SEAMAIL

Monthly Newsletter from
Software Engineers Association

Volume 1, Number 12 | December 1986

目　次

編集部から		1
SEA 情報化月間記念フォーラムから－その２－		
パネル討論「ニュー・テクノロジー・インパクト」	深瀬・柿下・熊谷・二木	2
セミナー 21 レポート－その３－		
設計方法論の最前線	落水浩一郎／渡辺雄一	12
ソフトウェアの法的保護	佐野稔／田中慎一郎	16
大型ビジネスアプリケーションの開発環境	盛田政敏／関崎邦夫	20
プログラミング言語雑感	引地　信之	23
中国旅行記－連載その３－	野村　敏次	26
幹事会，分科会および支部活動		30
幹事会　ＡＩ分科会　管理分科会　教育分科会　再利用分科会		
法的保護分科会　環境分科会　横浜支部　関西支部		
'86 データショウ見聞録	匿名子	32
9th ICSE 予告		33
カレンダー		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　皆藤慎一　木村高志　久保宏志　熊谷章　斎藤信男　三枝守正　芝原雄二　杉田義明　辻淳二　鳥居宏次　中園順三　針谷明　藤野晃延　松本崇純　松原友夫　水谷時雄　三浦信之　村井進

会計監事：　近藤秀朗　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)：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)：安倍昭敬　坂下秀　白井豊　高田佳彦　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)：鈴木弘

法的保護分科会(SIGSPL)：能登末之

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

ＳＥＡＭＡＩＬ編集グループ：大槻亮人　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．１２　　昭和６１年１２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

鈍行列車便で，今年最後のＳＥＡＭＡＩＬをお届けします．前号が船便でなかったときに，ある会員から，編集部の言い訳を読むたのしみがなくなったと文句がでたので，わざとらしい言い訳を少し：

本当はこの号は，クリスマス前に発行する予定だったのです．ところが，編集長が，３週間のアメリカ出張のあと，最終校正を終えるひまもなく（年のせいか，疲れてそれどころではなかった），またしても１０日間の海外出張に出掛けてしまったのです．

なんとか年内発行に間に合わせようと，向こうのホテルでも赤ペン片手に酒を飲み，帰国後は毎朝５時起きでがんばりましたので，ほめてやってください．印刷屋さんをだましすかしてくださったＯさんにも，感謝！

お正月にゆっくり読んでいただこうというわけで，今回は全部自宅あてに発送させていただくことにしました．

情報化月間記念フォーラム

ＳＥＡ情報化月間記念フォーラムの報告第２弾として，１０月２日の午前中に行われたパネルを，編集部でレポートしました．最初は４－５ページに圧縮するつもりだったのですが，内容が内容だけに，思い切って全部のせることにしました．

セミナー・ウィーク特集

今回は，Ｂ３，Ｅ３，Ｇ２の３セッションを掲載しました．これで，全２１セッションのうち，約半分の１２セッションを消化したことになります．

なお，Ｅ３セッションのレポートにはいっている「絵」（１４ページ）は，Ｍａｃ＆Ｐｉｃ（前月号参照）で作成したものです．お絵書きの所要時間は，約３時間でした（！？）．この程度の労力は編集部で十分提供できますので，これからは絵入りの原稿もどしどしお寄せください．

会員からの投稿

野村さんの「中国旅行記」は，今回で完結しました．引地さんの原稿は，ＳＩＧＥＮＶ世話人としてのものではなく，個人的なエッセイです．同じような経験を持つ会員の方々からの反応をお待ちします．また，匿名での投稿（データショー見聞記）を今回初めて掲載しました．ユーモアとしてお読み下さい．

幹事会から

幹事会の報告にもありますように，協会の財政確保の一貫と，会員への情報伝達を兼ねて，各種宣伝用メイルの発送代行を事務局が主体になって行うことにしました．セミナーその他の宣伝媒体として，どうかご活用ください．

第９回ＩＣＳＥについて

３３－３６ページの案内にあるように，来年３月末から４月初めにかけて，第９回のＩＣＳＥが開催されます．わがＳＥＡは，日本を代表する協賛団体であり，したがって，会員に対しては，参加費が特別割引になります．グループ・ツアーをいま企画中（案内は１月初め）ですので，ふるって御参加ください．

なお，今回の論文審査はきわめて狭き門（応募約２００篇中，採択されたのはわずか２５篇）でした．日本からは３篇の論文が採択されましたが，うち２人はＳＥＡ会員です（電総研の二木さんと日本電子計算の野村さん，おめでとうございました！）．

また，パネリストやセッション・チェアマンとして，会員のなかから，岸田（ＳＲＡ），斎藤（慶大），寺本（日電），横井（ＩＣＯＴ）の各氏が参加される予定です．

さあ，来年！

船便だ，合併号だといわれながら，なんとか月刊ペースを１年間死守することができました．来年は，ページ数も若干増やして，さらに頑張りたいと思っています．

１月号からは，表紙の色がブルーに変わります．前年度の日本シリーズを制覇したチームの色をというのが，創刊のさいの内規になっていました（これを決めた編集長は，ひそかに阪神の２連覇を願っていたようですが，そうは問屋がおろしませんでした）．

今月の標語

年末年始は，ＳＥＡＭＡＩＬを読んで原稿を書こう！

SEA情報化記念月間フォーラム
パネル討論

ニュー・テクノロジー・インパクト

コーディネータ：岸田　孝一
パネリスト　：深瀬　弘恭
：熊谷　章
：柿下　尚武
：二木　厚吉

1．はじめに

司会：　このところハード／ソフト両面で，技術的に面白そうな話題がちらほらしていて，９０年代にはどうなるのかという期待があります．しかし，その一方では，昨日の佐和先生のお話にもあったように，日本経済はすでに曲がり角を曲がってしまって，これから世紀末に向かって，われわれをとりまく経済的環境は暗くなる一方だという，悲観的な予言がある．

だとすれば，いろいろな新技術は，われわれの世紀末暗黒探検にとって，懐中電灯やスコップの役割を果たすにちがいない．それでは，いったいどんな道具立てがあって，それがプログラマの今後のライフスタイルにどんな影響を与えるのかを，４人のパネリストの方々と一緒に考えてみようというのが，このパネルの狙いです．

パネリストを順番に御紹介すると，まず，アスキーの深瀬さんは日本のＵＮＩＸコミュニティを代表する中年ハッカーとして有名ですが，経歴を拝見すると，原子力計算からバンキング・システム，汎用機のＯＳ，さらにはパソコン・ソフトまで，「無節操」にいろいろ手懸けて来られました．

次が，カスタム・テクノロジーの柿下さん．もともとは学者をめざしていたそうですが，いまはＣＡＤ／ＣＡＭやＡＩ関係のコンサルテーションを中心にお仕事をなさっています．

３人目はパナファコムの熊谷さんですが，ＬＩＳＰ，ＰＲＯＬＯＧ，ＳＭＡＬＬＴＡＬＫを同時並行的に使うという狂気のプログラミング環境を実際に開発してしまったという恐ろしい方です．

以上３人の血液型は，司会の私を含めて全員Ｂ型であり，なにをいいだすかわかりません（笑）．そこで，最後の砦として，唯一の非Ｂ型人間である電総研の二木さんには，フォーマル・スペシフィケーションという御専門を生かして，それまでにいろいろ出たインフォーマルな意見を，ややフォーマルにまとめていただこうと思います．

では，まずは深瀬さんから．

2．ソフトを作らない技術の重要性

深瀬：　ニュー・テクノロジーとは何かを考える前に，今までがどうだったのかを振り返ってみたいと思います．典型的な例として，１０年単位で進んでいるバンキング・オンライン・システムが，第１次から３次までの間にどれだけ変わったかを考えてみましょう．

ハードに対する投資額は，１０年ごとに３倍づつ増えています．第３次では総額３兆円，１行あたり７百～１千億の投資が予定されています．一方，ソフトの投資額の増加はそれをはるかに上回っており，もしこのあと第４次オンを開発するとすれば，ハードとソフトにほぼ同じ額の投資をしなければならないだろう，といわれています．

ではその間に，ＣＰＵの能力やソフトウェアの価格がどのくらい変わったかといえば，まずＣＰＵの能力は，１０年間に８倍，３年で２倍のペースで伸びて来ました．一方，ソフトの価格は扱い方が難しいのですが，同規模のソフトを作るのにいくらかかるかということで見ると，１０年間で約２倍になっています．たとえば，アプリケーション・プログラマの１人月あたりの値段は，いまスキルの高い人で１２０万位ですが，１０年前だと６０万まではいかなっかたと思います．

さて，それでは，これからの１０年はどうなるのでしょう．ＣＰＵの能力がさらに８倍になり，ソフトウェアの開発に対して今の何倍もの投資が行えるのでしょうか．

１９９５年に，ユーザは果たしてそれだけの負担に耐えられるのでしょうか．

これは，何とか手を打たなければならない．どうにかして既存のソフトウェアの再利用をはかろうとか，それを支える道具立てを用意するために，国家プロジェクトで環境開発をしようといった発想が，そのあたりから出てくるわけですが，そんなことでうまくいくのだろうかという素朴な疑問が，私にはあります．

この問題に対するアプローチは大きく分けると２つになるでしょう．

１つは，ソフトウェアの生産性を向上させようというものです．これは，ある意味で質よりも量を重視する考え方だと思いますが，数年後にはソフトウェア・エンジニアの数が何十万人か足りなくなる．だからＳＩＧＭＡプロジェクトなるものが必要だ，というお役所的なロジックは，これにあたります．

その場合，本当にそれだけの人手をかけて，そんなに膨大なソフトウェアを開発するニーズがあるのだろうかということの分析は，あまり真剣に行われないようです．ところが実際には，世の中では，本当は不必要なソフトウェアもずいぶん作られている．

だから，単純に量的な生産性向上だけを考えると，ごみくずのようなソフトを効率的に作りだし，そのメインテナンスに多くの人がさかれるという結果に終わってしまう危険があります．

もう１つのアプローチは，なるべくソフトを作らないようにするというものです．作らなければメインテナンスは発生しないし（笑），ソフトウェアの価格競争もおきません．なるべく不要なソフトは作らないで，必要なものだけをなるべく高品質で提供することを考えたほうがいい．

私は，個人的には，この「ソフトはなるべく作るな・使うな」という考え方が正しいと思っています．もともと質の悪いソフトにいくら手を入れても，お金ばかりかかって，結局何にもならない．いいコンセプトにもとづく良質のソフトウェアを世の中に供給し，みんなで守り育てていくにはどうしたらいいかを考えていきたいと思っています．

それを実現する上で重要なことの１つは，分散処理技術の確立です．先程お話したように，ハードウェアのコストはどんどん下がっていますが，それは相対的な話で，ソフトの肥大化に対しては，ハードの側でもそれなりに対処せざるをえないわけです．たとえば，ＣＰＵの能力は１桁のオーダーでしか増えていかないのですが，メモリーやディスクの容量は１０の２～３乗のオーダーで増えていく．

となれば，どうしてもＣＰＵパワーが足りなくなるのは目に見えている．したがって，単純な算術の論理からいって，しかるべき分散処理技術を確立しておかない限りは，それぞれが勝手に動くようなソフトウェアのあいだにはさまれて，われわれ自身が苦労をせざるをえない．

もう１つのやり方は，できあいのソフト（パッケージ）に，自分の仕事のスタイルをあわせてしまうことです．みんながこれを大胆におしすすめると，ソフトウェアの相対的な価格は，どんどん安くなるはずです．みんなで使えばやすくなる．

かなり深いスキルを持った人に対しても，なにも知らないユーザに対しても，まったく同じソフトで押し通す．もちろん，ユーザ・インタフェースに細工をして目先を変えることは必要ですが，ソフトウェアの骨格自体は変えないですませる．このことをまず実行しないと，どんなに多額の開発予算を持っていても，結局は，量に圧倒されてしまうでしょう．

現在，国家プロジェクトとしてシグマ計画が進められていますが，これまでの様子をみると，参加各メーカが独自につくるワークステーションをばらまいて，そのうえでソフト作りの仕事をする方向をめざしているようです．こうなると，しかし，各社は，それぞれのワークステーションの上に独自のソフトウェア・ツールをのせて他社と競争しようとするでしょうから，なるべくソフトウェアを作らないようにということ私の主張とは，まったく逆の方向に進んでいきます．

それぞれのメーカーが，また，たくさんの新しいソフトを作って商売をしていこうと考えている．ソフトウェア危機が大声で叫ばれているわりには，新しいソフトウェア産業を支えるべき人々が，きわめて認識不足であるように感じられます．

３．でも，世界はソフトを求めている

司会： それでは次に，柿下さん．

柿下： 現在の技術状況をひとことでいえば，ソフトのハード化がますます進みつつあるということでしょう．われわれが作るソフトがそのまま，新しいハードの機能仕様になりうる時代が近づいています．

そうした時代の方向性を一連のキイワードで表現すれば，ハードウェアは「ネットワーク化・ＶＬＳＩ化・パーソナル化・超高性能化・非フォンノイマン化」の方向へ向かっており，一方，ソフトウェアは「巨大化・広範囲化・統合化・パッケージ化・カスタム化・ユーザフレンドリ化」の方向へ向かっている，といってよいでしょう．

そして，私の専門分野であるＣＡＤ/ＣＡＭについていえば，それは，明らかに「統合化・専門化・低価格化・パッケージ化・カスタム化」に向かいつつあります．

つまり，現状のＣＡＤシステムは，汎用メインフレームの上にカスタム・メイドのソフトをのせ，専門のオペレータが操作するマル金のシステムから，簡単なお絵かきソフトと簡易３Ｄグラフィクス・パッケージをパソコンにのせ，素人でも手軽に使えるようなマルビのシステムまで，多極分化していますが，いずれについても共通にいえることは，きわめて使い勝手が悪く，また，ＣＡＥ－＞ＣＡＤ－＞ＣＡＭ－＞ＦＡという全体の作業プロセスの部分的な所しか支援してくれないということです．

しかし，最近のハードウェア技術の進歩は，こうした状況の限界を簡単に飛びこえて，われわれが本当に作りたいもの，使いたいものを，高性能の専用ワークステーションという形で実現することを可能にしてくれるかのように思われます．

で，もしそうした夢のＣＡＤ／ＣＡＭ支援環境を作ったとしますと，作ってはみたものの，どう使っていいのかわからない（笑い）という事態にきっと陥るだろうと予想されます．これまで自分が慣れ親しんでいた環境との落差にとまどってしまい，古い文化的感性を拭いさるまでに，かなりの時間がかかるでしょう．

それは，汎用ＴＳＳのプログラマを，いきなりビットマップ・ワークステーションの前に座らせて，マウスをにぎらせるようなものです．

そこで，昔の計算機用に作ったソフトウェアというのがいろいろいあって，そのソフトウェアを捨てるわけにはいかないから，それがそのまま動くように新しいハードを作るということになるかもしれません．

革新的なニュー・ハードウェアを考えるにせよ，漸進的な改革を狙うにせよ，いずれにしても，ソフトウェアの要求を土台にしてハードを作るということは変わりません．これは，大きな時代の転換といってよいでしょう．

そこで，ソフトウェアとは何かということになりますが，プログラムはデータがないと動かない．これからは，データの重要性が強調されてくると思います．では，そのデータ（もっとナウイいい方をすれば「知識ベース」ということになりますが）をだれが作り，だれがどう使っていくのかとが問題です．

ソフトウェア技術といったときに，流れがはっきり２つに分かれてくると思います．

１つは，決定論的なプログラミングです．これは，伝統的なソフトウェア・エンジニアリングの世界にあたり，今後ますます専門化・高度化が進でしょう．もう１つは，オープンで経験論的なプログラミングの世界で，これは，非プログラマのための，特定領域における問題解決知識の定式化に相当します．知識ベースやＡＩ技術を応用して，状況に応じて，どんどん変えていきやすいような形にする．

この２つの技術の流れは，いずれ新しい統合的ＡＩ言語の登場によって１つにまとまるのかも知れませんが，当分の間は，はっきり区別して考えなければならないでしょう．

そして，特に後者の世界では，ナレジ・エンジニアとかいわれる人など，今までのタイプと違う技術者を巻き込んでいかないと，絶対に駒が足りないといった状況がおきるのではないかと予想されます．

ソフトウェアを作るという仕事は，実にたくさんの言葉を駆使しないと達成できない．今までプログラミングというと，プログラミング言語の世界にいればよかったのですが，アプリケーション向きの言葉をしゃべらないと，だんだんプログラムができないようになってきています．

実は，研究室から出て，現場のソフト作りの世界にはいると，つくづく，「ソフトは人なり」ということばを実感させられます．そこでは，修羅場をくぐった回数というものが重要です（笑い），

これは冗談ではなくって，そういう人間は，いざというときパニックにならないですね．実際にチームのキーマンになれる人というのは，聞いてみるとよく遊んでいる．例えば，プラモデルを作ったりして，「これは仕様がはっきりしてるし，目にみえるからいい」なぁーんちゃって（笑い）．したがって，いいソフトウェア技術者を育てるにはどうすればいいかというと，遊ぶときには，しっかり遊ばしておくことが，意外に大切なのではと思います．

何かまとまりのない話になりましたが，このへんでひとまず終わっておきます．

４．社会の変化と技術の対応

司会： では次に，熊谷さんお願いします．

熊谷： これまでのコンピュータの発展を振り返ってみると，社会の変化に対応して，機械のイメージが変わってきていることがわかります．そういう歴史的展望に立って考えると，いま，コンピュータは，まさに質的な転換期にさしかかっているといえるでしょう．現在，コンピュータ産業，印刷産業，そして映像産業の３つが，それぞれお互いに分離独立した形で存在していますが，西暦２０００年位までには，これらが１つの産業として融合してしまことが予想されるからです．

社会環境の具体的な変化のあらわれとしては，コミニュケーションの形態が変わってきます．たとえば，紙や電話が電子メールにおきかわるといったことです．マシンの利用形態は，当然パーソナル・ユース，つまりかなり高性能な計算機を１人１台以上専有するかたちになる．

それで，いったい何が起こるかといえば，人間は大体わがままな動物ですから，「自分のしたいようにしたい，規制されるのはいやだ，自分にとって快適な環境が欲しい，機械が自分に合わせてくれる，そして，ときには自分の悪い癖を計算機に注意してもらいたい」といった要求を持ち出してくる．マン・マシン・コミュニケイションの媒体は当然マルチ・メディアにならざるをえない．

こうした変化によって，計算機も社会の動きとともに変わってきています．６０年代は汎用機の時代，７０年代における話題の中心はミニコンで，８０年代の前半はパソコンがそれにとってかわりました．これからはワークステーションが大きくクローズアップされてくる．

このように時代が変わっていくときに，依然としてＣＯＢＯＬやＦＯＲＴＲＡＮやＣを使っていたのでは，多分ダメなんですね（笑い）．

オブジェクト指向，動的な記号処理，フレーム表現等の新しい概念を取り入れたソフトウェア開発の新しいパラダイムが必要だと思います．先程いった３つの業界がどんどん一緒になっていく過程で，そうした新しいパラダイムが必ず出てくる．

ところで，オブジェクト指向，動的な記号処理，フレーム表現等を全部ひっくるめたシステムを，いま具体化しようとすると，私のやっているＬＩＰＳでもそうなんですが，まだコンピュータに知性が足りないとか，データベースが使いにくいとかいう問題があらわれて，やりたいことがなかなかできない．新しいパラダイムを作り出すには，ソフトだけではダメで，ハードウェアも一緒に作ることが必要なように思われます．

ところで，３つの業界が融合したあとのワークステーションについて考えるときには，大きくわけて，社会的な機能と個人的な機能の両面を見てゆく必要がありますが，考え方の基礎としては，「人間性中心，不易実行，マイノリティ重視」という３つのことが重要だと思います．

たとえば，いま設計者とコーダとを分離するといったことが多くのいわゆるソフトウェア工場で行われていますが，人間性からいえば，これはおかしなことで，プロトタイピングとかの新しい技法を使いながら，１人の人間がまとまった仕事を全部やるようにしなくてはダメなんじゃないでしょうか．

もともと，エンド・ユーザが思っていることを，たとえそれがどんなにマイナーな要求でも，ユーザ自身が計算機を使って実現できないから問題なので，それが可能になれば，プログラマはいらなくなる．つまり，現場のユーザが自分でマシンを使えるような環境を，われわれソフトウェア技術者が提供できないことに問題がある．

ソフトウェア・エンジニアリングのこれまでの目標は，単なる量的な生産性向上だったようですが，これからは，本を書くあるいは絵を描くといった知的活動を質的に支援する方向に向かわざるをえないようなきがする．そこで，ソフトウェア・エンジニアの仕事はどうなるかといえば，いま，ハードウェア，ソフトウェア，そしてエンド・ユーザの間には深い溝があって，それをみんなが嘆いているのですが，この溝を埋めることである．

そうした「量から質への変革」という観点にたって考えると，ソフトウェア技術を単純な技（わざ）の世界から，より深みを持った，いわゆるひとつの「アート」に変貌させることが必要になってくると思います．

プログラマのライフ・スタイルについていえば，技術文化には２つのタイプがある．１つは「自転車置き場型」の技術で，これは，だれでも作ろうと思えば作れる．そういったものを作る技術は１つのカルチャを形成しています．ところが，もう一方には，「アポロ打ち上げ型」の技術文化がある．これは，とにかくやってみなければわからない世界です．この２つを混同してソフトウェ

ア・エンジニアリングを論ずるからメチャクチャになる（笑い）．

前者つまり自転車置き場の世界では，生産性とか品質管理とかの概念が重要であり，ライフサイクルを通じての体系的あるいは標準的な技法が必要になる．各種の開発支援ツールを統合化したワークベンチによる自動化のパラダイムが近未来のイメージです．私個人は，あまりそうした日常的な世界には興味がない（笑い）．

では，後者のアポロ打ち上げの世界では，どういったことが重要視されるかといえば，ここでの最大の課題は，「創造と表現」だと思います．もう１つは，実験またはプロトタイピングの技術が必要で，それには当然検証の技術が対になっていますが，それを使いこなすためには，「非常識な常識」を持っていなければ，ダメだと思います．私自身がそうだとはいいませんが（笑い），ある種の常識と教養がなければ，プロトタイプの検証などできません．

５．研究世界からの展望

司会： では最後に二木さん，まとめをよろしくお願いします（笑い）．

二木： 私のここでの立場は，前のみなさんが実際に物をつくっている立場だとすれば，それを横目で見ているかたちになります．リサーチ・コミュニティという，いわば岡目八目の視点から見て，これからどういう技術があらわれてくるかをお話させていただきます．

前の３人の方々からのお話からもわかるように，ソフトウェアの話は総合的なものですから，単純にこれからの技術はこうなって，これが主流になるだろうというハッピーエンドのストーリーは，研究者の立場からも描けません．１９９０年代のソフトウェア工学をにらんで，いま，リサーチ・サイドでどんなものが芽生えており，実際にどんなものが生き残っていく見込みあるのか，またはないのかについての，きわめて個人的な予想としてお聞きください．

ソフトウェア工学というのは，非常に幅広い学問ですが，そこで対象にしているのは主としてインダストリです．つまり，イメージとしては，これから大規模なインダストリが成り立つような基礎技術を，従来の工学的センスでとらえようとしている．ですから，先程熊谷さんがお話しされた，きわめて人間的な話題とかアートうんぬんの議論は，いくらかそういう気配も見えてきてはいますが，まだ，研究コミュニティのなかでは，はっきりとらえられてはいないように思えます．

話の組立て方はいろいろ考えられますが，ここでは，（１）ソフトウェアの再利用，（２）人工知能とソフトウェア工学，（３）新しいプログラミング・パラダイム，という３つのことがらについて，私のまわりで起こっていることをお話します．未来予測に結論はありえませんので，結論は期待しないでください．

まず，ソフトウェアの再利用ですが，これは非常にわかりやすい，しかし，技術的には実に含蓄の深い概念であって，１９９０年ごろまでは，このことだけを考えていけば，技術の主軸としてはよいのではないかとも思われます．現在行われている再利用研究を眺めてみても，そのスペクトラムは非常に広く，いろいろな技術要素をカバーしています．

たとえば，ソフトウェア・ライブラリ，ソフトウェア・データベース，さらには知識ベース，プログラム変換，アプリケーション・ジェネレータまでを含めて再利用という言葉を使っています．ここまでを概念の拡大解釈を行えば，１９９０年どころでじゃなく，２０００年まで持ちこたえられるかも知れません．

再利用のレベルはいろいろあって，先程深瀬さんのいわれたのは，いわゆる気配りレベルでの再利用だと思いますが，さらにはモジュール構造やアルゴリズムの構造などの構文レベルの再利用も十分実用化の段階に達しています．また，プログラム・トランスフォーメーションやアプリケーション・ジェネレータを含めた言語・意味論レベルの再利用も，いま一歩のところまできていると思います．

次に，人工知能（ＡＩ）とソフトウェア工学との関連ですが，これはここ数年，どうなるかということで話題を集めていることは御承知の通りです．そもそもＡｉ技術が生まれて来た背景を考えると，それは，人間が新しい問題にぶちあたったて，それを解くために，計算機という道具をどう使ったらよいかに関する研究なのですから，別のいい方をすれば，「自動プログラミング」とほぼ同義語だといっていい．人工知能がめざしているのは，ある意味で理想的な自動プログラミング環境だともいえるわけです．

ところが，この自動プログラミングということばは，いろいろな背景を背負ってきているので，使い方がむずかしい．一昔前には，ＦＯＲＴＲＡＮコンパイラが自動

プログラミングだといわれていたこともありました．逆に最近では，第５世代あたりの宣伝が効き過ぎて，ＡＩ万能論が素人に行き渡っている．そこで，例のパルナスのＳＤＩ論文のように，そうした傾向に対する批判もあらわれてきた．

しかし，本当のところは，ＡＩとソフトウェア工学とは，うまく手をたずさえて進歩して行くことが望ましいし，また実際にそうなってゆくと思います．ＡＩ研究者たちがＬＩＳＰプログラミング環境のなかでつちかってきたいろいろな技法（動的メモリ管理とか，プロトタイピングとか）は，今日ソフトウェア工学の世界でも，ちゃんとその重要性が認識されていますし，これからもそうした技術移転はどんどん行われるでしょう．

最後に新しいプログラミング・パラダイムについてですが，これも話題としてとりあげられるようになったのは１９７０年代後半からですから，どうも今という時期は，技術的にいって１つの転換期であるという気がしますね．プログラミングだけに限らず，コンピュータに関する技術全体の基本骨格の見直しが必要だということではないでしょうか．

これからのパラダイムの移行は，大きくいって次の３つの流れにに分けられるように思います．

・命令型から宣言型へ
・手続き指向からオブジェクト指向へ
・逐次集中型から並行分散型へ

たしかに，熊谷さんがいわれたように，その背景には，社会状況の変化にともなう技術の変貌があり，その流れはもはや止めようがないでしょう．

司会の岸田さんから期待されたような，きちんとしたまとめにはなっていませんが，一応，私のプレゼンテーションはこんなところで．．．

司会： どうもありがとうございました．それではこれから自由討論にはいります．会場にマイクが２本立っていますので，アメリカ式に，何かいいたい方は順番にマイクの前に並んでいただくというスタイルで進めていきたいと思います．御自由に，どうぞ．

６．再利用について

Ｑ： ソフトウェア再利用のレベルについてのお話がありましたが，もう少し詳しく説明していただけますか．

二木： 先程いったように，再利用は非常にスペクトラムの広い概念ですから，どのレベルでやるのかというのが大事です．

気配りレベルというのは，研究というより会社などの再利用促進本部とかにとっての話で，つまり，組織として再利用をやる意志を持つか否かという問題です．深瀬さんがいったように，ソフトウェア・プログラムをみだりに作らない，という覚悟が必要なレベルです．それでするとしないでは，随分違います．日常的な問題点は，このレベルでかなり解決するのではないかと思います．

構文レベルの再利用とは，具体的にいえば，モジュール・レベルの再利用，つまりすでにあるソフトウェア資産を再利用することです．ビルディング・ブロックが何であるのかをアイデンティファイして，そのビルディング・ブロックの使い方を，いろいろ技術的に詰めていくということです．それをさらに押し進めれば，ロジック・パターンの再利用にまでゆきつくでしょう．

言語・意味論レベルの再利用というのは，平坦ないいかたをすれば，ジェネレータを再利用という立場でとらえることになります．プログラムをなるべく作らないようにするというのは，たしかに重要なことで，つまり，かなり信頼性の高い既存のプログラム部品をいかに有効に使うか，それをジェネレータを使って展開してゆく．

いくらソフトウェアを作らないといっても，末端では必要なソフトウェアの量は増えると思いますので，その作成をジェネレータで自動的に支援する，絶対に安心できる部品やロジックを自動的に取り込むという形のソフトウェア再利用です．

その根底には，同じような仕事や，似たような仕事は２度とやりたくないという意志があると思いますが，そのかわり最初のものは一生懸命作る．ここで多少余分なお金がかかってもいいという考え方です．このようにして，ソフトウェアのビルディング・ブロックを考えていくときに，オブジェクト指向のプログラミング的言語を使うと，技術的な見通しがいいということについては，すでにコンセンサスが得られていると思います．

深瀬： そういう意味で，実際に役立っている再利用技術としては，ＵＮＩＸのシェル・プログラミングやパイプの概念があります．ＵＮＩＸがもてはやされている大きな理由の１つは，過去の人々が作った資産をうまく利用できるからです．

研究コミュニティでの最先端の技術からすれば，もうとっくに次世代のＯＳを開発できなければいけないし，また，それは十分に可能なわけですが，にもかかわらず，

いまだにＵＮＩＸが主流で使われているというのは，単純に気配りレベルをこえた再利用のサポートがあるということであり，

これは産業界の立場から考えても，きわめて強力だといえます．ですから，１９９０年ごろまでは，ネットワークさえ整備されれば，万々歳ではないかと思います．

熊谷： 言語レベルの再利用についてですが，ここでもオブジェクト指向の考え方が非常に重要な概念でして，いろいろなパラダイムのなかでも，オブジェクト指向が一番効果的だといえます．他のものは技術的な見地からすれば重要な概念をいくつか含んでいますけれど，効果を考えたときには，オブジェクト指向だと思います．Ｓｍａｌｌｔａｌｋは，システム自体がＳｍａｌｌｔａｌｋ自身で書かれていて，ウィンドウベースのシステムが，あのように簡単に整備できているということが，一番すごいですね．

７．アプリケーション・ジェネレータ

Ｑ： アプリケーション・ジェネレータの今後について，お考えを聞かせてください．

二木： これはいま，いろいろなレベルのものが試作され，実験されています．アプリケーションのドメインに依存した部分が多いので，一般論はいいにくいのですが，実用的な技法が除々に明らかになりつつあります．次の時代の研究レベルでは，プログラム・トランスフォーメーションが，重要なテーマではないかと思います．

プログラミングを一般論でやろうとしてもダメだということは，１９７０年代の中頃からいわれていました．一般的にプログラムというものはこうすれば作れるというアプローチで具体的なアプリケーションの問題が解けるかというと，どうしても無理なんですね．

それぞれの分野ごとに問題解決に必要な知識を拾い集めて，ちゃんと整理しない限りむずかしい．プログラミングというのは問題の解決であり，問題というのは非常に具体的なもので，一般的な解法というのはありません．そのことが，研究者たちの間ではっきり認識されたのは，１９７０年後半からです．

それ以降，知識ベース・プログラミングやドメイン指向型のパラダイムという概念がでてきたわけです．これは，ある意味で，それぞれのアプリケーション・ドメインごとにプログラミングのエキスパートを作ろうという試みだといえます．

これまでに，かなりいろいろなものが作られていて，有効性が実証されつつあります．いまのところまだ研究室の中だけの話ですが，そろそろ実際の物作りに役立つところまで近づいたようです．

Ｑ： 設計プロセスを何らかの形でモデル化できたとすると，これはプログラミング過程全部のモデル化になります．つまり，人間がプログラミングをするという仕事を，人間サイドに立って深く研究し，どういうプロセスで何をやっているのか明らかにする．それがいわゆる認知科学のアプローチです．これは，先程熊谷さんがいわれたアートに近いところで，はたしてものになるかどうかはわかりませんが，非常に重要なテーマだと思うのですが．．．

二木： 本当の意味のＡＩというのは，人間を研究する学問です．人間がどういうことをやっているのか，ソフトウェアを作る段階で，人間がどうプロセスを経て，どういうメンタリティのもとに，いろいろなデシジョン・メイキングを行っているのかということを理解することが，最終目的ですが，いまのところ手掛かりすら摑めていない（笑い）．

認知科学といわれる学問は非常に重要な分野でして，われわれの研究所でも１つの流れになりつつありますが，気をつけないといけないのは，すこしばかり研究が進歩したからといって，すぐに実際の役に立つというものではないということです．

８．ＡＩの実用性について

Ｑ： 最近気になっているのですが，ＡＩ技術が非常に役に立つもの，すぐにも実用化されるものと一般には考えられているようですが，私はＡＩというのは，本当の意味の基礎科学じゃないか，人文科学までを含めた統合的な基礎科学で，役に立つか立たないかということからすれば，一番役に立たない学問の代表じゃないかと思います．そのへんをはっきりと区別しなければいけないのではないでしょうか．

ソフトウェア工学とかコンピュータ・サイエンスというのは，比較的決定論的な色彩が強い．一方，ＡＩは人間の学問ですから，経験論的だといえます．

二木： たしかに，御指摘の通りだと思います．Ｄ．パルナスも，ＳＤＩ批判の論文の中で，同じようなことを述べています．ＳＤＩシステムは，人間がこれまでに作ったことがない膨大なソフトウェアになり，現在のソフ

トウェア工学のレベルでは開発できない．ＳＤＩ推進論者たちは，ＡＩ的なテクニック，自動プログラミングを使えば開発は可能だといっているが，たとえそれらの技法を使ったとしても，あれだけの複雑性を持ったリアルタイム・ソフトは開発できるはずはないし，かえって信頼性が悪くなるだろうというのが，その趣旨です．

ところで，これはあまり注目された論文ではないのですが，Ｈ．サイモンが，第７回ＩＣＳＥの基調講演のなかで，ソフトウェア工学はもっと積極的にＡＩ技術を取り入れるべきだと述べています．そのころはまだＳＤＩなんてものはありませんでしたが，いまこそソフトウェア・エンジニアリングとＡＩが仲良くして，大規模なプロジェクトを推進すべきだ，と（笑い）．

どうも，学界の両巨頭の意見がまったく対立しているように，表面的には見える．困ったことです（笑い）．おそらく，結論は両者の中間あたりにあるのでしょう．

私個人の意見としては，非常に信頼性の高いプログラムを作るのに，現在のＡＩテクニックに全面的によりかかるのはかなり危険だというパルナスの見解には，かなりの説得力があると思います．ＡＩの目的は，極論すれば，人間の仕事を効果的に助けることにすぎません．ＡＩを万能視する考え方は，かなり危険だといえるでしょう．

９．ネットワークをめぐって

Ｑ： ネットワークがもたらすわれわれ技術者の仕事へのインパクトについて，みなさんにお聞きしたいと思います．ネットワーク社会では，プログラマのライフスタイルはどうなっていくのか，たとえば，田舎で地酒を飲みながら仕事ができるとか（笑い），そうしたことがが何年ぐらいで可能になるのか，といったことです．

深瀬： アメリカには，コンピュータ・サイエンスに関するネットワークがＡＲＰＡＮＥＴ，ＣＳＮＥＴ，ＵＳＥＮＥＴ等複数ありますが，これらのネットワークを使って，技術者や研究者がどういう仕事をしているかといいますと，たとえば，日本では学会に論文を書いて送ると，採用・落選などの手続きを経て，最後に電話帳みたいな論文集ができ上がる（笑い）．アメリカの場合には，ネットワークに対してペーパーを書く，それが正式の論文として認められるというような違いがあります．

こうした環境の違いというのはきわめて大きいものでして，ＡＲＰＮＥＴにつながっていないような企業には，就職はしたくないという学生も大勢いる．ネットワーク上でなんでも議論できるというのが一番大きいですね．なにかわからないことがあると，ネットワークに質問を出すと，誰かが答えてくれる．

日本では社内にだれか知っている人を探し出すだけでもたいへんですから，その差は大きいとおもいますね．

二木： アメリカで研究活動を成功させたかったら，ＡＲＰＡＮＥＴに情報を収集し，成果を宣伝するのが一番だといわれていますが，それは冗談でなくて，本当にその通りです．研究活動だけでなく，ソフトウェア開発でいう概念設計みたいなアイデア収集のプロセスにおいて，ＡＲＰＡやＣＳなどのネットワークが使えるか否かは，決定的な差を生みだしますね．

日本の大学，研究所でもそういったものを作りたいという動きはあります．通産省でもインタオペラティーブ・データベース・システムという大型プロジェクトが走っていまして，額面通りだとそれが終わる５～６年後には，そしたネットワークができるのでは，と思われます．

で，それでどうなるかといいますと，ネットワークというのは，技術的というより政治的社会的な問題がすごくありますから，質問の中であったような会社の外にいて仕事をするということは，いまでもやろうと思えばできる．それが社会的にもう少し認められやすくなる．といっても，だれでもできるようになるのは，なかなか大変だと思います．

柿下： いま２人が話されていたのは，ネットワークを使う人たちの知識なり情報なりを共有するというものですが，もう一方には，ひとつのプロジェクトを運用していくうえでのネットワークの利用ということもあると思います．そうした技術はまだ確立されていないのですから，ライフスタイルがどうなるかということより，俺はこうしたいんだ，ということをはっきりさせることのほうが，いまは大切なのではないでしょうか．

司会： ネットワークの政治的な問題ということについては，ＪＵＮＥＴやパソコンネットに関係している深瀬さん，いかがですか．

深瀬： ひとつは，企業内のセキュリティの問題が，最大のネックになると思いますね．ネットワークに対して情報を出すとき，だれが責任をとるのかということです．しかし，外から入ってくる情報というのは，いつでも受け入れる（笑い）．この落差が非常に困るわけです．

たとえば，ＪＵＮＥＴには２００以上のサイトがつな

がっています．半数以上はプライベート・カンパニーですが，ここらからは（とくに大企業のサイトからは）ほとんどメッセージが流れ出してこない．じゃあ，情報を全然みていないのかというと，全部読んでいる（笑い）．アメリカではそういうことはない．日本の場合は非常に閉鎖的ですね．

こうした文化のちがいが非常に大きな問題で，ネットワークで仕事ができるかできないかの分かれ目になるんじゃないかと思います．

司会： ネトワークが国際化，いまも少しされてきていますが，そうなったときに何らかの摩擦を引き起こすのではないでしょうか．

深瀬： すでになりつつありますね．ＪＵＮＥＴには，ＡＲＰＮＥＴを通して相当数の情報が，１ヶ月にして２０Ｍバイト位の情報が流れてきているんですが，日本からは一向に何もいってこないと文句をいわれています．アメリカと日本のソフトウェアのマーケットの大きさは，１０：１なんですが，情報のやりとりに関しては，１０００：１位です．日本の会社というのは非常に管理がいきとどいているので（笑い），それが一番の政治的問題です．

熊谷： ＵＮＩＸを電話でつないで協力会社の人たちに使ってもらったことがあるんですが，ともかく料金が高すぎる．外国にくらべて，ＮＴＴは勇気があるというか料金が高いですね（笑い）．これが最大のネックだと思います．

もう１つは，メーカーとか官庁には，常識人がいないんですね（笑い）．常識で考えればやらなくてはならないことが，常識がないばかりにできない．３番目には，ワークステーションの使い勝手のいいものを作らないとダメだと思います．

１０．再びＡＩと再利用について

Q： 二木さんのお話の中で，パルナスのことが出てきましたが，私自身ＡＩに興味をもっていますので，それに関連して質問したいと思います．パルナスの論文では，彼自身ＡＩに対して否定的な見解でしたが，ソフトウェア工学もかなりの成果をあげてきていますけれど，それで全部が解決されるということはないのではと思います．

いまのソフトウェア・マーケットを見ると，ビジネス・アプリケーションの需要が非常に大きいわけで，そこらへんでより効率的な生産していくとときに，ソフトウェア工学だけで対応できるのだろうか．もしできないとすれば，何らかの形でＡＩの応用はできないだろうか，どんなふうに応用していけばいいのか，といったことについて，具体的にみなさんにお話していただければと思います．

熊谷： 基本的には，ビジネス・アプリケーションの世界には，ＡＩ技術が応用できると考えます．具体的には，マンマシン・インタフェースの部分ですね．人間というのは，みんな好き勝手にやっていますので，業界ごとに住んでいる世界が違うと，そこで使うボキャブラリーも違っていますから，それをＡＩ風のマンマシン・インタフェイスで吸収するといったことが考えられます．

椿下： 信頼性という言葉がＡＩとしっくりいくのか，という問題提起がありましが，ＡＩ技術がそのままシステムのなかに入り込んでも，信頼性が上がるとは思えません．これは私のＡＩというものに対する考え方なんですが，機械はいったいどこがいいのかといえば，人間にできないことをやってくれるからいいのであって，人間ができることを機械がやってくれたからいって，それがどうしたの，ということがありますね（笑い）．

深瀬： みなさんＡＩの応用ということを考えておられるようですが，うちの会社はやりません．どうしてかといいますと，ビジネスとして対象にしていない．

世の中でＡｉ，ＡＩと騒いでいますと，その実体は別にして，企業のトップは使いたがるんですね．そこで，たとえば何とかいうエキスパート構築システムを持ってきて，それでシステムが作れるかといいますと，できない．最後に，実際のプログラムはＬＩＳＰでゴリゴリ書くようになる．どこがＡＩなのか（笑い）．

旧来の開発となにも変わらないですね．そうやっていくつかのソフトが実用化されていますが，一番いい例は，為替のビリング・システムですね．で，このシステムでもそうですが，どこが一番むずかしいかというと，知識ベースのところでして，それは現場でやっている連中にはかなわない．ですから，ビジネスとしてソフトウェア会社がやっても，デモンストレーション・パッケージにしかならないですね．

Q： いまの深瀬さんのお言葉を聞くと，ソフトウェア会社でＡＩで商売しようとしている身にとっては，反省させられるところもありますが，もうすこし泥臭い話をうかがいたいと思います．現場のほうがよく知っているとよくいわれますが，現場の知識がどれだけ役に立つの

か．

再利用を図っていく上で，再利用すべき対象物の抽出にはどういうアプローチをとったらいいのでしょうか．ＡＩでいう知識というのは，私は手続き型だと思います．極論すると，宣言型のデータは知識ではないと思うんですね．人間の頭の中で考えている知識は，かならず手続きです．手続きでないと困るわけです．

これからのソフトウェア技術者が，たとえ，飯のたねにありつくために（笑い）ＬＩＳＰやＰｒｏｌｏｇを使うとしても，やっぱり手続きの発想でプログラムを組んでいくことでしょう．そうなると，いまの知識のモデルというのは，どうもおかしいんじゃないか．宣言的な知識がどうの，フレームがどうのと騒いでいますが，そうじゃなくて，どちらかというとスクリプト技法の流れで押さえるほうが，知識ということでは主流ではないか，と考えます．

深瀬： 再利用ということでいえば，現場が一番知っているとは申し上げてません．知識ベースがどこにあるかについて，ＡＩがらみのアプリケーションを作るときにどうするかという問題をお話したつもりです．

旧来のソフトウェアの再利用ということを考えると，一番可能性の高いのは，ＣＯＢＯＬプログラムの再利用だと思います．あるフリーのプログラマに聞きますと，ＣＯＢＯＬというのは適用業務が事務処理と決まっていて，プログラムのパターンは２０数種しかない．お客から話しを聞きながら，それをちょっとずつ変えていくだけですむ，といっていました．それらのパターンを整理して，典型的なものを作ってしまうことのほうが，早道だと思います．

二木： 再利用に関しては，分野にディペンドしたものがやりやすいということはあります．問題領域がどういうものなのかに，大幅に依存する．領域がむずかしかったり，広かったりしたら，どうしょうもないですね．

ところで，プログラミング上の知識が手続きだというのはまさにその通りで，宣言的という表現は，手続きでないということを強調するためにそういっているだけです．最終的には計算機で動くのですから，本質的には手続き的です．

熊谷： 再利用の議論は，いつも社内でやっています．新しい製品を出すとき，前の製品で動いていたソフトをそのまま動かせないか，ということですね．そのさいのノウハウは何かといえば，これはもう決断しかないですね．やるという決断をしたら，やるっきゃない（笑い）．デシジョンをどう下すかで，すべての運命は決まってしまいます．

さて，知識表現についてですが，人間はいろいろ独自な表現方法を持っていますね．人が考えてものごとをやる背景には，１つの思想がある．表向きの形は少しずつ違うのですが，その奥にある意味は同じじゃないかと思います．同じようなことをやろうとしても，各自が自分の思っているイメージをより的確に表現できる表現形態をとっていくと思うんです．

したがって，私は，各自がそれぞれ自分に適切な表現方法をとって，それで共通の会話ができればいいと思うのですね．

そういう意味では，宣言がいいとか，いやフレームだスクリプトだという議論ではなく，もっと奥にあるものを本当にわかるような形で，コンピュータの上で表現することが重要なのだというような気がします．

Q： 世の中の言語を非常にシンプルに分けると，名詞と動詞のふたつになりますね（笑い）．名詞のほうは宣言型でやってもいいのですが，動的なところに関しては，経験的に手続き的にまとめて書いていく方がいい．ソフトウェア工学には，そうしたまとめ方の方法論があると思いますが，一方，ＡＩにはなぜ方法論がないのか．いっそ，ないならないで安心するのですが（笑い）．

二木： 本当にＡＩを好きでやっている人たちの動機は，そこに方法論がないからだという，うがった見方も一部にはあります（笑い）．

司会： いつものことながら，議論がおもしろくなってくると，時間ぎれになってしまいます．パネリストのみなさん，どうもお疲れさまでした（拍手）．

［文責：編集部］

SEA秋のセミナー・ウィーク
セッション E3

設計方法論の最前線

講 師： 落水 浩一郎
報告者： 渡辺 雄一

1．はじめに

９月号の予告にあったように，このセッションの前半では，講師の落水先生が大学院の講義を利用して読まれたいくつかの論文〔１〕を紹介され，種々のプログラミング・パラダイムの特徴や適用分野について解説された．そして，後半はそれに基づく自由討論が行われたが，このレポートでは，前半の要約とそれに続く自由討論をとりあげた．

2．論文の紹介

セッションの半分以上の時間が，この論文紹介に費やされた．その詳細をここに示すことは限られた誌面の関係でできないので，特に指摘されたポイントをかいつまんで紹介する．ここで話された目的・主旨は，言語の話ではなく，それら言語と実際のソフトウェアの開発との間にあるシステムの構造を作りあげてゆく手段・方法の最近の動向をサベイすることである．

2．1 Conventional Approach：

問題，要求の定義，システムの構造の解析設計，インプリメント，テストを行うという従来からの工程を踏むことを前提として，そのどこか一部あるいは全体の方式を改善し，精度を上げるエキスパート・システムの一部の成功に刺激されて，それらをシステム作りにも応用しようとする動き．また既存の方法論に対する反省，計算機そのものの変化に対する新たな方法論．

(a) マンマシン・インタフェースのシミュレーション

図書館の図書貸出しシステムを例にして，Ｕｎｉｘ上でのＲＡＰＩＤ／ＵＳＥを使い，ユーザ・インタフェースをいかに早く確定させるか，また改善してゆく〔２〕．

(b) 問題構造とソフトウェア構造の対応の良さ

問題世界のなかで，データフロー等を使って問題の構造を意識したときに，そこからモジュールの呼び出し関係に変換してゆくのではなく，オブジェクトという活性なデータを導入して，他とのデータのやりとり（メッセージのやりとり）という形でシステムを設計し，Ａｄａの持っている機能を利用して展開してゆく〔３〕．

落水先生のコメントとして，『オブジェクトの取捨選択は何に左右されるか，というとインプリメントする言語に依存する．Ｓｍａｌｌｔａｌｋ８０ならば既存のクラスをどのように使うか，ということでオブジェクトを選ぶし，Ａｄａならパッケージ機能やタスク機能に変換しやすいようなオブジェクトの認識の仕方をする．計算機の最終的な動作を記述しようとしている物（言語）によって，現実世界の認識の仕方が変わってくる』．

(c)並列処理システムの解析

人間ではなかなか分析しにくい，デッドロック等の並行システムの動作解析のときに，また普段直感的にやっているところを，フォーマルな形で定義して，計算機に解析を行わせて，そこから有用な情報を得る．設計記法には依存しないでそれができる〔４〕．

2．2 Operational Approch：

問題世界を表現するような言語（ないしは表現）をつくり〔実行可能仕様〕，それに対するインタプリタ等も作り，そこでテストを吸収しながら，全体の仕様を固めて，そこからコードを作ってしまう〔自動変換〕．

第３世代言語を使った設計方法論，またその上流に位置する管理工程に対する問題意識から，従来の工程を圧縮する形のプロトタイピング手法や，単に言語だけではなく，その周りの環境の進歩も含めて操作的アプローチを行うのだが，実行可能仕様をいかに記述するか，ということが問題になっている．

(a) 関数仕様の効果とプロトタイピング〔５,６〕
(b) データフロー図式〔７〕
(c) ペトリネット〔８〕
(d) Entity - Action モデル〔９〕

3．落水先生の感想，指摘

これらの新しい方法論にたいして，落水先生は以下のような指摘をされている：

「これら一連の論文を読み始めたときの印象は，一番

最初にシステムを作り始めるときに，システムが引き起す“動作”というものを，ユーザの世界に近いレベルや出来上がるシステムに近いレベルで，イメージしようとする試みが，積極的になされようとしている．

かつての方法論のもとでのドキュメンテーションは（ＪＣＬは除いて），ほとんど役に立たない．モジュール構造図が書いてあったり，インタフェースの説明が書いてあっても，そのシステムがどう動くのか，動いたときにどういう状況が起こって，どこらへんがおかしくなるのか，どこらへんを改善しなければいけないのかという見通しが，設計の中から導けない．

ソースコードを作って動かしてみて，おかしい所を集中的にやっつけるとかいうことぐらいしかできず，設計過程での情報をもとに，システムの改善をするということができない．上記の諸論文の方法論は，こうした欠点をおぎなう方向性を示しているように思える．」

４．パラダイムの特徴と適用分野

先の論文紹介の後を全体的に見直すかたちで，種々のプログラミング・パラダイムの特徴比較と適用分野に対する解説がなされたが，問題の種類に応じて，どのパラダイムを適用してゆけばよいのか，制御がどのように表現されるか，という観点で整理してみた（図１参照）．

(a) 論理型，関数型アプローチ

問題の構造が明らかな場合に適用可能．入出力にしても，ボトムアップでもトップダウンでも，ひとつひとつをきちんと決めてゆけることができる場合に適用が可能である．この２つの相違は，“値を次々と作ってゆく”と捉えるか，“値と元の値との関係が真か偽か”と捉えるかの違いであろう．

また関数型と手続き型との一番の違いは，手続き型ではデータの流れと制御の流れがバラバラであるのに対して，関数型ではデータの流れと制御の流れが一致している．

(b) 手続き型

サブルーチンがあって，それらを動かす順序がアルゴリズムとして書ける場合．ただしそのときに，対応する動作が大域変数を用いると非常に複雑になり，問題をひきおこしていた．

われわれがプログラムを作るとき，その動作を定義してみると，通常はステートベクタという表記手法を使う．ステートベクタとは，変数を置いておいて，時間軸の変化につれて変数がどう変わるか，というものを掃き出したものであるが，実は，分析しているプログラムの制御構造は，ステートベクタの時間軸の方をマクロに捉えたものである．すなわち，それがストラクチャード・プログラミングそのものである．

(c) データフロー型，ペトリネット，エンティティ・アクション（ＪＳＤ）

いきなり問題から，メインのフローチャートやサブルーチンが書けるわけではないので，段階的に落としてゆく，もしくは一度皮をかぶせたもので記述して，それでシステムの動作を確認し，コードに変換してゆく（ＪＳＤに関しては，ＳＥＡＭＡＩＬの１１月号を参照されたい）．

(d) オブジェクト型アプローチ

オブジェクト指向では，ステードベクタの時間軸に連続した式というのがなくなっている．つまり先の大域変数に相当するものの固有の変換系列というのを定義して，それぞれデータごとに変換を加え，それらをオブジェクトとして形成する．

そして，それらのデータが変わるとか変わらないとかいうことは，勝手に自分自身で決るのではなくて，ほかのものが変えたから，これも変えなさいというメッセージが出てきて，それに基づいてとデータを変換してゆく．

(e) ルールベース・エキスパートシステム

これでよく問題にされるのは，ｉｌｌ－ｓｔｒｕｃｔｕｒｅとｗｅｌｌ－ｓｔｒｕｃｔｕｒｅということである．これについてはまだ明確な定義はされていない．これは多分，システム化しようとする対象の世界に依存する．

例えば，ロジック・プログラミングの世界で，前向き演えきとか，後向き演えきと称されるものを扱うときには，それら問題の対象は非常に厳密なものでなければならない．

それに対してルールベース・エキスパートシステムの場合には，ロジックで捉えられなければいけないというようなことはなくて，広義に解釈すればパターンマッチングできるもの，また何か変化を引き起すものならなんでもいい．

５．自由討論

自由討論で話された内容を，ランダムにいくつか紹介する．

Q： 私はジェネレータの設計等を中心に仕事をしてき

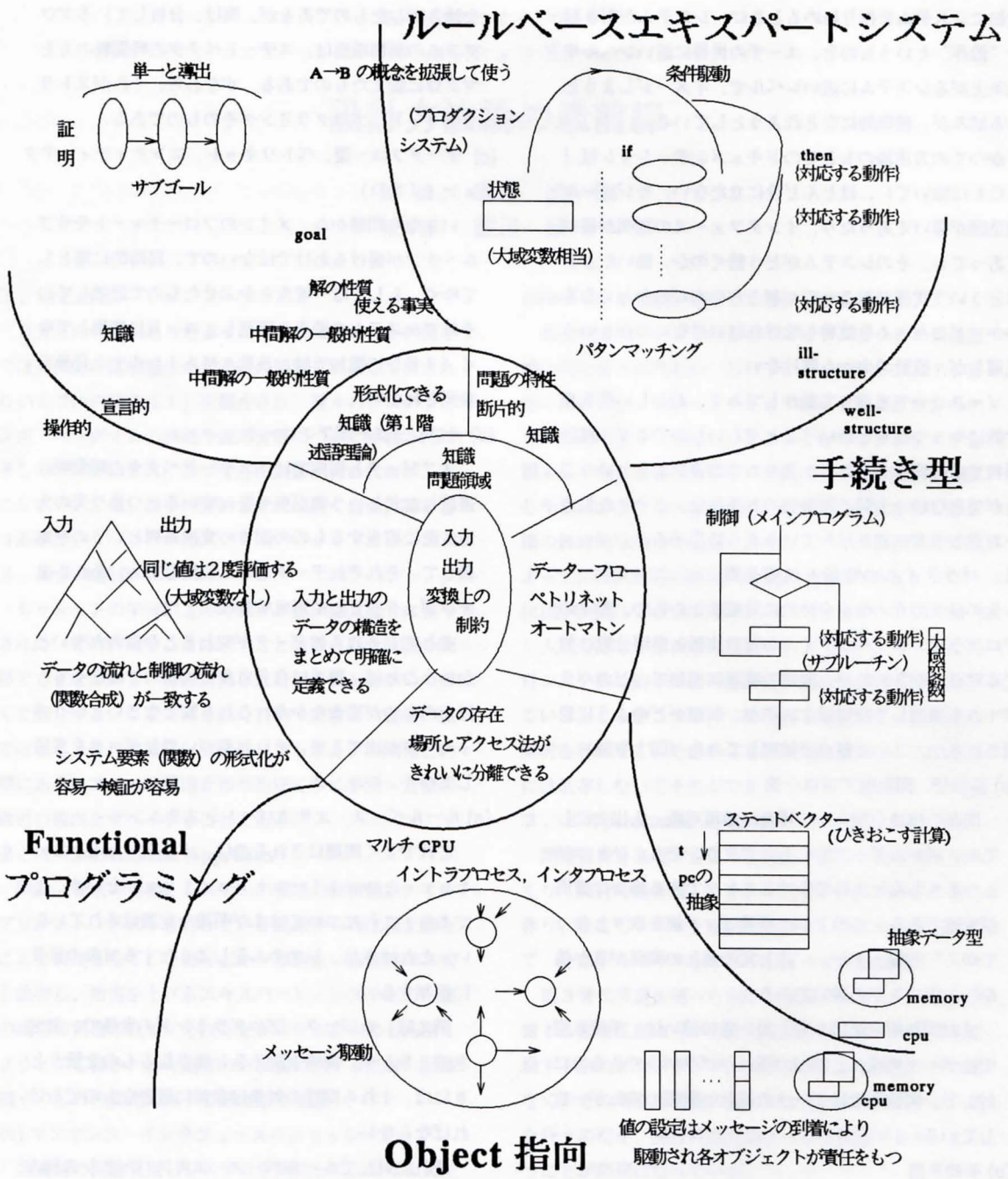

図1.

中心の円から外側に向かって,

問題の世界→仕様の世界→言語の世界→計算機（ハードウェア）の世界を表現している

ている．ＪＳＤ法やそれほど完成度は高くはないにしても，同様の考え方がいろいろあると思うが，そのなかでやはり一番難しいのは，名詞とか動詞とか（ＪＳＤ法でいうエンティティとかアクションに相当するもの）を抜き出すことで，かつ今問題になっているのは，システムをどう捉えるのかということだと思う．抽象的なものをどう整理すれば，マンマシンインタフェースの部分で，間違いなく問題を捉えることができるのか，というのがジェネレータを作る立場の人間としての一番難しいところだ．

落水： 私が思うには，ユーザの要求の観察観測をじっくりすると，そこらへんの問題は一挙に解決するような気がしている．一度そのようなことをやられてみてはどうだろうか．

Q： 簡単にいうと，現在要求定義はとにかく人間の世界にまかせられている．そしてその中で一番具体的にできあがるものは，画面レイアウト，ファイル・レイアウト及び帳票レイアウトの類である．プロセスというのは，データをどう加工するかということであって，具体的なものであるという位置付けで捉えたときに，要求定義をいかにして画面レイアウト，帳票レイアウト，ファイル・レイアウトにするかが今非常に大変で，システム設計そのものよりもひどい状態にある．

Q： データとプロセスを分離させるような話をされていたが，本来は，データとプロセスとは一緒になってあるもので（機能などはプロセスだけでは定義できない），そのデータとプロセスとの組合せが，どういう単位を認識して要求分析とか設計のときに位置付けてゆくのか，というところをしっかり整理しておくことが重要であると思う．

落水： それが事務アプリケーションの世界では，限られた言語の機能に展開できればよいわけで，そのなかでは，プロセスというのは非常にとらえやすい．データストリームの付帯的要件である考えてよいと思う．要求定義からファイル・レイアウトを作ったり，画面定義を作ったりするのが，極めて人間的な世界であるにもかかわらず，実は誤差が大きい．そこさえ良く解決するような手法が見いだされれば，事務アプリケーションの分野では，多くのものが自動プログラミング可能となる．

Q： それは，簡単なレポート・ジェネレータとか何かで，こなせないのか？

落水： どこにどのようなデータを，どのように出力するかという（エルゴノミクス的要素もふまえた），ベテランＳＥのノウハウを含んだ方法論が出てきてほしい．仕様が固まれば，システムの内側に向かって作り込んでいく環境は整備されつつある．

６．個人的感想

きわめて有益な話を聞けたというのが感想だが，一方で，それらを実践する環境がそろっていないことも事実である．受講者自身がこれらの設計方法を試みる機会は，たとえあったとしても限られるであろう．

また私がとくに感じたのは，各問題領域にそれぞれ適したパラダイムが存在し，それらを適用すれば，いままで実現が不可能であったような範ちゅうの分野への進出が可能であり，そしてそのための方法論が用意されているということである．

ただ現場では，それらの問題が複雑に（少なくとも２つ以上）含まれたシステム要求が多く，かつそれらを１つの限られたハードウェアと（多くの場合システムの統合性の要求から），１種類の言語で構築しなければならないことは事実です．ここらあたりの問題が，現場で早急に解決されなければ，折角の話も猫に小判のような気がした．

【参考文献】

〔１〕IEEE Tran. on Software Eng. Vol.12, No.2, Feb. 1986

以下は〔１〕のうち，講義で紹介されたもののリストである．

〔２〕Developinng Interactive Information Syatem with the User Software Engineering Methdology

〔３〕Object-Oriented Development

〔４〕Constrained Expression: Adding Analysis Capabilities to Design Methodos for Concurrent Software Syatem

〔５〕Functional Programming, Formal Specification, and Rapid Prototyping

〔６〕Sailent Features of Executable Specification Language and Its Environment

〔７〕The Transformation Schema: An Extesion of the Data Flow Diagram to Represent Control and Timing

〔８〕Process-Translatable Petri Nets for the Rapid Prototyping of Process Control Syatems

〔９〕An Overview of JSD

SEA秋のセミナー・ウィーク
セッション G2

ソフトウェアの法的保護

講 師： 佐野 稔
報告者： 田中 慎一郎

１．はじめに

私が担当するセミナーレポートの第２弾として，佐野先生の『ソフトウェアの法的保護』を取り上げる．自由討論を中心にまとめるつもりだが，そのまえに行われた講義から，プログラムを著作権法で保護することになったいきさつと，問題点を，まず整理しておく．

２．講義内容から

（１）なぜ著作権法か

プログラム保護に関して，日本では，通産省案の『プログラム権法』と，文化庁案の『著作権法の改正』という２つのアプローチがあった．結果的には，改正著作権法が今年の１月から施行されることになったが，これは多分に，『アメリカでもそうだから．．』，という理由があったように思われる．ではアメリカでは，なぜ著作権法でいくことになったのだろうか．

本来著作権法で保護されるものが，一体どういうものであるかは，私のようなソフトウェアの素人考えでも，著作権法でプログラムの保護を行うのは，無理があるように感じられる．アメリカでもＩＢＭ社は，著作権法ではうまく行かないと考えていたようである．それが著作権法で対応をとることになったのは，新しい法律を制定したとしても，それで望む結果が充分に得られない恐れがあったためである．つまり，新しい法律をもってプログラムの保護を行おうとした場合，外国，はっきりいってしまえば日本で，この法を侵害する行為があったとしても，その規制は難しい．ところが著作権法で行えば，万国著作権条約その他により，その内容が保証されることになる．これが実態ということだろう．

（２）著作権法とは何か

１５世紀半ば，活版印刷術が発明され，書物の発行が画期的に容易になったわけであるが，これは同時に，海賊版の普及にも絶大な効果を及ぼすことになった．この頃はちょうど古典復刻ブームでもあり，いろいろと手間をかけて古典を調べ，整理／編集したものが，海賊版に押されて，資金の回収ができない状態になったという．このため，まず，原稿をもっている原出版者に独占権があるという形で特許制度が成立した．

１７～１８世紀になると，『原稿をもっているから独占権があるというが，それならばむしろ書いた私の方に権利があるはずだ』，という当然の主張が起こり，フランス革命などの個人主義思想ともあいまって，著作物を著者の精神活動の所産とみとめ，これを保護するという方向に向かって行くことになった．ちなみに，日本における『著作物』の定義は次のようになっている．『思想または感情を創作的に表現したものであって，文芸，学術，美術又は音楽の範囲に属するものをいう』．

（３）問題は何か

では著作権法でプログラムの保護を行おうとすると，何が問題となるのであろうか．

日本では（取り敢えず）著作権法で行うと決って以来，本来なされるべき議論がほとんどされていない．これは，日本の場合，法律家はほとんど法学部出身であり，アメリカのようにコンピュータ・サイエンスをマスターしてから，法律家になることは少ないためである．このため，『プログラムは著作物である．著作権法で保護できる』と，一応の結論を出してしまうと，もう議論すべきことがなくなってしまった．しかし，現実的には多くの問題が残っている．つまり，著作権法で保護する以上，何が表現されているのか（プログラムにおける表現とアイデアの境界はどこか），どのように保護するのか（何をどこまで保護するのか），ということは非常に重要であり，この議論を軽視することはできない．

これまで，日本で発生した事件は，デッド・コピーといえる例ばかりである．これらはもう法律論以前の問題で，コピーをしたら悪いと決っている．アメリカでも，一昨年ぐらいまでは同じようなものであったが，最近は少し変ってきていて，『表現』が拡大解釈されているとしか思われないような判決が下されている．そのいくつ

かの例をあげる．

（a）ウェラン対ジャスロー事件（言語の書換え）

ＳＥＡＭＡＩＬ８月号の講座紹介でも扱われているが，『著作権では，アイデアを保護せず，アイデアの表現だけを保護している』としながらも，明らかに，コンピュータ内部でのデータの処理の仕方や，データの流れ方をも『表現』と認めている．このときの判旨は以下のとおりである．

『著作権法が保護しようとするのは，アイデアや概念を著者が表現している方法（ｍａｎｎｅｒ）であり，アイデア自体ではない．著作権はアイデアを保護せず，アイデアの表現だけを保護している．コンピュータがデータを編成し，集配し，保持し，検索し，使用するにはさまざまな方法がある．異なったコンピュータ・システムであっても，互いにコピーすることなく機能的に類似の目的を実現することは可能である．（中略）

コンピュータ・プログラムにおける「アイデアの表現」とは，コンピュータがディスプレイやプリンタあるいは音による信号を通じて有意な情報をうけ，集配し，計算し，保持し，相互に関連ずけ，あるいは他の情報を生みだすさいに，プログラムがどのようにして当該コンピュータを動かし，制御し，順序だてているかという，その「方法」のことを意味する．』（くわしくは，日経コンピュータ１９８５．１０．１４参照）

（b）サス・インスティテュート対Ｓ＆Ｈコンピュータ事件（ｐｏｒｔｉｎｇ）

小説のストーリを模倣する行為と対比して上で，プログラムのｏｒｇａｎｉｚａｔｉｏｎやｓｔｒｕｃｔｕｒｅを取り込むことが違法とされ，独自にｃｏｄｉｎｇしたということ自体が認められなかった．

『原告のｓ／ｗとおなじ機能を果たすｓ／ｗは，いろいろと考えられ，原告の採用したプログラムのｏｒｇａｎｉｚａｔｉｏｎやｓｔｒｕｃｔｕｒｅは，無数の方法論の内の一つにすぎない．それにもかかわらず被告は，原告のプログラムのｏｒｇａｎｉｚａｔｉｏｎやｓｔｒｕｃｔｕｒｅをそのまま取り込んでいる．

原告のプログラムのｏｒｇａｎｉｚａｔｉｏｎやｓｔｒｕｃｔｕｒｅを抽出し，それに基づいて独自にｃｏｄｉｎｇしたとしても，そのような行為は，やはり侵害である．（セミナーテキストから）』

（３）　ウィリアムズ対アーンド事件（書物からプログラム化）

これも講座紹介で取り上げられているが，商品相場の手引書をプログラム化した事件が，単なる翻訳と解釈され，違法とされたものである．やはりプログラムの処理手順を小説のストーリと同様のものと判断した例である．

・プログラムとは，人間が紙と鉛筆でできる作業を高速であるだけのこと．
・プログラム言語も自然言語の一つにすぎない．
・英語で書かれたｓｔｅｐ－ｂｙ－ｓｔｅｐのマニュアルに詳細な処理手順が書かれているならば，そのプログラム化は単なる他の自然言語の一つへの翻訳にすぎない．

（セミナーテキストから：くわしくは，日経コンピュータ１９８６．５．１２）

これらの例は，それぞれ裁判に持ち込まれるだけの背景もあったようではあるが，こうした判例が続出し，それらが前例として扱われることは，非常な危険をはらんでいる．つまり，これらの判例は一見，『おなじことをやるのであっても，違う方法でやるのであれば，全く構わない』，といっているようではあるが，その『方法』がくせもので，『考え方そのものと思われる性質のものであっても，いったん紙に書いた形を通せばもはや他人がおなじ考え方を紙に書くことは許されない』，という方向に進んでいるのである．

ある意味で，それが本当によいものであれば，なんらかの形で保護することは，必ずしも不当とはいえない．しかし，全てのケースにこれが無条件に適用されてしまっては，単に混乱を招くだけになろう．やはり著作権法では無理があるのではないだろうか．アメリカでは今年の４月に，やはり著作権法ではソフトウェアの保護はできないというレポート（米国議会技術評価局：ＯＴＡレポート）が議会に出されたので，今後また議論が活発になることだろう．

３．自由討論

法律的問題というのはなかなか難しく，討論というより，どうしてもわからないところを教えていただく形となってしまうようだ．

Ｑ：　プログラムの保護に関して，こうあったらよいだろうというところをお聞かせ願いたい．

佐野：　私はプログラムは著作権法では保護できないと考えます．それも，プログラムだけの問題に限らず，最近のコンピュータ・サイエンスのもたらす効果全般に関して，既存の法体系にとらわれるのではなく，何にもな

い白紙の状態から考えなくてはいけないと思います．これは他の先端技術全般に関しても言えると思います．

Q： ソースをユーザに提供する場合，それを保護するためには現状ではどのような手段が考えられるか．また，将来はどのような方向に向かうか．

佐野： まず，ソースでライセンスする場合，これはもう契約しかありません．現在の契約形態の最大の欠陥は，『使用権』という概念の定義がされていないことでしょう．やっていいこととやっていけないことを，もっときっちり定義しなくてはいけない．平たく言えば，使用権ではなく，まず『実行権』として，『指定機械の上で，かつ指定機械の制御の上で，プログラムの本来の機能を実現すること』というように定義をして，これを許諾する．つまり，許諾する範囲をきっちり定義することが必要でしょう．

将来ですが，著作権法で保護するという流れにあるうちは，このようにするしかないでしょう．法律には何にも書いてありませんから．

Q： コピーされたということを証明するために，やっておくべきことというものはあるのでしょうか．例えば，ソース・プログラムを公証人役場に登録するなどは，本当に意味があるのでしょうか．

佐野： 少なくとも，契約当事者間ではそう言った問題は起こり得ないでしょうから，第三者の場合になりますが，公証人役場で，確定日付をとっておくことはそれなりに意味があります．プログラムの登録などというものもありますが，それのもつ意味は，その日付に，そのプログラムが存在したという『推定』材料になるだけで，しかも登録した人が，著作権者かどうかに対しては，推定が働かない．それだけのことだったら，公証人役場の『証明』の方が有効です．

ちなみに，コピーはだめですが，表紙に「これは原本を写したものです」と手書きし，住所／氏名／日付をいれておけば，そういう原本になりますから，間接的に原本の証明にもなります．

Q： プログラムにおいて，ＣＲＴ画面の構成が非常に重要性をもっている場合，こういうものの保護はどうなっているのでしょうか．

佐野： これは，プログラムの問題とは，全然違うものになります．プログラムが全く違っても，おんなじ画面は作れますから，基本的に絵と同じになって，まさに著作権法の問題になります．あまり特別な問題にはなってきません．

Q： アイコンなんかは，意匠権に引っ掛かってくるのですか．

佐野： 意匠権というのは，『もの』のデザインに対してのもので，２次元のものは普通，対象になりません．絵であれば描いた時点で著作権法の保護下に入るので，そもそも登録の必要がありません．

Q： コピーライト表示というものは，法律的にはどういう意味を持っているのですか．

佐野： まず，日本の国内だけでは全く意味がありません．日本の本の著作権法はベルヌ条約系のもので，書いただけで保護下におかれますが，アメリカでは，コピーを登録／受理されて初めて保護下におかれるので，そのためのものです．逆に，コピーライト表示をしておけば，万国著作権条約に加盟している国に対して全部有効になります．日本などでは，最初に発行したときにコピーライト表示をしておけば，アメリカでその手続きをしたのと同じになります．

Q： 委託契約で作ったプログラムの権利は納入後はどうなるのですか．

佐野： 今まではあまり問題意識すら持たれていなかった．つまり，プログラムを作るのに用いた部品の権利は，作成者側にあると，何となく皆思っていたわけなんですが，契約で，『成果物に対する権利は委託者に帰属する』となっているので等，やっと問題意識が持たれるようになってきたところです．契約できちんとするのが一番いいでしょう．

Q： マニュアルなどドキュメントをもとにコーディングをすると全部違法となるわけですか．

佐野： アメリカでは違法であるという判決が出ています．ただし，アメリカの内部でも，それは，裁判所のプログラムに対する認識が薄いためという意見がではじめてます．ですから，いづれはこの考え方は変ると思います．現在のアメリカの裁判所のながれというのは，鑑定人（ＩＢＭ）の意見が重視されています．いまはコンパチを押えるために，アルゴリズムまで保護しようとしているようですから．

Q： 自動生成プログラミングの成果物等の著作権は，どこに帰属するのでしょうか．

佐野： これは今の時点では，難しい問題だとしかいえません．文化庁で，新たに小委員会が作られて，２年計画で動きはじめています．文化庁がやるのですから，中

で使用されるデータ（部品その他）を最初に作った人にあるという答になることは，ほぼ間違いないでしょう．

Q： ソフトウェアの盗用に対する処罰の法的根拠についてですが，現在は弱いのではないか．

佐野： プログラムの保護を著作権で行くといった場合には，ちっとも弱くありません．著作権法は，刑事罰が付いてきますから，違反すると手が後ろに回ります．懲役3年以下，罰金だと５０万円以下くらいですかね．アイデアと表現の境界がはっきりしていないのに，おかしいというのが私の考え方なんですが，罪刑法定主義という犯罪になる行為は，はっきりしていなければいけないという憲法上の思想からいっても，おかしいと思います．

Q： ある翻訳プログラムを，ＣＰＵの全く違う別のマシン上で，言語も換えて実現したとしても駄目というようなことですが．

佐野： アメリカの考え方では駄目ということになります．日本の場合は，文化庁がプログラムを著作物としていれるというときの審議会の報告の中では，言語を換える場合には，アルゴリズムに遡るから別個の著作物と見なすという見解を出しています．オフィシャルではないですが，当時の文化庁の見解はこうだったわけで，今後日本で問題が発生したとき，文化庁がどういう態度をとるか，非常に関心が高いですね．

Q： ソフトウェア著作権はどのように取り締まることができるのでしょうか．

佐野： 民事で，プログラムの使用の差し止め，ドキュメント類の破棄，ＲＯＭに焼いてあればそれも破棄，そういったことができます．刑事ですから，告訴することもできます．法人がやったときは，責任者も一緒に処罰されます．

Q： 著作権法でのプログラムの定義はどうなっているのですか．

佐野： 文化庁としては一生懸命考えて，『電子計算機を機能させて一つの結果を得ることができるように，これに対する指令を組み合せたものとして表現したもの．（2条1項１０号の２）』としてあります．

一番問題なのは，『機能させて』とあるが，これは『外から』の意味なのに，マイクロコード，マイクロプログラムというのは，中にあるわけです．実際には定義は難しいですね．契約論では，プログラムというのを直接定義しないで，いろいろな定義情報の中で明確になるようにされている．

Q： 契約できちんとするということですが，例としてはどのようにしておけばいいのでしょうか．

佐野： 大体次のようなことを決めておけばいいでしょう．

- 指定システム（ＣＰＵの製造番号など）
- 指定プログラム
- 実行
- 複製（プログラムをデータ列と見なして，同じデータの並びを再現すること）
- 改版権（改変権よりもこちらの表現の方がよい）
- 変更
- 訂正（ある機能を実現するための，論理的な誤りを取り除くこと）

佐野： ソフトの保護についてもし関心があるのなら，ＳＥＡの法的保護分科会で，月一回勉強会をやっていますから，そちらに参加ください．

４．終りにかえて

特許権法ではどうして保護できないかという話しもあったので，簡単に触れておく．特許権はアイデアを保護する法律だが，『この法律でいう発明とは，自然法則を利用した技術的思想の創作のうち高度のものをいう』となっている．プログラムの場合，この『自然法則を利用した』という部分が引っかかり，保護対象とならない．しかしこれがなんらかの回路に組み込まれて，装置になっていれば保護対象になるという．（例えば，電子ジャーの保温回路はオームの法則が入ってくるそうだ）．これは，あくまでもプログラムの保護ではないので，同じプログラムでも他の装置に使われてしまうと保護できない．結局この程度の間接的な保護に留まるようである．

いずれにしても，最近の技術進歩にともなって生まれるさまざまな問題を，現行の法体系で処理するにはもう無理な時期にきているのではないだろうか．最近米ＩＢＭは，先に上げたような『表現』の範囲を拡大解釈する判例を背景に，外部仕様情報（ＥＩ）にまで著作権が及ぶと主張している．これはいろいろな意味で危険をはらんでいる．わが国でも，米ＯＴＡレポートにもあるように，知的所有権法全体の抜本的改革を考える必要があるのではなかろうか．このあたりの情勢については，日経コンピュータ１９８６．１１．２４にかなりくわしく紹介されています．

SEA秋のセミナー・ウィーク
セッションB3

大型ビジネス・アプリケーションの開発環境

講　師：盛田　政敏
報告者：関崎　邦夫

１．はじめに

「環境」という言葉が，世の中でずいぶんと使われるようになったが，これまではどちらかというと，環境といえば，ＵＮＩＸの専売特許のような感があった．一方，ソフトウエア開発の大半をしめるビジネス・アプリケーションの開発環境はどうなっているのか，という真剣な議論はいままでほとんどされていない．

このセッションは，そんな不満（？）から開かれた．講師の盛田さん自身の開発環境の紹介，参加者の事例紹介，および自由討論など，かなり具体的な内容で行われた．それらを全て紹介できないが，私自身も大型ビジネス・アプリケーションの開発に関わっているので，特に興味をもったことに絞ってレポートした．

２．講師自身の開発環境とその問題点

大型ビジネス・アプリケーション分野の開発は，エンドユーザ，メーカ，ソフトウエアハウスの三者構成で行われることが多い．そこでは，アプリケーション・スペックは顧客主導であり，システム・スペックはメーカ主導で行われることがほとんどで，この状況はどこも大差ないのではなかろうかと思う．

開発期間も１～２年，長いものでは５年というものもある．作業場所は顧客先が中心となっている．このような条件にあってソフトウェアの開発をしているのだが，次のような点が開発を行っていくうえで問題となる．

１）プロジェクトの推進がユーザ主導，メーカ中心となるため，技術思想がメーカ・オリエンテッドになり，エンジニアだけでなくマネージャにも主体性，自立性が育ちにくい．

２）技術蓄積が個人ベース，またはユーザ／プロジェクト毎に分散し，組織としての体系だったフォーマルな蓄積が行えない．

３）同一マシン，同一ＯＳの場合でも，ユーザ／プロジェクト毎に標準が異なるため，作業効率が悪く，経営的にも売り上げを増やすためには，人員を増やさなくてならない．

４）作業場所が社内でないため，モラルの低下が懸念される．

以上の点を解決すべく努力は行っているのだが，結果はそうすぐにはでないようである．

大型ビジネス・アプリケーションの社内開発用マシン（ＯＳ）としてＵＮＩＸを考えると，ハードウエアからの独立，パブリック・ドメインのソフトウエアの活用，リサーチ・コミュニティの拡大等のメリットが考えられるが，まだわれわれが実際に開発をしている事務アプリケーション分野での利用人口が少なく，また事例も少ないため，導入をしたときの効果等の具体的イメージが掴めない．

さらに，開発マシンとターゲット・マシンの分離という面から考えると，ＵＮＩＸ上ではこの分野向けの支援ソフトウエアが少ない．このへんはシグマ計画でどこまでおこなわれるかは，いまの情報量では判断でききれない．

ターゲットＯＳを開発用とした場合，直接的な開発支援，つまり多くのプログラマの支援が行えるが，メーカ依存の精神的基盤は変えられず，コスト面でも採算が取れない．またＯＳとパッケージプログラムのエディションの整合性等，プロジェクト毎の開発環境の構築にはかなり無理がある．大型汎用機開発の場合，単に開発計算機を導入しても問題は解決しない．

このように，対象（プロジェクト，問題）が分散しており，プロジェクト毎に環境，方法論が異なる．そのため一つの集中された環境，方法論ではこれらの問題をすべて解決できない，というのがいつわらざる現状であろう．

そんなわけで，現在われわれのところでは，マルチホスト接続型の開発環境において，各ターゲット・ホストへのアクセス手段を確立し，問題毎に最適な方法論，ツールで対処し，それら方法論およびツールを蓄積しなが

ら，統合化を考えている．

３．参加者からの事例紹介および自由討論

盛田：　メインフレーム下でのツールということで，どなたか事例紹介をお願いします．

S氏：　ＣＯＢＯＬ開発環境支援の一例として，対話形ＪＣＬジェネレータとＣＯＢＯＬプログラムのスケルトン・ジェネレータを紹介します．

ＪＣＬジェネレータは対話的に，コンパイル・リストの要／不要等を質問に答えて行くことにより，コンパイル時のＪＣＬをジェネレートします．スケルトン・ジェネレータは，入出力ファイル等を対話的に答えて行くことにより，ある例ですと，約２０項目の入力で６５０ステップのＣＯＢＯＬソースをジェネレートします．枝葉の部分はオン・コーディングが必要ですが，キーワードを埋めて行くだけで，入力されない（不要な）部分は，自動的に削除されます．

私の考えでは，プログラミング工程以降の環境は，メインフレーム上で行い，それ以前の作業はワークステーションで行えば良いと思っています．

盛田：　今年の３月に，ＳＥＡの主催で長岡で行われた「環境を考えるワークショップ」（ＳＥＡから報告書がでていて，事務所に問い合わせると入手できる．また，ＳＥＡＭＡＩＬの４月号にもパネル討論が掲載されている）においても，ＣＯＢＯＬの問題が討論され，そこでの対処策としては，

（１）ＣＯＢＯＬ用開発環境を支援していく

（２）ＣＯＢＯＬを意識しないようにツール等で包む

（３）ＣＯＢＯＬを否定する

が出たと思いますが，（１）および（２）は，いま紹介のあったツールは該当すると思います．ＣＯＢＯＬを否定するということで，どなたか意見は？

Ｉ氏：　わが社もビジネス分野では，ＣＯＢＯＬ，ＰＬ／Ｉが主体ですが，言語の選択は，われわれがうんぬんする問題ではなく，カストマの指定により決まってしまい，善し悪しでなくて，やらざるをえない問題だと思います（笑い）．

盛田：　さきほど設計段階をワークステーションで行えばよいという意見がありましたが，開発環境の分離という形態でメインフレームの環境は，ワークステーション上で実現出来るのでしょうか？

Ｉ氏：　私の場合は，ＵＮＩＸをＯＳとしたワークステーションですが，いまのところは実現できていません．特にＤＢ／ＤＣ関係は無理だと思います．現在は，プロジェクト管理，ドキュメント支援等のＳＥのＯＡ化という形で，実現しています．

盛田：　要求定義，設計工程での支援ツール等の事例についてどなたかございませんか？

Ｉ氏：　現在，システム開発の全ライフサイクルを支援する統合化ツールと技法を開発しています．その鍵となるのは，ソフトウエア・データベース（ＳＤＢ）だと思います．すべての開発が標準的な作業で行えるよう，単一のコマンドでＪＣＬを作成しないで，つまりプログラム名を指定すれば，どこに格納されているか，記述言語は何か等の付加情報をＳＤＢから与え，自動的にコンパイラ，ライブラリ等を選択／実行するというものです．

プログラミング工程以降は，個別のツールが有りますが，設計段階で実用化したものはいまのところないと思います．設計段階では，文字情報以外に図形データ等も含まれ，それらを同時に扱える実現可能なワークステーションとしては，Ｊ－Ｓａｔｒぐらいしか無いと思います．Ｊ－Ｓｔａｒで，作成した設計ドキュメントをプログラミング工程でどのように結び付けていくかということで，どなたかにコメントを頂けたらと思います．

Ｈ氏：　Ｊ－Ｓｔａｒでは，ＲＪＥ機能があり，ホスト上へのファイル転送は可能です．しかし，ホストへ格納した後の処理ですが，現在文字データの解析は終了していますが，図形データについてはまだ解析中でして，実現化はされていません．

盛田：　プロトタイピングについての事例はございませんか？（場内沈黙）プロトタイプというのは，世の中でいわれているほど実際にはまだ利用されていないようですね．先程のソフトウエア・データベースの件ですが，ＣＰＵの負荷の点からはどうでしょうか．

Ｉ氏：　やはり３０人程利用しますと，かなり負荷がかかりますね．今後の課題としては，世の中の動向もそうですが，分散化が上げられています．

盛田：　大型システム開発の場合，数十人から数百人までの人間が作業しており，経験者によりプロジェクト管理が行われていると思いますが，管理用のツールがありましたら紹介をお願いします．

S氏：　先程のＪＣＬジェネレータ，スケルトン・ジェネレータの場合，使用履歴が取れるようになっており，誰がどのような事をしているかモニタができますが，実際には使用しておりません．結局は，マネージャの経験

により管理しています．

Ｉ氏：　先ほどのＳＤＢでは，担当者の情報も入るようになっており，担当者別のプログラム作成状況一覧が出力できます．ただし，どのプログラムが何回コンパイルしたかというレベルまでは管理していません．

盛田：　大型の場合，ＣＰＵ時間などへの課金の経費管理も必要と思いますが．

Ｉ氏：　あるメーカの場合，ＯＳの機能としてＣＰＵ時間の計画管理が行えます．プロジェクトの開始時，予想使用時間を登録しておき，それを越えたらセッションの開始ができないようになっています．また利用者は，セッション終了時に，残りのＣＰＵ時間が表示提供されますので，それを参考にしています．

４．おわりに

いくつかの情報処理の雑誌をみていると，設計工程を支援する開発ツールもいろいろとあるようだが，このセッションに参加された受講者の方々の開発環境では，浸透していないように思える．これは受講者の環境が貧弱ということではなく，まだこうしたツールが，現場レベルまで浸透していないということだと思われる．

そのためか，今回のセミナーでは，互いにどのような開発環境で行っているのかの確認にとどまっており，どう問題を解決していくかという点にまで，議論がすすまなかった．

参加者の一般的な意見としては，テスト工程以降の環境は，メーカ提供のテスト環境にそれなりに満足しているようである．参加者はソフトウエアハウスの人間が多かったが，そうした立場の人たちにとっては，設計工程のツール，大規模なシステム開発のマネージメント方法など，現実に対処しなければならない問題が多かった．こうした現場の声として，ＳＥＡＭＡＩＬ１０月号で，佐久間まゆみさんのエッセイが掲載されていたが，そのなかでは，ツールの使い勝手の悪さが指摘されており，興味深いので参考にしていただけたらと思う（ちなみに，佐久間さんは，講師の盛田さんの部下だそうである）．

セミナーを通じて，現場の方々の話しを聞けたのだが，残念なのは，メーカのツールを作成している方の意見が聞けなかった事である．これもＳＥＡＭＡＩＬの１０月号に，田中慎一郎さんの環境分科会のレポートで，メインフレームのＴＳＳ環境の紹介とそれを基にした討論の記事が掲載されていたが，そのレポートの最後に，ツールを供給する側（メーカ）と，使う側（ソフトハウス）の意見交換の接点をもっと多くすべきだという意見があったが，まったくその通りだと思う．こうした討論が，ネットワークを通じておこなわれると，もっといいのではとも思う．

来年２月にＳＥＡ主催で第２回めの，「環境ワークショップ」が開催されるようであるが，大型ビジネス・アプリケーションの問題も，ぜひとりあげてほしい．そして，そのなかに，メーカの人たちもぜひ参加して，使う側，作る側の両者の意見を吸収していただけたらと希望する．

＊　　＊　　＊　　＊

お知らせ

以下の２つの報告書の在庫があります．ご希望の方は，事務所までお申込ください．また，ＳＥＡＭＡＩＬの在庫は，７月号を除いてありますので，こちらもどうぞ（１部：５００円－送料別）．

実践的ソフトウェア開発環境報告書

今年の２月におこなわれた長岡ワークショップの成果をまとめたもので，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．

販売価格　ＳＥＡ会員：￥３０００
一　般　：￥５０００
（送料別）

第４回夏のプログラミング・ワークショップ報告書

テクニカル・マネージメントを中心に，という副題をつけて，「プロジェクト・プラニングのケース・スタディ」をテーマにしたワークショップで，プロジェクト管理を実践的に討論したもので，参加者のポジション・ペーパ，パネル討論等を，１８０ページにまとめたものです．

販売価格　ＳＥＡ会員：￥３０００
一　般　：￥５０００
（送料別）

プログラミング言語雑感

引地　信之
ソフトウェア・リサーチ・アソシエイツ

1．はじめに

アマ（大学院までを含めた学生時代）と，プロ（会社に入って給料をもらってプログラムをつくってから）を含めると，ずいぶんとプログラムを作ってきた．それがいいプログラムであったかどうかは，私の作ったプログラムに接した人たちの判断にまかせるしかないのだが，１０年間程のプログラミングをしてきた経験のなかで，さまざまな文献や他人のプログラムを見てるなかに，非常にきれいだとか，なるほどと思った経験が何度かある．そうした感動・感心をふまえて，私自身のプログラミング言語経験を述べてみたいと思う．最後に，これから出てきそうな言語，あるいは出てきて欲しい言語についても触れてみた．

2．プログラミング言語とプログラミングスタイル

(1) Fortran

人のプログラムで，文番号の付け方をうまくやっているのを見て，なるほどと思ったことがある．文番号の欄を見ると，最初から昇順に番号が振られていて，非常にみやすいものであった．たしかに，行き当りばったりの番号付けのプログラムから，目的のものを探すのはかなり手間取る．しかし最初から，整然と昇順に番号を振るようにプログラムを作成するのはたいへんである．経験者はこのへんはよくわかると思う．そのときやっていたのは，一度コーディングしてから，後で文番号をふりなおすことであった．今考えると，ツールを用いてやるか，あるいは最初から文番号のほとんどいらない Ratfor や EFL を採用するほうがいいことが判る．

私が Fortran を使っていたときは，構造化プログラミングがもてはやされていて，段付けを行うことが流行していた． do 文と continue 文の頭をそろえ，本体は２文字分くらい下げるのである（いまでもそうらしい）．また，小文字で書くことも流行っていた．つまり，それまでは FORTRAN とか DO 文といったように，大文字で書いていたが，それを小文字にしていたのは，小文字で書くと，目にもいいとかといっていたような気がする．しかしこれも考えてみると，大文字だけの文章（プログラム）なんて実に読みにくいもので，小文字で書くのがあたりまえといえる．いまだに大文字しか扱えない言語（処理系）があるが，その言語の設計者というのは，もしかしてプログラムをつくっていないのじゃないか，と常識を疑ってしまいたくなる．

Fortran 経験の中で，ちょっと人のプログラムを見て迷ったことに， format 文をどこに置くかということがあった．それを参照している read 文や write 文とかのすぐ後ろに書くか，あるいは，stop 文か return 文の後ろに，まとめて書くかということである．結局，read/write 文の後ろに書いていた．

(2) Algol

この言語の文法とプログラムを初めてみたとき，エレガントだと感じたが，実際にプログラムを作って実行してみる機会は，あまりなかった． call by name，call by value，call by procedure という概念と，再帰呼出しを理解するのに骨が折れた記憶がある．

(3) Pascal

本格的にいろいろなプログラムを作成したのは，この言語が最初である．最初は，セミコロンのルールを理解するのに手間取った．「アルゴリズム＋データ構造＝プログラミング」のなかにあるプログラムがきれいだと思ったものだが，とくに sentinel の使い方が巧妙ではあるが，無理があるとも感じた．

(4) Ratfor

段階的詳細化，あるいは programming tool という概念を，この言語に触れることによって得ることができた．「Software tools」は原書で（私が Ratfor を始めたときには，原書しかなかった），プログラムの部分だけ読んだ．本文はあまり読まなかったが，詳細化の過程が非常に美しい．

Ratfor という言語は，はなかなか優れていると思うのだが，エラー・メッセージが Fortran コンパイラからでた場合に， Ratfor のインタプリタで作成される Fortran

のプログラムを見ないとわからないのが難点である．

(5) C

C 言語は，非常にコンパクトであるという印象を最初もった．エラー・メッセージに慣れるまで時間がかかるのと，当時処理系が少なかったという欠点があったが，私にはその簡潔さが快感で，好んで使った．簡潔さのひとつとして，キーワードが短いということがある． Pascal の場合，必ずキーワードのミスタイプがあった．例えば，procedure とか function などはよくミスタイプをした．私個人の使用感でいえば，短いキーワードというのは，速くプログラミングを作成するのに役立つ．ただし，生産性（バグの少ないこと）とは相関関係はなさそうである．

エラー・メッセージを含めた簡潔さが，C 言語をとっつきにくいものにしているといわれているが，もうひつにポインタの使い方がある．これをうまく使うかで，エレガントなプログラムになるかの分かれ目になるが（先輩の筏井さんのプログラムはその意味で，実にエレガントなもので），初心者には，このポインタの使い方が難解である．私の場合は，C コンパイラの生成コードであるアセンブラを見て，そのメカニズムを納得したものである．CかFortranとのつき合いがもっとも長いだろう．

(6) Lisp

これも使ってみた印象では，非常に短い行（分量）であっても，やれることが大きいと思った．というのは次のような経験があったからである．Prolog のインタープリタが Lisp で書かれているものがあり，新しものがりやの私としては，早速動かしてみたのだが，例に依ってタイプミスがあり，すぐには動作しなかった．この原因を探るためにソース・リストを追ってみた．そのソース・リストの２００行くらいだが，ユニフィケーションの部分がうまく処理されていた．私にとって，リストの概念の習得が，Lisp に慣れるキーポイントとなった．

(7) Yacc/Lex

このふたつは，プログラミング言語といえないかもしれないが，非常に有効なツール（言語）である．このふたつの言語でつくったプログラムのデバッグに，最初はずいぶんと苦労した覚えがある． Yacc の conflict とか，実行時のデバッグなどである．

Yacc/Lex の出現は，プログラミングの過程を根底から変えたと思う． Yacc/Lex を使って，テーブル操作とコード生成の部分を付け加えれば，簡単なコンパイラならすぐに作れる．ちょっと努力すれば， CP/M のインタープリタくらいなら作るのは簡単．

(8) Prolog

ユニフィケーションとカットが難解である．一見するとプログラムではなく，まるで要求仕様の羅列のように見える．そんなことで，Prolog を仕様記述言語にという人もいるようだが．問題の記述のみで，本当に実際の動作が行われるのかと，感動したこともあった．しかし実際はなかなかそうはいかなくて，それはコーディングしてみるとすぐにわかる．さらに全体が非常にフラットな感じがした．変数のスコープが Global と local の２つしかなく，これで大規模プログラミング開発に使うのは，たいへんだろうということが推測される．

(9) Unix

Unix はプログラミング言語ではないが，改めて項を設けて述べてみたい． Unix の世界には，ソース・プログラムが沢山あって，新規にプログラムを作る際に，再利用可能な部分やヒントが，そうした先人が作ったソース・コードから簡単に得られる．つまり，プログラムを最初から作るのではなく，まず使えそうなソース・プログラムをコピーしてきて，次にそれを直して使うというスタイルに変わっていく．こうしたプログラミング能力というのは，プログラム作成能力ではなく，どのくらいソース・プログラムを知っているか，ということにかかってくると思われた．しかしながら，当然のことであるが，基本（正規表現とかソートの種類）を知っていないと，ソース・プログラムさえも読めない事態が生じるので注意が必要である．

(10) Smalltalk80

いうまでもなく，オブジェクト指向プログラミングというパラダイムを提供している．

最初は本（いわゆる blue/orange book）しかなく，実際に使うことができなっかたのだが，クラスとインスタンスの区別が難しく， super と self も判らなかった．原書を読んで，非常に短いステップで，新しい機能が追加できる， OS の中身でさえ再定義できる，ブラウザが

強力である，といった印象があった．実際にプログラムを作ってみた経験では，エディタが非力のようであるのと，世界が閉じているといった欠点が見受けられる．

(11) ESP

Prologにオブジェクト指向のパラダイムを追加した言語である．インスタンス，クラスの理解が一つの山となり，データの種類も豊富である．これを使用してプログラムを書く場合に，どういった過程で作成するかという方法の模索が最初は続いた．「Software Tools」とか，前述したSmallTalk80のblue/orange bookに相当する参考になる本もほとんどない．Unixの場合，参考にするソース・プログラムは山ほどあるが，ESPの場合ほとんど最初から考えなければならない．

コーディングの際に迷った点を次に示す．

・どれをグローバル変数にしてどれを入力変数にどれを出力変数にするか
・Prologと共通した問題であるが，Cut のデバッグ，あるいはどこでバックトラックしているか
・ループ，選択枝の表現

こうしたことから，プログラミング・ツール，部品集の整備，プログラミング・スタイルの確立などの仕事がまだまだ残っている．

私自身では，クラス／インスタンス／インヘリタンスをうまく使って，なんとかきれいにコーディングできないものかと考えたが，結局はうまくいかず，従来の手続き的なアルゴリズムを，ESPへのマッピングということになってしまっている．

プログラミング・スタイルをどのように統一するかということで，例えば，ESP/Prologで，次のような条件式のスタイルを考えてみる．

```
(Condition == 1, !,
        A = 10;
        B = 20),
```

これはC言語的に記述すると，次のようになる．

```
if (Condition == 1)
        A = 10;
else
        B = 20;
```

ところがESP/Prologの';'を','とミスタイプすると，全く違う次のような意味になってしまう．

```
if (Condition == 1) {
        A = 10;
        B = 20;
}
```

この点をはっきりさせた書式が必要である．例えば次のように書き換えればよいだろう．

```
(Condition == 1, !,
        A = 10
;
        B = 20
),
```

ESPのコメントの書き方も，統一した書き方を実行していない．私自身では，この言語で初めて日本語のコメントを書き出したのだが，慣れないせいもあって，まず日本語の動詞がうまくコメント化できない．例えば，英語だと「get the file name from disk」のコメントを日本語で書くとすると，「ディスクからファイル名を得る／とる」となり，なんとなく妙な感じを受ける．これは単に慣れの問題かもしれない．

3．これからの言語

以上私のプログラミング言語の体験を，簡単に述べてきたのだが，私の少ない経験から，これからでてきそうなプログラミング言語について，考えてみた．

これからも，いろいろなパラダイムを盛り込んだ新しい言語が出現するだろう．そのなかで，次のような特徴をもつ言語がこれからおもしろくなるように思う．．

・モジュール化（例えばModula-2など）
・マルチプロセス対応，分散処理用言語（occamなど）

さらに，インクリメンタル・コンパイルなどの特徴をもった言語も，これから出現するように思える．しかし，こうした言語が，自己記述性／移植性に優れたものであることを望む．

中国旅行記

連載 －その３－

野村敏次
日本電子計算

１６．上海到着

(1) 空港にて

ほぼ正午，無事上海空港に到着．真夏の陽射しが降り注いでいる．人口１３００万人というより世界最大の都市なのだから，かなり近代的な空港を予想していたが，設備はほとんど機械化されておらず，西安と大して変わらなかった．しかし，これは国内線だからで，国際線の発着は非常に近代化されていることが，出国時に初めてわかった．

飛行機からバスに乗りゲート（といっても，ちょうど駅の改札口の感じであったが）を出ると，そこは出迎えの人々で一杯であった．例によって，こちらの名前を書いた紙を持っている人と出会い，御互いに簡単な挨拶を交わした．出迎えて下さったのは，上海計算機ソフトウェア技術開発センタの Qiu 副主任以下３人の方々であった．

やがて荷物が運ばれてきたが，ターンテーブルもなく，荷物をカウンタの中に積み上げていくだけである．しかも，何百人もの荷物を，わずか数人で積んでいるので，なかなかはかどらない．その内に待ち切れなくなった乗客は次々にカウンタを乗り越えて中に入り，自分の荷物を捜しだし持ち出し始めた．われわれもそれにならって，自分の荷物を持ち出すことにした．

空港の正面は，大きな駐車場になっており，乗用車でほとんどいっぱいであった．われわれの乗った車は，中国車の［上海］．フォルクスワーゲンを一回り大きくしたような車で，中はかなりゆったりとしており，窓にはレースのカーテンが掛けられ，運転手の斜め上には扇風機が回り，運転席と助手席の間にはポットまで用意されていた．

ホテルまでは，約４０分位であったが，その間横に座った迎えにきてくれた若い人といろいろ話をしたり，街の中の説明をしていただいた．１０月に Qiu さんがビジネスで日本に来られるという話を聞かされたので，「あなたはどうか？」と尋ねると，自分のような若い人間が海外に出るのは非常に難しいということであった．

郊外の比較的広々した所を走っていた車は，やがて市街地に入る．北京とは異なり，道幅はせまく，人の数は今までのどこよりも多い．特に一番の繁華街である南京東路にさしかかると，両側の歩道はまさに人の波であった．ただ，自転車は他の都市よりも少なく中国らしさが消えたような感じを持った．道幅が狭い上に車と人が多く，街中では危なくて自転車に乗っていられないという説明であった．

(2) ホテル

われわれに用意されたホテルは，南京東路の東のはずれにある和平飯店（旧イギリス租界のキャセイ・ホテル）であった．ニクソン大統領もかつて泊まったことがあるそうで，玄関の扉の上には１９０６という創業の年号が刻まれていた．

車は玄関前に横付けされたが，日本やアメリカとちがって，ベル・ボーイが荷物を運んでくれわけではない．時間はすでに午後１時を過ぎていたが，チェックインは２時からということなので，荷物だけをフロントに預け，チャイニーズ・ブランチをとることにする．

ホテル内のレストランは奇麗で，注文に際しても英語が通じ，何か非常に安心感を覚えた．上海ビールと上海料理で腹一杯になった後，腹ごなしに隣を流れる揚子江の支流の黄浦江の辺まで散歩することにした．

河というより海という感じである．かなりの大型船が往き来している中を，ポンポン船や，水上生活者の船を引いた小型船が，その間を縫うように動いており，かなり活気のある港の様相を呈していた．河の色も空からみた時のような茶褐色ではなく，青と焦茶色を混ぜ合わせたような色で，御世辞にも奇麗な水とはいえない感じであった．

河岸には１ｍほどの高いコンクリートの壁があって，そのの壁沿いに遊歩道が作られていた．木陰にはベンチも置いてあり，市民の人々の憩の場所になっているようであった．ここもかなり混雑していたが，たしかに水の

側だから涼しいし，眺めもよいから，時間を潰すにはもってこいの場所である．２時過ぎ，ホテルに戻り，部屋に入ったが，中は古めかしい感じはするものの，きちんと掃除されており，重厚さが感じられた．家具や調度品はかなり立派なものが揃えられており，ヨーロッパの古いホテルに近いイメージであった．

１７．上海の街

(1) ショッピング

日曜日の朝，比較的ゆっくり休んだ後，散歩がてらショッピングを楽しむことにした．ホテルの前の南京東路を西に向かって第一百貨店まで，文字通りの人波をかきわけつつ，一軒づつ覗きながら歩いていった．

衣料品店とか，時計・貴金属店，それに食料品店が特に混雑していた．掛け軸や硯を売っている店とか，お茶屋さん等は比較的すいている感じであった．食料品店の前では，西瓜を立ち食いしている人が多く，食べ終わった皮が道端に積まれ，それらが日に当たって異臭を放っていた．

ガイドブックによれば，ホテルから第一百貨店まで３０分ということであったが，店を覗きながら歩いたせいか，１時間以上もかかってやっと着いた．中国の百貨店というものも一度は見ておかねばと思って，店の中に入ったのだが，あまりの人の多さに圧倒されて，とても買い物どころではなかった．

何を売っているか見ようとする前に，人をかきわけて前進するのに精一杯なのである．その上，冷房もないから，熱気で気持ちが悪くなる程であった．ほうほうのていで逃げ出し，この百貨店の前に最近できたという歩道橋の上で一休みして，やっと人心地がついた．

この歩道橋からの眺めは非常によく，上海の高い建物も見渡すことができ，異国情緒豊かな景色をゆっくりと楽しむことができた．

帰りは行きと反対側の店を覗きながら歩いたが，途中に第十百貨店というのがあったので入ってみた．ここは比較的空いていたし，日本のデパートの感じに近かった．オールシルクのスカーフやネクタイが数百円という値段で売られているのである．ただ女性のスカーフはまだしも，ネクタイは柄が日本人には合いそうもないものが多く，買う気は起こらなかった．

ホテルの近くに友誼商店もあったので行ってみた．かなりの広さであり，４階までぎっしりと土産物で埋まっていた．外国人専用であるため，空いているし，奇麗な物も多いが，街の中の普通の店に較べると値段はかなり高かったように思う．ただ，余り歩き回らなくてもいろいろな種類の土産物はあるし，売っている物も奇麗であるというイメージがあり，ここでの買い物が一番多かった．

(2) 街中で

上海の街を歩いていて強く感じたのは，ここでも，交通規則は，車を除いて，まったくといってよいほど守られていないことであった．北京で最初に感じたことは，上海でも同じで，人々が交通規則というものに慣れるまでには，未だかなりの時間と努力を要するような気がした．

北京では，交通信号はあったが車専用であり，歩行者用の信号というものはなかった．上海では日本とほぼ似た形の歩行者用の信号もついているのだが，誰一人として守る人はいない．

あちこちの交差点には，網を持った人がいて，歩行者用の信号が赤になると網を張って人を止めるのであるが，それを避けてどんどん赤信号を渡っていくのである．網を持った人はそれを見ても止めるのでもないし，怒りもしない．日本だったら，交通事故の連発であろうと思うような光景は到る所で見受けられたが，それでも，事故がそれ程多く起こっていないところをみると，運転者も歩行者もそれなりに慣れているということであろう．

(3) 魯迅の墓と故居

和平飯店の９階のレストランで，黄浦江を行き交う船を眺めながら，遅い昼飯を済ませた後，車で魯迅先生の墓と故居を見にいくことにした．ホテルで車を呼んでもらい，墓のある虹口公園へ向かった．ここは，市の東北にあり車で１０数分の所である．車に待っていてくれるように頼んで，公園への入場料５分（日本円にして約２円）を払い中に入った．

なかなか奇麗な公園で，大きな池にはボートが浮かび，小さな池には蓮の花が一杯であった．ナイター設備のあるサッカー場などもあり，木陰のベンチで語り会う人，ボールを蹴って遊ぶ子供達を見ていると，日本の何処かの公園に遊びに来ているような錯覚に陥った．

公園の奥にある魯迅の墓の前には，ゆったりと腰を降ろしている魯迅の像があり，その後ろの大きな大理石の墓碑には，「魯迅先生之墓」という毛沢東直筆の文字が書かれていた．その前には花輪が供えられており，ついているリボンには，日本国宮城県青年の船云々と書かれ

ていた．近くには人も殆どおらず，実に静かな所であった．

次に魯迅記念館を訪ねたが，ちょうど改修工事中ということで，ここは見ることはできなかった．公園の中をしばらく散歩してから，近くの山陰路というところにある魯迅の住んでいた旧宅［魯迅故居］を見に行くことにした．

地図を頼りに，その方向に歩いていったが，それらしき建物は何処にもない．たまたま通りかかった老人に訪ねたが知らないとのことだったが，そこへ恰度もう一人の老人が通りかかり，通りが一つ違っていることを教えてくれた．教えられた通りを，南の方へ歩いていったが，それらしき表示も家も見当たらない．

地図を片手にキョロキョロしていると，肩をたたかれたので振り返ると，先程の老人が立っており，来過ぎているから戻れと言う．少し後ろの方に白髪の老人が立っており，そこが故居だというのである．丁重に礼を述べて老人の立っている場所に戻ったが，そこは，アパートの入り口の門であり，何の表示もない．

すると，門の前に立っていた白髪の老人が近寄ってきて，門の中を指差した後，９というのである．意味が判らず首を傾げると，今度は手で９を示した．多分９番目の意味であろうと思い，礼を言って門から中に入ってみた．煉瓦色の３階建ての高級アパートが両側に建っており，それぞれの家の入り口には更に門がついており，そこには，入り口に近い所から，１，２，・・・と番号が振られていた．ここで初めて入り口にいた老人の９の意味を理解し，９番目の門の前に行くと，そこには門柱に［魯迅故居］と書かれていたのである．

鉄の門には鍵がかけられ，中へは入れないので，外から建物を眺めるだけであった．両側の家も普通に人が住んでおり，われわれの他には観光客は一人もおらず本当に目立たない所であった．

中国近代文学の創始者であり，毛沢東をして「魯迅の方向こそ中華民族の新文化の方向である」といわしめた大思想家の住んでいた場所を，やっと苦労して捜し当てた感激はひとしおであり，しばらく門の前で感慨に耽っていた．

ガイドブックによれば，２階の彼の書斎は，昔のままに復元されており，壁に掛かったカレンダーは，彼の亡くなった１９３６年１０月１９日を示しているということだ．今年はちょうど，魯迅が亡くなって５０年めになる．中国も日本も，そして世界全体が大きく変化している．

１８．上海計算機ソフトウェア技術開発センタ

帰国予定日の朝８時半，上海計算機ソフトウェア技術開発センタを訪問した．

ここは，略称上海ソフトウェアセンタ（Shanghai Software Center：SSC）と呼ばれており，２年半前の１９８４年１月に設立されたとのことである．場所は，ホテルから車で２０分位西の，市街地からは少しはずれたところにあった．こちらは，岸田さんと私の２人，先方は Qiu 副主任の他，通訳を交えて６名が出席した．互いに自己紹介の後，SSC におけるソフトウェア開発プロジェクトの進め方を聞き，こちらからは日本におけるソフトウェア開発の現状等をお話しして，意見交換を行った．

SSC は，日本の協同システム開発（株）と同じような組織であり，２０の大学や研究所あるいは，公司を部門としてもっている．予算は，国家科学技術委員会と上海市の両方からきており，主要な仕事は，

・政策の立案
・新しいソフトウェア技術の開発
・開発したソフトウェアの管理
・ソフトウェアの評価
・ソフトウェア技術者の養成
・海外との合弁による開発
・ソフトウェアの輸入と輸出
・ユーザに対するサービス

の８つだということであった．

このような組織は，他の地方にも存在するが，重点は北京と上海であり，その意味でも，国がかなり力を入れているとのことであった．

海外との協力は，ソフトウェアの共同開発の形で始まったが，今までの実績としては，アメリカと２つの協力があり，アメリカの専門家の要請を受けて，中国で実際の開発作業を行ってきた．これは，アメリカの専門家からも高い評価を受け，国際競争力もあると評価されている．これらの実績を踏まえて，アメリカとは合弁会社を設立するところまでいっている．

今後日本とも合弁会社方式で協力してやっていきたい．そのために，現在日本に技術者を研修生として派遣もしており，準備のための努力をしているところである．言葉の違いについては，英語を共通語としてやっていきたいが，ここ数年は日本との協力を考えて日本語の判る技

術者を養成中であり，少なくとも一つのプロジェクトに一人は日本語の解かる技術者を置くつもりであると，日本との協力に関してかなり意欲的に活動している様子であった．

われわれとしても，何か御互いに協力できることを見つけることについては異存はなく，とりあえず，来年あたり上海でソフトウェアの日中協同のシンポジウムかワークショップを開こうではないかということで，意見が一致した（この件に関しては，その後１０月に先方が来日されたおり，ＳＥＡセミナー委員長の杉田さんや事務局の中島さんを交えて，東京で再び話し合いが行われ，来年１０月開催を目標に企画を具体的に進めることになった）．

Qiu さんが中国語以外話せなかったこともあり，通訳を介して日本語と中国語での話し合いだったため，３時間ほど話したが実質は一時間半程であったと思われる．昼前にミーティングは終わり，SSC の招待でホテルの特別室で昼食会が持たれた．この時には上海市科学技術委員会の Cheng さんも同席され，歓談しながら楽しい食事をした．

この時の上海料理は，今まで我々が中国で食べた料理の中では，最も豪華であり，紹興酒もとてもおいしかった．話はつきなかったが，飛行機の出発時間の関係もあって，あまりゆっくりしているわけにもいかず，１時過ぎには，名残を惜しみつつ，お別れせざるを得なかった．

空港までは，Qiu さん，祝さんが一緒に車で送って下さった．空港までの間，祝さんは我々に上海が戦前まで７つの欧米諸国により分割されており，それらは租界と呼ばれる地域を構成しており，今でもその名残が沢山残っていること等を英語で丁寧に説明してくださった．また，アメリカには行ったことがあるが，日本にはまだ行ったことがないので，是非行きたいということや，中国はこれからもっともっと海外と協力関係を深めていく必要があることなどを熱心に話された．

話をしている内に，車は空港に着いた．国内線とはちがう立派な建物であった．日航の窓口で搭乗手続きを済ませ，荷物をあずけた後，送っていただいた２人の方と，日本での［再見］を固く約束して御別れをした．出国審査も，聞いていたような，うるさいことは全くなく，ほとんどフリーパスであった．

免税品店も，出発ロビーも明るく奇麗で，上海についた時と同じ空港とは思えない程であった．時間が少しあったので，免税品店で少し買い物をした後，残った中国元を日本円に交換してもらった．日本円を中国元に換える時と同じで，書類にパスポート番号と金額を記載するだけで，直ぐに交換してくれた．

空港内はかなり人も少なく，免税品をいろいろ覗いて歩いたが，外国の酒を除けば物の値段は一番高いような気がした．

３時少し前，ゲートが開かれ，我々は１０日間の中国の旅を終え，機上の人となった．

１９．中国を後にして

上海をでた飛行機は，九州の上を通り２時間半程で大阪空港に到着した．ここで，若干の乗る人／降りる人があったが，やがて夕闇せまる中を成田に向けて飛び立った．

静岡近辺では，左手の雲一つない夕焼けの中に，真っ黒な富士山がくっきりと浮かび上がり，今度の旅のフィナーレを飾るに相応しい，非常に美しい光景であった．同時に，日本に帰って来たことが，実感として湧いてきた．

［後記］

今回の旅行では，非常に多くの方々の御世話になりました．ヨーロッパやアメリカとはまったく異なった文化圏でのさまざまな体験は，今後の私の人生においていろいろな教訓を与えてくれるものと確信しております．さらに，私自身が微力ながらも，中国と日本のソフトウェア技術の交流の橋渡しになれれば，この上ない幸せに思います．今後，機会がありましたら，そのための努力を続けていきたいと思っております．

今回の会議や旅行で御世話になった，中国ソフトウェア技術開発センタの Liu さん Zhong さん，西北大学の Hao 教授，上海ソフトウェアセンタの Qiu さん，さらにわれわれの案内役をして下さった，Jin さん，Ge さん等々の中国の方々に紙面を借りて，心から感謝の意を表します．

最後に，今回の国際会議への出席を快く認めてて下さった日本電子計算（株）の役員の方々，会議への出席をアレンジして下さったＳＲＡの岸田専務に，改めて深く御礼を申し上げます．

幹事会報告

昭和６１年度第６回：１１月２０日（木）
会　場：機械振興会館
時　間：１８：３０～２１：００
議　題：

１．新幹事紹介

前号でもお知らせしたように，幹事会をより充実させるために，幹事の増加をはかりつつある．ＣＡＩ分科会が近々正式に発足することから，世話人をしている大木幹夫さんに新幹事をお願いすることになった．

幹事の定数は，会則に定められているが，その数に余裕があるうちは，新しい分科会・支部の結成にともなって，その世話人を幹事会のメンバにすることが基本的に承認された．

２．メイル類の発送代行

前回の幹事会で，事務局から会計の中間報告があり，そのなかで，会員（正会員，賛助会員）の増加がないと財政は赤字になることが判明している．そこで，会費以外の収入を得る手段の１つとして，各種メールの発送代行を事務局でおこなうことが承認された．

やり方は，ＳＥＡＭＡＩＬとの同封または協会所有のラベルを使用した発送代行という形をとる．会員ラベル自体の販売は当分おこなわない．

３．講習会の実施

収入の増加および協会の広報（会員増加に効果がある）のために，セミナー，ワークショップ以外に，ＳＥＡ主催の講習会を開催する提案がされ，７月に行われて好評だったＡＩ向き言語の講習会を近々実施することになった．

４．年会費の更新

ＳＥＡが正式に発足してから，約１年が経過しようとしている．したがって，今月から年会費の更新が順次おこなわれるわけだが，振込手続きが面倒だという声もちらほら聞こえるので，現金書留でも受付られるようにすることが提案され，次回の会員更新案内には，そのような主旨も盛り込むことになった．

（記：野辺）

ＡＩ分科会(SIGAI)

来年１月の月例会は，休会とさせていただきます．なお，２月からの予定は以下の通りです．

第７回月例会

日時：昭和62年2月12日（木）19:00-21:00
場所：機械振興会館 B3階 9会議室
テーマ：未定

第８回月例会

日時：昭和62年3月12日（木）19:00-21:00
場所：機械振興会館 B3階 9会議室
テーマ：「Smalltalk のアプリケーションへの応用」
　　その効果と問題点

ＡＩ分科会の月例会は，当面のテーマを，「ＡＩ技術のソフトウェア開発への適用」にして，講演，討論会形式で，毎月第２木曜日に開催しています（ただし１月は休会です）．参加のための特別の資格はいりません．また，参加するための手続きもありませんので，当日会場に直接おいで下さい．連絡および問い合わせは，協同システム開発(03-503-4981) の野辺良一まで．

管理分科会(SIGMAN)

次回会合予定

日時：１月２１日（木）１９：００～２１：００
場所：協同システム開発会議室（予定）
テーマ：「未定」

問い合わせは，沖ソフトウェア(03-454-0020-ex-3134) 芝原雄二まで．

ＣＡＩ分科会(SIGCAI)

コンピュータを使った教育を，実践的に研究しようという目的で発足する予定の分科会です．

現在，この分科会に興味のある会員に対してアンケートを発送し，それを回収中です．それを集計して，運営方法，世話人を決め，来年１月中にアンケートに回答くださったかたを中心に第１回の会合を開催する予定です．ＣＡＩに興味のあるかたは，是非ご参加ください．

問い合わせ先は：

大木幹雄（ JIP-03-668-1322 ）

法的保護分科会(SIGSLP)

第７回の研究会を以下の要領で開催致します．
日時：昭和６２年１月１３日（火）

１８：３０～２１：００

場所：ＳＲＡ会議室（予定）

議題：IBMの提唱するEI(External Infomation)の内，コマンドの文法について検討をしてみる

ミニコンから大型機のOS分野コマンドの表記とその機能を持ち寄り，コマンドの文法が著作権の保護対象として相応しいかどうかを検討します．当分科会には，弁護士を職業とする会員も出席していますので，専門家の意見をきくことが可能です．

SIGSLPへの問い合わせ等は，能登末之(SRA:03-234-2611)まで．

環境分科会(SIGENV)

第１３回月例会

日時：１月２１日（水）１９：００～２１：００

場所：機械振興会館

テーマ：未定

問合せ先：引地信之（SRA:03-234-2611）

環境分科会は，月例会として，第３水曜日に，ソフトウェア開発環境の諸問題の研究，討論をしています．参加は自由です．費用は，資料代，会場費として，１０００円程度をその場で集めています．

関西支部

来年１月からの各種会合，研究会の日程を以下のように予定しています．ネットワーク分科会，システム技術分科会のふたつの新しい分科会も発足します．

ＳＥＡ関西第８回研究会

日時：昭和６２年１月２１日(14:00-17:00)

場所：日本能率協会研修室

テーマ：パソコン・ネットワーク「ＣＯＭＣＯＭ」

倉橋（データブレーン）

アメリカにおけるソフトウェア開発環境の動向

佐藤（ＳＲＡ）

ＳＥＡ関西１月度月例会

日時：昭和６１年１月２８日(18:30-20:30)

場所：ＳＲＡ関西支社会議室

テーマ：「６２年度の活動方針について」

ＳＥＡ関西ＡＩ分科会

日時：昭和６２年２月４日(18:30-20:30)

場所：日本電子計算大阪支店会議室

テーマ：診断型エキスパート開発ツール「ＣＳＲＬ」

臼井（日本電子計算）

ビジネス分野におけるエキスパート・システム

高森（日本コンピューターシステム）

ＳＥＡ関西２月度月例会

日時：昭和６１年２月１２日(18:30-20:30)

場所：ＳＲＡ関西支社会議室

テーマ：未定

ＳＥＡ関西ネットワーク分科会

日時：昭和６１年２月１８日(18:30-20:30)

場所：日本タンデムコンピューターズ

テーマ：アパレル産業ＶＡＮの実例

ＳＥＡ関西システム技術分科会

日時：昭和６１年２月２５日(18:30-20:30)

場所：未定

テーマ：システム設計技術の現状と改善案

研究会・月例会・分科会を含めて，関西支部へのお問い合わせは，以下にどうぞ．

盛田政敏（KCS:078-391-8291）

臼井美義（JIP:06-488-6021）

横浜支部(Y-DOC)

１２月のＹＤＯＣは予定どうりＳＩＧＲＥＵＳＥ（再利用分科会）とのジョイントの忘年会です．話題も中身の濃いのを用意して，思いっきり愉しく過ごしたいと思います，奮ってご参加下さい．

来年１月からの予定は以下の通りです．

第１０回予定

日時：１月３０日（金）　１８：３０～

テーマ：『プログラミング環境の最新米国事情』

林　香　（ＳＲＡ）

場所：「神奈川県労働福祉センター」（横浜・関内）

連絡先：０３－３７８－８０１０

ｅｘ．６７２２　藤野

その他：ＹＤＯＣ新年会を兼ねる予定です

第１１回予定

日時：２月２７日（金）　１８：３０～

場所：「神奈川県労働福祉センター」（横浜・関内）

テーマ：未定

´86 データショウ見聞録

匿名子

１．はじめに

過日，１０月に東京・晴海の国際見本市会場でおこなわれましたが，ここでのセンセーショナルな話題を２つほど報告します．なお，文中のなかで意図的に（固有）名詞を変えてあります．読者のかたの知識ベースを活用して，正しいものに変換しながらお読みください．また，筆者自身の知識ベースの不足を，たくましい想像力で補った部分も何か所かありますので，そのへんも楽しんでいただけたらと思います．

２．IβMの驚異

某国の知識処理に関する国家プロジェクトは，数年前，マスコミの間でも非常な話題になりました．現在もそのときほどではないにしても，重要な発表がときどき見受けられます．今回発表されたIβM-309Xシリーズ（開発コード名：ミイラ）もそのひとつといえます．この発表の笑劇はすさまじく，３大新聞がこぞって取り上げるはずでしたが，現在遅れながらも進行中のΛプロジェクトとの関係から，通産省からの申し入れがあったものと思われ，一般にはひろくつたわっていないようです．この発表が国民に与える影響を考えると，妥当な処置のようでもあります．

その驚異的な事実とは，なんと「ＩＣＯＴ」なる隠れコマンドがあり，このコマンドによって，なんとIβMの大型汎用機が，

COMANND NOT FOUND
READY

というプロンプティングにより，Prologマシンになってしまうとういう，世にも恐ろしいものでした．さらに驚くべきことは（以前から噂はされていましたが），プログラマのプログラマによるプログラマのためのユーティリテーとして，世界的にも高い評価を受けて，熱狂的なファンをもっていたZAPに，J-Starのオプションが隠されていることです．これにより，いかなるマシンも，低級文房具FUDITU-HighTACとしてのコピー機のエミュレーターをすることが正式に発表されました．また，未確認の情報ですが，現在NtTが自社システムにこの機能を取り入れ，「ファクシミリアンＩＩ」として商品化するであろう，という観測が消息筋のあいだではなされています．

また，同社の東洋進出作戦の一環として，メモリ・ダンプ・リストの漢字化がなされました．これと並行して，ハングル語，アラビア語およびスワヒリ語への対応も，来年中には完了の見通しとか．このへんには賛否両論があり，学会などでも盛んに議論されています．現在Iβ Mは，この形式的日本語化されたデータを，理解しやすい日常会話表現にする翻訳システムの開発に着手しており，年内にはプロトタイプの発表ができる見通しだという噂があります．

３．SOMY ぐわんばる

家電メーカーのSOMYが8ビットのワークステーション OLDを発表しました．まだ詳細は十分につかめていませんが，現在一応の平静さを取り戻したこの世界に，あらたな波紋をなげかけています．

これはかっての SOMY の栄光を取り戻すべく，全社をあげて取り組んだ製品といわれ，とくにかってのベトナム戦争にアメリカ軍に採用されたような信頼性を得るために，ほこり，振動，熱，水等が悪条件であっても，正常に稼働すべく設計されているといいます．しかし残念ながら，筆者のみたかぎりでは，電源系やや問題を残しているようで，事実会場でも電源不足不安定のためか，完全に満足ができる程の稼働状況をみることができなかった． SOMY 側はこれに対して「一般にコンピュータはデモのときにしか稼働しないというジンクスがある．その点 OLD は，デモのときだけ機嫌が悪いというのが唯一の欠点で，そのことを紹介できただけでも，今回は大成功だった」と述べています．

この OLD のセンセーショナルなことは，価格が従来の８ビットのワークステーションの半分以下であることです．そのため，世の中のハッカーといわれる汚い子供たちが，十分に自費で購入することが可能で，さらには，自分の能力以上にマシンの性能を要求する彼らにかなりの満足をあたえそうな性能を持っていそうです．これによって，彼らが，学校の勉強をよけいにおろそかにして，ネットワークを張ったうえで，ゲームに熱中してしまうことが，世の良識派の技術者からは，指摘されています．

March 30 - April 2, 1987
Monterey, CA, USA

9TH Annual International Conference on SOFTWARE ENGINEERING

Formalizing and Automating the Software Process

The theme *Formalizing and Automating the Software Process* identifies the two essential activities for realizing the potential of Software Engineering. The first is to rigorously define software creation and evolution processes. The second is to create organizational and automated systems supporting these processes. This theme has guided the definition of a Technical Program organized into Process, Formalism and Automation Tracks, a Tutorial Program providing background information, and a Tools Fair in which a variety of available technology will be demonstrated.

Each of the tracks in the Technical Program includes major talks, papers, workshop reports and panels that collectively expose and integrate relevant, current activities. The Process Track concerns approaches to defining, measuring, assessing, modifying and evolving software processes. Automated and organizational systems supporting these processes are the subject of the Automation Track. The Formalism Track covers both the use of formalism within the processes and the rigorous definition of the processes, their constituent activities and the supporting organizational and automated systems.

The Tutorial Program precedes the other activities and provides in-depth, one-day introductions to a variety of technology pertinent to formalizing and automating the software process. The program covers database technology supporting software engineering, software development environments, knowledge-based programming environments, formal specification and verification techniques, and software reusability.

The Tools Fair includes demonstrations of both production and prototype tools. It also includes in-depth, technical presentations covering particularly interesting and innovative tools. The Tools Fair parallels the Technical Program and its activities are scheduled to allow attendance at both activities.

CHAIR
William E. Riddle
Software Productivity Consortium

PROGRAM CHAIRS
Robert M. Balzer
USC Information Sciences Institute
Kouichi Kishida
Software Research Associates, Inc.

TOOLS FAIR CHAIRS
Larry E. Druffel
Software Engineering Institute
Jack C. Wileden
University of Massachusetts

TUTORIAL CHAIR
Richard E. Fairley
Wang Institute

LOCAL ARRANGEMENTS CHAIR
William M. Murray
General Dynamics Corporation

SPONSORED BY:

The COMPUTER SOCIETY of the IEEE
Technical Committee on Software Engineering

ACM Special Interest Group on Software Engineering

Institute of Electrical and Electronics Engineers, Inc.
Association for Computing Machinery
Agence de l'Informatique
Association Francaise pour la Cybernetique Economique et Technique
The Fellowship of Engineering, United Kingdom
Information Processing Association of Israel
Israel Section of the IEEE
Singapore Computer Society
Singapore National Computer Board
Software Engineers Association of Japan

TECHNICAL PROGRAM

PROCESS

Software Processes Are Software Too
L. Osterweil

Understanding and Assessing the Software Process
W. Curtis, M. Dowson, W. Humphrey

Measuring and Modeling the Software Process
W. Ehrlich, T. Emerson, A. Goel, H. Mills, Y. Miyazaki, N. Murakami, J. Musa

Empirical Studies of the Software Process
participants to be determined

Processes for Achieving Dependable Software
participants to be determined

Some Existing Software Process Models
P. Feiler, G. Kaiser, D. Schwartz, D. Serbanati

Software Process Assessments and Improvements
M. Harandi, J. Kelly, M. Lubars, T. Nomura, D. Ogle, R. Ramnath, K. Schwan, S. Vasudevan

Software Process Management: Lessons Learned From History
B. Boehm, W. Royce and others to be determined

Approaches to Improving Software Processes
V. Basili, L. Druffel, J. Elmore, R. Fairley, K. Kishida, M. Lehman, W. Riddle, D. Rombach and others to be determined

FORMALISM

The Use of Formalisms in Software Engineering
D. Bjørner

Database Technology and Software Engineering
P. Bernstein, R. Fairley, N. Goldman and others to be determined

Formal Specification Languages
E. Berliner, K. Futatsugi, J. Goguen, B. Kramer, J. Mesequer, K. Okada, P. Zave

More Formal Specification Languages
P. Ladkin, D. Perry, G. Roman

State-of-the-Art in Transformation Technology
W. Feather, W. Scherlis

Practical Application of Specification Technology
participants to be determined

AUTOMATION

Automating the Software Process
V. Stenning

Artificial Intelligence and Software Engineering
D. Barstow, B. Boehm, C. Williams and others to be determined

Intelligence in User Interfaces: Is it Necessary?
participants to be determined

Supporting Version Control
D. Perry, J. Winkler

User Interface Support
C. Rathke, S. Reiss

A Sampling of Existing Environments
participants to be determined

Prospects for Future Environments
T. Cheatham, R. Taylor, C. Tully, T. Yokoi and others to be determined

Support for Evolution and Reuse
D. Embley, G. Fischer, S. Katz, A. Lemke, C. Rathke, C. Richter, K. The, S. Woodfield

Individual and International Property Rights: How Serious is the Problem?
R. Balzer, R. Banin, D. Bradin, P. Samuelson and others to be determined

TOOLS FAIR

31 March - 2 April 1987

The Tools Fair runs parallel to the Technical Program, scheduled to allow attendance at both activities. The purpose of the Tools Fair is to provide conference attendees an opportunity to view a variety of production and prototype software development tools, toolsets and environments from both the research community and industry.

Individual tools, integrated toolsets and full-scale environments will be demonstrated in an exhibition area, providing the opportunity to gain first-hand experience and informally discuss technical details. In addition, a sequence of special presentations will be devoted to the description and demonstration of the technical details of particularly interesting or innovative technology.

Applications for space within the exhibition area and proposals for special presentations must be received by 30 January 1987. Presentation proposals will be judged on the depth of technical content, freedom from marketing "hype," and suitability for illustrating interesting and novel aspects. Applicants for space in the exhibition area are advised that marketing activity will not be allowed.

To request exhibition space or propose a special presentation, contact either of the Tools Fair Chairs:

Larry E. Druffel ***(412) 268 7740***
Jack C. Wileden ***(413) 545 0289***

TUTORIALS

30 March 1987

The Tutorial Program includes several in-depth, one-day introductions to a variety of technology pertinent to formalizing and automating the software process:

Database Technology for Software Engineering

John Nestor
Software Engineering Institute
Richard Snodgrass
University of North Carolina, Chapel Hill

Software Development Environments

Peter B. Henderson
State University of New York at Stony Brook
Mark Ardis
Wang Institute

Formal Specification and Verification

Richard A. Kemmerer
University of California, Santa Barbara

Knowledge-based Programming Environments

David R. Barstow
Schlumberger-Doll Research

Software Reusability

John B. Goodenough
Software Engineering Institute

LODGING RESERVATION FORM

CUT-OFF DATE: 23 February 1987

PROCEDURE:

1. Send this form to:
   ICSE-9 Housing Bureau
   Monterey Peninsula Housing & Convention Bureau
   PO Box 1770
   Monterey, CA 93940
   Complete the charge authorization or enclose a check, for one night's lodging, payable to Monterey Peninsula Visitors and Convention Bureau.
2. Block reservations will not be accepted.
3. Phone reservations will not be accepted by either the bureau or the participating hotels/motels.
4. List a first, second and third choice; your reservation will be made at your most preferred, available hotel/motel.
5. You will receive your room confirmation directly from the hotel/motel.
6. The rates listed do not include room tax.
7. Suite requests must be made in writing to ICSE-9 Suite Request, MPV&CB, PO Box 1770, Monterey, CA 93940.
8. Notify the Monterey Peninsula Vistors and Convention Bureau, in writing, of any cancellations or changes in your reservation before 15 March 1987. After 15 March 1987, contact the confirming hotel/motel directly. Deposits will be refunded only with 72-hour cancellation notice.

(Please type or print)

Name: _____
Organization: _____
Address: _____
_____
Phone: ( ) _____

Room Preference: ___ Single ___ Double

Hotel/Motel Preference: please indicate your preference using 1st, 2nd, 3rd, etc.

	Single:	Double:
____ Casa Munras - standard	$65	$65
____ Casa Munras - delux	$75	$75
____ Doubletree	$95	$110
____ Hilton Inn Resort	$89	$99
____ Hyatt Regency	$95	$95
____ Monterey Beach - gardenside	$55	$55
____ Monterey Beach - oceanfront	$65	$65
____ Monterey Beach - hills view	$112	$112
____ Monterey Beach - cannery row view	$128	$128
____ Monterey Beach - ocean view	$144	$144
____ Monterey Beach - deluxe ocean view	$160	$160
____ Sheraton	$95	$105

Credit Card: __ Visa __ MasterCard __ Choice __ AmExpress
Credit Card Number: _____
Expiration Date: _____
Cardholder's Name: _____

Signature: _____

CONFERENCE REGISTRATION FORM

PROCEDURE:

1. Send this form to:
   ICSE-9 Registration
   Computer Society of the IEEE
   1730 Massachusetts Ave., NW
   Washington, DC 20036-1903
2. To guarantee your registration, complete the charge authorization or enclose a check payable to Computer Society of the IEEE.
3. Members of sponsoring organizations are eligible for the reduced Member Fees. Please indicate the sponsoring organization and your membership number when claiming these fees.
4. The Conference Fee includes admission to the technical sessions, admission to the Tools Fair, one copy of the Conference Proceedings (except students), refreshments at breaks, and receptions.
5. The Tutorial Registration Fee includes tutorial notes and lunch.
6. Refunds (less a $15 handling fee) will be made if a request is received in writing by 15 March 1987.

(Please type or print)

Name: _____
Organization: _____
Address: _____
_____
Phone: ( ) _____

TUTORIAL CHOICE (choose at most one):

___ Database (Nestor/Snodgrass)
___ Environments (Henderson/Ardis)
___ Formal (Kemmerer)
___ Knowledge-based (Barstow)
___ Reusability (Goodenough)

FEES:

Advance Registration (Prior to 25 Feb 1987)	Member	Non-Member	Student/Retired
Conference	$160	$200	$25
Tutorial	$125	$155	$50
Late Registration (After 25 Feb 1987)	**Member**	**Non-Member**	**Student/Retired**
Conference	$200	$250	$35
Tutorial	$155	$195	$60

Conference Fee: _____
Society: _____
Membership Number: _____
Tutorial Fee: _____
TOTAL FEE: _____

Credit Card: __ Visa __ MasterCard __ Choice __ AmExpress
Credit Card Number: _____
Expiration Date: _____
Cardholder's Name: _____

Signature: _____

ＳＥＡこれからの活動予定

－１９８６－

◆１２月(December)

２６日（金）

再利用分科・Ｙ－ＤＯＣ合同月例会

（神奈川県労働福祉センター）

－１９８７－

◆１月(January)

１３日（木）

第７回法的保護研究会（ＳＲＡ会議室）

２１日（水）

第１３回環境分科会月例会（機械振興会館）

関西支部第８回研究会（日本能率協会研修室）

管理分科会月例会（ＪＳＤ会議室）

２８日（水）

関西支部月例会（ＳＲＡ関西支社）

３０日（金）

第１０回ＹＤＯＣ（神奈川県労働福祉センター）

◆２月(February)

４日（水）

関西支部ＡＩ分科会研究会（ＪＩＰ大阪支店）

１２日（木）

第７回ＡＩ分科会会合（機械振興会館）

関西支部月例会（ＳＲＡ関西支社）

１２～１４日（木～土）

実践的ソフトウェア開発環境に関する集中討議

第２回 SEA ワークショップ in 長岡

（新潟県・長岡市）

１８日（水）

関西支部ネットワーク分科会

（日本タンデムコンピュータズ・大阪営業所）

２５日（水）

関西支部システム技術分科会（未定）

２７日（金）

第１０回ＹＤＯＣ（神奈川県労働福祉センター）

◆３月(March)

１２日（木）

第８回ＡＩ分科会会合（機械振興会館）

１８～２０日

春のセミナー・ウィーク（機械振興会館）

ＳＥＡ会員状況

昭和６１年１２月１７日現在の入会会員状況は以下の通りです．

正会員－－６７２名（先号比４３名増）

賛助会員－－２１社（先号比３社増）

＜地域分布＞	勤務	居住
北海道＝	1	1
岩手 ＝	4	4
新潟 ＝	5	5
栃木 ＝	4	3
茨城 ＝	3	6
埼玉 ＝	10	55
千葉 ＝	13	53
東京 ＝	448	267
神奈川＝	44	137
長野 ＝	9	9
富山 ＝	1	1
石川 ＝	2	2
静岡 ＝	4	4
岐阜 ＝	1	3
愛知 ＝	8	6
滋賀 ＝	1	2
京都 ＝	4	9
大阪 ＝	59	46
奈良 ＝	1	3
兵庫 ＝	20	27
福岡 ＝	5	5
熊本 ＝	6	6
長崎 ＝	1	1
鹿児島＝	3	3

＜男女分布＞

男 ＝ 632

女 ＝ 40

＜年齢分布＞

20-24 ＝ 38

25-29 ＝ 149

30-34 ＝ 184

35-39 ＝ 175

40-44 ＝ 66

45-49 ＝ 31

50-54 ＝ 11

55-59 ＝ 7

60以上 ＝ 5

会員の名簿管理は，パソコンを使用しておこなっています．住所，会社部署等の変更があったときは，なるべく早く事務局までお知らせください．

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