SEAMAIL

Monthly Newsletter from
Software Engineers Association

Volume 1, Number 3 | March 1986

目　次

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

現在のソフトウェア技術が抱える最大の課題は，ソフトウェア・エンジニアリング研究の最前線（ステイト・オブ・アート）と，その実践状況（ステイト・オブ・プラクティス）との間に横たわる大きなギャップを埋めることだといわれています．ソフトウェア技術の特徴は，他の工学諸分野の技術にくらべて属人性がきわめて強い点にあります．したがって，そうしたテクノロジー・トランスファの成否の鍵は，研究者や技術者が，既存の社会組織の壁を越えて，相互の交流を効果的に行うためのメカニズムが確立できるか否かにかっかっています．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．

ＳＥＡＭＡＩＬ　Ｖｏｌ．１，　Ｎｏ．３　　昭和６１年３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

定価　５００円

プログラマ定年説とＳＥＡの役割

辻　淳二
辻システム計画事務所

情報サービス業界においては、一方で「ソフトウェア技術者の絶対的不足」、他方で「プログラマ３５歳定年説」が話題とされている．これは大いなるディレンマであり，その着実な解消なしには，技術者の地位向上も情報化社会に向けての円滑な移行もむずかしい．ＳＥＡも，この問題解決に確かな貢献をなすことをその存立意義のひとつと考えて，正面から取り組む必要があろう．

「プログラマ定年」論議には，大きくわけてふたつの側面があると思う．ひとつは，ソフトウェア産業がユーザ産業各界の下請け的サービスから出発し，今なお大勢においてその域を脱しえていないという「産業としての受け身性」に由来するものである．もうひとつは，建築家やデザイナーなど，同じく専門技術に立脚する他の職種に比べて，まだ確たる専門性・社会性を確立しえていないという「専門職としての受け身性」に関わるものである．

この説の解消のためには，二つの側面をともに改革していかなければならない．前者すなわち「産業としての受け身性」を克服するカギは，業界各社が握っている．今日の需給関係のもとでは，仕事の受注に苦労することはないから，多くの企業が「大手ユーザまたはメーカーの安定的外注先」としての立場に安住しがちである．しかし，この姿勢をとり続ける限り，この説を「よその会社の話」とすることは難しい．規模の大小を問わず，経営資源に全力を蓄え組織力を備えた企業群が，「受け身的・下請け的受注形態の比率を引き下げ，リスクはあるが主導的な事業の比率を高める経営努力」に，それぞれ力を注ぐことが切望される所である．

後者すなわち「専門職としての受け身性」を克服するカギは，技術者一人一人が握っている．もちろん，その所属企業が顧客に対し自立性を保っている場合と，そうでない場合とでは，技術者が専門技術を磨く上でのベースが違うことは確かである．しかし，自立性を保っている企業に属していても，技術者の努力がなければ受け身の立場に甘んじることになるし，逆の企業にあっても，自ら鍛錬に努めていれば専門技術を伸ばし，駆使することが可能となる．

ＳＥＡがこの問題解決に向けて貢献するための視点は，いうまでもなくこのうち後者，すなわち「専門職としての受け身性」から一人でも多くの技術者が”自立”できるよう強力な後押しをすることであろう．

それでは，プログラマの専門性を揺ぎないものとするキー要件は何だろうか．専門性の高さは，専門家を必要とする重要局面における「余人への代え難さ」の度合いで測ることができよう．これに対し，ある人は専門分野に対する深さ・確かさで，また他の人は状況への幅広い適用性で，プロとしての力を発揮して応えようとする．このように，人それぞれにキラリと光る「持ち味」を創り出すことが求められるわけである．

社会の情報化が進みつつある今日，高度な情報システムは，社会及び企業の発展を支える仕組みとして，ますます重要なものとなってきた．これらのシステム安定性・健全性を高め，脆弱性をなくすという役割に，プログラマを含む情報システム技術者が深く関わりを持つことになる．このような時代ニーズを考える時、プログラマの専門性においても「取り組む仕事・役割と環境に対し，広く鳥かん的にとらえ，ゆとりを持って自らの位置きめができる力」が求められてくると考えられる．これまでの技術者像よりも一回り大きく，タフな技術力のみならず，全人格的な力の職務への投影が不可欠とされよう．

以上のような考察から，私は，ＳＥＡの活動が「ソフトウェア技術者の全人格的な能力開発」に焦点を当てたものとなることを念願している．ＳＥＡの活動がプログラマの人たちの自己啓発意欲を刺激し，会員の多くの人たちが努力してプロとしての確たる専門性を磨き上げることができ，さらにプログラマの仕事の中でルーチン的な作業を自動化するための技術向上に貢献できれば，３５歳定年説は影の薄いものとになり，同時にソフトウェア技術者不足も解消に向かうのではないだろうか．ＳＥＡが，この流れに向けてのテコの役割を果たせるよう，私も微力を尽くしたいと思っている．

プログラマ定年説をめぐる誌上討論

プログラマ３０代全盛期説

佐原　伸

１．根拠のないプログラマ３０才定年説

「プログラマの定年は３０才だ」という俗説がはびこって久しいが，その根拠となるデータが示されたことはない．せいぜい，

（１）３０才になると，そろそろ徹夜がきつくなる．

（２）持っている技術が陳腐化し，新しい技術についていけない．

（３）コーディングのような低レベルの仕事を，３０才過ぎてもやっているのはもったいない．

といった，一見もっともらしい理屈が下敷きになっているだけである．

しかし，これらのうちで，（１）は「徹夜業務３０才定年説」であるに過ぎないし，（２）は「勉強しないプログラマはそのうち駄目になる」というだけの話で，別に年齢と関係ない．（３）にいたっては，まったく間違った説であって，ほんとうは「コーディングのように高級な仕事を，３０才前の連中にやらせるのはアブナイ！」というのが正しい．

「低レベルなコーディング」とは，本来自動化すべき機械的な作業であり，現在すでに適切なツールを使いさえすれば自動化できるようなものを指す．しかし，本来のコーディングとは，人間の言葉で表現されたアルゴリズム的思考をコンピュータが理解できるかたちに変換するという，高度に抽象的な仕事なのである．

また，プログラム設計を行ってからコーディングをするという考え方自体ももはや時代遅れになりつつある．そうしたウォータフォール型のモデルでソフトウェア開発プロセスをとらえることは，ＣＯＢＯＬやＦＯＲＴＲＡＮなどという「低級」な言語を使っているときは，やむを得ない面もあるが，ちかごろの「よい」言語を使う「よい」プログラマは，コーディングをしながらプログラム設計をしているのである．

結局のところ，３０代プログラマの人数が現在あまり多くないのは，技術的な問題というよりは，３０才になると自動的にマネージメントの仕事をやらせるという日本の会社制度の方に問題があるように思われる．打ち合わせや電話による中断が頻繁に起こるような（つまり管理職的な）環境では，どんな天才プログラマでも，たいした業績を残すことはできないだろう．

２．２０代プログラマに創造力があるか？

プログラマ定年説の裏返しとして，「２０代のプログラマは創造力が豊かだ」という説がある．このことは，ごく一部のプログラマについては正しい．ＶＩＳＩＣＡＬＣやＳＭＡＬＬＴＡＬＫあるいはＭＡＣＩＮＴＯＳＨのソフトウェアは，たしかに２０代のプログラマたちの作品だ．しかし，これらのプログラマは，世界中に何百万といる２０代プログラマのうちでほんのわずかな，例外的な天才にすぎない．ほとんどの２０代プログラマは，いわゆる「低級なコーディング」で毎日をすごしているだけなのである．

３．３０代プログラマに栄光あれ

以上の考察により，「プログラマ３０才定年説」はまったく根拠のない説であり，むしろ「プログラマ３０代全盛期説」の方が有望なことが分かってきた．つまり，

－３０代プログラマは，先人が作ったプログラムを収集し，流用するすべを知っている．

－３０代プログラマには，自分の作業環境を整える力と金がある．

－３０代プログラマは，失敗を避ける経験と能力を豊富に持っている．

ということになる．

あとは俗説を完全に打破するために，３０代プログラマ全盛期説を流布するという作業が残っているだけである．ところで，敬愛する４０代や５０代の先輩プログラマ諸兄姉は，この問題についてどのようにお考えだろうか？

（野村コンピューターシステム）

プログラマ定年説の図式

藤野 晃延

1．その歴史的展開

この問題はなかなか難しい．しかし，現状ではそうした説が「存在」していると認めざるを得ない．

なぜなら，１つには経済的理由，つまり商売の観点から眺めたときに，現在のこの業界の若さから，当然のことながら１人当たりの付加価値，換言すれば売上高などタカが知れたものであること，２つめには一般の社会通念からして，いわゆる年功序列的な考えが根強いこと，およびソフトのような可視的でないものにオカネをかける習慣が少ない，といった事実が根底にあり，このことから

－年取ったプログラマは給料が高いからあまり歓迎しない．
－単価の高いベテラン・プログラマ１人よりは単価の安い並のプログラマを２，３人使った方が得策のようだ．
－それなら若いプログラマたちを使おう．
－若いプログラマに対するニーズが高くなる．
－若いプログラマが増える．
－相対的に年取ったプログラマの数が少ないように見えてくる．
－若いということがプログラマの特徴のように思える．
－プログラマは若いうちの仕事のように見える．
－プログラマの定年は３０才前後だろう．
－プログラマ定年説の誕生．

と，おおよそこのような過程で今日の「プログラマ定年説」ができあがったのではないかと推測される．

2．肯定の図式

それと同時に，いわゆるスーパープログラマないし年配のベテラン・プログラマたちが持つ一種独特の，よくいえば個性的，ハッキリいって風変わりなイメージが，より一層社会との距離を広げるような形で働いて，せっかくのすぐれた仕事の成果の付加価値が正当に評価されず，そのためにかれらの仕事は割高であるかのような印象を与えていることも，定年説に追い打ちを掛ける効果をもたらしている．

つまり，

－何だかよくわからないけれどスゴイものを作ってくれたようだ．
－しかしかかった費用も馬鹿にならない．
－確かにいいものを作ってくれたようだが，これでなくてはいけないのだろうか．
－それにしてももう少し愛想よく説明してくれないものだろうか．
－ソコソコのものができるなら，もっと愛想のある者を使ったほうが気が楽だ．
－次回は普通のプログラマに任せてみよう．
－多少のトラブルや稼働時期の遅れはあったが，まあまあなんとかなった．
－特にベテランを使わなければいけないということはないようだ．
－プログラマ定年説を否定はしない．

という図式が成立しているかのごとくに思われる．

そして，さらに問題なのは，当のベテランたちがこの説に対して反対の意志を明確に表わず，得意の無関心をよそおっているため，あたかもかれら自身がその説を肯定しているかのように解釈されていることである．

そうしたプログラマたちの態度が，

－少なくとも狭義のプログラミングにはベテランを使う必要性は少ない．
－要求分析や設計などの時点では豊富な経験を積んでいるベテランの方が望ましい．
－それらの人間にはそれなりの職名が与えられるので，たとえ本人がどんなに主張しようとも，一般的には最早プログラマとは見て貰えない，

という一種の既成概念を社会的に流通させる働きをし，今日の不孝な状況をもたらしたのだと断言してもいいだろう．

3．もはや手遅れ？

私は，個人的には，広義のプログラミングとは人間の本質的な頭脳活動そのものを指し，したがってプログラミング言語やコンピュータの使用とは無関係に，いつでも，どこでも，だれにでもできる仕事（この場合作業の質は問わないこととする）だと考えている．したがって，医学的にボケてしまわない限り定年などありえないのだが，今日ではすでに上記のような既成概念が一般に浸透してしまっている関係上，「定年説」は存在しているとの肯定的立場をとらざるをえない．もはや遅れというところか・・・

（三和エレクトロニクス）

余計な御世話

沢田 寿実

１．はじめに

最近の「プログラマ定年説」がどのような主張のされ方をしているのかはよく知らないが，いずれにしても余計な御世話といいたい．そのうちでも特にひどいのは，「経験２～３年が最盛期で，あとは燃えかす」といういい方で，ナンセンスを通り越して，完全にわれわれプログラマを馬鹿にしているとしか思えない．定年説が出現する背景には，まだプログラミングという仕事が，固有の評価尺度を持つ技術あるいは職業であり，才能を必要とするものだとは認められていない現状がある．

本稿では，定年説の根拠と称して従来からよくいわれていたもののうち，代表的と思われるものを取り上げて反論する．プログラマを高度な専門職と認めるならば，そもそも定年説などあり得ない．しかしこの場合には，「いつプログラマを引退するか？」という純粋に個人的な問いかけが意味を持って来ると思われる．これについても少し述べたい．

２．定年説の神話

（１）生産性の頭打ち

これは「経験２～４年の間に生産性が急激なカーブを描いて上昇し，以降は目立って伸びない」という経験的事実（？）をもって，定年説の根拠とするものである．経営者やマネジャーにとって，伸びの目立つしかも給料の安い若年プログラマに魅力を感じさせる要因ともなっている．

しかし，この単純なモデルにはいくつか問題がある．生産性を何で測るかというのが，まず非常に難しい問題である．これを無視するとしても，人間の持つ能力の限界からいって，どこまでも生産性を伸ばすなどというのはそもそも無理な話だ．また，一般に年長のプログラマほどより高度な問題に取り組むわけであるし，管理上の雑用を含めた直接数字には現れない部分に，多くの努力を注いでいる．

一番の問題はプログラマの作りだす成果物の質に関する視点が欠けていることである．多分このような主張をする人々は，すぐれたプログラマたちがどれだけすばらしい仕事をするか御存知ないのだろう．しかも，そのためには，深い知識と少なからぬ経験が必要なのである．

（２）現実に高年齢のプログラマは少ない

３０代半ばを過ぎて，現役ですぐれた仕事をしているプログラマは確かに少ない．だからといってこれを定年説の動かぬ証拠というのはどうか．

この現象には，多くの要因が複雑にからまり合っているが，基本的には，現実が，そのようなプログラマが育ちにくい状況になっているということである．決して年齢による限界を裏付けるものではない．

まず，企業に努めているプログラマにとっては，キャリア・パスとして（暗黙的にせよ）ある程度の年齢になったら生産現場を離れ，スタッフや管理職となって行くことが，ノーマルとされている場合が多い．日本の風土や業界の体質など多くの理由が考えられるが，背景には，プログラマが高度の専門職と考えられていないという事情がある．このことは，プログラマとして一流になろうと思う者の気を挫く要因にもなっている．

一流のプログラマになるには時間がかかる．例外的な人がいることは認めるが，５－６年で一流と言われても困るのである．確固とした開発技術がいまだに存在しないソフトウェアの世界において，それだけの期間でプログラミングのすべてを学びつくすなどということは，そもそもあり得ない．われわれの世界はオープン・エンドであり，今後もまだ，どこまでも新しい技術の探究がつづき，プログラマの仕事もそれとともに変化し，発展して行くだろう．

それとも，一流のプログラマなんか必要ない？！

３．いつ引退するか

プログラマに定年はない．引退を考えるとしたら，それは純粋に個人的な問題である．つまり引き際があるとすれば，いつそれと知るかということになる．

私にとって，それは仕事に興味を失ったとき，学ぼうとしなくなったときである．過去の成果を充分に学ぶ間もなく技術的な進展が急速に進む世界のなかで，つねに自らの精神をリフレッシュして学んで行くことは，やはりしんどいことである．しかし，「好奇心がまさっているうちは何とか」と考えている．今のところは，プロフェッショナルなプログラマになりたいと思っているのである．

いささか青臭い議論となってしまって気恥ずかしいが，正直な考えを述べさせてもらった．

（ソフトウェア・リサーチ・アソシエイツ）

私にとっては先のことだが・・・

渡辺雄一

1．プログラマとはあたかも・・・

「プログラマに定年はあるのか？」という問いかけの「プログラマ」ということばを「タクシーの運転手」と置き換えてみても，あまりおかしくないように思います．

タクシーの運転手は，似ているようで全く同じではないそれぞれの車のハンドルを握り，ときには夜も（人によっては夜のみ）頑張っています．会社員として仕事をしている人，１匹狼（個人）で営業している人，コラム・シフトの３速でダブル・クラッチングでエンジンをうならせている人，それとは対象的にオートマチックで楽々と運転している人・・・

それらはプログラマの世界とそっくりです．

大型汎用機でオンライン・データベースと格闘している人，アセンブラとダンプリストから逃れられない人，ＦＯＲＴＲＡＮ７７からＦＯＲＴＲＡＮＩＶへのコンバージョン（ユーザの望むことであれば，ときには時代に逆行することもある！）をしている人がいるかと思えば，かたやパソコンでグラフィックスやイメージ処理に没頭する人，ＰＲＯＬＯＧでエキスパート・システムの構築に情熱を傾けている人・・・

2．どこが違うか？

プログラマとタクシー運転手との大きな違いは，その環境の変化の度合ではないでしょうか．

実際，いままで数多くのコンピュータが作られ，そしてそのためのソフトウェアが作成されてきました．タクシーの運転手は，ほとんど同じ車ならなんなく乗りこなしてしまいます．ところが計算機の困ることは，ほとんどそっくりの言語を設計しても，ちょっとだけ違う部分で多くの人々がつまづいてしまうことです（それゆえ言語の標準化がいつも叫ばれ続けられている）．

またタクシーの運転手は，自分で自分の環境（車）を運転していってお客さんをのせる（古くて汚い車にはだれも乗ってくれない）のですが，プログラマの場合は，自分の環境をお客様に押しつけることは，大抵の場合できません（古い計算機がいやでも，ひたすらそこで仕事をしなければならない）．

3．定年は存在するか？

計算機にカードを読ませてリストをとっているころは，まだよかった．ＴＳＳやパソコンが普及するにつれ，環境は飛躍的に進歩したように見えました．

しかし，そこに登場したエディタに代表される数々のツール群は，多くのプログラマを喜ばせ，かつ苦悩の毎日へと誘ってきています．なぜなら，今日のプログラマには，単にプログラム言語に習熟することだけでなく，ターゲット・マシンやワークステーション上で，エディタやその他のツールを充分に使いこなすことが要求されているからです．

苦悩の毎日を送っているプログラマにとっては，時代の先端をいくテクニカルな話題も，ときには自分（や自分の仕事をしている環境）への皮肉にしか聞こえない．いや，そのように感じて，プログラムを組む仕事に疑問を抱き始めたときに，その人がプログラマとして定年を迎えたか否かが，はっきりするのではないでしょうか．

そのとき逃げ出してしまった人はもう定年なのです．これは個人の能力のみでなく周りの環境に依存することも大きいと思いますが，そのようなことはプログラマの世界だけでなく，ごく一般の社会常識と思われます．

一方，その場をなんとか改善する方向（それはときにはその場限りのプログラミングであったりすることもあるが）で対応して，次のステップにそれを自分の糧としていける人が，ほとんど定年を知らないプログラマなのではないでしょうか．だから人によっては定年は存在しない・・・？

4．やっぱり定年は存在する

定年ということを議論するからに，職業人（プロフェッショナル）としての具備すべき要件を考えてみると，納期までに仕事を仕上げるということが思い浮かびます．

プログラムを作るのが速いか遅いかは，絶対的な尺度ではなく他人との差という相対的尺度で議論されることが多いので，個人の趣味としてプログラムを作り楽しむことがたとえ死ぬまで出来たとしても，ある一定の期日内で仕事を成し遂げるのは，当然歳をとればつらくなってくると思います．

自分の「プログラマ」としての定年が一体何才ぐらいにやってくるのかは，分かりません．まだ私は２６才だ！それでも，いやそれだからこそ，そのときがくるまでに，納得できる仕事を一つでも多くなしとげたいと考えています．

（電力計算センター）

消極的賛成と絶対的反対

山内 徹

1．定年説への疑問

私は，世間で言われる「プログラマ定年説」について，「消極的」に賛成の立場です．しかし，従来のこの種の議論にはいろいろな問題があり，今後はもっと別の視点から検討されるべきだと考えています．

第一に，プログラマの定義が不明瞭であること．「定年説」がいつ頃から唱えられるようになったのか，私にはよく分かりませんが，時にプログラマが広義に解されて，ＳＥ等を含めたソフトウェア技術者全体に定年があると誤解されているような気がします．このような誤解が社会全体にひろまっているとすれば，それは，ソフトウェア業界にマイナス・イメージを与えこそすれ，決してプラスにはならないでしょう（なお，プログラマを広義に解しての定年説については，私はもちろん「ノー」です！）．

第二に，今日のシステム開発状況を考慮していないこと．「定年説」の理由は，プログラム開発に必要な注意力やち密さ，さらに次々とでてくる新しいハードウェア機器への適応性等の能力が，年齢とともに衰退するということにあると思います．しかし，私は，現在ではこの理由は，定年説の消極的理由にしかなりえないと考えています．すなわち，今日のようにプログラム開発の生産性が重視され，そのために属人性の排除を目的とした「プログラムの標準化」が推進されている状況下では，プログラム開発に必要な注意力やち密さは，以前のように職人的プログラマに依存していた時代に比べて低下していると思います．また，新しいハードウェア機器への適応性は，確かに年齢とともに低下することも考えられますが，その新機種の操作が容易でかつプログラム開発の生産性向上に役立つものであれば，導入時の一時的混乱はあっても，長期的に適応性が欠けるとはいいきれないと思います．新しいハードウェア機器への適応性については，年齢差とともに個人差もまた大きいのではないでしょうか．

以上のことを考えると，私は従来の「定年説論議」が，ソフトウェア産業やわれわれソフトウェア技術者の今後の成長と発展のために，どれほどの意義があるのだろうかという疑問を感じます．

いいかえれば，ソフトウェア産業が，大手ユーザ企業等のプログラム開発のための下請けではなく，システムの大規模化やライフサイクルの長期化に対応して，開発の生産性・信頼性向上やメインテナンスの改善などの課題に前向きに対処し，より付加価値の高いサービスを提供する産業となるために，「プログラマ定年説」という概念をいかに役立たせるかという視点からの見直しが必要だと考えています．

2．プログラマからＳＥへ

今日のシステム開発の課題は，すでに述べたように，システムの大型化と，メンテナンスやソフトウェア技術者の不足を考慮した上での，システム開発の効率化にあります．そして，これらの課題に対処するために，近年では，システム開発過程の中でもシステム分析・設計のフェーズが重視され，プログラム開発工程では誰でも容易に作業できるような「標準化」が叫ばれていることは，周知の通りです．つまり，システム開発の成否に対してＳＥの重要性がより高まるとともに，ＳＥの絶対的不足がさらに助長されることになります．そこで，私は，プログラマは一定期間をへた後はＳＥに成ることが望ましく，またこのような視点から「プログラマ定年説」の内容が再考されるべきだと考えます．

プログラマとＳＥの違いは，プログラム単位の仕事をするのかシステム全体の仕事をするのかにあり，このことからよく両者の適性の相違が指摘されています．特に，ＳＥには開発対象システムの業務知識が必要であり，現実問題としてプログラマ定年後のＳＥは業務知識の修得にある程度の努力を要しますが，私には，これが決定的な問題になるとは思われません．

現にＳＥとして従事している人でも全員が当初からその業務知識を有していたとは思えないし（それゆえに，ユーザとＳＥのコミュニケーション・ギャップがときどき問題になる），そもそも多くのソフトハウスでは業務知識の重要性そのものが十分認識され，また教育されているとは思われないからです．

3．プログラマ定年説を死語に

最後に，「プログラマ定年説」が従来のように人間の能力的側面からのみ議論されるのであれば，それは社会に対してわれわれソフトウェア技術者のマイナス・イメージを与えることが多く，もはや死語にすべきであることを強調しておきたいとおもいます．

（ビジネス・ブレイン昭和）

個人の意識が定年を変える

長井 修治

１．プログラマの寿命

日本人は，妙なところで物事にこだわるところがある．最近流行り出した「熟年」や「実年」などのことばもその一つで，これまでは「中・高年」と呼ばれていたものだ．今回取り上げる「定年」も，まさにこの年代になって訪れるものである．では何故この年代でなければいけないのか．答はいくつもあるだろうが，やはり老化にスポットを当ててみたい．年を取っても第一線で活躍している人は沢山いる．現にトップと言われる階層はおしなべて中高年かあるいはさらに年配である．政治家などは，４０代，５０代で新人だという．まったく奇妙なものである．

前置きはこの位にして，本題へ移ろう．まずプログラマーのスタート時期は，大体２０才とみてよいと思う．そして定年は，諸説あろうが高い方で３５才をとりたい．とすると活動期間は１５年ということになる．サラリーマン１５年といえば，まず間違いなくその企業の要を形成しているといってよい．その絶頂期にあたって定年，ボイ．それではあまりに悲惨というものだ．

たしかに，スポーツの世界などでは肉体的な衰えから来る定年は考えられる．ではプログラマーの世界での衰えとは何だろう．鉛筆が持てなくなるということでもあるまい．若者の誤字・脱字を考えると，むしろ数段上を行くに違いない．外見的にはマイナス要素は見つかりそうもない．とすれば，これはもうプログラムを作ること，それ自体に原因が潜んでいると考えざるをえない．

２．プログラミングは寿命を縮める

人の寿命をそんなにも簡単に縮めてしまうプログラミングとは，いったい何だろうか．よくいわれることに，創造するには頭がかたくてはダメ，つねに若者のように柔軟でいろいろな発想が泉のように湧きでてこなければという．私も硬直化してはダメだと思うが，それが即年齢と結びつくとは考えられない．

発想の豊かさというものは，天才ならいざ知らず，私は引き出しの中味で決まると考えている．個人がそれまでに体験し，修得してきた実にさまざまな内容が物をいう．それらが土壌としてあって，初めてその中からでてくるものだ．ボケーとしていては若者も，名実共に「若年寄」になってしまい，役には立たない．あくまで，本人の意識の問題である．いかに活性化するか，それはいってみればＶＢを起こすようなアントルプルヌリアルな姿勢を絶えず持ち続けることであり，そこにこそ創造する力が秘められていると思うのだ．

素粒子の分野で励起と言う概念がある．これはある系にエネルギーを与えることにより，その系を低い状態から高い状態へ遷移させることをいう．創造の世界では頭脳というカオス的倉庫にエネルギーを与えて，その中に蓄えられているものを取り出すという風にはいえないだろうか．とすれば，貯蔵量は年配者の勝ち．個人差はあるのだろうが，情報の蓄積量は，流通経路（チャネル）の数と時間量がものをいう（若い人，ごめんなさい）．それからその取りだし方，これが実際のポイントになる．いくら中味があっても，その中から創造物という結果を導いてくる手段がなければ何もならない．

アインシュタインという学者を知らない人は少ないと思う．彼が３５才で定年になったとはとても思えない．その彼が亡くなった時に頭脳を調べたところ，無数のひだが刻みこまれていたそうだ．赤ん坊の脳はツルツルしている．人間死ぬまでに，頭は１／３しか使わないという．そして，よく頭を使う人ほどこのひだが多いそうだ．乱暴な展開だがこのひだをシナプスと置き換えてもいいと思う．シナプスは細胞を連絡する構造で脳細胞を正にＶＡＮの網のように繋いでいる．このシナプスこそが，創造物という結果を取りだす手段だと思うのだ．

人間の脳細胞はどんどん死滅する．が，逆にシナプスはどんどん増加する．若者がすぐれているといわれるのは，単にこの細胞の数が多いことであり，空き倉庫が沢山あるにすぎない．その中に物を入れ，さらに頻繁に取りだすメディアの数は，時間に依存する．算術的表現を使わせてもらうと，細胞をＮ，シナプスをＭとして，$dN/dt<0$，$dM/dt>0$．多少余談になるが一説によれば $N(t=0)=140$億個，$dN/dt=-10$万個，他に，酒だと $dN/dt=-1$万個，タバコは$dN/dt=-500$個／本，コーヒーは$dN/dt=-300$個／杯だそうである．

結局のところ，プログラミングを創造の必要な分野と定めると，そこでの定年は肉体的なものではなく，このシナプスをいかに増加させるかである．たえずエネルギーを供給し続ける姿勢が重要なのである．個人の意識次第で，定年などどうにでもなるのではないだろうか．

（大興ファコム・データー・センター）

会員の声

以下に収録した会員の声は，本誌第１号で公募した＜プログラマ３０才定年説＞に対する会員アンケートの結果を，到着順に編集したものです．
ちなみに，「プログラマには定年がある」という意見が９件（４５才－１，４０才－３，３５才－４，３０才－１），「定年はない」という意見が１４件でした．

日比野修三

プログラマの定義が明確になされていないので判断しにくいが，プログラマがコーダーと同一語で使用されているのならば，３０才ぐらいが限界であろう．しかし，プログラマがプログラムの論理設計までやると考えるならば，定年はないと考える．プログラマが論理設計までやると考えると，

・ルーチンの作成
・テスト・ジェネレーターの作成
・プログラム・ジェネレータの作成
・使用言語の決定
・業務知識の探索

等，中高年のプログラマのやることは，実に多いと考える．

福島勲

会社がむりやり管理職にしないかぎり，仕事を続けることができる．私の会社に，部長の資格を持っているが，プログラマをしている人がいる．

中高年と若いプログラマとで，仕事のちがいは特にないと思う．ただ，私（２５才）は，いつも上司の視線を背中に感じながら，仕事をしている．なにか，常に監視されているようだ．その点，中年プログラマは，上の人がいないわけだから，気楽に仕事をしている．なかには，昼ごろ出勤する人もいる．

私も，はやく実力をつけ，そうした中年プログラマみたいになりたい．

久保宏志

ソフトウェア・クライシスはないという言説で評判になった興銀レポートからの引用をすると，

「メインフレーマや大手ユーザの下請けであるソフトウェア産業にとって，決められた納期に作業を間に合わせることは最大の関心事である．また，ユーザの無理な納期をのまざるをえないという事情もある．このとき，肉体的な無理のきかなくなり始めている３５才以上のソフトウェア技術者が企業内で問題になる．」

これが一般にいわれる「３５才定年説」の構図である．この指摘は，わが国のソフトウェア産業の体質をやゆしているのであり，ソフトウェアの特質を物語っているわけではない．このレポートに対しては，数多くの反論があげられているが，多くはソフトウェアの特質に沿った議論になっており，この点に関しては完全に３５才定年説は否定できる．

３５才で引退せざるをえない労働形態は若く優秀な人々にとって魅力のないものであろう．若い人々が仲間入りしてくれることを望むなら，この労働形態をかえていく仕事を，中高年の層が担わなければならない．この努力を怠ると，中高年になっても安い労働力として供給されつづける事態を迎えかねない．

手始めになすべきこととして，ソフトウェア・ハウスの経営者の考え方を知ることとプログラマの労働の実態をただしく認識することをあげたい．

大地英夫

プログラマに定年はないとおもいます．

プログラムは，単に新規に作るだけではなく，やはりベテランはプログラムを新規に作ろうとするより既存のものを作りかえて使おうとするのではないでしょうか．

柴田潤

プログラマに定年はない．自分自身の経験で，入社後年数を経るにつれ，プログラム設計の内容が広がり，ユーザーの立場で仕様を決定できるようになり，バグがs少なくなったことから，このベクトルが年齢を重ねると変

化すると思えないから．

中高年のプログラマは選択項目の多い場面で活躍する．たとえば，ユーザ・リクワィアメントの把握が上手で，相手の立場で仕様を決定してゆくことができる．ユーザ・インタフェースの多い場面で重宝される．調整力・交渉力を発揮する場面が多い，等いくらでもある．

小野寺斎

発想に年齢はない．ただ，プログラミング速度はおそくなる．

若いプログラマは新しい方式（ｏｂｊｅｃｔ指向等）を，中高年は経験を生かして，どっしりと仕事をしていけばよい．

野村行憲

企業人としての定年はあるだろうが，プログラマとしての定年はない．

現在，プログラマは，特殊な職業ではなくなってきている．もし，特殊な職業だとするならば，言語を操って著作活動をする形態から，作家を例にとってみると，作家は，中年でデビューし，高年で活動のピークとなる．もし，定年があるとすれば，本人の気力が失われた時点であろう．若年プログラマは，量の仕事，そして中高年（実年！）のプログラマは質の仕事．

松本崇純

体力・集中力等は年齢とともに衰えてくるが，その衰えには個人差があり，何才で定年とはいえない．プログラマとしての年齢によるちがいは，経験と体力だけである．

大木幹雄

ダイクストラやクヌースのように，創造的な作業のために巧妙なプログラムをコンピュータに入力する人（これをプログラマと云うなら）には，定年がない．一方，人から依頼されてコンピュータに人間の意図を伝えるだけの人であれば，定年はある．何故なら司法書士のようにプログラマには資格はいらない．したがって仕事が若年者に奪われることが一つ．二つめは，体力がなくなること．三つめは仕事を依頼してくる者に「それ位自分でやれ」という気持ちが目ばえること．逆に，このような事態をのりきれる人（会社から出家した人）には，定年がない．

プログラマ自体が不必要になる恐れはあるが，今後１０年間位に限定すれば，コスト・パーフォーマンスは中高年が断然よい（給料の割に，信頼性，効率，保守の容易なプログラムを作る）．したがって，信頼性・効率がよく，保守が容易なプログラムを作らせるには，中高年に限る．陶芸よりよほど奥が深いプログラマに，もし人間国宝の設定があったとしたら，また，プログラムに「銘」が入るとしたら，一層若い人との仕事上のちがいは明白になる．「絵つけ」は中高年，「かま焼き」は若手の分業作業が自然．

横山博司

本人が，その気をもってすれば，何才までででもできるのでは．もちろん，クビにされれば別ですが．しかし，一生プログラマ暮らしをしようというもの好きな人がいるのでしょうか？とにかく，その人の個性にもだいぶ左右されるでしょうが，中高年のプログラマは，まず，全体を見わたしてからコーディングにかかるのに対して，若いプログラマーは，コーディングしてから考えるのでは．だから論理的なバグは，中高年のプログラマーの方が少ないのではないでしょうか．

塩野富教

プログラマの定義が明確でないが，ソフトウェアを作る人とするならば，一般社会の定年と同じだと思う．ただし，現在の作業環境におけるソフトウェア開発を考えるなら，定年は，肉体的な年齢に左右されるだろう．あくまで，開発作業が改善されることを前提とした意見である．

そうした意味では，どこの会社でも同じだと思うが，技術を磨き上げるには，時間がかかる．磨き上げた技術をもつか，潜在的な技術をもつかの違いであると思う．同じ仕事をしたからといって，全く同じ技術力を身につけられるとは限らない．あくまで，技術年齢が大切であると思う．

高橋幸男

われわれの仕事には，経験も重要な要素の１つであると考える．プログラマ定年説を盲目的に信じて，経験豊富なプログラマ（だけに限る）をやめさせる（やめてしまう）のはもったいない．若い人より体力はないが，エ

レガントなプログラムは作るはず（と思いたい）．ただし経験・能力による仕事上の差は当然あると思う．

中村正三郎

作家，詩人，画家などに定年はない．書けなくなった時が定年とも言えるが，死ぬまで創作意欲の衰えない人も多い．いや，それどころか，技術・経験とも円熟し，すばらし作品を創造できるように思う．

プログラマも同じで，知的労働なら個人の能力に応じた成長を死ぬまで続けることができると信じている．実際，「松」の基本デザインは，かな漢字変換は４０代前半，エディタ部は３０代半ばの人が行った．

中高年プログラマは，いやなら，もうコーディングをやる必要はない．経験に裏打ちされた，しっかりした基本デザインを提出すればよい．経験の浅いプログラマには，一緒に仕事をすることで，多くの事を教えられるように思う．基本デザインも提出できず，若い連中を導けない中高年プログラマは，いわば修業がなってなかったわけだから，さっさと転身なさったほうが，世のため，人のためになる．

佐藤千明

プログラマの定義が明確でないが，私は「外部仕様が決められたあと登場しテスト完了までを行う人」と考えて意見をのべる．

日本では，年相応の給料をもらっているのが普通であるが，４０代のプログラマの生産性・信頼性は２０－３０才代以上とは思われないので，企業の論理として中高年のプログラマを必要としなくなる．定年は，本人の希望ではなく，組織の側からの強制である．私個人としては，いつまでもプログラミングしたいと思っている．給料に見合う仕事ができるなら定年はないだろう．

アプリケーションに精通し，システム・デザインを行う外部仕様を決める．システム開発のプロジェクト・リーダとなり，若いプログラマをコントロールする．プログラミングを商売とする会社であるならば，営業もしくはそのサポートを行う．独立してシステム設計事務所を開き，コンサルティングやシステム監査を受請う．等々，やることは一杯あり，後は本人の努力しだいだと思う．

中園順三

プログラマの定義を「何らかの言語を用いて，設計されたシステムの実用化に対して製造を行う人」とすると，以下の問題が生じてくる．

・プログラミングとは記憶力による所が大である．
・新しい言語が登場してきた場合の習得力が劣る．
・新しい開発環境についていけない．

このようなことから，従来の方法にしがみついて生産性が上がらなくなる．

中高年のプログラマは，その持っている資質によって，次のような仕事をやったらいいのではなかろうか．

発想豊かな人	設計者，企画者
貴重面な人	検査
口のうまい人	営業
金儲けのうまい人	経営者
知識豊富な人	コンサルティング
上記以外の人	ヤメル！

小須田正孝

やはり，体力もそうですが，発想の転換の速さ！やわらかい頭脳しか持ちえない慣れの速さは，４０－５０才が限度と思います．これもしかし，一概に４５才といえないのは，人により，環境により，千差万別だからでしょう．人によっては，何年たっても，デラックスな頭脳を持っておらます．そういう方はうらやましいです．

中高年プログラマのやる仕事としては，後続プログラマ養成（ノウハウの伝達）がよいでしょう．ただし，それも育成機関があり，ある程度ペイしなければ成立たないし，コンピュータ・メーカーのソフトウェア開発事業部の上の方には，４０－５０代の年齢層がいらっしゃるわけでして，そういう方々は，いわゆる管理屋さんであり，プログラマとしては扱ってもらっていないようです．それでも一応，お金をもらって成り立っているようです．

河村一樹

すべての人々にあてはまるとは思われませんが，少なくとも３０才になってもプログラムだけを組む人材であるとは，本人にとってふがいないことでしょう．たぶん自分自身の仕事に対するモチベーションが相当低下するでしょう．たしかに，プログラムが自分の思い通りに動くことは楽しいことですが，プログラムはシステムのごく一部分なのです．全体もわからぬまま，プログラミングの技術的テクニックだけが向上しても，自分の力を充分発揮していると思えないでしょう．あまりに小さな世

界だからです．自分自身との葛藤に耐えられなくなる限度は，たぶん３０才となるでしょう．

国鉄マンのように，エプロンをして駅のコーヒーショプの店員をやってみるなんて考えてみても，たぶん無理でしょう．プログラマはあまりにプライドが高いのです．結局はこの道を歩むしかないでしょう．そのためには，プログラマより上位の職種であるＳＥ，ＳＰ（システムプランナー）をめざしていく以外にはないでしょう．資格をとるなり，自分自身をアピールして，もっと上位の職種につくべきです．とにかく，経験を積んでいく以外に有能なＳＥへの道もないのです．着実に一歩一歩自分の将来をみつめながら，実力を養成していく必要があるのです．それによって，次の飛躍が期待できるのですから．

加藤義介

プログラマ全員，定年なんかないと思っている．しかし，社会がプログラマはそいうものだと思っている（建て前）．実は，私は，心から定年はないと思っている（これは本音）．プログラムができないじゃなくて，体力がたりなくなる．

自分で定年したいと思う人は，プログラマをやめる．そうでない人は，システム内のプログラム等の分割とまとめや，
若いプログラマたちの管理プラス少々のプログラミング，といった中高年のプログラマになる．

岡芹晃

会社組織上，いつまでも作成者ではいられない．４０才ぐらいの年齢になれば，管理・指導的立場にたって，後進を育てていかねばならない．また，一般人は，だんだん根気がなくなっていき，しこしこプログラムを作ることには，生理的に向かなくなると思われる．能力的には，継続して作成していかれば対応できると思うが，

管理，教育，検査（現在ここの部分が手薄と考えられる．中高年を導入し，過去の知識からきびしくチェックすれば信頼性向上につながる），要求定義（仕様打ち合わせ）等が，中高年に向いた仕事である．

長岡ワークショップ

ＳＥＡの第１回ワークショップは「実践的ソフトウェア開発環境」をテーマとして，２月１９－２２日の４日間，雪まだ深い新潟県長岡市で開催された．参加者数は３５名（会員３１名，一般２名，招待講演者２名）．

プログラム

２月１９日
- 13:30 - 14:00 オリエンテーション
- 14:00 - 16:30 自己紹介
- 16:30 - 17:00 グループ分け
- 18:00 - 20:00 情報交換パーティ

２月２０日
- 9:00 - 10:30 招待講演（１）
  米沢明憲：ＡＢＣＬについて
- 11:00 - 21:00 グループ討論（１）

２月２１日
- 9:00 - 10:30 招待講演（２）
  栗原正利：ＵＮＩＸ環境
- 11:00 - 12:00 グループ中間報告
- 13:00 - 21:00 グループ討論（２）

２月２２日
- 9:00 - 12:00 クローシング・パネル

今回のワークショップでは，これまでとちがって，グループ別にサブテーマを決めて討論するのではなく，参加者各自が日頃抱えている問題を持ち寄って，相互に批判や忠告をだしあうという形式をとった．

そして，最終日のパネルで，残された問題点についての自由討論が行われた．

結果は大成功であり，クロージング・パネルをもっと続けたかったというアンケートの声が強かった（ほんとうは，もう一晩泊まって，お酒を飲みたかったのかも知れないが）．

全員のポジション・ペーパや，招待講演，グループ討論，およびパネルの記録をまとめた報告書は，４月中旬発行の予定．それを機会に，４月２４日には，フォーラムが開催される（詳しくは３２ページの案内を参照されたい）．

シグマ・プロジェクトへの建設的提案

松原 友夫
日立ソフトウェア・エンジニアリング

1．増殖し独り歩きするシステム

物理学者フレッド・ホイルが書いたＳＦ小説の中に登場する「暗黒星雲」では，個々の星が互いに通信しあい，増殖し，全体が一つの生命体として行動する．私は，シグマ計画の説明を聞きながら，これを思い浮かべた．シグマは，今までわが国で作られたどのシステムにもない特異な性格を持っている．システムが，ある時点から計画者の意図を離れて，自らの行動原理の下で独り歩きをはじめる可能性がある，という点である．

アメリカには，似たような例としてＵＳＥＮＥＴがある．その加入者は，わずか１年の間に１万から４万に自己増殖し，加入した人々が次から次へと新たな使い道を見出して，ついには一つの文化を共有する新しいコミュニティを形成しつつあるという．

シグマでは，ネットワークばかりでなく，開発環境と結びつくことによって，さらに広い場が用意される．計画書に書かれたシステムを完成させることは，目的のごく一部にすぎない．システムの利用者をたきつけて前向きの自己発展を起こさせることが，究極の目的だといえる．自己発展・自己増殖といったいわゆる離陸現象が起こらない限り，世間はシグマが成功したとは認めないであろう．

2．技術移転と社会的慣性

この種のシステムを離陸させることや，独り歩きを望ましい方向に誘導することは，決して容易でない．学会などで華やかに発表される新しい開発手法，ツール・システムあるいは開発環境のうちで，離陸しているシステムがいかに少ないかを考えてみれば，よくわかる．

ソフトウェア・エンジニアリングの泰斗Ｌ．Ａ．ベラディは，ある講演の中で，「ソフトウェアの技術移転においては，技術的な側面ではなく社会的な側面の方がより重要である」と述べ，技術移転を妨げる最大の要因に，社会的慣性をあげている．

シグマの開発に携わる人々はもちろん，すべての利用者も含めて，シグマに関与する人々が，このシステムの持つ特異な性格を理解し，その物理的構築とは異なった視点からそれを見ることができるか否かで，計画の成果は大きく変わるであろう．

アメリカには，ＵＳＥＮＥＴの例にみられるように，開拓精神によってもたらされたいくつかの成功例がある．しかし，個人の自己主張が強いお国柄は，まさに個人の自発性やボランティアのみが頼りで，それが，システムの広域的組織化を妨げている面もある．石油危機を国民的コンセンサスによってうまく回避し得た日本は，もしかしたら，アメリカよりはるかにうまく，このナショナル・ソフトウェア開発環境をものにしうるかも知れない．

しかし，それには何かが必要である．

3．誘導者の視点

シグマが成功するポイントの第１は，あまり適切な表現ではないが，「大衆を乗せる」ことである．量は力を生じ，質を変え，それによって独り歩きが始まる．「大衆」とはシグマ・ワークステーションを購入し，それを使ってソフトウェアを開発する人たちであり，「乗せる」とは，シグマなしでは仕事ができないかのような状況を作りだすことである．

第２のポイントは，たくさんのハードウェア・ソフトウェアと，その上に多くの人を載せたシステム全体を，正しい方向，望ましい方向にいかにして誘導するか，という点である．巨大タンカーと同じで，巨大システムの方向は急には変わらない．先行する施策が時を経て効果をあらわす．

望ましい方向に，いくつかの視点がある．

一つは，情報処理の場における共存である．情報処理のグローバルな拡がりの中における情報の交換や共有は，シグマの必須条件である．これらは相手があり，しかも常に変化する．世界的な技術の標準化動向という動く標的を追うことは容易ではない．

しかし，もしそれを怠って，シグマだけが孤立した道

を歩むことになると，慣性の大きさのゆえに，方向転換はきわめて困難になる．それを避けるには，特にインタフェースについて，その標準化の動向を探る触角，その普及についての洞察，採用したシグマ標準の公知，などの策が必要であろう．

もう一つは，技術の水先案内の必要性についてである．このシステムは，実用システムとして計画され，それゆえに，現在あるいは近い将来に利用可能な技術要素だけを用いて構築されようとしている．これは当然なアプローチである．

しかし，一方において，世の中の技術は，われわれの意図とは関わりなしに進歩して行く．そうした状況のなかで，もし，システムに技術的な停滞が生じたとしたら，それはシステムの「死」を意味する．

これを避けるためには，シグマの将来の発展に関わり得るテーマに関して，先行的・先駆的な調査研究を，意識的に，しかも総合的に行う必要がある．それらの結果を水先案内に利用すれば，シグマは未知の海域により安全に進むことかできる．いいかえれば，５年をこえた視点での施策が，シグマの終局的な成功を導く．

最後の視点は，アダプティブな体制，すなわち利用者と開発者の間のフィードバック・ループの形成である．これさえしっかりしていれば，基本計画書に些細な矛盾や欠陥があったとしても，モニターに提供されるシステムに多少欠陥があったとしても，あまり問題にはなるまい．

体制に必要なものは，利用者からの意見の収集，ノイズの除去による建設的意見の抽出，およびそれへの敏速な対応であろう．各種のデータ収集については充分配慮されているようで心強いが，それらをシステムに効果的にフィードバックさせるためには，ウォーターフォールに捉われない柔軟な開発プロセスの運用と，心の通ったフィードバック体制が必要であろう．

４．いくつかの具体的施策案

シグマ・システム構築計画書は，大変な力作である．短期間にここまで計画を煮つめた関係者の努力には頭が下がる．恐らく，今後，シグマ開発本部の方々は，システムの物理的要件をまとめるために，日夜忙殺されることであろう．

しかし，前述したように，このシステムは，ただ完成して動けばよいというものではない．それが一つのトリガとなって，人々の行動や開発スタイルを変革し，社会にインパクトを与えつつ自らも発展することがほんとうの目的である．そう考えると，計画書に書かれたことの実現と並行して，冷静な眼で多面的な気配りをすることが，きわめて重要になってくる．

以下に，これらのいくつかについての私案を述べる．

４．１．ユーザ・イメージの確立

基本計画書の第３章には，利用者から見たシステム要件が，主な機能別に述べられている．しかし，いざシグマを利用しようという際に，末端の利用者にとって必要なことは，「自分がやっている仕事の流れがシグマを使えばどう変わるか」であろう．

つまり，シグマが支援対象と考えている典型的な作業の流れ，あるいはデータベースサービス機能や通信サービス機能を，単独または他の機能と組合わせてうまく生かせる業務の例を，できるだけ具体的に示すことであろう．

基本計画書上の機能記述をヨコ糸とすれば，仕事から見た機能記述はタテ糸である．「Ｙ氏の場合（概要説明書付録）」や「A Day after SIGMA（５９年１２月のJISA パネル討論での岸田氏のＯＨＰ）」に，作業とデータの流れのイメージが握める程度に肉付けし，まとめたドキュメントというのが，およその構成イメージになる．

これは，原案を，シグマの外で実際にさまざまなソフトウェアの開発に携わっている人たちが作成し，シグマ開発本部で，その実現性についてレビューするのがよいだろう．仮にこのドキュメントを「シグマによる開発事例集」と呼ぶことにする．これをまとめて公刊する狙いは二つある．

一つは，潜在的利用者に対するＰＲである．大衆に，「こんなことができるのなら使ってみよう」という気持ちを起こさせるためには，こま切れの機能記述では不充分であり，現実の業務からのアプローチが不可欠だと考える．

もう一つの狙いは，開発者側におけるイメージ統一である．ユーザ・サイドから見て「こんなことができるようにしなければならない」というコンセンサスを，適切なプライオリティをもって認識させることである．これは，もしシステムの開発途上で，開発項目と期限のコンフリクトが生じた際に，プライオリティ変更の判断を助けるであろうし，システム完成時のベンチマークにも使

える．また，ツールの詳細仕様が決まった段階で，開発事例集中の局面をリアルなものに置き換えることによって，フィージビリティの検討にも使えるであろう．

４．２．立ち上げ戦略

ソフトウェア開発に携わる人々は一般に保守的である．これが技術移転をむずかしいものにしている．シグマは，１企業内ですら多くの困難を伴うツール・システムの導入・活用を，国家レベルで成功させようというプロジェクトである．ベラディのいう社会学や心理学など，人間集団の行動特性を読んだ戦略が，重要な役割を果たすであろう．

利用開始の初期には，ワーク・ステーションへの投資を決意するに足る効果の確信，企業間の競争意識，未加入者たちの疎外感や不便さ，などがシグマ加入への動機となるであろう．ある程度軌道に乗った段階では，一般企業やメーカなどの発注者が，シグマを用いた開発を指定することによって，加入者拡充のダメ押しをすることは十分可能であろうが，成否を左右する最も大切な時期は，モニタ利用期間および初期の加入者立ち上げの時であろう．

初期の加入者拡大のためのポイントを，いくつかあげてみよう．

（１）モニター期間

まず，モニター期間中に，モニター企業にシグマがなくてはならぬものにしてしまうことである．

これは必ずしもツールである必要はない．どちらかというと，通信サービスやデータベース・サービスを生かして，魅力ある利用法を考え出すことができるであろう．アイディアの種は，ＵＳＥＮＥＴやＣＳＮＥＴなどに，いくらでも見出せる．

たとえばネットワーク上でのＲＦＣ（リクエスト・フォア・コメント）や投票機能（何かの問題提起に対して意見収集を行い，全体の結果がすぐわかる）など，双方向コミュニケーションを生かした応用の中に，魅力的なものが多い．

電子ニュースレター，電子掲示板，データベースは，ジャーナリズム・業界・官庁などの協力によって，その内容を充分興味あるものにすることができるだろう．やり方によっては，重要な情報をシグマの上で独占的に流すこともできよう（たとえば，情報処理技術者試験合格者ＤＢ検索など）．また，モニター期間中に，ツール利用者が思いついた意見メッセージを直接集めて電子掲示板に表示することも，シグマへの信頼を増すことに役立つであろう．

（２）一般へのＰＲ

次に考えるべきことは，シグマの効果的な利用法を，まだ使っていない人々に知らしめることである．もちろん，シグマ環境を利用しての効果的開発法が中心となるべきであろうが，ネットワークやＤＢも当然含まれよう．

それを，ＩＰＡはもちろん，各種の協会，ジャーナリズムを動員し，あるいはシンポジウム，セミナー，ワークショップなどのあらゆる手段を用いて，世の中に知らしめることである．これらは，また，すでに使っている人達の利用法を進歩させるよい刺激ともなるであろうし，このために情報処理月間などで，シグマの効果的利用法を表彰するのも一策であろう．

（３）ワークステーション

その気にさせる相手が経営者かワークステーションを使う人かによって，乗せるための作戦が異なるであろう．

経営者は，財務的なことと同程度に，人材の確保に心を悩ましている．もし，学生や教授の目にもシグマが魅力的に見えるなら，ワークステーションの廉価版を学校に普及させることや，シグマを前提としたカリキュラムや試験なども，有効であろう．その他，詳述を避けるが，法律面（例えば派遣法），税制面，官庁からの発注，など多くの面でシグマの加入が有効に働くような制度作りがもしできれば，経営者の支持は間違いなく得られるであろう．

しかし，ソフトウェア業界という人への依存によって成り立つ業界では，概してボトムアップの力が経営者を動かす．したがって、ワークステーションを使う技術者たちにそれを気に入ってもらえるかどうかが，勝負どころである．

２年先，あるいは５年先に業界に参入する若者の多くは，パソコンやファミコンのゲームの洗礼を受けた世代である．かれらにとって，単に仕事をうまく早くやるだけのツールは，魅力の対象とはならないだろう．それ以外の，何か気の利いたこと，楽しさをツールに求めるであろう．

このためには，ディスプレイの高精密化とカラーは必須である．それによって，タイトルなどの画面の芸術性に関心が高まれば，しめたものである．ある程度ツールが普及した後は，ベスト・ヒット・ツールの人気投票が，

ネットワーク上で行われるようになるかも知れない．

（4）ネットワーク

ネットワークも，若者にとって大きな魅力の対象となり得る．そのためには，ネットワーク上での情報交換を単に業務だけに限らず，ある節度をもって，ホビーの領域まで認めるのがよいだろう．すでにアメリカでは見られるように，ペンパルならぬネットパルが縁となって結ばれたり，ネット上のレポーターが現れ，新聞を賑わすことになるかも知れない．シグマがこうしたことに使われ出せば，それは成功の証しである．

もっとまじめなネットワークの利用法としては，規格原案作成のスピードアップがあげられる．

情報処理の分野では，旧態依然たる委員会形式のでの審議では，技術進歩に追いつけない．３～４年かけて新しい規格が完成した頃には，すでにそれが陳腐化してしまうことが多い．あまり頻般に集まることができない国際的標準化の場合，問題は深刻で，最近は審議のスピードアップと，そのための手続きの簡略化が，最重要課題となってきた．

この事情は国内でも変わりはない．情報処理分野での我が国の立ち遅れの危機感から，６０年７月に（財）日本規格協会に情報技術標準化研究センター（ＩＮＳＴＡＣ）が設置され，６０に近い委員会が新設されたが，その委員の多くはＩＳＯやＪＩＳ関連委員会とのかけ持ちであり，標準化のスピードアップにどれ程寄与するかは，疑問である．

これを大幅に改善する可能性を，シグマネットワークは持っている．つまり，色々な段階における案の審議や，最終段階では規格草案に委員の意見を反映した更新を，シグマネット上でできるようになれば，その効果は大きい．

これは，標準化に携わる委員およびその所属組織をシグマに加入させる大きな理由となるであろう．

４．３．国際的インタフェース

シグマのような多くの人を乗せたシステムは，一旦走り出すと止まらないばかりか，方向を変えることも難しい，さらに，定められたコースを走るのでなく，世界の情報処理インタフェースの潮流の中で，相手の位置を知り自らの位置を他に知らせつつ，他と同じ方向に進むように舵を採らねばならぬ．それは，インタフェースの潮流を先読みするアンテナを張ることと，自ら採用したインタフェースを広く知らしめることによって，可能になる．

ここで，あえて標準化といわずインタフェースという表現を用いたのは，「標準化」ということばが持つ広い意味に幻惑されて，それが技術進歩を阻害のではないかという拒絶反応が出ることを，恐れたからである．グローバルな情報世界での共存を考えた場合，際立って重要なのは，情報のアクセスや交換における互換性つまりインタフェースである．これは技術的取り決めであって，あまり技術進歩を阻害しない（もし阻害するようなら，変えられて然るべきである）ばかりか，逆に，大きく発展をうながす．

これまでの情報化の進展は，アメリカが過去に示してきた，「力による事実上の標準化」に支えられてきたが，いまや，それが曲がり角に来ている．

顧客は，機器やソフトウェアの多様な組み合わせの下でのインターオペラビリティを強く望むようになってきているにもかかわらず，業界はその望みを果たしていない．その最大の原因である「力による標準化」が反省され，ここ数年，アメリカを含めて，協調による標準化に努力が向けられるようになった．

シグマ・システムの構築に直接携わる人は，恐らくそのことだけに忙殺され，世界のインタフェースの動向に眼を向けたり，その方向を洞察したりする余裕はないであろう．

したがって，システムの中からインタフェースとなる部分をとり出し，独立に編成された別のチームが，各々の項目について，世の中にどんなものがあり，その中でどれがどういう理由でどのような勢いで伸び（衰え）ているかといった背景を調べ，そうした根拠にもとづいて，このインタフェースを採用した旨を，きちんと記録する必要がある．

そうしておけば，もし風向きが変わっても迅速に対処できるであろう．

また，シグマが採用したインタフェースは，技術的規格として公表すべきである．できれば，ＩＰＡだけでなく，参加業界団体も加わって承認するスタイルが望ましい．そのためには，標準化専門家グループ（ＩＮＳＴＡＣ委員会など）による検査も，ものによっては必要であろう．

シグマが国のシステムである以上，暴走は許されない．それを避けるには，インタフェースという枠組みだけは，

かなり慎重な気配りが必要であり，その枠の中で，かなり思い切った競争をさせるのがよいだろう．

４．４．先行技術の基礎研究

基本計画書は，５年後に実現すべきものに焦点を当てており，したがって，５年以内に実現できそうもない技術項目は当然省かれているが，実現しようとしている項目の中にも，いくつかの研究の余地が残されている．

たとえば，インプリメンテーション言語はほんとうにＣだけでよいのだろうか，ＡＤＡあるいはＭＯＤＵＬＡ－２のような新しい言語は不要なのだろうか．今回は，思いきってＣだけに限定する必要があったことは認めても，先行きはこれでよいのかとの不安が残る．また将来，サブセンタができたら，データベースの分散が必要となろうが，これなども先行した技術研究が必要だと思われる．

こうした計画書上でのペンディング事項や，５年間のタイムスパンを考えて切り捨てざるを得なかった課題を列挙して，それぞれの課題を，意図的に大学・研究所・業界団体等に研究委託する．５年後のシグマの発展のために，できるだけ広域な研究活動を今からスタートさせておく．

高度な研究ばかりでなく，シグマに関連した実態調査もこの中に加えるとよいだろう．しかも，これらのことの多くが，ネットワークを通じて行えるようになれば，これもシグマの普及に役立つであろう．

こうしたシグマに関連する先行技術を，実質的に検討し，まとめるグループが，やはり開発本部とは別に必要であろう．

４．５．システムの適応

ここでは，基本計画書の個別の内容には，あえてほとんど触れていない．精読すれば，おそらく，矛盾や欠落あるいは誤りがあるだろう．しかし，それらをここでつつき出すつもりはない．システムの構築にともなう自浄作用によって，自ずと修正されると考えるからだ．むしろ，気になるのは，システムが働き出してからの適応能力と，それを可能とする体制を，システム自身と開発本部が備えられるかどうか，という点である．

シグマにおいては，使用者との間で，ある時はクレームに答えて迅速にシステムを修正し，あるいはノイズを無視して理想へ誘導するといった，高度のフィードバック・フィードフォワードが成立することが生命である．これまでのシステムで，このようなことが要求された例は少ない．したがって，この点については，やはり，別の視点からの施策が必要である．

まず第一に必要なことは，問題を検知するしかけと高い感度である．この点は「ファクトデータの収集」の項に述べられていて心強いが，それに含まれるべき機能の一つとして，「シグマそのものに意見を言える電子掲示板」を特別に設けることを提案する．

ユーザはシグマを利用していて気づいたことを，気づいた時点でワークステーションから入力し，掲示できる．当然，それは誰からも見ることができる．恐らく，掲示板に寄せられるメッセージには，要求やクレームが多いであろう．シグマセンタは，これらの個々に答える必要はない．全体の動向をできるだけ生のままとらえ，ノイズを捨て，計画的に改善していけばよい．

議論を呼びそうな要求や意見は，ＲＣＦによって公開討議にかければよい．そうすれば，一部の人の偏った意見によって，システムがおかしな方向へ進むことを防げる．こうした定性的情報をもとにして，シグマ・センタは心の通った判断と利用者へのフィードバックを行う．

こうした末端の利用者とは別に，経営者に対するフィードバックシステムも必要である．これは課金システムなど，費用に関する制度に多くを依存するので，それが決まっていない現状では，具体案は提案できないが，たとえば，決められた負担額の中で，他よりも有効に利用し利益を引き出そうという心理を利用したフィードバック・システムなどが考えられる．また，こうしたことのために，特別なツールを用意して，経営者にワークステーションを使わせることも，将来考えてみてもよいのではないか．

そして，これらのダイナミックなフィードバックをつねに考え，実行することができる組織と体制とがあれば，シグマは安定して成長を続けるであろう．

５．おわりに

この小論は，シグマ・プロジェクトの概要説明書および基本計画書に触発されて，これからのシステム開発への個人的提案をまとめたものである．第１号の誌上フォーラムにつづいて，会員各位の議論の題材として役立てば幸いである．

ソフトウェア・シンポジウム’８６

主催：（社）情報サービス産業協会
会期：１９８６年6月4日（水）・5日（木）／会場：東京農林年金会館（東京・虎ノ門）

プログラム（予定）

6／4（9：00～19：30）

◇招待講演◇
大野豊<京都大学>

◇一般論文発表A（プログラミング）◇
・Ａｄａ言語によるソフトウェア開発経験
磧谷寛俊，川田直哉<構造計画研究所>
・ロジック・プログラミングによる設計支援システムの開発
西山聡，牧野京子<三菱総合研究所>
・多用な作業形態に対応できるソフトウェア開発支援システムの試行
伊勢進寿，長山清，末吉一夫
<日立ソフトウェアエンジニアリング>

◇一般論文発表B（ソフトウェアの管理）◇
・基本モデルの構築
加藤政男，高畑一郎
<ソフトウェア・リサーチ・アソシエイツ>
・大型プロジェクトにおける開発作業の標準化
松井悦男<富士通エフ・アイ・ピー>
・ソフトウェア部品の再利用とその体系的な管理方法
大木幹雄<日本電子計算>

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

◇チュートリアルA◇
・人工知能
小林重信<東京工業大学>

◇情報交換パーティー◇

6／5（9：00～17：00）

◇パネルディスカッションB◇
・仕様化技術
コーディネータ　岸田孝一
<ソフトウェア・リサーチ・アソシエイツ>
野辺良一<協同システム開発>　ほか

◇チュートリアルB◇
・シグマプロジェクト
シグマシステム開発本部<情報処理振興事業協会>

◇一般論文発表C（環境）◇
・ＳＰＥＸ：インフォーマル仕様と文書化
岩田成康，歌代和正，筏井幸夫
<ソフトウェア・リサーチ・アソシエイツ>
・ＩＷＢ(Integrated Work Bench)
大野仁勝<野村コンピュータシステム>
・パーソナル・ユニファイド・プログラミング環境
上林憲行<富士ゼロックス>
・ソフトウェア保守支援環境の今後の方向性
道正一郎，田中慎一郎<協同システム開発>

◇一般論文発表D（品質と信頼性）◇
・ソフトウェア信頼性モデルの実践的評価に関する調査研究
長井剛一郎<システム工学>
・潜在エラー数に与える再利用度の影響
小室豊，中山勝之，富田博章<東芝エンジニアリング>
・プログラム設計の要因分析
高田佳彦<日本電子計算>
・ソフトウェアの品質管理と品質保証
渡辺益秀<日本電気ソフトウェア>

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

ツール展示

6／4（12：00～18：00）　6／5（9：00～15：00）

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

問合先：〒１０５　東京都港区芝公園３－５－８　機械振興会館内／（社）情報サービス産業協会／ソフトウェア・シンポジウム実行委員会
ＴＥＬ：（03）４３６－３９３８／ＦＡＸ：（03）４３６－０３０８

シグマへの疑問と期待

芝原雄二
協同システム開発

１．はじめに

いま私は，ＪＳＤで，ソフトウェア開発におけるコスト・モデルや信頼性モデルの実用性評価を目的とする研究プロジェクトの世話役的な仕事をしている．そうした立場から，シグマ・プロジェクトに関していくつかの素朴な疑問を抱いている．それは，ソフトウェア開発の定量的管理とそのために必要なデータ収集のむずかしさが，シグマのなかで，どれだけ真剣に検討されているのか，についてである．

２．最初の疑問

だいぶ前にある雑誌で，「シグマは生産性を４倍に上げる」という見出しを見た．もちろん，これはある種のセールス・トークなのだろうが，ソフトウェア・データの統計的処理に苦しんでいる身からすれば，いくつかの疑問が心の中にわだかまってくるのは，やむをえない．

たとえば，ソフトウェアの生産性は何で測るのだろうか．４倍という数字の根拠となる比較対象物は，存在しているのだろうか，等々である．

これまでにいろいろな文献を読んだ限りでは，シグマの主なねらいは，ソフトウェアの生産性および品質向上を確保するための環境の構築であって，どこをどう読んでも，具体的に生産性を４倍に上げる根拠の存在を，明確にはしていない．つまり，シグマはユーザに対してよりよい環境を構築する手段を提供だけであって，シグマを導入したからといって，けっして棚からボタモチ式に生産性が向上したり，品質が保証されるわけではない．

ほんとうに生産性を４倍に上げるには，シグマを使う側が，じっくりとシグマを観察し，そこに用意されたどのツールがどう使えるのかを見極めたうえで，自分の環境に取り入れ，アレンジしなければならないかのように見える．

昨年１２月のウィンター・フォーラムのパネル討論において，私は上記のような疑問を，シグマからきたパネリストたちにぶつけてみた．一応，私の理解はまちがっていないことがわかって，疑問はとけたのだが，しかし，私がひそかにシグマに期待していたことは，少しちがっていた．

３．データ収集に関する素朴な疑問

ＪＳＤの定量化モデル・プロジェクトにおけるわれわれの悩みは，現実の開発プロジェクトでは，なかなか，生産性や信頼性に関する基礎データがとれないということである．

国家プロジェクトであるシグマなら，ある程度の強制力をもって，さまざまな定量的データをシグマ・システムの開発中に収集し，それを一般に提供することができるのではないだろうか．もちろん，シグマで開発されるものは，ツールの類が主であり，ふつうのアプリケーション・ソフトとは性質がちがうが，それでも，工数やエラーに関するデータなどがきちんと集められれば，大いにありがたい．

ただし，問題は，それらのデータ自体の信頼性である．

シグマがツール開発を外部に委託する場合，費用の積算は当然工数ベースになるだろう．受託した会社は，なんとかその工数より安く仕上げて，利益を出そうとする．そんな時，正直に実際かかった工数を申告してくれるだろうか．

実際のデータ収集はどうするのだろうか．シグマが完成し，開発工程のほとんどが自動化されたあとなら簡単だが，現状ではどうしても面倒な手作業にたよらざるをえない．

エラー・データも面倒な問題を含んでいる．デバッグやテストは，ふつう，ぎりぎりの時間的制約のなかで行われる火事場さわぎであり，一つ一つのバグについての詳細な分析データは，とかく，失われがちである．

アメリカの例では，そうした余分（！）なデータ収集にかかるコストとして，開発費の少なくとも１割以上は必要だそうだが，シグマでは，それだけのお金をツール開発費に上乗せする覚悟を，ほんとうに決めているのだろうか．

システムが完成し，稼働しはじめたあとはどうなるのだろう．

ユーザがシグマで用意されたツールを使って仕事をしたら，その時のデータは自動的に集められるようになるのだろうか．それぞれのシグマ・ユーザにとって，それ

らのデータは貴重な情報であり，そう簡単には，他人には見せたがらないと思うが，シグマ・センタのデータベースに，それらをたくさん集めるには，どんなメカニズムがかんがえられているのだろう．

3．データ収集・提供体制としてのシグマ

こんなことを考えるのは，十分な情報（データ）がなければ，ソフトウェアの生産性や信頼性についての正しい議論ができないと思うからである．少ない情報量による意志決定は，判断を狭くし，ときには誤った方向へわれわれを導く．企業間で純粋な競争をし，ソフトウェア業界を発展させるには，大規模な定量的データ収集・提供の体制が存在しなければならないと思う．

そのような体制は，すでにアメリカにはいくつかある．ＯＳ３６０の保守グループ，ＵＳ陸軍，ＴＲＷ，ＮＡＳＡ／ＳＥＬなどである．その日本版をぜひシグマに求めたい．

4．また再びの疑問

ここまで書いている間に，シグマ・ニュース第１号を入手した．そこには，

・プロジェクト管理ツール：ソフトウェア開発プロジェクトの管理のためのデータ作成やデータ分析を行うための機能を提供する．

・データベース・サービス機能：ソフトウェア開発の費用，期間，品質，生産性に関する事例をいう．

という記述があった．定量的データの収集・提供体制が明確ではないにしろできつつあることがわかった．

しかし，生来のへそ曲がりのため，また新しい疑問がでてきた．それらを箇条書きにすると，次のようになる．

- 総工数，総ステップ数，総計算機使用時間等の把握がきわめて困難ないまのソフトウェア開発現場の状況をどう考えるのか．
- データの品質を保証するために，どういう基準や，どんな体制を考えているのか．
- プロジェクトの管理用のデータとはどのようなものか．
- データ分析とは，統計手法なのか．特定のコスト・モデルを対象にしているのだろうか．

以上，勝手な疑問を書きつらねてみた．シグマ関係者の方々にお答えいただけるとうれしい．また，近くスタートする管理分科会でも，このことをぜひ議論したいと思っている．興味ある方々の参加をお待ちする．

ＳＥＡ人気アンケート

日経コンピュータ誌３月１７日号に，ＳＥＡに関する読者アンケートの結果が紹介されています．これは，同誌の読者２７０人をランダム・サンプリングし，「あなたはＳＥＡに加入しますか」と質問して，１４０人から寄せられた回答をまとめたものです．

回答者全体では

すぐに加入する	２．０％
将来加入する	７．５％
様子を見てから	５０．３％
たぶん加入しない	３６．７％
絶対加入しない	２．７％

ソフトウェア設計者は

加入する	１５．４％
様子を見てから	５７．７％
加入しない	２５．０％

プログラマは

加入する	１５．４％
様子を見てから	６９．２％
加入しない	１５．４％

ＳＥＡ加入への強い意向を持っている人は，まだ全体の１割弱ですが（それでも日本中のソフトウェア関係者の１割といったら大変！），「しばらく様子をみてから決める」という人々が約半数いるので，これからのわれわれの活動いかんによっては，大いに発展が期待できそうです．

組織の基礎を固めるために会員拡大にご協力を！

CALL FOR PAPERS

9TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING

FORMALIZING AND AUTOMATING THE SOFTWARE PROCESS

MONTEREY, CALIFORNIA, USA 30 March-2 April 1987

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

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

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

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

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

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

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

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

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

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

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

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

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

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

IMPORTANT DATES

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

SPONSORED BY:

ACM SIGSOFT

IEEE COMPUTER SOCIETY

IEEE THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC

シグマを巡る三題噺

長井修治
大興ファコム・データ・センター

その1　マラソンとシグマ

シグマ・システム研究会の手になる「ソフトウェア危機への挑戦」という小冊子を読まれた方も多いと思う．これは，冒頭にも述べられているようにシグマ・システムを利用した時の世界を描いている．分析・設計に始まり，具体的なメイキング，そしてテストとそれぞれのフェイズに於いて実に様々なツール群が活躍する．私にとっては夢物語りである．これは決して茶化しているのではない，しごく真面目にそう思う．

一方，ＳＥＡＭＡＩＬ第１号におけるシグマ紙上討論会が行われた．こうした開発担当者達の生の声を聞けたのは実によかった．これはシグマ・システムの開発当事者ということもあって，話がぐっと現実的，というよりは現状報告的な内容であったように思う．

この２つ，いうなればタイムスケール上の，ＳＴＡＲＴとＧＯＡＬを示しているとはいえまいか．折りからのマラソンブーム．最近は長く走ることが好まれているようだ．

その2　空隙の話

ボイジャーが飛行中である．これまで，惑星には土星にしか輪は無いと思われて来た．ところが実際に輪を持った仲間は他にもいたことが，ボイジャーの送ってきた写真で判明した．リングそれ自体も地球で見るよりもはるかに複雑で多くの空隙を持っていた．これらの発見は，地球を飛び立ってから，われわれがその存在を忘れていたほどの，長い空白期間を一気に埋めてしまった．

シグマ・システムの開発期間は５年だという．この期間，存在すら忘れられるようでは困ると思い，また，忘れられていても，空白期間を埋めてくれるものが，５年後に出来上がるという保証もないことから，ＳＥＡＭＡＩＬに，シグマへのメッセージを送ることにした．前段を引き継げば，ちょうど真中のタイムスケールを埋める部分を，これから描き出そうというのである．

ＳＥＡの会員ばかりでなく，ＳＥＡＭＡＩＬを読んだすべての人々が，シグマに関心を寄せることで，国家プロジェクトとしてのシグマの軌道を，正しい方向へもっていけるのではないだろうか．

その3　シグマを食べるグルメの条件

とりとめもなく書いて来たが，これ自体自分の頭の整理でもあった．私自身一体どれほど，シグマのことを理解しているのだろうか．夢の世界に憧れる少女のようなものだと云ってもよい．反面，自分の現実に立ち戻るとその隔たりの大きさにぼうぜんとする．

すると，当然出てくるのがもう一人の私で，つまり，さい疑心である．＜本当に出来るの？＞という疑いである．結局，この姿勢は何処から来るのかといえば，傍観者の眼だといえる．完成の暁にちゃっかりと利用させてもらって美味しい所だけ食べようというのだ．まあ，これでも悪くはないのだろう．でも，どうせ食べるなら，うまく食べたい．

うまい料理の三要素，それは素材，調理法（勿論シェフの存在も），そしておいしいと感じる舌．異論もあろうがシグマも同じだと思う．つまり，環境（ハード&ソフト），シグマ・プロジェクト，そして利用する人，これがシグマの三要素である．折角料理してもそれを味わう味覚がなければ何もならない．手間暇かけることもないし，第一，コックに注文できないではないか．で，今回三題噺の締めくくりとしていいたかったのは，利用する人は味覚をわかる舌を持つということである．

シグマではＵＮＩＸを利用するようである．ＵＮＩＸは，今やこの世界における共通言語ともいえる位置にある．いうなればソフトウェアのＥｎｇｌｉｓｈともいえる．世界を旅する時，好むと好まざるとにかかわらず英語が役に立つことは，海外旅行にいかれた方は思い当たると思う．シグマ・ワールドの旅行には，「舌」と「英語」が必要だ．

残されたタイムスパンを埋めていくのは，われわれユーザ側の役目だと思う．シグマ・プロジェクトへ大言に注文を出そう（でも，そこでは，三大珍味も料理してもらえるのだろうか）．

ソフトウェア開発環境構築に関する提言

岸 田 孝 一
ソフトウェア・リサーチ・アソシエイツ
野 村 敏 次
情報処理振興事業協会
早 川 雅 貴
野村コンピューターシステム

提言１： ソフトウェア開発の生産性を大幅に向上させるために，各種支援ツールの機能をさらに拡充し，その産業界への普及を促進すべきである．

現状におけるツールの利用状況を工程別にみると，プログラミング／テスティング工程支援のツールは，ある程度使われているものの，要求定義／設計工程を支援するツールは，ほとんど使われていない．また，保守や管理支援のツールについても，その利用度合は低い．

これは，ツール自体の使い勝手の悪さ，機能的な不備，ツール間インタフェースの不統一など，現場で使いものになる実用的なツールが少ないことに，ひとつの原因がある．さらに，どこにどんなツールがあるかを知るための仕掛けや，開発マシンとターゲット・マシンの混同，端末設置台数の不足といった物理的な環境の未整備も，また別の原因であろう．

したがって，使い易い，機能的にも満足できるツールにすべく，その機能拡充を図るとともに，柔軟なユーザインタフェースのもとに，それらのツール利用を図るための仕掛けを考えることが必要である．

提言２： ソフトウェア要求定義／設計工程の支援ツールに関しては，その実用化を図るための基礎的技術開発を促進すべきである．

ソフトウェア要求定義／設計は、開発ライフサイクルのなかでもっとも多くの問題を含む部分であることが，早くから認識されている．にもかかわらず，この工程を支援するツール類は，大学や各種研究機関のなかでは数多く試作されているものの，その実用化にはまだほとんど手がつけられていないというのが，現状である．

ソフトウェア開発の自動化を支援し，これまでの技術的蓄積の有効な再利用を行うためには，この要求定義／設計工程の機械化が必須であり，かつ，そのニーズはきわめて高いにもかかわらず，実用的なツールは，ほとんど存在していない．

したがって，実験的ツールの評価や改良など，その実用化を図るための基礎的技術開発を早急に行い，後のプログラミング／テスティング工程支援ツールへ，データベースを介してつないでいくことが，ライフサイクル全体の効率化を考えるうえで，最も重要なことである．

提言３： プログラミングおよびテスティング工程については，先進的な技法やツールの利用を促進するための手段を早急に講ずるべきである．

プログラミング／テスティング工程は，他の工程にくらべてツール利用の度合が高いとはいうものの，そこで使用されているツールの実態は，エディタ，コンパイラ，デバッガ，メモリ・ダンプなど，きわめて素朴なものだけに限られている．したがって，工程全体としてのツール利用度は，必ずしも高いとはいいがたい．

したがって，より進んだプログラミング技法やテスティングの方法論にもとづく，近代的なツールの利用が図られる必要があるが，そのためには，プログラマの意識改革を含めた教育が重要であるとともに，ツール自体をより使いやすくするための工夫（たとえば，開発専用マシンとターゲット・マシンと

の適切な機能分担など）が考えられる必要がある．

いずれにせよ，先進的技法／ツールの利用促進が，これらの工程における生産性向上には不可欠であり，そのための対策を早急に立てるべきであろう．

提言４： ソフトウェア保守に関する現状の問題を根本的に解決するためには，開発の全工程におけるドキュメントのコンピュータ化が不可欠である．

いま，ソフトウェア保守を担当する技術者にとって，信頼できるものはソースコードしかないというのが実状である．われわれ自身が参加したＪＳＤのソフトウェア保守技術開発計画（ＳＭＥＦプロジェクト）においても，現状の保守作業の生産性向上を意図して，ソースコードをベースにした支援ツールの開発が，その中心的な課題としてとりあげられた．しかし，こうしたアプローチでは，問題の究極的な解決にはならない．

本質的には，ソースコードとドキュメントが常に一致しておりドキュメントの保守がソースコードの保守につながるようなシステムが考えられなければならない．このためには，ドキュメントの機械化が前提であり，ドキュメントとソースコードの両者がコンピュータ上にあって，データベース上で相互に関連づけられている必要がある．

このドキュメントの機械化は，要求定義／設計支援工程ツールの実用化とも密接に関係しており，いずれ技術が進歩すれば，ソース・コードに頼らず，仕様ドキュメントを修正するだけで保守作業がおわることが，われわれの夢である．

提言５： ツール開発における重複投資を回避し，ツールの普及を促進させるために，産業界共有のツール・データベースを構築すべきである．

現在，ツールがあまり利用されていない原因のひとつとして，ツール情報を知るための適切なメカニズムがないことがあげられる．国または業界として，共通に利用できるツール・データベースを作り，それらを通しての利用普及を図ることが急務である．

ツール・データベースとしては，ツールの仕様や利用情報を入れた，いわゆる「カタログ・データベース」と，デモや試用が行えるように実際のツールを入れた，「ツール・データベース」の２つを用意する必要がある．

これらの利用に関しては，当然のことながら，ネットワークを利用して簡単に情報を検索できるとともに，遠隔地からも簡単に試用できる環境が整っていることが必要である．

提言６： パーソナル・ワークステーションをソフトウェア開発に活用するための，ツール統合化技術およびＬＡＮ利用技術を，早急に確立すべきである．

現在，ソフトウェア開発におけるコンピュータ利用形態の主流は，ＴＳＳが大半を占めており，パーソナル・ワークステーションやＬＡＮは，それほど広く普及しているわけではない．これは，現在のワークステーションが，まだ性能の割にはかなり高価であることに起因している．

しかし，将来において要求定義／設計工程からプログラミング／テスティング工程まで一貫した自動化支援ツールの活用を考えるなら，少なくとも１人１台のワークステーションと，それらをＬＡＮで結んだ環境が必要である．

一方，このところのハードウェア技術の動向から見ると，現在のスーパー・ミニ程度の処理能力と記憶容量を持ち，日本語処理やグラフィック機能をそなえた高性能ワークステーションが，かなりの低価格で商品化されそうな見通しになってきた．ソフトウェア両面からの技術開発が押し進められねばならない．

問題は，そうした「夢」のワークステーションが目の前にあらわれたとき，われわれの側にまだ，必要な各種のツールをその上に統合して，トータルな実用的環境を実現するための技術が用意されていないことである．ツール統合化ためのインタフェイス技術や，ソフトウェア・データベース技術，グラフィクスやマルチウィンドゥを含むマン・マシン・インタフェイス技術ＬＡＮ上でのデータや負荷の分散技術などを早急に確立しなければ，せっかくのワークステーションが宝のもちぐされに終わってしまう恐れが，ないわけではない．

提言７： 技術者のみならず管理者の生産性を向上させるために，柔軟なユーザインタフェースをそなえたマネジャーズ・ワークベンチを開発し，普及すべきである．

現状においては，管理者は一般に端末にさわっていないし，また，さわりたがらない．このことは，技術者による生産性向上の努力にも悪影響を与えるものであり，早急に何らかの対策が立てられねばならない．

まず，管理者でも容易に使えるマネジャーズ・ワークベンチが不可欠である．柔軟なユーザ・インタフェースを通じて，データベースに入っている各プログラマの進捗状況をはじめ，各種の管理データが簡単に見られ，適切な判断が下せる仕掛けになっている必要がある．管理者がコンピュータに慣れ，機械を通しての管理が容易となれば，管理面における生産性が向上するばかりでなく，プロジェクト全体としても生産性の向上が図れることになる．

提言８： 諸外国の技術動向と歩調を合わせつつ，ソフトウェア開発環境の標準的基盤を国として早急に整備すべきである．

環境構築技術は，諸外国においても研究開発がさかんに行われているが，まだ試行錯誤の段階であり，技術的には手探り状態であるといっても過言ではない．ソフトウェア開発環境を構築する要素としての，ソフトウェア・データベース，ＯＳ等については，産業界と国とで協力して標準的なものを作っていく必要がある．

シグマ・プロジェクトでは，ＵＮＩＸをベースにした標準的なＯＳを作ることが決まっているが，これは現時点においては最適の選択といえるだろう．

しかし，ワークステーションのマルチＣＰＵ化などの動向を考えると，より新しい構造を持ったＯＳが，いずれ近い将来に必要とされるにちがいない．そうした事態にそなえるための基礎的な研究開発をいまから手がけておくことが望ましい．

提言９： わが国ソフトウェア技術の水準を向上させ，それを維持していくために，公共的なコンピュータ・ネットワークを早急に構築すべきである．

コンピュータ先進諸国の中で，ソフトウェア研究者や技術者が情報交換を行うための電子ネットワークが存在しないのは日本だけだということが，最近あちこちでいわれはじめた．これは，ソフトウェアの技術水準を向上させるためにそうしたネットワークが必要であることを，研究者や技術者のみならず，多くの人々が感じてきたからにほかならない．

公共的なコンピュータ・ネットワークは，情報交換の促進を通じて技術者の全体的な能力向上に役立つばかりではなく，企業にとっては，仕事の場を地方に移すことが可能となり，地方振興にも一役買うことになる．このことは，ソフトウェア開発の進め方に大きなインパクトを与えるばかりでなく，新しいマーケッティング・チャネルの誕生というかたちで，営業活動にも大きな影響を与えることになる．

こうしたネットワークを構築する上でのもっとも重要なポイントは，インフォメーション・ソースとしてのアカデミック・コミュニティと産業界とをどうやってうまく結合させるか，であろう．

現状では，双方の側にいくらかのアレルギーが残っているようにみえるが，それがうまく解消して，ネットワーク上で両者が結ばれれば，きわめて大きな効果が期待できよう．

おわりに

以上の提言は，もともと昭和５９年度に情報サービス産業協会が実施した「ソフトウェア開発のための開発環境構築」と題された調査研究報告書にのせたものに，若干手を加えて再録したものである．

当時，筆者等は，ＪＳＤにおいて４年目を迎えたＳＭＥＦプロジェクト（わが国で最初のＵＮＩＸをベースとする実用環境構築計画）のメンバーであり，そうした実際の経験にもとづいて，旧ソフト協，情産協をはじめとする各種の団体が実施した関連調査を総合的に分析し，報告書の巻末に上記のような提言をまとめた．

同報告書は，不幸にして，あまり多くの人々の目にふれていないと思われるので，ここにあらためてＳＥＡメンバー各位の前に提示する次第である．これからのいろいろな議論の叩き台として役立てば幸いである．

幹事会報告

第２回：２月２６日　18:00 - 21:00
第３回：３月１８日　18:00 - 21:00
会　場：いずれも機械振興会館

議題：

１．会員状況

２月２６日時点における会員数は，正会員２６２名，賛助会員３社であった．３月１８日には，それが，正会員３０１名，賛助会員５社に増加した．そのほかに，入会希望の問い合わせが数十通よせられている．

これは，春のセミナー・ウィークにさいして行ったダイレクト・メイル作戦が，功を奏したものと考えられる．

なお，賛助会員の募集については，早急にパンフレットを作成し，勧誘活動をスタートする．もちろん，全幹事は，自分の会社に働きかける．

２．細則（会費規定）の改正

多くの入会希望者から，その不合理性が指摘されたので，年会費の有効期間を，正会員・賛助会員とも，入会の月から１年間とし，以後は，毎年その月に会費を更新するよう改め，これまでの入会者にも，さかのぼって適用することとなった．具体的には，細則第２条第３項が次のように変更された：

> ３　正会員，賛助会員とも，その会費の最初の有効期間は，入会した月から１ヶ年とし，以後一ヶ年毎に更新して行くものとする．

ただし，設立時の特別（ボランティア）会員は，全員この４月に会費の更新を行う．

３．総会の日程変更

先に５月１０日に予定されていた第１回総会の日時が，会場の都合で１週間後に変更された．

日時：　５月１７日（土）　14:00 - 17:00
会場：　機械振興会館　地下３階研修２号室

４．公開オープニング・イベント

第１回総会が終わった翌週の５月１９日（月）午後，東京農林年金会館パストラルにおいて，「ソフトウェア技術者に期待する」というテーマで，講演とパネル討論から成る公開イベントを開催することが決まった．なお夕刻には，引き続いて，ＳＥＡの設立を各界に披露するためのパーティを行う．

具体的なプログラムは，鈴木常任幹事と久保幹事が中心になって検討中であり，詳細は次号でご案内する．

５．エイプリル・フォーラム

２月１９日－２２日に長岡で開催されたＳＥＡの第１回ワークショップは，成功裡に終わった．そのレポートは４月中旬に刊行されるが，成果報告会を兼ねた半日のフォーラムを，４月２４日（木）に機械振興会館（地下２階ホール）で開催することが決まった．詳細は３２ページ参照．

６．セミナー・ウイーク

目標の６００名を上回る参加者を集めて，内容面からも，また財政的にも，大成功に終わった．ＳＥＡＭＡＩＬの次号では，仕事の都合で参加できなかった方々や，地方の会員のために，ＳＥＡトラックの全セッションの記録を特集する予定．

７．ＣＧセミナー

コンピュータ・グラフィクス学会から，４月２１日に行われる特別セミナーへの後援依頼があり，承認された．詳細は３１ページ，なおＳＥＡ会員は割引価格で参加できる．

８．ソフトウェア・シンポジウム

ＪＩＳＡ／ＳＴＣのシンポジウムが，例年通り６月に開催される（詳細は１７ページ），ＳＥＡの母体となった組織のイベントなので，積極的に後援し，来年以降はＪＩＳＡとＳＥＡの共催行事にするよう働きかけを行う．

９．ＵＮＩＸマシン

ＵＮＩＸユーザ会会長である斎藤幹事から，ユーザ会が近くＡＴ＆Ｔから借りる予定の３Ｂ２を，ＳＥＡの事務局に置かせてほしい旨要望があり，電子掲示板等に共同利用させてもらうことを条件に，承認された．

１０．次回会合

日時：４月１８日（金）18:30 - 21:00
会場：機械振興会館　地下３階１号室

第1回　再利用分科会（ＳＩＧＲＥＵＳＥ）報告

３月５日（水）に，１７名の参加を得て第１回会合を行いました．最初から内容のあることをやりたいという世話人のたっての希望により，主旨等の説明の後，早速発表会に入りました．車でいえば，急発進して突然ギアをトップに入れたようなもので，参加者の中には驚かれた方も多かったと思います．そこで次回（４月２日－水）は，第１回の反省を含めて，参加者がどういう風にＳＩＧＲＥＵＳＥを運営したいと思っているのか，十分に意見交換をしてみたいと思います．

１．主旨説明

ソフトウェアの生産性を飛躍的に向上させる可能性のある技法として，ソフトウェア再利用について調査研究する．調査研究の内容は，学術的なものと実用的なものの２本立てとし，どちらにもかたよらない．

２．運営方法

当面は以下の方法を採用する．

・会合は，月２回とする．

・第１水曜日は原則として誰かが発表をし，それに対する討論を行う．

・第三水曜日は，ある程度時間の取れる人が集まって研究会を行う．

３．ｂｉｔ別冊発行の件

ｂｉｔ編集部と交渉した結果，ＳＩＧＲＥＵＳＥの報告書をｂｉｔ別冊として，共立出版社から発行することに決定した．そこで当面は，第三水曜日を編集会議の日とし，第１水曜日を前回の編集会議の報告の日とする．（次号に載る予定のｂｉｔ別冊「ソフトウェア再利用」執筆者募集を参照）

４．研究発表会

研究発表会の発表内容に関しては，次号に載せます．

・村井進　「ソフトウェア・プロセスとソフトウェア環境の関連性，およびソフトウェア再利用の位置づけ」

３０分の発表後，１５分の討論を行った．討論は活発であったが，時間が短くて意見を言った人が限られたのが残念であった．また，話が抽象的すぎるという意見もあった．

・寺田賢二　「ＪＡＳＭＡＣ：ＣＯＢＯＬ再利用システムとして成功した理由」

ＣＯＢＯＬ部品化システムであるＪＡＳＭＡＣについて，具体的な説明を行った．やはり時間が短くて，具体例の説明と討論ができなかった．具体例の説明を十分に行なえば，非常に面白かったと思う．

５．当面のスケジュール

第１回編集会議

・日時：３月１９日（水）　７：００－９：００

・場所：機械振興会館　Ｂ３－９

・参加費：１０００円

・内容：ｂｉｔ別冊の編集方針の討議と研究会の運営方針を検討する

・参加資格：参加自由．

・連絡先：村井進（協同システム開発 503-4981）

（参加について，事前に連絡はいりません）

第２回発表会

・日時：４月２日（水）　７：００－９：００

・場所：機械振興会館　Ｂ３－１

・参加費：１０００円

・内容：

運営方針に関する意見交換（7:00-8:00）

研究発表：阿部正平「現状における再利用支援ツール」（8:00-9:00）

発表内容：

現在，ソフトウェア開発の生産性向上を目的とした開発支援システム／ツールが，徐々に成果をあげつつある．そこで，これらの中でも，ソフトウェアの再利用を重視しているツールを選び，共通した考え方・方法論・ツールの体系・機能ならびに各ツールの今後の方向性などについて紹介する．

・参加資格：参加自由

・連絡先：阿部正平（協同システム開発 503-4981）

（参加について，事前に連絡は必要ありません）

６．世話人について

現在世話人は，阿部正平（ＪＳＤ），村井進（ＪＳＤ），青島茂（ＳＲＡ）が引き受けております．世話人とは，会運営上の雑用を引き受ける人のことと考えていますので，世話人を引き受けてもよいと思われる方は，村井までご連絡下さい．

（村井進）

環境分科会(SIGENV)

３月の月例会

予定どうり３月１９日（水）夜，機械振興会館で行われた．討論内容については，次号で報告する．

４月の月例会予定

日時：４月１６日（水）　19:00 - 21:00

場所：機械振興会館　地下３階会議室

テーマ：環境事例研究

Gandalf や Toolpack など，アメリカでの環境プロジェクトをとりあげ，その位置づけや評価を議論したい．

連絡・問い合わせ先：松尾正敏（SRA 03-234-2611）

管理分科会(SIGMAN)

前号のアンケートその他で，１８人のメンバーが集まりました．最初のミーティングを次の予定で開きます．

日時：４月４日（金）18:30 - 20:30

場所：協同システム開発

話題：

（１）自己紹介

（２）ＪＳＤ定量化プロジェクトの報告と討論

報告者：芝原雄二

（３）今後の運営方法について

問い合わせ先：芝原雄二

（協同システム開発 03-508-4981）

教育分科会(SIGEDU)

現在メンバーはちょうど１０人．とりあえず，どんな風に会をすすめていくかを議論するための第１回会合を４月１２日夜に高円寺のＳＥＡ事務所で開く予定です．

連絡先：杉田義明（SRA 03-234-2611）

ＳＥＡ関西支部だより

今年は関西も時折雪の舞う寒い日が続いています．ＳＥＡ関西ではこの寒波にもめげず１月２５日に恒例の研究会を行いましたのでご報告します．

今回のテーマは，全国レベルでの話題として「シグマ・プロジェクトについて」，ＳＥＡ自身の最近の活動状況の紹介，そして関西地区における話題として「関西データベース協議会」を取り上げました．

講演者と演題は次の通りです．

１．シグマ・プロジェクトの動向について

柴田潤

本格的に活動を開始したシグマ・プロジェクトの最近の動向と，今後の方向などについてお話いただきました．プロジェクトの成果に期待するとともに，関西を初めとする地方にもなんらかの形で参加または利用できるような配慮を望みます．

２．ＳＥＡの活動について

岸田孝一

昨年１２月２０日のＳＥＡ設立の経過や，東京での活動状況についてお話いただきました．もっとも話の中心はＳＥＡの会員募集にありまして，盛んに誘惑の方法などを吹き込んでいただきました．ありがとうございます．

３．関西データベース協議会について

岡田正之

関西データベース協議会は関西地区の官民学が共同で，ユーザ・ニーズに立脚したデータベースはどうあるべきかを協議するために設立された会です．関西における最近のデータベースに関する話題を中心に紹介していただきました．ＳＥの皆さんが直面している問題と多少観点が異なるためか，聴衆に戸惑いが見られました．

いずれも，日頃ＳＥがあまり接することの少ない，ＳＥＡならではの組織を越えた話題を興味深く聞かせていただきました．今後ともソフトウェアに関する話題を幅広く提供するとともに，もっと身近な技術的問題をも取り上げるように努力をする積りですので，関西のＳＥ，プログラマの皆さんの多数のご支援をお願いします．

（臼井　義美）

3RD INTERNATIONAL SOFTWARE PROCESS WORKSHOP

Iteration in the Software Process
Breckenridge, Colorado, USA, 17-19 November 1986*

(To be sponsored by ACM SigSoft and IEEE-TCSE)

Most existing models of the software process are biased toward a view of the life cycle where verification and validation failures, and the backtracking and iteration that they cause, are the exception rather than the rule. This bias has led to insufficient emphasis on the control and management of iteration both in original system development, and in evolutionary system modification as the requirements change. The 3rd International Software Process Workshop will focus on iteration in the software process from three points of view:

The Iteration Mechanism

A change in the specification of a system, or failure of verification or validation at the end of a system development step, forces the repetition of part or all of one or more development steps. We need systematic ways to:

- determine the extent of iteration needed — i.e. how much of a given step or steps must be repeated?
- make use of the information gathered during preceding iterations (including the nature of verification and validation failures) to guide the current iteration.

Modeling Iteration

Software process models differ in the extent to which they represent and allow reasoning about iteration. For example, in the classical waterfall model, the rule at each phase of the life cycle is simply "do it again till it's right" if verification or validation fail. For the effective management of iteration we need models that represent it explicitly and in greater detail:

- what aspects of iteration need to be modeled, and how can we reason about it?
- does iteration caused by verification or validation failure need to be modeled differently from iteration caused by evolutionary change of system specification?

Iteration in the Small Versus Iteration in the Large

In some respects, the activities in large, multi-person, long-timescale software projects differ from those in smaller projects:

- is the nature of iteration fundamentally different in small and large projects?
- need the techniques for managing and controlling iteration differ for small and large projects?
- if so, what are the special demands of large projects?

The workshop, which will be held at the Beaver Run Conference Center, Breckenridge, Colorado, will consist of three days intensive consideration of these issues by, at most, 30 participants. Prospective participants should submit a maximum 3 page position paper by 1 June 1986, explicitly addressing one of the workshop issues and suitable for inclusion in the proceedings. A small number of participants will be requested to prepare short keynote presentations to initiate discussion. Position papers should be sent to one of the Workshop Chairs:

Robert M Balzer
Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina Del Rey
CA 90292, USA
tel: (213) 822 1511
BALZER@ISI-VAXA.ARPA

Mark Dowson
Imperial Software Technology
60 Albert Court
Prince Consort Road
London SW7 2BH, England
tel: 01 581 8155
mcvax!ist!mark@seismo

Inquiries about local arrangements should be made to the Local Arrangements Chair:

William E Riddle
PO Box 3521
Boulder, CO 80303, USA
tel: (303) 499 4782
RIDDLE@ECLB

*Please note revised date and venue

CALL FOR PAPERS

Conference on Computer–Supported Cooperative Work

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

Sponsored by

Microelectronics and Computer Technology Corporation (MCC)

Software Technology Program (STP)

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

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

Suggested Topics

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

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

Information for Authors

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

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

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

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

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

Important Dates

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

Program Committee

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

Conference Committee

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

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

Call for Papers

Second ACM SIGSOFT/SIGPLAN Symposium on Practical Software Development Environments

Sponsored by:

ACM-SIGSOFT
ACM-SIGPLAN
Office of Naval Research

Palo Alto, California

December 9-11, 1986

General Chairman

Peter B. Henderson (USA)

Program Chairman

Leon Osterweil (USA)

Program Committee

Richard Adrion (USA)
Robert Balzer (USA)
David R. Barstow (USA)
Barry W. Boehm (USA)
John N. Buxton (UK)
Lori A. Clarke (USA)
Stuart I. Feldman (USA)
Susan L. Graham (USA)
Kouichi Kishida (Japan)
Bernard Lang (France)
Ez Nahouraii (USA)
Patricia Oberndorf (USA)
Donald J. Reifer (USA)
William E. Riddle (USA)
Erik Sandewall (Sweden)
Richard N. Taylor (USA)

Scope of Symposium

Practical Software Development Environments assist with the development and maintenance of larger, better, and more reliable software systems. The symposium will address issues fundamental to the development of such practical environments. Suggested relevant issues include:

- Monolingual/Multilingual Environments
- Production Quality Environments
- Database support for Environments
- Knowledge-based Environments
- Workstation Based Environments
- Applications of AI
- Human Factors Studies
- Standardization of Environments
- Support for the Software Lifecycle
- Empirical Evaluation using Environments
- Environment Integration Strategies
- Distributed/Network Environments
- Role of Graphics
- User Interfaces
- Security concerns in Environments
- Comparative Analysis of Environments

This represents a flavor of the topics of interest; however, one purpose of the symposium is to help define the emerging discipline of software development environments. Accordingly, submissions addressing other topics relevant to the theme of the symposium are encouraged.

Of special interest are papers from industrial/government users and developers of environments, and papers defining the role which environments do, or will play in the "real world."

Information and Instructions for authors

Please send ten copies of an extended abstract (10 double-spaced typed pages/2,500 words) to the program chairman:

Leon Osterweil, Chair PSDE
Campus Box 430
Department of Computer Science
University of Colorado
Boulder, Colorado 80309

Extended abstracts will be read by the program committee. They should explain what is new and significant about the work presented and adequately address the following issues:

a) With regard to the contribution of the paper, how does it relate to similar work? Authors should list several key ways in which this contribution is unique or important.

b) If the paper describes an operational system, why is this different than other similar systems? What are its advantages and deficiencies?

Submission Deadline: April 18, 1986
Acceptance Notification: July 1, 1986
Camera-ready paper due: September 1, 1986

The program committee wishes to facilitate the publication of the best full length papers in the appropriate refeered journals. Accordingly, authors of selected, top-quality abstracts may either submit full length papers for publication in the proceedings, or for prompt review by the program committee for journal publication.

Authors of accepted papers will be requested to sign an ACM copyright release form.

Proceedings will be distributed at the symposium, as a special joint issue of Software Engineering Notes and SIGPLAN Notices, or may be purchased from ACM.

Demonstrations

Proposals for demonstrations of prototype or production quality environments and tools will be accepted by the program committee. Six copies of a short abstract (3 to 5 pages) describing the environment or tool, what is new and significant about the system, experience with the system, current status and availability of the system, and its hardware/software requirements should be submitted by April 18, 1986 to the program chairman.

コンピュータ・グラフィックス 特別セミナー

主催：コンピュータ・グラフィックス学会(CGS)
後援：ソフトウェア技術者協会（SEA）
協賛：図形処理情報センター（PIXEL編集部）

募集要項

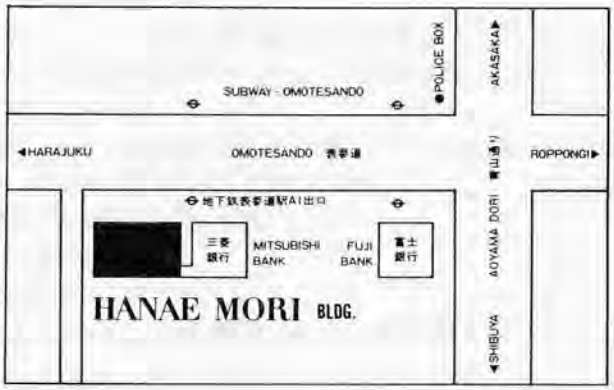
1.日時：昭和61年4月21日(月) 9:30(受付開始)
2.場所：ハナエ・モリ ビル 2F （東京都港区北青山3-6-1)
3.プログラム：

時間	テーマ	講師	所属
9:45-10:00	挨拶	Prof. Tosiyasu L. Kunii	CGS会長，東京大学教授
	モリキュラー・グラフィックスの世界		
10:00-12:00	筑波科学博でのビジュアル・イメージ作り	Dr.Nelson Max	Lawrence Livermore National Laboratory U.S.A
13:00-15:00	フランスのモリキュラ・グラフィックス神秘	Dr. Jacques-Emile Dubois	Institut de Topologie et de Dynamique des Systemes Universite Paris VII France
15:30-17:30	パーソナル・コンピュータ・グラフィックスの動向	Mr. Carl Machover	President Machover Associates Corporation U.S.A
18:00-20:00	懇親会	セミナー参加者，講師，CGS会員，CG-Tokyo'86海外参加者	

講義は逐次通訳付きで行ないます。OHP,16mm,スライドによるプレゼンテーション・デモも予定しています。

4.参加費：5万円（一般）,3万円（CGS会員,SEA会員）
5.申込：所定の申し込み票にご記入の上，下記宛て手紙，またはFAXにてお送り下さい。折り返し受講票及び請求書をお送りします。但し，定員（60名）に成り次第，締め切らせていただきますので，あしからずご了承下さい。

申込先
（〒107）東京都港区南青山5-9-12
アイサワビル7F
コンピュータ・グラフィックス学会（CGS）
特別セミナー係
FAX: 03-498-6943
TEL: 03-498-6941



(特別セミナー会場)

CGS 特別セミナー参加申込書

氏 名：＿＿＿＿＿＿＿＿＿＿ （○印：会員，一般 参加金額：＿＿＿＿＿＿円）

会社名：＿＿＿＿＿＿＿＿＿＿ 部門：＿＿＿＿＿＿＿＿ 役職：＿＿＿＿＿＿

住 所：（〒 ） ＿＿＿＿＿＿＿＿＿＿

TEL：＿＿＿＿－＿＿＿＿－＿＿＿＿（内線 ＿＿＿＿）

備 考：

SEA Forum
April '86

実践的ソフトウェア開発環境を考える

ソフトウェア技術者協会(SEA)
セミナー委員会

実践的開発環境をテーマとする第*1*回*SEA*ワークショップ（*2*月*19*日ー*22*日・於長岡）は，*30*数名の参加者を得て，成功裡に終了しました。討論結果をまとめた報告書は，*4*月中旬に発行されます。そこで，この機会に，ワークショップの報告会を兼ねたフォーラムを計画しました。日頃ソフトウェア開発環境の改善を心がけ，ツールの開発や導入を担当されている技術者・管理者の方々の積極的なご参加をお待ちしております。

(1) 日時：昭和 *61* 年 *4* 月 *24* 日（木） *13:30* ― *17:00*

(2) 場所：機械振興会館 地下 *2* 階大ホール （〒 *105* 東京都港区芝公園 *3-5-8* ）

(3) プログラム

基調講演：ソフトウェア開発支援環境の現状と動向（ *13:30* ― *14:30* ）
岸田 孝一 （ソフトウェア・リサーチ・アソシエイツ）

パネル討論：実践的開発環境の構築をめざして （ *15:00* ― *17:00* ）
コーディネータ： 安間 文彦 （情報処理振興事業協会）*
パネリスト ： 熊谷 章 （パナファコム）
佐藤 隆 （インテック）*
佐藤 千明 （長野県協同電算）*
盛田 政敏 （神戸コンピューターサービス）

（*は交渉中）

(4) 参加費： *1* 名当たり *10,000* 円（*SEA*会員）， *15,000* 円（一般）

(5) テキスト： 第１回*SEA*ワークショップ報告書（実践的ソフトウェア開発に関する集中討論）予価*5,000*円を参加者全員に配布します。

(6) 申し込み及び参加方法： 所定の申し込み票にご記入の上，下記宛てお送り下さい。折り返し受講票及び請求書をお送りします。ただし，定員（*200*名）になり次第，締め切らせていただきますので，あしからずご了承下さい。

〒 *166* 東京都杉並区高円寺南 *1-5-4* 高円寺サンハイツ *404*
ソフトウェア技術者協会
FAX 03-262-9719

問い合わせ：*forum*専用電話（*03-238-9909*）までお願いします。

---

*SEA Forum (April '86)*参加申込書

氏 名：＿＿＿＿＿＿＿＿＿＿（ふりがな：＿＿＿＿＿＿＿＿） 会費扱い（○印で指定）：会員，一般

会社名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿部門：＿＿＿＿＿＿＿＿役職：＿＿＿＿＿＿＿＿

住 所：（〒 ） ＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

TEL：＿＿＿＿ - ＿＿＿＿ - ＿＿＿＿（内線 ＿＿＿＿）

備考(m):

ソフトウェア技術者協会入会申込書（正会員）

（フリガナ）
氏　名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

勤務先名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

勤務先住所：〒(　　)＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

勤務先TEL：＿＿＿＿＿＿＿＿＿＿＿＿

自宅住所：〒(　　)＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

自宅TEL：＿＿＿＿＿＿＿＿＿＿＿＿

連絡先（どちらかにチェックしてください）　　□勤務先 □自宅

年令＿＿才　　性別（男・女）　　血液型（A・O・B・AB）

会費：　入会金3千円　＋　年会費（入会より1ヶ年分*）7千円　＝　1万円

ソフトウェア技術者協会入会申込書（賛助会員）

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

住所：〒(　　)＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

電話：＿＿＿＿＿＿＿＿＿＿＿＿

連絡担当者：
（フリガナ）
氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

所属：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

賛助会費：＿＿＿＿口（1口　5万円）　ただし入会より1ヶ年分*

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

＜会費振込先＞：三菱銀行本店公務部
普通預金口座 No. 0004830
口座名：ソフトウェア技術者協会
なるべく添付の振込用紙をお使い下さい。

*：　61年2月26日の幹事会での細則決定による

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