プログラミング言語研究に未来はあるか？
– スクリプト言語のすすめ –

田中 二郎
筑波大学 電子・情報工学系
305-8573 茨城県つくば市天王台 1-1-1
jiro@is.tsukuba.ac.jp

1　はじめに

コンピュータというのが人類の歴史の中で画期的な発明だということについては異論を持つ人は少ないだろう。コンピュータはハードウェアとソフトウェアから構成されるがどちらがより重要なのだろうか？（一部にはハードウェアが好きな人がいるかもしれないが）疑いなくソフトウェアの方である（と私は思う）。

そのソフトウェアの中心的な部分を担うのがプログラミング言語である。過去においては、コンピュータに興味を持つことはプログラムを書くことであった。

いまから 30 年以上も前になるが、私がまだ大学の学生の当時、コンピュータのプログラムを Fortran や Algol、あるいはアセンブラで書いて楽しんだ。また、80 年代のパソコンブームの時代でも、 Basic などで各種のプログラムを書いて楽しむことがコンピュータマニアの喜びであった。

コンピュータサイエンスという学問においてもプログラミング言語こそが中心であり、コンピュータサイエンスにおける King of Kings であると思ってきた。

2　プログラムを書かない時代

しかしながら、このコンピュータサイエンスの王道にも昨今暗い影が忍び寄っている。

昔は計算の結果として数字が出れば満足していたが最近では GUI(グラフィカルユーザインタフェース) というものが全盛となっていて、計算結果としてグラフィックスがでないと満足しなくなっている。コマンドベースであったオペレーティングシステムも Windows のような GUI に基づくものになっている。 GUI とはまことに便利なお化粧道具であり、これを用いれば普通のプログラムでも実に格好良く変身することができる。

現在のような Windows パソコンの先駆けは 80 年代後半から流行したマッキントッシュであろうが、このころから段々プログラムを書かない人間が増えてきた。ただプログラムを買ってきてインストールし実行するだけである。コンピュータとは計算をする道具から電子文房具になり、文書を書いたりメールを送ったりする道具、 Web ページを読む道具になった。

私もずいぶんとマッキントシュを愛好したが恥ずかしいことにマッキントッシュでプログラムを書いたことがない。実は GUI とは厚化粧のシステムで、このシステムを使いプログラムを書くのは至難の技という印象である。

その辺のプログラムの書きにくさ、操作性の悪さは今日の Windows でも変わっていない。通り一遍のことをやろうとすればアイコンをクリックすればよいが、何か自分流にやろうとする時には DOS 窓を開くことになる。 (先日も

Windows にはディレクトリのファイルの一覧を印刷する機能がないことを知って唖然とした。)

3 伝統芸能としてのプログラミング言語研究

プログラミング言語とはユーザに使ってもらってなんぼの世界である。新しい言語を発明してもユーザがつかないようでは何にもならない。

プログラミング言語とは保守的な世界である。 Fortran が発明されたのは 1954 年、 Lisp が発明されたのは 1962 年である。 Lisp についてはそろそろ処理系のメインテナンスが怪しくなっているという話も聞くが、 Fortran については、現在まだ第一線のバリバリである。

Fortran にしても Lisp にしても、まだパーサという概念もなかった頃のプログラミング言語である。 Fortran はバラバラに空白など関係なくプログラムを書くことができるし、 Lisp は括弧だらけである。そのような言語には早く御隠居願いたいというのが本音であるが特に Fortran については、今後およそ 100 年間は安泰であろう。

このように年寄りが頑張っていれば新人が活躍する余地はない。新しい言語が現れなければ、プログラミング言語研究は活力を失い、プログラミング言語研究者は細々と伝統芸能を守り続けるだけの集団となってしまう。

4 プログラミング言語の研究

プログラミング言語においても、 80 年代には、関数型言語、論理型言語、オブジェクト指向言語がこれからのプログラミング言語として注目された。しかしながら、この中で社会的に認知されたのはオブジェクト指向言語のみであり、関数型言語や論理型言語については、あいかわらずイバラの道を歩んでいる。

私自身も 80 年代の後半を、 ICOT という国策組織で第五世代コンピュータプロジェクトのために過ごした。当時、米国にいた私はこのプロジェクトに参加できることになり、日本人がプログラミング言語の世界で世界に貢献できるチャンスであると感じ、参加することに興奮を覚えたことを思い出す。このプロジェクトは論理型言語をベースとして世界征服を志したものであったが残念ながら我々のプログラミング言語は世界を征服することはなかった。

一般には新しいプログラミング言語が提案されそこに新しいプログラミング言語の研究成果が採用されて技術が進展するわけである。

最近のプログラミング言語界のニューフェイスは Java であろうが、この言語には、オブジェクト指向、仮想機械、リフレクションなどの概念がそれなりにスマートにとりいれられ、プログラミング言語に関心を持つものとしては嬉しい限りである。

5 国際的なハンディキャップ

米国にいた時にはあまり意識することもなかったが国際的なハンディキャップというのはたしかにあるかも知れない。コンピュータサイエンスという学問が米国を中心に回っており、コミュニティのテイストに合う形で論文を書かないと受け入れられない。同じ研究をしていても日本にいて影響力を保つのは難しい。

一方、ネットワークの発達によりコミュニティのバーチャル化が進行し、田舎にいても世界に向けてソフトウェアを発信することが可能となった。昔は東京にいないとトレンドを掴むことが難しかったが、現在は筑波にいる不便を感じることは少ない。

今後、国際的なハンディキャップはなくなる方向なのか、あるいは依然として何らかの障壁が残るのか、その辺は気になるところである。

6 そこでスクリプト言語

なぜ最近の人間はプログラムを書かないか？前述したようにその原因は GUI にある。パソコ

ン上でも Visual Basic, Visual C++ といった統合環境が出ていて、それらを使えばそこそこのプログラムを楽しむことができる。しかしながら、そこで面白くないのは作ったプログラムを既存の環境にシームレスに結合して使えないことである。

私が注目しているのはスクリプト言語である。5 年くらい前からプログラミング言語の未来形としてスクリプト言語を意識しており、これからはスクリプト言語の時代となると主張している [1]。

スクリプト言語は、いわば電報言語で、「一筆啓上、おせん泣かすな、馬肥やせ」と要点をささささっと書くとあとは処理系がやってくれるという「お便利言語」、もしくは Unix のコマンドをただで使えるようになっている「宿り木言語」、あるいは Unix の力を借りて敵を投げる「巴投げ言語」であるということができる。

プログラムを書かなくなった世代へプログラミング言語を復権させるにはスクリプト言語に活躍してもらうしかないと思っている。

またスクリプト言語には、日本人の感性にあった簡潔さがある。HAIKU の世界である。年とってプログラムを書かなくなった人でも、スクリプトをちょこちょこと書いて動かして見せるのは、竹内郁雄氏の好々爺、老人力の世界である。

7　スクリプト言語の分類

スクリプト言語を世代に分類して挙げると以下のようになる。

まずは sed，awk，shellscript。これが第一世代のスクリプト言語である。基本的には sed や awk はフィルター言語、shellscript は Unix のコマンド処理言語であるが、これらによりスクリプト言語の基礎が築かれた。

つぎが Tcl/Tk と Perl。これが第二世代のスクリプト言語で、スクリプト言語が実用言語としての体裁を持つようになった。Tcl/Tk については評価が分かれるかも知れない。「このようなおかしな言語はとても学生には教えられない」とか「Tcl/Tk は本まで書いたのに好きになれない」とか真顔でいう方がいて嬉しくなってしまう。(Tk の功績と、電報言語としての性格から私は Tcl/Tk を高く評価している。) Perl についてはもっとも使われているスクリプト言語と言っても良いかもしれない。Perler という言葉もあるくらいで学生が今プログラムを書くとするとだいたいがこれである。

第三世代が Ruby や Python のオブジェクト指向スクリプト言語、オブジェクト指向を採り入れて実用言語としての色彩がより強くなっている。また、動的な HTML を作るために、従来は CGI に Perl を使うことが多かったが、最近では、クライアントサイドスクリプトとして Javascript、サーバサイドスクリプトとして PHP, ASP などを使うことも増えてきた。こうした言語を第三世代のスクリプト言語と呼んでもいいかもしれない。

スクリプト言語の世界における Ruby の健闘については、いまさらいうまでもない。使ってみると Ruby は GUI との相性も良く、言語仕様も最近のプログラミング言語の研究成果を採り入れていてそこそこに洒落ている。Ruby は、世界に誇るスクリプト言語になっているのかもしれない。

8　原点に戻る

要するに伝統芸能化したプログラミング言語の研究、米国の追従だけのプログラミング言語研究から脱皮し、これからのトレンドであるスクリプト言語の研究を行なおうと言うのが本稿の趣旨である。

昔の IBM を中心とした大型コンピュータ全盛の時代には、国産メーカによって IBM 風の OS が作られ、その上にさまざまな言語処理系も作られた。しかしながら Wintel の時代となり、パーソナルコンピュータの CPU は Intel のマイクロプロセッサであり、Windows の上に Visual Basic や Visual C++ を買って載せるよ

うになった。そのため言語処理系の技術も一般には不要となりつつある。また、一般には言語処理系は買わず、 Office と Web ブラウザだけしか使わないユーザも増えつつある。時代はより悪くなっていると言えよう。

この辺で我々も原点に返って考える必要があるのではないか。プログラミング言語とはそも人間とコンピュータのインタフェースをとるためのものであり、人間のコンピュータに関する活動すべてがプログラミング言語となるはずである。そう言った意味ではスクリプト言語は、プログラミングの原点にもっとも近いところにある言語と言うことができる。

インタープリティブな言語には、インタープリティブであることによる反応の明解さがある。昔から Lisp はインタープリティブであることに拘ってきたし Java にしてもまた然りである。スクリプト言語は、動的であることを最大限に楽しめる言語になっている。

9　今後にむけて

今後のスクリプト言語を考えると、いくつかの可能性を挙げることができる。

一つは読みやすいスクリプト言語の開発。そもそもが電報言語でモジュール化も不十分なので読みにくい、そこで読みやすいスクリプト言語というのが流行るかもしれない。そう言った点をつきつめていくとスクリプトベースの仕様記述言語というのもありかもしれない。

もう一つは宿り木スクリプト言語への回帰。そもそもスクリプト言語は独立した言語としてではなく、何かに寄生し、その上にちょこちょことソフトを書くことで出発したものである。そういう点では最近のスクリプト言語は巨大化し、それ自体が独立した言語になってきた。私が Tcl/Tk を愛好するのは、それが言語としてほとんど目立たないところにある。くどくど書くスクリプト言語より、何かに寄生しそのくせ要所は押えるというのがスクリプト言語の原点である。この辺で再び宿り木スクリプト言語への回帰という揺り戻しがあるかも知れないと考えている。

また、 GUI とより親密になったビジュアルなスクリプト言語の開発というのが考えられるかもしれない。現在のアイコンベースの GUI は単にアイコンを押すだけで、コマンド言語のようなプログラミング構造をつくり出す能力を十分に持っていない。本来 GUI は二次元なので、一次元のコマンドと比べより多くの表現力を持つはずである。スクリプト言語の未来像としてビジュアルなスクリプト言語を描くのはまだ早過ぎるだろうか。

参考文献

[1] 座談会「スクリプト言語とは」 bit, 共立出版, 2001 年 1 月, pp.66-75.