SEAMAIL

Monthly Newsletter from
Software Engineers Association

Volume 1, Number 4 | April 1986

目　次

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っかています．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．

常任幹事： 岸田孝一　鈴木弘　長井剛一郎　吉村鉄太郎

幹事： 稲田博　臼井義美　岡本吉晴　落水浩一郎　皆藤慎一　木村高志　久保宏志　斎藤信男　三枝守正　杉田義明　辻淳二　鳥居宏次　中園順三　針谷明　松本崇純　松原友夫　水谷時雄　盛田政敏　三浦信之

会計監事： 近藤秀朗　吉村成弘

常任委員長： 鈴木弘（企画総務）　吉村鉄太郎（技術研究）　岸田孝一（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)：岡本吉晴　久保宏志　引地信之　松尾正敏　水谷時雄

管理分科会(SIGMAN)：岸田孝一　塩野富教　芝原雄二　鈴木信裕

教育分科会(SIGEDU)：大浦洋一　杉田義明

再利用分科会(SIGREUSE)：青島茂　阿倍正平　村井進

ＡＩ分科会(SIGAI)：白井豊　菅原勝彦　野辺良一　藤野晃延

ネットワーク分科会(SIGNET)：鈴木弘

支部世話人 関西支部：臼井義美　盛田政敏

横浜支部：藤野晃延

ＳＥＡＭＡＩＬ編集グループ：大西亮一　岸田孝一　佐原伸　沢田寿実　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　山内徹　渡辺雄一

ＳＥＡＭＡＩＬ　Ｖｏｌ．１，　Ｎｏ．４　　昭和６１年４月１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）

〒１６６　東京都杉並区高円寺南１－５－４　高円寺サンハイツ４０４

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　５００円

編集部から

ＳＥＡＭＡＩＬ第４号は，「セミナー・ワークショップ特集」ということで編集しました．

昨年１２月の設立以来，２月の長岡ワークショップ，そして３月にはセミナー・ウィーク（日本ＵＮＩＸユーザ会との共催）と，立て続けに２つの大きな行事が開催され，会員各位の御協力のおかげで，ともに成功裡に終わりましたが，今回の特集は，これらのイベントに参加できなかった方々（特に地方在住の会員の方々）への情報提供を意図しています．

当初の編集企画では，セミナー・レポートのみを特集するつもりだったのですが，ワークショップの報告書を編集している段階で，グループ討論のまとめが，なかなかおもしろい形にまとまってきたので，それらをぜひ，みなさんに紹介したいということになりました．そのため，発行が予定より若干遅れましたが，内容・ページ数ともに充実したものになったと自負しています．

長岡ワークショップ

第１回ＳＥＡワークショップは，前号で報告したように，「実践的ソフトウェア開発環境」をテーマとし，２月１９～２２日の４日間，新潟県長岡市に３５名の参加者を集めて開催されました．

従来のワークショップでは，グループ別にサブテーマを決めて討論するという形式がふつうでしたが，このワークショップでは，なるべく「実践的」な討論をしたいということから，参加者が日頃抱えている問題意識を持ち寄って，相互に批判や忠告をだしあうという形式をとりました．また，最後のパネル討論では，各グループの話し合いのなかで未解決に終わった問題を全員で議論しました．

なお，ここに収めたグループおよびパネル討論の記録は，ワークショップ報告書に収録されているオリジナル・レポートに若干手を加えて編集したものです．

春のセミナー・ウィーク

３月１０～１５日の間，ＳＥＡとしての初めてのセミナー・ウィークが，東京虎の門の日消会館で，ｊｕｓ（日本ＵＮＩＸユーザ会）とのジョイント・イベントとして開催されました．

ここにおさめたレポートは，初日の基調講演および第２日以降のＳＥＡ企画セッションのすべてを，編集部員を中心とするレポータが実際に講演を聞いてまとめたものです．

講演のテーマは，いずれも，現在ソフトウェアの開発に携わる者にとって，真剣に取り組まなくてはならない問題ばかりでした．それだけに，限られたスペースで３時間の講演をまとめるのは，かなりむずかしい仕事だったと思います．あらためてレポータ各位の御努力に感謝の意を表します．

セミナーに参加できなかった方々には，十分に講演の雰囲気が伝わることと思います．また，参加者には，もう一度講義内容を反すうする手助けになることでしょう．

以下に，セミナーの各セッションごとの有料参加者数の一覧を載せておきます．

NO	テーマ	申込
A0	ソフトウェアの技術移転	14
S1	シグマプロジェクト	36
S2	ソフトウェアの再利用	41
S3	ソフトウエアDBマネジメント	38
S4	J-starによるドキュメント	38
S5	エキスパートシステム	35
S6	オブジェクト指向プログラミ	37
S7	ソフトウエアの著作権	18
S8	品質保証と管理	25
S9	テクニカルマネジメント	31
J1	UNIXの現状と動向	26
J2	UNIXの高度な利用	42
J3	日本語UNIXとドキュメント	45
J4	Shellプログラミング	40
J5	Cプログラミング環境	37
J6	UNIXシステムプログラミング	46
J7	UNIXネットワークの運用	47
J8	スーパユーザのためのUNIX	48
	合　計	644

第１回ＳＥＡワークショップ

実践的ソフトウェア開発環境を考える

グループ討論Ａ

久保宏志
情報処理振興事業協会

１．グループメンバ

グループを構成したメンバの氏名・所属は次の通りである（５０音順・敬称略）

久保　宏志（ＩＰＡ）
近藤　一成（ＣＳＫ）
樋口　勝久（管理工学）
深瀬　弘恭（アスキー）
松原　友夫（日立ＳＫ）
三浦　信之（日本コンピュータ・システム）
盛田　政敏（神戸コンピューターサービス）

コーディネータは久保がつとめた．

２．グループ討論の経過

(1) 2月20日　11:00 - 17:30

コーディネータを除く６人が，樋口・深瀬・近藤・松原・盛田・三浦の順にそれぞれのポジションを説明し，質疑応答を通じて，お互に相手の問題を理解することにつとめた．それぞれのポジションは，報告書所収のポジションペーパを参照してもらうこととし，その紹介は省略する．

ここでの討論でのゴールは，翌日の討論テーマを決めることに置いた．その際，全員の関心の共通項をテーマに選ぶ方針はとらなかった．それぞれが，自分の問題を解決するためにどんなことを討論してもらいたいかを，自分できめることとした．テーマそのものは，この日の討論が一巡したところで，メンバ各位から出してもらった．それは次の通りである．

樋口：ポジションペーパの内容の外から，討論テーマを選んでもらうこととした．ポジションの説明をきいて，実行できることは実行していると理解したからである．また，実行できていないことは，討論仲間が手助けできるレベルを越えていると，感じたからでもある．

深瀬：２つのテーマを希望した．開発環境用の設備資金をどう調達するか．経営者にどう説くか．これが１つ．使いすてツールを推進してきたが，すべてのツールが使いすてであってよいと思っているわけではない．使いすてツールの是非について討論したい．進め方のアイデアも集めたい．これがもう１つである．

近藤：現在ＶＡＸを持ちＵＮＩＸをのせている．ＡＩツールものせている．これを大型のビジネス・アプリケーション開発に役立てるにはどうすべきか．この問題の検討を指示されている．一緒に考えてもらいたい．

松原：「ＣＯＢＯＬをどうするか」をテーマにしたい．特殊な問題にみえるかもしれないが，そうではない．ソフトウェア開発現場を象徴する問題である．これを論ずる中から，より広い範囲の問題に対する一般解を導びくことができる．日立ＳＫではこの問題への処方せんをつくり実践している．

盛田：開発用のマシンを入れたいと思っている．現在の開発業務の内容からすると，ターゲット・マシンと同種のものを入れるのが無難であるが，シグマ・プロジェクトが考えている方向に仕事のスタイルを寄せていくには，ＵＮＩＸマシンを入れておくのがよい．そのあたりで，いま悩んでいる．ＵＮＩＸマシンを開発用に入れるとして，現在の開発業務にどう使っていくか．どこでメリットを出すか．この問題について知恵を借りたい．

三浦：開発用のマシンを入れたいが，大きな機械，高い機械を入れる状勢にはない．パソコンを入れて開発環境の効用をデモすることによって，意識改革をすすめたい．社内でアンケート調査したら，ドキュメント標準化への要求が強いことがわかったので，デモ内容は，ドキュメント標準化支援を中心にしたい．その計画づくりを手伝ってもらいたい．

(2) 2月21日 9:00 - 11:00

（1）に書いたことを，ワークショップ参加者全員の会議で発表し，他のグループとの意見交換を行った．発表は久保が担当した．

(3) 2月21日 11:40 - 18:00

前日の討論の最後に出してもらった各自のテーマについて，討論を行った．討論は，各人が改めてテーマ説明を行い，他のメンバが問題解決に役立つ意見をのべる，類似事例を紹介する，あるいは参考になりそうな情報を提供する，という形ですすめた．これを深瀬・近藤・松原・盛田・三浦・樋口の順に行った．なお，樋口はコピー対策を討論テーマに選んだ．

(4) 2月21日 20:00 - 22:00

シグマ・プロジェクトの状況について，久保から若干の情報を提供し，意見交換を行った．２日間の討論のまとめとして，グループ単位で未解決の問題３点を挙げることが指示されていたので，最後にその整理を行い，翌日のクロージング・パネルでの発表は，松原が担当することに決まった．

当グループが挙げた未解決問題は，次の通りである．

・開発環境の中でＣＯＢＯＬをどう扱うべきか

・「使いすてツール」の是非とその進め方

・開発環境をターゲット・マシンから分離するための方策と仕掛け

(5) 2月22日 9:00 - 11:30

クロージング・パネルにおいて，松原が（４）に挙げた問題を発表した．次節は，松原が発表用につくったＯＨＰフィルムに基づいている．もちろん文章に対する責任は，全面的にこの稿を書いている久保にある．

３．３つの未解決問題

３．１．問題へのメンバの関わり

以下の表のように整理できる．

３つの問題を１つにまとめて，次のようにいうこともできる：

> ＣＯＢＯＬを主たるプログラミング言語としているソフトウェア開発現場の環境転換をいかにはかるべきか．「使い捨てツール」の対立概念としての「成長進化するツール」を示し，それによって切り開かれるであろう新しい境地への到達願望をドライビング・フォースにして環境転換を促進することは，むつかしそうである．「使い捨てツール」の思想を前面に押し出して環境転換を実践することを積み重ねない限り，環境転換は，永久に見果てぬ夢でありつづけるおそれがある．「使い捨てツール」の思想を実践しようにも，ＣＯＢＯＬを主たるプログラミング言語としている開発現場には，その思想を体得した実践者がいるとは限らない，という問題もある．さてどうするか．

	COBOL	使い捨てツール	環境転換
久保	関係あり	関係あり	関係あり
近藤	関係あり	－	問題提起
樋口	－	関係あり	関係あり
深瀬	－	問題提起	解決済
松原	問題提起	関係あり	過渡期
三浦	関係あり	－	関係あり
盛田	関係あり	－	問題提起

３．２．ＣＯＢＯＬ問題

開発環境の中でＣＯＢＯＬをどう扱うべきかという問題である．この問題へのアプローチとして，代表的なものを３つ挙げることができる．それを以下に示す．

（１）ＣＯＢＯＬ環境支援

ＣＯＢＯＬ用ツールを環境上に整備していくというアプローチである．このアプローチをとるときには，ＣＯＢＯＬの生存・繁栄を環境が助けてよいか，という疑問への答を用意しなければならない．

（２）ＣＯＢＯＬ封じ込め

ＣＯＢＯＬをツールや別の言語で包み隠そうというアプローチである．このアプローチはびほう的であるから，ほんとうにツールで隠せるか，隠しきってよいか，という問題がつきまとうことは避けられない．

（３）ＣＯＢＯＬ否定

代りの手段で置きかえるというアプローチである．置きかえの実践は，代りの手段と共存させながら陶汰をはかるという形をとることになろう．もっとも健全な態度であるが，代りの手段はあるかという，基本的な問いに答えなければならない．その答えを欠いたままでは，空理空論にすぎず，実践規範とは

なりえない．

われわれがとるべきはＣＯＢＯＬ否定の態度であろう．そして，当面なすべきは，味方をふやすこと，知恵を集めることである．

第１に，ＣＯＢＯＬの欠点を認識させる活動が必要である．ＣＯＢＯＬを教えるのではなく，プログラミングを教えるよう学校教育・職業教育を変えていかなければならない．ＣＯＢＯＬプログラマを採用するのではなく，プログラマを採用するよう，人の採用に従事する人の意識を変えていかなければならない．

第２に，代案を求める気運を醸成していかなければならない．データ指向型プログラミングの芽が出てきているので，これに便乗することは，作戦の一つでありうる．

第３に，代案を用意しようとしている動きを孤立させることなく，それらを連携させて，大きな波の形成に向わせなければならない．日立のＥＡＧＬＥや日電のＳＥＡ／１は，その動きの中に数えてよい．日立のＰＡＤ，ＮＴＴのＨＣＰ，富士通のＹＡＣⅡなど，プログラムの表記に図形を持ちこもうとする動きを，利用しない手はない．

第４に，代案づくりのためのテクノロジーの集積を図らなければならない．ＩＢＭのＱＢＥ，ＯＢＥ，またはＳＢＡ，ソードのＰＩＰＳなどに代表される表計算言語や，アプリケーション・プログラム・ジェネレータなどに含まれている技術は，利用できるものとして数えあげてよいであろう．

また，そうした利用可能な技術のレパートリーをふやすために，ビジネス分野の理論的研究を促すことも必要であろう．ＦＡの分野は，ＣＡＤやロボティクスをテーマとする研究者をもっているのに，ビジネス分野には，そうした研究者がきわめて少ない．ソフトウェア工学者を名のる人はいるが，ビジネス・アプリケーションのためのソフトウェア工学には，関心がうすいようにみえる．これは改められるべき風潮である．しかし，現場が期待するような研究者が多数派になるには，まだ時間がかかるであろう．それを待つのではなく，期待にこたえてくれそうな研究者のマップをつくり，交流を深めることあたりから，行動をはじめてはどうであろうか．

３．３．使い捨てツール

ツールには，「成長進化するツール」だけでなく，「使い捨てツール」も必要であり，後者はもっと助長されるべきだという問題提起である．「使い捨てツール」とは目的に特化したツールのことである．ハードウェアの世界では，「使い捨て治具」はごくあたり前の存在である．工作機械と治具を同列には扱っていない．「使い捨てツール」が話題になること自体，ソフトウェア生産の工業化がおくれていることを象徴しているということもできるかもしれない．

「使い捨てツール」の助長をなぜ主張するか．それは，小まわりの効用があるからである．多様な対象への対応を「使い捨てツール」なしに行おうとするのは非現実的だからである．気軽なツール作りを習慣づける必要があると思うからである．この習慣を身につけた人がいい仕事をすることは，現場で日常的に観察される事実である．この人々は常にアンテナを働かせていて，他人のツールをもってきて自分用にテイラリングしている．

「使い捨てツール」を現場に定着させるためには，その効用を説くだけでは不十分である．工夫がいる．仕様の文書化にエネルギーを使わせないのは，一つの工夫である．ＣＯＢＯＬプログラマにツールを作れといっても，作れないということもあるかもしれない．ツールも作れないでプログラマを名のることを許している社会制度を，改めなければならない．たとえば，通産省の情報処理技術者試験の内容を変えるくらいのことは，簡単にできそうに思える．

ソフトウェア事業を営む経営者の中には，ツールは出来合いのものを使うのがよいと，信じこんでいる人が多いように見える．これはまちがいである．ツールは自前で持つのが正しい．こういう経営者ばかりだと，アメリカの製造業のようになる．ソース・コードを提供しないという慣行も困りものである．ソース・コードの提供を促すような法制度の整備が，最低限なされなければならない．プログラマの閉鎖性が，「使い捨てツール」を作る行動を妨げている面もある．これをこわしていくために，ソフトウェア技術者協会があり，シグマ・プロジェクトがあるのだが，それだけで開かれたプログラマ社会ができるほど，ことがらは簡単ではない．

３．４．開発環境の転換

圧倒的なターゲット・マシン支持ムードの中で，開発環境のターゲット・マシンからの分離を，どうやって成功させるかという問題である．ＶＡＸを持ちＵＮＩＸをのせている．もともとは開発環境用に入れたものではないが，経営者は開発環境にも使えと迫ってくる．これは問題の例である．ありそうな例である．事業規模が大き

くなって，開発用のマシンを持てそうな身分になった．ＵＮＩＸマシンを入れるべきか，ターゲット・マシンを入れるべきか．前の例と同様，ありそうな例である．

開発環境のターゲット・マシンからの分離は，理論的に可能であり，かつ分離は現実的に実行可能であることを，証明しなければならない．それを納得させなければならない．その際，いつも話題になる，データベース・アプリケーションをどうするかという問題を，避けてはならない．ビジネス・アプリケーションは，間口は広いが底は浅い．内部ロジックに共通性がある．この命題はターゲット・マシンとは独立に真である．だからターゲット・マシンのメリットは小さい．この言い方は証明の論法の一つでありうる．アプリケーションの特性分析に基づいて，開発環境がターゲット・マシンから分離可能であることを説くものである．そして，何よりも有効な証明方法は，分離を成功させた事例を集めることである．

４．まとめ

いい問題を摘出することができたといえる．３．１節で論じたように，３つの問題は，１つの問題の最重要断片３つを切り出したものである，という見方もできる．まとめられた１つの問題は，産業界がかかえる最重要問題を適切にとり出しえている，と見ることもできる．シグマ・プロジェクトの向っている方向が正しいかどうか，を調べるリトマス試験紙には使える．シグマ・システムらしいものができたときに，それがどの程度役にたつものになりえているか，を調べるのにも使える．

一方で，２日間もかけてごくあたり前の問題しか摘出できなかった，という自覚も必要である．問題があまりに基本的すぎる．それは，産業界の現場の実態が深刻であることを物語っている．

解決行動のヒントを得ることもできたといえる．それらは現場体験から導かれた貴重なものである．しかし，問題解決の手ごたえを得たというには遠い．同じ問題を二度も摘出する経験はしたくはないが，１年たち２年たっても，やはり同じ問題が残っていて，得られる解決のヒントにも進歩がない，ということはありうる．そういう悪い予感を拭うことはできない．

この心情のつづきで，次のようなことを考えはじめている：

５年前にも開発環境による問題解決という思想はあったであろう．われわれがグループ討論から導いた問題も解決対象にあげられていたであろう．５年間という時間の経過がもたらした進歩とは何だろうか．こんな意識をもって５年間の歴史をたどってみたい．

環境分科会（ＳＩＧＥＮＶ）のテーマの１つにしてもよい．

５．結び

本レポートが，一緒に討論してくれた６人の諸兄に，討論の場面を想い起す手がかりを与えるものになっていれば幸いである．

同様の問題意識をもっている人々に，連帯のメッセージとして受けとってもらえると幸いである．環境による問題解決の営みを，それぞれの人がそれぞれの現場で続けられることを願っている．

筆者は，シグマ・プロジェクトというナショナル・プロジェクトに従事する立場から，それぞれの人の解決行動を手伝いたいと思う．

最後に，筆者のコーディネーションに協力くださった６人のグループ・メンバーの方々に深く感謝する．また，今回のワークショップの企画・運営に尽力された岸田孝一，安間文彦，杉田義明の各氏にも謝意を表したい．

お知らせ

前号でお知らせしましたが，長岡ワークショップの成果をまとめた「実践的ソフトウェア開発環境報告書」ができあがりました．参加者全員のポジション・ペーパ，招待講演，グループ討論，パネルの記録等をまとめたもので，約２００ページのボリュームです．

環境の問題に関心をもっておられる方の必読書です．お申込は，ＳＥＡ事務所まで．

販売価格　ＳＥＡ会員：￥３，０００
一　般：　￥５，０００
（送料別）

第１回ＳＥＡワークショップ

実践的ソフトウェア開発環境を考える

グループ討論Ｂ

臼井義美
日本電子計算

１．はじめに

まず，当グループのメンバーを紹介する．

青島　茂　（ＳＲＡ）
臼井　義美　（日本電子計算）
熊谷　章　（パナファコム）
佐久間まゆみ（神戸コンピューターサービス）
桜井　真理　（ＳＲＡ）
酒匂　寛　（ＳＲＡ）
林　香　（ＳＲＡ）

これらのグループ・メンバー各自の主義・主張をアトランダムに書き出すと，次のようなものであった（個人のプライバシーを尊重して，名前は匿名にした．興味ある方は，どの発言がだれのものか，推理されるとよい）．

Ａ：仕事は義務であり，何としてでもやり遂げねばならぬ．
Ｂ：現在の開発環境に満足している．何が不満なのかがよく分らん．
Ｃ：仕事をやるもやらぬも，環境のせいでなく自分の意思しだいである．
Ｄ：仕事とは個人のものでなく，あまねく人類愛から出発しなければならぬ．
Ｅ：環境や仕事がどうであれ，とにかく楽しければそれでよい．
Ｆ：開発環境の整備は得意とするところであるが，いまは混乱状態である．
Ｇ：われこそはどのような混乱でも収束させる，まとめの天才である．

以上のような各自の強力なパーソナリティーと自己主張に直面し，コーディネータとしては一瞬とまどったが，とりあえず，各自の考える問題点を出しあってみて，そのあとで共通の問題点（論点）を洗いだすことにした．このようにして得られた共通の問題点について論議することによって，互いに協力できる部分が発見できるのではないか，と考えたからである．

以下，議論の展開に沿って討議の内容を報告する．

２．現在の開発環境に対する検討

２．１．何に対する環境改善か

まず，何のためのソフトウェア開発環境の改善かを，明確に定義しておくことにした．これは基本的に次の３点に集約された．

（ａ）ＳＥ自身が快適にソフトウェアを開発するための環境
・創造的な活動が便利に行える環境
・やりたいことを楽しくやれる環境
（ｂ）企業が効率良くソフトウェアを開発するための環境
・生産性，信頼性の高い開発環境
・売れるソフトウェア作りができる環境
・現在抱えているバック・ログを解消するための開発環境
（ｃ）社会全体として考えなければならないソフトウェア開発環境
・より安全で便利な社会を構築するための開発環境
・ソフトウェア産業をさらに発展させるための開発環境

２．２．適用業務の現状はどうか．

（１）事務系処理

とにかく要求定義が問題である．つまりシステム要求者が何をいっているのかが分らないことが多い．その大きな原因は，対象となる業務の知識がなければ相手の要求を理解できないからである（たとえば，会計処理における法律，規則，慣習など）．

（２）制御系処理

サイエンス・ベースで話が通じる世界であり，要求が比較的明確であることが多い．

２．３．ソフトウェア開発環境の構成要因は何か？

（1）マネジメント

あらゆるソフトウェアの開発は，なんらかのマネジメントの下で行なわれているといえる．そして，ソフトウェア開発には人間的な要因がきわめて多いため，よいマネージメントが不可欠である．

（2）エンド・ユーザ

一般にソフトウェア開発は，エンド・ユーザが，開発要求者または利用者としてかかわっているため，かれらの要求を満足させることが，当面の目標となる．したがって，要求者・利用者とＳＥ・プログラマとのあいだの相互理解を容易にするための技術が必要である．

（3）ソフトウェア技術者

ソフトウェア技術者自身も環境を構成する一要因である．

（4）開発技術・開発ツール

創造的な開発にも，一般的な業務用システムの開発にも，それぞれに応じた各種の開発技術・開発ツールが必要である．

（5）ワークステーション

高性能文房具としてのワークステーションが必要である．

２．４．実践的開発環境に必要な特性

次に，ソフトウェア開発環境が備えておらねばならない特性にはどのようなものがあるか，という点について論議を行った．

（1）カストマイズの功罪

・カストマイズが常にいい結果をもたらすとは限らない．

（2）ソース・ターゲット・コンセプトへの対応

・開発マシンが有用であるための基本的要件とは何か？

（3）開発作業のおもしろさ

・作業がおもしろいと，開発がはかどるのでは？

（4）ネットワークなどのコミュニケーション機能

・必要な時に必要な情報を入手できる機能の存在は重要である．

（5）オープン・アーキテクチュア

・技術が公開されること，またその情報を容易に得られること．

（6）強力なマシン・パワー

・マシン・パワーが貧困であってはどうしようもない．

２．５．実践的環境構築のための外的要因

（1）標準化の意義

・標準化することのメリット／デメリットは充分検討する余地がある．

（2）エンド・ユーザの役割

・エンド・ユーザはソフトウェア開発作業にどのようにかかわるか？

（3）プログラマのモラール

・プログラムを作る人々のモラールも重要な要因ではないか？

（4）豊富なリソース

・リソースの制約は開発環境を直撃する．

（5）人類愛／隣人愛

・いい環境というものは，人類愛もしくは隣人愛あってこそ可能ではないか？

（6）自己実現欲

・最終的にＳＥ・プログラマ自身の自己満足である．

（7）パブリック・ドメイン

・共同で行う各種の活動も重要ではなかろうか．

３．求められるソフトウェア開発環境とは？

ソフトウェアの開発は，従来のような紙とエンピツによる手工業であってはならない．しかし，ソフトウェア工場のような単なる大量生産方式の導入によって，問題が解決するわけでもない．ソフトウェアの作成は，あくまで技術生産活動もしくは知的生産活動である（ＳＥ・プログラマの立場からは，そうあって欲しいと思う）．

さて，技術生産活動にはどのような環境が必要であろうか．たとえば，大工にいい仕事といい腕といい大工道具が必要であるように，ソフトウェア技術者にもいい仕事といい技術といいワークステーションが備わっておればいいのであろうか．また，知的生産活動の例としての作家に必要な環境としては，静かな部屋と作家としての素質の他には，性能のいいワープロがあればすむ程度であろう．いくら高性能なワープロを使ったからといって，いい小説がかけるわけではないからである．

とすれば，創造的ソフトウェアの開発者にとって本当に必要なものとは，快適な作業環境と，エンジニアとしての資質と，高性能なワークステーションだけではないだろうか．

３．１．快適な作業環境

（1）マネジメント

ソフトウェアの作業環境を左右する大きな要因の一

つとして，マネジメントが挙げられる．つまり，組織としてどのような目標を持ち，どのような方法を用い，何を開発するかによって，ソフトウェアの作業環境は大きく異なってくる．また，マネジャーの仕事に対する考え方や方針によっても，開発環境は大きく左右される．それでは，何がいいマネジメントであるのか．これは，ソフトウェア開発に限ったことでないが，特に知的な生産活動を管理する点では個人のスキルを重視し，技術者のモラールを高め，技術者自身の自己実現欲を満たしてやることが，大切である．

（2）対象業務

開発の対象となる業務によってソフトウェアの開発方法や手順は異なるので，画一的な方法論は適用できない．たとえば，制御系のシステムは比較的数値化しやすく，進捗の管理も行ないやすいが，事務系のシステムは，人間系の作業要因を数値で表現するようなもので，システムの範囲や完成の判定にさいしては，非常に困難が伴うことが多い．また，予測を伴うシステムやツールの開発，デザイン関係のシステムなどは，どこで完成なのかさえ，不明確な場合がある．ここではユーザとソフトウェア技術者の間のコミュニケーションの手段としての要求定義の技術が重要なポイントである．

また，この議論の中で次のような提示があったことを付記しておく．

コメント１：メタ・ツールがあれば，事務系処理開発支援システムの構築は容易ではないだろうか？

コメント２：オブジェクト指向は一見制御系に有効に見えるが，真に威力を発揮するのは事務系ではないだろうか？

（3）コミュニケーション

ソフトウェアの開発にあたって大きな問題の一つは，システム要求者の要望がとらえにくいこと，つまり要求定義のあいまいさが挙げられる．また，実際利用している立場の人達からの変更・補修要求に対応するための，システム保守の困難さも，重要な問題である．さらに，作成されたシステムを関係する人々に理解させるための資料，つまりドキュメントの不備，不明瞭，不正確が問題として指摘される．

このように開発に携る人々のコミュニケーション・テクノロジーが未熟なのである．この問題を解く鍵は強力なモデル化技法にあり，これこそ要求定義の泥沼を抜出すためのキーワードである．

ここで，アコーディオン・フロー（旧称ＳＰ－ＦＬＯＷ）と勝手に名付けた記述法を例にとって，どのような表記技法が役立つのかを具体的に検討した．

要求定義のような高いレベルからプログラム・レベルまでを，単一記法を用いて表現することができれば，要求提出者，システム設計者，プログラマが同じ設計書を見て，その内容を相互確認できる．こういう表記技法が利用できれば，それぞれのレベルにおける記述の違いは詳細度のみであり，問題・要求が拡散しないため，信頼性の高いドキュメントとして使用できるのではないか．

（4）快適さ，おもしろさ

さて，技術者の求める開発環境に共通の要素があるとすれば，ソフトウェアの開発がいかに快適に行えるか，ということであろう．単なる作業環境という点では，個人の趣向によるところが大きい．たとえば，静かな山の中の別荘であってもいいし，音楽がガンガンかかっている地下室でもいいかも知れない．

また，いいアシスタントが備わっていることも，大きな要因である．ここでいうアシスタントは，人間であってもいいし，機械であってもいい．コーヒーを入れてもらうことで快適になれば，それでもいいのである．もちろん，ワークステーション自体に，さまざまなアシスタント機能を持たせることができれば，さらに快適な環境を提供できよう．

３．２．ソフトウェア技術者自身について

（1）適用業務への取組み方

ソフトウェア技術者の側から見ると，かれ／かの女自身が，当面開発すべきソフトウェアに対して「どのようにかかわろうとしているのか？」が，重要である．すなわち，技術向上のためなのか，研究のためなのか，職務のためなのか，金儲けのためなのか，事業化のためなのかによって，対象業務についての取組み方や，開発環境に対する評価が，まったく違ってくるからである．

（2）自己実現

個人にとって最も重要なことは，自己実現欲をどのようにして満たすか，ということであろう．それは，技術者自身の知的生産の根源だからである．

ここでグループ内に突然カルチャーショックが発生した．関東文化圏に属する人びとが，不思議（？）に

も「名誉」なるものを自己実現の大きな要因としてあげたのに対して，関西文化圏の人間は何と「お金」以外に何の価値があろうかといいはなったのである．

喧々囂々の論議の末，「人類愛」でも「金類愛」でもいいから，とにかく自分の求める満足が得られればよかろう，ということでケリがついた．

（3）人類愛

知的作業環境は，ソフトウェア技術者のためだけにあるのではない．ソフトウェアの利用者はもちろん，ハードウェア・メーカや関連する会社，さらには社会全体に対しても，多大な影響を及ぼすものである．したがって，すべての人類への愛情が環境構築のベースとなっていなければならない．この意味からいっても，教養のない技術者がソフトウェア（とくに環境）の開発を行うなどは，とんでもないことであって，それ自体罪悪であるといっても過言ではない．

（4）楽しさ

ソフトウェアが知的生産活動であるとすれば，創造の悦びとともに，活動自体が楽しいものでなければならない．つまり，ソフトウェアの創造主自身にユーモアと遊びの精神がなくて，どうして完成した製品が使う人を喜ばせることができようか？

3．3．開発環境に必要なハードウェア，ツール

（1）スーパー文房具としてのワークステーション

先に述べたように，求められるワークステーションは，便利な機能を持つ単なる機械であっても，充分に開発に役立つものである．だとすれば，現在の技術に多少の改善をくわえるだけで，相当いいモノが提供できるのでないだろうか？

（a）情報の自在な加工が可能なこと

・パーソナルな高機能データベースを使用できること

・高度なグラフィック表示が可能であること

（b）知的コンサルテーション能力

・ある程度の常識と，ある程度のエキスパート機能があること

（c）マルチメディア

・同じ画面に，必要とするさまざまな情報が自由に表示されること

（d）コミュニケーション機能

・必要な情報を自在に収集できること

（e）ユーザ・フレンドリであること

・人間の思考をサポートしてくれる視覚的インターフェイス

・人間の失敗を力の限りサポートしてくれるフェイル・セイフ機能（これはバロースのアーキテクチュアに見るべきものがある）

（2）開発ツール

より知的なソフトウェア開発に当たるために，また，より快適な開発環境を提供するためには，次のような開発ツールがそなわっていれば，一層いいであろう．

（a）プロトタイプ機能

・各種のソフトウェア開発に関して，容易にプロトタイピングができるツール

（b）フォーム設計などの支援

・面倒な作業はできるだけ，機械に任せたい．

（c）システム構造の表記能力の付加

・システムの内容や構造が具体的に表示できるツール．

（e）定型パターンの検索と適合

・部品や過去のソフトウェア資産を自由に再利用できるツール．

（f）さまざまな静的チェック能力

・静的な状態で，できるかぎりのチェックが行えるツール．

以上のような実用的なモデルや技法が有用なツールを導くことになるであろう．

4．終りに

今回のワークショップは，文字通り非常に快適な環境の中で，しかも充分な時間をかけて討論できた．どういうわけか，笑い転げて論議中断に陥ったこともしばしばであったが，自分たちの置かれている開発環境について，このように白紙の状態からじっくり見直してみる機会が得られたことは，非常に有意義であった．

第1回ＳＥＡワークショップ

実践的ソフトウェア開発環境を考える

グループ討論Ｃ

岡村　博

三菱電機コントロールソフトウェア

１．はじめに

当グループのメンバー・リストは次のとおりである（５０音順・敬称略）：

太田靖彦（ＳＲＡ）
岡村　博（三菱電機コントロールソフトウェア）
奥山　充（日立ビジネス機器）
鎌田照彦（日本システム）
佐藤　隆（インテック）
高橋哲夫（ファコム・ハイタック）
中山勝之（東芝エンジニアリング）
西村孝治（神戸コンピューターサービス）

討論は次のようなプロセスで進められた．

（１）メンバーの相互理解：メンバーの立場と背景を明らかにし，それぞれが抱えている問題点を全員がよく理解する．

（２）問題の抽出：各人のプレゼンテーションから問題点を抽出し，一般化する．

（３）討論：問題点についてより深く議論し，解決のためのアイデアを出し合う．

（４）全体討論のテーマ設定：クロージング・パネルでさらに議論したいテーマを設定する．

２．メンバーの相互理解

「メンバー各自が抱えている身近な問題に対して，グループ全員が知恵と経験を出し合うことによって，問題を改善するためのアイデアを得る」という方針にしたがって，各人の問題点を十分に理解するために，それぞれの立場（背景），問題点についてプレゼンテーションを行った．

（１）高橋氏の発表

（ａ）立場

－　事務計算（ＣＯＢＯＬ）を対象としたソフトウェア開発のための支援ツールの開発

（ｂ）問題

構造化ソフトウェア開発システム（ＣＡＮＤＯ）において

－　客先の業務仕様をどのようにＤＢに入れるか？
－　また，そのときのマシンをどうするか？

（２）中山氏の発表

（ａ）立場

－　ＬＳＩ製造のためのＣＡＤ開発

（ｂ）問題

ソフトウェア業界は，技術者不足，生産性の向上，品質，等様々な問題を抱えている．こういった現状から見て：

－　見積もり，設計のための技法・方法やドキュメンテーション技術・品質等に関する，業界としての統一した規準（基準）・標準を設定する必要があるのでは？
－　また，このための提案はＳＥＡで行うべきではないか？
－　さらに，こういった提案はΣに先立って考えておく必要があるのでは？

（３）鎌田氏の発表

（ａ）立場

－　金融端末用ソフトウェア（μＰ）の開発
－　開発はおもにＵＤＩ（パソピア）で行っている

（ｂ）問題

－　ノウハウのデータベース化をどうするか？
－　再利用を促進したいのだが…？

（４）奥山氏の発表

（ａ）立場

－　ソフトウェア製造グループのとりまとめ
－　対象はオフコン用応用プログラム
－　開発方法はパターンを利用したＣＯＢＯＬソースのジェネレーション（パソコン）
－　テスト，ドキュメンテーションはＥＡＧＬＥを

使用

（b）問題

設計と製造を組織的に分離しているが，次のような問題がある．これらをどうしたらよいか？

- 設計サイドの仕様を正確に製造グループに引き継ぐには？
- 古い（経験的，年齢的）設計者はパターンを無視した独自の設計をしてしまう．こういった旧人の教育をどうするか？
- 設計・製造の分離というやりかたはこれでよいのか？

（5）佐藤氏の発表

（a）立場

- ソフトウェアツールの開発

（b）問題

- 異なったマシンで開発しているが，マシン間の言語仕様（C）のちがいに悩まされている
- ライブラリの違いにより，他のマシンで作ったプログラムが流用できない
- また，マシンの設置場所も異なる

（6）太田氏の発表

（a）立場

- 受託ソフトの開発環境の構築・整備
- 客先向けＵＮＩＸツールの開発
- そうした仕事の全体的管理

（b）問題

- 分散型環境について
- ＴＳＳとパーソナルＷＳでの開発方法論の違い等について
- 分野別デザインの方法論について

（7）西村氏の発表

（a）立場

- アプリケーション・プログラムの開発（ＣＯＢＯＬ，ＦＯＲＴＲＡＮ）
- 開発環境の整備

（b）問題

- 要員不足による外注依存（品質）
- 開発環境がない（現在はユーザーマシンを使用）
- コーディングレスのプログラム開発を目指したいが…？

３．問題の抽出

メンバーのプレゼンテーションと，このときに出た質疑から，議論すべきテーマを抽出し，まとめると，おおむね次のようになった．

（1）見積・設計のための技法・方法やドキュメンテーション技術・品質に関して

- ソフトウェア業界の統一基準・標準化は必要か？

（2）テクノロジー・トランスファに関して

- 知識，ノウハウをソフトウェア開発環境に反映させるには？

（3）再利用について

（4）設計と製造の分離について

- 是か否か？

（5）ＵＮＩＸ

（6）マシン間の言語の違いについて

（7）分散型環境（パーソナル環境）で考慮しておく点について

（8）ソフトウェアＣＡＤ

（9）ドキュメント量の増大について

（10）ソース・マシンとターゲット・マシンについて

以上示したテーマに関して，さまざまな角度から議論を繰り返していく中で，新たな問題の提示もあった．メンバーに与えられた課題はこれらの議論の中から有効なアイデアを各人の状況に合わせて選択し，いかにして問題解決に結び付けていくかということである．

４．討論

ここでは，当グループで議論した話題について，時間軸に沿って順を追って報告する．したがって，内容に関しての重複とか，若干テーマと異なるところがあると思うが御容赦願いたい．また，問題に関しては敢えて結論は求めていない．仮に結論が必要な問題であったとしても，それを導き出すのはメンバー各人の役割だ，と思われるからである．

４．１．ソース・マシンとターゲット・マシン

（1）分離すべきか？

- モジュールの再利用等を考えると分離すべきである．開発の過程で作成された様々なプロダクトを次の開発に利用したい．
- ユーザー・サイトからみると，開発専用マシンの購入は不可能な場合が多い．分離するなら，開発者側でマシンを持つ必要がある．
- ターゲット・マシンでも，汎用機にはひととおりの開発環境は揃っている（事務処理：ＣＯＢＯＬ

）．したがって，分離しにくい面もある．
- 否応なしに分離する必要があるものもある（μＰ等）．

（2）分離した場合，テスト環境をどうするか？
- ビジネス・アプリにおけるＤＢ／ＤＣ，またμＰにおけるＩ／Ｏ等を，開発マシンで完全にサポートするのは，現状では無理である．
- シミュレータ等のツールを与えても，テストのための環境設定（テスト・データ，ドライバ／スタブの作成等）が面倒なため，プログラマは使いたがらない．
- しかし，いきなりターゲット・マシンでテストをするより，モジュール・テストを行ったほうが，全体的には効率がよい．
- Σに期待するか？　「期待する，だがアテにはしない」（名言！）

４．２．パーソナル環境／分散環境

（1）１人１端末は必要か？

「端末が遊んでしまうのでは？」
- 多少コストが上がっても，使いたいときに使えるメリットは大きい．
- １人２端末欲しい場合もある（思いついたときにすぐにマシンにかけたい，たとえ家にいたとしても）．

（2）分散環境について

「現在のＴＳＳでは性能の問題は必ずつきまとう．したがって，１人１台パーソナルＷＳということになりそうだが」
- （共通リソースの更新等）リソース管理をどうするか？
- マネージメントは可能か？（進捗等がみえるか？最新ファイルのありかは？）
- 個人が孤立しないか？
- ホストとの作業分担をどうするか？
- ネットワークのトラフィック

等について十分検討しておく必要がある．

（3）ＷＳ／パソコンのシングル・タスクＯＳは？
- なくなっていくだろう．
- ただし，マルチ・ユーザーである必要はない．

４．３．各種技法の標準化

ソフトウェア業界の統一基準・標準化は必要か？
- ＮＯＴＡＴＩＯＮの標準化はあってもよいが….
- 意味があるだろうか？

（この問題に関しては後でもう一度議論した）

４．４．ノウハウのトランスファー

（1）ベテランの作業のやり方のシナリオをＤＢ化・ツール化した例はある（シナリオを作るのが大変）．

（2）ツールを固定する（作業スタイルの標準化）．
- 知らず知らずにノウハウを習得できる．
- しかし，技術的にスキルが平均化される恐れがある．

（以後は標準化について）
- クリエイティブな部分を残した標準化を考えるべきである．
- 生産性を上げるためには，多少の人間的な犠牲は仕方がない．

（3）ノウハウをＤＢに入れるには？
- 難しい．

４．５．ソフトウェアＣＡＥ（ＣＡＤ）
- 設計からプログラミングの自動化を考える（標準パターン，部品化等）．
- このとき問題となるのは，インフォーマルな情報（知識等）をどのようにＤＢ化するかということ．
- ＡＩ的アプローチか？あるいはΣに期待するか？（非常に困難な問題であるが，ある程度固定化した仕様についてはＤＢ化が可能なのでは？そのためには，まず従来のソフトウェアの機能名等の調査・検討が必要だと思われる）
- フォーマル言語
- 絵・図
- 矛盾のチェック，内容に対する検証機能が欲しい．

４．６．設計と製造の分離
- パターン化できるような繰り返し業務では分離すべきである．
- ただし，新しいものを作ろうとするときは不可能（企画物，ツール開発）．

４．７　ドキュメント量の増大

「倉庫に眠ってしまう．」

（1）じゃあ，何のために書くのか？
- 愛着心．契約．
- インプリメンテーションのため（開発用）．
- メンテナンスのため．これが無いから，開発用のドキュメントを捨てることができない．

（2）メンテナンス用ドキュメントは不要か？

「基本的には必要である．では，どうするか？」
- ソースコードにコメントとして入れる．しかし，これだけでは不充分．
- ドキュメント，ソースコード共ＤＢ化する（もちろん図や表も，レーザーディスクは？）．
- あらためてメンテナンス用ドキュメントを作るのは大変だが，開発用ドキュメントをうまく利用できるようなツールを考えるのが，ベストだろう（たとえばＳＲＡのＳＰＥＸ等）．

（３）ドキュメントに関する基準が必要では？
- 書式についての基準はあるが，内容についてのものはないのが現状である．
- しつけまで含めた教育システムが必要なのでは？
- とりあえずはレビュー時に対処する．

「たとえＪ－ＳＴＡＲ等を用いて機械化したとしても」
- ドキュメントのバージョン管理が必要である．

（４）結論（？）
- （今）不要なドキュメントは捨てるべきである（ただし，ゴミ箱へではなくて，レーザーディスクへ：資産は後に再利用できる）．

４．８．再利用

「今は個人レベルでしか再利用していないが，部単位（１０人程度）でうまくやっていくには？」

（１）何を？

「言語はＣとして，ロジック，サブルーチン，スケルトン，仕様というレベルがあるが，サブルーチン・レベルを考えている」
- サブルーチン・レベルではあまり普及しないのでは？
- 汎用性を考えるあまり，部品自体の性能等に問題がでてくるのでは？
- プログラムが部品に縛られる．

いや，縛られてもよい．

（２）部品の収集・登録について
- 大きく組織立って行うと，同じようなものがたくさん集まってしまう．しかも，実際には使えない（職制で部品登録を競争した例）．
- 完全に個人に任せておくと，集まらない（自由にした例）．

（３）部品の管理について

「ＩＮＤＥＸは必要だが，紙でもよいと考えている」
- いや，機械的なツールが必要である（キーワード検索）．
- 保管場所は一箇所にすべきである．

（４）運用・普及について
- ＵＮＩＸ流自然普及？
- 運用体制・部品に対する基準が必要である．
- プロジェクトのスタート時にまず再利用環境を作っておくと比較的うまくいく（再利用できるものの検討・整備等，成功例）．

（５）基準について
- 少なくともテストはしなくてよいもの．
- 制限がはっきりしていること．

（６）ドキュメントの再利用について
- ＳＰＥＸが参考になる（機械化ということを前提として）．
- 契約上の制約で再利用できない場合もある（著作権）．

４．９．知識・ノウハウのＤＢ化と設計支援

（この問題に関しては『難しい』の一言でかたずいた）
- 知識ベース？

（議論の焦点はＤＢ化から，技術移転へと方向が変わった）
- チェックリストを作成する（仕様書に現れないもの）．
- メイルを利用する．
- 人のローテーションを行う．
- 事例集を作成している（例）．

『問題に対してどう解決するか？』というものが欲しい．
- 事例，ユーザー環境，開発者等の情報をＤＢ化する．

「ノウハウと技術の違いは？」
- （あまり意味はないと思うが）ノウハウとは経験から生まれる知識である，ということにしておく．
- 知っている人にとっては，あたりまえのことである．

ということもノウハウが移転されない理由の一つである．

４．１０．基準について（ふたたび）

たとえば，ソフトウェア業界における

－ 品質の基準はあるか？

－ 作業のやり方についての基準はあるか？

－ 生産性のメジャーは？

（ＵＮＩＸの世界のように，プログラマが会社というワクを越えて同一の基盤で話ができるのは非常に望ましいことである）

－ 常識的に「あるといいな」といった程度．

４．１１．その他

（１）外注管理

「設計が異なる」

－ ツール化する．

「外注先にツールを開放したいが」

－ セキュリティの問題がある．

－ パスワードで逃げる．

（２）職場環境

－ 人の配置，机の並び等（話題はアメリカのソフトウェア業界について）．

－ 国民性，人間性の違い．

－ 在宅勤務．

－ 個室．

（３）ＯＢＪＥＣＴ指向（米沢先生の招待講演）

「われわれの開発に利用できるか？」

－ 利用できる分野もあるだろう．ただし，道具立てが必要．

（４）言語

－ ＣＯＢＯＬをＣに変える（困難では？）．

－ ＰＬ／ＭをＣに変える（やりやすい）．

－ 再利用のための言語．

（議論の中で佐藤氏の問題に対して）

－ 分散されている開発マシンをネットワークで結ぶ．

（等のアイデアが出たが，極めて個人的な問題だとわかったため）

「ＦＤＤシートを持ち歩く」ということで一件落着．

（５）ツール

－ 自動テスタ．

－ 工程進捗把握のためのツールが欲しい．

「ツールを開発する場合，方法論を考えるのか？」

－ 方法論から考えたツールはあまりない．

５．クロージングのためのテーマ設定

以上で，メンバー各人の問題についてひととおりの議論を終わったわけだが，メンバーから「是非に」という要望があったため，次の２つを当グループのクロージング・テーマとした．

（１）ソフトウェア開発環境におけるテクノロジー・トランスファ

－ 知識，ノウハウを開発環境に反映するには？

（２）ドキュメント量の増大

－ 膨大な量のドキュメントをどうするか？

これらを含めたクロージング・パネルについての報告はプログラム委員長にお願いすることとする．

６．最後に

「実践的ソフトウェア開発環境」をテーマとした今回のワークショップは，ＳＥＡ発足後初めての大きなイベントということも関係してか，各社において重要な地位にある方々が多く，こころもとないリーダーとしては，ますます不安を抱きながらのスタートであった．しかし，いざフタを開けてみると

－ 技術的に的確なアドバイスをしてくれた‘アドバイザー’

－ 長岡の夜の町を案内してくれた‘案内役’

そして，定年説を乗り切った，あるいは２，３年後には乗り切らなければならないといったグループの中で

－ 新しい感覚で議論に加わってくれた‘若者’

等々「たよりないリーダーを何とかしてやろう」という心やさしいメンバーのみなさんのおかげで，曲がりなりにも無事に終えることができた．心から感謝する次第である．

さて，レポートをまとめるにあたって討論の過程を振り返ってみると，「？」マークがやたらと多いことに気がついた．つまり，われわれソフトウェア技術者にとって開発環境という問題は，いかに重要で，かつ厄介なものであるか，ということをあらためて考えさせられた．今回議論した中に，メンバーにとってひとつでも問題解決の糸口になるものがあれば幸いである．

最後に，すばらしい環境の中で討論する場を与えてくださった関係者，事務局の方々に，メンバー全員を代表して，心から感謝の意を表する．

第１回ＳＥＡワークショップ

実践的ソフトウェア開発環境を考える

グループ討論Ｄ

伏見　論

情報数理研究所

１．討論の視点

今回のワークショップの目的は，ＳＥＡの実務的性格を表現すべく，現場のソフトウェア技術者の視点から見て「開発環境」の構築あるいは改善に関する現状の課題を整理し，方向付けを行うことにあった．

なぜ，「実務的」あるいは「実践的」という修飾語を強調するかといえば，望ましい開発環境を一般的に議論しようとすれば，それは，理想論のさらに先にある純粋な技術的可能性の領域に突入してしまうだろうことが，ほとんど必至であり，特に，今回のプログラム委員会に，その辺での議論を避けたい意向が強かった，ということである．したがって，ここでのグループ討論の目的は「それぞれのエンジニアないし企業の現場における，開発環境の改善をめざした実践的な知恵の獲得」ということに絞った．

ソフトウェア・エンジニアの職業活動を，一方で弁護士のような職業，他方で装置産業のエンジニアのような職業と対比してとらえると，どうなるだろうか．それは，弁護士に比べれば，明らかに設備との関連性が強く，ＳＥＡにおける議論の内容も，単に開発対象のソフトウェアがアプリケーションかベーシックかといった違いだけでなく，それぞれが所属する現場でのハード／ソフトの装備状況を無視して，一律に扱うことはむずかしい．

しかし，だからといって，装置産業におけるように，そもそも相手にしているプラントの種類が違えば共通に話せるのは一般論だけだ，というほどでもないように思える．ＳＥＡの存在理由は，一種微妙なその職業性の中にあるのではないだろうか．

２．グループ討論の出発点

実践的ということをそんな風に考えると，ややＱＣ運動風になって必ずしも個人的には好きでない面もあるが，本来は，同じような「環境」あるいは「現場」から来たエンジニアを１つのグループに集め，その改善方向をさぐるという形で，整理した討論が行われるべきだったのかも知れない．しかし，今回は初回ということもあり，ひとまずは，さまざまな立場からの問題点の持ちよりということになった．

これはこれで，また，ざっくばらんな議論をあれこれ聞いてみるメリットがあって，そう悪いものでもなかったというのが，素直な感想である．

さて，われわれのグループは，次に示すように，バックグラウンドにおいて，まったくまとまりのないメンバーの集まりであった．

伊野　誠（日本ユニバック）
　　メーカ　汎用機　ツール開発
籠谷正樹（日水コン）
　　ユーザ　汎用機　技術計算
栗原正利（ＳＲＡ）
　　ソフトハウス　ＵＮＩＸ　ツール開発
小山孝司（ＩＰＡ）
　　ソフトハウス　マイコン　ツール開発
佐藤千明（長野県共同電算）
　　ユーザ　汎用機　事務計算
玉本和明（神戸コンピュータ・サービス）
　　ソフトハウス　汎用機　事務計算
馬場充彦（日本システムサイエンス）
　　ソフトハウス　汎用機　ツール開発
伏見　論（情報数理研究所）
　　ソフトハウス　ミニコン　技術計算

したがって，グループ・コーディネータとしては，「むしろ一般論を楽しくやった方がよいのか」，「あまり議論のかみ合いそうにない実務的議論を押し通すか」，等々のオルタナティブを前にして，いささか迷ったのであるが，一応，初期の方針通りまとめさせていただくこととした．

３．グループ討論の過程

(a) 2月20日

まず午前中に，参加メンバー全員が，自己紹介をかねて，簡単な問題提起をおこなった．「ワークステーションあるいは小規模な専用マシン上にどうやって開発環境を整備してゆくか」という立場と，「汎用大型機上のそれなりに現存する環境をどうやって引き上げてゆくか」という立場の２つに大きく分かれたので，それらを別々に議論することにした．

午後は，主として，前者のカテゴリに属する問題を討論した．ツールをうまく使いこなす体制をどうやってつくるか，技術計算における標準化の進め方，事務処理やＣＡＤにおけるＤＢ環境と開発マシンとの関係，ソフトウェア部品ＤＢの使用上の問題点などが討論された．

途中，安間プログラム委員長が飛び入りで参加され，シグマ・プロジェクトにおける事務処理アプリケーション分野への対応の方向について，一通り説明された．また，今回のワークショップの招待講演者の１人である栗原氏から，ＳＲＡにおける過去５年間のＵＮＩＸ利用経験にもとづく，環境構築事例のくわしい説明があった．

(b) 2月21日

メインフレーム上での開発管理の話題，メインフレームとワークステーションとの接続や両者の機能分担をとりあげて，討論した．話題提供は，佐藤，玉本，伊野の３氏．

佐藤氏からは，自社の環境およびこれまでおこなってきた改善について詳細な説明があり（ポジション・ペーパ参照），今後のワークステーション導入戦略について討議した．次いで，玉本氏から，ＪＳＴＡＲによるドキュメントの機械化についてお話しいただき，ソフトウェア資産の再利用や，ネットワークの将来の活用法などを議論した．

最後に，伊野氏から，メインフレームとワークステーションとを組み合わせたソフトウェア設計支援システムの事例紹介があり，日本語，グラフィクス，イメージ処理など，ソフトウェア設計の作業からくるワークステーションへの機能的要求について，活発な議論が展開された．

４．グループ討論のまとめ

討論のまとめとしては，コーディネータにもいろいろ後から付け加えたいことはあるのだが，佐藤氏が最終日の発表のために整理したにもかかわらず，時間の関係で使わずじまいに終わったＯＨＰの原稿があるので，それをほとんどそのまま引用させていただく．

４．１　未解決の問題点

（１）伊野（大型汎用・ツール）

・設計書の機械化支援環境を安く作りたい．
　－　ＪＳＴＡＲなみのグラフィクス機能がほしい．
　－　Ｃでは大変なので，Smalltalkを使いたい．
　－　シグマＷＳでぜひSmalltalkをサポートしてほしい．

・設計書の管理をどうしたらよいか．
　－　ＷＳとホストの機能分担は？

（２）籠谷（大型汎用・技術計算）

・オープン・プログラマの技術標準化．
　－　まったくの放任はよくない．
　－　動機づけ．
　－　ツールの提供．
　－　１人ずつまきこむ（？）．

・現在ターゲット・マシン１台で，テスト環境が劣悪．
　－　リソース不足をどうやってトップに理解させるか．

（３）小山（マイコン・ツール開発）

・４ or ８ビット・マイコン用のプログラミング環境整備．
　－　ＵＮＩＸがまだ使いこなせていない．
　－　環境専門スタッフが必要だ．
　－　シグマに期待するところが大きい．

（４）佐藤（大型汎用・事務計算）

・設計支援ツールをなんとかしたい．
　－　マイクロ・メインフレーム結合をどうするか．
　－　設計情報の入力をどうするか．
　－　プログラムの自動合成へのつなぎは？

・データ・ディクショナリ（！？）．
　－　その構築は可能か．
　－　実用に耐えうるか．

（５）玉本（大型汎用・事務計算）

・受託ソフトの開発経験の再利用を図りたい．
　－　著作権の問題．
　－　ＪＳＴＡＲ上の設計情報をどうやってプログラム作成につなぐか．

・これからのネットワーク．
　－　在宅勤務：その管理，セキュリティ，など．

（６）馬場（大型汎用・ツール開発）

・ＪＡＳＭＡＣの開発支援環境．

－ 部品ライブラリの更新とソースへの反映．

（7）伏見（ミニコン・技術計算）

・派遣作業における諸問題．

－ 自社ツールが使えない．

－ 客先にツールが整備されていない．

－ 端末の台数がすくない．

・ミニコンと客先マシンとの接続．

－ セキュリティの問題．

－ 接続期間の長さ．

－ 費用負担．

－ 環境の変化への人間的抵抗．

4．2 なにが必要か

（1）大型汎用ユーザの要求

現状では，ＴＳＳ上に，プログラミング・テスティング・メインテナンスのための環境が，メーカ提供のツールおよび自社開発のツールを組み合わせた形で，ある程度ととのっている．しかし，要求分析・定義ツールの不在，設計機械化ツールの不足，ＴＳＳ端末のユーザ・インタフェイスの悪さなど，いくつかの不満もある．

こうした現状の問題点を解決するためには，日本語処理，図形処理，イメージ処理などの機能を持った専用のワークステーションを用意し，それらをメインフレームと組み合わせて利用するのがよい．そのさいに考えるべきことがらは，以下の通りである．

・ＷＳ上のグラフィクス・ソフトをどう作るか．

・設計情報の管理をどこでどう行うか．

・ＷＳ上のコマンドとホストのコマンドとの切り換えは？

・テストはどんな風にするか．

（2）ソフトウェア・ハウスの要求

ようやく「紙と鉛筆」の時代からぬけだして，自社内に専用の開発設備を持つところが多くなってきた．しかし，自社内のマシンで開発の全工程をカバーすることはむずかしく，なんらかのかたちで，客先のマシンを使わざるをえない．自社用のマシンとしては，ミニコンやワークステーションが多い．

現状における問題点は，まず，自社内の開発環境と客先の環境とのくいちがいである．また，一般的にいって，ツールの整備・普及が遅れている．さらに，受託開発ソフト（およびその開発ノウハウ）の蓄積・再利用が十分できていない．

こうした問題点を解決するためには，社内の環境をハード・ソフトの両面から強化するとともに，それをなんらかのかたちで客先の環境と接続し，エンジニアたちが，社内ですべての作業ができるようにすることが必要である．

環境の接続には，いろいろなかたちが考えられる（シグマ・ネットワークもその１つである）．その具体化には，セキュリティ対策，経済性評価など，いくつかのことがらが，もうすこし詳細に詰められなければならない．

5．蛇足としての追加

ソフトウェアの生産性の飛躍的向上ということがたとえばシグマ計画の目標であるわけだが，われわれの討論の中では，なかなか，そのような意味での飛躍が見えてこなかった．ワークステーションを基礎とする環境が，単なる抽象概念としてではなく，物理的な装置そのものとして生産性向上に貢献しそうなことは，予測の範囲内にある．

また，現在ソフトウェア・エンジニアに課せられているいくつかの不合理な労働を軽減するものとして，ネットワークが有効であることも，どうやら事実らしい．しかし，そのインパクトは，たとえばＣＡＤワークステーションが一般のエンジニアリングに与えるものに比べれば，全体として，弱いという印象をぬぐえない．

ソフトウェア・ハウスの立場から見て，新しい環境の構築による生産性の飛躍的な向上が十分確実であれば，企業間競争に打ち勝つためにも，多くの会社が「自社内に」重装備をするという方向へ動かざるをえないわけだが，そのことは，現状ではまだあまり明確ではない．

しかし，社会全体としては，ソフトウェア開発の生産性が非常に低いと見ることが一般的になっているので，そうしたいわゆる「ソフトウェア危機」の認識は，たとえばエンドユーザからソフトウェア・ハウスへの発注のメカニズムも含めた総合的な問題として，もっとポイントを明確にしておかなければならないように思える。

ただ，危機の克服へ向けての戦略のたて方が，政策論的なものか，または純エンジニアリング的なものか，の二者択一ではよくないように思う．

第１回ＳＥＡワークショップ

実践的ソフトウェア開発環境を考える

クロージング・パネル

安間文彦

情報処理振興事業協会

１．はじめに

安間： 今回のワークショップの最後のプログラムであるパネル討論をはじめます．最初のオリエンテーションでお話したように，ここでは，各グループの討論の中で未解決の，あるいは積み残したテーマについて発表していただき，参加者全員で，それらの問題を考えていきたいと思います．

２．グループからの課題提案

松原友夫（日立ソフトウェア・エンジニアリング）：
われわれのグループの特徴は，各社の実力者が集まったことだと思います．そうしたメンバーにふさわしく，十分に実のある討議がおこわれました．とくに，経営者をいかに口説くかとか，あるいは金を出させるにはどうしたらよいか，といったことがらについて，実際にいま，そうしたテーマを抱えている方がおられて，現実的な生活の知恵が話題になりました（笑）．

このグループでの未解決な問題は３つあります．

１つは，環境の中でＣＯＢＯＬをどう扱うべきかということです．この問題へのアプローチには，ＣＯＢＯＬ開発支援環境を充実させること，ＣＯＢＯＬを環境の内部に封じ込め他の言語で包んでしまうこと，ＣＯＢＯＬそのものを否定してしまうこと，の３通りがかんがえられます．

２番めの問題は，使い捨てツールの是非とその扱い方です．ツールには，その場限りの使い捨てツールと，将来に渡って成長・進化してゆくものとがあります．使い捨てツールは，目的別に特化したもので，文書化などに余分なエネルギーを使う必要はありません．このことは，ハードウエアの世界では常識になっています．

最後の問題は，開発環境を実機から分離するための，方策または仕掛けについてです．ＶＡＸが与えられてしまって，何とか開発環境を構築し，ソフトウェア開発の転換を迫られている方と，逆に，圧倒的な実機支持ムードの中で，メインフレームを入れるべきか，ＵＮＩＸマシンにするべきかを，悩んでいる方とがいました．いずれにせよ，開発環境と実機の分離をどうやって成功させるかという問題に対しては，まだ模範解答が見つかっていません．

臼井義美（日本電子計算）： われわれのグループでは，メンバー全員に共通の問題（論点）はあるのか，もしあるならばそれを論じたい．そのことについてお互いに協力しようというアプローチを最初に確立し，それにしたがって討論をすすめました．

まず，開発環境とは，マネジメント，ソフトウェア要求者，ソフトウェア利用者，ＯＳおよびツール，ハードウエア，さまざまな開発技法，などの要素から成り立ち，ＳＥやプログラマがそれらを適当に組み合わせて仕事をする場を指す，ということで合意に達しました．

それから，途中経過はいろいろあって，なぜか笑いすぎてお腹が痛くなった人も出たりしましたが，結論としては，よいＯＳとよいツールがすてきなワークステーションの上にのっていて，ＳＥが快適な仕事部屋のなかでそのワークステーションを使えるようになりさえすれば，自動的によい仕事ができるということになりました（笑）．そうしたワークステーションを「スーパー文房具（Super Stationary：略称Ｓ＊Ｓ）」と呼びますが，そこには，情報の自在な加工，知的コンサルテーション，マルチメディア通信などの高度な機能がふんだんに用意されており，ユーザ・フレンドリネスの観点からいえば，完全なフェイル・セイフ機能があって，人間の失敗を力の限り支援してくれるはずです．

このような討論のプロセスおよび結論からいって，われわれのグループには，未解決の問題は残らなかったことになります（一同爆笑）．

岡村博（三菱コントロール・ソフトウェア）： われわれのグループでの討論の経過は，およそ次の通りです．

まず，開発マシンとターゲット・マシンを分離すべきか否か，その場合のテスト環境をどうするか，といったことを議論しました．次に，１人１端末は必要か，パーソナルな環境とは何か，ワークステーションＯＳのあるべき姿，設計手法を統一することの必要性，ノウハウのトランスファー，ソフトウェアＣＡＤ，設計と製造の分離，作業環境などの問題について意見を交換しました．

そのあと，ドキュメント量の増大ということをめぐって，何のために書くのか，メンテナンスにドキュメントは必要か，ドキュメンテーション基準の問題，ドキュメントの機械化と管理といったことがらを話しあいました．設計支援のためのノウハウ／知識のＤＢ化や再利用についても，その体制・管理・運用・基準・普及など，さまざまな側面を議論しました．

ここで，もう少し突っこんで討論したいテーマは次の２つです．１つはテクノロジー・トランスファの問題，つまり，知識・ノウハウ・技術をいかにして環境に取り入れるかということであり，もう１つは，ソフトウェア開発において増大する一方のドキュメント対策です．

佐藤千明（長野県協同電算）：　われわれのグループでは，メンバーの関心が，ワークステーション上の環境整備と大型機上の環境改善の２つに分かれたので，それらを１日目と２日目に分けて順番に討論しました．

ワークステーション上での環境構築については，ツールを使いこなす体制をどうやって作るか，技術計算における標準化の進め方，事務処理とくにＤＢアプリケーションと開発マシンとの関係，ソフトウェア部品データベースの使用上の問題点などが，議論されました．２日目は，すでにある程度の環境をそなえているメインフレーム・マシンを中心として，それとワークステーションとをうまく組み合わせて環境を整備するにはどうすればよいかが，主要な話題になりました．

それらの中で，未解決に終った問題は，アプリケーション・ソフトウェアのモジュール化と再利用，および設計ドキュメントの機械化に関することがらです．後者については，玉本さん（神戸コンピュータ）のＪＳＴＡＲでの経験や，伊野さんのところ（日本ユニバック）でやられている「ジャクソン法」にもとづく設計支援やドキュメントの機械化がたいへん参考になりましたが，まだ結論が出てはいません．

安間：　ありがとうございました．それでは全体討論に入りたいと思います．まず，岡村さんのグループから提起されたテクノロジ・トランスファの問題について討論をはじめましょう．少し補足説明をお願いします．

３．テクノロジ・トランスファについて

奥山充（日立ビジネス機器）：　会社のなかには，新しい人，古い人，多くの経験を持った人，古いセンスだけの人，新しい技術動向に敏感な人など，さまざまな人がいますが，それらの人々のあいだの技術や経験の交流は，ＯＪＴという美名のもとに放任されているのが現状です．開発のノウハウや経験をいわゆる「知識ベース」にまとめるには，どのようにするのが有効か，そのシナリオを考えてみたいというのが，そもそもの問題意識です．

組織的な仕掛けをどこにどのようにつくるか，そうした組織ができたとして，現実の運用をどうしたらよいか，などをグループで話しあいました．

自分固有の技術やノウハウを最良のものと信じ，新しい技術や開発環境の変化を受け入れたがらない技術者が，どこの企業にもいると思いますが，このようなタイプの技術者をどう扱ったらよいか，みなさんのご意見をうかがいたいと思います．

臼井：　テクノロジ・トランスファをうまくやるには，それによって得られる利益を明確にすること，もっと端的にいえば，自分の技術を他人に教えた人が「トクをする」状況を作り出すことが大切です．人間は欲望の動物ですから，すべての関係者がそれぞれの欲求をみたせるような方向に向かっていればＯＫで，そういう環境にする必要があります．

「トクをする」かどうかという判断は，価値観の違いによって大きく左右されると思いますが，われわれのグループでも，東京人と大阪人とがそのへんで鋭く対決しました（笑）．

大阪的感覚では，金銭的欲望を満たす組織作りが大事で，たとえば，よく議論される「製造と設計の分離」についても，大阪の場合，新人の段階から金で縛ることによって，分離が比較的スムースに進むことになります．東京の場合とはちがって，名誉なんてものはどうでもよいのです（一同爆笑）．

安間：　古いタイプのＳＥをどうしたらよいかという質問に対して，どなたか経験を教えてください．どこの職場でもいろいろ苦労をかさねていると思いますが，ＳＲＡの場合，そのへんをうまくやっていらっしゃるという噂を聞きましたが，どうなのでしょうか，杉田さん？

杉田義明（ソフトウェア・リサーチ・アソシエイツ）：

あまりうまくいってないというのが，正直な現状だと思います．

私の個人的な意見をいわせていただくと，古いタイプの人は無視するのが一番です．新しい技術を導入したり，新しい環境を使って仕事をしてみようとする場合，今までの経験からいって，新しい人たちを中心に進めるのが効率的だと思います．

ＳＲＡでは１９８０年にＵＮＩＸを導入し，これまでツールや環境の整備をおこなってきましたが，その主たる原動力は若い人たちで，かれらに対しては，新入社員のときから，全面的にＵＮＩＸを取り入れた教育をおこなってきました．最初は，古い人にもＵＮＩＸの教育をしましたが，その効果はほとんどみられませんでした，そのような人には，無視する態度をとることが必要だと思います．

三浦信之（日本コンピュータ・システム）：　私も社内で教育に関係していますが，ほとんど杉田さんと同じ意見でして，何とか若い人たちを盛り上げていきたいと，いつも苦心しています．

松原：　古いタイプの人は，金（予算）を持っていて，口もうるさいが，自分で手を動かすことをしない．もしそうであれば，なんとかごまかして手を動かさせるようにしたほうがよい．ワープロなり端末なりを，そういう人たちにも１台ずつ与えて，どうしてもそれを使わなければ仕事がうまくいかないような環境を作ってしまうのがよいと思います．

臼井：　私の会社ではコンピュータの入れ換えがあって，これまでの古いがエレガントな環境の一部が，新しいがダサイ国産汎用機のそれに置き換えられました．しかも，新人は後者の環境で仕事をするように強制されています．私たち古い世代は，高性能なＯＳとコンパイラを統合したエレガントな環境に慣れ親しんでいるので，最近の新人はまったく原始人にしか見えない．技術の善し悪しや新旧は，年齢とは関係なく，教育や仕事の環境が重要だということです．

安間：　技術移転の問題はここまでにして，次のテーマに移りたいと思います．シグマ・プロジェクトのなかでも，最近ソフトウェアのドキュメンテーションが話題になっていますが，先ほどこの問題を提起された岡村さんのグループから，ひとこと補足説明をお願いします．

４．ドキュメントの機械化

佐藤隆（インテック）：　以前あるナショナル・プロジェクトで，安間さんが発注者，私が受注者側の立場で仕事をしたとき，ドキュメンテーションについて大いに疑問を持ったのが，この話題のそもそもの発端です（笑）．

なにが問題かといえば，詳細に定められたドキュメント作成基準にしたがって仕事をしてゆくと，開発工程のなかでドキュメンテーションの作業量が増大すること，そしてまた，成果物を保存しておくのに膨大なスペースが必要になること，などです，さらに，そうやって苦労して作ったドキュメントが，実際にはあまり使われません．にもかかわらず，見もしないし利用もしないドキュメントを欲しがるおかしなユーザがいる（笑）．ドキュメントの書き方はいろいろ決まっているが，捨て方はどこにも書いてない．ここで私的な反省をすると，この議論をするために，５枚もＯＨＰでムダなドキュメントを書いてしまいまった（一同爆笑）．

それでは，なぜ書くのかといえば，ソフトウェアを作るため，あるいは直すためです．この２つの目的は一致していません．それらを１つのドキュメントで間にあわせようとすると，量が多くなり，いろいろな情報が混在しているために，読むのが大変であり，そのうちだんだん読まれなくなる，ということになってしまう．

解決案として，メンテナンス・ドキュメントとしてはソース・コードを利用するようにしたらよいと考えます．そのためには，ソース・コードにメンテナンス用の情報を付け加えてデータベース化し，保守用ドキュメントの自動化を図るというのは，どうでしょうか？

安間：　佐藤さんのとらえているドキュメントは，紙に書くことを前提としていますか？

佐藤（隆）：　はい，そうです．

佐藤（千）：　私はこれまで，ステップチャートを書かずにプログラミングする方法はないかと，いろいろと試みてきました．なぜチャートを書いてきたかを分析し，書かないとすれば開発のスタイルはどうなるか，そしてその場合の保守を容易にする環境を考えてきました．

臼井：　ドキュメントが必要である理由は何か．いまのソース・コードでは，どんなソフトウェアなのかが要求者によく分からないから，ドキュメントを書いているのではないか．

それなら発想を逆にして，ユーザに分かるものを作ればドキュメントがいらなくなるはずだ．要求定義から設計，プログラミングまでのすべての結果を同一のフォーマットで扱えないか．ユーザ要求をそのまま残すことが

できれば，ユーザにとって見れば，自分のいったことがそのまま実現されているということになる．それで，プログラマ・ＳＥとユーザとのあいだのコミニュケーションがよくなると思われます．

私は自分でそのようなドキュメントの一部を実現しました．名付けてアコーディオン・フローと呼んでるもので，ポータブルで必要な時必要な場所を検索・削除することができるようになっています．サンプルを配りますのでご覧ください．

青島茂（ソフトウェア・リサーチ・アソシエイツ）：
生の情報をそのまま全部書こうとするから，ドキュメントの量が多くなる．データベース化といっても，手書きのドキュメントをそのままコンピュータに入れると，やはり量が多すぎて，検索も利用もままならなくなる．データベースの構成をもっと知的なものにし，スペースの削減を考えることが必要だとおもいます．

岸田孝一（ソフトウェア・リサーチ・アソシエイツ）：
ふつう，ドキュメントというと「もの」についての記述，成果物についての議論におわることが多いのですが．問題は，途中の開発プロセスをどうやって記録するかということではないでしょうか．われわれの仕事の成果物としてのソフトウェアが，最終的にいまのような形になる過程で，だれが，いつ，どのような意志決定をくだしたのか，それはなぜか，この重要な情報が抜けているから，ドキュメントが完全なものにならない．

完成したソフトウェアの静的な姿だけを記述するのではなく，開発プロセス全体をカバーする時間的な観点が必要になる．その意味で，たとえばバルザーなどが現在試作している新しいソフトウェア・データベース・モデルには，時間的概念が含まれていて，おもしろいと思う．

従来からあるような，完成したモジュールをそのまま記述しただけのドキュメントは，これまでの議論でわかるように，役に立たないのではないでしょうか．

盛田政敏（神戸コンピューターサービス）：　でも，そうだとすると，ドキュメントの量が現在よりずっと増加して，インプットがたいへんな話になってしまう．

岸田：　そうです．そのためにも，高性能のワークステーションによって，データの収集を自動化することが必要なわけです．

杉田：　今後ドキュメントの量はもっと増えるでしょう．また，開発や保守の過程で発生する間違った情報や，エラーについても，残しておいた方がよいと思われます．そうした情報は，プログラムを理解するさいにきわめて有用です．

樋口勝久（管理工学研究所）：　私は，マルチウインドウつきのパソコンを，いつも使っています．ドキュメントすべきことを思いついたらすぐ書くことが大切だと思います．パソコンにどんどん入れておけば，あとでそれをまとめてドキュメントにするのは，比較的簡単です．

臼井：　われわれの仕事は，本質的に「ものかき」の一種ですから，ワードプロセッサは必須です，先ほどのアコーディオンを，もっと伸縮自在にするためにも．

松原：　たしかに，ワープロが１人に１台普及するだけで，文化的にはきわめて大きな変化が起こるでしょう．そうしたパーソナルな環境への切り替えを促進するにはどうしたらよいか．そのことをもっと真剣に考えるべきでしょう．

安間：　たしかに，要求分析・定義の段階で，ドキュメントをどのように残すか，それらの工程で発生するインフォーマルなメモ類の資料をどうやってうまく生かすか，といったことの検討が必要だと思います．

佐藤（千）：　さきほど話のあったドキュメントの捨て方について，少し議論しましょう．メンテのことだけを考えれば，内部仕様書は捨ててもかまわない．とっておいても全然使えないからです．しかし，たしかに，ドキュメント基準には捨て方が決められていない．

中山勝之（東芝エンジニアリング）：　ソフトウェア開発の契約書に，ドキュメントの捨て方をきちんと決めて入れたらどうですか．　そのためには，ドキュメンテーション基準をきちんと作っておく必要があります．ただし，マイクロプロセッサ関係のアプリケーションでは，そのシステムが動いている限り，ドキュメントは残すというのが鉄則です．

岸田：　開発プロセスの記述を含むドキュメントは，絶対に捨ててはいけないと思います．ボリュームが問題なら，光ディスクかなにかを使えばよい．

酒匂寛（ソフトウェア・リサーチ・アソシエイツ）：
なぜドキュメントが増えるかということですが，１つには，ドキュメント基準にもとづく余分なフォーマット情報を一緒に入れるからではないでしょうか．もちろん，ワープロの場合はそうせざるをえないのですが，コンピュータを使えば，必要最小限の情報だけを機械の中に残し，人間が読むドキュメントは，その都度，適当なフォーマットを使って作りだすことが可能になる．実際，あ

るプロジェクトでそうした試みをやってみたのです，非常にうまくいきました．問題は，ソフトウェア開発用のデータベースをどう作るかです．

安間： アメリカのあるソフトウェア会社では，ドキュメントはコーディングしながら記述していくようになっている．すなわち，ソース・コードの中にコメントとして打ち込んで行く．コメントの書き方にある規約があって，あとでツールを用いてそれを取り出し，ドキュメントが自動的に作られるようになっている．これは技術計算分野のツールの例です．

５．ターゲットと開発マシンの分離

岡村： ターゲット上でのデータベースをどう扱うかが，きわめて大きな問題だと思います．結論的には，ＤＢアプリケーションのテストを開発マシン上ですべてやることには無理があり，ターゲットと開発用ワークステーションとを組み合わせたテスト環境が必要になるでしょう．

盛田： 開発マシンとして，実機（ターゲット）と同じ機械を持つメリットは，これまでの技術蓄積がそのまま利用できること，作業環境の同一性または連続性，ユーザとの一体感，また特にターゲットが汎用大型機の場合にはそれが一種のステータス・シンボルになること，などがあげられます．

一方，ＵＮＩＸに代表される開発専用マシンを導入することのメリットは，ソフトウェア・ハウスとして特定のハードウェアから独立できること，ツールの整備・活用が長い目で見ればやりやすいこと，などだと思います．しかし，長いことターゲット環境で仕事をしてきていると，なかなか新たにＵＮＩＸの導入には踏み切れない．

熊谷章（パナファコム）： コンピュータ発展の系譜をたどってみると，汎用コンピュータから，ミニコン，パソコンと進み，今後はネットワークで相互に連結されたワークステーションの時代に移ろうとしています．

ワークステーションに要求される機能には，２つの側面があると思います．１つは社会的な要求であり，たとえば，国際的電子コミュニティへの参加，マルチメディアの融合化，インターネットワーキング，低価格化，機能と操作の異機種間共通化などが，それにあたります．もう１つ，個人的な要求としては，より快適な作業環境，環境の標準化よりはカスタマイゼーション，ＤＩＹ，カウンセリングその他の知的支援などがあり，マルチメディアのスムーズな利用，知的なドキュメンテーション・サポート，使い込めば使い込むほどよくなるシステムなどが，具体的に望まれています．

今後，国際的ネットワーク網の整備など社会的環境の変化にともなって，電子メール，ファイル転送，リモートログイン，電子掲示板，ニュース，電子会議，ネーム・サーバなどの機能をそなえた電子コミュニティは，確実に広がってゆくでしょう．そして，ネットワークへの接続プラグとしてのワークステーションの上では，オブジェクト指向，動的記号処理，論理表現，フレーム表現などの新しいパラダイムが主流になるでしょう．

しかし，私自身は，そうした技術的なことがらよりも，むしろ，人間性中心主義，不易流行，マイノリティ重視といった思想的な側面のほうが大切だと思います．

６．ＣＯＢＯＬをどうするか？

安間： 時間がなくなってきたので，ワークステーションの議論はそれくらいにして次のテーマに移りましょう．松原さんから提案のあったＣＯＢＯＬをどう扱うかということについて，皆さんの意見をお願いします．

臼井： 問題は，言語としてのＣＯＢＯＬのよしあしではなく，多くの汎用機上でのプログラミング環境があまりにもヒドすぎるということでしょう．

私はバロースのＣＯＢＯＬをかなり長い間使っていましたが，きわめて快適でした．ところが，現在のＩＢＭ系統のＣＯＢＯＬ開発環境は，とても人間が使うようには設計されていない．簡単なプログラムを動かすために恐ろしいＪＣＬを書かなければならなかったり，ちょっとした間違いでシステムがすぐアベンドしたりといった話が日常茶飯事です．

こんなことは，バロースのシステムでは考えられません．コンピュータは，自分ができる最大限の努力をして，どうしてもダメな時に，さらに人間の指示を求めるという思想でシステムが作られているのです．ＵＮＩＸがいま，もてはやされているのは，同じような人間味を持っているからでしょう．もともと，２つのシステムは，同じルーツ（ＭＵＬＴＩＣＳ）から出た異母兄弟みたいなものですから．

松原： ＣＯＢＯＬで育った人は，他の言語で育った人とは違う「文化的」雰囲気を持っているような気がしてならない．機能的にＣＯＢＯＬ言語が融通がきかないので，それを使っている人もだんだん融通がきかなくなる（笑）．

臼井： いや，ＣＯＢＯＬはそれほどヒドイ言語じゃない．うまく使えば，ＣＯＢＯＬで書いたプログラムは，

他の人が読んで理解しやすい．言語のレベルと人間の知性とは直接関係はないと思います．

佐藤（千）： ＣＯＢＯＬやＦＯＲＴＲＡＮが現在広く使われている理由としては，これまでに蓄積された資産が捨てられない，ということが大きいと思います．つまり経済性を無視できないからです．たとえ，いくらよい言語だからといって，自然に広まるというわけではない．

松原： そうやって，いわば既成事実によってＣＯＢＯＬへの依存を続けると，自然にそれに規制されてしまうということは事実です．いつまでもそれでよいかどうかは，大きな問題だと思います．

熊谷： 現実に世の中の大規模なアプリケーション・システムでは，ずいぶんヒドイ言語がいまだに使われている．言語に対するすききらいやよしあしを議論するのはナンセンスであり，ＣＯＢＯＬだけしか知らない人には他の言語を教えて，それが選べるような環境を設定するのが大事で，もしやっていないとすれば，マネジメントの怠慢である．よくいわれるではないか，恋はフランス語，仕事はドイツ語と（一同爆笑）．

酒匂： 言語の完成度を議論してもしかたありません．重要なのは，上のレベルでどう扱うかということです．ＣＯＢＯＬにはファイル処理やデータ項目の扱いで便利な機能がたしかに存在しますが，所詮は事務用のアセンブラにすぎません．上のレベルでそれをカバーする仕掛けを考えることが必要です．

松原： 事務処理アプリケーションそのものにもっと光をあてなければなりません．この分野に取り組んでいる学者が少ないのは残念です．

佐藤（隆）： 私は，いまのガチガチのＣＯＢＯＬで十分だと思います．酒匂さんのいうように，プレプロセッサやなにかを工夫すれば，なんとかなる．

臼井： 私が経験したフレキシブルな開発・オペレーション環境では，ＣＯＢＯＬでほとんどすべてのことが可能であり，しかも便利だった．

７．最後に

安間： 「使いすてツール」の問題，その他まだまだ議論すべき問題がたくさん残っていますが，残念ながら時間がなくなりました．どうも，活発な討論をありがとうございました．

私は，いまシグマ開発本部で，いろいろと環境開発の計画を検討しているのですが，今回のワークショップの討論は，自分自身の問題意識と重複するところが多く，非常に参考になりました．このような議論を聞いてみてシグマがなにを目指すべきか，その実現のプロセスはいかにあるべきか，などを探る上で，ヒントになることがたくさんありました．とくに，ＣＯＢＯＬ環境については，一通りの結論が出たのではないかと思います．

新しい開発環境の構築は，新しい人々を中心として，新しい形態で進めなければならないということを，なお一層実感しました．

最後に，実行委員長の岸田さんから，一言．

岸田： ４日間活発な討論お疲れさまでした．今日，朝食のテーブルで，三菱総合研究所の反町さんからうかがったひとつの恐ろしいデータがありますので，ご紹介します．

昭和５５年から５９年までの５年間におけるソフトウェア開発の生産性の変化を，ソフトウェア業界に働く技術者の１人当たり年間売上げ高で見ると，昭和５５年が６２２万円，昭和５８年は６３６万円，昭和５９年（去年）は７０２万円という数字です．名目の伸び率は年間３．５％でしかなく，実質成長率はわずか１．１％にしかならない．これは，構造不況産業といわれている繊維やアルミ業界以下です．

一方で，ソフトウェアのマーケット自体は急速に拡大しています．そうした需要の増加に対応するためには，今回のワークショップのテーマである開発環境の改善を早急におしすすめて，生産性の向上に努めなければなりません．このことは，単に経営者だけにまかせておけばすむ話ではなく，われわれ技術者に課せられた緊急の課題だと思います．

今回のワークショップの討論を通じてそれぞれが受け止めた「メッセージ」を，これからの仕事のなかで具体化することにつとめ，その結果を持って，いつかまた同じメンバーで集まりたいと思います．

それではこれでワークショップを終ります．プログラム委員，グループリーダ，事務局のみなさん，大変ご苦労さまでした（拍手）．

CALL FOR PAPERS

9TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING

FORMALIZING AND AUTOMATING THE SOFTWARE PROCESS

MONTEREY, CALIFORNIA, USA 30 March-2 April 1987

CHAIRMAN
William E. Riddle
software design & analysis, inc.
PO Box 3521
Boulder, CO 80303 USA

PROGRAM CHAIRMEN
Robert Balzer
Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292 USA

Kouichi Kishida
Software Research Associates, Inc.
1-1-1, Hirakawa-cho, Chiyoda-ku
Tokyo 102 JAPAN

TOOLS FAIR CHAIRMEN
Larry E. Druffel
Rational
1501 Salado Drive
Mountain View, CA 94043 USA

Jack C. Wileden
Computer and Information Sciences
University of Massachusetts
Amherst, MASS 01003 USA

TUTORIALS CHAIRMAN
Richard Fairley
Wang Institute
Tyng Road
Tyngsboro, MASS 01879 USA

LOCAL ARRANGEMENTS CHAIRMAN
William M. Murray
General Dynamics Corporation
12101 Woodcrest Executive Drive
St. Louis, MO 63141 USA

Theme: Over the last decade, two important myths have gradually been discarded: that the waterfall model describes the software lifecycle and that code is the only product of that cycle. Significant systems arise by a evolutionary process rather than by implementing a carefully constructed specification in a single sequence of development phases. Code defines execution but provides an inadequate basis for understanding, and hence, maintaining and/or evolving, the implemented system.

These insights have shifted attention toward paradigms which recognize software development as an iterative design (development) process and techniques for recording the process itself to document and understand the resulting implementation. Formalizing the intellectual activity provides the basis for a new generation of development tools which automate and/or effectively support portions of the design process.

Objectives: The main objective of the conference will be presentation and discussion of progress in formalizing and automating the software process through provision of better models, methods, and tools, and identification of critical issues to be addressed. A secondary objective is to broaden participation beyond traditional Software Engineering to include the Artificial Intelligence and Database technologies needed to model, store, manipulate, access, reason about, and automate portions of the software process. The final objective is to provide wider access to the best work in this expanded field by republishing outstanding papers, summarizing important specialized workshops, and presenting summaries of recent advances in particular areas.

Original Papers: Authors are encouraged to submit papers that directly address the formalization and automation of the software process. Theoretical papers should indicate applicability. Papers about new developments should separate achievement from plans and evaluate usage. Papers reporting practical experience should include sound empirical evidence. Authors should be advised that conference time allocated to the newly incorporated workshop summaries, "Recent Advances In", and republished outstanding papers, will substantially reduce the number of accepted papers.

Tools Fair: A Software Tools Fair will be held in parallel with the conference to provide conference attendees with information about current software tools. Both experimental and commercial software will be demonstrated. In addition, the conference will include a special, separate track featuring presentation and demonstration of tools selected by the tools fair committee. Those interested in exhibiting in the tools fair, and especially authors interested in presenting a paper describing practice and experience with a particular tool in conjunction with a demonstration, should contact one of the Tools Fair chairmen.

Submission of Papers: Four copies (in English) should be submitted by 1 September 1986 to either Program Chair. Papers should be no longer than 6000 words. Full-page figures should be counted as 300 words. The paper should include a short abstract and a list of keywords indicating subject classification. Notification of acceptance will be sent by 1 December 1986. Camera-ready copy of the final version will be due 1 February 1987.

Further Information: For further information and/or a copy of the advance program when available, write to 9ICSE, c/o IEEE Computer Society, 1730 Massachusetts Ave., N.W., Washington, D.C. 20036 USA.

IMPORTANT DATES

Submission Deadline:	**September 1, 1986**
Acceptance Notification:	**December 1, 1986**
Final Versions Due:	**February 1, 1987**

SPONSORED BY:

ACM SIGSOFT

IEEE COMPUTER SOCIETY

IEEE THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

SEA & jus 春のセミナー・ウィーク

セッション A0

ソフトウェアにおける技術移転

William E. Riddle

software design & analysus,inc.

1．はじめに

講師のリドルさんは，すでにＪＳＤのセミナーなどで何回か来日されているので，御存じの方も多いと思う．ソフトウェア開発環境の構築技術に関しての世界的権威であり，現在アメリカ国防省のＳＴＡＲＳプロジェクトの企画推進役として，お忙しい中を，ＳＥＡのオープニング・セミナーのためにわざわざおいでいただいた．

今回の講演は，昨年夏の第８回ＩＣＳＥに発表された論文「ソフトウェア技術の成熟過程」にもとづくお話であり，１９８０年代の最大の課題である技術移転を主題に，予定の時間をかなり延長して，熱のこもったレクチャーが行われた．

ここでは，テキストとして配布されたＯＨＰのコピーをもとに，その概要を報告する．

お話の構成は，次のとおりであった：

・技術成熟のプロセス
・実践的技術の現状
・先進的技術の現状
・技術成熟の期間
・技術移転のアプローチ

2．技術成熟のプロセス

一般的にいって，ひとつの技術が生まれ，成熟して行くプロセスは，

・先進的研究・試作
・実用化のための開発
・産業界への移転・普及

というかたちをとる．

研究開発コミュニティにある先進的技術と産業界における実践的技術との間には，時間的な遅れと技術レベルのギャップとがある．また，ある時点での実践的技術の中にもバラツキ（レベルの差）がある．ここでは，主として，先進的技術と実践的技術との間の時間的な遅れについて述べる．

技術移転のプロセスは，およそ次のとおりである：

・選定：導入すべき技術をえらぶ．
・取得：ふさわしいバージョンを入手する．
・統合：既存の技術と組み合わせる．
・普及：組織の内外に広める．

3．実践的技術の現状

まず，２つの調査の結果を紹介するかたちで，ソフトウェア・エンジニアリングの実践状況の説明がなされた．

（１）メリーランド大学の調査

これは，ＩＢＭのスポンサーとして，１９８４年に行われた調査である．約３０の組織（ＩＢＭの社内部門－５，他の米国の会社－１２，日本の会社－１３）を対象にアンケートおよびインタビュー形式の調査が実施された．

結果は，およそ次のようであった．

－ツールや方法論が広く利用されているわけではない．
－評価用のデータが収集されていない．
－プロジェクト管理のトレーニングがあまりなされていない．
－トレーニングが活用されていない．
－メンテナンスが軽視されている．

特に米国では，開発とメンテナンスとが別契約であるため開発時にメンテナンスをあまり考慮しない．

－時代遅れのマニュアル．
－通常，先導者は研究部門におり，めったに本気にはならない．

また，ツールや技法の利用状況は次のとおりであった．

高級言語	１００％
オンライン・アクセス	９０％
レビュー	８０％
プログラム設計言語	６０％
形式的技法	５０％

テスト・ツール　３０％
コード・オーディタ　２０％
チーフプログラマ・チーム　１５％
ベリフィケーション　０％
形式的要求記述　０％

これは，それぞれの会社の平均的な人に質問し，全員の技術者が利用しているというわけではなくても，一部でも使っていれば，ＹＥＳとした．

（２）ＤＯＤの調査

これは，防衛システムのソフトウェア開発における現状調査である．８つのシステム（主要な防衛システム－６，ＮＡＳＡ－１，商業ベース－１）が対象であった．

方法論／ツールの利用状況は，およそ次のとおりである．

具体化／テスト：方法論／ツール共に利用率は高い
設計　：方法論は高いがツールは低い
要求定義　：方法論／ツール共に低い
見積　：方法論／ツール共に低い

以上の２つの調査から，実践的技術の現状に関してまとめると，次のようなことがいえる．

・ソフトウェアによって具体化されうるものは多種多様で，かつ精密になってきており，全てのプロジェクトが何か新しいことを試みている．
・ソフトウェアにつきものの変化への対応が要求されている．
・現状で満足している人はいない？？
・要求定義プロセスでは，体系的な方法論やツールが不足している．
・工数の過小見積と生産性の過大見積．
・よく利用されている技術は，契約時に使用が義務づけられたものである．
・一般的には，問題の発生が開発時からリリース後のサポートに移っている．
・現状の要求に対して，現在の実践的技術では不十分である．

４．先進的技術の現状

１１種類ののソフトウェア技術について，それぞれの現状を次の４段階により，評価した．

成熟：完成し，広く利用されている
出現：一般に利用可能であるが，広くは使われていない
理解：十分に研究されているが，一般には利用できない
未成熟：研究中

つまり，成熟段階の技術は実践的であり，理解段階の技術は先進的であるといえる．

（１）成熟段階の例

・方法論－機能的分解
　　構造的分析技法
・テスト技術－ad hoc testing
・静的解析技法－文法解析
・検証技術－ウォークスルー
・プログラム変換－コンパイラ
・モデリング技術－プログラム設計言語
・計測技術－サイズ・メトリックス
・コンパイラ生成技術－パーサ・ジェネレータ
・ソフトウェア工学環境－プログラミング支援環境
・エディター－フルスクリーン・エディタ
・コマンド言語－ジョブ・コントロール言語

（２）出現段階の例

・方法論－データ分解
　　ワーニエ法，ジャクソン法
・テスト技術－カバレージ計測技術
・静的解析技法－タイプ・チェック
・検証技術－数学的机上チェック
・プログラム変換－プログラム・フォーマッタ
・モデリング技術－抽象データ型
・計測技術－データ収集機能
・コンパイラ生成技術－レキシカル・アナライザ・ジェネレータ
・ソフトウェア工学環境－要求定義システム
・エディター－対話型エディタ
・コマンド言語－プログラム可能なコマンド言語

（３）理解段階の例

・方法論－フル・ライフサイクル法
・テスト技術－機能的テスト
・静的解析技法－データ・フロー解析
・検証技術－検証システム
・プログラム変換－コンバージョン支援
・モデリング技術－設計モデリング記法
・計測技術－プログラム・メトリックス
・コンパイラ生成技術－コンパイラ・コンパイラ
・ソフトウェア工学環境－ライフサイクル支援環境

・エディター構文エディタ
・コマンド言語－汎用コマンド言語

（４）未成熟段階の例

・方法論－non work product methods
・テスト技術－テスト・ケース選択技術
・静的解析技法－ユーザによる特性の指定
・検証技術－プログラムの変換・合成
・プログラム変換－プログラム・ジェネレータ
・モデリング技術－再利用可能なソフトウェア部品
・計測技術－経験（観察）的技術
・コンパイラ生成技術－ターゲット独立型コンパイラ
・ソフトウェア工学環境－知識ベース環境
・エディターグラフィック・エディタ
・コマンド言語－ツール構成機能

５．技術成熟期間

何が技術の成熟を加速させ，何が減速させるのかを，事例の紹介によって分析し，その結果，どうすれば早く技術を成熟させることができるかについて，議論がなされた．

技術成熟期間の算定は，以下にあげるような１２のソフトウェア技術の事例について，それぞれの技術分野の専門家による主観的判断にもとづいて，各技術のクリティカル・コンセプトが出現してから，一般に広まるまでの期間を算定した．

・主要技術分野
　(1) 知識ベース・システム
　(2) ソフトウェア工学
　(3) フォーマル・バリデーション
　(4) コンパイラ・コンストラクション
　(5) メトリックス
・技術概念
　(6) 抽象データ・タイプ
　(7) 構造化プログラミング
・方法論技術
　(8) パルナス法
　(9) 費用見積モデル
・統合化技術
　(10) Smalltalk-80
　(11) SREM
　(12) UNIX

成熟段階は，次のようなステップに分けて分析された．

・基礎研究
　技術の基礎となるキイ・アイデアや明確に認識された問題の出現．
・概念形成
　初期の論文やデモ・システムによる解決策の明確な定義．
・開発と拡張
　利用可能になる．
・内部増強
　開発グループの外での利用促進．
・外部増強
　価値と適用性を実際のプロジェクトで証明する．
・普及
　コミュニティの４０％までに普及．
　コミュニティの７０％までに普及．

５．１．実例

ここで，統合化技術の３つの例について，それぞれの成熟プロセスをみてみると，以下のようになる．

◇UNIX の成熟

UNIX は，ATTではソフトウェアの開発／販売に規制があったにもかかわらず，ほとんど自然発生的にうまれてきたものである．その成熟の過程は，およそ次のとおり．

基礎研究（1967）：MULTICS system
概念形成（1968->1971）：initial versions (PDP-7)
開発と拡張（1972->1974）：PDP 9 version
内部増強（1975->1976）：論文の収集，一般の利用開始
外部増強（1977->1981）：Ｃでのリライト，バークレイでの作業開始
普及（1982 ->）：米大学の９０％が利用

◇SREMの成熟

SREM は，成熟に時間がかかっている．それは，最初に選択されたハードウェアが，あまり一般に広まっていない大型機だったからである．のちに，それを VAX にかえることで，普及が促進された．政府の仕事なので，政治的理由により基礎研究に時間がかかったこともある．

その成熟の過程は，およそ次のとおり．

基礎研究（->1968）：ISDOS の出現
概念形成（1969->1974）：RSL & REVS の設計
開発と拡張（1975->1977）：first version

内部増強（1978->1981）：オプティマイズと機能強化

外部増強（1982->）：VAX バージョン，ユーザ・グループ

◇Smalltalk-80 の成熟

この場合の特徴は，基礎研究期間がながいことである．発案者（アラン・ケイ）に先見性がありすぎて，他人がかれのアイデアを理解できなかったことが，大きな原因である．その成熟の過程は，およそ次のとおり．

基礎研究（ ->?）：アラン・ケイが DYNABOOK を定義

概念形成（->1972）：smalltalk-72 の開発（基礎バージョンの出現）

開発と拡張（1973->1976）：smalltalk-76 の開発（新バージョンの出現）

内部増強（1977->1981）：拡張（他マシンへの移植可能）

外部増強（1982->1983）：テスト及び移植

普及（1984->）：販売

これらの実例から，成熟時間のケースごとの比較を試みてみよう．

概念形成までの期間は，

最短：4.0年　最長：10.0年　平均：6.3年

開発と拡張までの期間は，

最短：6.0年　最長：14.0年　平均：10.1年

内部増強までの期間は，

最短：9.0年　最長：18.5年　平均：13.8年

外部増強までの期間は，

最短：11.0年　最長：23.0年　平均：17.0年

というようになる．

５．２．技術成熟の一般的傾向

各カテゴリ別に，技術成熟の一般的傾向を分析すると，次のようなことがいえる．

・主要技術分野

いくつもの成功例がなければ一般には広まらない．成熟するには，ユーザが真に必要としていなければならない（見積技術は，政府関係者が必要とした）．

・技術概念

簡単に理解できるシンプルなアイデアは，はやく成熟する．概念であるため，利用者にたいする例題が必要である．

・方法論技術

概念形成の期間がながい．その理由は，基礎となる技術の成熟をまたねばならない，他の技術と併用するためのルールやガイドラインの設定が必要である，また効果をデモできねばならない，といったことである．

・統合化技術

統合すべきものが同時に現れなければならない．方法論技術に比べると，増強（ツール開発等）に時間がかかる．

５．３．技術移転をめぐる諸要因

技術移転のスピードをゆるめる主要な原因としては，次のようなものが考えられる．

・些細で単純なミス

SREM でのハードの選択ミス（大規模なマシンを選んでしまった）．

・おもわぬ時間がかかるものがある．

SREM ではユーザのための資料作成におもわぬ時間がかかった．

・導入コストが高い．

高価であるとか，教育費が高いとか．

・導入予算がない．

効果が高いとわかっても，導入予算がない．

・心理的障害

知的だと思っている作業が自動化されるのをこばむ．

・些細な変更による多種類のバージョン．

一方，技術移転をスピードアップする要因には，次のようなものがある．

・開発者による成功例の提示．

・利用を促進する．

SREMの場合，米陸軍が契約時の見積評価に使用したことから，受注側が事前に利用するようになった．

・技術指向の強い管理者が率先して導入する．

・スポークスマンによる宣伝．

・導入のための講師やテキストが準備されている．

・潜在ニーズをうまくつかんだ技術．

コスト見積技術等．

・シンプルな技術である．

・段階的に導入できる．

たとえば，UNIX　や SREM の導入は，変化が大きすぎるため広まらない．

その他，技術移転に影響を与える要因をあげると，次のようなものがある．

・概念的な統合性

定量化指標計測技術は，まだ学会でも論争中である．したがって普及しない．一方，UNIX のパイプ等の概念は美しい．したがって，だれでもが納得する．

・認識されたニーズ

・チューナビリティ

ユーザのニーズにあわせて変更できる．

・成功例

SREM は，ライフサイクル全体を支援するため，成功例がでるまでに時間がかかりすぎた．

・管理者の意欲

管理者に意欲があれば，予算問題の解決や将来を展望した活動が容易になる．

・スポークスマンによる宣伝や発表

成功者による説明等も効果的である．

・教育

５．４．技術移転に関する一般的結論

一つの技術がユーザコミュニティのなかに広まるには，１５年から２０年の期間が必要である．大規模ものであると，それを，組織の中に浸透させるには，さらに4から８年かかる．

効果的な技術移転プロセスとは

－試用を重ねながらの

－段階的な改良と

－効果の経験的な証明

をいう．

UNIX, Smalltalk, コンパイラ構築技術のように　研究者からの例題があれば，成熟は促進される．

また，技術の成熟や移転は，以下の要因によって促進される．

－明確なニーズ

国策等によりニーズを作りだすのはよくない（失敗するとチャレンジャの場合のように大問題となる）．

－効果が明確

－導入による環境の変化が少ない

－コミュニティに受入準備がある

６．技術移転のアプローチ

現時点での方法論やツールの利用率は，先進技術が示している可能性にくらべて，非常に低い．したがって，技術移転にかかる時間をもっと短くすることが望まれている．

技術移転を促進するには，受入側の現状の技術を革命的に変化させるのではなく，段階的に試用／評価を繰り返しながら導入していくアプローチをとるべきであろう．特に，気のきいたシンプルで小規模な技術を少しずつ提供し，受入側を「やめられない止まらない」という状態にしてしまうポテトチップ効果を狙ったアプローチが有効である．

質疑応答

◇見積技術の例は？

明確ではない．必ずしも見積モデル等を使っていることをさしているわけではない．ただし，ＴＲＷはＣＯＣＯＭＯを使用している．

◇開発とメンテナンスをＵＳでは別契約にするのはなぜか？

政府の仕事などでは，機会均等の原則で契約が別になっている．

◇第４世代言語（アプリケーション・ジェネレータ）を利用すると，開発とメンテネンスを別契約にできなくなるのでは？

ツールの開発込みの発注になるか，販売されているツールを購入して利用することが，契約の条件となるだろう．

◇技術移転において，加速要因である’チューナビリティ’は，減速要因である’些細な変更による多種類のバージョン’の出現を引き起こすように思えるが？

もちろん，コントロールが必要である．しかし，ここで’チューナビリティ’とは，SREM において，記述言語の仕様を，ユーザが SREM の内部を変更せずに変えられる機能をいうのであって，ツールの内部を勝手に変更することを意味しているのではない．

（レポータ：林　香）

SEA & jus 春のセミナー・ウィーク
セッション S1

シグマ・システム

久保宏　永島晃　川口隆
情報処理産業振興事業協会

１．はじめに

先日の機関誌編集会議で「無料でセミナーを聞かせてやるかわりに，レポートを書きなさい」という話があった．私はＪＳＤに出向している関係で，ときどき同プロジェクトに関する噂話みたいな情報は耳にしていたが，プロジェクト参加メンバーからの直接の話はあまり聞いたことがなかったので，よろこんで出席させてもらうことにした．

しかし，ＪＳＤでの「保守技術開発プロジェクト」の最終納品作業に忙殺され，レポート提出を怠っていたところ，「早く書け！」とのキツーイ催促．こうして休日のひととき，近くのＪＡＺＺ喫茶にコーヒー一杯で粘りながら，原稿執筆に励む結果となってしまった．

セミナーの内容は，プロジェクトの運営，ネットワーク，およびツールの話という３つの部分にわかれていたが，その忠実な紹介というよりは，話を聞いて感じたこと，そしてシグマについて日頃思っていることを，この場を借りて述べてみたい．

２．シグマ・プロジェクトとは？

昭和５９年の調査で，わが国のソフトウェア技術者は，現在の４０万人から，５年後には４倍の１６０万人が必要となる旨が報告された．しかるに，人口動態調査からすれば，可能な動員量は６０万人と指定され，５年後にはプログラマーが１００万人不足するという話が持ち上がり，いわゆる「ソフトウェア・クライシス」に対する懸念が生じてきた．

ソフトウェア業界の中では，すでに何年も前から開発の生産性向上に対する関心は高く，「開発環境」の整備や，「ソフトウェア・エンジニアリング」の実践が声高く叫ばれてきたが，最近では，ようやくこのようなことばを，エンド・ユーザ・サイドの人々も口にするようになってきた．こうした状況に対処すべく，ナショナル・プロジェクトとしてこれまでに，「ソフトウェア生産技術開発計画」（昭和５６－６０年度），「ソフトウェア保守技術開発計画」（昭和５６－６０年度）が実施された．しかし，このような努力によっても，なお８０万人分のバックログが解消されずに残る，ということがいわれている．

シグマ・プロジェクトは，これら過去のプロジェクトの結果を踏まえて，ツールの効率的利用を促進することにより，残る８０万人分の人員不足の解消をはかる，という位置付けにあると思う．

３．シグマ・ネットワークの利用法

シグマ・システムを，その名が示す工業化のイメージで捉えてみると，工具や工作機械をとりそろえたツール・センター，ツールや情報およびプロダクツの移動搬送手段，各工程や工場を渡り歩いて物を送るワークショップ，といえる．つまり，これまでのわが国ではあまり一般的でない，ネットワーク，ツールキット，ワークステーション，という三つの基本技術により構成されるシステムである．

まず，全国規模のネットワークに関しては，いかなる利用が考えられるであろうか？

シグマ・システムでは，ネットワーク上のツールを用いて，プログラム開発をスムーズに効率良く行うこと以外に，各サイト上で開発された個別システムの連繋を容易になしうること，つまり，システム・インテグレーションをこれまで以上に大規模化しうる可能性がある点に，大きな意味があると思う．

たとえば，全国に多数の工場を持つ製造業のような場合，工程や資材等の管理システムは，多くの場合，工場ごとに似かよったローカルな管理システムを持っており，ソフトまたはハード資源の重複運用，重複開発のむだが指摘されている．最近では，このような社内大規模システムの改善を行う会社が多くなってきた．一般的には，全社的なセンターを設置し，集中的に，もしくは分散集中的に管理していこうという方向にある．

しかし，各工場間の慣習，運用等の違いをこえて，統

一的なシステムを構築するとなると，ネットワークの構築等，新たに着手しなければならぬ部分が多くなったり，これまでの資源のほとんどを改編する必要に迫られたりして，費用のみならず，開発期間の長さに二の足を踏んでしまうことが多い．また，相異なる会社間やその他の団体をも含むネットワーク・システムとなると，その便利さは理解出来ても，投資意欲が湧くまでにはいたらないのが，一般ではなかろうかと思う．

シグマ・システムが目指す，全国規模のネットワークの建設は，このような問題に対処する１つの，大きな解決策だと思う．シグマ・ネットの持つメッセージ通信機能は，電話，ファクシミリ，郵便，伝言板等をミックスした統合的な通信手段を提供してくれるであろう．

こうした情報交換機能以外にも，シグマ・ネットは，情報流通を促すインフラストラクチャーとなる大きな可能性がある．ツールやアプリケーション・パッケージの流通が促進され，再利用を進展させる．また国内外データベースへのアクセスも容易になり，データベース業の発展を促す．シグマ・ネットワークは，日本初の大規模ネットワークであるため，単純なシステム開発環境というだけでなく，多大な利用形態が考えられる反面，セキュリティの重要性が，これまでになく大きな問題となるであろう．

４．シグマに対する提言

「５年後」に完成するシグマ・システムは，何かを作りだすための１つの土台であると考えられる．その上にさまざまな構造物を建築することを容易にするために，次のような提言をしておきたい．

（ａ）フレキシブルなシステムであること

５年目以後の拡張を考え，システム構成の面で自由な改変が可能であるようにしたい．基本ＯＳであるＵＮＩＸのバージョン・アップや，ユーザ・インタフェースまわりの改良が可能なこと．また，現時点でも変遷著しいハードウェア（たとえばターゲット・マシンやワークステーション等）のグレードアップに，柔軟に追随できるようであって欲しい．

また，シグマ・ツールとしての言語標準や，ツール規格，運営等に関する諸手続きが不必要に厳格であったり，拡張性を殺すことがないようにしてもらいたい．ナショナル・プロジェクトは，ともすれば融通性のない規則で縛られがちであるが，シグマ・システムは供給者と使用者全員で作り上げるものであることをよく認識し，このようなことがないようにしてもらいたい．

さらに，当然のことだが，ネットワークのインタフェイスが容易にとれることが必要である．シグマ・システムを利用するユーザは，供給されるツールを単純に使用することだけを考えている，というわけではない．それぞれが持つ既存の開発環境との融合や，シグマ・ネットワークの機能を最大限に利用したシステムを構築したいと考えるであろう．

このようなことを容易に行えるように，運用手続き面や技術的な側面で，ローカル・システムとの整合性の高さを保証してほしい．とくに目玉商品の一つである国際間ネットワークに関しては，運用や手続き面での整合性が大きな問題となる．

（ｂ）情報混乱ではなく，情報化を促進すること

現在，日本ではシステムのセキュリティの問題は，最近になってその重要性が叫ばれだした．今後，コンピュータ関連業種に就く人が増えると同時に，コンピュータ犯罪による社会に対する脅威の種も増えてくる．また，シグマ・システムの様な大規模ネットワーク・システムが発展するにつれ，そのインパクトは大きいものとなるため，セキュリティについては十分な配慮を望む．

（ｃ）新しさを失わぬこと

過去のナショナル・プロジェクトがそうであったように，開発当初は先進的であった技術も，完成する頃には凡庸なものとなることがある．今回のように，長期間に渡るプロジェクトでは，特に実現可能性を重視するあまり，先進性を殺し，完成時点で世の水準以下とならぬようにして欲しい．

５．おわりに

以上，話の内容がネットワーク利用に偏ってしまったが，何しろ日本ではほとんど未知の部分が多いため，将来のネットワーク社会がどのようなものになるのか，推測することはむずかしい．

セミナーで聞いたことよりも，自分勝手な空想の方が多くなってしまったが，何とかノルマを達成できただろうか．

最後に，このプロジェクトは，日米の通信会社，ハードおよびソフトウェア・メーカーをはじめとして，数多くのボランティア出資社によって遂行されている．これを機に，これら参加各社のネットワークが，一層強くなることを願う．

（レポータ：大西亮一）

SEA & jus 春のセミナー・ウィーク

セッション S2

ソフトウェアの再利用

大木幹雄

日本電子計算

1．はじめに

再利用についての世間の関心は，年々高まる一方のようである．このセッションでは，冒頭に講師もそう指摘しておられたが，受講者の年齢層がかなり高い（平均３０代後半？）ことが，ちょっぴり意外であった．いままでは，現場での自発的再利用が中心だったのだが，これからは，管理された組織内部での再利用が，積極的に行われていく前触れなのだろうか．

講師の大木さんは，昨年度ＪＩＳＡ再利用部会のリーダをつとめられた方で，すでに本誌創刊号のインタビューに，再利用に対する思い入れとともに登場されているので，御存知の方も多いと思う．

2．講演内容

（1）再利用の目的

それは，もちろん，ソフトウェア資産が蓄積されているにもかかわらず，うまく利用できていないからに他ならない．しかし，最近では，ＡＩ技術の進展，ソフトウェア需要の増大，要員不足，ハードウェアやＯＡ機器の進歩，コストダウン等により再利用技術が儲かるようになってきた．また，再利用のノウハウが商売になるであろうことも予想される．

（2）再利用の基本的な考え方

まず，いくつかの実例を収集・体系化する．こうして構築される部品もしくはソフトウェアＤＢを，新たな環境（業務）に適用し，結果をフィードバックするという再利用の手法は，知識処理のプロセス，すなわち，＜知識の獲得－知識の整理－知識ベースの構築－推論－知識の再構成＞と関連付けて説明することができる．

（3）再利用を図る上で必要な要因

次の３つが相互に関係している．

・再利用の範囲：主に体制に関する要因
・再利用の方法：支援ツール等に関する要因
・再利用の内容：対象に関する要因

このうち特に，最後のものについては，部品や既存のプログラム・モジュールのほかに，Ｐ．Ｆｒｅｅｍａｎが階層的に分類したように，各種のソフトウェア開発情報の再利用を考える必要がある．

（4）再利用の対象

アンケート調査の結果，現状では，

１．コピーまたはサブルーチン・ライブラリ
２．マクロ／プログラム論理パターン
３．設計ドキュメント

の順で再利用が行われている．１と２の差はあまりないが，２と３の開きは大きい．

（5）再利用の現状と問題点

再利用を阻害する要因の代表的なものとしては，次のようなことがあげられる．

・業務目的に合致した既存ドキュメント，プログラムを検索するツールがない．
・利用すべきプログラム，ドキュメントがＤＢ化されていない．
・ソフトウェア開発プロセスの標準化が遅れている．
・再利用を支援するよいツールがない．

また，以下に述べるような部品作成時の問題も，再利用を大きく阻害している．

・作成基準が明確でない．
・可変部分，不変部分の判断が明確でない．
・汎用的な部品は１つの具体的ものを作るよりむづかしい．
・部品のエラーは影響が大きい．
・使い方になれなければ効率が悪い．

（6）再利用促進の方策

再利用の体制について，

・管理部署
・標準化
・監査

の３つの視点から，促進のキーポイントになるであろう項目が説明された．このうち特に，監査に関する資料は

大いに役立ちそうである．

（7）再利用環境整備の方法

再利用のためのツールやアプリケーションが乏しい現状では，「こうすればうまくいく」ということはいえないので，現状で利用できる（もしくはできそうな）ツールの紹介があった．

またソース・コード再利用の新しい試みとして，部品の抽象化に関する非手続き型言語や論理型言語，さらにはオブジェクト指向型言語の考え方と，それらの再利用システムへの組み込み方法の応用例が解説された．ただし，これらの多くは現在まだ実験段階である．

さらに，設計の再利用の一例として J-Star による設計書が示された．

（8）再利用の評価基準

いくつかの評価尺度の紹介と，定量的および定性的効果が説明された．再利用技術がそれぞれツールによって異なるため，共通の評価尺度がないことが説明された．

（9）再利用の効果

再利用を進めることにより，生産性が飛躍的に向上することは明らかである．また，いままでむずかしかった見積りが，ファンクション・ポイント法で可能になることが示された．

（10）ソフトウェア部品普及の問題点

講義の中でも触れられた問題点のまとめとして

・部品の登録・蓄積
・標準部品規格はどうあるべきか？
・部品の権利保護（著作権）問題
・製品を試用している時の問題（再利用部品，ツールは概して製品そのものの仕掛けは簡単ではあるが，そこにあるノウハウにお金が掛かっている場合が多い）

等があげられた．

3．質疑応答

Q　ＡＩの技術が，今後どう再利用に使われるのか．

A　フレームの考え方を調べていくと，その多くの部分は，実はＡＩの持っている推論エンジンと同様の機能になってくる．

Q　調査された再利用ツールで，コメントはどのように評価したのか．

A　部品のなかにコメントがあっても，部品の中をみないかぎり意味を持たない．また，ジェネレートされたときの効果としても特に評価しなかった．

Q　部品するか否かの基準はあるか．

A　一番のポイントだと思うが，決定的なことはわからない．どれくらい使われたら部品とすべきか，ということもよくわかっていない．

Q　ソース・コードの再利用が話の中心だったが，現時点では，このあたりが限界なのか．

A　そのとおりだと思う．その原因は，要求仕様等の情報が機械化されていないことによる．なお，このような問題に対する国家プロジェクトＦＡＳＥＴが，去年からＪＳＤで開始されている．

4．個人的な感想

今日の再利用は，プログラム・ソースだけではなく，要求仕様定義やそれらの決定に至った要因，さらにはテクノロジ・トランスファの問題等も含めて議論されることがある．しかし，今回の講義では，ソース・コードの再利用を中心に考えたときの現状の自動化技術の概要が，話の中心であった．非常によい内容選択であったと思う．

また，知識処理やそこで用いられているＰｒｏｌｏｇ，Ｓｍａｌｌｔａｌｋ等の言語，あるいはフレーム，メッセージ・パッシング等の技法が，今後の再利用を促進していく一つの鍵になっていくという見解も，講義の流れのなかで，きわめて自然に理解することができた．

こういうセミナーでは無理かもしれないが，そうした先進技術と開発現場とをつなぐためには，現状の再利用ツール・アプリケーションの使いごごち，機能のよしあしが，今後しばらくは，再利用化技法の普及促進に大きく影響を与えることが明白であり，それについても，さらに突っ込んだ見解がほしかったように思う．

5．おわりに

つたない文章で，当日の熱気あふれる会場の雰囲気を，どの程度読者のみなさんに感じとっていただけたか，あまり自信がない．また，それ以上に，私の誤解による記述があることをおそれる．末筆ながら，記念すべき第1回セミナーに，すばらしい内容のプレゼンテーションをしてくださった大木さんに，ＳＥＡ会員の１人として，お礼申しあげたい．

（レポータ：渡辺雄一）

SEA & jus 春のセミナー・ウィーク

セッション S3

ソフトウェア・データベース管理

林香

ソフトウェア・リサーチ・アソシェイツ

磯部裕一

情報処理産業振興事業協会

1．はじめに

このセッションでは，林氏がＡＣＭ「将来のＡＤＡ環境ワークショップ」（１９８４年９月，サンタ・バーバラ）での討論をもとに，また磯部氏がＩＰＡシグマ・プロジェクトでの検討内容を中心にして，以下のような講演が行われた．

2．ソフトウェア・データベースの概念（林）

ソフトウェア・データベース管理システム（以下，ＳＷＤＢＭＳと略す）は，ソフトウェア開発において大きなウェイトを占めている情報の蓄積・検索活動を支援する「環境」を提供するために，次のように概念付けされる．

（１）ソフトウェア開発に関する各種情報の蓄積

開発システムに関するドキュメントの保存・管理と，スケジュールやソフトウェアの品質管理のためのデータの収集

（２）ソフトウェア・ツール間のインタフェース

ツールの集合体からなるソフトウェア開発環境を，単なるツールのよせ集めとしないため．

また，ＳＷＤＢＭＳの持つべき特性として，下記のような項目があげられる．

- プロジェクトのすべての情報（管理・技術両面の情報と生産物のシステム構成管理）を格納し検索できること．
- ソフトウェア開発プロセス（ウォーターフォール・モデル等）から独立していること．
- 開発対象アプリケーション（科学技術計算や事務処理）から独立していること．
- 各種ユーザ（管理者，設計者，プログラマ等）のニーズにこたえられること．
- 監視機能（設計者の許可なしには仕様変更を許さない等）．
- ＤＢインタフェースのレベルでの操作の共通化（理想的には，データ構造を意識しないでツールがＳＷＤＢＭＳをアクセスできること）．
- 拡張性（データ構造の変更が容易でかつツールの手直しが不要など）．
- 外界とのインタフェースの確保（ｉｎｐｏｒｔ／ｅｘｐｏｒｔ機能）．
- ロック機構や一貫性（仕様とソースの一致），無矛盾性（仕様間での無矛盾）を保持する機能．
- その他（許容範囲内のレスポンスタイム，長期保存／バックアップのためのセーブ機能）．

以上の特性には，相互に矛盾する事項が含まれている．したがって，実用的なソフトウェア環境を構築するためには，まず既存の環境の中でＳＷＤＢＭＳに対する評価を行い，つぎに評価結果にもとづいたプロトタイピングを行って実用的なＳＷＤＢＭＳの機能や特性を明らかにしていく必要がある．

3．ケース・スタディ（林）

ケース・スタディとして，インフォーマルな仕様のデータベース化の試み（ＳＰＥＸ）と，事務アプリケーション・システム開発用ＳＷＤＢＭＳの例（ＺＯＤＩＡＣ）が説明されたが，ここでは後者について述べる．

ＺＯＤＩＡＣの基本思想は，次の通りである．

- システム内のデータ相互矛盾の排除
- 要求定義－プログラミング－テスト作業全般のＯＮ－ＬＩＮＥ化
- プログラム（ＣＯＢＯＬ）の自動生成
- ドキュメントの自動生成
- 管理データの収集と活用

ＳＷＤＢＭＳとしてのＺＯＤＩＡＣの特徴を評価すると，次のようになる．各項目の（）内の印は，実現程度を表している．

・プロジェクトの全ての情報を格納し検索できる（X）
・生産物のバージョン管理（X）
・生産物のシステム構成管理（O）
・ソフトウェア開発プロセスからの独立（X）
・開発対象アプリケーションの独立（X）
・各種ユーザのニーズにこたえる（X）
・監視機能（O）
・DBインタフェースのレベルでは操作を共通（O）
・拡張性（X）
・外界とのインタフェースの確保（O）
・ロック機構や一貫性，無矛盾性を保持する機能（O）
・許容範囲内のレスポンスタイム（△）
・長期保存／バックアップのためのセーブ機能（O）

4．SWDBMSを前提とした開発環境（礒部）

SWDBMSは，ソフトウェアのライフサイクル全段階で発生または利用されるデータを一元的に保守・管理し，これらを利用容易な形で提供するものである．しかし，その機能については今なお不明確なこともある．理由は，SWDBに格納されるデータ，SWDBを利用するツール，ユーザの利用形態等が明確でないためである．

（1）ツールとSWDBの関連

統合的開発支援環境下では，データはユーザから見れば論理的にはSWDBとして一元管理されているように見えるが，内部的にはファイル形式で保存され，その構造を各ツールが利用できなければならない．また，ツール間の依存関係を定義した情報をSWDBMSに保存し，ユーザが自動的に最適なツールを利用できるようにする必要もある．

（2）開発プロセス上の作業とSWDBの関連イメージ

SWDBMSは，データ資源を一元管理し開発作業を効率化するために，その機能としてデータ・ディクショナリィを持つ．これによって，データをリレーションやモジュール化だけでなく，情報システムの体系に合せて管理する．また，ライブラリ管理などの管理機能との連結も必要である．

（3）分野別の作業特性

ソフトウェアのシステムに依存した管理機能はSWDBMSが提供し，技法・言語・開発体制に依存したデータの処理はツールや運用で支援する．ただし，各ツールや運用を体系付けるプロジェクト管理ツール間インタフェース情報定義などをSSWDBMSに取り込むことは必要である．

5．SWDBMSのモデル化（礒部）

（1）モデル化の前提条件

・ソフトウェアを構成部品に分解し，作業に合わせて統合化する．
・SWDBMSをブラックボックスとする．
・SWDBMSはデータを構造化するメカニズムを提供し，その定義はツール間で決める．
・ソフトウェアの再利用を支援する．
・プロジェクト内の共有財産となるものを一元管理をする．
・SWDBMSはデータ管理を主目的とし，ブラウジング・編集・解析は各ツールが行う．
・ユーザや各ツールからSWDBMSへのインターフェースを共通化する．

6．将来への展望（礒部）

（1）SWDBMSを構築するためには，現状ではHW／OS／DBMS／通信等が制約となることが多く，これらの性能や機能の改善が必要である．

（2）理想的なSWDBMSになるほど，その内部構造や操作性が複雑になるが，その場合でも，ユーザが利用しやすいように，インターフェースを向上させる．

（3）ソフトウェアの開発環境は，スタンドアロンから分散化に向かっており，分散DBの実現が必要である．

（4）SWDBMSは，ソフトウェアの再利用・部品化技術を促進させるために，部品の検索や，再利用のためのガイダンス等の，支援機能を持つ必要がある．

（5）オブジェクト指向言語専用システムのような，プロトタイピング技術を導入する．

（6）SWDBMSは，知識ベース等の知識応用技術との連動で，その応用範囲を格段に広げることができる．

7．感想

現時点において利用可能なSWDBMSの開発をめざせば，理想からはるかに低いレベルのものになってしまうし，また逆に理想的なものの開発を夢見ると21世紀まで待たなければならないのが，技術の現状である．当面は，シグマでどのようなシステムが開発されるかを期待し，さらにその先に理想的なSWDBMSが現れることを，技術者の夢の一つとしたいと思う．

（レポータ：山内　徹）

SEA & jus 春のセミナー・ウィーク

セッション S4

J-Star によるドキュメントの機械化

盛田政敏

神戸コンピューターサービス

１．はじめに

ある程度以上の規模のソフトウェアを開発する場合，ドキュメンテーションをどうするかが，きわめて大きな問題である．「紺屋の白袴」のたとえどおり，これまでソフトウェアのドキュメントは，ソース・コードを唯一の例外として，ほとんどの場合，手作業で作られてきた．

このセッションでは，この問題にワークステーションによる機械化のアプローチで取り組んでいる貴重なケース・スタディの説明があった．講師の盛田さんのひととなりについては，本誌第２号の「プログラマ！！」を参照されたい．

２．導入の経緯

現状における手書きのドキュメントやその作成作業には，以下のような問題がある．

- ・メンテナンスされないドキュメント－＞現状のシステムやソースコードと不一致
- ・再利用できないドキュメント－＞類似ドキュメントの氾濫
- ・どこにあるか不明なドキュメン保存－＞個人レベルでの保管
- ・癖があり読みにくい手書ドキュメント－＞形式，書き方，内容，用語，文字が不統一

このような問題は，程度の差はあっても，どの会社も抱えているのではなかろうか．これらの問題の根は深く，改善といっても一朝一夕にはむずかしい．結果としてドキュメントの品質や生産性が低下する．もっと何とかならないか，ということになってくる．

そこで，ドキュメンテーション作業の改善をはかりながら，ソフトウェアＣＡＤ／ＣＡＭの実現をめざして，５９年夏より検討を開始し，テスト使用を経て６０年４月よりＪ－Ｓｔａｒを導入，現在に至っている．

３．改善へのアプローチ（方向性）

前述の諸問題を次のように考えた．

（１）ソフトウェア開発のドキュメンテーション作業を対象とする：ソフトウェアＣＡＤ／ＣＡＭへの第一ステップとして，まず，分析，設計段階におけるドキュメントおよびドキュメンテーション作業の改善をはかる．

（２）ソフトウェア・ドキュメント作業の改善：まず，ペーパードキュメントを電子化し，文書作成・修正・複写・印刷・伝達・保管・検索等の作業について方法論を確立する．そのために，ツールを開発導入し，環境を創造し，基準やルールだけでなく，物理的な仕組みを環境としてシステム化する．

（３）ツール手段としての高性能ＷＳの導入：高性能ＷＳ（ワークステーション）の導入とそのシステム構築を考えた．ドキュメント作成だけであれば，ワープロで充分であるが，作られたドキュメントが，開発・テスト・運用・保守といったライフサイクル全般で使われいくためには，ＷＳの機能が必要になってくる．

（４）電子ドキュメント化の推進：ペーパードキュメントを少しでも多く機械化する．組織をあげて有効なものにしていくには，より多くの電子ドキュメントが作成される必要がある．

（５）ドキュメント・データーベース（ＤＤＢ）およびネットワークの構築：これらの電子ドキュメントを蓄積し，利用するためには，まず共通的・一元的に管理できるデータベースおよびコミュニケーションを可能にするネットワークの構築が必須となる．

（６）オフイスにおける環境基盤の確立とエンジニアリング環境へのアプローチ：ドキュメント・データーベースとネットワークが構築できれば，ソフトウェア開発のオフイスにおける環境基盤が確立する．これをベースに，エンジニアリングを支援する環境へアプローチする．

（７）ＳＷＤＢへの足掛かり：ドキュメント資産が，電

子化，データベース化されることで，既にコンピュータ内にあるソフトウェア・プロダクト資産と結合可能になる．これによって，ソフトウェア・データーベース（ＳＷＤＢ）構築への足掛かりをえることになる．

（8）ドキュメントの品質向上とドキュメント作業の生産性向上：これらのプロセスを通じてドキュメントの品質，ドキュメント作業の生産性を向上させる．

４．取り組みの経過

５９年７月に導入検討を開始し，他社導入事例を見学したり，セミナーに出席した．５９年１２月にテスト機を設置．６０年４月に正式導入（３台）し，部内に研究会を発足させた．また，ＨＯＳＴマシン（Ｆａｃｏｍ Ｍ）と接続した．６０年８月から利用状況をまとめ，標準化の検討を開始した．６０年１０月に増設（６台）をし，ホスト接続での展開を行った．

５．利用状況

導入後半年間（前期）はまだ普及期間であり，Ｊ－Ｓｔａｒの持つ可能性，実用性の確認を行うトライアル期間であった．研究会も普及活動が中心となっており，これらにより利用人口はかなり増大した．しかし，全員がフルに使える状況でもなく，本格的に使っているグループがある一方，全く使用経験のない人もいる．

使用者は，マシンの絶対数の不足から，予約管理による利用規制を受けている．使用者が多い部所ほど，利用規制による弊害が大きくなっている．ＴＳＳ端末としての使用も多い．これはＴＳＳ専用端末が少ないための現象であるが，このことによりＪ－Ｓｔａｒの本来の使用が制限されている．

月別使用状況を見ると，１日７時間は確実に使用している．

６．生産性について

使用者へのアンケート調査によると，３５人中９人は，ほぼ一通りＪ－Ｓｔａｒを使える．その使用した結果として以下のようになっている．

Ｊ－Ｓｔａｒを使用することで

・作業時間が短縮された（4人－44.5%）
・作業時間が長くなった（2人－22.2%）
・無回答（3人－33.3%）

作業時間が短縮された人で

・２０%程度作業が短縮された（2人）
・３０%程度作業が短縮された（1人）
・５０%程度作業が短縮された（1人）

作業時間が短縮された理由として

・既存文章を流用できる
・修正が簡単にできる

また，作業時間が長くなった理由としては

・新規作成時手間取る
・必ずしも十分に使いきれない

という結果がでている．

操作になれ，ドキュメントの蓄積及び流用がはじまるとドキュメント作業は２０～３０%効率化する．また，このデータは，さらにうまく利用すれば５０%前後の効率アップも可能であることを示している．

７．問題点

・利用者のスキルに片寄りがある
・台数が少ない（使いたい時につかえない－予約制）
・ＴＳＳ端末として使われることも多く，本来の使用ができない
・まだフルに使えていない
・標準化があまり進んでいない

８．評価

Ｊ－Ｓｔａｒは，ドキュメント・プロセッサーとしての機能は高く，ほぼ期待どおりであったが，まだそれをうまく使いこなしているとはいえない．

今後の展望として，開発マシン，ターゲット・マシンとの接続とプログラム開発環境としての整備を行いたいが，後者は，現状ではユーザに充分解放されていない．

また，詳細設計書，プログラム設計書フェィズに利用をすることも検討している．

９．感想

ソフトウェア開発において，紙と鉛筆によるドキュメントの作成がかなりのウエイト（ネックでもある）を占めるものであることは事実である．生産性向上のために，作業の機械化を考えている管理者はいると思うが，それを実行している例はあまりないのではないだろうか．Ｊ－Ｓｔａｒがそれにふさわしかという議論は別にしても，そのアプローチによる成果は期待したい．

講演では，機械化に取り組んできた苦労話があり，同じような問題を検討している管理者には，ぜひ聞いてもらいたい内容でもあった．

（レポータ：野辺良一）

SEA & jus 春のセミナー・ウィーク

セッション S5

エキスパート・システム

玉井哲雄

三菱総合研究所

１．はじめに

比較的早くから積極的にＡＩを導入しようという姿勢を見せている企業の業種に注目して見ると，すでに成熟してしまっていると考えられる製造業関係あるいは鉄鋼業等の重工業関連の企業が多いことに気がつく．つまり，現状の打開やＡＩにより新たな展望，あるは業務分野を開拓し活性化を図りたいといった要望が，そこにあるからであろう．

２．ＡＩの歴史

現在の米国におけるＡＩ研究の主流，つまりＭＩＴ，スタンフォード大，ＣＭＵの３大学の現在までの研究の足跡，そして各々の大学のリーダーであるＭｉｎｓｋｙ，ＭａＣａｒｔｈｙ等の，果たしてきた役割が述べられた．

Ｆｅｉｇｅｎｂａｕｍが，知識工学という言葉を初めて使用し，それが一般に受け入れられ，その研究の産物として生まれてきた実用システムが，エキスパート・システムであり，知識ベース・システムであると位置付けられる．

３．エキスパート・システムの特徴

エキスパート・システムの主要な特徴として，以下の５つが示され，その詳細が述べられた．

（１）ＡＩに於ける，記号処理，推論，探索，ヒューリスティックのような典型的な手段を駆使する．

（２）限定した問題領域を対象とし，そこでの固有な知識（専門知識）を利用する．

（３）あい昧さを含む知識を取り扱い得る．

（４）知識と問題解決手続きとが分離されている．

（５）システムが行う判断に関しシステム自身がそこに至る過程，理由を説明する能力を持つ．

エキスパート・システムを「単に知識ベースに格納された知識をそのまま取り出すだけでなく，推論を通してより豊かな知識として活用したり，与えられた問題に対する解決を提供するもの」として定義がなされた．

４．エキスパート・システムの構成と知識表現

典型的なエキスパート・システムの構成は，知識ベース部，知識獲得部，推論機構部，説明機構部，ユーザ・インタフェース部とから成る．最近の傾向として，人間への心理的側面からも，説明機構部，およびユーザ・インターフェース部の機能が重要視されて来ている，と指摘された．

知識表現については，ルール表現，フレーム表現，述語論理，意味ネットワーク，黒板モデルが掲げられた．

ルール表現ではプロダクション・システムの例としてＭＹＣＩＮを引用し，「風が吹けば桶屋がもうかる」という諺を，ｉｆ～ｔｈｅｎ形式のプロダクション・ルールとして書いたものを用いてわかりやすく解説された．

フレーム表現ではＭｉｎｓｋｙのフレーム理論等を背景に，そのフレーム，ＡＫＯ（Ａ－Ｋｉｎｄ－Ｏｆ）あるいはＩＳＡ（ＩＳ－Ａ）スロット，ファセット等の意味と関連が解説された．

５．エキスパート・システムの効能

エキスパート・システムの効能として，

（１）希少な専門能力の保護

（２）知識の（明示的な）形式化・体系化

（３）専門能力の均質化

（４）（専門家の行う作業の中の）誤り易く退屈な仕事の自動化

の４つを期待できる効果として，それぞれに具体的な例が示された．

６．専門家との関係

専門家とエキスパート・システムとの役割関係については，エキスパート・システムを，専門家の代替・補助・教育（育成）のいずれに対応させるかといった問題がある．現時点では，代替する程の能力を持ったエキスパート・システムがまだほとんどないこともあって，あまり問題になってないが，今後，エキスパート・システムが専門家の代替を努められるほどの能力を有するようになると，はたしてエキスパート・システムの開発に専門

家の協力が得られるかどうかといった，微妙な問題も生じかねない．

7．エキスパート・システムの現状

まず現状のフィールド調査については，その説明を行うにあたって，適用分野を，医療，科学，機械・プラント，鉱物・石油資源，航空・運輸，電気・コンピュータ，ビジネスの7分野に分類し，各分野の概要と代表的システムの例が示された．なお，（）の中の数字は，大体のルール数を表している．

【医療】

MYCIN（600）・・・血液の感染症の診断と治療に関する助言

INTERNIST（15，0000）・・・内科全般に亙る診断，（開発進行中）

【科学】

MACSYMA（ルール・ベースではない）・・・数式処理，かなり大規模

DENDRAL（400）・・・有機化合物の分子構造の決定

【機械・プラント】

ACE（100）・・・電話ケーブルの保守・管理，AT&Tで開発，大規模データ・ベースと接続した実用に徹したシステム，バッチ・モードで動く．

【鉱物・石油資源】

DIPMETER ADVISOR（90）・・・油井のログ・データから地層構造を解釈，シュルンベルジュ社で開発，現在盛んにフィールド・テスト中

PROSPECTOR（2，000）・・・鉱物資源の鉱脈探査の助言，SRIで開発されたが，現在では保守・利用されてはいない模様

【航空・運輸】

CATS－1（530）・・・機関車の故障診断，GEによる実用化を目指したシステム

【電気・コンピュータ】

DART（不明）・・・コンピュータ回路の故障診断，IBMの委託によりスタンフォード大で述語論理推論システムMARS上で開発

XCON（5，000）・・・顧客向けにVAXコンピュータのシステム構成を選定，おそらく最も実用的に使われているシステム

【ビジネス】

実験段階のものが多く，余り本格的なものはない．EPISTLE（IBM，文書作成支援），AUDITOR（会計監査支援），税金相談，法令判断などの実験システムがある．最も期待されているのは，為替ディーリング，融資判断，投資選択などの金融業分野への適用で，実際，SRIのPROSPECTOR開発者たちの多くもこの分野にスピンアウトしている．

実際に米国で調査した結果として，現存するエキスパート・システムのうち実用レベルに達しているものの割合はかなり低い．その理由として，

（1）知識工学の歴史が浅く，技術蓄積が十分でないこと，

（2）産業界が本格的に開発に着手してから日が浅く，実用レベルまでの完成に至っているものが少ないこと，

等があげられる．

また機能からみると，一般に解釈，診断などは比較的実現しやすく，一方，予測，設計などはむずかしく，つまり，機能として創造性を強く必要とするほど，エキスパート・システムの実現はむずかしくなる．

（ここで会場より「株の相場予測のようなことはできないか」との質問があった．玉井さんは「現在ではちょっとムリですね，それができるなら私はここでこんな話はしていませんが．．」とのことに場内大爆笑）

8．開発上の問題点

開発のアプローチとして，知識がすでに体系化されている分野から手を着けるべきで，エキスパート自身がどのような知識の基に判断を下しているのか明確化していない分野，あるいは対象分野に人間的要素が大く含まれるもの等は，今後の研究課題であり，実用を意図するなら避けた方がよい．

さらに要員確保（いわゆるKE：Knowledge Engineerのこと），必要なハードウェア／ソフトウェアについてなど，実際の開発経験からの意見も混ぜた（玉井さんはエキスパート・システム・シェルのZeus開発に関与）話があった．開発コストの目安として，5人年工，または，$3，000／ルールといわれている．たしかにエキスパート・システム構築には時間とオカネがかかるようである．

（レポータ：藤野晃延）

SEA & jus 春のセミナー・ウィーク

セッション S6

オブジェクト指向プログラミング

米沢明憲

東京工業大学

1．はじめに

講義では，非常にていねいにオブジェクト指向プログラミングの考え方について説明があった．全体の要約は無理だが，私が理解し得た（と思う）ところを報告する．誤りがあれば当然私の責任である．

2．オブジェクト指向プログラミングの考え方

まず，オブジェクトとは一体何だろう，ということから話が始まった．セミナーのテキスト［1］から引用すると，「オブジェクト指向プログラミングにおけるオブジェクトとは，問題とその解決をできるだけ素直に表現・記述するために導入された概念で，実際の問題領域に登場するもの（ｔｈｉｎｇ）がもつ機能や知識をモデル化した概念的実体（ｃｏｎｃｅｐｕｔｕａｌ　ｏｂｊｅｃｔ）である」ということになる．

プログラムは，現実の世界を計算機上でモデル化しシミュレートするものだ，と考えられる．われわれが物事をとらえるときは，たいてい（想像の産物であれ現実のものであれ）具体的に目で見，手で触れることができる”もの（オブジェクト）”をイメージして考えている．この，”もの”を主体としてモデルを作り，これに基いてプログラミングするというのが，オブジェクト指向プログラミングの考え方である．

こうすることにより，より人間の考え方に近い，問題やその解決の論理構造を素直に表現したプログラミングができる道が開けてくる．

プログラムを作る場合，データ構造とそれを扱う手続き，処理の流れ，データの流れといったもので問題をモデル化，抽象化する．このとき，手続き，データのモジュール化，カプセル化という概念が非常に有用である．手続きやデータを抽象化してモジュールし，またカプセル化することを実現するために，抽象データ型というものが考えられ，このしくみを提供する言語がいくつも生まれてきた．オブジェクトは，この抽象データ型をさらに進展させたものであるともいえる．

抽象データ型の場合は，あるクラスのデータに対する演算群（手続き・関数）を用意しておき，それらの演算によってのみそのクラスのデータを操作できる．利用者には，データの内部表現はまったく見えず，外部へのインタフェースとして定義された演算群の仕様のみが見える．オブジェクトもこのような手続きやデータの抽象化，カプセル化，モジュール性などの概念を持っている．

オブジェクト指向プログラミングの場合は，まず”もの”を主体として抽象化を考えるという基本的な抽象化の方針が第一にある．この抽象化された”もの”がオブジェクトであり，抽象データ型における，あるクラスのデータとそれに対になった演算群に対応する．しかし，オブジェクトの場合は，データというものをより主体的，能動的なものと見なし，それ自身で独立したプロセスのようなものと考える．

データ（オブジェクト）に対する操作は，それに対してメッセージを送って行うという考え方をする．オブジェクトはメッセージを受け取ると，それにしたがって，何らかの動作を行い，内部状態を変化させたり必要に応じ返事のメッセージを返す．計算はどうするかといえば，オブジェクト同士のメッセージのやり取りによって行う．各オブジェクトは独立並行に動作をすることができる．このようなオブジェクトに基いた計算モデルは，ＡＣＴＯＲ［2］によって厳密に記述される．

講義では，オブジェクトをわかりやすく説明する具体的な例が，いくつか示された．オブジェクト指向の考え方を用いてプログラムを作成した例としては，同じくセミナーのテキストとして使われた文献［3］を参照していただきたい．

3．機能／知識の分割・分散

オブジェクト指向の考え方を用いることによって，さまざまな利点が得られる．講義では，機能や知識の分割といった観点から，オブジェクトという概念の持つ有用な特質として，以下のものが示された．

（1）機能／知識の切り分けが柔軟であること

これは問題の性質に合わせて，何をオブジェクトとするかを柔軟に決定できるということである．逆に何をオブジェクトと思うかによってプログラムの作り方に決定的な影響を及ぼすことにもなってしまう（これはプログラマの腕次第？）．

（2）同じ性質／振舞のオブジェクト群のグループ化

これは共通の機能や知識をグループ化し，使用上も管理上も扱うというもので，部品化や再利用を考える上で重要となる．オブジェクト指向ではクラスとインスタンスという概念があり，グループ化が非常に自然に行える．

人間は，複数の概念間で共通しているところがあればこれらをまとめて，さらに抽象度の高い概念を作りあげてこれらを整理する，といったことを行っている（この”概念”がオブジェクトに相当する）．たとえば，わが家のポチも隣のタローも犬といったぐあいである．ここでポチやタローをインスタンス，犬をクラスと呼ぶ．つまりクラスは，インスタンスを抽象化したものである．いくつかのクラスをまとめて抽象化することも行われ，この場合抽象化されたものをそれらのクラスのスーパークラスという（動物は犬のスーパークラス）．オブジェクト指向プログラミングを提供する言語には，何らかの形でこの機構がそなわっている．

（3）知識／機能の共有ー継承(inheritance)

これは（2）のクラス，インスタンスの概念から自然に導かれる．上位クラスは，下位クラス（あるいはインスタンス）に共通な性質をまとめたものである．下位クラスやインスタンスの記述をする際，まったく同じことを書かなければならないとしたら非常に不便である．そこで，下位のものは上位のものの持つ知識や機能を受け継ぐと考える．これを機能や知識の継承（ｉｎｈｅｒｉｔａｎｃｅ）という．

ここで知識や機能といっているのは，あるメッセージのパターンを受け取ったときの，それに対する所定の動作のことである．継承により記述量が減るほか，機能の変更，修正が容易になるという利点がある．しかし継承関係が複雑になって来ると，上位のクラスの機能を修正することにより，どれだけの波及が下位に及ぶかを把握することが，困難になって来る．特に多重継承（複数の上位クラスを持つ継承）の場合むずかしいことになる．また，下位クラスのオブジェクトの定義が逆に困難となり，継承が足かせともなりかねないので注意を要する．

継承の実現のしかたには色々な方式があり，システムによって考え方異なる［4］．しかし，何らかの形で継承という概念は取り入れられており，オブジェクト指向プログラミングの大きな特徴の一つとなっている．

（4）型多形態(type polymorhism)

オブジェクトには，型（ｔｙｐｅ）という概念がない．たとえば，表のオブジェクトやキューのオブジェクトを考え，同じメッセージ”項目数は？”に対して表に入っている項目の数やキューにたまっている要素の数を返答する機能を定義することができる．

型がないので型検査という概念もない．したがって，汎用的なオブジェクトを定義するのも容易である．たとえば項目数や要素の順序関係の定義されたオブジェクトを与えられて，それをソートするというオブジェクトを作ることができる．型概念がないことは，システムの拡張も容易にするが，反面でコンパイル時の静的な型検査によるエラーの検出が困難となる．これは，今後解決されなければならない問題である．

４．おわりに

セミナーでは，今まで述べたことのほかに，オブジェクト指向の考え方を，ハイレベルな実行可能仕様の記述に使うといった応用や，分散型ＯＳの記述等の，興味深い話題や，講師の米沢氏らが開発した並列型オブジェクト指向言語ＡＢＣＬやその背景にある計算モデルについての説明など，おもしろい話があったが，これらについてはこの報告では触れられなかった．

［参考文献］

［1］米沢明憲：オブジェクト指向型プログラミングについて，コンピュータソフトウェア，Ｖｏｌ．１，Ｎｏ．１，ｐｐ．２９－４１（１９８４）

［2］米沢明憲：ＡＣＴＯＲ論理について，情報処理，Ｖｏｌ．２０，Ｎｏ．７（１９７９），ＰＰ．５８０－５８９

［3］柴山悦哉，松田裕幸，米沢明憲：並列オブジェクト指向言語ＡＢＣＬによるプログラミング，”オブジェクト指向”鈴木則久編，共立出版（１９８５）

［4］大里延康：Ｓｍａｌｌｔａｌｋ－８０ ｖｓ Ｆｌａｖｏｒ，ＬＯＯＰＳ，ＴＡＯ，Ｃｏｍｐｕｔｅｒ Ｔｏｄａｙ，Ｎｏ．４，（１９８４），ｐｐ．４２－５３

（レポータ：沢田寿実）

SEA & jus 春のセミナー・ウィーク

セッション S7

ソフトウェアの著作権

石原寿夫

ソフトウェア流通促進センター

１．はじめに

本セッションでは，ソフトウェアの著作権について，

・ソフトウェアの法的保護のこれまでの経緯

・ソフトウェア法的保護の現状

　著作権法の一部改正

　プログラム著作物の登録

・国際動向

・今後の課題

のテーマで講演が行われた．

２．ソフトウェア法的保護のこれまでの経緯

昭和４６年，ＩＢＭから米国特許局に提出された”ギャルビー試案”が日本政府に入り，それまで特に検討されていなかったソフトウェアの法的保護に関して，国内での検討が始まった．

この試案では，実質審査のない簡単な手続きで所有者に無断でする複製・翻訳，または使用，もしくは譲渡からプログラムを保護し，しかもそのプログラムと類似のものが独立して作成される可能性を考慮して，所有者に絶対の権利を与えない方法で，プログラムを保護する登録制度の設定を主張している．

この試案に対して，ソフト協（現在のＪＩＳＡ）がソフトウェア業界の意見をまとめ，政府に提出した．このソフト協レポートは，

・使用権の設定

・権利保護期間の短縮

を主張した．このレポートを参考にし，文化庁と通産省から中間報告書が提出された．

この頃，「タイトー事件」（スペース・インベーダ複製）について，東京地裁により「本件プログラムは，作成者の独自の学術的思想の創作表現であり，著作権法上保護される著作物に当たる」との司法的判断が，プログラムに対して初めて出された．

その後，通産省は，通産大臣の諮問機関である産業構造審議会情報産業部会ソフトウェア基盤整備委員会において，文化庁は，著作権審議委員会第６小委員会において，検討が重ねられた．

この通産省と文化庁との基本的な考え方の違いは

・プログラムは産業所産か文化所産か

・オブジェクト・コードはヒューマンリーダブルか

である．ソフトウェア業界は新立法化である通産省案を支持し，学識者は文化庁を支持した．いわゆるプログラム権法騒動が始まった．結果的には，米国通商局の介入により日米貿易摩擦の余波をうけ，政治的決着により，著作権法での保護となった．

３．アメリカ政府の方針

米国政府は，いくつかの理由から，コンピュータ・ソフトウェアを日本において著作権法による保護から排除する提案について深い懸念を持っていた．

第１に，著作権保護から著作物を排除することは，万国ベルヌ両著作権条約に含まれる国際著作権法の精神と規定を侵害することとなろう．

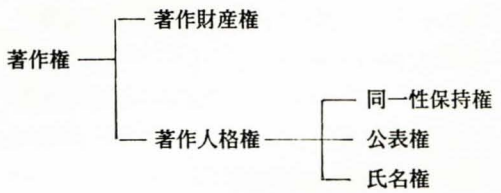
第２に，この提案の特定の条項は，実際に，日米両国のソフトウェア創作者に実質的な不利益を与え，したがって，日本のソフトウェア産業の継続的発展を促進するどころか疎外することとなろう．

第３に，コンピュータ・ソフトウェアの保護だけのために新しい法体系を創設することは，日米両国を含む工業国と世界中の取り引き相手国の長期的利益に反する．

４．ソフトウェア法的保護の現状

（１）著作権法の一部改正

現在ソフトウェアは著作権法下での保護を受けて，



から成り立っている．同一性保持権等ソフトウェアには，無理なものもあり，ソフトウェア保護のために，以下の

点が改正されている．

ａ）プログラムの定義を定めること．

ｂ）著作物の例示にプログラムの著作物を加えるとともに，それに用いられているプログラム言語，規約および解法は著作権法で保護するものでないことを明らかにすること．

ｃ）法人等の発意に基づきその業務に従事する者が，職務上作成するプログラムの著作権は，作成時の契約等に別段の定めがない限り，その法人等とすること．

ｄ）特定の電子計算機において，より効果的に利用し得るようにするために，必要なプログラムの改変について，同一性保持権の例示を定めること．

ｆ）プログラムの著作物の複製物の所有者が，電子計算機において利用するために必要な場合に行う複製または翻案について，著作権を制限すること．

ｇ）プログラムの著作物の創作年月日登録の制度を設けるとともに，その登録に関して必要な事項は，別に法律で定めることとすること．

ｈ）プログラムの著作物の著作権を侵害する行為によって作成された物を，業務上電子計算機で使用する行為は，当該複製物の使用の権原を取得した時に情を知っていた場合に限り，著作権を侵害する行為とみなすこと．

（２）プログラムの著作物の登録

著作物としての権利発生時期を，プログラムの登録制度により定義する．ベルヌ条約では，無方式（創作時に自然発生）であるが，万国著作権条約では，形式的要件（登録制度）により定義している．なお，この制度は，来年４月施行予定であり，現在国会に提出されている段階である．その概要について文化庁の説明によれば，以下の通りである．

・プログラムの種類

ａ）アプリケーション・ＯＳ等の種類は問わない．

ｂ）ソースコード，オブジェクトコードのいづれでも可能である．

ｃ）著作物としての一定のまとまりがあれば，モジュール単位でも可能である．

・登録の種類

ａ）実名の登録

無名または変名で公表された著作物の著作者は，現にその著作権を有するかどうかにかかわらず，その著作物についてその実名の登録を受けることができる．

ｂ）第１発行年月日等の登録

著作権等は，その著作物について第１発行年月日の登録または第１公表年月日の登録を受けることができる．

ｃ）創作年月日の登録

プログラムの著作物の著作権は，その著作物について創作後６ヶ月以内に限り，創作年月日の登録を受けることができる．

ｄ）著作権の転移または処分の制限，著作権を目的とする質権の設定，移転，変更もしくは消滅，または処分の制限について登録できる．

３．国際動向

フランスは，保護期間を２５年と定め，プログラムは応用美術との観点から，ベルヌ条約の規定を侵害しないと考えている．

オーストラリアでは，新立法化の動きがあり，日本でのチップ法と同程度である．

４．今後の課題

６１年度は，文化庁第９小委員会で検討が始まる．また，シグマ計画でのソフトウェアの著作権などいろいろ問題はある．著作権法下での保護では，まだプログラム翻案の範囲など，具体的な詳細についての基準が明確でないところがあり，業界としては個々の判断ができない部分がある．そのため，判断できない部分を洗いあげ，再度検討していきたい．また国際問題の解決へ一翼をになうように，日本がならなければいけない．持ってほしい．

５．感想

得てして面白くない法律の話をアカデミックでなく，本来の意味合い，成り立ち，認識等を大変面白く聞くことができた．年々，法律が改正，整備されてはきているが，技術の進歩に伴って，思いもよらぬ問題が出現する可能性は否定出来ない．本来，産業が発展していくための，阻害要因を取り除くことが，法律の役割でなければならない．

要は優秀なソフトウェアを開発し，ユーザが安心して利用できるような環境を整備することである．そのためには，技術的実態と法律構造のむすびつきをわれわれ自身が考え，常に問題意識を持っていなければならない．

（レポータ：関崎邦夫）

SEA & jus 春のセミナー・ウィーク
セッション－Ｓ８

ソフトウェアの品質保証と管理

小室豊
東芝エンジニアリング

１．はじめに

このセミナーでは，

・品質とはどういうものか
・品質について業界ではどういうふうにとらえているか，つまりどういう認識か？
・品質保証については，どういう体制にあるのか？

ということで講演がすすめられた．

ソフトウェアの品質問題が論じられて久しいが，品質管理（ＱＣ）方法や品質保証（ＱＡ）方法が確立したとは耳にしていない．それは，ソフトウェア自身が持つ特性のためで，固定した方法論を確立させることが困難なためと思われる．

このＱＣ／ＱＡ方法を確立させるために，ハードウェアの手法を導入すべきであるという考えのもとに，いろいろな試みが行われている．しかし，決定的な成果は期待できないようである．それは，ハードウェアが理論的に定量化できるのに対して，ソフトウェアの技術は，芸術や音楽と同じような創作である，という人間の個性や能力によって決まるヒューマン・ファクタが大きな部分を占めているからである．

現在行われているハードウェアのＱＣ／ＱＡ手法の応用は，設計された仕様書にもとづいてコード化されたり，テストを行うことである．ここの部分は，ソフトウェア技術の中でも，比較的機械化や標準化が進んでいるところである．

これ以外の工程に関しては，前述したヒューマン・ファクタが大きな部分を占めているために，ほとんど応用できないというのが現状である．

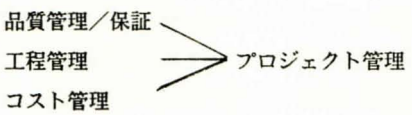
２．ソフトウェアの品質について

（ａ）定義を考える

ソフトウェアの品質について，次のようなことを念頭において，品質保証と品質管理を考えた方がよい．

・品質保証とは，一般に「ユーザの要求する商品・サービスを保証する生産者の体系的活動」であると考えられる．
・ＪＩＳでは，品質管理は「買手の要求に合った品質の製品を経済的に作り出す為の手段の体系」であると定義されている．

近代的な品質管理は，統計的な手段を採用しているので，特に統計的品質管理ということがある．これらを，ソフトウェアの世界でも用いてもよいのではということを示している．ただし，ソフトウェアの品質保証／管理は，それ単独では行えない．次のように，プロジェクト管理の一要素として考える必要がある．



（ｂ）品質項目

ソフトウェアの品質特性としては，

・ＴＲＷ社のＢｏｅｈｍによる品質特性
・ＩＢＭの製品品質特性
・Ｗｕｌｆの品質特性
・武田の品質特性
・花田の品質特性

等，さまざまな考え方があることが示された．これらの中から，Ｂｏｅｈｍの品質特性を見て感じることは，携帯性，信頼性等の縦のレベルまでは，なんとなく把握できるが，それ以上の詳細なことになると，なかなか理解できないことである．まだまだ，品質特性については，論議の対象となっている．

（ｃ）ＱＣ／ＱＡの特徴

ソフトウェアのＱＣ／ＱＡの特徴として，

（１）品質の作り込み
（２）品質（信頼性）の時間的変化
（３）品質（信頼性）とコストの関係

があげられる．

（1）については，特に，ソフトウェアの品質の向上のため，

・ユーザの要求の品質の作り込み

・不良点（エラー）の除去

があげられ，品質の作り込みに重点をおき，レビューの重要性が示された．

（2）については，ハードウェアのバスタブ曲線が，ソフトウェアの障害の発生についても，同様に考えられる．一般には，劣化現象はソフトウェアの製品が完成した場合はないとされるが，ここでは，ソフトウェアの改造ということを考えると，あてはまるとしている．

（3）については，一定レベル以上の品質を確保しようとすると，それに反比例してコストが上昇するという話であった．

３．ＱＣ／ＱＡの現状

品質について，業界ではどういうふうに取られているか，また，どういう認識かということについて，話がされた．

ここでの話は，主に，昭和５９年度情報サービス作業協会の「ソフトウェアプロダクト品質保証に関する調査報告書」に掲載されている，２２１社の技術者を対象にしたアンケートの結果を基になされた．

ここでの話で特に気付いた点は，次のようなことである．

・品質特性に対する重要性の認識として，携帯性，信頼性，効率性，操作性，理解性，更新性，テスト容易性，満足度の中から，信頼性についてがもっとも多かった点である．最近のＩＥＥＥの論文でも，信頼性の特集が取り上げられており，私なりに興味深い点であった．

・ＱＡの実施レベルは，一般的に，メーカーは，社会に取り組んでいると話を聞くが，それを裏打ちする結果を示していた．

・品質問題として取り上げられてから，１０数年たつが，ＱＡの活動上で最も問題になるのは，保証の基準がない，品質評価が難しい，とういうことが上げられており，ソフトウェアには人－ｈｕｍａｎ　ｆａｃｔｏｒが入り込んでいることで，さらに，品質保証をむずかしくしている．このことを，われわれは考えていかなければいけない．

４．ソフトウェアの品質管理

ソフトウェアの品質管理ということで，ソフトウェア開発におけるＱＣ活動は，プログラムの中のエラーを除去することが中心になっている傾向がある．しかし，それだけでなく，開発環境の整備，技術者の教育，設計手法の改善，テスト方法の開発など数多くの活動が必要となる．

代表的なＱＣ項目として，次のものが上げられる．

・要求仕様の的確な把握

・開発過程でのＤＲ

・データ収集と分析による問題点把握

・標準化

もっとも興味深いのは，一般に使用されている品質管理用の手法（ＱＣの７つ道具）だけでも，データの整理・分析に利用すると非常に便利であるということである．

次に，ＱＣの新旧の７つ道具をあげると，

ＱＣ旧７つ道具

・パレート図

・特性要因図

・グラフ

・チェック・リスト

・散布図

・ヒストグラム

・管理図

ＱＣ新７つ道具

・系統図

・マトリックス図

・マトリックス・データ解析図

・ＰＤＰＣ（過程決定計画法）

・関連図法

・親和図法

・フロー・ダイアグラム

のようになる．

最後に，当り前の品質から魅力的な品質へとういうことで，パッケージ・ソフトウェアによるユーザの要求を事前に組み込むＱＣ／ＱＡ活動が必要であることが述べられた．また，魅力的な品質を作り出す一方法として，数年前から品質管理学会を中心にソフトウェアへの「品質展開」手法の適用の研究が行われてきているし，今年４月よりＩＰＡの「品質評価」プロジェクトの１サブシステムとして，実験が試みられていることも紹介された．

（レポータ：芝原雄二）

SEA & jus 春のセミナー・ウィーク

セッションS9

テクニカル・マネジメント

岸田孝一

ソフトウェア・リサーチ・アソシェイツ

１．マネジメントの人格化

これまで管理というと，コストの見積り，工程の管理，品質管理といったことが多いが，これらはマネジメントの非人格化ということである．しかし，ソフトウェアのプロジェクトには大勢の人が関係する．そのために，マネジメントの人間化，個性化ということが大事である．

テクニカル・マネジメントとは，ソフトウェアを作るときに，どういう技法をどう取り入れたらうまくいくかということでなくて，ソフトウェアをつくる技術が非常に属人的なので，技術の人間的側面をどうやったらうまくマネジメントできるかということである．

この説の裏付けのデータとして，ソフトウェアの生産性向上に関する調査の結果でみてみる．生産性が向上したのと低下した原因では，大きなギャップがあった．それは，向上した原因として人の問題がなく，低下した原因として人の質等があげられたことである．

これは，失敗したプロジェクトのマネジャの言い訳が多いと思われるが，こうした生産性向上に関する人々の意識，ソフトウェア業界の人達の意識の現状をみてみて，結論としていえるのは，プロジェクト・マネジメントにおいて，マネジメントというのは技術的な色彩が非常に強いということである．その技術を重視してプロジェクトを進めないと，生産性向上は望めない．したがって，生産性を向上させるためには，技術的な観点でマネジメントをおこなわなければならない．

２．仕事の生き甲斐を与える

ソフトウェア技術者の生き甲斐というアンケートの結果では，「与えられた一つの仕事の完成」ということが一番になっている（これはアメリカでも同じ）．さらに，技術者たちは，自己中心的で，閉鎖的であるという結論がでている．それは，何か一つ仕事をして，その仕事を通じて自分が向上すればよいという考えかたである．

エンジニアに悩みをきくと，一番に出てくるのは，チーム・メンバーとの人間関係で，次がやりたくない仕事をやらされたり，同じような仕事をやらされる，という答えがかえってきている．

プロジェクト・マネジャはこうしたことを考えておかなければならない．つまり，自己中心的に動く技術者の集団をどうコントロールするか．自己中心的な人を相手にして，チーム・メンバーの信頼関係をどうやっていくか，ということになる．

こうしたことを逆手にとると，プロジェクト必勝法が考えられる．エンジニアは達成感を第一にしているのだから，達成できないような目標はあげない．さらに仕事を通じて新しい技術が習得できるか，少し難しい技術課題に挑戦させる．あまり簡単なところのコーディングなどは与えない．つぎに，仕事の目先を変えてあげる．つまり，同じような仕事を続けてやらせないようにする．さらに，チーム・メンバーとの人間関係に悩んでいることから，プロジェクト・マネジャは，普段から皆と酒を飲まなくてはいけない，ということになる．

技術をマネジメントすることは，人間を管理することにもなる．こうしたことから，プロジェクト・マネジャは，社会心理学を勉強しなくてはならない．

３．社会心理学からのマネジメント

プロジェクト・チームをつくったときに，マネジャが悩むのは，メンバの個人の能力差がある．

１９６２年頃にアメリカのサックマンがとったデータでは，コーディング能力で３０倍の差があった．そのとき，プログラマ適性検査の成績，大学のときの成績，経験年数といったいろいろなファクタは，生産性と相関関係がなかった．唯一，相関関係があったのは，生産性が高いプログラマが作ったプログラムほど品質がいいということであった．

しかし，プログラミングという仕事で，個人の能力の差が一概にとらえることができない．どういう仕事の与えかたをするかによって能力がかわる（この実験がワインバーグによって行われている）．

つまり，プロジェクト・マネジャが，部下にたいして何を望んでいるかということを，彼等がどうとらえるかということで，仕事ぶりが違ってくる．

4．目標を明確に示す

マネジャは，仕事をやるうえでその目標をはっきり示すことが大事である．ひとつのソフトウェアを作るという点においても，いいソフトウェアを作ればいいのか，それとも早く作るのがいいのかといったいろいろの目標が設定できるが，それを明確に示す．

しかし，指示をするときには，多くてもいけない．全部満足するようにというのは，何も指示をしないのと同じで，適当にやっていいというのと等しくなる．命令を受けて仕事をする人を考えたら，いろいろあるけれども，ウエイトの一番はこれだから，これを一番と考えて仕事をして欲しい，というように明確に指示をする．

5．マネジャはプログラムを読まなくてはいけない

一般的に，プログラマは他人のプログラムを読まない．これはプログラマの職業意識が未熟だからで（ＵＮＩＸが普及してきて，事情は変わってきつつはある），ものを書く技術をどうやって訓練するかというと，まず読むということがあって，そのうえで書くということがある．つまり，読めなくては書けないということになる．

マネジャもおなじで，部下の作った（作っている）プログラムを読めなければ，マネジメントはできない．読めなければ，何%おわったという単なる報告を受けるだけのプロジェクト管理になってしまう．

6．人間は誤りをするものである

プログラミングというのは，非常に人間的である．そのため，マネジメントは難しくなるが，人間は誤りをおかすものであるという前提で考える必要がある．

たとえば，レビューとかウォークスルーをおこなうとき，この前提を把握していないと，ミスをおかした個人への攻撃をもった査問会になってしまい，プロジェクトがうまく機能しなくなってしまう．

7．チームの問題

次に考えなくてはならないのは，チームの問題である．コンウエイの法則というのがあり，これは，あるソフトウェアの内部のアーキテクチャは，それを開発するチームの人間関係を正確に反映していなくてはならない，というものである．つまり，チームの誰がどこを担当するかということで，技術的確認が全員に徹底されていなくてはならず，機械的な分割はダメである．やりたくない仕事にアサインされたプログラマをどう心理的に救うかということも含まれる．

8．グループで大切なこと

ここでいうグループとは，ソフトウェア技術者の集団とか，ある部門・部署とかをさしており，このグループの中で自由にコミュニケーションができることが大切である．つまり，組織風土として，同じ人間だから誤りがあるということで，おたがいにフランクに仕事ができる，ということが大切である．

9．チームには父親と母親が必要である

グループの中のチームを管理するためには，チームのリーダシップが重要になってくる．目標の設定とその徹底を行わなければならない．

そのために，チームには２種類の性格をもったリーダが必要である．つまり，技術的にひっぱていく父親と，チームをまとめていく母親である．

また，チームを運営していくうえで大切なのは，うまくいくという前提で役割を分担するが，かならずなにか問題がおこることを想定して，そのための立て直し策をはかっておくことである．うまくいかなくなったときに，どこで歯止めをするかを，あらかじめ考えておく．

１０．キーマンを作らない

プロジェクトが存続している間に，人間的環境が変化するが，そのために技術的中心人物をつくらないようにする．その人間がいないと，プロジェクトがうまくいかなくなる，ということは避けなくてはならない．

１１．感想

講演はこのあとケース・スタディということで，これまでに述べられたことの実際例が，判りやすく話された．

岸田さんの講演で定評があることのひとつに，講演する内容が豊富な知識と経験の裏打ちがあり，聞くものに説得力があるということである．今回もその例にもれず，というより，人間を扱うのが重要であるということからも，その経験からの言葉に説得力がとくに感じられた．このレーポートでそのへんの感じが伝えられないのが残念である．秋のセミナー・ウィークで再演をお願いして，多くのマネジャに聞かせたい．

最後に，この講演とは別にＳＥＡの「管理分科会」の席上で岸田さんがいったことばで終わりにする．「管理者とは，野球の監督みたいなもので，選手であるエンジニアたちの力を発揮させてやることが大切である」．

（レポータ：野辺良一）

SEA & jus 春のセミナー・ウィーク

受講者の声

以下に収録した受講者の声は，セミナーを受講をされた方々から頂いたアンケートを編集したものです．

アンケートの結果は，全体的には好評でしたが，今後のセミナーやフォーラムの参考にしたいと思います．

セッション：A0

－アメリカにおけるソフトウェア工学の最近の状況がよく判った．外人講師の場合，通訳が入るので，実質的に日本人の場合の半分になる．今後は一日コース（６時間）としたら，突っ込んだ内容を聞けると思う．特に今回の講演の内容がよかっただけに惜しまれる．

－最近，いそがしさにかまけ，外からの情報を得ていませんでした．久々に外気にふれる機会を得られたことを感謝しています．

－現在自分が働いているセクションにかなりかかわることが多かったので，有効的だった．午後のパネルは，もうすこしテーマを絞ったほうがよい．

セッション：S1

－Q&Aで，さまざまな会社の人たちが，それぞれの立場でシグマ・プロジェクトに関心をもっていることがうかがえた．内容的には，もっとシグマ特有のトピックを期待していたのですが．

－講師の声が小さく，ハッキリしない話し方で非常に聞きずらかった．また，ＯＨＰの文字が小さく，よく見えないので，コピーするなりして事前に配布してほしい．

－一人で話して一人で納得しないでほしい．いくら開発途中であるとはいえ，訴えるものがあるはずである．もうすこし，効果的なプレゼンテーションを考えてほしい．

－もうすこし具体的な所が固まったころに，また聞いてみたい．そのときは，ＯＨＰで使う資料は事前に配布してほしい．

セッション：S2

－既存ツールの紹介だけでなく，今後のありかたとして試行・希望等があり，ツール開発にたずさわっている者として，大変参考になった．今度はぜひ大阪で．

－再利用の概念的なことと，具体的な例がうまくバランスがとれていて，わかりやすかった．セミナーは，たいていどちらかに偏るものだが，それでは退屈するから，ちょうどよっかた．　－テキストにないＯＨＰがあったが，そちらのほうがおもしろかった．これに関しては，テキストにしてほしかった（とくに要点をまとめた絵）．

セッション：S3

－開発環境を検討するうえで，参考になりました．今後，ソフトウェアの概念について，より厳密な定義ないしは，共通の意識が必要になろう，ということを感じた．私は，ソフトウェアの中に，作成者のアイデア，ノウハウもでも含めて考えるべきだと思っている．

－ソフトウェア・データベースの概念がほぼ理解できた．ただ，理論的な解説が主だったようだが，現実的な話と，理想的な話を整備したようがいいのでは．

セッション：S4

－他の会社の利用状況等が，とても参考になった．

－Ｊ－ＳＴＡＲに惚れた人が日本にもいることが判って

嬉しかった．私も国内外合わせてＪ／Ａ－ＳＴＡＲを約７０台，ファイルの容量はGの単位になっているＳＴＡＲ　Ｓｙｓｔｅｍを扱っています．盛田さんの御苦労はよく判ります，頑張って下さい．

セッション：Ｓ５

－全くエキスパート・システムについての知識なしで受講したのですが，説明がひじょうに理解しやすく，ふんいきだけでもつかめました．

－話題になっているエキスパート・システムについての概要を知ることができた．全体的にまとまっていて，ポイントが判りやすく，最後まで興味を持って受講することができた．今度は，もう少し詳しい話を聞きたい．

－エキスパート・システムをまったく知らない人に対して説明するときの助けとなる．無理な注文かと思いますが，実際のエキスパート・システムのプロトタイプの実演などがあれば，もっとよかった．

セッション：Ｓ６

－オブジェクト間のメッセージ・パッシングについて知識の整理ができた．

－オブジェクト指向は言語としてとらえるのでなく，ひとつのシステムの記述方法であるという点が明確になった．このような新しい考えかたを教えるときは，ある問題にたいして，他の方法との比較をあげてほしい．

セッション：Ｓ７

－非常に楽しく聞かせていただきました．もっと多くの人が，これらの問題に関心を持つべきだと感じると共に，自分自身の不勉強も強く感じました．

－著作権法とソフトウェア保護法との間の検討過程がよく判った．ソフトウェアには，プログラムとマニュアルから構成されており，もうすこしマニュアルの面から見た著作権法との関連も知りたかった．

－法律整備の背景というか，文化庁，通産省などの実情的な話がわかって，興味深いものがあった．石原さんのレクチャーは，大変判りやすく有益でした．できれば今後も，新しい動きがあった場合，フォローしていただきたいと希望します．本講演内容は，大勢の人に聞かせたいものである．ぜひ，Ｉ，ＩＩといったカリキュラムを組んでいただき，再講されたらよいと思う．

セッション：Ｓ８

－ソフトウェアの品質について新しい視点を得た．ソフトウェアは，結局人に帰着するのだと感じている．そうした意味からも，興味深いデータがでていて，おもしろく聞かせていただいた．

セッション：Ｓ９

－プロジェクト・マネジメントにおいて，心理的な側面の重要性が感じられた．もっと長い時間聞きたかった．

－豊富な実例や経験が話題の中心だったので，理解しやすく，大変参考になった．また，同じテーマできいてみたいと思います．

ＳＥＡでは，今年９月末に，「秋のセミナー・ウィーク」を企画しています．

この企画に参加するボランティアを募集することにしました．セミナーなどの企画に興味のある方の参加を待っています（ボランティアには，セミナーを無料で聴ける特典があります）．申込は，ＳＥＡ事務所まで郵便で申し出ください．お待ちしています．

CALL FOR PAPERS

Conference on Computer–Supported Cooperative Work

December 3–5, 1986 ● **Austin, Texas**

Sponsored by

Microelectronics and Computer Technology Corporation (MCC)

Software Technology Program (STP)

This conference takes an interdisciplinary look at computer-supported cooperative work from technological, sociological, organizational, cognitive and task domain points of view. It grows from two past conferences: the DEC/MIT Workshop on Computer-Supported Cooperative Work in August, 1984 and the MCC Interdisciplinary Design Symposium in May, 1985. The previous conferences drew participants from computer science, organization design, cognitive science, sociology, artificial intelligence, design theory and practical engineering disciplines. We hope to incorporate an even broader range of research and application perspectives on groups and group work at this meeting.

In order to encourage an informal and informative atmosphere, the conference's size will be limited. The program will include invited speakers, paper sessions, panel sessions and informal interest groups. We invite proposals for panels and discussion groups as well as papers.

Suggested Topics

We are soliciting new papers on the following representative topics which include, but are not limited to:

* Experiences with technology for cooperative work
* Computer-based environments that support cooperation: co-authorship, project management, large-scale design of computer systems, planning
* Empirical studies of cooperation/teamwork
* Impact of computer technology on group behavior, organizational structures and work practices
* Underlying technologies: data bases, structured documents and hypertext, access controls and privacy
* Theoretical models for analyzing group work: "roles," communication protocols, coordination constraints
* Multi-media conferencing
* Group decision support systems
* Domain-specific requirements for computer-supported group work

Information for Authors

Fifteen copies (15) of a double-spaced extended abstract of 10–12 pages in length should be submitted to:

Dr. Irene Greif
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139
phone (617) 255-5987
e-mail: greif@mit-xx.lcs.mit.edu

People who have limited access to copiers, or for whom overseas airmail costs will be a burden, should submit only one copy.

Suggestions for panels and interest group meetings should be 1–2 pages long. Submit these short proposals either by sending fifteen copies to the address above, or by mailing one copy to the email address above.

A contract is under negotiation with a publisher for the conference proceedings which will be available at the meeting.

Important Dates

Submission deadline:	July 1, 1986
Acceptance notification:	September 1, 1986
Final version due:	October 1, 1986
Conference date:	December 3–5, 1986

Program Committee

John Seely Brown, Xerox PARC
Christine Bullen, MIT Center for Info. Systems Research
Paul Cashman, DEC
Bill Curtis, MCC STP
Clarence A. Ellis, MCC STP
Douglas C. Engelbart, McDonnell Douglas
George Huber, University of Texas
Thomas Malone, MIT Sloan School of Management
Margrethe H. Olson, NYU Graduate School of Bus. Admin.
Ben Shneiderman, University of Maryland
Mark Stefik, Xerox PARC
Lucy Suchman, Xerox PARC
Terry Winograd, Stanford University

Conference Committee

Conference Chair
Herb Krasner, MCC STP
Program Chair
Irene Greif, MIT Laboratory for Computer Science
Local Arrangements Chair
Bryan Fugate, MCC STP

For more information call Barbara Smith, MCC STP at (512) 834-3336 or by netmail to basmith@mcc.arpa

環境分科会（ＳＩＧＥＮＶ）

第３回会合報告

日時：昭和６１年３月１９日（水）
１９：００～２１：１０
場所：機械振興会館６階６１会議室
出席者：１８名

議事内容：
今回は担当幹事水谷時雄の司会で”現状のソフトウェア開発環境とその問題あるいは改善について”というテーマでフリーディスカッション形式で行われた．

１．はじめに

はじめて出席された方も多かったので，ＳＩＧＥＮＶの活動経過を久保宏志から説明していただいた．

２．メンバの環境に関する討論

１人につき約５分の持ち時間でメンバの自己紹介（＝環境の紹介）をした．

その後，各人の環境に対する問題点の解決を含めて討論を行った．時間が足りず十分に議論はできなかったが，これを通じて今後のＳＩＧＥＮＶのテーマを探ることもある程度できた．この中で，長岡のワークショップの報告が佐藤千明，久保宏志からあった．

メンバ各人の環境は，同じ会社の人がほぼ同じ環境を経験していたということを除いては全くバラバラであった（ＵＮＩＸ，大型汎用機，パソコン，Ｐｒｏｌｏｇマシン・・・）が，このことは会の運営をさまたげるものでは一切なく，むしろ当分科会の活動を触発させるものばかりであった．紙面の関係で個々に紹介することは出来ないのが残念であるが，まとめとしては，

・環境からみた言語の動向
・ＷＳ（ワークステーション）とホスト（ターゲット）との環境の融合
・ソフトウェア受託開発の環境（いかにターゲット環境の変化に対応してゆくか？），移植の問題
・パソコンでの環境はいかにあるべきか？
・大型汎用機（ＴＳＳ，ＶＭ／ＣＭＳ）とＵＮＩＸの環境の良し悪し

等が各人の共通した問題（キーポイント）であるようだ．いずれこれらを順次検討してゆくこととなろう．

３．次回の開催について

（１）第４回月例会
日時：昭和６１年４月１６日（水）
１９：００～２１：００
場所：機械振興会館　地下３階会議室
テーマ：Ｔｏｏｌｐａｃｋの勉強会
ＬＡＮを利用したソフトウェア開発の経験
（２）第５回月例会
日時：昭和６１年５月１４日（水）
１９：００～２１：００
場所：機械振興会館　地下３階会議室
テーマ：未定
連絡先：久保宏志－ＩＰＡシグマ・システム開発本部
０３－２５５－０４２１
（３）第６回月例会
日時：昭和６１年６月１８日（水）
１９：００～２１：００
場所：機械振興会館　地下３階会議室
テーマ：未定

（渡辺雄一）

お知らせ

ＳＥＡＭＡＩＬの１号，２号，３号の在庫が若干あります．
ご希望のかたは，１冊につき７００円（送料込み）の切手を同封のうえ，事務局までお申しこみください．

再利用分科会（ＳＩＧＲＥＵＳＥ）

第２回討論会報告および今後の予定

１．第２回討論会報告

４月２日（水）７：００－９：００，機械振興会館Ｂ３－１において，１９名の参加を得て第２回研究会を行いました．ＳＩＧＲＥＵＳＥの運営方針，ｂｉｔ別冊目次案の検討，および研究発表「現状における再利用支援ツール」とそれに対する討論を行いました．

その結果，以前パンフレットでお知らせした内容とほぼ同じですが，以下の通り決定しました．

１）編集会議（第１水曜日），研究会（第２水曜日），討論会（第４水曜日）の３つの会合を設ける．

２）世話人（阿部，青島，村井，大西），会計（村井）をおく．

３）編集委員，執筆者をおく．以下の章立てが検討の結果，ｂｉｔ別冊目次案として今後のたたき台とすることになった．

ソフトウェア開発の現状と問題点
ソフトウェア・プロセス・モデルと
　ソフトウェア環境
ソフトウェア環境に於ける
　ソフトウェア再利用環境の位置付け
ソフトウェア再利用環境の分類
　・再利用対象による分類
　・再利用方法による分類
ソフトウェア再利用環境の
　分類に従った具体例
ソフトウェア再利用環境を
　構築するための要因
　・標準化
　・ソフトウェアの標準化
　　・開発プロセスの標準化
　・ソフトウェア再利用とＡＩ
　・ソフトウェア・データベース
　・ソフトウェア環境のためのハードウェア／ＯＳ等

研究発表：
「現状における再利用支援ツール」と討論会

阿部氏が，現在使用されているツールでソフトウェア再利用の傾向を持つものについて，共通した考え方や方法論，ツールの体系，機能ならびに今後の方向性などについて発表を行い，それに対して討論を行いました．参加者のなかには，種々のメーカ系列のツールやソフトウェアハウスのツールを使用している人がおり，使用経験に基づいた具体的な話が聞けました．

２．今後の予定

現在決定しているＳＩＧＲＥＵＳＥとしての活動予定をしるします．まだ参加されていない方でも，興味のある方は是非参加をしてください．

１）第２回研究会
　４月２３日（水）７：００－９：００
　場所：機械振興会館 Ｂ３－９
　参加：自由
　内容：詳細化された目次案に沿った記載内容の検討
　連絡先：阿部 正平（０３－５０３－４９８１）
　協同システム開発（株）
２）第３回編集会議
　５月７日（水）７：００－９：００
３）第３回研究会
　５月１４日（水）７：００－９：００
　場所：機械振興会館 Ｂ３－９
４）第３回討論会
　５月２８日（水）７：００－９：００
　場所：機械振興会館 Ｂ３－１

３．問い合わせおよび連絡先

ＳＩＧＲＥＵＳＥに関する問い合わせおよび連絡先は，下記に願います．

連絡先：村井進（０３－４３３－８１７１）
日本コンピューター・システム（株）

（村井進）

管理分科会（ＳＩＧＭＡＮ）第１回研究会報告

４月４日（金）に，プロ野球開幕にあわせて，第１回研究会を行いました．この分科会には，現在１８名の参加申し込みがありますが，６名の参加者を得て，プレイボールとなりました．

第３回ＳＥＡＭＡＩＬでお知らせした話題に基づいて，進行していく予定でしたが，とにかく興味ある人が集まって，息の長い，実のある分科会を行うために，どのような事をやっていくかを，自己紹介を含めて話し合い，当面の運営方針と予定を決定しました．

１．自己紹介

各自のバックグランド，ＳＩＧＭＡＮに何を期待し，各自が何をやりたいかを自由に発表してもらいました．

前島仁：ソフトウェアの知的作業（思考を集中・持続・発揮する為に）は，心技体が必要であり，今，心技に興味をもっており，心とは，心理学，技とはソフトウェア工学，体とは，生体工学，とういうような世界である．

岸田孝一：各自が困っている問題を定量化されたデータと共に提案していただて，問題解決の為の議論をこの分科会でやり，各自の会社に持ち帰って役立たせてもらっては．

土崎直樹：パソコンによるソフトウェア開発管理システムを個人的に開発中で，現在基本設計の段階であり，ＳＩＧＭＡＮでの討論，事例紹介，文献紹介を基にして，科学的方法等を取り込んで行きたい．

高瀬尚彦：現在数万本のプログラムの保守管理に苦労しており，また，２年間で，２倍に増えた社員数の技術移転・管理にも苦労しているので，そのような管理の解決のヒントをＳＩＧＭＡＮで得られることをのぞんでいる．

野村敏次：生産性向上という事で，数年間，様々なデータを取っていたが，現在，様々な点について疑問を持っていのるで，それについての意見をいただけるような分科会としたい．

臼井義美：大阪の方で，今回，出席できませんということで，ＯＨＰを送ってくれました．商品開発プロジェクト管理の業務を担当しており，モットーとしては，楽しく，優雅に，儲けよう．

芝原雄二：現在，ソフトウェア工学グループの定量化プロジェクトの世話人的立場にいる．様々な文献の紹介を行っていけるのではと思っている．

２．運営方法

（１）方針

各自が困っている問題について，定量的データを基に，提案していただき，ＳＩＧＭＡＮで意見交換を行い，各自の問題解決に役立たせる．定量的データは，その場でＯＨＰで発表するだけにし，コピーも記録も取らないとういう紳士協定を結ぶ．また，報告者は，発表内容を自由に原稿にしていただく．

（２）運営

当面，会合は，月１回，第２金曜日　６：３０～８：３０とする．

３．その他

当面，事務局として，芝原が担当し，会計も兼ねる．世話人希望の方，芝原まで御連絡下さい．

連絡先：協同システム開発（０３－５０３－４９８１）

（芝原雄二）

次回会合

日時：５月９日（金）６：３０～８：３０
場所：ＪＳＤ会議室
話題：コスト管理の面から，ＪＳＤで収集したコストデータを紹介し，議論する
報告者：芝原　雄二
会費：１，０００円

第３回会合

日時：６月１３日（金）６：３０～８：３０
場所：ＪＳＤ会議室
話題：デザインレビューに関するデータの紹介
報告者：前島　仁

教育分科会（ＳＩＧＥＤＵ）

第１回　教育分科会会合報告

４月１２日（土）に，５名の参加を得て第１回会合を高円寺のＳＥＡ事務所にて開催しました．

分科会に申し込まれた方は総勢１１名だったのですが，所用のため当日欠席された方を気にしながらも，今後の運営方針，活動予定を決めました．

ちょうど今は各社で新入社員教育の真っ最中ということで，分科会はもっぱら新人類に関する文化的考察（？）の話題になるのではないかと思われましたが，中堅技術者教育，効果的な技術移転をどうするか，ＣＡＩなどの真面目な話で，延々と３時間近くつづきました．運営については，参加された方が何をやりたいのか，その素直な希望を大事にして，各人がそれぞれの立場で発展させることが出来るような，雰囲気作りを第一優先にしたいと考えています．

１．参加者自己紹介

参加されたのは，

河村一樹，松尾育子，杉田義明，大浦洋一，中園順三

以上の方々で，自己紹介に続いて教育に関し興味を持っている事柄について意見を述べた．

２．運営方針

ＳＥＡという団体をうまく活用し，参加者各人が関心のあるテーマを追求し，何等かのかたちで後に残すことにする．しばらくは効果的な教育や，技術移転に関する情報収集と意見交換を中心に運営する．

３．テーマ

関心のあるテーマとして：

- 新入社員教育
- 中堅社員教育，
- 技術移転
- ＣＡＩ

などが挙げられるが重なり合っている領域が多いため，当面は限定せず，各人の経験とアイデアの情報交換を狙う．

４．運営方法

月ごとに幹事役を決めて，その月の運営内容を一任する．

- 原則として月１回
- 平日の夜１．５－２．０時間程度

通常は幹事が設定した発表者（幹事自らでもよい）による発表と，討論が主体．年に２回は一般公開の討論セミナー，年に１度はワークショップを企画する．それぞれの活動毎に記録を残し，ＳＥＡＭＡＩＬに成果をまとめる．これらの成果物がソフトウェア技術者教育のガイドラインとして使われるのが理想の姿である．各月の幹事と一応のテーマは以下の通り：

- ５月：河村（新入社員教育）
- ６月：大浦（新技術）
- ７月：中園（中堅技術者）
- ９月：杉田（公開討論セミナー）
- １０月：松尾（新入社員教育の評価）

５．当面の計画

第一回発表会：

- 日時：５月１５日（木）
  １８：３０－２０：００
- 場所：機械振興会館：Ｂ３－８部屋
- 参加費：１０００円
- 発表者：河村一樹（日本電子専門学校）
- 内容：新入社員教育のカリキュラム，内容，効果測定，視聴覚教材の効果的利用についての経験発表と意見交換
- プログラム
  １８：３０－１９：３０：発表と討論
  １９：３０－２０：００：分科会運営に関する意見交換
- 参加資格：参加自由
- 連絡先：杉田　義明
  （ＴＥＬ０３－２３９－５４７３）

（参加については事前連絡は不要です）

６．会員の募集及び世話人について

教育に関して興味を持っている方の教育分科会への積極的なご参加をお待しております．現在世話人という名の雑用係を大浦（シーイーシー），と杉田（ＳＲＡ）が引き受けています．世話人をしてもよいと思われる方もお申し出ください．

横浜支部発足

大変素晴らしいニュースを皆さんにお伝えできる事を嬉しく思います．

そのニュースとはＳＥＡ横浜支部，名付けて「Ｙｏｋｏｈａｍａ　ＤＯＣ」が正式に開設の運びとなったことです．そして，その記念すべき第一回目の会合が来る４月２８日（月），場所もまたこれ相応しく横浜の名門ホテル，ニュー・グランドに於て開催されます．

支部としての誕生は関西支部に続いて２番目ですが，港Ｙｏｋｏｈａｍａらしく世界に開いた先進的な活動をして行きたいと思います．

とりあえずは，中華料理を食べながら今後の活動方針を決めたいと思います．

基本的には多摩川を境にして，コッチ側に住んでいる方，またはいは職場がある方を対象に考えていますが，勿論そこは世界に開かれたＹｏｋｏｈａｍａ　ＤＯＣの事，参加したいと云う方には資格を問わず門戸を解放して歓迎いたします．どうか，奮って御参加下さい．

ＳＥＡ東京本部，そして関西支部に劣らぬ活動を展開して行きたいと思います．我こそはとお思いのハマッ子諸子，ソフトウェア，コンピュータに関し意見・アイディアを有して居られる諸子は，是非是非，御参加下さいます様お願い致します．

以下にＹｏｋｏｈａｍａ　ＤＯＣ，第一回ミーティング開催の要領を示します．

日時：４月２８日，ＰＭ．１８：００～２０：３０
集合場所：横浜，ホテル　ニュー・グランド　１階ロビー
参加費：１名　￥５，０００円位（高くてゴメン！）
内容：Ｙｏｋｏｈａｍａ　ＤＯＣ　結成式　ｅｔｃ．
　　南京町（中華街）中心に夜の横浜探訪を予定
申し込み：４／２８（月）迄に下記宛て電話で申し込みか，または，現地（ホテル　ニュー・グランド）に直接おいでください．
　ＡＭ．１０：００－ＡＭ．１２：００
　ＰＭ．１３：００－ＰＭ．１７：００の間のみ
　　電話０３－４９６－８５２１（内線４５０）
　　担当：　藤野

ＳＥＡＭＡＩＬ次号予告

次号第５号は，５月中旬には皆様の手元に届くよう，編集長はガンバルといっていますので，ご期待ください．

◆誌上フォーラム
「ＳＴＡＲＷＡＲＳ」

◆Ａｐｒｉｌ　フォーラム報告
４月２４日に行われるフォーラムをレポートします．

◆プログラマ！！
１号，２号で好評だった，「プログラマ！！」の連載が再開されます．

◆文献紹介
次号から，ひとつのテーマ毎に，最新の，あるいは必読の文献の紹介を初めます．
次号では，「ソフトウェアの信頼性」の文献です．

◆ネットワーク・セミナー・レポート
昨年１０月に，ｊｕｓとの共催でおこなった「国際ネットワーク・セミナー」のレポートを会員の要望に答えてレポートをします．

◆会員の投稿
会員からの投稿を掲載します．

これからの誌上フォーラムの予定

５号以降の「誌上フォーラム」の予定は以下の通りです．

各テーマの会員からの投稿を待っています．会員以外からの投稿も歓迎します．なお，投稿は，ワープロ出力，または手書きの原稿を２，０００字でＳＥＡ事務所までお願いします．

◇第６号－「ソフトウェア産業論」
締切：５月１５日

◇第７号－「パソコン」
締切：６月２０日

◇第８号－「情報処理者試験の是非」
締切：７月１５

◇第９号－「ドキュメントの機械化」
締切：８月１５

究極のワークステーション
『楽夢駄』が切り開く
創造環境のニュー・パラダイム

『楽夢駄』は電子黒板の夢を見るか？

そのコンセプトの発表と共に，既成のワークステーション・メーカーを顔色なからしめ，世界に轟々たる反響を巻起こしたスーパー・ワークステーション『楽夢駄』．殺到する引き合いに，製造／発売元である（株）ＫＥＭＥＣは，２４時間体制で生産を続けている．ここでは，出荷を間近に控えた『楽夢駄』の驚異のスペックの数々を紹介する．

ハードウエア

表示システム

その数々の特徴のうち，最も異彩を放つのが表示システムである．ここには，ＣＲＴなどという，熱した電極から叩き出した電子を頼り無い磁力線で制御する様な野蛮なデバイスは存在しない．生物／物性に関する工学的知見が高度に応用された全く新しいものが採用されている．厚さ０．５ｍｍ広さ無限大のこの表示システムは，まるで，カーペットの様に単位面積あたりいくら，の方式で販売される．

『楽夢駄』を購入しようとする者は，先ず表示システムの大きさを決定しなければならない．表示システムは机の上，壁，天井，床，時計のバンド電気スタンドの傘など，あらゆる場所に設置することが可能である．柔軟な可塑性と，裏に強力な接着帯を有するため，**いかなる自由曲面上にでも**フィットさせることができる．人々は程無く，街中の至る所にこの表示システムを見掛ける様になることとなろう．

入力システム

現在以下の入力方法が可能である

伝統的入力方法
- マウス
- ライトペン
- ディジタイザ
- トラック・ボール
- キイボード
- ペダル
- ボイス（キイワード方式）

先進的入力方法
- ボイス（自然言語方式）
- ヴィジュアル・イメージ
- エキストラ・センサリィ・パーセプション
- 毛筆

伝統的入力方法であるキイボードやマウスなどにも，ユーザーの趣味に合わせた様々なヴァリエーションが存在している．例えば，キイボードは従来の様な一枚型から，両手両足で操作できるセパレートタイプ迄，基本タイプで１０種類，色合いなどを合わせると３００種類以上のものが用意されている．極端な話，ユーザーは，安楽椅子の上で腕組みをしながらでも入力操作を行う事ができる．

情報処理装置

使用されているのは，新設計の完全なる並列型プロセッサで，任意の個数の『楽夢駄』を並列に動作させる事が可能である．処理能力は一台当たり２００ＭＪＰＳ（Million Jokes Per Second）を誇っており，ユーザーの思考を中断しない充分な速度が確保されている．

ソフトウエア

基本ソフトウエア

高度なモデル操作能力がこのシステムの基本である．ＯＳそれ自身高度に抽象化されたモデルであるため，システムを再構成する事が非常に容易である．整備されつつある**知覚／表象／認知／記憶**のモデルは高度に知的で柔軟な処理を行うことを可能にしている．このシステムの登場によって世の中の雑用的な事務が減少していくのは必至であり，産業構造に大きな変革を与える事になるだろう．先日代々木公園で行われた全東京秘書同盟の抗議集会は読者の記憶に新しいものと思われる．

インターフェイス・ソフトウエア

高度なインターフェイス能力は，単に電算機同士のものに留まらない，通信の相手によって最も適当であろう表現モデルを使いながらインターフェイスを行うことが可能な為望むなら，音声で２台以上の『楽夢駄』を（しかも，自然言語で）対話させる事も可能である．簡単な用件の電話なら横着をして，『楽夢駄』に任せることもできる．勿論相手も『楽夢駄』に応答させていることは充分考えられる．

次なる世代へ

簡単な紹介ではあるが，『楽夢駄』の持つ革命的意義は御理解頂けたと思う．全世界の情報に自由に手が届く日が近付いている．

『楽夢駄』開発プロジェクトは１９８５年夏岩手県盛岡市で発足し，爾来７年の歳月をかけて終了した．当初のメンバー名は以下に挙げる諸氏ある（敬称略）．

松野圭子 目黒和博 畠山一郎 小松一彦
佐藤正美 大地英夫 山本修一 酒匂寛

文部省
コンピュータ言語審議委員会を設置か？！

言語の不統一による弊害を危ぐ

文部省の意向としては，諮問機関である国語審議委員会と同様の目的で，コンピュータ言語に就いても審議会を設けたいとしており，「乱れたコンピュータ言語」の是正に向け基準作りを行っていきたいとしている．

これは昨今のファミコン・ブームなどを背景にコンピュータ言語が予想以上に早く普及してきており，同時に言語の不統一や乱れによる弊害が目立って来ていることもあって，このまま看過することに，将来への影響を危ぐする声が内部に上がっているためとみられる．

言語の統一化へ

具体的な作業としては，「旧言語使い」の「新言語使い」への統一，発音に忠実な表記法への統一，翻訳の際の「送り仮名（ｏｐｔｉｍｉｚａｔｉｏｎのこと）」の付け方の統一，更にはいわゆる「当用言語」の設定まで持って行きたいとしている．

通産省は反対を表明

これに対しはやくもソフトウェア技術者協会（ＳＥＡ）等から反対の意見が表明されている．一方，ナワバリを荒された格好の通産省も今回の文部省の発表を「全く寝耳に水，ｖｉ（いま流行のＵＮＩＸの画面エディタ）編集中にマシン・ダウンが起きた様な感じだ」として不快感を隠せずにおり，「今回のは明らかに記憶保護例外，および特権命令例外だ」として直ちにａｂｅｎｄするよう要請した模様．

プログラマは大反対

またＳＥＡ等に属している純正ハッカー等は，「米国内で制定されつつあるＣｏｍｍｏｎ　Ｌｉｓｐの向こうを張って，日本国内に公文式プロログを制定したいじゃないの，一部教育関係者の不穏な動きがあるのを察知していたが，今回の文部省の発表はこれが事実である事を裏付けるとともに，自らが関係していることを表明したもので，ソフトウェアの未来を阻害する様な今回の発表は遺憾であり，全力で阻止する」としている．さらに，「特に新言語使いへの統一などは迷信でしかない，”プログラマ３０才定年説”を何とか定着させようとの魂胆がミエミエ，表記法や送り仮名の統一等は厳しいレスポンスの要求される実時間処理のデッド・ラインをクリヤーしようとしている者にとっては全くナンセンス，当用言語の設定に至っては，ＡＤＡの真似がしたくてしょうがないのだろうが，もともと問題分野毎に最適の言語を用いるべきであって，最初から一種類の言語で全て対応しようとするその根本概念からしてバグだらけ」として激しく非難している．

反対が続出

亦，長年に亙り国語審議委員会の措置に対し反対してきている文人および知識人からも「国語審議委員会が無限ループに陥り，その失敗は誰が観ても明らかなのに，再び他の分野に同じ過ちを繰り返そうとしている愚挙，直ちに止めるべき」とした批判が出ている．

シグマにさきがけて
SEA-InterNet運用開始

かねてから計画中だったＳＥＡのコンピュータ・ネットワークが，１９８６年４月１日運用を開始した．あまりの突然さに，関係筋とくにシグマ・システム開発本部では，困惑を隠し切れないでいるもよう．第１号の電子メイルとしては，モスクワのＫＧＢＶＡＸから，ＡＲＰＡネット経由で，祝電が寄せられた．

事務局からのお知らせ

会員の住所管理をコンピュータで行っていますが，最近郵便物が，一部宛先人不明で戻ってきます．
住所変更は早めに事務所へお知らせ下さい．
なお，連絡には会員番号をお忘れなく．

ソフトウェア・シンポジウム’８６

プログラム（予定）

6／4（9：00～19：30）

◇招待講演◇

大野豊＜京都大学＞

◇一般論文発表A（プログラミング）◇

・Ａｄａ言語によるソフトウェア開発経験
積谷寛俊，川田直哉＜構造計画研究所＞

・ロジック・プログラミングによる設計支援システムの開発
西山聡，牧野京子＜三菱総合研究所＞

・多用な作業形態に対応できるソフトウェア開発支援システムの試行
伊勢進寿，長山清，末吉一夫
＜日立ソフトウェアエンジニアリング＞

◇一般論文発表B（ソフトウェアの管理）◇

・基本モデルの構築
加藤政男，高畑一郎
＜ソフトウェア・リサーチ・アソシエイツ＞

・大型プロジェクトにおける開発作業の標準化
松井悦男＜富士通エフ・アイ・ピー＞

・ソフトウェア部品の再利用とその体系的な管理方法
大木幹雄＜日本電子計算＞

◇パネルディスカッションA◇

・ソフトウェア技術者の教育
コーディネータ　落水浩一郎＜静岡大学＞
河村一樹＜日本電子専門学校＞
杉田義明
＜ソフトウェア・リサーチ・アソシエイツ＞
竹田昌弘
＜日本ディジタルイクイップメント＞

◇チュートリアルA◇

・人工知能
小林重信＜東京工業大学＞

◇情報交換パーティー◇

6／5（9：00～17：00）

◇パネルディスカッションB◇

・仕様化技術
コーディネータ　岸田孝一
＜ソフトウェア・リサーチ・アソシエイツ＞
野辺良一＜協同システム開発＞
ほか

◇チュートリアルB◇

・シグマプロジェクト
シグマシステム開発本部＜情報処理振興事業協会＞

◇一般論文発表C（環境）◇

・ＳＰＥＸ：インフォーマル仕様と文書化
岩田成康，歌代和正，筏井幸夫
＜ソフトウェア・リサーチ・アソシエイツ＞

・ＩＷＢ(Integrated Work Bench)
大野仁勝＜野村コンピュータシステム＞

・パーソナル・ユニファイド・プログラミング環境
上林憲行＜富士ゼロックス＞

・ソフトウェア保守支援環境の今後の方向性
道正一郎，田中慎一郎＜協同システム開発＞

◇一般論文発表D（品質と信頼性）◇

・ソフトウェア信頼性モデルの実践的評価に関する調査研究
長井剛一郎＜システム工学＞

・潜在エラー数に与える再利用度の影響
小室豊，中山勝之，富田博章＜東芝エンジニアリング＞

・プログラム設計の要因分析
高田佳彦＜日本電子計算＞

・ソフトウェアの品質管理と品質保証
渡辺益秀＜日本電気ソフトウェア＞

◇パネルディスカッションC◇

・ナショナルプロジェクトの成果の技術移転
コーディネータ　斉藤信男＜慶応大学＞
川合英俊＜情報処理振興事業協会＞
花田収悦＜日本電信電話＞
棟上昭男＜電子技術総合研究所＞
ほか

ツール展示

6／4（12：00～18：00）　6／5（9：00～15：00）

◇ＨＣＰチャートプロセッサ	＜ユニオンシンク＞
◇パソコン運用管理システム	
◇ＣＯＢＯＬシンボリック・エバリュエータ	＜ソフトウェア・リサーチ・アソシエイツ＞
◇ＩＷＢ（Ｉｎｔｅｇｒａｔｅｄ　Ｗｏｒｋ　Ｂｅｎｃｈ）	＜野村コンピュータシステム＞
◇ＳＰＥＥＤＩＩ－ソフトウェア生産技術とツールの活用	＜三井情報開発＞
◇Ｃ－ＢＡＳ　Ｉ	＜三井情報開発＞
◇ＵＮＩＸにおけるツール合成によるアプリケーション生成システム	＜日本電子計算＞
◇ＦＡＳＥ－ＦＯＲＴＲＡＮプログラム開発保守支援ツール	＜三菱総合研究所＞
◇保守用テストデータ生成支援ツール	＜日本タイムシェア＞
◇ウォークスルー支援ツール	＜日本電気ソフトウェア＞
ほか	

ソフトウェア・シンポジウム’８６

主催：　（社）情報サービス産業協会

会期：　１９８６年6月4日（水）・5日（木）

会場：　東京農林年金会館（東京・虎ノ門）

◇シンポジウム’８６実行委員会◇

実行委員長：　村山公士<ティーディーシー>

プログラム委員長：　岡田正志<日本電気ソフトウェア>

委員：　浅川新一<東京システム技研>
遠藤靖彦<コンピュータ・アプリケーションズ>
河崎　浩<フジミック>
木間教大<日本ソフトウェア開発>

委員：　佐原　伸<野村コンピュータ・システム>
新美　論<協同システム開発>
鈴木　弘<構造計画研究所>
高田佳彦<日本電子計算>
玉井哲雄<三菱総合研究所>
林　　香<ソフトウェア・リサーチ・アソシエイツ>
別役高明<ジェーエムエーシステムズ>
吉村鉄太郎<管理工学研究所>

参加申込

下記に必要事項を記入の上、事務局までお送り下さい。到着しだい参加券と請求書をお送りします。

参加費（1名）　２２，０００円（正会員会社、賛助会員会社）
２５，０００円（会員外）

（5月２５日以降は、当日会場にてお申し込み下さい）

ソフトウェア・シンポジウム’８６申込用紙

申込者（氏名、年齢）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

所属（会社名）　＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

（部署）　＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

（会社住所）　〒＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

（ＴＥＬ）　＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

連絡先（請求書送付先）〒＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿
（申込者と異なる場合のみ記入）

参加費（○でかこむ）
２２，０００円　　２５，０００円

（２名以上の場合はコピーして御記入下さい）

事務局：〒１０５　東京都港区芝公園３－５－８　機械振興会館内
（社）情報サービス産業協会
ソフトウェア・シンポジウム実行委員会

ＴＥＬ：（０３）４３６－３９３８／ＦＡＸ：（０３）４３６－０３０８

ＳＥＡこれからの活動予定

－１９８６－

◆４月(April)

２３日（水）
再利用分科会研究会（機械振興会館）

２４日（木）
ＳＥＡ Ｆｏｒｕｍ Ａｐｒｉｌ’８６
（機械振興会館）

２８日（月）
横浜支部発足会（横浜：ホテル・ニューグランド）

◆５月(May)

８日（木）
第５回幹事会（機械振興会館）

９日（金）
管理分科会（協同システム）

１４日（水）
環境分科会第４回月例会（機械振興会館）

１４日（水）
再利用分科会研究会（機械振興会館）

１５日（木）
教育分科会研究会（機械振興会館）

１７日（土）
ＳＥＡ総会（機械振興会館）

１９日（月）
ＳＥＡ設立記念フォーラム（農林年金会館）
「ソフトウェア技術者に期待する」

３１日（土）
関西支部第４回研究会

５月末
ＡＩフォーラム（未定）

◆６月(June)

１３日（金）
管理分科会（未定）

１８日（水）
環境分科会第５回月例会（機械振興会館）

６月末
ツールの利用状況に関するフォーラム（未定）

◆７月(July)

１６日（水）
環境分科会第６回月例会（機械振興会館）

◆８月末
若手プログラマのワークショップ（岩手県盛岡）

◆９月末
秋のセミナー・ウィーク

◆１０月末
フォーラム　ｉｎ　関西（神戸）

ＳＥＡ会員状況

昭和５６年４月１８日現在の入会会員状況は以下の通りです.

正会員－－３２３名
賛助会員－－５社

正会員の年齢分布

２０－２４歳	１７
２５－２９歳	６３
３０－３４歳	１０６
３５－３９歳	７８
４０－４４歳	３１
４５－４９歳	１６
５０－５４歳	４
５５－５９歳	６
６０歳以上	２

ソフトウェア技術者協会入会申込書（正会員）

（フリガナ）
氏　名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

勤務先名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

勤務先住所：〒(　　)＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

勤務先TEL：＿＿＿＿＿＿＿＿＿＿＿＿

自宅住所：〒(　　)＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

自宅TEL：＿＿＿＿＿＿＿＿＿＿＿＿

連絡先（どちらかにチェックしてください）　　□勤務先 □自宅

年令＿＿才　　性別（男・女）　　血液型（A・O・B・AB）

会費：　入会金３千円　＋　年会費（入会より１ヶ年分）７千円　＝　１万円

ソフトウェア技術者協会入会申込書（賛助会員）

会社名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿
（フリガナ）
代表者：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

住所：〒(　　) ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

電話：＿＿＿＿＿＿＿＿＿＿＿＿

連絡担当者：
（フリガナ）
氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

所属：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

賛助会費：＿＿＿＿＿口（1口　5万円）　　ただし入会より１ヶ年分

＜申込書送付先＞：〒166　東京都杉並区高円寺南1-5-4 高円寺サンハイツ404
ソフトウェア技術者協会

＜会費振込先＞：三菱銀行本店公務部
普通預金口座 No. 0004830
口座名：ソフトウェア技術者協会

ソフトウェア技術者協会
〒166 東京都杉並区高円寺南1-5-4 高円寺サンハイツ404
TEL. 03-312-3256