SEAMAIL

Monthly Newsletter from
Software Engineers Association

Volume 2, Number 4 April 1987

目　次

編集部から		1
長岡ワークショップ報告・その1		2
Session 1: プロセス・プログラミング		2
ソフトウェア・プロセス・モデルの変遷	岸田孝一	2
討論記録	野村敏次	5
Session 2: 技術移転		10
環境の構築・利用と技術移転	杉田義明	10
討論記録	盛田政敏	12
春のセミナー・ウィーク報告・その1		18
要求定義支援ツール	松本吉弘／渡辺雄一	18
設計方法論の比較評価	落水浩一郎／村井進	22
ＯＡを取り巻く運営管理の諸問題	芝原雄二	26
長野支部設立報告	佐藤千明	28
会員の声		30
＜夏のプログラミング・ワークショップ＞へのお誘い	野村行憲	34
会員状況		37

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

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

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

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

代表幹事：　鈴木弘

常任幹事：　岸田孝一　長井剛一郎　盛田政敏　吉村鉄太郎

幹事：　稲田博　臼井義美　大木幹雄　岡本吉晴　落水浩一郎　柿下尚武　木村高志　久保宏志　熊谷章　斎藤信男　佐藤千明　芝原雄二　杉田義明　鳥居宏次　中園順三　西尾出　能登末之　針谷明　藤野晃延　松本崇純　松原友夫　丸尾浩一　水谷時雄　三浦信之　村井進

会計監事：　辻淳二　吉村成弘

常任委員長：　岸田孝一（会誌編集）　盛田政敏（企画総務）　吉村鉄太郎（技術研究）　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)：歌代和正　北村昌人　田中慎一郎

管理分科会(SIGMAN)：相沢圭一　芝原雄二　野々下幸治

教育分科会(SIGEDU)：大浦洋一　杉田義明　中園順三

再利用分科会(SIGREUSE)：青島茂　阿倍正平　村井進

ＡＩ分科会(SIGAI)：安倍昭敬　梅林信之　広川昭八　野辺良一　藤野晃延　横山憲一

ネットワーク分科会(SIGNET)：青島茂　鈴木弘　野中哲

法的保護分科会(SIGSPL)：能登末之

ＣＡＩ分科会(SIGCAI)：大木幹雄　寺嶋裕一　中谷多哉子　中西昌武

ドキュメント分科会(SIGDOC)：田中慎一郎　丸尾浩一

ＣＡＤ分科会(SIGCAD)：柿下尚武

支部世話人　関西支部：臼井義美　盛田政敏

横浜支部：熊谷章　林香　藤野晃延　松下和隆

長野支部：青柳和男　佐藤千明

名古屋支部：岩田康　鈴木智　西村享

ＳＥＡＭＡＩＬ編集グループ：大槻亮人　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　渡邊雄一

ＳＥＡＭＡＩＬ　Ｖｏｌ．２，　Ｎｏ．４　　昭和６２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

例によっていいわけ

お待たせしました．３ヶ月おくれの船便です．今度はどんないいわけをしようかと考えていたら，まわりから声あり：

「ヤクルトのホーナーがホームランを打ちすぎて，阪神優勝の望みが完全に消え失せ，編集長がやる気をなくした，というのはどうかしら？」

ちなみに，事務局のＮさんと関西支部屈指の暴走族Ｕさんは，熱烈なヤクルト・ファンです．

Good News そして，お願い

６月中旬，ついに会員数が待望の１０００人を突破しました！（７月９日現在１０３５人）．次の目標は２０００人です．実は，ＳＥＡの運営予算は，２０００人規模をベースに組んであるので，そのラインに到達するまでは，赤字をイベント収入で埋めなければならないのです．

そういえば，昨年度の赤字を補うべく，急拠企画された七夕（マネジメント）セミナーは，みなさんの御協力のおかげで大成功に終わりました．

次のメジャー・イベントは，９月のセミナー・ウィークです．気紛れなセミナー委員長が，また少し，新しい試み（パネルとチュートリアルの２本立て）を取り入れました．同封のチラシを御覧の上，７月末までに予約申込をぜひ（２割引きです！）．

もうひとつ Good News

今月中に，事務局が杉並の田舎（Ｎさんの表現）から，都心（千代田区隼町）に移転します．いまのところ，７月２４日の月例フォーラムが終わった後の週末に引っ越し，遅くとも８月１日からは完全に新事務所で業務を始めます．

新住所：〒102 東京都千代田区隼町 2 - 12
藤和半蔵門コープビル 505号

電　話：03 - 234 - 9455　（FAXの番号は未定）

いままでより，ずっと便利になります．お近くへお越しの節は，お立ち寄りください．

長岡レポート

２月の環境ワークショップについては，すでに報告書が出ていますが，多くの会員の方は読んでおられないと思いますので，各セッションから，チェアマンのまとめと討論記録を抜き出し，編集部で整理することにしました．

もともと，この号はそれだけを特集するはずだったのですが，懲り屋の編集長のアンカーとしての筆が進まず，全体で５セッションのうち，第１部として，２セッションだけを収録しました（残りは次号）．

春のセミナー

恒例のセミナー・ウィークのレポートですが，こちらのほうは，担当した各レポータからの原稿がなかなか出てきません．とりあえず，今回は２セッション分だけです．次号以降に御期待ください．

投　稿

会員からの投稿は２篇．１つは，管理分科会の世話人・芝原さんからの報告です．他の分科会の方々も，なるべくマメに報告をまとめるよう，この場を借りてお願いしておきます．もう１つは，長野の佐藤さんからの支部設立報告です．地方支部としては，続いて，名古屋でキックオフ・ミーティングが行われました．

会員の声

いくつかのアンケートや，入会申込書に書かれた会員からのメッセージをまとめてみました（女性技術者問題についての第２回アンケート，環境アンケートの結果は，あらためて別の号にのせます）．

若手の会

ポジション・ペーパ募集の最終案内です．よろしく．

次号予告

次号は，８月中旬（？）発行の予定．いままでの遅れを取り戻すべく，５～６月合併号とし，ページ数を倍増して，長岡レポート，ＩＣＳＥ参加記，セミナー記録，会員からの投稿と，盛り沢山の内容を考えています．これで，手持ちの原稿在庫のかなりの分を消費してしまいますので，その補充になにとぞ御協力ください．

第２回　実践的開発環境に関する集中討論 in 長岡

Session 1:　プロセス・プログラミング

ソフトウェア・プロセス・モデルの変遷

チェアマン
岸 田 孝 一

１．はじめに

このセッションでは，米国コロラド大学のＬ．オスターワイル教授が昨年提唱したプロセス・プログラミングの概念［１，２］をテーマとして取り上げ，それが何を意味しているのか，またわれわれの環境改善にとってどんな効果をもたらすのなどについて，討論することを意図した．

プロセス・プログラミングの根本思想は，「ソフトウェア開発プロセスそれ自身が１つのソフトウェアであり，プログラム可能である」という一言につきる．ある意味でそれは，これまでに提唱されてきた一群のソフトウェア・プロセス・モデルを統合化し，ソフトウェア・エンジニアリングの方法論に，統一的かつ具象的なフレームワークを与えようとする野心的な試みだとみなすことができよう．

２．なぜプロセス・モデルか？

ソフトウェア開発プロセスを形式的にモデル化することの重要性が初めて認識されたのは，１０７０年代初頭のことであった．そうしたモデル化の動機づけは，主として，米国における大規模システム開発において続発したプロジェクト・トラブルを回避するためであり，こうしてまず，Ｗ．ロイス（ロッキード）やＢ．ベーム（ＴＲＷ）によって古典的なウォータフォール・モデル［３］が提唱され，一般に普及した．

しかし，このモデルの現実との不適合性は，やがてメインテナンス問題の顕在化とともに明らかになり，以後，それに代わるものとして，Ｌ．ベラディおよびＭ．レーマンによるソフトウェア進化論のモデル［４］や，Ｂ．ベームのスパイラル・モデル［５］などの代案が提唱されたが，決定的な解はまだ見つかっていない．

一方，７０年代終わりから８０年代前半にかけては，ソフトウェア・プロセス全体の効率化を支援するためのツールを統合化した「環境」の概念が多くの人々の関心を集めるようになり，環境を構築するさいのアーキテクチャ・デザインの土台としてのプロセス・モデルについて，新しい角度からの議論が多角的に展開されるようになった．

１９８４年春以降，約１年半の周期でこれまでに３回開かれた「ソフトウェア・プロセスに関する国際ワークショップ」（略称：ＩＳＰＷ）は，そうした議論の重要な舞台であり，進化論モデルの新しい形であるＰＷモデル（第１回）や契約モデル（第３回），あるいはスパイラルモデル（第２回）など，いくつかの重要なモデルがその席上で提案されてきた．

プロセス・プログラミングの概念は，昨年１１月の第３回ＩＳＰＷで発表され，多くの論議を呼んだきわめてホットなトピックである［６，７，８］．

３．プロセス・プログラミングの背景

「プロセス自身がソフトウェアである」というオスターワイル教授のテーゼは，一見ごくあたりまえのように聞こえる．

たとえば，「多くの組織で作られている標準開発手順書は，１つのプロセス・プログラムではないのか？」といった素朴な質問が，ＩＳＰＷでも今回のワークショップでも，ちらほら出たりした．

たしかに，そうした古典的マニュアルの類も，きわめて原始的な形のプロセス・プログラムの萌芽であるにはちがいないのだが，プロセス・プログラミングのほんとうのねらいは，コンカレントなマルチ・プログラムとしての開発プロセスを，自然語ではなく，厳密でフォーマルな言語を用いて記述し，それについていろいろな分析をおこなったり，できればその支援環境を半自動的に生成しよう，というあたりにある．

その背景には，オスターワイル教授自身が８０年代前

半の５年間を費やして行ったＴＯＯＬＰＡＣＫプロジェクトでの環境アーキテクチャ・デザインの経験が，しっかりと踏まえられている．

ＴＯＯＬＰＡＣＫの技術的特徴は，環境を構成する要素ツールや，それらが扱うさまざまなファイルの取り扱いを，オブジェクト指向の概念で統合化しようとしたことだと思われるが，プロセス・プログラミングでは，その考え方をさらに１歩進めて，ソフトウェア・プロセスを１つのオブジェクトとしてとらえようとしているわけである．

環境の設計とは，つまるところ，それが支援するプロセスの設計にほかならないのであり，その意味で，オブジェクト指向アプローチの導入によって，プロセスを目に見えるよう形式化することがきわめて重要であることは，いまさら論を待たない．

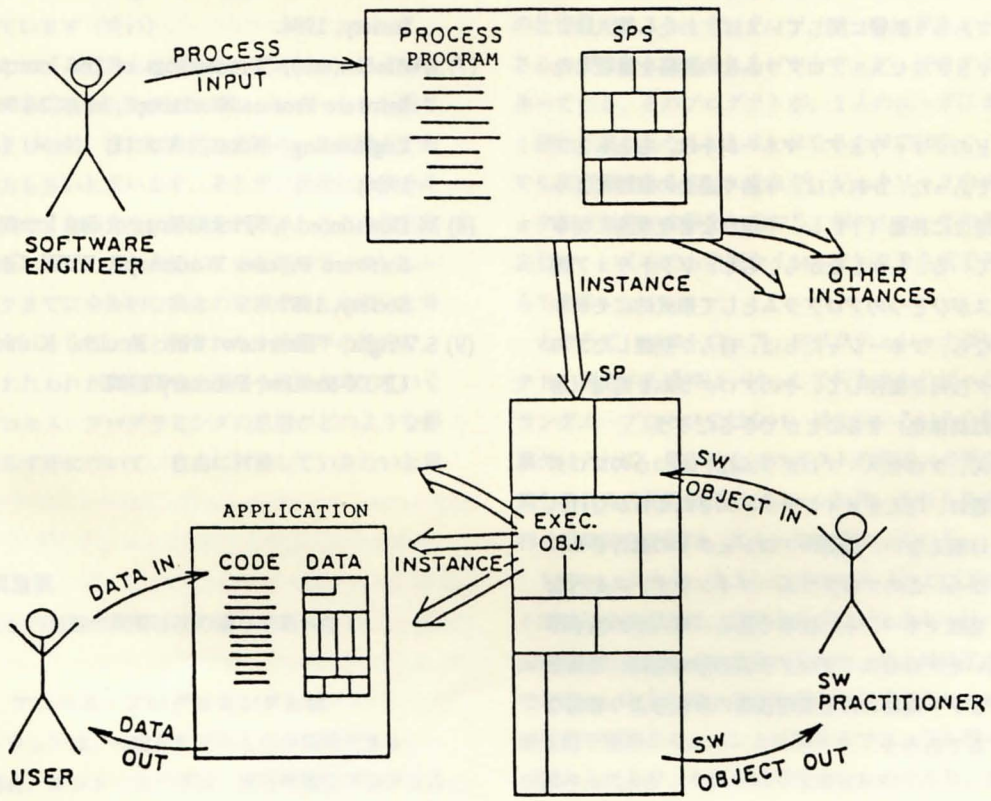
ただし，そこには，プロセスの設計者（ソフトウェア・エンジニア）と実行者（ソフトウェア開発者）のあいだの一種感情的な対立関係（あるいは反発関係）を不必要に増幅してしまう危険が，常につきまとっているように思われる．そのことは，今回のワークショップの議論でも，ときどき感じられた（下図参照）．

４．おわりに

今回のセッションでは，斎藤先生のキーノートで，この新しい概念について，チュートリアルを兼ねた紹介をしていただき，参加者の方々の新鮮な反応を得て，できれば日本的なプロセス・プログラミングの可能性を探るところまでいけたらよいと考えていた．このねらいは，ほぼ６０％程度は達成することができたと思っている．

最後に，プロセス・プログラミングについての個人的感想を２点述べておきたい．

わたし自身，時間的にほぼＴＯＯＬＰＡＣＫと並行してすすめられたＪＳＤのＳＭＥＦプロジェクトを技術的に管理する立場にあった．ＳＭＥＦは，わが国ソフト業



界にとって最初のＵＮＩＸプロジェクトであり，その社会的な成功が今のシグマ計画を生みだしたことは，よく知られている．しかし，にもかかわらず，ＳＭＥＦの技術的な遺産が，シグマにはほとんど引き継がれていないことも事実である．

それには，もちろん，さまざまな政治的理由がある［９］．しかし，また，一方において，われわれＳＭＥＦの関係者が，プロジェクトの技術的な成果を，個々のツールや環境プロダクトとは別に，もう一段抽象的に昇華したかたちでまとめることができなかったことも，大きな問題だと反省せざるをえない．

ＴＯＯＬＰＡＣＫが研究開発プロジェクトだったのに対して，ＳＭＥＦは，ＵＮＩＸをわが業界に技術移転するというハンディキャップを負っていたのだからというのは，単なる慰めのための言い訳にしかならない．

われわれプログラマにとって，すべてのプロジェクトはひとつの例題にすぎず，われわれは，その例題の解決を通じて，もっと別の抽象化されたシステムの構築の実現をめざしているのだという命題は，国家プロジェクトや協同開発の場合にも，やはり真実なのである．

プロセス・プログラミングが今後われわれ自身の仕事に与えるであろう影響に関していえば，わたし個人は，マネージャとプロセス・プログラムとの関係を重視したい．

ほとんどのソフトウェア・マネージャは，もともとプログラマであった．かれらは，年齢や過去の業績によって現在の地位に昇進（？）し，不慣れな管理業務に従事させられている．しかしながら，もし，ソフトウェア開発プロセスがひとつのプログラムとして形式的にモデル化できるなら，マネージャたちは，自らが熟練したプログラミング技術を駆使して，そのプログラムを処理（解釈あるいは具体化）することができるだろう．

もちろん，プロセス・プログラムは，ふつうのプログラムと同様に，たえずメインテナンスされて行かなければ実用には耐えない．実際のプロジェクトの進行をモニターしながら，このプログラム・メインテナンスを行なうのは，当然マネージャの仕事である．何らかの高水準言語を用いたプロセス・プログラムの形式化は，相異なるプロジェクト間における管理技術の移転をより容易なものにするであろう．

５．参考文献

(1) L.J.Osterweil, "Software Process Interpretation and Software Environments", Technical Report CU-CS-324-86, University of Colorado, April 1986 (ソフトウェア・シンポジウム８７論文集に収録).

(2) L.J.Osterweil, "Software Processes Are Software Too", Proc. of 9th ICSE, March 1987.

(3) W.W.Royce, "Managing the Development of Large Software Systems: Concepts and Techniques", PRoc. of IEEE WESCON, August 1970 (Proc. of 9th ICSE に再録).

(4) L.A.Belady & M.M.Lehman, "Program Evolution - Processes of Software Change", Academic Press, 1985.

(5) B.W.Boehm, "A Spiral Model of Software development and Enhancement", in ACM Software Engineering Notes, Vol.11, No.4 (August 1986).

(6) M.Lehman(ed.), "Proceedings of 1st International Software Process Workshop", IEEE Computer Society, 1984.

(7) J.Wileden(ed.), "Proceedings of 2nd International Software Process Workshop", in ACM Software Engineering Notes, Vol.11, No.4 (August 1986).

(8) M.Dowson(ed.), "Proceedings of 3rd International Software Process Workshop", IEEE Computer Society, 1987.

(9) S.Wright, "Interview with Kouichi Kishida", in UNIX Review, February 1987.

第２回　実践的開発環境に関する集中討論

Session 1:　プロセス・プログラミング

討　論　記　録

レポータ
野　村　敏　次

１．はじめに

まずは，司会の岸田さんの挨拶から：

このセッションの狙いは，６月のソフトウェア・シンポジウムに招待講演者として来日が予定されているオスターワイルさんが，昨年の１１月にロッキー山中で行われたワークショップで発表し，多くの議論を呼んだプロセス・プログラミングの概念について，少し詰めてみようということです．そのワークショップには，私と鳥居先生，斎藤先生の３人が出席しましたが，英語での討論になかなか参加できなくてイライラしました．今日は日本語で話せるので，楽な気持ちで議論できるのではないかと考えています（笑い）．

あらかじめ，みなさんの御手もとにオスターワイルさんが去年の春に書かれたテクニカル・レポートをお配りしておきましたが，なにせ英語ですから，まだ読んでおられない方も多いと思います．そこで，最初に３０分くらいの時間で，斎藤先生に解説をお願いし，その後いろいろ質問をフロアの方から出していただいて，コーヒー・ブレークまでに全体的な概念の定義をまとめたいと考えています．そして，あと残された１時間半の枠内で，これからわれわれが開発環境の構築や改善を考えていくとき，プロセス・プログラミングの思想がどのような影響をもたらすかについて，自由に討論していきたいと思います．

２．基調講演

斎藤先生のお話の概要は次の通りであった：

２．１　プロセス・プログラミングとは

ソフトウェアは，一度作ればみんなが利用できる．

この場合，エンド・ユーザは，実行可能なプログラムの１つのインスタンスとのあいだで，いろいろなデータをやりとりして，自分が目的とする問題を解決しているのである．このエンド・ユーザが使うプログラム（正確には，そのインスタンス）は，データの処理手順をあらわすコード部分と，データをいれておくためのストレージという，２つの部分から構成されている．

プログラム（より正確にいえば，個々のエンド・ユーザが使うインスタンスを生成するもとになるジェネリックなプログラム）は，プログラマ（オスタワイルの定義によれば「ソフトウェア・プラクティショナ」）の手によって作られる．

かれは，プログラム（すなわちエンド・ユーザが直接利用する実行可能コード）だけを作るのではなく，各種の仕様書やテスト・データ，ソース・コードなど，たくさんの要素から構成されるソフトウェア・プロダクトを作っている．そのプロダクトが，１人のユーザによって１回だけ使われたらおしまいになるわけではなく，何度でも再利用できるのは，それが，ジェネリックなオブジェクト（対象）の形をしており，個々のユーザの希望に応じて，いくらでも必要なインスタンスを生成できるからである．

ところで，ソフトウェア・プラクティショナがプロダクトを開発する過程は，けっして行き当たりばったりのランダム・プロセスではない．過去のいろいろな経験に裏付けられて，標準的なプロダクトの構造（必要な各要素とそれらの相互関係のパターン）や，それらを作って行く具体的な手順は，あらかじめ決まっている．

しかし，ふつう，そうしたプロセス（およびプロダクト構造）のモデルは，経験あるプロジェクト・リーダの脳細胞のなかにたくわえられており，他人にはアクセスできない．ときには，それを他人に伝達するために，標準工程や標準ドキュメントに関するマニュアル類の作成が試みられるが，それらは不完全なものであり，また，自然語で書いてあるために，多くのあいまいさを含んで

いる．

プロセス・プログラミングが意図しているのは，このソフトウェア開発プロセス自体を１つのジェネリックなオブジェクト（すなわちソフトウェア）として客体化し，何らかの言語を用いて厳密かつ形式的に記述することによって，その再利用を容易にしようということである．

いいかえれば，それは，われわれがこれまで使ってきたプログラミングという方法論を，エンド・ユーザの問題解決プロセスにではなく，われわれ自身のソフトウェア開発プロセスに適用しようということに他ならない．

オスターワイルによれば，それこそが「ソフトウェア・エンジニア」のなすべき仕事であり，もしそれがうまくいけば，「プラクティショナ」は，汎用的なプロセス・オブジェクトの特定のインスタンスを操作するかたちで，いままでより効率的に開発作業を進めることができるようになる．

２．２　その可能性と言語

ふつうのプログラムのステートメントは，すべてコンピュータによって実行されるが，プロセス・プログラムの場合には，あるステートメントは機械（ツール）によって，また，別のステートメントは人間によって実行される．そして，ときには，こうした実行媒体のバインディングがダイナミックに行われる．多くの場合，開発プロセスを構成するサブプロセスは同時並行的に実行される．もちろん，実行媒体（人間および機械）は複数存在し，プロセスの実行は分散型で行われる．

プロセスのなかで扱われるオブジェクトには，いろいろな種類がある．コードやデータ，ドキュメントなどは当然そうである．ツールもオブジェクトの一種だと考えられる．ところで，ツールが実行すべき機能は，何らかのサブプロセスをあらわしている．サブプロセスと実行媒体とのバインディングや，ときにはサブプロセスの生成が，プロセスの実行中に動的に行われるとすれば，プログラム中にそのことを記述する必要がある．すなわち，プロセスもまた，１つのオブジェクトである．

したがって，プロセス・プログラミング言語は，こうした多種類のオブジェクトのタイプを区別して取り扱うことができなければならない．

現存するプログラミング言語は，どれもこのような機能を十分に備えてはいない．オスターワイルは，さしあたり，ＡＤＡとＵＮＩＸシェルの組み合わせで，プロセス・プログラムのプロトタイプを書こうと考えているようだが，それは，あくまで実験であり，その目標は，プロセス・プログラミング言語の仕様を洗い出すことにある．もちろん，希望はある．プロセス・プログラミングは，このところやや停滞気味だった言語研究の分野にとっても，きわめてよい刺激になるであろう．

２．３　プロセス・プログラミングの意味

さて，環境開発というわれわれの関心に照らしていえば，プロセス・プログラミングは，次のような意味を持っている．

まず，第１に，それは，環境のアーキテクチャ設計に対して，プロセス指向という統一的な枠組みを与える．これまでの環境は，ＵＮＩＸでもそうだが，ツールの寄せ集めという感じが強かった．プロセス・プログラムを解釈し実行するのが環境の使命であるという視点は，そうした状況下で，環境設計者に正しい方向を指し示すものである．

第２に，プロセス・オブジェクトをインスタンシエイトすることで実際の開発プロセスを進めて行くという考え方は，ツールのカスタマイゼーション機能を論ずる場合に，きわめて有用である．

第３に，これからの環境は当然のことながら，分散型になって行くだろうが，プロセス・プログラムの各サブプロセスを解釈・実行するのは，そうした分散環境におけるサーバ・ノードの役割だと考えられる．環境のなかで，どこにどんなサーバを配置するかは，きわめて重要な設計上の問題であるが，その答えは，その環境で実行すべきプロセス・プログラムがどんなものであるか，によって決まる．

２．４　われわれは何をなすべきか

アメリカでは，すでにＡＲＣＡＤＩＡと呼ばれる共同研究プロジェクトがスタートし，プロセス・プログラミングの概念を実用化する方向で，いくつかの大学や企業が実験を始めようとしている．これまで，ソフトウェア技術の日米格差について，いろいろいわれてきたが，このままほうっておけば，ますます差が開くことが予想される．

今回のワークショップでの議論をきっかけに，われわれも力を合わせて，何とか独自性を持った「日本的」プロセス・プログラムを作りあげて行きたいと考える．

3．討論

以下，ほぼ時間的な経過を追いながら，今回のワークショップで，どんな問題がとりあげられ，どのような意見がたたかわされたかを中心に，討論の内容を要約して報告する．

3．1　最初の素朴な質問

ワークショップ参加者のほとんどにとって，プロセス・プログラミングという「奇妙な」ことばを耳にするのは，初めてであった．そこで，当然のことながら，次のような素朴な質問が，次々と投げかけられた：

- すべてのプログラムには，バグがある．もし，プロセス・プログラムにバグがあったら，開発プロセスに取り返しのつかない被害を与えるだろう．いったい，そのデバッグは，どのようにして行われるのか？
- 現実のソフト開発では，特にトラブル発生時に，プロジェクトをどうやって終わらせるかが問題になる．プロセス・プログラムの終了条件はどうなるのだろう．無限ループすることはないのだろうか？
- プロセス・プログラムという形での開発工程のモデリングは，論理的には可能かもしれない．しかし，そのインプリメンテーション，つまり現実のプロジェクトにおけるプロセス・プログラムの解釈・実行メカニズムの実現は，きわめてむずかしいのではないか？

これらに対して，斎藤先生および岸田さんからのコメントをまとめると，おおよそ次のような趣旨になる：

オスターワイルは，環境問題を手がける以前はテスト・ツールの専門家であり，彼が学界で名前を上げたのは，ＤＡＶＥと呼ばれるＦＯＲＴＲＡＮのデータ・フロー・アナライザの開発によってである．したがって，プロセス・プログラミングについても，テスティングとのアナロジーでものを考えているように思われる．

ソフトウェア・エンジニアがプロセス・プログラムを書くとき，何もないところから仕事を始めるわけではなく，何らかのリクワイアメントをもとに，過去における開発経験を集約する形で，プロセスのアルゴリズムやプロダクトの構造を設計して行く．

こうして作られたプロセス・オブジェクトが，個々のプロジェクトに対してインスタンシエイトされるわけだが，その前に，当然ある種のスタティックなチェックが行われるだろう．プロセス・オブジェクトが厳密な言語を用いて記述してあれば，そうしたチェックは容易になる．

正当性の証明ができるかどうかは，プロセス・プログラミング以前の，ソフトウェア開発に対するわれわれのリクワイヤメントが，どれだけきちんとまとめられるかどうかにかかっている．

ソフトウェア・プラクティショナが，実際にプロジェクトを進めて行くさいには，ダイナミックなモニタリングが行なわれる．これまで，定量的なプロジェクト管理ということがいわれてきたが，どこで何をどう定量化すべかが，はっきりしていなかった．プロセス・プログラミングが，もし，完全な形で書けるなら，その実行プロセスのモニタリングこそが，一番正確な定量化指標になるにちがいない．

プロセス・プログラムが無限ループを含む危険性については，何ともいえない．そこには，おそらくリカーシブ・コールが含まれるだろうし，あるステートメントは機械でなく，人間によって実行されるわけだから，従来のプログラムとは話が少し異なる．

プロセスを「プログラムする」という表現は，どちらかといえば，誤解を生みやすい．人間とハードとソフトと一体になった仮想機械がプロセス・プログラムを実行するといわれると，何か，人間がロボットになったかのような感じを受ける．

しかし，「設計上の意志決定を下す」とか，「テストの妥当性を判断する」とかいったプロセス・ステップは，まさに人間がそのもっとも人間的な知性をもって行うべき仕事であり，プロセス・プログラム上では１種の入力ステートメントとして表現されるというのが，オスターワイルの考えである．

3．2　歴史的経緯

プロセス・プログラミングの概念が出てきた背景を環境の歴史的発展の中で位置づけるとどうなるかについて，昨年暮れのＰＳＤＥシンポジウムに参加した方々から，次のようなコメントがあった：

第１世代の環境といえば，まずＵＮＩＸがあげられる

が，そのアーキテクチャ上の特徴は「ツール指向」の一言で表現できよう．つまり，環境はツールの集まりであり，それらの機能をどのように分解し統合化するかという観点に立って，インタフェイスの統一（単純化）を図り，それなり（！）の成功をおさめた．しかし，ツールの機能的統合という面から見ると，ＵＮＩＸはあまりにアナーキーである．

そこで，８０年代になって，第２世代の環境構築があちこちで試みられた．オスターワイルたちのＴＯＯＬＰＡＣＫは，その代表的な例である．これらの第２世代の環境における主流のアーキテクチャは，「データ指向」であった．たとえば，ＴＯＯＬＰＡＣＫでは，ユーザ・コマンドの名前は，ツールの機能ではなく，ユーザが最終的に必要とするもの（オブジェクトまたはファイルの種類）になっている．環境が，そうしたユーザの要求を解釈して，必要なツールを自動的に起動してくれるのである．

プロセス・プログラミングの概念は，明らかにこの延長線上にあるが，環境アーキテクチャの考え方は「プロセス指向」に変わってきている．すなわち，ソース・コードやテスト仕様のようなデータと同様に，コンパイラやパーサ・ジェネレータも，環境を構成するオブジェクトであり，それらの論理的な相互関係をあらわすプロセス・モデルを中心に，環境を組み立てて行こうというわけである．

３．３　実体はあるのか？

ところで，このような新しい考え方は，見方によってはハッタリに見えるし，何となく眉唾的な印象を与えるものである．今回の参加者の多くも，半信半疑で討論に参加していたのではないかと思われる．こうした人々の意見を集約すると，おおよそ次のようになる：

プロセス・プログラミングの概念は，話題性を狙っただけで，実体はまだほとんど存在しないのではないか．ただ，ＡＩやオブジェクト指向に関連する重要なポイントは大部分押さられえており，その意味では非常におもしろい．うまく行けば，ソフトウェア工学がほんとうに学問として認められるきっかけになるだろう．

その意味で，オスターワイルの頭の中で，プロセス・プログラミング言語の仕様がどの程度固まっているのかが，興味深い．すでに，この１月からコロラド大学の学生を使って実験を行うということは，何等かのイメージがすでにあるのだろうが，テクニカル・レポートに出ている例題は，単純なものばかりであり，実際のプロセス記述が，こういったアルゴリズミックな言語だけでうまく行くのかどうかは，はなはだ疑問である．

３．４　プロセス・モデルとの関係

ソフトウェア開発環境を構築する際に，支援対象としてのプロセスはきわめて重要な基盤的要素である．プロセスを的確に表現できなければ，有効な環境構築は望めない．

この，プロセス・モデルとプロセス・プログラミングの関係については，次のような意見が出された：

開発プロセスのモデル化については，いままでにも，ウォーターフォールを初めとして，進化論，スパイラル，フラワーなど，いろいろ提案されているが，どのモデルも十分に現実を表現できずに行き詰まっているのが，現状であろう．そういう意味からすれば，どんなモデルでも自由に表現できるプロセス・プログラミングのアプローチは，かなり魅力的に見える．

オスターワイルの考えは，ＴＯＯＬＰＡＣＫにおけるオブジェクト指向型ユーザ・インタフェースを発展させて，プロセス・オブジェクトをうまく整理し，コントロールできれば，モデル化のフレキシビリティを高められるということであろう．

現状では，プロセス・モデルを一意的に定めるのはむずかしいので，プロセスの表現手段を確立することは，それに対する支援環境を作る上からも，きわめて有用だと思われる．

３．５　プロセスのメインテナンス

ジェネリックなプロセス・オブジェクトというアイデアは，一種のメタ・プログラムを書くということに通じると思われるが，それはだれかが１度だけ書いてしまえばおしまいになる．

ただし，最終プロダクトとしてのプログラムですら満足に作れないような技術の現状からいって，はたしてそうしたメタ・プログラムがうまく作れるのか，責任の所在を雲の上に棚上げして，何だかわけがわからなくしてしまうだけではないのか，といった意見が出された．

これに対して，メタ・プログラムを１度書けばおしま

いという考え方は，メインテナンスを無視しているという反論があった．ふつうのプログラムの場合と同様に，プロセス・プログラムにもメインテナンスは発生する．

テクニカル・レポートを読んだかぎりでは，オスターワイル自身，次の３種類のメンテナンスを区別して考えているようだ：

１つは，ユーザが使うソフトウェア・プロダクトそのもののメインテナンスであり，これはふつうの意味でのメインテナンスに他ならない．

第２は，ソフトウェア・プラクティショナが実際にプロダクトを開発するさいに，与えられたソフトウェア・プロダクトの標準構造（ＳＰＳ）が現実と一致しないという場合であり，３番目は，指定されたプロセス・アルゴリズムが実際の開発作業とくいちがった場合である．

逆にいえば，これら３種類のメンテナンスをしっかり行うためにも，プロセス・プログラミングの考え方は重要だということである．

３．６　プロセス・プログラミングの目的は？

プロセス・プログラムがなんらかの言語できちんと書けたとした場合，それは人間が読むばかりでなく，当然コンピュータで処理して何かをやらせることになると思われるが，いったい何が起こるのか．

その辺のところをオスターワイル自身がどのように考えているのか．テクニカル・レポートから類推すると，次のようにいえるだろう：

プロセス・プログラムを書くことの目的の第１は，プロセスに関する関係者間のコミュニケーションの改善である．第２は，プロセスの善し悪しに関する分析や計測ができること，そして第３に，ツールの利用形態や環境アーキテクチャについてのガイダンスが得られることである．

これは，オスターワイルの夢であろうが，プロセス・プログラムの記述から，環境を自動的に生成しようと考えているものと思われる．おそらくＧＡＮＤＡＬＦ等のパラメトリックな環境生成のコンセプトが頭にあるものと思われる．はたしてそこまで行けるのかどうかは，オペレーションの解釈・実行者としての人間について，どこまでの記述が可能かにかかっていよう．

開発環境のアーキテクチャ・デザインについては，まだよくわかっていないことが多いが，逆にプロセス・プログラムを実行するための環境を試作することを通じて，アーキテクチャがわかってくるということもあり得る．いずれにせよ，これからの環境構築を考える上で，なかなかおもしろい問題を含んでいる．

４．おわりに

以上にあげたことがら以外にも，

- プロセス・プログラムのインスタンシェーションとは何か？
- カストマイズをどのように行うのか？
- メイン・フレーマのソフトウェア・ファクトリにおける設計・製造の分離とプロセス・プログラムとの関連
- ＵＮＩＸのＭＡＫＥコマンドのプロセス・プログラム的解釈

など，多彩な話題に関して，かなり白熱した討論が展開され，３時間が非常に短く感じられたセッションであった．

新しい話題の割に，討論がかなり細かな部分にまで踏み込んだ形で盛り上がったのは，わが国でも，ソフトウェア開発のパラダイム変革やプロセス・モデルの改善に対する技術者の関心が高まっているからであろう．

報告者である私自身，ＪＳＤのＳＭＥＦプロジェクトの発足当初からＴＯＯＬＰＡＣＫとの情報交換を通じて，オスターワイル教授のこれまでの仕事のついては，ある程度知っていたのだが，プロセス・プログラミングという概念に接するのは初めてであり，いくらか戸惑いながらも，何か新鮮な感動を受けた．

今後も，こうしたワークショップのプログラムのなかに，その時点での世界の最先端のトピックスをとりあげて，積極的な討論を行う機会を持つことが望まれる．

第２回　実践的開発環境に関する集中討論 in 長岡

Session 2:　技術移転

環境の構築・利用と技術移転

チェアマン
杉　田　義　明

１．はじめに

１９７０年代における各種方法論の提唱を受けて，８０年代は，より具体的なソフトウェア・エンジニアリングの実践，つまり技術移転の時代になるだろうと期待されていた．

しかしながら，ワークステーションの急速な高性能化に代表されるハードウェア技術の進歩や，ＡＩ・知識工学を初めとする周辺技術の話題の豊富さに比較すると，実践的ソフトウェア技術の展開スピードはきわめて遅く，また地味なかたちであって，一般の目からは，技術そのものの進歩があたかも停滞しているかのように，見える．

わずかに，デマゴーキッシュなプログラマ６０万人不足説が一人歩きをはじめ，産業界・教育界にさまざまな思惑を抱かせながら，ソフトウェア産業の未来に対する社会的な関心をいくらか高めているだけにすぎない．

ソフトウェア開発の現場では，１０年前と比べて，どんな新しい技術が導入されて，作業工程がどのように改善され，品質や生産性がどれだけ向上してきたのだろうか．

現在のソフトウェア需要に大きな割合を占めている第３次金融オンライン・システムの開発現場における技術課題を，あるワークショップで取り上げたことがある．

ほとんど信じられないような人数の技術者を動員して実施されているそれらのプロジェクトにおける開発作業手順や支援環境を，過去の第１～２次プロジェクトと詳細に比較して得られた結論は，ユーザ要求が指数的な増加を見せている反面，ソフトウェア開発の環境は，コンピュータの利用形態がバッチからＴＳＳに代わっただけで，そこで使われているツールや工程モデルは，まったく１０年前と変わらないということであった．

また，通信業界の某大手ユーザでは，１０数年がかりで整備されたシステム開発の標準マニュアルがまるで六法全書のように部厚くなり，それが創造的な仕事のほとんどをガイドしてくれるという神話があって，上級ソフトウェア技術者になるためには，六法全書を活用できる弁護士なみのスキルが要求されるといわれている．

これらは，いささか極端で特殊な事例なのかもしれない．しかし，全体的に見て，ソフトウェア・エンジニアリング研究の最前線と，現実世界の開発現場におけるその応用や実践の状況とのあいだに，かなりのへだたりがあるというのが実態であろう．

２．環境における技術移転

一般に，技術の成熟には，＜研究・開発・移転＞という３つの段階がある．そしてまた，移転段階は，＜選択・導入・統合・広報＞という４つの段階に分かれる．教育訓練や各種マニュアルの整備は，技術移転プロセスの最後の部分である＜広報＞のなかにに位置づけられる．

統合的ソフトウェア開発支援環境を構築し利用するために必要な要素技術は，これまで数年にわたって研究・開発され，現在は明らかに移転段階に入っている．

技術移転とは，ある意味で「経験の経済学」だといってよい．今回のワークショップ参加者は，それぞれ，環境関連技術の教育や移転について，なんらかの経験があり，意見を持っているものと予想される．このセッションでは，それらの経験や意見を，おたがいに評価しあい，交換するためのフリー・マーケットがうまく構成できればと考えた．

一応，チェアマンとして用意した討論の話題は次のようなものであった：

- 対象とすべき技術：いま技術移転の対象として，どんな技術を取り上げるべきか．その成熟の度合いは，など．
- 技術移転プロセス：そのモデルはどうなるのか，プロセス・プログラミングは可能か．

－ 国家プロジェクトの役割：ＩＣＯＴ，ＪＳＤ，ＳＩＧＭＡなどを，技術移転という観点から見直す．
－ 大学・研究所に望むこと：産学の分業・協力体制はいまのままでよいのか．その日米比較など．
－ 人間的側面：技術移転のグループ・ダイナミクス．日本的社会構造は技術移転に敵しているか．
－ 技術促進の加速メカニズム：ショーケイス・プロジェクトや企業内遊園地などの概念の有用性について．

しかし，実際のセッション運営は，これらの話題にとらわれることなく，会場からの発言や問題提起を中心に，いわば成り行きまかせで行おうと考えていた．ソフトウェア・プロジェクト運営における成功のための第１原則「メンバー性善説」にしたがったわけである．

３．討論結果を振り返って

これまで，わたし自身，技術移転をテーマとするパネル討論を，チェアマンとして２回，スピーカとして２回ほど経験していたが，今回のワークショップが，最もエキサイティングであった．

３．１ 基調講演

まず，君島浩さん（富士通）に，企業内で教育を担当しておられる観点から，ソフトウェア技術一般の移転の問題について，日頃実行したり，考えたりしていることをお話いただいた．概要は次の通りであった：

１．背景
２．技術移転の構成要素
３．現在のソフトウェア企業の技術水準
４．先端技術と実践技術の格差
５．技術移転を阻害する要因
６．技術移転作法：７つの心得
７．技術移転の体験：ソフトウェア設計の自動化
８．技術の利用容易性：タスク指向の移転方法
９．技術移転の目標管理とボトムアップ運動
１０．まとめ

君島さんのポジション・ペーパの内容を，こうやって目次ふうに整理すると，何となく堅いレクチャーのように見えるが，実際は，大学教育の現状を批判したかと思えば，返す刀で企業内での技術管理の問題点を切り，採用問題とからめて現代学生気質を論ずるといった，大変楽しいお話であった．

３．２ 討論

その後の討論は，初めのうち，教育問題だけに焦点が合ってしまって，それだけで時間切れになってしまうのではないかと，いささか心配した．実は，昨年１０月のＵＮＩＸワークショップ（ｂｉｔ臨時増刊「最新ＵＮＩＸ」参照）でも，斎藤先生が司会された開発環境のセッションで，教育の問題をだれかが提起したところ，３時間の討論時間のほとんどがそれで終始してしまった苦い経験があるからである．

一般に，教育についての議論は，全員が参加できる反面，ほとんど皆にとってうなづけるステートメントばかりが続出して，盛り上がりに欠けることが多いが，今回は企業サイドと大学人との論争がリアルタイムで行われ，きわめておもしろかった．

また，大学教育に期待する以前に（いや，教育一般に期待する以前に），広い意味での開発支援環境を整備し，充実させることの重要性が認識できたのは，はなはだ有意義であった．

討論後半の話題として，ソース・コードを取り上げたのは，休憩時間の何気ない立ち話の中で，ヒントを得たからであるが，単純なソース・コード論議を経て，パブリックドメイン・ビジネス戦略・特許問題と，議論がどんどん進展し，活発な意見交換が行われたのはうれしかった．

今になって振り返ると，ソース・コードより上位の設計や要求定義に関する知識やノウハウの移転をどうするかについて，あるいは，環境そのものに関するノウハウについても，もう少し突っ込んで議論してみたかった．

たとえば，メインフレーム上での開発環境と今後実用化が進むであろう高性能なワークステーション上の環境とを，どんな風に使い分け組み合わせるかが，これからのかなり重要な技術的課題になると思われる．

こうした課題に対して，また来年以降のワークショップで，ぜひ討論してみたいと思う．

第２回　実践的開発環境に関する集中討論 in 長岡

Session 2:　技術移転

討　論　記　録

レポータ
盛　田　政　敏

１．大学における教育と研究

杉田義明（ＳＲＡ）：　それでは，君島さんのキーノートを受けて，これから自由討論に入りたいと思います．

鳥居宏次（阪大）：　大学教育について説明されたＯＨＰをもう一度見せて下さい．

君島：　これでしょうか？　これは，大学でのコンピュータ教育に対する一般的な批判を箇条書きにしたものです．要点をまとめれば，おおよそ次のようになる：

（１）　底の浅いプログラミング教育に終始しがちである．

（２）　現実のソフト開発との関連や応用についての説明なしに，むずかしい理論だけを教えている．

（３）　ソフトウェアとあまり関係のないシステム工学やハードウェアの教育がかなりの部分を占めている．

（４）　どうも日本語の文章教育がなっていない．大学でもっと教えてほしい．

私は上の３つには同感ですが，最後のドキュメンテーション教育は，企業でやるべきだと考えます．技術文書の書き方は，学習院大学の木下先生などが熱心にやられていますが，こういう基本的な教育は大学というよりむしろ小中学校段階の問題だと思います．

ところで，こうした批判が出る裏側には，逆に，大学の情報学科で何を教えて欲しいのか，企業サイドの要求がはっきりしていないということがあるでしょう．望ましい講座のタイトルを５つ挙げよといわれて，さっと答えを出せる人はほとんどいません．

鳥居：　最近の新入社員の知的能力が低下したといわれましたが，能力評価の基準は？

君島：　あまりよくないかも知れませんが，出身大学の偏差値でマクロに傾向を見ています．入試の時に使われている偏差値です．

鳥居：　ということは，入学から卒業までの４年間は無視しているわけですね？（笑い）．

君島：　そうです（大爆笑）．

片山卓也（東工大）：　大学のソフトウェア研究者が，積極的に研究資金を集める努力をしないのは，志が低いというようなニュアンスのお話があったが....

君島：　アメリカとくらべると，日本の大学は，気迫や意気込みの点で，やや欠けるような気がします．

片山：　しかし，お金ばかり集めてもソフトの研究はできません．なぜかといえば，日本の大学の制度では，アメリカと違って，お金があっても人（研究開発に必要な学生アルバイトやプログラマ）を雇えないからです．

研究用の機械装置を買うことも必要だが，それ以前に，問題はいかに良い人材（学生）を研究室に残しておくかということなのですが，それがなかなかむずかしい．ソフトウェアの研究は，ご承知のように，概念をいじるような仕事が多いので，ただ頭数だけそろえてもしかたがない．私はそんな風に考えていますが....

君島：　たしかにそうでしょうが，最近，新しい形の産学共同方式で何とか現状を打破してみようという動きもあるようです．そのへんに注目したいと思っています．

片山：　それから，私自身いままで理論的な研究をやってきたので，理論が役に立たないということについて，一言いわせてください（笑い）．

役に立つことがはじめからわかっていれば，本来，技術移転などということは問題にならない．もともと理論的な研究の成果は，直接現場の仕事には使えないので，我々としては，一歩退がって哲学を教え，間接的な波及効果として，世の中のお役に立てばよいのではないかと考えているのですが....

君島：　ＣＭＵのウルフたちが書いた計算機科学の入門書を見ると，ソフト開発のどんな所にどんな理論が使えるかが，きわめてわかりやすく示してある．私がいいたかったのは，研究レベルの話ではなく，教育のなかで，

理論をどこでどう応用するかを，明らかにしたらどうかということです．

片山： リクルートの話ですが，最近うちの大学でも，やはり，大企業へ行く人が減っている．学生時代からソフトハウスを作ったりして，小さいところで能力を発揮しようと考えている人が割合多いようです．かと思えば，マスターを優秀な成績で出て，銀行へいった人もいるし，N社へ行った人もいる．

鳥居： ということは，F社の魅力が減っているのでは？（笑い）．

君島： たしかに給料や勤務場所の点では不利になっています．問題は，仕事の内容に魅力がなくなっているのかどうかです．もっと，特許や論文の数を増やすよう，尻を叩かなければいけないと感じています．

杉田： 大学の先生に一言文句をいうと，こういう具合いにいろいろな反論がはね返って来るので，気をつけて下さい（爆笑）．

中野秀男（阪大）： 杉田さんのポジション・ペーパに，日本の大学ではソフトウェア・エンジニアリングよりコンピュータ・サイエンスに重点が置かれていると書いてありましたが，具体的にどういうことなのですか？ それから，ソフトウェア設計につながらない基礎理論教育が多い，といわれてもよくわからないのですが....

杉田： それは私への質問ですか？ 困ったな（笑い）．実は，今年の正月にハワイで行われたソフトウェア工学に関する日米ワークショップに，オブザーバとして出席しました．

日米それぞれの大学の先生方が，自分たちの研究の概要を紹介し合い，新しい共同研究の可能性を考えようといろいろ議論したのですが，その時の率直な感想として，日本での研究がコンピュータ・サイエンスに偏っていると思ったわけです．

具体的にいえば，今回のワークショップのテーマである開発環境とか，プロジェクト管理（これはちょっと難しいかも知れませんが）を研究されている先生はほとんどいない．逆に，コンパイラの基礎理論とか，ＡＩ，オブジェクト指向，自然言語処理などの分野には，研究者が集中している．

エンジニアリングには，研究の成果を何らかの形で実験をし，その結果を反映して次のモデルを作るというようなフィードバック・ループが必要ですが，研究と実務とのあいだのそういった連携が，ほとんどとれていない．

中野： それは，簡単にいえば論文にならないということだと思うのですが，斎藤先生いかがでしょう？

斎藤信男（慶大）： あまり論文を書いていないので，よくわかりません（笑い）．たしかに，大学では論文を書く事も重要かも知れないが，ほんとうに大切なのは人材教育だと思います．

そういう意味では，本来，大学でＣ言語の構文なんかを教える必要はない．どうやってプログラムを設計するかとか，ソフトウェアの要求仕様とは何かを教えなければならないが，なかなかむずかしく，準備も大変なので，とりあえずプログラミング言語の文法を教えているというのが現状ではないかと思います．

日米の大学教育を比較した調査結果によると，日本の大学は無用の長物であり，大学院はもっと悪いといわれている（笑い）．それに対して誰も反論できない．私の大学でも，コンピュータの教育はマシンに電源を入れる所から始めなければという頑迷な信念にもとづく教育を１０年以上続けている先生がいます．こういう所から，世の中に役立つ人材は決して育たないわけで，情報学科なんかなくても大丈夫という産業界の人たちの意見も十分理解できる．

そういう意味では，偉い先生方はいろいろ反論されましたが，私は，君島さんや杉田さんの意見に全面的に賛成です（爆笑）．

２．プロはどうすれば育つか？

佐原伸（野村コンピュータ・システム）： いままでの議論で，１つ疑問があります．つまり，よい教育をすれば自動的に技術移転が進むという暗黙の前提で話が進んでいるようですが，私自身のサッカーや囲碁の世界での経験に照らしてみると，教育にはそれほどの効用は期待できないと思います．

プロフェッショナルを養成するには教えてはいけない．彼らが自分で何かを学ぼうとするさいの環境を整えてやることは必要だが，教えてしまうとプロは育たない．せいぜい，よくできるアマチュアにしかならない．ソフトウェア・エンジニアリングでもそうだと思うのですが，皆さんのお考えは？

柿下尚武（カスタム・テクノロジー）： プロになれる人間の数には限度があります．どう考えても人口比３％くらいでしょう．ソフトウェアが産業になるには，この３％の枠からはみだした人間をどれだけ集められるかが

問題で，そこでは，プロとアマチュアをいかにうまく使いわけるかを考える必要があると思います．

建築業界を見ると，いわゆるデザイン・ハウスとゼネコンの２種類の会社がある．前者の方は，徒弟制度で（人間のローテーションの）サイクルは１０年位です．一方後者の場合は，サイクルが短く人の出入りが多く，組織も大きなピラミット型になっている．仕組みの違う会社が二重構造を構成しています．

オリジナルなアイデアにもとづいてまったく新しいソフトを創り出そうという場合と，とにかくお金を稼ごうという場合とでは，状況が違うのではないでしょうか？

ソフトウェアを十把一からげにする時代はそろそろ終わりで，どういう種類のソフトをどんな組織で作るか，またそのための人材をどういうメカニズムで育てるかを，明確にしなくてはならないのではないでしょうか？

たとえば，金融オンライン・システムと，ＣＡＤ／ＣＡＭでは，必要な技術がまったく違うでしょう．道具としてのソフトウェアにも，きわめてプリティブなものから，素人向けツールまでいろいろあります．そうすると，情報学科でしかソフトウェアを教えられないというのはおかしい．

もっと細かく分けて考えた方がよいのではないか，単純に「技術」の一言ではかたづけられないような気がしているのですが....

君島： お二人の話に対してコメントさせていただくと，まず，プログラミングの技能教育をするのはよくないと思います．やはり理論的な面の教育にとどめて，応用面は各自が考える．そういう風にしないと，高度な技術は習得できません．私は，理論教育がだめといっているわけではなく，どこでどう使うかをはっきりさせた形で，理論教育を強化すべきだと思うのです．

次に，いろいろな学科で情報教育を充実しなければならないという点は，まさにそのとおりだと思いますが，あまりそのことを強調しすぎると，情報学科プロパーの教育の印象が薄くなってしまう．両方やらなければならないでしょう．しかし，一方では，生産性がもっと上がれば，専門プログラマの人数はもっと少なくてすむという皮肉な意見もあります．

日本ではソフトウェア・エンジニアリング教育の影が薄いという杉田さんの話に関連していえば，アメリカのＡＣＭのカリキュラムが，昔は１種類だけだったのですが，いまはコンピュータ・サイエンスとインフォメーション・エンジニアリングの２種類に分かれています．

ところが，日本ではこうした体系化の動きがなく，ただあちこちの大学の理・工学部に，次から次へと，情報関連の学科が作られているだけなのです．

杉田： さきほど柿下さんがいわれた３％の話に，基本的に賛成します．ところで，残りの９７％の人をどうやってチームの中に組み入れて，生産性の高い仕事をしてもらうか，そのための具体的な仕掛けや戦略が，非常に大切だと思うのですが....

鳥居： ＯＪＴのような話ですか？ まず，下手でもよいからやらせてみれば，除々に仕事のやり方を覚えて行くだろうという．もちろん，できる人はすぐできるようになるでしょうが，できない人はどうなるのか？ 結局，３％の人を選んでいることになりませんか？

今後，もし，平均的な技術者の人数をふやさなければ，と企業サイドが考えているのだとしたら，別なアプローチを取らざるをえないのではないですか？

大学教育でも同じような問題があります．私の教室では，何も教えないでまずプログラムを作らせ，その後で設計法をもう一度教えていますが，他の先生からは，そんな無駄なことをするな，初めから教えておけば時間が節約できるではないか，と批判されています．その辺のことを聞かせてほしい．

佐原： ３％の話，日本ではとてもプロの比率が３％に達していない．１％ぐらいしか育たずに，残りの２％は途中でつぶされてしまう（笑い）．

それは，明らかに，鳥居先生がいまおっしゃった，時間をかけないで合理的に教えた方がよいという，生産性至上主義の弊害でしょう．小学校からしてそうですが，人間の促成栽培を目指した結果，レベルの低いアマチュアしか育てられない．全体の底上げを狙った教育で，はたして何が解決できるのか，という問題だと思います．

ソフトウェア・プロフェッショナルを育てるには，底上げ教育ではなくて，岸田さんあたりが主張され，自分の会社やＪＳＤで実践して来られた「企業内遊園地」の考え方，かなり誤解を招きそうな危険な呼び名ですが（笑い），こっちのほうが効果があるでしょう．ともかく，若い人をそういう環境にほうりこめば，おそらくあるレベルまではスーッと伸びることはたしかです．そこから先はまた別の問題ですが....

第二種情報処理技術者受験講座というのが，最近テレビで放送されていますが，見ていると，どう考えても，

底上げというより，むしろ底を下げている感じしかしない（笑い）．これが，いまの日本の教育の典型的なパターンのように思われます．

企業内（あるいは大学内）遊園地の概念は，昔の旧制高校なんかと相通ずるところがありますが，ある意味で本来の目的と無関係なことをして遊びながら，実は平均レベルがスーッとあがって行く．もちろん，その中でも本当にプロになるのは小人数しかいないでしょうが，いまは，教育制度とは別な所で技術レベルの向上がなされていて，教育は逆にレベルを下げている．

柿下：　私の所のような小さい会社は，別に優秀な人が集まっているわけではありませんが，不思議なことに，開発チームのなかに一種の興奮状態ができると，能力が急に上がることがある．３％の人が意外に身近で発見されたりします．

いずれにしても，技術者を育てるには，かなり時間がかかるものだというのが，私の実感です．私の会社では，ソフトだけでなく，ＬＳＩの設計もやっていますが，１０年ぐらい同じ仕事を技術屋としてやらせてあげると，大企業にはいないような人材が育って来る．技術の熟成には時間がかかるので，途中で担当を変えたりすると，だめになってしまう．むしろ，そういう点では小さい会社の方がやりやすいかも知れません．

大企業からうちへ来た人のなかには，管理職にさせられるのがいやだから，ずっと技術屋を続けていたいから，という理由の人が結構います．そういう人たちを，大きな会社では救ってあげられないのか？　もっとも，下手に救われたら，こちらが困りますが（笑い）．

我々のような小さな会社は，仕事の枠組をそんなに広げることはできませんが，逆に，あるひとつの領域に特化して，それ一筋に１０年間ずっと同じことに取り組める．というより，それしかできないのです．ところが，そんなことをやっている人はあまりいないから，自然に他人より優位に立てるということになります．

だから，３％と９７％の議論は，だれが３％かと血眼で探すのではなく，仕事の組み立て方しだいで，３％の人間を育てられるような気がしています．

杉田：　どうもありがとうございました．一応この辺でコーヒー・ブレイクにしましょう．そして，また気分を変えて議論を続けたいと思います．

――――休　憩――――

３．技術移転とソース・コード

杉田：　さて，ここで話題を教育問題から一転させて，開発環境に関する実践的な技術の移転を取り上げてみたいと思います．それでは，話のきっかけを，ＩＰＡの永島さんからお願いします．

永島晃（ＩＰＡ）：　シグマ・プロジェクトでは，環境を構成する各種の支援ツールを開発するために，いろいろな会社からアイデアを募集し，提案された内容の検討や調整を行っています．その中で，ソース・コードの問題がでてきました．

つまり，ツールをソース・コード付きでユーザに提供しようと考えて，何社かに相談したところ，開発者および利用者の双方から，こちらが期待していたのとは逆に，拒否反応がありました．このことについて，みなさんはどうお考えですか？

小池幸徳（日本システム・サイエンス）：　私も，ソース・コードを受け取ってもしかたがないと思います．ソフトウェア・プロダクトを開発する場合，できるだけ自分たちだけの知恵で仕事を進めるのが，正しいやり方でしょう．

結果的に，その内容がノウハウとして残り，技術の付加価値を高めていくことになります．他人のアイデアを盗むことを考えない限り，ソース・コードをわざわざ見る必要はありません．

林香（ＳＲＡ）：　いや，ソース・コードは絶対にツールの利用者に提供されるべきだと思います．その理由は，使ってみてすばらしいと思ったソフトウェアについて，そのソース・コードを見ると，また更に感動が深まる場合があるからです．

ソース・コードを読むと，開発者の考えた筋書が読めてきます．このことは，また，プログラムの再利用を推進する上でも，きわめて重要です．

岸田孝一（ＳＲＡ）：　私は，若いとき，プログラマになる前，画家を志して勉強したことがあります．そのとき，現代美術の新しい世界を切りひらいた先駆者の一人である画家パウル・クレエの造型講義テキストを読みました．その冒頭に，「芸術における分析は，自然科学のそれとちがって，単に作品を構成要素に分解することではなく，作品が創造された道程を再びたどってみることだ」という趣旨のことが書いてありました．

ソフトウェアは芸術作品ではありませんが，その成り立ちを分析するにあたって，ソース・コードが貴重な手

がかりになることはたしかです．

既存の環境やツールを実際の開発現場に生かすためには，それぞれのプロジェクトの状況に応じたカストマイゼイションが必要になりますが，その場合，移転対象の環境やツールが，もともと，どんな考えで開発されてきたのかを知るのに，それ以外に有用なドキュメントが存在しない現在の一般的状況を考えれば，ソース・コードはきわめて重要な材料だと思います．

藤野晃延（富士ゼロックス情報システム）： ＳＭＡＬＬＴＡＬＫの場合も，全般的な技術移転をスムースにするために，一応はライセンスの保護メカニズムを設定した上で，ソース・コードを利用者に提供するようになっています．

伊野誠（日本ユニバック）： ＳＭＡＬＬＴＡＬＫを使ってユーザが開発したソフトウェア・プロダクトも，ほとんどが，そのポリシーを引き継いで，商品としてソース・コードを提供するのが原則になっているようです．プログラミング言語としてのＳＭＡＬＬＴＡＬＫの魅力は，このことによって随分大きくなっているように感じられます．

４．パブリック・ドメイン

二木厚吉（電総研）： たしか，オーランドで開催された第７回ＩＣＳＥの時に，フリー・ソフトウェアという考えをハーバード大学ＡＩラボのリチャード・ストールマンという人が提案していました．これは，すぐれた基本ソフトウェアを無料でドンドン配布し，それに関する技術情報の交換を進めようという，非常にラディカルなアイデアです．

かれは，その後その信念にもとづいて自由ソフトウェア財団を起こし，ＧＮＵプロジェクトをスタートさせて，自分が開発したＥＭＡＣＳエディタの無料配布を始めています．いずれは，Ｃコンパイラや０Ｓカーネルも無料で配るそうです．１ユーザの立場からすれば，歓迎すべき動きですが，こんなことが一般化したら，問題は起きないのでしょうか？

村井純（東工大）： この問題は，パブリック・ドメイン・ソフトウェアを具体的にどう活用し，自らも広げていく努力をするかという問題につながると思います．

この点については，津田塾大の小川先生が開発したＳＳＥエディタのパブリック・ドメインとしての普及方法が参考になるかも知れません．これは，不特定多数を対象にせず，そのソフトウェアの良さを理解してくれそうな人だけを対象にして，使用結果の評価と交換に配布するという形です．

ＧＮＵの場合ほど自由ではありませんが，多分，こうすればそのソフトウェアの真の理解者が増え，着実に利用者層が拡大して行くものと予想されます．

斎藤： パブリック・ドメインに流通させたらよいと思われるものの１つに，文字フォントがあります．

デスクトップ・パブリッシングと呼ばれる，ワークステーションを活用しての電子出版技術が話題になっており，我々の研究室でも，ＴｅＸの応用についていろいろと研究開発を行っています．しかし，日本語での電子出版を考えた場合，どうしてもそのフォントの入手が問題になります．

通常は，レーザ・プリンタのファームウエア上のＲＯＭ内に格納されていますが，それぞれのハードウエア・メーカごとに，その体系や，書体が異なっており，一般に電子出版を普及させるためにはどうしてもパブリック・ドメインのフォントが必要です．

この問題を１度，シグマの技術委員会で提案したのですが，まだ返事が返ってきません．

永島： 斎藤先生の提案が今どうなっているのか，私のレベルではよくわかりませんが，提案書を御提出いただいてあれば，検討が進んでいるに違いありません．

５．ソース公開とビジネス

片山： パブリック・ドメイン扱いでソフトウェアがどんどん普及していった場合，それを販売しようとしているソフトウェア会社は，商売がなりたたなくなるのではないかと心配するのですが，実際にソフトウェア・ハウスの方々はこの問題に対してどのようにお考えなのでしょうか．

岸田： ビジネスに関連してパブリック・ドメインの問題を考えると，２つのとらえ方があると思います．

１つは，研究開発の段階で一応できあがったプロトタイプの評価や改善を行うための場として，パブリック・ドメインを活用することです．つまり，テスト・マーケティングおよびβ－テストを，パブリック・ドメインのユーザを対象に実施し，その結果をベースに，商品を完成させていくという戦略です．

こうすれば，最終商品が世の中に出たときには，その評判はクチコミですでに広まっており，余計な宣伝は不

要になります．また，プロトタイプのテスト・ユーザも，当然，より機能が充実した本番の商品を買ってくれるでしょう．

２番目のより重要な戦略は，ソース・コードを一般に公開することによって，単にビジネスそのプロダクトの販売だけに終わらせることなく，より大きな，新しいマーケットの創造を狙うという考え方です．ＳＵＮマイクロシステムは，オープン・システム・アーキテキチャという旗印のもとに，そうしたマーケティングを展開してかなり成功しているようですが，この点については，村井さんに補足していただければ幸いです．

村井： ＳＵＮは，無条件にすべてのソースを公開しているわけではありません．ひとつの技術戦略にもとづいてやっています．このことをよく理解する必要があるでしょう．

ソース・ライセンシングに関しては，ＵＮＩＸの場合と同様に，大学向けと企業向けとでは，価格や制約条件などの点ではっきり差別をしています．カーネルとのかかわりある微妙な部分は，巧妙にそのノウハウの公開を隠しているようです．

現在ＳＵＮの主力ソフトウェア商品であるＮＦＳ（ネットワーク・ファイル・システム）については，ソース・コードをはじめ，ほとんどすべてのノウハウを公開して，そのアーキテクチュアに合致す関連関プロダクトを，周辺のハード／ソフト・メーカにどんどん開発してもらおうとしています．これは，一見ボランティア風の，単純に気の合う仲間を増やそうという動きのように見えますが，もちろん，それだけではありません．その奥には，ＳＵＮ独自のビジネス戦略があるようにうかがわれます．

１つは，いわゆる第３世界コンピューティング，つまり，ソフト非公開のクローズド・アーキテクチャ戦略でこれまでの世の中を支配してきたＩＢＭ（第１世界）や第２世界（ＤＥＣ）を，オープン・アーキテクチャ思想に共鳴する同志を集めて包囲するという長期的な狙いです．

また，短期的には，当面の競合プロダクトであるＡＴ＆ＴのＲＦＳが市場に出て来る以前に，ＮＦＳの絶対的優位を確立し，その結果として，ワークステーションやその他のソフトウェア・プロダクトの今後のマーケッティングを有利に進められるようにしようと考えているようです．

６．ソフトウェアの特許

二木： いままでの議論を聞いて，ソース・コードの公開やライセンシングとビジネス戦略との関係は，おおよそ理解できました．それに関連して，さきほど君島さんがソフトウェアの「特許」のことをいわれたような気がしますが，ちょっと解説してください．

君島： ソフトウェアの権利保護は，いま，著作権が中心になっていて，それ以外に特許を使う方法があることは，一般にはあまり知られていません．

これは，１９７６年から出願の受付が始まっていて，いまでは，毎月何十件かが公告されています．この中の相当部分が１９９２年ごろに特許になるでしょう．出願は月千件ぐらいだと聞いています．現在は，計算機メーカが主に出願していますが，ソフトハウスの方もぜひ特許を出願するようにおすすめします．

二木： ソフトの特許というのは，たとえばどんなものですか？

君島： パソコンやゲーム・マシンのアニメーションにおいて，画面上で何かを移動するときに，背景が隠れるので一時的に隠れる部分の絵を退避する．そのときに排他的論理和を３回繰り返してデータを交換するという有名なテクニックがありますが，これは特許になっていて，もしこの方法を使いたければ，特許使用契約をしてお金を払う必要があります．

鳥居： 特許として認められるようなすばらしいアイデアや商品があるのなら，それをどうして論文の形でまとめようとしないのですか？

大体，いろいろなコンファレンスやシンポジウムに産業界からの論文が少なすぎるということが，現在の日本におけるソフトウェアの技術移転を困難にしている大きな原因だと思います．開発現場における現状の問題点の分析や，実践の中から生まれたいろいろなアイデアを，積極的に論文にまとめて発表していただきたいと，強く希望します．

杉田： ちょうど予定の時間が近づいたところで，鳥居先生からクロージングにふさわしいお言葉が出ました．どうも，我々産業界の人間は「論文を書け！」といわれると，ギャフンとなってしまいますが（笑い），この辺で討論を終わりにしたいと思います．

春のセミナー・ウィーク：　Session A1

要求定義支援ツール

講　師：　松本　吉弘（東芝）
報告者：　渡邊　雄一（電力計算センター）

1．はじめに

ふだん仕事をしていて思うことがある．

エンド・ユーザが出してくるさまざまな要求（それは要求というより，むしろ“ぐち”に近いものが多い）をワープロみたいな機械を使って順番に入力しておく．そしてある日，「みんなの勝手なわがままも聞きあきたから，そろそろシステム作りに取りかかろうか」と，おもむろに特殊なコマンドを入力すると，「いくつかのシステム・デザインを試みてみましたので，お好きなものを選んで，機能の修正または追加をして下さい」と問い返してくれる．

究極の要求定義支援ツールとは，そんなものだろうという期待が，いつも心のどこかにある．今回の松本さんの講義のなかにも，きっとそうした夢のツールが出てくるものと，勝手に思い込んでいたのだが，やはり現実はきびしく，甘い夢物語は聞かせてもらえなかった．

では，セミナーに参加しても無意味だったかといえば，答は「ノー」だ．結論として，「ウ〜ン，要求定義ってやっぱりむずしいんだなぁ！」というごく平凡な事実を再認識しただけではあるが，むずかしさの中身についての理解を，以前よりずっと深めることができたと感じている．

以下，今回の講義の，主として前半の内容をまとめた個人的なノートを，恥ずかしながら披露することで，レポータとしての責務にかえさせていただきたい．

また，ことわりしておくが，当日説明された内容は，講師自らの筆による参考文献［2］〜［5］に詳しく書かれてある．しかし，「詳細はそちらを読めばよい」といってしまっては，身もふたもない．実際，論文を読むだけで理解できるなら，何も苦労はしない．

私自身，このセミナーで直接に松本さんのお話をお聞きして，やっとその真髄が理解できたという気がする．あらためて，人の文章の行間を読み取ることのむずかしさを痛感させられた．

2．観点（view）の重要性

現在，我々がソフトウェアを作っている手順を振り返ってみると，図1のようになる．

この図は，一見あたりまえのように見えて，実は，現状におけるシステム開発の問題点を示す重要な意味を持っている．なかでも，ポイントは，中央に位置する抽象機械作成過程である．

すなわち，この過程で，システム分析者あるいは設計者は，「与えられた要求をどうやって実現するか？」という抽象的なモデル化の作業を繰り返すわけだが，これらの実現モデルは，その時点では人間の頭の中にしか存在せず，たとえ外部に出力されたとしても，あくまで非公式なメモの形でしかない．

次の資源割り付けの過程に入ると，ようやく前段階での成果が，基本設計書の形にまとめられ，詳細設計書からプログラムへと，機械にのせるための記述手段を用いて表現される．

以上2つの過程は，その性格が本質的に大きく異なっている．実際，これらの間のギャップは非常に大きく，その間の変換はきわめてむずかしい．

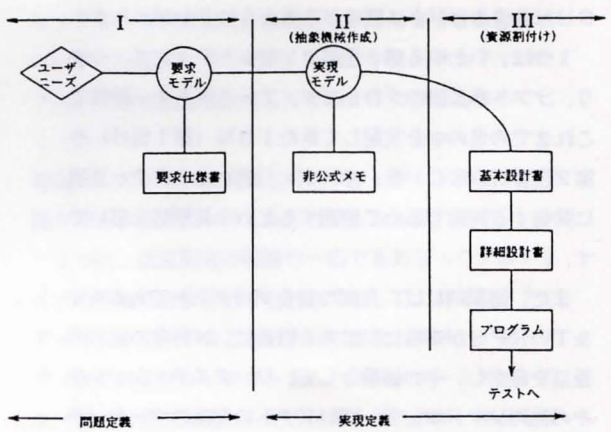


図　1

それでは実際にこれらの過程をもう少し詳しく見てみよう．

まず，利用者からの要求はどのようにして決まるか？

一般に，目的システムに対する制約と，システムによってもたらされる効用とを折衷したものが，利用者のニーズであると考えられる．それにもとづいて，計算機内で処理しなければならないことがらをまとめたものが，要求である．それは，利用者が与える入力から目的とする出力を得る過程での，さまざまな制約条件や計算機環境上の規制を，何らかの形で含んでいる．

こうして一応は定められた利用者要求を，次に，さまざまな観点から分析することが，きわめて重要である．今回の講義では，有名な酒倉庫問題［８］をとりあげ，いくつかの典型的な観点による分析の例が示された：

(1) Morphological View（形態的観点）

絵による表現を用いる．きわめて初歩的なアプローチではあるが，文章でごちゃごちゃ書いてあるよりはわかりやすい．記述に関する形式上のルールは存在しない．

(2) Functional View（機能的観点）

まず，システムの持つべき機能を洗い出して，それぞれに対する入力データ，出力データ，動作の起動条件，機能を実現するための資源を図に示す．代表的なシステムとしてＳＡＤＴがある．

(3) Contexual View（前後関係的観点）

目的としている対象の中に，どのような単位があるか？ そしてそれら単位間には，どういう相互関係，前後関係があるかを記述する．代表的なシステムとしてＰＳＬ／ＰＳＡやＳＲＥＭがある．ＥＲ（実体関係）モデルとも呼ばれる．

(4) Dynamic View（動的観点）

事象とその時間関係，論理関係を明らかにしようとする観点である．タイミング・チャートや状態遷移図がよく知られている．

ここで重要なことは，どのような問題（対象）に対しても，万能な観点は存在しないということである．したがって，我々としては，つねに複数の観点から，利用者の要求を記述し分析してみる必要性が生じる．さまざまな観点からの「展開と縮約の繰り返し」が行われるわけである．

そのさい，個々の観点からの記述が，それぞれ１つの要求モデルとして，コンピュータ上に保持されていると便利である．それらの要求モデルは，システム分析者の指定に応じて，必要な情報を印刷表示してくれる．それが「要求定義支援ツール」として望ましい．

３．現実の問題

モデルだけでは要求は記述出来ない．また顧客（ユーザ）の側からは，自然言語で書かれた仕様書を求められることも，一般的である．講師は，以下の４点を，現実の問題点として指摘された：

- 要求が外の世界，例えばプロセス制御のように他の機械（例えば発電機，タービン等）と強く関連する場合には，それらの仕様が決まらないためにソフトウェアの要求定義が，それに付随して非常に決まりにくいことがある．
- 顧客の持っている知識やノウハウが，ソフトウェア開発者側になかなか伝達できない．そのため，仕様決定作業に，顧客が立ち入ってこなければならず，責任の境界があいまいになる．システムの規模が大きくなればなるほど，この問題は大きくなってくる．
- 最近のシステムでは，ヒューマン・ファクタ（人間的要因）がますます重要になってきているが，それらを十分に取り込むことができない．
- 要求仕様ばかりいくら作ってみても始まらない．その意味で，実現可能性を持った要求をいかに形成して行くか，後の工程との整合性を持った要求仕様をいかに作るか，ということが問題である．

４．既存の手法に対する疑問等

ここまでに示した手法を，講師は「古典的手法」と呼び，参考文献［２］で，次のように総括している：

『これら一連の思想の流れのなかには，構造化プログラミングや段階的詳細化，実務向けの設計方法論に代表されるトップダウン思想と，コードとの対応を明確にしながら抽象度を高めていこうとするボトムアップ思想が共存していた．〔中略〕 以上に述べたようなものを古典的アプローチと考えているが，その残念な点は，トップダウンとボトムアップが個々にそれぞれ独立に進められていて，どちらも要求からプログラミングまでをカバ

ーするような一貫性のある方法論にまでは到達できず，中途半端なものに終わっているということである．実務では，かならず途中のどこかで，トップダウンとボトムアップの接合点が現われ，そこで相互の間の編み合わせが行われているが，これに関するまともな扱いが後回しにされている点が残念である』

このような主張に対して，私は次のような疑問を抱いた．

「古典的」手法という割には，それらに対する適当なガイドブック（参考書）が世の中に見当たらない．またここに示されたモデルに限らず，一般に要求定義や設計に関する情報が計算機化されていない．

そういしたことが，これらの手法の一般への普及を妨げているのではなかろうか？　コンピュータ化されていない手法に対しては，それを採用しようとした場合のリスクに対する経営者側の態度は一般に厳しくなりがちである．

我々がプログラミング言語を習得するときにも，まず，コンパイラなりインタプリタなりを入手し，それを使って簡単な例題を作成し，ミスを繰り返しながら言語の特徴や機能を学習得して行く．それと同じように，特定のモデル化手法について，「ああ書けば良い」「こういう書き方はまずい」と指導してくれる人間や支援環境が存在しない限り，その習得は困難である．

「古典的」手法が，あくまで書物の上での手法にとどまっているかのように見えるのは，実にもったいないことだと思う．

この問題に関連して，松本さんは，要求定義を完全にするにはどうしたらよいかという立場で，文献［3］で次のように主張されている：

『これまでは，要求仕様を書く際に用いられた観点，すなわちセマンティックスが実現体のそれとは異なることを前提としてきている．〔中略〕　これだけ生産性の向上が厳しくなった今日，ひとつのライフサイクルのなかで，複数のセマンティックスを仕様記述に際して許すことは好ましくない．全ライフサイクル・フェーズを一本のユニークなセマンティックスで押し通すことを考えなければならない．したがって，一歩進めて，要求仕様を含めてすべての仕様書が，唯ひとつのセマンティックスで記述できないかということについてこれから考えてみよう．さて，ユニーク・セマンティックスに耐えることのできるセマンティックスが現実に存在するであろうか．さきに，種々のモデルや仕様について紹介してきたが，これらのもつセマンティックスはいずれも帯に短し，たすきに長しで適当なものがない．あるものは実現性は十分であるが，要求レベルでの記述にはあまりにも精密過ぎて適当でないし，あるものは検証可能性や理論的裏付けにはすぐれているが，実現性には乏しい』

しかし，また一方で，「インプリメントする言語により，システムの設計（例えばオブジェクトの選び方，タスクの設計の仕方）が変わってくる」という意見も，世の中には根強く存在する．理想的には，ボトムアップに関する情報を知識ベースとして取り込んでおくような何らかの仕掛けが必要なのであろうか？

なお，蛇足ながら，私には「古典的」という言葉があまりふさわしくないように思えた．「従来型」くらいのほうが自然のような気がする．

５．これからのアプローチ

さて，図１に示したソフトウェア開発手順の問題点を改善することから，現代的アプローチが始まる．

すなわち，図２に示すように，まず，実現モデル作りにおいて，いままで非公式メモでしかなかったものを，精密化し，厳密なセマンティックスで記述する．そして，それを何らかの形で実行して，実現モデルとして顧客に見せる．今まではコンセプトだけを説明していたものを，抽象機械作成段階で顧客にデモンストレーションし，隠れたニーズをすべて吸い上げてしまおうという考え方である．

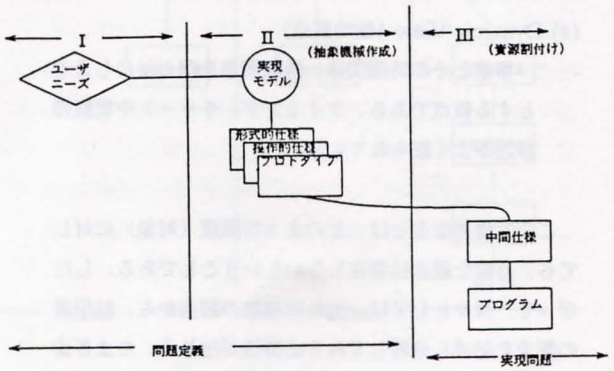


図　２

こうして要求仕様がまとまった後は，それを「出来るだけ手をかけずに」プログラムに変換して行くことが指向されている．完全な自動変換はすぐには不可能だけれども，できるだけ機械の助けを借りて変換を行なうやり方が，かなり将来性があると考えられている．

いいかえれば，ここで作られるプロトタイプや操作的仕様，あるいは形式的仕様は，要求そのものではなくて，その実現までを考えた１つの機構である．ただ，それを顧客が見て内容を修正することができるという仕掛けを，要求分析の段階における分析者と顧客とのインタフェースとして採用していることになる．

以下にそれら機構の特徴を示す．

(1) プロトタイピング
- 動作可能なモデルである．
- デモ，改良そして変更を繰り返す．
- 以下の３種類に大きく分類される：
  (a) シナリオ型（紙芝居型）
  (b) 原版型（バージョン０型）
  (c) 機能検証型

(2) 操作的仕様
- 資源割り付けを考えたものであってはならない（その必要はない）．
- 実現したいシステムの振る舞いがわかるものでなければならない．
- 実行可能．
- 設計のように聞こえるがそうではない（設計では資源割り付けを考慮する）．

「このパラダイムはプロトタイピングと次の点で異なる．プロトタイプはコードであり，プロトタイピングを通じて，それ自体が目的コードへと転換される．操作的仕様の場合には，正当性の検証を終わった仕様が，設計仕様，またはプログラミング仕様へ形式的に変換される」［７］

(3) 形式的仕様

要求定義段階で，実現へ向けて変換可能な形式的仕様を作成し，これを一定の変換規則を用いて目的コードへ逐次変換していこうとする方式．

６．ツール構築の実際

先に示したような個々の独立した観点を計算機がそなえていて，それから要求モデルを形成するようなツールを実際に構築することは，至難の技である．

「関係型データベースを用いて，いろいろな観点から定義した結果をしまっておき，所定の観点について必要なデータだけを編集して取り出してくる」というのが，要求定義ツールの一般的なものである．よって要求定義を実現するためには，非常に大きなデータベース機能と，それの上に皮をかぶせた“仕掛け”が必要となってくる．

７．おわりに

セミナーの後半では，実際に松本さんたちが開発・運用している要求定義支援ツールの一部が紹介された．

私のように，要求定義にしろシステム設計にしろ，ある意味でいい加減に仕事を片付けていてそれですんでしまう環境で暮らしているものの目には，初めのうち，セミナーの内容が何となく堅苦しいものに映ったことは事実である．しかし，講義を聞いているうちに，「いろいろな観点からのシステム・イメージを整理し保持して行く」ことの重要さが，よく理解できた．

いままで，現場では，形式的・理論的な話をしても実りが少ないからと，目先の成果にとらわれた行動が少なくなかったように思う．インフォーマル・スペックの実践にもそれなりの進歩は見られるが，ここで話のあったような王道をいくアプローチこそ，これから積極的に試みていかねばならないということを強く認識させられた．

【参考文献】

[1] 松本：ソフトウェア工学演習，朝倉書店，(1984)
[2] 松本：オブジェクト・モデルに基づいたソフトウェア設計(1), bit, Vol.17, pp.1157-1168 (1985.10)
[3] 松本：同上(2),同誌，pp.1331-1336 (1985.11)
[4] 松本：同上(3), 同誌，pp.1436-1448 (1985.12)
[5] 松本，田中：データ構造指向の要求分析技法，情報処理，Vol.27, No.2, pp160-170, (1986.2)
[6] 松本ほか：要求技術の目指すべき方向（パネル討論会），情報処理，Vol.27, pp.1415-1429, (1986.12)
[7]松本：要求定義・プロトタイピング技術の現状と課題，ソフトウェア工学の現状と動向，（社）情報処理学会，pp.11-22 (1986.12)
[8] 山崎：共通問題によるプログラム設計技法解説，情報処理，Vol.25, pp.934, (1984.9)

SEA春のセミナー・ウィーク：Session A2

設計方法論の比較

講　師：　落水浩一郎（静岡大学）
報告者：　村　井　進（フリー）

１．はじめに

このセッションでは，まず，言語・環境・設計方法論の定義から始まって，ソフトウェア開発プロセスの中での設計方法論の位置づけ，７０年代の方法論の特徴，８０年代における新しい動向，さらには環境構築のための設計方法論の適用例と，流れるように話が進み，最後に，最近話題になっているＦ．ブルックスの「銀の弾丸」論文を取り上げて，その概要が紹介された．

ＡＢ型の熱血漢（よく考えると矛盾した表現だ！？）として知られる落水先生の雄弁にのせられて，そのときは何となく，世の中の動きすべてが理解できたような気になったのだが，いざこのレポートを書こうと思って講義ノートを読み直してみると，短い時間の割には，きわめて幅広い話題がとりあげられており，その１つ１つを理解することさえ，いまの自分には容易でないことが明らかになってきた．

したがって，以下のレポートは，あくまで，私という偏光フィルターを通した講義内容のエッセンスでしかない．

２．言語・環境・設計方法論

プログラミング言語は，ソフトウェア開発のための最も重要かつ基本的な道具だというのが，落水先生の考えである．方法論は，言語の不備を補う機能を果たすことを目的としている．そして，環境とは，開発・検査・管理等の工程に含まれる作業の一部を，人手からソフトウェア・ツールに置き換えるための仕掛けである．

プログラミングとという仕事の目的は，比ゆ的にいえば，コンピュータと問題世界とをつなぐ何らかの「回路」を設計することである．

しかし，現在のソフトウェア工学技術のレベルでは，現実世界の問題が，十分には定義しきれない．つまり，一応の問題定義がなされたとしても，まだ，それは，コンピュータからの距離が，かなり遠いところに放置されている．

ソフトウェア開発プロセスの中でこの距離を埋める作業は，いまのところ，もっぱら人間の知恵に委ねられており，そのさい制御すべき対象は，質・量どちらの面でも，明確に限定されてはいない．

現在，ソフトウェア・エンジニアは，自分に許された範囲内で，与えられた言語や環境などの道具を使って，設計作業に従事している．そして，先にも述べたように，それらの道具を使いこなすためのノウハウを「方法論」と呼んでいる．

それぞれの会社，プロジェクト，あるいはエンジニア個人が，現実世界のイメージをコンピュータの内部にマッピングするために，独自の工夫をこらしている．そうした努力の中で，個別的な成功の事例が除々に積み上げられ，しだいに会社内あるいはプロジェクト内での標準的方法論として，整理されてきつつある．

設計方法論とは，「要求分析・定義」の工程と「言語・環境」とをつなぐための一種のブリッジである．過去に試みられたさまざまなアプローチの中から，何度でも再現することができ，そうすることによって必ず一定の効果を達成できるものだけが，現在，「名前のついた」方法論として，いくつか残っているのだと考えられる．

ところで，環境としてのＵＮＩＸの特徴は，すべての資源をＣという一つの言語で使用できるようにしてあるという意味で，環境の中での言語の位置づけが非常にはっきりしていることである．

たしかに，プログラミング支援環境としては有効であり，静岡大学の場合でいえば，以前のハード・ソフト環境では，１人の学生が２年間の時間を必要とした作業（演習課題）が，ＶＡＸ／ＵＮＩＸの導入後は，１年以内でできるようになった．

３．７０年代の方法論の特徴

７０年代におけるソフトウェア技術の特徴は，コンピュータの側（つまり言語と環境のまわり）で，機械が持つはたらきを仮想化し，少しでも問題世界に近づけよう

という努力が，さかんに行われたことである．

その結果，各種の高水準言語（コンパイラ言語から第４世代言語まで），制御構造またはデータ構造の整理をベースとする設計技法群，抽象データ型やオブジェクト指向の概念にもとづく新しいプログラミング・パラダイムなどが登場し，そのうちのいくつかは実用化のレベルに到達して，数多くの技法・ツール・環境などが開発され，利用されるにいたった．

一方，現実世界の側，つまり要求分析や定義に関しても，問題世界をそれに密着した記述で書きとめ，解析し安定化させるための努力が，ある程度は行われた．その結果，疑似自然語，データフロー図式，事象／反応図式など，やはり数多くの新しい記述技法が提唱されたが，実用化という意味からすれば，いま一歩の感は否定できない．

そうした動きの中で，設計方法論もいくつか提案されたが，その役割は，上述の２つの記述の間のギャップを何らかの形で埋めるための手続きを定式化することであった．つまり，機能構造あるいはソフトウェア構造の形で記述された内容を，どういう過程を経て，マクロやサブルーチン，または言語に落とすかということである．

下図に，７０年代における設計方法論関連のテクノロジー・マップを示す．

４．８０年代の新しい動向

８０年代になると，７０年代の研究および実践の結果（あるいは失敗の反省）を踏まえて，さらに数々の新しい試みが，次々に登場してきた．

それらをランダムにリスト・アップすれば：

- オペレーショナル・アプローチ
- 図式プログラミング
- 論理型プログラミング
- 関数型プログラミング
- オブジェクト指向プログラミング
- マルチ・パラダイム・プログラミング
- 知識工学アプローチ
- ルールベース・エキスパートシステム

などである．

	基礎理論	方法論	仕様化・言語	支援環境
要求分析			SADT	
			ISDOS	
要求定義	実体関係モデル		PSL	PSA
			SREM	
確認			RSL(R_NET)	シミュレーション
分解	Parnas法			GAP
	情報隠蔽	仮想機械	Parnas仕様	
			複合設計	
設計		強度・結合度	複合分析	PL/I
	構造化プログラミング		高水準言語 ＋	その支援環境
プログラミング			PASCAL	Unix
			C	構文エディタ
		真珠のネックレス	段階的詳細化	
			Jackson法	COBOL
検査		構造テスト		Test Harness
変更・再利用				構成管理

これらについては，すでに，昨年の秋のセミナー・ウィークでも，落水先生自身がお話されており，そのときのレポートも，すでに昨年１２月号の本誌に掲載されているので，ここでは詳細は触れない．

８０年代の方法論の特徴をひとことでいえば，それは，きわめて豊かなハードウェア資源を持つ支援環境を前提としていることであろう．今日の設計支援環境は，ＣＰＵ能力においても，また，記憶容量の面でも，７０年代とは比較にならないほど強力であり，ユーザ・インタフェイスも洗練されてきている．

５．銀の弾丸

講義の最後の部分で，落水先生は，最近話題になっているＦ．ブルックスの論文「銀の弾丸は存在しない：ソフトウェア工学における本質と偶然」をとりあげて，その概要を紹介された．ブルックスについては，ＯＳ３６０の開発・管理に携わったときの経験にもとづく名著「ソフトウェア開発の神話」が邦訳されているので，御存じの方も多いだろう．

この「銀の弾丸」論文は，昨年８月アイルランドで開かれたＩＦＩＰ８６で発表され，世界中にきわめて大きな反響を呼んだものである．原文でお読みになりたい方は，ＩＥＥＥ　Ｃｏｍｐｕｔｅｒ誌の４月号（今年）に再録されているので，御覧になられるとよい．

５．１　それはなぜ必要か？

「銀の弾丸」とは，ドラキュラや狼男などの怪物を退治するための，必殺の武器を指す．つまり，ブルックスは，ソフトウェアという「生き物」を，ふだんはおとなしいが，ある日突然変身してわれわれに襲いかかってくる狼男にたとえているわけである．

かれは，ソフトウェア開発の仕事，特にその過程でわれわれが遭遇する困難を，古代ギリシャの哲学者アリストテレスにならって，「本質的」なものと「偶然的」なものとに区分して考える．

ソフトウェア開発における本質的なむずかしさは，複雑な構造を持つ抽象実体（データセット，データ項目間の関係，アルゴリズム，関数呼び出し等の相互関連）を記述し，設計し，その概念構成の妥当性を検証することにある．

もちろん，そうした抽象的記述を何らかのコンピュータ言語で表現し，その表現が仕様をきちんと反映しているか否かをテストするには，かなりの労力を必要とするが，それらは，しかし，あくまで偶然的な困難であって，決して本質的なものではない．

銀の弾丸とは，当然，ソフトウェア開発にともなう本質的な困難を解決してくれるような技法，ツール，環境あるいは方法論のことである．

ブルックスは，ソフトウェアが持つ複雑性，調和性，変更性，不可視性からいって，現在のソフトウェア工学技術の中からは，まだ，銀の弾丸は見出せないという分析を，ていねいに展開している．

５．２　いくつかの候補と欠陥

かれによれば，７０年代以降続けられてきた努力の大半は，偶然的困難を解消するためのものであった．以下に，銀の弾丸の候補と考えうる各種の技法・ツールに関するブルックスの分析を要約しておく：

(1) Ada その他の高級言語

特徴としては，モジュール化，抽象データ型，階層構造の導入などがあげられる．従来の言語では，方法論がカバーしていた内容を，構文に取り込んでいる．したがって，近代的なプログラミング技法を訓練するのには，有効に使えるだろう．しかし，基本的には偶然的困難の緩和にしかならない．

(2) オブジェクト指向プログラミング

抽象データ型あるいは階層的データ型の概念を，さらに一歩押し進めたのが特徴である．それは，たしかに，設計者が余分な構文上の苦労なしに自分のアイデアを表現することを可能にするが，よく考えてみると，それは，抽象的なレベルでの偶然的な困難を解決しているだけのことである．

(3) エキスパート・システム

アプリケーション独立な推論エンジンや，現実世界を正確に反映した知識ベースなど，実現性はともかく，夢を持たせるアイデアは多い．期待効果としては，アプリケーションの持つむずかしさと，プログラムの複雑さとを分離することの可能性があげられる．未熟なプログラマを，経験豊かなプログラマのレベルに近付けるのには，役立つだろう．銀色の弾丸足りうるか否かは，疑問である．

(4) 自動プログラミング

ごく少数の例外的な問題に対しては，簡単なパラメ

－タ指定によってプログラムを自動的に生成することが可能になった．しかし，汎用の自動プログラミングは，いつになったら実現するかわからない．

(5) グラフィカル・プログラミング

ビットマップ・ディスプレイの普及にともなって，さまざまな図式を用いる視覚的プログラミング・ツールの試作や研究が花盛りであるが，成果は疑わしい．ソフトウェアはハードエウェアに比較して複雑すぎるので，ＣＡＤやＶＬＳＩ設計のアナロジーは成り立ちそうにない．

(6) プログラム・ベリフィケーション

安全性の高いＯＳを開発するような，特殊ケースを除いて，一般的には役立たない．つまり，プログラムが仕様と一致しているか否かの証明以前に，仕様そのものの「デバッグ」が問題である．

(7) 環境とツール

階層ファイル，ファイル・フォーマットの統一，プログラム・インターフェースの統一，汎用ツールの使用などにより，開発の生産性や信頼性は，たしかに向上してきた．今後は，データベースの統合化により重点が置かれるべきであろう．しかし，環境やツール（あるいはワークステーション）は，あくまで補助的な手段であって，それ自体だけで，本質的な問題が解決するわけではない．

５．３　グレート・デザイナ待望論

ブルックスは，論文の最後で，将来の希望について触れ，既成ソフトの積極的活用，ラピッド・プロトタイピング，段階的開発（進化論的アプローチ）など，ソフトウェア開発の概念的本質に攻撃をかけるための，いくつかの見込みがありそうなアイデアについてのべたのち，結局は「人間」の問題がもっとも重要なので，われわれとしては，偉大な設計者になりうる可能性をもった若手技術者を，早期に発見し，かれ（またはかの女）にしかるべき知的刺激を与えて，その成長を促進したらどうか，と提言している．

設計方法論の比較・評価を行うにあたって，そういう観点から考えてみるのも，ひとつのおもしろい比較になると，講義を聞きながら感じた．つまり，単に設計作業の生産性向上だけを考えるのではなく，その方法論を利用することで，設計者がどれだけの刺激を受け，どれだけ成長するか，という視点である．

６．質疑応答

Ｑ：　本質的な困難を克服する能力は，教育あるいはテクノロジ・トランスファ可能か．当然ある程度の能力や経験が必要だと思うが，もし基本的に教育不可能なものだとすると，その代替としてどうすればよいのか．

落水：　４０人の学生がいるとすると，そのうちの１０人は偶然的困難をこなすという意味で素晴らしい能力を持っている．１０人は全く駄目で，２０人はまあまあのレベルだろう．上位１０人のうち，本質的な仕事がやれる潜在能力をそなえた人間は，１回見本をみせてやると，次からは自分でできるようになる．その意味で，選別し，伸ばすことはできるが，教えることはできないような気がする．

ソフトウェアの本質的な問題に取り組むためには，何か総合的な知的能力が必要とされるように思う．そういう学生は，毎年，せいぜい１人か２人しかいない．しかし，毎年の卒業生のうち，１５人位はちゃんとしたＳＥになっているので，割合はもう少し高いのかも知れない．

Ｑ　（このレポートでは省略したが，適用事例で扱った例に対する質問）保守時に，設計時点の判断材料を使うことに関してはどう思うか．また，新しいプログラミング・パラダイムは，具体化する言語によって変わるといわれたが，データの構造もあわせて考える必要があるのか．データ独立にはできないのか．

落水：　設計時の判断材料は残せると思う．確定的なことはいえないが，構築上の具体的な情報と打ち合わせの情報をまとめたものが必要だと思う．

新しいプログラミング・パラダイムに関しては，イメージするデータ構造の全体は同じだと思うが，そのまとめ方が異なってくると思う．私個人は，ルール指向はもうあきたので，これからは論理型かオブジェクト指向でやってみようと思っている．

ＯＡを取り巻く運営管理の諸問題

芝原 雄二
沖ソフトウェア

１．はじめに

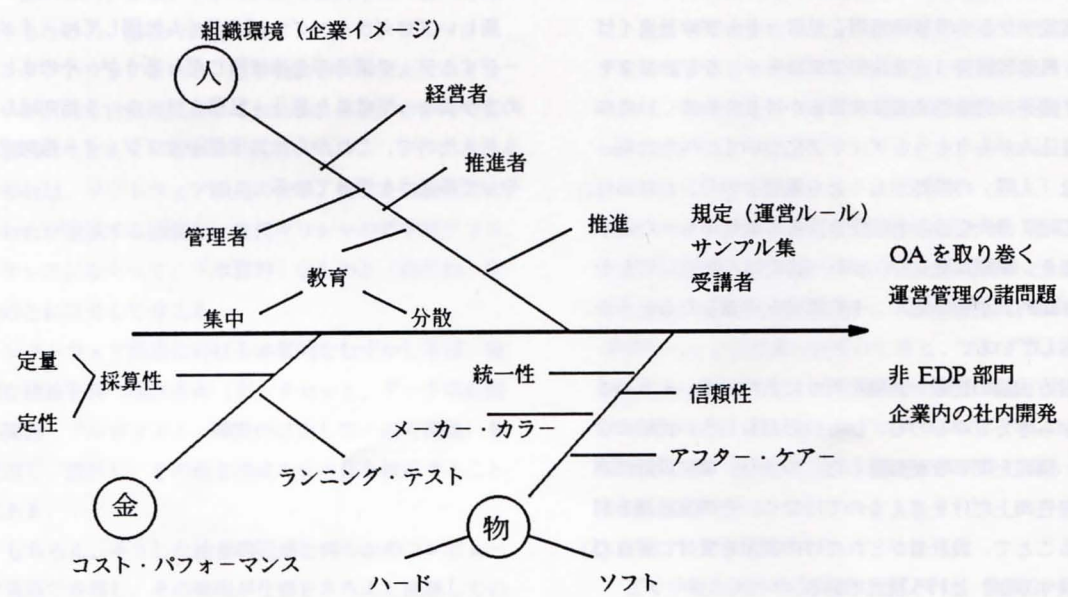
管理分科会では，毎回，メンバの１人が，コーディネータ兼スピーカとして，話題を提供することになっています．今年２月の会合では，Ｔさんが，表記のテーマを取り上げられました．

Ｔさんは，かって，会社のＥＤＰ部門にいて，オフコン等を社内に導入し業務のＯＡ化を行なう仕事を，担当されていたそうです．そうした経験を踏まえたお話をうかがい，その後，参加者全員で討論しました．

私自身は，メーカのＯＡ部門に属していますが，ふだんはあまりユーザと接触する機会が多くないので，ＯＡ利用現場の御意見がいろいろうかがえて，たいへん参考になりました．このレポートは，Ｔさんの当日のお話の要旨と，討論のポイントをまとめたものです．

２．特性要因

ＯＡの運営管理を取り巻く主な問題点を，Ｔさんは「人，物，金」の３要素の観点から分析され，さまざまな構成要因のあいだの関係を，下に示すような特性要因図で説明されました．



（１）人について

Ｔさんの経験によれば，人に関する問題のポイントは，管理者や経営者に対して，コンピュータ導入の理由やメリットをどう説明するか，ということでした．

Ｔさんの会社では，以前は，個々の部門単位で機種を選定し，部門長の主導のもとにシステム開発が行なわれていたのですが，人事移動などによって生ずる混乱を避けるため，何年か前に，ＯＡ推進を専門に担当する人間を決め，この推進者の手で，運営ルール規定やサンプル集の作成，人の教育などを，統一的・集中的に行なわれるようにしたのだそうです．

マシンの導入に関しては，現場でいろいろ検討していた結果が，上層部の鶴の一声で勝手にひっくり返され，現場を白けさせることがあるという経験談が，他の参加者から披露されました．

現場主導型でものごとを進めるには，管理者や経営者に対して，ソフトウェアを含めたシステム化の構想を，わかりやすくプレゼンテーションし，納得してもらうことが大切だというのが，みんなの討論の結論でした．

ここで，筆者が興味を持ったのは，コンピュータやＯ

Ａ機器の企業への導入が，経営者の手で企画されるのか？　それとも現場から上がってくるのか？　どちらのケースが多いのだろうか？　ということです．組織が大きくなればなるほど，経営側と現場との意識の差は大きくなるでしょうから，一概にはいえないかも知れませんが，どちらが主導権をとったほうが機械化がスムースに進行するかは，十分に比較・検討する価値がある問題ではないかと思います．

（２）金について

ＯＡ機器の導入にさいして，採算性はどれだけ真剣に考えられているのだろうか？　という単純な疑問から出発して，いろいろな議論がランダムにとびかいました．

たとえば，ワープロを例にとってみても，最近はやたらと何でもワープロで「清書」したがる傾向があり，書類の作成に要以上の時間がかかってしまう．この傾向は，特にふだん採算や経費削減に口うるさい年配者によくみられる，という若者らしい愚痴も出たりしました．

ともあれ，マシン導入時には，採算性がどうか，どれだけの合理化ができるかを明確なデータで示すようにいわれることが多いのですが，ＯＡ化・機械化によって仕事のやり方がまったく変わってしまうような新しい環境の設定を計画している場合には，こうしたデータを作ることがきわめてむずかしくなります．

したがって，そんなときには，将来もたらされるであろうメリット，他社または他部門での類似システムの導入例，現状システムの問題点や改善の方向付けなどを，素直に示したほうが効果があるのでは，ということになりました．

（３）物について

ここでの話題の中心は，マシン導入後のアフター・ケアでした．特にメーカのサービスについての議論がかなり行われました．

ユーザがマシンを導入するとき，しかもそれを全国的な規模で採用するときには，メーカ側に全国ネットワークの教育・サービス体制があることが絶対の条件になるという話が出ました．なぜなら，ユーザが自前で要員を中央に集めて教育するには，予想以上に膨大な費用がかかるからです．

技術者教育については，教育分科会やＣＡＩ分科会でいろいろ討論されているですが，これからは，こうしたエンド・ユーザ教育も，大きな問題になるのではないでしょうか？

運営に関しては，もうひとつ，マシン・トラブルの問題が取り上げられました．事務のＯＡ化が進み，機械なしではやっていけない状態に到達したあと，マシンの故障修理をどれだけ迅速にやってくれるかが重要になります．一般ユーザは，コンピュータは故障するものだという「常識」をほとんど持っていないので，何度も故障が繰り返されると，システムそのものに不信感を持ってしまうからです．

３．まとめにかえて

筆者は，最初にも触れたように，メーカのＯＡ部門にいる関係上，ともかく売ることが第一で，今回議論されたようなことがらは，日常ほとんど考えたことありませんでした．ＯＡ機器導入のメリットや先例は，われわれメーカ側で，きちんと整理し，ユーザに提供する必要があることを，今回の議論を通じて，感じさせられました．

ここにあげられたような問題は，ＯＡ機器の導入だけでなく，大型コンピュータの場合も同様です（むしろ，費用面で，はるかに高価であることからすれば，よりシビアに考えられなくてはならないと思います）．

コンピュータを作るメーカ，その上で動くシステムを作るソフトハウスの人間と，そうしたハード・ソフトを購入して利用する側の人間とのコミュニケーションが，これからは，ますます重要になってくるのではないでしょうか？

既存の会社組織の壁をこえたＳＥＡのような場に課せられた期待も，また，それだけ大きくなるように感じられます．まだまだ低空飛行中のＳＩＧＭＡＮですが，今後，より多くの方々の参加を得て，こうした境界領域の問題を積極的に取り上げて行くつもりです．よろしく．

長野支部設立報告

佐藤　千明
長野県協同電算

１．支部設立，みんなでやれば恐くない

これからの地方でのＳＥＡの活動のプロトタイプという使命を背負って（？），長野支部が５月中旬に設立されました．なぜ？　と疑問に思われる方もいるでしょうから，ここで長野が先行した理由をあげると：

－　信州という閉じた県民意識によるまとまり．
－　理屈っぽく議論好きな県民性．
－　県当局が，重厚長大型から知識集約型で軽薄短小な産業への転換を目指しており，ソフトウェアに対する過大なる期待がある．
－　東京でのイベントに毎回顔を出す経済的・時間的ロスを軽減させたいという単純な要求．
－　週末は信州に遊びに行けるということで中央からの支援があるのではないかという甘い期待．
－　本当は忙しいが，暇そうに見られている世話人．

といったところです．

２．活動方針

５月のキックオフ・ミーティングでは，活動の基本方針と，メンバー獲得作戦を協議しました．そして，芋づる式に一本釣りを展開した結果，支部会員数は２５名にふえて，なんとか体裁だけは整えることができました．

当面の活動方針は，以下の通りです：

－　毎月第１土曜日の午後に月例会を開催する．
－　テーマは，まず「身近な情報交換」．
－　場所は長野市（ときには松本市）．
－　会費は場所代とコーヒー（？）代で１０００円．
－　活動内容はＳＥＡＭＡＩＬ・ＮＡＧＡＮＯ（毎月発行）で支部会員に知らせる．
－　中央からのスピーカの押しかけは大歓迎．
－　自然環境のいい時期には合宿もやりたい．

３．第１回月例会

日時：　昭和 62年 6月13日　15：00～21：00
場所：　長野市ケイケン研修センタ
参加者：支部会員 10名　＋　新聞記者
　　（翌日の経済面に写真入りで宣伝）

議題：　設立経過報告と自己紹介

事前に各メンバの仕事の環境や関心あるテーマ，発表材料の有無等について，アンケートを集めました．そして，当日はそれをもとに自己紹介を一巡させ，課題の掘り起こしと研究テーマの絞り込みを行うというつもりだったのですが，ふたを開けて見たら，なんと最初の３人の自己紹介だけで６時間が過ぎてしまいました（この辺が，信州人の議論好きのあらわれ！）．

そこで急拠方針を変えて，当分はこの調子で各メンバの自己紹介と問題提起を続けて行い，その中から今後の研究テーマを抽出することにしました．現在の支部会員数が２５名なので，年内はスピーカの手配という厄介な仕事から逃れることができると，世話人は内心大喜び．

以下に，１回目の討論内容を整理してみましたが，これらが長野県の特殊事情を反映しているものか否かは，定かではありません：

(1) ＳＥＡＮＥＴはいつどんな形で動くのか？
・　パソコンネットも可能にして欲しい．
・　将来はＳＮＮＥＴ（ＳＥＡ・ＮＡＧＡＮＯ・ＮＥＴ）を作りたいが，今は池田屋の二階で我慢．
・　ＪＵＮＥＴサイトを支部会員に巻き込もう！

(2) 開発生産性の定量的把握がむずかしい．
・　まず標準化，機械化の推進が必要．
・　個人の能力差をどう評価したらよいか？

(3) 工数見積を勘でやっているための弊害．
・　基本設計後でなければ工数は捉えられないのに，概要設計段階で営業が受注額を決めてしまう（決めざるをえない）ため，納期や予算がオーバーする．
・　発注者側の見積もりはあてにならない．
・　設計完了後に，再度契約し直せるだけの力を業界として持つ必要がある．
・　経験したシステム開発のデータ収集と分析．
・　ファンクション・ポイント等の新しい見積り技法をカストマイズして使いこなそう．
・　建築でいう意匠設計のように，概要設計や工数見積もりを専門に行うビジネスが成立するのでは？ただし，客観性と技術に裏付けされた権威が必要．

(4) ステップ数と工数に相関関係はあるか？
- １Ｋステップ以内なら大体比例する．
- 低級言語と高級言語とでは，ステップ単価に差があって然るべきだが，実態はそうなっていない．
- 高級言語のステップ数の裏にはかなりの投資が隠されている．

(5) 部品化・再利用の遅れ．
- １回しか使われない部品が多い．つまり，モジュール化はされているが，部品化になっていない．
- 客先での開発のため自社内での部品の蓄積が進まない．
- 部品化の前に標準化をすべきだ．
- 部品化とパターン化は別個に考えるべきだ．事務処理では，ロジックのパターン化の方が有効ではないか．

(6) プログラマのキャリアパス
- 生涯１プログラマでいたい人間にも，ＳＥになれと強制するのはよくない．
- プログラマとコーダは区別されるべきだ．
- プログラマに業務知識，外部仕様記述能力は不要か？　答えはノーだろう．
- ＳＥとプログラマの分担は，ハードの世界の論理屋さんと回路屋さんに対比できる．
- 一億総プログラマ時代にプロとして生きる道は？
- 経営者は，＜ＳＥ→プログラマ→コーダ→パンチャ＞という単価のハイアラキを有効活用しようと考えている．
- ワープロは女子職員に清書させるといった発想と同じか？
- 新人にいきなりＳＥ教育をして，単価の高い仕事をさせるという発想もあるらしい．
- 経営者と技術者の観点の違いをどう調整するかが問題だ．

(7) 非メインフレームでの開発環境
- メインフレームとは世界が少し違うようだ．
- 総じてまともな仕様書やドキュメントが残っていないらしい．
- シンボリック・デバッガなしの環境なんて考えられない．
- 一方，メインフレームの世界では，シンボリック・デバッガはあまり使われていない．

(8) テスト，そしてΣ
- プログラマとは別のチェッカーがいて，コード・インスペクションやウォーク・スルーを専門に受け持つ体制が必要．
- ΣＷＳ上で，本当にメインフレーム向けのプログラム・テストができるのか？
- メインフレームのプログラムにはＤＢ／ＤＣ環境が必要であり，Σに繋がるテスト用メインフレームはその環境をオープンにしてくれるのか？

４．今後の計画

メンバの自己紹介が一巡した後は，次のようなことをしようと，いまのところ考えています：

－ 情報処理学会の「酒屋の在庫管理」みたいな共通問題を設定し，各社が実際に使っている方法論・技法・ツールを用いてプログラムを開発するコンペを開きたい．

－ そのプログラミングの環境説明も兼ねて，各社の開発現場を相互に訪問し合う．

－ 自己紹介を通じて浮き彫りにされたいくつかの問題点やテーマに関して，中央から専門家を招へいして解説してもらう．

５．おわりに

さて，何はともあれ，支部はスタートしました．ここまでは，そんなにむずかしくないのです．問題は，これからどうやって一定レベルの活動を継続して行くか，いかにして魅力ある支部に仕立て上げるか，です．

長野県内には，いま，大中小合わせて１００社を越える情報処理関連会社が存在するようですが，幸いにして，県内のソフトウェア業界をリードされている中堅技術者の方々が，多数ＳＥＡの会員になってくださいました．ということは，大袈裟に表現すれば，長野でソフトの商売をする限り，この支部の集まりに顔を出していないと，貴重な情報を取り逃がす可能性が大きいということにほかなりません．

そのうち，「ＳＥＡ－ＮＡＧＡＮＯに行けば，儲かるかどうかは別として，とにかくおもしろい話が聞けるらしい」という噂が立つよう，心がけたいと思っています．

会員の声

安西　宏

現在走っている分科会全部に関心がある．時々刻々重点も変わる．情報処理に携わる人は皆そうなのではなかろうか．全部に顔は出せないないので，結局ユーレイ会員になりそうだ．困るネ．

石井　悟

先日，３２ビット開発環境をテーマにして，日本パーソナル・コンピュータ・ソフトウェア協会主催のセミナーを企画して，開催しました．ところが，このセミナーに対して，お上からクレームがついてしまいました．セミナー自体はたいへん盛況で，アンケートの結果も好評でした．しかし，分散環境の実演に用いたマシンが，お上のお気に召さなかったようです．この分では，プログラムの使い捨てと，税金の無駄使いがまだまだ続きそうです．

伊藤　素

レベルの高い話しばかりでなく，初心者用，入門用といった話しも，ＳＥＡＭＡＩＬ誌上でたまには取り上げてほしい．ハイレベルな話しは，情報処理学会で十分すぎる位である．

［編集部注：ＳＥＡＭＡＩＬは商業雑誌でなく，会員相互の情報交換メディアです．いまのところ，編集部企画の入門講座風のページをつくる余裕は，印刷費の面でも，また，編集ボランティアの人的資源の面でも，残念ながらありません．唯一の可能性はセミナー・レポートです．セミナー委員会宛に，必要と思われる入門講座トピックスのアイデアをお寄せください］

一井生也

管理職の在り方についてもっとご意見を！！

［編集部注：ＳＩＧＭＡＮにぜひ御参加を！］

大久保功

パソコン・ネットワーク上で，諸問題のディスカッション，特に各分科会で現在行なわれているディスカッションができるといいな！　当社は，そのような社内ネットワークがあるけれど，アメリカ製で英語だから，どうもやりにくいのです．

［編集部注：ＳＩＧＮＥＴで考えてください］

小川昌邦

私は社内のＥＤＰの仕事をしていますが，自分の仕事に直接関係のない（役に立たない）話しが聞きたいものです．名古屋地区での会員の交流の場がほしい．

［編集部注：名古屋支部設立へ向けて，去る６月２６日，キックオフ・ミーティングが開かれました］

加藤　朗

ＳＥＡのイベントはことごとく１万円単位の参加費が必要で，これでは私のような学生で自腹を切って行く人が困る．領収書はいらないから，もっと安価で参加できるようにしてほしい．学生はＳＥＡにはお呼びでない，というのならそれでも結構．更新しないだけだけど

［編集部注：御要望に応えて，６月のフォーラムから，きわめて安い学生料金が設定されました］

熊谷　章

ＡＩ，分散環境，ディスクトップ・パブリッシング等，我々を囲む環境は複雑化の一途をたどっている．読者の立場に立って，「いま何が重要で何をなすべきか」というアンケートが面白いような気がする．これは編集部への提案です．意外と，多くの人々は，最近の流行のものをなにひとつ必要としていないような気がする．

［編集部注：早速実施してみたいと思います．アンケート案を作ってください．また，結果の集計と分析もよろしく］

河村一樹

最近ソフトウェア工学の限界を感じている．１９６０年代後半に叫ばれたソフトウェア危機は，あいかわらず続いている．企業内ＥＤＰ部門のバックログはたまる一方．業界全体が泥沼にのめりこんでいるような状態だ．どうもみんながプロセス指向になりすぎているのではないか．ユーザは，構造化されたプログラムよりも正しい情報のほうが結果としてほしがっているのではないか．そのためには，プロセス指向でなくデータ指向になるべ

きである．ソフトウェア工学から情報工学への転換が，これから必要とされると思われる．そうした技術的なアプローチを行なう人間を，ＳＥでなくＩＥ（Information Engineer）と名付けたい．

木村一郎

ＳＥＡＭＡＩＬには，ＳＥＡ主催，共催のフォーラムやシンポジウムの内容がかなりあり，なかなか出席できない会員としてはよろこんでいます．しかし，紙面の制約もあり全てを掲載してもらえない「うらみ」もあります（編集部にはきっと全てを掲載できない「うらみ」が残っているでしょう）．そこで，フォーラムやシンポジュムにビデオを持ち込み収録をしてはいかがでしょうか．後日，希望者に実費でダビングしてもらえるとありがたいと思います．

［編集部注：たいへんよいアイデアで思いますが，いまのセミナー委員会には，そのための余裕がありません．どなたか，ボランティアしていただけませんか？］

栗本幸男

ＤＷＢ，ＰＷＢについて一言．これらのツールがドキュメント作成，保守工数を削減し，プログラム自動生成につながる有効なものであることはわかっていても，ソフトやコード（漢字を含む）などの統一化・標準化がされていないために，なかなか導入に踏みきれない．また，標準か待ちきれず，どんどん自社流のやり方を押し進めていったあとで，ハード・ソフトの仕様標準が決定した場合に，メーカーはどこまでそうした先行投資の面倒をみてくれるのだろうか．

小柴信子

画像処理，ＣＡＤ，ＣＡＥ等についての情報があまりないようですね．ＳＩＧＣＡＤに期待しています．

小林伸二

ソフトウェアは科学か芸術か．そこが問題だ！！

佐々木義博

入社して足掛け５年目の若輩ですが，最近は「ソフトウェアの生産環境」に興味を持ちはじめました．この分野の勉強をはじめましたが，なにしろ言葉が難しくて，なかなか前進しない毎日です．この分野で面白い話しがありましたら，是非教えを請いたいと思っております．

［編集部注：ＳＩＧＥＮＶへぜひ！］

咲間　繁

Ａという仕様に対してプログラムａを作成し，Ｂという仕様に対してプログラムｂを作成しているのが現状である．早くこの手作り的な工程から脱出し，大量生産を図りたい．その方法は？

鈴木裕信

最近思っていることは「すべては個人の問題意識に還元される」ということです．ツールがどうのＯＳがどうのなんて，実は小さいことなんじゃないかな？

曽根浩士

８６年１２月に，ＳＥＡに加入しましたが，静岡県の加入者が少ないのには驚きました．長野支部の記事は大変参考になりました．静岡のソフトウェア産業の発展のためにも，静岡支部をぜひつくりたいものです．

［編集部注：がんばってください．まずは，身近の会員拡大から！］

田中康和

現在，色々なシンポジウムが大都会でたくさん開催されています．私は参加したいのですが，福岡という地方都市に住んでいるために，なかなか参加できないのが現状です．たまには福岡地区でも開催してもらえたら，九州の人々にとってプラスになるのではないでしょうか？

［編集部注：イベントの地方開催を実現する上では，やはり採算が問題です．つまり，いくらの参加費でどのくらいの人数が集められるか，適当なスポンサーをみつけられるか，といったことですが，そのへんを調べて事務局宛に具体的な提案をお寄せいただければ，セミナー委員会で検討します］

高尾　猛

ＯＡの普及・拡大が進み，非ＥＤＰ部門の職場にもいろいろな機器が設置され，利用者の関心度合も年々高まっています．そこで，さまざまな問題点，失敗／成功例等を，エンド・ユーザ（非ＥＤＰ部門）活用事例分科会（仮称）を作って交流できるようにしたら，新しい層の会員も増えるのではないでしょうか？　非ＥＤＰ部門の

人々からみた場合，ＳＥＡに入会しづらいということがあるようです．専門家集団についていけないと感じる人も多いように思われます．専門家と一般利用者とのあいだの相互理解を深める必要はないでしょうか？

［編集部注：これ以外にも，先日のアンケートで，こんな分科会が欲しいという御希望が，いくつか寄せられました．ＳＥＡは「草の根」型の組織ですから，分科会は，すべて，いいだした人が「この指止まれ」方式でスタートさせるルールになっています．具体的企画を立てた上で，幹事会に提案するか，ＳＥＡＭＡＩＬ誌上で直接会員に呼びかけてください（Ｖｏｌ２．Ｎｏ２掲載のＳＩＧＤＯＣ設立の呼びかけ参照）．もし，あなたにそれだけのエネルギーがないというのであれば，いつの日か，だれかが自分の希望をかなえてくれるのを待つ以外にありません］

田村　整

地方会員という表現に少し抵抗はあるが，自分としては東京から地方（？）に移った者には，ＳＥＡＭＡＩＬは唯一の中央（？）とのパイプであることはカクセナイことである．また，自分のソフトウェア環境が大きく変化した（ＵＮＩＸ系からＩＢＭ大型に）者にとっては，世界の動き（情報）がわからないので，船便であってもとても必要な情報源である．

辻　英夫

ＳＥＡＭＡＩＬの記事の内容がアカデミックすぎてすこし違和感があります．技術者の労務問題，作業環境，キャリアパスなどもとりあげていただきたい

［編集部注：ＳＥＡＭＡＩＬの記事は，基本的に会員からの自発的投稿で成り立っており，セミナーやイベントの記録以外に，編集部企画の記事はありません．もし，御自分の期待と違うようでしたら，期待される内容の記事を増やすために，まず，そのきっかけとなる原稿をお寄せください］

中小路久美代

これまで，自分が計算機に関わっていたにもかかわらず，ソフトウェア業界というと，なんとなく関係ないという目でみていた私ですが，ＳＥＡ会員になって半年，今までこの偏見で随分損をしていたことに気がつきました．

野中　哲

私は，マイクロ・コンピュータで装置制御をおこなうソフトの開発をやっていますが，我々のような者の立場からみると，ＯＳや言語処理のようなソフトをやっている人は，たいへん高級な仕事をしているように思えてしまうのです．数ミリ秒という制限のもとでは，プログラムの美しさなどは，あまり追求できません．

主にＹ－ＤＯＣに出席していますが，ＬＩＳＰとかＳＭＡＬＬＴＡＬＫとかで飯を食べている人がけっこう多いのにおどろきました．自分の仕事に全く関係のないいろいろな人と会えるのが，ＳＥＡの楽しみのひとつです．

個人的には，労働時間，タイムカード，フレックス・タイム，残業，休暇，給与，評価，在宅勤務，転職など，広い意味での作業環境について議論したいと思います．

野々下幸治

以前，池沢忠男さんから，年間の文献をまとめてくれるとことはありませんか，というメッセージがありましたが，ＪＯＩＳ（日本科学技術情報センターのオンライン・データベース）を利用すると便利です．特に特定のテーマに関して，継続的に調査がしたい場合は，ＳＤＩ検索を利用すれば，最新から２－３ヶ月程遅れますが，定期的に情報を手に入れることができます．月々のお金は使用しなければとられません．無料で講習会も開いていますから，一度行ってみてはいかかでしょうか．

藤原利晴

いま，私の会社に待望のＮＥＷＳワークステーションが導入され，ＦＲＡＮＺ　ＬＩＳＰをゴチョゴチョいじっているのですが，マニュアルが全部横文字で悪戦苦闘しています．何かいい本（方法）があれば教えて下さい．

細野広水

地方の会員としては，都会（？）で開かれるＳＥＡの各種の会に参加するのが，やや困難であるが，それは単に個人の都合である．会が地方で開かれたら，東京・大阪の方は困るでしょう？　それよりも，毎月（？）のＳＥＡＭＡＩＬに感謝，感謝．船便でも定期であれば文句なし．

藤原千尋

まだ駆け出しの私は，将来のために先輩方の意見を聞

き，日進月歩のこの世界で目標を見失わないために入会しました．当然行事への参加や原稿を送ることは無理な現状ですが，今後もよろしくお願いします．

水谷哲也

要求定義，概念設計の段階で，いかにうまくエンド・ユーザの要望を引き出すか，またエンド・ユーザ自身のもやもやとしたシステム・イメージを固めることができるかが，その後のシステム開発の経費（期間・コスト）短縮の一番大きな要因になるのではと感じています．インタビュー技術等に関するセミナーは開催されないのでしょうか？

［編集部注：どんな講師のどんな話が聞きたいのですか？　事務局宛に具体的なリクエストをどうぞ］

宮内興治

いろいろなイベントがありますが，平日の昼間に行なわれているものが多いように思います．なるべく，１９：００以降にして頂くと参加しやすくなるのではないでしょうか？

［編集部注：分科会はそうなっています．セミナーやフォーラムは，会員サービスの他に，一般への活動ＰＲ，および会の運営のための資金調達を兼ねていますので，どうしても通常の時間帯にならざるを得ません．ぜひ，会社の教育予算を使って，堂々と出席してください］

水野誠一

ＳＥＡ内部連絡用に，パソコン・ネットワークを利用する予定はないのでしょうか？　すでに運用しているネットワークの一部を間借りするとかすれば，技術的な問題もないと考えますが．ＳＥＡ会員の方々ならば，かなり多くの方々がこのようなネットワークを利用できると思うのですが．それとも，パソコン・ネットワークの普及率はまだまだ低いのでしょうか？

［編集部注：ぜひ，ネットワーク分科会に参加して，具体化を図ってください］

村瀬訓雄

データフロー・モデルを利用したシステム設計支援エキスパートシステムを設計中です．よいアイデアがあったらお聞かせください．

柳田俊一

東京以外の地で，東京と同等レベルの仕事をこなしていくためにはどのようにしていったらよいのかを，真剣に考えています．このテーマについての誌上討論会をぜひともお願いします．東京で仕事をしている人からの上記テーマについてのアドバイスも期待しています．ただし，ばくぜんとした討論では意味がないので，テーマを次のようにしぼってみたらいかがでしょうか？

（１）必要とされる環境
（２）東京に出先機関を設けることは必要か？
（３）東京のＳＥＡ会員の東京に対する満足・不満足度
（４）東京との情報伝達ルート運用のノウハウ

［編集部注：前述の熊谷さんの提案と同様ですが，ぜひ，具体的にアンケートの質問をデザインしてください．そして，回答の分析もよろしく］

山崎　桂

ＪＵＮＥＴを用いて，ＳＥＡのニュース・グループを作ったらどうですか？

［編集部注：これもまたＳＩＧＮＥＴへ］

山城英彦

ハードな環境で作業を行なっている技術者が多いと思いますが，個人のライフ・スタイルをどう守っていくべきか悩んでいる人もいると思います．そこで，このようなテーマを取り上げて誌上討論をされたらいかがでしょう．管理上のポイントとなる項目が浮き上がってくるのではないでしょうか．

［編集部注：ＳＥＡは労働組合ではないので，取り上げ方がむずかしいと思いますが，プロフェッショナル・エンジニアの自己管理は，たしかに重要なテーマです．具体的にどんなテーマでみなさんの原稿を集めたらいいでしょうか？　そのきっかけになる一文をお寄せいただけますか？］

第5回

夏のプログラミング・ワークショップへのお誘い

実行委員長
野 村 行 憲
岩手電子計算センター

1．はじめに

第5回目を迎える夏のプログラミング・ワークショップ（略称：若手の会 in 盛岡）が，今年は『バクのないソフトウェアの実現を目指して』をテーマとして，例年どおり夏の終わり（9月2日～5日）に開催されます．

ワークショップの環境として，すなわち，エネルギッシュな若手プログラマたちが，日頃の業務を離れ，何かひとつの問題について，じっくりと議論を闘わせる場所としては，わがホーム・タウン盛岡こそ，もっともふさわしいと考えます．さわやかな空気，静かな町並み，おいしい食べ物，そして甘露なる酒・・・これだけそろえば，あとは集まるメンバーの質だけが問題です．

みなさんも，ソフトウェア界の宮沢賢治や石川啄木になったつもりで，9月の虫に心を馳せてみませんか！

といっても，残念ながら，ＳＥＡの会員の平均年齢は，今回のワークショップの参加制限条件をやや（？）オーバーしています．あなたが，もし，無資格である場合は，どうぞ身近にいる若い方々に，この案内文と参加案内パンフレット（３６ページ）を至急回覧してください．

2．若手の会の歴史

若手の会のこれまでの輝かしい実績をご存じでない方のために，少しその足跡を紹介しておきます．

このワークショップは，毎年ほぼ同じ時期，同じ場所で過去4年間開催され，今年で5回目を迎えます．第1回から第4回までの開催年月，基調テーマ，チェアマンと参加人数は，次の通りでした：

第1回（５８年8月）
　特定テーマなし
　　岸田孝一　　４７名
第2回（５９年8月）
　もしも私がプロジェクト・リーダになったら
　　野村敏次　　５０名
第3回（６０年9月）
　これからのソフトウェア開発を考える
　　三浦信之　　６１名
第4回（６１年9月）
　プロジェクト・プランニングのケース・スタディ
　　堀江　進　　４１名

3．今年の企画の特徴

今年は，今までと趣を異にして，このワークショップのＯＢ（つまり３０代のエンジニア）だけで運営スタッフを編成しました．それは，参加者との世代ギャップがほとんどないことを意味します．しかも，それぞれソフトウェア・エンジニアとして，いろいろな分野の第一線で活躍中の強者ぞろいです．ですから，討論が，いままで以上に盛り上がるのではないかと期待されます．

討論の基調テーマとして，今年は，プログラマにとってもっとも基本的な問題である「バク」の問題をとりあげました．バクについてはすでに語りつくされたと思っている方が多いなかで，あえてこの問題を取り上げたのは，新しいプログラミング言語・パラダイム・ツール・環境などが次々に登場し，虫の棲息する環境も様変わりしてきた今こそ，あらためてバグについて考え直す時期だと思ったからです．

4．予備的考察

（1）その分類

さて，ここでバグについて少し考えてみることにしましょう．バグはこの世にコンピュータが登場したのと同時に生まれ，以来，プログラマとのよき（？）隣人またはライバルとして，影のようにつきまとってきたことは，いまさらいうまでもありません．

英語での bug という単語は「昆虫」という意味のほかに，もともと「故障」とか「欠点」などの意味があり，本来こちらの意味で使われていると思いますが，日本で

は「虫」と訳されていますので，その名にこだわって普通の昆虫と比較してみたいと思います．

昆虫	バグ
よく移動する	ほとんど移動しない
役に立つものもいる	百害あって一利なし
五分の魂有り	自我なし
殺虫剤で死ぬ	特効薬なし
単独で生きている	必ず何かに寄生する
生態解明が進んでいる	生態が掴めていない

（2）バグの誕生？

いうまでもなく，バグは生まれて来るものではなく，プログラマ自身が作り出し，プログラム中に埋め込んでいるものにほかなりません．でも，どうして作り込んだプログラマ自身が，そのことに気がつかないのでしょうか？　このあたりにも解決の糸口がありそうです．

（3）バグはいつ作られる？

バグの作り込まれるタイミングとしては，主として設計時とコーディング時の２つが考えられます．それぞれの時点での対策は？　レビューの効果と限界は？

（4）害虫駆除

デバックの方法にはいろいろありますが，それぞれの長所，短所は？　たとえば，開発用のマシンと実行用のマシンが違う場合の効果的なデバックのやり方は？

（5）虫のないプログラム！

虫が棲んでいないプログラムは存在するだろうか？　また，動く機会がほとんど来ないような虫がいたとしても，それは虫のないプログラムと呼んでいいか？

（6）網戸

プログラムの作成時にバグが入り込まないようにするための方策として，簡単なものでは，ロジック・スケルトンの利用や既存モジュールの再利用などが，比較的多く利用されているのではないでしょうか．また，ランゲージ・オリエンテッドなプログラム・エディタの利用も，シンタクス・レベルのバグには効果があります．これらの「網戸」の目の大きさは一体どれくらいのものでしょうか？

（7）それぞれの分野で

オブジェクト・オリエンテッドなプログラミングと従来の機能オリエンテッドなプログラミングの場合では，デバッグ上の問題点はどう変わるか？また，ＡＩを利用したプログラミングの自動化やデバッギング・ツールなども，興味深いテーマでしょう．さらに，ＡＩプログラム自身のデバッグや保守をどうするかも，おもしろそうです．

５．まとめ

このようにいろいろ考えてみると，話題がどんどん発散して，議論のまとまりがつかなくなってきそうですが，参加される方々のポジション・ペーパを拝見してから，ある程度の方向性を定めて，多角的で実りの多い討論が行えるようなプログラムを作りたいと考えています．

ワークショップのフォーマットについて，全員でのパネルとグループ討論との時間的バランスをどう取っていくか，招待講演のテーマや時間配分をどうするかなどをあれこれ考えていると，頭がだんだんバギーになってバグ・ハウスにお邪魔したくなりそうですが，各プログラム委員にご協力をいただいて，今まで以上の成功を達成できることは，ほぼまちがいないでしょう．

６．始まりに向かって

ポジション・ペーパの締め切りも，７月２７日に迫ってきました．３０歳未満のバリバリ・エンジニアたちからの自由で過激なポジション・ペーパを，スタッフ一同楽しみにしています．例年約２倍以上の狭き門ですので，くれぐれも遅れることのないように....

それでは９月の盛岡のさわやかな空気の中でお会いしましょう．

Call for Position Papers

第 5 回　夏のプログラミング・ワークショップ

「バグのないソフトウェア」の実現を目指して

主催：ソフトウェア技術者協会（SEA）

後援：（社）情報サービス産業協会（JISA）

第５回目を迎える恒例の「若手の会　in　盛岡」です．今年は，過去にこのワークショップに参加したＯＢ（つまり３０代の現役エンジニア）だけで運営スタッフを構成し，ソフトウェア開発の現状が抱える問題点について，より生々しい議論を展開したいと考えています．

設定された討論テーマは，「広い意味でのバグのないソフトウェアはどうすれば実現できるか？」というマクロな視点から，現状を多角的にとらえることを意図したものです．具体的な論点としては，たとえば：

・保守の困難さやバックログの増加をどう解消して行くか？
・バグの発生を防ぐために開発環境やツールをどう工夫しているか？
・ＡＩやその他の新パラダイムはどのように役立つだろうか？
・要員の資質や教育訓練は製品の品質とどう関わってくるか？
・プロダクトあるいはプロジェクト・マネジメントのあり方は？
・ソフトウェアの保護あるいはセキュリティとバグとの関係は？

など，いろいろ考えられるでしょう．

これまでの仕事の中での経験を踏まえた意見や提言を，なるべく具体的な形でポジション・ペーパにまとめてください．多くの若手技術者からの積極的な御応募をお待ちしています．

開　催　要　領

１．期日：　昭和 62年 9月 2日（水）午後から 5日（土）正午まで（3泊 4日）

２．会場：　ホテル・ニューカリーナ（岩手県・盛岡市）

３．定員：　25 名

４．参加資格：　年齢 **30歳未満**，経験 **2年以上**のソフトウェア技術者

５．費用：　65,000円（ SEA会員，JISA会員），　75,000円（一般）

現地集合・現地解散とし，期間中の宿泊費（朝食付き）および資料代を含みます．

６．スタッフ：

実行委員長：　野村行憲（岩手電子計算センター）
プログラム委員長：　藤野晃延（富士ゼロックス情報システム）
プログラム委員：　沢田寿美（ＳＲＡ）　月岡義範（中央システム）
松下和隆（三菱電気東部コンピュータ）　村井進（テクニカル・ライタ）　他

７．申込方法：　上述の要領でまとめたポジション・ペーパ（A　4判　2～　5枚）を　5部コピーの上，下記の申込票を添えて，**昭和 62年 7月 27日（月）必着**までに，プログラム委員長宛お送り下さい．申込多数の場合（例年定員の　2～　3倍の応募があります）には，プログラム委員会でポジション・ペーパの内容を審査の上，　8月初旬には参加者を決定し，結果を全員に御連絡します．

ポジション・ペーパ送付先：

〒160　東京都新宿区西新宿 3 － 16 － 6 西新宿水野ビル
富士ゼロックス情報システム（株）
技術部　　藤　野　晃　延
Tel: 03 － 378 － 8010　Fax: 03 － 378 － 7298

.....

第 5回　夏のプログラミング・ワークショップ　参加申込票

氏　名：＿＿＿＿＿＿＿＿＿＿＿＿（年齢：＿＿＿＿ 歳，性別：男　女）　申込種別（SEA　JISA　一般）

会社名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿　所属：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

住　所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

TEL：　＿＿＿＿＿－＿＿＿＿＿－＿＿＿＿＿＿＿＿＿

ＳＥＡ会員状況（昭和 62 年 6 月 30 日現在）

正会員	1032名（5月1日から134名増）
賛助会員	27社（5月1日から0社増）

＜正会員の勤務地および居住地域分布＞

	勤務地域	居住地域
北海道＝	6	6
山形　＝	1	1
宮城　＝	2	2
岩手　＝	4	4
福島　＝	1	1
新潟　＝	6	6
栃木　＝	6	5
群馬　＝	1	1
茨城　＝	6	8
埼玉　＝	11	67
千葉　＝	14	68
東京　＝	620	377
神奈川＝	78	200
長野　＝	21	21
富山　＝	3	3
石川　＝	2	2
静岡　＝	12	10
岐阜　＝	2	5
愛知　＝	23	19
和歌山＝	1	0
三重　＝	1	1
滋賀　＝	2	3
京都　＝	11	16
大阪　＝	110	87
奈良　＝	2	6
兵庫　＝	25	39
愛媛　＝	1	1
徳島　＝	3	3
岡山　＝	1	1
広島　＝	4	4
福岡　＝	10	10
熊本　＝	12	12
大分　＝	4	4
佐賀　＝	1	1
宮崎　＝	1	1
鹿児島＝	2	2
沖縄　＝	1	1

＜男女分布＞

男　＝	974
女　＝	57

＜年齢分布＞

20以下　＝	0
20_24　＝	45
25_29　＝	232
30_34　＝	282
35_39　＝	250
40_44　＝	131
45_49　＝	51
50_54　＝	17
55_59　＝	10
60以上　＝	7

＜血液型分布＞

A型　＝	413
O型　＝	289
B型　＝	224
AB型＝	106

＜賛助会員会社名＞

ＩＮ情報センター，ＳＢＣソフトウェア，エムテイシー，コンピュータサービス，サンビルト印刷，ジェーエムエーシステムズ，セントラル・コンピュータ・サービス，ソニー，ソフトウェア・リサーチ・アソシエイツ，ニッポンダイナミックシステムズ，マイクロキャビン，パナファコム，リクルート，リコーシステム開発，近畿日本ツーリスト，構造計画研究所，神戸コンピューターサービス，経調，辻システム計画事務所，東電ソフトウェア，日進ソフトウェア，日本システム，日本システムサイエンス，日本能率コンサルタント，日立製作所，富士ゼロックス情報システム，富士通，富士通ビジネスシステム

（アイウエオ順）

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　FAX.03-318-3909