ソフトウェア技術の動向*

和　田　英　一**

1. はじめに

情報処理学会の理事会から，ソフトウェア技術の動向についてはなすように依頼がありましたが，実はあまりおはなしするようなこともなくて，こまっています．いろいろなかたに，ソフトウェアの最近の動向はなんだろうとたずねても，どなたも，さあ，といってくびをかしげるばかり，反対に大会当日ご高説をうかがいますという次第で，きょうはこのなかにかなりの野次馬がいるとみうけられます．このたびFJCCに出席されて，アメリカから帰国された電気試験所の渕さんに一昨日おあいして，はなしのたねをしこんでおきましたので，きょうのはなしには，かなりその方の情報源があることをご了承ください．

きのうの特別講演はハードウェアで，そのうちでも高橋さんは超ハードをはなされました．計算機をハードとソフトにわけて，HとSとかきます．超ハードは特にHyperhardの意でHHとかきます．ソフトの方にもただのソフトと超ソフトがあるようで，超ソフトはSupersoftのつもりでSSとかくことにします．そこでかたい方からやわらかい方へ，HH, H, S, SSとならぶわけです．きのうの高橋さんのおはなしは，HHとHのうちHHの方でした．きょうはSとSSのうちSについてのべようとおもいます．Sは大体システムプログラム，最近のよび方ではOSです．SSにはユーザープログラムやアプリケーションパッケージなどが分類されます．もし最後に時間があれば，すこしSSのうちで，ぼくが興味をもっているものについてふれたいとおもいます．

ところでソフトウェアについて，特に目をみはるようなことはないのですが，しかし，いろいろな概念が整理されてきていることは事実です．ここの地下4階の電気試験所の計算機のへやへゆくと，どれかの装置にはりがみがしてあって，「ソフトウェアはつかえばつかうほどよくなりますが，ハードウェアはつかうほどへることをおわすれなく」とかいてあります．異論もなくはないのですが一応これに賛成しておいて，さていいたいのは，ソフトウェアはそのうえつくればつくるほど整理されて，すっきりしたものができるということです．

IBMなどはシステムプログラムを相当つくった経験がありますから，つかいやすい，いいものをつくります．日本のは，うわべはにてますが，つかいにくいところを沢山もっています（立教大学の島内先生が，システムプログラムは一度できあがったら，おしげもなくそれを全部すてて，すぐもう一度つくると，いいものができるといわれたことがあります）．フィジカル，ロジカル，リエンタラブルなどの概念は，システムをあたらしくつくるときに発生したものということができましょう．

システムプログラムの概念が整理されてくると，使用が容易になり，計算機ののびにプログラマののびが比例している必要はありません．きのう開会のあいさつで会長のいわれた情報処理学会の10年間に，計算機ののびは10なん倍，会員ののびは約3倍，これは残念な数字だというのは，考えかたをかえれば，これはシステムプログラマの努力のあらわれであるともみることができるとおもいます．

2. モニタまわり

2.1 TSSの様子

TSSが提唱されたのは1959年頃で，まずMITあたりが実験をおこない，その後かずおおくの機械にTSSがくみこまれてきていて，小型の計算機の実験の報告もあります．IEEE Computer Group Newsにでていた1966年のハイライトのうち，ソフトウェア関係に，事務計算のTSS化の一項がありました．Keydataなどをさすのでしょうが，この夏，渡米されてKeydataをみてこられた日立製作所の永井さんのおはなしでは，フォーマットのかぎられた作業しかできない形式のものだそうです．TSSのターミナルは，いろいろなところにすえつけているようで，先日のComputers and Automationには，ボートのなかにターミナルをもちこんだ写真があって，フローティングソフトウェアハウスだなどという説明がでていま

* Trends in Software, by Eiiti Wada (Faculty of Engineering, University of Tokyo)
** 東京大学工学部計数工学科

した.

TSS をつかうとバッチ処理よりどのくらいよいかという比較実験が方々でおこなわれています. 一応 TSS の方がよいような結果がでています[1]. それで TSS を促進することになって, MIT では MULTICS をつくるといっています. MULTICS は 1965 年の FJCC に発表されて, われわれは 1965 年の年末には 2次元メモリーとはなにものだろうとくびをひねったものでした.

1966 年夏にはご存知のように Saltzer さんが来日され, MULTICS についてくわしい説明をされ, ちょうどいまごろ GE 645 が般入されている, ということだったので, その後の進展をおおいに期待していました. この春に Organick という人の "A Guide to MULTICS for Subsystem Writers" がでて, そのコピーを電気試験所からいただいて, 大変おもしろくよみましたが, FJCC と Saltzer さんのはなしがもうずい分ちがっていたように, この本では去年の夏とさらに様子がかわっていました. 日立製作所の永井さんは, 前述の渡米の際 MIT にもよってこられ, MULTICS がターミナル1台でテストを開始したと報告してくださいました. 先月, 渕さんが MIT にゆかれたとき, いまなん台のターミナルでテストしているのかと質問したら, まだ1台だという返事だったそうで, MAC は MAC でも目下 Mono Access Computer になっているようです.

まえの CTSS は今からみるせいか, あっというまにできてしまいましたが, それはシステム製作に参加した人数がすくなかったからだそうで, それにくらべると MULTICS は大型のプロジェクトでもありますが大勢でよってたかっているので, なにか, 余計なかなかできないという感じです. Saltzer さんからこの夏うけとったてがみでは, 60 人くらいのフルタイムプログラマでやっているそうです.

それならこういう大型のシステムは人力にあまるかといえばそうではなくて, FORTRAN でもはじめは大勢でゆっくりやっていたわけですが, ちかごろはわりに少数で短時間でできるようになりました. それとおなじで, もうなん年かたつと, 最近の FORTRAN のコンパイラみたいな感じで MULTICS ができるようになるのではないかとおもいます. いまのゆっくりな MULTICS も経験をつみ, 考えかたをまとめるという点では相当な業績をあげているといえるでしょう.

2.2 TSS の批判

なかなか MULTICS ができないということだけではなく, TSS にはまたたくさんの批判があります[2]. その代表的なものには Computers and Automation の 1965 年の TSS 特集号にのったのがあります. この号は E.C. Berkeley が TSS の大宣伝をやっている一方, 批判ものっているのだから愉快ですが, その批判の第一はなんといってもオーバーヘッドです. これについてはあとまわしにしますが, つぎはあまり膨大なシステムにしてしまうため, 故障がおきたときの影響がおおきくなるというのがあります. 1965 年の秋の, New York の停電みたいなことになります.

MULTICS でもインクレメンタルダンプなど対策は講じているようですがどうなることやら, たしかに心配な点です. TSS は大変便利なので, 利用希望者が殺到するだろう, そのため優先度をもうけなくてはならなくなる. するとトーテムポールのしたの方の利用者はほとんどつかえなくなる, そんなのより, 小型の機械を自分でもっていたいというのもその批判のひとつですが, これは計算希望量より処理量がすくないときには TSS であれ, バッチであれおきることです. オーバーヘッドがふえたため処理量がへるということもあります.

バッチ処理では希望量が処理量よりおおいと, ターンアラウンドタイムがのびて, 時間あたりの希望量を処理量まで自動的にさげてしまいます. TSS だといくらダイヤルしてもおはなし中ということになります. バッチとちがって, 先着順ということにならず, 根気よくダイヤルしつづけるのがかちです. きっと回線をつかまえるまでくりかえしダイヤルする機械が発明されるでしょう. こういう状態ではバックグラウンドジョブはほとんど処理できませんから, 東大の大型計算機センターのようにジョブがおすなおすなできているところは, TSS なんかやっていられないかもしれません.

オーバーヘッドが大きいから TSS はいやだということもよくききますが, オーバーヘッドというのはなんでしょうか. それはリスポンスタイムをみじかくしたためのコストです. TSS にするのはとにかくバッチ処理よりもリスポンスタイム (Scherr[3] の本にあるコンソールパートオブインタラクション) をみじかくしたので, いささかのオーバーヘッドはしかたのないことです (バッチ処理でも東大の大型計算機センターのように, 短時間ジョブははやく処理されるというふ

うにすると，プログラムを複製し，データをわけてみじかいジョブを複数にして依頼する利用者がでてきます．この利用者の出力は，まとめて長時間ジョブでだすよりもはやくかえってきますが，分割したためにいろいろなところにオーバーヘッドがかさむことになっています)．

とかなんとかいっても，日本でも着々と TSS がつくられているようで，きのうは電気試験所の ETSS のおはなしがありましたし，きょうは日電の方から発表がおこなわれます．慶応の工学部が TOSBAC でおこなった TSS はこの夏公開され，万年暦や素数の計算のデモンストレーションをやっていました．今月のはじめに京大数理解析研で計算機構論のシンポジウムがあり，そのかえり，大阪まであしをのばして阪大の MAC をみてきました．この TSS はひるまは残念ながらうごいていませんで，出力された紙しかみることができませんでしたが，なんでもひるまはバッチ処理で，夜間，TSS の試験をやっているのだそうです．そこで阪大の MAC は反対の MAC だなんて，わるいことをいう人もいますが，ここのは TSS といってもリモートバッチの性格のものですから，これでもよいのかもしれません．

2.3 もっと一般に OS

はじめにのべたように OS もなんどもつくりかえられましたから，まえの OS の使用中にコブのついたのがつぎの OS ではすんなりととりいれられるというふうに，つぎつぎとシステムが膨大になるにもかかわらず，整理されたかたちになって登場します．マネジメントということばには，どうもよくわからない点もあるのですが，最近の IBM system 360 の OS をみますとこの OS はタスクマネジメント，ジョブマネジメント，データマネジメントの三つのうえにくみたてられているとか，Saltzer さんの MULTICS のトラフィックコントローラ[4]にはハードウェアマネジメントとリソースマネジメントをどうこうというのがあって，マネジメントということばで，システムの機能を整理する傾向にあるようにおもわれます．

これらのマネジメントのおかげで，利用者はむずかしいことをやらないで，てがるに計算機が利用できるようになったのですが，しかし，こういうゆきかたにも痛烈な批判が，すでに 1963 年の Comm. ACM にでています[5]．それによるとせっかく，かなもののすべての機能に対してレンタルをはらっているのに，このモニタシステムでは，これこれはつかえないぞといわれる（たとえばセンススイッチ）のは，けしからんというのからはじまって，システムプログラムはつぎつぎと磁気コアを占領してけしからん，モニタはよけいなチェックまでするからけしからん，センターによって，おなじ機械なのに操作の仕様がちがうのはけしからん（このセンターでは些細な変更のほかは，標準のシステムを使用しているといっていても，その些細が，とりもなおさずソースプログラムをかきかえなければならない程度のものであることがしばしばである）といっています．とくに一般の利用者はふつうのシステムでよいが，高級プログラマにも，その腕をふるわせないのはけしからん，高級プログラマ用モニタもつくれという主張でした．

もちろん OS の設計者もそこも考えてはいるのですが，具体的につくるとなると，やはりいろいろと制限がくわわりがちなことはいなめません．

最近の Comm. ACM などには，OS の解析，統計がだんだんとのるようになりました．Comm. ACM だけでなく，九大の牛島さんたちの解析もあり，また MIT の CTSS の調査は前述の Scherr という人の本にのっています[3]．それでただ，どこがつかいにくいということのほかにどこに無駄があるかということがわかってくれば，またその点を改善した OS が登場するということになるわけです．

こうやって OS を修正すると，またまえにかかっていたプログラムがかからなくなったという，頭のいたいことがでてきて，結局，ふるいのや，高級プログラマ用でセンターには複数の OS が存在するということもあるようです．些細な変更をしたセンター独特の OS が各センターにできて，1 センター 1 OSから（これだけでも各センターの OS のメンテナンスをメーカーがやったのではやりきれなくなりますが）もっとふえて新聞をにぎわしている用語をつかえば “a few” OS's というふうになりました．

しかし実際 OS も発達したものだとおもいます．せんだって東芝の米田さんにおねがいして GE 635 をみせていただきました．これはマルチタスクをやるモニタで，コンソールからたずねると，いま実行中のジョブはこれ，これ，これ．待期中のジョブはこれ，これ，これ，これときもちよい返事がもどってきました．

OS に関してひとつメーカーにおねがいしたいのは，OS のマニュアルの計算機による編集です．IBM のマニュアルはご存知のように計算機でつくられています．Saltzer さんのトラフィックコントローラの本も

そうです[4]. 英語の方がその点はたしかに便利ですが，漢字テレタイプをもちいれば，日本でだって実用になるとおもわれます．もっとも現在の漢字テレタイプには多少問題はありますが，きのう電気試験所の真名子さんが ETSS の EDITOR のはなしをされました．そのマニュアルをみせていただいたこともありますが，あれも EDITOR にかかるかたちになっているとよいとおもいます（カナ文字でよいのではないでしょうか）.

3. ランゲージまわり

3.1 JIS 規格など

最近のランゲージまわりの一つの話題は FORTRAN と ALGOL の JIS がきまったことです．あまり内容をまだよくはよんでないのですが FORTRAN 7000 をながめた感じではプロセッサ（処理系）に親切だなあという気がしました．すでにある各種の FORTRAN は今回制定された規格と多少でいりがあるでしょうが，これからは包含関係になって，つかいよくなるとおもわれます．一体マニュアルというのはどれにも類似のことが書いてあって，すでによんだものとの相違点をしらべながらあとのをよまなければならず大変ゆううつになります．それなのに，またよまなければならないマニュアルがいくつかできてしまったという印象もあります．

この規格ができてよいとおもったのは，日本語の用語がこれからこれに統一されてゆくだろうということです．むかし Report on the Algorithmic Language ALGOL 60 がでたときもそれによって用語がずい分統一されてきたと感じました[6].

こういううえからの規格制定とは反対に，したから利用者のこえとして FORTRAN を統一してほしいというごきもあります．それは今度京大，九大，東北大に FACOM や NEAC がはいって東大同様の共同利用センターができることになりましたが，東大のセンターで使用したプログラム（単位）がそのまま他のセンターででもかかるようにという要求です．この方はちょうど JIS がきまったので，相談は簡単，JIS FORTRAN 7000 というはなしになっているそうです．問題はカードパンチで，東大ではすでに IBM の 26 型のカードパンチを全国にばらまいていますが，これからのセンターではむしろ新型を設置するでしょうから，その両方どちらのコードでも，制御カードの指定によってよめるようにしてもらう必要がでてきています．

COBOL に関しては 1965 年版仕様書ができて，和訳が進行中です[7].

3.2 PL/I サブセットなど

またさきほどの IEEE Computer Group News ですが，これに PL/I について，フル PL/I のコンパイラは，どろぬまにはいったようだという記述があります．これが本当かどうかは別として，またサブセットの PL/I ならうまくゆくのか，サブセット用コンパイラはいろいろなところで計画されているようです．IBM system 360 にはF コンパイラというのがありました（その後 360 の PL/I がどうなっているかはあまりよくはしりません）．Saltzer さんが MULTICS のはなしをされたときはしきりに EPL (early PL) というのがでました．TSS のファイルの例も EPL だし，MULTICS 自体も EPL でかかれているのだということでした．こんなシステムプログラムを EPL でかいて大丈夫なのかという質問に，一応うごかしてあまりにもコンパイラ言語のためおそい部分は，あとで機械語でチューンアップするんだといっていたようでした（この講演の直後に Saltzer さんからきた手紙によると，PL/I のコンパイラのつくるプログラムが，ながくてそくて，コンパイラの改善が焦眉の急だということです）.

そのほか PLIW とか PL/I* が日立中研や日本ソフトウェアで計画されているようなこともきいていますし，通研にも PL/I のコンパイラがあるらしく，また電子協のソフトウェア委員会でも，minimum PL/I subset の案をつくり，それが先日配布されてきました[8].

3.3 コンパイラつくり

かってはコンパイラを一つつくるのも大仕事で，だれかがコンパイラを完成したとなるとニュースになったり，情報処理の論文になったりしたものでしたが，最近では FORTRAN や ALGOL のコンパイラなどふつうのつくりかたでは話題にもならなくなりました．またコンパイラの特長を一覧表にしたものも何度かつくられたりしましたが，これも前述の JIS のおかげでもうあまりつくられなくなるでしょう．まえに情報処理の座談会で島内先生が，コンパイラをつくる技術はスタックとチェインとリストしかないといっておられたと記憶しますが[9]，これらの手法はもう上記のような宣言をしなくても誰もしっている標準のものになってしまいました．

スタックの方はスタックつきのオートマトンなどに

分どられてしまったようです．リスト処理は，そのものずばりのリスト処理用言語がつぎつぎとつくられていますが，リストの残骸処理のガーベージコレクション（このことばもLISP 1.5で使用されてから，リスト処理の標準語になりました．LISP 1.5の用語はCAR, CDRからはじまって，妙なものがおおく，ATOMなどのほかはそう標準になりそうなことばもないのですが，ガーベージコレクタ（LISPではリクレーマーというよびかたもしますが）はリスト処理で標準になった例です）の手法がこのところいくつかCACMや情報処理にのったりしたのがめだちました．

その他コンパイラつくりに関したものにはインコアコンパイラや，シンタックスデイレクテッドコンパイラというものがあるようですが，くわしいことは勉強していませんので，名称をあげておくだけとします．

コンパイラをつくるときいつも問題になるのは，コンパイラの目標で，コンパイラを凝ると能率のよい目的プログラムができるが，そのかわりコンパイルに時間がかかる．コンパイルの方を凝らないで，コンパイルの速度をあげるとできあがった目的プログラムが，凝ってコンパイルしたものほどよくはないということになり，コンパイルをはやくするか，目的プログラムをはやくするかの二者択一にせまられます．コンパイラつくりのしたうけなどすると，コンパイル時間がうんとはやくて，実行もはやい目的プログラムをつくりだすコンパイラにしてくれという注文があって，安くて良いものをかいたいというのと似た難題になります．よくいわれていることは結局コンパイラを二つつくる．その両方ともランゲージの仕様はまったくおなじで，一つは目的プログラムはそんなによくはないが，コンパイルのはやいもの，もう一つはじっくりコンパイルするが，目的プログラムは効率のよいもの，の二つです．

デバッグ中のプログラムは目的プログラムがはしりだせばすぐエラーにひっかかるか，配列もループの回数もデータの数も本番のときよりずっとすくなくしてあるから，実行の時間はそうかからない．したがって，目的プログラムは効率がわるくてもかまわない．むしろムシがいるたびになんどもコンパイルをやりなおすから，コンパイルははやい方がよいわけで，前者のコンパイラをつかいます．ところが一たんできてしまったプログラムは，もうコンパイルしなおしませんから，一度だけゆっくりコンパイルして，あとはフルスピードで目的プログラムをはしらせてプロダクションをやろうというのですが，なかなかこの2種類のコンパイラを用意しているというはなしをききません．

ところでこのデバッグとかなり性格の似ているのが実習のプログラムで，学生や初心者のつくるプログラムはあまり大きいものではないため，デバッグ用のコンパイラでことたりるわけです．特にこういうプログラムはFORTRANばかりでアセンブリーもないとなると，FORTRANのコンパイラがコア内に常駐してつぎつぎの実習用，デバッグ用のプログラムを処理することができます（デバッグ中のプログラムはコア占有面積もすくないのでコンパイラの常駐が可能です）．こうして，ミニマムオーバーヘッドトランスレータPUFFTが誕生したのは意味あることです．PUFFTはPurdue大学のものですが，Cornell大学やWaterloo大学にも同様なコンパイラがあるようにもきいています（CORC, WATFOR）．

デバッグはいつも大変なせいか，いろいろ工夫されていて，タイムシェアリングをもちいたデバッグ用システムや，グラフィックディスプレイをもちいたシステムなどが開発されているようです．後者では，スナップショット（プログラムの途中で適当なレジスターの内容をうちだす）をブラウン管にだすものがあって，ブラウン管上にプログラムで指定した場所で，指定した数個の記憶場所の内容をだします．プリンターにだすのとちがって，ずっとはやく変化がわかるわけですが，はやくかわっている記憶場所は（一つの記憶場所はいつもおなじ場所にうつる），数字がボケて，ちょうどテレビで水泳や陸上の中継をみていると，秒の下のけたがボーっとまわってみえているようなぐあいだそうです．

リスト処理言語もあいかわらず沢山つくられているようで，日本でもそのコンパイラが完成しつつあります．この方の情報もあまりつかんではいないのですが，LISPは使用法がむずかしいのですが，LISPのコンパイラ（というかインタプリタ）がだいぶあちこちにできているようです．東大の大型計算機センターのHITAC 5020にかかるLISP 1.5がありますが，この方はガーベージコレクタがないので，あまり大きなことはできません．日立中研でもLISPをつくられたようですし，ほかにもLISPインタプリタをつくられたところがあるようです．SNOBOLは京都大学の方がやはりHITAC 5020用につくられたのがあるようです．

ところでまえからそうおもっているのですが，日本ではコンパイラつくりは大変さかんで，そのくせ言語

があまりできないようです．通研の ELIS などは言語からつくられたのでしょうが，どちらかというと，外国（といってもアメリカ）の言語をもらってきて，そのコンパイラをそのままつくるということになっています．これは言語をつくることがあまりじょうずでなく，だれかが言語をきめてくれれば，そのコンパイラはじょうずにつくれる，ということかもしれません（どうしても言語をつくらなくてはならないのは，機械ごとにことなるアセンブラです．この方はぎこちないアセンブラがおおいようにおもうのですが，やはり言語つくりに問題があるからでしょう）．しかし最近さとったのは，どうも言語をつくった人はあっても自分の言語の PR に徹しきれないため，せっかくの言語がひろくつかわれない．それに対して LISP, SNOBOL というのは，実に宣伝がじょうずで，また強引にだれかれに使用させる．LISP など MIT の大学院の学生がいろいろな言語システムをつくるのに LISP でかく，あるいはかかされる，といった事情にあって，それでますます LISP, LISP とさわがれるようになってきたのだとおもわれます．

また宣伝に必要なマニュアルを用意することも，日本ではなおざりにされていて，せっかくの言語が蒸発してしまうことになるようです．

先日，こんなはなしが，前述の京大数理解析研でシンポジウムのときにもちょっと話題になり，マニュアルかきももっと業績としてみとめられなければならないということになりました．もう一つ，前述の東大の LISP の場合で感じることですが，ほかのセンターの，ほかの言語でも同様と察せられることに，やっと LISP のインタプリタができても，わりあいに利用してくれる人がすくない．これもこんどはインタプリタ作製側の PR 不足なのでしょうが，そういう事情で，なかなかむしがとりきれません．大体作製者は簡単な，あるいは常識的な，あるいは自分で処理系に工夫をした部分のテストしかないので，一応完成したようにみえるのですが，いざ他人がつかってみるとむしが沢山発見されます（東大大型計算機センターをつかって，実に沢山の人が HITAC 5020 の FORTRAN コンパイラのデバッグに協力しました）．いずれにしても，世は PR の時代です．その点 PR とは直接関係ありませんが，前述の京都のシンポジウムで，電気試験所の西村さんの紹介された機械翻訳用に開発された言語を，みんなが利用しているというのは立派だということになりました．

プログラム言語の方があまりはなばなしくすすまないのは，すこしまえまでは，計算機がないからだということになっていましたが，現在はたしかにいまでも東大の大型計算機センターもおすなおすなですが，どうもプログラムをつくる人，さらにはリーダーの不足にあるようです（東大のおすなおすなもプログラマーやリーダーが充分いて，プログラムをたえず改良していれば，それだけでもいくらか緩和されるかもしれません）．

4. その他

あとは，最初の分類で SS といわれるところで，ぼくがおもしろいとおもった話題についてすこしふれようとおもいます．

一つは MIT のプロジェクト MAC でやった情報検索システム TIP です．これはぼくがやってみたいとおもっていて，機械も機会もなくしてやらずにいたことをやられてしまったものですが，それは文献の計算機による本当の意味の「まごびき」です．文献の最後には参照文献の一覧表がのっていて，それ以前に出版されたその主題に関係ある文献がたどれるようになっています．通常これをまごびきとよんでいますが，これをここでは以前のものにもどるので，先祖びき，または「じじびき」とよびましょう．

さて，この主題についてその後どういう文献がでたかは，別の方法によってしるよりしかたがありませんでした．計算機に情報検索用のファイルをつくった場合，文献末にある参照文献から，参照されている文献のレコードに，この文献をリスト構造でむすびつけると，以後，この文献を参照している文献はどれかということがわかるようになります．これはやはり計算機でやる種類の作業ですので，この提案をまえに国会図書館におつとめの坪井忠二先生におはなししたことがありました．そしてそのままになっていたのですが，プロジェクト MAC のプログレスリポートなどで TIP の説明をみるとちゃんとこの機能がついていました．

情報検索ではもう一つ，グラフィックディスプレイをもちいて，文献の図面から文献をさがすのがよいのではないかとおもっていますが，あまり具体的に考えたわけではありません．すでにやっているというはなしもきいてはいません．

プロジェクト MAC のリポートには，アーティフィシャルインテリジェンスのところに，計算機にテレビカメラとマジックハンドをつけて，作業をさせるはな

しがのっています[10]．計算機めがけてボールをほうるとテレビカメラでボールを追跡して，マジックハンドでキャッチするのだそうですが，これもやはり大変なのはソフトウェアです．渕さんがこのマジックハンドもみておいでになったそうなので，うかがったら，うでをそろそろうごかして，このくらいのはやさですということで，とても長島みたいにはゆかないようです．

もう一つ最近おもしろいとおもってよんだものはグラフィックディスプレイの諸問題[11]と，そのためのプログラムのはなし[12]です．グラフィックディスプレイでは，かくべき図面の情報をディスプレイファイルにいれて，ディスプレイの制御装置がこのファイルをよみ，図面をえがきだすわけです．制御装置が，良質の図面，文字をかくために複雑化して，ドットパターンジェネレータ，ストロークパターンジェネレータなどをもつようになると，ディスプレイファイルのなかみも複雑なマシンコード（CRT ランゲージ）になるという記述があります．つまり計算機を直接うごかしているプログラムが機械語だとおもったら，そのプログラムからみてもっと機械的な機械語があることになり，計算機はかきたい図面をコンパイラをとおして機械語にかえてディスプレイファイルにいれていくというみかたです．また図面は記憶装置にハイヤーレベルシンボリックピクチュアデスクリプションではいっていてジェネレータだのトランスレータだのが活躍しているということです．このあたりはわれわれのところで経験はありませんが，いわれてみればなるほどそんなぐあいになりそうだなあという気もしました．

参考文献

1) M. Schatzoff, R. Tsao and R. Wiig: "An Experimental Comparison of Time Sharing and Batch Processing" Comm. ACM, Vol. 10, No. 5, May, 1967, pp. 261～265
2) Neil Macdonald: "A Time-Shared Computer System, The Disadvantages" Computers and Automation, Sept. 1965
3) Allan L. Scherr: "An Analysis of Time-Shared Computer Systems", M.I.T. Press, 1966
4) Jerome H. Saltzer: "Traffic Control in a Multiplexed Computer System", MAC-TR-30, 1966
5) John M. Blatt: "Ye Indiscreet Monitor" Comm. ACM, Vol. 6, No. 8, Sept., 1963, pp. 506～510
6) Peter Naur: "Report on the Algorithmic Langnage ALGOL 60", Comm, ACM, Vol. 3, No. 5, May, 1960, pp. 299～314
7) DOD: "COBOL, Edition 1965", DOD, 1965
8) 日本電子工業振興協会："Minimum PL/I Subset（中間報告" 昭和42年6月）
9) 森口ほか："ソフトウェアの動向"，情報処理，Vol. 5, No. 6, Nov., 1964, pp. 354～368
10) Marvin L. Minsky: "An Automomous Manipulator System", Project MAC Progress Report Ⅲ, July 1965 to July 1966, pp.11～17, 1966, MIT.
11) Ivan E. Sutherland: "Computer Graphics, Ten Unsolved Problems" Datamation, May, 1966, pp. 22～27
12) Morton H. Leuin: "An Introduction to Computer Graphic Terminals" Proc. IEEE, Vol. 55, No. 9, Sept. 1967, pp. 1544～1552