

このように、ハードウェア価格のみの安価な構成から、UNIX を使用した高価な構成まで、Mono の登場によって.NET のシステム構成の幅は飛躍的に広がる。.NET はマルチプラットフォームの点でも Java に見劣りしない環境に発展していっている。

	.NET	Java
開発ツール	VisualStudio.NET（商用） C#Builder（商用） Sharp Develop（フリー） Web Matrix（フリー）等	JBuilder（商用） Eclipse（フリー）等
動作環境（OS）	Windows（商用） Linux（フリー）等	Windows（商用） Linux（フリー） AIX（商用） Solaris（商用）等
動作環境（アプリケーションサーバー）	不要	Tomcat（フリー） Enhydra（フリー） Web Sphere（商用） Interstage（商用）等
動作環境（Web サーバー）	IIS（Windows に付属） Web Matrix サーバー （Web Matrix に付属） XSP（Mono に付属）等	Apache（フリー） Web Sphere（商用） Interstage（商用）等

表 3.1 .NET と Java での Web アプリケーション開発・動作環境

3.2 移植

Mono を採用したアプリケーションを.NET で構築した場合にどのような問題や作業負荷が発生するのかアプリケーションを構築して検証を行った。

作成するアプリケーションは作業期間などを考慮した結果、昨年度の部会が作成したものを採用した。

昨年度はデータをデータベースに保持する方式を採用していたが、Mono はデータベース関連の実装が完了していない等の理由もあり、.NET との親和性の高い XML ファイルにデータを保持する方式に変更する事で、プログラム修正の負荷を調査した。

昨年度作成システムの概要は以下の通りである。

【機能概要】

OISA 部会員の名簿の管理する

【機能詳細】

（1）部会員情報の照会、登録、修正、削除をする。

（2）操作は部会単位で表示された一覧表にて行う。

（3）Window プログラム、およびブラウザから利用可能で、情報はデータベースにて管理する。

3.2.1 XML 化について

3.2.1.1 XML 化に伴う作業

（1）XML ファイルの作成

［テンプレート］から XML ファイルを選び、［開く］ボタンを押すと、新しい XML 文書の編集画面となる。

図 3.1　XML 文書の編集画面

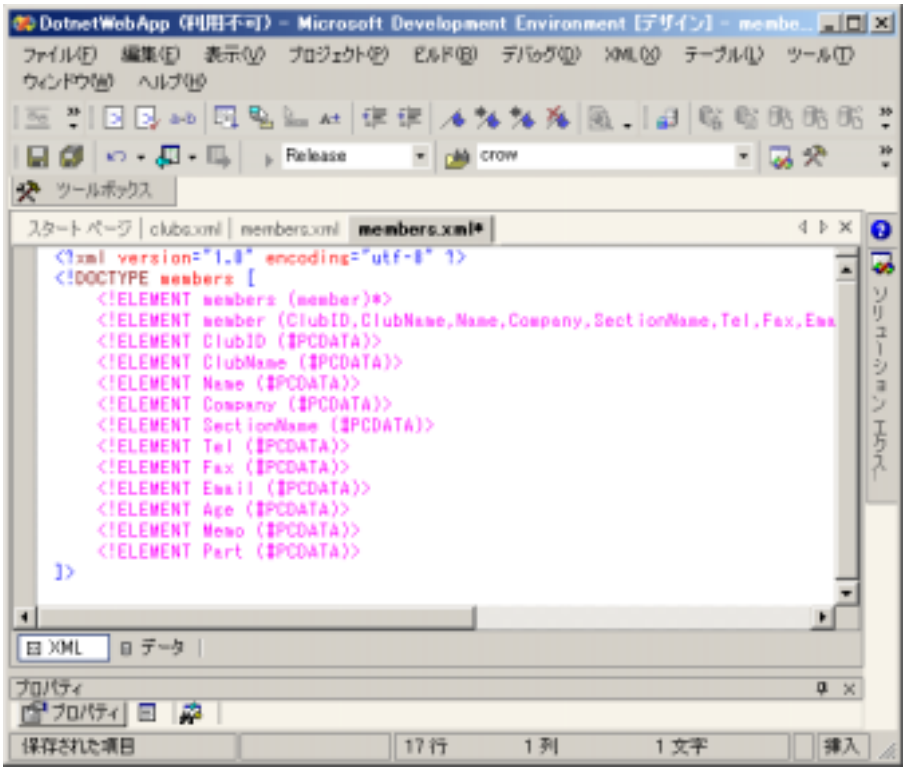

図 3.2　XML 文書の編集

（2） XML 保存ディレクトリに対するユーザアカウントの権限付与

データを保持する XML ファイルへの書き込みを可能にするため、その保存ディレクトリに書き込み権限を付与する必要があった。

ASP.NET では Windows2000 と XP 上では、ASPNET というアカウント、Windows2003 の場合は、Network Service というアカウント上で動作する。

保存ディレクトリに上記のうち該当するアカウントに書き込み権限を与える。

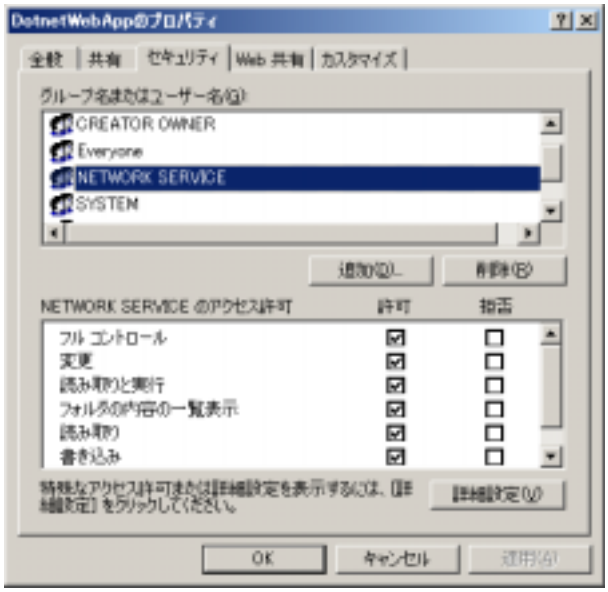

図 3.3　XML 保存ディレクトリに書き込み権限付与

（3） データベース使用部分ロジックの修正

XML ファイルにてデータを管理するよう変更したため、データベース使用部分を、XML ファイルを読み書きするように修正した。

```
/// <summary>
/// [編集]ボタン押下時の処理
/// </summary>
/// <param name="source"></param>
/// <param name="e"></param>
private void DataGrid1_EditCommand(object source, System.Web.UI.WebControls.DataGridCommandEventArgs e)
{
    try
    {
        if ((Label)e.Item.Cells[2].Controls[1]).Text == "")          // Cells[2]:hiddenなIDﾌｨｰﾙﾄﾞ:0:編集, 1:新規
        {
            // XMLファイル使用のため修正
            //MemberListService ws = new MemberListService();
            //DataSet ds = ws.reference(int.Parse(DropDownList1.SelectedItem.Value));
            //ds.Tables[0].Rows.Add(ds.Tables[0].NewRow());
            DataSet ds = new DataSet("members");
            ds.ReadXml(calFile_Member);
            DataSet cloneDs = getMembersOfClub(DropDownList1.SelectedItem.Value, ds);
            DataGrid1.DataSource = cloneDs;
            //DataGrid1.DataSource = ds;
            cloneDs.Tables[0].Rows.Add(cloneDs.Tables[0].NewRow());
            DataGrid1.EditItemIndex = e.Item.ItemIndex;
            DataGrid1.DataBind();
        }
        else
        {
            // 既存行
            DataGrid1.EditItemIndex = e.Item.ItemIndex;
            BindData();
        }
    }
    catch(Exception ex)
    {
        Label1.Text = string.Format("[編集]コマンドでエラーが発生しました。<BR>{0}", ex.Message);
    }
}
```

※コメント行が昨年のコード

図 3.4　C#のソースコード

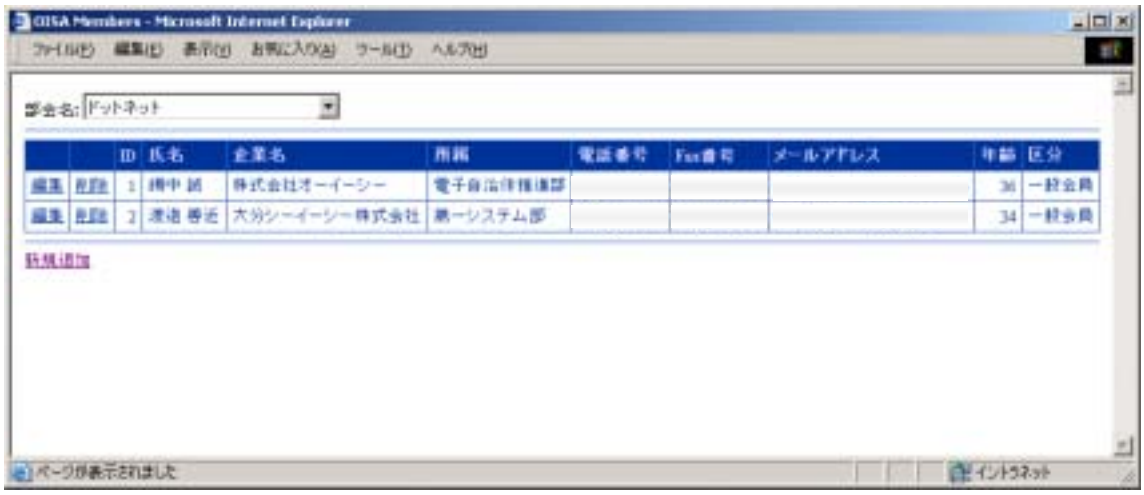

図 3.5　改造前の画面（昨年の状態）

図 3.6　改造後の画面

※機能は同じで見た目も変わりなし。

3.2.1.2 作業負荷について

今回の作業によって、下記のような作業負荷データを得ることができた。

＜実績値（未経験者の場合）＞

XML ファイルへの移行　：0.5H(1TBL 11 項目)
ロジックの変更　　：2H(134 ステップ)
動作検証　　：4H(ユーザアカウント付与に関する調査：2H)

＜適正値（経験者を想定）＞

XML ファイルへの移行　：0.5H
ロジックの変更　　：2H
動作検証　　：2H

3.2.2 Linux での検証

XML 化された Web アプリケーションの Linux への移植に関する作業負荷を検証した。

＜検証手順＞

１．Windows 上の VisualStudio.NET にてソースの修正を行う。
２．Windows 上の VisualStudio.NET にてコンパイルを行う。
３．FTP にて実行モジュールの Linux への転送を行う。
４．Linux 上の Web サーバー(XSP)にてモジュールの実行を行う。
５．ブラウザ(IE / Mozilla)にて動作検証を行う。

＜検証に使用した機器＞

CPU：PentiumⅢ866MHz
Memory：256MB
HDD：20GB
O S：RedHat 9
Mono：Release 0.29

※Mono は専用ツールが用意されている為、簡単にインストールが可能であった。

3.2.2.1 無修正による動作確認

Windows 上で動作したソースに手を加えることなく Linux 上で動作可能かどうか検証を行ったが、下記の様な問題が発生した。

問題点１：XML ファイルを正常にオープンできない。

（原因）OS の違いから発生するファイルパスの指定方法の違い。

図 3.7　動作状況

問題点２：日本語文字の文字化け。

（原因）OS の違いから発生する文字コードの違い。

図 3.8　動作状況

いずれの問題も OS の違いから発生するものである為、.NET 自体の問題ではなかった。

3.2.2.2　ソースの修正作業

＜修正箇所：ソースコード＞

XML ファイルへのパスを設定している２ステップを下記の様にパスの記述を変更した。

```
public class members : System.Web.UI.Page
{
    protected System.Web.UI.WebControls.DropDownList DropDownList1;
    protected System.Web.UI.WebControls.LinkButton LinkButton1;
    protected System.Web.UI.WebControls.Label Label1;
    protected System.Web.UI.WebControls.DataGrid DataGrid1;
    //Linux向けカスタマイズ Start
    //protected static String xmlFile_Member= @"C:¥Inetpub¥wwwroot¥DotnetWebApp¥members.xml";
    //protected static String xmlFile_Club  = @"C:¥Inetpub¥wwwroot¥DotnetWebApp¥clubs.xml";
    protected static String xmlFile_Member  = @"/home/dotnet/DotnetWebApp/members.xml";
    protected static String xmlFile_Club    = @"/home/dotnet/DotnetWebApp/clubs.xml";
    //Linux向けカスタマイズ End

    /// <summary>
    /// ページロード
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void Page_Load(object sender, System.EventArgs e)
    {
```

図 3.9　ソースコードの修正

＜修正箇所：GUI 部品プロパティ＞

漢字コードを使用して設定している GUI 部品のプロパティの箇所を下記の様に英文字

を使用した表記に変更した。

図 3.10　DataGrid のプロパティ

＜修正したプロパティ群＞

- 部会名コンボボックスのラベル
- 入力項目のヘッダー名（氏名、企業名など）
- 編集、更新、削除、キャンセルボタン
- ページングボタン（前ページ、次ページ）

3.2.2.3 修正後のソースによる動作確認

前述の問題点を解決し動作確認を行ったところ、下記の様に正常に動作することを確認した。

＜修正後の動作画面＞

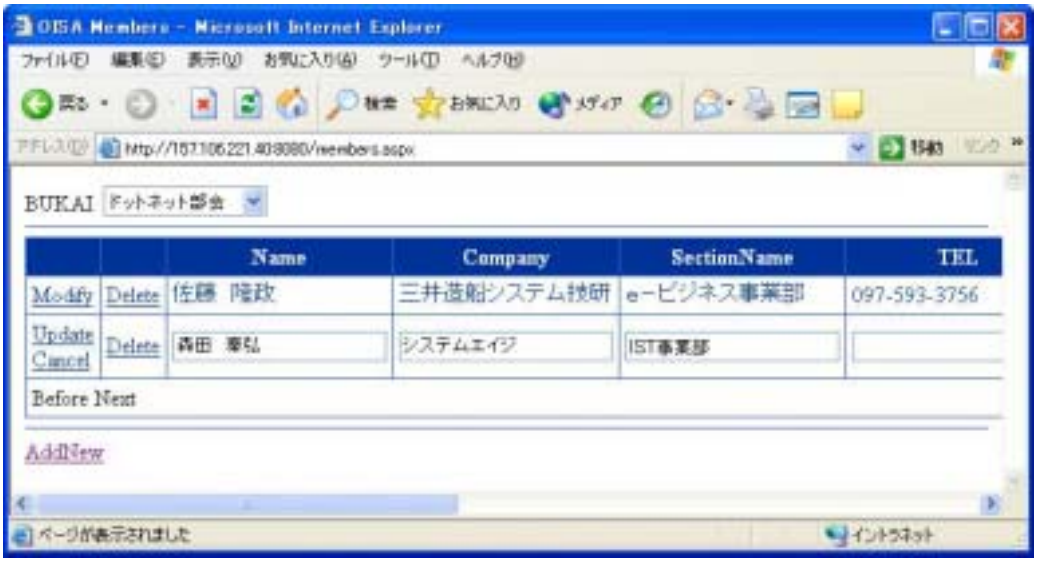

図 3.11　修正後の画面

3.2.2.4 作業負荷について

今回の検証によって、下記のような作業負荷データを得ることが出来た。

＜実績値（未経験者の場合）＞

Linux の環境構築：8H
ロジックの変更　：2H(2 ステップ、18 プロパティ)
動作検証　　：1H

＜適正値（経験者を想定）＞

Linux の環境構築：5H
ロジックの変更　：0.5H
動作検証　　：1H

3.2.3 検証結果

検証作業中に発生した問題点を以下のようにまとめてみた。

①ユーザアカウントの付与

②ソース変更（プロパティの変更）

①ユーザアカウントの付与

.NET が動く仕組などの前提知識が不足していた為に発生した作業の為、Web アプリケーションの開発経験があれば発生しなかった作業と言える。

②ソース変更（プロパティの変更）

.NET の DataGrid コントロールに対する知識が不足していた為に、発生した修正量に対しての作業コストが高いものとなっているが、DataGrid コントロールに関する知識を持っている経験者であれば発生する作業コストは低いと言える。

以上のように、今回の検証作業中に発生した問題点はすべて.NET に関する経験不足に起因するものであり、.NET の経験者であれば発生しなかった問題ばかりである。よって、.NET で開発した Web アプリケーションは低コストで Linux へ移植可能であると言える。

4. まとめ

4.1 .NET の利点

.NET の長所について以下のようにまとめてみた。

① 将来性

② コストパフォーマンス

① 将来性

.NET は Windows という圧倒的に多い既存導入環境をベースにコンピューティングのあり方を Web サービスへといざなう役割を担っている。しかも、今後の Windows は全て最初から.NET の実行環境と位置付けられている。

以下の点で.NET は Java より優位であると言える。

・既存の導入環境(Windows 端末)の多さ

・.NET プラットフォームの展開の早さ

・開発コスト（RAD 開発スタイル、互換性）

現存する既存の Windows 端末のほとんどは.NET の実行環境になりえる。その意味での.NET の実行環境の普及度はかなり進んでいると言える。Java が業界やインフラを司る団体が共同して仕様を策定してから普及するのに対して、.NET は Microsoft 社が主導的な立場で動作環境を広められることが結果として素早い展開を生んでいる。

開発における優位性も将来性に大きく影響する。.NET は、フロントエンドからバックエンドに至るまで、フロントエンドのアプリケーション開発に主流の RAD スタイルで開発を行える。Windows のアプリケーションが IDE に統合された RAD スタイルの開発ツールの登場と共に飛躍的に生産性が向上したことは周知の通りであるが、このことは Web アプリケーションを使ったシステムの開発・メンテ・保守などに関わるコストの低減も可能ということを示している。

また、.NET では Web アプリに必要な必須ソフトウェア(Web サーバーやフレームワーク)も Microsoft 社の製品で構成されており、そのほとんどが無償で提供されている。このことでサーバー側、クライアント側ともに実行環境の品質を定め、Web アプリケーションの互換性を容易に保つことができている。互換性が高いことで Web アプリの利用範囲を広げた場合にも開発コストの上乗せは最小限でとどまり、インフラの更新に左右されにくいシステムが出来上がるものと期待できる。

.NET は携帯電話やモバイル端末など PC 以外の様々なコンピュータに発展し

ていっているが、過去において Microsoft 社の提供する Win32API は、Windows CE などの非 PC 端末において非常に高い互換性を証明してきた。今回の.NET フレームワークも.NET に対応した OS の提供と同時に様々なプラットフォームに向けて提供されており、互換性の高さは保障されるだろう。

そして.NET フレームワークの拡充が進めば、.NET で構築された Web アプリケーションはどんな端末でも利用でき、「いつでも・どこでも」恩恵に授かれることになる。すなわち、.NET の社会への浸透がそのまま Web サービスの拡大につながり、企業がその恩恵に授かるのみならず、一般の人々にも Web を通して様々な生活に必要な便利さが提供されてくるのである。また、本部会で.NET アプリを Linux 上に容易に移植できたことは、Microsoft 社が.NET をオープンにしようとする姿勢の成果と捉えることができる。.NET は低コストと既存環境(Windows)を武器にして急速に活躍の場を広げていくであろう。将来性には大いに期待できる。

② コストパフォーマンス

.NET と Java は同じようなことを目指し、同じような Web サービスを構築できるが、.NET は、その開発環境における導入コストの低さ、実績のある複数の開発言語が使えることで習熟に必要とする時間が小さいこと。加えて、最も OS として普及している Windows の上で動作する事で、実行環境の導入コストや利用者の操作に対する学習コストが小さく、高いコストパフォーマンスを期待できるプラットフォームと言える。

以下の点で.NET は Java より優位にあると言える。

・利用者端末の選択肢の多さ
・利用者のアプリケーションに対する学習コスト
・開発コスト（複数言語対応、開発環境、互換性）

.NET はモバイル端末などにも普及しているが、どれも PC に搭載されている Windows と同じ GUI を持った OS である。それらがお互いに互換性を持った.NET のプラットフォームとして機能するので利用者にはさまざまな選択肢が与えられる。ハンドヘルド型の PC や PDA といった端末でも Web アプリケーションを利用したシステムを効率良く運用できるので、端末コストを低く抑えられる。

Java では互換性のある JavaVM が搭載されていることが前提となるが、過去の例から同じバージョンの JavaVM においても実行環境が違うと互換性が低くなっている。すなわち、互換性の低さの分が.NET より高コストになる。また、Java の GUI は Windows の GUI とは多少であるが異なっており、利用者の学習コスト

も.NET と比較すると高い。これはリッチコンテンツな GUI を構成した場合に差が顕著になる傾向にある。

開発コストはそのまま利用者の負担として反映されるので、開発コストも低い方が良い。Java を使ったシステムは Sun Microsystems 社の製品だけではまかない切れない。Web サーバーとアプリケーションを実行するためのアプリサーバーを OS とは別に用意しなければならず、概して高コストであり、導入する製品毎に決して少なくない知識量が必要になるため、開発者の負荷は大きく、開発コストも高いと容易に想像可能である。対して、.NET システムを構築する場合、Microsoft 社１社が提供する製品で DB サーバー、Web サーバー、アプリケーションまで全て可能である。しかも Web サーバーや実行環境はほとんど無料で提供されており、製品間の連携に高い品質が期待できるという安心材料もあるので開発者への負担は小さく、開発コストも低い。

本部会で.NET アプリを Linux 上に容易に移植できたのは、Mono の高い互換性とクラスライブラリの出来の良さのおかげであったが、このことは.NET の生産性の高さも物語っている。.NET は既存の環境に対して安価に高品質な Web サービスを提供するだけの素質を十分に持っている。

4.2 まとめ

.NET は企業のインフラを十分に活かすプラットフォームとして十分に期待できる。それは、低い開発コストと既存環境との高い親和性がもたらす恩恵が、新しい情報システムについて前向きな姿勢を与えてくれるからである。

企業は、既存のシステムを機能拡張する場合に、使い古された専用の技術を用いると、専門の保守要員の確保と特定ベンダーとのコストの駆け引きを背負う。しかし、Web サービスに代表されるオープンな技術で構成される仕組みを取り入れると、それらのレガシーなシステムを維持する時に発生する諸問題に取り組む必要が無くなり、革新的なシステムの導入も実現可能になる。

.NET は、オープンな技術で構築されるシステムであるのに加えて GUI に慣れ親しんだ Windows の操作性を再現できているので現在の企業のシステムに容易く浸透しうる。最初に企業に導入される時のシナリオは、おそらくレガシーシステムは XML で通信するドライバで.NETで作ったシステムと通信し、利用者は.NETで作ったWebアプリの画面を使うというものであろう。このとき、.NET は利用者から透過的に見えるが、それこそが.NET の恩恵なのである。

ERP(Enterprise Resource Planning)、CRM(Customer Relationship Management)などの基幹業務アプリケーションのフロントエンドや社内システム統合(EAI：Enterprise Application Integration)にオープンな Web サービスを利用することで低

コストに実現できるだろうし、EDI(Electronic Data Interchange)、B2B(Business to Business)、B2C(Business to Consumer)に Web サービスを利用することで、オープンな技術、XML によるオープンなインターフェイスに基づいて企業を超えたアプリケーションの連携が可能となるだろう。そうすれば、.NET はビルゲイツの描く将来像となる。

.NET は、安く確実に将来性のあるシステムを構築できる優れた物であると結論付けたい。

5. 参考資料

※１　http://www.go-mono.com

※２　http://internet.watch.impress.co.jp/cda/news/2003/11/19/1180.html

※３　http://www.zdnet.co.jp/news/0110/31/e_mono.html

※４　http://www.zdnet.co.jp/news/0108/07/e_mono.html

http://www.zdnet.co.jp/news/0110/31/e_mono.html

6. メンバー紹介

姫野　公洋	（部会長）	株式会社 富士通大分ソフトウェアラボラトリ
佐藤　隆政	（副部会長）	三井造船システム技研株式会社
金光　しほり		新日鉄ソリューションズ株式会社
小林　弘幸		株式会社 オーイーシー
近澤　邦雄		新日鉄ソリューションズ株式会社
広崎　克敏		大銀コンピュータサービス株式会社
宮下　一樹		新日鉄ソリューションズ株式会社
宮成　亨治		エスティケイテクノロジー株式会社
森田　泰弘		システムエイジ株式会社
重光　貞彦	（技術委員）	大分シーイーシー株式会社
佐藤　清孝	（技術委員）	(株) ウィルウェイ