Java vs C#
—競争原理はプログラミング言語にも働くか?—

前田 敦司／筑波大学電子・情報工学系

委員会が作ったプログラミング言語には魅力がないなどと言われる．1人またはごく少数の人間が，その思想や嗜好を色濃く反映させた言語は当然に個性的で，多くの場合エレガントで，しばしば使いやすい．では企業の開発チームが設計した言語はどうなのだろうか？

ここでは，JavaとC#という2つの言語について考えてみたい．これらは，それぞれSunとMicrosoftという企業がその経営戦略上の重要な武器として位置づけている言語である．最初に設計したのは少数の個人であったのだろうが，現在の形になるまでに，企業戦略を強く反映させたものになっていると思われる．

これらは「魅力的な言語」だろうか？　プログラミング言語のどういう点が人々を魅了するのだろうか？　広く使われ，永く生き残る言語の条件とはどういうものだろうか？

言語の人気

たいていの実用品と同様，プログラミング言語にも生態系における地位に似たものがあり，それをめぐった争いが常に続いている．生物界の生存競争や，他の工業製品の競争と違っているのは，その「慣性」の大きさである．プログラム言語がこの先生き残るかどうかは，その言語が過去にどれだけ使われたかによるところが大きい．

形ある工業製品が，やがては損耗して使えなくなってしまうのに対して，プログラミング言語を使って書いたコードは永久に残る．それを捨て去って新たな言語に切り換えるのはコストがかかるから，永く使われた言語というものは容易に消えない．プログラミング言語の生態系では，先住者が常に有利な立場にある．

教育の現場では，レガシーコードの問題がそれほど深刻ではないため，プログラミング言語の流行りすたりはソフトウェア開発現場よりも速い（はずなのだが...）．研究の分野でも，論文でアルゴリズムを説明するような場合には慣性の問題など皆無に等しいので，その時々に研究者が「主流」あるいは「かっこいい」と考える言語が使われる．そこでCOBOLやBASICが使われることは決してない（失礼）．かつてはAlogolやPascalが多かった．その後，Adaが使われた時期もあったが，現在はCやC++が使われることが多い．何年かたてば流行は次の言語に移り，今まで主流だった言語はもはや古臭いものとみなされるようになる．そのうちJavaだって広く使われるようになるかもしれない．

Java?

そう，Java．CやC++にとって代わったとまではいえないし，Javaでアルゴリズムを説明する論文はまだ少ないが，少なくとも研究対象としてならJavaはすでにニッチを確保したといってよい．情報処理学会のWebページで検索してみると，Javaをキーワードに含む論文や記事が400件以上ヒットする．ACMのdigital libraryで検索すると，700件近く見つかる．

1995年にJavaが発表された当座[1]，Javaの処理系は実行速度が遅く，またクラスライブラリもごく非力なものだったにもかかわらず，多くの人がJavaに関心を持った．実際のアプリケーション開発に使われるより早く飛び付いたのは，学生や研究者たちであり，中でも早くから興味を持ったのはプログラミング言語の研究者たちだった．

研究者にとってみれば，格好の飯の種を見つけたというところかもしれない．なにせ研究者だってお金と無縁ではいられない．研究費をとってくるには産業界が関心を持つ言語の方が簡単だし，解説書を書くにしてもポピュラーな言語の方がよい．研究するにしてもメジャーな言語の方がインパクトがあるし，大学で教えるにしても，ほかではお目にかかれないようなマイナーな言語よりは，どこでも使える言語の方が望ましいに決まっている．

しかし，研究者たちが飛び付いたのは，単に「広まりそうだから」という理由だけではないと思うのだ．そもそも「広まりそうだ」と思うにはそれなりの理由があったはずだ．また，メジャーだというだけでは論文にしたり学生に教えたりする気にはならないだろう．VisualBasicの処理系に関する論文など見たことがないし，情報系の学生にVBでプログラミングを教えている大学は聞いたことがない（きっとあるのだろうけど...）．

まず言語自体がそこそこモダンな仕様であったのが重要だと思う．たとえばオブジェクト指向というパラダイムが役に立つということは，専門家にとってはとっくに常識になっていたのだが，ポピュラーで・いろんな環境で使えて・安価な（できればフリーな）処理系が手に入る言語で

は，なぜか1970年代そのままの「こーぞーかってすごいでしょ」と言ってるレベルのものしか存在しなかった．新しい概念を取り込んでいて，落し穴が少なくて，手近な環境でも動く言語がやっと出てきたか，という喜びが大きかったのではなかろうか．

手近な環境とは，WindowsだったりMacだったりするのかもしれないが，私のまわりでは一般にUnix系のOSが好まれているようだ．内部構造がよく見えて，しかもシステムの働きに手を加えることが容易であることがその理由だろう．自動車の構造を教えたり，「新たなエンジンのチューニング法」を研究したりするのに，ボンネットを空けることのできない自動車を使ったのでは都合が悪いのと同じことだ．

中身が見えない・触れない環境の方が世の中では多くなってきて，性能も使い心地もどんどん向上しているのに対して，自分たちの環境でプログラミングの教育に使える言語が相変わらずCか，せいぜいC++くらいにとどまっていることにいらだちを感じていた人は多いだろう．そのうち世の中は中身が見えない，コンピュータの仕組みを教えたり研究したりするのにきわめて不都合な環境だけになってしまって，その他の環境は滅びてしまうのではないかという不安だってあったはずだ．そういう環境をふたたびモダンなものにしてくれるという意味でも，Javaはありがたい選択肢だったのではないか．

C#

一方，「中身が見えない環境」で世の中を覆いつくそうとしていた側にとっては，そうでない環境の有用性が増すのは面白いことではないから，当然いろいろな対抗手段をとってくる．自製品のシェアを高めようとするのは商行為としては当然であるから，別にそれ自体を非難する理由はないのかもしれない．

いろいろあった対抗手段のうちの1つが，昨年発表されたC#言語である．プログラミング言語オタクとしては仕様のディテールをあげつらって，どちらが良いかのディベートをやってみたり，後知恵の優位な立場から設計者たちをけなしてみたりするのも楽しいと思うのだが，大多数の読者にとって，細かな仕様上の差異などに興味はないだろう．

ここでは，両者の言語仕様のコア部分はほとんどそっくりだとだけ言っておく（仕様書[3]にはJavaへの言及は皆無だが）．個人的な印象としては，Javaがどちらかといえばストイックで，多少書くのが面倒でも言語の単純さを保とうとする傾向が強いのに対して，C#の方は，少しくらい不統一でも便利なものはどんどん採り入れようとする傾向があるように感じる．その程度の違いである．

さて，自由経済の世の中では，同じニッチを奪い合う企業間の競争によって技術が進歩し，製品の価格性能比が向上し，全体の生産性が向上するということになっている．この2つの言語も健全な競争によって使う側が利益を得る ということになるのだろうか？

徹底してOS非依存を狙ったJavaに対して，C#は特定のプラットフォームの機能をなるべく引き出すことに主眼を置いている．しかし，C#の言語自体はプラットフォームに依存したものではなく，ECMAに言語仕様を提出して規格化するのだという[2]．また，C#が前提とする.NET Frameworkも，ネットワーク透過で，OSに依存しないプラットフォームになるのだという．つまり，Javaが目指している世界とほとんど変わるところはないように思える．やはりプログラマはJavaとC# をどちらでも自由に選べて，競争の利益を享受できるのだろうか？

不安

そうはいかない気が強くするのはなぜだろう？ もし競争を仕掛けたC#が勝ち残って，Javaを使う開発者がほとんどいなくなったとすると，開発者のコミュニティは競争がなかったときに比べて利益を得たことになるのだろうか？ それよりも，コミュニティがそもそも分裂せず，負けた言語から乗り換える苦労などしないですむ方が利益になると思うのは，競争原理に反する「抵抗勢力」の考え方か？

もしJavaが開発現場から消えたと仮定してみても，.NET FrameworkとC#を使った「オープンな環境」でプログラミングを教える自分が想像できないのは 単なる偏見，あるいは想像力不足だろうか？ 「中身が見えるかどうか」などにこだわらず，勝ち残って世の中で広く使われているものを教えるべきなのだろうか？

C#が勝ってJavaが消えると世界が闇に閉ざされるように感じるのは，私が滅びつつある環境にしがみついているだけのオールドタイプだからか？ もしそうなっても実は教育にも研究にも何の害もなくて，今Javaを対象としている研究者たちは，フリーなC#コンパイラのソースコードを改変して新しい研究を行うようになるのだろうか？

いったいこの不安が根拠のないものなのかどうか．どなたか教えてください．

参考文献

1) Byous, J.: Java™ Technology: An Early History.
2) Wong, W. and Ricciuti, M.: Microsoft Strikes at Sun's Java with New standard.
3) Microsoft Corporation: C#標準リファレンス, 日経BPソフトプレス (2001).
4) Archer, T.: プログラミングC#, 日経BPソフトプレス (2001)
5) 後藤弘茂: .NETの正体はバーチャルWindows構想

(2001.9.15)

周辺環境も含めた言語比較

矢嶋 聡／NRIラーニングネットワーク(株)

言語の良し悪しを考える上で重要なこと

単にC#とJavaの構文的な違いだけでは，それぞれの利用価値を測ることは難しい．そもそもC#とJavaではその実行基盤が異なり，C#は .NET Framework，JavaはJava 2 Platformで動作するため，コード実装に必要となるテクノロジーも異なる．また，全体的な開発生産性にも影響を与える開発環境の整備状況や言語習得に要する時間なども，言語比較での重要な項目となるだろう．これらもふまえて，2つの言語を比較してみることにする．

実行基盤からくるコード実装の違い

そもそも，JavaもC#もそのルーツがC++であることから，オブジェクト指向プログラミング言語として類似している点は多いが，実行基盤が異なることからくる違いがある．たとえば，以下の2点は実行基盤の違いからくるものだ．

デリゲートによるメッセージ通知

JavaもC#もインタフェースを経由してメッセージのやり取りをする場合，受信側はインタフェースの全メソッドの実装コードを持たなければならない．C#には，デリゲートと呼ばれる関数ポインタを抽象化したオブジェクトがあり，メソッド単位の動的な連携が可能である．

実装コードとファイルとの関係（パッケージ，名前空間，アセンブリ）

Javaでは，クラスのグルーピングであるパッケージはディレクトリ（フォルダ）に相当し，またパブリックなクラスはソースファイル名と一致するなど，ソースコード上の実装とファイルシステム上の構造が密に連携している．

C#では，開発者の便宜を図るクラスのグルーピングである「名前空間」と，配布者の便宜を図るファイルのグルーピングである「アセンブリ」を別々に管理する．ソースコード内の記述と，ファイル名とは切り離して管理できる．

これ以外にも，詳細は省略するが，参照型と値型の扱い，多次元配列，メソッドの引数，スレッド同期，例外処理，オブジェクトの扱いなどの記述にも相違点がある．

このことから，どちらが一方的に良いか断言することはできないが，全般的に，C#の方が，「かゆいところに手が届くような」より細かい文法を多く持っている感がある．たとえば，Javaのオブジェクトは参照型であるが，C#では参照型と値型の2種類を表現できる．また，Javaのインスタンスメソッドは常にC++のvirtual関数に当たるが，C#ではvirutal関数と非virtual関数の2種類の表現が可能である．C#のメソッド定義では，引数に修飾子をつけて，ref（入出力引数），out（出力引数），params（可変の引数リスト）などもある．

また見方を変えれば，Javaの方がコンパクトであると言えるかもしれない．

修得しやすさ

企業などでシステム開発をする場合，少ないコストで，より短期間で技術を習得し，技術者を養成するすることは，言語を選択する上で重要な要因の1つだ．

Windowsの開発環境では，C/C++よりもVisual Basicが普及していった．とある法人向けの教育企業の公開コースを調べてみると，ここ6カ月で予定されているWindowsプログラミングコースのうち，C/C++関連が6コースであるのに対して，Viusal Basic関連は21コースである．これはVBに対する需要ととらえることもできるだろう（Windowsに限定してないC/C++コースは，このほかに15コースあった）．

VBの詳細は省略するが，短期間で習得できる点は魅力の1つである．難易度という点では，おそらくVB < C# == Java < C++ であろう．C#やJavaは，C++をよりシンプルにしたものであり，VBよりもオブジェクト指向的な知識が必要になる．構文的には，C#もJavaも難易度は同じくらいであろう．

ただし，難易度の点で強いてC#とJavaの違いを上げるとすれば，C#では文字列の扱いがより直感的に利用できる点だ．以下は，文字列のデータ自体を評価する分岐表現である．むしろその発想はVBに近い．

```
if ( s == "abc" ) { ... //(C#) 文字列変数 s の値が "abc" であるか

switch(s) {          //(C#) 文字列変数 s の値による多分岐
case "tokyo":   ...
case "yokahama": ...
}
```

JavaやC言語プログラマにしてみると，参照比較（アドレス比較）に思え，不自然に感ずるかもしれない．しかし，一方ではVB開発者にとっては分かりやすいという側面もある．

修得しやすさは実装知識の難易度にもよる

C#とJavaとの間に難易度に違いが出るとすれば，個々の実装で必要な知識の差である．たとえば，Javaでサーブレットを作りたいのであればJava文法だけでなく，サーブレットに関する実装知識が必要になる．C#でWebサービスを公開したいのであれば，ASP.NETの実装知識が必要になる．今後，C#で注目すべき点の1つは，属性という宣言的な記述がある点だ．たとえば，

```
[WebMethod]
public void Print () {
   return "Hello, world!";
}
```

と，メソッド名に [WebMethod] という「属性」を宣言しただけで，このメソッドはWeb上に公開され，XML形式のHello, world!というデータをHTTP Responseとして返すことができる（厳密には，これは.NET Frameworkの機能の1つ）．

もちろん，これだけでC#がJavaより優位と言うつもりはないが，このような属性指定の記述が .NET Frameworkにおいて整備されていけば，高度な実装を限られた知識で簡単に行うことが可能になる．

今後の「属性」の成り行きには，C#の有用性を語る上で注目する価値があるといえるだろう．

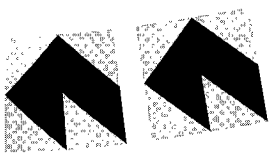
今後のC#，Java

C#やJavaが，今後利用されるかどうかは，.NET FrameworkやJava 2 Platformがどれだけ受け入れられるかにかかっているだろう．その場合，コード実装における簡易さも見逃せない要因である．

また，もう1つ加えると，.NET FrameworkではC# だけでなく，現在20種類以上のコンパイラがサードベンダによって開発中である．その中には，Javaも含まれる．いずれ，C#とJavaの比較の論点も，より実行基盤に関する内容になるであろう．

参考文献
1) C# Language Specification (Visual Studio .NET 日本語版 Beta 2 添付ドキュメント)
2) Java Language Specification.

(2001.9.17)



Open Your Mind

千葉　滋／東京工業大学

C# が勝ってJavaが消えると，世界は闇に閉ざされるだろうか

C# はいたって普通の言語として開発されているようである．Windows系OSの開発者にとって，C# はC++，Visual Basicに続く第3の選択肢となる．Javaが提供しているモダンなオブジェクト指向やメモリ管理技術を使いたい開発者にとっては朗報であろう．標準化作業も進められるようで，無事標準化されれば，C++ のように他のOS（プラットフォーム）上でも使えるようになるだろう．フリーなC# 処理系も開発されるに違いない．

ただし，標準化されるC# の言語仕様は必要最小限のもので，各ベンダは自由に言語仕様を拡張することができるようである．ライブラリについても同様で，標準化されるのはCやC++ と同様，簡単なファイルの入出力や数値計算のための機能が中心で，GUIやネットワークに関する機能は，各ベンダが独自ライブラリで提供することになると想像される．たとえばWindows系OS用には，.NETに対応したライブラリが用意される．このライブラリは標準外なので，他のOSには簡単には移植されないだろう．C# と .NETは一体であるともとれるが，標準C# と .NETは一応独立なようである．

要するに，Windows系OS陣営としては，C# の標準化がどのような結末になろうと，製品戦略に無関係なのである．同陣営は，独自拡張をほどこしたC# と .NET対応ライブラリを配布するだけである．Windows系OSの寡占状態の下では，独自拡張により他のOSへの移植性がなくなっても問題は起こらない．

一方のJavaは，C# に比べてより野心的な言語として開発されている．Java言語の仕様をベンダが独自に拡張することは許されていない．Javaの標準ライブラリは，GUIやネットワーク，暗号化といったものまでを広範に含む巨大なものである．したがって，移植性を高めるため，標準ライブラリだけを使って書いたプログラムでも，GUIやネットワークを利用することができる．

このようなJavaの標準ライブラリは，OSが提供する機能

を標準化しているようなものである．Java陣営が呼びかけていることは，結局，各OS（プラットフォーム）で同じ機能セットを提供することにし，少なくともJavaプログラムから見たOSの機能を統一し，標準化しよう，ということである．Unix系OSの標準仕様であるPOSIXのようなものを，より広範なOSに対して作ろう，と呼びかけていると考えることもできる．

Javaは決してオープンソースではなく，非商業利用でも再配布が強く制限されているが，「オープンな環境」を支持する人々の間でも人気が高いようである．制限つきながらソースが公開されていることも理由だろうが，Windows系OSの寡占を崩そうとしており，前田氏が述べているように，「オープンな」Unix系のOSにも，再び光を当ててくれそうだという期待があるからだろう．OS機能の標準化によって，Unix系OSも再びデスクトップOSとして主流になれるかもしれない．

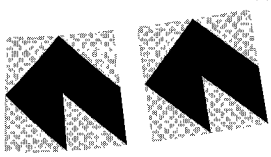
オープンな環境

そう考えると，もし仮にC#が勝ってJavaが消えてしまうと，「オープンな環境」を支持する人たちにとって，世界が闇に閉ざされたように思えるのもうなずける．ただ，そう思うからといって，必要以上にC#を嫌うのはいかがなものか．特に，仮にJavaが開発現場から消えてしまった場合，その後も大学で細々とJavaを教えつづける人が出ないことを祈りたい．潔く標準C#を教えて欲しいものだ．

商用ソフトを開発しなくてもよいからか，学生や教員といった大学人には，オープン環境が好きな人が多い．確かにオープンソースは重要な概念であるし，反Windows系OSを標榜する気持ちは理解できる．しかしオープン環境の方がシステムソフトウェアの研究の対象にしやすいからといって，オープン環境の中に閉じこもるのはいただけない．少なくとも教育では，オープンかそうでないかにかかわらず，その時々の最先端技術を教えるように努めるべきである．極端なたとえだが，GUI全盛のこの時代に，テキストベースのソフトウェアでメールを読ませたり，プログラムを開発させたりするような教育を，一般の学生に対して実施するようなことになっては悲劇である．もっとも，将来，もし私が標準C#を教える羽目になったら，きっとよいプログラミング言語の設計とは，という話を随所でしてしまうだろうが...

参考文献
1) PDC#, http://msdn.microsoft.com/library/en-us/dncscol/html/deep07202000.asp
2) ECMA activities, http://www.ecma.ch/ecma1/TOPICS/survey.HTM

(2001.9.18)



Java, C#の次に来るのは?

脇田　建／東京工業大学

C#を提案しているマイクロソフトは，今でこそWindowsを世界戦略の中心に据えています．しかし，その昔はBasicプログラミング言語で覇権を争い，勝利してきました．前田さんは昔を懐古して，プログラミング言語は個性的だったとおっしゃいます．その意味でいうと，かつてのBasicほど個性がなく，退屈なプログラミング言語もなかったように思います．

ところで，最近，知ったのですが，ちょうどMicrosoftがBasicをひっさげて成長していたころ，Bill GatesはAPLというプログラミング言語の開発に夢中になっていたのだそうです．APLといえば，特殊なグラフィック文字で表示される演算子を利用できて，パソコンをいじり始めた私には，普通にいう個性を超えた奇妙なプログラミング言語として映りました．今，思うと退屈な堅実さの中に茶目を秘めていることに，成功の一因があるのかもしれません．

Javaが投げかけた光

Javaが登場するまでは，たしかにプログラミング言語は多様性に富んでいたように思います．オブジェクト指向言語はもちろん，論理型言語も健在で，関数型言語もファンを増やしていたように思います．わたしも一時はApple社のDylanなぞに夢中になったことがあります．

そんな中で，Javaは多くの人に好意的に受けいれられました．C++に比べて単純化して洗練された設計，自動的なメモリ管理，並行プログラミング，モバイルコード，型安全なバイトコード，動的コンパイル，セキュリティなどなど．Javaはパラダイムを超えて，プログラマ，教育者，研究者とさまざまな人を引きつける要素にこと欠きません．

その一方で，Java以降には，個性的なプログラミング言語の提案は以前よりも減ってしまいました．これは，ある意味ではJavaという，かなり洗練された言語の登場で多くの人がそこそこ満足したという点で喜ばしいことです．しかし，逆に，ほとんどの人が80%程度の満足感で我慢しているのではないかという寂しい疑問を感じているのは私だけでしょうか．

C#が投げかける光

C#というプログラミング言語も，Javaとの対比でしか語られないという点では，ある意味不幸なのかもしれない．確かに，みなさんがすでに述べられたように，言語仕様からC#に独自の特徴を見いだすことは難しい．プラットフォームに固有の機能が活かせるという特長（？）も，千葉さんがおっしゃるようにJavaが目指した先進性に比較すると霞むという指摘ももっともかもしれません．

私は，C#は.NETの枠のなかで評価すべきだという矢嶋さんの意見に賛成します．私が，興味深く思うのは.NETがC#だけでなく，さまざまな言語の共通のプラットフォームとして設計されている点です．マイクロソフトでは，.NETの設計に，自社のBasic, C++, C#の開発者だけでなく，関数型言語や論理型言語の研究者の意見が反映したそうです．

かつて，プログラミング言語は個人，あるいは少数の人々によって開発されていました．しかし，今日，高度な機能や機構を提供するJavaに対抗できるような言語システムを開発することはきわめて困難になっています．それに対して，これまでJavaが独占的に持っていた高度な機能が，.NETの実行系を通して個人が設計した言語でも利用できるとすれば，うれしいことではないでしょうか．.NETという共通の舞台の上で再び，JavaやC#としのぎを削る競争が展開されるとしたらわくわくしませんか？

(2001.9.21)

闇よ落ちるなかれ

前田 敦司／筑波大学電子・情報工学系

言語と周辺環境

実は，最初の原稿を「C#のフリーな処理系は現れないだろう」という予言で終えようと思っていたのだが，探してみるとすぐにフリーなC#のプロジェクトがいくつか見つかった[1), 2)]．いずれも，「多言語VM」にあたるCLI（Common Language Infrastructure）や，その下のAPIを実装する.NETプラットフォームにあたるものからすべてフリーで作り直そうというプロジェクトである．

.NET抜きのC#というのは，どうやら誰も問題にしていないようだ．言語だけでなく周辺環境も含めて考えなければならないという矢嶋氏の指摘は当を得ているものと思う．

プラットフォームの統一――その両極端

Javaは言語だけでなく，言語から見えるプラットフォームまで厳密に規定してしまい，現実のプラットフォームの差異を無効にしようと試みた．そこでは，「プラットフォームに依存しない」という大義名分だけでなく，「ある特定のプラットフォーム」もっとはっきり言えば「Windowsプラットフォームに依存しない」ことが重要な意味を持っていた．

これに対して.NETは，言語は制限せず，その下のプラットフォームを統一しようというものである．ただし，そのプラットフォームは何かオープンなプロセスによって決められたものではなく，既存のIEEEのOS規格などともまったく無関係なものである．ありていに言えば「我々のプラットフォームで世界を統一しよう」という主張にほかならない．

いずれにせよ，使い勝手が統一されるならば使う側にとっては同じようなもの...だろうか？　あるいは，脇田氏が言うように，共通のプラットフォームの上で言語間の競争が活発になるのだろうか？

.NETなしの「標準C#」

コメンテータ諸氏の一致するとおり，C#は，いわば.NETの申し子である．言語仕様書を見ても，CやJavaなどの他の言語とは違って，標準ライブラリにあたるものはまったく含まれていない．.NETのAPIの潜在能力をフルに引き出し，.NETプラットフォームの用途の全域で高い生産性を発揮することがC#に求められた使命なのである．

すると，.NETを離れた言語としての「標準C#」はどういうものになるのだろうか？　千葉氏の言葉どおり，Javaがすたれても標準C#を教えていけばよいのだろうか？　ひょっとして，標準C#はオープンな環境でも動いて，それを教えれば開発現場でもその知識がすぐに活かせるのだろうか？

千葉氏も予測していることだが，標準C#にライブラリはほとんど含まれそうもない．C#の標準化を行っているECMAで，以前にJavaScriptを標準化した結果であるECMAScript[4)]を見ても，Boolean, Number, Stringなどの基本的なオブジェクトや文法の定義はあるが，GUIにあたるものは何もない．今回も，CLIに含まれる基本クラスを除いて，プラットフォームの豊富な機能は標準化案に含まれないと

いう[5]．要するに，それだけでは事実上プログラムが書けないのである．自らが統一しようとしている .NETプラットフォームについては標準や規格などおくびにも出さず，C# 言語についてのみ形だけの標準化を強調するのは，Sunに対する攻撃以上のさしたる意味はないのだ．

ならば，標準C# を大学で教えるというのは，すたれてしまった言語を教えるのと同様に，実用的な意味は薄いと言えるのではないだろうか．

ギャップは埋まらない

議論の結論が出たとはとても言えないが，個人的な意見としては，ソフトウェア開発の現場でポピュラーな言語と，教育や研究の場でポピュラーな言語とは，やはりそう簡単に一致しそうもないと感じる．Javaは，そのギャップを狭める可能性のある希有な例であった（まだ今もそうだと思う）．願わくばその可能性が閉ざされざらんことを．

シニカルな言い方をすれば，その時代に世の中でいちばん使われている言語が教育の場で主流であったことなどないのだし，企業もそういう期待はしていないのだから，気にせずに教えたい言語を教えればよいというのは開き直りがすぎるだろうか．もう少し積極的に「ギャップ」を擁護すれば，いちばん使われている言語は必ずしも最新の技術などではない（例：COBOL, Basic, C, VB, ...）し，最新の技術を追いかけてもすぐ陳腐化してしまう（たとえば，DDEやOLEを教えたとしても，2年もすれば陳腐化してしまった）のだから，そんなものにとらわれずにプログラミングの概念を教えるのに適したものを使うべきだ，と言えるのではないだろうか．もちろん，開発現場でもポピュラーなものであればそれに越したことはない．

教える側からすれば，教えやすい（教えたい）SmalltalkやSchemeやHaskellのような言語が，開発現場でポピュラーになってくれれば最もありがたいのだが... なぜならないのか，の方をあるいは考えるべきなのかもしれない．

参考文献

1) Mono, http://www.go-mono.com/
2) DotGNU Portable.NET, http://www.southern-storm.com.au/portable_net.html
3) DotGNU, http://www.dotgnu.org/
4) Standard ECMA-262 ECMAScript Language Specification 3rd edition, ECMA (Dec. 1999). http://www.ecma.ch/ecma1/stand/ecma-262.htm
5) Zeichick, A.: Microsoft Embraces FreeBSD, ISO, http://www.sdtimes.com/news/034/story4.htm

(2001.9.25)

議論の続きは，次のURLをご覧ください．http://www.ipsj.or.jp/magazine/interessay.html