座　談　会

タイムシェアリング・システムの問題点をめぐって

高橋延匡・牛島和夫・新田謙治郎
淵　一博・土居範久・山地克郎
亀田壽夫・真子ユリ子

出席者の TSS に関する略歴

高橋　延匡　(日立製作所中央研究所，1966～1968 HITAC 5020 TSS の方式設計に従事した．同システムは稼動中)

牛島　和夫　(九州大学工学部通信工学科，1966～1967 OKITAC 5090 H タイムシェアリング・システムの“構造を持った言語に対する対話的プロセッサ”の研究に従事した．)

新田謙治郎　(日本電気データ通信システム事業部，1966～1967 NEAC-TSS (NEAC シリーズ 2200 モデル 500 による) のシステム設計に従事した．同システムは大阪大学，日本情報処理開発センタ，東北大学および日本電気社内で稼動中，また 1968 年から DEMOS のシステム設計に従事して現在に至る．使用機種は NEAC シリーズ 2200 モデル 500.)

淵　一博　(電子技術総合研究所パタン情報部．1966～1969 ETSS の設計・開発・解析・改良に従事した．使用機種は HITAC 8400, 同システムは 1971 年 3 月まで電話回線を介して所外端末からも用いられた.)

土居　範久　(慶応義塾大学情報科学研究所，1965～1967 KEIO-TOSBAC TSS のコントロール・プログラム設計，TRC および端末装置の開発，会話形 FORTRAN コンパイラ設計などに従事した．使用機種は TOSBAC 3400-30．同システムは凍結中)

山地　克郎　(富士通第2ソフトウェア技術部，1966 より FACOM 230-60/75 TSS の MONITOR-V，会話形言語 BACCUS，コントロール・プログラムの設計に従事している．同システムは稼動中)

亀田　壽夫(東京大学理学部，1966 より 5020 TSS の研究に従事した.)

司会，編集

真子ユリ子　(情報処理学会誌編集幹事，電子技術総合研究所電子計算機部，1966～68 ETSS に関する研究に従事した.)

「TSS はもはや研究段階ではないのではないだろうか」という意見は，わが国では実際に稼動している TSS がひとつもない時代からすでに存在していた．そして現在は，どうであろうか．TSS の曙時代から第一線に立って研究および開発に携ってこられた方々に一堂に会していただき，TSS の問題点をめぐってご討論いただいた．ここに収録するのはその一部にすぎないが，その内容には貴重な体験と示唆が織り込まれている．

なお，編集者の意向により，問題点をしぼるため，意識的に設計者サイドに強く偏向した視座をとっていただくよう，あらかじめ出席者の皆様にお願いしてあったことを，しるしておく．

MULTICS の響き

牛島　1965 年に MULTICS の論文が出たわけですね．九州におりますとあの論文が手にはいったのは 66 年の 6 月ごろでしたが，それを見てえらく感心しまして，これはもう革命的なものだと思いました．たまたま大槻さんや私などが計算センタに集まっておりましたので，「とにかく体験してみよう」ということで，2，3 箇月討論してから始めました．われわれとしては，かなり一般性のあるものを作ったつもりだったのですが，いまから振り返ってみると随分と見落としていることもあり，おはずかしい次第です．ファイル関係のことなど，当時はその重要性がわかりませんでした．

司会　オンライン・ファイルですね．

牛島　はい．一方，会話形言語の処理などについては，最近の学会の発表を聞いておりましても，われわれの試みの折に問題になったのと同じことが繰り返し問題にされており，その意味で技術の交流の悪さを感じます．

司会　TSS の個別的製作に伴って，技術の成果がローカルに蓄積する傾向にあり，拡散していかないということでしょうか．

山地　それは TSS に限らずソフトウェア全般に関していえることで，数年前から社内でも問題になっています．いま，“ローカルに蓄積”とおっしゃいましたが，現状では社内でさえソフトウェアの蓄積はなかなかなされないのですよ．もちろん技術の積み上げは大切ですから，図面の管理とか，“コンパイラ作製の手引き”をくばるとか努力はしております．だが，会社同士で互いに努力の結晶を交換し合えるかというと，そうもいきませんね．いずれは日立さんが特許を公開されたように，ソフトウェアのノウハウを交換し合える日がくるでしょうが．

新田　65年，66年というお話しがありましたが，私どもの会社ではそのちょっと前に，CTSS を見てきた人が大変結構なものであるというので，ここで採用されたメモリ・プロテクションやリローケーションなどの機能をモデル 500 の設計にもとり入れました．これで TSS をやる基本的な準備ができたわけで，たまたま大阪大学計算センター設立の話しがあり，大学におけるコンピュータ共同利用形態に TSS をとり入れたらどうか，という私達の提案に同意していただきました．その結果，私達の TSS はまず阪大のユーザに利用されることを前提に開発することになりました．

私は 66 年の夏からかり出されて TSS の開発に参画したのですが当時発表された MULTICS の美しさには強くひかれるものがありました．ところが一方では，何とか早く実用に耐えるように，もの作りをしなければならなかったので，あまり冒険的なこともできず，とにかく 68 年 1 月の開所式には間に合わせました．私個人としては，その後それをベースにして，データの収集，解析，評価，さらに次期システムへの反映という問題と取り組みたかったのですが，引き続いて電電公社のデータ通信サービス*1の設計に携わることになりました．これは不特定多数のユーザに対する公共サービスという意味で，いろいろ大変な仕事なのですが，私としては最初のシステムを十分育てる時間がとれなかったことがいま思えば心残りです．

司会　K-TSS の動機と意義をお話し下さい．

土居　1964年の夏ごろ，例の赤い表紙の本*2を読んで，こういうことをやってみたいと思って始めました．しかし，すぐに金の面でひっかかってきましたが，幸いなことにたまたま TOSBAC-3400 が導入されることになったわけです．

TSS は研究として非常に大きな，汎用性のあるシステムをねらうのと，多少特殊なインハウスのものをねらうのと二つの方向がありますが，私達は後者を選びました．私自身いまでも後者に関心があります．

さて，イムプリメントしてみると，いろいろな問題があって，論文は何となく格好よく書いてあるのですが，実際となると，かなりヤクザな部分が出てくるのですね．これは会社でも大学でも同じで，つまらないところに関して，つまらない同じような苦労を重ねてしまう．さきほどお話しのあった技術の交流の悪さもありますが，けれどもそれ自体，つまり経験を積み重ねること自体が交流以前に大切なことだと思います．われわれのやったことは，MULTICS のような大きなシステムにくらべたら，ほんの一部にすぎないかもしれませんが，攻め方が異なっていたともいえます．一部分をやったにしても，その経験により汎用の大きなシステムについてもいろいろわかるようになり，ここはこういう仕方がよいのではないかという意見も出てきます．岡目八目ということもありますし．

オペレーティング・システムとアーキテクチャ

司会　5020 TSS については，お金も人も十分な状態で進められたのではないでしょうか．（笑）

高橋　ところが世の中そう簡単ではなくて，（笑）理想ときびしい現実の間には，バーチャルとフィジカルの差があるのですよ．（大笑）　しかし，人材には恵まれていました．

新田　大きなシステムというのは，とにかくキャリアのある優秀な人間が，少数で作らないとうまくできないという実感は昔から持っているのですが，実情ではこういう人間はなかなか集まらないので，若い人をたくさん入れてインターフェースがふえ，ディスカッションの時間がふえ，ドキュメントがふえる．（笑）

高橋　私は 1965 年からソフト工場に行っておりましたが，うちの島田正三さんが実際に MIT に調査に行ってきまして，その結果 TSS を会社として取り上げることに決めました．その年の FJCC で MULTICS の論文が発表されたわけですが，それを見まして，われわれ随分カッカッとしたものです．

*1 DEMOS（電電公社のデータ通信サービスのうち科学技術計算サービス）

*2 “The Compatible Time-Sharing System, A Programmer's Guide” The MIT Computation Center, 1963.

私はそのときすでに東大のモニタ 1, 2, 3 でマルチ・プログラム・ベースで動かすことは経験してしまっておりましたから，ただ端末の数だけのジョブを動かすだけでは，もう研究的興味をそそられませんでした．そこで，どうせやるならセグメンテーション・ページングのものをやってみようということになりました．われわれがそれまで OS をやってきてつくづく考えたのは，ユーザにバーチャル・プロセッサなりバーチャル・メモリなりを提供しなければならないはずだということでした．そこで，それを実現するという MULTICS の思想にほれたのです．その概念にほれてしまったのです．

そしてわれわれは 66 年から，ここにおいでの亀田さんや，高橋秀俊先生，和田英一先生と共同研究を始めました．67 年の 11 月までに 48 回ぐらい討論会を持ちました．また同時に，穂坂衛先生，大須賀節雄先生にはプログラムの内部構成について検討していただきました．

5020 TSS のスペックは大体全部，その段階で決まったといえます．そこでイムプリメントにかかりましたが，ここでも最初は理想と現実のギャップはバーチャルとフィジカルのギャップでありまして．(大笑) しかし，ものになりました．

ここで飛んで結論を出すのは少し早いようでありますが，(笑)，世の中の将来の方向は，やっぱりこの方向だろうと思っております．異論もありましょうが，個人的にはそう結論づけております．

土居　5020 TSS は，東大では DAT*3 をつけず，日立中研では DAT をつけたわけですが，DAT をつけるべきか，つけるべきでないか，良い悪いの比較をやってみようというお話しがあったそうですが，どうだったのですか．

高橋　イムプリメントは，ある場合とない場合についてやってみようということになりましたが，論理的にどちらがよいかというような討論はありませんでした．パーフォーマンスの比較については論じたように思います．DAT をつけない場合というのは，限られたリソースでマン・マシン・インタラクションの実験だけをやるのが目的でありましょうし，その場合には DAT は本質的な問題でないので無くてもかまわないということになります．

亀田　DAT，あるいはバーチャル・メモリというのは，メモリ・アロケーションの問題に対する一つの解決法をめざすものであると考えられますが，確かに東大大型機センタで作成した実験的 TSS の研究のねらいは，そのようなメモリ・アロケーションの問題ではなく，マン・マシン・インターフェースの問題，さらに割り込み処理などの観点からの OS の一般的な構成原理の探究におかれました．私自身としては，バーチャル・メモリの与える夢には大きな期待を寄せていました．

高橋　しかし，われわれのように，次期の計算機システムはどうあるべきかを追究する立場であれば，DAT をつけないシステムでは開発の意義がないことになります．その立場から見れば，DAT の必要性は非常にクリアなものであって，つけるべきとかつけるべきでないとかを議論する性質の問題ではなかったのです．

土居　パーフォーマンスの比較を行なうためには，つけた場合とつけない場合につき，両方の動作を調べなければなりませんね．そういう比較が問題意識としてあったかということをうかがいたかったのです．

高橋　イムプリシットにはあったかもしれません．具体的には，ダイナミック・リンク，ロールイン・ロールアウトを中心に検討しました．ページ・セグメント・メカニズムをソフトで行なおうとするとどうなるかを検討しました．

司会　一般的にいってメーカ内ではそういう検討の結果，ハードウェアを直すべきだということになったら，アーキテクチャを変更してもらえるのですか．

高橋　それはなんとも．われわれの場合，最初にアーキテクチャをやったのは，村田さん，中沢さん，竹屋さん達です．最初に彼らがハードウェアのイメージを出して，そのうちにソフトの検討の進行とともに，こういう機能がないとまずい，とか変更して欲しい，とか希望を出しました．変えられないところは次期システムへ活かすことにして，その回は打ち切るわけです．

司会　両者のやりとりがあるわけですね．

高橋　それがなかったら何のためにシステムを開発しているのかわかりません．

1966 年の史的意味

司会　皆様のいままでのお話しで，MULTICS の与えた衝撃とその波紋が浮び上って参りましたが，その 1966 年という時刻はわが国の計算機技術史上で重要な意味を持っていると思われますが．

*3 Dynamic Address Translator

淵　たしかに 1966 年というのは，日本の計算機の歴史の上で，大きな意味を持っていると思います．

TSS という略語ではなく，タイムシェアリングということについていえば，一つの計算機を皆で使うというイメージで，1960 年ごろから技術者の間では語られていたわけです．しかし，そのころの私の当面の関心は別のほうにありました．

司会　この発言は歴史研究会の資料になります．(笑)

淵　つまり，60 年代の初期においては，いろいろな計算機を作ってみるということがわが国では技術的な課題であったし，私自身のおもな関心もそこにあったのです．それで，ソフト・ハードといいますか，計算機方式のことをやっていたわけです．

TSS に対しても興味を持っていたのですが，それはおもに計算機方式の立場からでした．1966年に ETSS を始めるまでは，OS のソフトウェア作製の経験は，ゼロに近いといってよかったのですが，しかし，個人的な思い出をいえば，シミュレーション言語に興味を持ち，1964 年ごろでしたか，SOL をイムプリメントしてみようかと思い，作り方をひとりで考えてみたことがあります．そのとき，TSS の作り方については大筋はこんなところだろうと，わかったような気がしました．CTSS の話しは前から聞いていて，タイム・スライスですか，私達はフォースド・タイム・シェアリング*4といっていましたが，そのアイデアには感心していました．

1965年ごろは，アメリカを見てきた偉い人達がTSS について騒ぎ始め，MIT が MULTICS のプロジェクトをはなばなしく打ち上げたこともあって，TSS ブームが起こったんですね．ちょうどそのころ，私達の方でもプロジェクトの話しが持ちあがっていて ETL*5 でやることとしては TSS だろうと思ったわけです．そして ETSS を始めたわけです．

そのとき，TSS などはもう研究所ではやる必要がないという声もあったし，大変だから止めとけという話しもあったのですが，ETL はソフトウェアの伝統がないので，ここで技術的な基盤を作っておかなければならないだろうと相磯さん達と大決心をしました．範囲を限ればそう大変ではないだろうとも，内心思ったのですが，しかし，やったことがないことは不安でもあるし，大変だろうという同情を幸いとばかり，まず人をたくさんもらいました．新人ばかりでしたが．(笑)

TSS は大問題だから本場の MULTICS の話を聞いたらどうか，という話が上からあり，それも幸いと MIT に講師をお願いしました．はじめは，偉い年とった人がくるかと思っていたら，若い，本当に若い人がきました．(笑) その人がソルツァでした．実に名講義でしたね．ソフトウェアに対する見方がスッキリとして，ソフトウェアとはなかなかよいものだ，と再認識しました．この講義は，うちばかりでなく，ほかのところでも TSS の研究の実質に大いに寄与したと思います．

＊＊＊＊＊ こもごも語られる思い出の中に，鮮やかに浮び出る MULTICS，それを一層印象づけたのは 1966 年 8 月，MIT の Jerome Saltzer 氏の来日であった．ときに彼は 26 才，MULTICS の Traffic Control に関する研究で Ph. D を得たばかりであり，当の博士論文の内容を含めて MULTICS の全貌を見事に語り尽くした．その名講義ぶりは，MIT において teaching assistant 時代に teaching award (教え方が優秀であることをたたえる賞) を与えられたのもむべなるかなと思わせるものがあった．この講義は多くの若い研究者のその後の活力の源泉となった．＊＊＊＊＊

かつての展望を顧みて

司会　1966 年の TSS につき，技術と研究段階ということが話題に出ましたが，TSS に限らず研究所で仕事をしていく場合，それを握っているべき最適の期間というものがあります．いつまでも執着してそれを離しそこねれば，研究所としての機能を果たせなくなるし，世間一般の技術レベルが高まらず，機が熟さないうちに早過ぎて手離してしまえば，次期の研究への基盤がくずれてしまいます．その点 1966 年における決断の是否，そうしてその時点における「TSS を研究すべき期間」の見通し，そしてそれらが果たして狂ってはいなかったのかどうか，をお聞かせ下さい．

淵　1966 年という時点は，ETL のような研究所で TSS を手がけるとすれば，ほとんど最後のチャンスであったといえます．もう 1 年遅ければ，意味を失っていたでしょう．ほんとはもっと早くやるべきであったものを金もない，人もない，体制もない，で手がつけられなかったところを，やっとギリギリですべりこんだという感じです．新人ばかりだし，しかも ETL にはシステム作りに向かない体質もあるので，ETSS は成功するはずがないというウワサもあったようですが．

次に，ETSS を始めた時点では，世の中全体に楽観的な見通しがあって，数年のうちに経験を積んで何がしかの技術的な寄与を果たしたら，そのあとはちゃん

*4 forced time sharing
*5 電子技術総合研究所，当時の電気試験所

としたコマーシャル・ベースのシステムに乗りかえるつもりでした.

しかし,これはうまくいきませんでした.それでは66年における見通しが狂っていたのかといわれると……弱りますね.見通しが甘かったのでしょう.ETSSは予定どおりいったが,わが国の TSS ビジネスはいまだに軌道に乗っていないし.システムのスイッチには所の方針がしっかりしている必要もあるし….

とにかく 66 年というのは非常な転機であったと同時に,最も楽観的な時代であったと思います.これはMULTICS 自体についてもいえることですし.

司会 さきほど,高橋さんが OS ソフトウェア作成過程から次回のアーキテクチャ設計へと意見が伝えられることがなければ,OS の研究は無意味であるとおっしゃいましたが,ETSS の場合はそれは果たされたのでしょうか.ETSS 作成中には,担当者の間ではハードウェアに対する不満,そして改善への望みがいろいろあったのですけれども,具体的な提案として,まとまった形を成したとはいえないように思います.最近,IBM システム 370 のマニュアルを見ましたところ,ETSS 作成中にハードウェアに対して感じた夢のいくつかが実現されていると思いました.たとえば,RAS です.

淵 私自身も,もともと方式屋だし,ハードについていろいろ意見がありますが,日本ではなかなか実現しませんね.メーカの体質というか,方針もあるようだし.

土居 動作解析などからシステムのオーバヘッドを証明していくことが大切ではないでしょうか.

淵 システム評価のためのデータを積み重ねていくことにより,ハードウェアを改良していくという努力は重要です.

システムが大きくなりますと,万事組織で動くようになりますから,「自分に任せてくれればこう直すのだが」と個人の力で発言したのでは無力であって,蓄積したデータを持って向かうのでなければ組織は動かないのです.これが現実ですね.

しかし,データ万能あるいは多数決的なやり方で,スッキリしたシステムができるとは,私は思いません.具体的なデータの蓄積の上に立って,個人の創意やセンスがいかされるという形でなければならないと思います.その意味ではわが国には硬直した組織が多いように思います.わが国でもメーカの技術担当者はいろいろ改良の意欲を持っていると思いますが,それが十分結実するような組織になっていないことが多いようですね.

それはともかく,データを集めるということは非常に重要です.「ETSS の解析*6」なども,そのような力になればよいと思います.

新田 メーカからレポートを出す前に,淵さんに「ETSS の解析」を出されたのは非常にショックでした.ああいうものは,本来メーカの側のほうがきちんと追究しなければならないのです.

いったんできあがったソフトウェアを改善することの重要さを私達が認識したのは少し意味は違いますが,IBM が FORTRAN G から FORTRAN H にかえてオブジェクト・コードの最適化をはかり,数倍のパーフォーマンスをあげたという実績を作ったことが発表されたころだと思います.大形ソフトウェアという,開発に金がかかり長期間使われる耐用生産財に,少し手を加えることでコスト・パーフォーマンスを大幅にあげられるとすれば,これは投資効率という点でメーカが目をつぶるわけにはいかない.

システムの解析,評価,精製

司会 自分達が作ってきたものであっても,できたものを性能測定してデータを解析し,こつこつと改善していくより,次のシステムの設計製作にとりかかるほうに魅かれるのが,一般的傾向ではないでしょうか.これは自分自身の反省でありますが.(笑) 最初のシステムの合成に比べて,解析後の再合成に対しては,世間でも低く評価されすぎているのではないでしょうか.

山地 世間一般のことは知りませんが,メーカの立場からは,少なくとも当社では,チュンアップ*7は非常に重要であると認識されてきております.その理由の一つはアーキテクチャ部隊との関連です.アーキテクチャ部隊を説得するのがむずかしいのです.それはさきほど,淵さんがおっしゃったような意味でのむずかしさもありますが,OS 作成部隊がアーキテクチャ部隊を説得しうるだけの思想を示せないからではないか,と私は考えております.ハードウェアに対してなぜこうして欲しいのか,というフィロソフィを示すことができない.つまり,OS 作成者自身がよくわかっていないのではないか.たとえば,マルチ CPU にするといって2台がよいか4台がよいか議論しても,

*6 淵,弓場,他:タイムシェアリング・システムの解析および評価について,第11回プログラミング・シンポジウム報告集,C 65.
*7 tune up

なぜ2台がよいか4台がよいかわからないのです．それは結局データがないからです．

私達のアーキテクチャ部隊とソフトウェア部隊との結び付きは，5年くらい前から始まっております．昔はハードウェアのスペックがきてから，あわててソフトを作っていたのですが，いまでは両者の第一線の者達が集まって，どういうものを作り出すのか討論します．そのような場のあることが，データの重要性に対する認識が高まってきている一つの理由となっています．もう一つは，ハードウェアにしろソフトウェアにしろ，暗中模索で次から次へと新しいものを作り出す時代が続いてきたのが，ようやくおちついてきたということです．以前のような時代では，とにかく何とか動かすということで精一杯だったのですから．それから，計算機が大型化するにつれてOSの寿命が少しずつ長くなってきて，チューンアップの価値が出てきたと思います．CDCは6000シリーズに10年かけているし．そうなるとチューンアップするほうもやりがいがあるわけだし，マンパワーの投資効率も上がりますね．ですから，チューンアップはこれから一般的にも盛んになり，皆が当然やることになるのではないでしょうか．

＊＊＊＊＊1970年10月，あのJerome Saltzer氏がアメリカ・コンピュータ・ショウに出席するため，再度来日した．そしてMULTICSの研究経過と現状について，各方面で報告を行ない，“MULTICS健在”の印象を残して去った．＊＊＊＊＊

MULTICSの精製

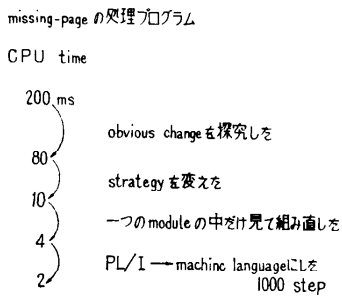
新田　この前ソルツァが来日したとき，この1年間に108回システムを更新したというのです．108回というと1週間に2回ですね．私どももシステムのデバッギングの途中で週に何回かシステム・ジェネレーションをやっておりますが，そのためにクローズド・タイムをかなりくってしまいます．1週間2回も数時間のロスを出すのは大変なことだと思って念のため，それにどのくらい時間がかかるか尋ねてみますと，5分くらいだというのです．そこでハタとわかりました．ダイナミック・リンキングのおかげなのですね．

高橋　ダイナミック・リンキングだとデバッグが非常にやりやすい，実感としてそう思いますね，私達が非常に少ない人数でシステム開発ができるのは，PL/Iとダイナミック・リンキングとTSSの三種の神器のおかげだといえます．

ダイナミック・リンキングだとパーフォーマンスが落ちるという話しを聞きますけれども，あれはうそだと思います．むしろ，ダイナミック・リンキングのほうがスタティック・リンキングより簡単だと思いますね．両者のプロシージァを比較したら，ダイナミック・リンキングのほうがはるかに小さい．

亀田　一部分のモジュール間のリンケージの形式にまちがいがあっても，当のモジュールが呼び出される段階になるまでは，ほかの部分を動かすことができるという点でも，ダイナミック・リンキングは非常に有利ですが，リンケージの誤りが初めに見つかるというスタテイック・リンキング方式の有利さを経験したこともありました．

司会　先日ソルツァが2度目に来日したおりに，MULTICSにおいてチューンアップによりめざましい成果を上げた例をうかがってみたところ，こんな例を示してくれました（**第1図**）．これによるとCPUタイムで100倍くらい速くなっているわけです．全体的に，どんなモジュールも最低3回書き直したそうです．



第1図

新田　PL/Iコンパイラの改良も含めて，3年間で3回書き直したおかげで，能率が10倍上がって，メモリ・サイズが10分の1になったということでした．

司会　チューンアップすれば，速く，簡単に小さくなる．（笑）

新田　ソルツァの来日のおり，実はチューンアップに関して非常に面白い話しがあったのです．ある人が「もしプログラムがアセンブラで非常にアイデアルに書かれていたとしたらPL/Iと比較してどうだろう」と聞いたのです．するとその比は無限大だと彼はいうのです．というのは，アセンブラで書いたらまだ働いていないだろうから比較できないという意味なのです．日本人というのは，できるにしろ，できないにしろ，まずアイデアルな仮想状態を頭に設定して，それとの比較で価値を云々しますが，アメリカ人というのは，まず作り，次第にそれを精製していく．そこにタ

イム・スケールの意識がある．それは非常に学ぶべきだと思いました．

＊＊＊＊＊それにしても限りなき MULTICS の影．この5年間，あるときは濃くあるときは淡く．だが，さらに話題がすすんでいくと，全くそれに疑問がないわけではないのである．＊＊＊＊＊

システム記述言語をめぐって

亀田　MULTICS の最初の研究課題の一つに，PL/I でシステムを記述することがあったわけですが，さらに機械語で書き直す程度のチュンアップをやらざるを得なかったというのは，システムの基本設計のどこかに無理があったのでしょうか．（第1図参照）それから，チュンアップは速く簡単に小さくするばかりが能ではありません．やり過ぎて，かえって記述の見通しを悪くしてしまってはダメで，どこかに最適のレベルがあると思うのですが．

司会　高橋さん，PL/I による記述を試みられたわけですが，PL/I に対する評価はいかがでしょうか．

高橋　よいか悪いかは別として，私どもメーカで新しいシステム記述用言語を開発するということは非常に冒険があると思うのです．言語を設計するのは格好がいいし，ペーパの種になるといっては何だが(笑)，そのつもりになればいくらでも言語は作れる．しかし，われわれメーカの立場としては，さきざきその言語をサービスしていかなければならないから，慎重で保守的であらねばならない．

土居　自分から見た使いやすさばかりではなく，普遍性のある皆に受け入れてもらえるものを提供しなければならないのですね．

高橋　そうです．新しいシステム記述言語を開発したとき，PL/I という既成の言語に比べて，それがどのくらい価値があるかということは，ある同じシステムを両方で記述してみなければならないわけですが．それだけの金と人はない．したがって，私は PL/I でやって来た．

亀田　人間の言語になぞらえると，エスペラント語で書くより英語で書く，ということですね．

司会　システム記述に関するメーカの現状をうかがわせてください．

山地　現在は圧倒的にアセンブラですが，システム・ランゲージも使いはじめています．SL 45，SPL 60 など．しかし賛否両論でいまのところ何ともいえません．

新田　私のところもアセンブラで非常に苦労してきました．それと，現在苦労しているのは内部スペックの記述で，いまやっている電電公社の仕事については数万ページもあり，書くのも大変だし，小数の人間がたん念に目を通すことも大変です．それに日本語の記述はやはり曖昧です．トラブルの多くが記述の曖昧さによるインターフェースの誤りです．かつて，シングル・ドキュメント・アプローチというのを試みたことがあります．これはドキュメントとプログラムを一緒に書いてしまう，簡単にいうとコメントが多いプログラムということです．ところがカード・パンチを使う関係上，このコメントは英語で書くわけですね．それも勝手な造語まじりの英語で．そこでまた曖昧になってくる．

山地　そういう言語の面の未熟さとともに，いまはあまりに OS 自体が構造論を欠く．そのため OS の記述といっても，どのくらいに焦点をあてて言語設計したらよいのかわからないということがいえると思います．モニタ部のファームウェア化にしても理論の欠除がネックになっているといえる．

亀田　システム作成グループの一員が作ったサブシステムの外部仕様が，ほかの人の作ったサブシステムとのインターフェースになるわけですが，その外部仕様を皆がよく知っているようなものに似せると，皆が“気の合った”状態になって，曖昧さ，了解の悪さによるトラブルが少なくなると思います．われわれの場合でも実際にそのようなことを経験したことがあります．ここで，皆が共通してよく知っているものが多いと，ずっと仕事がやりやすくなると思います．このような共通の理解物，概念として便利なものを発見し整理し，習得しやすい形にすることが，プログラミングの理論化の一つの方向であると思います．

OS の構造論

司会　OS の構造論もまた，いまお話しがあったような“共有概念”に関係すると思いますが，構造論の立つ見通しについてお話しください．

土居　ひと口に OS の構造といってもさまざまで，たとえば亀田さんの論文[*8]に示された構造法がありますが，あれに全員がモロ手をあげて賛成するかというとそれはうそだと思うのですよ．

亀田　たしかに各メーカのシステムはお互いに異なっていますが，よく検討すれば“システムAのこの機能はシステムBのこの機能に対応する”というよう

*8　亀田：オペレーティング・システムの構成原理について，オペレーティングシステムズ・シンポジウム報告集，情報処理学会，pp. 64～79.

に対応性が見い出されます．システムごとに具体的な表現は異なっていても，本筋としては皆同じようになっていると考えられます．たとえば，この前の9月のOS シンポジウム*9でもお話ししたことですが*8，割込み処理という一面をとってみても，システムごとにいろいろな表現がとられていますが，共通の基本概念を用いて記述できるようです．またタスク間の通信や，ジョブ制御言語などについては，メーカごとに異なった表現がとられていますが，満足すべき共通の必要条件を抽出して，プリンシプルとしてまとめることができると思います．

山地　しかし，現実にはOSを作るたびにいちいち記述から考え直しているのですから，テクニックの蓄積さえうまくいかないわけです．ここいらで考え直さねば．「大体，コンピュータなんてものは，この辺でいいんだ」というものがおいおいにふえていかなければと思います．せめて，できそうなところから手をつけていかないと，構造論どころかシステム記述もうまくいきませんね．

高橋　現に，山地さんのおっしゃったようなものがふえてきていますね．たとえば，360 におけるプログラム・ステータス・ワードのようなものは，マルチプログラムのOSの開発をすれば，誰でもああいう形にまとめたくなるものでしょう．

山地　そういうものがふえていけば，OS の構造を論ずることもあながち夢ではないと思いますね．

淵　構造論になれば，構造を論じるための言語，プログラミング言語ではなく人々が使うことばとしての言語，ランゲージ・フォア・ディスコース*10ですか，それがかなり統一されていかなければなりません．マーケット・シェアの理由で IBM 流のことば使いがかなり支配的なのが現状ですが，大学などがOSを手がけるようになれば，体系付けとそこで使う言語が洗練されてきて，多くの人々の間で話しが通じるようになるでしょう．MULTICS の意義の一つは，そういうところにもあると思います．

亀田　メーカには機密があり，交流がやりにくいので，研究所や教育機関が媒介を果たすという意味でOS をやってみる意義があると思います．

牛島　シミュレーション言語は共通の言語として，ある程度役に立つと思われますが，その可能性はどうでしょう．いまから考えると，あれは実によくできていると思うのですけれど．シミュレーション言語の中には，OS の構造論と評価論において価値のあるものがあると思います．

司会　それについてはあちこちで関心が持ち続けられているので，そのうち成果が出るかもしれません．

ソフトウェア危機について

司会　TSS からははずれますが，ちょっと話題を変えまして，ソフトウェア危機*11についての皆様のご意見を交換していただきましょう．

土居　ソフトウェア危機ではなく，問題はシステム全体のまとめ方にあるのですから，その意味では計算機システム全体の危機というべきではないでしょうか．

新田　現在では，ソフトウェア生産が膨大な人をくってしまうこと，その割にはソフトウェアが代価を支払えないのではないかというマネージャとしての危機ならわかります．ただし，ソフトウェア生産性の向上については，まだまだ改善の余地も見通しもあり，私はむしろ希望を持っています．

淵　人の膨張は経営者の感ずる危機ですが，しかし私の見方からすると，計算機産業は全体の規模がまだまだ飛躍的に拡大する可能性を持っていると思います．それは1桁などというオーダではありません．人がふえるのは当然です．ただ，そのための対策，のびのびと人が育つための土壌がふんだんに用意されていないという現状には危機を感じますね．

牛島　私はプログラミングの自動化・機械化といってもよいかも知れませんが，これの困難性についていうと，自動化を実現するためにはそれ自体において，“アセンブラ言語的な努力”をしなければならないような気がします．

高橋　危機論ではいろいろなことが課題になりますが部品の点数が10の5乗とか6乗とかを越えると，すべてシステム的にものを考えなければならない．プログラミングにしても100Kくらいのものはどこでも作っています．すると徹底的なシステム・アプローチが必要であるが，技術的にはその段階ではない．こういう意味で危機を感じます．

亀田　私には，大きなシステムに対する需要があるのに，技術的にできない，というところにソフトウェア危機のイメージがあります．いま，高橋さんのお

*9　1970年9月2日～5日
*10　a language for discourse.
*11　高橋秀俊：ソフトウェア危機，情報処理，Vol. 10, No. 6, 1969, pp. 373～374.

っしゃったシステム的アプローチ，あるいは多人数のときのインターフェースというように問題がはっきりしてきたということであれば，これは数年以内に解決するかあるいは当分の間停滞かのどちらかだというところかと思います．すなわち，20年くらい前に，原子力産業が非常にバラ色のものとして語られたことがありますが，現実にはなかなかそうなっていかないようです．計算機産業もこれと同じことになるのかも知れません．

土居　それは教育の問題になるのではないでしょうか．人間に対する需要が満たされない，つまりソフトウェア人口の底辺が満たされないという現状が打壊されれば……．

亀田　私の申す停滞の場合とは，そういう教育が全部うまくいっているのに，ただそれに対処するための技術がまだでき上がっていなくて停滞する，ということです．それからもし危機あるいは停滞が解決されるとしたら，アインシュタインのような英雄の出現によるのではなくて，どこかの組織が，組織的な活動によって停滞を克服するような技術を開拓し，蓄積するのではないかと思っております．

山地　私は皆さんと違って，日本にはまだソフトウェア危機はきていないと思います．というのは，計算機産業はまだのびつつあるのですから，いずれ危機にぶつかると思いますが，技術が弁証法的に発展するのは大きな危機を乗り越えることによるのですから，私は，早く危機がくるとよいと思っております．

司会　「冬よ早くきたれ」という表現による春を待つ心ですね．

潮流の中で

司会　それでは最後に，計算機技術，特にOSの技術の流れにおいて，TSSはいかなる役割を果たしてきたのでしょうか．また，今後 TSS は計算機技術全体に対して何を提供しうるでしょうか．

淵　これはむずかしい質問です．一つには，TSSでは非常に複雑なシステムを実現しなければならないという課題があって，これは技術を育てるよい土壌だったわけです．一方，いままでに使い方に応じて，それに合わせたいろいろなシステムが存在しうるということになってきたことからすると，TSS だけが，あるいは MULTICS だけが最適のシステムであるとはいえない．かなりよいが，最適ではない．したがって，それだけが生き残るのではない．とすれば結局 TSS をめぐって様々の複雑なシステムを実現する過程において生れた技術こそが 70 年代への遺産となるということではないでしょうか　これが 70 年代に様々なシステムを実現させる“もと”になるわけですから，

高橋　私は非常に単純に考えまして，汎用の大形の TSS は，実は OS の典形的なものであると，思っています．

司会　そうしますと，OS の標準イメージが TSS であるとお考えですか．

高橋　ええ．私はそういうイメージを持っております．これからの OS については，TSS にするかバッチにするか，という議論は自然解消するでしょう．そのときどきに応じて，リモート・バッチとかいろいろの形態をとり得ますけれども．

司会　それらは標準形としての汎用 TSS の縮退形とみなすのですね．

亀田　それから，原子核物理学の計算など，OS と呼べるものをほとんど必要とせず，CPU が計算に専念しているようなものも縮退形といえるでしょうね．

新田　TSS を OS 技術の標準形とみなすのには賛成ですが，個々の計算センタで用いられる OS が，汎用 OS からの選択的なモジュールの組合せで作られたもので運用されるかどうかには疑問が残ります．むしろ通信ネットワークとの結びつきを考えるとき，個々の計算センタはそれぞれ得意不得意な点で特徴を持っていて，ユーザはダイアルを通して業務目的に応じてセンターが選べるという形に発展していくと思うのですが……．

高橋　TSS といったときの各人のイメージの違いがあると，さっきから思っておりました．私が TSS といっているのは，タイム・シェアばかりでなく，データのシェアを含めた情報システムを指しているのです．その点で，単に計算の手段としてミニコンを幾つも提供するのと大きく異なります．

山地　タイム・シェアというのはよくないですね．むしろ，リソース・シェアというべきでしょう．

司会　これは，初めにことばの定義をよくしておくべきでした．（笑）

牛島　私も座談会に出席する前から TSS の定義には，ひっかかるものを感じておりました．（笑）

飛行機がうちに落ちてからは*12ユーザになってしまいましたが，マニュアルというのは，われわれが見てさえなかなかわかりにくいものですね．それでもわ

*12　米軍機が九大の計算センタの建物に落ちた件をさす．

れわれはシステムの内部をよく知っているので，かなりの推察をしてうまく使いこなしていると思います．計算機の操作方式の過程では，バッチ処理方式でユーザがコントロール・カードの使い方を覚えていったように，TSS については今後ユーザの質が高まっていくことが必要だと思います．

高橋　OS 設計者の立場からいえば，技術的にファイルをどうマネージメントするかという問題よりずっと大切なことは，プロテクションを代表とするような問題です．これからは，ビヘビアル・サイエンス的なアプローチを真剣に考えなければならない時期であると思います．

司会　新種の公害についての技術者の責任ですね．

高橋　ゆゆしきことになる可能性があります．今後のハードウェアについていうといままでは，ギブソン・ミックスのアプローチで追っかけてきたのですが，極端にいえばその時代は去った，といいたいですね．

将来の計算機は大形のものを回線を通じて家庭電器を使う感覚で使うようになることには皆さん異論はないと思いますが，そうなると発電機に相当する計算機が故障したり，ダウンしたらどうなるか．もうこれはとんでもないことになります．われわれ設計者は，いままでとは質的に考え方を変えて，事を運ばなければなりません．

司会　皆様どうもありがとうございました．

(昭和 46 年 3 月 24 日受付)