セッション3
ワークステーション＋ネットワークによる分散環境の構築

チェアマン　　中野秀男（大阪大学）

プレゼンテータ　加藤朗（東京工業大学）

酒匂寛（ソフトウェア・リサーチ・アソシエイツ）

西岡健自（横河電機）

－　内容　－

- 事前アンケート
- プレゼンテータのポジション・ペーパ
- セッション・レポート
- 事後アンケート
- セッション・サマリ

開発環境の土台がワークステーションを中心としたものに変化することにより、必然的にワークステーション間はＬＡＮによって結合されてきました。しかし、単に物理的に結合しファイルの転送ができる程度では、資源の無駄使いといわれてもしかたがないでしょう。一時代前のミニコンに匹敵するハードウェアを各個人が１台ずつ手にしたならば、より快適なソフトウェア開発環境を目指して、工夫をしないわけがありません。分散環境構築の成功／失敗の例、現在構築中のもの、構築プラン等のあくなき追及の実体を披露してください。

熊谷　章／ＰＦＵ　研究開発センター第三開発室

◆不要になりつつある WS をどのように活用するか

WS の発達は激しく、購入して２年目位から誰も相手にしてくれない WS が出現し始める。例えば、SUN2 とか 1100SIP がそうだ。減価償却が終了していない陳腐化した WS は多分全国に散在していると思われる。現在我々には、ファイルサーバと教育用としてこれらを活用しているが、ここ暫らくはこの問題が WS につきまとっているように思える。まだあった。Apollo Domain DN-570 である。これも使い道がない。

◆分散システムのアーキテクチャ

セッション１とどのような関係が必要か。分散システムのハードウェアとソフトウェアのアーキテクチャをどうするか。このテーマで、"beyond UNIX"の形で考えてみたい。最終的には、社会構造の中の家庭や個人生活のようなアーキテクチャになりそうだが、現在からそこまで行く間には沢山の段階的なモデルがありそうだ。

岡本隆一／神戸コンピューターサービス　ソフト開発部技術第一課

◆Σワークステーションは本当につながるか…？

関係各社からボチボチとΣワークステーションが出回りだしたが、どうも方言システムが多発しているように感じてならない。本当に、機種を問わずに仲良くつながるのだろうか<例えば、A社の WS とＢ社の汎用機（コミュニケーション･ツール）>…？

◆多国籍（？）ファイル･システムの必要性について

異機種間をネットワークすることは、わりと一般的になってきたが、それぞれのコンピュータが入出力したファイルの互換性については、まだまだである。この問題を解決できる秩序の実現も望まれるが、それ以前に、コードの異なるファイルをハンドルするファイル･システムの実現が必要だと考えている。

中野秀男／大阪大学　工学部通信工学化

今、私の回りがネットワーク＋ WS の話ばかりで、すべての話が全部もしくは一部の相談がまわってきます。このごろ、少し責任を感じてきましたが、私には思っているまま喋って欲しいようなので、難しく考える事なく、jus や sea での話の中でのこれは正しいと思うことを、いいですよと広言しています。DEC や xerox や TANDEM などの（sra も）ネットワークを見たり聞いたり、所属するのが通信ですからデータ通信の研究の流れからいろいろ対象とする機関や企業の技術力や資金力をみて種々の形態を考えています。セッションの座長らしいので、既に構築しているところと、これからの所を考えながらやっていきたいですね。宜しく。

藤野晃延／富士ゼロックス情報システム　技術部

◆分散環境の問題点

XNS の Network では、Network 上の Server が Common な strage になっている。従って Access 権等を比較的 Rigorous に設定可能である。一方、NFS 等は state-less といった強みを持っている反面、Security の面では殆んど無防備である。この辺、両者の機能を比較して、分散環境の標準的な機能スペックも考えたい。

◆ IPC の実現方法と分散 OS

分散環境では IPC/RPC は不可欠となる。しかし、OS としての UNIX では process-context-switching に over-load が懸る。多重プロセス空間方式の OS では、こういった demerits がある。一方、単一空間方式では共有メモリ等が簡単に扱える反面、Address-Fault に対しリスクが大きくなる。分散 OS/IPC はどうあるべきか？

久保宏志／富士通　システム本部パッケージ企画統括部

◆地方にいて首都圏在住宛客のためのカスタムソフトを開発するための専用略字分散開発環境

長野にあって、受託開発に従事している人の環境へのニーズを聞く。

◆メインフレームの莫大な資産の、ワークステーションへのオフロードパス

実在の情報システムに即して、問題提起をしてくれる人がいるとよい。アプリケーションは、開発環境にこだわら

なくてもよい。

岸田孝一／ＳＲＡ

◆ネットワーク環境全体の Administration をどうするか？

いま、私自身の周囲にも LAN とたくさんの Workstation が導入されて、かなりの混乱が生じている。

とりあえずは、現状の問題点を拾いだして、それらの重要性に Priority をつけ、解決の方向を探るしか手はないのか？

◆分散環境での共同作業について

これは、セッション１ー議題１と同じである。

ソフトウェア開発の本質がチームワークであるなら、そのための有効な支援を早く考えるべきだ。

御喜家貴子／横河ヒューレット・パッカード　マーケティング部門

◆まずは、ネットワーク化環境で遊ぶためのマナー（意識）改革から…

分散環境を構築できるようになり、個人（ユーザ）の立場では、コンピューティングパサーを専有できるという点で TSS 時代から比べると環境は充実してきている。しかし一方、個々のワークステーションをネットワークでつないだという分散環境は、単に TSS 時代の 232c ケーブルが Ethernet に変化しただけでかえって、システムの保守・管理の面では、システムの数が１台から複数台に急増し、工数の増加を起こしている。個人（ユーザ）にとって、複数のシステムを意識させないで作業できる環境を思考する方向に技術は進歩しているが、とりあえずは、ネットワーク上での作業に対するマナー（常識）についての意識改革が必要ではないかと思う。

伊野　誠／ＳＲＡ

◆ UNIX は分散環境 OS になりうるか

分散環境に望むものは何か。プリント・サーバと充分大きなファイル・サーバが利用できているとすると、AI や知識ベース・システムのように大きなメモリーと CPU パワーを必要とするプログラムをうまく走らせてほしいということである。そのために OS は分散環境内の全資源の管理と空いている資源を有効利用するための負荷分散機能を持たねばならない。

森　幸一／ＰＦＵ　開発企画部ソフトウェア検査課

◆マルチメディア情報の有効利用

ワークステーションと LAN によるネットワーク環境は、マルチメディア情報の流通を可能にしてくれる。開発環境におけるマルチメディア情報が適切に利用されれば、非常に大きな効率が期待される。

◆セキュリティ対策

情報の共有のために全ユーザをあるホストにアクセス可能とすると、他の個々のホスト間では直接アクセスできなかったホストでもこの共用ホストを介してアクセスされる危険がある。こうした場合に有効なセキュリティ対策を考えたい。

村川貴広／ HST

◆分散環境の中の集中

現在、分散環境と呼べる環境の中で仕事をしているのですが、確かに、分散した資源をあたかもひとつの集中した資源として利用できるのは喜ばしいことですが、逆に同一の資源が分散してしまうのは、管理面からも困ったものがあります。ここで、何が分散できる資源で、何が集中しなければならない資源なのかを明確にする必要があると思います。わたしはこのテーマについて、さらにつっこんだ討議をしたいと考えています。

西岡健自／横河電機　研究開発４部第１研究室

◆プロジェクト共有資源の分散対応

頻繁に参照される共有資源（共通の include ファイル等）は各ワークステーション上に置くことがパフォーマンス上必要。そのような各ワークステーション上で重複する資源をどのように管理するか、重複しておくべき資源にはどのようなものがあるのか、対応を考える必要がある。

斎藤信男／慶応大学　数理工学科

◆究極の分散環境

分散環境（特に大規模）には、本質的にどんな問題があるのか。メール、ファイル、ロードバランス、セキュリティなどについて、もう一度議論してみたい。

◆マルチメディアワークステーションの展望

（マンマシンインタフェースと同じ）

和田　勉／長野大学　産業社会学部

現在長野大学に構築中の新計算機システムのうち、教育用サブシステムは VAX86001 台を中核とする集中システムですが、もう一つの研究用サブシステムは、SUN＋数台に加えて Xerox1121 や MacII やターミナルサーバをイーサネットでつないだ分散システムです。しかしい

くつかの先進的な大学・研究所・会社ですでに構築しておられるものに比べて、特に目新しいところはありません。強いて言えば、まだあまり使われていない（らしい）Thin イーサネットケーブルを使い、それを DEC のリピータ DEMPR を介して標準イーサネットケーブルにつなぐ予定である、ということぐらいです。

佐原　伸／ＳＲＡ

◆分散型オブジェクト指向言語

ワークステーションとネットワークによる分散環境は、実行環境としても注目すべきではないか？

そのための基盤として「分散型オブジェクト指向言語」が必要になると思う。

三浦あさ子／ＳＲＡ環境開発部

◆分散環境の構築の成功

とはどういうことかが知りたいと思う。

歌代和正／ＳＲＡ　環境開発部

◆余っているマシンはどうしましょう

たとえば、休日出勤をしたら１００台のワークステションが余っているけど、自分のマシンだけフルに動いているという事態が起こってしまいます。こういうのは有効に利用できたほうがいいですね。

◆分散環境というのはよいものか

私は、はっきり言って自分の机の上にマシンは置きたくない。うるさいし、あついし、管理はしなくちゃいけないし、バックアップはとらなきゃいけないし、あまり良いことはない。速いマシン一台をみんなで使ったほうがよっぽどいいことがあると思うのだが...

桜井麻里／ＳＲＡ　環境開発部

◆個人、グループ、そして、全体のバランス感覚

人とマシンが一対一の時代になっても、人間が集まって小人数のグループをつくって仕事をするという状況は変わらない。そしてそのグループが幾つか集まって一つの組織となり、それがあつまって...となっていくわけである。だから、情報の交換を必要とする度合いに合わせて、うまくグループを作りそれにしたがってネットワークを構築していけばよいわけで、そのへんのバランス感覚が重要であろう。むやみにファイルの共有をしたりすると、誰が面倒を見るかとか、そこがこけたらどうするかとか、アクセスに時間が掛かるとか、が問題となるから、扱う人間とマシンのことを考えて無理のないように構築しなければならない。ネットワークがこけるとみなこけるという状況がある以上、ネットワークのグループの切りわけの問題は無視する訳には行かない。

◆分散環境の運用・構築に関するよもやまばなしよもやまばなし

分散環境を構築すると言っても、かな漢字変換サーバーとかプリンターサーバーとかニュース用のファイルサーバーとかゲートウェイマシンとか、それぞれのマシンにに役割を持たせて、サーバーとしてそれをみんなで意識せずに使うという話と、一緒に仕事をする仲間の中で、自分達のプロジェクト間の情報交換を密にし、プロジェクトのリソースをうまく管理していくという話と二つがあるように思う。前者に関しては、組織単位で計画を立てて構築し専任者が面倒をみていくのがよいし、後者に関してはプロジェクトもしくはグループ単位で責任を持って面倒を見ていくしかないであろう。私の経験上、ひとりでまともに面倒を見られるワークテーションの数は、せいぜい２台である。それ以上になると、面倒ばかり見ていて仕事ができなくなってしまう。

田中正則／ＳＲＡ　開発第１部

◆大規摸システムのドキュメント作成支援

数千ページにもなるドキュメントの作成支援を考えたときに、どのような環境が必要であろうか？

・ユーザ・インタフェースには、ユーザ・フレンドリな WYSIWYG 方式

・データ管理には、分散ファイルシステム十階層的なデータ管理

◆大型ホスト機とのネットワーク接続

WS と WS のネットワーク接続だけではなく、WS と大型機の接続についても議論すべきである。なぜ接続が難しいのか、接続したらどのようなことが出来るのかなど色々あると思う。

・メーカーのプロトコルの公開

・設計情報とソースコードのリンク、開発支援、テスト支援

北野義明／ KCS　ソフト開発部技術第一課

◆実行環境を意識した分散型開発環境

私のチームは、大型汎用機上での金融アプリケーションの開発をサポートしようとしているため、分散型の開発環境であっても、ホスト・マシンを無視できない。分散環境上の WS でホストしかないツールや機能についてどう扱

うのか。例えば、ホストのユーティリティをうまくシュミレーションするようなツールがWS上か、LAN上に必要ではないのか？等が課題であると思う。このような実行環境としてのホストを意識した開発が行なえる分散環境について課題を洗いだし、有効なアイデアを探りたい。

◆異機種WSの利用法

メインフレーマが分散型開発環境を意識したEWSを発売し、またΣの実用化と合わせ、今私のチームが行なっている「WSとホストとの環境の相互利用」や別のチームで「UNIX-LANの利用」等の自分達の取り組みがある。これら各々のWSをLANの上にどう乗せ、どう利用すればうまく分散され、かつ統合化もされた環境が出来るのか考えたい。

田中一夫／山一コンピュータ・
◆分散開発センター　開発第二部基本システム課

ワークステーションでの分散開発が可能になったといわれるが、実際問題コーディング・単体テスト等までは可能と思われるが、結合テスト以降は不可能だと聞いている。DB/DCをどうやってテストするか？この辺はやはりホストを使うしかないのかも知れない？教えてください。現在、YCCでは日立機に於いて2050のSEWBを使用し、画面設計・プログラム設計・プログラム作成等を実施しようとしている。また、ユニバック機に於いては2200という分散コンピュータを使用してプログラム作成～単体テストまでを実施しようとしている。ユニバックの世界ではこれぐらいしか手が出ない。

参考資料：日立機に於ける開発方式、ユニバック機に於ける分散開発

田中慎一郎／ＳＲＡソフト工学研究所
◆ワークステーション利用者の実態

少し前、４ケ月ほど作業をしていたところでは、作業場に１０台ほどのWSがありファイルサーバとLANで結ばれていた。私はその内の“１台”を作業用にわりあてられ、２人で共用していた。他もほぼ同様で、“１台”を２～３人で使っていたが、“使っていない”WSも５台ぐらいあった。私はそこでマシンをお借りしている立場だったので何も出来ませんでしたが、WSが余っているのに何が悲しくて、紙にコーディングしたり、リストを広げてデバッグをしなければいけないのか、とても疑問でした。ちなみにそこでは、WS間でファイルを受け渡す時、LANで結ばれているにも関わらず、FDでやっていました。他にもそのようなところはあるのでしょうか？

小池幸徳／日本システムサイエンス　システム部システム技術課
◆ DEENの提案

現在、私が開発中のDEEN（知的分散型システム開発環境）において、メインフレーム（巨大なファイルサーバー、ライブラリ管理）、32ビット機（マン・マシン・インターフェースを利用した開発機）、パソコン（安価な実行用計算機）といった位置づけをしており、このようにワークステーション＋ネットワークにより従来のメインフレーム中心の世界、あるいはパソコンによるネットワークの世界がどのように変化すべきかを考えたい。

青木　淳／富士ゼロックス情報システム　技術推進室
◆オブジェクト指向的な分散環境

ネットワーク（LAN）におけるノード（ワークステーション）がひとつの仮想マシンのように見えることが大切です。プロセスの概念がどのCPUかを意識しなくてもよく、それぞれのプロセスはオブジェクトとして自立し、その間をメッセージが送受信されて、仕事が進むような分散環境が必要です。

古田とおる／富士通ビーエスシー　商品企画開発部開発課
◆通信言語について

Sunのウィンドウシステム Newsにはサーバ・クライアント間の通信に Postscriptが使われていますが、データの小量化などは行なわれています。Postscriptでより軽いものも必要だと思います。

荻生準一／日本システム　府中事業所　第２システム技術部
◆Σワークステーションを利用した電力供給システムソフトの開発

当社では、ΣWSのモニタ・テストを計画しています。このテストでは、ΣWSを用いて電力供給システムのソフト開発がどこまで可能か？を目的としています。電力供給システムのソフト開発がどこまで可能か？また、何が問題か？を検証し、今後のWS時代に対処することを目的としています。個人の意見としては、プロセス制御のソフトではターゲット・マシンのリソースに大きく依存している為、WSでの開発では難しい点が多いと思います。

野中　哲／日本電気　マイクロ波衛生通信事業部

◆電子メールの普及について

電子メールは、便利であるが、多くの人がつながっていなくては、意味がない。最近、FAX の普及が著しいが、この程度の手軽さで、電子メールが使えるようになって欲しい。技術的というよりは、政治的な問題であると思う。

◆ ICE とのネットワーク化

ファームウェアの開発に ICE は欠かせないツールであるが、開発マシンとオンライン化する事例は、まだ一般的でない。高級言語のソースレベルのデバッグがターゲット装置上で出来ると有り難い。各社によってインターフェースが異なるのも困る。また、出荷後現地でのデバッグにも、電話を通じて、ソースコードにアクセス出来るような環境を作りたい。バイナリのバッチをあてるなどという非人間的な作業は、早くやめたい。

堀内泰輔／長野工業高等専門学校　機械工学科

◆教育用 WS はいかにあるべきか？

現在、ホスト 50 台のパソコン端末を使って、学生の実習に供している。パソコンを WS にグレードアップしたいのだが、まだまだ高すぎる。教育用 WS というのも必要ではなかろうか。そしてそれに具備すべき機能とは？

井上　尚司／ソフトバンク総合研究所

◆リモートログインとネットワークファイルシステムの関係

NFS は確かに良いと思う。しかし、本当に良い分散ファイルシステムは、リモートログインを必要としなくするものだと思う。我が社では、NFS すら行きわたっていないので、大きなことはいえないが、ソフトウェア開発において、ホスト名を意識しながら行なっているのは、「まだまだ子供だ」と考えて、構築ということをもって追及する必要があるのではないか。NFS でがんじがらめになると、最終的にどうなっているのかわからなくりそう。結局、スーパールート的使い方がよいのだろうか？

小林明彦／東電ソフトウェア

◆分散環境における各種データの最適管理方法

ワークステーションのように分散化された環境においても一元管理したいデータ、あるいは一台のマシンを利用したほうが有効である場合もある。結局分散化してもホスト的な役割を果たすマシンを要求する声があるならば、何もそのマシンはワークステーションである必要はないのではないか。ワークステーション＋ネットワークにホスト機あるいは DB マシンを負荷した形での構築が有効なのではないだろうか？

◆事務処理分野における分散環境のあり方

事務処理分野での大規模開発は、各社ともホスト依存型であると思われる。こうした中で、ターゲットマシンとは異なるメーカーのワークステーションを導入した際の、ワークステーションの役割分担と、同メーカーの端末と比較した場合の有無を参加者の方に聞いてみたい。

河田　亨／シャープ

◆分散環境とシステム全体の一元管理

分散環境の利用者がすべてシステム管理者能力を有していればよいが、育成するにしてもそう均整とれて分散するとは考えにくい。一方これを集中管理しようとすると、これまでの TSS システムと同様に行なえるとも言えない。どこまでをワークステーション利用者に任せ、全体の管理者が TSS 時代をこえる労力を要せず、マネージ出来るだろうか。

近藤康二／ソニー　スーパーマイクロ事業本部ワークステーション事業部

◆独り占めしたいという気持ちと分散環境

人間は、本来物を独り占めにしたい、又は自分のものにしたいという独占欲が強い動物であると思う。それと、分散環境というものは時として衝突をおこすことがある。例えば、データ分散の問題でも、特定ファイルを別マシンで管理する場合、誰が(どのマシンが)持つか取り合い(又は押しつけあい)になることがよくある。　また、CPU 資源についても同様で、自分の CPU(機械)を使われることを極端に嫌う人が多い。これらの人間の性質をかい潜っていかに分散環境を構築するか。まさに人間の心理との戦いのような気がする。

井川裕基／日本電子計算　開発本部ファイプロジェクト

◆一般の WS によるネットワークとホスト機の利用

現在、社内で UNIX (BSD) を中心としたネットワークを構築中です。その一部に大型機を組み込み、大型機のプログラム開発時において、従来の“メーカが提供した環境しか使えない”状態からの脱却を試みようとしています。柔軟性／レスポンスの点では絶対有利、のワークステーションですが、“既存大型機ユーザに使わせる”という点ではなかなか大変です。すでにそんな経験をお持ちの方はいらっしゃらないでしょうか？また、これからこういう使い方にはどのような技術的発展が予想できるのでしょう

か？（ホスト機と WS のの LAN 接続、NFS3...？？）

加藤　朗／東工大情報工学部

◆ワークステーション＋ LAN による分散環境の構築

適当な規模のネットワークなら、金さえ出せば誰にでも構築はできる。問題は、それをいかに使いこなしていくか、いかに管理していくか、ということにある。現在、ワークステーションを相当数活用しているような組織では、このあたりの問題は、導入時に活躍したエンジニアによって解決されているが、新たに参入する場合、これらの技術はお店では売っておらず、（某社のセミナーでとりあげれば別だけど）どうするんでしょうね。ところで、分散環境、即ち、計算機の名前を意識させない環境の構築は、あまり進んでいない。これはひとえに、分散環境の研究には、好きな時に自由に落とせ、カーネルを書き換えることができる、相当の能力のある計算機が複数と、それなりに鋭い研究者が沢山必要あるが、日本ではそのどちらも十分ではない。MACH をあてにしていていいのだろうか？

中村暢夫／日立ビジネス機器　技術部

◆ソフトウェア資源の配置について

分散環境の構築プランを検討したとき、ネックとなったのは、データの互換性、伝送速度、ユーザの操作性でした。その要因は、色々な機器を結合するということと、ソフトウェア資源をどのようには位置するかに拠っているという気がします。

3rd SEA Environment Workshop Position Paper

氏名　加藤　朗
種別　■会員 no. 850677　□一般
所属　東京工業大学　工学部　情報工学科　当麻研究室
住所　〒１５２　東京都　目黒区　大岡山　２～１２～１
TEL　03-726-1111 ext.2566
FAX　N.A.
EMAIL　kato@cs.titech.junet

仕　事

大規模ネットワークに於ける経路制御等に関する研究

作業環境

SUN3's with Ethernet + X.10/11 + wnn + NEmacs/kemacs

現在の問題

安価で効率的な経路制御手続きの開発及び評価
非定常リンクの取り扱いとアドレス割り振り問題
JUNET の管理水準を保ったまま、いかに手を抜くことができるか
どのようにして、短期間で効率的に学位を取得するか

広域ネットワークの問題点

加藤　朗
東京工業大学

January 6, 1988

Ethernet などの局所ネットワークの発展は我が国でも著しいが，それらの局所ネットワークの相互接続に関しては，今一つの感がある．アメリカではすでに１０年以上にも渡って，ARPANET などの経験を有しており，さまざまな問題点が洗い出され，解決されてきた．

ところで，広域ネットワークということばの定義であるが，ここでは次のように考えたい．

- 地理的な分散はあまり問題ではない．しかしながら，同一敷地内に終始するものは広域とは言えない．
- 広域ネットワークを構成する複数の局所ネットワークが，異なった管理母体によって管理運営されていること．この意味では，ノードプロセッサが BBN によって一元的に管理されている ARPANET は厳密な意味では広域ネットワークとは言えない．また，いくら距離が離れていても，学内ネットワークや社内ネットワークは広域ネットワークではない．

このような広域ネットワークを実現し，運営していくためには，勿論エライさんの了解や費用などの政治的・経済的問題を解決することは必須であるが，これらを無視したとしても，次の様な技術的問題を解決する必要があるだろう．

- さまざまな種類のリンク（通信媒体）が用いられており，それらの速度も異なっていることが多い．そのため，複雑な経路制御技術や輻輳制御技術が必要になる．
- 管理母体が異なることにより，部分ネットワークの管理レベルが異なる．一部の管理不良ネットワークによって，全体に悪影響が及ぶような脆弱さは避けなければならない．
- アプリケーションのレベルでの，セキュリティの確保も重要であるが，経路制御レベルでの幇助も必要である．

（広域）ネットワークの最大の目的は，計算機によるデータ（メッセージ）交換（やその管理）を通じて，組織間に存在するポテンシャルギャップの低減である．UUCP を利用した広域ネットワークである JUNET も，この目的をかなり達したと考えられるが，まだ十分ではない．また，すでに UUCP のみによるネットワーク構築は限界に近づいており，より快適で手間の掛からない広域ネットワークへ発展させていく努力が必要である．

現段階で JUNET を ARPANET 並み，もしくはそれ以上の帯域幅を持ったリンクを全面的に使用して，UUCP を置き替えるのは不可能である．しかしながら，UUCP の様なただ単にファイル転送をバッチ的に行うリンクを，TCP/IP の様な上位プロトコルに依存しないプロトコルをサポートするリンクに置き換え，信頼性の高いデータ交換を実現するようにすることは急務である．このため，新たにリンクを引き直さなくても TCP/IP による通信を可能にする Dialup IP（DL）が開発されており，最近普及してきたダイアルアップ回線用の高速モデムを使用して，FTP を用いて約 0.8KB/sec の信頼性あるファイル転送を行うことができるようになる．

これらのことを踏まえ，『実践的環境』としての広域ネットワーク JUNET をより信頼性のある快適なネットワークに成長させていくためのプランについて議論する．

3rd SEA Environment Workshop Position Paper

氏名	酒匂 寛
種別	☑会員 no 850202 □一般
所属	(株) ソフトウェア リサーチ アソシエイツ 環境開発部
住所	〒(102) 千代田区平河町 1-1-1
TEL	(03) 234-2611 (ext 571) FAX (03) 262-9719

仕　事

◎ ソフトウェア環境の整備

- 開発環境
- 利用環境

上記の目標のため 現在 2つのプロジェクトに関わっている.

1: SDA (Software Designer's Associates) → ソフトウェア設計支援システムの基本アーキテクチャを研究するプロジェクト

2.: Zodiac → 小規模事務アプリケーション開発支援環境. 現在拡張中

作業環境

SONY NEWS-830 (4.2BSD) 8M(mem) / 182M(disk)

(日本語環境有)

現在の問題

ツールを自由に統合し, データを管理するための枠組が存在しない.
分散環境で, それを行えなければならない.

次ページの討論テーマ票にも, 必要事項をご記入の上本紙とあわせて 12 月 15 日まで (締め切り厳守!) に下記の SEA 事務局までお送りください. もちろん, 本式の Position Paper も同封していただくほうがベターです (既発表の論文を添付していただいても構いません).

ソフトウェア技術者協会 (SEA) 〒102 千代田区単町2-12 藤和半蔵門コーブビル505
TEL: 03(234)9455 FAX: 03(234)9454

ツール統合と環境改善について

酒匂 寛

Software Research Associates, inc.
環境開発部
e-mail : sakoh@sra.junet

ABSTRACT

ワークステーションの急激な値下がりにより、その普及台数は飛躍的に伸びている。筆者の職場でも、数多くのワークステーションが稼働しているが、その使われ方は、残念ながら単にＴＳＳの端末が複数画面上に表示されているだけ、という場合が多い。これは非常に残念なことである。そこで筆者は、**X-Window** 上で動作する、簡易ユーザーインターフェイス構築ツール **XME** を作成した。利用者は既存のツールを **XME** を用いて統合し、使用の便に供する事ができる。このポジションペーパーでは、**XME** を用いたツール統合例を示し、その効果に関しての考察を行った後、将来のツール統合構想について述べる。

1. 夜明け前[1]

ワークステーションが比較的安価に手に入るようになって、ビットマップディスプレイも昔ほど珍しくなくなって来た。その大部分は **UNIX + Window System** という構成である。この変化は作業環境をどの程度改善したのだろうか。残念ながら現時点では、ある特定のツール内ではマウス、マルチウインドウ、マルチフォントなどの恩恵に浴せるものの、既存の、あるいはユーザーが用意した様々なツールを統合する段になると途端に旧態然とした作業環境になってしまう。即ち、キャラクタディスプレイが沢山手に入っただけ[2]、という結果にしかならないのである。**UNIX** の特徴を表す言葉にツールボックスというものがある。これは、既存のツールを組み合わせて新しいツールを構築し易いといった意味合いであるが、この特徴はシェルと呼ばれるコマンドプロセッサの助力に負う所が多い。特にシェルスクリプトと呼ばれるツール統合手段は、かなり複雑な処理でも既存のツールを組み合わせて実行させることができる。ところが残念なことに、シェルはそれ自身では非常に貧弱な入出力機構しか持っていない。印字端末で使われていた頃のイメージそのままに、文字列の入出力を行なえるだけである。マウスからの入力や、ビットマップディスプレイの操作等は標準的なものが無かったせいもあり、それぞれのツールが独自に対応しているのが現状である。

ここに至り、筆者はビットマップ環境でシェルと協力して動作しツールの統合を支援するシステムの構築実験を行う事にした。実験を開始するのに際して、

- 簡単にインターフェイス形状を定義、更新できること
- シェルプログラムと組み合わせて使用できること
- プロセス間通信による分散処理が可能であること
- 当面 **X-Window** を対象とするが、将来それ以外のウインドウシステムにも適用可能であること

などを当座の目標とした。この実験過程で最初に作成されたのが **XME** である。

2. XME とは

先に述べた通り、**XME** とは、**X-Window** 環境下で動作する簡易ユーザーインターフェイス構築ツールである、簡単な定義ファイルを記述することにより、パネルをデザインすることが可能であり、簡単な手続きや、プロセスの呼び出しを実行させることができる。

---

[1] この有名な小説のタイトルは、明治維新以前を暗黒時代と考えたところから来ているが、今にして思えば皮肉なタイトルである。私達は本当に明るい時代に生きているのだろうか？

[2] それだけでもとりあえず十分、という方も居られると思うが。

図１：アプリケーションパネルの例

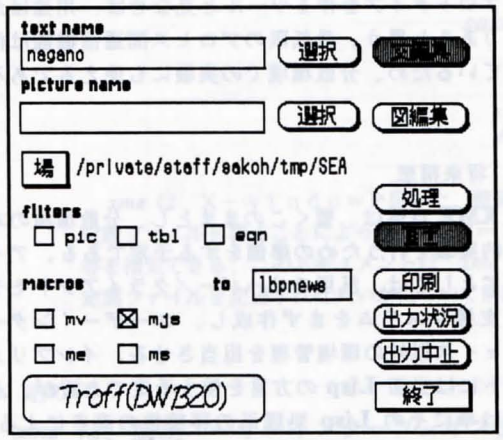
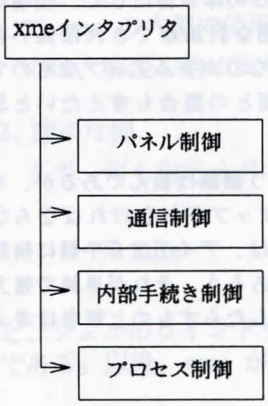


図１は、**XME** で作成されたアプリケーションの一例である。このアプリケーションは、**UNIX** のテキストフォーマッタである **troff** を使用する場合に必要な手続きを実行する機能を持っている。[3] 詳細は後述するが、このパネルから troff, vi, make, pic, tbl, eqn, lpr, xp, xpic, etc .. などの様々なツールが呼び出されるのである。呼び出しはツールを意識せず、作業目的に適った場所(編集、印刷など)をマウスで選択すれば良いようになっている。

ここで、**XME** のアーキテクチャを図２に示そう。

図２：**xme** のアーキテクチャ



パネル制御

基本的なパネル動作を行う部分である。ユーザーの入力(キイボード、マウス)を処理しインタプリタに結果を返す。

[3] 当然このポジションペーパーもこのアプリケーションを使って書かれている

通信制御

プロセス間通信を支援する。複数の **XME** 同士でメッセージが交換出来る。

内部手続き制御

パネル上の項目に関係付けられた手続きを実行する(簡単な演算、代入、条件判断など)。

プロセス制御

他のツールを呼び出し、その実行を制御する。

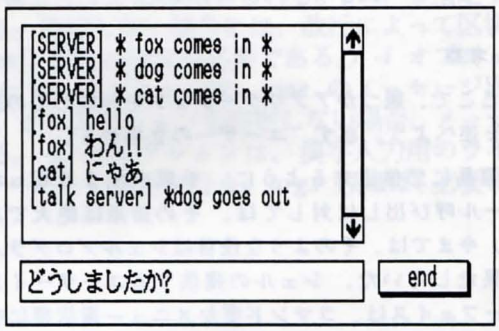
3. ツール統合例

3.1. docs

先に例を示した **troff** 環境を支援するアプリケーションである。実行画面は、図１に示されている。

3.2. xmetalk

複数の **XME** を使い、二人以上の人間が画面上で対話するためのツールである。プロセス間通信機能を使用している。下の画面では、**fox, dog, cat,** の三人が会話を始め、**dog** が去った様子が示されている。[4]

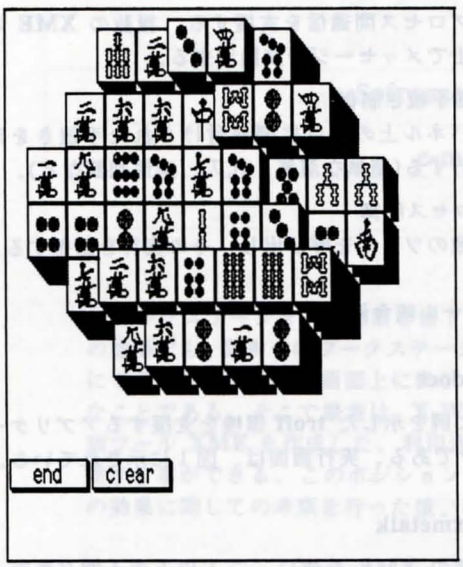


3.3. shanghai

あの『上海』[5]の **XME** 版である。少し凝ればこのようなものも作れるという例である。

[4] **fox** がわん!!と言ったため、**dog** は出鼻をくじかれて去ってしまったようだ

[5]『あの』、といっても御存知ない向きにはさっぱりであろうが、元はマッキントッシュの上に作られたゲームである。



3.4. その他

他に現在存在しているアプリケーションとしては、ファイルのバージョン管理、ネットワークを通したリモートホストへのアクセス管理、ノートファイルの管理、伝言管理、スケジュール管理、電卓、ゲーム、漢字記号入力、などがある。

4. 考察

ここで、幾つかアプリケーションを作成しての感想を述べよう。まず、ユーザーの立場から。

容易に想像出来るように、手順の決まり切ったツール呼び出しに対しては、その効果は絶大である。今までは、そのような役目はシェルプログラムが果たしていた。シェルの提供するユーザーインターフェイスは、コマンド型かメニュー選択型になる場合が多いが、**XME** は、状態を目で確認し易くかつ選べる対象が表示されているため、初心者にも使い易いインターフェイスを提供出来る。一般にエキスパートはコマンド型を好むと言うが、実行時に変更するオプションが多いツールを相手にする場合には、このようなインターフェイスも役立つと思う。筆者は、もはや先に紹介した **docs** の助けなしには、文書作成のルーチンワークは苦痛である。パネルのレイアウトもその気があれば簡単に変更することができる[6]ので、ユーザー個人の趣味もある程度反映させることができる。

アプリケーション作成の側からいえば、気軽にビットマップディスプレイを使ったアプリケーションを作れることは魅力である。しかしながら、あくまでも簡便さを追及したものであるから、製品レベルのものをこれで作るには無理がある。ある程度動くプロトタイプを作るツールと見なせば、用途はかなりあると思う。最低限のプロセス間通信機能は備えているため、分散環境での実験にも使えるであろう。

[6] パネルデザインのための専用ツール **XMED** が用意されている。

5. 将来構想

XME 自体は、暫くこのままとし、分散環境の本格的実験を行うための準備をする予定である。アイデアとしては、汎用のサーバー／クライアントモデル支援システムをまず作成し、ユーザーインターフェイス以外の環境管理を担当させる。インプリメントには当面 **Lisp** の方言を使う予定であるが、それは単にその **Lisp** 処理系の移植性の高さによる。管理システムができた時点で、**XME** に対して出されている改良要求を吸収しユーザーインターフェイス部分を担当させる予定である（この場合 **XME** はクライアントの一つとなる）。

両者が揃ったとき、『分散開発環境のありかたとツール統合に関する実験』の下準備ができたことになるであろう。最初の実験としては、筆者がかつて **UNIX** の上に構築した **COBOL** 開発支援環境 **Zodiac** の再構成を行う予定である。

6. ワークショップに向けて

と、上では偉そうなことを書いているが、問題点はまだ山積みしている、今回のワークショップでは **XME** のアプローチを一つの叩き台にして、環境構築の方法に関する広範囲な討論ができれば良いと思っている。この話はかなりボトムアップなものであるから、理論的な側面との整合も考えたいと思う。

開発環境とは何かという議論は盛んであるが、モデルはいつかは現実にマップされなければならない。この橋渡しをするには、アイデアを手軽に検証出来るシステムが必要であろう。それが単純で強力であるほど、良い結果をもたらすものと筆者は考えている。

7. 謝辞

XME のパネル管理の部分は、その大部分をＳＲＡの同僚である有座道春氏の作成した **XControl** ライブラリに助けられている。この場を借りて感謝の意を表したい。また、バグの多かった時代から **XME** におつき合い願い、貴重な意見を戴いた大勢の方々にも感謝を捧げたい。

Xme User's Guide (ver1.0)

酒匂 寛
sakoh@sra.junet

ABSTRACT

xme は、Ｘ－ｗｉｎｄｏｗを用いた、簡易ユーザーインターフェイス構築ツールである。利用者は、定義ファイルを書くことによって、インターフェイス･パネルの形状、そこから起動される簡単な手続き等を指定できる。このドキュメントは、*xme* がどのようなものであり、かつその利用者が、どのように定義ファイルを記述すれば良いかについて解説したものである。

1. 始めに

xme は、簡易ユーザーインターフェイス構築ツールである。起動するとスクリーン上にウインドウが現われ、ユーザーとの対話を行ない、様々な機能を果たす。利用者はウインドウのレイアウトやその振る舞いを、簡単な定義ファイルを用意することによって変更することができる。

詳しい解説は、後ほど行なうとして、先ず ***xme*** の起動方法を述べよう。次のように行う。

```
% xme [ -f file ] [ -o[ STR ] ] [ -i ] [ -w ] [ -K ] [ host:num ] [ args .....  ]
```

file は、***xme*** が解釈実行するための定義ファイルである。省略するとディフォルトの定義で立ち上がって来る。***-o*** オプションは、実行結果を標準出力に書き出すために用いられる。***STR*** を指定すると、各出力項目は ***STR*** によって区切られて出力される。指定しない場合には、改行によって区切られる。(実際に書き出すためには、後述の **pl (put at last)** 属性の指定が必要である。) ***-i*** オプションは、最初にアイコンを表示するための指定である。また ***-w*** オプションは、***xme*** のメッセージ出力を行なうための専用ウインドウを開くことを指定する。このオプションを指定しない場合、メッセージ出力は ***xme*** を起動したシェルウインドウに現われる。***-K*** のオプションは、漢字入力用のウインドウを生成することを指定する。***host:num*** で、出力サーバーの指定を行う。***args*** 以降は、必要に応じて ***xme*** の内部で使用されるものである。

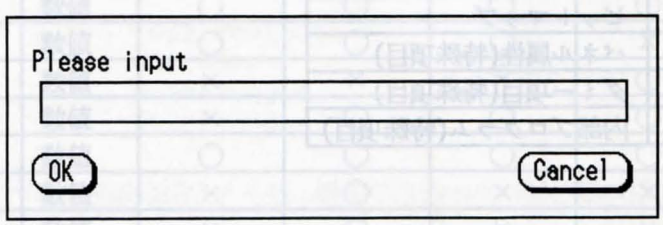
では、解説を始めよう。

2. 簡単な例

まず、何も指定せずに ***xme*** を起動してみよう。

```
% xme
```

と、シェルのウインドウ内で入力してみる。すると、画面中央に下のようなウインドウが現われる筈である。以後、***xme*** が使用するウインドウをパネルと呼ぶことにしよう。



キイボードから入力された文字は、パネル上のテキスト編集枠の中に反映される。編集枠内では、マウスオペレーションとして、以下の操作が可能である。

操作	効果
左マウスボタンをクリック	テキスト挿入ポイントの変更
左マウスボタンを押しながらマウスを移動	テキスト選択
右マウスボタンを押してメニュー選択	選択されたテキストに対する操作

OK または **CANCEL** のボタンをマウスでクリックするか、リターンキイを入力するとプログラムは終了する。

さて操作ばかりを述べたが、上のプログラムで何ができるのだろうか。ディフォルトで、***xme*** が外部に与えられる影響は2種類ある。ひとつはプログラムのｅｘｉｔステータスで、もうひとつはカットバッファの内容である。前者に関して言えば、**OK** がクリックされたか、リターンキイが叩かれたとき0が返され、**CANCEL** がクリックされたとき1が返されるのである。即ち、

```
% xme || echo FALSE
```

として、ツールを起動した場合、**CANCEL** ボタンをクリックした場合にのみ、**FALSE** がエコーバックされるのである。後者に関しては、Ｘ－ｗｉｎｄｏｗ上で実行中の、他のアプリケーションとの間で、テキストのカットアンドペーストが出来るということである。

3. 定義ファイルの利用

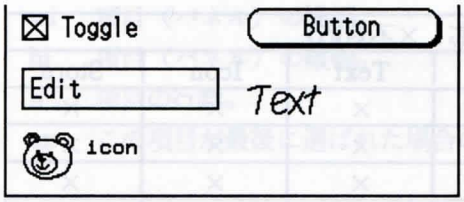
ディフォルトで ***xme*** を利用している場合、その用途は非常に簡単なものに限られてしまう。そこで登場するのが、定義ファイルによるパネルとその振る舞いの定義である。定義ファイルを利用する場合、***xme*** の起動方法は次の形になる。

```
% xme -f "定義ファイル名"
```

定義ファイルには、パネル自身とパネル上の項目に関する属性が記述される。定義ファイル名に **-** を指定した場合には、定義は標準入力から読み込まれる。現在許されている項目は、次のものである。

項目名	説明
Button	クリックボタン
Toggle	トグルボタン
Edit	テキスト編集枠
Text	表示のみのテキスト
Icon	ビットマップ
Self	パネル属性(特殊項目)
Store	ダミー項目(特殊項目)
Program	内部プログラム(特殊項目)

下に各項目の表示例を示す。



ちなみに、このパネルを表示するための定義は：

```
Self|my dialog window:¥
    :hi#120:wi#280:
Toggle|toggle button:¥
    :xp#10:yp#2:wi#120:hi#25:tx=Toggle:on:tp=CheckBox:
Button|button:¥
    :xp#150:yp#2:wi#120:hi#25:tx=Button:
Edit|edit box:¥
    :xp#10:yp#40:wi#120:li#1:tx=Edit:
Text|plain text:¥
    :xp#150:yp#40:wi#120:li#1:tx=Text:fn=macLosAngels24:
Icon|icon:¥
    :xp#10:yp#80:bm=skuma.ptn:
Text:¥
    :xp#50:yp#80:wi#120:hi#25:tx=icon:fn=vtsingle:
```

といったものになる。

さて、定義ファイルの内容を見て行こう。このファイルは

項目名：属性定義：属性定義：属性定義…：

という形式の行が、複数繰り返し現われるという形をしている。（行の最後に\があると、継続していると見なされる）各行がそれぞれ、パネル上の項目の一つ一つに対応している。なお、特別な項目名として **Self** と **Store** , **Program** というものがある（上の例で、Self は最初に書かれている）。**Self** はパネル自身の属性を記述するために用いられ、**Store** は、パネル上に現われない疑似項目の記述に使われる（使い方は後述）。そして、**Program** は、パネル内の簡単な動作記述を行なうために用いられるものである。

以下に、項目の属性一覧を示し、その解説を行う。ただし、**Program** に関しては先の別章で扱う。

属性	区分	項目への対応（○対応：×不対応）						
		Self	Button	Toggle	Edit	Text	Icon	Store
xp	数値	○	○	○	○	○	○	×
yp	数値	○	○	○	○	○	○	×
wi	数値	○	○	○	○	○	○	×
hi	数値	○	○	○	×	×	○	×
li	数値	×	×	×	○	○	×	×
ex	数値	×	○	○	○	×	×	×
bw	数値	○	○	○	○	○	×	×
dm	数値	×	○	×	×	×	×	×
tk	数値	×	○	×	×	×	×	×
mc	数値	×	×	×	○	×	×	×

属性	区分	項目への対応（○対応：×不対応）						
		Self	Button	Toggle	Edit	Text	Icon	Store
sw	文字列	×	○	×	×	×	×	×
ic	文字列	○	×	×	×	×	×	×
im	文字列	○	×	×	×	×	×	×
ux	文字列	×	○	○	○	×	×	○
fn	文字列	×	○	○	○	○	×	×
tx	文字列	×	○	○	○	○	×	○
tv	文字列	×	○	○	×	×	×	×
fv	文字列	×	○	○	×	×	×	×
tp	文字列	×	×	○	×	×	×	×
bm	文字列	×	×	×	×	×	○	×
ct	文字列	×	○	○	○	×	×	×
sh	文字列	○	×	×	×	×	×	×
pg	文字列	×	○	○	○	×	×	×
wp	文字列	×	○	○	○	×	×	×
id	文字列	×	○	○	○	×	×	○
fs	文字列	×	×	×	○	×	×	×
bs	文字列	×	×	×	○	×	×	×
nf	文字列	×	×	×	○	×	×	×
nc	フラグ	○	×	×	×	×	×	×
et	フラグ	×	○	○	○	×	×	×
on	フラグ	×	×	○	×	×	×	×
wt	フラグ	×	○	○	○	×	×	×
pe	フラグ	×	○	○	○	×	×	×
wt	フラグ	×	○	○	○	×	×	○
pl	フラグ	×	○	○	○	○	○	○
dv	フラグ	×	×	○	×	×	×	×
sb	フラグ	×	×	×	○	×	×	×
lt	フラグ	×	×	×	○	×	×	×
as	フラグ	×	×	×	○	×	×	×

上の表で、区分というのはデータの与え方を示している。即ち、ｔｅｒｍｃａｐのエントリイと同様に、

数値　　:wi#120:
文字列　:ic=/usr/lib/icons/appare.icn:
フラグ　:wt:

といった与え方を、*xme* の定義ファイル内でも行うのである。

さて、各属性について説明しよう。

xp　項目（パネル）の左肩のＸ座標。項目の場合はパネル内での座標、パネルの場合はスクリーン上での座標。

yp　項目（パネル）の左肩のＹ座標。項目の場合はパネル内での座標、パネルの場合はスクリーン上での座標。

wi　項目（パネル）の横幅。

hi　項目（パネル）の縦幅。

li　項目の行数。

ex　この項目が最後に選ばれた場合のｅｘｉｔステータスを指定する。

```
:ex#1:
```

と書くと、この項目が選ばれて終了する場合に、ｅｘｉｔステータスとして１が返される。

bw　境界の太さを指定

dm　ボタンの丸みを指定する

tk　ボタンの厚みを指定する

mc　**Edit** 項目に入力可能な文字数（バイト数）を指定する。

sw　ボタンの影の種類を指定する、指定できるものは現在

```
NoShadow
Forenoon     ForenoonFloat
Highnoon     HighnoonFloat
Afternoon    AfternoonFloat
CandleLight  CandleLightFloat
```

といったものである。

ic　*xme* のアイコン指定。ウインドウマネージャを使用している場合、アイコンとして表示させたいビットマップファイルを指定する。ビットマップファイルの形式は、***bitmap(1)*** で扱われるものと同じである。

im　*xme* のアイコンマスク指定。上で指定されたアイコンと同じ大きさでなければならない。

ux　この項目が選ばれた場合に呼び出されるコマンド。例えば：

```
:ux=xterm -fn vtsingle:
```

と書いておくと、この項目が選ばれた場合に、***xterm(1)*** が起動される。

fn　この項目が使用するフォント。

tx　この項目が使用するタイトル（初期文字列）。即ち、**Edit** では編集対象の初期文字列。**Button, Toggle** ではその名前。**Text** では表示される文字列を与える。

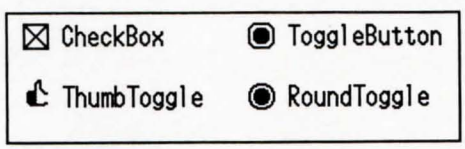
tv　**TRUE** の場合に返す値。

fv　**FALSE** の場合に返す値。

tp　トグルボタンのタイプ。現在指定できるのは：

```
ToggleButton
CheckBox
RoundToggle
ThumbToggle
```

の４種類である。下にそれぞれの形状を示す。（すべて **TRUE** の状態である）



bm　パネル上に表示するアイコン。

ct　この項目が選ばれた場合カットバッファに格納する文字列を指定する。

sh　*ux* で指定したコマンドは、通常"/bin/sh"によって実行されるが、このオプションはそれを変更する。例えば:

```
Self:sh=/usr/local/bin/jsh:
```

のような指定をすれば、日本語シェルが立ち上がるのでコマンドの引き数に漢字を渡すことが可能になる。

pg　この項目が選ばれた場合呼び出す手続き（**Program** 項目）を指定する。(**Program** 項目の書き方は後述)

wp　ここで指定されるものは、**wt** で待たれているプロセスが終了した場合に呼び出されるプログラム項目名である。属性 **wt**（後述）を持つ項目に限り意味がある。

id　項目に名前を付ける。ここで指定された名前は、**ux** 、あるいは **Program** 項目実行時に評価される。詳細は後述される。

fs　ここで指定した文字が **Edit** 項目内で入力されると、次の **Edit** 項目に、キャレットが移動する。文字は複数指定して良い。例えば:

```
Edit:fs=¥012¥015:
```

という指定を行なうと、改行（^J）もしくは復改（^M）が入力された場合に、次の **Edit** 項目にキャレットが移動する。

bs　**fs** 項目の逆で、入力されると前の **Edit** 項目に、キャレットが移動する。

nf　**fs** で指定された文字が入力されて、項目の移動が行なわれるとき、移動する先の項目は、通常定義ファイルに書かれた次の **Edit** 項目である。**nf** の指定はこれに対し、移動先の項目を明示的に指定するもので、**id** で指定する項目名を書くことができる。

nc　改行（リターン）が叩かれると、ディフォルトでは項目番号（後述）0が選択されたと解釈される。このフラグは、そのような効果を打ち消すためのものである。

et　この項目が選ばれた場合終了することを示す。

on　トグルボタンの初期値を **TRUE** にする。

wt　呼び出されたコマンドの終了まで項目を選択不可にする。例えば:

```
:ux=xterm -fn vtsingle:wt:
```

と指定しておくと、ｘｔｅｒｍが起動された後その実行が終了するまで項目の表示がアミカケになり、選択ができなくなる。

pe　項目が選択される度に、現在値を出力することの指定。

um　**ux** で指定されたコマンド実行時に、*xme* のウインドウを消去するか否かの指定。

pl　*xme* の起動時に *-o* オプションが指定されていた場合、その現在値を *xme* 終了時に標準出力に書き出すことを指定する。

dv　トグルのテキストは、通常 **tx** で指定したテキストが表示されているが、**dv** を指定することによって、**TRUE , FALSE** に対応したテキストが表示されるようになる。

sb　**Edit** 項目にスクロールバーを付ける。

lt　**Edit** 項目を、リスト選択形式にする。即ち、ボタンのクリックによって、一行が選ばれ、選ばれた範囲が値となる。ドラッグすれば複数行を選ぶこともできる。この項目はキイボード入力を受け付けない。

as　**mc** で項目の最大長が定められている場合、指定された文字数の入力が行なわれると同時に、次の項目に移動することを指定するフラグである。

4. その他の機能

4.1. 引数の取り込み

xme を起動する際に、最終的なレイアウト等を行いたい場合がある。この場合、定義ファイルの中に＄ｎ（ｎは整数）と書くことにより、コマンドラインの引数を取り込むことができる。ｎは０から始まる整数である。コマンドライン上で、オプション以外の引数が現われると、それ以降を＄０、＄１、＄２、、、として定義ファイル中で参照できるのである。

```
% xme -f desc -o arg0 arg1 arg2 ...
                $0   $1   $2 ...
```

例えば、テキスト編集枠に最初に表示する文字列を外部から与えたい場合には、定義ファイル中に：

```
Edit:¥
    :xp#10:yp#10:wi#100:hi#25:tx=$0:
```

と書き、

```
% xme -f xxxx InitText
```

と呼び出すことにより、編集枠内に **InitText** の文字列を表示させることができるのである。

4.2. 環境変数の取り込み

記述ファイル内に **${id}** のような記述があると、それは環境変数の取り込みを意味する。例えば以下の記述は、編集枠内にユーザーのホームディレクトリを初期設定する。

```
Edit:¥
    :xp#10:yp#10:wi#100:hi#25:tx=${HOME}:
```

4.3. コマンド呼び出し時の文字列置換

ux= の指定によってコマンドを呼び出す場合に、その引数を他の項目の状態から決めることが可能である。各項目は定義ファイル内の **id** 属性で指定された項目ＩＤを持っている。(**Self, Program** には項目番号はない）定義ファイル中に！｛項目ＩＤ｝と書くことにより、その表記は項目の現在値に置換される。[1] 項目の現在値は項目の種類と状態によって決まる。下にその一覧を示す。

---

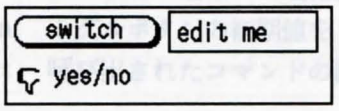
[1] 注意：古い(Oct 6 1987 以前) **xme** のバージョンでは、!n（ｎは整数）という参照を認めていたが、今後この機能は保証できなくなる。現在(Oct 7 1987) **Edit** に **sb**（スクロールバー）を用いなければ上位互換性は保たれるようになっている。

項目	現在値の評価
Button	評価直前に選択された項目自身のとき1、通常は0。tv, fv が定義されている場合には、それらが代わりに用いられる。
Toggle	TRUE のとき1、FALSE のとき0。tv, fv が定義されている場合には、上と同じ。
Edit	現在書かれている文字列。ただし lt 属性が指定されている場合は、反転した部分。
Text	現在書かれている文字列
Store	現在保持している文字列
それ以外	未定義

これも例を使って説明しよう。次のような定義ファイルがあるとする。

```
Self:¥
    :xp#10:yp#10:wi#200:hi#60:
Button:¥
    :xp#5:yp#5:wi#90:hi#20:tx=switch:¥
    :ux=echo !{switch} !{edit} !{yes/no}:id=switch:pl:
Edit:¥
    :xp#100:yp#2:wi#90:li#1:tx=edit me:id=edit:pl:
Toggle:¥
    :xp#5:yp#30:wi#100:hi#25:tx=yes/no:tv=yes:id=yes/no:pl:
```

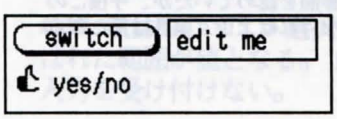
Button, Edit, Toggle には、それぞれ **switch, edit, yes/no** の項目ＩＤが割り当てられている。この定義で作られるパネルは、下図のようなものである。



ここで、**switch** というボタンの上をクリックしてみよう。***xme*** を起動した画面に、次のような出力が現われる。

```
% xme -f desc        ☞ xme の起動（desc は定義ファイル）
1 edit me 0      ☞ switch をクリックした
```

即ち、"echo 1 edit me 0"というコマンドが呼ばれたのである。最初の1は!{switch}に対応するもので、これは自分自身（switch）であるからいつでも1である。次の"edit me"は !{edit}に対応するもので、テキスト編集枠内を変えるとこの出力も変化する。最後の0は !{yes/no}に対応している **yes/no** トグルスイッチの値で、現在 **FALSE** であるから0が返されたのである。それでは、**yes/no** のトグルスイッチをクリックしてみよう、トグルは **TRUE** になり、画面は下のようになる。



ここで、先のように **switch** をクリックすると、出力は:

```
1 edit me yes          ☞ switch をクリックした
```

となる、最後の"yes"は **Toggle** に **tv** が指定されていたために、１の代わりに返されたのである。

4.4. 標準出力への結果の出力

前章で簡単に触れたが項目属性に **pe** を与えた項目は、その項目が選択されるたびに（クリックされたり、キイ入力される度に）その現在値を標準出力に書き出す。この指定方法以外に、***xme*** の起動時のオプションとして ***-o*** を与えることができ、この場合属性 **pl** を持つ全項目の現在値が、標準出力に指定されたデリミタ（省略時は改行）を用いて出力される。前節の例で、

```
% xme -f desc -o
```

として起動し、何もせずに終了した場合

```
1
edit me
0
```

といった出力が得られる。もし、

```
% xme -f desc -o"/"
```

として起動した場合には、出力は次のようになる。

```
1/edit me/0/
```

現在は意味がないが、**Icon** の現在値としては空行が出力される。

5. Program 項目の使用

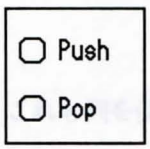
さて、より凝った使い方を望む利用者のために、**Program** 項目の書き方を説明する場所に来た。項目は皆、以下の形式をしている。

```
Program|プログラム名: 機能 引数 引数 ...:[ 機能 引数 引数 ...: ....: ]
```

即ち、あるプログラム名に対して、いくつかの機能(+引数)の組が対応している。各機能は順番に実行される。例を示そう。例えば２つの **Toggle** があり、どちらか一方だけしか **TRUE** になれないものとしよう。記述ファイルは例えば次のようになる。

```
Self:xp#100:yp#100:wi#85:hi#85:
Toggle:xp#8:yp#10:wi#80:hi#33:¥
    :tx=Push:¥
    :tp=ToggleButton:id=push:pg=push:
Toggle:xp#8:yp#47:wi#80:hi#33:¥
    :tx=Pop:¥
    :tp=ToggleButton:id=pop:pg=pop:
Program|push:off pop :
Program|pop :off push:
```

下に表示されるパネルを示す。



ここで定義されている **Program** 項目は2つあり、それぞれ **push pop** というプログラム名を持っている。いま **id=push** の属性を持つ項目がクリックされたものとしよう。**pg=push** という記述がその直後にあるため、プログラム項目 **push** が呼び出される。さて、**Program|push** に指定されている機能は **off pop** である。これは、" **pop** の項目ＩＤを持つ項目を **off** にする" という意味である。即ち、**push** のトグルがクリックされると、**pop** のトグルは **FALSE** になるのである。現在次の機能が提供されている。

機能	記法	説明
on	on id1 id2 ..	id で指定されるトグルスイッチを TRUE にする
off	off id1 id2 ..	id で指定されるトグルスイッチを FALSE にする
up	up id1 id2 ..	id で指定される Edit 項目をスクロールアップする
down	down id1 id2 ..	id で指定される Edit 項目をスクロールダウンする
pup	pup id1 id2 ..	id で指定される Edit 項目をページ単位でスクロールアップする
pdown	pdown id1 id2 ..	id で指定される Edit 項目をページ単位でスクロールダウンする
＝＝＝	= id value	id で指定される項目に値を代入する。単純に文字列を書いた場合、その値が入るが "< ファイル"でそのファイルの内容、"`コマンド`" でコマンドの出力が代入される。id で指定された項目がトグルスイッチの場合、設定する値が 1 あるいは tv と等しいとき TRUE、それ以外のとき FALSE になる。
enable	enable id1 id2 ..	id で指定される項目を選択可能(通常の状態)にする
disable	disable id1 id2 ..	id で指定される項目を選択不可能(アミカケ)にする
kill	kill [-n] id1 id2 ..	id で指定される項目から ux で呼び出されているプロセスに n (n は整数) で指定されるシグナルを送る。-n が指定されないときは、SIGINT が送られる。
exec	exec id1 [id2 id3 ..]	id1 で指定される項目の"ux=項目"を実行する。id2, id3...などの項目が指定されている場合には、実行結果がそれぞれの項目に代入される。

機能	記法	説明
cd	cd dir	dir で指定されるディレクトリにカレントディレクトリを変更する。
selfile	selfile id1 [id2 ..]	検索ウインドウが表示され、ファイル名を選ばせる。選ばれたファイル名は id1, id2 ..の各項目に代入される。
cdselfile	cdselfile id1 [id2 ..]	検索ウインドウが表示され、ファイル名を選ばせる。選ばれたファイル名は id1, id2 ..の各項目に代入される。同時に、選んだファイルが存在したディレクトリに、カレントディレクトリを変更する。
seldir	seldir id1 [id2 ..]	検索ウインドウが表示され、ディレクトリ名を選ばせる。選ばれたディレクトリ名は id1, id2 ..の各項目に代入される。
cdseldir	cdseldir id1 [id2 ..]	検索ウインドウが表示され、ディレクトリ名を選ばせる。選ばれたディレクトリ名は id1, id2 ..の各項目に代入される。同時に、選んだディレクトリに、カレントディレクトリを変更する。
selpath	selpath id1 [id2 ..]	検索ウインドウが表示され、パス名を選ばせる。選ばれたパス名は id1, id2 ..の各項目に代入される。
cdselpath	cdselpath id1 [id2 ..]	検索ウインドウが表示され、パス名を選ばせる。選ばれたパス名は id1, id2 ..の各項目に代入される。同時に、選んだパスに、カレントディレクトリを変更する。
kanji	kanji [on | off]	漢字入力ウインドウを立ち上げる(on のとき)、あるいは消去する(off のとき)。
setfont	setfont font id1 [id2 ..]	id で指定される項目のフォントを font に変更する。
ptyout	ptyout arg1 [arg2 ..]	arg1, [arg2 ..]をコントロールターミナル(/dev/tty)に出力する。この出力はキイボード入力と共存することになる。
callprog	callprog arg	arg で表される名前の項目を Program 項目として実行する。
ifmatch	ifmatch arg1 arg2 [prog1 [prog2]]	arg1 と arg2 の値を比較して、等しければ prog1 をもしそうでなければ prog2 を Program 項目として実行する。 prog 項目が省略されたときは、"Default"Program を呼び出す。
push	push arg	arg で表される値をスタックにプッシュする
pop	pop id	スタックから値をポップして id で示される項目に代入する
peek	peek id	スタックの先頭値を id で示される項目に代入する
eval	eval id	スタックを評価して結果を id で示される項目に代入する[2]。演算子の種類は後述。
clear	clear	スタックをクリアする
unmap	unmap id	id で指定される項目をアンマップする。[3]
map	map id	id で指定される項目をマップする。
exit	exit [arg]	arg で指定される値を Exit Status(省略値は0)として、xme を終了する。

[3] スタックの評価はスタックの先頭から正ポーランド方式で行っている。

例:

なお、機能ならびに引数の表記には !{項目ＩＤ} を用いることが可能である。例えば、

```
Program|prog:= edit !{switch}:
```

といった表記がある場合、**prog** が実行されると、**id=edit** で示される項目に、**id=switch** で示される項目の現在値がはいる。

5.1. Program 項目に関する特記事項

5.1.1. 特別なプログラム名について

特別なプログラム名として、**BEGIN , END** がある。これらの名前を持つプログラム項目は、それぞれ *xme* の開始直後、終了直前に呼び出される。（ａｗｋを思い出しましょう）

また、**pg** で指定されたプログラム項目が存在しない場合、通常は単純に無視されるだけであるが、**Default** という名前のプログラム項目があると、その項目が実行される。

プログラム項目内で、!{_SELF_}という参照は特別な意味を持つ、その場合、呼び出した側の項目の値が置換される。また単に_SELF_と書いた場合は呼び出した項目自身を意味する。

5.1.2. スタック評価について

スタックで現在評価に使える演算子は以下のものである。

表記	意味
+	加算
-	減算
*	乗算
/	除算
%	剰余
==	数値として等しい
<	数値として小さい
<=	数値として小さいか等しい
>	数値として大きい
>=	数値として大きいか等しい
~==	文字列として等しい
~<	文字列として小さい
~<=	文字列として小さいか等しい
~>	文字列として大きい
~>=	文字列として大きいか等しい

尚、数値演算は全て３２ビットの符号付き整数として行われる。

---

push 1:push 2:push +:eval ☞ 答3
push 1:push 3:push -:eval ☞ 答2
push 1:push 3:push -:push 1:push 2:push +:push +:eval ☞ 答5

[3] 現在 map/unmap できるのは、**Button** と **Toggle** だけである。

なお、機能ならびに引数の表記には !{項目ＩＤ} を用いることが可能である。例えば、

```
Program|prog:= edit !{switch}:
```

といった表記がある場合、**prog** が実行されると、**id=edit** で示される項目に、**id=switch** で示される項目の現在値がはいる。

5.1. Program 項目に関する特記事項

5.1.1. 特別なプログラム名について

特別なプログラム名として、**BEGIN , END** がある。これらの名前を持つプログラム項目は、それぞれ *xme* の開始直後、終了直前に呼び出される。（ａｗｋを思い出しましょう）

また、**pg** で指定されたプログラム項目が存在しない場合、通常は単純に無視されるだけであるが、**Default** という名前のプログラム項目があると、その項目が実行される。

プログラム項目内で、!{_SELF_}という参照は特別な意味を持つ、その場合、呼び出した側の項目の値が置換される。また単に_SELF_と書いた場合は呼び出した項目自身を意味する。

5.1.2. スタック評価について

スタックで現在評価に使える演算子は以下のものである。

表記	意味
+	加算
-	減算
*	乗算
/	除算
%	剰余
==	数値として等しい
<	数値として小さい
<=	数値として小さいか等しい
>	数値として大きい
>=	数値として大きいか等しい
~==	文字列として等しい
~<	文字列として小さい
~<=	文字列として小さいか等しい
~>	文字列として大きい
~>=	文字列として大きいか等しい

尚、数値演算は全て３２ビットの符号付き整数として行われる。

---

push 1:push 2:push +:eval ☞ 答3
push 1:push 3:push -:eval ☞ 答2
push 1:push 3:push -:push 1:push 2:push +:push +:eval ☞ 答5

[3] 現在 map/unmap できるのは、**Button** と **Toggle** だけである。

5.2. Store 項目の使い方

擬似項目 **Store** は、パネル上には表示されない項目であるが、**id** の指定が可能であるため、一時的な値の保持、実行プログラムの指定などに用いられる。例えば、

```
Self:¥
    :xp#10:yp#10:wi#200:hi#60:
Button:¥
    :xp#5:yp#5:wi#90:hi#20:tx=switch:¥
    :ux=echo !{switch} !{edit} !{yes/no} !{store}:id=switch:
Edit:¥
    :xp#100:yp#2:wi#90:li#1:tx=edit me:id=edit:
Toggle:¥
    :xp#5:yp#30:wi#100:hi#25:tx=yes/no:tv=yes:id=yes/no:pg=ah:
Store:¥
    :id=store:tx=myValue:ux=echo AH:
Programlah:¥
    :exec store edit:
```

といった書き方が可能になる。**!{store}** は、**myValue** を参照し、**:exec store edit:** は、**echo AH** を実行して、その結果を **edit** に代入する。

5.3. ux, pg, wp の関係

ux , pg , wp は、それぞれ何かを実行するという定義であるが、現バージョンにおける、各者間の関係について述べよう。以下のような項目があると考える。

```
Button:ux=MyCommand:wt:wp=Done:pg=Init:
```

この場合、この項目が選択された場合の典型的な実行順序は、以下のようになる。

pg ☞ ux ☞ (ux が終了) ☞ wp

もし、**wt** が設定されていないとすると単に:

pg ☞ ux

のように実行されるだけである。

実はこの議論は完全ではない、厳密に言えば **pg=** の中で自分自身の **exec** をおこなう場合、事態はもっと複雑になる。正確な議論は以後の版に譲るものとする。ただし注意として、絶対にやってはいけないことだけを述べておこう。

注意

wp= で呼ばれるプログラム項目内で、呼んだ項目を **exec** してはいけない。例えば

```
Button:ux=MyCommand:wt:wp=Done:pg=Init:id=Caller:
Program|Done:exec Caller:
```

と書くと、これは **Caller** ☞ **Done** の無限リカーシブコールを引き起こす結果になる。

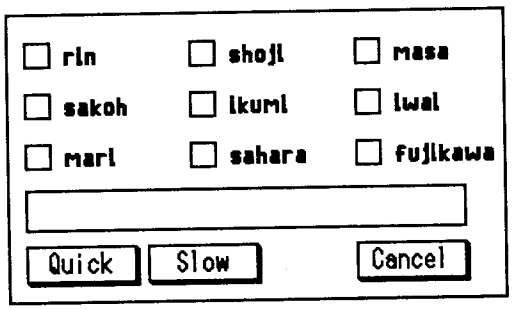
6. 例題

以下に示す例題は、***ksh*** と ***xme*** を使った簡単なゲームプログラムである。仕様としては:

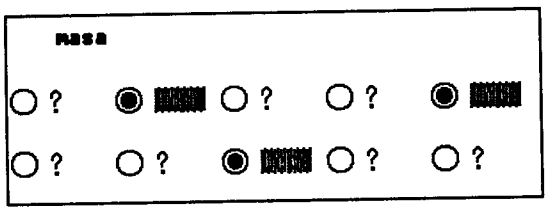
参加者の名前を登録する ☞ ゲームを行なう

という二段階に分れている。下に各画面を示しておく。

参加者登録画面



ゲーム実行画面



以下にリストを示す。

```
#!/bin/ksh
# get names
BITMAP=/usr/mmb/sakoh/lib/bitmaps
set $(xme -o' ' -f - << __EOF__
Self:wi#306:hi#180:¥
    :nc:¥
    :ic=${BITMAP}/coffee.ic:¥
    :im=${BITMAP}/coffee_mask.ic:
Button:dm#2:xp#10:yp#145:wi#70:hi#25:tx=Quick:et:pl:tv=quick:fv=:
Button:dm#2:xp#85:yp#145:wi#70:hi#25:tx=Slow:et:pl:tv=slow:fv=:
Button:dm#2:xp#214:yp#145:wi#70:hi#25:tx=Cancel:et:pl:tv=cancel:fv=:
Edit:xp#10:yp#110:wi#270:li#1:pl:
Toggle:xp#10:yp#10:wi#100:hi#30:pl:tp=CheckBox:fn=accordb:fv=:¥
    :tv=rin:tx=rin:
Toggle:xp#112:yp#10:wi#100:hi#30:pl:tp=CheckBox:fn=accordb:fv=:¥
    :tv=shoji:tx=shoji:
Toggle:xp#214:yp#10:wi#100:hi#30:pl:tp=CheckBox:fn=accordb:fv=:¥
    :tv=masa:tx=masa:
Toggle:xp#10:yp#42:wi#100:hi#30:pl:tp=CheckBox:fn=accordb:fv=:¥
    :tv=sakoh:tx=sakoh:
Toggle:xp#112:yp#42:wi#100:hi#30:pl:tp=CheckBox:fn=accordb:fv=:¥
    :tv=ikumi:tx=ikumi:
Toggle:xp#214:yp#42:wi#100:hi#30:pl:tp=CheckBox:fn=accordb:fv=:¥
    :tv=iwai:tx=iwai:
Toggle:xp#10:yp#74:wi#100:hi#30:pl:tp=CheckBox:fn=accordb:fv=:¥
    :tv=mari:tx=mari:
Toggle:xp#112:yp#74:wi#100:hi#30:pl:tp=CheckBox:fn=accordb:fv=:¥
    :tv=sahara:tx=sahara:
Toggle:xp#214:yp#74:wi#100:hi#30:pl:tp=CheckBox:fn=accordb:fv=:¥
    :tv=fujikawa:tx=fujikawa:
__EOF__)
# initial proc
# echo PLAY="$*"
#
if [ $1 = "cancel"  -o $# = 1 ]
then
    exit 0
else
    KIND=$1 ; shift
fi
LOG=~sakoh/lib/clog.${KIND}
#
typeset -i I=0
for P
do
    ATT[ ${I} ]=${P} ; (( I = I + 1 ))
done
# randomize
(( CNT = I ))
while (( I > 0 ))
do
    (( R = RANDOM % CNT )) ; (( I = I - 1 ))
    TMP=${ATT[ ${I} ]} ; ATT[ ${I} ]=${ATT[ ${R} ]} ; ATT[ ${R} ]=${TMP}
```

```
done
# game
echo " "
#
DAT=$(date)
echo "*DATE*" ${DAT} >> ${LOG}
echo "${KIND} : Game starts at ${DAT}"
echo "Players are:"
PLAY="${ATTI * ])"
for NAME in ${PLAY}
do
    echo "* ${NAME}"
done
#
# game function
#
GAME () {
awk '
BEGIN {
hit='$3';cnt=0;WI=60;HI=35;GAP=5;YN=2;XN=5;YO=2;
printf "Self:wi#"(WI+GAP)*XN+YO*2":hi#"(HI+GAP)*YN+YO*2+HI":0;
printf "Text:xp#"WI/2":yp#"YO":wi#"(WI+GAP)*XN+YO*2":li#1:tx='$1':fn=kilterb:0;
for (y = 0; y < YN ; y ++)
 for (x = 0; x < XN ; x ++) {
  if (cnt++ == hit)
   printf "Toggle:tp=RoundToggle:xp#%d:yp#%d:wi#"WI":hi#"HI" ☞ 折り返し
            :dv:tv=Hit!:fv=?:et:pe:0, x*(WI+GAP) + YO,      ☞ 折り返し
             y*(HI+GAP) + YO + HI;
  else
   printf "Toggle:tp=RoundToggle:xp#%d:yp#%d:wi#"WI":hi#"HI" ☞ 折り返し
             :dv:tv='$2':fv=?:pe:pg=p%d:id=p%d:0,           ☞ 折り返し
              x*(WI+GAP) + YO, y*(HI+GAP) + YO + HI, cnt, cnt;
  printf "Programlp%d:disable p%d:0, cnt, cnt;
 }
}' < /dev/null | xme -f - | wc -l
;}
# game
#
while true
do
    TMP_ATT=
    (( SAME = 1 ))
    if [ ${KIND} = "quick" ]
    then
    (( MAX  = 0  ))
    for n in ${PLAY}
    do
        (( SEED = RANDOM % 10 ))
        S=$(GAME ${n} "Oops" ${SEED})
        echo ${n}:   ${S}
        echo "${n} : ${S}" >> ${LOG}
        if (( ${S} == ${MAX} ))
        then
        TMP_ATT="${TMP_ATT} ${n}" : (( SAME = SAME + 1 ))
```

```
        elif (( ${S} > ${MAX} ))
        then
        TMP_ATT=${n} ; MAX=${S} ; (( SAME = 1 ))
        fi
    done
    else
    (( MIN  = 11 ))
    for n in ${PLAY}
    do
        (( SEED = RANDOM % 10 ))
        S=$(GAME ${n} "Safe" ${SEED})
        echo ${n}:   ${S}
        echo "${n} : ${S}" >> ${LOG}
        if (( ${S} == ${MIN} ))
        then
        TMP_ATT="${TMP_ATT} ${n}" ; (( SAME = SAME + 1 ))
        elif (( ${S} < ${MIN} ))
        then
        TMP_ATT=${n} ; MIN=${S} ; (( SAME = 1 ))
        fi
    done
    fi
    if (( SAME > 1 ))
    then
    echo "決戦です : ${TMP_ATT}"
    echo "*RETRY* ${TMP_ATT}" >> ${LOG}
    PLAY=${TMP_ATT}
    else
    echo "${TMP_ATT}さんの負けです"
    echo "*LOST* ${TMP_ATT}" >> ${LOG}
    break
    fi
done
#
```

3rd SEA Environment Workshop Position Paper

氏名	西岡健自　　種別 ☑会員 no 871190 ☐一般
所属	横河電機(株) 研究開発 4部第1研究室
住所	〒(180) 武蔵野市中町 2-9-32
TEL	(0422) 54-1111 (3370) FAX (0422) 55-0461

仕　事

ソフトウェア研究開発。
特に，ソフトウェア開発環境の構築。

作業環境

VAX 11/780 (VMS)
HP 9000/840 (HP UX)
HP 9000/350 (〃)
9801 (MS DOS)

} LAN + Net 1 (ターミナルネットワーク)

現在の問題

SPIDER モデルに沿った C 言語向け統合化開発環境の構築。(SPIDER モデルについてはポジションペーパ参照)

* 統合化データベースモデル
* カスタマイズ可能な情報抽出方式
* フィルタ生成系

次ページの討論テーマ票にも，必要事項をご記入の上本紙とあわせて 12 月 15 日まで（締め切り厳守！）に下記の SEA 事務局までお送りください．もちろん，本式の Position Paper も同封していただくほうがベターです（既発表の論文を添付していただいても構いません）．

ソフトウェア技術者協会 (SEA) 〒102 千代田区隼町 2-12 藤和半蔵門コーブビル 505
TEL: 03(234)9455　FAX: 03(234)9454

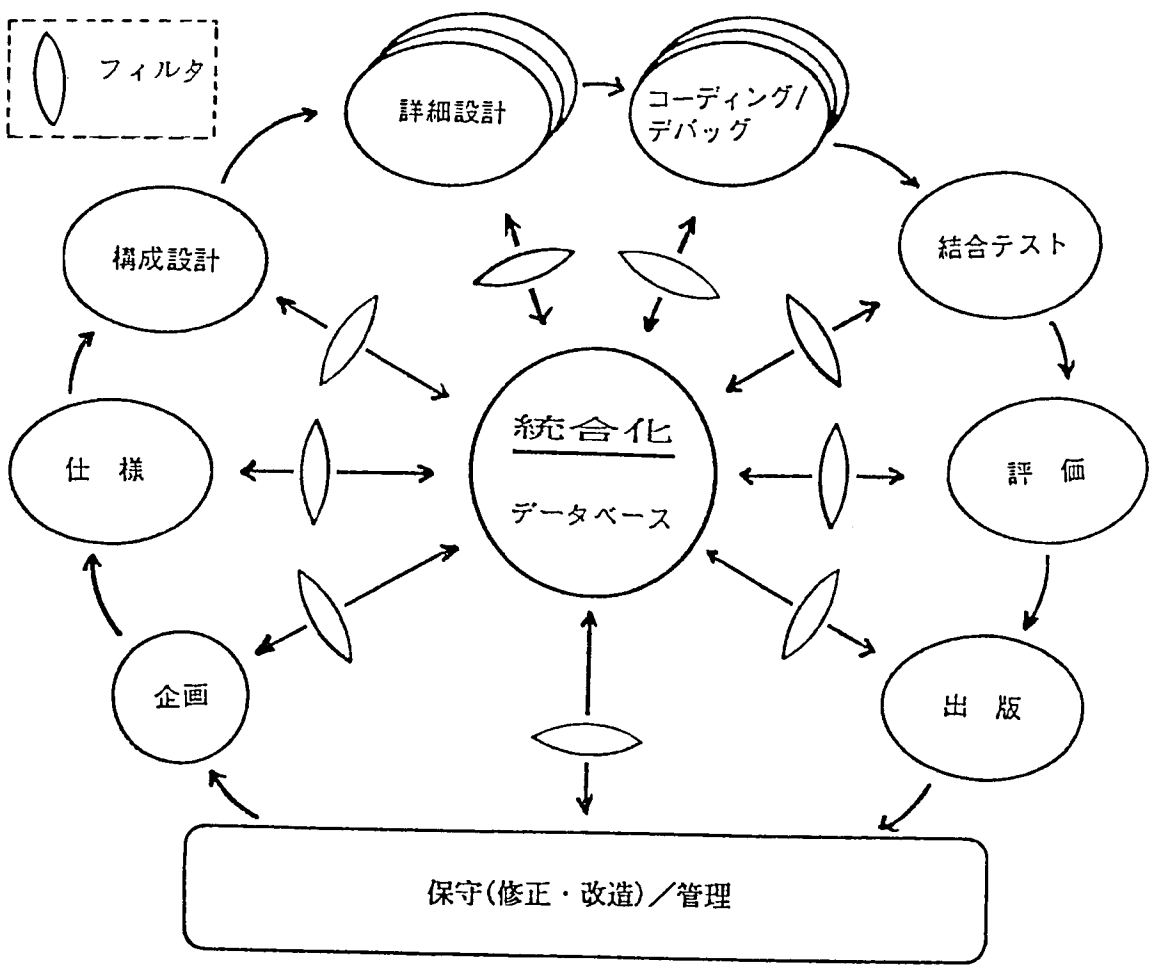
3rd SEA Environment Workshop (1988.1.27-29)

ポジションペーパ

横河電機株式会社
西岡健自

ＦＡ、ＰＡ、ＬＡ分野を軸に営業展開を行なう弊社では、全社技術部に於けるソフトウエア生産性向上を目的としたプロジェクト　ＳＰＩＤＥＲ（Software Productivity Improvement Developer）を三年前にスタートさせた。ＳＰＩＤＥＲプロジェクトは我々の属する研究開発部を推進母体として、ＵＮＩＸ、Ｃ言語，ＬＡＮを中心に据えた開発環境の展開を狙い、かつ，その後スタートしたΣプロジェクトへの参加・協力も行なってきた。

ＵＮＩＸ上の開発環境を構築する上で、我々は従来からのウォータフォールモデルに代わるものとして，中心に統合化データベースを据えた下図のようなＳＰＩＤＥＲモデルを提唱した。



ＳＰＩＤＥＲモデル

このＳＰＩＤＥＲモデルの特徴は各フェーズの出力ドキュメントをフィルタと呼ぶ一組のソフトウェアツール群経由で統合化データベースにリンクさせることにより，ソフトウェア開発途上の全技術情報（開発中のソフトウェア自体の情報＋管理情報）を一元管理する点にある。フィルタは次の３つのツールから構成される。

１）フレームエディタ：ドキュメントスタイルからの逸脱を防止するとともに，記述作業を機械化する構造エディタ。
２）情報抽出ツール　：フレームエディタで記述されたドキュメントの含む全技術情報をドキュメントスタイルに依存しない，ドキュメント間で共通の形式で抽出し統合化データベースに格納するツール。
３）逆生成ツール　　：統合化データベースの情報に基づいてドキュメントスタイルに沿ったテキストファイルを生成するツール。

設計書，ソースプログラム，テスト成績書といったソフトウェア開発途上で出力されるドキュメント毎にこのようなフィルタを用意することによって，次のような効果を期待できる。

１）ドキュメント間で重複する情報を統合化データベース上で一元管理でき，開発フェーズ間の情報の整合性を保証できるとともに，逆生成／抽出処理により他のドキュメント情報に基づく別のドキュメントの部分合成，ドキュメント間の変更／修正の自動反映が可能となる。
２）フレームエディタにより，記述作業が効率化する。
３）汎用性の高いソフトウェア部分を統合化データベースより直接切り出せ，信頼性の高いソフトウェア部品が自然に蓄積される。
４）統合化データベースの監視によりプロジェクト管理を可視的に行える。

さらに，ＳＰＩＤＥＲモデルには次のような利点がある。

１）従来のウォータフォール的開発形態から違和感の少ない形で新しい開発環境を提供できる。
２）逐次的に構築でき，ユーザ負担の少ない形で逐次的に提供できる。
３）フィルタのカスタマイズにより多様なソフトウェアプロセスに対応できる。

ＳＰＩＤＥＲモデルに基づく開発環境は、当面，統合化Ｃ言語プログラミングシステムとして，詳細設計とコーディングフェーズに焦点をあわせて構築中であり，プロトタイプによる実現性の評価を終えている。現在は統合化データベースのデータ構造に知識処理的手法を加味して，より実用性の高いものとするとともに，フィルタ生成系の実現，統合化データベースの管理を含むマンマシンインタフェースの統合化，分散環境への対応を図りつつある。
なお，現在の統合化データベースモデルには，次のような情報が含まれる。

１）ソフトウェアプロセスに依存しない情報：
　・不可分の要素に分解された開発フェーズ全体に亘る技術情報
２）ソフトウェアプロセスに依存する情報：
　・各ドキュメントの属性情報，及び，技術情報との関係情報
　・各ドキュメントのテキストスタイル情報

第３回　SEA環境ワークショップ

セッション３

ワークステーション＋ネットワークによる分散環境の構築

チェアマン　：**中野　秀男**（大阪大学）
プレゼンテータ：**加藤　　朗**（東京工業大学）
：**酒匂　　寛**（ソフトウエア･リサーチ･アソシエイツ）
：**西岡　健自**（横河電機）
レポータ　：**石坂　幸一**（ケイケン長野データセンター）

1.はじめに

中野：ポジションペーパーを見て，こういうものを作りました．（ＯＨＰによるアイコン－話題と話題提供者名が書いてある）話題が途切れましたら，アイコンをさします(爆笑)クリックしたら喋って下さい．(爆笑)では，宜しくお願いします．

2.プレゼンテーション

2.1.広域ネットワークの問題点

2.1.1.環境としてのＪＵＮＥＴ

加藤：先ず，ＪＵＮＥＴの簡単な現状報告をしておきます．組織を超えた研究者相互のコミュニケーションを計る目的で３年前から始め，現在アクティブにメールが届く所が大体１１０組織（実験中が約１０），計算機の数で１０００（多分１３００位になっている筈）大体全国をカバーしていますが，東京と大阪に集中している．今の処電子メールと電子ニュースだけです．そのうちもっと色々なことをやりたい．

海外のアカデミック．ネットワークと情報交換をしています．ただ，トラフィックを見ていると，向こうから入ってくるばかりで，出て行かない．ＵＵＮＥＴはＵＳＥＮＩＸがお金を出して運営していて，情報の流れが５０／５０なら，お金も５０／５０でいいよと言っているのですが，結局日本からは何も出て行かないのが現状です．

2.1.2ＪＵＮＥＴの「実践的環境」への貢献（１）

広域ネットワークに限って話をします．僕がここで広域と言っているのは「組織を超えた」という意味で，隣合ったビルでも会社が違えば広域ネットワークです．ＪＵＮＥＴは広域ネットワークです．明日のスキーの集合場所は等のメールも飛び交っていますが，一応表向きにはアカデミックな議論がされている．ＡＲＰＡみたいにリアルタイムではないが，半日位で届く．学会などはポジションペーパーを出して，アブストラクト，本チャンというサイクルだと，研究会レベルで２カ月，国際会議だと１年位かかってしまう．ＪＵＮＥＴでは今の議論が出来ると言う訳です．

2.1.3.ＪＵＮＥＴの「実践的環境」への貢献（２）

ソフトウェアとはプログラムとかマニュアルだけでなく，どう使いこなしてゆくかも利用技術に入っていると思うが，そういう利用技術はお店には売っていない．ウチではこうやっていますという情報が非常に必要になる．大学などはビンボーですから，なかなかＷＳなど買ってくれない．１台のＳｕｎ－３を１０人で使うなど，どうやって旨く生かすかという情報交換がされています．管理面でも，例えばＮｅｗｓのあの辺はどうしたら良いのとか皆の環境を良くして行こうとする情報交換が，結構されています．

もう１つ大きなのはＰＤＳ．ｒｍａｐ，ｌｅｓｓ，ｓｐｓ，ｐｈｏｎｅなどはＪＵＮＥＴに繋がっている処は総て使っているが，そうでない処にはあまり知られていない便利なツール．ライセンスの制約がないので，ソースコードから皆思い思いにバグフィックスをする，アップデートをするなど，ＰＤＳはネットワークを通してどんどん成長して行くので，下手な売っているプログラムよりは信頼できるものが少なくない．こういうことを通じて，お互いの環境向上に寄与しているのではないか．

2.1.4.広域ネットワーク環境としてのＪＵＮＥＴ

ＡＲＰＡは，始めにＡＲＰＡがあってそれからローカルエリアネットワークが発達してきたが，日本は逆で始めにＬＡＮがあった．ネットワークを運営して行く上での問

題は，もともと独立だったのを一緒にまとめるわけですから，管理レベルが違う．例えば計算機がメンテで３時間止まりますと言ってくる所もあれば，１週間呼んでこないところもある．それから，計算機資源の貧富の差が激しい．(笑い) 例えば大学で，物理屋さんとか化学屋さんがネットワークが必要だと言うと大学側では計算機をアロケート出来るけれども，計算機屋だけがギャーギャー言っても全然買ってくれない．

管理者はみんな忙しい．新たなサイトが入ってきた時に，どうやって広域ネットの運用技術を持ち上げて行くかが問題になっている．それを放っておくと，１週間に１回僕がどこかへ行かなくてはいけないという事になってしまう．今の問題点は，良く知っている人の時間を占有してしまう事．管理レベルを下げないで如何に手を抜くかを真面目に考えなければいけない．

2.1.5.分散環境に向けて

ＷＳがあってＮｅｔｗｏｒｋがあるから，分散環境ですというのはウソです．せいぜい協調処理環境と言うにとどめて頂きたい．分散処理とは「ホスト名を意識させない」という性質が要求される訳です．あそこのマシンは軽いからと，プロセスが自らＭｉｇｒａｔｅして行ってそこで仕事をして返す．(Process Migration をどうしたら良いかと言う話は，今度のＵＳＥＮＩＸで Battle of Process Migrator というセッションがあって，３名の発表が有ります♪) 何処の計算機へ行っても同じ様に走れるというホストを意識させない環境は，何処の計算機でもきちっと管理されている状態でないと実現できない．こういう前提の上に良い計算機と速いネットワークがあり，そこで始めて Process Migration を如何するか，リソース･ネーミングを如何するか，ネーム･サーバーを如何するかという基礎技術によって，だんだん分散環境に近づいて来る訳です．これぞ分散環境と言うのは，今はあまりなくて，例えばＭａｃｈがどれ位良いかというのは使った事が無いので分かりません．そろそろ日本に入って来そうなので遊んでみたいと思っています．

2.1.6.会場より質問

藤野：　Ｍａｃｈはロードリプレースをやってくれるのですか？

加藤：　よく分かりません．もしほんとうにやっているとすると，アブストラクトのペーパに書いてある筈ですが，あまり書いてないので多分これからだと思います．それで，１番議論がしたかった事は次の様な事です．

・ところで，本当に分散環境は必要か？

例えばＶＡＸとか大きな計算機が１台あって，専任管理者が居て，ダンプも取ってくれるし，あれが欲しいと言えばインストールもしてくれる，そういう環境とＷＳがズラズラという環境とどっちが良いのだろう．ひとつの使い方としてＷＳには仕事をさせない，例えばウィンドのマニュピレーションとか画面回りとか入出力の管理だけをやらせ，計算等はＥＴＡ１０やＣＲＡＹなどにやらせる．ＷＳはただ賢い端末として動く．こういう使い方が１番良いのではないか．

僕が気になっているのは，分散環境でなければ出来ない仕事はどれ位有るのかということです．１台の計算機にやらせておけばよい仕事を，単に振り分けて，問題を複雑にしている．盲腸を叩きつぶして，腹膜炎を起こしたようなイメージがある．今みたいにＶＡＸにＷｉｎｄｏｗがのるカードをガシガシ刺して，そこから各々の端末に引いて来てＸやＳｍａｌｌｔａｌｋを走らせる，という程度なら集中の方が絶対楽なわけで，何でわざわざ散らさなければいけないのか．

今は分散環境は実現されていなくて，基礎技術が色々な所で研究されている．今後どうしたら良いかがクローズアップされている．単に論文が発表されて，それで御仕舞になるのか，本当に分散環境がちりばめられるのか….

取り留めのない話ですが以上で終ります．(拍手)

中野：　何か質問がございましたらお願いします．

野中：　私は通信衛星の地上用の通信装置を作っているのですが，通信衛星というのは割とコストが高く，大容量の所しかなかなか使われないという現状があります．大陸間とか通信量の多い幹線には既に通信網が行き渡っていて，これから商売として余り伸びないのではないか．新しい衛星通信をどこに使って行くかが，当社の事業部の問題となっている訳です．

今の地上の有線系の通信路に衛星がどう対抗して行くか，衛星の有利な特性，１つには広域同報性を生かして行くしかない．それに向いているのが，いわゆるＪＵＮＥＴのようなニュースシステム．地上をバケツリレー式で広がって行ったものが，いっぺんに東京，大阪，九州でワッと受けられたら，トラフィックの上で大変有利になるのではないかと思っている．

今まで通信と言うのは１対１で発展して来て，放送というものと完全に区別されていると思うのですが，ニュースというアプリケーションが出て，通信と放送が大分近付い

て来た．その辺の研究をやったら良いんじゃないかという話を周りの上司にしているのですが，日本電気という会社は電話通信で発達してきたものですから，やはり通信の主体は「音声」であるということが言われているのです．

そんなことで，ニュースに衛生通信が使えるんじゃないかと思うんですが，その辺もし何か有りましたらコメントを頂きたいのですが．

加藤： 多分使えます．米国のスターゲート･プロジェクトはそういう話でして，これはＵＳＥＮＩＸがプッシュしているのですが，今のアメリカのＵＵＣＰｎｅｔはパンクしているので，そういう方法でやりたいというプロポーザルをローレン･ワインシュタインという人が出しています．マーク･ホートンとローレンとあと２人がやっている．アメリカはラッキーなことに衛星のチャネルが取れたので，出来るようになったという話を聞いたことがある．日本でも出来なくはないが，僕らのネットワークはまだそこまでのパワーがない．ひとつの問題は，寝ている人が居るように落ちているマシーンというものが有って，それが結構難しい問題です．ノードがフォールトレラントに設計されていれば良いが，日本では金に糸目をつけずにそういう事をするという考えが市民権を得ていませんので直ぐには実用化しそうに無い．只，もしチャネルを提供して頂けるという話があれば(笑い)これはもう前向きに考えたいと思いますので宜しくお願いします．

伊野： 私は仕事柄，アメリカ人と電子メールのやり取りをしたい事が非常に多い．現在ＪＵＮＥＴに入っていて，アメリカのＣＳＮＥＴとかＡＲＰＡＮＥＴに入っている人と電子メールを交換したいんですが，日本では東京大学の何とかという先生が非常にうるさい手紙を出しているという話で,..... 今後の見通しをお伺いしたいのですが．

加藤： １つの方法は，ＳＲＡがＣＳＮＥＴに入ることです．(笑い)それが１番確かな方法です．東大のゲートウエイは文部省だからということではなしに，ＣＳＮＥＴとの契約によって大学だけにしてくれと言われています．

もう１つの方法はＫＤＤのＫＤＤＬＡＢ（国際電信電話でなしに別団体）というマシーンがありまして，そこからＵＵＮＥＴへコンタクトするというパスがある．毎日ガンガン使うようでしたら，そこの何処かへ直接リンクを張るのが多分１番正しいやり方だと思います．ＡＲＰＡとかＣＳＮＥＴではセキュリティが問題になっていますが，ＫＤＤ経由で出した場合はＵＵＮＥＴに行って，そこはＡＲＰＡのお墨付があってそこからはちゃんと行く．今はＫＤＤのリンクは皆でお金を持ちましょうということでやっていますので，通信量が増えている事は確かですので，もっと速いレスポンスの要求があるのも確かで，お金はちゃんと払いますという事ですからもっとまめに呼ぶということは多分，現実的な問題になって来ると思います．今度の国際科学技術通信網利用クラブの幹事会があったら報告しておきます．

ＵＵＮＥＴのマシンがよく落ちるという問題等色々あるんですが，確かに企業の方にとってはなかなか遠くなってしまっているというのが現実だなというのはよく分かります．

中野： あんまりやるとＪＵＮＥＴの説明会になってしまうので．とりあえずここは一旦切らせて頂きます．どうも有難うございました．(拍手)

2.2.ツール統合と環境改善

酒匂： 元々はユーザ･インターフェイスの所に出したのですが，分散環境の方になってしまいましたので…．今日の話は，分散環境はどうあるべきかという概念的なお話はスキップして，ツールをネットワーク環境の中に取り込んで統合化したり，使いやすいユーザーインターフェースを作って行く礎となるシステムとしては，どの様なものが適当か，それについて幾つか実験したので，その報告という形にしたいと思います．

ＳＲＡではＵＮＩＸを１９８０年頃から使っていますが，ずっとキャラクターベースのＩ／Ｏしかサポートされて来ませんでした．究極のキャラクター．アプリケーションというのがｅｍａｃｓですが，あれ以上の世界を求めようとすると，キャラクターベースからは抜け出さなければならない．それで，Ｘｅｒｏｘのマシーンなどでアプリケーションが提案されて来て，遅ればせながらＵＮＩＸもウィンドゥを使わないと世の中に取り残されてしまうという事で， やっとＸ－ｗｉｎｄｏｗとかＧＭＷ，ＳｕｎＮｅＷＳ等が用意されて来た．

2.2.1.XME

(An User Interface Construction Mechanism)

Ｗｉｎｄｏｗシステムを構築するライブラリーは確かに提供されていて，それを用いればユーザーフレンドリーなアプリケーションが作れると能書には書いてあるのですが，Ｘ－Ｗｉｎｄｏｗを使って例えば Kernighan の “hellow world “を書く例題をやろうとすると，１００行近くのＣのソースを書かなければならない．そういった事を救う為にＴｏｏｌ－Ｋｉｔ等用意されたが，それでも

ＵＮＩＸ上にはｔｏｏｌを統合する為の優れた手段としてＳｈｅｌｌ　Ｓｃｒｉｐｔというものが存在している．Ｓｈｅｌｌの基本的な思想は，パイプメカニズムとか制御文構造等で，作られたツールを旨く統合して行こうというもの．その優れた支援システムがＷｉｎｄｏｗを導入したとたん失われてしまう，ということになると，それはＵＮＩＸではなく単なるウインドウアプリケーションが１個ずつ走っている，普通のマシンと同じになってしまう．

そこでウインドゥ環境下で，Ｓｈｅｌｌプログラムの様に手軽に使えるシステムを作って見ようと思った訳です．これがＸＭＥなるツールです．最初の目標としては，インターフェースの形状を気軽に定義できる事，Ｓｈｅｌｌプログラムの様な強力なツール統合手段と組み合わせてウインドゥ･インターフェースを構築出来ること，それからプロセス間通信を用いて複数の処理の間で自由に通信が出来る事を考えました．現在のＳｈｅｌｌプログラムでは，Ｓｈｅｌｌの中でのプロセス間通信はパイプしか無く，後から追加されてきたネットワーク関係の通信機能はＳｈｅｌｌの中にちゃんと入っているとは言い難い．ネットワーク･ファシリティが有るにもかかわらずＳｈｅｌｌから触れない．

Ｘ－ｗｉｎｄｏｗの上に作られているので，現在ＸＭＥと名付けられているが，そのうちにこのＸが取れて普通の汎用ウィンドゥ･システムの上に乗せられる形で，こういった物を構築して行きたい．

2.2.2.Architecture

ＸＭＥというのはユーザー･インターフェースのコンストラクション･メカニズムということで定義しています．インタプリターが天辺にあって，その下にパネル･コントロール，ネットワーク･コントロール，プロセス･コントロール，インターナル･コントロールのファシリティが有ります．これらは総てインタプリターで，Ｌｉｓｐ等と同じ様に送られてきた文字列をプログラムとして解析し，答える．

2.2.3.XME I/O Diagram

ＸＭＥはＵＮＩＸの持っているリソースにとりあえずタッチ出来る．パイプも使えるし，ＰＴＹにもアクセス出来るし，ネットワークを通してよそのプロセスともコミュニケート出来る．この様にシステムにディペンドしている部分とインディペンデントな領域とがある．(図２－２－１参照) 右下の部分がＵＮＩＸに依存している部分．

message exchange と書いてある所は，ＸＭＥ同士が理解できるプロトコルでメッセージがやり取りされており，このメッセージ自身がプログラムでも有り得ます．これを受け取った相手側のＸＭＥは，「自分がやらなければいけない事だな」と解釈して実行する．例えて言えば，Ｌｉｓｐのインタープリタがｓ式をいきなり送ってｅｖａｌｕａｔｅさせている，といった感じの物．

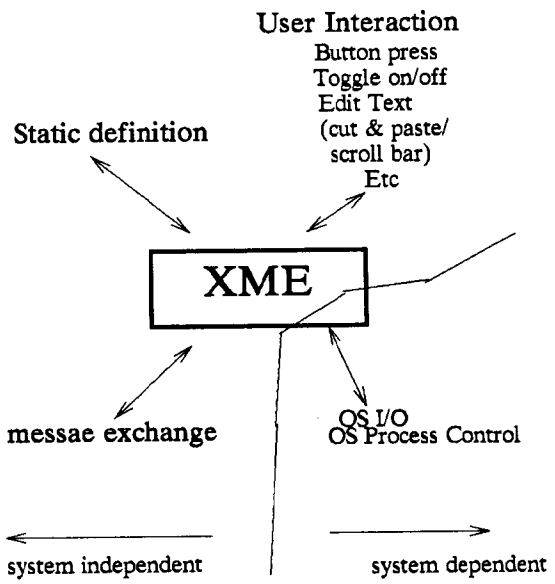


図２－２－１　XME I/O Diagram

Static definition は，ユーザーが普通のテキスト･ファイルを然るべきエディターを用いて書くとＢｕｔｔｏｎとかＴｏｇｇｌｅとかを簡単に定義出来て，なおかつ，中でＯＳの持っているリソースに自由にタッチ出来る．当然プロセスを呼び出したり，普通のストリームＩ／Ｏを呼び出したりする事も出来る訳です．

2.2.4.Easy to Design Panels

具体的には，これは現在用意されているＸＭＥのためのエデイターの画面ですが，(図２－２－２参照) この様に，アイコンにテキストをバインドする，テキスト･ベースで文字列をバインド出来るという所がＸＭＥの利点であります．すべてのユーザーインターフェースの定義が，単なるテキストで出来るという事です．するとＵＮＩＸの豊富なテキスト処理系に簡単にインプリメント出来る．

2.2.5.Discussion Board system

次は，プロセス間通信も（ｎ人Ｔａｌｋ）出来ますというお話しですが，ダイヤグラムはこんな風になって居ます．(図２－２－３参照) 天辺にマネージャーが居て，Ｔａｌｋ

Easy to Design Panels

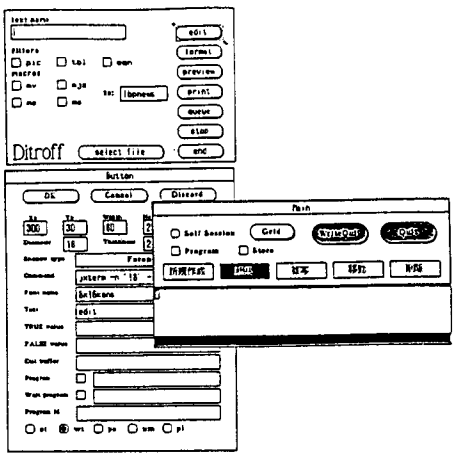


図２－２－２

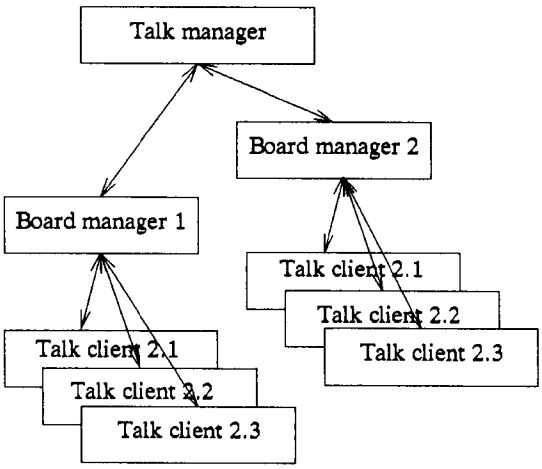


図２－２－３　Discussion Board system

のクライアントが「ディスカッション･ボードでお話ししたいよ」と言うと，「このボードでお話ししなさい」とマネージャーから言われる訳です．あと何人か入って来ると，同じボードの上でゴチャゴチャお話しする．この様なアーキテクチャーの為には，当然プロセス間通信が自由に出来なければいけない．しかも，Ｓｈｅｌｌ　Ｓｃｒｉｐｔの中で使うという事は，「あのサーバーの云々‥」という風にシンボリックにテキストで書けなければいけない訳で，これはそういった物をサポートしている．（図２－２－４参照）

2.2.6.Tool Integration Capability

これは最初に作ったシンプルなアプリケーションです（図２－２－５参照）．ｔｒｏｆｆというテキスト･フォーマッターが有りますが，これはそれを作業する時のフロント･エンド･パネルです．ｐｉｃをかけるとか，ｔｂｌをやるとか，ｍａｃｒｏに何を選ぶとか，プリンターは何処に出すか，そういったルーチンワークを総て定義したのが，このパネルです．例えばここで「印刷」を指定すると，勝手にパイプを作ってｔｒｏｆｆとか色んなフィルターを掛けて，プリンターに送る所まで全部やってしまう．

2.2.7.Amusement（上海）

アプリケーションとしては，こんな物も出来ます（図２－２－６参照）．こういった物も定義ファイルを書けば出来ます．

2.2.8.Conceptual Architecture of a Prototype

概念的な話をちょっとだけします．これから先どうしたら良いか，という話ですが，基本的に，ツール統合という事を考えると，シェルに代わるものがあれば良い．ある人はＳｍａｌｌｔａｌｋの中で全部出来る，いやｅｍａｃｓで出来ると言ったりするが，それは大変結構なんですが，それらのものを更に組み合わせて上位のものを作って行くにはどうしたら良いか．ツールとツールをどうやって切り貼りするか，ということが重要になって来る．

ＸＭＥは単なるユーザーインターフェースのコンストラクション･キットですが，今考えて居る事は，話をもう少し進めて，ある程度汎用なサーバー･クライアント･モデル

Network Distributed

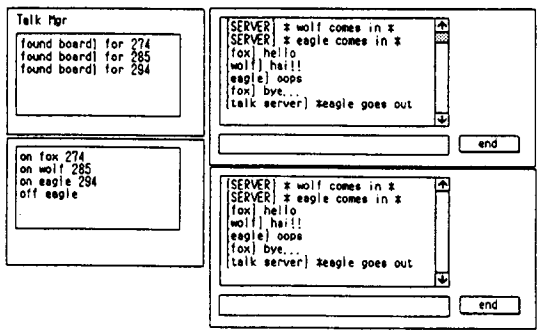


図２－２－４

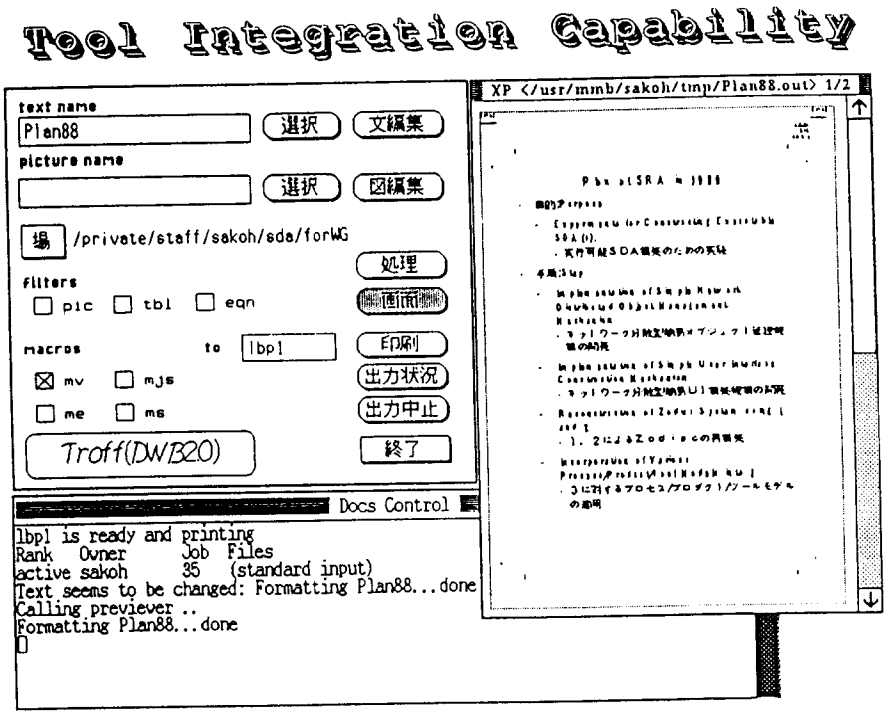


図2－2－5

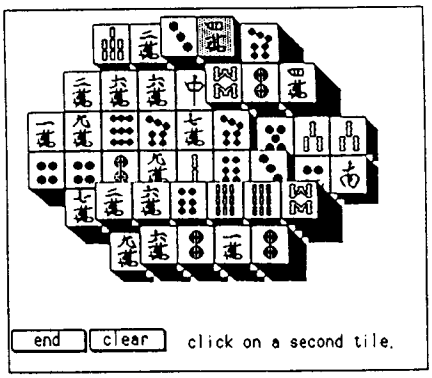


図2－2－6 Amusement

を構築出来る様なキットがあれば良いだろうという事です．XMEは，この中のあるクライアントの位置づけであったりサーバーになったりするが，こういった物を作って，簡単なデータとかプログラムをやり取りして，不用なものはいつも抱え込んで置かない，必要な物はプログラムを動的にロードして，要らなくなったら捨ててしまう．必要な情報は1箇所に置いておき，奇妙な分散，無用の分散を行わせない，といった考え方が良いのではないかと思っています．

2.2.9.The Reconstructed Zodiac system

Part1： for a specific project

どういう風にしてツールを組み合わせるか．例えば，私はかつてＺｏｄｉａｃシステムに関わっていましたが，このシステムは色々なツールがバラバラに存在している，１種の統合環境です．しかし，今現在うまく統合されているとは言い難い．この様にばら蒔かれたツールとかデーターベースに，それぞれ，こういうシンプルなサーバーをペタペタと貼り付けて行って，ユーザー･インターフェースの部分はさっきのＸＭＥの様な物を使う．自分の好きなパネルをペタペタ切り貼りしてアレンジして行けば良いのではないか．

Part2： for a Organization

これは，ひとつの開発環境のモデルです．複数のプロジェクトが１つ１つのセットを使っていた時に，ひとつのプロジェクトの中で使われている環境－Ｐｒｏｊｅｃｔ ｓｐｅｃｉｆｉｃ Ｚｏｄｉａｃ－の外側では先ほどのシンプルなサーバーを持って来て，インター･プロジェクト･サーバーという形で，これらと他のプロジェクトとが連携しながらリソース等を参照するという形になると非常に良いのではないか．こんな所で，いかがでしょうか．

2.2.10.会場よりの質問

中野： １つぐらい質問を受けましょう．

藤野： うちの社内ではサーバー･クライアント･モデルはダサいと言われている．何故かと言うと，ＵＮＩＸみたいに６０年代のＯＳだからこそ，サーバーなんていう物を作らなくちゃいけなかった．Ｍｅｓａを使って居る連中に言わせると，あんなサーバーなんてダサイやり方するからいけない，だいたいプロセス間通信をあんな風にオーバーヘッドの大きいやり方でやっているのはおかしい，小さなプロセスがいっぱいある様にしておけば，それらが勝手にお話してやってくれる‥という事になっている．

例えばＭａｃｈなんかでもスレッドみたいな感覚で出てくると思うんですが，今後の分散環境を考えて行く上で，ほんとにサーバー･クライアント･モデルってのは正しいのかな？という質問ですが．

酒匂： 良い質問です．(笑) 私自身も，大きなサーバーがドンと居てそれが何でもかんでも面倒を見るという環境が良いと思っている訳ではありません．私としても，各々の責任範囲がはっきりしている様な幾つかのプロセスがあって，それらがうまくｃｏｎｓｉｓｔｅｎｃｙを保ちながらやってくれるのが良いだろうと思っています．今私が考えているサーバーは特異なサーバーではなく，それぞれの実態はひとつひとつしかない，ちょっとしたメッセージを送ってやることによって特化する様なサーバー，しかも非常に軽いという様なものを作って，それをツールの切り貼りに使おうと思っています．

今のご質問は，サーバー･クライアント･モデルは主流に成り得るかという事ですが，確かに，でかいデータベース･サーバーが居て，それが何千件ものトランザクションを一気に処理するという様なやり方では，当然ＵＮＩＸマシンはもたない．アーキテクチャーはもともと悪い．そういったものは，やはり主流にはならないですね．どちらかというと，こういうアナーキーな，皆が勝手に自立を主張して「責任はおまえだ」とか言いながら，やり取りしている‥というのが良いのではないか．(要らない時には死んでいる) 必要なときにポンと出てきて，やりたい事だけやって，死んでしまう．

藤野： そうすると，どうしてもＩＰＣのメカニズムというのは問題になって来ると思うのですが‥．

酒匂： はい．私は現在ＢＳＤのＵＮＩＸの上でやっているから，ＩＰＣを使っている訳で，別にそれに組するものではありません．確かにＩＰＣのオーバーヘッドが大きいのは認めます．だから，無くせる物なら，無くした方が良いと思っています．要するに，ここの部分（図２－２－１で，システム依存のＩ／Ｏ部分とＸＭＥとのインタラクション）が今重たくてどうしようもない訳です．

とは言え，例えば小さなワーク･セットとかインターフェースの部分（インターフェースの部分は色々あると言いながら，ある程度定型があった方がよい）とかロジカルなＭｅｓｓａｇｅ ｅｘｃｈａｎｇｅとか，そういった物は，どのマシンに行っても変わらないのではないか，と言うことです．

深瀬： 酒匂さんを弁護する訳ではないのですが，ＵＮＩＸというＯＳですと，どうしてもサーバー･クライアント･モデルで作らざるを得ない．何故かと言うと，私も以前ｎ人Ｔａｌｋという様なチャティングのシステムで，１００チャネル位動かしてみた事があるんですが（笑い），そうすると，プロセスだらけになる訳です．それは，今のＵＮＩＸのメカニズム（バークレーだろうが，Ｓｙｓｔｅｍ－Ｖだろうが）ではもたない．カーネギー･メロンでやっているＭａｃｈプロジェクトのスレッドという様な，新しい概念を持ち込まない限りは，それは無理だろう．現状のＵＮＩＸの仕掛ですと，妥当な選択ではないかと，僕は思います．

酒匂: 有難うございました.

中野: では，ここら辺で取り合えず切る事に致します.

2.3.SPIDER モデルによる環境統合化

西岡: 我々が，この「統合化」にどうアプローチしているかお話して，ディスカッションの切掛けになればと思います.

2.3.1.開発環境統合化の新しい難題

私は卒業して１３年ソフトウェアの仕事をしていますが，その間８－９年は事業部でプロセス制御関係のソフトウェアを作って居ました．今は研究開発部門でソフトウェアの開発環境を作ろうとしています.

結論から言うと，ＷＳ＋ネットワークによる統合化は，次の様な新しい難題を持ち込んだのではないか.

(1) ユーザー（プログラムを作る人）の負担増大

(2) 類似コピーの散乱（バージョン管理が難しい）

(3) 開発環境の保守，拡張コスト増大

(4) セキュリティーの低下

ただ，基本的にはＷＳ＋ネットワークという環境は捨て難い.

2.3.2.統合化開発環境の実用化条件

では，統合化をどういう風に考えたか．統合化のユーザー要件としては，

(1) 困った所で役に立たなければいけない.

仕様とか設計の段階でなく，ほんとに困っているのは詳細設計以降，コーディング･デバックの部分.

(2) なにげなく使える.

従来の作業方針がガラリと変わる様では，絶対に使ってもらえない．例えば再利用ライブラリー化する為に，余計な仕事が増える様ではダメ．(従来方式との調和)

(2) 余計なことを考えずに開発に専念できる.

ネットワークであちこちつながると，何が何処に在るか余計な事を考えなければならない．(情報隠蔽)

(3) いつの間にか生産性が上がる.

始めのうちは，今までの延長上でちょっとしたツールを使っていて，遂次拡大して，気が付いて見たら飛躍的に生産性が上がっていた，というのが良い．(遂次構築，遂次運用)

(4) 自分達好みのやり方で仕事ができる.

この「達」が重要．各プロジェクト毎に何らかの標準的なやり方で仕事ができること．(カスタマイズ可)

ただ，これだけでは統合化はこう在らねばならぬというビジョンは見えて来ない．もう１つ別な観点から考える必要がある.

2.3.3.統合化開発環境のねらい

そこで，統合化の狙いは一体何なのかを考える．最終的には「ソフトウェアの生産性向上」(安易に使われる言葉だが，こう言わざるを得ない)．これには，再利用とか自動生成がよく言われる．自動生成は事務処理分野では，ある程度出来ているかも知れないが，プロセス制御，ＦＡ，ＬＡ関係ではまだまだ実用化できない．再利用できる部品は信頼性が高くなければいけない訳だが，信頼性を高くする事と作業効率とが両立出来る様な環境，これが統合化だ．とは言っても，これだけでは何も指導理念が出て来ない．そこで…

2.3.4.どこを改善するか

モジュール仕様書（詳細設計書）からソースプログラムに落とす時にエラーが起き易い．逆にソースを直した時に仕様書を直すのを，つい忘れてしまう．この様に，仕様書とソースプログラムとの不整合が起き易く，又変更作業の二重手間が起こる．更に，これが全て人手に任されているので，信頼性と作業効率の低下を招く．つまり情報の二重管理と機械化の不足が信頼性と作業効率の低下を招く大きな要因ではないかと考えた．これに対応する方法として情報統合化を考えた.

2.3.5.プログラム情報の統合化とは

上流側には詳細設計書(その上もあるが当面は考えない)がある．その中の構成要素として，データ構造について書いてある部分と，個々の関数について書いてある部分とがある．下流のプログラムについてもやはりデータに関する部分と関数に関する部分とがある．これがブレーク･ダウンされて行ってリンケージの段階になって，パラメータ名とか，パラメータ用途のコメントとか，データタイプと分かれて来ると，この辺ではドキュメントのフォーマットとは独立した形に出来るのではないか．フォーマット独立な情報として，バラバラに解きほぐす事が出来るのではないか，というのが出発点です．(図２－２－７参照)

2.3.6.グローバル･データのデータ構造と内部形式

解きほぐしたのが，これ．(図２－２－８参照) グローバル･データ名は属性と関係に分かれ，その各々のデータの中身は自然言語でもＣ言語でもない独特な内部表現形式でバラバラに解きほぐしてしまう．これを自動的にやるためにツールを用いる.

2.3.7.情報統合化プログラミング･システムの実現方式

詳細設計デザイナーは「モジュール仕様フレーム･エ

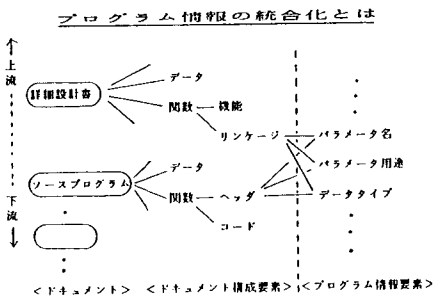
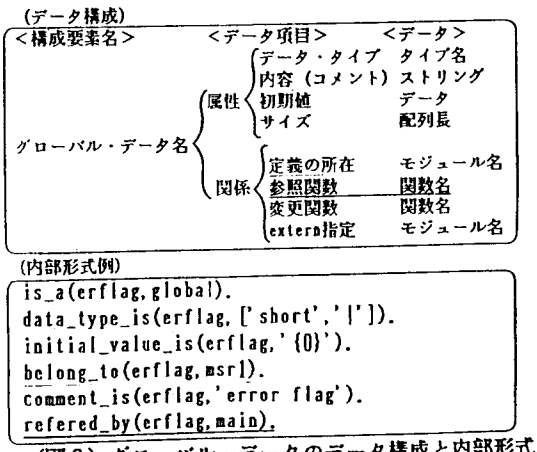


図2－2－7



(図8) グローバル・データのデータ構成と内部形式

図2－2－8

ディタ」なるツールを用いて，モジュール仕様書を書く．プログラマは「Cプログラム･ソース･フレーム･エディタ」なるツールを用いてCソースを書く．仕様情報を抽出するツールを用いて，フォーマットに依存しない形にして統合化データベースに放り込んでやると，ここで一元管理が出来る．この様にする為には，ユーザーが勝手なフォーマットで詳細設計書とかソースプログラムを書いてもらっては困るので，フォーマットを限定する．一旦統合化データベースに放り込まれた物は，各々の逆生成ツールを用いてモジュール仕様なり，ソースに戻す事も出来る．これをやると，どんな良い事があるか．

2.3.8.統合された環境上のプログラミング

モジュール仕様書から抽出ツールを使って統合化データ･ベースに放り込み，さらに逆生成ツールを使ってソース･プログラムを逆生成させると，そこには詳細設計レベルで確定した部分だけが出て来て，コードの部分や細かい部分は空白のまま出て来る．そこで，これをソース･フレーム･エディタに掛けて空欄部分をどんどん埋めて行けば良い．こうすると記述の繰り返しの手間が省けるばかりでなく，ソースを修正した場合も自動的に仕様書に反映される．

メリットは，これだけでなく次の様なものが挙げられる．

－情報統合化開発環境の直接効果－

(1) プログラミング作業の効率化

＊エラー発生防止（ソース，詳細設計書の部分合成，変更の自動反映）

＊エラー早期発見（整合性のチェック）

＊プログラム情報参照サービス

(2)管理作業の効率化

＊プロジェクト進捗状況の可視化－データ･ベースの溜り具合いを見れば良い

＊バージョン管理の自動化

－情報統合化開発環境の間接効果－

(1) ソフトウエア再利用の促進

＊汎用部分の切り出しによるソフトウェアの部品化容易

＊柔軟性の高いソフトウェア部品の実現

(2) 機械化可能性の向上

＊フレーム･エディタによる記述作業の機械化

＊他：プロジェクト管理の機械化－製品管理の機械化，単体テストの自動化，等

(3) 分散環境への対応

＊コマンド統一によるユーザ負担軽減

＊統合化データベースによる情報の一元管理

(4) 多様なソフトウェアプロセスへの対応（ＳＰＩＤＥＲモデル）

2.3.9.ソフトウェア開発環境（ＳＰＩＤＥＲモデル）

現在では詳細設計とコーディング／デバッグのフェーズでの環境整備だが，社内各部課（企画，仕様など）から出て来るフォーマットの全く違うドキュメントを，フレーム･エディタと抽出ツール，逆生成ツールから成るフィルタを通してやって，全社の統合化データ･ベースが出来るのではないか．この様なＳＰＩＤＥＲモデルを現在考えている．

我々の研究開発部門は出来て４年位だが，これから実際に使い物になるシステムを作ろう，という段階に達しています．以上です．御意見をお聞かせ下さい．

有難うございました．（拍手）

2.3.10.会場より質問

中野：　何か質問が有りましたら，１つ位お願いします．

落水：　高級言語をひとつ設計して，その言語周りのツールを整えるのと比較をする必要があるのではないか．例えばＡｄａが持っている機能はどうか．詳細設計を書くから悪いのではないか，あるいはＣで書くから悪いのではないかという気もするが．

西岡：　オブジェクトの段階になるとコメントなど消えてしまうが，我々はコメントが割と重要だと考えます．詳細設計書までブレークダウンしてコーダーに渡すというサイクルなので詳細設計書は大きなプロジェクトになるほど必要．基本設計から急にコードにするより，詳細設計書があった方が人間にとっては分かり易い．

落水：　という事になると，「フレーム」の内容をもう少し説明して頂きたい．

西岡：　これは詳細設計書から逆生成されたソース･プログラムの部分ですが，(図２－２－９参照)このフレーム･エディタでは関数名，関数の機能，パラメータの説明，コメント等を書く場所が自動的に決められる．この様にフレームというのはコーディング･スタイルの枠組みを規定します．

歌代：　詳細仕様書の部分からインターフェース部分を抜き出して，ソース･プログラムの雛形を作る，という様な事を仰っていたと思うのですが，それで作ったソース･プログラムの枠組みからコーディングを始める位だったら，詳細仕様書の１番最後に，コーディングという欄を設ければ，プログラムを最後まで自動生成できてしまうのではないか．例えばコーディングの様式についても，プログラマにＣのプログラムを書かせるから，つまりどんなプログラムを書いてもコンパイル出来てしまうからいけない訳で，仕様書のレベルだったら，ある程度決められたフォーマットを定義して，それから最終的なＣのフォーマットに落とすとか…そう言う事も可能じゃないかと思うのですが．

西岡：　確かに，今までのアプローチとしては，ソース･プログラムの中に全ての情報を放り込んでしまうという考え方だった．どうしてそれが旨く行かないかと言うと，ただでさえ膨大なソース･プログラムが，いよいよ膨大になってしまうから．ある意味で統合化データ･ベースは全ての情報を持っている訳ですから，直接統合化データ･ベースにいろんな情報を放り込んで行けば，それで出来て仕舞うという事にもなるんですが，そうすると，多分プログラム開発担当者は「とてもやり切れない」という事になる．人間の為には，やはり階層化した仕様－設計－詳細設計－コーディングという流れが分かりやすい．だから，詳細設計というフェーズも大事にしようと考えています．

中野：　よろしいでしょうか．セッションのテーマと段々離れて行くような気がしますので，(笑い) ここら辺で休憩にしたいと思います．

(休憩)

逆生成されたソース･プログラム

```
/*>>>  LANGUAGE C FUNCTION  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
>
>     > checkd <
>           — input parameter check routine —
>
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                /*                                  *
                 * ***  INFORMATION  ***  *
                 *                                  *
/*>>> faculties of this function are ...
    1.        check ASCII parameter or not
    2.        erflag on if input data are illegal
    3.        [redacted]
              /*                                    *
               * ***  TYPE & PARAMETERS  ***  *
               *                                    *
/*>>> type of this function is ...
     (type)    (name)  /* parameters */
      char    *checkd (p,[redacted])
/*>>> parameters are ...
(type) (parameters)          /* comment */
char    *p ;          /* input parameter string pointer
                       */
/* >>>>>>>>>>>>>>>  "checkd" VARIABLE DEFINITION <<<<<<<<<<<<<< */
{/*>>> return_value(p) is ...
      parameter "p" is used as return value.  <<<*/
/* >>>>>>>>>>>>>>>>  "checkd" EXECUTIONAL TEXT  <<<<<<<<<<<<<<<<
          [redacted]
          return(p);
} /* >>>>>>>>>>>>>>>>>>>  "checkd" TEXT END  <<<<<<<<<<<<<<<<<<
```

図２－２－９

3.討論

中野：　分散環境とワークステーションというのは，ここに書いてある様に１番上の藤野さんの大好きな話（分散ＯＳ）から，要らないワークステーションをどうしましょうという話まで有ります．非常に困って仕舞いまして，どんな話が出ても私は構わないと思っていますが，話がこんがらかって来ましたらここに書いて有ります様にデフォルトが切ってあって，深瀬さんがバカヤローと言って，その議論は止めると‥(爆笑)．それが深瀬さんのプログラム委員としての役目だと聞いています．深瀬さんが言うと終わって仕舞いますが，私が言うとアホかいなと言う程度で又話が続いて仕舞いますので困ったなと思っています．それで発散しても仕様がないのでチェアマンがこの２つ(分散環境は良いのか？，ＷＳの正しい使い方) を中心にや

りたいと思っています．取り合えずプレゼンテータに対する質問から．

中野： 先ず私から加藤さんに質問．毎日ＪＵＮＥＴの管理にどれ位の時間を使われていますか．

加藤： もし東工大がＪＵＮＥＴの末端であれば３分とか１分で良いと思いますが，割とまん中の方に居る．東工大は Mail が通るから fail し易いとは言わせたくないので(笑い)－それは屈辱ですからね－ある程度の時間，１日に２～３回見るだけで，ほんとは良い筈です．今，もう少し手を抜こうといじってますので，その分時間を取られてます．１時間に３０秒位，ちらっと気を付けてやれば普通の場合は大丈夫です．只ディスクがクラッシュしたりすると２～３日掛かったりしますけど．

堀川： 我々の所でＷＳを使って分散か協調か分かりませんが，兎に角バス形のＬＡＮに繋いでＴＣＰ／ＩＰで何かやって行こうと，今やり始めている所です．２０台位ＷＳを入れて使って居ますが，問題なのは UNIX 系のＷＳですとパソコンじゃなくて大型の計算機に近い面を持っている．例えば電源切ったら止まるもんじゃあない，要するにシャットダウンしなければいけない．そういうのを２０何台に対して，いったい誰がするのかという問題が有って，結局使う人がやらないといけない．それからパスワードの話でも全部一緒にして置かないとダメなんですね．隣の所に持って行くと，誰の持ち物でもないようなファイルが出来て仕舞うとか，それからもしＮＦＳの話になって行くとしたら当然パスワードも一緒にして置かないといけないでしょうし，２０何台も，或は５０台位になった段階では，パニックに成ってしまう－マシン管理とかＯＳの版管理とか－．先ほど「能力のない者は使うな」という話がありましたが，それは「マシンを御守できない人は使うな」という意味に解ってよろしいのでしょうか．

加藤： 確かに問題です．ＷＳを並べると言っても色んな意味合があって，僕が思っている良い並べ方は，大きいＷＳというのがあって小さいＷＳがいっぱいある－こういう並べ方が良いと思って居ます．つまり１台のファイル･サーバにＮＦＳとかのメカニズムに依存してしまう．そうするとｒｅｖｉｓｉｏｎ管理の問題，例えばここのＧＮＵは１８．４１だけど，こっちは４７といった問題は減ります．サーバだけにコピーしておけば良いから．もう１つは，パスワードはＹＰを使えば良い．シャットダウンの問題に関しては，サーバだけ気を付ければクライアントは多分(ディスクがなければ)いつ電源を落としても，その人が若干ファイルが死ぬ程度の被害で済む筈です．２００万円位の同じ様なＷＳがズラリ並んでいる有る会社では，あのソースは何処だって言うとＮ１３の誰のフォームの下だと（笑い）…笑うから何処の会社かバレてしまうんですけど，それは多分良くない方法でして（笑い）グループ毎にＳｕｎを１個買ってＥＡＧＬＥをボンボンとくっつけて置くという風にすると，大分手を抜ける面が有ります．ユーザに徹している人はクライアントを使えば良いし，パワーのある人はサーバの御守を少しやって頂ければ良い…こう言うことが多分いいと思います．

落水： 加藤さんと西岡さんに質問．先ず加藤さんに．私はＪＵＮＥＴではＥ－Ｍａｉｌを出したり貰ったりして居る位ですが，そういうのを維持したりｆａｃｉｌｉｔｙを高めたりする為に何か陰のグループが居ると思うのですが，どういう所に，そういう拠点が有るのか，ちょっと教えて頂きたい．

加藤： 裏の組織としては日本ＵＮＩＸユーザ会のシンポジウムの会場の脇とか，そういう所で情報交換されています．只最近はそういう所でやると僕らの話しか伝わらないし，Ｎｅｔｗｏｒｋで話をするともっと他の人にも聞いて貰えるという事も有って，ガイド的なプロジェクトを楠本さんにやって頂いてますし，静岡大学に関しては多分大木さんが全部やって頂いてますので大変ｗｅｌｌ－ｍａｎａｇｅｄな環境にあるんじゃないかと思っています．

落水： では，その背景をお聞きした上で，先程こちらから情報が出て行かないという話が有りましたが，結局言語の問題がかなり有る訳で，例えば皆が日本語でドンドン書いてゲートウェイの所で翻訳されてピョッと向こうに出て行くというのを－そんなサービスは無用という意見もあると思いますが－事の是非はさておき，そういう事をやる人ってのはどういう風な…結局ボランティアですか？そういう人達が居る訳ですか？

加藤： 現在翻訳をやろうという働きかけは有りません．ネットワークに興味がある人と，翻訳に興味がある人の交わりというのは死語に近い(笑い)．ですから本質的に両方のパワーが非常に少ない．それからボランティアの件は今後非常に大きな社会問題となる事が予想されまして，ネットワークを管理しているが為にＰｈ．Ｄ．が取れない学生がいっぱい出て来ると困るので(笑い)本来であれば×××学会とか，そういう所が音頭を取って徐々に移行して頂けると嬉しい．只日本の現在の電気通信事業法では

専ら金を取ってＪＵＮＥＴをやるという組織は許されないので，その辺の法律的問題と社会的問題と政治的問題が絡み合って来て，今後どうしたら良いのかなーという所で…．学生がやっていると，そういう社会的な問題は解決しないので落水先生の様な理解者が居ると….

落水：　どこそこの学会が主導権を取るという形ではなくて，そういう問題の中で技術的に整理すると色々出て来ると思うんですよ．中にはそれでＰｈ．Ｄ．を取れる位のテーマも出て来るかもしれない，その辺を皆でやれる様な雰囲気作りの方が大事なんじゃないかと思うんですが．どこそこの学会に任せると言うよりは….

加藤：　勿論そういう事でして，そういうａｃｔｉｖｉｔｙがネットワークの上で広がりつつ有ります．まだアカデミックなペーパーが出る様な話は少ないが，「うちではこうやっている」という話は，情報交換としては大きい．自分の所でいちいち全部の方式をやってみて是非を問うという事でなくて，Ｎｅｔｗｏｒｋの上で情報交換をするという気運が高まっていて，それに関するニュース･グループは非常にアクティブになっている．ですから，皆に読んで頂くというのが，そういう人の負担を軽減するのじゃないかと期待している．

落水：　西岡さんに質問ですが，１０万ステップのプログラムを１０人に各々ＷＳを１台ずつ与えてという作業環境で，それが旨く行くコツ，ポイントを教えて頂きたいのですが．どんな問題点が有るか，という事でも結構です．

西岡：　１番問題なのは，そのプロジェクトで共通のインクルード･ファイル等を何処に置くかという問題で，各々自分の所へコピーして使っているといつか又変える事になる．コンパイルしてリンクした時にインクルード･ファイルが違っているという問題がきっと出て来る．１０万ステップを１０人で１０個のＷＳという状態は，社内ではここ２～３年は起きないと思うので深く検討はしていないのですが，今後問題になるのでどうしようかと…ひとつお教え頂きたいと思います．

落水：　そういうのはシュミレーション等してもらうとか…．実際にそこで起こった問題点を整理して頂いたら，その辺どうやったら良いかを皆で考える事が出来ると思うのですが．

中野：　落水先生が喋るとすぐ論文のネタになってしまいますので‥（笑い）

臼井：　先程青木さんなんかのデモを見てまして，Ｓｍａｌｌｔａｌｋの非常に良い環境を使ってＵＮＩＸの使い易い環境を作ろうとか，ソフトウェア･データベースを作ろうとの話があったのですが，門外漢から見てそれよりもっと大変だなと思うのは，ネットワークを繋いで廻ったり管理したりする事だろうと思います．ネットワークというのは「関係」を示す訳ですから，関係図が画面に書いてあってマウスでチョンとやればネットワークが繋がる様に，なぜならないのかというのが私の疑問です．

勿論ハード的に繋がないといけない物については人間が出て行って繋ぐのですが，一旦構築されたネットワークのシステムを変えるというのは，どちらかと言うと，メカ的な話だと思うんですね．先程のソフトウェア･データベースなんかは，ヒューマンな要素があって非常に大変だと思いますが，ネットワークはプロトコルがどうこうという問題も有りますが，何とかすれば何とかなる世界だと思うんです．ネームサーバの管理システムみたいなのが有って，マウスでチョンチョンとやれば直ぐプロトコルを合わせて繋げてくれる様な環境は将来技術的に可能なのかどうかが１点と，そういう事をするのが意味があるかという事をお聞きしたいのですが．どなたでも結構です．

深瀬：　今の話はＵＮＩＸマシンだけでネットワークを構成していると難しいと思います．物理的なネットワークのトポロジーと論理的なトポロジーを別に考えて，ダイナミックに動かしたいというお話だと思いますが，ネットワークとネットワークを繋ぐ箱が，ＵＮＩＸマシンである場合もありますしゲートウェイ･サーバであるブラックボックスみたいな仕掛の場合もありますが，ブラックボックスを使っている場合は，それが可能です．というのは，ゲートウェイ･サーバ自体はＴＣＰ／ＩＰ以外のプロトコルも持っていて，トラフィックのモニタリングとかネットワークのサーバとしてのネーム･データ･ベースの交換とかいう事をやって居るのです．ネットワーク上でそのゲートウェイ･サーバのスーパー･ユーザとして，そのジェネレーションを変えてやれば何処からでもダイナミックに変えられる様な仕掛になって居ます．ですからＵＮＩＸマシンだけでＮｅｔｗｏｒｋを構成すると，それは難しいだろうと思います．ユーザ･インタフェースとしてマウスを使ってやるというだけで，技術的には今も可能な話です．

酒匂：　臼井さんのご質問には深瀬さんが的確な答をして下さいましたので，特に追加する事はないのですが，要するに今まではＵＮＩＸはプロセスの壁が高いからダメよという事でしたが，今やホストの壁が高くてダメだと言われ始めている訳です．ディスプレイだけ持ってきて付け

たいと思う時が有りますが，それが出来ない．何故かと言うと，ディスプレイとかキーボードとかワンセットになっていて，全部含めた形でしか移動出来ない．ネットワークにワインドして居るからです．理想を言えば，コンセントにディスクを差し込んで通常はディスプレイとセットで使っている，ディスプレイがもう１個必要になったら横にくっつける，キーボードが要らなくなったら外して使えるという感じになれば良いと思います．スティーブン·ジョブスがＮＥＸＴとかいうコンピュータを設計しているそうですが，噂に依るとそれに近い世界が展開しているとの事です．

加藤： 別ネタで質問です．ＷＳ＋ネットワークの環境で会社でひとつのプロジェクトを走らせる場合，仕様が決まったものは割と楽だと思いますが，「こういうツールを作って見よう」というとやはり仕様は常にフィードバックして変わって行く．プログラム·テキストだけ考えても，共通であるべきファイルがどんどん書き変えられて行くと，非常に困ってしまう人も出て来る訳ですね．そういうネットワーク＋ＷＳで，使っているユーザが隣の机に居れば良いが，隣の部屋とか隣の町とか隣の国に居た場合に，電話を使わずにＮｅｔｗｏｒｋだけでプロジェクト管理が旨く出来るのかなーという疑問が有りまして，その辺を特に企業に居る方にお聞きしたいのですが．

中野： 分散環境の共同作業と言う所で，名前が５人並んで居ます．まず岸田さんかな…．

岸田： 僕はアンケートには，質問を書いただけで回答は書いてない(笑い)‥．理想の答は見つかっていないと思いますが，どうしたら良いかという試みは色々されている．スペックが固まらない時点で，１人でなく大勢で考える場合にどうしたら良いかという話ですね．ソフトウェア·データ·ベースとも絡む訳だけれど，ネットワーク·ワイドに跨ったフワフワしたデータ·ベースみたいな物が有って，そこに皆が情報を追加したり検索したりして仕事をして行く，という仕掛を作れば良い．

最近私が仕入れたネタは，アメリカのＭＣＣという所で最初ｇｎｅｗｓというツールを作った．これは普通の電子掲示板のＮＥＷＳは見にくいので，ネットワークをグラフィックにモデル化して，此のＮＥＷＳと此のＮＥＷＳは関係がある，それに対して誰がＲｅｓｐｏｎｄしたという風な格好で，Ｂｒｏｗｓｅｒが蜘蛛の巣の様になっていて，そこをクリックすると見えるという物．

それをさらに発展させたｇｉｂｉｓという物がある．これは元々Ｉｓｓｕｅ－Ｂａｓｅｄ　Ｄｅｓｉｇｎという設計の方法論が有って，設計するという事は，何等かの色んな問題点を解決して行く過程との考えに根ざしている．何か，あるソフトを作ろうとした場合に，問題ａ，ｂ，ｃとあって，問題ａを誰かが検討したらそれは問題ａ－１，ａ－２，ａ－３に分かれた．ａ－１に関して解決方法はこういう方法が有る…と．そういう格好で，ソフトウェアの設計に関する問題のツリー構造は最初からある訳ではなくて，皆が討論しながら段々出来上がって行く，そういうのをネットワーク上でサポートして，一種の設計チーム用特殊電子掲示板にスペックを貯め込んで行く．最後にドキュメントを作るなら作れば良い，という風な話が有ります．

それに似た仕掛は，各々のプロジェクトに応じて割と簡単にセットアップ出来ると思いますが如何な物でしょうか．

青木： 私どもの会社でもＮｅｔｗｏｒｋがビシッと引かれていまして，メールによってプログラムを作る事がしばしば有ります．Ｎｅｔｗｏｒｋがないとプログラムは作れない．メールで色々なディスカッションがされて，製品の仕様が決まります．仕様書として出来上がった段階は面白いものではなくて，寧ろメールで意見交換された過程が非常に重要だと思います．仕様書は１つの結果であり，重要ではない．ネットワークに残るログが大事．これを何とか管理出来る様な機構，例えば今岸田先生が言われた様なツールを，私のＳｍａｌｌｔａｌｋの環境で１週間位(？)で作って見ようと思います．

野中： 私は，マイクロ·プロセッサが乗った装置へＲＯＭを焼いてソフトを組み込む様な仕事をしていますが，現在の開発環境はどうかというと，計算機間はネットワークで結ばれていてもターゲット装置上ではＩＣＥというＩｎ－Ｃｉｒｃｕｉｔ　Ｅｍｕｌａｔｏｒを使わなければいけない．これはターゲットのＣＰＵを引き抜きＣＰＵのＥｍｕｌａｔｉｏｎをする様な装置を差し込んで，それでトレースしたりブレーク·ポイントを設定して変数値を見たりする物です．ところが，此の部分が完全にチョン切れて居て，パソコン上でデバッグをやると結局ＲＯＭに焼いてしまうのでバイナリの情報しか渡らない，最終的には１６進で何番地という形で見なければいけない．この部分が１番繋がって欲しいと思う．世の中には色々なＩＣＥが有るが，Ｃのソース·レベルのデバッグが出来るなどと言ってもメーカ毎に全く違うとか，Ｗｈｉｔｅ

Ｓｍｉｔｈでなければ駄目だとか，何々メーカのシンボル･リストを出すコンパイラでなければ駄目だとか，せいぜいフロッピーのフォーマットが合っているから渡せるという程度です．　ＵＮＩＸのＷＳ間ならＥｔｈｅｒｎｅｔとＴＣＰ／ＩＰで繋がる様になっているが，それがもっと外の世界に広がって，究極は洗濯機や掃除機にＥｔｈｅｒｎｅｔのボードがあるとか(笑い)Ｃ－ＴＲＯＮみたいな話になるのかも知れない．今は切実な問題としてＩＣＥが完全に離れている事が１番困っています．

中野：最後の洗濯機の話は兎も角，これはＩＣＥの話で，ＳＲＡにそういうツールが有りますね？それと松下電器の高野さんの所でやっている．ＩＣＥに関して言えば，今までＲＳ－２３２Ｃの１０Ｋｂｐｓでやっていたのが１０メガｂｐｓになる，それが効率が上がったという事で，その他の話は分散環境の問題ではなく，プログラム開発環境上でソース･デバッグをどのレベルで何処までやるか，という問題だと思いますが…どなたか．

加藤：それは，ＩＣＥにＥｔｈｅｒｎｅｔが刺さっていて，そこへ，ドライブ用のしかるべきプロトコルが定義されていれば，こっち側から適当に送ってやれる‥そういう装置があればいいなあ，という話ですか？

熊谷：ＨＰ社にＬＡＮで繋がるＩＣＥがありますよ．それを買うと解決するんじゃないですか？(笑い)現在うちでは，ＩＣＥを繋いで机の上から６８０００でも８６でもコンパイラを動かしてダウン･ロードしてデバッグをやっています．それは悲観しなくても，世の中ちゃんと出来ています．

野中：僕の希望としては，早く業界標準が出てくれればいいなあと…．有るという事は判ったが，まだバラバラの状態で．ΣはＧＰ－ＩＢを使うと言って居るが，兎に角何でも繋がった方がよい．その標準が出来てくれればいいという希望です．

鶴見：私の会社ではワープロ，表計算，販売管理等パッケージ･ソフトを作っています．ＷＳはあまりなくて分散環境と言えないが．先程加藤さんからコミュニケーションの話がありましたが，我々がパッケージ等を作っている時，電子メールでやり取りしているセクションもあるが，我々のパソコン･プロダクツで１番大切なのは，画面の綺麗さとかファンクションキーをどう押せばユーザが使い易いかとか，結果で見ている所がある．分散と言うか離れてのコミュニケーションは，当初は良いのだが，製品の仕上がり段階ではそれが出来ない．製品が仕上がって来ると徹夜でそれを見直して，またやる．

コミュニケーションの話でもう１つ．在宅勤務というのが一時話題になったが結局旨く行かなかった．在宅勤務だと大切なコンセプトが伝わらない．断片的にしか伝わらないので，結局時間が経つと違っていたという様な事がある．それは在宅勤務に限らず，お客でもそうだが．プログラム自身にはバグが付き物ですが，コンセプトをどうやって出すかというのが，プロジェクトをやる時に苦労をして居る所です．

歌代：ちょっと席を外していたので，分かりませんが…．分散環境に問題が起きていてどうしようという話の様ですが．

１０人で１０台のＷＳを使っていたら，ファイルのConsistency はどうするかとの問題は確かに UNIX で商売をしている人達の間では盛り上がっている分野です．しかしそれはＮＦＳやＲＦＳを使えば解決してしまう問題ですし，パスワードにしても加藤さんが言った様にＹＰを使えば何の問題も起きない．そういう問題は，ここ５年位で安いＷＳが出て来たが，それを売っている人達は既に考えて居た事で，その技術は確立されている（良い悪いは別にして）．地理的に分散した環境でプロジェクトを運用する場合の，マネジメントの問題を話題にして居るのですか？

加藤：共通ファイルが書き変えられた事を知らずに他の人が使っている…普通はこういう事が起こる（積極的に全てをｌｓするとかｍａｋｅにガリガリ書くということをしない限り）．「こういう風に変わりました」という事を何とかツールを使って旨くやるのが良いのか，ｎｏｔｅｓのような別の形態が良いのか，その辺を聞きたいのです．

歌代：そういう問題は分散環境とかネットワークには関係ない．Ｓｍａｌｌｔａｌｋの様なインタープリタ型で，今ある状態が全てというシステムでは問題ないが，ＵＮＩＸの様なコンパイラ指向ＯＳは全て持っている問題．ＵＮＩＸを使っていると，同じマシンを使っていても誰かがＩｎｃｌｕｄｅ　Ｆｉｌｅを変えてしまったら，古いＩｎｃｌｕｄｅ　Ｆｉｌｅでコンパイルしたプログラムは全部今のシステムの状態と食い違ってしまう．だから，それはネットワークとはあまり関係ないという気がします．そういう OS は早く無くなった方が良い．

地理的に分散したプロジェクトの管理は，多分アメリカの方が進んでいると思います．私は以前Ｓｕｎに居た事

があるのですが，Ｓｕｎの環境はＷＳを使った分散環境としては１番進んでいる．しかし彼らは，いつもＷＳに向かって仕事をしている訳ではなくて，少なくとも１週間に１回はミーティングをします．有名な人はアドバイザーの様な立場で引っ張りだこで，その様な人が異なる場所から会議に参画する時は，机のまん中にスピーカ付きの会議用電話を置き，その人の声がスピーカから聞こえる，という形で会議をしている．現状のワークステーション程度の分散環境は，コミュニケーションの手段としては音声にかなわないと思う．テレビとかはＳｕｎでは有りませんでした，五角形の建物では知りませんが…(笑い)．今まで聞いていた話は，あまり大した問題ではないと思いますが．

西岡： Ｃｏｎｓｉｓｔｅｎｃｙの問題ですが，ＮＦＳ，ＲＦＳではよそのＷＳのディレクトリーが，あたかも自分のシステムに付いている様にアクセスする事が出来る訳ですから，Ｃｏｎｓｉｓｔｅｎｃｙは問題ではない．ところが問題なのは，今のソフトウェアーは幾つかにモジュール化してしまったら，モジュール間のＣｏｎｓｉｓｔｅｎｃｙすら旨く調整出来ないというのが実状です．それを何とかする為に例えばソフトウェア・データ・ベースで情報を一元管理して，Ｃｏｎｓｉｓｔｅｎｃｙを旨く取ってやる．ここでデータ・ベースの問題が出てくる．そのデータ・ベースを例えば５０個のＷＳのどれか１つのホストに置いて，みんながワーっと来てやる．そこでＣｏｎｓｉｓｔｅｎｃｙ Ｃｈｅｃｋが頻繁に起こる様なシステムを構築した場合，ネットワークがそれに耐えられるか，という問題があるのですが…耐えられるのですか？

深瀬： ネットワークを使わなくてもそんな事は出来ます．世代管理のソフトウェア・ツールという物が有って，それの最も弱い所はモジュール間の整合性をどうするかという点で，そこまで踏み込んで世代管理をしようというシステムが今実験的に幾つも有る訳です．それを使えば何でもない事で，取り立てて問題にする事ではないと思いますが．

西岡： もし分散環境を使ったとした場合は，という前提がある訳です．分散環境を使わなくてもいいじゃないかと言うと，これからはそうは行かないのではという気がします．

深瀬： それはトポロジーとしては分散しているが，プロジェクトの１つな訳です．たまたまソースがあちこちに分散していたと言うだけで，Ｏｒｇａｎｉｚａｔｉｏｎをどう組むかという問題ですから，分散環境とは関係のない話だと思います．要するにツールがネットワークを意識して作られているかという問題で，もしそうでなければ当然そういう問題は出てきますが，今我々が手に入れられるツールはネットワークを前提としていますので….

西岡： そのツールは，例えばＣｏｎｓｉｓｔｅｎｃｙを保存する為にデータ・ベースを置くのかどうか分かりませんが，そういったデータ・ベースはＷＳ毎に置くのではなくて，何処か１箇所に置くという発想なのですか？

坂下： あの，ずっと疑問だったのですが，どうして分散しなければいけないんですか？何故ネットワークを入れなければいけないんですか？単なる流行で入れるのですか？おかしいと思うのですが…．でっかいＶＡＸを入れれば良いと思うのですが．ネットワークとかＬＡＮで解決する問題ではないですよね．どうしてＷＳ，ネットワークを入れようと思われたのですか？その辺をお聞きしたい．

酒匂： 確かに流行でネットワークを張っている所も有りますが，広い画面を使って使い易いインタフェイスを使おうと思ったら，何故か黄色いケーブルを引きずって来てしまった，さてどうしようというのが私の感想です．ＷＳにはインタフェイスだけやらせて置けば良いと言うか，それを付けたらディスクとケーブルという重たい物を引きずって来てしまったので，問題が複雑になった．先程のＣｏｎｓｉｓｔｅｎｃｙの問題ですが，色々なＷＳに無理矢理ファイルをバラ蒔いて皆が勝手に行くとすれば問題が起きるのは当然で，それは１つのホストでも同じです．最初はＲＣＳというソース単位の世代管理方法があって，これは縦の糸ですけれど横の糸を何とかしようという事でＧＲＩＤというシステムが出来た．複数の人間のプロジェクト管理は色々と研究が進んでいる．

ＮＦＳ，ＲＦＳが良いとかファイルのコピーはどうするとかの話は歌代君が言ったようにどうでも良い事であって，このパネルで話し合うべき事は－さっきからずっと言いたかったのですが(笑い)－ソフトウェア・データ・ベースの続きの話みたいな，更に上位の概念が大事だと思います．

歌代： 分散環境という言葉を皆さん勘違いしている方が多いと思います．何故ネットワークを使うかという話をまずします．分散したいからやっている訳ではなくて，たまたま１００倍速いＶＡＸを作るよりは値段が１００分の１のＶＡＸを作るほうが簡単だったから１００個出来て仕舞ったという事．今までは１個のＶＡＸのもとに皆

が共通の環境で，ＵＮＩＸって便利だな，嬉しいなと思っていたのが，別々のマシンになって人は何をやって居るのか分からなくなって仕舞った．そこで取り合えずＥｔｈｅｒｎｅｔという結構速いＮｅｔｗｏｒｋで繋いだが，しかしそれは今までＲＳ－２３２Ｃで繋がっていたのが単に速くなったという環境に過ぎない．当然そこでＵＮＩＸマシンを売る事を生業としている人達はＮＦＳとかＲＦＳ等の技術を一生懸命考える訳です．ＳｕｎのＮＦＳとかＯＮＣとか，Ａｐｏｌｌｏのネットワーク･コンピュティング･システムとかフォーラムとか…．これが現時点での分散環境と言われる物だと思います．

そう考えると分散環境とは，実は分散させる為の技術ではなくて分散してしまったものを如何に集中させるかという技術なんです．ＮＦＳと言うと「分散している」という雰囲気が有るが，実は１個のファイル･システムを皆で使う技術であって，決して「分散ファイル･システム」ではなくて「集中ファイル･システム」なんです．

もう少し新しい技術として分散ＯＳというのが有りますが，あれはタイトに結合したネットワークを使った複数ＣＰＵのシステムで，その中のＣＰＵ同志は結合が強く，１０個のＣＰＵの内１個が壊れてしまっても，トータルな能力が１０分の９になるだけで何の障害もなく仕事が出来るという技術です．ファイルにしても何処に有るかは分からないが，全体が１つのコンピュータとして機能していて，何処かのディスクが壊れてしまってもファイルは絶対無くならないという様に，分散したものから如何に１つのグローバルな空間を作り出すかという事です．この様に分散から如何に集中したものを生み出すかという技術だと思います．分散と言うと何かをバラ蒔くという風に考えますが，そうではないという事を私は強く主張したいと思います．(拍手)

中野：ここで手を叩かれると結論になってしまいますので(笑い)…．

小池：我々の会社は元々メインフレームのパッケージを扱っている会社です．今ディーンというモデルを商品化しようとしていますが，これはパソコン等の下位レベルに処理を集めファイルのＩ／ＯやＣｏｎｓｉｓｔｅｎｃｙのチェック等はファイル･サーバにやらせてしまおうという物．分散と言うと開発と実行とに分かれると思うが，ファイルが何処に有るかという問題はどちらかと言うと，開発している時の問題と思う．逆に実行の段階だと完全に分散させてパソコン側に負荷を掛ける方が良いのじゃないか．メインフレームではオンライン･モニタとか色々あって，パソコン側に落として仕舞えばオンラインを意識せずに可搬性が良くなる．開発に於ての分散と，処理に於ての分散とは環境が違うし，コンピュータの適材適所な配置が必要と思って居ます．

盛田：先程話そうと思ったら歌代さんが随分話されたので引っ込んでしまったのですが．分散環境の名の下に「問題」を分散してるんじゃないか．「集中」という言葉が殆ど出て来ないが，私は集中も必要だと思っている．ほんとうに分散システムで解決しなければいけない問題は一体何かを考えたい．

もう１つ，組織の中の開発環境を考えるとき，分散システムは組織を支援すると言う意味で重要な役割を持っていると思う．例えばトップは紙と鉛筆と電話があれば良い，プログラマーはメインフレームの端末でゴリゴリやっている，デザイナーはワープロや J-star を使っているという様に組織上のいろんな階層は住んでいる世界が違う．これを何とか繋いで，組織の問題解決に役立てたいと思って居るが，御意見を聞かせて下さい．

中野：メインフレームを含めた分散環境の話が出ると，一部シラケるんじゃないかと恐かったのですが，是非やりたいと思います．「あんなのは過去の遺物だ」という様な言い方でも結構です．

酒匂：あんなのは過去の遺物だとは申しません．社会の大部分，金の大部分がメインフレームに注ぎ込まれている以上捨て去るわけには行かない．分散環境とは何ぞやというこのパネルで，一体何を話して居るのか先程からよく分からないのですが(笑い)…．アーキテクチャーの話なのか，マイコンとメインの負荷分散なのか，そこへ来て盛田さんが文化の違う人達をどうやって１つの器に入れるかみたいな事を仰る．ソフトウェア開発という抽象的な話になれば，如何に人の和を崩さずに皆を助けてあげられるか，というのが分散環境又はコンピュータがしてあげられる最大の事です…．全然まとまっていませんね．(笑い)

深瀬：メインフレームで仕事をして居る人は必ずＭＭＬの事を持ち出す．あれは，結局垂直分散してホストの負荷を下げ，周辺型端末の仕事をパソコンにやらせようという事．ところが UNIX での分散は横への分散で，皆で共同で仕事をして１つの格好にしようと言う物．その違いはかなり大きなものがありますが，最近はＩＢＭもそこに気が付いて，例えばシルバー･レイクという新しいモデルは

Ｅｔｈｅｒｎｅｔのコントローラを持っていて，水平分散も出来ます．シェラも８２５０というＥｔｈｅｒｎｅｔにチャネル直結する装置を用意している．我々は単に負荷分散の為だけに垂直分散をするのではなく，幾つか特異なサーバを集めてシステムを作り上げようというコンセプトで分散環境を考えているのだから，その視点で，メインフレームで仕事をされている方々もこれからどしどし仕事をして頂きたいと思います．

中野： 分散の話になると，メインフレームの方々からは意見が出ないのが普通ですが，ここで忌憚のない話を聞かせて頂きたい．日本電子計算の井川さんにお願いします．臼井さんに聞いても分かりませんので（爆笑）．

井川： そうです．臼井さんとか，当社でＳＥＡの中で有名な人達は，計算機をブンブン動かしてお金を儲けるという事には関係のない人達ですので．当社はホストを回して，１秒幾らで商売をしている会社でして，その時々で最高の大型，超大型を幾つか買って来たのですが，商売をするには足りない．支店等のノードにホストをバラ蒔いてどんどん使わせる‥そのうちにそれでは不便だという事で，いわゆるホスト間のネットワークを引いた．ここ５，６年そこから一歩も前へ進んでいない．そこでＵＮＩＸを手に入れ，ホストのネットワークに重ねてホストの人達が抱えている問題を軽減しようと言うアプローチをしています．

当社では日立，バロースという，今話題にしているＮｅｔｗｏｒｋとは関係のないＮｅｔｗｏｒｋを持っている機種を保有しているので，ＴＳＳ，ＲＪＥという過去の物で仕事をしている．しかも端末が高い上に，端末を増やすとドンドンＣＰＵのパワーが落ちて行く．我々は機械を回して商売をしている訳だから，端末を増やしてＣＰＵの負荷を上げてしまったら商売にならない．そこでＷＳをバラ蒔いてＮｅｔｗｏｒｋに繋ぎ，そのＮｅｔｗｏｒｋの一端にホストを接続し，日々の生活はＷＳでやって，最終的な結果だけをホストに送る，という使い方を考えている．５，６年前に話題になっていた事を現実に今やろうとしている．

問題は，ＴＳＳの世界の人達がホストの世界から離れられない．文化が違う，ＴＳＳがないと仕事が出来ない．マルチウィンドがあるじゃないと言っても，３０個位ファンクション･キーがないとこれは端末じゃないという話になってしまう．しょうがないのでＷＳ上に画面の動きをエミュレーション出来る様なものを作って，ホストと通信させる．ホストの周りに住んでいる人達をもう少し普通のと言うかｒｅａｓｏｎａｂｌｅな世界に連れて来ようとしている．こういうアプローチの方法には不満が有るのですが，現実にはこんな方法を取らざるを得ないと思います．

臼井： 同じ会社の人間ですから，その通りです．只何故ＷＳに切り替えないかというと，会社がつぶれるからです．メインフレームで売上を上げて，ＷＳで遊んでいる人間を養っているのが現状です．商売の方向をきちんと定めない限り，解決出来ないだろうと思っています．ＫＣＳさんは如何でしょう．

盛田： 基本的にはそう思います．我々の仕事はメインフレームのビジネス･アプリケーション開発が主ですが，メインフレームが今まで何をサポートして来たかと言うと，プログラミングだけな訳です．つまり端末でソースコードを相手にすると言う所だけ．ところが，ソフトウェアを作るという事は要求分析から始まってドキュメント作成の時間の方が遥かに大きい．全体の３分の１もやっていない様なプログラミングの工程に高いメインフレームを使っている一方で，ミーティングとか SPEC.を決めるといったプロセスには全く金を掛けていない．その辺のロジカルな仕組みとか，情報間のＲｅｌａｔｉｏｎがすっきり割り切れて，旨い切口からアプローチ出来れば良いが，それが侭ならない．その辺にＪ－ｓｔａｒが存在している．

当社ではメインフレームでプログラミングやってＪ－ｓｔａｒでドキュメンテーションをやる，事務所の中はＳｔａｒ系のＥｔｈｅｒｎｅｔとＴＣＰ／ＩＰのＥｔｈｅｒｎｅｔが混在して居て，結合度は弱いのですが両者間でやり取りをして居る．メインフレームとも強くはないがやり取りして居る．こんな状態が暫くは続くんじゃないかと思います．

先程臼井さんが仰って居た様に，稼いでいる部隊は圧倒的にメインフレームなのですから，そういう人達に元気を出してやってもらわないと，それ以外の所も成り立たない．その辺がソフトハウスとして難しいところだと思います．

岡本： 社内では垂直分散なのか水平分散なのか，よく議論しますが，どちらかと言うとホストを中心にした垂直分散の話になり易い．その中で私は小数派として水平分散を意識しながら環境構築を進めている者の１人です．何故水平分散なのかですが，企業の中は，人格の違う人間が集まって仕事をする「単位」(例えばプログラマ，デザイナ，

プロジェクト･マネージャといったグループ）が存在し，更にオフィスの中にはもっと違った立場の人が居て組織体としての活動をしている．ところが，その人達が仕事をする為に使う道具というのは，みんな違う．従って，各々の人にはそれに相応しい環境を提供して行きながら，個人個人又はグループを横に連携させて仕事が出来る様な環境を作って行きたい．分散環境で，しかも横に旨く繋がる様な事を考えて行きたいと思って居ます．

只僕自身「ＷＳの正しい使い方」には悩んで居て，「能力のない者は使うな」と言いたいのですが，私など勝手に好きなＷＳを社内に持込みますので，「人に押し付けておいて旨く使えとは横暴だ」という反発もある．水平分散した中にある，１つのＷＳの正しい使い方について，パネラーの方々のお考えをお聞きしたいと思います．

中野： 私からも答えたい．岡本さんは，もっと外へ出られれば良いと思います．ＪＵＮＥＴも，凄く良いニュースが流れて居て，例えば加藤さんなんかも，年齢は兎も角Ｎｅｔｗｏｒｋでは私よりもの凄く先輩なので，どんどん教わって良い管理者に成りたいと思っている訳で，梁瀬さん（ＫＣＳのＰｏｓｔ　Ｍａｓｔｅｒ）を１人にせずに，そういう人をどんどん増やす方向で努力される，もっと外へ出られるのが良いのじゃないかと思います．

加藤： ＷＳの正しい使い方，というのは良くない言い方で，「好きな様に使えば良い」というのが僕の持論です．ｅｄを使いたい人，ｅｍａｃｓを使いたい人それぞれに使えば良い．ただ，ちょっと手助けしてやるだけで良い環境に移れるという場合，教えてあげる人が居ると良い．

私はメインフレームは速いので使いたい．だけど，あの端末からでなくＷＳにあるプロセスに使わせる，こういうアプローチが良いのではないか．其の為の異機種分散みたいな技術，メインフレーム側からネットワーク的サービスを提供する必要がある．それと，本当に分散でないと出来ない問題を捜さなければいけない．ここに居る方々は，ＷＳとかＮｅｔｗｏｒｋとか分散とか，ラベルで判断しないで中身で勝負をして頂きたい．（笑い，拍手）

酒匂： ＷＳの正しい使い方－好きな様に使う－は加藤さんと同じ意見です．分散環境の許に集まって何の話をしたのか…言葉だけ有って，実は実態がなかったのでは…．此の問題を提示された人と聞いている人の意図が食い違っていた．「分散環境」だけでは問題提起にすらなっていなかったという事が，今日のパネルで良く分かりました．

西岡： 当社ではミニコンやＷＳが漸く繋がったばかりですが，以前はファイルひとつハンドリングするにもＭＴやフロッピーに落し，フォーマットが違っていると言っては大騒ぎしていましたが，Ｎｅｔｗｏｒｋを１本張っただけで何でも送って仕舞う，此は素晴らしい環境だと思います．もう１つ，「統合化」というのが私のテーマだったのですが，統合と分散とは反対の様な気がしますが実は同意語だった．ＷＳ間でＮｅｔｗｏｒｋを張った様な環境は統合化という点から見ると問題がありますが，良いこともあります．それはＷＳ単体で何ＭＩＰＳものパフォーマンスを独占出来るから．しかし，分散環境を開発環境として生かし切っているかと言うと疑問で，分散環境でなければ出来ない環境，例えば生産性などのメリットが見つからない．それを何とか捜し出したいと思います．

中野： このテーマは初めからまとまらないと思って居ましたので，こんな風に終わりたいと思います．燃え尽きない方はＢＦＯの方へどうぞ．

本当はメインフレームを出したく無かったのですが，ＳＥＡ関西では「ゼニだ，ゼニだ」という集団が多くて（爆笑）儲らないとやらないという所があって…．今日集まったのも何かの縁ですし，外へ出て何か言うのも Network 環境の１つ…そういう事でエンディングにしたいと思います．どうもお疲れ様でした．（拍手）

杉田　義明

加藤さん：グローバル・ネットワークを構成する上でのむつかしさがよくわかった．

酒匂：テーマとして、ユーザインターフェイス向きではなかったかなあ．

西岡：これもテーマとしては、ソフトウェア、データベース範囲に入るものと思われる．

統計的開発環境というネーミングをつける場合には、当然プロセスについての情報の取り扱いが含まれていると予想したが、成果物を中心としたオブジェクト管理が中心であるように見覚えられた．その点で少し物足らなかった．

以上すべてに言えることであるが、分散そのものの評論・発表が少ないのは、去年と同様無理なのでしょうか．

三浦あさ子　ＳＲＡ

分散環境は本当に必要かという話があるということがわかった．

桜井　麻里

「統合化」と一口にいっても、それぞれ描いているイメージがちがうのですね．

近藤　康二

現実的な話でわかりやすかった．

junet：ボランティアでどこまでいけるか？

XME：UNIX でどこまでいけるか？

開発環境：実用階段へ行けるかな？（かなり独断と偏見にとんだ話だったなぁ）

新田　SRA

JUNET には本当におせわになっています．再利用と自動化で本当に生産性は上がるか？

林　香

WS + LAN では分散環境ではない．

北野　義明 KCS

分散環境はどのようなもの？という話に対して、現状では、UNIX-WS を中心とした高性能 WS と Network が浮かぶが、ソフトウェアの実行環境も、その WS 上や Network 上にあり、インターフェースもそれた場合と、もともと集中環境であったものを不便を感じて分散に手を差し延べていて、実行環境が別の場合とがあると思う．

前者の発表ばかりで、後者の我々はやはりなかなか良い解決策が見付からないようで頭をかかえ込んでしまいそうです．

堀川　博史

３名のうちのプレゼンテーションとセッションとにずれがあった．

井川　裕基

・VAX よりは WS の方がよい．

・現実に VAX 11/785 上りは NEWS-820 の方が速い．

・WS だと、だれかがドジっても自分が困らない．

小林　貞幸

・セッションテーマとプレゼンターの話が１人しかマッチしていないのでは？

・とは言え、ソフトウェア開発環境の改善という大きなテーマには合っている．

・小生の作業環境から西岡氏のテーマについて詳細討論を望む．

具体的には

１．詳細ドキュメントとプログラムソースの関連性（特に電子的には）はどうか？・ただ単にフレームジェネレーターを提供しているだけなのか．・仕様、変更があった場合の両者の関連は？

２．データベース化されているのか？されているとすれば、その仕組みは？

深瀬　弘恭

加藤さん　センタマシンと WS は両方ともあって一人前だと思います．酒匂さん Window Constraction kit はこれからも色々出て来ますが、Geveval な Window Manager の Body はどんなものが良いのかはっきりしないのが現状ではないかと思います．

久保　宏志

西岡さんの時には、ソフトウェア、データベース構築論になっている．彼のはなしは、ソフトウェア・データベースセッションのあたまにもってくると、discussion はもっと produdive になったのではないか．

中村　真

ネットワークが当然の世界（例えば MACH から）はもうじきやってくるだろう．一方、古いシステム（多分 UNIX の TCP/IP 系もそうなのだがと進んだものとの接続も（捨ててしまえない以上は）大きな問題になりつつある．

熊谷　章

加藤：JUNET の説明「米国とのやりとりは、会社が直接 CSNET に加わった方がよい」とは知らなかった．
酒匂：X-Window の短所を補習する開発環境 XME の説明 UNIX 下の実用的なアプリ作成のインプリの制約と技術がよく分かる．どこでも同じことが起きている．
西岡：何を分散させるかという議論がない．

荻生　準一

西岡さんの意思に賛同．

森　幸一

(3-1)・ 分散と集中の協調が必要．・ローカル処理は WS が有利．
(3-2)
(3-3)・フレームエディタで作成された情報がどれ程汎用性をもたせることが可能か．

歌代　和正

最後の発表
例1：最初の OHP の問題点をネットワークのせいにしたらネットワークはきっとぬれぎぬだと言っておこると思う．
例2：最初の問題から開発環境の統合化を思い付くという発想は偉大だと思う．

田中　正則　ＳＲＡ

WS ＋ ネットワークという環境で資源をいかに有効利用していくかという話をききたかった．

御喜家　貴子

ソフトウェアの生産性向上の定義が再利用性向上である点が面白かった．

佐藤　千明

・水平分散よりも垂直分散の方がいいのは賛成・spiDer：統合化 D/B のモデルはその対象上程毎に別口のモデルなのか．工程間の連結方法が不明．（今は詳細設計→ Code skelefon のみなのでよくわかるが）

田中　一夫

西岡氏 SPIDER は、汎用機・事務処理分野でも充分使える物である．

小池　幸徳

UNIX, MAC を中心とした分散環境もあるが、分散環境というネットワークに接続可能なコンピュータすべてが対象になると思う．そこで、メインフレーム、１６ビットの存在も忘れてはいけないのではとおもいます．

藤野　ＦＸＩＳ

加藤：JUNET の volantea 御苦労さま、本当の分散環境を考えるのは、使い方（文化？文明？）と同時に手探りしていくのは、仕方ないでしょう．
酒匂：IPC は分散への１つの核技術である事は確か．しかし、OS（分散 OS）として提供されるべき機能と、サーバとは区別しておいた方がよいように考える．
西岡：統合化環境としての分散環境という見方で、一寸、変わった見方だと思った．「再利用」とか「共用」と言う分散環境の共同作業の進め方の指針は、確かに重要な Factor かも知れない．

岡本　KCS

・東工大の加藤さん教育的問題については、同感です．また、分散環境が必要かどうか？についてですが、要は、何を分散させるかが問題で、ただ、WS とネットワークでの話にとどまるならば、分散化ではなく、独立化でしかないでしょう．同感です．
・ＳＲＡの酒匂さん酒匂さんは、何を分散されるかについて、ひとつの実例と考え方を示されたのだと思います．私は常日頃から、環境（協調作業の為の仕組み）を切り出すと言う考え方のもとに分散環境を構築していきたいと考えていますがいかがでしょうか．
・横河電機の西岡さん環境の構築に際しての考え方の中で、自分に好ましいではなく、自分達に好ましい環境＿と提案されている点に特に共感しました．

吉村　智香子

(3-1) なかなかおもしろい話だった．ワークステーション＋イーサネットで分散環境が構築されているというのはまちがいで、本当の分散環境とは何かを考えもっとおもしろい使い方の分散環境を確立していくために論議を行いたい．
(3-2) xme の宣伝を聞いているようだった．xme 自体は今後、改良かなされてどんどん成長していくだろう．今後の分散環境との構築を期待する．
(3-3) なんか、分散環境の話　というよりデータベースの話を聞いているようであった．

鎗　友良
西岡さんの発表は面白かった．こちらも興味を持っています．

青木　淳
セッションがばらばらョ！

鶴見　泰久
ほとんど寝ていた．

村川　貴広
・コンピュータ通信も、衛星通信になると面白いと思う．JUNET を今後もっと活用して、SEA の各メンバとの連絡を密にしたい！
・BSD のサーバクライアントモデルについて、批判がありましたが、そこまで考えているとは、まったく、驚きました．BSD では、サーバ・クライアントがもっともだと思っていたのですが、問題は、あるところにはあるものですね．今後もっと UNIX の将来について考えてみたい．

小林　明彦
分散環境は本当に必要か？私もそう思います．

海尻　賢二
bitmap ＋ mouse が使えれば現在のところ分散の必要はないような気がする．lan を経由して使うのはそのソフトが直接つながっている
machine にないから．その意味から一台に十分なリソースの余裕があり、ソフトも十分あればそれでもよいのでは
(WS base のユーザインタフェイスは別)
本当か？

伊野　誠
１．加藤さんの発表は、junet の現状を知る意味で参考になった．
２．酒匂さんは興味深いツールの話で今後の発展が期待できる．
３．西岡さんの発表は、分散環境というより、ソフトウェア・データベースのセッションにふさわしいもので、なかなか参考になった．

市川　寛
１．分散か集中かの再検討 (?) を加藤氏が提示した理由は何か．
２．集中 ＋ 分散というのは構築できないだろうか．

野中　哲
Network はなんでもつながらないといけない．計算機だけではいけない．マイコンのソフト開発では、ICE（インサーキットエシュレータ）がはなれてしまっている．

田中　慎一郎
もっと技術的にこみいった話ばかりになってしまうかと思ったがそうではなくとても良かった．
西岡さんの話は、何かひっかかるところがある．ちょっとおかしいような気がする．テーマも少しちがうような気もするし…

中野　秀男
(1) kato@cs.titech　junet の管理者としての苦労をぶちまけてほしかった．

井上　尚司
３人ともほとんど重なる点がない発表であったが、「ばらばら」という感はなく、「段階的」だなぁと思った．こんなのもおもしろいですね．

中野　秀男

(1) Chairman です．思ったとおり Spectrum が大きくて(参加者の)、左と右の統一見解はむつかしいですネ．

佐藤　千明

・隣の人との会話まで WS で、Mail する必要はどこまであるのか．その会話内容が LOG さめて誰かが管理／再利用／検索できれば価値ある？

・「 WS は、MMI のみ担当すればよい」は、賛成．

・ホストは，おそい，高い，U/I がまづい　→ WS を活用したい．

但し、D/B まで分散しようとは思わない．

盛田　政敏

分散環境で何をしようとするのか？問題を分散するのか？それとも、分散にいる問題を解決(統合的に)しようとするのか？現状では議論がはっきりしていない．分散でならなければならない理由、分散の方がメリットと思われる様な理由をはっきりさせたい．

歌代　和正

疲れました．

岡本　ＫＣＳ

１．分散環境は良いのか？何故（目的）分散させるのか！何を分散させるのか！によって良し悪しは変わるが、要は、分散の中での協調をどう考えていくかが重要に思う．

２．WS の正しい使い方

ss@astec

今の分散環境のこまかい管理の know -how を教えてほしかった．

堀川　博史

加藤さん歌代さんの「分散において、ここで話している内容は、集中として見せることで何も問題がないという」話はおもしろいかった．

ただ歌代さんの討論テーマ「自分の机上にワークステーションをおきたくない」という話は、おかせたい人間がどうするかという問題があると思う．これは加藤さんの「ディスクレスにすれば良い」という解であろうか

吉井　孝　日立ビジネス機器

ネットワーク分散管理の難しさがさらに認識できた．分散環境というのは、実は分散しているものを集中して使おうという事であるという意見には大変興味を持ちました．

近藤　康二

・造る過程が大切であるという話は、なかなかおもしろい．青木さんの次のデモがたのしみです．

・WS の正しい使い方→同感です．

市川　寛

・分散処理のアプリケーションはなにか．

・メインフレームはプログラミング作業のみをサポートしていたという考えは賛同したい．

西岡

・ネットワーク　ミニコン間 WS 間

混合

各々開発環境としての意義が不明確．

・ハードウェアのパフォーマンスさえあれば１台の CPU でこと足りるのか．WS のネットワーク環境独特のメリットはソフト開発の上にはないか？

三浦　あさ子

最初にコンピュータにさわった時には、６０人ぐらいで、１コの、あの、とても近よりがたい．”神秘的な”マシンルームにある大きな箱を使っていたのに…

なぜか、そのうち机の上にある端末に、日本語がでるようになっていた．その頃には、少しは、はいってもだいじょうぶになったマシンルームの２つの箱を使ってもよくなっていた．なんか知らないうちに、わたしのとなりの席に、小さな箱と大きいテレビ(画面)がある．これはすごいできごとだ．

新田

桜井　麻里

ネットワーク　＋　ワークステーション

去年よりは、ましみたいですが、まだまだイメージがちがう人がいるみたいですね．それぞれかかえている問題がちがうのですが、てきとうにちらばってよかったのではないですか？

ワークショップですから、問題提起だけでおわってもぜーんぜんかまわないと思います.

加藤　朗

·会場の人々のモデルがＳＥＡの方が、ＪＵＳより分散が大きく、なかなかまとまった議論になりにくい.
·もっと細かい技術的な話なら＿ 僕は、この方が好きなのですが＿

酒匂　寛

「分散開発環境」という言葉に意味があるか？
·いろんなマシンを使いましょう（モチはモチ屋）
·リソース分散の礎は何か.（分散基礎技術）
·安くあげましょう

森　幸一

青木氏のいう network にソフト開発プロセスの情報を蓄積して利用する機構は有用と思う.
WS 利用は１人１台であっても各個人の上にグループ、部門等の階層があって、分散環境は個人適レベルだけでとらえられるべきではない、部門が違えば、この部分ハ、ローカルに処理し、蓄積したいテーマも存在する.

福田　充利

·負荷は分散、情報は集中ということでないでしょうか.
·このセッションの必要性が？

藤野

分散環境と WS
分散環境とは、分散するのではなく、何処に仮想的に集中しているかのイルージョンを与へるかと言う事だと思う.
それは今、まだ試行錯誤の段階を「洗礼」として受けなければならないかも知れない.

野中　哲

WS + Network までは、良くわかった？というかある程度の議論になった. ような気がする.
分散環境とはなにかよくわからなかった. サコウ氏と同じ意見か？

臼井　義美

分散環境で問題になっている多くの事項は集中環境でも同様に問題であることが多い. やはり MMI と管理方法につきるか.

小池　幸徳

分散環境は水平と垂直をうまく組み合わせることから考えなくてはならないのでは

田中　正則

井川さんがおっしゃったホスト、WS 接続の話に大変興味を持ちました. BOF でまた、話したいと思います.

鐘　友良

分散環境の共同作業について、たしかどうするようにいい要求／設計仕様記述を作る必要があるじゃないかなの気がする.

北野　義明

やっと「何を討論するのか」ということが見えてきたところで、残りあと２０分＿ 歌代さん（ＳＲＡ）の話がでてくるということに問題がある. 当たり前の認識（細かな技術的な事は別にして）であり、それを前提に討論が進むと思ったが、
最後の加藤さん(東工大)の話が全てで、それを基に突っ込んでゆきたかった.
途中では、深瀬さん（アスキー）の話が最もな事で、JIP（臼井さん、井川さん）や、うち（ＫＣＳ＝盛田、岡本）の世界の問題の重要性をご理解して頂きたかった. 中野先生御苦労をかけましたが有難うございました.

伊野　誠

分散環境についてかなり本質的な討論ができたと思う.
ＵＮＩＸの次のプロジェクトとしして、分散ＯＳが考えられていると聞くが、分散 OS について討論したかった.
junet の発展が必要だと思うが、何かいいアイデアがないだろうか.
メイン·フレームの世界は大変だが、Σ計画で何とかなるのであろうか.

田中　慎一郎

話がおかしい. そういう話をしてもしかたがないような気がする. これは具体的なノウハウ·レベルの話だと思う. どこか別のところでレクチャーすれば良いのではな

いだろうか（問題としては、大事だとは思うけど_）

分散したら困ったよという話をしてもここでは意味がない．

前置もの話をしているうちに終わってしまった感じ．

発表のスタートの印象とくらべて、後ろ向きの話からぬけられなくて残念最後少しもちなおしたが、まだちがうみたい．明日のセッションもこのようになるおそれがある．こわい．

村川　貴広

今日の討論で分散環境の真(?)の意味がわかったような気がして、本当によかったと思う．討論もなかなか面白くきけたが、私が意見をいうまでにはいたらないのが残念！！帰ってからは、自社に正しい分散環境を構築して、その成果を今後発表したいと思う．しかし、まだ分散環境という環境は、意味が広いようなので、それぞれの立場(環境）で望みたい．

みなさん、それぞれに自社の環境というものの改善にいどみ、その成果を発表されているようですが、私ももっとソフトウェアの開発における環境というものを考えたいと思いました．そのためにも、今自分に与えられた仕事を完璧にやりとげ、ふりまわされないようにしなければならないなと感じます．

吉村　智香子

各人の分散環境のとらえ方にズレがあったように思う．

分散環境の分散ということば、集中という意味であると思う．分散とは、資源の分散ではなく、処理の分散であると思う．ここで、分散環境のアーキテクチャを討論する必要はないと思う．

今いわれている分散システムで本当に解決しなければならない問題はいったい何なのか．そしてその解決のために何をすべきかを考える必要があると思う．

メインフレームの話はよくわかりません．すみません．

久保　宏志

「なぜ分散環境なのか」から出発するのは、いい問題のたてかたである．

平凡の不毛な議論にこりずに、何度も何度も行うのがよい．

井川　裕基

分散開発環境は、まだあまり確立していない事がわかった．

それでは WS とバラまいても無駄か_?

鶴見　泰久

分散環境の定義は如何に（基本的な質問）WS の定義にかかわらかず、パソコンでは分散環境になりえない．WS とはどのような位置／機能を言うのか．生産性が一番上る環境はどのように

CTC では WS があるが、分散環境がない．（物理的／人的）

小林　貞幸

・集中化のデメリットに対して分散が現れたのであり、一方、分散の中にあってよりメリットを生かすために集中が見直されたと思う．いかに水平と垂直の性格の違いがあろうと事質は同じはず．（IBM VS DE この事質論か？）

・ UNIX や W/S 環境で楽しんでいる SEA のメンバーが、事務分野のホスト環境で苦労している．

悩める者へ救いの手を！

みきや

分散開発環境としうのは、ある目的を個人に上手に分散させて効率よく処理する手だてではなく、ある目的にむかって多勢の人の意見を交換．集中させる共同作業を支援するものであると確認できた．

深瀬　弘恭

パネルのタイトルは "WS + Network による分散環境の構築" ということであったが、何が問題となっているのか良くわからなかった．

上記環境で特別の問題があるとは思えない．もちろん、管理上のもろもろの問題はあるであろうが、使いたい人は使えば良いのでは！

熊谷　章

分散とネットワークのテーマ中味（情報として飛び交う意味のあるデータ）の流通について話したい．でも、中味が詰るかな？ information provider が必要．

井上　尚司

ホスト名を意識しながら使う必要がないのが good な分散環境であろう．

ホスト名を常に考えねばならないようなのは、分散環境で

はなく、ただ単にコンピュータが電線でつながっているだけだとおもう.

荻生　準一

私が WS に着目していたのは、大 Project のプログラム製作ピーク時に、WS を使用して target machine の質問(= mashine time, TSS 端末数) が軽減出来るのでは？と思っている. 但し、制御末の Program では、target mashine の resource (= File, Grobal, 周辺機器)に依存する部分が多く、これらをいかに WS 内に構築し(シミュレート) プログラムデベッグが、出来るかが問題点である. (制御系内部の計算 Pro. ではメガ・バイト単位のデータを使用するケースが多いです.)

小林　明彦

(1) 大量ワークステーションの一元管理(ユーザ、ソフトウェア　etc) どのような管理が理想的かと考えていたが、議論の中では、何らかの問題があるということでとまどっている.

(2) 分散環境のとらえ方がかなりの広範囲にわたるということが認識できた.

(3) JIP 井川氏の発表は参考になった. 当社も同感です.

林　香

分散環境だとどうも話が噛み合わない. 人により経験／知識が違いすぎるのかな？やれば良いだけ？

分散環境セッション・サマリ

大阪大学工学部通信工学科
中野秀男
nakano@oucom.osaka-u.junet

0．まえがき

昨年もそうだったが今年も卒業研究の時期とぶつかり、そうのような環境で書かれたことを含めて読んでください。

1．WSと分散環境のチェアパーソンをつとめて

ポジション・ペーパが分散環境なのかどうかしらないが、プログラム委員の分担として座長をつとめた。以下、思いついたまま書き連ねることにする。サマリにはならないので、ご容赦を。でも、真面目にサマリを書くのは佐藤さんだけだと確信しています。

ワークショプの始まる前に、カントリーベア実行委員長からこのセッションは一番難しいねと言われ、前夜と前々夜のお酒も控えめに望んだ。ポジションペーパもメインフレームを含んだホスト派から、真の分散環境を目指すものまであり、どこに焦点をあてるかが難しいところである。一応、分散環境とは、「(1)すべてのコンピュータがネットワークで結ばれており、(2)各ユーザは一体どのコンピュータで、プロセスやファイルが動いているかを意識しなくて、(3)快適に仕事のできる環境」と定義しておく。

2．私のネットワーク環境

WSとネットワークという時には、本人の計算機環境を示しておかないと、議論が空回りする。以下はネットワークにつながった、私のまわりの計算機群である。一応、私のマシンはsumiである。

```
NEWS830(oucom2)-NEWS841(oucomh)-NEWS711(sumi)
                                -NEWS711(tomoko)
               -SX9100(?)(4月からモニタ)
AS3052M(oucom5)
NEWS841(oucom3)
NEWS841(oueln2)-NEWS830(oueln1)
                Balance
Mac AppleTalk
PC9801 PC-NFS
PC9801 FUSION
<- Terminal Server <- PC9801
-> Terminal Server -> Modem -> ACOS2000
<- Terminal Server <- PC <- Radio Ham Packet
```

ようするに、メインフレームも含めた恐ろしい異機種間ネットワークである。ボストマスタがもの好きなので何でもつないだとも言われているし、大型を使う人が多いのでしかたがないとも言われている。

3．S＊Aをみて

すごい水平分散だと思っている。栗原さんから苦労話も聞いており、開発環境としては「1人1台EWSを」運動の1つの典型であろう。私もこの線で動いているが、やってみて分かったことは、WSを使う人の技術力が非常に必要だと痛感した。大学では、お金がないから、「1人1台WSを」運動は、遅々として進まないが、もし有ったとしてもWSが100%以上に使われる所と、WSが腐っている所に大きく分かれそうである。だから、加藤さんのいう「分からないものはWSを使うな」論や、阪大基礎工の下条さんの「分散環境の資本主義」論がでるのだろう。

岸田さんの言う「企業内遊園地」論が長い目でみて、企業利益につながるのかという辺りが興味がある。FXISもだが、分散環境は、チームとしてのソフトウェア開発に向くのだろうか。

4．J＊Pをみて

大手ソフトウェア・ハウスの典型だと思っている。WSとメインフレームの差がないのが分かった時どうなるのだろう。現在はもうかるだろうが、どのようにいわゆるソフトウェア危機を乗り切るのだろうか。東京はともかく、大阪の方は良く研究会なぞで遊びにゆくので、フロッピーやカード(ICカードではない)をもって走り回っているらしいので、真の分散派はめざせないであろう。なにはともかく、現在稼いでいるのはメインフレームであるという議論からは何もでてこないのではないだろうか。

5．分散アルゴリズム

分散環境でなければ出来ないアルゴリズムを議論したいとの加藤さんの発言が冒頭にあった。座長も同じ大学人なので、この展開は歓迎だか、参加者の分布から考えて、この話は取り上げなかった。ここらあたりは、プロトコルの話で、プロトコルの検証や多くのプロトコルが提案されているので、加藤さんの気持ちはWSをノードとした分散

環境で本当に役に立つ(効率が良くて、かつハングしない)プロトコルを含めた分散アルゴリズムを考えようという事だと思う。

座長は本当は計算機科学の基礎論が専門なので、国際会議でもこの頃は普通の直列アルゴリズムの提案はなかなか採録されなくて、並列や分散アルゴリズムだと比較的簡単に通るので、安易に並列や分散アルゴリズムの研究をしているが、まだまだやることはあるようである。特に、実際に分散環境を構築してからの実験例も含めた研究はこれからだと思っている。

6．ネットワークの管理

これはいつも出てくる問題で、要するにネットワークの管理者が「とろい」かどうかという問題と、良く分かっていないユーザを切り捨てるかどうかの問題だろう。

管理者が努力を怠らなくて、ユーザが良く分かっていて、分かっていなければ、千尋の谷底に落とすのなら、みんなでより良き環境をめざして頑張ろうで話はおしまい。

管理者がとろくて、ユーザがダメ社員ならどうするのでしょうね。メインフレームのMMLを使うのでしょうか。

管理者は一応努力するんだけど、ユーザは中高年だとかいう環境がたいへんです。そのうちアマさん用のMMIができて変なものをさわると、「あっ。そ、そ、そ、そこはダメよ」と音声で出て、ログアウトしたりすると、本日のあなたの得点なぞが出たりするといいですけどね。「ウィンドウはすごいんだけど、字をもっと大きく表示しないと(老眼鏡でも)見えない」なんて変なクレームばかりがきて、管理者の気持ちをくじくことしか考えないのかとも思います。

7．垂直分散

このごろ証券会社や保険会社等で、メインフレームとパソコンのリンクMML構築の話が良く新聞にでている。メインフレーム屋さんから見ると、メインフレーム－WS－パソコンの垂直分散に見えるらしいけど、WSの力の過小評価のような気がしてならない。深瀬さんから、IBMだって今は水平分散を真剣に考えていて、今のメインフレーム1台のパワーで数百(数千かな)のTSSをこなしきれないから、MMLとかっこのいい言葉をつけてパソコン側に負荷を分散させているのだという意見は正論だろう。

大阪大学の情報処理教育センターでも進んだCAIシステムとしてIBM3090にトークンリングで数百のIBM55xxをつないで情報処理演習をしているけど、55xxから3090を使っても55xxのLotus123を使っても課金される。前者はともかく、後者の課金にたいしてはなにを考えているのだろうかと思っているけど。話をもとに戻すと、FORTRANの授業をしていると、面倒だから家のパソコンで、たとえばMS－FORTRANでプログラムを組んで動かしたけど、IBMマシンでは精度が悪いとか動かないと言って来る。精度については、浮動小数点の計算機での表現を教えないとだめなので、マンテッサから説明すると理解してもらえる。実はそういうことは授業と並行して教えてこそ演習なのに。MMIなんて、はるかにパソコンの方がいいし、演習ぐらいならパソコンでやってもハードディスクならあまり変わらないのにと思っているのに。

8．水平分散

いろいろ分散環境の話(各人が各様に思っている分散環境)が出たが、それは分散環境ではなくて、

協調処理環境

ではないかとの意見がでた。NFSだとかRFSだとかが出てきて、ホスト名を意識しない環境が正しい分散環境だというのは皆さん頭の中に共通点としてあるようだが、水平分散といってもLANで横につなげば水平分散と思っていて(まだ構築してない)その頭で議論するのと、既にLAN等でネットワークを張っていて、NFSもガリガリかけていて、その中でさらに問題点を突き詰めようとする人たちとのギャップがどうしようもなかったという感じです。

9．MMIとWS

WSはMMIだけやっていればいいという極端な意見もあった。MMIを広義に解釈すると本当に正しい認識かもしれない。今は回りの計算機パワーが足りないからとか、大型計算機は使いにくいとかいう理由でWSを字のとおり「計算機」として使っている。我が研究室もそうだが、ditroffで卒業研究のレポート(論文?)を書いている裏で、暗号解読のプログラムが日夜動いているのが現状である。「WSはMMIのために使おう。単なる端末のかたまりにしたくない」との雲地区の酒匂氏のうん蓄は正しい。

１０．ｊｕｎｅｔ

ｊｕｎｅｔの管理者の加藤さんがプレゼンテータなので末端ポストマスタがあまりサマリを述べるのはおこがましいが、昨年の長岡の環境ワークショップ以来、私のいるｏｕｃｏｍ２もｊｕｎｅｔにつながって、海外ニュースまでもらってと思っていたら、現在下に６つもサイトがあって、朝と夕方には３台の電話がつながりっぱなしの盛況である。管理者としてたいへんだが、なかなか楽しいネットランドです。１ユーザとしてもニュースやメイルは役に立っているし、何よりも学生の視野が広くなってきたのがうれしい。「ａｒｂｉｔｒｏｎ」でサイトのニュース読者を毎週調査しているが、１００ユーザ（学科のＷＳなので）で４５人程が読者である。ローカルのメインＮＧは常時３０名ぐらいの読者がいて活況を呈している。Ｎ１ネットだとかややこしいネットもあるので、ｊｕｎｅｔは実験ネットとおまじないを唱えながら、大学内公認ネットワーク計画を眺めている今日このごろです。

１１．ＮＦＳ

ＮＦＳの話は歌代さんがぽろっと言った以外出なかった。出なかったというより出るように議論は前進しなかった。座長が意識的に後進させて、後ろにいる人に合わせようとしたふしもある。

ＮＦＳは今、私の所はバリバリかけて卒研の追込みをしている。ＹＰはやっていないが、メインのサーバのＷＳがこけるとわいわい言ってみんな楽しそうに集まって来る。フォントの塊を空いているエリアに移して、ＮＦＳで使ったりしてやっているけど、一番いいのは学生のネットワークをつかった計算機環境に対する驚き。嫌いな（？）夜の集まりに出て情報を集め、一応まあまあのＷＳの環境と、八方手をつくして集めたソフトと利用技術を使わせて、安心して子を実業界に送り出す気分である。

１２．関西のＮＥＷＳ

昨年、ＮＥＷＳを買ってネットをはってから、いろいろ私に聞きに来られることが多くなった。私は某社から情報を仕入れるし、ネットランドやＳＥＡ居酒屋連からも正しい情報を仕入れるからいいけど、お金があってＮＥＷＳが安そうだから買ったし、ついでにネットも張りたいから教えてといった話がたくさんある。ＷＳが百万円となったらみんなパソコン感覚で買うから、ますますコンサル気味の話が多くなるだろう。普通の人がＮＥＷＳを買っても、普通の人しかいなければどうするんでしょうね。いくらＭＭＩがいいＭａｃ感覚のＷＳがあってもａｄｍｉｎは必要だから、ディラーさんがやるのでしょうね。いつも「しっかりしたディラーから買って、私の手を煩わせないでね」と電話の届かないところでそっと言います。

１３．おまけ

阪大医学部への手紙です。阪大の医学部は３年後に吹田キャンパス(我々の所に来る)に移転されます。現在、インデリジェント医学部を目指して、とりあえずＬＡＮシステムを構築しています。やっとｊｕｎｅｔをつないだあと医学部のポストマスタに送ったメイルです。

「ＵＮＩＸというのは、あくまでマニアのマシンです。黒沢先生のいわれるように、誰でもつかえるようにとの事は、現在２つのアプローチでなされています。

１つはシｘｘプのＯＡ２１０や、日ｘの２０５０で対象を一般のＯＡ人(？)としており、何も難しいことをしなくともやれることをめざしています。だから、最初の内はすごいと思うのですが、使い込んでいるうちに遅いとか、こういうことがやれないとかの不満がでてきます。

だから、簡単なことには便利なんですがちょっと細工がしたいときにはとたんに駄目になったり、法外な開発費を要求されます。このようなマシンは、２年先の技術進歩にはついていけないのではと思っています。理由は、ウィンドウやネットワークを独自に開発しているから、今の

ウィンドウはＸウィンドウ

ネットワークはイーサのＴＣＰ／ＩＰプロトコルとＮＦＳ

といわれる世界標準の考え方からはなれていくからです。

例えば、取引先の関係で日ｘでないと駄目というのは仕方がありませんが(悲しいですが現実です)、そうでないのなら、このアプローチはとらない方がいいでしょう。

２番目は、ＳＵＮやＮＥＷＳをフロント･エンドにおいて、大型機やスーパーコンピュータをバックエンドに置くアプローチです。大型機はデータベース、スーパーコンピュータは高速計算用で、大型機も分散型データベースが普遍化すると、たんなるバックアップ･マシンかもしれません。

これ方法はかっこがいいのですが、一つだけ悪いのはまだまだＵＮＩＸフリーク用になっていることで、ＯＡ人にはマニアックで使いにくいでしょう。このＥＷＳをＯＡ

用に使いこなすためのマン・マシン・インターフェイスの研究は非常に盛んに研究されだしています。

Ｍａｃのハイパーカードの感覚で、ハイパーテキストや、音声・画像・動画も含んだハイパーメディアの研究も前者は２年ぐらい先を、後者は５年ぐらい先を実用化のめどに進められています。

まあ、いろいろ意見を書きましたが、まだ言いたいことはたくさんありますが、ここらで今日はやめます。

最後に１点、ｊｕｎｅｔについてですが、昨日、黒沢先生がいわれた、そろそろｊｕｎｅｔも一般のユーザに使えるようにとの言葉ですが、それぞれの社会にはやはりエチケツトがありますし、ｊｕｎｅｔの維持だけで、多くの人が自分の研究以外の時間を奉仕しています。だから、辰巳さんのような人がなんらかの時間をあてないと、ニュース・システムの維持はできないでしょう。

医学部のｊｕｎｅｔにたいする責任は１つ上位の私にもありますので、共同で各ユーザーにエチケットを守らせるしかありません。それがいやなら、パソコン通信を使うか、大型計算機センタにたよるか、システムハウスに作ってもらうしかないでしょう。

ｊｕｎｅｔがどうなるかは、わかりませんが、現在５０００人ぐらいが読んでいるニュースシステムですから、入れ物としてのシステムではなく、その中での使い方のノウハウがすごくあります。これは、使ってみないとわかりません。

セッション4
開発環境管理の実際

チェアマン　臼井義美（日本電子計算）

プレゼンテータ　山浦恒央（日立ソフトウェアエンジニアリング）

田中一夫（山一コンピュータ・センター）

桜井麻里（ソフトウェア・リサーチ・アソシエイツ）

－ 内容 －

- 事前アンケート
- プレゼンテータのポジション・ペーパ
- セッション・レポート
- 事後アンケート
- セッション・サマリ

新しいハードウェアやツールの導入、ファイルのバックアップといったことに始まり、開発環境の維持改善、管理には多くの労力を必要とします。この苦労はＴＳＳ環境、分散環境を問わず付きまとうものです。開発環境管理のために様々なツールが生み出され、多くのノウハウが蓄積されているものと思います。すぐ目の前に来ている１人１台ワークステーション時代の開発環境管理をうまく乗り切るために、いままでのツールやノウハウを公開し、作を練りましょう。

熊谷　章／ＰＦＵ　研究開発センター第三開発室

◆よいアイデアや効率的な方法を産み出す環境

昔から交通が盛んで豊穣な風土に文明が栄えた。そこでは、科学と工学が発達し人々は快適な暮らしをした、ようにみえる。コンピュータを利用した世界で、交通が盛んでかつ豊穣な風土とは何か。これは開発環境そのものである。交通は、グローバルとローカルなネットワークであり、豊穣な風土とはコミュニケーションの実体となる言葉、ツール、方法論、ソフトウェアプロダクト等ではなかろうか。この考えの具体案を考えてみたい。

◆サイバネテックスな開発環境

集中管理を必要としない開発環境がよいと思う。各ノードで自分にあった管理を臨機応変に、かつ自主的にやるようにしなければならない。そのためには、道が分かればよい。道が分かれば、姿勢が分かり、全体と部分が分かりサイバネテックスな開発環境ができ、そこにサイバネテックスな人が住むようになる。

岡本隆一／神戸コンピューターサービス　ソフト開発部技術第一課

◆ネットワークの維持と管理をどうするか…！？

分散化や異機種間接続の推進によって、あるいは効果的なコミュニケーション・ツールとしてのネットワークは、拡大、複雑化が進むにつれて維持工数や管理工数の増大がバカにならなくなる。

中野秀男／大阪大学　工学部通信工学化

１年弱しかいない学生と、３年弱の学生がいて、計算機科学の授業もしっかり受けていないので、１から始めるため苦労しています。今年から UNIX とか Ｃ言語のビデオを入れたり、既存のソースプログラムの利用など、UNIX 環境を軸に楽しくやりたい。

深瀬弘恭／アスキー　システムソフトウェア事業部

◆ WS のアドミニストレータの教育

新人プログラマ及び経験者でもアプリケーションプログラマの場合は自分のかかえている WS の管理が出来ない。

効率の良い環境保守（ネットワーク）の方法は？

◆ WS 間のソフトウェアツールの世代管理

WS の数が多数になってくると、同じ機種の WS 間でもその上のソフトウェアツール群のバージョンは統一がむずかしい！これに対する解答は？

藤野晃延／富士ゼロックス情報システム　技術部

◆ Network への Node の追加／削除／再編成

Network 化された環境で問題なのは、新規の Node 追加、あるいは削除といった事が smooth に行える必要があることで、この場合、Node は個人用 WS でも良いし、共用の Server であっても良い。

また、Node の増加に伴う Network 構成の再編成なども Dynamik に Flexible に行える事が望ましい。しかし、現実には Net の維持・管理には絶望的に人手が懸かる。この辺はどう対応していけば良いのか。

久保宏志／富士通　システム本部パッケージ企画統括部

◆環境論理の構造化（論理体系と実理系の対応）

環境特性をどう整理するか。それを問起の構造とみたとき、解法の構造であるところの、環境の階層構造との対応関係を整理する。

◆環境論理の構造化（環境の評価）

議題１の意味での解法構造をテンプレートにして、実際の環境を評価してみる。

岸田孝一／ＳＲＡ

◆環境構築における技術移転

関連技術がめざましい発展を続ける現在、環境の構築は技術移転の色彩を強く帯びざるをえない。

新技術の開発や導入とその実用化による利益（生産成向上など）の獲得とのあいだのバランスをどうとるか？そのための組織運営上の配慮は？といった問題を議論したい。

私自身の解答は「企業内遊園地」の建設である。

◆いまシグマを考える

この技術移転の観点に立って、シグマのあり方について考えてみたい。

私自身の考えは、周知のごとく、現在のシグマの運営方針とはまったく対立している。

御喜家貴子／横河ヒューレット・パッカード　マーケティング部門

◆自分の責任と他人との協調は忘れられない！？

現状では、ワークステーションの導入化がはじまったばかりで、管理については、各ワークステーションユーザが、自分で決め手バックアップを取るなり、バージョンアップをするなりしている。基本的には、自分のディスクは自分で守れ、ということになっているが、ネットワーク上で活躍するサーバなどは、自分かってにはできない。しかたのないことだが、共有するシステムに関しては、声をかけあって、お互いが失敗のないように心を配るのみである。従来の TSS 環境プラス個人の環境管理の２頭だてで、高速なバックアップ装置や、フォールトトレラントなシステムが我々の心がけを支援してくれるよう手ごろに登場することに期待しています。

伊野　誠／ＳＲＡ

◆ OS にもっと仕事をさせるべし

１台の大型コンピュータ上での管理は容易であるし、必要な管理はされている。しかしワークステーション＋ネットワークによる分散環境ではいろいろな困難が発生するであろう。全体としての OS のバージョン・コントロール、CPU やディスクの利用状況、ネットワークのこみぐあいなどをウォッチし、システムの管理者に適切なレポートを出す管理者用 OS が必要である。

鶴見泰久／東海クリエイト　VAR 推進室

◆パソコン関係開発環境の問題点

パソコン関係は商品化激化で、開発のクローズが非常に困難。その時のライブラリー管理、仕様固定等で非常な問題となっている。あとに続かない、開発の終結になっている。次のレベルアップに又同じ事をくりかえす事になっている。

山浦恒央／日立ソフトウェアエンジニアリング　第１技術本部第５設計部第４課

◆開発環境をどのように進化させるか

開発環境に格納されている資源（再利用されるソース、オブジェクト、仕様書、また開発環境支援のツール等）をどのように進化させるかは非常に重要な問題であるユーザの新しいソースやツールを簡単に登録でき、また全使用者が共有できる仕掛けが必要である。

森　幸一／ＰＦＵ　開発企画部ソフトウェア検査課

◆分散開発環境管理の自動化

分散開発環境では、個別のマシンは利用部門が管理するが、ネットワーク全体にかかわる部分はどこかの部門がまとめて管理することになる。個別の管理は個々のマシンの事情で自動化は不可能と思われるが、ネットワーク全体にかかわる管理はある程度まで自動化が可能と思われる。

村川貴広／ HST

◆バックアップ体制

ファイルのバックアップは管理上重要であることは言うまでもありませんが、現在のような分散環境の中でのファイルの量は、莫大なものになってきています。そこで、これらネットワークを介したバックアップの方法やその計画について討議したいと考えています。

西岡健自／横河電機　研究開発 4 部第 1 研究室

◆管理コストの同定

開発環境の必要性の認識は得られつつあるが、その維持／改善は構築担当者が仕事のあい間に行わざるを得ない状況も少なくない。本来この種の管理は専任の担当者がいてしかるべきと考えるが、その議論には管理のコストと効果をある程度把握する必要がある。

和田　勉／長野大学　産業社会学部

◆開発環境の管理の実際

いままでのところ大規模な開発環境を維持管理した経験がありませんので、省略させていただきます。

佐原　伸／ＳＲＡ

◆環境定義言語としての HyperCard

HyperCard の概念を拡張して、環境定義記述言語を作る必要がある。

プロセス・プログラミング言語としても Ada ベースのものより可能性がある。

歌代和正／ＳＲＡ　環境開発部

◆ネットワークの管理

ネットワークの管理は大変である。日本ではまだ広域のネットワークが発達していないために、それほど問題になっていないが、大規模ネットワークの管理は本当に大変そうである。早く広域分散環境の技術を確立しなくては。

桜井麻里／ＳＲＡ　環境開発部

◆社内(組織内)における開発環境の管理

自分達の環境は、自分達で管理していくものである。開

発環境は、そこで仕事をする人達がやりやすいように自分達の手によって改善しなければならないものである。といっても皆が勝手なことをしていると全体としては統制の取れないものになってしまい、結局、仕事がしにくくなる。したがって、上の方で誰かが方針を打ち出して、それに沿ってある程度同じ方向になるように各人が努力していくしかない。細かい方法論などはいろいろあると思うけど、大切なことは、みんながそういう意識を持つこと。

◆とはいいながら...悩むところはみな同じ

とはいえ、みんな似たような悩みを持っているわけである。だから、ローカルに悩んだことは、(ある程度抽象化した上で) なるべくみんなに公開するのが望ましい。そういう情報を交換できる場を多く持ち、そこで得たことをまた自分の環境にとりこんでいく。そういう姿勢が大切である。そして、そういう作業は往々にして、ある特定の人にたよりっぱなしになるものであるが、その人達が雑用のみで忙しくなる事態は何とか避けなければならない。できれば、組織外の情報を取り込んで自分達のものにして行くための専門の部隊を作り、そこが中心になって整備して皆に広めていくのがよいであろう。でも、結局は個人個人の意識の問題である。

田中正則／ＳＲＡ　開発第１部

◆開発環境の管理に決定打はあるか？

１人１ワークステーションの時代を想定した場合、開発環境を全体主義的に一つの環境と考えないかぎり、管理の決定打はないと思う。又、環境を全体主義的に考えたのでは、１人１WSという特性を損なってしまうであろう。なぜならば、個人の能力及び趣味は異なるからである。ではどうすれば良いか？　まず開発環境を個人もしくはグループでオプティマイズ可能なように提供する。(あくまでもツール単体の話しではなく環境の話である。) 出来るならば、あらゆる能力の人用に変えた同一内容の情報を適確に流すことがベストと思われる。これが出来れば、開発環境の決定打となるであろう。

北野義明／ KCS　ソフト開発部技術第一課

開発環境の管理と言っても、TSS 環境化でのファイルのバックアップや、　　ライブラリの管理を、メーカーの提供するツールをマニュアル通りに使って行なったことがあるという程度である。他には、UNIX-WS をほんの少しかじったが、(当時、スタンド・アローンであった) その際、自分の思うままにディレクトリやファイルをさわれることの便利さと逆に不安のためにバックアップを頻繁に行なっていたので手間がかかったという印象が強い。これらの経験から自分によって便利な「開発環境の管理方法」を考えてみたい。

田中一夫／山一コンピュータ・センター　開発第二部基本システム課

◆開発資源管理

ユニバック　開発用テープ: 4000 本　ディスク: 5Gx12＝60G

日立　開発用テープ: 700 本ディスク: 5Gx4+2.5Gx2＝25G

特に、ユニバックはいとも簡単にディスクが使える様 (ルールを作っても関係なく) になっているので、ひどい使い方をしていると思う。これは自由に使えるということではよいが、人間が増えてくると管理できなくなり苦労している。そこでとりあえず、管理用 DB を作成して、何とか管理しようと考えている。

田中慎一郎／ＳＲＡソフトウェア工学研究所

わかりません。セッション３の話もこれに関連するとは思いますが... 環境の構築と同時に使用者の教育も大切だなと思います。

小池幸徳／日本システムサイエンス　システム部システム技術課

◆セカンドファイルサーバーの利用

常に開発に利用されるデータは、メインフレーム上の強力なファイル管理機能により保護され、その一部 (頻繁にアクセスされるもの) を WS のネットワーク上のセカンドファイルサーバー上にダウンロードし、WS から高速にアクセス出来るようにすることにより、システム開発の情報管理を効果的に行なうことができるような気がする。ただし、ファイルサーバーとセカンドファイルサーバーはあるタイミングで同期を取る。

青木　淳／富士ゼロックス情報システム　技術推進室

◆開発環境はひとつの言語の下に生まれる

開発環境は、それを使用する人が自分の好みのものにできなければなりません。それは、プログラマブル開発環境であり、何らかのプログラム言語を必要とします。そういう言語はユニフォームな構造を持ち、インタフェース記述に向いた言語でなければなりません。例えば、Smalltalk-80 のような...

荻生準一／日本システム　府中事業所　第２システム技術部

◆１CPUでの複数性番のソフト開発

新事業所にてターゲット・マシンと同等機能のシステムを構築し、単体デバッグを行なう予定になっているが、複数の性番が同時にデバッグするため、ファイル、UG等の重複を避ける工夫が必要となってくる。このために、現在、様々な検討が進行中である。

◆デバッグ・ツールの開発

プロセス制御のオンライン・プログラム開発に当たっても、マシンを効率よく使用するためにバッチ処理としてデバッグできるよう添付資料のデバッグ・サポート・ツールを開発した。

井上　尚司／ソフトバンク総合研究所

◆マイコン的発想の技術者への対応

我が社においては、まだ、環境の大切さがあまり認識されていず、システムをまわりとの調整を考えて整備しようとする技術者（結局マイコン的発想をするので、関連性を考えてくれん！！）が少ないので、八方広がり的になってしまいます。結局、「ツール」以前の問題だ！！

近藤康二／ソニー　スーパーマイクロ事業本部ワークステーション事業部

◆一人一台のワークステーション時代の環境管理

現在は、非常に低価格なコンピューターが半導体の急速な進歩によって生まれてきている。昔一つの機械を多くの人が使っていた時代には、その多くの人の中の好きものの一人が環境の管理をやればよかった。しかし一人一台の時代が訪れた今、すべての人が管理者にならざる終えないのである。当然のことながら、このような状況に合わせてシステムも変化する必要があると考えられる。しかし現存の多くのシステムはそうではない。しかし、環境の管理を良く分析してみると、意外に単純なものの繰り返しや、めったに使わないものがあり、このようなものをまとめて、プログラム化するのは意外に簡単なことがわかったので、そのようなツールを試作してみた。本ツールの概要書を添付する。

井川裕基／日本電子計算開発本部ファイプロジェクト

◆ WSベースの環境での管理ツールは？

実際に WSをたくさん抱えておられる domainでは、マシン／ユーザの環境をどのようにされているのでしょう。JIPでは、このままでは"無法地帯"になりそうです。

加藤　朗／東工大情報工学部

◆開発環境の管理の実際

一人の、あるいは数人の管理できる人間によって管理されているネットワークは、管理者の犠牲によって運営されていると言わざるを得ない（無論、専属の管理者がいる場合は別だけど、それにしても、給料を貰って管理できる人を育てるのに必要なパワーはどうするんだ？専属の管理者養成講師というのがそのうちでてくるんだろうな、ある程度の需要が見込めるようになれば。）ので、誰にでも管理できる、特別な管理技術がなくてもなんとかなる、つまり、メンテナンスフリー・ネットワーク環境を構築することが必要になる。これは従来の研究でほとんど顧みられることのなかった分野で、今後の発展に期待したい。

中村暢夫／日立ビジネス機器　技術部

◆開発環境とターゲット・マシンとのインターフェース

に大型機の開発支援ツールを試行して感じるのは、OSが異なるとなぜこれほど使い勝手が違うのか、ということです。全く住み心地が違う。ターゲットレスという開発環境で作成したプログラムをターゲット・マシンにもってくる場合、解決すべき問題が発生しているのは確かです。

◆技術移転について

新しいハードウェアやツールを導入した場合、どうやってそれを普及させ、ノウハウを蓄積し、フィードバックするか、ノウハウとかマニュアルを開発環境に組み込んでしまいたいのですが。

細野広水／エム・ケー・シー

◆ソフトウェア環境（資産）の価値の認識

開発環境を整えるうえで、ソフトウェア面のサポート（ツール等の導入）についての認識、価値観が、ハードウェアの導入の際と大きく異なり、軽視されている。

3rd SEA Environment Workshop Position Paper

氏名　田中　一夫	種別　■会員no 860264
所属　山一コンピュータ･センター　開発第二部　基本システム課	
住所　〒273　船橋市浜町　2-2-2	
TEL　0474-37-3183	FAX　0474-37-3282

仕事

開発環境整備と標準化
- YCCの開発環境を最高の状態に保つ為に、H/W,S/Wの面でのシステム改善策の提案及び実施を行う。
- ソフトウェア開発の生産性･保守性向上の為に、最新の技術･技法･ソフトウェア･パッケージ等を調査･把握し、自社システムへの適用を検討する。効果の高いものについては採用を提案し、促進する。
- システム開発標準手順(AURORA)の推進と改善を行う。

作業環境

メイン･フレーム(ユニバック1100/90,日立M680H,IBM4381)
開発用端末　ユニバック　約160台、日立　約60台、IBM　約30台
船橋を拠点に数ヶ所に設置している。

J-Star　W/S　50台、P/S　12台、F/S　13台、C/S　6台
船橋を拠点に3ヶ所をIRS接続している。現在は9.6Kbpsだが近日中に54Kbps又は48Kbpsの高速デジタル回線に変更予定。

上記以外にオフコン、ミニコン、PC、WP等が何台かあり？

現在の問題点

第三次オンライン開発に伴うバック･ログの増大

若年層･女性及び外注の増加に伴うスキル不足

システム開発サイクルの短縮化

ユーザ･ニーズの多様化

人員増加に伴う管理の難しさ

マルチ･ベンダーになり用語、環境、操作等の違い

エンド･ユーザ･コンピューティングに於けるEDP部門のサポート方法

保守用DD(VEGA)概要

内容

開発済みのソース･プログラムから、使用しているレコード･ファイルの名称やフィールドの名称、桁数、データ形式などの情報を抜き出し、TSX1100のデータ･ディクショナリに反映させる。
このDDを検索することにより、既存システムを変更する際の影響度分析を機械的に行えるようになり、システム保守作業の効率化が図れる。

1. 目的
2. 構築環境
3. 構築方法概要
4. 構築後のDD状況
5. DDへのロード実行時間
6. 保守用DD利用上の留意点

昭和 62 年 12 月 1 日

作成者 日本ユニバック

氏名 坂口 健二郎

所属 開発第二部基本システム課

氏名 田中 一夫

YCC

㈱ 山一コンピュータ･センター

1. 保守用DD(VEGA)構築の目的

現行システムで使用しているデータ項目を登録し、プログラム(COBOL)との関連を設定することにより、

・項目変更により影響を受けるプログラムの検索

を可能とすると共に、保守用DD構築の過程で得られた情報及びDDより

・データ項目の標準化作業支援

を目的とし構築する。

1.1 保守用DDによる検索

現行システムの保守支援及びTSX1100のフィーズビリティ・スタディを兼ねて構築を行う。又、このDDは保守用DDとして固定させ、このDDの情報を変更して開発用DDとはしない。
さらに、保守用DDを常に最新の状態に保つ為、現行プログラムを変更する時、保守用DDに反映させる。

開発用DD(ZEUS)とはTSX1100の本来の目的であるソフトウェア・データベース化であり、システムの開発・保守・管理に関するすべてのソフトウェア資源を、統合開発支援データベースとして蓄積・更新し、これを標準化したディクショナリーによって制御することによって、生産性・保守性・管理性の高い環境を実現する。
さらに、これらのソフトウェア資源を活用し、また開発の工程間で情報を受け渡し再利用することによって、生産性・信頼性を向上させる。

1.2 保守用DDによるデータ項目の標準化

今後、開発用DDを構築するが、基本となるデータ項目はこの保守用DDを参照して、データ項目の正規化を行う。
保守用DDには同名異義項目(表示あり)、異名同義項目(表示なし)が多数あると思われるので、これの正規化を目指す。

2. 構築環境

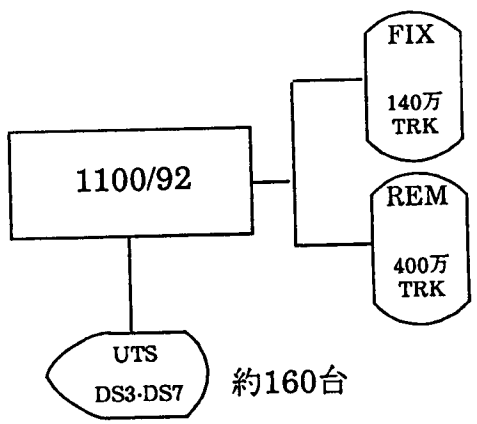
2.1 実施日、使用マシン、対象データ

・昭和62年11月19日(木)~22日(日)

・山一開発ホスト　1100/92　　メモリ 16MW

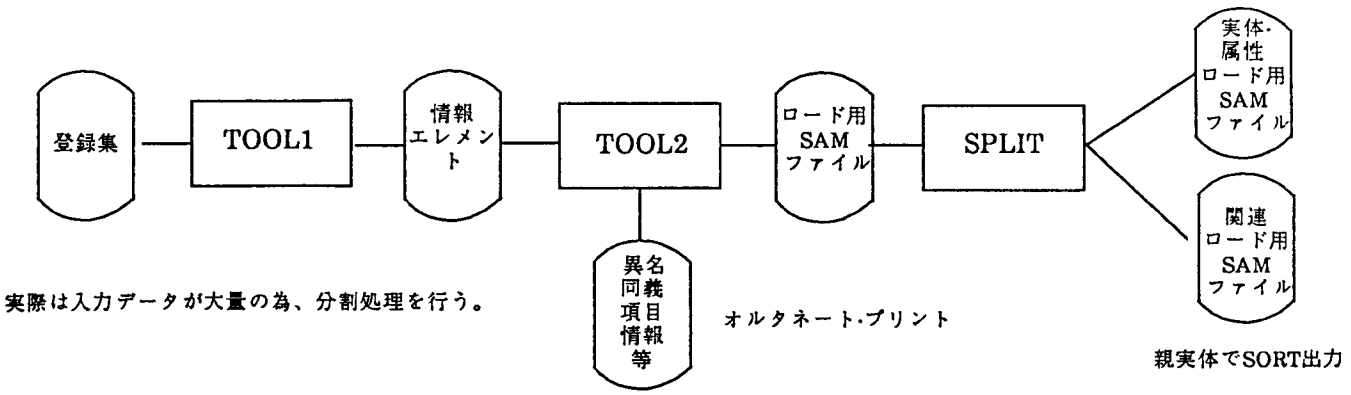
・通常の開発作業と並行して実施(但し、休日は、ほぼ独占)

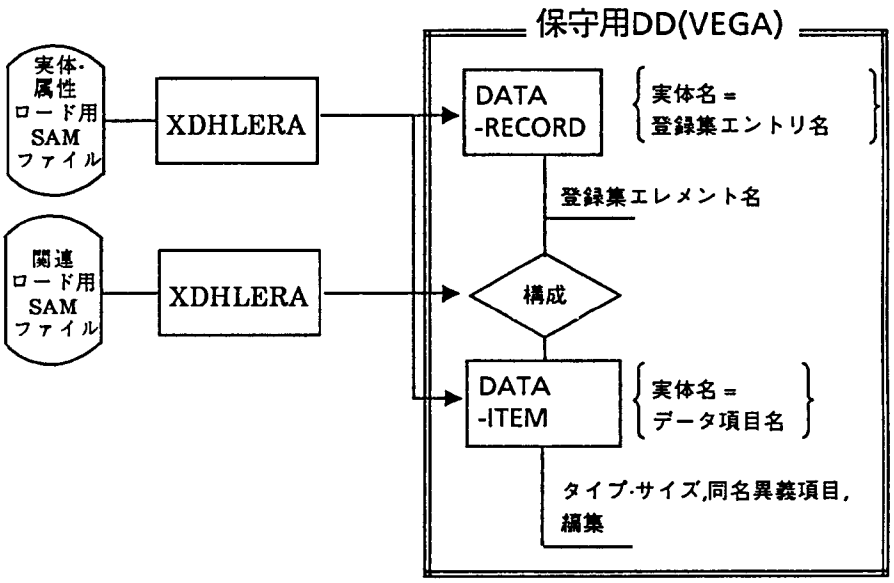
・現在本番運用しているプログラムの一部を対象とした。



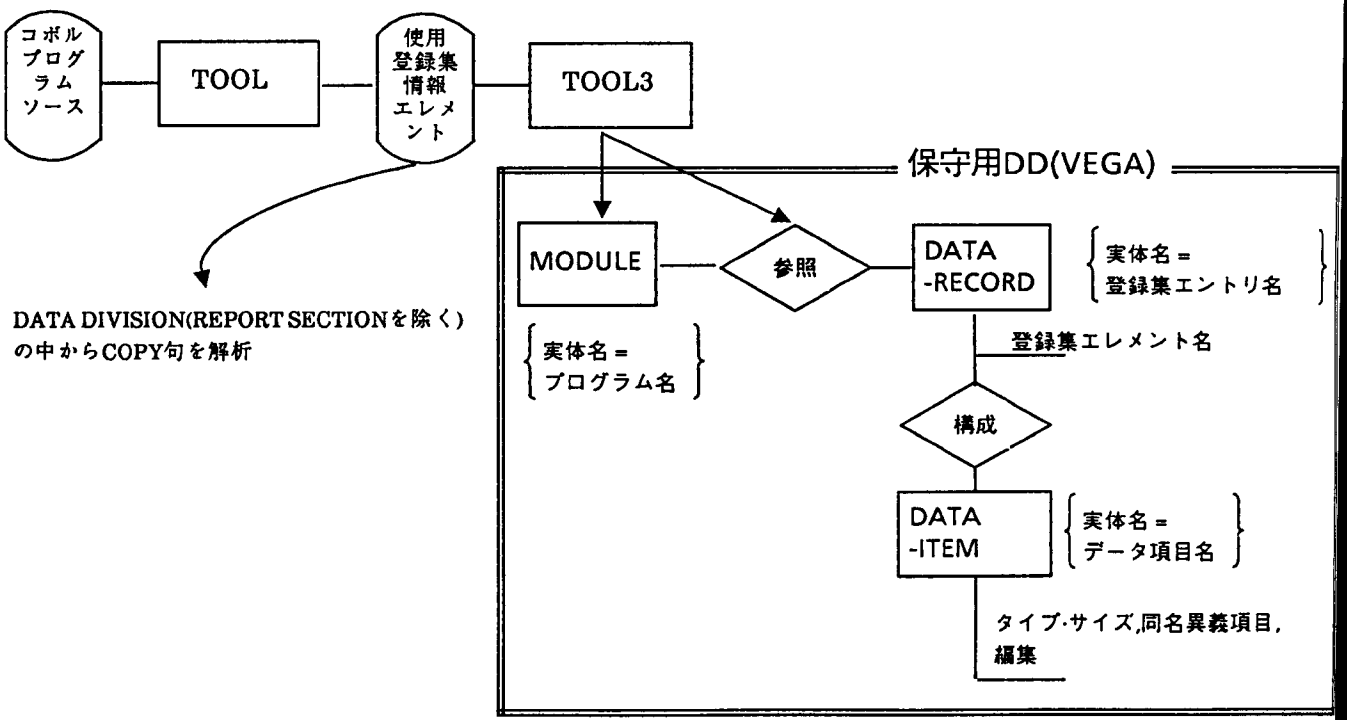
3. 構築方法概要

3.1 登録集エレメントを入力とし、入口名と使用データ項目の情報を蓄積する。



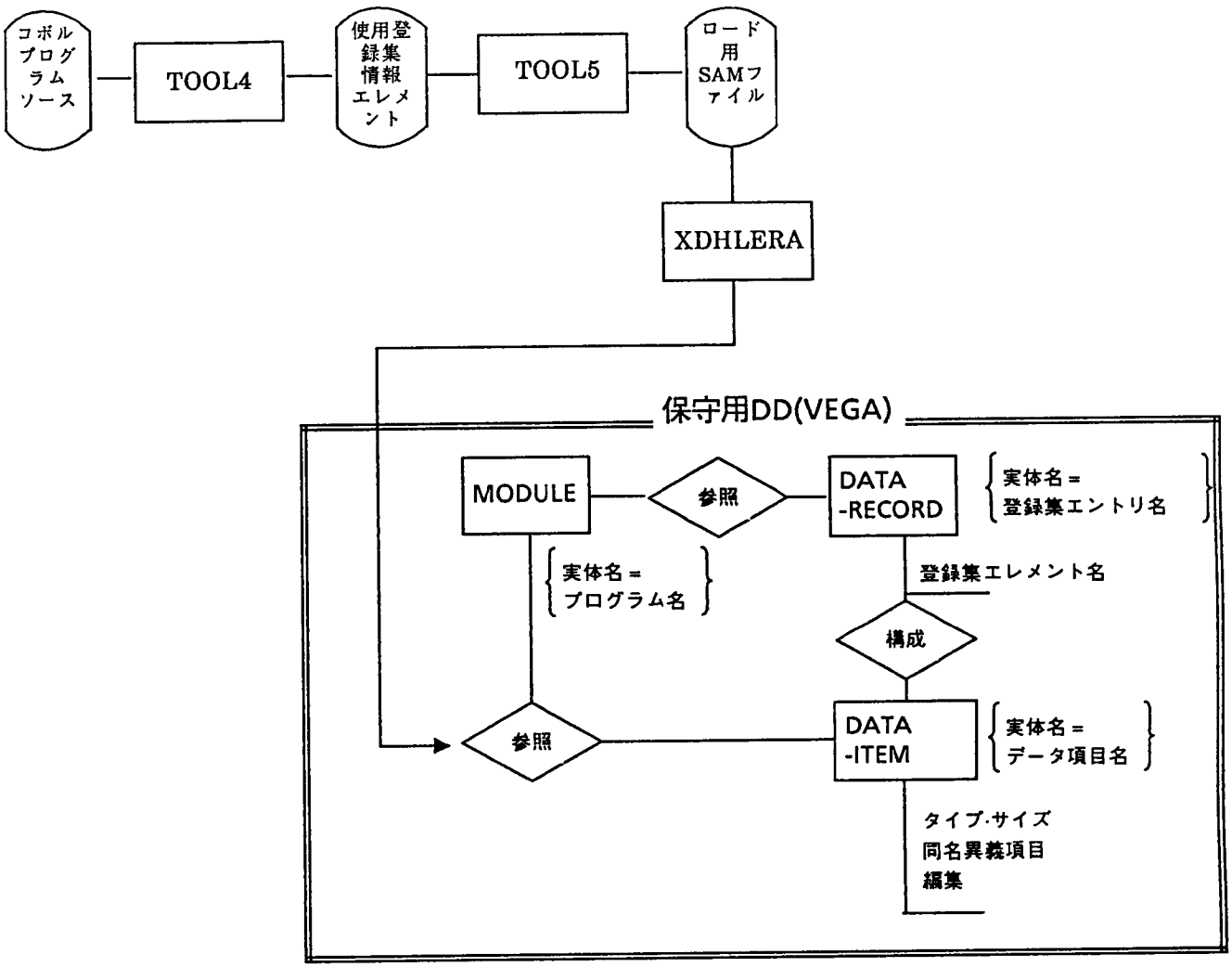


3.2 プログラム・エレメントを入力とし、使用している登録集の情報を蓄積する。



3.3 プログラムが使用しているデータ項目の情報を蓄積する。

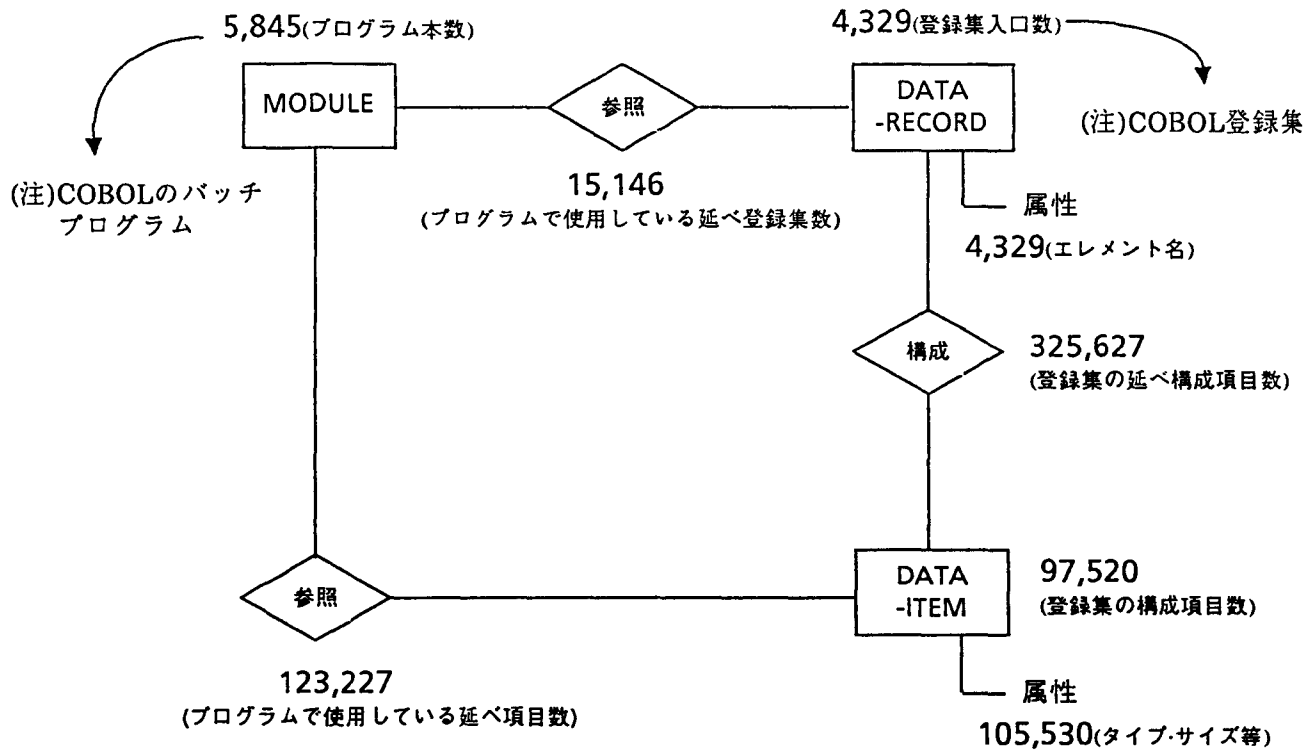
(注) PROCEDURE DIVISION以降で使用しているデータ項目のうち、登録集で定義されている項目との関連を蓄積する。



前述、3-1~3-3の順に実行する。

実際はデータ量が多い為、各ステップは複数RUNに分割して実行している。

4. 構築後のDD状況



	実体	関連	属性
MODULE	5,845		
DATA-RECORD	4,329		4,329
DATA-ITEM	97,520		105,530
MODULEとDATA-RECORD		15,146	
MODULEとDATA-ITEM		123,227	
DATA-RECORDとDATA-ITEM		325,627	
小 計	107,624	464,000	109,895
大 計	**681,483**		

参考) ① 同名異義項目(同一項目名でタイプ、サイズが違う項目)の数 = 8,010

② 同名異義項目でなくて複数の登録集の構成項目となっている項目の数 = 40,208

③ 単独項目数 = 97,520 - (8,010 + 40,208) = 49,302

④ 一つのプログラムが使用している登録集の平均数　15,146 ÷ 5845 ≒ 2.6

⑤ 一つのプログラムで使用しているデータ項目の平均数　123,227 ÷ 5,845 ≒ 21
(データ項目は登録集の構成項目となっているものが対象)

⑥ 一つの登録集の構成項目の平均数　325.627 ÷ 4,329 ≒ 75

5. 保守用DD利用上の留意点

主目的である項目の変更によるインパクト調査を行う上で、現在のDDでは次の点に注意が必要である。

(1) インパクト調査で指定した項目が集団項目の構成項目で且つ、プログラムではその項目は使用してなく、集団項目のみを使用している場合は、調査から抜けてしまう。

例.

```
PROC*                                   PROG1
     01  A.
         02  B   PIC     X(2).              COPY   PROC.
         02  C   PIC     X(2).
                                            MOVE  A   TO AA.
```

Bのサイズを X(2)から X(5)に変更した場合
Bでインパクトを調査しても、PROG1は対象から抜けてしまうが、本当は影響を受ける。

原因
DDでは登録集の構成項目について階層構造を表現していない為

対策
Bの項目が変更になった時、Bを構成項目にもつ登録集エレメント名はDDを検索することにより分かるので、その登録集エレメントのソースイメージから上位の集団項目名を調べ、その集団項目をインパクト調査の対象項目(この例ではA)として指定して調べる。

YCC作成の検索ツールではBを指定した時、PROG1は影響を受ける可能性のあるプログラムとしてフラッグをセットして出力される。

(2) プログラムのWORKで定義している項目と同じ名前の項目が、このプログラムで使用している登録集以外の登録集の構成項目となっている場合は、この項目でのインパクト調査を行うと対象となってしまう。

例.

```
PROC3*                                  PROG1
     :
     W-STATUS-CD  PIC   X(2).               COPY   PROC1.
     :                                      COPY   PROC2.
                                       WORKING
                                            :
                                            W-STATUS-CD  PIC X(5).

                                            MOVE  A   TO W-STATUS-CD.
```

原因
プログラムと項目間の関連を下記仕様でつけている為、
→PROCEDURE DIVISION以降で使用されている項目の接頭語を外し、その項目名を実体名とするDATA-ITEM実体が存在していれば参照の関連を設定する。

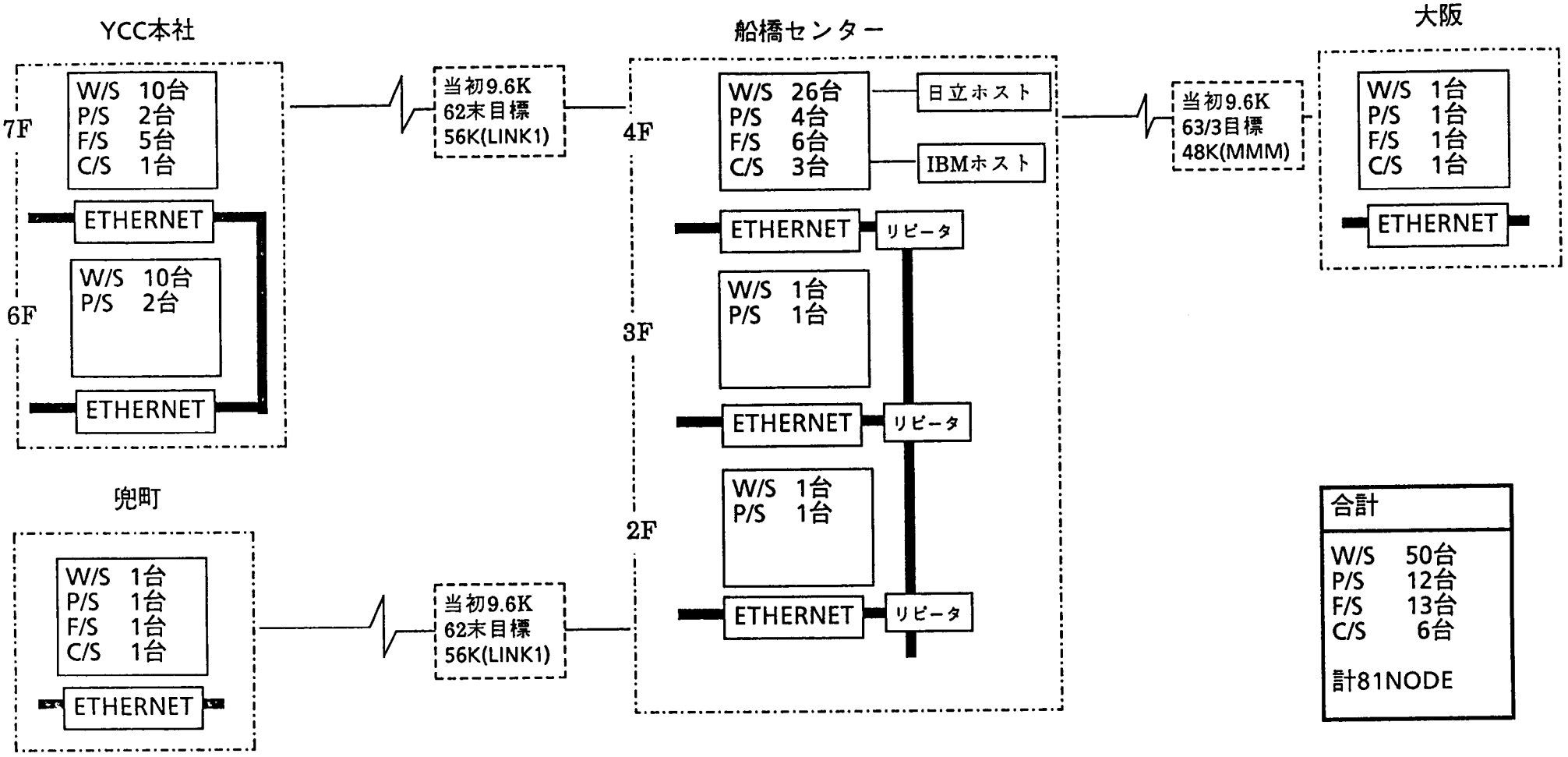
対策
YCC用検索ツール(新規作成)にて下記チェックを行い、WORKINGで定義された項目については表示しない。
→インパクト調査項目として指定された項目が、使用されている(MOD→◇→D-I間に関連が存在する)プログラム中でCOPYしている登録集(MOD→◇→D-R間の関連より調べる)の構成項目となっているか(D-R→◇→D-I間の関連より調べる)で判定する。どの登録集の構成項目にもなっていないものはWORKINGで定義された項目であり、このプログラムは影響を受けない。

ドキュメントのJ-Starに於ける適用領域

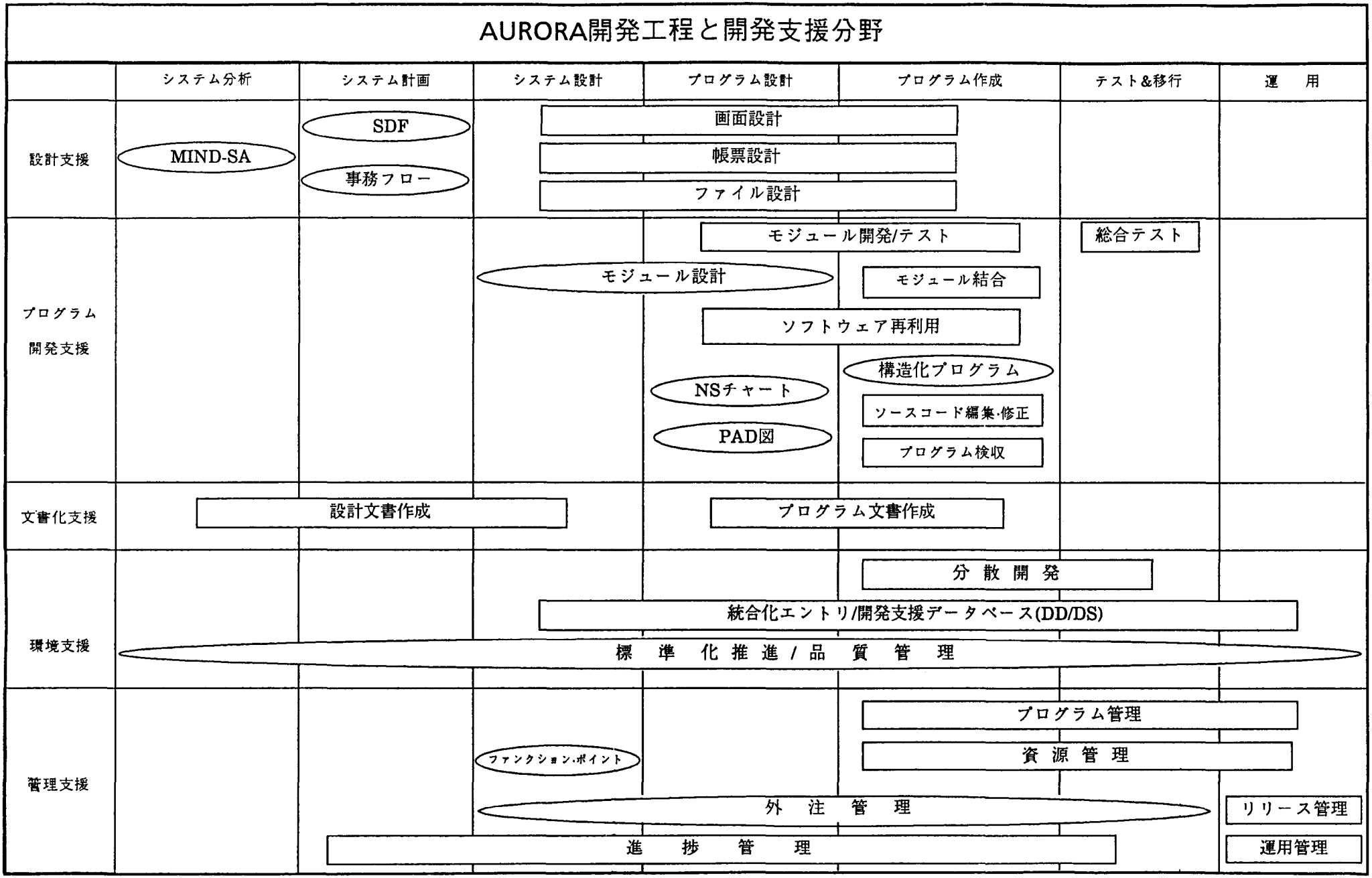
	システム分析	システム計画	システム設計	プログラム 設計	プログラム 作成	テスト&移行	管理
J-Star適用範囲	←……→	←――→	←――→				←――→
手書き			←――→	←――→	←――→	←――→	
ホスト出力(保守用)			←――→	←――→	←――→	←――→	←――→
ワークシート種類(比率)	51(33.6%)	16(10.5%)	51(33.6%)	11(7.2%)	12(7.9%)	11(7.2%)	

- 分析·計画·設計(一部)フェーズは重要なドキュメントが多く、そのワークシート種類の比率も高い(77.7%)ので、特にこの部分にJ-starを適用する。
- システム設計(一部)フェーズ以降は、設計用より保守用の意味合いが濃いので、J-Star作成とはせずホスト出力とする。(日立のADCAS,ユニバックのTSX-DOC使用)

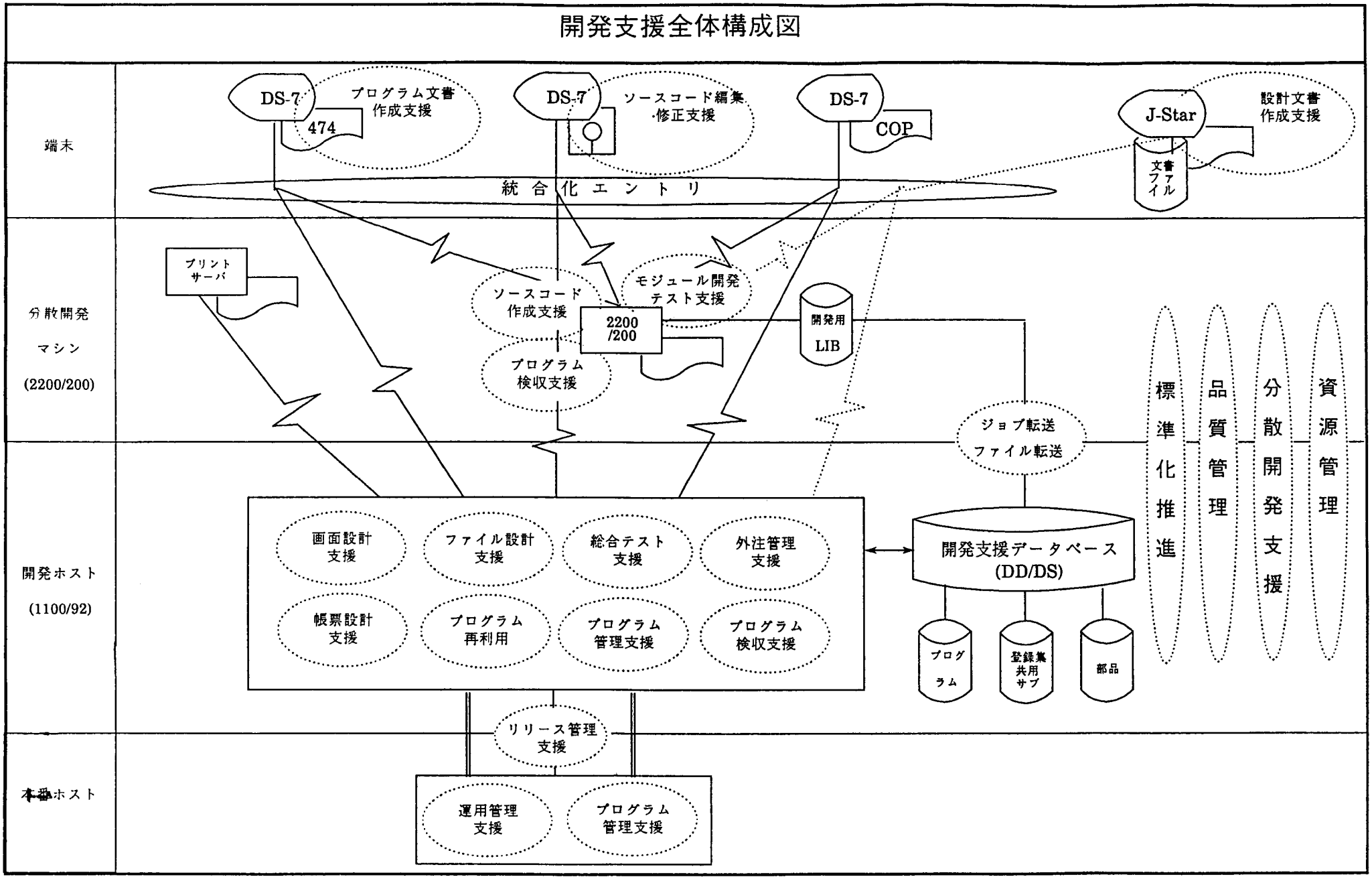
J-StarIIの構成と設置場所



AURORA開発工程と開発支援分野



開発支援全体構成図


端末
分散開発
マシン
(2200/200)
開発ホスト
(1100/92)
本番ホスト
DS-7
プログラム文書
作成支援
474
DS-7
ソースコード編集
・修正支援
DS-7
COP
J-Star
設計文書
作成支援
文書ファイル
統合化エントリ
プリント
サーバ
ソースコード
作成支援
モジュール開発
テスト支援
2200
/200
プログラム
検収支援
開発用
LIB
ジョブ転送
ファイル転送
画面設計
支援
ファイル設計
支援
総合テスト
支援
外注管理
支援
帳票設計
支援
プログラム
再利用
プログラム
管理支援
プログラム
検収支援
開発支援データベース
(DD/DS)
プログラム
登録集
共用
サブ
部品
リリース管理
支援
運用管理
支援
プログラム
管理支援
標準化推進
品質管理
分散開発支援
資源管理

日立機に於ける開発方式

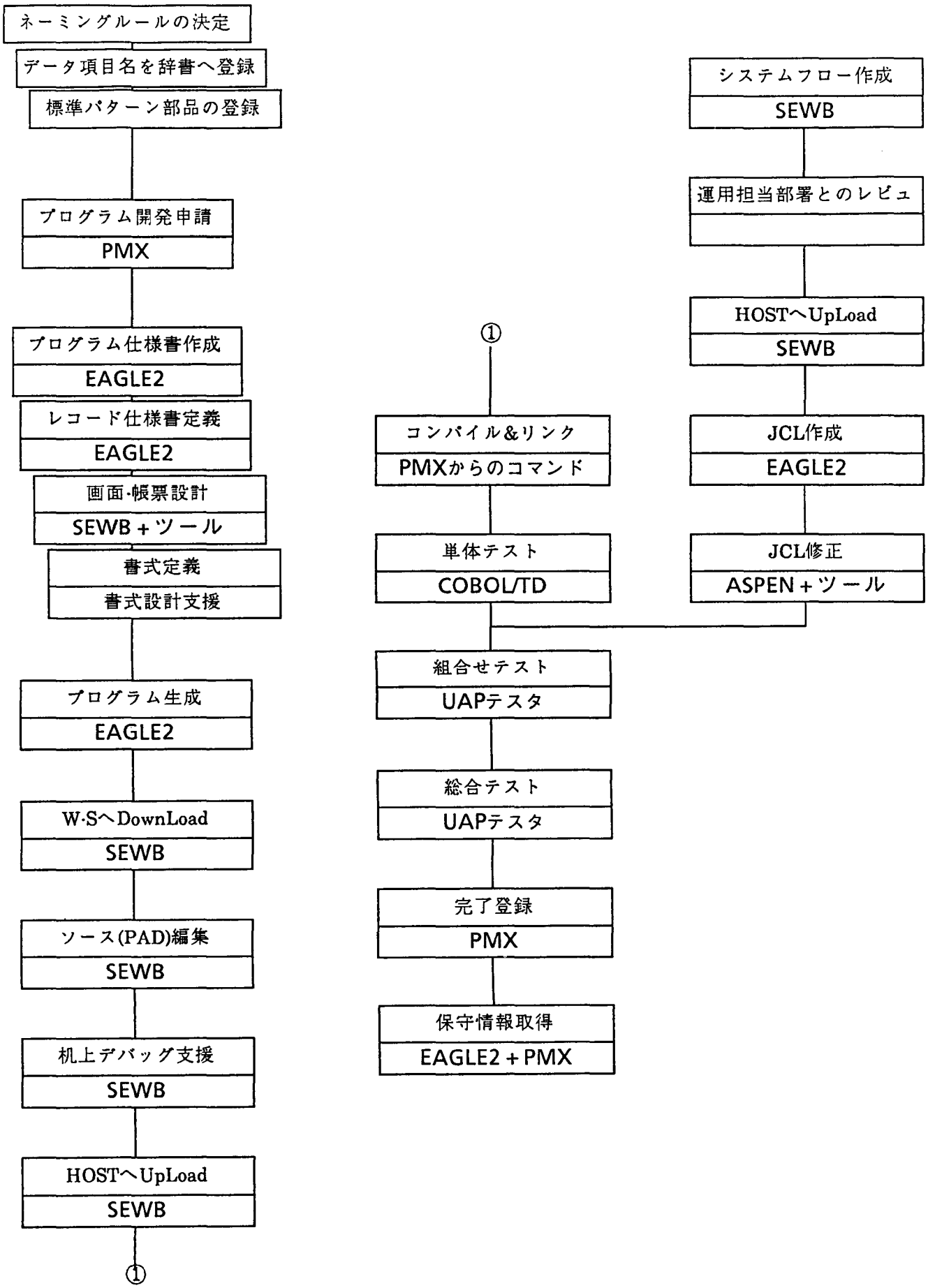
特徴 ・データ･デクショナリィによる項目名称の一元管理
・パーソナルCADを利用した書式設計の効率化
・標準パターン部品の利用によるプログラムの標準化･均質化とプログラム作成効率の向上
・SEWBによる分散開発環境を導入し、システム開発担当者にとって快適な開発環境を提供する
・組合せテスト、総合テスト支援ツールによるオンライン･プログラムテストの効率化

開発方式と手順

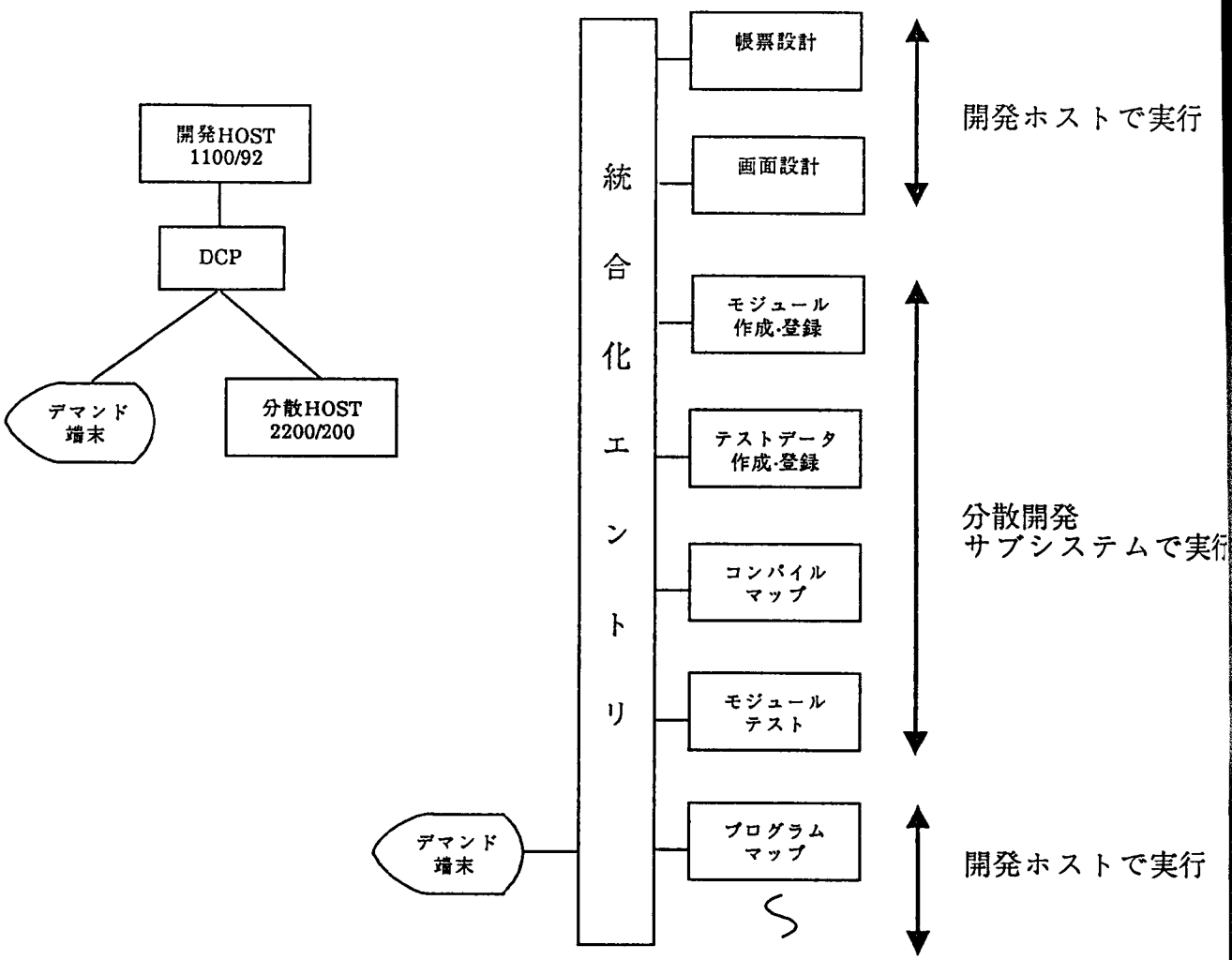
開発支援システムの構成を以下に示す。

開発支援システム	2050 SEWB	T560/20,2020,2050 EAGLE2	T560/20,2020,2050 支援ツール
システム分析計画	･画面･帳票定義 ラフスケッチ テスト表示 ･画面遷移シミュレーション		
システム設計	･画面･帳票定義 ラフスケッチ テスト表示 ･画面遷移シミュレーション ･システムフロー作成 運用とのレビュ JCL作成	･データ項目の登録/更新 項目名称の一元管理 ･ファイル仕様書定義 ･レコード仕様書定義	･画面･帳票定義 YCCローカル部分 ･書式定義 CAD方式 テスト印字
プログラム設計作成	･COBOLソース編集 ･PAD編集 ･COBOL↔PAD自動変換	･プログラム仕様書定義 ･プログラム生成	･マップ生成 ･書式オーバレイ生成
テスト	･COBOLソース実行テスト ･PAD実行テスト	･単体テスト COBOL/TD ･JCL生成	･組合せテスト ･総合テスト ･テストデータ作成 ･テスト結果編集･表示

日立機に於ける開発手順



ユニバック機に於ける分散開発



1) 利用者は統合化インターフェースによって作業(処理)指示を行う。
ジョブ転送、又はファイル転送の必要な処理を指示すると、パラメータ入力が要求され、それに応答する事によって実行される。
2) 帳票設計、画面設計の処理はジョブ転送によって開発ホスト側で実行される。必要に応じて、結果を分散開発コンピュータにファイル転送を行う。
3) ソフトウェア再利用システムをアクセスし、目的とするソフト部品の原形を自分のワークファイルに読み込み、それを基にモジュールを作成する。
4) テスト·データを作成し、モジュールをコンパイル、マップし、モジュール·テストを実行する。
5) モジュールのリロケータブルを開発ホストにファイル転送し、以降のフェーズでの作業はジョブ転送によって実行する。
6) 接続ホストの切り替え操作による開発ホストへのデマンド·アクセスは可能である。
7) システムの整合性を保つための共通ファイルの同期処理の仕組み·運用を行う。
8) プログラム·モジュールの管理は開発ホストにて行う。

3rd SEA Environment Workshop Position Paper

氏名	山浦 恒央　　種別 ☑会員 no 871181 □一般
所属	日立ソフトウェアエンジニアリング
住所	〒(244) 横浜市 戸塚区 品濃町 549-6
TEL	045(824)2111 ex.2515　　FAX 045(824)2729

仕　事

Σシステムの開発

作業環境

現在の問題

- ドキュメントそいかん整備ぜんか
- バージョン管理の問題
- ソフトウェア資源の保守方法

次ページの討論テーマ票にも，必要事項をご記入の上本紙とあわせて 12 月 15 日まで（締め切り厳守！）に下記の SEA 事務局までお送りください．もちろん，本式の Position Paper も同封していただくほうがベターです（既発表の論文を添付していただいても構いません）．

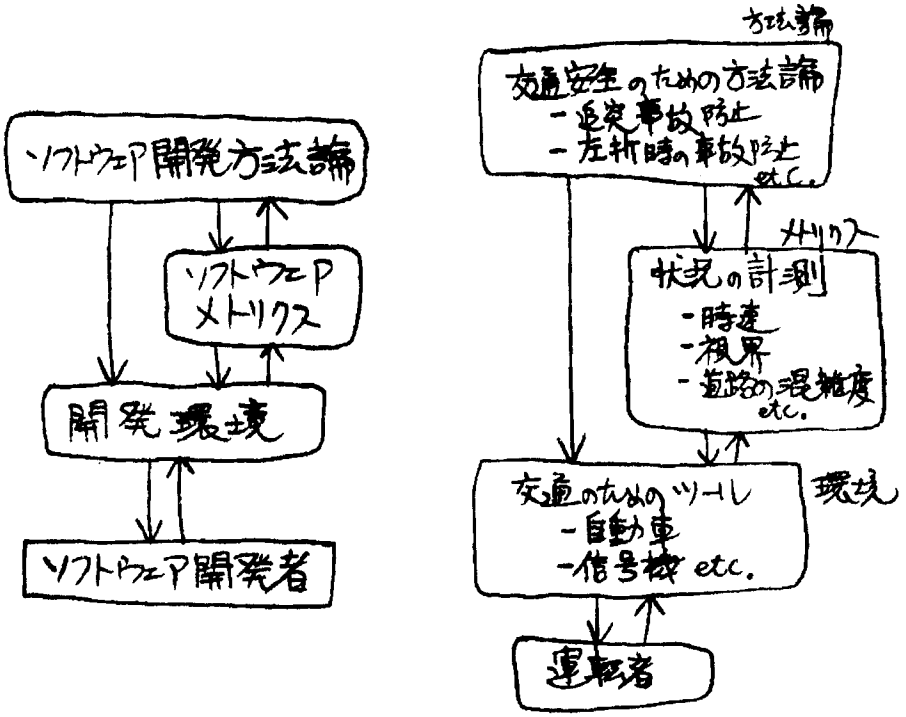
ソフトウェア技術者協会 (SEA) 〒102 千代田区単町2-12 藤和半蔵門コーアビル506

TEL: 03(234)9455　FAX: 03(234)9454

会員# 871181　山浦 恒央
日立ソフトウェアエンジニアリング (045-824-21

開発環境，開発方法論，ソフトウェアメトリクス の 三身一体 (SOFTWARE TRINITY)

現在 何十という開発環境が提案されているが，その実体は 単なるツールの寄集め (just a bund of tools) という感が否めない。個々のツールは 確かに すばらしいものではあるが，ツールをどのように使って 何をどうするか が不明確であるため 単なるツールの土俵を出ることができない。従って開発環境 以前として その開発者が どのような ソフトウェア開発方法論を考えているのか，またその開発環境は ソフトウェア開発方法論をどのように実現しているのかが非常に重要であると考える。また開発環境と ソフトウェア開発方法論の橋わたしをする ソフトウェアメトリックス 及び ソフトウェアメトリックスによる フィードバック も同時に考えなければならない問題である。このことは 交通安全上の諸事を考えても 明白である (図参照)。



今回のワークショップでのセッションとして データベース化，LAN化などがあるが，それらは 方法論の具体化として論じられるべきものであり，開発環境の第1次として現われる項目では ない。平均的プログラマの生産性，プログラムの品質等を1桁向上させるには 単なる思いつきのツールではなく 総合的な方法論に立脚した開発環境が必要である。歩行 (時速4km) 自動車 (時速40km) 飛行機 (時速400km) ロケット (時速4000km) のように スピードに関しても 新しい概念の導入により 1桁向上が可能となった。現在の開発環境を見ていると いかに速く歩行させるかに重点が置かれ 新しい概念である自動車については 考えられていないようである。ソフトウェア開発環境，ソフトウェア開発方法論，ソフトウェアメトリクスは 各々単独に存在するのではなく，同時に総合的に考慮されるべきものである。

3rd SEA Environment Workshop Position Paper

氏名 桜井 麻里	種別 ☑会員 no 85009 ☐一般
所属 (株)ソフトウェア・リサーチ・アソシエイツ 環境開発部	
住所 〒(102)東京都千代田区平河町1-1-1	
TEL 03(234)2611	FAX 03(262)9719

仕　事　「小規模事務アプリケーション開発支援ツール」のメンテナンス

約5年ほど前から、あるオフコン・パッケージやさんと共同で「『オフコンやパソコンで動くような定型化されているが少しずつちがう小規模事務アプリケーション』の開発を設計からソースコード(COBOL)生成まで総合的に支援する環境」を構築してきた。2年半前に最初のバージョンがリリースされ、1年ほど前から本格的に動いているものである。

その後、このツールは最初のユーザのところではどんどんカスタマイズされ良く使われているようであるが、他の所には導入されていない。

せっかく開発したツールなので、ほかの所にも使ってもらいたい。その為に使いやすくするということが現在の私の仕事である。

ちなみに、このツールはUNIX上に構築されている。

作業環境　UNIX 4.2 BSD + X window V10R4 + NFS + かな漢字変換機能付き端末エミュレータ

このツールのソース、ライブラリ、バイナリイ、ドキュメントは2台のSONY NEWS (フォーマット時のディスク容量156M)に分散して置いてある。その内1台のほうからは、すべてが一つのツリーに見えるようにNFSを使ってマウントしてある。現在このツールは実質2人で面倒を見ていて、どちらも自分の机の上にNWS-830とそのビットマップディスプレイが在る(必ずしもこのツールのリソースが置いてあるマシンと同一ではない)。

この2台は、当然、社内のLANにつながっている。

このプロジェクトの開発初期には日本語環境が使えるUNIXがなかったため、そのころに書かれたドキュメントは紙で残っているが、操作マニュアルやメンテナンス用のドキュメントはすべてroffでつくった。出力イメージを確認するためにnroffのプレビュアを使っている。ドキュメントを気持ちよく作る環境も徐々にではあるが、整いつつある。

現在の問題

①このツールを使ってもらうための問題点として:

他人に対して説明しにくい。わかりにくい。使いにくい。導入時の負担が大きすぎる。

だから、なかなか導入してもらえない。どうすればよいか。

②このツールをメンテナンスする側としての問題点:

いろいろできるような仕掛けを作ったためにシステムが複雑になってしまった。

今後予想されるカスタマイズに対してどう対応していけば自分たちもユーザも楽になるか。

③バージョン管理が難しい。

この手のツールは、データベースのスキーマやスケルトンなどの変更を避けられない。

それらのバージョンとツールのバージョン、そして蓄積してきた定義情報との対応、ならびに生成されたソースとドキュメントのバージョンをうまく管理するのが問題である。

など.

次ページの討論テーマ票にも，必要事項をご記入の上本紙とあわせて 12 月15 日まで（締め切り厳守！）に下記のSEA 事務局までお送りください．もちろん，本式の Position Paper も同封していただくほうがベターです（既発表の論文を添付していただいても構いません）．

使ってもらえる開発支援環境に変身させるための構想

― 小規模事務アプリケーション開発支援ツール『Zodiac』を例に ―

(株)ソフトウェア・リサーチ・アソシェイツ
環境開発部

桜井 麻里
E-Mail: mari@sra.junet

ABSTRACT

世の中のソフトウェア開発支援ツールはどれも使うために開発される筈である。しかし、一生懸命作って見たけど結局使われていないというケースが多いように見受けられる。せっかく苦労していいものを作ったのなら、使ってもらいたいと思うのが人情というものである。

筆者の所にも、機能的には優れているのに余りの使いにくさに使ってもらえないツールがある。小規模事務アプリケーション開発支援ツール Zodiac である。使ってもらえないのは哀しいから、使ってもらえるように変身させようという試みを始めた。

このポジションペーパーは、その Zodiac を例に、使ってもらえない理由を考察し、事務処理アプリケーションの開発環境全体を支援するツールとして使ってもらえるようにするための今後の構想とその問題点を述べる。

1. 背景

ソフトウェアの生産性向上ということを旗印に、いわゆるソフトウェア・データベースを構築し、ソフトウェアの再利用並びに自動生成を推進しようという試みが、数年前から各地でなされている。筆者の所でも、支援対象を小規模事務アプリケーションの世界に絞り、設計から COBOL ソースコード生成までを一貫して支援するツール Zodiac をあるオフコンメーカーと共同で開発してきた。２年ほど前から一応動いているとは言うものの、その最初のユーザー(つまり、依頼人)にしか使ってもらっていない。そこでは、自分たちのためにどんどんカスタマイズを続け、使い易いものになっているようであるが、ソフトウェアハウスである筆者の所には裸のままの使いにくい Zodiac が取り残された状況になっている。

一方、近年の技術の目覚ましい進歩により、メモリーとディスクのたくさんついた UNIX ワークステーションとビットマップディスプレイ、そしてその上で動くウィンドウシステムが、組織内ネットワークの中で一人一台という形で使えるようになってきた。つまり、開発環境支援ツールの再構築をするための道具立てはできてきたのである。

そこで、他の所にも使ってもらえるように使い易くしていこうという構想を練っているところである。本題に入る前に、簡単に Zodiac の紹介をしておきたい。

1.1. Zodiac とは

Zodiac は、小規模事務アプリケーションの開発作業全般を支援することを目的に開発された統合環境である。

その対象ユーザーは、定型パターンの存在する似たようなアプリケーションをたくさん作り、エンドユーザの要求に合わせて細かい修正を繰り返しているような所 (たとえば、オフコン・メーカーとかディーラ)を想定している。

その基本思想は、以下に示す通りである。

(1) システム内のデータの相互矛盾を排除する。
同一のＩＤを持つデータが、システムのどこに現われても、同じ意味を持つ様にする。即ちデータ項目の管理をＤＤ[1] により行ない。各種の項目属性が同一に保たれる様にする。

[1] ＤＤとは、データディクショナリのことである。

(2) 要求定義～プログラミング～テスト作業を通して ON-LINE 化を図る。
通常の設計作業は、その大部分を単純な転記に依存している、そのため転記時の誤りが発生しやすく、再利用も困難である。Zodiac では、すべての設計情報を ON-LINE/DB によって管理し、容易に情報のアクセスが可能になるようにする。

(3) プログラム（COBOL）の自動生成を行なう。
処理の内容によって、いくつかのパターン[2] を用意し、設計情報から必要な事項を取り出すことによってソースコードの自動生成を行なうようにする。

(4)ドキュメントの自動生成を行なう。
設計情報はＤＢに格納されているので、通常必要となるドキュメント類は適当にフォーマットを決めて出力することができる。

(5) 管理データの収集と活用を支援する。
現在、開発がどの程度進んでいるかの情報をＤＢから抽出して報告するようにする。

1.2. 現在の Zodiac

以上のような基本思想のもとに作成されたのであるが、現在の Zodiac は、その全部を完全にサポートしたものではない。しかも、そのための基本的な機能を持ったツールのよせ集まりであるにすぎない。それぞれのツールは UNIX のコマンドとしてシェルから独立に起動されるようになっている。具体的には、次のようなツールからなっている。

A. 設計支援系
- ·エントリイ管理ツール
- ·ＤＤ登録ツール
- ·帳票定義ツール
- ·画面定義ツール
- ·ファイル定義ツール
- ·ボリューム定義ツール
- ·更新定義ツール
- ·処理定義ツール [3]

B. プログラム生成系
- ·プログラム定義ツール [4]

C. ドキュメント生成支援系
- 各種ドキュメント生成ツール群 [5]

2. 使ってもらえない理由

使ってもらえないのは哀しい。だから、その理由を考えて今後の改善を図りたい。

開発支援ツールを使うユーザーがやりたいことは、ツールを使うことではなくツールを使ってアプリケーションを開発することである。そういう見方で、Zodiac を見直して見ると、使ってもらえない理由としては、大きく分けると以下の三つになると考えられる。

(1)使いこなすためには、導入してから運用するまでのスタッフの負担が大き過ぎる。

(2)いろんな作業フェーズに対するトータルなサポートに欠ける。

(3)全体として使いにくい。

それぞれを、もう少し詳しく説明すると次のようになる。

2.1. 本格的に使うまでのスタッフの負担

日本語 UNIX 上に構築されたツールであるから、まず日本語 UNIX を使えるようにしなければならない。さらに、X ウィンドウや NFS も今や不可欠である。もし、UNIX ワークステーションを多数用意できたとしても、その運用·管理が容易ではないことはいまさらいうまでもない。それだけでも、いままで COBOL の世界にどっぷり使っていた人達にとっては相当な負担であろう。

しかし、Zodiac を導入し、本格的に使うまでの負担はそれだけではない。組織内におけるアプリケーション開発の標準に基づいて、Zodiac のカスタマイズを行うと共に、標準部品を作成[6] しなければならないのである。これが、実に大変な作業である。さらに、標準部品の使い方に関する説明書も書かなければならない。そのへんに関し

---

[2] いわゆるジェネレーションルールである。Zodiac では、このスケルトンと呼んでいる。おなじレイアウトでもこのスケルトンによっては全く違うことをするソースコードが生成される。このスケルトンを書くことはユーザーに任されている。そのことが Zodiac の良い所であるが、同時に導入を難しくさせている要因ともなっている。

[3] 処理の流れを定義するツールではなく、処理(プログラム)が使うファイルや帳票がどれかを定義するツールである。

[4] スケルトンにしたがって、設計支援系によって入力された情報やプログラマからの追加情報をもとに文法エラーのない COBOL ソースコードを生成する。

[5] データベース内の設計情報を書式化しレーザプリンタに出力する。それぞれのドキュメントに対応して、独立なコマンドが存在する。

[6] まず、既存のアプリケーションを幾つかのパターンにわけ、それに応じてスケルトンを作成しなければならない。これには、対象アプリケーションに関する豊富な経験が必要である。そして、データ項目名を標準化し、組織内標準のデータシクショナリを作成しなければならない。そして、プログラマが直接使うことになっている中間言語(PDL)のマクロや OWN コーディング集などを作成し、最終的には標準となるアプリケーションを最低一つはサンプルとして作成する必要がある。

て、現状の Zodiac ではなんのサポートもしていないのである。あるのは、「だれかが標準部品を作れば、アプリケーションプログラマは、それを利用することにより、レイアウトを定義して質問に答えるだけで、望んでいるような処理をする COBOL のプログラムが自動的に出てきますよ。」という機能だけなのである。もちろん、それだけでも素晴らしいことなので苦労して導入するだけの価値はあると思うのだが、スタッフの負担の大きさは導入をためらわせる大きな要因となっている。

もちろん、どんなツールであれ、それを導入し使いこなして行くためにはそれなりの努力が必要なのであるが、それも程度問題である。

2.2. いろんな作業フェーズに対するトータルなサポート

さて、すでに標準部品があるとして、Zodiac を用いたアプリケーション開発の標準的な作業の流れは、次の様になる。

a.受注

b.分析
ここでは、受注したシステムが過去に開発したもののどれに似ているかを分析する。どれにも似ていない場合は、Zodiac を使ってソースコードまで生成するのは困難である。過去に開発したシステムで、参考になるものがあれば、それをコピーする形で開発を進めればよい。特に重要なことは、既存のＤＤやスケルトンがそのまま使えるかどうかということである。

c.対象システムの登録

d.エントリイの登録
そのシステムで必要な帳票/ファイル/画面などのエントリイを登録する。登録しないと、レイアウト定義ができない。

e.設計系定義ツールを使ってレイアウト定義
·最初にＤＤ登録(たいていの場合、流用) ·ファイル定義と処理定義·そのあと、帳票定義と画面定義および更新定義

f.生成系ツールを使ってプログラム定義および生成

g.デバッグ、テスト

i.納品
ドキュメント生成

j.保守

x.トラブル処理

y.ディスク、MT などの資源管理およびバックアップ

z.いわゆる運用管理

以上の様になる分けであるが、残念ながら現在の Zodiac がまともにサポートしているのは、e,f だけである。そのほかの部分は、データベースのメンテナンス用のプリミティブなコマンドが存在しているだけという状態である。筆者らが Zodiac のメンテナンスをして行く上では、それらともともと UNIX にあるコマンドだけで必要十分だったからそうなったのである。

しかし、Zodiac を使ってアプリケーションを開発しようとする人達は、今までやっていた作業のほとんどをそのツールを使って行おうとするのだから、もっと広範囲の総合的なサポートが欲しい筈である。これは手作業、あれは Zodiac をつかって、その結果を紙に出して目で見て、そして＿などとやっていると段々面倒臭くなってくるのは自分のことを考えれば直ぐ分かる。だから、日常接しているものだから、なるべく全部を一つの世界で終わらせることができたらうれしいだろう。

そこで、ユーザー(アプリケーション開発者)の目で見ると足りないものがいっぱいあるということになる。とくに、b.分析, j.保守の部分は楽に使えるユーテリティを用意してあげないと、せっかくデータベースに保存してある情報が有効に使われないことになってしまう。また、再利用ができるように、設計情報と生成情報をデータベースにいれてあるのに、他のシステムを流用するための手段が繁雑なのもいただけない。

2.3. 全体としての使いにくさ

さて、なるべく全部を一つの世界でやりたいということは、それぞれの機能をもったコマンドやツールが存在すれば良いというだけでは不十分なのである。確かに、一つ一つのツールはなるべく単純で使い易いことが好ましい。それは、UNIX のツール·ボックスという考え方が多くの人に受け入れられていることをみても明らかである。だから、機能単位にツールを分けておくことは良いことである。そして、Zodiac の場合、確かに個々のツールはそれなりに使い易いように工夫されている。しかし、個々のツールがばらばらに存在しているだけで、全体としては使いにくいものになっている。

例えば、一つのことをやりたいだけなのに、そのために 5 個もユーテリティを組み合わせなければならなかったり、引き数を知りたいだけなのにわざわざ他のコマンドを実行しなければならなかったり、関連する情報も一度に見たいのにそれができなかったり、急に他のことをやりたくなった時、わざわざそのツールを抜けなければならなかったり、とにかく何かと不便である。

ユーザーの目で見て、使い易いように統合化していかなければ使いたくなくなる気持ちも良く分かる。

2.4. 使いにくさの元凶

使いにくさの元凶は、ツールを使う人の見方を考えずに、ツールを設計する立場の見方、場合によっては、ツールを実現する人の立場をそのまま使う人に見せているからである。そして、作る方のいいわけとして

·やればできるとわかっていることはすぐにはやらない

·使う方だってプログラマなんだから、好きなように直して頂戴よ、ソース全部あげるから_

という気持ちがあり、わかっていながら放置しておいたということは否めない。

3. 改善のための構想

理由は分かっているのだからそれを改善すれば良いのは明らかである。では、この状況をどのように改善して行くか構想を述べて見たい。

やらなければならないことは沢山ある。しかし、今はそれを全部やることはできない。そこで、次のような段階を経て、使ってもらえるような開発支援環境ツールにしていきたいと考えている。

第一段階

今あるツールに手を入れずに、ツールを統合化して全体として使い易くすることを目指す。後々ツールの追加·変更が予想されるのでそのことも考慮する。

第二段階

今あるツールはそのままにしておいて、足りないツールを用意し、日常の作業全般をサポートできるような総合的な環境を目指す。そのとき、第一段階で行った統合化環境の中にうまく溶け込ませるように工夫する。

第三段階

今あるツールを使い易く、かつカスタマイズし易くなるように根本的に見直す。この際、データベースの機構についても根本的に見直したい。

第四段階

サンプルとして提供できるような標準部品とそのマニュアルを試作し、導入側のスタッフの負担を軽減させることを目指す。また、分かり易い導入解説書やスケルトンの正しい作り方などの解説書も用意したい。

道は遠い。。

4. 全体としての使い易さを目指して

では、第一段階の「ツールを統合化して全体として使い易くすることを目指す」ためにどうすれば良いかを考えたい。

4.1. 目標

·やりたいことがすぐにやれること。

·やりたいことをやるための方法がすぐ分かること。

·ユーザーに見える部分は、抽象度のレベルが同じであること。

·見たい時に見たいものがすぐ見られること。

·矛盾を起こさないかぎり、やり方はなるべくユーザーの自由にできること。

たとえば、システムの設計を始めてからプログラムを完成させるための手順についても、トップダウンに考えられる時もあれば、ボトムアップに考えたい時もあるので、どちらからでもできるようにしたい。エディタにしても、仮名漢字変換にしても、なれているものを使いたいし、なれている方法でやりたいと思うのは、使う立場としては当然の要求である。

4.2. 基本的なアイデア

戦略としては、次のようなもの。

· 「もの」を主体にして、「機能」を選択できるようにする。つまり、以下のようにする。

(1)「もの」を主体にメニューを出す。

(2)何かを選ばれたら、まず、一覧とか構成図などを表示する。

(3)そして、その「もの」たちに対してできることをボタン表示してクリックされるのを待つ。パラメータが必要なものに対しては、一覧の中からクリックしてもらうとか、キーボードによる入力を待つ。

· 「もの」とそれに対してできる「機能」を全部並べただけでは、かえって、何をして良いか分からなくなってしまう危険性がある。それを少しでも和らげるために、その時の状態によって、やってはいけないとかできないなどの「機能」に関しては選べないようにしておく。

· ○○一覧を出している場面では、その一覧に対して、ただ全部を表示するだけでなく、フィールド指定によるソート及び grep などの加工ができるようにする。

· ある機能単位の中では、その手順が分かるようにガイダンス表示する。

ちょっと細かい話で恐縮ですが、戦術としては例えば次のようなもの。

· ○○定義、○○一覧、○○構成図など、一つのシステムの中の何かを表示する時は、ＩＤだけでなく対象システム名がわかるようにする。これは、今自分が扱っているのがどのシステムであるかを利用者に認識させ、他のシステムと区別を付けるためである。他のシステムを参照したりする時には、この機能はどうしても必

要になる。

- ユーザーからの入力を受け付ける時は、以下のような点に注意する。

  (1)あらかじめ選択の幅が決まっている時(SELECT ASK 形式)には、それらを表示しどれか一つを選べるようにしておく。

  (2)ID 入力などの場合には、選択対象のＩＤの一覧を表示し、入力の助けとする。できれば、クリックしただけで入力フィールドに ID が入った方がよい。

- 全てのウィンドウに対して、可能なかぎりカットアンドペーストをサポートする。

その他、アイデア募集中。

4.3. 実現のための方法

では、どのように実現するか。これが問題なのである。

現在のところでは、シェルと、酒匂氏の XME[7] に頼ろうと考えている。足りない部分はＣ言語でプログラムするしかないであろうが、それはなるべく避けたい。

4.4. 問題点

ツールをメンテナンス／カスタマイズする立場から考えると、問題点は山積みされているのだが、ここでそれらを一々述べることは避ける。

要は、これ以上 Zodiac システムを複雑で説明し難いものにしないようにしながら、使い易くするためのツール(コマンド)やライブラリを増やすためにはどうやって実現していけばよいかということである。結局の所、ユーザーにカスタマイズしてもらうしかないので、そのための機構と支援環境を今から考えて構築しておかないと、いつまでたっても自分が抜けられなくなる。

5. ワークショップに向けて

この構想は、まだ考え始めたばかりで、いろいろ悩んでいます。きっと、他にも似たようなことで悩んでいる人がいるであろうと期待し、ワークショップに参加することを希望しました。皆様の経験談とご意見をお聞きし、皆様と共に議論できればと思っています。

---

[7] X-Window 環境下で動作する簡易ユーザーインターファイス構築ツールで、SRA の酒匂氏が実験的に作成したもの。今回のワークショップのポジションペーパとして提出されている筈であるので、そちらを参照のこと。

第３回　SEA環境ワークショップ

セッション４

開発環境管理の実際

チェアマン　：臼井　義美（日本電子計算）
プレゼンテータ：山浦　恒央（日立ソフトウェアエンジニアリング）
：田中　一夫（山一コンピュータ･センター）
：桜井　麻里（ソフトウエア･リサーチ･アソシエイツ）
レポータ　：細野　広水（エム･ケー･シー）

１．はじめに

臼井：最後のセッションを受け持つ日本電子計算の臼井です．よろしくお願い致します．

討論テーマは大きく２つに分かれていまして，１つは分散環境に対する管理をどうしたらよいかという問題，もう１つは人を含めて開発環境をどう管理するかというマネージメントの観点です．

分散環境については，昨日のセッションでかなり討論されましたので，このセッションではできるだけマネージメントや開発環境を管理するためのツールに話を絞っていきたいと思います．

それでは，プレゼンテータの１人目は，田中さんにお願い致します．

２．プレゼンテーション

２．１．保守用ＤＤ（ＶＥＧＡ）

田中：山一コンピュータ･センターの田中です．よろしくお願い致します．

まず，簡単に今の環境を説明しますと，完全にメインフレーマです．メインフレームしか有りません．使っているのはＴＳＳです．ＵＮＩＸのような流行の世界は知りません．会場の皆さんからみると古い世界ということになります．言語も相変わらずコボルです．ユニバック，ＩＢＭ，日立いずれもコボルです．いろいろなツールは使っていますが，基本的にはメーカーのもの，そして自分たちで作ったものです．ただ，ユニバックは以前から使っていたので，（ＩＢＭ，日立は最近なので）ユニバックのツールが多いです．ただ，ユニバックはマイナーな世界なので，知らない人が多いと思います．

開発手順などを規定していますが，一般と同じだと思います．オーロラと名付けた開発手順に沿って工程を分けています．理由としては，人間が多いことで，社員，パート合わせて約１，０００人がプログラム開発をやっています．そこで分析からテスト／運用まで手順を作っています．１番困っているのは上流工程，つまり要求定義です．このあたりで，徹夜も有ります．理由は納期が厳しいこと．証券会社で，今だと税制改正の対応で，ハードな仕事です．いろいろなツールも使っていますが，あくまでもメンイフレームの世界です．ただ我々としては，これが精いっぱいの仕事だと思っています．今回お話したいのはこのあたりで，ＤＤ／ＤＳと書いてある部分をやりましたので，そこをお話します．（これは，昨日のセッションで話をする予定だった部分ですが，チェアマンからの指名の際に何故かＳＲＡの田中さんが出て，話の展開を面白くしてくれましたので言わずじまいでした）

ＤＤ／ＤＳはユニバックの上で運用されています．話のキッカケは昭和６１年ＮＴＴ株の上場の時，桁数が多くて納まらなくなりプログラムの修正が必要になったことです．その時はＣＯＰＹ－ＬＩＢ(登録集)などを全て人手で処理したわけですが，そんなやり方はどうも旨くないということで，保守用ＤＤを作ることになったわけです．ＤＤの目的は，人手で探していた情報を早く見つけられるということだけです．ともかく，いま動いているプログラムを対象にしています．データ項目は標準化してあるのですが，現状はメチャクチャなので，今後の参考にもしようという意味もあります．昨年１２月に１週間くらいで，プログラムの一部ですが，ＤＤを作っています．

ＤＤの作り方の概要は，まず，ＣＯＰＹ－ＬＩＢ（登録集）からデータを抜き出して，保守用ＤＤにデータレコードとデータアイテムを作るというものです．ユニバック

のＴＳＸというツールのＥＲＡモデルの概念を使っています．

次にソースプログラムを同様に処理して，保守用ＤＤに情報を入れます．最終的に欲しいのはコボルのソースプログラムとＣＯＰＹ－ＬＩＢのデータ項目との関連で，特定のデータ項をどのプログラムが使っているかということで，そのためにかなりの手がかかっています．

データの件数ですが，一部を処理したのですが，プログラムが５，８００本，登録集が４，２００本，データ項目は９８，０００件で，参照／更新等の関連は５５０，０００件に上り，属性は９２，０００で合計で７５４，０００件が登録されたわけです．但し，ネーミングの問題で，同名異義や異名同義などが多数あるので，今後の重要な課題です．

3. 構築方法概要

3.1登録集エレメントを入力とし、入口名と使用データ項目の情報を蓄積。

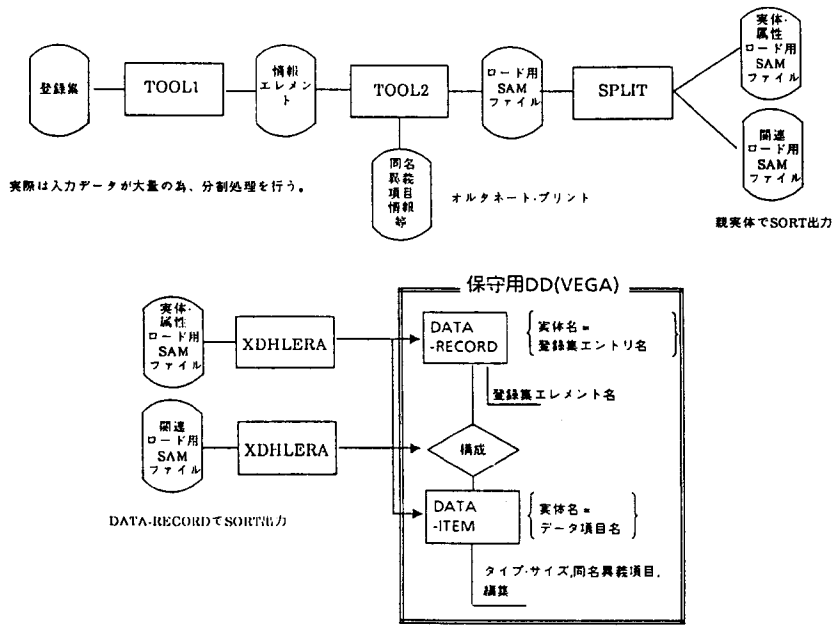


図２－１－１

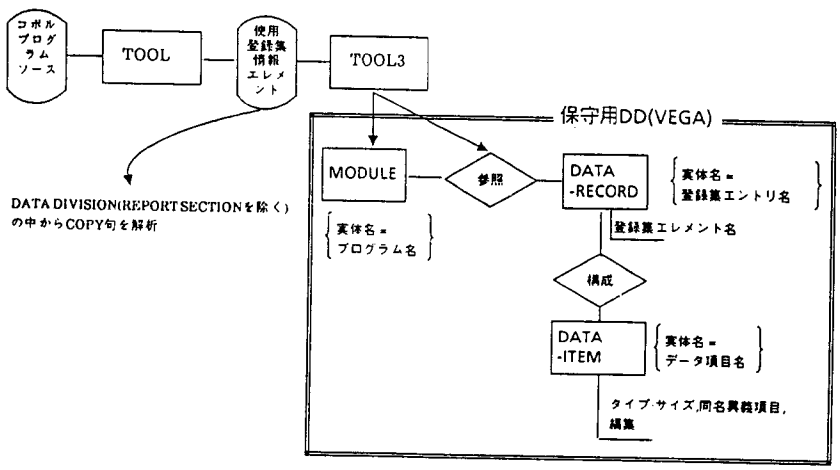


図２－１－２

3.3プログラムが使用しているデータ項目の情報を蓄積。

(注) PROCEDURE DIVISION以降で使用しているデータ項目のうち、登録集で定義されている項目との関連を蓄積する。

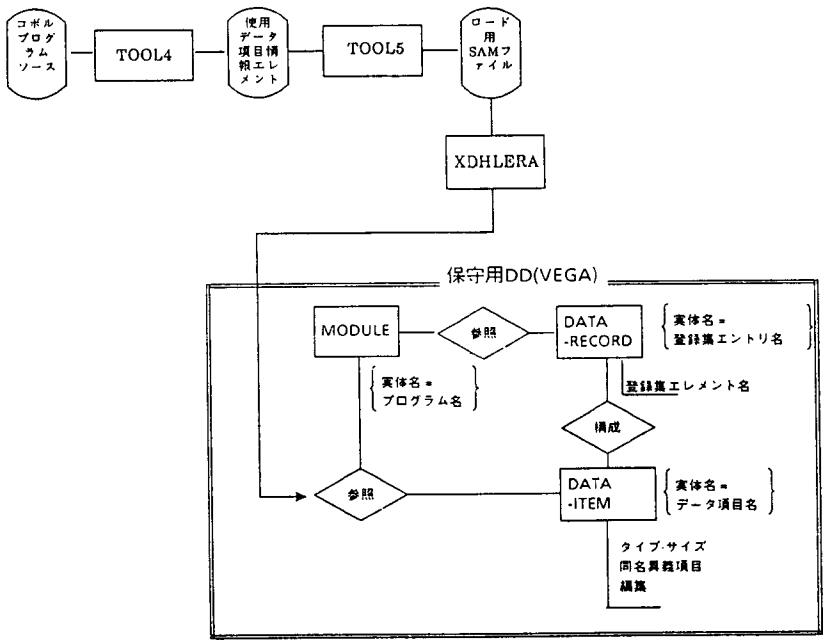


図2－1－3

構築にかかった時間は，ｃｐｕで４５時間，実行延べ７３時間で予想を越えていました．

今後のメンテナンスについては，現状は初期構築ということで無からＤＤを作ったわけですが，日々のプログラム修正があるわけで，本番に移す際に必ずＤＤにも登録するようにしています．しかし１日１００本もの修正が重なるとＤＤの更新に数時間かかるので，運用面では大変です．ただ，使う側からみると何も無い時代と比べて関連項目の検索は早くなっています．

ＤＤを使っての参照方法は，まずスタートすると検索画面が出て，

(1) データ項目の変更によって影響を受けるプログラムとＰＲＯＣとＣＯＰＹ－ＬＩＢ

(2) ＣＯＰＹ－ＬＩＢの変更によって影響を受けるプログラム

(3) ＣＯＰＹ－ＬＩＢで定義しているデータ項目

(4) プログラムで使用しているＣＯＰＹ－ＬＩＢ

(5) プログラムで使用しているデータ項目

といったものを選択するようになります．

これは，まだ使い出したばかりですが，問題になっているのは検索時間です(1) のデータ項目とプログラムの関連が何千件にもなるわけです．データベースを全てなめることになりますので，ちょっと時間がかかります．場合によっては１時間弱，データ項目の多いものはバッチで流すということで対処しています．

以上が保守用ＤＤの概要です．これが開発環境の管理ということになっているかどうかは分かりませんが，メインフレームを使う者にとっては最も悩んでいる部分なので作ってみたという次第です．

本当はＵＮＩＸとか分散環境を使ってみれば良いのかも知れませんが，メインフレーム上でなんとかならないかと考えたわけです．

臼井：　有難うございました．

次は山浦さんにお願いします．

２．２．ソフトウェア開発に於ける三身一体

山浦：　日立ソフトウェアエンジニアリングの山浦です．今日の私のトピックスは「ソフトウェア開発に於ける三身一体」と題し，ソフトウェア開発方法論，ツール及びソフトウェア・メトリックスについてです．よろしくお願い致します．

まず，エンジニアリングの評価基準というのがありますが，最初にくるのは「高品質のものを安価で速く」ということになります．テレビのコマーシャルで吉野屋の牛丼は「旨い，安い，速い」で，エンジニアリングの心髄をついていると思います．次に「楽して儲けよう」というのがあります．少ない労力と少ないお金で儲けよう，儲からなければエンジニアリングではないと思っています．じつは，大森の平和島競艇の白タクの呼び込み文句なんです．

さて，技術の進歩について，人はどれくらいの速さで移動できるかを見てみると，まず歩く場合は４ｋｍ／ｈ，１０倍に速めようとすると自動車が出てきます．さらに１０倍に上げようとすると飛行機ということで，地上から空へと発想の転換があるわけです．さらに４，０００ｋｍ／ｈまで上げようとすると空気に頼れずロケットになる．３倍くらいなら歩行から自転車くらいで済むけれど，１０倍位にすると新しい考え方が必要になります．これは力まずに進める巡航速度で考えています．

それで，ソフトウェアの開発環境ということになりますが，先ず，ソフトウェア開発の方法論と，メトリックスとツール群は，３つがうまく噛み合っていないと開発環境とは言えないと思います．交通安全というのを考えてみますと（交通システムと考えても良いですが），まず交通安全のための方法論というのが有ります．追突防止にはこうしなさいとか，左折時の歩行者巻き込み防止はこうしなさいといった具合いです．そうしたルールのようなものが，ソフトウェア開発での方法論になります．こういうふうにすれば誰でも一定の結果を得られるであろうというものです．それに対して，状況の計測というのがあります，時速何ｋｍなのか，視界が何ｍであるとか．道路の混雑度とか，路面の状況はどうかといったもの，これがソフトウェアのメトリックスという事で方法論と密接に関係があります．最後にそれを具体化するツールがあります．例えば自動車であるとか，信号機であるとか，こういった物が全て噛み合わないとチャンとした交通システムというものが出来ないわけです．ですから，メトリックスが無いのは，メーターが無くて自動車を運転するようなもので，ほとんど狂気の世界です．

ソフトウェアメトリックスの無い環境，これは伝統工芸の世界です．工業ではない．工業と言うからには自分でどんな物を作っているのかを定量的に把握できなければならない．イギリスのロード･ケルビン卿の言葉で，「計測できて初めて自分はそれについて知っていると言える」，知らないものをどうやって作るんだ，フィードバックの掛けようが無い．これを私は「技を盗め症候群」と呼んでいます．別の言葉で言うとドンブリ勘定の世界，感覚だけで処理をする，ほとんど芸術です．

次にツール群の無い環境，これは，今は殆ど無いと思いますが，ツール＝＝環境と誤解されている．ツールの無い環境は１番救いがあると思う．これは手工業です．「今日も１日ご苦労様症候群」．

３番目に開発方法論の無い環境というのが有りますが，これは何十％とある．メトリックスほどではないが，多い．これは無法地帯です．「俺たちに明日はない症候群」と呼びます．暴走族の話を聞いてみると，無法の世界でも厳しいルールが有って，決して楽でははいようです．

それではソフトウェア･メトリックスについて簡単にやりたいと思います．実はこのＯＨＰは再利用なんですが，いつも最初に言うのは，「計測できたり数値で表現できたとき，初めてそれについて知っていると言うことが出来る」というロード･ケルビン卿の言葉です．これが全ての根底にあります．ソフトウェア･メトリックスとは何であるかというと，まず，ソフトウェアの品質／性質を定量的に計測することが有ります．それをどう使うかというと，計量化されたデータを基に，現実の生産活動をコントロールするものでなければいけない．単に数字があるだけでは何の意味も無いのです．言い訳の材料にしかならないのです．もう１つは，過去の活動／経過を分析し，現在の生産形態に反映させて，より効率的な生産形態へ向上させるためです．

ソフトウェア･メトリックスというのは，やらなければいけないと言われてはいましたが，比較的新しい分野でして，歴史ですが，私の独断で，第１期は始まりで１９７６～１９８２年頃，１９７６年はマケイブという人がサイクロマティック･ナンバーというのでプログラムの複雑性というのを計測しようと言い出した年です．まだ，複雑度というのが定義されていませんでしたから，まず手近なデータを基に計ってみようというわけです．例えば，ステップ数とか仕様書のページ数とか，サイクロマティック･ナンバーとか，ソフトウェア･サイエンス，こういうものが出てきました．とにかく計ってみようというのです．

サイクロマティック･ナンバーというのは簡単で，フローチャートがあるとしますと，それが幾つの閉じた領域に分かれているかというものです．全体が１つで，各々の部分が２，３，４というわけで，この図の場合は４という

ことになります．数学の世界でオイラー数というのがあって，それに相当するのだとマケイブは言っています．もう１つのフローチャートでも複雑度は４というふうになりますが，たぶん皆さんはそんなバカなと思うでしょう．そうなんです．サイクロマティック･ナンバーというのは言葉は立派ですが定義がハッキリしていない．何でこう出てきたのか科学的な根拠が無いのです．タダ数字を計ってそれが複雑度だと言っているわけで宗教の世界なんです．

次に出てきたのはソフトウェア･サイエンスというので，１９７７年にハルステッドという人が発表したんですが，やる事は，先ずオペレータの種類の数，ｉｆとかｇｏｔｏとか+やーとか，これを出す．それから，オペランドの種類の総数(変数の総数)，それに，各々の出現数を出して，これを式に入れるとプログラムの作成とか理解に要する時間が「秒」で出てくるんです．すごいですね！！　これはほとんど冗談ですけどマジメに検証する人がいるんですよ．いまでも情報処理学会などで論文が出ています．困ったものです．直感的にみてどうかなと思うんですが，問題は科学的根拠が無いことですね，つまり値が何を表現しているか不明であるとか，精度が低いということです．また，コーディング以降の工程にのみ適用可ということです．このうち，科学的根拠が無いというのは，使ってみて良ければ納得できるんですが，２番目のは製品が作られてから計測することになって，悪い値が測定されたらどうするか，結婚してからやめとけば良かったかなと思うのと同じで，やり直しは大変なことです．さらに，単一の値で品質／性質を評価するのは，品質のコントロールやフィードバックが困難になります．例えば，４と８とでどう違うか？

２倍なのか，＋４なのか単位が明らかでないし，ｚｅｒｏが定義されていないわけで比較の意味も無いわけです．

これではダメということで１９８２年頃から第２期ですが，実用化への始動ということで，ソフトウェア品質のある部分を計測しようというふうになってきました．品質という言葉がやっと見直されてきました．値の意味を追求しようとする．そしてメトリックスを意味のあるものにしよう，役に立つものにしようと研究され始めた．第３期に至る過渡期のようなものです．

第３期が１９８４年以降で，私がメトリックスの関係を研究し始めたのが１９８４年だから，第３期ということにしています．品質制御の追求という時期です．

まず，メトリックスというのは生産活動を制御するためのツールである．あくまで命題ですが，だからソフトウェア･ライフサイクルの最初から最後まで，つまり要求仕様からテスト段階まで必要です．また，メトリックスと直結したソフトウェア開発方法論というのが必要です．それに，計測の自動化とか隠ぺいとかも必要です．現状は自動化されていなくて，各々のプログラマーの自己申告になりますから作業が大変ですし，データの操作という問題もあります．

次に，品質そのものを計測しようというもので，私がやったアプローチというのは，ソフトウェアの品質を分解して各々について要求を具体的に表現する値を抽出しようとしたものです．つまり，１つの値でやるのでなくて，例えば信頼性というものについて，こういうふうに計測したら良いのではというようなもんをいろいろ使って．これにはスペクトラム･メトリックスという名前を付けたんですけど．何故こんな事をやるのかというと，例えば美人コンテストで９.６と８.０の人がいたとすると，９.６とか８.０とかは何だろうか，そんなことを言われるよりも，身長／体重／趣味といったもので具体的な数字が数多くあった方が，選ぶ方としても見方が違っているから有難いんです．数値が小さい人の方が劣っているとは言えないわけです．例えば，伴侶を求めているのか友達を求めているのか会社で働く人を求めているのか，ということです．１つの数値で代表させていることの危険性があります．

メトリックスによって方法論を制御してやろうという考えで，メトリックス･ガイディッド･メソロジーという名前を付けまして（略称ＭＧＭ），名前はすごいんですが，まだ実体は無いんです．メトリックスの適用により生産活動をより良い結果に導くという考え方で，全ての工程にメトリックスを適用……など，先ほどと同じ事なんですが，最後に測定された値というのは客観的なんですが，その意味するところは各人違うわけでして，それを主観的に使おうということです．ファジーセット(曖昧論理)というのを適用したら出来るかも知れないと思っています．まだやってはいませんが．

品質の分解というところでは，いろいろな見方がありますが，私は１０個に分解しています．まだ十分に検討されているわけではないんですが，どうするかというと，例えばインティグリティについて考えてみますと，システムの外敵からの脅威に対して安全かということ……ハッカー

であるとか，悪い人が侵入してこないかということですけれど，メトリックスとしましては，

(1) 堅牢さ，どれだけ外に対して強いかということですが，まず，パスワードの強さとか，どんな方式を使っているか，あと，機密階層レベルといったものを計ります．必ずしも数値で表せるとは思いませんが，これも１つのメトリックスと考えています．

(2) システムの外の人，つまり悪いことをする人にとってデータとかシステムにどれだけの価値があるのか，価値の全く無いものに対して金をかけてもしかたないわけです．

(3) システムの中の人にとってどのくらい価値の有るものなのか，

(4) 機密保護のための実際の費用です．つまり装置とか，オーバヘッドとかシステムの監視というものです．

などです．こういったものからシステムは安全なのかどうか，保護をすべきかどうかといったことがおぼろげながら判ってきます．先程のように全部まとめて４.0 では訳が判らないわけです．

他の項目については省略します．

次にファジーセットについてですが，これはカリフォルニア大学のザデー先生が１９６５年に発表したんですが，曖昧論理，例えば暑いとか寒いとか，人によって表現が違うというのをどうやって表現しようかということで，確率のようなものを導入して定量的に表現しようとするわけです．曖昧な概念を数学の世界できれいに捕まえようとしたのです．例えば，寒がりの人と暑がりの人がいて，暑がりの人は１０℃位ではまだ寒い，２０℃あたりで確率０.5 くらいで暑いと思う．３０℃を過ぎると暑くてたまらない．０.0 から１.0 の間に確率のようなメンバーシップ関数で割り当てようというのです．寒がりの人は３０度を過ぎてもまだちょっと寒い，４０℃辺りでやっと暑い，こんな人いるかどうか判りませんが．ソフトウェア開発の例でいうと，初心者の場合モジュールの数が１０であると繁雑である，それに対してエキスパートでは４０位で苦しくなるという具合いで，モジュールの数と難しさというグラフにすれば対策を論じられるわけです．

結論としまして，メトリックスについて私が言いたいことですが，

(1) ソフトウェアの生産がエンジニアリングとなるためには，メトリックスによる品質の定量的把握が不可欠である．メトリックスの無いものは開発環境ではないと言い切ります．

(2) メトリックスをどう使うかという点で，メトリックスは品質の各要素を明確に表現でき，生産活動をより良い結果に導くものでなければならない．作った後で計ってもつまらないのです．私の結婚は失敗であったという理由は出ても，どうすればもっと良かったかという良い方向にはいかないのです．これではちょっと寂しいですね．私も後悔していますが．

(3) メトリックスは開発方法論と切り離すことは出来ない．計器飛行の世界なんですね，飛行機をいかに正しく目的地に飛ばすかは計器が無いとダメなんです．

(4) メトリックスはソフトウェア開発環境との融合により，その結果が最大限に発揮できるのです．

ということです．

さて，何でも最初と最後が面白いと思います，小説でもそうですが．私は要求仕様の定義あたりと，最後の検証あたりが１番面白いと思っています．いつも気になるんですが，検証系と言うのはいい地位にない，冷飯を喰わされていまして泥臭い仕事が多いんで，何とか自動化できないかと思って開発環境の中でテストの自動化を考えています．この根底にドメイン･テスティングというのがありますので，それについて説明します．このＯＨＰも２～３年前からの再利用です．１９８０年にホワイト先生が考えたものです．

まず，プログラムの概念ですが，プログラムは入力データのドメインと，それに対応する処理，この組合せの集まりと捉えています．例えば，１８才以上の人に対してはこういう処理，１８才未満の人にはこういう処理という具合いです．あるデータの集まりに対して，同じ処理をするということですね．図でかくとこんな具合いです．

ここで，処理が誤っている……コンピューテーション･エラーと言いますが，…Ｆｉの誤りがあります．つまり，処理が誤っているのですね．次に分け方の誤り……１８才未満のつもりが２０才未満になってしまったというような……ドメイン･エラー，Ｄｉの誤りです．３つ目にミッシング･パス･エラーというのがあります．（Ｄｉ，ｆｉ）のセットの欠落ですね．機能もれです．処理そのものが抜けている．この３つがありまして，ドメイン･テスティングで何をするかというと，ドメインがどう歪んでいるかということを自動的にテストデータを作って上げようということです．それに，コンピューテーション･エラーの一部も検証できるんです．マイヤーズの書いた本

に「テストの技法」というのがありますが，その中に同値分割がありますが，その概念の一部がこれに入っています．

概略ですが，コーディングでｘ，ｙを読んでｙ≧１ならａが０，そうでなければａを２だとします．今，ｙ≧１という判定が正しいか否か判らないんですが，ここに注意して下さい．ｙ＝１の線があって（これが正しいかどうか判っていれば苦労はないんですが），その上にあればａ＝０下ならａ＝２と言うことになります．これをどうやって検証しようかということですが，まず，ｙ＝１の線の両側，十分離れたところの線上に２つの点ｐ１，ｐ３をとります．またまん中あたりで線に乗らない僅か下の点をとります（ｐ２）．両側の点のところはａ＝０，まん中の点のところはａ＝２になります．この３点をとればドメインの歪み，つまり線が上にいってるか下なのかが判るだろうというわけです．実際に値をいれてみると，ｐ１，ｐ３は線上つまりｙ＝１だからａ＝０，ｐ２は僅かに外れてａ＝２になります．しかし実は，真の境界が別のところにあって，仕様書とか頭で考えた値を入れてみると全てａ＝０になるんです．ここで実行値とズレているというのが，境界線がズレているな，バグが有るなと判るわけです．斜めのズレというのも同じように検証が出来ます．

なんとなく騙されているような気がして，良く読んでみるとやっぱり騙されている．というわけで，私も人に聞いて初めて理解できた次第です．騙されていると思っている人も騙されているんではなくて本当なんです．

ホワイト先生はフォートランの上でこれを構築してテストデータを作っています．もっと複雑になってｘとｙを読んでｘ≧０の場合はこうして，さもなければこう，そして，ａ≧１ならこうで，さもなくばこう，といったプログラムで考えてみますと，ｘ≧０のところで境界が出来まして，右半分と左半分で２つ目の判定ａ≧１の意味が変わってくるんです．それでいくつかの領域に分かれて，ドメイン･テストでは各々の境界線について３つの点をとって，境界のズレを検出してやろうというんです．今は２次元ですが，ｎ次元でも同じで，ｎ個のｏｎ（線上）のポイントと１個のｏｆｆのポイント（線から外れた点）で，少なくともドメインのエラーは判るというのがこのテストの主旨です．ホワイト先生のはフォートランですが，Ｃとか良く使われている言語に応用したいと思っています．いろいろ制限がありまして，それを解除しようとしています．ドメインテストにも欠点がありまして，ＣＯＭＰＳＡＣのとき発表したんですが深入りしないようにしましょう．

私が言いたいのは，テストを自動化したいんという事ですが，１番言いたいのはこれです．

「ＳＥＡにもテスト分科会を作りませんか」

ついでに，私が昔考えていた，開発はこう有るべきだというのがありまして，ＯＰＵＳＤＥＩと名付けて，こうするんだと考えた３年ぐらい前の話をします．

まず，昔の典型的な環境というので，シンタクス･ディレクテッド･エディタというのが核にありまして，それを使って高い品質／生産性を上げようというんで，まず道具が先にあって，それをうまく使おうとしているんでは無かろうか，ガンダウルフとか，コーネルのプログラム･シンセサイザーとかいったものだと思います．それではソフトウェアは滅びますので，私のアプローチというのは，まず何をしなければいけないか，というところから，それに向けてどんなツールが必要かという順で，洗練された，よくまとまったツール(実は紙の上でまとまっただけですが)を考えよう．特徴としまして，全てのフェーズをカバーしなければいけない．そして管理面でのサポートもできなければいけない……つまり金，時間といった面ですね．あとペーパーレスです．ツールとして考えたものは後で述べるとして，最後は健全な方法論の上に立っているよ，ということです（言ってるだけですが）．

ツールの１つですが，ＳＷＯＲＤというのを考えまして，半形式要求仕様記述言語です．例えば，ソートマージの機能は「入力ファイルよりレコードを入力して，その入力レコードをキーに従ってソートした後，それらを出力ファイルへ出力する」となります．自然言語の解析というのは他に願うとして，名詞か動詞かというのが判れば有る程度機能かデータかというのが出てくるのではないだろうかということです．その名詞／動詞にあるラベルを付ける．それは人間がやります．ラベルは普通は隠して見せません（指定は出来ますが）ただ，入力ファイルとかレコードとかいった言葉を次のデザインとかコーディングの時に同じ言葉を使って割り振っていけば，何とかなるんじゃないかと思いまして考えて，何にもやっていませんが．もし，同じ言葉で仕様からテストまでコードが出来ますと，どんな良いことが有るかといいますと，リップル･エフェクトつまり波及効果というのがありますね．ある所で修正すると，他のいろいろなところに波及してくるんです．それがある程度自動的に出来るのではないかと考えたんです．リップル･エフェクト･アナライザと名前はスゴイんですが，モノが無いだけに説得力が無いですね．

もう１つはＳＣＩＳＳＯＲＳ．道具ということで無理やり合わせた言葉なんですが，　Ｓｔａｔｕｓ　Ｃｏｎｔｒｔｏｌ　ｐｒｏｃｅＳＳＯＲＳという面白い言葉で，本人は結構満足しています．このベースにあるアイデアは，各々のフェーズ，例えばコーディングで１つのモジュールを作ったりとかテストケースを作ったりとか，その１つのフェーズについてモジュールのステータスを作ってやる．コーディングが終わっていないときは「何もない」というステータスを持ちます．次に「やっている最中」というステータスに移り，「一応終わった」というステータスになります．そして「デバッグしてます」，「バグが出ました」．また，先のリップル･エフェクト･アナライザで，他のプログラム操作から勝手に「チェックして下さい」というステータスが入ってきます．という具合いです．この効果としては各モジュールの状態がよく判る．マネージャに便利です，１つのメトリックスになるんですね．出来ていないとか，出来ているのが幾つとか，有る程度の目安になる．それから，ツールにこのステータスを認識させようということで，エラーが出ているときに何がエラーだか判らずにエラーを修正しようとするとそれは許さない．エディターが動かない．というふうに，ツールとモジュール･ステータスを結び付けるんです．今は，ツールはツール，モノはモノとバラバラになっているんで，それをまとめようと思ったのです．

さらに，実際にどんな物が必要かというので，先ず，原仕様の段階で，オリジナル･リクワイアメントというのがありまして，要求仕様を書く前の段階なんです．そこで，こういう物が必要だろうというのをまとめてみると，先ず，フィージビリティ･スタディというのがありまして，出来るか出来ないかをまじめにやらなければならない．問題の本質，本当にコンピュータが必要かということで，例えば道路が混雑するんでコンピュータで信号を制御して混雑緩和というのがありますが，ドライバーに要求を聞いてみて，いらいらを解消したいということならコンピュータを導入しなくてもよい，交差点では直進できないというルールを作れば，右か左へ曲がることになって時間はかかりますが，とにかく動いているからイライラは無くなるかも知れない．つまり，何をどうしようとしているかをハッキリさせないといけない．コンピュータは本当に必要かと確認しなければならない．自動車屋さんに聞くと，車の設計ではまず，エンジンが必要かというところから始めるそうです．モノを作る人としては，必要でしょうと言わせたいのですが．

次に今後何年くらい使いたいか，ということで，耐久性を考えなければならない．次に安全かどうか，セーフティ･アナリシス，これには２つの意味があって，外敵から安全かというのと，自分が悪いことをしないかということ……例えば，原子力発電所のように．

あと，スケジュール，コスト……どれくらいかかるだろうか，そして，法律の面です．

あとは，まあ，モノが無いだけに迫力に欠けますんで，この辺で終わります．

２．３．ツールの管理／改善

臼井： ５人分ぐらい話して頂き，様々な内容がありました．

質問は後に回させて頂きまして，最後の発表を桜井さんにお願いします．

桜井： ソフトウェア･リサーチ･アソシエイツの桜井です．

何を話すかというと，ある開発支援環境を作ったのですけど，それを使ってもらえなかったんです．４～５年前から開発を初めて，２年ほど前に実用化されました．あるユーザさんで使ってもらっていましたが他に使われていなくて，それは悲しいからなんとかしたいということで，現実的な話です．順序としては，先ず，どんな開発支援環境を作ったのか，次に，なぜ使ってもらえないか，最後にどう改善しようと考えているか，という話をします．

作ったのはＺＯＤＩＡＣといいまして，オフコンなどの小規模コンピュータで実行される事務アプリケーションを開発するための　総合開発支援システムです．開発当初はＶＡＸでＵＮＩＸの上にパソコンを沢山つないで，そこで絵を書かせながら設計からコボルの生成までやりましょうということで，レベルＩＩ－コボルを生成しています．

絵にするとこんな具合いですが（図２－３－１参照），ＺＯＤＩＡＣはオフコンのパッケージ屋さんなどを想定してまして，いつもいつも同じパターンのソフトウェアを作るデザイナーということです．ほとんどが帳票と画面からの伝票入力ですが，ユーザさんによって書式がちょっとずつ違うから修正が必要ですけど，その手作業を何とか機械化しようというわけです．それを支援するツールです．

これは，実は２年前の第１回環境ワークショップの時に既に出来ていて，その時に酒匂さん（ＳＲＡ）が苦労話などをしていて，今回はそれを直すのにどうしようかという話

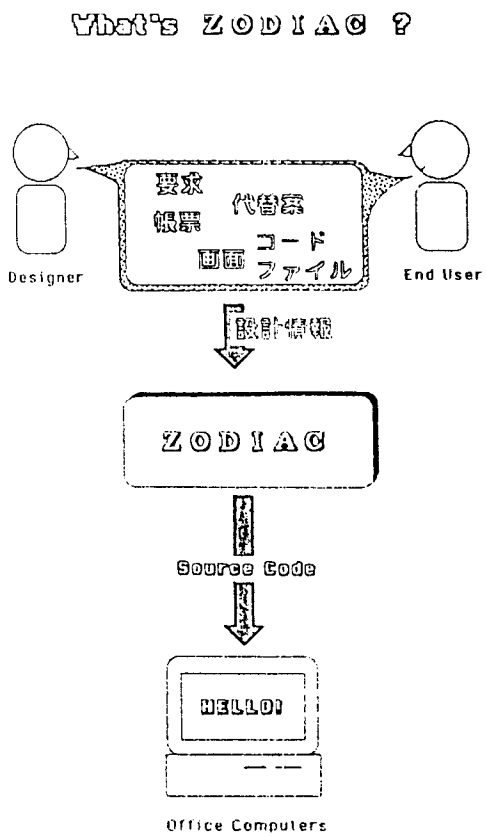


図２－３－１

になります．

当初の予定として，基本思想は，

(1)「システム内のデータ相互矛盾の排除」ということで，先ほどの田中さん（山一）の話にもありましたが，新規開発の段階でデータの矛盾を避けようということで，ＤＤを作ろうとしています．

(2) 要求定義，プログラミング，テストの作業全般のオンライン化

(3) コボルの自動生成

(4) ドキュメントの自動生成

といったものです．

稼働環境は，当初はＵＮＩＸにパソコンをつないで疑似的にマルチウィンドを構成していました．今は，ＮＥＷＳ（ソニー）のようなワークステーションをつないで，Ｘウィンドの上で動いています（図２－３－２参照）．

ＺＯＤＩＡＣのシステム構成（図２－３－３参照）は，設計からジェネレートまでサポートするということで，設計

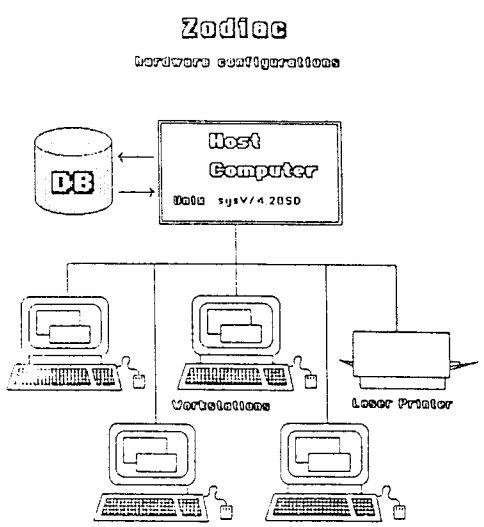


図２－３－２

情報データベースをもって（昨日の話に出たソフトウェアデータベースに相当すると思いますが），設計に関する情報を全部ここに蓄えておきます．ユーザが，つまりアプリケーションを設計する人ですが，設計支援系のツールを使って設計情報を登録します．ここで設計情報というのは画面や帳票のレイアウト，ファイルの設計といったものです．もう１つ，ソースコードをジェネレートする為のルールというのをパターンとして予め登録しておきますが，それと合わせてコボルのソースコードをジェネレートします．あとは，同じＤＢからドキュメントを出します．現状はこんな形です．

特徴を挙げますと，当初の考え方が「構文エラーの無いコボルプログラムをジェネレートする」という事でしたから，そういったものが特徴になっています．

具体的にどんなツールがあるかといいますと，現在は同じ様なものがいっぱいありますから理解できると思いますが，画面設計ツール，帳票定義ツール，更新条件などを定義する更新定義ツール，各処理で使う入出力ファイルや帳票などとの関連付け，補足説明の入力をする処理定義ツール，それに，帳票と共に用いられる定型フォームの定義を行う書式オーバレイ定義ツール，それに，ファイルのレコードを定義するファイル定義ツールといったものがあります．他に，ＤＤの項目を登録するツールがあります．

全体としては，設計情報があって，システム･リソースがあって，ソースコードがあって，それを全てユーザ，つまりアプリケーション設計者がみられるようになっていま

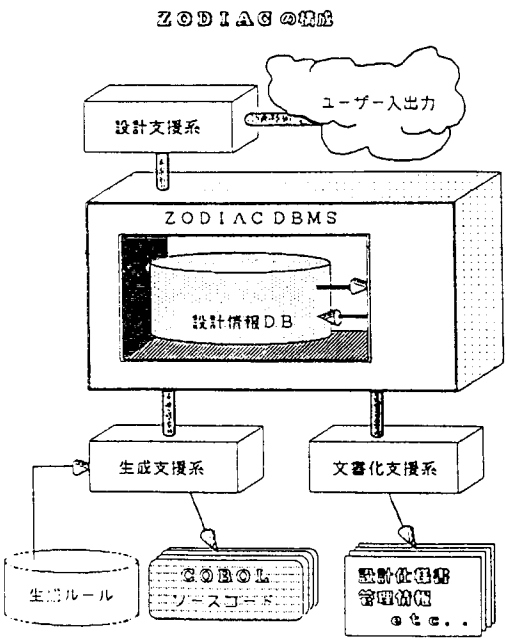


図２－３－３

す．が，実は，使いづらいというもののようです．

画面の形というのを見て頂いた方が分かりやすいと思います．Ｘウィンド上でマルチウィンドになっています．昔はＰＣ上に出ていました。メニューから起動すると，帳票定義，画面定義，ファイル定義，ボリューム定義，……のどれを選びますかという形になって（図２－３－４参照），例えば，１番の帳票を選ぶと，次にどんな帳票が登録されているか一覧が出て(図２－３－５参照)，１つを選ぶと定義ツールが動くという仕掛です．対象となる帳票のレイアウトはこうなります（図２－３－６参照)．

どういう項目を何処におくか，といった指示が出来ます．そこで，各々の項目をクリックすると，ＩＤが何であるかというようなものが項目ＤＢから出てきて，項目の選択･定義が出来ます．項目の名を忘れたときは，ファイルの情報を出して，ここから項目を見つけられます．

ファイル定義ツールでは，最終的にはＤＡＴＡ ＤＥＶＩＳＩＯＮに落ちる部分ですが，項目のＩＤ，ＰＩＣＴＵＲＥ句，用途，長さ，といったものが出てますが，これらは項目ＤＤに入っていまして，このツールは単にファイル上の項目の位置付け(並べ変え)をするだけのものです．さらにキーの指定とか，マルチレコードの指定もできます．

画面の定義ツールは画面のレイアウトを作って，何処に

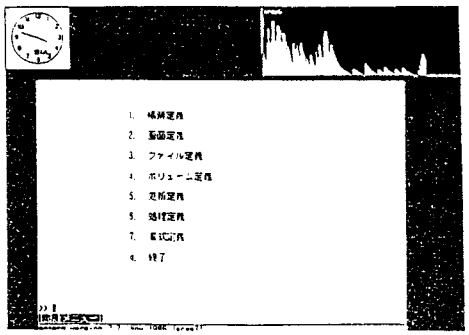


図２－３－４

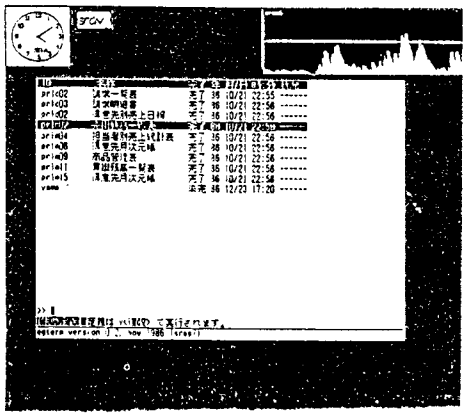


図２－３－５

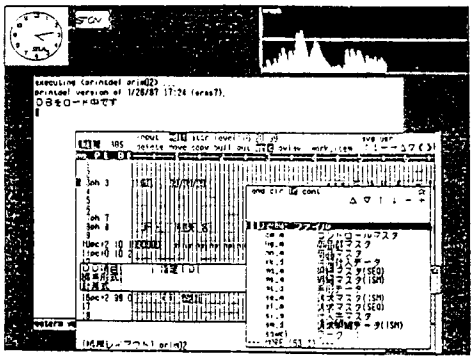


図２－３－６

どの項目を入れるかを指定する，さらに中身として，どういう条件かという様なものを入れるツールです．

こういったものでバラバラに定義されたら，処理定義ツールでプログラムとの関連付けを行います．例えば，売上伝票を発行するプログラムｂｌｌｄ０１を作ろうとす

るとき，予め定義した画面，帳票，更新条件などを使い，それにファイルは各々参照／入力等を定義していくわけです．そこまで出来たら，プログラム生成ツールを起動して，コボルのプログラムをジェネレートします．それをターゲット･マシンに送ってコンパイルすれば出来上りです．

というわけで，結構うまく考えたツールですが，実際にはあまり使ってもらえません．

それで，何故使ってもらえないかという点ですが，一応名誉のために言っておきますと，このツールは，あるオフコン･パッケージ･メーカとの共同開発で，２年程前に開発を完了しました．その後，先方は使いながら独自に改良して現在も使っています．当方は当方でＰＣ－９８００に移植したり，Ｘウィンドに移植したり，という具合いに進めて，一般向けに開発したんですが，基本的な機能には手を加えませんでした．共同開発の先方は独自に改良しながら使っているんですが，商品として売ろうとしたＳＲＡの側ではデモンストレーションなどをやっても反応が今一歩で，どうも使い難そうだということです．

その理由を探ってみますと，使いこなすためにはスタッフの負担が大きすぎるということです．まず，動作環境としてのＵＮＩＸ，Ｘウィンド，ＮＦＳといった基本部分の導入，運用，管理というのが大変……これはＺＯＤＩＡＣ以外でも同じだと思いますが．

もう１つはシステムを使う前にやらなければならないことが多いことです．それは何かというと，開発標準の設定です．共同開発したユーザさんは，もともと標準化が進んでいて，システムを導入し易い状況だったんですが，一般にはその努力が必要です．さらに，ジェネレーション……ＺＯＤＩＡＤＣではスケルトンと呼んでいますが，それを作っておかなければならないわけです．我々としては，スケルトンをサポートしていませんで，スケルトンが無いと何も役に立たないんです．項目を定義したり，計算式を定義したりはできますが，それが，プログラムにどう反映されるかはスケルトンによって決まるわけです．真に使えるものにするには，スケルトンをバッチリ固める必要がありますが，かなりの負担になります．スケルトンさえしっかりしていれば，その他大勢のプログラマは，項目などを決めるだけで同じ様な処理は簡単に出来るんです．デモンストレーションなどでは結局，スケルトンをどうするんですかという話になって，スケルトンはユーザが御自由にというと，大変だなあとなる．また，スケルトンはジェネレーションの大きなルールだけを決めるので，詳細の変化に対応できないということで，オウンコーディングを可能にしていますが，オウンコーディングは中間言語を使うため（ＰＤＬ）サンプルの整備がいる．

結局，使い易くするために標準部品を作るという点で，導入側の大きな努力が必要になるということです．この辺りを我々がやらなければならないのは重々判っているけれども，それが出来ない．何故なら，対象のアプリケーションの内容を知らないわけで，パターンなどはユーザでないと判らないし，開発標準というのもユーザ個々に違うわけです．売る側としてそこまで作るのは困難で，なんとかお客様にやって下さいということになるんです．骨組み，メタ･ルール，或はメタ･メタ･スケルトンといったものを用意すれば良いのでしょうが，なかなか難しい問題だと思います．

２番目の問題は，いろいろな作業フェーズに対するトータルなサポートに欠けるという点です．ＺＯＤＩＡＣがサポートしているのは，レイアウトの部分からソースコードまでですが，本当はどうかということです．

ＺＯＤＩＡＣを用いたアプリケーション開発がどうなるかというと，オフコン･パッケージが対象ですから，まず，ユーザとの話があって，受付して，分析．ここで分析というのは過去にどういった類似があったかということで，似たものがあればそれを利用するということです．適当なものがなければ新しく作るわけで，対象システムの登録，エントリの登録，という順で，ここからやっとレイアウト登録に入るわけです．そしてコボルを作るということです．その後，デバッグ，テスト，ドキュメント生成，保守……となりますが，ＺＯＤＩＡＣはここまでです．

ＺＯＤＩＡＣは登録するというだけで，例えば，対象にしているシステムが見えるだけなんで，結果，似たようなシステムをいっぱい作ってムダになる．というわけで，今１番欲しいのは，過去にどういうシステムを作ったかということが全部判る仕掛です．ですがそれが出来ていない．また，登録後に同じだからコピーし直すというような操作で，いろいろなコマンドがあって，たぶん細かいところまで入れると８０～１００個位あって使いこなすのが大変だろうと思います．またバラバラというのもあるでしょう．

それで，どうすれば良いかということですが，誰でもやりたいことは直ぐやりたい，簡単になることは簡単にやりたい．ＺＯＤＩＡＣはＩＤを知らないと何もできない，

ＤＢの機能を良く知らないと使えない，それは使い難いわけです．ある程度のカスタマイズは可能ですけど，もっとユーザの自由に出来ないといけない．ではどうしようか？　いきなり話が飛んで，ＺＯＤＩＡＣのツールはメジャーなもので２０個ぐらい，これらをコントロールするような，コンフィギュレーション･サーバみたいなものを用意して，そしていきなりＥＲＡがりまして，バラバラにいろいろなものが存在している世界をどう表現するかということで，エンティティとリレーションとアトリビュートのモデルでなんとかなるのではという話になります．

そうすると，いまのＺＯＤＩＡＣのＤＢはＣＯＤＡＳＹＬになっていますが，それだけでは難しくて，その上に汎用的なＥＲＡのＤＢみたいなものを構築しておいて，その先はＩｎｆｏｒｍｉｘでもＮＷＤＢでもなんでも良いとして，コンフィギュレーション･サーバがＥＲＡのＤＢとやり取りをする．それでもう１つ，ＺＯＤＩＡＣの最上位のインタフェースとして，ＥＲＡツールみたいなものを用意して，エンティティというのは，ここに帳票レイアウト，これとこれがどういう関係で，というのを見ながら定義できるものですが，そういったインタフェースを使って，そこで帳票というのをクリックすると，それに対して定義ツールが立ち上がるとか，進捗状況が見れるとか．そういったふうになっていれば良いので，アプリケーションを作るユーザには，その辺りを見せて中ではこうしようと，このツールはコンフィギュレーション･サーバに対して，追加，削除，コピー，リンクみたいなやり取りをすると，そして，コンフィギュレーション･サーバがＺＯＤＩＡＣのツールとやり取りをするというので良いのではないかと思います．(図２－３－７参照)

そこで，先ず初めは，今あるツールはそのままにしておいて，先ず統合化して，使う人にはダイレクトに帳票などを触れているというイメージにしておいて，それで中の機構を徐々に変えていこうと思っています．

以上です．

臼井： 有難うございました．

３．討論

臼井： それでは質問を受けたいと思います．３人に発表して頂いたんですが，みなツールを企画されて作成されていますが，そういったものを使ってどれだけ役に立っているかということが重要だと思います．皆さんの所でもいろいろ作られ，使われているんでしょうが本当にどれくらい役だっているのかなというところが私にも気になります．

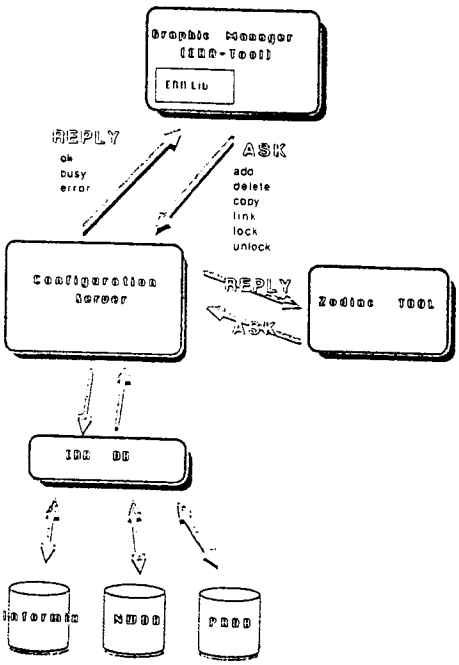


図２－３－７

野中： 桜井さんに質問ですが，使ってもらえないのはユーザ･インタフェースが悪いというのも有るかと思うんですが，それより，マーケティングの問題の方が遥かに大きくて，例えばコボルのアプリケーションを書く人が，ＵＮＩＸを持っているかとか，値段が高いのではとかいった問題の方が大きいと思います．その辺りはどう考えておられますか．

桜井： 独断と偏見でいうと，事務アプリケーションを作っている人達がＵＮＩＸの世界に馴染めないのではとか，ＵＮＩＸを導入しないのではというような話だとしたら，今はそんなに問題ではないと思います．ＺＯＤＩＡＣが始まった頃は心配されていましたが，今はそれほど心配ではない，それよりも大事なことは，ＵＮＩＸでも何でも良くて，ともかくこういうものを使えばこうなるというふうにすればいい，これだけ簡単に出来るんだよというものを見せれば良いんですね．見せたいし，実際かなりの部分でみせていて，デモでも，ア，これはすごいねと言ってくれる．でも見てくれる人は部門で重要な位置にいて，導入すると自分が大変になるという危機感を持っていて，つまり新しくＵＮＩＸを導入すれば，自分がＵＮＩＸの面倒を

見なくてはならなくなるとか，で，大変そうに見えるわけで，マーケティングについては心配していません．

佐原： 野中さんの質問にお答えします．実はコボルのジェネレータが売れない理由は，コボルという言語が存在しないからです．同じメーカー，１つの機種の中でもコボルが５種類くらい有るとか．だからジェネレータはそれに全て対応していかなくてはならない．ジェネレータを買いたいといっても，コボルの仕様がちょっと違うから，直すためにお金がかかる．それで躊躇してしまうのが多いんです．ですから，コボルが標準化されていれば，そういう事がかなり無くなると思います．

桜井： 私もそう思います．実際，名目としてはどんなコボルもできます，ここだけ変えればいいですよ，と言っていますが，そのちょっと変えるのが大変で．何種類ものコボルに対応しようとすると（機能的にはありますが），結局我々がずっと張り付いていなくてはならないんです．あるいは，ユーザサイドにカスタマイズ出来る人がいれば良いのですが，それを考えると大変だねということでヤメてしまうのです．

佐藤： ２年前にＺＯＤＩＡＣを聞いて，これはスバラシイと思って，さぞ売れているだろうと思っていたら，パートナーの方はカスタマイズ出来ていて使っているけど一般には使われていないという．なぜ一般のところではカスタマイズ出来ないのかと考えるとき，レベルの問題なのかどうか判らないけど，私はスケルトンを自分で作らなければならないということじゃないかと思います．で，それに対するアプローチとしてＥＲＡモデル，ＥＲＡインタフェースを作ったりとか，いろいろなツールを作ろうとしておられるけど．使いたい人がなぜ使わないかというと，スケルトンのような１番根本となるところが提供されていないわけで，その辺りの充実を先にやった方が売れるような気がしますが．

もう１つ，なぜ，コボルなんだろうという疑問が有ります．つまり，画面が定義でき，処理内容を定義しました，と言うのだから，そのまま実行できれば良い，インタプリット出来れば良いじゃないか．今後はそういうソフトウェアを狙っていけば良いと思います．ピッタリ合った話じゃないけれど，昨今のパソコンで，Ｒｄｂという言葉は珍しくもない．例えば，Ｒ－ＢＡＳＥ　ＳＹＳＴＥＭ－Ｖというのが有りまして，ちょっと触れてみたんですが，画面を定義してテーブルを定義して，処理内容を定義するともう動いちゃうんです．画面をちょうっと変えたいとか，キーを変えたいとかいったものをその場で直して，すぐ実行できるんですね．エンドユーザが，やりたいことを定義すればそのまま動くソフトウェアが出てきているんです．ただ，それがパソコンでしか動かないので，ｃｐｕ，ディスク，メモリの問題なんかで大きな処理は出来ないんですが，そうしたものがワークステーションに載るとか，あるいはロータス･１－２－３みたいなものが，ものすごく小さいけどユーザ･インタフェースが良くて，ユーザが楽だというソフトがいっぱいあって，裏には高級なアイデアとか思想が潜んでいるんだけど表にみせるのは簡単なインタフェースで，やりたいことを書けば動きますよといったものを．そういうノウハウをワークステーションなりオフコンなり，メインフレームなりに取り入れた方が，ツールの技法云々よりも良い，ソフトウェアのアプローチとして取入れた方が良いと考えます．

桜井： 私もそう思います．

酒匂： 元責任者から一言．佐藤さんのおっしゃる通りです．パソコンのＲｄｂのアプリケーション環境などをみると，なぜ，コボルを出さなきゃいけないか，という疑問は確かに有ります．ＺＯＤＩＡＣはコンポーネントにうまく分けられていて，最終的にコボルでなくても良いように作られています．それでオンラインとかＤＢの充実とか，パソコンで出来ないような大規模な処理が出来るというアドバンテージが有るわけです．で，ダイレクトにインタープリットしても良いのですが，それは環境の中に組み込みのものではない，つまり，ＺＯＤＩＡＣでは環境でもあるけれど，その中の１つとしてコボルの処理系が入っているけれど，その部分は差替え可能にしておきたいという気持ちで作ってきました．今後の展開としては，幾つか優れたＤＢなどがありますから，そういったもののサポートも出来なくてはならないと思うし，また，スケルトンの話もその通りで，何度かスケルトン整備作業という話はあったけれど，現場のニーズみたいなものを出して，それと一緒に作っていかなくてはならない．でも，そうしましょうといっても，手応えが無い．それでいながら，ツールを作る側としてはスケルトンまで考えないと売れない，そういう難しい点があります．これから，その辺を追々解決していこうと思っています．

堀川： 桜井さんの，なぜ使われないかという分析の中で，２点共感するところがありまして，１点はツールがあったときにＮＦＳとかＸウィンドとかを持ってきて管理するというのが先ず難しい．ワークステーションというのは

良いものだと思うんですが，まだ余り広まっていない．もっと広めるべきだと思いますけれど，ある種，敷居の高いところがありまして．実は昨日の分散環境の時に話したかったことで，その時歌代さんがおっしゃっていたのは，例えばＮＦＳでイエロー･ページを使えば出来るとか，最終的に分散環境というのは仮想的なマシンを提供するのだから，それはそれでいいんです．でも，それはあくまでもＵＮＩＸワークステーションの中だけの話で，例えば異機種間ネットワークというのをやったときには，ある種の計算機はＴＣＰ／ＩＰしかサポートできないかも知れない．そこには，ディレクトリが無いものも有り得るわけで，そういった場合はイエロー･ページを使えないんです．つまり，良いものと，実際に使い難いところがあって，ワークステーションをバラ蒔こうと思っても，使う人は思っている程には使えないように思う．そういう人を扱っていくような活動をしていかないと，ワークステーション自身が主張を得られないという壁にぶつかるという気がします．

もう１点の共感は，ユーザインタフェースの話で，カスタマイズするのは重要なことだけれど，それすら出来ないというユーザがいて，そういう人ほどカスタマイズしたがっている．で，そこでいいツールが出来たときに，何か定義を入れられれば自分の好きなエディタが出来るというのは，いい方向なんですけど，何か問題を含んでいると感じています．質問になっていませんけど．

歌代：いろいろなことを言われたんで，全部に答えられませんが．ＵＮＩＸだけでなくて，いろいろな機種のマシンが結合された状態で，今の状況とは違った問題が出てくるのでは，ということについてですけれど．ここ２～３年ＵＮＩＸというのがコマーシャルベースで一般化して，地下鉄の吊り広告に出ていたりしまして，ＵＮＩＸが普及してきたというのは事実です．１０年くらい前はメーカー毎に違ったＯＳを作っていて，全然インタフェースがなかったところへＵＮＩＸというスタンダード的なＯＳが出てきて，みんながＵＮＩＸを使うようになったんです．そこではＵＮＩＸ上のプログラムだったら大体は持っていってコンパイルすれば動きますよ，という１つのスタンダードが出てきて，そういうところでＵＮＩＸを使うと皆で話が出来て嬉しいねという良い状況が出てきました．最近はちょっと違ってきていて，ＵＮＩＸを使うということはみんながやっているわけで，プログラムが何処でも動くなんてのはＵＮＩＸの世界では当り前．そういった中でＮＦＳであるとかＸウィンドであるとか，ある種のスタンダードみたいなのが出てきている．ＵＮＩＸマシンを使って何でも出来る．何でも出来るからもっとスゴイ事をやろうと思ったときに，じゃあファイルシステムに関するアクセス方法というのはこういうのがありますよ，だったら幾つかのプリミティブな機能に集約できるんだったら，それをネットワークを使ってやり取りすることもできるだろうというわけで，ＮＦＳみたいな技術が出てきている．ウィンド･システムでも，最近のマシンではビットマップ･ディスプレイをもっていますから，それが自分のマシンでしか使えないのはつまらないね，というんで，じゃウィンドウシステムをもっと抽象化して考えると，こういうインタフェースが有ればみんなお話が出来るんじゃないか．というんでグラフィックに関するスタンダードみたいなのが出てきているわけです．そういう状況になってくると，よく考えてみると，今の状況でもＮＦＳをサポートしていてＸウィンドをサポートしていると，ＳＵＮでもＮＥＷＳでもＯＳがＵＮＩＸの必要性は無くなっているんですね．例えばＭＳ－ＤＯＳをつないでみようかとか，全然関係無いスーパーコンピュータをつないでみようかとか，最近ではＬｉｓｐマシンをＸウィンドつないで使ってしまおうとか，特化したコンピュータをＵＮＩＸ間での結合で生まれたスタンダードを使っていろいろなマシンをつないでいこうというような時代になっていると思います．

だから，もはや，ＵＮＩＸの必要性は無くて，ＵＮＩＸの普及によってかえって１つのＯＳとかツールが持っている機能がかなり抽象化されているレベルになっているんで，これからは機械の特性を活かしたようなＯＳであるとか，というものに発展していくような気がしています．今の技術というのはＵＮＩＸのいいところを抽出して，ＯＳとかユーザ･インタフェースとかのスタンダードを作っていくところにあると思います．今やっているのは，ＮＦＳであるとか，イエロー･ページにしても，ＵＮＩＸｉｔｙというものを廃止する方向です．多分，そういうのは無くなる方向に進んでいると思ってます．その辺は安心していいと思うんですけど，そういう動きが始まったところなんで，問題はいっぱい出てくると思うんですが，そのあたりをどうやっている人は解決しようと思ってやっているようなので，いい方向に向かうでしょう．

深瀬：ワークステーションを配ると，スキルの低い人が困るという話ですが，これは，負荷分散して仕事をすると

か，集中するとかいう話に絡むんですけど，今まで顕在化していなかった問題なんです．センターマシンに環境を置いて誰かがおもりをしてくれるという形だと，明らかにはならないんですね．カスタマイズしたいというスキルの低い人間をどうするか，管理させたら危ないんではとか．

ところが，私は逆に考えてまして，社内の話をしますと，ＰＣのソフトを開発していますが，ＰＣはそれこそ誰でも使うんですね．ワークステーションも仕掛としては大差無いものですから，やらせればやると思ってました．当然ネットワークにつながっているという，重大な違いがありますが．やらせてみると，大体はファイルを潰しますね．間違いなくＦＳＣＫとマニュアルに書いてある．ＮＥＷＦＳと書いてある．やってみる．パソコンだとやってみるわけですね．マニュアルを読んでやってみるんですね．そうすると，あるコマンドを叩いた瞬間に，次から何も出来なくなるという事実があるんですが，幸いにもワークステーションというのは１人の環境ですから，他に迷惑を掛けないんです．自分だけが困るわけです．しかも，かなり困るんですね．今までにやってきた仕事が消えてしまうんですから．そうすると，２度とそんな事はしないで，マニュアルには読む順序があるんだなとか，コマンドには危険なものと安全なものとあるんだなと，身を持って体験していくわけですね．それはある意味では手間のかからない教育なんです．本人だけが困って周囲は放って置けばよい．スキルの高くないエンジニアのレベルアップを図るにはワークステーションを配るのが逆に１番良いのではと思っています．

佐原： 大型機を使ってコボルを作っている技術者にスキルが無いというのはおかしい．あんな複雑なＯＳを，あんな複雑な言語でどうにも効率の悪いものを使って何とか動くシステムを作っている人達が，能力が無いわけがないと思います．実際にマネージャクラスがそういうものを使えと言っているから彼らは使っているだけで，決して能力が無いわけではない．大型機の場合もファイルを消してしまうというのは良くある話で，新人が入ってくると大体そういう騒ぎが起こって，我々はそれを管理する立場なんで走り回ることが多いんです．この場合影響は当然，ひどい場合はシステムを使っているセクション全部に及ぶわけで，大騒ぎになるけれど，ワークステーションだと自分だけが被害を受けるわけで，進歩してきたなという気がします．

臼井： 田中さん，今の件をどう思われますか？

田中： 佐原さんは以前，私と同じ様なところで，もっと先進的なことをやっていると噂に聞いていました．今の話のように我が社も同じ様な状態で，ワークステーションについてはやっていないので余り判りませんが，大型に関してはそういう事だと思います．今の話を聞いて，ワークステーションは個人だけなら我々からみると楽だなという気がします．

臼井： 山浦さんも意見をどうぞ．

山浦： 他人がファイルを消したというのは，まだ許せまして，自分で消すと悲劇的です．私もよくやりますが，ＯＨＰの原稿を作っていたんですが，いつの間にか消えてまして，というふうに体で覚えるというよりも，ある機構で防げないかという気がするんですが．例えば，暗室というのは光が入っては困るんですが，必ず扉が２つあって，一方が開いているときは他方は開かない．だから，初心者がいきなり扉を開けてダメにしちゃうということが無いんです．そういうメカニズムで救ってやるということが出来ないかというのが私の考え方です．技を盗むというのは好きではないので，話が合わないかも知れませんが，私はそう思います．

臼井： 質問をどうぞ．

三浦： ２つありまして，１つは先ほど堀川さんが話された事と歌代さんの話を聞いて思ったことですが，これは性格の違いで，これから難しいことがいっぱいあって大変だなと思う人と，でも大丈夫と楽観的に思う人の違いだと思います．

もう１つは，田中さんに質問したいのですが，ＤＤ導入の時に，もっと大変だったんじゃないかな，作る作業自体は大変でなくても，そういうのに踏み切るところが大変じゃなかったかなと思って，そのあたりを説明して頂きたい．それと，導入してこれから先の見通しみたいなものがどうなっているのかなというところを聞きたいのですが．

田中： 導入の際にはあまり深く考えませんでした．ちょっと前まで困っていたんだから作ってみようという軽い気持ちでした．なんとなく出来てしまったという感じです．今後の展開については，とりあえずなんとか使ってくれればいいやと．ただ，いいものを作ろうとは思っていませんでした．使い易いものを作ろうと，プログラマーとかＳＥとかが使い易いものを作ろうと，中身なんかチョットもいいと思っていません．使い易さに重点を置いて作って，今，使い方などを考えているところです．将来はどうなるか判りませんね．

野中： 先ほどの発言に対する弁解ですが，本当に言いたいのはユーザインタフェースが悪いからじゃないんだよということで，悲観することはないんで，正義は迫害されるんですよ．

別の質問なんですが，近々，職場にＭｉｃｒｏ ＶＡＸが入って，ＵＮＩＸ(バークレー版)を使うんですけど，同時にイーサネットケーブルを張ろうとして，いよいようちにもそんな環境がやってくるんだなと思ってワクワクしています．やっとルートに成れるとかホスト･マスターになれるとか，ワクワクしているんですが，周りから，大変だね，可愛そうにと言われるんですね．それがナゼかというのを知りたくて，会場にはＵＮＩＸのベテランがいらっしゃるので，どういう苦労があるか教えて頂けたらと思います．

加藤： 苦労話というのは聞かない方が良いと思います．野中さん御自身が苦労されて，どういうふうにしてきたか，というテーマで来年発表して頂ければそれが１番良いと思います．いろいろ細かいことはありますが，それをキッチリして頂いて，野中さんの次にスーパーユーザになる人が楽をする方法だと思います．私たちはゴチャゴチャした中でやってきて，ここがいけないとうまく指摘が出来ない．今後，被害に遭われる方はチャンと被害届を出して頂きたい．

坂下： 規模は違いますが，加藤さんのところと似たような仕事をしています．今の話は言い出せばきりが無い．加藤さんがほとんど言ってしまいましたが，今でも，あの時どうすれば良かったんだろうとか，あんなふうにうまくいったんだろうとか，人に会う度にいろいろ思うわけです．ぜひ，そういうフィードバックを掛ける場を設けて頂ければうれしいです．

中村： お二方とチョット意見が違いまして，ある意味で幸せだなと思うんです．１番最初にスーパーユーザになれる方は１番幸せだと思います．何故かというと，そのチョンボは波及効果が少ない．つまり，ＵＮＩＸマシンが入ったからといって，真っ先に業務ソフトが動き出すなんて事はまず無い．先ず使ってみようというフェーズがあるわけです．この間にシステムがダウンしようがカーネルが潰れようがファイルが消えようが，誰も文句を言わないんです．

この時期にはいろいろな冒険が出来ると思うんですね．ある意味で教えられないいろいろなことがあるし，自分で失敗することが結局，その時に育ったっていうんですか，その時にスーパーユーザをなさっている方は，血となり肉となる事がいろいろ生まれるんではないかと思います．これが業務に組み込まれて動き出すと，ある面，そういう冒険を出来なくするのも大切だとは思いますが，やりにくい．そうなってくると前任の管理者というのが存在して，これはやってはダメです，これはこうして下さい，こういう話が伝わってくるんですよね．でも，そうしなかったらいったいどうなるのか，例えばアイノードをクリアしてみたらどういう事が起こるんだといったことは，決して行わなくなるんです．で，行わないということは，もしも何かの障害が起こった時に，いったいどうしたら良いのか，ナゼそういう事が起こったのか，ということがなかなか対処し難くなると思うんです．

で，先ほどからファイルが消えたらどうするのか，それをプロテクトするにはどうしたら良いか，という話が出てるんですけど，ボク自身も幾つかソースファイルを消してしまったことがあるんです．簡単なチョンボです．あの．Ｏを消すハズが ．の前にスペースが１個入っただけで＊だけになって全部消えてしまったと． で，ｒｅｍｏｖｅにはもちろん，それを回避するために必ずｙｅｓと打たなければいけないという人も多いんですけど，私はそういうまどろっこしい事が嫌いで，パッパッと消える方が便利だと思って，オプションがそうなっていますので，こうして消えるんですけど．ただ，ある意味でボクは幸福だと思っているんです．というのは，人間ていうのは頑固ですので，前にあるものは必ずそれをどこかいじって使おうと思うんですね．そうすると，だんだんコーディングが汚くなってきてですね，最後はバケモノになるわけです．これがナゼか事故で消える．しまったと思うんですね，何とかしなければ．その時人間はいままでやってきた反省が出来ていて，それがフィードバックされて，非常にコードがきれいになるんですね．その途端に．文書でも今までの中で整理された文書になるんです．ですから私は失敗を恐れることはないと思っています．人間は頑固ですから，外的なそういう圧力がかからないことには，なかなか心を入れ換えない．悪いもん，邪悪なもんですから，そういった意味でいうと，まとまり無いですが，事故が起こってもいいじゃないかと，それはそれでいいじゃないかと，こういうふうに思う次第です．

歌代： 野中さんが神妙に聞いておられるんで，一言いっておこうと思うんですが．関係無いんですが，このあいだスキーにいきまして，会社で借りているペンションに行っ

たんです．そのパンフレットに載っている地図が非常にひどいもので，夜中に行ったんですけど，その地図を見て行くと絶対にそこへ行き着かないような地図が書いてありました．我々は，それを見て行ってえらい苦労をして，最後はペンションに電話をして迎えにきてもらったんです．会社で借りているペンションですから，当然，毎週誰かが行っているんですね，で，次の週はウチの後輩が行くことになっていたんですけど，アイツにこれを教えるのヤメようね，となりました．今，３人が野中さんの質問に答えたんですけど，皆，そういう心境に違いありません．それを神妙に聞いていては絶対にいけません．それだけです．

新田： 私もハワイ大学のコンピュータ･サイエンスに初めてＶＡＸ･７８５が入った時にＵＮＩＸをインストールしたんですけど，やはり，自分が最初にスーパーユーザとなってマシンを使うというのはスバラシイ事だと思います．システムのコンフィギュレーションなんかも１５分くらいで全部出来ますから，やってみて，ア，間違えたとか，ＤＺ１１をもう１個増やさなくては，とか，そういうふうにしてゴチャゴチャやって．その時はディスクのパーティションの切り方を間違えてですね，どうも１週間に１回くらい何かがおかしくなる．誰もそれを判らない．ディスクの蓋を開けてみたら白い粉が溜っていてどうもおかしいな．そういう事があって，その時はハードとソフトの両方がおかしかったんですが，そういういろんな，面白い事があるので，管理者になるのは，それほど割に合わない仕事だとは思いません．

臼井： 質問は有りませんか？ 意見発表は多いんですが．

ツールを一生懸命作っている人もあるし，欲しがっている人もあるのに，使ってくれないと嘆いている人もあるし．使って大成功というところは無いんですか，どこか無いですか．全然無いんだったら，今まで皆さん何をしていたんだかと，本当に思っちゃいますね．

酒匂： 感想を述べさせて頂きますけど．話題がワークステーションの話になっちゃいましたけれど，開発環境の管理という点からいくとですね，例えばＺＯＤＩＡＣのようなシステムを作ってですね，１つの，今の開発環境がいけないからその上に新しい環境を作ろうということで．で，実は薄々感じているのは，ちょっと抽象度は上がりましたが，新しい管理の問題を引き起こすわけで．話は元に戻りますが，メタ･スケルトンといわれるパターンが無いから，それを整備しなければならないっていう話ですけど，例えばＥＭＡＣＳだったらですね，ライブラリをハッキングしてくれる人はいっぱいいるんですが，メタ･スケルトンをハッキングして，コボルがジェネレートされるようなパターンを作ってくれる人はいないわけです．つまり，文化として根付かないんですね．で，要するに何かを，新しいシステムを，どんなに素晴らしいシステムを作っていっても，そこにはまた，新しいハッカーを必要とするような土壌が，現在の技術レベルでは現れてしまう．つまり，どんなに良いものを作る，新しい抽象化イベントを作り論理的にどんなにきれいなものを作ったとしても，今度はそのレベルで新しいハッカーが必要になってしまう．で，この無限連鎖をですね，断ち切ることが果して可能かどうかというのが，今チョット感じている疑問です．

臼井： どなたか，断ち切れそうな人．

藤野： それは，断ち切っちゃいけない．逆説的なやり方というのは，最近ソフトウェアで悩んでいるのは，実は，１番必要なんだということで．それに気が付き出したのは，自動車工業とかをやっているところで，ホンダとか日産とかトヨタとか．そういうところでやり出したのが，ハッカーみたいな人達が車自体を面白くするとかいう，社内でのコンテストみたいなので，やっぱり，そういうのが，次のブレークスルーのために絶対必要で，だから，ＳＭＡＬＬ ＴＡＬＫだとかＪ－ｓｔａｒとかっていう某Ｘ社のが，なぜマイナなのかっていうと，やっぱりそういうハッカーが育たないようなやり方をやって，戦略としてきたのがいけないんですよね．やっぱりそういう意味で，文化とか文明というのは，自分たちで草の根的に誰かがコチョコチョとやって出来ていくわけで，決してトップダウン的に某政治家が立てばなるというものでは絶対無いんで，やはりそれは断ち切らないで，それを育てる方向でやらないといけないと，いけないんじゃないかなと思っている某Ｘ社でした．

佐原： 役に立ったツールが２個だけありまして，前の会社で，最初の３台を導入するのに２年半かかったんですが，その後１年で２００台導入されたゼロックスのＪ－ｓｔａｒというのがありまして，これは，多分役に立ったんだろうなと思います．各々の，今までだとツールを使わないような技術者が，一応Ｊ－ｓｔａｒに取り付いて，何か仕事をしているというのはかなり見られたんで．そういう意味で，今のツールはユーザ･インタフェースがまだまだチョット甘いな，というところは有るんです．Ｊ－ｓｔａｒはＯＡという面からみれば，そこそこ使える

(ちょっとスピードは遅いですが) １０倍くらいになれば，まあ使えるなというツールですから，そういう意味では役に立つツールが確かに存在するということは証明されていると思います．

臼井： Ｊ－ｓｔａｒが出たところで，盛田さん何か無いですか？無いですか，それでは次の方．

佐藤： Ｊ－ｓｔａｒのユーザですが，その話はしないで，メインフレームの事をしゃべろうと思うんですが．メインフレームの上でハッカーがいるかどうかという議論は全然無かったと思うんですね．メインフレームで何故ハッカーがいないかというと，いましたか？ いました．私もその１人だと自分では思っているんですが，誰も認めてはくれないんです．メインフレームの上にも実は，それなりのソフトウェアは有るんですね．それを自分達でそれなりにカスタマイズしながら自分達の環境を作るというツールはある程度有るんです．ところが，実社会で，それらを使っていて作るのが面白い人が何人かいて，作っていて，当り前になっていて，取り立てて話題にならないだけの事なんですけど．ところが，ＵＮＩＸの上で何かをやると，すごいツールだなとなって，ワっと広まるというのは，世界が広いですから，ある意味で効果というか，実績以上の波及効果が出ているんじゃないかと思うんですけどね．で，例えば田中さんが先ほどＶＥＧＡというのを作られた話をしましたが，僕は，最終的にＥＲＡまで落としてないんですが，アレと同じ機能のものは６年ぐらい前に考えて４年ぐらい前に動いているんですね．ところがナゼ評価されないかというと，世界がマイナーなだけだと思うんです．そういう意味で，ハッカーみたいな，ＵＮＩＸみたいな上だけでハッカーがいるんじゃなくてメインフレームとか，そういうところにもハッカーがいる．ただ，ハッカーにとって興味のあるのはそれなりの，自分達がソフトウェアを作るためのもうちょっとベーシックとなるハードウェアとソフトウェアですね．そういったものが必要だろう，つまり，面白さが必要だと．で，たまたまＵＮＩＸの上にはそういうオモチャがいろいろ有る．メインフレームの上に無いかというと有る．ということを言いたかったのです．以上です．

深瀬： 私は以前メインフレームでＯＳのおもりをしていたんですけど，ハッカーはＯＳ屋さんにはいるんですね．ＵＮＩＸは最初からその部分をある程度諦めているんですね．楽なんですが，チューニング･ツールというものは無いものですから，それをエンジニアが全部書く．自分で作るというような事を私もやっていました．ところがですね，例えばメインフレームの場合は，つい，このあいだまで，空間は１６ＭＢしかなかったんですね．アプリケーションが納まらない．どうしたら良いか．空間を２枚つなぐ．ＯＳを直しちゃうんですね．そういう様な事で何とかＯＳを動かす，という事をやっていたんですが，アドレッシングが３１ビットになった時に意味が無くなるわけですね．非常に空しいんですね．チューニング･ツールなんてものを作っていても．バカみたいに．僕は今は別の仕事をやっているんですが．確かにハッカーはいると思います．ところが，それは，エンドユーザに近い方，というか，エンドユーザに使ってもらうためのツールを作っているんじゃなくて，何とかメインフレームらしく動くようなツールを使っている人達の中には，非常に優れたハッカーがいると思います．現に，一緒に仕事をしていた若い人なんですが，メインフレームの場合は設計してコーディングする人ってみんな分かれていますけど，その人間は３７０のアセンブラをボールペンで書き下しちゃうんですね．そういうような人は沢山いますんでね．ただ，そういう人達がエンドユーザ用のツールを作るというような立場にはなかなか現れてこないというような事だなと思います．

臼井： ツールの話が多いんですが，役に立っているという答えがあまり有りません．我々はツールを使って何かを一生懸命やろうとしていうハズなんですけど．

西岡： 最初は事業部にいたんですけが，その頃はアセンブラでプログラムを作っていて，ツールを作っていて，１３年の内，多分，半分くらいはツールを作っていたと思うんです．その半分の間に作ったツールが使われないというのはあまりにも可愛そう過ぎる．で，何というか，いろいろ考えて役に立っていないのではないかと思うんですけど．残念ながらですね．前に作ったツールは全て消え失せて，無くなっている．それはナゼかというと，アセンブラ用のツールを作っていたからですね．時代がＣ言語の時代にアセンブラは使われない．アセンブラの場合はｃｐｕ毎に全部違うわけですね．で，汎用性が無いから全然駄目だということになるわけです．しかし，使っているというものも有るわけです．そういうところを考えて，まあ，ある程度商品として売られているものも有りますので，それだったら少しは良かったのかも知れない．何しろ環境がどんどん変わっていくわけです．私が入ったときは紙テープでやっていて，それが今やワークステーション

の上で何かをやるんですが，そういった時，そういった時代の動きに対して耐えられるツールというのはいったい何であるのか．基本的にはそんなものが有るハズは無い．ただ，逆に，どんどん時代が変わっていくんですけど，その時代の長い差を出しているだけで，あれが出来た，これが出来た，それを見栄えだけはスゴイのにするといったようにしていくと，それは，後にも先にも絶対に残らないツールになってしまう．結局ツールを作るときはですね，いったい何が本質かという事を考えなくっちゃいけないと．これは自分への戒めなんですが．思うんですけど，そうした時代が変わって，インタフェースとかパフォーマンスがどんどん変わっても使えるようなツールというのを目指したいと思うのですが．

臼井： ツールは時代と共にどんどん変わりますから，なかなか，それが定着する前に消えちゃうのが沢山有るでしょうね．私がやったのも，やってもやっても消えていきそうな気がします．

坂下： メインフレームは大学の頃使っていて，今思うんですが，ちょっと振り返ってみてですね．ＵＮＩＸというのは成功したツールが無いとか何とか言っていますね．Ｊ－ｓｔａｒは確かにあれは重たいですが，成功していますよね．で，ｃａｔとかｍｏｒｅとかｙａｃｒｅｃｓとかもツールですよね．成功していると思うのですが，どうして成功していないと皆言うんでしょうか．それから，もう１つ，後ろで加藤さんとしゃべっていたんですが，ツールはシンプルでなければいけない．ｍｏｒｅとかｌｅｓｓとか出てきてもやっぱりｃａｔを使う．というのが真実ではないかと思うのですが，いかがでしょうか．

臼井： それに関して何かございませんか．

林： 全く同じ事を考えていたんで，追加コメントですが．何でもそうなんですけど，本当に身に付いてしまったらですね，もう，ツールという感じは無くなるんですね．だから役に立つものを言えといわれても，判らないんですよ．

例えば，ボールペンを毎日使っている人に，あなたは筆記用具として何が１番便利かと聞くのと似ているような気がします．つまり，次にボールペンを越える様なものを考えている人達にとって，次のものがツールらしく見える．例えば，電子ペンとか，といったものがすごくツールらしくみえるんです．そういうのは，今ツールとして存在しているとだいたい失敗しているんですね．それは役に立たないとかいう話になっているんですが，身に付いているから，多分出てこないんだろう，という気がして．もともと環境というものについて失敗談ばかり聞きながら，役に立ったら身に付いてしまうという，これも無限の繰り返しをしているんでは無かろうかという気がしているんですね．

臼井： 企業内遊園地では，いろいろなツールを作られていると思うんですが．

林： 役に立っているものは，例えば，先ずｔｒｏｆｆがあります．無いと生きてゆけないですね．ｒｏｆｆが無いとＯＨＰが作れませんから，今や．それにプロポーザルを書くときですね，きれいでないと見てくれませんから．そういうのはスゴクいいのかも知れないですね．金槌をいい道具だと言えば言えないことはないですが，ツールボックスの中には金槌があってペンチが有ると皆言いますが，普段道具だなんて思っていないでしょう．いい道具と言うと，電動ノコみたいにウィーンというのをみると，これはツールだと思うけれど，今，金槌を見てもツールだと思う人はいない．けど，使っていますよね．

臼井： 先ほど発表された方は，３社とも，トータルな総合ツールと言いますか，総合的な見地からみたツールがいるのではないか，という話があったのですが．今成功しているとおっしゃったのはツールでも，ノコギリとか金槌であって，オートメーション･ツールみたいなのは出てこなかったように思います．

林： 大上段に構える奴は，よく，失敗するんですね．それで，山浦さんにも一言いっておきたかったんですけど，あまり構えると死にますよ．

山浦： では，一言．いきなり走ってから後で考えると後悔すると思うんですね．ですから，結婚しちゃって後から悔やむことが無いように，僕は今から後悔しようと，そういうふうに考えているのです．私の経験から後輩に是非，そう言いたいですね．

歌代： 結婚してから奥さんのアラ探しをするよりも結婚してしまったら良いところを見つける……この人の良いところは何なんだ．そういう事を考える努力をするのが重要じゃないかと，私はチョット違うアプローチになると思いますが．

山浦： それは，僕は，諦めの境地だと思うんですね．何とか現状に甘んじようとするにはいいところを探さなくてはいけない．ですから，結婚みたいに大したもんじゃないんで初めにしっかり考えて，変なものが出来てしまったらしょうがないんで，いきなり物を作って後で悩むよりも,,というふうに考えているのです．最初にリクワイアメン

トのレベルで真面目にやって，後で悩まなきゃいけない事を前の方で悩もうと．ですから，コーディングまでいって捨ててしまうのはもったいないと思います．お金と労力をもっと有効に使いましょう．大阪人ですからゼニの世界ですね．

臼井： 今の話に関連して何か有りましたら．

中村： 計画に挫折有り．無計画に敵無し．こんな感じがします．ちょっと話が違うかも知れませんが，1つ目は社内で自分達のツールと言うのを叩き上げれば，果して皆に使ってもらえるか，つまり売り物になるツールが出来るか．

それから，人間が道具に合わせるのか，道具を人間に合わせるのか．ある意味で，先ほどｒｏｆｆの話が出てきましたが，あれも人間が機械似合わせている部分が多いですね．あれを見て，最初に使えといわれた人がガンとくるのは判ります．あれを見て最後に出来るイメージが描けるかというと，私はダメでプレビュアーかなんか出してみて，ア，しまったと思いながら頑張るわけですが．ただ，使い込んでくると逆に人間を合わせた方が便利だというのも多いですね．それは，いかに有るべきか．先ほどの話を聞いていると，道具を人間に合わせてやろうという親切心のツールが多くて，どうだどうだ，これではどうだ，という感じもするけれど，その辺はいかがでしょうか．

臼井： 女性の方からは何かございませんか．

桜井： 作る前にいろいろ考えて，なんていう話をしてましたけれど，それはそれで，大切なんですが，とりあえず作ってみるというのも絶対に必要で．取り合えず作ってみて，これが出来ちゃったけどどうしよう．それを，また，使えるように考えていくのも重要で，出来てみると以外と便利だっていうのも結構有って．考えているだけだと，便利かどうか判らないですね．作ってみて，実際使ってみる．自分が作った後，何人かにバラ蒔いて使ってもらうと，案外，初めに思ったのと違う使い方をしてくれる人もいるし．だから，ある程度考えたら，後は作ってみるのがいいんじゃないかと思います．

佐原： 役に立つツールが無いと臼井さんはおっしゃっているんですが，そうじゃない．ホストで第2次オンラインを作っていた頃ですが，非常に役に立つツールがあって，ＭＡＲＫ□で，第２次オンラインのトランザクション４００件をですね，何通りかソートしてプリントするという，なかなかすごいツールがあって，増刷に増刷を重ねて，１００部くらい配ったんですが．それを考えてみると，ＵＮＩＸの世界では，そんなものはツールと呼ばないんですね．ナザなら当り前にすぐ出来てしまうから．ただ，そういうのがコボルの世界ではツールと呼ばれていて，時々役に立つものが出てくる．ＵＮＩＸの世界では日常そういうものが横行していて，例えば，会場にあるＮＥＷＳというのは，私のワークステーションだったのですが，実は，先週の日曜日に深瀬さんからくるメールを見ようと思い，それから，ある事をメモしておいたのでそれを見ようとしてｌｏｇｉｎしたら自分のワークステーションが見つからない．おかしいなと思ったら，実は金曜日のうちに長野に発送してしまっていた，というふうに自分の生活に密着してしまっている．ですから，私は会社へ行ったら殆どツールを使いっぱなしという状態です．ＵＮＩＸはそういうぐあいにツールは活かされていますが，大型機のコボル環境では殆どツールが無くて，時々役に立つツールがあるが，それはＵＮＩＸからみるとツールとはう呼べないようなしろものであるということです．

佐藤： 一言だけ言いたいと思います．ツールがなぜ使われていないか，という話だったんですが，基本はですね，What I make is what I need.自分が使いたいものを自分が作ればよい．ところが，今のツールの作り方というのは，What I make is what you may need. なのです．つまり，必要は発明の母と言いますが，自分でやりたいとか，困っていることからいけばいいんで．そうじゃなくて，シーズという価値観も有るんですけど．例えば企業がものを売りたいために，技術が先にあってそこにモノが付いてくるということで発展することもあると思いますが．それに関して使われないと捉えているのは，そういうツールが出てきているんであって，もうちょっと地に付いたツールというものを開発していかなくてはいけないんじゃないかなという気がします．

落水： 袋叩きになるのを覚悟の上で．

ハッカーというのは非常に尊敬すると共に，1つだけ非常に不信感をもっているところがありまして．先ほどから聞いていてもそういう雰囲気があったんで．他人の作ったものの善し悪しとか，そういう価値判断とか比較とかはすごくうまいし，改良もうまいんですけど，その先を見た話がなかなか無い．

新田： 先を見た話をしなくてはならなくなったんですが．言いたかったことを言います．

ツールを作る立場から一言．ツールというのは（ＺＯＤＩＡＣみたいなツールの事です）どういうふうにツールを作っているかというとですね．例えば

ＺＯＤＩＡＣのようなコボル関係の作るときに，アー，このツールを導入した企業はきっと利益が上がるだろうな，とかは決して思っていないんですね．例えば，コボルのツールを作る人は，多分，過去にコボルのプログラミングで苦労したことがあって，その苦労をヤメたいとか新しい人にはさせたくないというふうな意味で作っていると思います．従って，ツールが成功するというのと，役に立つ，つまりそのツールが企業の中で利益を上げるために役に立つということは，作る人の態度として，何か違うことを言っているんではないかな，という気がします．

桜井：　最後に一言．

今，ＺＯＤＩＡＣの話が出たんですけど，先ほど佐藤さんが，使いたい人が作れば良いという話をしました．それはそうなんですけど，例えば，ＺＯＤＩＡＣみたいなツール，使われていないと，さんざん言ったんですが，実は初めに話したんですが，初めに欲しいといった人達は使っています．本当によく使っています．欲しいといっていた人達と一緒に協力して作って，そちらでは，欲しいと言ったから真剣にカスタマイズして使って，自分達で使い易くしようとして使っています．その意味で，使われていないわけではないんですし，無駄だったわけではないし，役に立っています．ただ，それ以外の人達，せっかく作ったんだからそれ以外の人達に使ってもらおうとすると，結構大変だという話になって．そういう視点で，先ほど述べたんです．欲しいなと思った人が勝手に作っていると，ただツールが増えて，気が付いてみると同じ様なツールがいっぱい有って，と言う話がもう何年か前から言われていたわけで，じゃ，せっかく作ったものは広めようと．そういう話でやってたわけですね．

結局，広めようと思っても，人によってチョットちょっとだけ違うとかいう話で，また複雑さを増しているということなんです．その辺で今悩んでいるということなんだと思うのです．

そういう意味で，カスタマイズ性を持ったツールとかいった話とか，使い方をどうしようかという話に，段々移行してきていると思うわけですね．だから，作ったら，その後でしばらくはそれを眺めて，いろいろ使い方を考えるとかっていうのも，重要なことなんじゃないかなと思っています．

臼井：　有難うございました．時間がきましたので，そろそろ終わりにしたいのですが．

いずれにしても，我々のソフトウェアをやっている人間のですね，自分らでツールを作れる技術があって，自分らの生産活動に活かせる立場にありながら，毎日毎日，徹夜だ，残業だ，と言っているのは，全然改善がみられないで自分を苦しめているというふうに，矛盾を起こしている業界じゃないかと思うんですね．でも，他の業界だったら，多分，自分で作ったら自分で利用するというところは，殆ど無いと思うんですね．どこかに頼まないといけないんですが．そういう力を備えて，それを利用する方法を知っているくせに，それが有効に活かされていない．で，少なくとも，いつまでたっても楽しくなっていないような気がするんですね．楽しくやりたいと言いながら，生産性を上げたいと言いながら，なかなか，なってない．本当は上がっているのかも知れませんけれど，それがまだ，十分うまく機能していないような気がします．せっかくＳＥＡなんかで集まっていますんで，ＳＥＡ認定○○ツールかなんかを制定して，皆さんに普及するような活動もやらないといけないんじゃないかと思うんですけれども．

これでこのセッションを終わります．皆さん長い間，御苦労様でした．

盛田　政敏

山浦さん：ソフトウェアメトリックスのない環境は工場ではないというのはよく分かるが、ソフトウェアは工場でだけ生産されるわけではない．そういうソフトウェアの持つ、ある種の力も必要だし、大切である．と思うが、
桜井さん：

吉井　孝

山浦氏の発表を非常に興味深く聞かせて頂きました．彼の主張は本質的には正しいと思います．ただ現状は、目先の事に追われ思想を持って仕事をしておりません（私は）が、コンピュータ世界にも哲学が必要である事を痛感しました．
桜井さんの発表ですが、ユーザーにとって一番重要な事の一つはスケルトンをどのように提供するかという事で、ユーザーで好きに作って下さいというのでは、どうなのでしょうか？

近藤　康二

３つの発表がそれぞれ個性があってもおもしろかった．HSK の山浦さんの SEA TST 分科会はよいと思いますが、TEST の分野には、じみな人が多く分科会として成立させるのはむずかしいかもしれませんね．SRA の桜井さんは、言葉の端々に苦労がにじみでていて、とても実感があってよかった．

小林　明彦

個人的には、ソフトウェアの開発環境をソフト面で考えたことが、ほとんどなかった(ホスト・ワークステーション両方）ため、あまりピンとくるのがなかった．山浦氏の話は比ゆが大変愉快でした．

野中　哲

Zodiac が使ってもらえないのは、ユーザーインターフェースが悪いことよりも、マーケッティングの問題が大きいのではないか？
メインフレームの環境が悲しいのは、その環境がおそまつなのだからではなく閉じられているからなのだ、と感じました．

藤野

環境管理

田中（山一）：保守用 D%D の構造等、参考になった．又、名前(Term?)付けの Guide-line の必要性は常々感じていたので、一寸、Basic なシソーラス研究をやる事も考えないといけないかな？
山浦(日立 SK)：メトリックスの重要性は認識しているのだが、主観を「数値」に置き換えられるのかは、今一つ疑問が残る．例えば、Shakespea の文の何も伝へるのは出来ないの同じように、 Software のある局面の集合が、Software そのものと表す事になるのだろうか．まぁ試す価値はありそうですね．

荻生　準一

山浦氏の Domein Test に興味があった．但し、多種多様の機能を持ったプログラムに対し、Test Point（３点）を定義するのがむずかしいのでは？

田中　慎一郎

今回も出入が多く、田中さんのはきけなかった．山浦さんのも半分ぐらいしか聞けなかったが、非常に楽しく聞けた．（長すぎたが）ぜひ、SIGENV で話していただきたい．（ENV の中で Test やってもいいですね）

井川　裕基

山一 DD の入力は、どのように？ユーザが自分で可？
日立やっぱり E1 はほしいですね．
ＳＲＡ Zodiac は、おもしろそうです．使ってみたいんだけど＿ 自分では COBOL の AP は作らないし＿

市川　寛

ソフトウェア開発環境については、物理的な部分（ホストマシン、NTT 回線、構内関係、端末）と論理的な部分（物理的な部分の定義、開発ソフトウェア群）があり、物理的な部分の維持・管理も大変である．ネットワークベースの開発環境が進んでくると、この部分はどのようになるのだろうか．

村川　貴広

・保守用ＤＤを作られたということですが、現状の問題をどのように分析し、整理されたのか．その過程について場を改めてうかがいたい．
・山浦氏は、過去のいろいろなドキュメントを読んでされを仕事に活用し、また、仕事をやりながら、論文（？）等を

出すようなことをされていますが、なかなかたいしたものです．私も社内論文を出してみようかと思います．それを大集させて、外にも出せるようになったらと思います．
・桜井さんの話を聞いて、ユーザに対して使ってもらえるツールをつくるというのは、いろいろ大変だなということを痛感した．私もある現場の開発環境を支援するシステムを手がけていますが、同様なことを感じました．これにどう対処していったらよいのか．今後検討し、その成果発表を行いたいと思う．

吉村　智香子

田中氏
私の場合、メインフレームを全く知りませんので、あまりなんとも言えませんが、メインフレームを使ってらっしゃる方の今悩んでいる問題がわかってこの解決のために保守用ＤＤを考えるしかないという現状がよく伝わってきた．
山浦氏
論理的非常におもしろい話であったが、ものがないぶん迫力が今一歩であった(？)．これらの話が、すべて構築されていけば、実にすばらしいものといえる．「三身一体法」は非常にうなずける話であった．
桜井氏
使用するにあたって、スタッフの負担の大きいツール、前以ってやらなければならないことが多すぎるツールというものは、使ってもらえないことは確かである．まずは今あるツールを統合化すること．ERA DB を作るとうのアプローチをぜひ進めてほしい．

中村　眞

初心者用 s/w 開発管理システムとエキスパート用システムに移行できるのか．特に「プロ」と呼ばれる人々にとってあまりそれまでの環境から移行する事にも抵抗が大きい事と思います．

久保　宏志

Zodiac Refirement Efforts の健闘を祈る．

歌代　和正

やっぱり結婚してしまったら良い所をさがす努力をしなくちゃねぇ．

田中　正則

大規模システムの管理にはデータ・ディクショナリが必要と思うし、又私も構築してみた．

？？？

みんなで Zodiac を使おう．

岡本

私はメトリックが無ければ、開発環境でないと言う山浦さんの考えに賛成です．メトリックそのものに関する考え方については、同じかどうか分かりませんが、環境とマネージさる上で、環境そのもののモンタリング結果を実体化することは重要で、その度のメトリックの設定に私も今悩んでいます．

堀川　博史

田中さん地道な活動として興味深い．山浦さん優秀な人だ．桜井さんなぜ使われないかという分析過程が知りたい．

熊谷　章

発表のスタイルが各人各様で面白い．

酒匂　寛

山浦さん　貴方はエライ！今度は動いているものの話も聞かせて下さい．SIGTST をやるときには、お話を聞きに参上いたします．

北野　義明

田中さん(山一)の話では、多分「こんなことはあまりやりたくないが、やらざるを得ないし、やればそれだけの効果があるから、しょうがないのでやろう」というのがあるのでは_ と勝手ながらそう思い同調しています．
山浦さん(ＫＳＫ)は、アカデミックな研究を自分の仕事と結びつけて実線的にやろうとされている様で、非常に面白い．が、周りの人を説得するのは大変だろうなと思いました．
桜井さん（ＳＲＡ）の様に WS の世界の人が COBOL ソースを Generate する仕組みを作らせている人はうちなんかがやろうとしても、WS 側のスキルを今、身につけようとしているところで、色々考えて頂きたいし、また、大型の世界をもっと知って頂いてうまく liNK してほし

いと思います.

森　幸一

(4-2)

(4-1) ·計測が困難でメトリックスを適用デキナイケースでもあるだろうが、ソフト開発管理に秩序を与える面で興味深い.

·メリット中心になるし、ソフト開発の現象面だけの認識におちいる危険が　　に思う. 構造的なモデルが見えるといいと思う.

(4-3)

ss@astec

· YCC の話は、ふだん縁がなく、内容を理解するのに時間がかかった.

· HSK は、困ってしまう. 特に、検証の問題をあんなふうにされてはいかん.

· Zodiac は大変そう

三浦あや子

小池　幸徳

山一の田中さんのプログラムから保守用 D/D を自動生成できるという点とその利用方法について興味をもった.

山浦さんのメイトリックスについてはたしかにソフトウェア開発に取り入れなくてはならないものが数多くあると思われた.

小林　貞幸

田中さんのアプローチについて、既存のソフトウェアについては、やむを得ないアプローチかもしれないが、新規及び今後については設計段階での DB によって問題解決を図ったらどうか. ソースプログラム全体なめるのはナンセンス

山浦さんのメトリックス論は共感を覚える. 現実的にツールなり開発環境なりを提供するにしても、生産性を計る. 尺度が存在しない(計数的に)のが問題であり、感覚的な「上がった」「下がった」だけでは論外と感じている者が大多数だろう.

具体的にどのような計測方法が可能なのか. 一緒に考えたい.

Zodiac との欠点はユーザ多すぎることだとは解るか. その解決策にいまいち考え不明.

佐藤　千明

·田中 DD は簡単に構築できれば使いたいが、その結果を考えるとソース解析でも問いに合っている現状です. 何か決定的な PUSH トリガーがあれば本気で考えたい.

·山浦あの話を imple した時、再度話を聞きたい.

·桜井オフコン用ソフトは COBOL で作るにも RDB を中心としてソフトウェアの方がいいのではないか. スケルトンがないツール　= DD の構築でしょうが、それでは一般プログラムの現実的課題の解決にはならない. いいものなんだからもう少し頑張って広めて欲しいと思う. 難しいツールでなく、まずスケルトンをいくつか用意すべきでしょう.

新田

マシンをコンボウしていたのでよくききませんでした.

堀内　泰

·「管理」の面をもっと前面に出て欲しかった.

·スピーカーの時間配分はどうなっているの？チェアマンは時間 "管理" をしっかり.

·討論テーマにある程度関連した話をもらしてほしい. (３人の話の中で)

酒匂寛

今回は、でませんでしたが、ツールを使う使わないという話は、いつでもヤル気の問題に還元されがちです．これは、そんなもので、しょうがないのでしょうか？

堀川博史

「ツールが使われていない」という話題が興味深い．

御喜家

人間は、どちらかというと、保守的な面があると思う．
最初に好きになったものから、また少し違う環境へ移るのは、良いと思っていてもなかなかできない．
例えば、健康の為に、ジョギングしようと思っていても朝になると、やっぱり起きられないのと似ている．
だから、ソフトウェア生産性向上というような目標がもしあれば、そこへ向けて、フェーズアプローチで、あせらずに、また、一つのステップの高さを上げ過ぎないで、いつのまにかレベル、意識が改革されていくように、常にそのような仕掛けに、心をさいておかねばならないと思った．

野中哲

メインフレーム系の人と Unix 系の人で、なかなか話がしにくいのが気になりました．同じバックグラウンドで話が進められるような世界になれば良いと思いました．

西岡健自

・計画性のない　Tool 開発は、趣味の世界のことにして欲しい．
趣味が悪いというのではないけれど、少なくとも企業の研究機関では許されない．
・ソフトウェア工学が、工学だとすれば、次の手順が適当．
(1) 仮説 (2) 検証（開発 ＋ 試用）(3) 仮説の洗い直し、改善
この観点から、桜井嬢のアプローチは、出色．

北野義明

大型と W.S では、文化の違う世界で、技術力がどうのこうのというのは、大間違いである．
ただ、異文化との接触に慣れているかどうかの差で、大型の世界では、せまく深く知っている人が多いのだと思う．
最後まで、世界、文化の違いがどこかで討論をさまたげていた．
でも、それは、それで有意義なことでした．

歌代和正

途中から入ったら、結局最後まで、何がテーマなのか分からなかった．でも、面白かった．

杉田義明

なぜ急に、Tool に関する議論になったか、その経過が良く分からなかったが、なかなか盛り上がった討論セッションであった．Tool そのものが役に立ったかどうかということに関しては、あまりにも役にたつことは、当然であり、話をする気にならなかった．
一概に Tool といっても、重いものと軽いものがあり、ごちゃまぜに評価はできないと思う．
それよりも、仕事をする上で、まず、その仕事に合った Tool を利用して、目的の仕事をすることの姿勢を作ることが、大事だと思う．
技術者は、Tool によって、評価されるものと思われる．

吉村智香子

ツールはシンプルでないといけない．
便利なツールは、定着してしまっているので、役に立っているツールを言えと言われてもなかなかでてこない．
地についた役に立つ便利なツールを作って、楽しく仕事ができるようにしていきたい．

村川貴広

・討論の内容が、よくわからない世界の話へ飛んでしまって、本来話してみたい開発環境のマネージメントの話ができない．私のパワー不足でしょうか？
・ツール作りは、やはり、自分たちが使ってみて使いやすいものであって、客先に合わせようとして、ツールを使いにくくしてはいけないと思いました．

田中慎一郎

今回も少し、テーマが違ったかとも思うが、これはなかなかよかった．

荻生準一

Picture Generator（画面定義ツール）は、私の行っているミニコン制御系の世界でも作成したことがありました．ただ、やはり、使われなかったようです．

理由としては、
(1) ユーザー（画面作成者）に対する PR 不足
(2) 種々の要求定義に対し、満足する機能を持っていなかった.
(3) Hard 自体（ファームも含む）の変更に対処できなかった. 等です. Zodeac は、画面の定義だけで、プログラムまで作れるのは、スゴイ！！

小林貞幸
今回のワークショップ参加メンバーの中では、UNIX は、メジャーかも知れないが、事務用汎用機（オフコン・事務用も含め）の世界では、マイナーなのです.（反論もあって当然）
そのマイナーな UNIX をオフコンの世界にせっかく持ち込んだ Zodiac なのだから、そのターゲットオフコンのオリジナルな環境下で動けるような仕組みが必要であり、尚且つそれは、ソースコードレスでも良いと思う.

鐘友良
実用的、汎用的なツール、例えば、UNIX 上での LEX、YACC ようなものは、本当にたくさん作ってほしい.

井川裕基
・ツールの有効性
・身についたツール(cat-ditroff) が多いので、そうすると、UNIX は、ツール不毛（もうすでに色々なことができる）の世界かも知れません.

新田
COBOL は大変だ.
パソコン上には、画面定義と動作定義を行うとそのまま動く便利なツールがあるが、Zodiac もそうすれば良いとか言っていたが、パソコン上にそういうツールがあるのは、パソコンを使う本人が要求者であり、設計者であり、作成者であるからである.
大型機を使用しているユーザは、「ユーザ」は一人の人間ではない. 要求者は、どんな画面で、どういう動きがほしいと思ってはいるが、必要な画面は、100-1,000 ぐらいあって、その人だけでそういうツールを使っているわけにはいかない.
「ユーザ」の中には、COBOL のソースが欲しいと思っている人もいる筈で、やはり、COBOL ソースを出すツールは必要.

三浦あさ子
田中一夫さんのような人が、きっと、もっとがんばると、すごい発表ができるような気がする.

近藤康二
私は、まずやってみることが大切だと思います. 人間の能力には、限界があり、すべての先を見直すことは、無理であると思います.（少なくとも自分には）
どうもありがとうございました.

田中正則
テーマの内容があってなかったように思ったが、まあよかった.

中村眞
有料ツールが使われるのか、タダだったら使う感のある物も多い. なぜなのだろう.

佐藤千明
・ラスターが広まったのは、簡単に仕えるからでしょう. ツールは、簡単に使えなくては%%%
・WHAT I MAKE IS WHAT WE NEED
WHAT I MAKE IS NOT WHAT THEY MAY NEED

小池幸徳
開発環境を整備する為の一つの方法として、優れた MMI により可視的な環境で作業を行うということだと思う.
そのために、現在心地好い可視的な環境を得るためには、32 bit キーが必要となってくる（？）. 現在 32 bit キーには、OS として、UNIX がかなり使用されている. そのために、UNIX を利用したパッケージ（開発環境の）が多くなるのでは？だから、私も UNIX を使っています.

森幸一
ユーザに使って貰えるツールは、ユーザの役にたたねばならないが、作るほうは、ユーザの使用についての理解が必要との指摘があったが、ユーザ要求の分析手法が作れると良いのだが. 社内標準、ユーザ標準、ユーザにするカトマイズのツールへの埋め込みが必要な気がする.

藤野晃延

全体と個という問題が、管理の問題だと思う．個としての使い易さの追求と、全体としての使い易さ(管理のし易さ)の追求は、相反するものなだけに、整合がむずかしいのだろう．

ホロン的アプローチが必要だと考えるが、具体的にどうやっていくかは、その仕組みを皆が考えて、現場へ応用しながら、より良いものを求めるべきなのだろう．

また、水平分散していく環境への対応も課題だろう．A-priori でないだけに工夫が必要かな．

小林明彦

三菱電気、堀川氏の意見に対し、全面的に賛成です．優秀でないユーザに対し、提供されるソフトウェア及び、環境がどのような形で提供されるかが、今後の UNIX (別に UNIX でなくても良いが)の拡大の方向性が左右されるのではないかと思われる．SRA 佐原氏の指摘があった "COBOL ユーザは優秀ではないことはない" という意見は、私もそう思います．しかし、異文化に触れたことのない人間が初めて触れる文化として、UNIX は、現在の所、親切ではない．多くを占める (日本で) メインフレーマーに UNIX または、ワークステーションが、拡大するためには、カルチャーショックを起こさないような配慮が「当初」いるのではないかと考える．

結局、発言はしませんでした．後悔はありません．

久保宏志

Recommendation な麻里

(1) Zodiac を環境プロダクトとして売るのではなく、Zodiac 環境を顧客にインプリメントするサービスを含めて売ること．顧客が希望する COBOL に合わせて手を入れることもサービスに含めること．

(2)Configuration service や ERA tool は、ビジネスのノウハウとして、ためこんでいく作戦が良い．プロダクト商品にするのを急いではいけない．

(3) スケルトンノウハウもサービスしながら、ためこんでいくのが良い．

(4) SRA が、オフコン用ソフトを受託開発するときの Environment として実践的に使い込んでいくのも有効なアプローチでありうる．

昭和６３年３月２日

ＳＥＡ環境ワークショップ
ＣＨＡＩＲＭＡＮ’Ｓ　ＳＵＭＭＡＲＹ

日本電子計算株式会社
臼井義美

このセッションは最後のセッションであったため、前日からの疲れで遅刻した人、酒がまだ残っている人、デモ用マシンの後片付けに追われている人、チェックアウトのため席をはずす人がいたりして、ちょっとざわざわした雰囲気のなかでの討論となり、申し訳なく思っている。しかし、実際のソフトウェア開発環境は決して理想的であるとはいえず、かえって苦境に耐えているＳＥの精鋭らしい雰囲気であると妙な感心をしてしまった。参加され方々もＳＥＡ風の討論に慣れてきたせいか間断なく討論への参加者が続き、活発な意見交換が行なわれた。

セッションテーマは「開発環境管理の実際」となっており、皆さんのポジションペーパによると、分散環境に即した開発環境の管理方法と組織を運用するための管理方法の２つが討論の対象と思われたが、分散環境については前のセッションで討議されたので、ここではマネージメントや開発環境のツールに焦点をあてて討論していただいた。

まず、３名のプレゼンテータの方からはいずれも実際に開発されたツールや企画されているツールについて具体的な紹介があったため、主にツールの問題についての討論となった。山一コンピュータの田中さんからは、最近話題から見放されそうになっているメインフレーム上でメンテナンスのための環境作りを行なった報告をしていただいた。日立ソフトウェアエンジニアリングの山浦さんからは開発環境とメトリックスとツールは三位一体であるとする、ソフトウェアエンジニアリングにおける正統信仰箇条ともいうべき山浦理論が展開された。その内容の多様性とほとばしるエネルギーでチェアマンが圧倒されて止める気力を喪失してしまい、時間を超過してしまいました。最後に、ＳＲＡの桜井さんからは、企業内遊園地でワークステーションをベースとした開発支援ツールを作っているという、現時点では理想的とも思える環境で作ったツールを、ほかの人が使ってくれないという聞くも涙の物語を語っていただいた。

その後会場の参加者を交えて討論したが、ツールの必要性は十分認識されており、我々にはツールを作る技術があることも了解した上で、ツールとはそもそも何か、また、ツールは本当に役立っているのかなどについて活発に論議された。

私自身の考えでは、理想的な開発環境とはどのようなものかという共通のコンセプトが明確でないため、どうしても個人のおかれたスタンスから見た改善の方法やアドバイス、懺悔の言葉が飛び交うことになったように思う。もちろん、いろいろな立場の人が自由に意見を出し合ってより良い道を探ろうということに意義があることは疑いないのであるが、できれば我々の求める方向をある程度見定めた上で解決の方法を探りたいものである。そのためにはわれわれの求める環境についての基本的なスタンスをはっきりさせたい。つまり、我々は開発環境について本当に困っているのか、なぜ困っているのか、何を解決すればいいのか、解決する技術は持ち合わせているのか、解決するためのコストともたらされるメリットは想定できるのか、などなど。

例えば、我々が討論し実現しようとしているのは、一体誰にとって理想的な環境なのかということを考えてみよう。

まず、開発担当者つまりＳＥ自身のための理想的環境を考えてみる。ＳＥのための環境であれば、便利で楽しくて思い通りのシステムが容易に構築できればいいであろう。

しかし、ＳＥとって便利な環境は管理者にとって理想的であるとはかぎらない。管理者は開発担当者が一心不乱にプログラミングしていることに満足するかも知れないし、開発者の能力を的確に判断したいと考えているかも知れない。管理用ツールは管理者個人の目指す方針によって、ＳＥの管理強化につながる可能性もあり、ＳＥの独自性を重んじ柔軟な管理を行なえるような方法を見いだすこともできる。

一方、経営者にとっての理想的な環境とは、生産性・信頼性の向上に代表されるように一般には大変経費がかかるソフトウェアを、優れたコストパフォーマンスで開発し終えることにある。とすれば、他の企業のツールよりも優れたツールが必要であるわけで、他の企業と同じ程度のツールを利用するのであれば、ツールが無かったときの状況となんら変わるところは無いといえる。

さて、社会においてのソフトウェア開発に関する理想的環境は、人間の本能的な狂気の進化論に沿って、より便利で快適な方向、いいかえれば怠惰の促進される方向へ向かっていくことは間違いないと思われる。このパワーの源泉はタテマエ世界の関東人好みの「人類愛」もしくはホンネで生きている関西人の好きな「金類愛」つまり欲望の果てということになる。

考えてみると人類愛から出発するのは、根底にＳＥがより快適に仕事のできる環境を提供しようという考えであり、いわば開発担当者の立場から見た理想的環境のことである。また、金類愛から出発するのは管理もしくは経営的な観点から見た理想的環境ということができる。したがって、いずれにしても開発環境が立場の違いを越えてより良くなっていくことは必然的な結果である。問題はどのように良くなっていくかである。ＳＥが自分自身の開発環境を改善または改革していくことは、よりＳＥにとって理想的な環境を取り上げることになる。従って、ＳＥの立場から見れば、技術の進化（人間の退化と同義かも知れないが）が避けられないものであるならば、管理的な観点から提供されたツールでなく、ＳＥの好みのツールを採用したいのは間違いない。例えば、ＳＥの好きな環境の一つであるＵＮＩＸを例にとると、ＳＥの欲望をかなり満たしているが、管理者や経営者に対してはほとんど考慮されていない。従ってこういう開発環境についてはどのように管理すれば良いかということを、考えなければならないのである。

いずれにしろ、本当に理想的な開発環境とはこれらのすべての立場の人々を満足させるものであろうが、それはどのようなものであればいいのだろうか。今回お話しいただいたプレゼンテータの報告や参加者の討議の中から、われわれＳＥにとって有益なキーワードを読み取り、近い将来多くのＳＥが楽しい開発環境で作業をしていることを望みたい。しかし、その成否は多分に我々自身にかかっているのではなかろうか。