SEAMAIL

Newsletter from Software Engineers Association

Volume 3, Number 3-4 1988

目　次

編集部から		1
第6回　ソフトウェア信頼性シンポジウム（報告・その1）		2
Opening Session		
ソフトウェア品質指標の標準化に関するIEEE試案の位置づけと評価	鳥居宏次／岸田孝一	2
日本的品質管理とソフトウェア	保田勝通	6
ソフトウェア・メトリクスとモデル	大場　充	12
第3回　実践的ソフトウェア開発環境に関する集中討論（報告・その1）		18
Session 1　マン・マシン・インタフェイスの改善		
発表1　Summitの発想	青木　淳	18
発表2　ユーザ・インタフェイスに関する一考察	吉村智香子	24
発表3　ERASについて	新田　稔	28
発表を聞いての感想	参加者アンケート	32
討　論		36
討論を聞いての感想	参加者アンケート	44
九州支部設立報告	平尾　一浩	50
Hackerに関する考察	浜野　剛至	60
不純人工知能礼賛	柳田　俊一	64
ウォータフォールモデルの変形表現	細野　広水	66
幹事会／総会／その他の報告		67
Call for Papers/Participation		71
Forum in Yokohaka: "Hyper-Textをめぐっての論議　Part 2"		71
第4回　ソフトウェア技術交流会 in 徳島		72
11th International Conference on Software Engineering		73
ソフトウェア・シンポジウム'89		74
ソフトウェア信頼性の定量的管理に関するワークショップ		75
第6回　夏のプログラミング・ワークショップ（若手の会 in 八ヶ岳）		77
第2回　ソフトウェア教育に関するワークショップ		79
第8回　ソフトウェア信頼性シンポジウム		81
第4回　実践的ソフトウェア開発環境に関する集中討論		83
会員状況		85

ソフトウェア技術者協会（ＳＥＡ）は，ソフトウェア･エンジニアの，ソフトウェア･エンジニアによる，ソフトウェア･エンジニアのための団体であり，これまでに日本になかった新しいタイプのプロフェッショナル･ソサイエティたることを目指して，１９８５年１２月２０日に設立されました．

現在のソフトウェア技術が抱える最大の課題は，ソフトウェア･エンジニアリング研究の最前線（ステイト･オブ･アート）と，その実践状況（ステイト･オブ･プラクティス）との間に横たわる大きなギャップを埋めることだといわれています．ソフトウェア技術の特徴は，他の工学諸分野の技術にくらべて属人性がきわめて強い点にあります．したがって，そうしたテクノロジー･トランスファの成否の鍵は，研究者や技術者が，既存の社会組織の壁を越えて，相互の交流を効果的に行うためのメカニズムが確立できるか否かにかかっています．ＳＥＡは，ソフトウェア･ハウス，計算センタ，システム･ハウス，コンピュータ･メーカ，一般ユーザ，大学，研究所など，さまざまな職場で働く人々が，技術的･人間的交流を行うための自由な＜場＞であることをを目指しています．

ＳＥＡの具体的な活動としては，特定のテーマに関する研究分科会（ＳＩＧ）や地方支部の運営，月刊機関誌（ＳＥＡＭＡＩＬ）の発行，各種のセミナー，ワークショップ，シンポジウムなどのイベントの開催，既存の学会や業界団体の活動への協力，また，さなざまな国際交流の促進等があげられます．

なおＳＥＡは，個人参加を原則とする専門家団体です．その運営は，つねに中立かつ技術オリエンテッドな視点に立って行われ，特定の企業や組織あるいは業界の利益を代表することはありません．

代表幹事：　岸田孝一
常任幹事：　臼井義美　久保宏志　熊谷章　佐藤千明　藤野晃延　松原友夫　吉村鉄太郎
幹事：　青島茂　稲田博　岡田正志　落水浩一郎　片山禎昭　川北秀夫　杉田義明　鈴木弘　武田知久　田中慎一郎　長井剛一郎　中園順三　中野秀男　西尾出　野村敏次　野村行憲　針谷明　深瀬弘恭　藤本司郎　村井進　盛田政敏
会計監事：　辻淳二　吉村成弘
常任委員長：臼井義美（技術研究）　久保宏志（企画総務）　藤野晃延（会誌編集）　杉田義明（セミナー･ワークショップ）
分科会世話人　環境分科会(SIGENV)：田中慎一郎　渡邊雄一
管理分科会(SIGMAN)：相沢圭一　川北秀夫　芝原雄二　野々下幸治
教育分科会(SIGEDU)：大浦洋一　杉田義明　中園順三
再利用分科会(SIGREUSE)：青島茂　阿倍正平　村井進
ネットワーク分科会(SIGNET)：青島茂　野中哲
法的保護分科会(SIGSPL)：能登末之
ＣＡＩ分科会(SIGCAI)：大木幹雄　寺嶋裕一　中谷多哉子　中西昌武
ドキュメント分科会(SIGDOC)：田中慎一郎　野辺良一
支部世話人　関西支部：臼井義美　盛田政敏
横浜支部：熊谷章　林香　藤野晃延　松下和隆
長野支部：小林貞幸　佐藤千明　細野広水
名古屋支部：岩田康　鈴木智　西村亨
九州支部：植村正伸　小田七生　藤本良子　平尾一浩　松本初美　中島泰彦　能見巧　後藤芳美
ＳＥＡＭＡＩＬ編集グループ：大槻亮人　岸田孝一　佐原伸　芝原雄二　関崎邦夫　田中慎一郎　長井修治　野辺良一　藤野晃延　渡邊雄一

ＳＥＡＭＡＩＬ　Ｖｏｌ．３，Ｎｏ．３－４　昭和６３年６月２３日発行
編集人　岸田孝一
発行人　ソフトウェア技術者協会（ＳＥＡ）
〒１０２　東京都千代田区隼町２－１２　藤和半蔵門コープビル５０５
印刷所　サンビルト印刷株式会社　〒１６２　東京都新宿区築地町８番地
定価　１，０００円

編集部から

1. 最後のいいわけ

創刊以来これまで SEAMAIL の編集を担当してきた岸田です．新年度の役員改選で，鈴木弘さんにかわって代表幹事に選ばれ，編集長のポストは藤野晃延さんにゆずることになりました．船便シンドローム解決のメドがまだ立っていないので，しばらくは，まだ，１編集ボランティアとして新編集長のお手伝いをさせていただきます．

この号は，最初の予定では，私がシンガポールの ICSE に行っているあいだに材料を揃え終わり，帰国後すぐ編集にかかって，ゴールデン・ウィーク明けには発行の予定だったのですが，テープから起こされた原稿の整理に予想外の時間がかかってしまいました．５～６月は，総会やシンポジウムなどイベントも多く，編集作業のための時間がなかなかとれなかったこともありますが，基本的には，テープから完全原稿までの距離が遠いということです．

2. 信頼性シンポジウム

去年１０月の第６回信頼性シンポジウムから，オープニング・セッションの記録を，とりあえずお届けします．原稿作成にあたっては，シンポジウム参加者の１人佐藤貴司さんにテープ起こしをしていただきました．

残りのセッション（品質定量化指標の設定・計測の具体化・結果の評価に関する連続パネル）は，いま，何人かのボランティアの方々にお願いして，磁気から紙への媒体変換プロセスの途中段階にあります．この春行われた第７回のテープも，大阪大学でやはり音から文字へのコンバージョンが進行中です（だれか，早く音声入力のワープロを製品化してくれませんか？）．次号以降に御期待ください．

なお，このシンポジウムは，御存じのように，大阪大学の鳥居先生が主宰されている高信頼性研究会と SAE 関西支部が協力して，約半年ごとの周期で開催しているものです．次回（第８回）は，１０月の末に予定されており，現在，一般からの発表者を募集しています．８１～８２ページの案内をお読みの上，ふるって御応募ください．

また，高信頼性研究会（というより，その母体になっている日本規格協会の鳥居委員会）では，いまソフトウェア信頼性の定量的管理のための標準的ガイドラインを策定中ですが，その試案の妥当性を現実のプロジェクトの状況と照らしあわせて議論するワークショップが，夏の終わりに計画されています（SEA 後援）．興味のある方は，７５～７６ページの参加案内を見て，お申込みください．

3. 環境ワークショップ

２月に長野でおこなわれた（暖冬のため残念ながら去年の長岡に続いて雪見酒はできませんでした）第３回環境ワークショップ（実践的ソフトウェア開発環境に関する集中討論）の記録から，第１セッション「マン・マシン・インタフェイスの改善」についての発表・討論・および参加者アンケートをレポートにまとめました．

このワークショップの場合は，長野支部のボランティア・メンバーの努力で，すでに全セッションの記録がコンピュータに入っており，あとは編集の手間だけです（これが，実は大変）．生の記録と参加者のポジション・ペーパを集めたワークショップ・レポートは近く刊行されます．お急ぎのかたは，そちらをお読みください．

なお，ここで発表された Summit およびロボット・プログラミング環境は，それぞれ先日のソフトウェア・シンポジウム’８８に，論文の形でまとめられています．詳細を知りたい方は，論文集をお求めください．

次回の環境ワークショップは，この秋神戸で開催の予定で，いま発表者を募集中です（８３～８４ページ参照）．

4. 最後にふたたび，原稿をよろしく！

この号では，九州支部のキックオフまでの事情を描いた平尾さんの力作ほか３篇を掲載しました．これで手持ちの原稿はほぼ底をつきました．

藤野新編集長のもとでの今後の SAEMAIL 発行予定も，会員のみなさんからの原稿がどしどし寄せられないかぎり，これまでのペースを大幅に改善できるとは思えません．

支部や分科会の議論の記録，技術的な小論文，随筆，書評，愚痴，呼びかけ等，何でも結構です．なにぶんともに，よろしく．

第6回 ソフトウェア信頼性シンポジウム（報告・その1）

Opening Session

1987年10月23日　AM 10:00～12:00
於　大阪ガーデンパレス

チェアマン：　**浦野義頼**（国際電信電話）
報告者　：　**鳥居宏次**（大阪大学）
岸田孝一（ソフトウェア・リサーチ・アソシエイツ）
保田勝通（日立製作所）
大場　充（日本アイ・ビー・エム）

浦野：　このシンポジウムも今回が第6回目，東京から大阪へ会場を移して4回目になります．あとで主催者の鳥居先生からご紹介いただきますが，ソフトウェア品質指標に関する IEEE の標準化試案というものが，アメリカで作られておりまして，鳥居先生がやっておられる日本規格協会の委員会のメンバーを中心に，8月の末に伊豆でその内容を検討するワークショップを開きました．今回のシンポジウムは，このワークショップの成果を踏まえて，IEEE 試案を肴に，ソフトウェア品質の定量的管理について，いろいろ討論しようと考えています．

オープニング・セッションでは，まず，鳥居先生他の方々にチュートリアル風のお話をしていただき，その後の討論の話題を提供をしていただこうと思います．

ソフトウェアの品質指標の標準化に関する IEEE 試案の位置づけと評価

鳥居　宏次／岸田　孝一

鳥居：　おはようございます．これだけたくさんの方が集まられることは，まったく予想外でした．1968年に，有名な NATO のソフトウェア工学ワークショップがありましたが，この会合が，それに似たものになって，ひょっとしたら何かこう将来に対するインパクトを与えられるかなと，さっきから思っております．

今年の春にモントレーの ICSE へ行ったときに，IEEE で品質定量化のワーキング・グループを主宰されている Schneidewind 先生にお目にかかり，お互いにやっていることを話しあって，あちらでいま検討中の試案（この1月に作られた Version 12）をもらってきました．たまたま，私がやっているソフトウェア信頼性委員会の活動テーマとうまくフィットするような感じなので，今年これを取り上げてみようかということになりました．

IEEE のこの試案作りは，すでに3～4年から始まっているようです．先々週，東京で開かれた COMPSAC の会場で，NBS の Osborne という黒人女性に会いました．かの女がたまたま Schineidewind 委員会のメンバでして，その後2月に出た Version 12b を持っていました．両方を比べると，内容はかなり変わっています．そして，かの女の話では，いま，さらにもう1つ新しいバージョンが走っているそうです．

「なぜそんなに変わるのか？」と聞くと，委員会といっても，要するにボランティアの集まりですから，お互いに E-mail で意見をどしどし交換しあう．そして，誰かが幹事役になってみんなの意見を編集し，適当なタイミングで全員に配布するという，私どもからすると考えられないようなスタイルで，作業が進んでいるようです．

来年の春までに，試案としては一応完成させる目標だったらしいのですが，いまの状況では，いつ終わるのか誰にもわからないと，Osborne 女史はいっていました．私どもとしては，別に，そうした向こうの作業ペースは気にすることなく，あくまで1つのたたき台として利用させてもらって，日本なりのソフトウェア信頼性定量化試案を作り上げていこうと思っています．

いま，お手もとに原文のコピーをお配りしたこの試案を，はじめて見た方は驚かれると思いますが，概念的なフレー

ムワークの絵は描いてあるのだけれど，内容はまだほとんど詰っていません．例題などもきわめて不十分です．伊豆のワークショップでも，そのことがかなり問題になりました．フレームワークは，図 1-1 に示す通りですが，参加者それぞれの環境や背景の違いから，この図の解釈についても，ずいぶんいろいろな意見が出ました．

今回，みなさんにお配りしたワークショップ・レポートは，そうした討論の結果を含めて，参加者全員にポジション・ペーパを整理をしていただいたものです．

私のこの場での担当は，伊豆ワークショップでの討論の概要を報告することなのですが，この仕事は，おそらく私より，ワークショップの世話人をやられた岸田さんのほうが適任だと思います．よろしくお願いします．

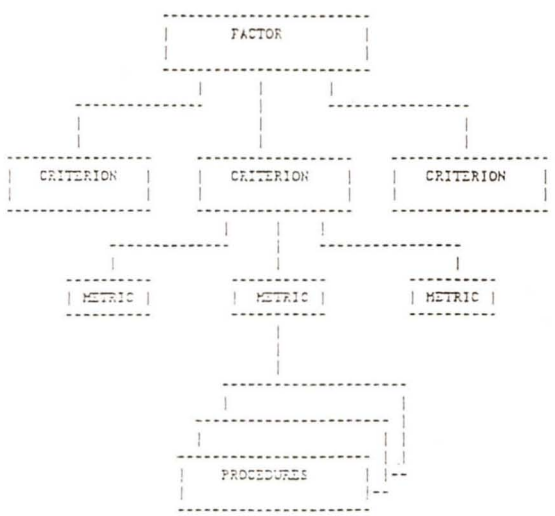


図 1-1

岸田： という突然の御指名ですので，しかたなく報告させていただきます．いいかげんな大学の先生とつきあっているとひどい目にあうという，格好なサンプルになったような感じがします（笑い）．

(1) IEEE の試案の視点

まず，ソフトウェアの品質に関して，この IEEE の試案が，どういう立場でものを考えているのかということから，お話したいと思います．一般的にいえば，ソフトウェアを使うユーザの視点と，ソフトウェアを作る開発者の視点の２つがあります．

表 1-2 は，試案の中に書いてある文章を取りだして，日本語に訳してみたものですが，IEEE の試案は，どうやら後者の方の立場，つまり，ソフトウェアを作ったり，テストしたり，あるいは動かしたりする環境や，そういう仕事をするプロセスとの絡み合いで，品質というものを考えようという立場をとっているようにおもわれます．

ユーザの視点	ソフトウェア製品が，望ましい属性の組み合せをそなえている度合い
開発者の視点	ソフトウェアの購入，開発，検証（検査），および稼働のための環境や関連するプロセスの関数

表 1-2 ソフトウェアの品質

ソフトウェアの開発・稼働環境，あるいは，それに関連するプロセスと，そこから作り出されるソフトウェアの品質との関連は，非常に複雑であり，したがって，こうした観点からは，品質というものを，なかなか，はっきり定義することができません．それは，いいかえれば，品質に関する要求仕様がきちんと書けないということになります．

このように土台が固まっていない状態では，いくらいろいろなデータを取ってみても，データそのものの信憑性，あるいはデータを加工して得られる統計的指標の妥当性がよくわからない，という根本的な問題が生じます．

一口にソフトウェアといっても，パソコンのソフトウェアから，大型機のソフトウェアまでいろいろありますし，そのソフトウェアをつくるプロセスもさまざまです．プロジェクトにもいろいろな形態があります．ということから，品質に関するわれわれの議論はいつもかみ合わない．

しかし，いつまでもこのままではいけないので，なんらかの共通の考え方，少なくとも概念上の共通の枠組みを設定しておく必要があるのではないかというのが，この試案を検討している人たちの姿勢であるように思われます．

その共通の枠組みとはいったい何かといえば，（１）品質の要求仕様をきちんと定め，（２）その仕様にもとづいて適当な指標を設定し，その指標の計測を実際に実施する．そして，（３）測ってみた結果，その指標が妥当かどうかをきちんと検証する．こういう３つのステップは，どんな場合にも必要だろうし，そういうことをやるための方法論を標準的に考えて，そのためのガイドラインを与えよう，ということが目的のようです．

(2) IEEE 試案のフレームワーク

図 1-1 に示してあるのが概念上のフレームワークです．これは昔からある McCall や，Boehm のモデルと同じものですけれども，ソフトウェアの品質を３段階にわけています．トップレベルに要因（Factor）があって，それをいくつかの判断基準（Criteria）に分割し，そしてそれぞれの Criteria に対していくつかの指標（Metrics）が対応する，という階層構造を考えています．

Factor とは，いってみれば，マネージャが品質というものを考える場合のコンセプトです． Criteria と Metrics というのは，実際に開発にあたるエンジニア・レベルの問題です．これらの３つの階層の項目の間に，マッピング（対応づけ）を行なっていくのですが，そのための標準的なガイドラインを示そうという意気込みは，大変よいことのように思えます．

ただ，伊豆でもいろいろ議論があったのですが，この図 1-1 の意味することは理解できる．しかし，実際に品質の Factor として何と何を考えるのか，その Factor はどのような Criteria に分解されるのかが，現在の段階の試案では，まだ具体的なかたちでは述べられていません．

たとえば，Factor に関しては，このようなものがありうるという例しか示されていません．これもたくさんあり過ぎて，素人は困るんじゃないかなと思いますし，Criteria に関しても，同じような実例が，何ページかにわたって列挙されているだけです．それぞれの間のマッピングは，まだ全然示されていません．

手続き上のフレームワークは，

- 品質要求の Establishment（確立・設定）
- Metrics 計測の Implimentation（具体化・実施）
- 品質 Metrics の Validation（妥当性検証）

という３つのことの標準的な方法論上のガイドラインを与えることを意図しています．試案の第５章に，手続き上のステップ（の名前）だけは，かなりきちんと書いてあります．

伊豆ワークショップの報告書の中では，日本電子計算の野村さんが，これに従ってやってみようということから，日本語に訳されていますが，しかし，試案の本文を読んでみると，１つの項目について１行ぐらいしか説明が書いてないところが多かったりして，実際のガイドラインとしては，まだ不十分だということがわかると思います．

試案の最後には，付録がいくつかついています．先ほどの Factor や Criteria の例は付録 B に入っていますが，付録 A には，Complete Examples という立派な見出しがついています．これは，IEEE の方法論にもとづいて実施されたプロジェクトの完全な実例集ということです．

その A-1 は，国防省関連のあるシステムの実例です．しかし，A-2 以下はありません（笑い）．A-1 自体もしりきれとんぼで終わっていて，後ろの方は空白です．まだ，この状態では，Incomplete Example といったほうがよい（笑い）．作業途中だからしょうがないのですけれども，そういう状況です．

(3) 伊豆での討論

そうした不満はあっても，この試案を作ろうとしている人たちの意気込みはなかなかよいし，全体の方向もそんなに狂っていないという認識のもとに，伊豆でいろいろと議論をしました．

ワークショップの冒頭では，早稲田大学の東先生においでいただいて，ISO で現在進められている標準化の話をしていただき，それを IEEE 案と比較しました．参加メンバーのなかには，日本的なソフトウェア QC をやられている方が何人かいて，それと IEEE 試案との対比という議論もありました．

また，伊豆に集まる前に鳥居先生から出されていた宿題は，それぞれ自分の身のまわりのプロジェクトを取り上げ，そのプロジェクトにおいて，仮にこの IEEE のフレームワークで品質の定量化を実施したらどうなるのか，思考実験をしてみて，その結果を持ち寄りましょうということになっていました．何人かの方が，非常にまじめにそれをやってこられました．

たとえば，SRA の松村さんのものは， IEEE 試案の付録 A-1 と全く同じことを，銀行オンラインのアプリケーションについて展開してみたというものです（SEAMAIL Vol 2, No 10-12 参照）．そういう試みを実際にやってみた結果として，当然のことですが，現状のこの骨組みだけしかないガイドラインでは非常にやりにくい．できればガイドラインとしてはこの点を充実して欲しい，という具体的な要望事項が何人かの方から出ました．

しかし，それにもかかわらず，こういうふうな標準的な枠組みの存在は，品質に関する議論を展開するためのコミュニケーション・プロトコルとして非常によいということで，みなさんの意見が一致したと思います．

(4) ISO/TC97/SC7

少し個々について触れます．

ISO で現在作業しているのは，基本的にユーザの視点に立って，ソフトウェアの品質特性（Factor　でなく，Characteristics と呼んでいますが）の標準化をしようというものです．IEEE 試案よりも数を少なく整理して，次の６項目にまとめています：

- Functionality
- Reliability
- Ease of Use
- Efficiency
- Maintainability
- Portability

これらの６項目を，それぞれさらに複数の Sub-Characteristics に展開し，きちんと標準を決めようというわけです．手続き上のフレームワークとしては，Characteristics の定義（Definition），計測（Measurement），Rating，アセスメント（Assessment）というようなモデルを考えています．品質の Characteristics の定義を標準化しておいて，そこから後の方は，プロジェクトによって異なるというアプローチです．

ですから，ここでのポイントは，品質の問題を考える場合に，ISO のようにユーザの観点に立つのがよいのか，あるいは IEEE のように開発者の観点に立つのが適切か，もちろん最終的には両方とも必要なのでしょうけれども，それぞれの立場にどういう特徴があるのかを，よく見極めておく必要があるということです．

(5) 日本的 QC

日本的な品質管理との絡み合いでは，IEEE 試案にのっている国防省関連の例題（しかも，これが現状では唯一の例題です）が，非常に問題になりました．やっぱりアメリカでは，何でもかんでもみんな国防省ではないか．同じ受託開発でも，日本では民間企業が相手だし，まして，ソフトハウスが自分でプロダクトを作る場合だとか，メーカが OS を開発する場合とかは，かなり事情が異なる．とりあえずの例題が DOD １つしかないということは，どうもこの試案の狙いが少しわれわれの考えているものとは違うのではないか，という話が出ました．日本の QC はもっと幅が広い．

IEEE の試案はどちらかというと，ニュアンスとして，マネージャの意志決定のもとづいて，トップダウンに品質の定量化を実施するという形になっています．しかし，日本的 QC はご存じのように，草の根の QC サークルから出発するボトムアップ形式です．どちらがよいのかということで，日本の風土では，やはり，ボトムアップのほうが似合っているという意見が支配的でした．

日本的 QC では，品質特性とか代用特性といったガイドラインがありますが，それと IEEE の Factor や Criteria が，１対１にマッピングできるのかどうかも論議されました．先ほどいいましたように，IEEE 試案は，現状ではまだ，こうした点で具体性に欠けています．

鳥居先生の委員会で，今後日本的な標準化試案を作られるとしたら，豊富な具体例をつけてほしい，という要望がでました．こうしたガイドブックを読むとき，現場では，例題編から先に読みはじめます．大学の先生がトップダウンだとすれば，現場はどうしてもボトムアップなので，そういうふうな人たちに使えるような試案にしてほしいということでした．

(6) フレームワークの重要性

といった具合いに，IEEE 試案の現状についてみんなが文句をいったわけですけれども，たとえ不完全ではあっても，こうした概念的な枠組み上のたたき台があると，その上にのっかって，大学の先生も，メーカの OS 開発者も，パソコン・ソフト屋さんも，あるいはエンド・ユーザのわけのわからない要求に日夜振り回されているアプリケーション開発者も，いろいろな立場の人々が，それぞれ自分の事情は事情として，お互いに技術的なコミュニケーションができる．そのことが証明されたというのが，こんどのワークショップの最大の収穫だったと思います．

じゃあこれからどうするということですが，もちろん，鳥居先生やIEEEの方にがんばっていただきたいのですが，われわれとしても，いずれは何か出てくるだろうと，ただ受け身で待っているだけではいけない．みんなの意見をどんどん出して，それを IEEE とか，鳥居先生の委員会になるべく幅広く吸収してもらおう，今回のシンポジウムも，そのための場として設定されているのではないかと思います．

鳥居： ありがとうございました．いま最後におっしゃていただいたのですが，今回のこのシンポジウムで，現場からの声をなんらかの格好で吸い上げていきたいと思っていますので，単にお話を聞いて帰るだけでなくて，何でも結構ですから積極的にコメントしてください．

日本的品質管理とソフトウェア

保田　勝通

保田：　日立の保田（やすだ）でございます．タイトルがちょっと大きすぎたと思うのですが，私なりに，わが国におけるソフトウェアQCの現状を整理させていただきます．

(1) ソフトウェア QC の現状

昨年の今頃ですが，日経コンピュータに掲載されたソフトウェア品質管理実態調査という中から，適当に抜き書きしてみたのが図 2-1 です．

SW 品質管理部門あり	17%
システム部門で TQC 実施	36%
小集団活動の導入	12%
システム開発の工程管理	25%
ドキュメンテーション管理・レビュー	27%
基本設計レビュー	37%
構造化プログラミング	30%
プログラムの部品化	24%
テストカバレージ測定	5%
バグ曲線等でエラー統計分析	3%
本稼働後のエラー発生件数を記録	5%

－稼働後の平均エラー発生件数＝ 0.2 件／KS

図 2-1 ソフトウェア品質管理実態調査

この調査は，一部上場企業のシステム部門を主な対象にしたものです．総論的にいえば，，約３分の１の会社で TQC を実施しており，品質管理専任者がいるところが２割弱．基本設計レビュー，構造化プログラミング，ドキュメンテーション管理とかは，他のにくらべると高い数字がでていて，まあまあ進んでいる，ということができます．

しかし，統計的品質管理の基本となるような，たとえば，本番稼働後のエラー発生件数の記録とか，バグ曲線とエラー分析とかの実施度合いが低いのは，ちょっと気になるところです．

(2) 日本的 QC のアプローチ

ソフトウェアの品質とは何かといえば，それは「ソフトウェアまたはそれがもたらすサービスが利用者の目的を満たしているかどうかを決定するための評価の対象となる固有の特質および性能の全体」だと定義されます．これをもう少し簡単にすると，石川馨先生の表現を借りれば「品質とはユーザの満足度の度合である」ということになります．

品質のよしあしを考えるとき，ハードの場合ですと，製品の規格がかなり明確になっていて，その通りに製品ができているか，ということがポイントです．ところが，規格というのは，たいてい後追いになるわけでして，それができた頃には，ユーザ・ニーズはもっと先の方へ進んでいるといったことが，よく起こります．

ハードウェアでいう製品規格は，ソフトウェアの場合には，仕様にあたります．プログラムの品質のよしあしは，それが仕様書通りに動いているかどうかで判断される．ただ，ほんとうの要求仕様が，ユーザ自身にもよくわかっていないということがありますので，ユーザ・ニーズがうまく仕様書に取り込まれているかが，かなり大きな問題だと思います．

日本的 QC の特徴を，石川先生は，「全員参加，品質第一，消費者指向，事実・データで物をいう，次工程はお客様」という一連のキャッチフレーズで示されています．

まずは，「全員参加」ということですが，QC 部門だけでなくて，間接部門，管理部門，それからトップからも積極的な指示も必要だし，担当者からのボトムアップの参加も重要です．これは，いわゆる QC サークルの形です．

２番目の「品質第一」．よく自動車とかが例にあげられますが，アメリカなどでは，短期的な成果特に利益が優先されすぎて，長期的にはうまくいかないということがあります．ソフトウェアの場合，特にそうした傾向が強いと思います．不良品の発生による手直しは，場合によっては，全体コストの半分くらい占めている場合もあるわけで，手直しが少なくすめば生産性は向上するし，納期もきちんと守れるということで，日本的 QC では，品質第一が重要な合い言葉になっています．

３番目の「消費者指向」は，マーケット・インともいわれます．この反対概念として，プロダクト・アウトというういい方があります．メーカの技術者は，一般に自信過剰で，オレがつくった物はいいのだから使えという傾向があり，えてして市場の反発を買います．本当にお客様がなにを欲しているのかをよく調べなければならないということです．

４番目の「事実・データで物を言う」は，ソフトウェアの場合はなかなかむずかしいと思いますが，事実をよく見るのは大切なことだと思います．

それから最後の，「次工程はお客様」というのは，お客と直接接触している営業第一線の人以外にも，マーケット

指向の感覚を身につけさせるために，次の工程をお客様だと思って責任を持ってやりなさい，ということですね．ハードウェアの場合ですと，明確に，設計・製造・検査とか分かれていますが，ソフトウェアの場合は，多少手工業的に自分でやってしまいますので，次工程というのがピンとこないのかもしれませんが，少なくともやるべきことをやるべき時にきちんとやっていく，という意味にとらえればいいのではないか，と思います．

管理とはどういうことかを，正面きっていうのは難しいわけですが，「目的を継続的に効率よく達成するための活動」というふうに考えます．そのために，いわゆるデミング・サークル Plan-Do-Check-Action をうまく回していくことが必要です．特に，Plan-Do は当然やらなくてはなりませんが，Check & Action をするというのが，日本的 QC ミソだと思われます．

(3) 日本的ソフトウェア QC の特徴

以上はふつうの QC 談義で，格別問題はないと思いますが，ソフトウェアにおける日本的QCということになると，さまざまな意見がありえます．以下に申し上げるのは，私なりの少し独断が入った考えです．

日本的なソフトウェアの QC は，どちらかといえば，メーカが従来の品質管理を導入したのとおなじような状況を，あえて想定した上で進められています．図 2-2 に，ソフトウェアの品質管理の特徴を，欧米と日本とで簡単に比較してみました．

日本	欧米
工場制度の導入 ３権分立 （品質，工程，原価）	プロジェクト制度 プロジェクト・マネージャ
検査の導入 ・実際の検査行為による合否の判定 ・品質保証活動のかなめ	日本的検査の概念なし
中間品質管理 ・テスト工程の不良管理	品質データ収集が困難
ライン部門の活動が主体	品質管理スタッフ主体
QC サークルの導入	なし？

図 2-2 ソフトウェアの品質管理の特徴

日本では，ソフトウェアを作るときに工場制度を導入しています．いわゆるソフトウェア・ファクトリです．この特徴は，品質・工程・原価の３要素を，それぞれ別の部署で管理できるということです．それに対して典型的な欧米でのやり方は，プロジェクト制度でして，日本でもソフトウェア・ハウスではかなり取り入れられているようですが，ほとんどの管理をプロジェクト・マネージャが統括しています．

ソフトウェアの検査は，最近かなり普及してきているようですが，これにもいろんなレベルがあります．ここで検査というのは，日本的 QC の枠組みにおける検査を意味しており，開発部門や設計部門とは別に，検査部門で行なうもののことです．この検査部門がラインの最終過程を担っており，品質保証活動の要になります．

次に中間品質管理ですが，テスト工程において，不良率を中心に考えた QC 的な管理が，日本ではけっこううまくやられているようです．欧米の場合は個人主義的な色彩が強く，個人が不良率を出したがらない，データがなかなか集めにくいという話を聞いたことがあります．

それから，先ほどいいましたように，日本的 QC は，ライン部門の全員参加という形で進められますが，欧米の場合には，品質管理スタッフという専門職がいます．その人たちは非常によく勉強しているのですが，いわゆるスタッフ意識が強いように思います．

(4) 検査のあり方

JIS で定義している検査は，非常に限定的な意味になっていまして，最終的に合否の判定をすることを指しています．ソフトウェアの場合には，テストという言葉がありますが，テストと検査はどうちがうのかという質問をよく受けることがあります．これは，図 2-3 を見ていただければわかるでしょう．

テストも検査も，その技術的な内容は，プログラムの静的または動的な特性を調べることです．ここに，バグというものが絡んできます．テストの目的は，それをなおすことです．一方，検査の場合は，最終的に合否を判定すること，つまり悪い製品は外へ出さないという判定が目的です．ここが違うんだと思います．

したがって，検査がどうあるべきかを考えると，合否判定を公正にするために，検査部門を設計・製造部門から独立した組織にしないといけない．この独立組織のよい点は，お客さんの観点に立った検査を行なうことができることで

す．所定の基準をクリヤーできない製品は合格させないわけです．

項目	説明
検査組織	製品の合否判定を公正にするため，設計製造部門から独立した組織とする
検査の職務	1) 顧客の立場に立った検査を行い，所定の品質に適合しない製品を合格，納入させないこと． 2) 製品納入後も，顧客の立場に立って，品質を保証するための活動を行なうこと
検査の活動範囲	1) 社内 ・工程別検査 ・基本設計から最終検査までの品質管理の総まとめ 2) 社外 ・品質保証活動 ・顧客ニーズ，クレームの窓口

図 2-3 ソフトウェア検査のあり方

検査部門に，このように強い権限を持たせた場合には，それなりの責任も与えなければいけません．たとえば，日立の場合には，ソフトウェアの出荷を承認した検査部門が，顧客に対する責任を取るようになっています．品質保証の窓口としての役割を担当しているわけです．

もちろん，バグ（クレーム）が出たら，それを実際になおすという作業は，あくまでも設計・製造部門がやります．検査部門は，ソフトウェアを自分でいじってはいけません．お客さんに対して一定の品質を約束し，迷惑をかけないようにするために，社内でのフィードバックをきちんと組織的におこなう部署として，存在しているのです．

社内における工程別検査は，ハードウェアの場合は，いろいろな部品の検査に相当しますが，ソフトウェアの場合には，たとえば，設計段階での中間生産物としての各種ドキュメントを検査するという形になります．最終的な検査としては，単なる選別検査でなく，基本設計から最終検査までの品質管理の総まとめをおこないます．

社外に対しては，品質保証活動があります．本来顧客のニーズを聞くためには別の部門があるのですが，クレーム処理に関連して，お客さんと折衝することになりますから，ニーズも含めてあわせて吸収していくようにしています．

中間工程の品質管理というのは，比較的分かりやすいといえます．図 2-4 に例をあげましたが， Plan-Do-Check-Action で，テスト工程だけで閉じた形で書いています．

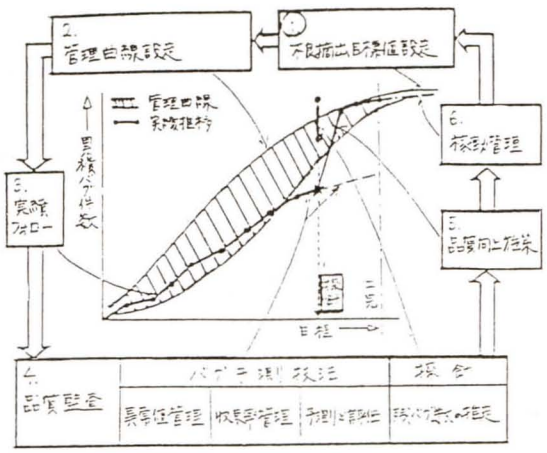


図 2-4 品質目標値管理の概念

横軸に日程を取っていますので，コンパイルが終わってテストをはじめるという時点が，工完となっています．縦軸は累積のバグ件数をとっています．そうしますと，かならず上昇曲線になるわけです．

ここでは不良だけに限定していますが，最初の Plan の段階では，目標値を決めなくてはいけません．どう決めるかは少し難しい問題ですが，まあ，エイヤでもいいから，ともかく「不良摘出目標値」を決めなければいけない．

その後，実際の管理作業には，工程がかなり長いですから，ガイドラインが必要になります．ハードウェアでよくやっている管理図や管理限界とに相当する考え方をとって，管理曲線を設定します．パターンとしては S 字形です．ここにある程度幅をもたせて，上限・下限というように目標値に対してカーブを描きます．ここまでが Planning にあたります．

そして，テストが始まったら，結果をプロットして行きます．その後，結果をどう評価して行くかですが，管理限界を飛び出ている場合は警告を出します．ここで難しいのは，品質がよいからバグが出ないのか，それともテストをさぼっているのかが，よく判断できないということです．

ただ，実際には，データをちゃんと取っていれば，だいたいわかります．それでも水掛け論になってきて，検査・

設計両部門間の力関係で決着がついたりします．そのさい，一種の抜取り検査を事前にしておき，これだけあるじゃないか，と証拠を見せて納得させるやり方も考えられます．

いずれにせよ，ここで Check をしているわけですね．最終的には，さらに Action が必要なのですが，それは品質向上施策になります．ここで諦めてはいけないので，バグが出てきたらばいろいろとフォローをします．

図 2-4 に示した例程度ならいいのですが，すっと気持ちよくバグが伸びて行く場合は問題で（笑い），不良目標値がおかしかったのではないか，プロセスの過程に問題があったのではないかとか，いろいろフィードバックして行かなければなりません．これが典型的なソフトウェア開発における品質管理です．

(5) 再発防止

不良の再発防止が大切です．私の考えでは，日本的 QC のポイントはこのあたりにあると思います．

要するに，フィードバックをきちんとして，本質的な原因をきちんと直していこうということです．どういうふうに考えればいいかといいますと，プログラムの問題ではなく，プログラマの問題ではないかと考える．さらにいえば，プロセスをきちんとなおしておかないと，類似の不良が再発することは防げません．

そのときの基本的なアプローチは，不良をどうして作り込んでしまったのかを徹底的に追求して行くことです．事前に発見できなかったことに問題があるわけですから，それはなぜか？という問いかけが必要なのです．そのさい，あまり精神論にならず，制度や仕組みのレベルで改善していけるほうが望ましい．

とはいっても，ソフトウェア開発の現状を見ると，非常に人間に依存している部分が多く，いくら仕組みや規則を作っても，すぐ抜け道を見つけられてしまう．チェックリストは立派なものがあるのに，いいかげんに○をつけるというようなぐあい（笑い）．規則なんて，作り出したらきりがない．あまりないほうがいいわけです．

とすると，基本的にやらなければならないのは，規則以前の問題になります．いわゆる「しつけ」だとか，教育といった側面も重要になってきます．

個人で仕事をしているなら，これは単に技術力の問題でしかありません．しかし，組織でやっている場合には，制度とか仕掛けとか技法だとか，いろいろなカバーの仕方があるはずです．個人ができなかったり，知らなかったからといって，それを簡単に技術力の不足だと総括するのは，いけないと思います．

(6) 品質機能展開の考え方

よいソフトウェアとは何だろうと，正面きっていわれると，なかなか答えにくいものです．さきほど，品質の定義は，ユーザの満足度だと申し上げました．それでは満足度とは何か，とさらに突っ込んで考える場合に，日本の QC 運動のなかで発明されたのが，品質機能展開といわれるやり方です．

品質検査あるいは保証をしっかりやって，ユーザからのクレームをうまく組織内にフィードバックし，不良の再発を防止することは，たしかに大切ですが，それだけでは，たとえば新製品開発のような場合には，ちょっと困ってしまいます．そうしたことから考え出されたのが，品質展開です．

これは，基本的には，図 2-5 に示したような設計アプローチにたよって，ユーザ・ニーズをきちんと把握して行こうという考え方です．

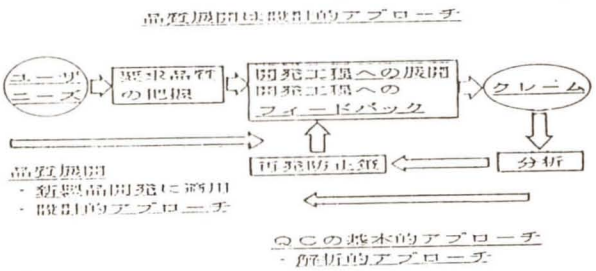


図 2-5 品質展開のアプローチ

ソフトウェアにおける品質機能展開の具体的な手順も，最近いくつか試みられています．ここでは，今年の３月，IPA 技術センタから出された報告をご紹介しておきましょう．図 2-6 がその概念図です．

まず，ユーザの品質に対する要求を定め，それをユーザが理解できるコトバで書くことから始まります．これは IEEE 試案でいっている Factor，または ISO でいっている Characteristics と同じレベルです．

次に，それらを，実際にソフトウェア開発者が計測できる特性に置き換えてやる必要があります．代用特性を決め，その一覧表を作ります．ユーザ視点からみた要求品質また

は品質特性と，代用特性とは，そもそも見方が違うわけですから，一般的には，両者はM:Nの関係にある．つまり，マトリックスによる表現が必要になってきます．

それをあらわしたものが，品質表といわれているものでして，要求品質を縦軸に，代用特性を横軸に書くことによって，それらの関係を示します．また，特定の工程において，代用特性はどういうふうに測定し，チェックするかを具体的に管理項目によって示したものを，QC工程表とか，期日管理工程表といいます．

ここまでくると，あとは，それらを実際の開発工程に適用して行くだけです．

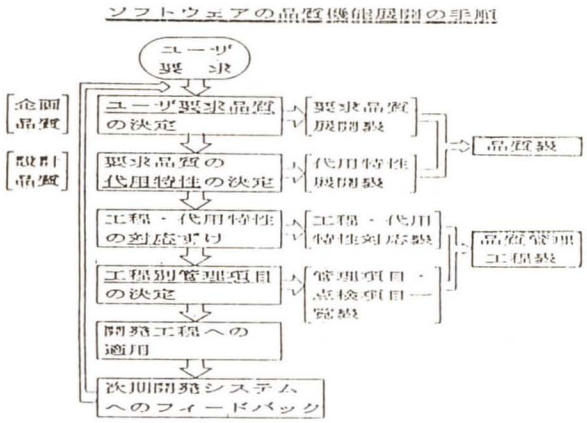
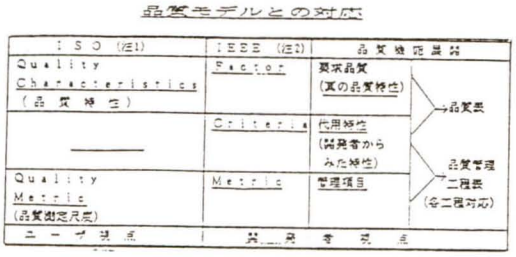


図 2-6 品質機能展開の手順

さて，IEEE 試案のフレームワークと，この品質機能展開との関係はどうかといえば，Factor はほぼユーザのレベルの品質要求に近いという対応づけができそうです．Criteria はあんまりはっきりしませんが，開発者向けという意味でほぼ代用特性と同じではないかと思われます．そして，Metrics は具体的に計測するものですから，管理項目に対応するのではないか．こんな対応づけができそうな気がします．

一方，ISO でいまやっている作業は， IEEE 試案のやり方とは違って，ソフトウェア品質というものの内容を，ユーザの視点に立って標準化しようというアプローチです．

そこでは，品質特性つまり Characteristics を６つに絞りこんでいます．しかし，これだけではあまりにも大まかすぎるので，直接扱いにくい．そこで，いま日本案では，２１個の Sub-Characteristics に展開しています．このレベルまでブレークダウンすれば，ユーザにもわかりやすい Metrics への対応づけが可能だろうと考えています．もちろん，それは必ずしもツリー構造ではなく，M:N のマトリクス形式になるでしょう．



(注1) ISO DP9126 Software Quality Characteristics (ISO/TC97/SC7/WG3)
(注2) ISO P1061 V.12 A Draft Standard for Software Quality Metric (IEEE Quality Metrics Standard Committee)

図 2-7 品質モデル間の対比

(7) 終りに

ソフトウェアにおける品質管理の歩みを振り返って見ると，最初は品質なんて考えずにとにかくものを作っていた．そうすると，いろいろ故障やクレームが出てきて，これは大変だということになった．そこで，いろいろな対応の仕方があるわけですが，１つの対策として，ハードウェアの製造と同じように，検査をきちんとやろうじゃないか，というアプローチが考えられた．

検査の目的は出荷品質の向上であり，選別検査的な色彩が強い．それを最終工程でやり，悪いものは関所を通さないようにする．そうすると，工程遅延だとか，開発効率の低下という現象が出てきます．一応品質だけはよくなるが，それでは企業として困るし，お客さんにも迷惑をかける．

そこで，もう少し前工程でやろうじゃないか，ということになります．かなり前のほうから不良を摘出して品質を上げて行く．そうすると，かなり工程的にも安定してきて，品質も向上する．ただし，不良を作りだすプロセスには目をつぶっておいて，結果を漏らさずチェックしようということですから，ライフサイクル全体で見れば別ですが，各工程の開発効率は逆に落ちる場合もある．

したがって，その次のステップとしては，設計の品質を上げて行こうということになります．つまり，事後のチェックではなく，事前にあるレベル以上の品質を作りこんで行こうという考え方です．こうすれば，かなり開発効率は上がってきます．

しかし，問題は一番最初にいったように，ユーザの満足度を満たすということですから，それだけではまずいわけで，品質機能展開のような形でユーザ・ニーズの正確な把握に努力して行く必要があります．そうすれば，いろいろ付加価値を含めて，多少の円高でも売れるような製品が作れるようになるでしょう．

(8) 質疑応答

Q： プログラム品質のチェックについてですが，日本でも，プログラマというのは，自分が作ったプログラムのソース・コードをチェックされることを非常にいやがる傾向がある．そうしたことに関して，何かお考えがありましたらお聞かせください．

A： ユーザの満足度を得るというのは，確かに大切なことではありますが，ユーザがもし完全に満足しきってしまうと，企業としてはもうその次の製品が売れなくなる．それでは会社が潰れてしまうという困った状況が生じるわけで（笑い），会社を長続きさせながら，それなりのユーザの満足を得て，社会に貢献していかなければいけない．

たとえば，保守性を問題にするときには，どちらかといえば，ユーザ視点というよりは，開発者の立場からみて，表面に出てきていない特性をどういうふうに扱って行くかということになります．これは，メーカ側としてはかなり真剣に考えなければいけないことです．

しかし，保守のことを考えてうまく構造設計をやりなさいということを，あんまりしつこくいってソース・コードを点検したりすると，プログラマたちは，たしかにいやがります．問題は，それにどんな意味があるかということを，うまく動機付けして，かれら自身が，これは何とかしなければならない，と思うようにもって行ければいいわけです．

それが QC サークルなどで可能になる．ただ，漫然とサロン的にやっているだけですと，企業目的に合わなくなってくるといけないので，トップダウン的な方針は必要だろうと思います．

Q： ユーザの満足度についてですが，ただ満足度といってもよくわからない．そうすると，何らかの代用特性を考えて行かなければならない．MTTRはその１例でしょうが，他にも，システムの生存率（どのくらいの期間，機能とか性能とかが満足できるか）といったもので，長期的にユーザの満足を高めるといった配慮も，高信頼性システムとして必要でしょう．ただ単にユーザ満足度といっているだけでは無理じゃないかと思うのですが．

A： ISO の６つの Characteristics には， Maintainability とか Portability が入っている．これはやはりライフサイクル的にみて，最終的にたぶんそういうものがいるだろうという考えです．ただ，具体的にどう展開するかは，これからの課題になっています．

ソフトウェア・メトリクスとモデル

大場 充

大場： 私の用意してきたプレゼンテーションのタイトルは，事前の案内とは少しちがって，「ソフトウェア・メトリクスとモデル」です．そういうことでお話しさせていただきます．

(1)プログラムの大きさと複雑さ

メトリクスは，基本的には管理のためのツールですが，もう1つ大きな目的があります．何か新しいアイデアが浮かんだとき，それにもとづくプロセスやそこから作られるソフトウェアが，どれくらいよいものになりうるかを，きちんと評価することが必要になります．メトリクスは，そうした評価のためのツールでもあるのです．

そういう観点から考えると，重要なメトリクスの第1は，プログラムの大きさです．ここでいうプログラムとは，単なるソースコードよりもう少し広い意味で，その裏にあるものも含んだソフトウェアの大きさをどうやって測るか，ということです．大きさという概念も少し比喩的で，複雑さとか，そういったものも含んでいます．生産性や品質のメトリクスも，この大きさにもとづいて定義されます．

プログラムの大きさについては，いくつかの古典的なメトリクスがあります．たとえば，LOC (Line of Code) とか，ステップ数などです．生産性でいえば，1日に何行書けるか（LOC/Programmer Day）とか，品質でいえば，1000 行当りのエラー数（Error/KLOC）などがおなじみです．

しかし，これらはあまりに古典的すぎて，その有用性は，今日ではもう，ほとんど疑問視されています．それでは，他にいったい何があるかということになりますが，大きく分けて，次の2つのものがあると思います．

- 動的な複雑性：
  アルゴリズムの評価
- 静的な複雑性：
  プログラムの評価

第1は昔からある考え方で，「動的な複雑性」という表現を使っていますが，一般的には，アルゴリズムがよいか悪いかを評価するものです．計算時間が長いプログラム（アルゴリズム）ほど複雑だという単純な考え方です．メモリ・スペースも，ときには問題にされますが，だいたいは計算時間が尺度として用いられます．

この評価は，アルゴリズムを記述する言語のレベルに依存しないことが必要です．たとえば，チューリング・マシンを用いて，アルゴリズムの動的な複雑さを評価するといった具合いです．実際，ソーティングとかのアルゴリズムになると，たぶんこうした方法でしか評価できないでしょう．

次に，少しいいかげんでもいいから，ともかく，人間がどれくらいの労力（頭）を使ってそのプログラムを作ったかを，簡単に知ることはできないかということで出てきたのが，静的な複雑性の考え方です．

これは，ふつうの文章の評価とまったく同じような感じで，プログラムを評価してみようというアプローチだといえます．その裏にある考え方は，見た目に複雑なプログラムは複雑だ，ということです．つまり，LOC をこの観点で解釈すれば，行数が多ければ多いほど難しいプログラムだということになります．

この種の評価は，プログラムをどういう言語で書くかで大きく変わってしまいます．Prolog で書けば 3 行ですむプログラムが，Pascal でば 100 行になるといったことは，ごく自然に起こります．だから，言語が変わると簡単には適用できない．言語別に考えて行かなければなりません．

(2) 複雑さの尺度

動的な複雑さは少しむずかしい問題を含んでいるので，ここでは省略します．静的な複雑さの尺度として，これまでどんなものが考えられてきたか，有名なものをいくつか紹介します．

(a) Halstead のソフトウェア・サイエンス指標

あるプログラミング言語で用意されているオペレータ（足し算記号の ＋ その他）の種類を n とします．そして実際にプログラム中におけるオペレータの出現個数を m とします．一方，オペランド（つまり変数とか定数とか，プログラマが勝手に作り出す記号）の種類を N とし，その出現個数を M とすれば，プログラム複雑さ C は次の式で与えられます：

$$C = m \log n + M \log N$$

この式を基本として，ほかにもいくつかのバリエイションはありますが，これが Halstead のソフトウェア・サイエンス指標と呼ばれるものです．背後にある考え方は，プログラムの中で，どれくらいの語いを使っているか，語いが豊富であればあるほどむずかしいプログラムだということです．

たしかに，ある意味でそれはいえるでしょう．変数の種類が多ければ，それを頭の中に考慮に入れながら，プログラミングしなければならないのはたしかです．

図 3-1 に示す Pascal の簡単なプログラムを見てみましょう．

```
begin
  .
  .
 x := 0;
  .
  .
 for i = 1 · ·
   begin
     .
     .
     y := a[i];
     .
     .
     if · · then
       begin
         .
         .
         x := x + y;
         .
         .
       end
     else
       begin
         .
         .
       end;
   .
   .
 end;
  .
  .
end;
```

図3-1 簡単な Pascal のプログラム

この場合，オペレータとは，たとえば beginとか end とか，言語の Syntax で決められているものを指します．for, if, then, else, :=, +, [], などもそうです．出現個数とは，それらが何回プログラム中にあらわれるか，その個数を勘定したものです．表 3-2 に示すように，その合計は 18 になります．

オペレータ	出現個数
begin	4
ens	4
for	1
if	1
then	1
else	1
:=	3
=	1
+	1
[]	1
	18

表3-2 オペレータの出現個数

一方オペランドとは，先ほどいったように，プログラマが勝手に定義した変数名やコンスタントを指します．図 3-1 のプログラムでは x とか a とか y とかです．それらの出現個数は，表 3-3 のようになります．

オペランド	出現個数
x	3
a	1
y	2
i	2
0	1
1	1
	10

表3-3 オペランド出現個数

以上の価を先ほどの式にあてはめて，プログラムの複雑性 C を求めると：

$$C = 18 \log 10 + 10 \log 6$$

となります．

この Halstead の指標については，これまでさまざまな実験が行われ，いくつかの相反する結果が導き出されています．したがって，その有用性については，まだ，いろいろな議論があるところです．

(b) McCabe のサイクロマチック数

次に，もう 1 つ有名な指標は，McCabe のサイクロマチック数と呼ばれるものです．これは，グラフ理論でいうオイラーの定義を拡張して使ったものです．プログラムを無向グラフであらわしたときの，辺の数を e，ノードの数を

v とすれは，プログラムの複雑度 C は，次の式であらわされます：

$$C = e - v + 2$$

図 3-1 のプログラムを例にとって考えると，図 3-4 のようなグラフが書けます．

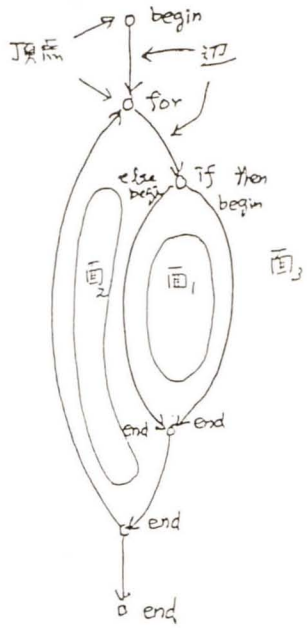


図3-4 プログラム・グラフ

ここには，辺の数が7つ，ノードが6つあります．これを先ほどの式にあてはめると，

$$C = 7 - 6 + 2$$

ですから，複雑度は3ということになります．この複雑度3の意味ですが，オイラーの定義によれば，外側に Bound されない面を含めて，図 3-4 に示したように3個の面があるということです．

構造化プログラミングの場合には，この解釈があてはまりますが，オペレーティング・システムなんかのように，あんまり構造化されておらず， GOTO 文がいっぱい入ったプログラムでは，グラフがよじれたりして平面になりませんから，視覚的な解釈はむずかしくなります．

このモデルの問題点は，プログラムを無向グラフとして考えているので，if 文も do 文も for 文も，全く同じように評価されてしまうということです．エラーを引き起こす可能性は，if よりも do とか for とか while のほうが高い．その点でちょっと正確さに欠けるのではないかというのが，多くの人々の実感でしょう．

(3) 生産性や品質の尺度

生産性というのは非常にむずかしい問題で，いまのところは，プログラムの行数を開発にかかった工数で割るぐらいしか表現手段はないんじゃないかと思いますが，本来は，ある生産工程に投入した資源と，得られるアウトプットの比が，生産性の定義ですから，次のような式であらわされなければなりません：

$$\text{生産性} = \text{成果物} / \text{投入資源}$$

しかし，ここで分子のほうが問題で，ソフトウェア開発における成果物の量をどうやって測ればよいかが，まだわかっていない．オブジェクト・コードの大きさとか，LOC とか，いろいろな指標はあるが，どれも適切でない．そこで，Halstead や McCabe など，いくつかの提案がなされてきたが，まだ，どれをとっても問題は残るというのが現状です．

そして，次に品質ですけれども，保田さんのお話にもあったように，品質というのは，機能性，信頼性，操作性，効率性，保守性，移植性，導入容易性といったいろいろな要素からなっています．

これらのうち，機能性とか，信頼性，操作性が品質の構成要因だということは，誰がみてもわかる．ユーザにとってもはっきりしています．また，保守性や移植性も，比較的わかりやすい概念です．しかし，導入容易性になると，前の4つ程クリアーではありません．

効率については，先ほど代用特性というお話がありましたが，そういう目で見ると，スループットとか応答時間，またはメモリの使用量などの指標を用いて，より具体的定義できます．というより，開発者の立場から，このような評価が可能なわけです．

(4) 信頼性の評価

ところで，今回のシンポジウムのテーマである信頼性についていえば，プログラムの複雑さは，その信頼性を測る Criteria または代用特性だと考えられます．つまり，複雑なものほど信頼性が低くなるという経験則があるからです．

具体的なメトリクスとして何があるかといえば，先ほど紹介したサイクロマチック数があります．これは IBM に

おける経験ですけれども，１モジュール当りのサイクロマチック数が３０以上になったら危ないとか，そういうことがいえる．

テストがどれくらい行なわれたかということも，信頼性を計測するメトリクスとして，非常に大切だと考えられます．もちろん，テストだけで信頼性を保証することはできませんが，信頼性の向上にはかなり寄与しています．

このテスト充分性という指標の裏にあるのは何かといえば，それは，テストが不充分だと信頼性が悪くなるという仮定です．これはきわめて自然な考え方ですね．もちろん，最初からプログラムがよければ別ですが，一般的にはそうでもないので，テストが不十分だと，どうしてもエラーが残ってしまい，信頼性が悪くなります．

それでは，テストの充分性をどうやって測るかといえば，みなさんよく御存じのテスト・カバレージ指標です．ブランチ・テストのカバレージ指標が 85% 以上にならなければ，エラーが残る危険率が高いという経験則があります．

また，ユーザの観点からよく使われているメトリクスの１つに，MTTF があります．これは，ある故障が起きてから，つまりあるバグが発見されてから，次のバグが発見されるまでの時間を測るものです．

これは，単にバグの数だけでなく，ユーザの数やプログラムの使用頻度に関係します．いくらバグがあっても，１年に１回しか使われないプログラムなら，バグは１年に１回しか見つからない．しかし，１万のユーザがいて，それぞれが毎日８時間とか１２時間使うプログラムだと，バグの数がかなり少なくても，毎日のように新しいバグが出てくる．

明らかに，ユーザの立場からすれば， MTTF が一番わかりやすい信頼性指標でしょう．しかし，MTTF は各ユーザ特有の環境に依存するので，メーカとかソフトウェア・ハウスとか，ソフトウェアを供給する立場から見ると，かならずしもよい指標とはいえません．特に，不特定多数のユーザが存在する場合には，非常に危険です．

たとえば OS のように，不特定多数のユーザ向けにたくさん出荷されるソフトウェアの場合には，将来を通して何個くらいエラーが発見されるだろうか，ということが問題になります．最初にリリースするとき，何が手がかりになるかといえば，過去の経験です．

OS には，１５年とか２０年といった長い寿命がありますので，以前のOSが，リリースされてから２０年の間に，何個くらいのエラーがみつかったかというデータが残っています．そうすると，最初のリリース時点で，大体何個ぐらいの残存バグ数であればユーザが満足できるか，ということがわかっているわけです．

ユーザが限定されいる場合や，稼働環境が限定されている場合，たとえば，軍用システムとか銀行のアプリケーションとか，電話の交換機のソフトなどは，一般に，かなり MTTF が長いようです．Bell あたりの交換機のソフトの場合，MTTF が２０年というような規定になっているという話を聞きました．

いずれにせよ，メーカやソフトウェア・ハウスの立場では，何らかの形で，自分たちが作っているソフトの信頼性を予測しなければいけない．そのさい，何が手がかりになるかといえば，開発途中で測定した MTTF とか，発見されたバグの数が，時間と共にどう変化してきたかを見るぐらいしか，今のところは手がかりがなさそうです．ひょっとしたら，将来もっとよい信頼性推定指標が見つかるかもしれませんが，現在はこうしたものに頼らざるを得ない．先ほどの保田さんのチャートにあった累積バグ数というのも，その１つですね．

開発過程の中で，テスト・フェイズを考えて見ると，テストが始まった時期はエラーがたくさん出ますから，エラーが発見される時間間隔（MTTF）は短い．そして，テストが進むにつれて，残存バグはだんだん減ってきますから，その発見間隔は長くなってきます．ほとんど理想的なテストができれば，MTTF は無限大に近くなり，まあ次の１００年くらいは新しいバグは見つからないだろうというのがソフトウェアの世界ですね．ハードウェアの場合はそれとはちがって，MTTF はテスト時間に関係なくだいたい一定ですね．ある時期がくると部品の劣化でわっと増える，これがハードウェアの信頼性の特徴です．

(5) モデルについて

メトリクスを考えるさいに重要なことがらは，ただやみくもにデータを収集し，そのデータに合うような適当なモデルを探してきて，それで問題を論じるという態度は，非常に危険だということです．H.Mills は，それを Psude Metricsと呼んでいます．何かわからないけれどこうなる，でもグラフを描くと線がこう引けるから，こんなことがいえるのではないか，といった議論は，結局何にも説明していないのと同じだ，というわけです．

やはり，そこには何かの理由づけが必要なのです．たとえば，ふつうにプログラムのテストが終わった時には，１

０００行あたり２０行のエラーが残っているといわれます．それは，確かに今までの経験ではそうかもしれないけれども，いま自分がやっているプロジェクトにとって，何の説明にもなっていない．そこで，モデルという概念が必要になってくるのです．

簡単な比喩で説明しましょう．敵陣と味方の陣地の間に地雷原があり，この地雷源をわたって敵陣を攻撃することを想定してください．

ここで，いくつかの仮定をします．攻撃の兵隊さんは１人ずつ出ていきます．そのさい，前に行った兵隊さんと次にい行く兵隊さんは関係がない．どういうふうな道をたどるかというのは，基本的に独立であると考えます．また，兵隊さんは任意の瞬間どちらを向いていてもよい，しかし，ある時間間隔をおいてスナップショットをとったとき，常に前進していなければならない．こういった仮定を立てます．

実は，これらの仮定の裏に，いろいろな意味（からくり）があるのですが，今日は時間がないので，詳細は説明しません．

さて，第１の兵隊さんが出発します．途中で，残念ながら，地雷に触れて爆死します．しかし，兵隊さんも死にますが，地雷もなくなるわけですね．２番目の人が行くときは，地雷が１個減った状態になります．当然まだ他にたくさん地雷が残っていますから，２人目の兵隊さんも多分死ぬ確率が多い．こういう攻撃を繰り返して行くとどうなるでしょうか？

まあ，日露戦争の乃木将軍みたいな心境ですけれども（笑い），時間がたつにつれて，たくさんの兵隊さんが死にます．しかし，それとともに，最後のほうになってくると，ほとんど地雷がなくなってくる．そうすると，ほとんどの兵隊さんは，地雷に当たらずに，敵陣に到達できるようになる．

このプロセスを神様が上から眺めていたらどう見えるか，それが，MTTF の考え方なのです．地雷がバグで，兵隊さんがテストケースだと思えばよい．同じ状況を，まったく違う観点から見ると，何人兵隊さんが死に，その間にどれくらい時間が経っていくか，といったモデルを作ることもできます．

本当は，ここにはもっと重要な仮定が隠されています．簡単にいえば，いまのたとえばなしの地雷源では，兵隊の通る道が決まっていません．しかし，プログラムの場合は，道が存在します．もう少し数学的にいえば，ここでは，地雷相互の間に独立性が仮定されていますが，現実はそうではありません．

OS をテストする場合を考えてみて下さい，たくさんのバグがありますが，まず IPL のテストを終わって，そのバグを取り除かない限り，他のテストには進めません．このように，バグのありかたに関係がある．あるバグが発見されない限り，次のものが見つからない．プログラムのある実行経路の上に３つのバグがあったとします．一番エントリーポイントに近いバグがとれない限り，残りのバグはとれないのですね．こういうふうに，バグの見つかる順序がある．それぞれのバグは独立ではないのです．しかし，テストが進んで，バグの数が減ってくると独立性が高くなってきます．

つまり，ここで何がいいたかったのかといえば，定量的指標の意味づけとしてのモデルの重要性，それも単に数学的にきれいなモデルを作ってやるだけでは駄目で，どうしてそのモデルでいいのかを現実ときちんと対応させなければならない，ということを申し上げたかったわけです．

以上で予定の時間もきましたので，私の話を終わります．

(7) 質疑応答

Q： 現在の信頼性推定モデルはどの程度現実的なのですか？

A： 非常にシビアな質問ですが，今のほとんどのモデルは，バグが見つかったときに，それを完全に取り除き，新しいバグを作り出すことはないと仮定しています．しかし，現実はどうかといえば，たぶん１～３割の場合に，古いバグの修正とともに新しいバグが入ってくると思うのですね．そうすると，今の信頼性推定モデルはあてはまらない．実際に３割の確率で新しいバグが入ってきたら，今のほとんど全てのモデルの推定は狂ってしまいます．

信頼性モデルによる推定がどのくらい当たるかを，実際のプロジェクトで調べたところ，誤差が，大きいもので３～４割ありました．一般的にいえば，推定値よりも現実の値のほうが大きい．ですから，まだモデルとしてはかなり貧弱だというのが，正直なところでしょう．

Q： 誤りが完全には除去できないという仮定にもとづくモデルはむずかしいのですか？

A: それをやろうとすると，ふつうは，マルコフ過程の考え方を導入する必要があります．あるバグを修正したとき，どれくらいの確率で新しいバグが入ってくるか考えて，それを無限に連鎖させるわけです．そうするとモデルがきわ

めてむずかしくなって，ほとんど一般的には計算不可能になってしまう．いままで，あまり成功した例は聞いていません．

しかし，私がいま吟味している新しいモデルでは，たぶんこれまでの仮定をはずしてもよいと思います．このモデルが成立するとすれば，もしバグが完全に取り除けない場合でも，ある変数変換を行なうだけで同じモデル（数学的にいえば同型のモデル）が得られるはずで，そうすると，たぶん推定は同じ方向でうまくいきます．

ただ，こうした形をしているデータはそう多くないので，それをうまく当てはめられるケースがどのくらいあるかは少し問題です．交換機系のソフトだと結構これでいけるのかな，と思っています．アプリケーションならいいのですが，OS みたいな基本ソフトの場合はほとんどあてはまらないでしょう．

Q： いま，交換機系といわれましたが，それはどうしてですか？

A： OS の場合は，システム・ジェネレーションという用語があるように，インストールされる環境によって違ってきます．そうすると名目的には１つのソフトなのですけれど，実際は，全然違ったソフトがたくさん動いているのと同じことになります．

一方，交換機系のソフトは，だいたい似たような使われ方をする．そうすると，ある程度まで行けば，メイン・ラインのバグが取れてなくなるので，バグ相互間の独立性という仮定が成立しやすくなる．ですから，必然的にモデル化が容易になってくるのです．

Q： 日本では，バグの累積曲線にＳ字形が多いといわれていますが，それは，テストのやり方とかに関係があるのでしょうか？

A： はい，ものすごく関係していると思います．Adhoc にテストを行なうと，絶対Ｓ字形の累積曲線になるといってもいいでしょう．

Q： その場合，原点をどこに取るかが問題になると思うのですが....？

A： ふつうは，システム・テストの開始時点を取ります．ある場合には，システム・テストの後で，リグレッション・テストのデータを分析することもある．そうすると，先ほどのように，たくさんのバグが残っているという状況ではないので，Ｓ字形の曲線にはならない．

したがって，どこからデータをとりはじめるかということは大切です．リグレッション・テストとシステム・テストとでは，同じテストでも，全然内容が違うわけですね．テストのやり方も違いますし，本質的に違う作業ですから，まとめてデータを取って，システム・テストの初めからリグレッションテストの終わりまでを，一遍に分析するのは少し乱暴だと思います．

Q： 先ほどの地雷原モデルですが，このモデルを見て不満に思うのは，バグの重みづけが考慮されていないことです．大きい地雷だと１つ爆発したら，周りの人間がたくさん死ぬし（笑い），かんしゃく玉みたいものなら，たとえ爆発しても，痛いなあと思う程度でそのまま前進できるということがあると思うのです．それに，バグというのは，ふつう，プログラムの正しさだけが問題にされるのですが，たとえばある操作を１０秒以内にやらなければいけないのに，２０秒かかってしまったとしたら，それも１つのバグと考えていらしゃるのでしょうか？

A： 後半の御質問の答えは簡単で，それはあきらかにバグです．それから，地雷の重みづけは，発見されたバグに対してそれぞれランクをつけることで処理できます．たとえば，あんまりたいしたことないバグと，重要な（むずかしい）バグとに分けて考える．

なぜそうした総別が必要かといえば，バグを処理（修正）するスピードをモデルのなかで扱うようにしたいからです．簡単なバグと難しいバグとでは，そのスピードがものすごく違うんですね．したがって，どうしてもそうした区別をせざるを得ないというのが現実です．

浦野： どうもありがとうございました．かなり時間が延びてしまったのですけれども，これでオープニング・セッションを終わりにさせていただきます（拍手）．

このレポートは，昨年秋，大阪で開かれた第６回ソフトウェア信頼性シンポジウムから，第１日午前のチュートリアル・セッションにおける３つのプレゼンテーションをまとめたものです．テープ起こしは，シンポジウム参加者の１人であった佐藤貴司さんにお願いし，最終原稿は編集部で整理しました．したがって，文責はすべて編集部にあります．引き続き行われた討論セッションの記録は，いま編集ボランティアの手で整理中です．次号をお待ちください．

第３回 実践的開発環境に関する集中討論（報告・その１）

Session 1

マン・マシン・インタフェイスの改善

1988 年 1 月 27 日　PM 2:00 ～ 5:00
於　長野国際会館

チェアマン：　坂下　　秀（アステック）
スピーカ　：　青木　　淳（富士ゼロックス情報システム：FXIS）
　　：　吉村智香子（ヒラタソフトウエアテクノロジー：HST）
　　：　新田　　稔（ソフトウエア・リサーチ・アソシエイツ：SRA）
レポータ　：　小林　貞幸（電算）

坂下：　いまや，開発環境をささえる土台は，TSS+キャラクター端末の時代から，ワークステーション＋ビットマップ・ディスプレイの時代へと移り変わりつつあります．ウィンドゥ・システムもいくつか登場し，どれがメジャーになるか興味が持たれています．

このような変化に伴い，従来のツールのユーザ・インタフェイスを改良したり，斬新なアイデアに基づいたマン・マンシン・インタフェイスを基礎にして，開発環境を再構築するなどの活動が行なわれているものと思われます．

このセッションでは，マン・マシン・インタフェイスのアイデアの提示や，ユーザ・インタフェイスの改良事例の発表などを通じて，人とマシンとの「親和性」の本質に迫れるような議論ができれば，と考えています．

みなさんのポジション・ペーパを審査した結果，プログラム委員会で３人の発表者を選んであります．まず最初に，富士ゼロックス情報システム（FXIS）の青木さん，お願いします．

Summit の発想

青木　淳

「破壊と建設を繰り返して美が生まれる」というのが，ソフトウェア作りにさいしての私の美学です．既存のものをできるかぎり壊して，その中からまた新しいもの，美しいものを作り出してゆこうということが，いつも心の底にあります．

私は言語屋です．いまは富士ゼロックス情報システムにいますが，その前は富士通ビーエスシーで，アセンブラや C の処理系を作っていました．その，後熊谷さんのところ（PFU）に派遣されて，外注として働きました．その時に，Lisp，Prolog，Smalltalk の処理系を作るという仕事をしました．

そんなわけで，マン・マシン・インタフェイスについて，言語屋としての立場から，考えをお話してみたいと思います．

(1) よいプログラミング言語とは

まず，「よいマン・マシン・インタフェイスは，よいプログラム言語とともにある」と申し上げたい．よいプログラム言語とは何かといえば，自動メモリ管理が，その第１番目の条件です．これは，農地解放のように，プログラマを平等にするための根本だと思います．この機構を持っていない言語は，近い将来に亡びるでしょう．ALOC とか FREE とかいうコマンドを，プログラマがわざわざ使わなければいけないような言語は，もうダメだと思います．

対話的で自己拡張的であるというのが，２番目の条件です．つまり，コンパイル言語は，やっぱりもうおしまいで，インタプリタのほうがいいということですが，実行速度が問題になるので，その辺を何とか解決して，コンパイラとインタプリタのあいの子みたいなものを作ればいいのではないか，と思っています．

ソースを書いて一括コンパイルしてメモリに乗せて走らせるような言語は，もはやナンセンスだという感じです．少なくとも，そういう言語の場合には，インクルメンタル・ローダを自分で作っておいて，動的リンクをしながらやっていくようにすべきでしょう．そうすれば，プログラムがまちがっていたら，その部分だけ入れ換えることができます．マン・マシン・インタフェイスを作っていくときに，これはきわめて重要なことだと思います．

３番目には，アルゴリズムをデータとして扱えるという特性が，特にマン・マシン・インタフェイスにとっては，重要になります．ユニバーサル・メソッド，Lisp の万能関数のことなんですが，それが簡単に実現できる言語構造が必要だろうと思います．プログラマブル・アイコン，可視的なイメージに対してプログラムを束縛できるような性質が，マン・マシン・インタフェイスを作っていく時に必要だと考えています．

アルゴリズム自体をデータとして扱える言語としては，Lisp，Prolog，Smalltalk があげられます．C にも関数へのポインタがありますが，環境自体が存在しない．つまり，スタックの内容を保持して動くようなことができません．アセンブラは問題外です．

それから，４番目には，対象物が自然に表現できること，すんなりモデル化できて扱えることが重要です．これは，Smalltalk が得意だといわれていることですけど，一応オブジェクト指向の考え方です．

オブジェクトがいっぱいあると，対象物間の関係を表現できることが必要になります．つまり，あるマシンとマシンの間のネットワークとかがスンナリ表現できる言語体系が必要だろうと思っています．Smalltalk で対象物を自然に表現することは，勉強すればそれなりにできるようになるんですが，そうやってできたオブジェクト間の関係を記述するのはむずかしい．

ですから，この２つができるような言語が，これから望まれるんじゃないかと思います．で，そういう言語の上に，初めて，よりよいマン・マシン・インタフェイスが作られるだろうというのが，私の持論です．

(2) Smalltalk の MMI

それで私，このところずっとSmalltalkをやっています．昔は処理系を作っていたんですけど，最近は，それを使って，いろいろなツールを作っています．

Smalltalk のマン・マシン・インタフェイスは，ユニフォームな構造に特徴があります．NTT の竹内さんあたりは，Smalltalk はインタフェイス言語だといいきっていますが，私もそれに賛成です．ものとものとが話しあうようなことを，根底からきれいに記述するのに向いていると思います．そのユニフォームな構造がどんなものかということを，少しお話します．

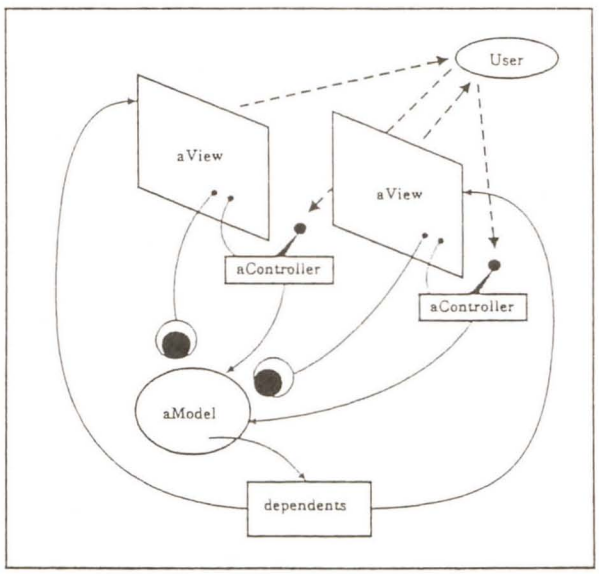


図１ MVC の概念

Smalltalk の中には，すべてのオブジェクトを見るためのツールがそろっています．たとえば，整数を見るためのウィンドゥすらあります．実際には，インスペクタと呼ばれるものです．それを専用化していくと，システム・ブラウザとかファイル・リスタとかになっていきます．よくいわれる MVC つまりモデル・ビュー・コントローラとは，あるものをある視点から見ている，そのビューにはコントローラがついていて，ユーザは，レバー操作によって，モデルを見る視点を変えられる．

ふつう，MVC については，このことばかりが強調されていて，１番大事な「依存性」の概念が抜けていると思います．これを拡張すると，エンカプセレータとかデーモンになっていくわけですが，そのことを無視した MVC の解説書が多いのは，困りものです．

どういうことかといえば，同じものを，複数の人が違う視点から見ている．そのときに，ある人がコントローラをいじれば，モデルの内容が変ります．同時に，違う視点から同じものを見ていた人に対しても，眺めが変化して欲しいわけです．そうすると，ふつうの MVC ではだめで，モ

デルに対してビュー，コントローラが依存しているような形態が必要になります．

それが，Smalltalk の中には用意されています．コントローラによって変化が起こると，自分が変化したよと伝えて，それが見ている人達にブロードキャストされていくという形です．MVC 自体はそんなに大したことはなくて，その先にある依存性とか，それをもっと発展させますとカプセル化の概念になるんですが，オブジェクト間の関係を記述するものです．オブジェクト指向に関係の概念まで取り入れたような考え方です．Smalltalk のマン・マシン・インタフェイスは，すべてこの方式で作られています．

(3) よりよい MMI のために

私は，よいマン・マシン・インタフェイスのためには，拡張性と親和性が重要だろうと思っています．

先程から述べてきたプログラマブル・アイコン（プログラマブル・イメージ）では，視覚的なものにプログラムを束縛できることが，この拡張性を生み出しています．1つの絵を描き，それに対して何かのプログラムを作ってバインドできる仕組みが必要です．その背後には，アルゴリズムをデータとして扱える言語構造が必要だと思います．

それともう1点，ダイレクト・マニュピレーション，直接操作性と呼ばれているものが重要です．対象となるものを選択してそれに対して動作を指定できるということです．私は家に Mac を持っているんですけど，フライト・シミュレータを動かすと，何か自分で飛行機を操縦しているような奮囲気になりますね．このように，対象物を直接操作している感じが，やっぱり大事だろうと思います．それは，ユーザが本当にそのものに触っているという感覚を与えてくれます．CPUのパワーが上がったら，MMIでは，まず，ここをがんばるべきだと思います．

次の図は，Smalltalk の小宇宙を示しています．オブジェクトを中心にして，これが1番抽象的なものですが，それをどんどん具体化していったクラスの階層構造を示しています．オブジェクトとコントローラの間には，サブクラスという関係があるんですけれども，この関係を線で結んで示してあります．しかし，それをただ線でつながっていると考えるだけでは無意味であり，オブジェクト間にサブクラスという関係があるということが処理できなければいけない．

ここに出ている文字列は，それ自体，プログラマブル・アイコンになっています．ただ単に文字がそこに書いてあるだけではなくて，これをひっぱたたくと，クラス・ブラウザが開きます．つまり，このアイコンに，「ブラウザを開く」というプログラムがバインドしてある．ここをダブル・クリックすると，コントローラを持ったクラス・ブラウザが開いてくるようになっているのです．

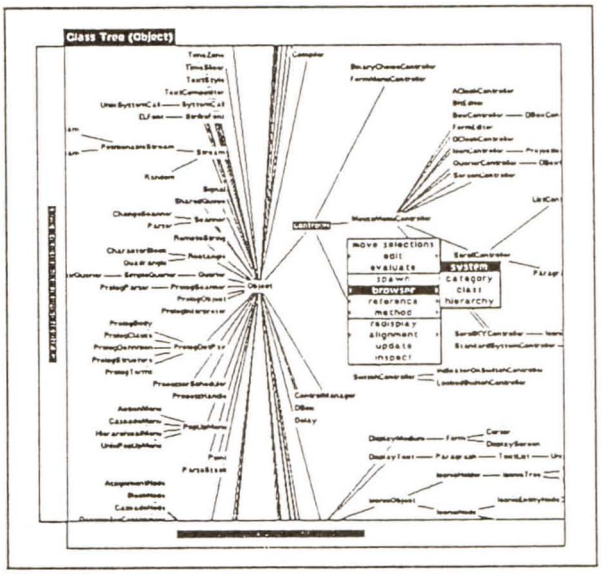


図 2 Smalltalk の宇宙

この図の中には，たくさんのものが描いてありますが，そのうちのどれかを選択し，その後で，それに対して動作を指定する．実際には，ポップアップ・メニューを使うのがふつうです．メニューを開けて，たとえばブラウザーを開くという順序です．

Smalltalk における MMI は，ほとんどこの形態で統一されています．Mac と同様に，ツールによって使い方が異なるということがほとんどなく，この原則を犯すこともタブーとされています．ある意味では，非常にプログラマが苦労する．つまり，ときにはそのタブーを破ることが，本当は必要なんだけど，やっぱり皆に使ってもらうためには，そのフレームワークを守らなければいけない，というジレンマに陥ることもあるのです．

(4) 視覚的 Unix の試み

それでは，Smalltalk を使って，いったいどんなものを作ったか，1つの事例をお話します．それは，Unix をできるだけ視覚的に見せてあげようという試みです．Summit と呼んでいます．絶頂とか頂上とかいう意味ですが，実は Smalltalk で Unix の MMI を作ったというだけのこ

とです．

このアイディアの鍵になっているのは，Smalltalk の中にある Unix のシステムコール・クラスです．Smalltalk の中から Unix のシステムコールをオブジェクト指向風に使えるようにしたものです．システムコールの中では，最近話題の IPC を使ってあって，ソケット，メッセージ・キュー，共有メモリなど，いろいろ IPC のやり方はあると思うんですけど，そうしたものが全部使えます．いままでずっと C でやってきたけれど，なにも Unix のシステムプログラミング言語は C ばかりではない，Smalltalk で書けば記述力が 100 倍になる（笑い）．

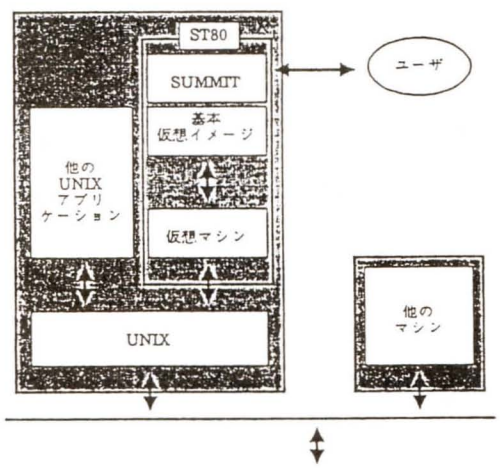


図 3 Summit

先程のブラウザがいっぱい出てきた仕掛けも，本質的には2～3日で作れます．これから説明するシステムも，実際にアイディアを考えついてから動き出すまで，数日しかかかっていない．その後の時間は，チューンナップやこまかいバグつぶしにかかっているだけで，とりあえず動くものができるのは，非常に早い．これが Smalltalk の特徴です．

さて，Summit の内部的な構造ですが，Xerox の 1161 というマシンに Unix (System V) が乗っていて，その上の1プロセスとして Smalltalk が走っています．その中には，仮想マシン，仮想イメージ，その上にこの視覚的 Unix ツールを乗せています．ここことやりとりしているのが Unix System Call クラスと呼ばれているものです．仮想イメージの中にあるんですけれども，仮想マシンの中はほとんど素通して，そのまま Unix に伝えられます．で，その機能を持って上がってきます．

ですからこの Unix を通して IPC で他の Unix アプリケーションの結果を Smalltalk の中に取り込むこともできますし，他の Unix アプリケーションから Smalltalk の中を呼び出すこともできます．ソケットとか何かを使えば，それはもうネットワークに出て行きます．残念ながら 1161 にはソケットがついていないので，最近では Sun を使っているんですけれども（笑い），その場合には，これ自体が何か Smalltalk の仮想マシンに見え，それが他のマシンと交信をし合うという形になります．

(5) ファイル・ホルダとエディタ

視覚的 Unix のツール群にどんなものがあるか，全部は無理ですので，目新しいところだけをご紹介します．

Unix の特徴は，プロセスとファイルにあると思うんですけれど，1161 というマシンは，その辺があまり強くない．プロセスを視覚的にしようと思うとパワーが足りない．今後は，そっちに手をつけて行こうと思っていますが，とりあえず安易にできるものはファイルなので，それをできるだけ視覚的に扱えるようにしようと考えてみました．

まず，ファイル・ホルダとエディタを作ってみました．エディタは，ほとんど Smalltalk のパラグラフ・エディタです． Unix の世界では，ふつう，フォントの幅が一定ですから，そういう風にしただけのものです．

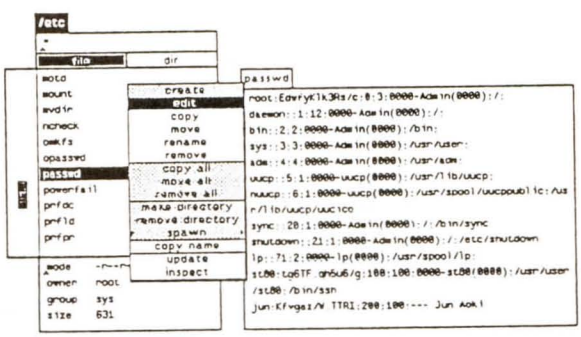


図 4 ファイル・ホルダ

一方，こちらはホルダですが，たとえば，パスワード・ファイルなんかが入っている /etc のディレクトリはこのパターンで表されます．ファイルの内容がここに出ていて，その中から適当に選べば，下に 'ls -l' コマンドで得られる詳細情報が表示されます．ものを選んで，ポップアップ・メニューを開け，編集という操作を選ぶと，望みのファイルのウィンドゥが開いてくるという感覚です．

日常，ファイルを探して内容を見る，エディタにかけるという一連動作を，コマンドでやっていますと，いくらスクロールバーをつけても，出力が上に逃げて行ってしまいます．そこで，それらをウィンドゥとしてホールドすることが大事だと思います．これからの ls 風コマンドは，こういう形であるべきじゃないか，と思って作ってみたものです．

(6) ファイル・ツリー

それから，ファイル・ツリーですけれども，とにかく人間は５階層以上，相当な天才でも７階層以上のイメージは頭の中にないという話です．それくらいが，記憶の限界らしいですね．そこで，ディレクトリの階層構造を視覚的に表現してあげようと考えました．ここに，枠で囲われているのが，実際に私が作業しているディレクトリ構造です．

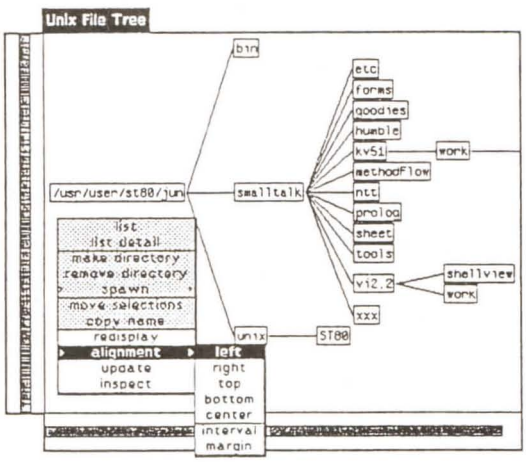


図 5　ファイル・ツリー

この上で何かを選んで，マウスをクリックすると，あたかもチェンジ・ディレクトリしたような効果が得られます．もう'cd'なんてコマンドはいらない．この中のアイコンに，そのためのプログラムがバインドされています，それをパチッとやると動作が起こる．つまり，「おれはいまここに行きたい！」っていう感じですね．

今度は，こいつをダブル・クリックすると，また違うプログラムが動き出して，自分のホルダーを開く，先程の /etc というファイル，そこをパチパチッとダブル・クリックするとホルダーが開いて，中にあるファイル群が見える．そのプログラムがこのアイコンに束縛されているのです．このメニューは，Smalltalk と Mac を足したような感じです．選択ができないところはシェードがかかっていて選べません．一応，ユーザをガイドしているわけです．

(7) 直接操作

今度はダイレクト・マニュピレーションの話です．ディレクトリ階層全体の移動をどうやって実現するかを例として，お話します．

System V では 'tar' コマンドを使うんですが（笑い），Summit では，対象物となるものをマウスでクリックするだけです．この図では Smalltalk という文字列のちょっと下の部分です．皆が集まったところをクリックする，これで，そこにつながっているサブディレクトリ全部が選択されます．次にメニューを開き，選択したものを移動したいという意志を指示します．

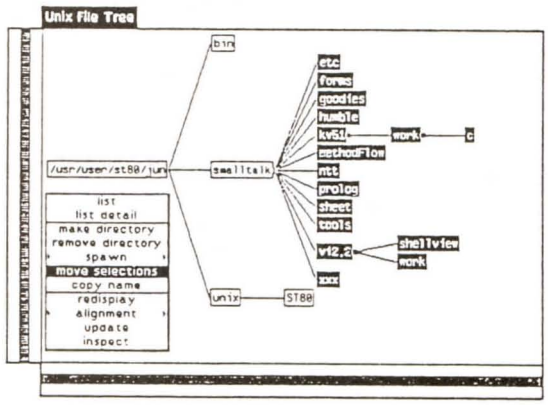


図 6　直接操作（選択）

で，いったい何が起こるかというと，この木全体がアイコンから離れ出します．それを望みの場所へ移動するだけでいいのです．

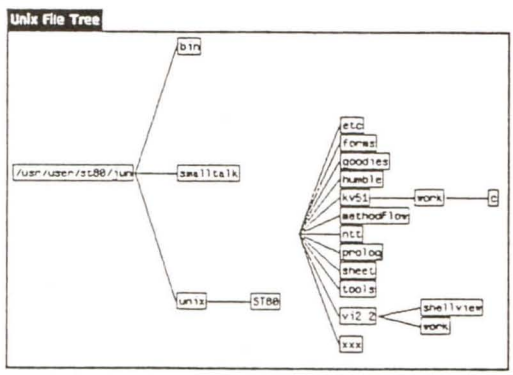


図 7　直接操作（移動）

(8) ファイル・リスタとコマンド・インタプリタ

ファイル・リスタと呼ばれるものがあります．これは，前述のホルダとツリーをくっつけたようなものです．ここをパシッとやると，その中のファイル群がこちらに現われる．それを選んでポップアップ・メニューを開け，編集その他の操作ができますす．ツリーとホルダーの統合，一連のコマンドのスムーズな流れを実現したものです．

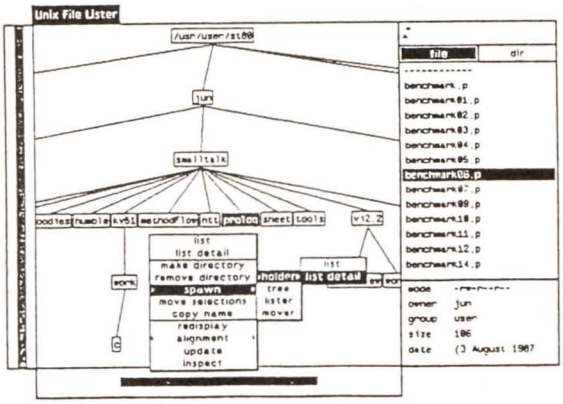


図 8 ファイル・リスタ

ツールの紹介の最後になりますけれども，とりあえずのコマンド・インタプリタがこれです．Smalltalk の世界には，コマンドを入れて復改キーを押すという概念が最初からないので，こうしたシェルを作ること自体がナンセンスではあるのですが（笑い），1161 自体がそんなに強力ではなく，プロセスを視覚化できないので，仕方なくとりあえずシェルを作ったというわけです（笑い）．

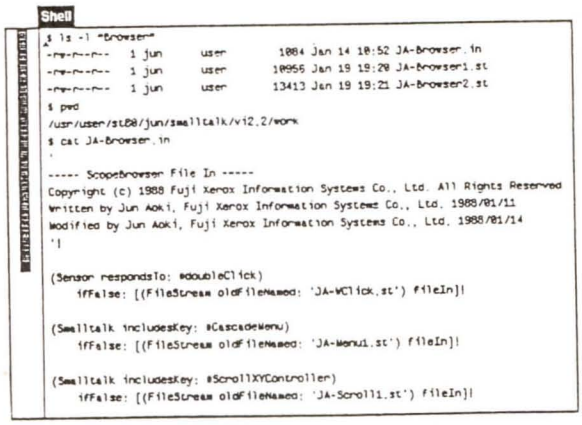


図 9 コマンド・インタプリタ

これはもう，通常のウィンドゥ・システムの MMI と変わりません．たとえば，X とか SunView とかは，Unix にウィンドゥ・システムを乗せたといっても，本質的にはこの機能を提供しているだけです．インタフェイスの立場からすれば，Unix のコマンド群の寿命は，もう尽きていると思います．しかし，これからもやっぱり過去を引きずって行かなければならないのなら，それをウィンドゥ・システムに合うような形に変えなきゃいけないと思ったわけです（笑い）．

以上で，私の発表を終りますが，ハイパーカードをはじめとして，非常に MMI のすぐれたソフトが最近出て来ていますけれども，その基本的な技術は，プログラマブル・アイコンと，アルゴリズムをデータとして使える構造の構築にかかっていると思います．そうすれば，ダイレクト・マニュピレーションみたいなものがスンナリ記述できますので，そういう方向で，これから努力して行くべきだと考えています．

坂下： どうもありがとうございました（拍手）．もし，いまどうしても聞きたいという質問があればうかがいますが，ありませんか．

それでは，引き続いて，HST の吉村さんに発表をお願いします．

ユーザ・インタフェイスに関する一考察

吉村智香子

SEA のワークショップに参加するのは今回が初めてです，どうぞよろしくお願いします．

ポジション・ペーパにも書きましたが，わが社で，ロボット・プログラミング開発支援システムの再構築という仕事をやりました．その中で，ユーザ・インタフェイスに関していろいろ考えたことを，これからお話したいと思います．

まず，仕事の内容について申し上げると，HST という会社は，自動化省力機メーカーの平田機工から，FA におけるソフト部門の強化を目的として独立した会社です．まだ独立して約1年ですが，これまでに FA 関連のソフトウェアの開発，主に製造管理システムなどをいくつか手がけてきています．

(1) ロボット・プログラミング開発支援システム

私自身，昨年の夏，ロボット・プログラミング開発支援システムの再構築というプロジェクトに携わりました．

これは，従来平田機工で IBM-PC 上に構築されてきたシステムですが，それを今回新たに Mac に移植するという仕事でした．名前が示す通り，ロボット・プログラムの開発におけるコーディングからテストまでを支援するシステムです．対象となるユーザは，主に工場の作業員ですね．

これまでコンピュータに一度も触れたことがない，コンピュータって恐いと思っているような人たちです．また，平田機工で生産ラインを作っている技術者たち，いままで IBM-PC 版のシステムを使って仕事をしてきた人たち．そういうユーザも対象としています．

開発環境としては，ユーザ・インタフェイスおよび通信部分を除いて，すべて Sony NEWS 上で開発しました．そこには，御存じのように，ビットマップ・ディスプレイ，Xウィンドゥ，マウスといったユーザ・インタフェイスが提供されています．この仕事は，HST で Mac に触る仕事として初めてのもので，かつユーザ・インタフェイスを作成するという意味でも，初めてのプロジェクトでした．

そういう環境の中でシステムを構築していったのですが，それを通して，たとえば，まず自分たちのための開発環境を作るさいに，いろいろユーザ・インタフェースについて考えました．また，構築したシステムを従来のものと比べてみて考えたこと，できあがったシステムに対するユーザ評価などをまじえて，お話したいと思います．

本当に，まだ考えただけで，まったく実践は伴っていないのですが，これから，それを具体化する方法についてのヒントを，討論のなかで得られたら幸いです．

では，まず簡単に，従来のものと Mac 版の方のシステムを比較説明したいと思います．

(2) 従来システムのユーザ・インタフェイス

これが，従来の IBM-PC 版のユーザ・インタフェイスですが，すべてのメニューが階層構造になっています．メニューの入力は，ファンクション・キーによるキーボードからの入力のみです．たとえば，システムを立ち上げると，ここに示すような初期メニュー画面が現われます．

Program Name	H-3525
Function	HARL-II Compiler for IBM-PC
Edited on	1987 - 1 - 28　　Ver. 3.01
(R) (C) All rights reserved by Hirata Industrial Machineries Co., Ltd.	

```
F1  : EDIT mode ----- Create HARL-II program using IBM-PC BASIC
F2  : COMPILE mode -- Compile HARL-II program
F3  : LINK mode ----- Link HARL-II program
F4  : DEBUG mode ---- Debugging, patching, and verifying HARL-II program
F5  : KEY-IN mode --- Edit position data, SG data or Definition Table
F6  : SET-UP mode --- Tune your IBM PC by editing set-up file
F7  : FILES --------- Show HARL-II program file names in disk B:
F8  : LLIST --------- Print out HARL-II program in disk B:
F9  : HELP  --------- Explanations on HARL-II compiler
F10 : EXIT  --------- Return to IBM PC DOS

Select function key < F1 - F10 > ?

1EDIT  2COMPIL 3LINK  4DEBUG  5KEY-IN 6SET-UP 7FILES  8LLIST  9HELP  10EXIT
```

図 1　従来システムのユーザ・インタフェイス

ロボット・プログラムを編集する場合は，F1 キーを押すと，編集モードのメニュー画面が現われて，そこで仕事をします．終了したら，そこで EXIT(F10)キーを押して，初期メニュー画面に戻ります．

次は当然コンパイルでしょうから，コンパイル・モードをここで選びます．するとまた，同じようなコンパイル・メニュー画面が現われます．コンパイルが終了したらメイン・メニューに戻って，次は，リンクのために，また同じような操作を繰り返すわけです．

リンクをしようと思ってリンク・モードに入った時に，もう1つプログラムの編集を忘れていたとかいう場合，また元に戻って，そこへ行ってというように，非常に面倒で複雑な操作を繰り返さなければなりません．ユーザにとっては，やりたいことがすぐやれるというのが一番いいので

はないかと思いますが，これでは，やりたいことに行きつくまでの道が，非常に長くて苦痛です．

(3) 新システムのユーザ・インタフェイス

一方，Mac 版の方ですが，これはビットマップ・ディスプレイとマウスを駆使しています．メニュー・システムは，Mac のメニュー・マネージャーを利用して，プルダウン式になっています．

スクリーン上では，このように，いくつかのウィンドゥを重ね合せることができ，別のウィンドゥを通して違うメニューをいつでも起動することができます．自分がいまどういうことをやっているのかを，いつでも把握することができます．

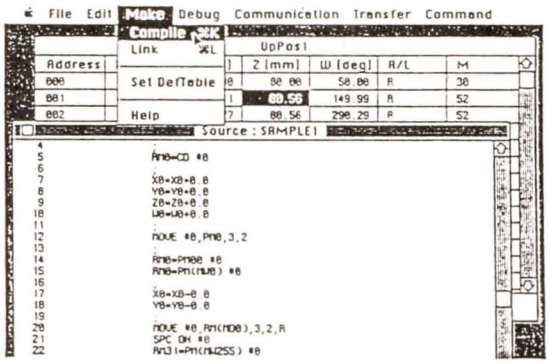


図 2 新システムのユーザ・インタフェイス

たとえば，あるウィンドゥでソース・プログラムを編集していて，別のウィンドゥで，そのソースプログラムをコンパイル・リンク・ダウンロードして，実行させながら，別のウィンドゥでは，ロボットの位置決めデータの編集をやってダウンロードさせるといった具合いです．

つまり，いままでのロボットの動きが，新らしくダウンロードされたデータにしたがって変わってきます．マルチウィンドゥですので，そういうことが，同時に行えるわけです．

このように，関連する事柄をいつも一緒に見ておきたいとか，気分転換に他の作業をしたいとか，やりたいことをすぐやりたいとかいうのが，ふつうの人間の気持ちだと思います．マルチウィンドゥのモードレス・メニューは，そういう意味で，私たちの気分を満足させてくれるもののように思います．

(4) ユーザは十人十色

以上のように，ロボット開発支援システムを再構築したわけですが，その中でいろいろユーザ・インタフェイスについて考えたこと（先程もいいましたが本当に考えただけなんですが）を以下に述べたいと思います．まあ，雑感を述べるというような感じですね．

まず，第１に，ユーザはさまざまだということです．よくいわれるんですが，初心者だったり，経験者だったり，頭のやわらかい場合もあれば，逆にモノスゴク頭が堅い人であったり（笑い），本当にさまざまなんですね．

たとえば，今回のロボット開発支援システムですが，わが社の新入社員は「メニュー・ガイド方式じゃないと非常にやりにくい」とかいうんですね．ところが，以前 IBM-PC を使っていた人は，Mac を見て「わあ，こいつは可愛い」ですとか「こいつはすごい，面白い」とか「工場で作業するのには，こういう可愛いのが必要ですね」とか，結構そういうことをいったりする．

でも，長い間プログラムを組んできたような人になりますと「やっぱりキーボードから自分で入力をしてリターン・キーをポンと押したら，そのコマンドが起動される，そういう方がいいね」などと，本当にユーザーが思うことというのはさまざまなんですね．

そういうことを考えると，やはり，ユーザの好みに応じて変更できるようなインタフェイスを提供していく必要があるんじゃないかと思います．それは，非常に大変なことかも知れませんが....

また，使いながら自然とユーザ自身が成長して行けるようなインタフェイスですとか，逆にユーザの成長に合せてインタフェイスそのものも成長していくようになっていれば，なお素晴らしいんじゃないかなと思います．

ユーザにとって，習得しやすく，信頼性が高く，発展性があって，しかも自分の好みに合うというインタフェイスを，今後提供して行かなければならないと思います．

(5) 利用目的／状況

また，ユーザがそのシステムをいったいどう使うのかを，しっかり把握しておかなければいけないと思います．

今回のシステムでは，我が社の方でハードも選択したのですが，たとえば，IBM7532 という非常に大きな工業用のマシンがありまして，過去に，そういうものにこのシステムを乗せたわけですね，そうすると，実際の生産ラインというのは，長いものになると２～３００ｍあって，その

中にロボットが2～30台くっついている．その個々のロボット向けにプログラミングをしていくんですが，ある場所で作業していて，今度は向こうへ行かなきゃいけないという時には，何と3人掛りで IBN7532 を運んで行ってました（笑い）．

何でそのとき IBM のマシンを選んだのかというと，IBM は大会社だからだというそれだけの理由だったらしいんです（笑い）．

それが Mac になれば，首にマウスをちょっと引っかけて両手でこう持って行けるわけですね．そういうことを考えてあげるだけで，ユーザさんというのは結構喜んだりしますので，そこまで考えなければいけないと，最近大声でいってるんですけど．

(6) ユーザ心理

それから，当然のことですが，ユーザの心理を忘れてはいけません．

たとえば，マルチウィンドゥを例にとれば，これはたしかにいいんですけど，欠点といったら悪いんですが，ちょっと嫌なところがありますね．それは，仕事をしているうちに，ウィンドゥが山のように重なってくるんですね．

その状況は，机の上にいっぱい書類を重ねているのと何ら変らない．あるウィンドゥでテスト仕様書を書いたはずだと思いながら，それを探すのにマウスを何回もカチカチカチやりながら，自分でイライラすることがある．

私の机の上は，月曜日の朝はキレイに片づいていて，土曜日の午後になると山のようにちらかってしまう．それを整理してまた月曜の朝キレイになってというサイクルを繰り返しているんですが，何かその状況と全然変わらないような気がする．

でもマルチウィンドゥの本来の機能は，それなんですね．机の上の書類をそのままディスプレイにポンと入れるような TV コマーシャルがありますが，まったくそれでしかない．

私たちは，もっとそれを整理したのが欲しいといつも思っていて，たとえばファイル・キャビネットをつけたりするわけですから，マルチウィンドゥに対しても，それを管理するもう1つ上のレベルの何かを考えて行くべきでしょう．

書類の山の中に手を突っこんで，いろいろ探しまわるのと，マルチウィンドゥのなかをマウスで探しまわるのも，何となく同じことですから，もうちょっと何とかしたいものです．

(7) ユーザ・インタフェイスと手順・状況

最後に，ユーザ・インタフェイスというのは手順・状況そのものだろうと思います．人間は，内容は忘れても状況とか手順とかは覚えていることが結構多い．たとえば，あの書類はあそこに斜めにして置いたとか，そういう状況は，覚えていたりする．ユーザ・インタフェイスは，この手順・状況そのもののように思います．

つまりユーザ・インタフェイスというのは，覚え易いように考慮されていなければならないと思います．そのために，はユーザ・インタフェイスには統一した全体的な思想というのが，1つ必ず貫かれていかなければならないと思います．たとえば，首尾一貫した操作ができるとか，視覚的な連続性が保たれるとかですね．

それと手順・状況によって検索ができるようなユーザ・インタフェースが望ましい．それには，ユーザとの対話が，常に保存されていないといけません．そうした過去の記録を学習して，ユーザ好みのインタフェイスに変わっていくようなシステムが夢です．

(8) 終わりに

以上，いろいろ雑感を述べてきましたが，いいたいことは，どうもふだん私たちは，ユーザ・インタフェース開発する場合に，「いまはこれしかない」とかいうふうにあきらめているような気がして，本当にユーザのことを考えているんだろうか，という反省がいつも心のどこかにあります．

最後に，わが社としては，先程も述べたように，この仕事がユーザ・インタフェイスを作るという初めての仕事でした．その後，いまもう1つ別のシステムを作っているんですが，まだその2つしか経験がありません．

どうも会社では，ユーザ・インタフェイスがメジャーじゃなくて他の部分で一生懸命がんばっていて，「ユーザ・インタフェイスは適当に作ればいいや」なんていうような意識で皆が仕事している．というより，むしろそこまで手がまわらないというような感じを受けています．

でもよく考えれば，どんなに機能が充実していても，ユーザ・インタフェイスの使い勝手が悪ければ，使ってもらえない．今後わが社でも，ユーザ・インタフェース専門のグループを作って，エキスパートを育てて行かなければいけないと思います．もちろん，センスのない人や，根気の

ない人は，いくら教えてもダメでしょうが（笑い）．そういうエキスパートの養成を是非やりたいなという考えています．

また，ユーザ・インタフェイスのライブラリ化も，是非やって行く必要がある．ユーザ・インタフェイスの設計からテストまでに要する時間は，かなり莫大なものです．

いまのところ，うちは各システムごとに1品生産をしているために，時間がかかっていますが，ライブラリ化によって，時間が大幅に短縮され，その分より高品質のユーザ・インタフェイスを考える時間がとれることになります．またライブラリ化されていますと品質面のバラツキも減ってくるでしょう．

(9) 質疑応答

坂下： どうもありがとうございました．何かここで質問がありましたら，どうぞ．

Q： ソース・プログラム編集画面の OHP で，横に行番号がついていたのですが，なぜあんなものがついているのでしょうか？また，ロボットがコントロールされるようすが，画面上でグラフィカルに見れたりすると面白いと思うのですが，そのあたりについてはいかがですか？

吉村： まずこの行番号のことですが，Mac のテキスト・エディタにどうしても行番号をつけて欲しいという要求がありましたので（笑い），ニューラインを判別して，そこに入れこんで行くようにしました．たとえば，第 16 行の次に何かを挿入したいときは，ここでリターンキーを打つと，次に 17 という行番号があらわれて，以下の行全部がインクリメントされるように，いまは作ってあります．これはずいぶん苦労しました（笑い）．

それから，ロボットを画面で動かすという話ですが，いろいろ考えはしましたが時間が足りなくてまだ具体化はしていません．この種のプログラムは，コンパイルした後，特殊なコントローラにダウンロードして実行されます．

今回の開発に際しては，News 上にコントローラのシュミレーション・プログラムを作りました．それは，いまの段階では，ただ何かを受け取って何かを返すということしかやっていません．今後は，X ウィンドゥを使って，いかにもロボットがアームを動かすようなシミュレーションをやって行こうという計画を立てています．

Q： プログラム言語は特殊なものなのですか？

吉村： 名前は，HARL2 コンパイラです．すなわちヒラタ・アセンブリ・ロボット・ランゲージという言語がありまして，それで実際にロボットを動かすプログラムを書きます．この言語は，Basic によく似ていて，IF 文とかが使えます（笑い）．

坂下： なかなか面白かったんですが，さっきの行番号なんか，いろいろなユーザがいるというよい例だと思いますね．

吉村： 平田機工が，元来 Basic しか使わない会社だったんですね．

坂下： すでに汚染されていたわけですね（笑い）．

吉村： もう「言語は Basic」という固定観念がありまして，絶対に行番号をつけてほしいといわれ（笑い），仕方なくつけているわけです．

坂下： なるほど．なかなか現実の例としては面白かったと思います．どうもありがとうございました（拍手）．

それでは最後に，SRA の新田さんにお願いします．

ERAS について

新田稔

新田： 用意してある OHP の数が少ないので，なるべく節約して使って行きます（笑い）．なぜかというと，今回発表対象のツールが，実はまだ未完成で，一昨夜も，Mac を今回のワークショップのデモ用に送り出す直前まで，コンパイルしていました（笑い）．そういうわけで，ウソの絵を見せるよりは明日デモを見ていただいた方がいいと思うのです．

(1) Mac について

最初の OHP をお見せする前に，Mac について少しお話しようと思います．たとえば，一般には，Smalltalk がよくユーザ・インタフェイスのよい例に上げられるんですが，Mac を持って感じたのは，Smalltalk のユーザ・インタフェイスはとてもヒドかったなということです（笑い）．

Unix を初めて触ったとき，昔の（？）大型機の OS ってヒドかったなあと思うのと同じように，Mac を使ってみると Smalltalk はヒドイと感じます．どこがヒドイんだと反論される方が多分いらっしゃると思うので，例を上げておきましょう．

ファイル・アウトという処理があります．Smalltalk のイメージの中に持っているプログラムなりデータをディスクに書き込む作業です．それをやろうとして，メニューからファイル・アウトを選びます．すると，フロッピー・ディスクを入れなさいといわれる．そこで，フロッピー・ディスクを入れる．すると，このフロッピーはフォーマットしてない，ダメだといって，やらしてくれない．

フォーマッティングは Smalltalk の中からはできない（笑い）．しかたなく，一旦 Smalltalk を抜けて，上の OS に戻る．ところが，ディスクを何枚フォーマットしていいのか，その時点では私にはわからない（笑い），適当に４枚位でいいだろうと思ってフォーマットして，再びファイルアウトを始める．すると，４枚では入り切れずに，５枚目を入れなさいといってくる（大笑い）．

つまり，概念的に閉じた世界でユーザ・インタフェイスのよしあしを論ずるのではなくて，これからは周辺機器その他，人間が手に触れるものはすべてシステムの要素として，インタフェイスのことを考えて行かなきゃいけないんじゃないかと，私は思うのです．

Mac のことをもう少し話しますと，人生というのは，いろんな人との出合いとか１冊の本を読むとかで変わると思うんですけれども，私の場合は Macintosh を買ったことで，すべてが変わってしまった（笑い）．

どういうふうに変わったかというと，あれを買ったことで私のプログラマとしての寿命がかなり縮まってしまった．どうしてといえば，Mac が世の中に存在しなければ，ああいうものを開発するために，あと何年か仕事ができたわけですけれども，その分が減ってしまった．そろそろ私のプログラマ寿命も尽きるんじゃないかなと思うので，ここらで最終兵器を作っておこうと思っています，

その最終兵器こそが，いまからお話しする ERAS というツールです．

(2) 背景

この ERAS の歴史をちょっと述べると，３～４年前にハワイ大学で仕事をしていた時に ERA（時代）という名前のエディタを作りました．

どういうエディタかというと，いろんなものを絵（アイコン）であらわして，エンティティ（実体）と呼びます．いろんなエンティティとその間に成り立っているリレーション（関係），それとそれらの性質を数値とか言葉とかで表現したアトリビュート（属性）の３つを使って，世の中のものを何でもあらわしてしまおうというものです．

この，エンティティ・リレーション・アトリビュートの頭文字をとって，ERA と名付けました．このエディタのもう１つの特徴は，何だったかというと，絵の描き方に関する規則をユーザが指定できるという点でした．

どうして，そんなことを考えたかというと，ハワイ大学に宮本勲先生という人がいて，学生に対しても無茶苦茶な要求を出す人ですが（笑い），CMU に ALOE という構文エディタ・ジェネレータがある．そのグラフィック版を作ってほしいという，ほとんど不可能ともいえることを，私に要求して来たのです（笑い）．

ところが，はじめは不可能のように思ったんですけれども，Mac を使っていろいろ試行錯誤している内に，これはできると確信できるようになった．そこで，作ってみたというわけです．

もう一度 ERA の特徴をいえば，エンティティとリレーションとアトリビュートでものを記述しようということ，ただのお絵描きツールではなく，絵を描く上でのルールに

照らして，正しく描いてあるかどうかのチェックもしてくれる．その判断ルールは，ユーザが自分で定義できるというものでした．

これが，だんだん発展してきて，先程熊谷さんがちょっとおっしゃったプロタイパーズ・ワークベンチというプロジェクトの中で，ユーザの要求を絵でモデル化し，それをプロトタイピングと呼ぼうと考えました．

しかし，どういう絵を描いたらいいか，最初のうちはわからなかった．その時に，この ERA という汎用のモデリング・エディタを使ったわけです．その時点では，ERA は Mac の上でしか動いていなかったんですが，宮本先生の真似をして，１人の学生にソースコードを見せて「これを Smalltalk に書き直せ」と要求したら，すぐやってくれました（笑い）．

(3) ERAS の狙い

以上が ERA の歴史ですが，どうしてそれに S がついたかというと，さっきいったように，もうそろそろ最後のツールをと思って，ERA のスーパー・バージョンという意味で，お手持の資料には，SUPERA と書いてありますけれども，あま語呂がよくないで ERAS にしました（笑い）．

この S の意味は，明日のデモをご覧になればわかるはずですが，本当は Scene（場面）の S です．いろんなものを ERAで描くと，ごちゃごちゃしてくるんですね．そこで，「場面」という概念を新たに導入して，このシーンではこれとこれだけが見えるようにしようというふうな意味合いです．いずれは，もっと発展して S の後に A とか O がつき，「偉そう」とか「偉さ」に結びつくはずです（笑い）．

何をしようとしているかといえば，いまはただ静的な絵を描いているだけですが，いずれは，描いた絵をそのまま動かしてしまおうと考えています．そのアルゴリズムは，大体もうわかかっています．

そうすると，ERAS は，それ自体がエディタであり，プログラミング環境であり，実行機構にもなる．それを作ってしまえば，あとはもう何もいらないというようにしよう，と考えています．

(4) ユーザ・インタフェイス設計の基本方針

ERAS にもユーザ・インタフェイスがあります．どういう考えを基本にして，それを作ったかというと，次の４つです：

- メニュー＋ウィンドゥ，キーボード＋マウス
- 有効なコマンドだけ選べる＋いつでも全部のコマンドが見える
- 極力モードレス．モードへの出入りを意識させない
- コマンドの数を少なく，操作の種類＋操作対象の組み合わせ

これらは，実は，Mac のアプリケーションの作り方に，こういうことを守りなさいと，きつく書かれているものです．

まず，ユーザがわかるものはメニューとウィンドゥ，それから手で触るものはキーボードとマウス，もちろんフロッピーも触ります．Mac の場合は，フォーマットが必要な場合はフォーマットをしてくれます（笑い）．プロセスの途中でフォーマットをしてくれて，続いて仕事をやってくれるということです．

メニューについてですが，ユーザがこのツールを使っていて，ある所に来ています，そこではすべてのコマンドが使えるわけではない．使えるコマンドだけが選べる．先程，青木さんの Smalltalk のメニューでシェードがかかっていて選べないというのがありましたが，Mac でも同じです．しかし，いつでもすべてのコマンドが見えている．つまり，これだけのコマンドが用意されているけれど，いまは使えませんよ，ということがユーザにわかるようになっています．

それから，いわゆるモードをなくします．たとえば，いまから名前とかテキストを書換えるからテキスト・編集のモードに入る，いまからエンティティを作るからエンティティを作るモードに入る，終ったらモードから出てくる，といった切り換えをなくしました．ユーザから見えないだけで，プログラムの中ではちゃんと区別があり，あることをしようとすると自然にそのモードに入るようにしてあります．

コマンドの数は，できるだけ少なくしようと考えています．たとえば，ファイルを消去するのとディレクトリを消去するのと，２種類のコマンドは必要ない．ファイルに消えろといえばファイルが消えるし，ディレクトリに消えろといえばディレクトリが消えればいい．そういうふうにしてコマンドの数を減らす．その代り操作の種類と対象物で，何をするかということを判断する．

以上は，基本と思っていることで，既にやっていることです．これからいっぱいしたいことがあるんですけど，その中の１つでマウスの利用ということを，ちょっと説明し

たいと思います．

(5) マウスについて

マウスには，ご存じの通りボタンがあって，ボタンを押す，はなす，それからマウスを動かすという３つの操作（イベント）を起こすことができます．それを，人間の目で分類すると，クリック，ダブルクリック，ドラッグという操作になる．つまり，コンピュータのイベントと，人間のものの見方は異なる．

すでに，Mac や Smalltalk がやっていることの他に，拡張機能として，速く押すとか，ゆっくり押すとかの区別を取入れてみました．人間はそういうことができる．実際身の回りにある機械を操作する時には，そうしている．たとえば，壊れそうなものは軽く持つとか，重いものはしっかり持つとかしています．それが，マウスやキーボードになると，持つか持たないか，押すか押さないかという単純なコードに変換されてしまう．

これはどうもまずいのではないか．われわれの仕事の手助けをするはずのコンピュータが，逆に仕事を制限している．そこで，速く押すとかゆっくり押すとかいうことを考えました．本当はマウス自体，というかハードウェアがもっと進歩して，そのスピードを検出してくれればいいんですけれども，いまはまだ，そういうことをしてくれませんので，自分の中で押された時間と離された時間を見て，区別する．

実際にどう使うかといえば，たとえば，あるアイコンを速くカチッと押してパッと持ってくると，ここに移動する．今度は，ゆっくり押す．するとカーソルの形が変わる．で，ピッと持ってくるとここにリレーションが張れる．つまり，持って来るのとつなぐのとは，別の動作だという使い分けができるわけです．

将来は，さらにマウスの動きによって動作の種類まで表現してしまいたい．いまは，対象物を選んでメニューから動作を選んでいますが，われわれはふつうの生活の中では，そんなことは特に意識していません．

たとえば，字を消したい時は消したい字を選んでから消しゴムを取るかというと，そうじゃない（笑い）．消しゴムを持って来て適当にガチャガチャやる．同じように人間の動作そのものを，やりたいことに変換してしまいたい．つまり，ものを包むように動かすと，選んでそれから包むというメニューを選ぶ．消したい時には，ガシャガシャガシャとマウスを動かすと，その文字列を選んで消去するメニューを選ぶのと同じような操作ができればいい．

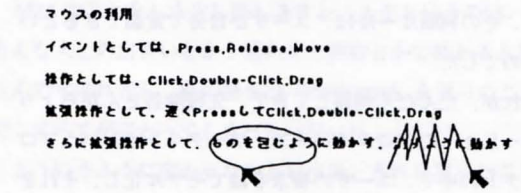


図 1　マウスの動き

つまり，ユーザ・インタフェイスとしての人間がいろんなことができるのに，その選択の幅を切ってしまわないで，違いをコンピュータに伝えたいということです．

(6) 終わりに

いままでお話ししたことは，ツールの操作という面でのユーザ・インタフェースの話題でしたが，最後にもっと大きな視点で，つまりわれわれ人間の世界にツールがどう入ってくるか，逆にいえば，たとえばソフトウェア開発の環境があって，その中に人間がどういうふうに入って行くかという意味で，ちょっと意見をいいたいと思います．

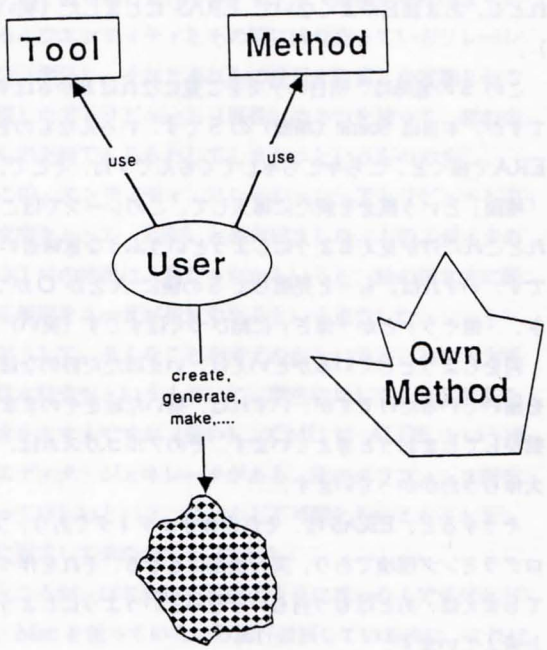


図 2　ツールと方法論

ユーザがツールを使う．方法論を使う．ツールと方法論が密接に結びついている場合もあれば，全然無関係な場合もある．たとえば，開発の能率を上げるために，それを目

標にしたツールが多くなって来ています．技術者がそれらを使おうとするとき，自分のいままでの仕事のやり方とまったく別の方法論を支援するツールを使いなさいといわれても，なかなかむずかしい．

ERASでは，そこをどうするかといえば，さっきいったように，ユーザ自身がルールを定義する，つまり ERASは，メタ方法論しか提供しません，あなたがいままでやってきたノウハウをここに入れて下さい．それで，全体をあなたの仕事の役に立てて下さい，という方針です．多分この機能は，これからのツールにとって１つの大切な方向ではないかと考えています．

それで，いま，ERASの開発が遅れているのはなぜかというと，ツールとメソッドを切り離してしまった．ツールを美しくするために，自分自身をルール（メタルール）で定義してしまおう，つまり自分で自分をモデル化してしまおうと考えているのです．蛇が自分の尻尾を食べているようなものなんですけれども，いろいろむずかしいことが起りまして，あっちを直すとこっちも直さないといけない．それで遅れているんですけれども，方針としてはなるべくツールと方法論を切り離してしまう，つまりペンチで釘を打ってもいいんではないかというふうな考えです（笑い）．

ちょっと話が中途半端で終ってしまいましたが，以上です．（拍手）

(7) 質疑応答坂下： なかなか面白いお話でしたが，何か質問はございませんでしょうか．

Q： ERASのルールを ERASで記述するということですが，もう少し詳しく説明してください．

新田： 実はブート・ルールがあり，それで自分が作ったメタルールを読み込みます．するとある程度のメタルールができる．つまり，すべてのことは書けないが，ある種のことは正確に書けるというルールをまず作ります．それでいまから作ろうとするものをデバッグするわけです．

Mac の上で作っているのですが，Mac 上のアプリケーションというかファイルはトリッキーで，すべてのファイルが２つの部分から成り立っているんですね．１つはソース部分，１つはデータ部分と呼んでいます．ソース部分の中にプログラムが入っています．データ部分は，ふつうわれわれがデータと呼んでいるものが入っている．

アプリケーションはふつう，データは空っぽで，ソース部分がいっぱい．ふつうのデータ・ファイルはデータがいっぱいで，ソース部分はゼロなんですけれど，ERASは自分のメタルールを自分のデータ部分に持っている．

それで，さっきいいましたブート・ルールで新しいルールを作って cat する．これは Unix の cat と同じで，すると新しい ERASができて，それでいま作ったメタルールを読み込んで，構文チェックをやる．そういうふうなデバッグをしています．

Q： 具体化で特殊なことをしているのですか？

新田： プログラムの部分では，ルールに依存する部分と独立の部分に分かれて，ユーザ・インタフェイスはどんなルールになっても一定，ということになっています．最初に ERASが動くと自分自身をもう一度読むんですけれども，その時にルールをもう一度コンパイルするんです．自分で自分をコンパイルして動き出すんですけれども，そのコンパイラの部分がメタルールに依存しているだけで，あとは全部フリーです．

Q： そのことと，ユーザが自分のルールを定義できることとの関係は？

新田： ERAS については，ユーザが自分でルールを定義するといいましたけど，どうやって定義するかというと，それを ERAのダイアグラムで描く，ということです．

それで，ERA が立ち上がった時にまず自分自身のメタルールを読んで，コンパイルして，自分自身の動作を決めます．それからユーザに「あなたがいまから書きたい図式は，どのルールにのっとっていますか」と聞いて来ますから，ユーザは自分で作ったルールを指定して．これで書きますと指定するわけです．

坂下： それでは約１０分間休憩を入れてから，討論に移りたいと思います．

発表をきいての感想
(参加者アンケート)

山浦恒央(日立 SK)

吉村さんの発表は，ユーザ・インタフェイスの本質をつくもので，現場とプログラマのギャップの例がおもしろく，興味深かった．しかし，雑感で述べられたことは，ソフトウェア工学の教科書に書いてある範囲を出ていない．「たとえばこういう方法はどうだろう」といった形で，たたき台的なものを提案してほしかった．

「方法論とツールを分割し，使用者が方法論に適合した環境を構築する」という新田さんのアプローチは，私自身の考えとまったく異なっており，新鮮であった．ただ，マウスの機能を拡大して，いろいろなオペレーションを行わせるというのには，反対．マウスが大きくなりすぎると，結局は，キーボードと同じになってしまう？

杉田義明(SRA)

青木さんの発表は，Smalltalk 屋さんたちに固有の閉じた世界を感じた．もう少し専門用語を省略して，一般人にわかりやすく話してほしかった．しかし，着想(Smalltalk と Unix をつなぐブリッジ・ツール)はきわめてユニークであり，成果が期待される．吉村さんの話には，まさに「現場」の匂いを感じた．このような個々の改善を積み上げていけば，実践的な環境の整備が進むと思われる．新田君の ERAS のプレゼンテーションはおもしろかったが，究極のインタフェイスを目ざすという狙いがよく伝わらなかったように感じた．

佐藤千明(長野県協同電算)

ツールと方法論の分離は疑問．何らかの方法論を支持するツールが必要だと思います．Smalltalk on UNIX の操作性はすばらしい．J-Star(MESA) でも似たようなことができます．これらのツールは，何気なく当たり前に使っていますが，ビジュアルなインタフェイスは，これから必須でしょうね．

佐原　伸(SRA)

自分にない考え方や発想を聞けておもしろかった．しかし，行番号付のエディタが，Mac の上にも生き残っていたとは！

田中慎一郎(SRA)

事務局の手伝いでちょろちょろしていたので，あまりよく聞いていなかったが，ほんとうの雑感を記すと..... 青木さん: SUMMIT の命名に感心．吉村さん: ウィンドゥの汚い重なりは私も嫌いです．新田さん: マウスの利用法のは同感．

歌代和正(SRA)

吉村さんの発表は，現場をこの間見てきたので，おもしろく聞かせていただきました．HST は若い人が多いので，ユニークな考えがあふれているようですね．

新田さんの ERAS は，内容的にはおもしろかったのですが，何に使うのか(使えるのか)がまだよくわかりません．

加藤　朗(東工大)

「....がよい(または悪い)」は，すべてのこと(対象，領域，分野 etc.)についていえるのでしょうか．もし，そうでないとしたら，「....の場合には」という前提を明確にすべきでしょう．吉村さんのいうように，話は「ユーザごとに異なる」わけなのですから．

「××で○○というのがありますが」すみません．○○を知らないものでわかりません．10 秒程度でもいいから，説明してほしかった．ところで，青木さんのシステムですが，「ほんとうに」ユーザ・インタフェイスは，改善されているのでしょうか？

吉村さんの雑感(？)は，(性的ではあるが)するどい指摘だと感じました．学習するユーザ・インタフェイスの実現には，プログラムの実行部と，ユーザ・インタフェイスを分離することが必要でしょう．行番号の件でいじめられましたが，やはり，あれは便利．それほどヒドイものではないと思います．

時間軸の導入という新田さんのアイデアは鋭い．が，慣れていないとうまくいかないかな，という気がします．「ユーザ・インタフェイスは単純に，作業内容は複雑に」という正反対の要求のトレード・オフは？

吉村鉄太郎(管理工学研究所)

個々のユーザの好みや要求に応じて MMI をチューンナップすることが，もっと行われるとよいと思う．

村川貴広（HST）

コマンドの流れを可視的にした SUMMIT のユーザ・インタフェイスや，ERAS のように，エンティティーやそのリレーションもグラフィックに表わすユーザ・インタフェイスは，なかなかおもしろいと思ったが，やはり，ビットマップディスプレイ＋マウスの域は出られないのだろうか？（実践的という意味ではいいと思うが）．これから，われわれのところでも，こうしたビジュアル化を試みて行きたい．

三浦あさ子（SRA）

「ユーザの好みに合わせて，違ったユーザ・インタフェイスが提供できる」という考え方は，すごいと思った．しかし，動作の「対象」と「動作」「動作の仕方」でうまく区別できるか？　これはむずかしい．

熊谷　章（PFU）

青木さんの話の前半はよかったが，後半はターゲットが UNIX というのが気に入らない．したがって，前半から後半への議論の連続性が，少し，希薄になったような気がする．オブジェクト表現にさいして，リレーションをどう表現するのか？

吉村さんの話は，ユーザ・インタフェイスとして何を作ったのかが，よくわからなかった．OS と開発者とのあいだのインタフェイスのように聞こえたが，そうなのだろうか？　ロボット自体のユーザ・インタフェイスの周辺には，問題はないのだろうか？

新田さんの話は，内容が実におもしろかった．ERAS の用途は，どこら辺なのだろうか？

臼井義美（JIP）

オブジェクト間の関連を的確に表現できるようなデーモンも，マルチウィンドゥによる書類の山も，現実の世界の混乱をコンピュータの世界に持ち込むことになる．新たなパラダイムが必要ではないか？

吉村さんの話で，マルチウィンドゥを管理するウィンドゥを作ることは，メニューに戻るのと同じではなかろうか？　現実世界の「もの」のイメージを画面にとりこむことが，ほんとうに最もよい環境なのだろうか？　かなりよいことは事実だけれど....

岸田孝一（SRA）

Smalltalk や Mac など，いま世の中で「よい」といわれているユーザ・インタフェイスについて，「よい点」「悪い点」を，新田君があげた Disk & Formatting の話のように，具体的な例をあげて討論してほしい．

井川裕基（JIP）

青木さんのいうように，UNIX コマンドの作り直し(?)は，必要だと思います．どのような形でやればいいのでしょうか？新田さんの ERAS は，動きがわからないので，デモに期待します．

小林貞幸（電算）

話を聞いていて，「Smalltalk を使ってみたい，Mac を使ってみたい」と思ったところで，汎用機の「は」の字，コボルの「コ」の字も触れられていなかったのは，ショック！　にもかかわらず，話の内容は共通の概念として理解できる．不思議．

林　香（SRA）

青木さんの話は，英語にすれば USENIX 向きのプレゼンテーションだと思う．

マルチウィンドゥ・システムの限界に関する吉村さんの指摘は正しい．まさに，机の上の混乱をディスプレイ上に移動しただけ．ユーザ・インタフェイスは「手順と状況」だという考え方は，おもしろい．人間がおぼえやすい．

新田君の話は，現在の新しいユーザ・インタフェイスの動向を非常にうまくまとめている．いまやユーザ・インタフェイスを論ずるとき，Mac や Smalltalk が話題の中心であり，キャラクタ端末は死んでしまった（満足！）．ユーザ・インタフェイスのメタ化というアイデアはすごい．

中野秀男（阪大）

(1) 思っていること：

マルチウィンドゥ上の各ウィンドゥの最適配置は，後ろでプロセスが走って，勝手に（もちろん各人の好みに合わせて）変える．

対象としているもの（文章中の１つの単語など）をクリックすると，関連する他のウィンドゥの内容も自動的に変わる仕掛けはできないか（どうやら，Hypercard のスイッチのしかけや，青木さんのいうように，データにメソッドを埋め込めばいいみたいだ）．

(2) 感じたこと：

いろいろすぐれたインタフェイスができてくるが，ユーザが好みのものを選択できるようになるのか，それとも標準化するのか，どっちだろう？

寺井　孝（日立ビジネス機器）

３氏の発言をお聞きすると，共通点として，各ユーザからのさまざまな要求に答えるために，マシンをより人間に近づけることを目指しているように感じました．

井上尚司（ソフトバンク総研）

コンピュータが人間の行動を制限しないような機械として実現されるのは，まだまだ先のことだと思っています．マン・マシン・インタフェイスの最終目標が達成されるのは，いつの日のことだろう？

西岡健自（横河電機）

青木さんの発表がおもしろかった．絵にして動かすときの一般的戦略があれば聞きたい．

盛田政敏（KCS）

ユニークな話ばかりででたいへんおもしろい．

鎧　友良（FXIS）

３つの発表は，いずれも，マルチウィンドゥを用いて，インタフェイスの改善を試みているが，他にだれか何かいい方法を知りませんか？　アイデアがあれば聞きたい．

藤野晃延（FXIS）

青木君：もう１つ上のレベル，つまり，メタ・レベルへの普遍性を求めることが，今後の課題でしょう．

吉村さん：おもしろい．Macはたしかにユーザ・インタフェイスに革命を起こしうるポテンシャルを持っているし，進むべき方法としては，よい．

新田さん：メタ定義，かつ自己内包的な汎用化の指向で，すばらしいと思う．ただ，ユーザが定義するルールの意味論的な正しさのチェックをどうするかは，問題だと思う．

小林明彦（東電ソフトウェア）

私の興味は，エンド・ユーザへの提供をどのようにしたらよいかというあたりににある．その意味で，エンド・ユーザよりのマン・マシン・インタフェイスの発表がなかったのが残念．

御喜家貴子（YHP）

マウスのボタン操作の早さ（速い／遅い）で，ユーザの気持ちが伝わるというのは，とても興味がある．おもしろいなぁ．

海尻賢二（信州大）

やはり日本語が大切だと感じた．遅れてはいるが，通常のCRT上のユーザ・インタフェイスは？

堀川博史（三菱電機）

青木氏：参考になった．実行速度はどんなものか？　明日のデモが楽しみ．吉村さん：実践的な話で，こういう世界もあるのか（こういう世界ばかりかも知れないが）と思った．新田氏：斬新で，おもしろかった．

セッションを通じて，発表者にバラエティがあり，たいへんよかったと思う．

市川　寛（電算）

マン・マシン・インタフェイスの話はよくわかるが，マシンとマシンとのあいだのインタフェイスはどうなっているのか．もっと，システム全体（開発環境全体）のことを考えてくれる人（グループ）はいないのだろうか．

小池幸徳（日本システムサイエンス）

これからは，ユーザのレベルに合わせたMMIが重要となってくるものと思われる．たとえば，キーボード，マウス，マルチウィンドゥ，タッチセンサーなどの選択．こうしたMMIの選択をどのような方法で行えばよいかが，よくわからない．

田中一夫（山一コンピュータ）

青木氏：わからん．吉村さん：いっているとおりだと思う．新田氏：おもしろかった．

中村　眞（シャープ）

エディタって何？　ソフト作りの仕事は所詮は「もの書き」だから，ただ文房具として，よい鉛筆が必要だといっているのと同じだよね！

近藤康二（ソニー）

青木さん：Smalltalk-80には以前から興味があったので，とてもおもしろかった．内容はまったく同感．UNIXも時

代に合わせて変わらなければ取り残されてしまうという感を強くした．

吉村さん：ロボット・プログラムの現実的な話題（特に質問に関連して）は，苦労がよく感じられた．私も会社に戻るとそうだなぁ．

ERAS：とても理論的な話のように聞こえた．自分で自分を作るというパラダイムは，なかなかのものだと思う．明日のデモが楽しみ．

久保宏志（富士通）

「Algorithm を value に持てる変数の概念が MMI 言語にとって本質的だ」との意見は，傾聴に値する．

酒匂　寛（SRA）

青木さん：Smalltalk を UI 言語とする考え方は，正しいアプローチでしょう．よくいわれる言語の閉鎖性を打ち破る鍵の１つがここにあるように思われます．

吉村さん：「人間が実際に使う」という視点にたったインタフェイスを忘れてはいけないと感じました．

新田さん：ERAS の将来に期待しています．ERA モデリングの極限を見てみたいと思います．

荻生準一（日本システム）

私自身の仕事は，電力供給システムの要求定義やシステム設計が主であるが，このセッションで発表された開発環境における MMI の考え方は，これからの客先との MMI 機能仕様決定に参考になるものが多く，興味深かった．

現在の電力供給システムでは，まだマルチウィンドゥの概念すら取り入れられていない状況である．次のシステムの機能仕様決定には，このセッションでの発表の内容をぜひ取り入れてみたい．

落水浩一郎（静岡大）

青木さん：勉強になった．Smalltalk　を持っていないので，やはり欲しいと思った．吉村さん：がんばってください．ただ，もう少し，話の技術的内容に具体性がほしかった．新田さん：マウスの移動速度等のあいまいなもの（人によって違う）を基準にしてよいのか，どうか？　少し疑問に思う．

岡本隆一（KCS）

３人の方々から，人間の５感に触れるユニークな発表があり，楽しく聞かせていただきました．が，耳に訴える話はなかったようですね．いま，私の職場では，この「耳」を中心としたアプローチを試みている若者がいて，机が隣接している私は，毎日非常に迷惑しています．

こんな話を参加者の皆さんから聞けるといいですね．

桜井麻里（SRA）

青木さんの発表は，整理されてわかりやすかった．吉村さんの発表も，現場の苦労がわかっておもしろかった．人間ができることをコンピュータによって制限されたくないという新田さんの発想は，とても自然だと思うが，それが夢でなく現実になりつつある点が凄いなぁと感じた．

北野義明（KCS）

Mac が実務の場で，こんなに使われているとは思わなかった．やはり，C や Smalltalk-80 でアプリケーションを作っている環境と，COBOL を使ってホスト上でプログラミングせざるを得ない環境との差を痛感した．

MMI のすぐれたシステムを持つマシン（OS）の上で動く COBOL コンパイラが欲しい！　欲しければ自ら作れ，といわれそうだが....

森　幸一（PFU）

アイコンにバインドされるプログラムの種類は，マウスのクリック操作に限定されるのか？　そうだとすると，ユーザ・インタフェイスの上で，かなりの制約になるのではないか？　成長するユーザ・インタフェイスの考え方には賛成．

伊野　誠（SRA）

MMI　はむずかしいが，いろいろな考え方，アプローチを聞けておもしろかった．しかし，もっと基本的な問題があると思う．つまり，MMI に関する基本的な作法とか作り方の指針が存在しないという点である．

討　論

坂下： それでは，これから討論に入ります．御意見または御質問をお持ちの方は，会場のマイクの前に並んでください．

(1) マウスをめぐる論議

林　香（SRA）： 発表全体を通じて，もはや CRT 端末を前提にしてユーザ・インタフェイスを語るのは江戸時代の人間だ，という感じがしました（笑い）．

大変よいことだと思いますが，一方では，TRON の坂村健さんみたいに，「マウスは原始人の道具だ．まるで石で絵を描いているのと同じではないか？」という意見もあります．

発表者３人は，それぞれ，マウスを用いたインタフェイスのお話をされたので，その点についてどう思われるかを，お聞きしたい．

青木： さきほど MVC のお話をしましたが，Smalltalk の中には，ユーザ・インタフェイスをつかさどるコントローラと呼ばれる部分がある．それが，ただ単に，いまはマウスにバインドされているというだけにすぎません．

将来，もしマウスよりもすぐれたインタフェイスが，ハード的に開発されれば，そこを置換えるだけですむ．ですから，マウスうんぬんは，あまりたいした問題ではないんですね．どんなものでも，オブジェクトとしてとらえているので，統一的な形で扱うことができます．

吉村（智）： いま，世の中の風潮としては，「ビットマップ・ディスプレイとマウスさえあれば，たちどころに，よいユーザ・インタフェイスが提供される」というようなムードがあるみたいですが，それはちょっとまちがっていると思います．

たとえば，たくさんのウィンドゥを開いて仕事をしているとき，さっきもいいましたが，何かをやりたい場所にいちいちマウスを持って行かねばならないのは，ものすごく，わずらわしい．もうちょっと，たとえば音声で判断するとか，視線のやり方でできるだとか，そういう仕掛けができたらスゴイなぁと思うんですけど.....

新田： TRON の悪口をいってもいいのかな？

坂下： どうぞ，別に，かまなわいんじゃないですか？（笑い）

新田： たしかに，マウスは石鹸です（笑い）．去年，TRON シンポジウムに行ったら，ワイヤレスのペンで書くタブレットが展示されていて，関心しました．TRON チップに次いで，すばらしい発明だと思います．

しかし，マウスを使うのは原始人だという悪口に対して反論すると，TRON のあのキーボードは，人間以前の代物です（笑い）．

なぜかといえば，人間工学を考えて設計したというのですが，人間を肉体的にしかとらえていない．ところが，人間には精神というものがある．そうした観点から見ると，あのキーボードは非常に使いにくい．あれを使って，ちゃんとタイプしている人を見たことがない．

坂下： その，精神工学的によくないという点を，もう少し説明してください．

新田： たとえば，この指とこの指が交互に打ちやすい所にあるというのは，肉体的工学から見ればよいことですが，精神的な目で見ると，文字がアイウエオ順に並んでいる方が，よほど使いやすいわけですね．

もし，TRON が専門タイピストを目標とするプロジェクトなら，あのキーボードは，すばらしいといえるでしょう．しかし，TRON は，もっと広く一般の人が使えるコンピュータを提唱しています．そういう人たちにとっては，少しぐらい指の動きが遅くても，キーがアイウエオ順やＡＢＣ順に並んでいるほうが，親しみやすいのではないかと思います．

(2) 現実世界の混乱をどう整理するか？

臼井義美（JIP）： ３人の発表者のお話は，いずれも，マルチウィンドゥを使ってマン・マシン・インタフェイスをよくしよう，という趣旨のように聞こえました．世の中にある実物と同じようなものを，画面の上で操作して動かせる．そういうのが使いやすいインタフェイスじゃないかという方向へ，何となく議論が収束して行くように思われます．

しかし，世の中にすでに実在するものを，わざわざコンピュータの中に，絵画的なイメージで再現しなければならないというのは，まったく，進歩したことにはなっていないと思うんですね．さきほど吉村さんがいわれたように，スクリーンの上が，いつのまにか，机の上のゴミの山と変わらないようになって行くのは，防ぎようがない．

また，いろいろなオブジェクトの間をつなぐ（関係づける）デーモンについても，大変面倒な問題があると思いま

す．

たとえば，私がいまこうやって話しているというのは，ここにおられるみなさん，つまり，１人１人の人間というオブジェクトに，何らかのメッセージを送っていることにほかなりません．

そのとき，受け手であるみなさん（各オブジェクト）は，同じ私の話を聞いて，それぞれ少しずつ異なった解釈をしているにちがいない．そうした関係づけをどうするかは，非常にむずかしい．仮に何とか関連づけができたとしても，１つのオブジェクトを取りあげたら何千何百という糸がついてくるようでは，ほんとうにシステム全体の整合性をデバッグできるとは思えません．

つまり，図式化されて一見使いやすそうなマン・マシン・インタフェイスを，機械の中に組み込もうとしているのだけれど，よく考えると，現実世界の混乱をそのままコンピュータに放り込もうとしているのではないか，という気がしてならない（笑い）．

これまで，われわれはそういう現実の混乱した姿を無視して，というか抽象化して，コンピュータで動く数学的にきれいなモデルを作り上げ，なるべく理論的な，ロジカルな話に持って行こうとしてきた．現実世界の混乱をどんどん取り込んでも，ほんとうに役に立つコンピュータ・システムができるのかな？　という根本的な疑問を感じます．そのあたりのことをどう思われているのでしょうか？

青木：　たしかに，オブジェクトをたくさん作って行くと，それらのあいだの関係を選り分けることができる何らかの機構が必要になるでしょう．

つまり，それぞれのオブジェクトが，自分のテリトリ，自分が見ることのできる範囲をはっきりわかっていて，その中でさまざまな関係を操作できるような仕掛けです．

そうした関係の操作という場面では，Plorog　みたいな言語が，これから力を発揮してくるのではないかと，言語屋としては，考えています．

新田：　直接の答にはならないのですが，私の考えていることをお話します．

現実世界とコンピュータの世界とのちがいは，たとえば，私の机の上がどんなにちらかっていても，欲しいファイルはちょっと探すだけで手に入るわけですね．ところがそれをコンピュータの中へ入れてしまうと，なかなか探すのがむずかしい，というあたりにあります．

現実の世界にはいろいろな混乱があるけれども，人間は何とかうまくやっている．それを，コンピュータ化した瞬間にうまくいかなくなるのはなぜか？

私が思うには，現実世界のオブジェクトのあいだの境目は，ディジタルではなく，もっとあいまいです．境界領域では，あるオブジェクトの濃度がだんだん薄くなり，そのうちに他のオブジェクトに変わって行く．そんなふうになっていると思うんですね．

それをモデル化して，コンピュータの世界に入れるときには，ここからここまでが１つのオブジェクトというふうに，はっきり区別しなければならない．

現実世界では，混乱しながらも何とかうまくやっていることを，コンピュータの中へ持ち込んだ時に，あいまいさを取り除きすぎて失敗しているということだと思います．

(3) 人間の動作のモデル化

落水浩一郎（静岡大）：　新田さんのプレゼンテーションの中で，人間には不器用な人も器用な人もいるので，それに合せて調整するというお話はわかるのですが，マウスの動かし方のように，基準のはっきりしないものをマン・マシン・インタフェイスに取り込むのは，いささか危険ではないでしょうか？

マウスを動かす速度は，人によって違うから判定がむずかしいでしょうし，また，マウスをガシャガシャ動かしてモノを消すということですが，何かの加減でカーソルがどこかへ飛んで行ってしまった場合，それを戻そうとしたときに画面が全部消えてしまわないのですか？（笑い）

新田：　TRON の坂村先生は，「自動車の運転はメーカがちがっても標準化されているのに，なぜコンピュータはそうなっていないのか？」とおっしゃるのですが，ハンドルを同じ角度だけ曲げたとき，すべての車が同じだけ曲がるかといえば，そうではない．アクセルを同じ角度だけ踏むと，同じだけスピードが出るかというと，そうでもない．そういう意味においては，自動車は，１台１台バラバラなんですね．

ところが人間は，そういうバラバラな特性を持った自動車をうまく乗りこなしています．それは，人間に学習能力があり，フィードバックの能力があるからだと思います．

私が ERAS に組み込もうとしているマウスの動作速度やパターンの認識についても，人間がその働きをうまく学習できるということが，１つのポイントです．

つまり，マウスをガシャガシャやって，自分としてはただ向こうに持って行くつもりだったのに，画面が消えてしまった．「あ，これは間違いだ」と気づく．もちろん，そ

う気づいたら，すぐ元へ戻す機能（Undo）が用意されています．

今度は，意図的にガシャガシャやったら，うまく消えた．「なるほど，こういうふうにやれば消えるんだな」と学習できる．マウスのスピードについても，同じように，ゆっくり動かすとカーソルの形が変わって，これだけのスピードならこうなるんだなということをユーザが発見し，学習できるような，そういうユーザ・インタフェイスを作りたいと思っているのです．

井上尚司（ソフトバンク総研）： その話の続きなんですが，コンピュータの方には学習機能はないんですか？

たとえば，消しゴムがA，B２種類あって，実際に消してみたらAの消しゴムはよく消え，Bは消えが悪いとすれば，人間はAのほうしか使わなくなると思うんですが，コンピュータの上にはBタイプの消しゴムしか用意されていなかったりすると，何となく「そんな環境に馴らされるのはいやだ」という人が出てくると思うんでが，そういうことは考えていませんか？

新田： その問題には，ある種の対処法を考えてあります．いま作っている ERAS では，たとえばマウスのスピードに関するパラメータを，手の感覚で何となく違うなという３つか４つの種類の中から，１つだけ選択するようふうにしています．そんなふうに，自分の好みを機械に伝えられるわけです．

坂下： その選択にさいしても，インタフェイスがあるんですか．

新田： Mac には，コントロール・パネルというものがあって，マウスの動作速度とかキーの反復のスピードとかを，ユーザが自分の好みに合せてある程度調節できるんですが，その仕掛けを利用しています．

(4)「マウス＝鉛筆」論

中村 真（シャープ）： これまでの議論を聞いていると，マウスは，いわばわれわれにとっての鉛筆であり，それが最近なかなか使いやすくなったというお話のように聞こえます．

ことばを変えていえば，木彫りをやるのに，一刀彫りで行くか，各種セットの揃ったのを使うか，２通りあるわけですね．非常にセンスのよいツールを作って，かゆいところの隅々にまで手がとどくような機能を持たせるのか，あるいは，もっとシンプルなものにして，人間の方がそれに適応するか，ということです．

現実の社会の中でも，ときには靴に足を合わせ，また，別の場合には足に靴を合せるという状況が存在して，それぞれ一長一短あります．「それなら両方選べるようにしてしまえ」というのは，乱暴ないい方でしょうか？

いま，マン・マシン・インタフェイスは非常に多種多彩なものが出てきて，たしかにそれぞれ便利そうに見えるんですが，はたして，ほんとうに仕事をする時に便利なのかどうか，疑問を感じることがあります．そのあたりをどうお考えなのか，うかがいたいと思います．

青木： 私がお話したプログラマブル・アイコンの概念は，まったく，いまおっしゃったのと同じ考え方です．マン・マシン・インタフェイスの拡張という見方からすれば，１つのアイコンの中にバインドできるプログラムの質，それが一刀彫りであったり，多様なセットであったりします．

それをユーザが選べるような仕掛けを持ったシステムを作り，そういうものを使って，その上に一刀彫りのインタフェイスをつけて出すのが，現状で考えられるベストだろうと思います．

新田： 私は，さっき Mac の上で仕事をしているといいましたが，実は Mac の上で Shell と vi と C コンパイラを動かして，コマンド・インタプリタで仕事をしています．要するに一刀彫りでやっているということですね．基本的には，人はそれぞれ自分の好きなものを選べばいい．だれからも選ばれないツールは消えて行く，そういうことではないでしょうか？

佐原 伸（SRA）： マルチウィンドゥとマウスは，どこかまだおかしいけれど，取りあえず使いやすいからいいのではないか，という論議のようですね．

私はいま Mac と Unix の両方を使っていて感じるのは，どちらもたしかに使いやすいんですが，使いにくいところもある．それは，目的によって違っていて，たとえば Mac は，頭にひらめいたことをすぐに具体化するときは使いやすいけれど，いわゆるバッチ処理的に，大量に同じことをやるのには向いていない．

つまり，ユーザ・インタフェイスには，大きく分けて２種類のものがあり，その片方の話しか，みなさんしていないんじゃないかと思います．

さきほど臼井さんが，人間のモデルを混乱状態のまま取り込むのはいけないといわれましたが，現在のユーザ・インタフェイスのほとんどは，しっかりしたモデルなしで作られています．それよりは，人間が頭のなかに持っているモデルの方がまだマシだ．そこで，一刀彫りのかたちで，

人間のモデルを利用しよう，ということになっているのではないでしょうか？

混乱を持ち込むのは人間の方が悪いので，コンピュータやソフトウェアが悪いわけではないという気がします．

(5) Smalltalk と Unix，そして行番号の問題

佐原： ところで，青木さんにうかがいたいのは，どうして Smalltalk で Unix をいじろうとしたのかということです．

私が考えるに，Unix はそれなりに使いやすいけれど，すでに過去の異物になりつつある（笑い）．だったら，別に Smalltalk で一生懸命 Unix の面倒を見る必要はないんじゃないかと思います．

それから，吉村さんへの質問は，さきほどの行番号のことですが，やっぱり，ああいうところで妥協してはいけないんじゃないか（笑い）と思うのですが，いかがでしょうか？

行番号をなくすというのが，Mac や Unix の１つの基本思想であるわけで，そこを曲げちゃったら，ユーザ・インタフェイスの基礎としてのモデル自体が，意味をなくしてしまうんではないかという気がします．

青木： Xerox 自体は独自の OS を持っていて，全然 Unix なんかに目もくれなかったというのが，過去の歴史です．ところが，いつのまにか Unix が世の中にはびこってしまい，このままでは Xerox のマシンが売れなくなった（笑い）．そんな流れの中で Unix の上に Smalltalk の仮想マシンを乗せようか，などというプロジェクトが起こってくるわけです．

ひとことでいえば，世の中に流されているわけですが，私個人にとってのモティベーションは何だったかといえば，周囲にこれまで Smalltalk や J-Star の環境に慣れ親しんだ人たちが大勢いて，かれらがなかなか Unix になじめなかったんですね．佐原さんのように Unix も Mac も両方好きという人ばかりだといいんですが，ふつうはそうではない．そこで，その人たちのために，むかし Unix を経験して C 言語が分る人間が一肌脱ごうということで，たまたま手がけた仕事です．

吉村（智）： 行番号の話ですが，開発しながら，たしかにMacの思想と行番号は一致しないなと感じていました．

でも，ユーザの要求をまず第一義に考えてあげたいという気持ちがありまして，そうするとあちらは，どうしても行番号をつけて欲しいというわけですね．こちらがそんなものはいらないといっても，これまでずっと Basic に慣れ親しんで来た人たちですから，どうしても 'GOTO 10' をしたいとか，'GOSUB 1000' と書きたいという（笑い）．

そういう要求がまず１番目に出てきたので，あえて今回は，システムとしての整合性に目をつぶって行番号をつけました．

佐原： そうした表面的なユーザの要求に応じても，開発生産性の向上は，せいぜい２倍がいいところです．しかも，'GOSUB 1000' というのは，初期のマイクロソフト系 Basic の話で，現在はそうしたステートメントを使わなくてもすむように，Basic ですら改良されて来ています．それを元に戻してしまうのでは，せっかく Mac を使うのに悲しいんじゃないかな．もう少しがんばって，ユーザを何とか説き伏せて欲しかった．

私も，むかし COBOL の構文エディタを作るプロジェクトをやったとき，行番号をつけろという圧力がありましたが，「ダメだ．たとえこのツールが使われなくてもいい．行番号はつけない」とつっぱって，結局，そのエディタはあまり使われなかった（笑い）．

それは，かれらが行番号と一緒にそのうち滅亡して行くんだから，それでもかまわないと思っています（笑い）．それくらい過激にやったほうがよいのではないでしょうか？

吉村（智）： 今回は，たしかに私の力不足で，「行番号なんていりませんよ」という説得が通用しませんでした．次はもう少し強硬にやってみます．

新田： 私は少し佐原さんとちがう考えで，たとえば，デジタル・ウオッチの時代になっても，やはりアナログにこだわる人がいるように，むかしながらの習慣を好む人がいつもいるんだと思います．そういう人たちのことを，少しは考えてあげなければいけないという気がします．

ところで，行番号をなくすにはどうすればいいかというと，「このツールはとても便利ですが，行番号を使っていると，時々落ちることがあります」（笑い）という意地悪な仕掛けを組み込んでおけば，自然にみんな行番号を使わなくなると思います．

深瀬弘恭（アスキー）： かつて，Basic のディストリビュータであった立場から（笑い），いまのお話の状況説明をさせていただきます．

たぶん，お使いになっている Basic は，もともと小さく作られたシステムなんですよね．あれは８バイトだからこそ，ああせざるを得なかったという事情がある．一方，

Mac の上で動く Basic のインタプリタやコンパイラの場合，そんな制約はまったくないわけです．構造化 Basic をちゃんとサポートできるわけですから，なければしょうがないけれども，あるものを使わないというのは，私もちょっとおかしいなと思います．

そういう歴史的な背景はあるんですが，言語もコンパイラも単なる商品ですから，よくないと思えば捨てればいい（笑い）．いまあるものの中から，何を選ぶかということだと思うんです．

(6) ウィンドゥとユーザ・モデル

深瀬： それから，私からの問題提起ですが，みなさん「マルチウィンドゥを開いてマウスで仕事をするのが便利だ」といわれますが，私自身 Mac を使ってみて，必ずしもそうは思いません．

Mac のスクリーンの情報量は非常に小さいので，あの上でウィンドゥを開いて仕事をするのは，机に書類が山と積まれている．そのどこか片隅にすきまを見つけてメモをとる（笑い）というイメージに近い．

もちろん Mac には他に，サイドキックなどのように，画面を全部入換えてしまう仕掛けも用意されていますが，ユーザに提供できるインタフェイスの環境は，どんな場合にも程度の制約があるので，その中から何を選ぶかは，かなり重要なファクターだと思うのです．

中野秀男（阪大）： 悲しいことに，うちの大学ではまだ 'GOTO 100' を教えています（笑い）．

それはそれとして，いま，卒研のテーマで学生が１人，ウィンドゥの話をやっていますので，もしよろしければ，みなさんの御意見をお聞かせいただければと思います．

まず，ウィンドゥが見にくい．これについては，かなりお節介なウィンドゥ・マネージャを考えて，ウィンドゥの最適配置を行おうとしています．最適配置というのはちょっといいすぎで，各ユーザ固有のルールに合せて，画面に触っているうちに，ウィンドゥが自然に動いてくれればいい．

それから，ちらかった書類については，一種の電子秘書みたいなプロセスを，バックグラウンドで動かせばいいんじゃないかと考えています．

もう１つのアイディアは，ユーザが視点（つまり目）を動かすと，関連するウィンドゥが（お節介マネージャーとか電子秘書の手で）自動的に連動するようにしたらどうかな（笑い），ということです．ハイパーテキストなんかで，もう，そんな機能が試みられているのかも知れませんが.....

落水： 率直な感想をいわせてもらば，ちょっと凝りすぎではないかと思います．

さきほど吉村さんがおっしゃった，状況とか手順によって情報にアクセスするというのは，非常に面白と思いましたが，視線の動きにつれて画面がススッと動く仕掛けは，作るのに苦労する割には，あまり効果はないんじゃないかと思います．ひとことでいえば，あまりよくないといってるんですが....（笑い）．

岸田孝一（SRA）： お手元に配布したアメリカの環境ワークショップのレポート（SEAMAIL Vol.2, No.9）の３４ページに，ユーザ・インタフェイスの項目があって，さっき臼井さんが出された問題と，いまの中野先生のお話の両方に関連するような指摘があります．

マウスとかビットマップとかウィンドゥは，単なる仕掛けというか，ユーザ・インタフェイスを実現するためのメディアに過ぎないわけで，インタフェイスを考える上で１番大切なことは，ここに書いてあるように，「システムがユーザの行動をどんなモデルとしてとらえているか？」だと思います．

つまり，ユーザがワークステーションを使って仕事をするプロセスや，結果として得られる（ドキュメントとかプログラムとかテスト・データなどの）プロダクト全体の体系をきちんとモデル化しなければ，よいユーザ・インタフェイスは作れないだろうということです．

ただ問題は，モデルというのはある種のスタンダードですから，それが確立されると，ユーザの行動はかくあるべしだとか，開発工程はこういう順序でなければならないとかいった形で，何らかの強制力が働いてくる．

いま Smalltalk とか Mac とかの評判がいいのは，そういう意味でのモデルを持たないシステムだから，ユーザが勝手に自分の頭の中にあるモデルに合わせて，ユーザ・インタフェイスの仕掛を使えるからでしょう．そういう特徴が，ごく一部の人たち（ハッカーたち？）に非常に好まれているのではないでしょうか？

しかし，「ふつう」のユーザにとっては，あらかじめシステムの中に仕事の手順についての大まかな標準モデルが存在し，その細部は自分でカストマイズできるようになっていたほうが，便利だろうと思います．

いまの中野先生のお話は，その１つの具体例のように感じました．また，さっき臼井さんがいわれたことに私は賛

成します．この世の混乱をモデル化せずにそのまま持ち込むのではなく，少し整理すれば，ユーザ・インタフェイスはずいぶん改善されるのではないでしょうか？

佐原： たぶん，事実関係をまちがえていたと思うので訂正しておきます．

私や青木さんがいっているマウスとかマルチウィンドゥとかは，オブジェクト指向風の一種のモデルをベースとするものであり，そうしたソフトウェアと密接に結びついています．ひょっとして，一部の人（たとえば，むかしのPC100ユーザとか）は，物理的にウィンドゥが複数になれば，それだけで使いやすくなると誤解しておられるような気がする．

SmalltalkやMacのように，何らかのモデルをベースにマルチウィンドゥやマウスを使っているシステムと，そうではないものとは，はっりり区別しなければいけません．

ところで，中野先生のいわれた視線の動きを追うという試みですが，これはすでに誰かやっています．それによると，人間の視線というのは，きちんと１点だけを見ていないらしいですね．ウィンドゥを見ていながら，ときどき遠くの女性に目を走らせたりとかで（笑い），どうもその線はダメなようです．

(7) 定量的分析の必要性

加藤　朗（東工大）： ユーザ・インタフェイスを考える時に，提供しているシステムのモデルの話はもちろん重要で，いままで議論されていたところですが，もっと大きな問題は，たぶん，ユーザのモデルのほうにあると思います．

ところが，そうした研究はこれまであまりされていない．きわめて原始的な問題，たとえば，ある入力を受けつけるのに，メニューを出してマウスでたたかせるべきか，キーボードから選ばせるべきかというような場合に，設計者の趣味で，やっぱりマウスの方がいいだろうとマウスにしてしまうというのが現状でしょう．

しかし，この程度のレベルでは，それなりにきっちり研究している学者が何人かいて，ユーザがキーを打つのにどれくらい時間がかかるとか，キーボードから手をはなしてマウスをつかみ，さらにポインティングするまでにはどれくらい時間がかかるかといった定量的な測定や分析がなされている．そうなってくると，ユーザ・インタフェイスの評価の方法もだいぶ変ってきます．

ところが，もう少し複雑な問題になると，いまのところ，そういう定量的な分析がないので，議論が宗教的な方向に流れてしまう．たとえば，行番号があってはいけないという人と，欲しい（または必要だ）という人とのあいだの哲学論争になってしまうわけですね（笑い）．

行番号があってはいけないという人は，なぜあってはいけないかを具体的かつ定量的に説明すべきであり，それができないなら両方用意しておくのがツール提供者としては正しい．ですから，吉村さんはそれ程いじめられる必要はない（笑い）というのが私の意見です．

佐原： プログラム・エディタについては，かなり定量的な評価が行われていて，たとえば，Xeroxでの分析結果によれば，行番号つきのコマンドを主にするようなエディタの場合，エラーの発生率が非常に高いという数字が，すでに出ています．その数字をどこまで信用するかは別として，大体５倍位の開きがあるといわれています．

加藤： プログラムの作り方にもよりますが，１つの手続きが長いものを作る人は，行番号があると自分がいま手続きのどの辺にいるかがよくわかる（笑い）．だから，フラットテキストの場合なんかよくないんだろうけど，うまく使えばそれなりに使える．使いたい人は，たぶん，そういうところがうれしいから使っているのでしょう．そういう人がいるかぎり，やはり行番号が必要だと私は思います．そういう人は，多少効率が悪いのを容認しつつ，「やっぱりいいもんね」といってるわけですから．

(8) リレーションについて

熊谷　章（PFU）： これまでの討論を聞いて感じたことは，コンピュータをビジュアルに使おうという考えが，これからますます重要になってくるということですね．人間は，他人に意味を伝える時に，あらゆるメディアを使っていますが，マシンとの意志伝達も，それと同じようにマルチメディアで考えるべきでしょう．

また，コンピュータ自体も，単なる計算機械ではなく，マルチメディアのコミュニケーション装置である．そう考えると，コンピュータの中に何を入れてあるかによって，マン・マシン・インタフェイスを議論すべきであり，当然話の切り口が変ってくる．青木さんのシステムも，ただUnixの上にSmalltalkの皮をかぶせたという観点だけから論じても，あまりおもしろくありません．それでは，Unixにできることしか表現できない．本来，マウスとかマルチウィンドゥは，マン・マシン・インタフェイスを論じる場合のメインテーマではないはずです．

ところで，１つ単純な質問なんですが，リレーションを

どうプログラミングすればいいか，工夫があったら聞かせて下さい．

青木： 要は，関係性だけを操作できるものを，オブジェクトの世界の上に持つんです．そこからいろいろ操作するという形でリレーションを具体化しています．オブジェクト指向でリレーションを扱って行くと，結局はオブジェクトになっちゃうんですけれども，私自身は，リレーションはオブジェクトではないと考えていますので，そういう具体化をして行きたいと思います．

落水： その点に関して，どういうリレーションを中に埋め込まれているのか，人間にまかせる問題は何で，ユーザ・インタフェイスで処理する問題は何で，どういうタイプのものがユーザ・インタフェイスの改善に貢献するのか，というお話をおうかがいしたい．

それから，吉村さんには，ユーザ・フレンドリというのはわかるんですが，個別にいちいち対応していたらきりがない．だから，何かジェネリックな仕掛けを用意して，カスタマイズして行くことになるのでしょうが，メカニズムの話に入る以前に何らかのポリシーが必要だと思います．そのあたりに関して，大学の人間にはわからないような，開発現場での実感をお聞かせ下さい．

青木： 本質的には，オブジェクトがなくとも関係性だけはこの世の中に存在するだろう．関係からすべてのものを起こせるんじゃないかという考えを，自分の根底に持っています．実際にものを扱う時には，対象世界へ落しオブジェクト化して，みんなに見せる．ですから，さきほどお見せした Unix も，その１つのアプリケーションに過ぎません．本質的には，関係とかオブジェクトの方向へ，自分ではどんどん行きたいなと思っています．それをどうやって作るかは，いま模索中です．

(9) ユーザ・モデルについて

吉村（智）： ほんとうは，ユーザ１人１人の好みに合ったものを作りたいんですが，現状としては，ジェネリックなものをいくつか提供して，それをカスタマイズするアプローチしかできません．そうした過程で，ユーザが使っていくうちに，１人１人のくせや作業スタイルに応じてユーザ・インタフェイス自体が変化していく，というようなものを提供したいと考えています．

新田： 私の ERAS の開発方針は，青木さんと逆で，"What You Sea Is What It Is." つまり変なモデルを持ち込んで（笑い）何かの皮をかぶせるのでなく，ユーザが見ているものがそれ自身だということです．ユーザの目から何かをかくして，ほんとうにあるものじゃないものを見せるようなやり方は，いけないんじゃないかと思っています．

藤野： たぶん新田さんのいうあるものをあるがままに見せるというのは，それも１つのモデルなわけで，ユーザに幻想を見せるという Smalltalk のやり方も，また１つのモデルです．さらにまた，さきほど落水先生がいわれたジェネリックなモデルも考えられる，これもまた１つメタ・モデルだと思います．

要は，どこからスタートするかということだけで，基本的には，ユーザが自分でカスタマイズできるジェネリックなものが必要でしょうね．１番最初にどこから始めるかが，まだよくわかっていない．さきほど，コンピュータの世界へ現実の混乱を持ちこむことのよしあしが議論されましたが，実は，コンピュータの上では，現実の世界ではできないこともできるわけで，そこまで考えて踏みこんで行かないといけないと思います．

中村： 私は，いまの仕事に入る前は，大学で脳の研究をやっていたんですが，われわれ人間は，たとえば，歩行するとかものを掴むとかいう動作を，意識せずにやっています．それは，小脳の部分に内在したモデルを持っており，そのへんが下位の神経組織とフィードバックをやりながら動いて行くんです．

マン・マシン・インタフェイスに関する議論では，よく，ユーザ・フレンドリであること（つまりある意味でユーザにこびること）が大切だといわれますが，しかし，実は，これが人間のほんとうの姿ですよと押しつけても，ほんとうにそれが正しければ，それを使い慣れると使いやすいということがある．

だから，いまここでいろいろと論じられている問題のなかで大切なのは，いったいどういうモデルが人間の行動を正しくとらえているかを考えることであり，また，いかにしてユーザにそうしたモデルを認識させ，それに移行させて行くかが，ポイントになるのではないかと思うのです．

さきほど，キーボードを打つのに何秒かかるかといった話がありましたが，それよりは，人間に内在するモデルをいかにして構築できるかということのほうが，ユーザと機械とのあいだのインタフェイスを豊かなものにする上で，重要だと思います．そのへんで，最適なモデルが何かについては，自分自身もまだ，答を出せずにいるわけですけれども.....

堀川博史（三菱電機）： インタフェイスがユーザごとに

どんどん変るというのは，アプローチとしてはよくないのではないかという気がします．

ユーザは，スイッチを入れただけで，自分の好きな環境が動きだすようになっていて欲しいわけで，わざわざ苦労して，いろいろな定義を変えたりしたくない．ユーザ・インタフェイスが自由に変えられるような機構を用意し，個々のユーザに対し，そのジェネリックな機構を使って，バラエティに富むインタフェイスを提供した場合，あとのお守りがきわめて大変になるでしょう．

したがって，システムに内在すべき本質的なモデルを，たとえ若干まちがっている部分があっても，あらかじめ作っておくことが本筋ではないかと考えています．ユーザの目的とその作業のモデル化がまずあって，その上でインタフェイスというのを考えるべきであり，個々のユーザの恣意的な要求にしたがって流されて行くのは，決して正しいやり方とはいえないように思います．

ただ，モデル化の議論のなかで，ジェネリックなユーザ・インタフェイスのモデルというのは，どうもよくわからない．もう少し個別的なモデルの話を徹底すべきだと考えます．

われわれのところでツールを作るときどうやってユーザ・インタフェイスを決めるかといえば，ターゲット・マシンの上で利用可能なインタフェイスのメカニズムを，一通り使ってみてから，適当に選択・組み合わせを行なうという形になります．あくまで，コンピュータあってのインタフェイスですから，対象とする機械に依存して変わって行く．

そういう風に考えると，ユーザ・インタフェイスというのは，これが基本だという確固としたよりどころがない領域ではないでしょうか．頼りになるのは人間だけ，ユーザとしての人間だけではないかという気がします．

林： 最後の御指摘はまさにその通りで，ユーザ・インタフェイスという名前の示す通り，ユーザしか頼るところはない．

しかし，人間をモデル化して，それに合わせてインタフェイスを作ればいいといっても，はたして人間は人間をモデル化できるのだろうかという疑問がある．それができないから，いろいろとジェネリックな仕掛けやメタ・ツールが作られるのでしょう．そうすれば，ユーザがどんなにわがままなことをいいだしても，何とか対応しうる．それでいいとはいいませんが，それしか道はないんじゃないかと思います．

たとえば，そのことが顕著に出ているのが X ウィンドゥです．何でもかんでも変えられるようにしてあって，それをがしかも簡単にできるようになっているから，みんなが文句もいわずに使うんですね．

だいたい，ユーザ・インタフェイスの基礎は，フィーリングとか感性ですから，十人十色でバラエティがあるほうが自然で，変に統一されたら気持ちが悪い（笑い）．やっぱり，自由に変えられるようにするしかないだろうと思います．

(10) ふたたび Smalltalk と Unix について

林： ところで，さきほど熊谷さんから，「Unix の上に Smalltalk の皮をかぶせたのでは，Unix にできることしか表現できない」という否定的な発言がありました．

しかし，いま Unix の上でウィンドゥ・システムを作っている人たちが大勢いて，大変なお金と時間をかけて仕事をしているわけです．ところが，青木さんのアプローチでは，わずか４～５日の労力で，ほとんどそれらと同じ成果を達成してしまった．これはスゴイことじゃないかと私は思います．

しかも，Unix の上に Smalltalk のよさまで取り入れている．いまや，単に Unix の上にウィンドゥ・システムを作るだけでは，あまりおもしろくない（笑い）．坂下さん，すみません，別に GMW がつまらないというわけじゃありません（爆笑）．ただ，青木さんのやったことは，それよりスゴイんじゃないかと私には思えるんですが.....

熊谷： どうも誤解されているようですが，私は別に，青木さんのお仕事をそんなに否定しているわけじゃありません．何を隠そそう，実はうちでも同じようなことをやっていますから（笑い）．

ただ，Smalltalk を介したときの Unix のプロセスの取り扱いなどは，ある意味でおもしろくない．なぜつまらないかといえば，Smalltalk 自体の制約が結構あって，たとえマルチ・ウィンドゥで動いたとしても，X ウィンドゥには遥かに及ばないことが，想像すればすぐわかることでしょう．

全面的に否定しているのではなくて，いろんなことをやらなくちゃいけないというのは，よくわかっているんです．私がいいたかったのは，最初に，何をどう表現するかという MMI の基本思想のほうが重要だということです．そういうコンテクストのなかでの発言だと理解してください．

岡本： 私自身は Unix が好きで，自分に１番フィットす

る MMI のような気がしています．ところが，おそらく日頃 Jstar だとか Smalltalk マシンだとかを使われている人たちは，そちらのインタフェイスのほうが気に入っているのでしょう．

音楽にたとえると，１５年ぐらい前に，ジャズ・ロックとかロック・ジャズという話があって，それが何年かたつと，フュージョンという呼び方に変わり，それぞれ分化した独自の世界を作り上げてきています．そういった意味で，これからは，いろんな世界観をうまく組み合せて，どんどん新しい試みをやって行くことが重要ではないかな，というふうに思います．

(11) まとめ

坂下： ありがとうございました．なかなかいまのはいいまとめだったと思います（笑い）．私がチェアマンとしていおうと思っていたことを，いわれてしまいました．

たしかに，ユーザ・インタフェイスについてみなさんいろいろな意見を持っておられる．その多様性を，この場での討論を通じて強烈に意識しました．いままでのユーザ・インタフェイスというのは，未熟なモデルというか，多様性があるようでないようなシステムを，どっちゃりとぶちまけて来たものであるような感じがします．

なかなか楽しいセッションでした．３人のスピーカの方々，また討論参加者のみなさん，どうもありがとうございました（拍手）．

討論をきいての感想
（参加者アンケート）

田中正則（SRA）

MMI は，やはり，ユーザ自身が作成することが望ましい．このセッションを聞き，そのための方策としては，ジェネリックな MMI を提供し，それをユーザにカスタマイズさせるのがよいと思った．その場合，ジェネリックな仕掛けをどう作るかがわかったとして，残るのはユーザ教育の問題である．どうしたら，ユーザをその気（カスタマイズをやる気）にさせることができるか？ だれか教えてほしい．

藤野晃延（FXIS）

MMI のモデルをジェネリックなものにすると，以下のような問題が起こるだろう：

(1) カスタマイズ容易性と一貫性とのトレード・オフ．

(2) カストマイズは，ユーザに自分の仕事をあらためて「意識」しなおすことを強制する．

(3) カストマイズしたあと，それがセマンティクスにおよぼす変化や正当性を，だれがどう保証するのか？

これらの問題をある種のメタモデルとして定義する試みを，だれかがやってみるべきだろう．

佐藤千明（長野県協同電算）

これから MMI は，マン・モデルまたはマン・メソッドのインタフェイスと解釈することにしよう．

吉村智香子（HST）

ビットマップとマウスがあれば，それだけでよいユーザ・インタフェイスが提供されたと誤解する風潮があるように思います．

「ユーザの好みにあった」インタフェイスを提供すればよいのか．「人間の内在するモデル」を作っていけばよいのかで，議論はかなり盛り上がりましたが，結局は，いまあるものを積み上げながら，ユーザ・インタフェイスは進化して行くということのようです．

田中慎一郎（SRA）

TRON のキーボードの話．あれがいいものかどうかは

知らないが，アイウエオ順はどうかと思う．ほんとうに普及させるつもりであれば，最初おぼえるのが大変でも人間工学的に使いやすい配置になっているほうがよいこともたしかである．

だれにも選ばれないツールは消えて行くという考え方は，危険．人間は，食わずぎらいが多いので，よいと思われるものは，それを使わせる努力なり方法を考えることも重要である．

ユーザに流されるのはよくないが，ユーザに任せる部分があるほうがよいと思う．

桜井麻里（SRA）

「材料は豊富に，組み合わせ選択はいろいろできるように」といっても，モデルは必要である．どんなに便利なカスタマイズ機能があっても，自分では何もしない人がほとんどだから，使いやすいインタフェイスの基本形も必要だ．それにしても，みんな同じことを悩んでいますね．私も悩んでいます．

三浦あさ子（SRA）

中村さん，佐原さんが話した２種類のユーザ・インタフェイス．

一口にユーザ・インタフェイスといっても，あまりに多様で，話についていくのはけっこう大変だ．私もいま，X ウィンドゥの上でユーザ・インタフェイスを作っていて，カスタマイズ機能が１つのポイントだと感じている．

コンテクストの重要性は理解できるのだが，それがうまく結びつかない．

小林明彦（東電ソフト）

聞いている分には，大変おもしろい議論でした．私も，ユーザ・インタフェイスは好みが強いと思っています．結局，声の大きい人間が力説したものがその組織での標準として取り入れられるようです．

西岡健自（横河電機）

MMI はソフトウェア本体と独立に存在しうるのか．その目的は何なのか．試行錯誤の連続とモデル不在の焦りを感じる．けれども，とてもおもしろそうだ．

森　幸一（PFU）

ユーザ・インタフェイスを考えるとき，人とコンピュータとのコミュニケーション（＝メッセージの交換）という視点が必要だと思う．メディアとしてのビットマップ＋マウスやメッセージの手段としてのグラフィックだけでは不充分である．コードとコンテクトへの考慮が今後必要だと感じる．

北野義明（KCS）

キャラクター端末＋キーボード：　JCL を編集してから，Home　キーを押し（コマンド行へカーソルが移る），"SUB"（Submit ＝ 実行）と入力してリターン・キーをたたくのは，慣れた人ならよそ見してでもできる．

メニュー＋ビットマップ＋マウス：　メニューの項目をよく選び損ねる（すぐ上や下の項目を選んでしまうことがある）．よくメニューの項目を見ていないとダメ．手先の細かな動きが苦手だと，編集のさい１文字ずれた所をポインティングしてしまう．

TSS　のダム端末や，I-Star，J-Star上でのエミュレータを使って思うこと：だれが何をするのか，どの程度....といった観点から MMI の要求を探すとキリがない．好きなものを各々が選ぶのが一番のような気がします．

古田とおる（富士通 BSC）

Mac，Smalltalkなど，マウスとウインドゥによる MMI だけでなく，キーボード，タッチペン，ジョイ・スティック，さらに今後新しいポインティング装置が登場したときの MMI も考えたいと思います．

井上尚司（ソフトバンク総研）

コンピュータを使うだけのユーザ，ソフトを作るために使うユーザをはっきり区別して MMI を考える必要が，少なくとも現在では，ある．

「いまのマウスやビットマップを前提にソフトを作る」という思想を悪いとは思わない．これに対して，「マウスでがまんできるか？」という質問は，的を得ていない．

しかし，いますぐにそのためのソフトを作れなくてもよいから，考えうる理想のハード環境を想像してみることも意味があるのでは？

寺井　孝（日立ビジネス機器）

今回の討論に参加された方々は，みなさん，それぞれ開発者の立場に立って，ユーザの感じ方，考え方を自分なりに勝手に判断しておられますが，かなり思い込みも入って

いるように感じました．さらに，使い勝手のよいものを開発するためにもユーザの考え方を積極的に聞くようにしていただければ，と思います．

小池幸徳（日本システム・サイエンス）

MMI をそのユーザの技術レベルで考えれば，明らかに開発者とエンド・ユーザが同じ MMI を持つのは，最適ではないと思う．

また，利用目的に則して考えると，ポインティング装置としてドット（ピクセル）レベルのポインティングを必要とするか否かによって，マウス，ライトペン，タッチセンサーなどいろいろな種類がある．しかし，これはらの選択の幅はまだ狭く，人間の日常生活におきかえて見るとなんとなく「かたわ」のような感じがする．音声など，もっとちがったメディアを扱える装置を考慮しなくてはならないのではないかと思う．

海尻賢二（信州大）

ユーザ・インタフェイスとそのボディとなるソフトウェアとの関係は？

鐘　友良（FXIS）

MMI の中心テーマは，熊谷さんのおっしゃった通りだと思う．人間がマシンに何を入れたいか，どういうふうにこの「入れたい」ものを簡単に表現するか，逆に，マシンが何を出せるか，どういうふうに人間がよく理解できるように表現するか，さらには，人間とマシンとの対話を人間どうしの対話のようにすべきであるか，など．

盛田政敏（KCS）

「使う人によって変わる，また，使う人の成長によって変わるインタフェイス」というアイデアは重要である．どうすれば実現できるのか，また，そのためのフレームワークは？

村川貴広（HST）

討論の内容が，発表者に対する質問に終始した．もう少し，フロアの参加者のあいだでの質疑や討論があってもよかった．もっとも，そんな環境づくりを，われわれ自身がしなければならなかったのだが.....

近藤康二（ソニー）

現実とのギャップについての議論はおもしろかった．

インタフェイスがユーザの手で自由に変えられるようなものでなければいけないという点では，多くの人の意見は一致している．では，具体的にどうしたらよいのか？　いくつかのアイデアをプロトタイプしてみる必要性を感じる．その意味で，青木さんの成果はすばらしいと思いました．

久保宏志（富士通）

ユーザ・インタフェイスつまり MMI の概念を構造的に理解することが，もっと行われてよい．考察対象の全体モデルについてのコンセンサスを持つ必要がある，といってもよいのかもしれない．

- レスポンスタイムは，MMIの特性の一部である．親しみやすさも特性の一部である．
- マウス，電子ペン等は，人がマシンに語りかけるための MMI 言語仕様の一部である．
- マルチウィンドゥは，人が知的生産を行うときに使う仕事場という MMI の一つである．
- 個人の好みにあった MMI というのは，MMI の具体化に関係する．
- Smalltalk の MVC + 依存関係は，世界を人が対象化するためのフレームワークという意味での MMI である．
- Smalltalk が，フロッピーの初期化をやってくれないという欠陥は，MMI の具体化における情報隠蔽の問題にかかわっている．
- ユーザ・インタフェイスがツール間で整合性が取れているかどうかも，MMI の特性の一部である．

以上，全体に包括されるであろう断片を思いつくままに書いた．

深瀬弘恭（アスキー）

ジェネリックなインタフェイスは，一般のユーザにとって，逆に使いにくいのではないかと思う．適応可能な部分と骨格となる部分の切り分け作業に，かなりの時間を費やしたほうが，よいのではなかろうか？

近藤隆彦（通産省）

MMIの多様性やむずかしさがよくわかりました．また，それぞれの技術者が第一線でさまざまな工夫をしておられることに，感銘いたしました．

落水浩一郎（静岡大）

おもしろかった．ランダムに，心に引っかかったキーワードを列挙すると：

・MMIの議論の範囲
・操作
・らくに操作できる
・関係のマネジメント
・視点
・オブジェクト管理

市川　寛（電算）

もう少し討論を続けたかった．最新の MMI を持つツールを試用してみたい．

田中一夫（山一コンピュータ）

MMI をどうやって作るかは非常にむずかしい．でも何となくわかったような気がする．

堀川博史（三菱電機）

リレーションの話とモデルの話が概念的で，十分理解できなかった．その他は，非常に有意義だった．

佐原　伸（SRA）

「マルチメディア・アーキテクチャのユーザ・インタフェイス，どうすべきか？」という問題を発見できた．近来まれに見るおもしろいセッションだった．

山浦恒央（日立 SK）

いろいろな意見を聞いて，自分の考えの狭さを思い知りました．今日は，ずいぶん勉強になり，少し自分が大きくなったような気がします．

林　香（SRA）

「SUMMITのアプローチでは UNIX のできることしかできない」とは，ほんとうか？　Smalltalkのよい部分と Unix との融合とは考えられないだろうか？　しかも，簡単に実現できたのなら，X ウィンドゥや GMW よりすぐれたアプローチかも....とはいいすぎかな？

言語のユーザ・インタフェイスは，シンタックスかな？

岸田孝一（SRA）

Smalltalk や Mac が好きな人たちは，これらのシステムに内在するモデルが好きなのだろうか？　私には，それらが持つユーザ・モデルやプロセス・モデルは，あまりにもナイーブ（原始的）すぎるように見える．

歌代和正（SRA）

オブジェクト間の関係がキーとなっているとの指摘は，どうもそのようである．でも，関係をメタなレベルで定義するというのはどうであろうか？

自然界は，だれかがまとめて管理しているのではなくて，すべてのオブジェクトがそれぞれ独特の性質を持ち，それにもとづいて行動している．やはり，オブジェクトが個々にもっている他のオブジェクトとの関係にもとづいてコントロールしなければ，最終的には無理なような気がする．

そこで，中野先生がいっていたウインドゥ・マネージャについて思いついたことがあるが，ウインドゥの１つ１つに性質を与えて，それぞれのウインドゥが自立的に行動するようなモデルで，ウインドゥ・マネージャを考えてみてはどうか？

たとえば，ウインドゥAは，Bについて動きたがるとか，Cはきらいなので近づいたら逃げる，といったぐあいである．当然，その時どきの状況なども影響するわけで，ウインドゥ間の力関係は，時間とともに変化して行く．

加藤　朗（東工大）

ユーザ・インタフェイスの問題があまりに広いので，議論が散失してしまいがちである．

プログラム（システム）が提供するモデルと，モデルが固定された時点での入出力の技術．この２つの議論のどちらを好むかは，あきらかに「人種」によってちがうようだ．

堀内泰輔（長野工専）

MMI を考える場合，われわれ々ソフトウェア技術者が開発作業を行う場面でのそれと，エンド・ユーザにとってのそれとは，別個に考えなければいけないと思う（たぶん，両者が一致することはないだろう）．

このセッションでは，ワークショップの性格上，前者についての話が中心であったが，エンドユーザ向けの MMI についても討論する場がほしい．

臼井義美（JIP）

よいマン・マシン・インタフェイスとは，生産性や信頼性なのか，気持ちがよく楽しいことなのか？　両方であれ

ばいいのは当然だが.....

杉田義明（SRA）

MMI を討論する場合，よくハードウェア上の問題とソフトウェア上のそれとの混乱が起こるが，今回も一部そうしたことがあった．キーボードやマウスとそのウインドゥ表現などがそうである．

私が一番興味を持っているのは，ソフトウェアの可視的な表現であり，どんな表現方法がユーザにとって大切か，そのメカニズムや定量的な評価手段はどうか，などの問題であるが，あまり議論がはずまなかった．

酒匂　寛（SRA）

マウスは，位置を決め，ボタン・イベントを起こし，移動を表現することしかしていない．より進んだ入力装置とは何か？

ウインドゥが多いと使いにくい．何とか，仕事の本質に関係ない作業を自動化できないだろうか？

セクレタリ・システム，つまり，勝手に整理される電子手帳があれば，いまのオンライン・ドキュメントの大部分はいらない．

ユーザ・インタフェイスに関しては，個々のツールの出来・不出来は，あまり問題（重要）ではないような気がする．それらを整理，統合する仕掛けと思想のほうが重要ではないだろうか？

すべての構造を平たく見せようとするところに問題がある．紙を漉くところから始めたくはない．私たちの「ペンと紙」の標準は何か？

３人の発表者の意見は：

新田：「ペンの作り方」を与える方法論

吉村：「新しいペンを作った」

青木：「ペンの作り方」

といったところか？

荻生準一（日本システム）

Unixを使い始めたばかりで，いまいち話がつかめない．参考資料のつもりで聞いています．

吉村鉄太郎（管理工学研究所）

キーボードとマウスは，本質的に異質なものであり，その２種類を併用することをユーザに強制するシステムはきらいだ．たとえば，タッチ・タイプができる人は，「すべての操作」をキーボードから手を離さずに行いたいのである．エンド・ユーザのためのインタフェイスと，プログラマのためのそれとを，ハッキリ区別して討論すべきだったと思う．

井川裕基（JIP）

インタフェイスはユーザの好みに合わせて：　ユーザのモデリング．

抽象的なモデル：　コンピュータ化しやすいが，まちっていたら，もうおしまい．「ちょっとした例外処理」がうまく表現できない．

多彩（?）なモデル：　コンピュータ化しにくい（要素が多いから）．ユーザは，自分が何をどうしてよいのか，わからない．

さて，これらの中間の位置で，どんなものを選べばいいのか？　この選択はむずかしい．

熊谷　章（PFU）

(1) マルチウインドゥとマウスの使い方

それなりにおもしろい．やはり，Mac の影響は大である．ロボット作成にも Mac が使われている．

(2) マルチウインドゥの改善案

具体策は，連動するメカニズムが必要．　br (3) MMI のモデル

リレーションの操作と表現．オブジェクトとリレーションは別物と考える．　What You See Is What It Is だと，リレーションはオブジェクトの属性になる．

(4) MMI の方向

マルチメデイアと自然指向．

小林貞幸（電算）

やはり，みんながそれぞれ，自分で使っているツール（ハード，ソフトともに）の話に終始してしまった．討論に参加するには，使った経験がなくともよいはずであるが，つい汎用機ユーザは，こういう場では弱気になってしまう．

汎用機の Cobol の世界から見た MMI についても，話したり意見を聞きたかった．

中野秀男（大阪大）

マウスはいいのか？　マウスに視点を代表させ，お絵かきをさせているいま．ポインティング装置がマウス，お絵かきはペン，耳と鼻と皮膚は？

MMI の定量評価： 各ユーザの能力の差をどう取り込めばよいか，などの問題はともかく，定量評価は，どんどん試みてほしい．ただし，プログラムの複雑性論議の二の舞にならなければよいが.....

MMI のモデル化： 正直いってよくわかりません．

MMI の多様性： 大賛成です．

新田 稔（SRA）

オブジェクトの境界： 現実世界の混乱をコンピュータの中へ持ち込むと，人間とコンピュータのどちらにアダプトすべきか？ 好きなほうを選べばよいのではないか？

行番号： 古い人たちのことも考えてあげたい．人が慣れてくれるまで待つか？ インタフェイスの基本は，叫ぶことだと思う．

御喜家貴子（YHP）

人間すなわちユーザというのは，いろいろと勝手なことをいいだすものだと思った．たとえば，マルチウインドゥができたことにより，同時にいろいろな情報を入手できて嬉しいといいながら，ウインドゥが多すぎると，整理がつかないと文句をいいだす．

あーいえばこーという感じで，とても対応しきれない．やはり，そうなってくると，何でもユーザの希望を聞いてあげるようなインタフェイスじゃなくて，ユーザの目的とモデルに合わせて，ユーザに自分のワガママを気づかせてやる必要がありそうだ．

強制的に，ワガママは許さないという態度で，マン・マシン・インタフェイスを提供すべきだ．それは，たとえば，車に乗るためには教習所で訓練しなければならず，スピードを出しすぎると「ピューピュー」と注意のブザーが鳴り出すのに似ている．

車の場合，そうした仕組みは，ユーザを守る目的や，安全のために，さらにはまたマナーを指導するために考えられている．このような配慮こそが，喜ばれるマン・マシン・インタフェイスを生みだすのではないかしら？

このレポートは，去る２月に行われたワークショップの討論記録および参加者アンケートを編集部で整理したものです．残りのセッションは次号以降に．

SEA 刊行

報告書・論文集のご案内

これまでに SEA から刊行されたワークショップ・レポートやシンポジウム・プロシーディングスのうち，次のものはまだ若干在庫が残っています．ご希望の方にお頒けしますので，事務局まで郵便または FAX でお申込み下さい．なお，定価はいずれも 3000 円（郵送料込み）です．

第１回
実践的ソフトウェア開発環境に関する集中討論
1986 年 2 月　新潟県長岡市

第２回
実践的ソフトウェア開発環境に関する集中討論
1987 年 2 月　新潟県長岡市

第４回
夏のプログラミング・ワークショップ
－プロジェクト・プランニングのケース・スタディ
1986 年 9 月　岩手県盛岡市

第５回
夏のプログラミング・ワークショップ
－バグのないソフトウェアを目指して
1987 年 9 月　岩手県盛岡市

第１回
ソフトウェア教育に関するワークショップ
－実践的ソフトウェア技術者教育を考える
1987 年 11 月　山梨県八ヶ岳高原

ソフトウェア・シンポジウム'87 論文集
基調講演："Software Process Programming"
Prof. L.Osterweil (Univ. of Colorado)
1987 年 6 月　東京虎の門パストラル

ソフトウェア・シンポジウム'88 論文集
基調講演："USC System Factory Project"
Prof. W.Scacchi (USC)
1988 年 6 月　東京虎の門パストラル

九州支部設立報告

平尾　一浩
ヒラタ・ソフトウェア・テクノロジー

1. はじめに

SEA としては，関西，横浜，長野，名古屋に続く５番目の支部として，九州支部が３月１２日（土）に設立されました．

九州の地にも SEA の支部をという声や動きは，かなり前からあったのですが，ともかく動きはじめましたので，その報告をしたいと思います．どういう形で報告しようかということでいろいろ考えたのですが，これまでの Seamail の方向を踏襲するものと，九州支部らいしいもの（世話人の趣味）を合わせて，載せてもらうことにしました．

特に，ドキュメンタリー・タッチのものは，私と藤本さん（彼女が支部設立に一番熱心でした）がかいたものを，ＨＳＴの２０歳になる福島という内職（？）で小説を書いている者にまとめてもらいました． Seamail にも軽い読物を求める会員の方がいるようなので，そんな要望に答えられるかもしれません．

2. 設立の呼びかけ

２月１９日の日付で，九州の SEA 会員を中心に設立の呼びかけをしました．

****

九州の会員も福岡：１２，熊本：１３（＋３　３月入会予定），大分：４，佐賀：１，宮崎：１，鹿児島：３，沖縄：１　の３５名（３８名）となりましたが，その活動への参加の場は東京をとる場合が多く，地理的問題で参加が難しいものとなっています．そこで，九州支部の設立を行ない，独自の活動形態を確立し，セミナー，ワークショップなどの企画，技術活動の活発化などにより，情報の中央集権を意識することなく最新情報，最先端の技術を我々のものとしていきたいと考えます．

つきましては設立のための第１回ミーティングを下記の要領で開催いたします．関心のある皆様方の積極的な参加をお待ち申し上げます．

****

こうした案内状だけで，忙しい（！？）みなさんが集まれるとは思っていないので，当然電話での勧誘もしました．そして，長野支部の例をみならってプレスにも載せました．

****

（熊本日日新聞：昭和６３年３月１０日）

九州のソフトウェア技術者

企業ワク越え交流

県内を中心に九州のソフトウェア技術者が，企業の枠を越えて参加する研究会を結成する．技術者が個人の資格で参加をし，技術と情報の交流を図るのが目的．各県単位では県やテクノポリスなどを中心に，企業同士の交流の場はあるが，技術者同士が研究会を結成するのは九州で初めて．

研究会は，６０年１２月に全国のソフトウェア技術者でつくった「ソフトウェア技術者協会（略称・ＳＥＡ，約1,200人）」の九州支部として１２日発足する．

平田機工（熊本市）の関連企業「ヒラタソフトウェアテクノロジー」の技術者らが発起人となっており，県内から１３人が参加するのをはじめとして，福岡県・大分県１８人など合わせて３１人の技術者で結成する．

１２日は発足式を兼ねた第１回会合を熊本市のヒラタビルで開くが，年１－２回の支部会合を開くほか，各県単位で定期的に会合やセミナーを開催，ソフトウェアの開発技術や情報を交換する－などの活動計画を決める．

ソフトウェア業界は情報化社会の進展を背景に急成長しているが，人手不足から深夜に及ぶ勤務を要求されることが多い．加えて，世界的な技術の進歩には目覚しいものがあるため技術者は情報不足に陥り，「企業という壁の中で取り残される不安感がある」という．

同支部は企業内技術者のほか大学，研究機関の研究者にも参加を呼びかけており，広く技術レベルの交流を図る方針．推進役の平尾一浩さん（ヒラタソフトウェアテクノロジー）は「企業の枠を超えた勉強は技術者はもちろん企業にもプラスになる」と話している．

****

(2) プログラム

設立ミーティングのプログラムと内容に関しては，この後の『熱きこころにおくる』を参考にしてください．また，参加された方々がどう受け取ったかというのも，この後の『会員の声』を読めば，わかっていただけるとおもいます．

(3) これからの運営

いろいろな心配も終ってみれば，ただの杞憂，とばかりに元気がでてきます．ともかく，当面は次にあげた体制で九州支部を運営していくことになりました．

九州支部の最大の特徴は，ほかの支部と違い８県を対象としているところにあり，それゆえ課題も多い．支部内での情報交換と全会員が集まるには，九州は広すぎて時間的問題などもあってそう簡単には集まれない．これではアクティブな活動はむずかしい．これを受けて，九州支部では九州を３地区に分け，それぞれ３名の世話人を置くことになった．

福岡，佐賀，長崎，沖縄　地区

世話人：　能見，松本，小田

＊沖縄は九州で一番飛行機の便が多い福岡が最適との判断．

熊本，大分　地区

世話人：　中島，後藤，平尾

＊ちなみに後藤さんは２４歳のＳＥ兼，主婦．

宮崎，鹿児島　地区

世話人：　植村，森，藤本

＊藤本さんは宮崎，鹿児島地区会員３名の助っ人として参加の主婦．

あとは，各地区の世話人を中心に活発な活動を行ない，年１～２回予定している支部会合の席において，ワークショップ・タイプの地区活動報告を行なうこととなった．年１～２回の支部活動では，充分な意見交換は行なえないとの声に，ネットワークを張りませんかと声が返る．ＨＳＴにホストを置き，伝言板形式でとにかくやってみようという意見もあったが，ネットワーク事情（ネットワーク情事というのも趣があっていいなぁ）として，junet の状況や，YOU-NET，QUEENS の説明後，とにかく分科会を作り検討することになった．SIGNET は既にあるので，名称を SIGQNET として，SEA 初の支部内ネットワークに向けて活動して行くことになった．この報告は，この後に世話人にしていただきます．

その他の意見としては，次のようなものがありました．

・会合の会場は各地区で持ち回りとし，ラフな形で行なう．

・各県にある情産協のようなものと協力してやっていくことはできないか．

・世話人に任期を決め，全員一度は世話人をやってみるのもいいのでは．

これらは，今後の課題ということになりました．

(4) 運営資金

運営資金は，当面は次のようにして行ないます．長野からのカンパ金３万＋第一回ミーティング夜の部からのカンパ金１万２百円は郵送代などに使い，会合での会場費などはその都度集める．

3. 会員の声

当日参加された方々から「声」を寄せていただきました（順不同）．

北川清介

新しい動き（九州の若い技術者たちの交流）が始まったことを大変うれしく思っています．こだわりや立場などをはずして，色々な話が出来，お互いに楽しい仕事（？）が出来ることを期待しております．

後藤芳美

大分という小さな地域でソフト開発を行っていると，どうしてもグローバルに物を見る機会を持てないというのが現状です．そういった状態の中で，SEA は大きな情報源でした．しかし，結局 SEAMAIL を読むだけで，受け身の一年を過ごして来ました．又その内容もレベルが高く，読んでも興味をもっていたのは会員の声でした．九州支部が設立したからには，もっと受け身ではない参加をしたいと思います．

田浦寿敏

たいへんおもしろそうなのでぜひ成功させたいと思います．ただ，個人的な意見としてあまりシステマティックにする必要は無いと思います．

鶴田真理子

SEAの会員に入らせていただきます.

永田和也

SEAの地方支部第５番目として九州支部が設立されたわけだが，中央の後追い，他の地方支部のまねをしたりして，九州支部独特の色を失うようなことはしたくない.

中島泰彦

九州のソフトウェア技術者が集まり何かいいこと話しています.

能見　巧

やっとスタートになりました．さぼっていてもうしわけない．熱い熱い九州支部にしたい.

飯富章博

技術者の方々の意見が聞ける有意義な会だと思う．どんどん参加して，利用させて頂きたいと思います．知人にも紹介します.

上妻健一郎

どんどん交流の輪を拡げて欲しい.

小田七生

今回の九州支部設立を非常に喜んでいます．常々，地方にいるため，業界情報の不足に悩んでいただけに，九州支部設立に期待します．会合に参加して，他の人々の活発な意見を聞くことが出来，自分自身をリフレッシュできました．これから，支部運営のために頑張りたいと思います.

松本初美

いろいろな人の集まりで，かつ，どこかで結びついている．ストレスの解消の場であり，我々ソフトウェアにたずさわる者の Communication の場であり，我々の仕事をもっと分かり易く，世間に認知させたく，努力したいと思ってます．今日の会合で，その様な活動の芽が出てきた事を共に慶びます．今日はどうもありがとう.

上川清士

東京との差があるとの事でしたが，九州の地域性を利点として拡大していってほしいと思う.

竹内正幸

第一回目にしては盛り上がったと思います．今後の方針，進め方，次回の議題等についても決まっており，今後が楽しみで期待できると思います．私個人としてもネットワークに興味があるので，色々聞きたいと思っています.

久保宏志

「太いが短い」の生命を燃えつきさせることなく，「細くとも長い」生命の維持を期待します．横浜に在住して東京に勤務している会員からのメッセージです.

今別府芳暢

キックオフのためにご苦労願いましたＨＳＴの方々ごくろう様でした．このような会合がほしかった．今後の活動に協力して大きく育てていきたい（ほしい）.

桑原たか子

なかなかおもしろそうな集まりだと思います．次回の集まりにも期待したいと思います．しかし，時間的なものが（予定通りに行くのは難しいですが）ちょっと気になります．又，もっとリラックスした場所というか，雰囲気をつくればいいですね（次第にそうなるでしょうが）．何か手伝うことがあればやりたいと思います.

杉野　緑

いろんな興味深い内容で，とても刺激になりました．ソフトウェアに対して，問題意識をみなさんもたれていて，ただなんとなく仕事をこなしている自分を少しはずかしく思いました．これからの活動を期待します.

植村正伸

非常に有意義な一日でした．今まで，他の支部や本部のワークショップやセミナー参加していたが，九州内で，こういうことができてうれしく思います.

森カンナ

興味深く，おもしろい集まりだと思います．企業を越えたつながりを目的としているので，会社単位の集まりでなく，もっと，個人のつながりを作ってもらいたいと思います.

門岡春雄

九州にソフトウェアの首都をおきたい！！

富永伸哉

さあ，今から何をやろうかな！

吉村智香子

今度は，一杯飲みながらやりましょう．ＢＯＦが大事です．

内田京子

様々な分野の方々（企業，学校関係など）が多くてたいへん勉強になりました．しかし，個人的な希望としては，もっと大学関係の方などが参加されると，もっと，刺激されると思うので，そのような方にも参加していただきたいと思いました．何度か話はあったと思いますが，SEAというのは，ソフトウェア技術者の集まりなので，企業とか企業名というカベがなくなって技術的な交流が深まればとても素晴らしいと思います．今回は，初めてということもあって，”肩書き　＋　名前”という親交がありましたが，今後，名前同士で話ができるとよいと思います．

中島敬彦

うまい料理を作る為によい材料を集めようと思った．

藤本良子

九州支部設立バンザーーイ！　主婦のストレスをSEAで解消するわ！！

村川貴広

SEAでも，雑用係（環境整備）をやりたいな！！

古閑幸一

今，九州が熱いと言われたい．

野口幸一

種々の業種を持つ人が同じソフトウェアという共通したツールをいかに生かすかということで会を持ったことは，非常に有意義なことだと思います．これからは，いかに発展させていくかということに焦られてくると考えます．相互にメリットがあること．問題点を提起し合い，皆で考える会であって欲しい．

松村幸一

大変面白い「会」になりそうなので今後が楽しみです．ただ大分のため水前寺公園発１６：５０しかありませんので，時間的に問題があります．ＨＳＴの関係者各位，大変お世話になりました．

田中則嘉

自己紹介において少しばかりのプログラマの方々の問題点が見うけられた．が，具体的なものが意見として出される事をこれからも期待しております．

平尾一浩

正直に言います．今日は，緊張をやわらげるため昼に焼酎のんでしまいました．ごめんなさい．

”すっごい男がいたもんだぁ〜〜

SEAでばったり出会ったら

西南の役もちだしたぁ〜〜

ビールを回せっ　朝まで飲もぉ〜〜〜

あんたが１番

SEAは２番〜

はあっ，どんどん！！”

田村

今日は，SEAの方の貴重な声を聞くことができましてありがとうございました．なごやかで楽しい雰囲気でよかったとおもいます．

4. 九州地区ネットワーク（SIGQNET）発足について

去る３月１２日のSEA九州支部設立総会で，九州地区のＢＢＳを作ろうという提案があり，現在発足に向けてＨＳＴの鏡沼和友氏と私を中心にして準備活動を行なっています．

実現方法として，junetを使う，オリジナルのBBSを作る等の，提案がなされましたが，誰にも負担をかけない方法として，既存のBBSを使う方針で検討がなされました．とりあえず熊本市のNTTがホスト局をしているYOU-NETというBBSに間借りすることを考えており，既に内諾は得てあります．他県の方には，電話代の面でご迷惑をおかけすることになりますが，会員の熱気がさめないうちに，発言の場，連絡の場を作っておくことが大事と考えています．

九州地区会員の方に配布するアンケートを現在作成して

おりますが，この文が読まれる頃にはもうお手元に届いていると思います．どうぞご意見をお送りください．ボード開設の際には，SEAMAIL誌上でお知らせしますので，どうぞ九州地区以外の方も気軽にアクセスしてください．

皆さんのご理解とご支援をお願いします．

上妻健一郎
電子応用機械技術研究所

5. 事後プレス

事前プレスがあれば，事後プレスもありました．

****

（電波新聞：昭和６３年３月１６日）

ソフト技術者交流組織

九州でも支部設立

ソフトウェア技術者および研究者が，社会組織の壁を超えて経験や技術を交流しあう，ソフトウェア技術者協会（ＳＥＡ）九州支部設立のための第１回会合は１２日，熊本市内ヒラタビルに関係者約５０人が参加して開催，九州支部の発足を決めるとともに，世話人の選出，年間活動計画，ネットワーク構築などについて話し合った．

会合では，通産省の情報処理振興課近藤隆彦課長が「２０００年の推計では，少なくとも４０万人以上のソフトウェア技術者の不足が予測されている．このような状況の中，ＳＥＡの活動には非常に関心があり，注目している」とあいさつ．続いて同協会東京本部代表幹事の鈴木弘氏（熊本構造計画研究所長）が，東京一点中心主義はやがて崩れるーなど「情報の中央集中から分散へ」の題で講演した．

続いて長崎，沖縄を除く九州６県から集まった約５０人が自己紹介した後，参加者全員で今後の活動計画を協議(1)九州支部連絡先をヒラタソフトウェア・テクノロジ（ＨＳＴ，平尾一浩代表）とする．(2)九州を３地区に区分，それぞれ３人の世話人を選出．(3)会費を取らないボランティア団体とするーなどを決定，今後の支部活動，ネットワークの構築問題については，世話人で検討，次回の会合を５月中旬にする，などを決定した．

SEAは，ソフトウェアハウス，コンピュータメーカ，計算センタ，エンドユーザ，大学，研究所など，それぞれ異なった環境に置かれているソフトウェア技術者または研究者が個人の資格で参加，技術と情報の交流を図る場として６０年１２月発足．機関誌の発行，支部および研究分科会の運営，セミナ・ワークショップなどイベントの開催，内外の関係諸団体との交流などを主な活動としている．

九州地区からも約４０人が参加しているが，活動の場は東京が中心になるため，個人の資格では地理適に参加が難しく，九州独自に活動する目的で九州支部を設立したもので，横浜，関西，長野，名古屋に続き全国５番目の支部結成となった．平尾一浩氏（ヒラタソフトウェア・テクノロジー）は「企業の枠を超えた勉強は，技術者はもちろん企業にもプラスになる」と話している．

****

熱きこころにおくる

原作：**平尾　一浩・藤本　良子**

作　：**福島　光則**

☆1988 年 3月 12日　ー熊本ー

その日，平尾一浩は浅い夜の眠りから目覚めた．熊本の春は早い．

「おはようございます」

「あっ，おはよう...」

いつのも時間に出社したものの，朝の挨拶もどことなく元気がない．いつもは熊本の荒技師と呼ばれている彼なのだが．（とうとう今日か...　うまくいくだろうか....）．いいしれぬ不安がこころを占めてくる．

「やることはやったが....」

煙草に火をつけながら，数キロ先に煙る熊本城の勇姿をみつめながら独り言ちた，「さしずめ，援軍待ちの篭城主というところか」

☆Welcom ！！　（平尾一浩）

「ようこそ　SEA Kyushuu start-up meeting へ！

本日は私のような弱輩者の呼びかけに対して，これほどたくさんのかたにお集まりいただき，誠に感謝しています．

九州支部設立の動きは，昨年秋，ＨＳＴの藤本が皆様のところへお手紙をだしたのがきっかけとなっています．しかし，返事の手紙，電話などはひとつもなく，「九州の方達は何も問題意識がないんじゃないだろうか！？」などと話しておりました．そこで，気の短い私としましては，「えぇ〜い，かけてしまえ！」と電話をかけてしまい，今日こうして皆様に集まっていただきました．はじめ疑ってい

ました皆様の問題意識の大きさや，また，なんとかしたいというその気持ちが，今日の会合でだされることを期待します．

現在 SEA の活動は，どうしても東京，大阪を中心としたものが多く，われわれ九州の人間にとって，地理的条件からどうしても受け身の参加しかできないのが現状です．しかしわれわれは，なにも中央ばかり見続ける必要はないと思います．身近なところでも，興味ある研究や仕事をされてる方々がたくさんいらっしゃいます．われわれ独自で，中央に遅れ，負けることなく，セミナーやワークショップなどを企画し，大学や，研究機関，企業の人間がそれらの組織の壁を取りはらった形で，多分野にわたる話や議論をして，最新の情報，最先端の技術をわれわれのものにしていくことが可能であり，またそうしてこそ Happy になることができるんじゃないかと思います．ちなみに，いま注目されてますファジー理論の山川先生（熊本大学）も SEA にお誘いしたところ，先生のほうでも興味を持たれたようで，資料を下さいといわれてますので，近いうちに御返事がいただけると思います．

さてここで会員状況をみていただきたいのですが，ご覧のように九州全県あわせますと，３５名の会員が登録されています．福岡－１２，熊本－１３と争っていますが，今日すでに熊本からは６名のかたが入会申込書をだされ，今日中には２０名を越えそうです．福岡の方にはすみませんが，熊本が勝ったな，と内心喜んでいます．ということで，鹿児島，大分，宮崎のかたも頑張ってください．

今回の九州支部設立の話を聞かれた通産省のほうから，是非参加したいとの連絡が一昨日ありまして，情報処理振興課の近藤課長さんがいらっしゃってます．近藤さんには後ほどご挨拶をお願いしてあります．

さて，今日これからのプログラムですが，なんといっても今後の活動に対するミーティングが，今日のメインとなっていますので，よろしくお願いいたします．

ところで，みなさま気付かれたと思いますが，部屋のうしろにコップを手でだしてセットするという，最新鋭のコーヒー自動販売機が用意してあります．今日はオープンにしていますのでご自由にお飲みください．このコーヒー代は，１月に行なわれました長野でのワークショップの際，有志の方々のビールもう一杯という誘惑を押さえ，カンパしていただいて集まった援助金を使わさせていただいています（有志のみなさんありがとうございました）．それでは次のプログラムに移りたいと思います...」

☆独白

ちっちっちっちっち.....

目覚まし時計の律儀で世話しない音をＢＧＭに，一本のジュースの空き缶を前にして，この原稿を書いている．ペンが止まる．あのときのジュースがそもそも間違いのもとだった．

「よお，おごるから，ちょっとジュースでも飲みにいこうぜ」

いつもはけして，そんなことはいわない平尾一浩２６歳既婚が，わたしを誘ったのは２日前のことだ．その数分ほど前に，彼が「Semail の原稿は.....」という言葉をもらしていたのを偶然にも耳にしていた．とっさに防御体制にはいった．

「え，いや，今仕事が....」

力のある人間にはかなわない．山法師のあだ名は伊達には付けられていない．数分後，ジュースの自動販売機の前に立たされていた．

「さっ，どれがいい」彼の不気味な笑顔におびえていたものの，私の右手は１００円のボタンへ伸びていた．

あのときはっきりと断ってさえおけば．いくら悔やんでも悔やみきれない．たかだかジュース１本のために，SEA の会員でもないのに，なぁんでこんな目にあわなければならないのだ．

仕方がない，次の資料は....と，今回の支部設立のきっかけを作った藤本良子主婦による「手紙」，か．

☆手紙（藤本良子）

ことは，1986年 6月のソフトウェア・シンポジウムに始まりました．

夜の情報交換パーティで，SEA の会員の人たちに，長野支部設立の影響もあって，

「九州支部つくってくださいよ」っていったら，

「SEA は，ボランティアだから，自分たちで動かないとダメ」なんて冷たく（！？）突き放されてしまった．

そうでした，そうでした．世話人って人がいるんですよね．次の日のチュートリアルで，TRON のことでもいろんな刺激を受けて，地方でもなにかやりたいって意気込みはできた．

熊本に帰って，「さあっ」と思ってはいたものの，プロジェクトに追われ，一日は，あっという間に一週間，そして一ヶ月．そこへ，福岡の能見さんが，

「SEA 会員の熊谷さんにＨＳＴに行ってみたらどうか」

といわれたそうで，遠路はるばるやってきた．話を聞いてみると，福岡では１人１人と会って，SEA福岡支部設立にがんばっているらしい．今度，ミーティングも開くらしい．実をいうと私，その前の夏の盛岡のワークショップで，熊谷さんのグループにいて風土って言葉を学んだのでした．そのときの熱い想いもよみがえり，「遅れてはいけない，熊本支部をつくろー」と初心を思い出したのでした．でも，考えてみると，福岡地区とはかなり違って，熊本の会員はほとんどＨＳＴばかりだし．これでは，SEAの趣旨でもある，さまざまな職場で働く人々が技術的・文化的交流を行なう，がむずかしい．うーん．よし，どうせつくるんだったら九州支部だ．なにから始めよう．福岡支部設立の様子をみながらなんとかしなくっちゃ．

しかし，その意気込みもむなしく，時間だけがたって季節は夏が過ぎ，気づいたら秋も深まってきていた．そんなプロジェクトも終わりに近い一日，席換えもあって気分も一新したことだし，SEAの会員の人たちに九州支部設立についてのお手紙でもと思って書きはじめた．

（拝啓，って硬い言葉うまく使えないのよねえ．突然だし，顔も知らない人々になんて書いたらいいのかな？　）なんて思いながら．

ＳＥＡ会員の皆様へ

拝啓，初冬の候　益々ご清祥の段，お慶び申し上げます．

SEAの九州地区会員も３０名をこえ，福岡地区では独自に支部結成を行いつつある中，熊本でも何か支部を結成して，分科会なるものを行いたいと，日々，思っていました．ところが，ここ熊本においては，１２人の会員のなか，１０名ほどが弊社，（ヒラタ・ソフトウエア・テクノロジー）所属ということで，どうしても内部構成となってしまいます．そのため，SEAの目的である”技術者の，既存の社会組織の壁をこえた相互の”交流”という面で，効果をあげることができにくくなります．そこで，熊本だけでなく九州全体で，支部を結成し，相互の交流を行いたいと思っています．

つきましては，会員の皆様のご意見，ご要望を承りたく，ここにお便りした次第です．率直なご意見，ご要望をお待ちしております．

敬具

そして全く同じ内容をＨＳＴのＳＥＡ会員にもメイル．後は，返事を待つだけ．そして，わたしは最後の出張に東京にでかけた．２週間なが－いホテル暮らしもなんのその．帰るころには，返事がた－くさん届いているだろうって楽しみにして帰ってきました．仕事も一段落．思うところあって今年いっぱいで退職するけど，SEAの会員は更新したし，退職までにはなんとか形にして，残りはＨＳＴの誰かに押しつけてって考えながら，SEA九州支部設立に力を入れる準備をしたのです．さあ，後はSEA九州支部設立にむけてがんばるだけ．

結果は電子メイルが２通きました．まあ，とっても明るいメイルで，”九州支部設立後は，年に４回は温泉でセミナーやワークショップをして親睦をふかめましょう”って書いてきたのは，なにを隠そうＨさんです．でも，返事の手紙は一通もありませんでした．まだ，２週間しかたってないしね．みなさん，じっくり考えていらっしゃるのだろう．まっ，いっか．

・・（また，２週間経過）・・

まだ，一ヶ月しかたってないしね．手紙の文面が悪かったのかしら，突然すぎて失礼だったかしら．まっ，いっか．年末で，みなさん忙しいんだろうしなあ．いまごろ暇なのは，わたしだけかな？　って思いつつ，少しづつ退職する日は近づいてきたのです．そして，とうとう．来年はなんとかなるかなあ，今度は往復ハガキでだしてみようかなあ，なんて考えながら，心はＳＥＡ九州支部設立より専業主婦の方へと傾きつつ，87年は暮れて，88年になってしまったのでした．

☆独白

そういうことだったのか．こういうきっかけから，ＳＥＡ九州支部設立の会合，ひいてはこの原稿書きの苦しみへとつながってきたのだな．うむむ．とりあえず初心をまっとうしたということか．それでいつに似合わず優しい雰囲気だったのか．なるほどなるほど．

まあそれはいいとして，人の文のつなぎつなぎを書くなんて，初めてのことだから，書くことも試行錯誤になってしまう．人はいつでも初体験．

まあ，今までの資料に目を通してみると，なかなかSEAというのもおもしろそうではないか．ボランティアでやってる，なんてところがかっこいい．　生臭い金や権威の話しになってきたら，いっきに興醒めするところだが，どうもそれらしき話はないみたいであるし，なかなかよい

のではないだろうか.

☆1988 年某日

「返事来ました？　おや，何むくれてんです？　なにっ，全然来てないの！　二ヶ月になろうとしてるのに！　そらでけん！　そっじゃ，あんまなさけなか．返事いっちょでけんごたるやつばかりじゃ九州はすくわれんばい．みんななんいっちょ考えとらんとだろか？　おっが，どぎゃんかせんといかん．藤本さん，続きば俺にさしてくれんね」

☆独白

いま，わたしの心は無である．

いくら元は他人が書いたものとはいえ，このまま載せてしまってよいのだろうか．確かに「どんなことを書いてもいいから」ということで引受けたのだが．いくらなんでもこれはまずい．わたしの良心までが疑われてしまう．せめて訳をつけることにしよう．

☆再び 1988 年某日

平尾一浩は前々から気になっていた，SEA 九州支部設立の動きを確かめるべく，そのいいだしの藤本の所まで足を運んだ．

「返事来ました？　おや，何むくれてんです？　なにっ，全然来てないの！　二ヶ月になろうとしてるのに！　それはいけないことだなあ．ふっ，そんなことでは，あまりにもなさけないではありませんか．返事ひとつできないような人ばかりでは，九州はすくわれないですよ．みんな何ひとつ問題意識はないのだろうか．私が何とかしなければ．藤本さん，続きを私にやらせてくださいませんか．

☆独白

これでよし，と．これならきれいで完璧な標準語だ．さすがに”あれが噂のゴーストライタ”と噂されている（陰口を叩かれる）ことはある．

まさか「みんななんいっちょ考えとらんとだろか？」を，困ったなあ．みんななにひとつ考えていないのじゃないだろうか？　やはりこの世で信頼できるのは自分と金だけだな，はっはっは」と，正直に訳しては，むざむざ自分の身に危険がおよぶことになる．さっさと次に移ってと....

☆電話（平尾一浩）

熊本の会員に電話をかけはじめたのが１月の中旬．

手順としては，「九州支部設立に向けて動いている，ぜひ参加して欲しい」という内容と，「第１回会合を開きたいが，日時に対する希望は？　平日？　休日？」などを聞いて，より多くの人が参加できる日時を選ぼうと，とにかく電話しまくった．しかし，みなさんとても忙しらしく，なかなかつかまらない．こんなとき電子メイルが使えたら，と思ったことが随分あった．

退職されて連絡が取れない２名の方を除いて，どうにかひととおりかけ終えて，やっとのことで日時を選択した．次はその日時を知らせて確認を取らなければいけない．しかし，私もこればっかりやっているわけにもいかない．本来の仕事であるプロジェクトの客先立会は迫るし，バグ対応もしなくちゃいけない．おまけに，論文も書かなければならない．結局すべての電話をかけ終わったのは，一ヶ月半後の２月も終わりの頃となっていた．はじめに作った名簿は，真黒になるほど個人個人の情報が書き込まれていた．

何人かの人は，返事を前日まで待ってくれとのこと．それでも，３０人位の方が出席できるとの返事をもらい，少しは安心した．一番恐れていたのは，熊本だけの会合になってしまうのではということだったが，それがなくなっただけ気が楽になった．

これで準備が終ったわけでは当然ない．会場の選択や会員以外の方へ配る資料作りなど，やることは予想より結構多かった．東京の本部のボランティアの苦労が実感として判ったのが，妙な感激としてこころに残った．ボランティア活動は，一体感を伴った熱きころこをたぎらせてくれる．

会場の選択では，いろんな所へ問い合わせをおこなったが，どこも高い！　どこかいい場所はないものか？　ある，ある，熊本城がみえ，市内が一望でき，阿蘇さえもよくみえる．そう，会社の会議室があるじゃないか．そこで会社との交渉がはじまる．なんと，意外にもと早くに OK がでた．「ただで使える」初めての会合だから，できたら参加費無料でやりたいと思っていただけに嬉しさひとしおだった．よし，もう一踏ん張り.....

「平尾さん電話ですよ」

「はい，平尾ですが.... えっ，これなくなった...」電話なんか嫌いだ！！

☆独白

電話が悪いわけではないと判っていても，最後の言葉がでるところに，平尾の全てが表れているような気もする．

☆前夜 （平尾一浩）

会場のセッティングは終わった．スピーチの依頼もちゃんとやった．そうそう，幹事の鈴木さんに明日のスピーチの題名をしらせなきゃ．

roff－fax，と．

はぁ～もうこんな時間だ．資料のコピーはあすの朝一番にやろう．でも，どうもプログラムがしっくりこない．企業の壁をとって，といいながらちょっと堅すぎる感じがする．SEAの持っているラフな感じでやりたいが，

九州支部活動計画

↓

九州支部活動計画だよ

これじゃ顰蹙買いそうだしなー．

「そんなときは英語でやってみたら」と煙の向こうから声．さすが，困った時の Mr. Ikadai.

「ついでに司会も全部英語でやっちゃったら」

「odamari nasai！」それこそ，バグがあるとわかってプログラムを納品にいくようなものでしょうが．で，結果出来上がったプログラムは以下の通り．

SEA Kyushuu Start-Up Meeting

(preliminary program)

1:00 Welcome(by Mr.hirao)

1:10 What's SEA and What can we expext from? (by Mr.Suzuki)

1:25 Greeting from MITI (by Mr.Kondo)

1:30 Speach I (by Mr.Suzuki)

Towards the distributed paradigm

2:10 coffee break

2:20 Discussing SEA Kyushuu

3:30 coffee break

4:00 Speach II (by Mr.Ikadai)

"American Programmer"

4:50 close

なるほど，堅くなくてなかなかいいじゃないか．どうして漢字を使うと，堅い感じを与えてしまうのだろうか？やっぱり日本は肩書き社会なんだな，とへんに納得してしまう自分がおかしい．

準備がひととおり終ったと思ったら，今度は急に極度の緊張感に襲われる．みんなちゃんと集まってくれるだろうか…． うまく進行できるだろうか…． 会議が盛り上がるだろうか…． 設立後は….

とにかく焼酎飲んで帰ろう．

☆独白

わたしは焼酎よりも，ビールのほうが好きである．だからどうだというのではない．のどがかわいただとかおなかがすいただとか，そんなことをいうきはこれっぽっちもないのである．それにしてはすこしかんじがすくなくなってきてる．かんじょうがおもてにでやすいたちはかわらない．

さて，次は肝心の当日の部分．はてさていったいどうなることやら．

☆さて （平尾一浩）

さて当日．（はぁ，はぁ，う，はぁ，はぁ…）

コピーは終わった．会場案内も手配した．駐車場の案内も手配した．受付も頼んだ．（はぁ，はぁ，う，はぁ，はぁ…）

引き出しに手が延びる．手には徹夜用にとっておいた焼酎が握られている．う～ぷっ．知らず知らずのうちに２杯目へと… 心地よい酔いが私の体を包み始めたとき，「２人みえられました」と受付からの知らせ．まずは挨拶．ところが，開始予定時間の５分前になってもほかのかたがこない．

なぜ，どうして….よぎる不吉な思い．そこへ突然の人の群れ．よかった．

種々の理由により，３０分遅れの 1:30．九州支部はスタートした．

☆Welcome Kyusyuu SEA

代表幹事の鈴木氏より， SEAとはどのようなもので，われわれは SEAになにを期待できるのか，という内容で SEAの紹介が行なわれた．

続いて，通産省情報処理振興課の近藤課長より挨拶．「九州の技術者の方々の熱い意気込みを感じ参加した．2000年には， 97万人ものソフトウェア技術者が不足するといわれている．ゆくゆくは自動プログラミングが出てくるだろう．しかし，その自動化の中心となるものはやはりソフトであり，ソフトは人間の課題としていつまでものこり，ますます問題は深刻化してくる．そのような中で，技術者が企業の壁をとった結合をおこなうことはたいへん興味あるところで，今は通産省としては直接お手伝いできないが，このような SEAの活動に期待する」

続いて代表幹事の鈴木氏より，「情報の集中から分散へ」と題したスピーチが行なわれた．「情報の一点中心主義はやがて崩れる．物理的（土地，通勤）問題から，地方での開発が注目されてきている．そこで，地域的，広域的な分散開発環境と開発管理をスムーズにおこなうことで，地方でも技術者が満足のいく仕事ができる体制を，ネットワークなどを活用し作り上げていく必要がある」

質疑では某会員から”Σ批判”がでて場内をわかせた．

すべて順調に進み，いよいよ活動形態のミーティングに入った．やっぱりはじまった，自己紹介，自己紹介．初めの予想に反して話すわ話すわ．日頃の不満から自分の夢まで．しかし共通していたことは，地方でもいい仕事がしたい．新しい情報をいちはやくつかみたい．それらを可能にするために，SEA九州支部をアクティブな支部にしたい．そしてすばらしい仲間を持ちたい．

☆九州はいま，熱くなってきている

討議は終わった．すでに予定から１時間３０分の遅れだ．最後に予定していたＨＳＴ筏井氏のスピーチ "American Programmer"ができない．しかし，そこは筏井氏．さっと前にでるがはやいか，話すがはやいか，電光石火．ささっと要約を話し，コメントをエドワード氏に送りたいと岸田さんがいわれてますので，それぞれお配りした資料を読まれたあと，コメントを岸田さんまで送ってくださいと業務連絡．さすがＢ型，素早い機転！（かくゆう私も同型）．

すべて終わった．かくして，今夜の熊本組１７人は，”馬刺し，馬刺し”と呟きながら，夜の街へと消えていく．

☆1988年　深夜　－熊本－

宴の後，一浩は夜の街に一人たたずんでいた．早春の夜気が疲れた身体に気持ちがいい．冷たいコンクリートの壁にもたれかかると，心地よい疲労感が全身を包んでいく．思わずもれる溜息も，いまはもうない．静かな微笑が，一浩を一事をなしとげた男の顔にしている．

ついさきほどまで互いの意見を交わし，熱論を交わしあった新たな友たちは，それぞれの宿へと帰っていった．またね．また会おう．．．．再会の言葉は疲れをいやし，こころにしみ渡る．

「またね，か」

一浩は思い返し，また笑みが顔にひろがっていく．静けさのなか，一浩のもらした言葉が，夜の街に溶けこんでいった．

「SEA you again ．．．．」

☆独白

実際には参加しなかったため，当日の様子はよく分からないのだが，この資料を見るに限ってはなかなか盛り上がったみたいだ．そういえば，ミーティングを終えた後のSEAの会員の面々が，えらく高揚してたような気がしないでもない．なにはともあれ，無事に終わったことはいいことだ．今後の活動についてなどなかなか具体的な所まで決まっているみたいではあるし，竜頭蛇尾にならなそうだ．今後に期待がもてるのかな．今度のミーティングの予定が５月か．個人的に参加してみようかなという気がおきなくもない．うむ．そう書いたら本当にそんな気がしてきた．よし，今度はのぞくだけでものぞいてみることしよう，と．

平尾氏の結びをBGMにして，わたしも，

SEA you again．．．．

☆結び　（平尾一浩）

さてさて，とにかく動きだした九州支部．あとはＢ型特有の飽きっぽい性格を克服し，なんとか軌道にのせていきたいと思っている．現段階では東京，関西，そして他支部との厚さ，力の差，は歴然としている．ひとつだけ，たったひとつだけ勝てるのは，”変なやつ”の多さだけだろう．その変なやつら約５０人が九州でなにかやろうとしている．各支部，分科会，そしてすべてのSEA会員の方々のご支援，アドバイスをお願いします．

また，九州支部設立に際し，多大なるご支援をいただいた常任幹事の岸田さん，事務局の中島さんに心から感謝し，そして今後も九州支部に対して熱い期待を持って見守ってくださることをお願いして，終わりにしたいと思います．最後に，とんでもない原稿をまとめ上げてくれた，福島君の労にたいして，紙面を借りて一言いわせていただきます．忙しい中，ご苦労さまでした．どうもありがとう．

Hackerに関する考察

浜野 剛至
ソフトウェア・リサーチ・アソシエイツ

1．Hackerの定義

コンピュータとしばらく付き合っていますと，「私はハッカーでありません」とか，「あなたはハッカーだ！」とかいったやりとりを，会話のなかにみかけますし，ハッカーに関する本もいくつか書店でみかけますが，ハッカーとは一体どんな存在なのでしょうか？ ハッカーと聞いただけでいろいろな感じを受けるのではないでしょうか？

なんかうっとうしい存在だ，現代社会はハッカー抜きでは語れない，ハッカーは反社会的である，常識人ではない，根暗だ，お人好しだ，運動神経が0だ，ポルノ映画には興味がないがアニメーションで鼻血をだす，スニーカーを履いている，子供がいればそれは女の子だとか，定時に会社に来る事は稀である，いわれていることを数え上げるときりがないくらい，いろいろな断片的特徴が報告されています．そのような愛すべきハッカーについて，気楽にあるいは真剣に考えてみたいと思います（ちなみに私はハッカーではありません）．

(1) MITの周辺(by S.Levy's "Hackers")

Hackerという表現は，1950年代のMITの鉄道模型クラブあたりから使われるようになった．配線を巧みに繋ぎ変える，鍵のかかったものはかたっぱしからはずそうと試みるといった，好奇心の塊のようなパーソナリティを持っている．しかしながら運動神経等は人並以下と，アンバランスな感じの学生たち．Hackの本来の意味は，樵などが斧で木を叩き切るところからきている．

(2) UNIXの世界(News from Junet)

ハッカーについての定義が，国内外からネット・ワークのニュースに流れていた．アメリカから流れてきたものにおもしろいものがあったので，一部を紹介しますが，それによると， Hackerにいたるまでには，いろいろな呼称のついたレベルが存在するようだ．まず，beginner，novice，user，knowledgeable user，expert，hacker．さらにその上があって， guru，wizardとなっています．その記事による hackerの特徴として次ぎのことがあげられてます．

- uses sed and awk with comfort
- uses undodumented feature of vi
- writes C code with "cat >" and compiles with "!cc"
- uses adb because she doesn't trust source debuggers
- figured out how environment variables are propagated
- writes her own nroff macros to supplement the standard ones
- writes Bourne shell scripts
- installs bug fixes from the net
- uses egrep because she timed it

wizardになると名前に Dennis とか Bill とか Ken とかあるなんていうのは笑ってしまいました（くわしくはbit増刊「最新 UNIX」に掲載されましたので，興味のあるかたはどうぞ）．

また別の記事には，ハッカーを次ぎのように定義しています．

- 一般のユーザが，必要最小限ですませたがるのに対し，プログラミング・システムの細部を学び，（プログラミングの）能力を伸ばすことを楽しむ人．
- 楽しんでプログラムを作る人．または，プログラミングの理論をこねまわすよりも，プログラムを作る方が楽しい人．
- Hack valueが正しく認識できる人．
- プログラミングが非常に早い人．
- ある特定のプログラムのエキスパート，またはその上で頻繁に作業を行う人．
- あちこち覗きまわることによって，情報を発見しようとする悪意の，または単純な好奇心の覗き見屋．転じて "password hacker"，"network hacker"．

また，別の記事では，

- HAPPY HACKING：別れの挨拶（じゃあね．またね．）

・HOW'S HACKING？: ハッカー同士の親しげな挨拶.
・HACK HACK: 深い意味はないが，親しい間で使うコメント．しばしば，一般的な別れの挨拶に用いられる．

というハッカー言葉を紹介しています．どれも愉快な記事ですし，思わず納得してしまいます．どうやらハッカーの元祖というのは，アメリカのコンピュータを触っている連中から生まれてきたようなもののようです．ヒッピー発祥の地 UCB のあたりなんかは，ハッカーがうじゃうじゃいるような感じですね．

2．Hacker 倫理(by S.Levy's "HACKERS")

"HACKERS" という本を読んで感じることは，ハッカーというのは建設的な無政府主義というのか，コスモポリタリズムというか，そんなものを感じることが出来ます．いくつか抜き出してみますが，私個人としては最後の二つの項目は好きです．

・コンピュータへのアクセス，加えて，何であれ，世界の機能の仕方について，教えてくれるものへのアクセスは，無制限かつ全面的でなければならない．実地体験の要求を決して拒んではならない．
・情報はすべて自由に利用できなければならない．
・権威を信用するな（反中央集権を進めよう）．
・Hacker は，成績，年齢，地位のような，まやかしの基準ではなく，そのハッキングによって判断されなければならない．
・芸術や美をコンピュータで作り出すことは可能である．
・コンピュータは人生を良いほうに変えうる．

3．マス・コミの功罪

(1) Hacker という言葉のイメージの固定化・側面的な解釈

なんといっても「高エネルギー研究所」のアクセス事件に対しての報道のしかたは，納得が行くものではなかった．この報道のあとに JUNET で，この事件，あるいは報道に対する活発な議論が起こっていました．その多くはハッカーたちの嘆きにも聞こえました．

「筑波高エネ研に西独のハッカー侵入」と書けば，世間の人は，「ハッカー＝悪人」という理解をしてしまうのではないか？ 「私はハッカーです」と表明したら，ただちに逮捕されるということがおこったり，魔女狩りならぬハッカー狩りが起こったらどうしてくれるねん．ネットワークの維持だって，みんなのボランティアでなりたってんのだぞ．こういった誤解を拭うことにマスコミの人はたちは関心があるのか？ 私はとにかく，日本のマスコミのセンセーショナリズムを煽るような報道は好きになれません．といっても，もともとマスコミはイェロウ・ジャーナリズムから起こってきたもので，それに振りまわされるわれわれ受け手の問題かもしれませんが．．．．どこかで飛行機事故があったとして，その損害賠償とか責任とかにマスコミはよく言及しますが，マスコミが侵した事故とか，責任というのはきちんととられているのでしょうか？

こういったマスコミの功罪については，もっと考えなくてはいけない．たとえば，情報処理業界は暗いというイメージを定着させたのはどこのだれだ，とか．

(2) ネットワークを利用した犯罪と Hacker との相関はあるのか？

銀行のオンラインなどから，大量の現金が瞬時にして引き出されたりする事件は記憶に新しいですが，これらのほとんどが内部からのものが多く，悪いことの仕方が時代と共に変わってきたということではないでしょうか？ もちろんネットワークを通して悪いことをする人もいるでしょうが，これをハッカーと呼んでは困りますよね．

3．セキュリティの問題

ハッカーとセキュリティとはあまりり相関がない話とは思いますが，セキュリティを守るも破るもハッカーさんはその気になればなんでも致しますというところでしょうか．

(1) トロイの木馬

破る側のテクニックとしていろいろあるとは思いますが，そのなかで有名なものとして，トロイの木馬というので，パスワードなどを盗むのに古典的な方法（トリック）があります．社内で，私もやられたことがあります．最初は何が起こっているのか全く見当がつかなかったのですが，今になってみるとなかなか楽しいものです．だまされたことのない人は，一度だまされてみて下さい．そのトリックにはいろんなレベルがあるので結構おもしろい．

(2) 守るための方法

(a) パスワードの選択（推測が容易では何の為のパスワード？）

銀行のキャシュ・ディスペンサーの暗証番号は，生年月

日が多いとききます．そうでなかったとしても，その個人の属性に関わるものが多いようです．定期的にかえればより安全でしょうが，その分忘れやすい，とかいって端末の所にメモでも張り付けたりすると，なんのためのパスワードか分からなくなってしまうし．．

(b) コールバック方式

電話回線を通じて，コンピュータにアクセスすることを制限する方法で，アメリカの大学・企業ではこの方式にしている所が増えているらしい．具体的には，電話でこっちからアクセスしようとすれば，コンピューターがその人の所に電話をかけ直して，接続をしてくれるもの．日本では電話料金などが高くてまだまだ無理だとは思うが，今後増えそうな方法ではある．

(c) スタンド・アロン（ネットワークに繋がない）

とにかく，機密を保持しなければならないマシンは，外界との接触を一切断ち切った状態で使えば一応大丈夫なはずである．つまりマシンルームには鍵がかかっており，常に監視され，マシンそのものもネットワークには繋がっておらず，端末も同じく管理されているといった運用状況．逆に，ネット・ワークなどにいったんつなげれば，ある程度事故は覚悟すべきものではないか．システムもいろいろなパニックを通じて成長と変化をし続けるものでしょう．

(d) システムの弱点を明らかにする（オープンなシステム）

内緒，内緒としてしまうほうが，かえってなにかやんちゃしてやろう，という動機づけを促す場合が多いのではないか？ また，UNIX のマニュアルの中にあるように，セキュリティはこういう所が弱いと，はっきり書いてしまうほうが，かえって安全ではないか？

(e) 暗号化

古くて新しいアプローチ．このへんの話は大阪大学の中野先生にお話して頂くとよいかと思います．

(f) いじりたがりはシステムの味方にしてしまう

なにかとシステムにちょっかいを出したがる人間は，システム側の人間にしてしまうとなにかと便利である．少なくともうちわで悪戯をするような人間はいなくなると思います．

いずれにしろ，技術の進歩と共に新たな道具が生活に入り込んでくるわけですが，問題は，それらを使って生きている人間の心持ち次第ではないでしょうか．なんだって制約と可能性の両面を持っていますものね．

４．GNU について

(1) 最後の真の Hacker R.M. Stallman(by S.Levy)

外見も，することもハッカーの手本のような感じのする彼．仕草は大阪弁でいうと「けったいな」です．「最後の真のハッカーの Stallman 」なんていうのは， Levy が勝手にいっているだけだと，あっさりいってのけるのはさすがに Stallman という感じです．こういうような感じの人間が，次々と生まれているというのがアメリカの怖さというか，強さの感じがしますが，いかがなものでしょうか？

(2) GNU Manifeato

ぜひ一読をすすめます．彼の主張はもっともだと肯づかされることが多い文章です．コピーは自由に行ってよい，しかしいかなる所有も許されない．彼はこれを，コピー・フリーと呼んでいます．彼は信念のもとに独自の運動を行っているわけですが，これは私が説明できるものではないので，彼の Manifesto を読んで欲しいと思います．

(3) Free Software Fundation

ボランティア・ベースの組織で数名のスタッフで構成されており，パートタイムで仕事をしているメンバーもいるようです．彼は自分の作ったソフトウェアで億万長者になろうと思えばなれたと思います．でもそうしなかったのは，彼なりに危機的なものを感じることがあったのだろうと思います． HACKERS の中にも彼の焦りのような描写がありました．

しかし，アメリカのソフトウェア工学をやっている人たちには，彼の思想を危険なものだという評価があるのも事実です．

５．Hacker へ愛を込めて

・良心に忠実であること

山を超すプログラムがあっても，愛が無ければ無に等しい（ホンマかいな）．

・人間が好きであること

コンピュータの向こうには生身の人間がいる．その人達の存在を忘れていてはだめなのではないか？

・日本的な企業文化との闘い

日本的ということに関しては，さまざまな見解や議論がなされていますが，そもそも日本人的なんてことをいうこと自体，とっても日本的な感じがします．そういっ

た議論がどうであろうと，とにかく日本の社会はホモジニアスであるわけで，その中では同じであることとか，全体のなんやらかんやらといった話になりがちで，議論といっても，相手とは違うということを明確にしていくプロセスとは違う．ヘテロジニアスなことに対して，なんとなく恐れがあるのではないか．そういった風土は効率的にはなるだろうが，創造的なものが生まれにくいのではないか．

いわゆる日本的な企業，組織はハッカーにとって生きやすく，ハッカーたちの自己実現が可能なところなのか？　学校の先生，ハッカーを育成しましょう．上司のみなさん，ハッカーの芽をその人にみつけたなら摘まないで下さい．経営者のみなさん，ハッカーを大事にしましょう．

６．社内では？

私が現在所属をしている会社での状況を少しみてみますと．．

・勝手気侭

これはいい意味での自由があります．

・セキュリティ

無関心では無いが，比較的イージーなようです．どうなっているのでしょうか？　あまり深刻に考えていないのではないかと思います．

・いいだしっぺがやる

「これがあったほうが便利じゃない？」「じゃー，お前やれよ」

「こんなツールがあるよ」「ううー，シブイ！！」

こんな会話がよく聞こえます．

・Hackする事に理解を示す上司，人々

ハッカーだった（現在も？）役員，上司がいる．

・やっぱりUNIXはHacker好み

ハッカーたちは，UNIXが一番良いとは考えていないが，その本質的な良さを認めている．またUNIXのスタイルというか，カルチャーはハッカーが作り上げていったものだと思います．こんなことを書くと，UNIXを導入することをためらってはいけません．UNIXは自分のところの技術者のレベルを示してくれるバロメータにもなります．

６．エピローグ

(1) セキュリティの意識の高まり

セキュリティを高めようとするなら，まずセキュリティに問題はないかと，自分のところのシステムをチェックすればよい．無関心・無責任が一番よろしくない．それなりに投資も必要になってくるので，真剣にやると大変なものになる．

(2) 管理者の脱次第

御承知のように，高性能なハードウェアが低価格で市場に出現しています．近い将来，かなり大規模なネットワークが構築されていくことになると思いますが，従来のような中央集中型のマシン環境であれば，状況を把握するのは比較的容易だったわけですが，何十台，何百台というネットワーク上のマシンを管理するのはとてもとても大変．

そこで何らかのアイデアが必要になってくるわけですが，このあたりの所が，今後の管理，更には分散環境を構築する上での要点になるように思えます．また，個人が１台のワークステーションを占有していれば，当然そのマシンはその個人が使用者であり管理者でもあるので，ワークステーションを手に入れた人は，管理者になることを拒否できないことになると思います．

(3) ゲマイントシャフト or ゲゼルシャフト？？

社会学でしばしば使われる概念ですが，われわれが生きているこの社会・企業・組織というのは，ゲマイントシャフト（共同体社会）なのか，あるいはゲセルシャフト（利益社会）なのか，その中間なのか．

ソフトウェアの開発においてあなたは工房派エンジニアなのか，あるいは工場派エンジニアなのかという話が，Seamailのセミナー・ウィークの報告のなかにありましたが（Seamail Vol.2 No.1），工房派・工場派という分類と，この概念とは，対応しているように思います．ハッカーがその本質を生かせるには，一体どのような組織が望ましいのでしょうか？

７．おわりに

昨年５月に大阪でおこなわれた関西Forum（そこでのパネル討論がVol.2 No.8に掲載されています）で思いつくままに話した内容を，書き直したものです．まとまりのない話になりました．なにか御意見・御質問のある方は，編集部にお寄せください．

不純人工知能礼賛

柳田　俊一
日本コンピュータ・システム

1．はじめに

この題はいったいなんだ！？　と思われるかたが多いと思います．その説明から．

最近，ドレイフュスという合衆国の哲学者が，「純粋人工知能批判」という，イマニュエル・カントも飛び上がって驚いた，という批判の書物を著し，いちはやくわが国にも翻訳され，書店の AI コーナーに並んでいますから，ご覧になった人もいると思います．関西 SEA の AI 分科会でも，臼井氏と中野先生のお二人に，内容を紹介していただきました．かくいう私は今，AI トゥールのベンダー側に立って仕事をしています．仕事上からも，個人的感想からも，Mr. ドレイフュスの意見には真っ向から反対です．しかし，そんなことはどうでもよろしい．「AI 賛成－反対」の議論には，私は全く興味がありません．AI トゥールを使ってシステムを作ったら，「これのどこが AI なのですか？」，「COBOL でも作れなかったのですか？」という意見がかならずでます．COBOL でも作れたかも知れません．AI トゥールを使って作るのに要した工数の５～６倍，かければね．

中国のある人の言葉に「黒い猫でも白い猫でも，ネズミをとる猫はよい猫だ」というのがあるそうです．私の考え（考えなんて大層なものじゃない，カンカクとでもいったほうがよい）も，それに近いところにあります．

「純粋人工知能」について私はうんぬんしたくありませんし，その資格もありません．これから私が述べさせて頂こうと思っているのは，不純極まりない，カッコウの悪い，ブサイクな人工知能へのアプローチの１方法です．

2．不純人工知能とは？

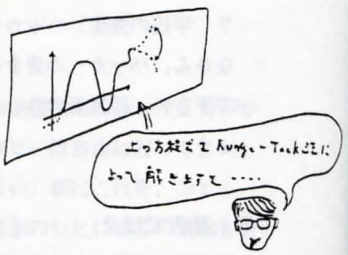
ここで，大胆な予想を申し上げましょう．

今後２～３年は，
不純人工知能の全盛期となるであろう

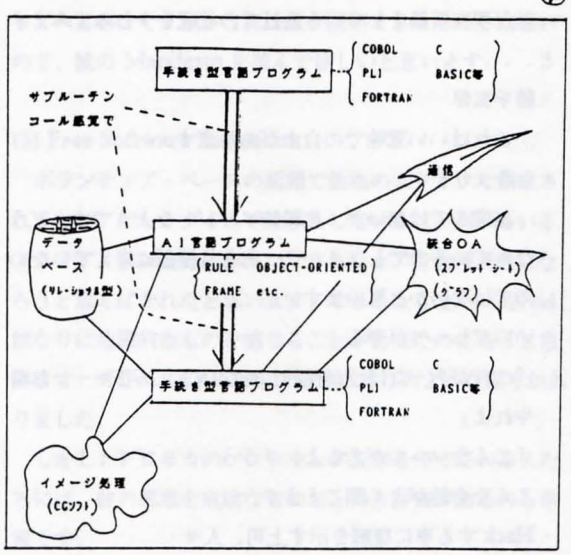
これは，私の勝手な予想ではありません．IBM の動きをみて下さい．不純人工知能の定義を申し上げましょう．

不純人工知能方程式：

不純人工知能　＝
手続き型言語　＋　ＡＩ　＋　データベース
＋　統合ＯＡ　＋　イメージ処理　＋　ｅｔｃ



その概念図を書きますならば：



不純人工知能の大黒柱は図の中央を上から下に力強く走る２重線です．まず，手続き型言語で書いたプログラムから，AI 言語で書いたプログラムをサブルーチンコールの感覚で呼び出さなければなりません．次に，AI 言語で書いたプログラムからも，手続き型言語で書いたプログラムをサブルーチンコールの感覚で呼び出さなければなりません．

この２方向のリンクのないものを「不純人工知能」と呼ぶことは，たとえ，もう一枚舌が生えてきたとしても許されることではありません．さらに，これらのプログラムのどこからでも，ＤＢ，統合ＯＡ等のモデュールを手足のご

とく，自由自在に呼び出すことが出来なければなりません．頭だけいくら立派でも，手足なしでは何程のことが出来えましょうや？

3．なぜ，このようなワイザツ極まるものが必要なのか？

そもそも，人間は不純な世界に傾いていくものなのだ，それが自然の姿なのだ，と一発で片付けたいところですが，それでは納得していただけないでしょうね．

純潔　AI　世界と古典的言語世界（例えば，COBOL，FORTRAN）の比較をしてみましょう．

純潔 AI は：

・トロい！

・メモリーをメチャ食いする

・繰り返し機構の実現方法が不自然（バックトラックとか，再帰とか）．

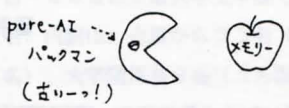
この方が自然なんだ，という人もいますが

・非決定的制御（つまり，何をしでかすかわからん，糸のきれた凧みたい）．

だからすばらしいのだ，という議論ももちろん成り立つ．

・処理が柔軟

手続き型の言語みたいにガチガチに組んでないから



古典的言語世界は：

ハシコい！

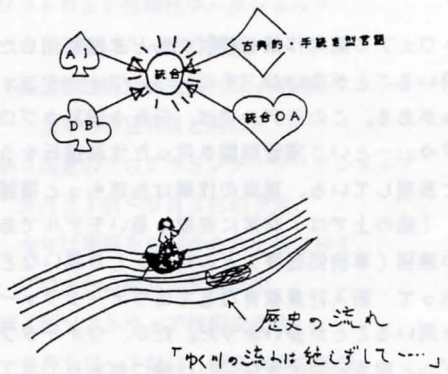
・つつましやか（爪に火をともしてメモリー節約）

・繰り返しは実に自然（と思うのは私だけかな？）

・決定的制御（安心してみとれる？）

・処理が硬直的（キメ打ちしてるからなぁ）

このように，両者，一長一短あり，優劣つけがたし，という状況です．すると，人間って欲張りですから，両方のいいところを取って，なんて考える人がすぐでてきますね．モチはモチ屋．AI の得意な問題は AI 言語で，他の手続き型で，ヒューマン・インターフェースは統合ＯＡで，データ蓄積はＤＢで，そして，全てをリンクして統合，というように進んでいくのは，実に自然な歴史の流れでは，とお思いになりませんか？



4．そんな AI トゥールあるのか？

現在のところ，このようなコンセプトにはもう１歩，というのは，私の知る限りでは，「GURU」というパソコン AI トゥールだけです．「大創玄」というのでは，この方向を目指していますが，発売が１９８８年９月まで延びてしまいました．残念です．

IBMは既に PL/I と統合できる推論エンジンを，たしか，もう発表したはずです．COBOL（このラテン語のごとき，永遠なる崇高，神聖なる言語に栄あれ！）との統合の実現も IBM のことだから，時間の問題でしょう．国産メインフレーム・メーカ各社からも，そのうち雪崩のごとく，同種のものがでてくるものと，予想しています．

5．終わりに

余談ですが，３年後には，イメージと音声がヒューマン・インタフェイスの主流になると思います．推論結果は音声の説明付きで，アニメーション，または静止画像で表示されることになるでしょう．推論途中の質問も，質問内容がイメージにより表示され，音声による説明が行われた後，ユーザからも音声で答える，といった形になるでしょう（文字をエンエンと画面に連ねても，だれも読んでくれなくなるのではないか）．３年後にそなえて，Seamail の媒体を何にするか，ということも，そろそろ考えておいたほうがいいのではないか，という気もしています．

ウォータフォールモデルの変形表現

細野広水
（株）エム・ケー・シー

ソフトウェアの開発作業を検討するとき抽象化したモデルを用いることが多いが、その一つにウォータフォールモデルがある。このモデルでは、分析⇨設計⇨プログラミング⇨……という滞留期間を伴った作業進行をうまく線形に表現している。現実の作業は当然もっと複雑なのだが、（紙の上では）非常に理解し易いモデルであり、また他の業務（事務処理等）との対応をとり易いなどの長所もあって、新入社員教育などではウォータフォールモデルを用いることが多いようだ。だが、ウォータフォールモデルと現実との大きなズレは幾つかあり、各工程でのフィードバックの状況や、設計とテストの関連などをうまく表現する他のモデルが求められる。そしてフラワーモデル、スパイダーモデルといった提案がなされることになる。しかしここでは、難く考えず、ウォータフォールモデルの「変形表現」で対応を試みる。

ソフトウェアの開発作業を企業レベルで考えるときの本質は「管理された作業」であろう。そして、その管理を確実なものにするために、チェックポイントの明確化と適切なフィードバックが必要となる。つまり、

◆一連の作業の細分
⇨多数のチェックポイント
⇨短いパスでのフィードバック
⇨「管理された作業」

ということになろう。ここまでについては通常のウォータフォールモデルの表現で何等支障はないが、ここに作業の複雑さ、抽象度、ＷＨＡＴとＨＯＷなどの概念、テストの位置づけといったものを付加して適切な表現を求めた場合、どうも不足感がある。そこで、ちょっと手を加えて出来たのが次の図である。

ここでは、従来、線形に表現されていたものをＵ字形に変形してある。その結果、要求定義等の人間臭い世界（ＷＨＡＴの世界と言えるかな）とプログラミング等の機械的な世界（ＨＯＷの世界と言えるかな）との関連、設計段階とテスト段階を対応させるドキュメントの扱いなどが旨く表現されたような気がする。この表現方法の妥当性の評価や、それに関する工学的な研究などという立派なことは私ごときに出きるはずがなく、また全く考えてもいないが、例えば、新入社員に「機械化の波」……プログラマー不要論とまでは言わないが……を理解させる場合などに重宝しており、なかなか使える図であるという気がするので、ここに提案してみた。

ついでに言わせてもらえば、場面によってはＹ字型（プログラミングのレベルの完全機械化）や、逆Ｎ型（一過性のもので、運用開始後は改善などをいっさい考えずに単純に運用するバカチョンシステム）なども有効かも知れない。

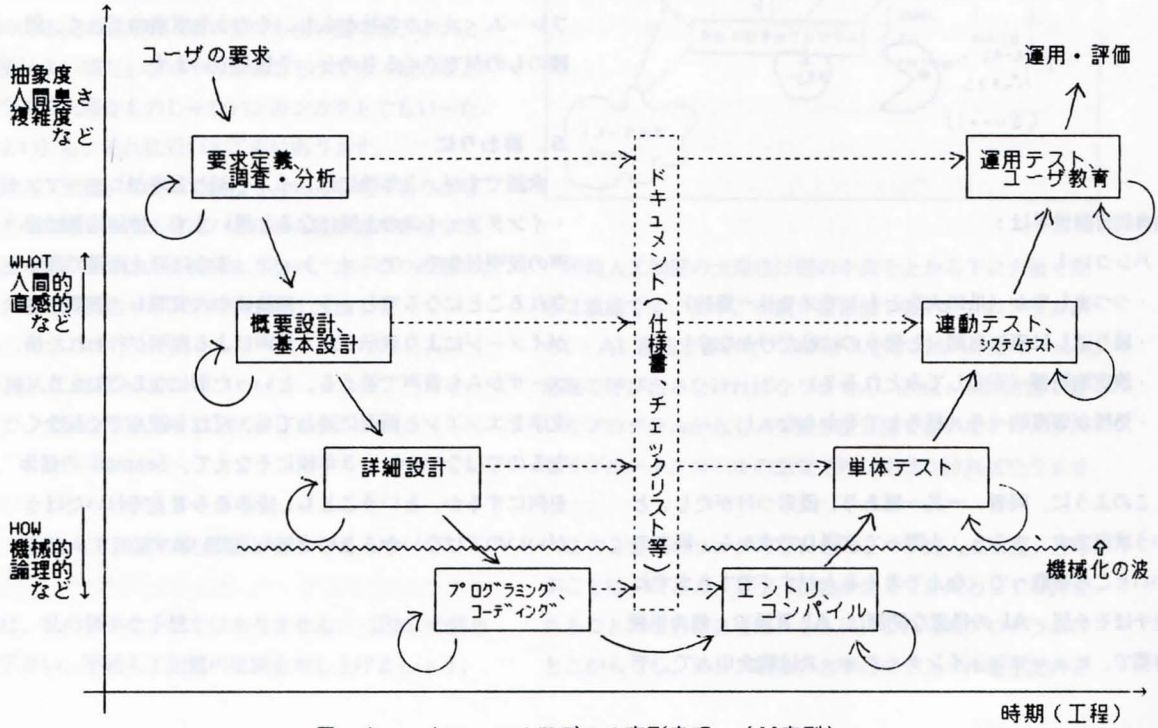


図　ウォータフォールモデルの変形表現　（Ｕ字型）

幹事会・総会・その他の報告

４月の幹事会

４月の幹事会は，４月２７日（水）に東京・機械振興会館で，６２年度幹事と６３年度幹事候補者の双方が参加して行なわれた．出席者は以下の通り：青島，大木，岡田，片山，川北，岸田，熊谷，芝原，杉田，武田，田中，野村（敏），野村（行），針谷，深瀬，藤野，藤本，松原，吉村，林［シンポジウム委］，中島［事務局］．

1. イベント報告

(1) 10th ICSE 研修ツアー

４月１０日～１６日にシンガポールにおける第１０回ICSE への SEA 研修ツアーは，５３名の参加があった（報告は次号で）．会議全体の参加者は約６００名．内シンガポール以外からの参加者は約３００名で，日本からの参加は，米国（１００名強）に次いで多く，約８０名であった．次回の米国 Pittsbergh での第１１回会議も，SEA は日本を代表する後援団体として協力することとなった．論文の締切りは９月１５日，詳細は７３ページ参照．

(2) 第３回ソフトウェア技術交流会

４月１９日～２０日に大阪大学工学部で行なわれた．参加者は５１名．内訳は，企業から３３名（SEA　会員３０名，一般３名），大学関係者９名（うち阪大９名）．

(3) 第７回ソフトウェア信頼性シンポジウム

４月２１日～２２日に大阪ガーデンパレスで行なわれた．参加者は４９名．

(4) ４月フォーラム

４月２５日に東京・機械振興会館で，10th ICSE の報告を兼ねて行なわれた．参加者は３８名．

2. これからのイベント（予定，計画，その他）

(1) 総会およびシンポジウム

目前にせまった５月１０日夜の第３回総会（同日午後には月例フォーラム），および６月初旬のソフトウェア・シンポジウム８８について準備状況および参加申込みの中間報告がなされた．

(2) 夏以降に計画されているイベント

以下の各イベントについて，おおよその計画が企画に関係している幹事から説明された：

・ソフトウェア信頼性ワークショップ
　８月下旬に八ヶ岳で開催．
　高信頼性研究会主催，SEA 後援．
　（去年の伊豆ＷＳと同じ）
・第６回夏のプログラミング・ワークショップ
　８月３１日～９月３日の予定．
　今年は場所を盛岡から八ヶ岳に移す．
　プログラム委員会を編成中．
・第４回ソフトウェア技術交流会
　９月６日～７日に，徳島大学で開催．
　プログラムはパネルを中心にする．
・第２回教育ワークショップ
　９月２９日～１０月１日に東北地方で．
・第２回日中ソフトウェア・シンポジウム
　現在１５名の申込みがある．
・第８回ソフトウェア信頼性シンポジウム
　１０月２８日～２９日・
　いつもの通り，大阪ガーデンパレスで開催．

(3) 今後の月例フォーラム

６月以降の月例フォーラムについての討論が行われ，開催期日とテーマがおよそ以下のように決まった：

・６月：Hyper Media（２７日）
・７月：テーマ未定（２９日に横浜で開催）
・８月：夏休み
・９月：セミナー・ウィークと重なるので休み
・１０月：漢字コードと日本語処理（２１日）
・１１月：　教育（１６日）

3. その他

(1) 常任委員長選出

常任委員長の分担は，一応次のように決まった：

総務：久保　　セミナー：杉田
会誌：藤野　　技術研究：臼井

(2) 支部・分科会について

各分科会や支部のこれからの活動について，次回以降に報告を受け，もし何か問題があれば検討することになった．

新年度総会

SEA 昭和６３年度総会は，５月９日夜，機械振興会館（東京）で開催された．出席者は３１名，他に委任状１１７通．

1. 昨年度決算

昨年度の決算案（辻・吉村両監事の監査済）が，次のように事務局から提示され，満場一致で承認された．

収支計算書

自昭和６２年４月１日
至昭和６３年３月３１日

支出の部	
人件費	506,813
事務所費	4,510,791
印刷費	3,556,700
通信費	2,290,611
会議費	362,000
広報費	129,926
消耗品費	232,140
雑　費	200,658
当期収支差額	5,814,837
合　計	17,604,476
収入の部	
新入会費	3,540,000
更新会費	5,495,000
賛助会費	1,800,000
寄付金	4,656,332
雑収入	2,113,144
合　計	17,604,476

この収支計算書は，公益法人会計基準に示されているような収支ベースのものではなく，損益ベースによって作成してある（企業会計の損益計算書に該当するものである）．

貸借対照表

昭和６３年３月３１日現在

資産の部	
流動資産	
現預金	3,612,696
未収入金	2,215,000
固定資産	
敷　金	2,400,000
合　計	8,227,696
負債及び正味財産の部	
流動負債	
未払金	725,100
借入金	4,000,000
正味財産	
正味財産	3,502,596
（当期収支差額	5,814,837）
合　計	8,227,696

2. 新年度役員

次の役員候補案が鈴木代表幹事から提案され，満場一致で承認された．

代表幹事　岸田孝一
常任幹事　臼井義美　久保宏志　熊谷章
　　佐藤千明　藤野晃延　松原友夫
　　吉村鉄太郎
幹　事　青島茂　稲田博　岡田正志
　　落水浩一郎　片山禎昭　川北秀夫
　　杉田義明　鈴木弘　武田知久
　　田中慎一郎　長井剛一郎　中園順三
　　中野秀男　西尾出　野村敏次
　　野村行憲　針谷明　深瀬弘恭
　　藤本司郎　村井進　盛田政敏
会計監事　辻淳二　吉村成弘

3. 今年度予算

引き続き，次のような新年度予算案が提案され，やはり満場一致で承認された．

予算書

自昭和６３年４月１日

至昭和６４年３月３１日

支出の部	
人件費	1,200,000
事務所費	4,200,000
印刷費	4,800,000
通信費	3,600,000
会議費	500,000
広報費	300,000
消耗品費	300,000
雑　費	200,000
当期収支差額	2,600,000
合　計	17,700,000
収入の部	
新入会費	4,200,000
更新会費	5,000,000
賛助会費	2,000,000
寄付金	4,500,000
雑収入	2,000,000
合　計	17,700,000

4. その他

以上で正式の議題を終わり，続いて，新常任幹事に選ばれた各氏からそれぞれ挨拶があって，総会を終了した．

なお，総会に先立ち，同日午後同じ会場で月例フォーラム「CASE ツールの現状と動向」が開催され，約４０名が参加して，熱心な討論が行われた．

また，総会終了後，機械振興会館６回のレストラン・パークヒルで，第４回環境ワークショップ（実践的開発環境に関する集中討論）の企画ミーティングが有志を集めて行われた．

５月の幹事会

５月の幹事会は，５月２６日（木）夜，機械振興会館で開催された．出席者は以下の通り：岸田，熊谷，佐藤，松原，青島，岡田，片山，川北，杉田，田中，中園，針谷，深瀬，盛田，高田［シンポジウム委］，中島［事務局］．

1. 報告

５月１０日の総会および月例フォーラムについての報告があった．フォーラムの収支は引き続き好調で，約５万円の収益が本体に寄付された．

2. ６月のイベント

高田委員長から，シンポジウムの最終準備状況の報告があった．参加申込みは採算ラインを突破，目標の３００人まであと一息ということで，各自さらに追い込みの勧誘を行なうことを申し合わせた．

また，６月フォーラム「ハイパーメディア」は，テーマの選定がタイムリーだったせいか，これまでの月例フォーラムのなかでは参加予約の出足が最高であることが，事務局から報告された．

3. 夏以降のイベント企画

次の各イベントの企画の詳細が説明された（ただし＊印の２つについては，担当幹事欠席のため概要のみ）：

・７月フォーラム(*)
・信頼性ワークショップ
・第６回若手の会(*)
・第４回技術交流会
・第２回教育ワークショップ
・第８回信頼性シンポジウム
・第４回実践的開発環境に関する集中討論

これらの各イベントについては，論文または参加者募集のチラシをシンポジウムの会場で配布し，また全会員に郵送することが決まった（会員への配布はすでに終わっているが，念のため，本誌７１ページ以降に再録してある）．

また，熊谷幹事から，来年１月に盛岡で，中年管理者のためのワークショップ（若手に対抗する会？）を行いたいという提案があり，基本的に了承された．詳細は後日あらためて討論する．

4. 秋のセミナー・ウィーク

９月下旬に開催の予定．現在会場を機械振興会館に予約してあるが，月～水の３日間なので，別の場所（青年会議

所会館）に移し，火～木に変更を考えている旨，事務局から報告があった．プログラムの詳細は，来月の幹事会で検討することになった．

5. 分科会および支部の活動状況

出席の各幹事から，それぞれが関係している分科会や支部の最近の活動状況や今後の計画について，説明があった．

現在いくつかの分科会の活動が低迷し，休眠状態にあるが，その立て直しをどうするかについては，来月以降あらためて検討することになった．

ソフトウェア・シンポジウム'88

当協会の主要年中行事であるソフトウェア・シンポジウム'８８は，６月８～９日に東京・虎の門パストラルで開催された．有料参加者は約３００名（うち SEA および JISA 会員はそれぞれ 45%，一般 10%），他に招待者を含めて合計３５０人が参加した．シンポジウムの前日に機械振興会館で行われた併設チュートリアルは，４セッションあわせて１１０名が受講した．

今回のプログラムは，米国から W. Scacchi (USC)，中国から張然の両氏を招待講演者としてお迎えし，また，一般応募論文の発表の他に，８大学・１研究所からのツール展示およびプレゼンテーションをあわせて，３トラック併列の形で構成された．例年に比較して論文の粒もそろい，内容的にもかなり充実したものであった．

最優秀論文賞（旧ソフト協の会長であられた故・服部正氏を記念して１９８３年から制定された）には，大阪大学の松本健一・井上克郎・菊野亨・鳥居宏次共同執筆による「ソフトウェアの残存エラー数推定モデルの定量的比較」（発表者は松本健一氏）が選ばれた．大学関係者の受賞は初めてである．ちなみに，これまでの受賞者は：

８３年　臼井義美（JIP）
８４年　佐原　伸（NCC）
８５年　伊藤春男（リコーシステム開発）
８６年　西山　総（MRI）および渡辺益秀（NES）
８７年　大形和宏（NES）

シンポジウムの論文集は，まだ若干の在庫があるので，購入を希望される方は事務局まで申し込まれたい．なお，基調講演および２つのパネル討論の模様はテープに収録した．いずれ本誌上でレポートできるだろう．

日中シンポジウム

１０月に予定されている第２回日中ソフトウェア・シンポジウムの参加者打ち合わせ会（顔合わせと最初の企画ミーティング）が，ソフトウェア・シンポジウム'８８第２日目の昼食休憩時間に，来日中の張然先生（上海・復旦大学）を交えて，虎の門パストラルで開かれた．日本側２０名の参加者を４名ずつ５つの討論グループに分けることで，ほぼ意見の一致を見た．

とりあえず，全員がそれぞれのグループ討論テーマに関する簡単なポジション・ステートメントを書き，別件で６月末に上海を訪問する岸田・杉田両幹事が，中国側と最終的な打ち合わせをすることになった．

また，シンポジウム冒頭の全体セッションにおける日本側の基調報告者は，落水浩一郎（静岡大）・野村敏次（JIP）の両氏に決まった．

６～７月のフォーラム

６月２７日夜に開催が予定されている月例フォーラム「Hyper-Text と hyper-Media」は，とりあげたテーマがきわめてホットな話題であるせいか，参加申込みは，これまでの月例フォーラムで最高の６２人に達した（６月２１日現在）．

なお，引き続いて同じテーマでの討論（Part II）を考えている７月の横浜ブルーナイト・フォーラムも，すでに申込みが２０人を越えている．参加希望者はお早めに事務局宛 FAX を！

SEA Forum July '88 in Yokohama

Hyper-Text をめぐっての論議　Part II

— Visual Programming のあるべき姿を求めて —

または

— Hyper Card ≧ 5GL！ —

主催：ソフトウェア技術者協会

7月の Forum は，会場を横浜に移して，YDOC の主催により開催します．もともとは船を借り切って洋上 Forum にする積りでしたが，費用その他の関係で，通常の YDOC 例会の形式を踏襲します．討論テーマは，6月の Forum の続きで，Mac の Hyper Card によるさまざまなアプリケーション・システムのデモンストレーションを交えて，これからのソフトウェア開発のイメージを探ることにしました．

おそらく，議論は予定の時間内には終わらないでしょうが，YDOC では，話の続きを中華街のどこかで深夜まで行なうことが恒例になっています．それに参加し，ミナト・ヨコハマで1夜を過ごしたいという方は，6月末までにお申込みいただければ，1泊5千円〜1万円のシングル・ルームをどこかのホテルに予約できるだろうと，世話人の藤野さんがいっています（横浜の週末のホテルは，通常どこも満員です）．希望者はお早めにどうぞ．

Let's talk together in the 'Blue Light' Yokohama!

開　催　要　領

(1) 日　時：　昭和 63年 7月 29日（金）　19:00 〜 20:30

(2) 会　場：　神奈川県労働福祉センター（JR 関内駅徒歩 4分，横浜球場並び）

(3) コーディネータ：　藤野　晃延（FXIS）

パネリスト（予定）：　青　島　茂（FXIS）　熊　谷　章（PFU）

佐　原　伸（SRA）　杉　原　誠（日本シュルンベルジュ）

野　中　哲（日本電気）

(4) 参加費：　3000円（SEA会員）　6000円（一般）　1000円（学生）

(5) 申し込み及び参加方法：　下記の申込書にご記入の上，郵便または FAX でお送り下さい．折り返し受講ハガキをお送りします．受講料は，**当日現金で受付にてお支払下さい**．ただし，定員（40名）になり次第，締め切らせていただきますので，あしからずご了承下さい．申込後のキャンセルはお断わりします．

申込先：　〒 102　東京都千代田区隼町 2 − 12 藤和半蔵門コープビル 505

ソフトウェア技術者協会・7月フォーラム係

TEL 03 − 234 − 9455, FAX 03 − 234 − 9454

SEA Forum July '88 in Yokohama **参加申込書**（申込日：＿＿月＿＿日）

氏名（ふりがな）：＿＿＿＿＿＿＿＿＿＿＿＿＿＿（　　）

種別（いずれかにチェック）：☐ *SEA*会員（*NO.*＿＿＿＿＿＿）　☐ 一般　☐ 学生

参加費：＿＿＿＿＿円

宿　泊：☐ 希望する　☐ 希望しない

会社（学校）名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

部門：＿＿＿＿＿＿＿＿＿＿＿＿＿＿　役職：＿＿＿＿＿＿

住所：（〒＿＿＿＿＿）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

TEL：＿＿＿＿ − ＿＿＿＿ − ＿＿＿＿＿＿（内線 ＿＿＿＿＿）

第 4 回

ソフトウェア技術交流会

参加者募集

主　催

ソフトウェア技術者協会（SEA）

ソフトウェア分野におけるテクノロジー・トランスファの促進を目指す SEA ならではのこのユニークなイベントも，4回目をむかえます．今回は，徳島大学の山本米雄先生にコーディネーションをお願いし，また「一太郎」のジャスト・システム社のご協力を得て，会場を四国に移し，下記の要領により開催します．これまで通り，企業サイドからの多数の参加をお待ちします．

開 催 要 領

1．期日： 昭和 63 年 9 月 6 日（火）～ 7 日（水）

2．会場： 徳島大学工学部知能情報工学科 C10 教室

3．定員： 50 名

4．会費： SEA 会員 15,000円，一般 20,000円，大学関係者 2,000円（宿泊費は含みません）

5．プログラム：（予定）

9 月 6 日（火） 13:00 - 20:00

・［基調講演］ソフトウェア国富論： 浮川和宣（ジャストシステム）

・パネル討論 1： ワープロ，日本語処理，DTP

［コーディネータ］ 岸田孝一（SRA）

［パネリスト］ 浮川和宣（ジャストシステム） 神田泰典（富士通） Edgar Cooke（SRA）
中村正三郎（管理工学） 西林瑞夫（エルゴソフト） 安田知弘（東海クリエイト）

・徳島大学知能情報工学科の紹介

並列処理（高橋研究室） 自然言語処理（青江研究室） CAI（山本研究室）

・懇親パーティ

9 月 7 日（水） 09:30 - 15:00

・［講演］Hyper Card ≧ 5GL ？： 藤野晃延（FXIS）

・パネル討論 2： エキスパート・システムの応用

［コーディネータ］ 熊谷章（PFU）

［パネリスト］ 伊野誠（SRA） 大木幹雄（JIP） 山口高平（阪大） 他

・パネル討論 3： CAI とソフトウェア技術者教育

［コーディネータ］ 山本米雄（徳島大）

［パネリスト］ 落水浩一郎（静岡大） 君島浩（富士通） 杉田義明（SRA） 寺嶋祐一（日電ソフト）

・［講演］ セマンティック・データベース・モデル： 有澤博（横浜国大）

・ 会議終了後，希望者はジャストシステム見学（15:30 - 17:00）

6．申込方法： 下記の申込書に御記入の上

〒770 徳島市南常三島町 2 - 1 徳島大学工学部知能情報工学科 山本米雄

TEL 0886 - 23 - 2311 （内 4712） FAX 0886 - 23 - 2761

宛に郵便（または FAX）でお申込ください．ただし，受付は先着順とし，定員オーバーの場合はお断りすることがありますので，あらかじめ御承知おきください．会費は当日会場で申し受けます．

第1回技術交流会参加申込書

氏 名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（□ SEA 会員 □ 一般 □ 大学）

会社（大学）名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

所 属：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

住 所：〒（ ）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

電 話：（ ）－（ ）－（ ）

Call for Papers

Twenty Years Of Software Engineering: Looking Forward, Looking Back

Pittsburgh, Pennsylvania, USA May 15-18, 1989

11TH INTERNATIONAL CONFERENCE ON
SOFTWARE ENGINEERING

CHAIR
Larry Druffel
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA
(412) 268-7740

PROGRAM CO-CHAIRS
Richard Fairley
School of Information Technology
George Mason University
Fairfax, VA 22030, USA
(703) 764-6195

Dines Bjørner
Dansk Datamatik Center
Lundtoftevej 1 C
Lyngby DK-2800, Denmark
(45) 2-872-622

TOOLS FAIR CHAIR
Grady Booch
Rational
835 South Moore Street
Lakewood, CO 80226, USA
(303) 986-2405

TUTORIALS CHAIR
Norman Gibbs
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA
(412) 268-7703

LOCAL ARRANGEMENTS CHAIR
Miles Barel
Formative Technologies, Inc
Foster Plaza VII
661 Anderson Drive
Pittsburgh, PA 15220, USA
(412) 937-4942

To Be Sponsored By

ACM SIGSOFT

THE COMPUTER SOCIETY OF THE IEEE

Other sponsorships have been requested.

The term "software engineering" was coined in 1968 to denote a new discipline concerned with the technological aspects of developing and modifying computer software systems. Twenty years have elapsed since the 1968-69 NATO Workshops on Software Engineering, which are generally acknowledged to be the seminal meetings. It is now appropriate to commemorate the past twenty years of software engineering and to look forward to the next twenty years.

In surveying the accomplishments of the past twenty years, the Conference will attempt to identify recurring themes and lasting contributions. In looking forward to the next twenty years, an attempt will be made to identify the major trends in hardware, software, and society that will shape the future of software engineering.

The Conference program will include both invited and contributed papers. Authors from all countries are encouraged to submit high-quality contributions in the mainstream topical areas of software engineering and in areas that are of considerable relevance to software engineering, such as database technology, expert systems, user interfaces, and hardware support for software engineering environments. Each submission will be evaluated by the Program Committee, who will assess the quality and relevance of the submitted papers. Proposals for panel sessions are also welcome. In addition, the Conference will feature a Software Tools Fair and a tools presentation track.

Submitting a Paper: Four copies (in English) should be submitted by September 15, 1988, to either Program Co-Chair:

- Richard Fairley
  School of Information Technology
  George Mason University
  Fairfax, VA 22030 USA
- Dines Bjørner
  Dansk Datamatik Center
  Lundtoftevej 1 C
  Lyngby DK-2800, Denmark

Papers should be limited to 6000 words, full-page figures being counted as 300 words. Each paper should include a short abstract and a list of keywords indicating subject classification. Notification of acceptance will be sent by December 15, 1988, and camera-ready copy of the final version will be due on February 15, 1989.

Tools Fair: A Software Tools Fair will be held in parallel with the Conference. The Tools Fair will allow conference attendees to see and learn about current software tools. Both experimental and commercial tools will be demonstrated. In addition, the conference will feature a program track dedicated to presentation and demonstration of software tools selected by the Tools Fair Committee. Those interested in exhibiting a tool, and especially those authors interested in presenting a paper describing a particular tool and demonstrating that tool should contact the Tools Fair Chair.

- Grady Booch
  Rational
  835 South Moore Street
  Lakewood, CO 80226 USA

Further Information: For further information and/or a copy of the advance program when available, write to ICSE11, c/o IEEE Computer Society, 1730 Massachusetts Ave., N.W., Washington, D.C. 20036 USA.

Important Dates:

Submission Deadline: September 15, 1988

Acceptance Notification: December 15, 1988

Final Versions Due: February 15, 1989

IEEE THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

論 文 募 集

ソフトウェア・シンポジウム '89

1989 年 6 月 14-15 日 (予定) ◇ 農林年金会館 (東京・虎ノ門)

共 催 : (社) 情報サービス産業協会 (JISA)
ソフトウェア技術者協会 (SEA)

はやいもので、ソフトウェア・シンポジウムも第9回目を迎えます。 産業界の最前線で、日頃忙しくソフトウェアの開発・保守・運用・管理に従事しているエンジニアたちが、それぞれの仕事のなかから得た経験や知識、アイデア、あるいは意見を交換しあうための貴重な場として、多くの方々の賛同を頂き、内容もますます充実してまいりました。

ここ1-2年、ネットワークの整備や高機能ワークステーションの普及、あるいはマルチ・メディアなど、各種先進的技法/ツールの実用化など、ソフトウェア業界をとりまく状況は急速に変わりつつあります。今回のシンポジウムでは、こうした新しい社会的・技術的環境への対応を発表や討論の主題として、従来にもまして魅力的なプログラムを組みたいと考えております。

実践的技術の経験、先進的技術の試行結果、基礎研究の成果等に関して、ヴァライティに富む多数の論文がよせられることを期待いたします。

募集論文テーマ

特に限定はしていませんが、たとえば、次のような話題がセッションーテーマの候補として考えられます:

- ソフトウェア開発におけるAI関連技術の応用
- 実際のプロジェクト支援環境における問題点
- 新しい開発/保守方法論の実践とその評価
- ユニークなアプリケーション開発の事例
- 技術者教育のケース・スタディ
- ドキュメンテーションの機械化/合理化
- ネットワークのセキュリティ/維持/管理
- プロジェクトにおける技術移転
- LANを用いた分散型環境の構築・利用
- 先進的ソフトウェア・ツールの試作・実験
- ソフトウェア・プロダクトの流通と権利保護
- 定量的マネジメント - 各種モデルの適用と評価

等々。

応募要領

応募論文の概要を4千字~6千字程度にまとめたもの (ワープロまたはコンピュータ出力でA4版4~5ページ)のコピー10部を、

1988 年 11 月 30 日までに

2人のプログラム委員長(藤野 晃延または飯沢 恒)のいずれか宛にお送りください。

プログラム委員会で応募論文の審査を行い、結果を1989年1月末までに、応募者全員にお知らせします。

審査を通過した方々には、発表用の最終論文(予稿集に載せるためのカメラ・レディ原稿)を1989年3月末までに仕上げていただくことになります。

その他、パネル討論のテーマや招待講演、チュートリアルなどに関して、何かアイデアがありましたら、実行委員長までご連絡ください。

シンポジウム・スタッフ

実行委員長
林 香
ソフトウェア・リサーチ・アソシエイツ
環境開発本部
〒102 千代田区平河町 1-1-1
Tel. 03-234-2611

プログラム委員長
飯沢 恒
三菱電機東部コンピュータ・システム
コントロール第一部
〒247 鎌倉市大船 2-19-28
Tel. 0467-45-7315

ツール展示委員長
高田 佳彦
日本電子計算
科学技術事業部
〒103 中央区日本橋兜町 6-7
Tel. 03-668-6171

藤野 晃延
富士ゼロックス情報システム
技術推進室企画課
〒160 新宿区西新宿 3-16-6 西新宿水野ビル
Tel. 03-378-8010

ソフトウェア信頼性の
定量的管理に関する
ワークショップ

参加者募集

主催：高信頼性ソフトウェア研究会
後援：ソフトウェア技術者協会（SEA）

昨年の夏，高信頼性ソフトウェア研究会では，IEEE Draft Standard for software Quality Metrics の内容を検討するワークショップを伊豆高原で開催し，大きな成果を収めました（報告書は昨年１０月に SEA から刊行ずみ）．

今回のワークショップでは，議論をさらに一歩進めて，当研究会メンバーの手で現在取りまとめているソフトウェア信頼性管理（各種の定量化指標をどのように設定し，現実のプロジェクトにおいてどのように計測を実施し，結果の妥当性をどうやって評価するか）に関するガイドライン試案を現実のプロジェクトに適用した場合，どのような問題点が生じるか，それはどうすれば回避できるか，等などのことがらを，さまざまな角度から検討したいと考えています．

具体的には，参加者の方々にそれぞれ，身のまわりにある実際のプロジェクトをサンプルとして持ち寄っていただき，それらのプロジェクトにおいて，このガイドラインにもとづく信頼性の管理を実施したらどうなるかを，ケース・スタディ方式で考えてみようということです．

討論の性格上，参加者の人数は 25 名（当研究会メンバーを含めて）に限定させていただきます．関心ある方々の積極的な御参加をお待ちします．

開 催 要 領

１．期日： 昭和 63 年 8 月 24 日（水）午後から 26 日（金）正午まで（2 泊 3 日）

２．会場： 八ヶ岳高原・泉郷（山梨県北巨摩郡・JR 中央本線小淵沢駅からバス 10 分）

３．定員： 25 名

４．費用： 50,000円

現地集合・現地解散とし，期間中の宿泊費（３食付き）および資料代を含みます．

５．スタッフ（高信頼性ソフトウェア研究会）：

代表： 鳥居宏次（大阪大学）

幹事： 浦野義頼（KDD 上福岡研究所）

会員： 天池学（沖電気） 飯沢恒（三菱電気東部コンピュータ） 岩丸良明（三井銀総研） 大場充（IBM）
岸田孝一（SRA） 久保宏志（富士通） 春原猛（三菱電機） 寺本雅則（日本電気）
花田収悦（NTT） 保田勝通（日立） 山田淳（東芝）

７．申込方法： 裏面の参加申込用紙に必要事項を記入の上，昭和 63 年 7 月 18 日（月）までに，郵便または FAX で，下記宛お送り下さい．申込者多数の場合には，ポジション・ステートメントの内容を審査の上，参加者を選考させていただきますので，あらかじめ御了承ください

申込み先：

〒356 埼玉県上福岡市大原 2 － 1 － 15
KDD 上福岡研究所 主幹研究員
浦 野 義 頼
Tel: 0492 － 66 － 7304
Fax: 0492 － 66 － 7510

***** 裏面に申込用紙があります *****

送付先：

〒356　埼玉県上福岡市大原 2 - 1 - 15
KDD 上福岡研究所　主幹研究員　浦野義頼
Fax: 0492 - 66 - 7510

申込日付：　月　日

ソフトウェア信頼性の定量的管理に関するワークショップ
参加申込用紙

氏　名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（ふりがな）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

年　齢：＿＿＿＿＿＿（歳）　性　別：□ 男 □ 女　血液型：＿＿＿＿＿＿

会社名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

部　門：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿　役　職：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

住　所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

TEL:（　　）－（　　）－（　　）内線（　　）

ポジション・ステートメント：

信頼性定量化のケース・スタディとして，どのようなプロジェクトをワークショップで提示できますか？　そのプロジェクトは，すでに完了したものですか？　現在進行中ですか？　それとも，これからスタートする予定のものですか？　プロジェクトの特性は？　開発チームのサイズ，メンバー構成，開発期間，開発内容・手順，信頼性やその他の品質上のリクワイヤメントなど，なるべく具体的にお書きください．紙面が足りない場合は，別紙を追加していただいてかまいません．

第 6 回 夏のプログラミング・ワークショップ

ソフトウェア・エンジニアリング技法のケース・スタディ

－ ソフトウェア開発の「銀の弾丸」を求めて －

参加者募集

主催：ソフトウェア技術者協会

後援：（社）情報サービス産業協会

これまで５年間「若手の会 in 盛岡」として開催されてきたこのワークショップも，いまではすっかり定着し，業界各社からも，次世代のエースを鍛える格好なトレーニングの「場」として高く評価されています．

第６回目を迎える今年は，会場を八ヶ岳山麓の別荘村に移し，運営スタイルも「若手の会」ＯＢを中心とするケース・スタディ方式を採用して，より具体的な課題を全員参加のグループ討論を通じて徹底追求することにしました．今回の討論テーマとしては，ややもすれば名前だけが先行し現場への浸透が遅れている開発技法のいくつかを取り上げ，それぞれの長所・短所を十分に比較検討しようと考えています．

具体的には：

・形式的仕様記述を用いたアプローチ
・オブジェクト指向のアプローチ
・エキスパート・システムによるアプローチ
・CASE ツールを用いたアプローチ
・プロトタイピングによるアプローチ

などを用いて現実的な例題にアタックし，F. Brooks の高名なエッセイに述べられた「銀の弾丸」（ソフトウェアという魔物を倒す最終兵器）の可能性を探ろうというのが，その狙いです．

多くの若手技術者方々の積極的な御参加をお待ちします．

開 催 要 領

１．期日： 昭和 63 年 8 月 31 日（水）午後から 9 月 3 日（土）正午まで（3 泊 4 日）

２．会場： 八ヶ岳高原・泉郷（山梨県北巨摩郡・JR 中央本線小淵沢駅からバス 10 分）

３．定員： 35 名

４．参加資格： 年齢 30 歳未満，経験 2 年以上のソフトウェア技術者

５．費用： 60,000円(SEA 会員，JISA 会員)， 80,000円(一般)

現地集合・現地解散とし，期間中の宿泊費（３食付き）および資料代を含みます．

６．スタッフ：

実行委員長： 藤野晃延（富士ゼロックス情報システム）

プログラム委員長： 坂下 秀（アステック）

プログラム委員： 岡芹晃（東芝エンジニアリング） 沢田寿美（ＳＲＡ）

松下和隆（三菱電気東部コンピュータ） 渡辺雄一（電力計算センター） 他・数名

招待講演者（予定）：柴山悦哉（東京工業大学） 寺野隆雄（電力中央研究所）

７．申込方法： 裏面の参加申込用紙に必要事項を記入の上，昭和 63 年 7 月 18 日（月）までに，郵便または FAX で，実行委員長宛お送り下さい．申込者多数の場合には，プログラム委員会でポジション・ステートメントの内容を審査の上，参加者を選定させていただきますので，あらかじめ御了承ください

申込み先：

〒160 東京都新宿区西新宿 3 － 16 － 6 西新宿水野ビル

富士ゼロックス情報システム（株）

技術推進室 企画課 藤 野 晃 延

Tel: 03 － 378 － 8010 (ext 6791) Fax: 03 － 378 － 7298

***** 裏面に申込用紙があります *****

送付先：　　申込日付：　　月　　日

〒160　東京都新宿区西新宿 3 - 16 - 6 西新宿水野ビル
富士ゼロックス情報システム（株）　技術推進室企画課　　藤野晃延
Fax: 03 - 378 - 7298

第 6 回 夏のプログラミング・ワークショップ
参加申込用紙

氏　名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（ふりがな）＿＿＿＿＿＿＿＿＿＿＿＿

年　齢：＿＿＿＿＿（歳）　性　別：□ 男 □ 女　血液型：＿＿＿＿＿

会社名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

部　門：＿＿＿＿＿＿＿＿＿＿＿＿　役　職：＿＿＿＿＿＿＿＿＿＿＿＿

住　所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

TEL：（　　）－（　　）－（　　）内線（　　）

これまでの主な開発経験：

ポジション・ステートメント：

ソフトウェア開発工程（特にシステム分析や設計などの上流工程）の生産性や品質の改善について，今後どんな方策を講じるべきかを，自分のこれまでの経験や知識に則して，なるべく具体的に書いてください．

Call for Position Papers

第２回　ソフトウェア教育に関するワークショップ

ソフトウェア教育への技術的接近

主催

教育分科会（SIGEDU）

ソフトウェア技術者協会（SEA）

ＳＥＡ　ＳＩＧＥＤＵは分科会の発足以来、各社の教育活動の情報交換や各種の見学会を開催し、活発に活動してまいりました．昨年のワークショップにおいては，さいわいにも第一線でソフトウェア教育に関心のある個性豊かな参加者に恵まれ、率直で新鮮な議論をすることができました．

今回は、さらに一歩突っ込んで、より具体的で実践的なテーマを取り上げ議論を行ないたいと思います．特に教育の手法、テクニックや現場の技術マップと教育体系との対比や、新しい教育のスタイルなどについても議論してみたいと考えています．教育体系を中心として、それをどうやって導き出すか、どんな形で教えるかを技術的に究明しようという方や新しい技術の開発現場への移転を試みておられる方々の参加を募ります．今回のサブテーマとして、

・教育技法・テクニック、
・ＣＡＩ・マルチメディア教育
・ＡＩ・オブジェクト指向、環境技術教育
・教育評価、効果測定

取り上げたいと考えていますが、以下の分野について問題意識を持っておられる方や話題を提供して下さる方も歓迎致します．気軽にお申込をしてくださるようお願いします．

・新入社員教育　・入門教育　・中堅社員教育　・管理者教育　・学校教育と企業内教育
・先進的ソフトウェア工学と教育　・ソフトウェア技術移転　・キャリアパスの設定と教育　・経営戦略と教育活動

開催要領

１．**期日：**　昭和 63年 9月 29日（木）午後から 10月 1日（土）正午まで（2泊 3日）

２．**会場：**　滝の湯ホテル（山形県・天童市　TEL：0236-54-2211）

３．**定員：**　30名

４．**費用：**　65,000円（SEA会員），75,000円（一般）
現地集合（ 9月 29日（木） 13:30 ）・現地解散（ 10月 1日（土） 12:00）とし、期間中の宿泊費（食事付き）および資料代を含みます．なお、ワークショップ終了後、討議の内容を報告書にまとめ、参加された方に配布致します．

５．**申込方法：**　参加を希望される方は、裏面のポジション・アンケート付きの参加申込書に必要事項を記入の上、プログラム委員長宛てに、ＦＡＸまたは郵送にてお送り下さい．折り返し、ポジション・ペーパを 8月19日（金）までに書いて頂くようご案内を差し上げます．申込が多数の場合には、プログラム委員会でポジション・ペーパの内容を審査の上、 9月上旬までに参加者を決定し、結果を応募された方全員にお知らせします．

申込書送付先：

〒410-03　静岡県沼津市宮本140　富士通(株)
ソフトウェア開発本部・教育課
君島　浩
Tel: 0559-23-2222 (EX.5730) Fax: 0559-23-1991

６．**その他：**昨年実施したワークショップの報告書を１冊３，０００円にて頒布しております，希望される方はＳＥＡ事務局（０３－２３４－９４５５）まで，お問い合わせください．

（スタッフ）	（敬称略）
杉田　義明	［実行委員長］（ＳＲＡ）
君島　浩	［プログラム委員長］（富士通）
	［プログラム委員］
大浦　洋一	（シーイーシー）
大木　幹雄	（日本電子計算）
大久保　功	（横河ヒューレットパッカード）
河村　一樹	（日本電子専門学校）
逆瀬川　明宏	（月島機械）
篠崎　直二郎	（日本電気ソフトウエア）
中園　順三	（富士通ＢＳＣ）
中山　照章	（富士通ＳＳＬ）
平本　巌	（電力計算センター）
平山　伸一	（神戸コンピューターサービス）
山口　圭一	（ＳＲＡ）

（ワークショップ参加申込書は裏面にあります．）

宛先:
富士通・ソフトウエア開発本部・教育課
第2回ＳＥＡ教育ワークショップ
プログラム委員長
君島　浩　様
(FAX: 0559-23-1991)

昭和　　年　　月　　日

第２回教育ワークショップ・参加申込書

参加を希望される方はこの申込書に記入の上、プログラム委員長宛に，このままＦＡＸにて申込んでください。

氏　名：＿＿＿＿＿＿＿＿（年齢：＿＿歳，性別：男　女）　申込種別（SEA　一般）
会社名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿　所属：＿＿＿＿＿＿＿＿
住　所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿
TEL: ＿＿＿－＿＿＿－＿＿＿＿＿＿　FAX: ＿＿＿－＿＿＿－＿＿＿＿＿＿

現在の職種(＿＿＿＿＿＿＿＿＿＿＿＿＿＿)、業務・役割(＿＿＿＿＿＿＿＿＿＿＿＿)

興味のあるテーマ(＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿)

＊＊＊＊＊＊＊＊＊＊＊　ポジション・アンケート　＊＊＊＊＊＊＊＊＊＊＊

1. **教育技法・テクニック：** 教育そのものの内容を有効でかつおもしろく、エキサイティングなものにするためにさまざまな試みが考えられます。技術として教育をとらえた場合、あなたの意見は：

2. **ＣＡＩ，マルチメディアの活用：** 伝統的・教室スタイルの研修方法から、個人の特性に応じた新しいタイプの教育に人気が高まっているようです。教育環境としてさらに有効なものにするためにあなたの意見は：

3. **ＡＩ，オブジェクト指向、環境技術など新しい技術の教育に関して：** ソフトウエア開発の諸問題を解決するために、新しい技術の現場への導入が期待されています。特に実践的な環境技術の普及について、具体的な方針・方策を考えると？

4. **教育の評価、効果測定**一連の活動をどのように評価するか、評価のためのメカニズム、ツールは？定量的なアプローチを前提に、あなたの経験、アイデア、意見は？

5. **オフレコ・セッション：** 討論の内容を記録に残さないで、自由に話し合おうということで昨年は採用問題を取り上げました。今回も同じ形式にて実施したいと思います。ついては、期待される話題、狙いについてお書きください。

第８回

ソフトウェア信頼性シンポジウム

発表者募集

主催：高信頼性ソフトウェア研究会
ソフトウェア技術者協会関西支部
電子情報通信学会ＦＴＣＳ研究専門委員会（予定）

ソフトウェア開発の信頼性向上に関するさまざまなトピックスをめぐって，インフォーマルな（しかし，だからこそ核心を突いた）議論を展開できるユニークな１つの場として運営されてきたこのシンポジウムも，回を重ねることすでに８回目を迎えることになりました．

開催要領

１．期日： 昭和 63 年 10 月 28 日（金）～ 29 日（土）
２．会場： 大阪ガーデンパレス（JR 新大阪駅から徒歩１０分）
３．定員： 70 名
４．費用（予定）： SEA 会員 7,000円， 一般 10,000円， 学生 3,000円
６．スタッフ：

実行委員長： 鳥居宏次（阪大）
プログラム委員長： 臼井義美（JIP） 浦野義頼（KDD）
プログラム委員： 天池学（沖電気） 飯沢恒（三菱電気東部コンピュータ） 岩丸良明（三井銀総研）
大場充（IBM） 岸田孝一（SRA） 久保宏志（富士通） 春原猛（三菱電機）
寺本雅則（日本電気） 花田収悦（NTT） 保田勝通（日立） 山田淳（東芝）

昨秋と今春は，ソフトウェア品質の定量化指標に関する IEEE 標準化試案を題材として，品質管理のフレームワークについての議論を２回続けて行いましたが，今回は，ふたたび以前の自由なスタイルに戻って，一般参加者からの事例発表を中心にプログラムを構成したいと考えています．

たとえば：

・ソフトウェア信頼性向上を目指して何らかの方策を講じたプロジェクトの事例
（成功例，失敗例いずれでも）
・信頼性の定量的な把握のための試みや提案
・各種設計技法やテスト技法の有効性に関する経験的評価
・プログラム・バグの分類や原因追跡

など，何でも結構です．身近な経験にもとづく具体的な報告を，ふるってお寄せください．

発表希望者は，裏面の応募用紙に必要事項を記入の上，昭和 63 年 8 月 22 日（月）までに，郵便または FAX で，プログラム委員長宛お送り下さい．応募者多数の場合には，プログラム委員会で応募内容を検討した上，プログラム全体でのバランスを考慮して，採否を決めさせていただきます．

審査結果の応募者への通知と，確定したプログラムにもとづく一般参加者の募集は，9 月下旬になる予定です．

申込み先：

〒356 埼玉県上福岡市大原 2 － 1 － 15
KDD 上福岡研究所
主幹研究員 浦野義頼
Tel: 0492 － 66 － 7304 Fax: 0492 － 66 － 7510

***** 裏面に申込用紙があります *****

送付先：

〒356　埼玉県上福岡市大原 2 - 1 - 15
KDD 上福岡研究所　主幹研究員　浦野義頼
Fax: 0492 - 66 - 7510

申込日付：　　月　　日

第 8 回 ソフトウェア信頼性シンポジウム
発表応募用紙

氏　名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（ふりがな）＿＿＿＿＿＿＿＿＿＿＿＿＿＿

年　齢：＿＿＿＿＿＿（歳）　　性　別：□ 男　□ 女　　血液型：＿＿＿＿＿＿

会社名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

部　門：＿＿＿＿＿＿＿＿＿＿＿＿＿＿　役　職：＿＿＿＿＿＿＿＿＿＿＿＿＿＿

住　所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

TEL：（　　）－（　　）－（　　）内線（　　）

発表の題目：

発表の要旨：

紙面が足りない場合は別紙を追加していただいてかまいません．また，関連する論文または技術資料などがありましたら，添付してください．

第 4 回 SEA 環境ワークショップ

実践的ソフトウェア開発環境に関する集中討論

発 表 者 募 集

主催：ソフトウェア技術者協会

今度で第４回目を迎える恒例の SEA 環境ワークショップは，開催時期を冬から秋に，また会場をこれまでの雪国から異国情緒豊かな港町・神戸に移して，気分を新たに，熱気あふれる集中討論を展開したいと考えています．

開 催 要 領

１．期日： 昭和 63 年 11 月 24 日（木）～ 11 月 26 日（土）

２．会場： 神戸国際会議場（神戸市ポートアイランド）

３．宿泊： 新神戸オリエンタル・ホテル（JR 新神戸駅前に今秋オープン）

４．定員： 50 名

５．費用（予定）： SEA 会員 70,000円（宿泊含む）／30,000円（宿泊なし）
一般 80,000円（宿泊含む）／40,000円（宿泊なし）

６．スタッフ：

実行委員長： 盛田政敏（KCS）
プログラム委員長： 臼井義美（JIP） 中野秀男（阪大）
プログラム委員： 青木 淳（FXIS） 久保宏志（富士通） 熊谷 章（PFU）
坂下 秀（アステック） 佐藤千明（長野県協同電算） 塩谷和範（SRA）
高野 豊（松下電器） 野村行憲（ICS） 平尾一浩（HST）
深瀬 弘恭（アスキー） 松浦敏雄（阪大） 他

運営形態は，第２～３回目と同じように，いくつかのテーマを中心とする連続パネル形式を採用します．先日開かれた企画ミーティングでは，これまでのワークショップにおける討論がいくらか「とびすぎて」いたことが反省され，より地に足のついた具体的な議論ができるようにという配慮のもとに，次の４つのセッション・テーマが選ばれました：

(1) Hyper Media のソフトウェア開発への応用
(2) Man-Machine Interface の改善
(3) Micro-Mainframe Link の構築と利用
(4) Workstation 上での CASE ツールの活用

各３時間ずつのセッションでは，まず，これらのテーマに関連するいくつかの技術発表または問題提起をお聞きしたあと，アトランダムな自由討論に入る予定です．また，前回初めて試みてたいへん好評だったツール展示（ワークステーションまたは PC を会場に持ち込んでのデモ）も引き続きプログラムに組み入れたいと思います．

そこで，上記４つのテーマに関する発表および関連ツール展示のプロポーザルを募集します．裏面の応募用紙に必要事項を記入の上，昭和 63 年 8 月 1 日（月）までに，郵便または FAX で，実行委員長宛お送り下さい．応募者多数の場合には，プログラム委員会で応募内容を審査の上，採否を決めさせていただきます．

審査結果の応募者への通知と，確定したプログラムにもとづく一般参加者の募集は，9 月中旬になる予定です．

申込み先：

〒650 神戸市中央区浪花町 64 三宮電電ビル 6F
（株）神戸コンピューターサービス
ソフトウェア事業部 盛田政敏
Tel: 078 - 391 - 8291 Fax: 078 - 391 - 5452

***** 裏面に申込用紙があります *****

送付先：　　申込日付：　　月　　日

〒650　神戸市中央区浪花町 64 三宮電電ビル 6F
（株）神戸コンピューターサービス　ソフトウェア事業部　盛田政敏
Fax: 078 - 391 - 5452

第 4 回 実践的ソフトウェア開発環境に関する集中討論
発表／ツール展示応募用紙

氏　名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿（ふりがな）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

年　齢：＿＿＿＿＿＿（歳）　　性　別：□ 男 □ 女　　血液型：＿＿＿＿＿＿

会社名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

部　門：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿　役　職：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

住　所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

TEL：（　　）－（　　）－（　　）内線（　　）

種別：　□ 技術発表　　□ ツール展示

発表の題目 または ツールの名称：

発表の要旨 または ツールの概要：

紙面が足りない場合は別紙を追加していただいてかまいません．また，関連する論文または技術資料などがありましたら，添付してください．

SEA会員状況（昭和 63 年 6 月 22 日現在）

正会員	1425 名（4 月 4 日から 66 名増）
賛助会員	49 社（4 月 4 日から 7 社増）

<正会員の勤務地および居住地域分布>

	勤務地域	居住地域
北海道＝	8	8
山形 ＝	1	1
宮城 ＝	4	3
岩手 ＝	6	6
福島 ＝	1	0
秋田 ＝	1	1
新潟 ＝	7	7
栃木 ＝	7	6
群馬 ＝	3	3
茨城 ＝	8	10
埼玉 ＝	18	97
千葉 ＝	17	89
東京 ＝	775	473
神奈川＝	109	251
長野 ＝	36	36
富山 ＝	3	3
福井 ＝	4	2
石川 ＝	4	4
静岡 ＝	18	18
岐阜 ＝	2	8
愛知 ＝	45	36
和歌山＝	2	3
三重 ＝	1	3
滋賀 ＝	9	9
京都 ＝	17	27
大阪 ＝	188	142
奈良 ＝	3	10
兵庫 ＝	33	58
愛媛 ＝	2	2
徳島 ＝	3	3
岡山 ＝	1	1
広島 ＝	4	4
鳥取 ＝	0	1
島根 ＝	1	1
福岡 ＝	12	13
熊本 ＝	27	26
大分 ＝	4	4
佐賀 ＝	1	1
宮崎 ＝	1	1
鹿児島＝	6	6
沖縄 ＝	1	1
韓国 ＝	1	1

<男女分布>

男 ＝	1351
女 ＝	74

<年齢分布>

20 以下 ＝	0
20_24 ＝	50
25_29 ＝	305
30_34 ＝	384
35_39 ＝	335
40_44 ＝	215
45_49 ＝	80
50_54 ＝	23
55_59 ＝	17
60 以上 ＝	14

<血液型分布>

A型 ＝	555
O型 ＝	414
B型 ＝	209
AB型＝	148

<賛助会員会社名>

IN情報センター，PFU，インターナショナル·データ·リサーチ，エヌ·ティ·ティ·システム，エムテイシー，サン·ビルト印刷，ジェーエムエーシステムズ，セントラル·コンピュータ·サービス，センチュリーリサーチセンター（大阪），ソフトウェア·リサーチ·アソシエイツ，ニコンシステム，ニッポンダイナミックシステムズ，ヒラタ·ソフトウェア·テクノロジー，マイクロキャビン，リクルート，リコーシステム開発，旭化成工業，伊藤忠エレクトロニクス，構造計画研究所，神戸コンピューターサービス，近畿日本ツーリスト，経調，三菱電機，三菱電機セミコンダクトソフトウェア，村田技研，千代田製作所，辻システム計画事務所，情報システムサービス，東海クリエイト，東電ソフトウェア，昭和電工コンピュータサービス，阪神計算センター，日進ソフトウェア，日本エム·アイ·シー，日本システム，日本システムサイエンス，日本情報システムサービス，日本能率コンサルタント，日本ビジネスデータプロセシングセンター，日立製作所，日立ビジネス機器，富士ゼロックス情報システム，富士通，富士通ビジネスシステム，富士通ビー·エス·シー

（アイウエオ順）

ソフトウェア技術者協会
〒102 東京都千代田区隼町2-12 藤和半蔵門コープビル505
TEL.03-234-9455 FAX.03-234-9454