SEAMAIL

Newsletter from Software Engineers Association

Volume 8, Number 3 August, 1993

目　次

ソフトウェア技術者協会
Software Engineers Asociation

ソフトウェア技術者協会(SEA)は，ソフトウェアハウス，コンピュータメーカ，計算センタ，エンドユーザ，大学，研究所など，それぞれ異なった環境に置かれているソフトウェア技術者または研究者が，そうした社会組織の壁を越えて，各自の経験や技術を自由に交流しあうための「場」として，1985年12月に設立されました．

その主な活動は，機関誌SEAMAILの発行，支部および研究分科会の運営，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，および内外の関係諸団体との交流です．発足当初約200人にすぎなかった会員数もその後飛躍的に増加し，現在，北は北海道から南は沖縄まで，900余名を越えるメンバーを擁するにいたりました．法人賛助会員も50社を数えます．支部は，東京以外に，関西，横浜，長野，名古屋，九州，東北の各地区で設立されており，その他の地域でも設立準備をしています．分科会は，東京，関西，名古屋で，それぞれいくつかが活動しており，その他の支部でも，月例会やフォーラムが定期的に開催されています．

「現在のソフトウェア界における最大の課題は，技術移転の促進である」といわれています．これまでわが国には，そのための適切な社会的メカニズムが欠けていたように思われます．SEAは，そうした欠落を補うべく，これからますます活発な活動を展開して行きたいと考えています．いままで日本にはなかったこの新しいプロフェッショナル・ソサイエティの発展のために，ぜひとも，あなたのお力を貸してください．

ＳＥＡＭＡＩＬ　Ｖｏｌ．８，Ｎｏ．３　　１９９３年８月３１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）
〒１６０　東京都新宿区四谷３－１２　丸正ビル５Ｆ
TEL: 03-3356-1077　FAX: 03-3356-1072
印刷所　サンビルド印刷株式会社　　〒１６２　東京都新宿区築地町８番地
定価　５００円（禁無断転載）

事務局から

☆

残暑お見舞い申し上げます．冷夏のためか夏風邪が流行っているようですが皆様にはいかがお過ごしでしょうか?? なぜか，わがSEAMAILは順調に毎月発行が続き，ここにVol.8, No.3をお届けします．これもまた，エルニーニョ現象の影響か(!?),

☆☆

巻頭に載せた青木さんの原稿は，Edward Yourdon氏編集のAmerican Programmer誌Vol.6, No.7 (July, 1993)に寄稿されたものの日本語訳です．実際は，この日本語が最初にあって，それを英語に直すのに悪戦苦闘されたとか，風の便りに聞いていますが....

☆☆☆

新幹事一番の張りきりボーイ（いや，おじさんかな? 失礼）大場さんからは，前号に引き続いて，SEAリストラ計画の第2弾が寄せられました．実は，No.3もすでに編集長のところに届いているそうです．これをきっかけに，これからSEAをどうして行くべきかについて，会員のみなさんの積極的な御意見をお待ちしています．

☆☆☆☆

編集長からのかなり強引な執筆依頼に応えて，今月は3人の幹事の方々から原稿が届きました．しかも，伊藤さんのはなんと16ページもの長編レポートでした．この分だと，あと1～2ヶ月は，幹事会メンバーが入れ替わり立ち変わり誌面に登場してくれそうですが，一般会員の方々もぜひ，エッセイや論文をお寄せください．

☆☆☆☆☆

夏がとうとう来ないうちに，もうすぐ秋．若手の会，泰安シンポジウム，教育ワークショップ，信頼性シンポジウムと，またイベントが目白押しに並んでいます! 月例Seminar & Forumも始まるし，だれか助けて!

☆☆☆☆☆☆

ソフトウェア技術者の「筏」

ピープルウェアにおけるオブジェクト指向の影響

青木 淳
(SRA)

1. はじめに

私は，3年前，日本のある会社において，オブジェクト指向の作業環境を構築するというプロジェクトを組織しました．このプロジェクトを通して，オブジェクト指向に関する多くのことを学んだのですが，人的資産を，どのように，捜し，雇い，教え，動機づけし，育て，管理したのかというピープルウェアについても多くのことを学びました．ここに書かれていることは，技術的というよりも，日本のソフトウェア開発組織における人間的側面に関するエッセイです．

2. プロジェクトの成果

まず，どのようにプロジェクトを遂行したのかに言及する前に，そのプロジェクトの成果を先に述べたいと思います．Smalltalk技術者の小さなチームが，各種のワークステーションをネットワークで接続した環境上に，かなりの大きさのオブジェクト指向プログラムを作成しました．クラス数は約700個，メソッド数は約14,000個でした．既存のクラスライブラリを合算すると，クラス数は約1,000個，メソッド数は約25,000個となりました．作成したものを下記に示します．

(1) オブジェクト指向のためのCASEツール(ObjectCast)
(2) RDBまたはOODBを利用したリポジトリのための基本機構
(3) オブジェクト指向的な3次元グラフィカルライブラリ

ここで開発されたものの一部であるオブジェクト指向のためのCASEツールは，ObjectCastとして，実際に日本市場で販売されています．

3. プロジェクトの発案発足

会社内にオブジェクト指向の技術者を養成できないか，あわよくば，会社自体がオブジェクト指向技術で業界のリーダシップを獲得できないものか，できればプロフィットも生みたいと，先新技術に明るい先輩と私は思案を重ねていました．マルチクライアント方式の実践的な研究開発プロジェクトを起こすという案が，二人の中に芽生えてきました．自分たちがサーバになり，オブジェクト指向の技術動向を調査研究するとともに，その成果をクライアントへ提供し，調査だけではなく，Smalltalkというオブジェクト指向の環境をベースにして，その上に有用なものを構築し，後々それらを販売できるようにしようと考えたのです．

社長や常務などのトップマネージャに嘆願書などを書き，何回もミーティングを設けて，このプロジェクトの発足を熱意で迫りました．儲かる市場を捜すのではなく，自分たちで市場を作り，そこへみんなを誘導することの大切さを訴え，そのための企画案，詳細な計画書などをこしらえていきました．会社にオブジェクト指向の色を鮮明につけるために，国内に限らず，世界中の有識者を招いて，講演会や技術セミナを開催することなども企画されました．会社の対外的なイメージアップと内部的なオブジェクト指向技術力の向上を大前提として，トップマネージャから許可を取り付けることができたのです．

このプロジェクトの参加費はクライアントあたり約1,000万円となり，期間や成果物の取り扱いなど，金銭

面や法律面などの詳細が決められました．プロジェクトへの参加を呼びかけるために，Yourdon 氏を招いた講演会を催し，Blizz ard-90 プロジェクトの発足を宣伝しました．この頃，Yourdon 氏は，Coad 氏とともに，オブジェクト指向分析の本を出版されたばかりでした [4]．この講演会は大成功でした．また，それに便乗した宣伝活動も功を奏し，10 社にのぼるクライアントを獲得して，予想以上に好調な滑り出しでした．

4. プロジェクトのメンバ募集

私たちは，このプロジェクトを構成するメンバを，社内公募制を採用して集めました．このプロジェクトに興味を持って参加したいと思っている人に，自分の動機や意欲や将来などを書いたポジションペーパ (小論文) を提出してもらい，このプロジェクトに適切な人材であるかを判断しました．また，会社に入社を希望してくる人の情報をトップマネージャから見せてもらい，適切な経歴の持ち主を外部から発掘することにも力を尽くしました．

このプロジェクトのために，ポジションペーパを提出するような意欲のある人は，その人が所属する部門の切り札的な存在であり，そのような人を，その部門から引っぱって来るためには，大きな抵抗にあいます．また，人事部門からも，相当な抵抗がありました．人事に関係ない技術者が，特別な人事権を持ったようなものですから当然の帰結でしょう．これらの抵抗に対抗するためには，トップマネージャからの同意を取り付けておくことが大切なことです．

このようにして，技術者 10 名とスタッフ 5 名のメンバを集めることができました．各メンバへの動機づけとして，まず従来の慣習や常識というものを疑ってかかり，もう一度考察し直し，プロジェクトに適合しないときには，あっさりと棄て去ることにしました．各々のプロジェクトメンバの中に，新しいことや変わったことをやるのだという意識を刷り込んだのです．明日のことを自分たちで決めるということは，既成の管理組織の中に埋もれていた何かを刺激するようでした．

従来の慣習や常識というものを疑ってかかるということを，プロジェクトメンバが自覚するために，実際に用いた絵を紹介しましょう．図1 と図 2 に示した二枚の絵を見て下さい．図 1 は直方体であり，図 2 が人間の顔らしいことが分かります．

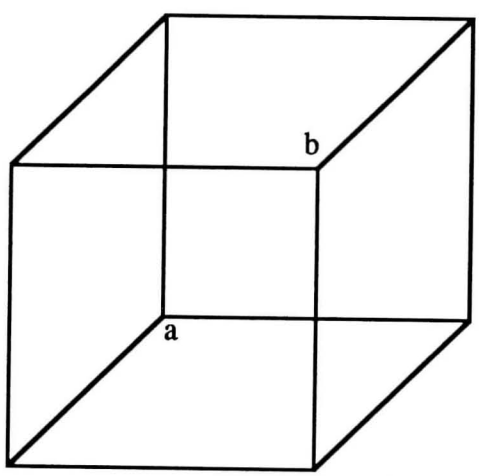

図 1. 直方体のワイヤフレーム

直方体の方はワイヤフレームで表現されており，図中の a 点と b 点のどちらが前にあっても，直方体に見ることができます．つまり前後を簡単に逆転できます．また，a 点と b 点が同一平面上にあるように見ることもできるでしょう．一方，人間の顔らしい絵ですが，実は，これは凹面なのです．皆さんが読まれているエッセ

イの上から水をたらせば，人間の顔の中に水がたまります．これはデスマスクと呼ばれるもので，人間が死んだ際に，顔の上に石膏を塗って，その形を取ったものです [10]．この凹面のデスマスクが直方体のワイヤフレームと異なる点は，簡単に前後 (凹凸) を逆転できないことでしょう．凹面なのに凸面にしか見ることができません．幼い頃から私たちが，人間の顔が凸面であると思い込んでいることに由来するのです．

図 2.　人間のデスマスク

この思い込みを常識とか体系と呼びます．口さがない言い方をすれば偏見です．常識とは認知の可塑性を制限して生まれたものなのです．この二枚の絵を提示した理由は，私たちがソフトウェアと呼ばれるものを考える際に，デスマスクを見るようになっていないか，常識や偏見にとらわれ過ぎていないかということを自省するためです．プロジェクトメンバの合言葉は「デスマスクになるな」でした．

5. プロジェクトの組織構造

日本の多くの会社の勢力構造は，ピラミッドのような階層構造であり，終身雇用に基づいた年功序列と呼ばれる制度を採用しています．また，各部門への配属は，人事部と呼ばれる部門が一手に握っています．

このプロジェクトでは，ピラミッドのような階層的な勢力構造の組織に依存しない平坦な勢力構造の機動部隊をつくることにしました．どうしてこのようにしたかというと，古典になりますが，心理学者である French 氏の社会勢力構造の形式理論に基づきます [7]．勢力構造の如何によって，いろいろな意見がどのように変化するかを表しており，図 3 のようになります．

左端に書かれた A,B,C,D は個人であり，これら個人の間に引かれた矢印は，始点の個人が終点の個人より勢力が強いことを表わします．A->B を企業の組織で考えれば，A が B の上司となります．最初，A,B,C,D の個人は異なった意見を持っています．時間が経過するにしたがって，勢力構造の影響を受けて，個人の意見が推移していく様子が見て取れます．

(1) と (2) の勢力構造では，意見の一斉化がみられません．(1) はプロジェクトのメンバが二つのグループに割れているような状況であり，組織の力を発揮することが困難です．(2) も同様であり，図中の B は二つの勢力の板挟みにあって戸惑っています．(1) と (2) の勢力構造は，このプロジェクトの組織構造として不適当となります．

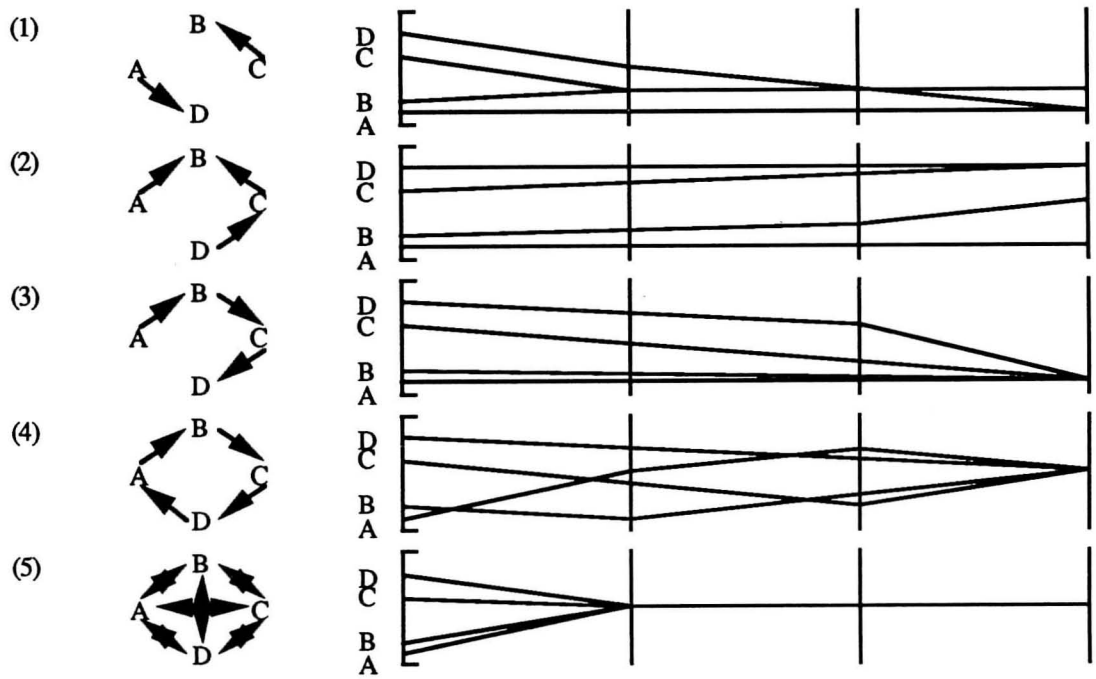

図 3. 勢力構造の恐怖

通常の企業組織の勢力構造はピラミッドのような階層的な構造をしているので，図 3 の (3) に対応します．例えば，D が社員，C が課長，B が部長，A が社長に相当します．(3) の勢力構造での意見の推移をみると，知らず知らずの間に勢力の強い人の意見になびいていきます．結局，上司の意見になるのです．ウォークスルーを通して話し合い，意見を調整する意義が薄くなります．意見の一斉化がみられるものの，その意見は組織内において妥当な意見とは言い難くなります．

(4) と (5) の勢力構造でも，意見の一斉化がみられます．(4) の方は時間がかかりますが，妥当な意見に収斂していきます．一方，(5) の方は短時間で意見の一斉化が達成されます．ウォークスルーを通して話し合い，意見を調整する意義があります．

このプロジェクトには，(4) または (5) の勢力構造が適当と思われました．私たちは，自分たちのことを，従来のどこの部門にも属さず，平坦な勢力構造を持ち，みんなが解け合っているという意味で「ジェルドチーム」と呼びました．

6. 作業スペースのレイアウト

私たちは，自らが働くスペースを第一に考えました．高速なコンピュータや高性能のネットーワークの導入を考えるのは二番目にしたのです．自らの働くスペースを自らによってレイアウトすることの重要性は，パタン･ランゲージやピープルウェアなどの文献に指摘されています [1][5]．日本のほとんどのソウトウェア開発の作業スペースは，図 4 の (a) から (e) のどれかに当てはまります．

(a) は，多くの会社で見かけるオフィスのスタイルです．窓側に部門長 (部長や課長など) の机があり，その前には，成員の机が対面するように置かれています．その横に描かれているのは，大きな机とホワイトボードで，ミーティングスペースを意味しています．私たち日本人は，あるスペースを確保すると，暗黙のうちに，このような机の配置をしてしまうのです．(b) は，(a) の作業環境に，ガラス張りの部屋が加わります．この部屋は完全な空調設備がなされ，大きなコンピュータ (メインフレーム) が中に配置されて，電子計算機室などと呼ばれます．しかし，技術者の働くスペースは(a) と同様です．(c) は，(b) で登場した電子計算機室の外側に端末と

呼ばれるものが出現します．しかし，技術者の働くスペースは (a) と変わりばえしません．(d) は，パーソナルコンピュータやワークステーションと呼ばれるものが広範に使用されるようになり，メインフレームコンピュータがなりを潜め，電子計算機室などが消失します．そのかわり，小さなコンピュータが，技術者の働くスペースに入り込んできます．(e) は，(d) の作業環境に仕切りが設けられ，個人のスペースを確立したように錯覚させられています．しかし実のところ，トイレの中で仕事をしているのとさして変わりがない状況なのです．

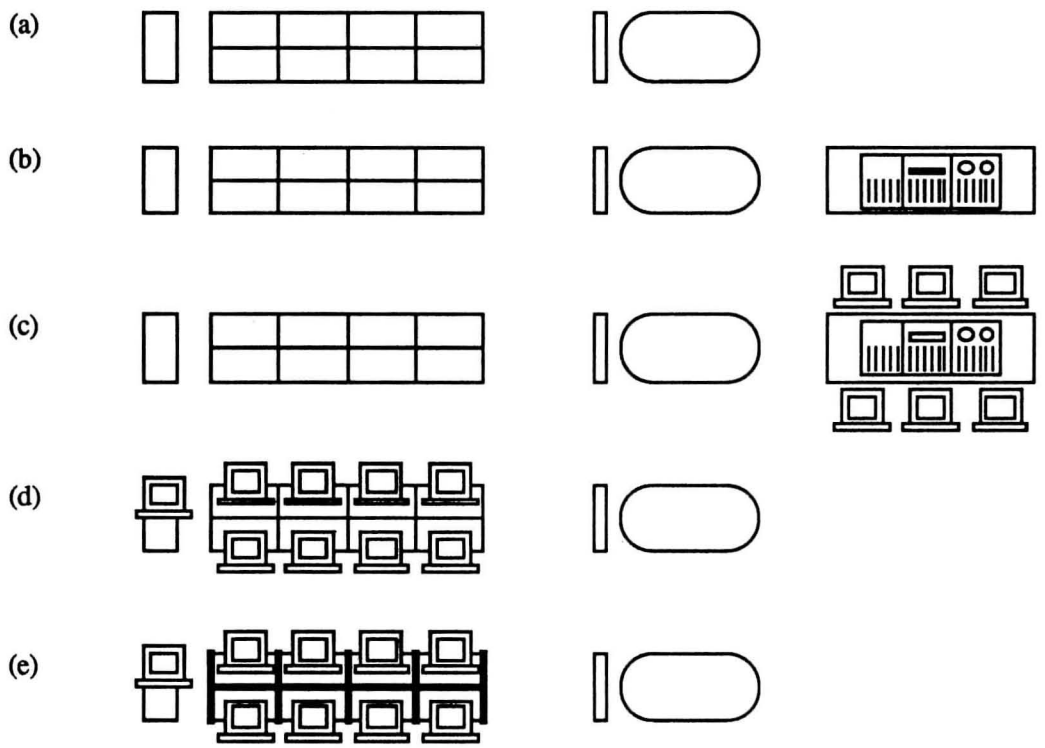

図 4. 作業スペースの変遷

以上，私たちの作業環境を概観しましたが，(a) から (e) に共通しているところは，会社のどこへいっても似たような構造が続き，オフィスの外に出ても，近代建築の画一さに延長されて，まるで金太郎飴です．はたして，これが私たち裁量労働者の作業環境なのでしょうか．

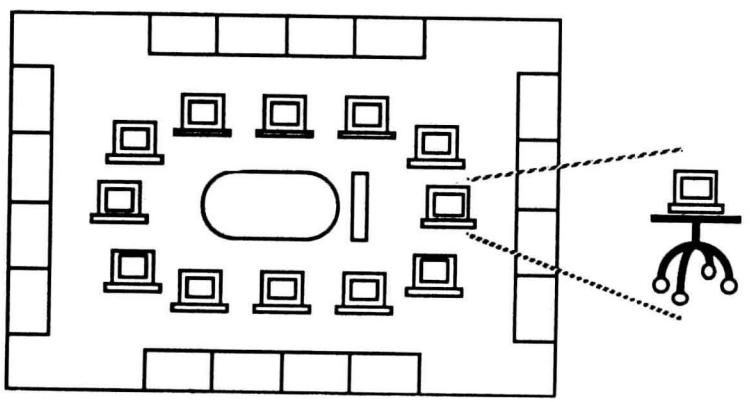

図 5. 試みた作業スペースのレイアウト

私たちは，プロジェクトの作業スペースとして，図4の(a)から(e)に示したような作業スペースを捨て去り，独立したフロアを確保して，図5に示すようなレイアウトを実際に行なってみました．ウォークスルーやレビューをするためのミィーテングスペースを真ん中に配置して，そのまわりに各個人の机を配置したのです．そして，その間をキャスタ付きの台に乗せたコンピュータを配置しました．

コンピュータの画面を手軽に拡大して映し出せるプロジェクタや，ビデオカメラと再生装置，プレゼンテーションツールなどの購入を行いました．コンピュータは，ドキュメント作成用のワークステーションとしても，プログラム開発用のワークステーションとしても機能するものを選定し，必ずネットワークに接続させて，国内や海外を問わずアクセスする状況を構築しました．

これは成功でした．まわりの画一化された空間とは異にする作業環境を作り出し，プロジェクトメンバの士気も向上し，開発に対する活力も増加しました．その上，ミィーテングスペースの予約をしなくとも，いつでもウォークスルーやレビューができる空間があるので，開発時の困難などに迅速に対応するための話し合いが度々行なわれました．コンピュータの画面を投影するプロジェクタやプレゼンテーションツールの有効利用も可能となり，自分たちが開発したものをビデオカメラで録画し，自分たちの活動を内外の人々に知ってもらおうという努力も自発的になされました．

6. プロジェクトの技術教育

プロジェクトの技術教育として，自らのための教育カリキュラムおよび教材を作成しました．言語やOSの使い方などの教育よりも，ソフトウェア工学的な教育を重視した内容です．オブジェクト指向も，ソフトウェア工学全般から眺めるような視点で捉えました．後になって，この教育システムは，社外や社内を問わない教育コースに昇格し，教育事業として発足しました．

また，プロジェクト進行中に，オブジェクト指向に関する調査研究の目的で，世界中の有識者を招いたセミナを，このプロジェクトで企画し開催しました．Youdon氏に続いて，Coad氏，Jacobson氏，Rumbaugh氏など，研究一辺倒ではなく，実践的に活動されている方々を次々に招きました．様々なシンポジウムやフォーラムそしてワークショップへ，プロジェクトメンバを参加させるよりも，最新情報を得るためのより積極的な方法といえるでしょう．なぜなら，主催が自らの会社であり，自分たちのプロジェクトであるわけですから，対外的に相当なイメージアップになりました．

その上，Rumbaugh氏らが書かれたOMTの本を日本語に翻訳する仕事も行われ，その翻訳が日本で出版されるに至りました[8]．私自身も，このプロジェクトの経験を踏まえて，オブジェクト指向分析設計に関する本を著作し，日本で出版されています[7]．

7. プロジェクトの評価方法

オブジェクト指向のソフトウェアを作る喜びや誇りは，自分が作成したクラスが，今まで先人によって開発されてきたクラスライブラリの中のどこに位置するのかを見定めることから始まります．そして，改良や拡張を加え，その位置がどのように変遷していくのかを観察することです．部分が全体で占める場所を確認し，その部分が全体に不可欠であることを知ったとき，大きな喜びと誇りを味わうでしょう．

そのために，クラスライブラリの中で，あるクラスの位置を数値化する三つの指標を，このプロジェクトで考案しました．この三つの指標を利用して，クラスライブラリの位相空間を創り出し，あるクラスと他のクラスが似ているか似ていないかを距離(遠近)のようなものに翻訳するのです．三つの指標は，図6に示す三つの式を利用して求めました．

最初の汎化-特化構造の指標(HF)は，きわめて単純ですが，抽象クラスの数を大まかに見積る指標になります．二番目の全体-部分構造の指標(RF)は，クラスライブラリ内で，部品としてのクラス種を見積る指標にな

ります．最後のポリモルフィズムの指標 (PF) は，クラスライブラリ内で，どれだけ語彙 (メッセージ名) が統一されているのかを見積る指標になります．

$$HF_A = \frac{\text{number of superclasses of class A}}{\text{number of superclasses of class A} + 1 + \text{number of subclasses of class A}}$$

$$RF_A = \frac{\text{ranking number of class A in topological sort}}{\text{total number of classes (total number of elements in topological sort)}}$$

$$PF_A = \frac{\text{number of messages that is defined in class A and other classes}}{\text{number of messages that is defined in class A}}$$

図 6. 評価のための三つの指標

測定結果を図 7 に示します．このような指標を，開発段階に順次測定して履歴を取ると，開発過程の視覚化 (アニメーション) が可能です．オブジェクト指向の開発過程である間違い削減プロセスを，形の変遷として観察することができます．

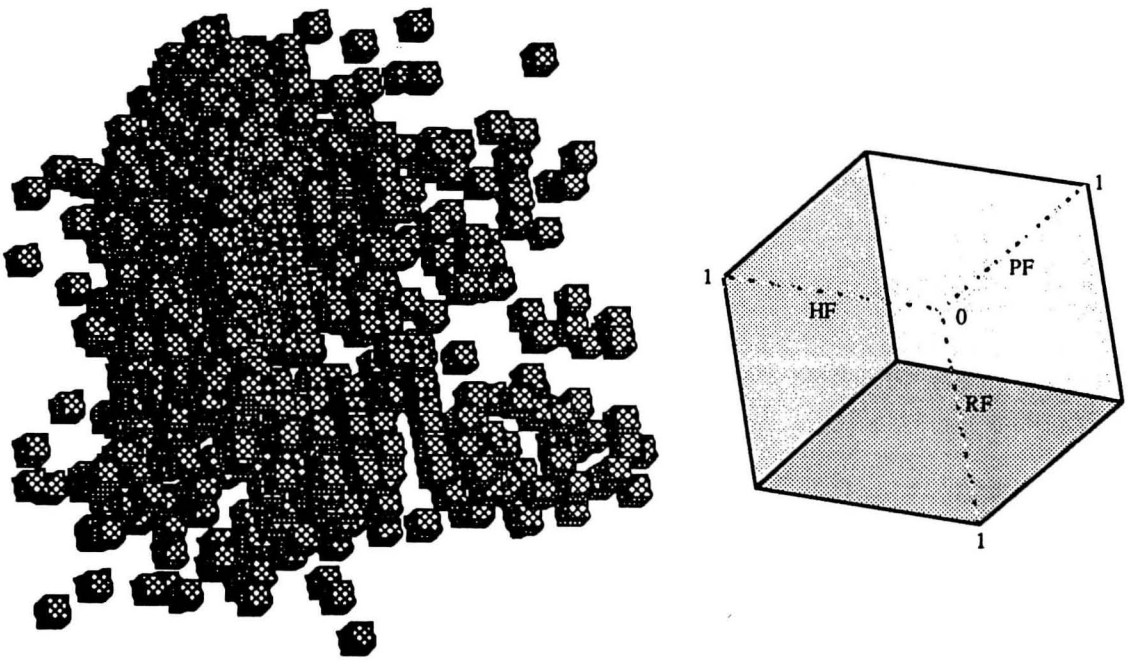

図 7. オブジェクト指向クラスライブラリの位相空間

これらの詳細は，いくつかの文献 [2][3][6] を参照してもらうことにして，ここでは，HF と RF の相関だけを考察してみましょう．HF を横軸に，RF をと縦軸にとると，図 8 に示すような傾向が得られます．HF も RF も低いクラスは，抽象的で部品として使われるので，ライブラリとしての性質が強くなります．逆に，HF も RF も高いクラスは，具体的で多くの部品を必要とするので，アプリケーションとしての性質が強くなります．

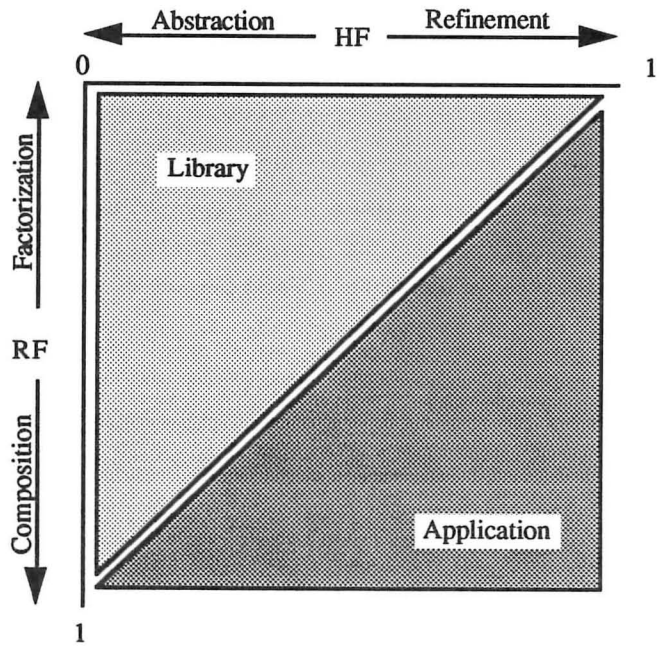

図 8. HF-RF の相関

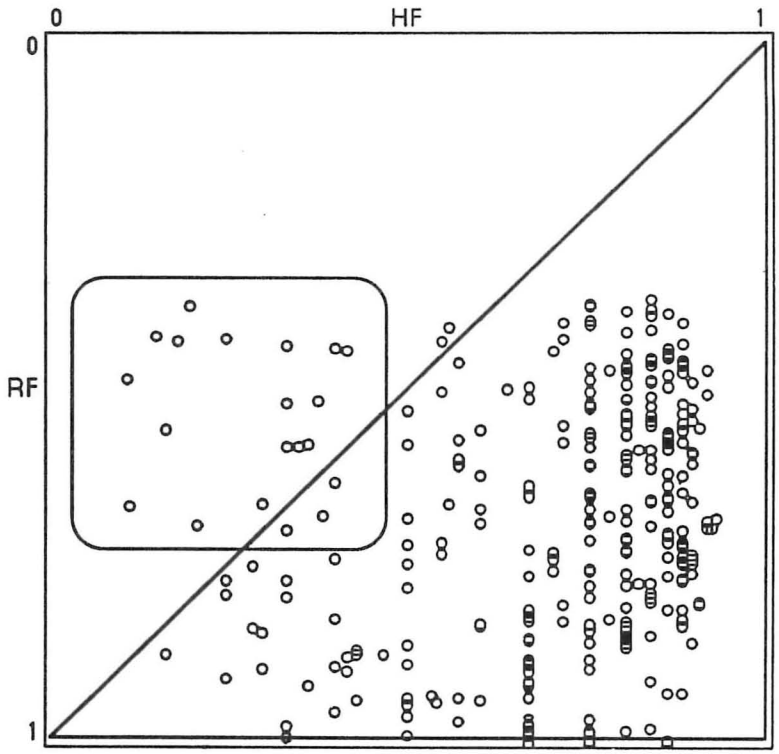

図 9. プロジェクトの HF-RF

図9は，その中から，私たちが作成したクラス群だけを取り出してプロットしたものです．点がプロットされていない場所は，既存のクラスライブラリが存在していた場所です．この図によって，私たちは先人が作成したしたクラスライブラリを利用して，アプリケーションを作っていたのだという結果が顕著に出ていることになります．しかし，多少ですが，私たちが作成したクラスの中に，ライブラリとしてのクラスが含まれていたことも示されています．

ところが，図9をプロジェクトのメンバで色分けをしてみた結果，二つのグループに峻別できることが判明しました．一つのグループは図10に示すようなプロット結果になり，フレームワーク開発グループと名付けました．もう一つのグループは図11に示すようなプロット結果なり，アプリケーション開発グループと名付けました．

フレームワーク開発グループが作成したクラス群は，図10に示すように，多くの抽象クラスを含んでいます．このグループに属しているメンバは，モデル，パラダイム，方法論，理論などに興味があり，メタシステムを作成する能力や抽象能力を有しています．クラスライブラリの根幹になる部分を作成し，少々の具体的な例示するだけに留めて，ある問題領域に特化したようなクラスを作成することを好まないようです．

一方，アプリケーション開発グループが作成したクラス群は，図11に示すように，抽象クラスを含まず，多量の具象クラスのみを作成しています．このグループに属しているメンバは，スタイル，ルック＆フィール，モード，移植などに興味があり，様々な実例を組み合わせる能力や具象能力を有しています．クラスライブラリの応用事例をたくさん作成し，誰かが作成したベースになるものを改良したり拡張したりすることを好むようです．

このプロジェクトでは，3:7の割合でフレームワーク開発グループよりもアプリケーション開発グループに属すメンバが多かったです．実際のプロジェクトを遂行するには，フレームワーク開発グループも，アプリケーション開発グループも，どちらも必要です．どちらが偉いわけでもありません．この二つのグループが力を合わせてこそ，より良いソフトウェアシステムが開発できるのです．

7. おわりに

このプロジェクトは二年間で終了しました．その成果は，20個程度の分厚いバインダと幾つかの光磁気ディスクとして，スチール製のキャビネットの中に格納されています．このプロジェクトのメンバも，ばらばらになり，新たなところで頑張っています．また，開発されたものの一部は，Coad/Yourdon法に基づくオブジェクト指向の分析ツールObjectCastとして，実際に日本で販売されています．しかし，本当の成果とは何だったのでしょうか．

「ここに，ひとりの人がいて，長い旅を続け，とあるところで大きな河を見て，こう思った．この河のこちらの岸は危いが，向こうの岸は安らかに見える．そこで，筏を作り，その筏によって，向こうの岸に安らかに着くことができた．そこで『この筏は，私を安らかにこちらの岸へ渡してくれた．大変役に立った筏である．だから，この筏を捨てることなく，肩に担いで，行く先へ持って行こう.』と思ったのである．このとき，この人は筏に対して，しなければならないことをしたといわれるであろうか．そうではない．この比喩は，『正しいことさえ執着すべきではなく，捨て離れなければならない．まして，正しくないことは，なおさら捨てなければならない.』ということを示している.」----パーリ，中部3-22「蛇喩経」[11]

本当の成果は，オブジェクト指向のような技術が，より良いソフトウェアを開発するための「筏」であったことを，プロジェクトのメンバ全員がしっかりと認識したことではないかと思います．上述した様々なピープルウェア的な試みも，新しいことを行なうプロジェクトに大きく寄与しました．しかし，多くの方法論や新しい技術，そして様々なピープルウェア的な試みも，すべてがソフトウェア技術者が乗るための「筏」に過ぎないのです．私たちは色々な執着から離れていなければなりません．すべては終わりのない変化なのですから．

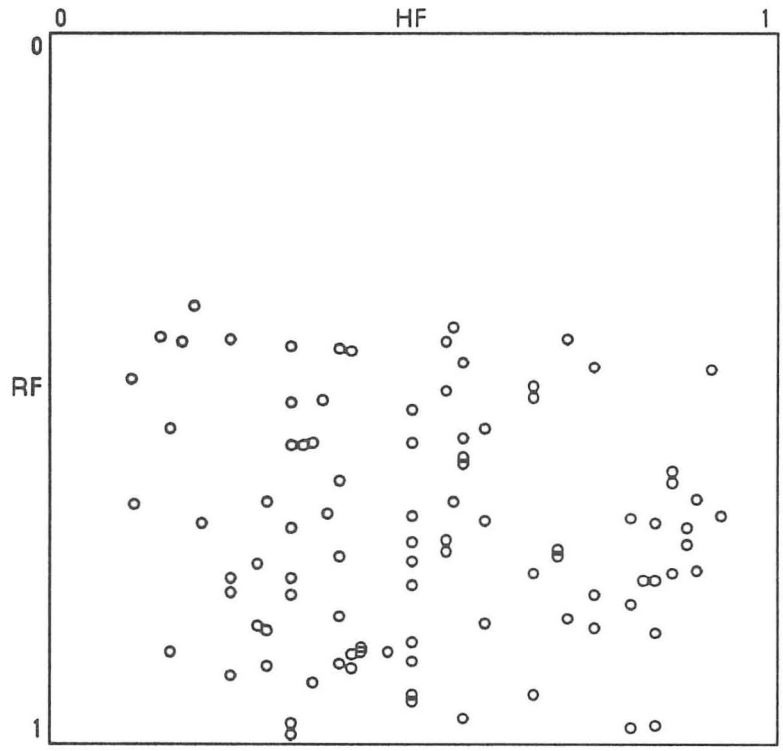

図 10. フレームワーク開発グループの HF-RF

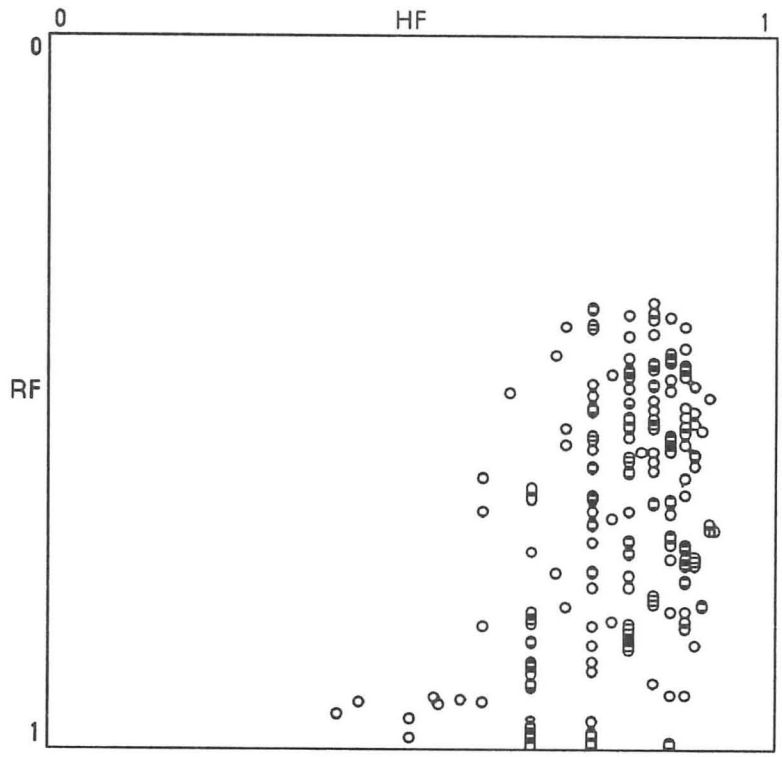

図 11. アプリケーション開発グループの HF-R

謝 辞

このエッセイを執筆するにあたっていろいろと助力をいただいた Software Engineering Research の Lloyd G. Williams さん, SRA Boulfer Lab の酒匂寛さん, 林香さん, 中小路久美代さんにお礼を申し上げます. また, FXIS の藤野晃延さんと Blizzard-90 プロジェクトの皆さんに感謝いたします.

参考文献

[1] C. Alexander, S. Ishikawa, and M. Silverstein, with M. Jacobson, I niversity Press, 1977. (平田翰那訳. パタン・ランゲージ. 東京: 鹿島出版会, 1984).

[2] 青木淳. オブジェクト指向プログラミング環境におけるインテグレーション過程, ソフトウェア・シンポジウム'90 論文集, pp.168-175. 東京: ソフトウェア技術者協会, 1990.

[3] 青木淳. オブジェクト指向システム分析設計入門. 東京: ソフト・リサーチ・センター, 1993.

[4] P. Coad, and E. Yourdon. Object-Oriented Analysis, 2nd ed. Englewo od Cliffs, New Jersey: Yourdon Press/Prentice Hall, 1990.

[5] T. DeMarco, and T. Lister. Peopleware. New York: Doeset House, 198 7. (日立ソフトウェアエンジニアリング生産技術研究会訳. ピープルウェア.
東京: 日経 BP 社, 1989).

[6] G. Fischer, D. Redmiles, L. Williams, G. Puhr, A. Aoki, and K. Nak akoji. "Beyond Object-Oriented Development: Where Current Object-Orien ted Approaches Fall Short," Colorado University, 1993. (Submitted to t he Special Issue of Human-Computer Interaction on "Empirical Studies o f Object-Oriented Design").

[7] J.R.P.French, Jr. "A Formal Theory of Social Power," Psychological
Review, Vol. 3 (1956), pp. 181-195.

[8] Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. O bject-Oriented Modeling and Design. Englewood Cliffs, New Jersey: Pren tice-Hall, 1991. (羽生田栄一監訳. オブジェクト指向方法論 OMT. 東京: トッパン, 1992).

[9] 仏教聖典. 東京: 仏教伝道教会, 1985.

[10] 養老孟司. ヒトの見方. 東京: 筑摩書房, 1985.

SEAのリストラクチャリングについて

(その2)

大場 充
(日本IBM)

残暑お見舞い申し上げます.

IBMの大場です. 新任の幹事として, 暑さにもめげず頑張っています. 先号に続きまして, 今回も『SEAのリストラクチャリング』について議論したいと思います. あまりパッとしない, 暗い, 景気のよくない話ですが, 明日のSEAのために, みなさんもどうか一緒に考えてみてください.

前回は, 今日の情報産業のおかれた環境を考えると, SEAの『リストラクチャリング』が急務であることについて, 私なりの見解を述べさせていただきました. 今回は, 『SEAのダウンサイジング』を, どう実施するかについての私なりの分析を述べます. これ以外のアイデアも, いくつかあるかも知れません. 今後, この問題に関して, 会員の間で, さまざまな, よいアイデアが提案され, 議論され, 検討されてゆくことを期待します.

まず, SEAの財政を見てみましょう.

SEAの運営は, 利益を出すことを目標としているわけではありません. しかし, この協会の健全な運営と発展のためには, それなりの『健全な経済的基盤』が必要です. 私は, ソフトウェア・メトリクスやソフトウェア品質管理なども研究して来たせいか, 長期的な意味で, 『組織の基盤が組織の発展に与える影響は大きい』と信じています. この観点から, 現在の協会の経済状態を見ると, よくいえば『非常にダイナミック』, 悪くいえば『自転車操業』といえるでしょう.

これは, 一概に悪いこととはいえません. 組織の誕生期から成長期には, そのような『ダイナミック』な活力が, 成長の原動力になるからです. しかし, 組織がいったん成長期から成熟期に移行した後では, この『ダイナミックさ』は, 長期的な展望のない, その日暮らし的な組織運営の原因になりかねません. 私は, SEAは『組織として, 既にその成熟期に入っている』と見ます. もし, これが正しいとすれば, 『ダイナミックな運営』はそろそろ危険になって来ている頃ではないでしょうか.

支出の部		収入の部	
変動費	9,391,959	会員会費収入	6,279,000
固定費	9,092,804	雑収入	10,712,941
計	18,484763	計	16,991,941

表1. 1992年度SEA会計報告

表1を見てください. 昨年度の協会の会計報告です. これを見ると, 協会の現在の財政状態がわかります. 現時点では, 『収入と支出はほぼバランス』しています. いま, 問題があるという訳ではありません. しかしよく見ると,

(1) 前年度からの繰り越しや賛助会員の会費収入によって救われていること,
(2) 支出が会員個人からの会費収入を上回っていること,

がわかります. 経済用語でいえば, 繰り越し金は『ストック』であり, 会費収入は『入ってくるフロー』, 支出は『出て行くフロー』です. 出て行くフロー(流れ)が, 入ってくるフロー(流れ)よりも大きければ, ストック(たまっている水量)のレベルは徐々に下がってゆき, 最後にはなくなります. 『ストックを食いつぶす』のです.

同じことを, もっとわかりやすい(?)微分方程式で説明しましょう. SEAの財政状態を, 変数zで表します. 変数xは収入, 変数yは支出です. ある時点における財政状態の変化(zの微分)は, そのときの収入と支出の差です. つまり,

$$z' = x - y$$

です. ここで, 時刻tにおけるSEAの会員数を変数w(t)で表せば,

$$x = a \cdot w(t) + b$$

$$y = c \cdot w(t) + d$$

となります．ただし，a, b, c, d は定数です．a は，会員 1 人当たりの会費です．b は，その他の会員数に関係ない収入 (賛助会員からの会費やセミナーの事業収入など) です．c は，会員一人当たりにかかる諸費用 (SEAMAIL の印刷費や郵便費など), d は，協会の運営に必要な固定費 (事務所の家賃や固定資産の償却，人件費など) です．

z に関する微分方程式を整理すると，

$$z' = (a - c)w(t) + b - d$$

が得られます．ここで，簡単のために $e = a - c$, $f = b - d$ とすれば，

$$z' = e \cdot w(t) + f$$

となります．この微分方程式は，簡単な「変数分離形」です．高校の数学を思い出してください．この形の微分方程式は，両辺をそれぞれ積分して解が得られます．いま，簡単のために

$$w(t) = \alpha t + \beta$$

とします．そして，左辺を z，右辺を独立変数 t で積分します．すると，

$$z = e\alpha \ t**2 / 2 + (e\beta + f)t + K$$

となります．ここで，K は，有名な積分定数です．

得られた式をよく見てみましょう．z は，独立変数 t(時間) に対して 2 次式で説明されています．したがって，α が正か負か，またはゼロかによって，結果は大きく違って来ます．つまり，協会の会員数が「増加している」のか，「減少傾向にある」のか，「定常で変化しない」のかによって，結論が大きく変わります．会員数が「増加傾向にある」ならば，一時的な，少々の赤字は問題になりません．これが，世にいわれる「バブル経済」の実態です．反対に「減少傾向にある」ならは，少しの赤字でも，2 ～ 3 年続けば，財政は破綻するでしょう．これは，会員数の変化を示すパラメータ α が「ゼロ」でなければ，時間の経過の 2 乗の速度で，財政状態が変化するからです．

ここでは，最も現実的と思われる，「会員数が変化していない」とする仮定で，検討を進めてみましょう．その場合，z は，独立変数 t に対して線型な 1 次式表現されます．すなわち，

$$z = (e\beta + f)\ t + K$$

となります．したがって，$e\beta + f \geqq 0$ であれば，財政は健全で赤字になる心配はなく，そうでなければ，財政は不健全で，いつか赤字になります．$e\beta$ は，「会員の会費から会員一人当たりにかかる費用を差し引いたものに会員数を乗じて得られる」ものです．また，f は，「雑収入から協会運営のための固定費を差し引いて得られる」ものです．昨年度の実績からいえば，$e\beta$ が「約 311 万円のマイナス」であり，f が「約 162 万円のプラス (黒字!)」です．つまり，$e\beta + f$ は大幅マイナスであり，「財政はよくない状態」にあります．

財政は現在「健全な状態」ではありません．それは，f が黒字でも，$e\beta$ がマイナス (赤字) だからです．実は，逆のほうが健全です．それは，f を黒字にしている雑収入は，本質的に不安定だからです．雑収入は，景気の動向に左右されます．景気が悪くなれば，減少するでしょう．リストラクチャリングをしなければ，SEA の運営は将来，行き詰まるかも知れません．そのリストラは，$e\beta$ をゼロにするためのものを最優先しなければなりません．もちろん今の財政は，リストラをしないからといって，明日行き詰まる訳ではありませんが....

少し悲観的な話になりますが，最近の業界の状況を反映して，「会員数が，ほんのわずかずつ減少し，併せて賛助会員費収入が半減，イベント収入がゼロになってゆくケース」も検討してみましょう．仮定として，現在の (実) 会員数を 900 とし，「4 年間にわたり，毎年，50 名ずつ減ってゆく」と，しましょう．昨年度の変動費を会員数 900 で割って，一人当たりの経費を求めると，10,436 円になります．これが，e を構成している c の値です．a は年会費ですから，7,000 円です．したがって，e は，

$$e = a - c = 7000 - 10436 = -3436$$

となります．仮定より，賛助会員費収入は 2,550,000 円，イベント収入はゼロ．それ以外の雑収入を 240 万円とすれば，雑収入 b は 4,950,000 円になります．また，α は－ 50，β は 900 ですから，積分定数を無視すれば，

$$z = -3436 \times (-50)t**2 / 2 + (-3436 \times 900 + (4950000 - 9092804))t$$
$$= 85900t**2 - 1050404t$$

が嵩>られます．これは，4 年間で，280 万円の赤字を意味します．この間の会員数の減少は，200 名です．z

の式からも明らかなように，SEA の場合，『会員数が減った方が，財政状態がよくなる』(?) 財政構造になっています．この簡単なシミュレーションから，我々の財政基盤がいかに脆弱なものか，ご理解いただけたと思います．

このような状態にある SEA の財政を立て直すにはどうしたらようのでしょうか？ それは，簡単 (?) です．$e\beta + f \geqq 0$，かつ $e \geqq 0$ が成立するようにすればよいのです．すなわち，

$$e\beta \geqq -f$$

とすればよいのですから，この両辺を β で割れば，

$$e \geqq -f / \beta$$

なる条件が得られます．ところで，$e = a - c$，$f = b - d$ ですから，

$$a - c \geqq (d - b) / \beta$$

が，その条件ということになります．つまり，『会費から会員 1 人当たりにかかる経費を差し引いたものがゼロ』で，かつ『会費から会員 1 人当たりの経費を差し引いたものが，事務局の諸経費などの固定費から雑収入を差し引いたものを会員数で割った金額より，大きければよい』のです．早い話が，『会費の値上げ！』です．

昨年度の会計報告の数字を前提にすれば，悲観的なシナリオの場合，$b - d$ は約 414 万円のマイナスですから，1 人当たり約 4,600 円の赤字です．また，$a - c = 0$ にするように，1 人当たりの経費に見合うだけの会費にすることが必要です．昨年度の実績を参考にすれば，c はほぼ 10,500 円であり，会費の 7,000 円を上回っています．つまり，イベント収入や賛助会費が減ってしまえば，単年度決算では，1 人当たり約 8,000 円の赤字が発生してしまいます．これが問題なのです．会費は，最低でも 3,500 円の値上げが必要です．さらに，財政基盤を改善するには，5,000 円から 8,000 円の値上げが，妥当な線でしょう．

ところで，『値上げ幅』は，『会員数がどうなるか』，『雑収入をどうするか』によって，答えが変わって来ます．会員数が多ければ，値上げ幅は，短期的には大きく (!) なります．逆に，会員数が少なければ，各会員の負担は軽くなります．イベント開催による収益などを増やすことなどによっても，値上げ幅を押さえることができます．ここで，前回議論した，SEA をどうするかの戦略が重要になります．つまり，『拡大·大衆化路線を採るか』『専門家集団として，少数精鋭主義を採るか』の選択です．これは，会員のみなさんの合意が必要なことです．つまり，拡大路線をとれば，会費は，採算ラインぎりぎりでも 15,000 円になるかも知れません．逆に，少数精鋭のダウンサイジング路線をとるならば，会員数を 700 人として，12,000 円ぐらいが妥当になるのではないでしょうか．

今回は，ここまでとします．読者諸兄のなかには，長年使っていなかった大脳のある部分を使って疲れてしまった人もいるかも知れません．しかし，このような算数は，組織を運営するための ABC(基礎) です．まあ，高校で学んだことが，役に立つこともあるという，まれな例です．ところで，使った式の中に誤っているものがあります．結論には，影響しません．その誤りを見つけた人にはコーヒーをごちそうします．

次回は，私のシナリオを提案したいと思います．乞う，ご期待！

ソフト屋さんたちは，いま....

筏井　美枝子
(SRA)

1. ハッカーは堕落だ!

たしかかあれは，ソフトウエア·シンポジウム'91 のプログラム委員 (PC) の勉強会 (?) だったから，いまから 2 年半以上も前のことである．

SEA にはずいぶん前から入会してはいたが，そのころの私は，まったくのペーパー会員であり，SEA の活動とは無縁に，社内の開発部門の近くで，技術支援みたいなことを仕事にしていた．具体的には，そのころ開発に使われ出した CASE ツールの導入サポートである．

そこへ，会社の上司でもある SEA 幹事の岸田さんから，名古屋でシンポジウムを開くことになったからプログラム委員を引き受けないかという声がかかり，「いや，なに，ちょっとしたローカル·アレンジメントをすればいいだけだし，名古屋に適当な (ヒマな?) 人がいないのだから，あなたが是非おやんなさい」みたいな，よくわけのわからない話ではあったのだが，そこはそれ，こちらも他人のことはとやかくいえないくらいのいい加減な性格なので，「まあいいか」くらいな軽いノリで引き受けたのだった．

いきさつはどうあれ，シンポジウムの論文集めを前に，PC の勉強会が名古屋で開かれた時の話である．

実行委員長かプログラム委員長か忘れたが，司会はたしか熊谷さんだったと思う．**「PC は，各自，自分が興味のある技術的テーマに対して，ポジション·ステートメントを示せ!」**みたいな課題があったように記憶しているのだが，何せ 2 年半前のことなので (場合によると，昨日のことも判然としないこともある私だから)，誰がどんな話をしたか，ほとんどおぼえていないのだが，その中で，某氏 (仮に Mr.X と呼ぶ) が，名古屋までの新幹線の中で作ったという手書きの OHP を使ってのプレゼン (あれは，Mr.X ふう現状認識とでもいうのかな?) が，「いいえて妙」で，長く印象に残った．

なかでも，**「····このころ Unix が生まれ，ハッカーと呼ばれる人たちが出てくる．かれらは，現状に満足しきったお坊っちゃまたちであり，過去のソフトウエア遺産を相続しながら，それを食いつぶすだけで，後に残すことを考えない．これは，当然やらねばならないことを否定しているのに等しく，まさしく堕落だ!」**というくだりは，オドロキ·モモの木·サンショの木だった．

どうして，オドロキだったかは後で述べることにして，これだけでは，どのような背景で，このようなくだりになったかよくおわかりいただけないと思うので，その時のメモを頼りに，当時の**「Mr.X ふう現状認識論？」**を筏井ふうに多少熱造して以下に再現してみた．

∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞

現在のソフトウェアの世界を見ると，まずは，**「ソフト屋さんたちの世界」**がある．これは，典型的な大衆社会であって，平凡な人が平凡であることを強烈に主張する．「平凡で何が悪い!」という開き直りが，立派に通用する世界である．一方で，**「ソフトウェア技術イコール近代科学技術」**と考えて (信じて) いる人たちの世界がある．

この 2 つの世界はそれぞれの方向に発展していく．つまり，一方は，**「単純な頭脳労働への道」**であり，「(テレビコマーシャルふうに) あなたにもできます」の世界である．もう一方は，**「専門化への道」**である．この方向は，自分の力に自信を持つ専門家大衆をたくさん生む．かれらは，ジェネラリストを軽蔑し (**「ふん，ディレッタントめ!」**)，専門以外のことにも大きな顔で口を出す (参加者は誰かを思い描いて爆笑)．

一方，歴史的に見ると，'50 ～'60 年代は，プログラミングにおける西部劇時代 (いわゆるコード·カウボーイの世界) で，私 (Mr.X 自身) が現役プログラマだったころの，古きよき時代である．

'70年代は，数学のアナロジーから構造化論が，製造工学のアナロジーからは，ウォータフォール型の機械論的パラダイムが提唱される．これらは，他の世界の理論をこの世界に単純にマッピングしたものに過ぎない．

'80年代に入ると，Unixが生まれ，ハッカーと呼ばれる人たちが登場する．かれらは，いわば現状に満足しきったお坊っちゃまたちであり，過去のソフトウエア遺産を相続しながら，それを食いつぶすだけで，後に残すことを考えない．これは，当然やらねばならないことを否定しているのに等しく，まさしく堕落だ!

そして,'90年代に入るわけだが，フォーマリズム，プロセス，オブジェクト指向などと，色とりどりのキーワードや流行語で世の中は賑わっているものの，いったい未来に希望はあるのか? 全体的な視野は開けるのだろうか? ところで，あなたは，この中のどの派?

∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞∞

と，まあ，こんな感じの論旨だったかと思う．このころは，まだバブルが充分膨らんでいたころなので，ダウンサイジングやリストラばやりの今日から見ると，なんとも，おおらかな雰囲気で，たった2年半前だというのに，妙に懐かしい．

話を戻して，前のくだりがオドロキだったのは，技術移転などというコトバが一部でたしかにささやかれてはいたが，それとまったく無縁と思える世界で，ハッカーといわれる人たちの，ずばぬけた頭のよさと革命的エネルギーが，かれらの自己中心性を差し引いても(あるいは，それすらも)礼賛されていたように，少なくとも私には感じられていたからである．

しかも，これまでハッカー礼賛者の一人であると思っていたMr.Xが，**「堕落だ!」**ときめつけたことは，私にとって，モモの木を通り越して，まさにサンショの木だったわけである．(最近は，ハッカーのイメージも大分変ってきて，**「コンピュータを駆使して悪さをする人」**という定義に近かったりするが...)

この時の**「ハッカーは堕落だ」**というコトバは，私に，ある方向の到来を確実に感じさせた．つまり，**「エントロピー減少へ」**，**「拡散から収縮へ」**，**「革命から進歩へ」**，**「混沌から秩序へ」**，**「自己満足から世界の幸福へ」**という流れである．その方向の行きつく先が何なのかはさておいて....

ここで，話はグイと横町に入っちゃうのあるが，**「堕落人ハッカー」**というイメージも，ちょっとすね者っぽくて，暗くって，ニヒルで... と自分にまったくないイメージだから，正直にいえば，ぐぐっと気を引かれてしまうのである．

かといって，ハッカーなどという，すぐれ者の堕落人に，私は到底なれないから，それならば，正当路線を地道でつつましく進んで，けなげな自分を振り返って，ほれぼれしてみたいと思うわけである．そして，開発の技術支援スタッフという位置付けは，地味でむくわれることも少なくて，まさにうってつけのところなのであった．

2. バブルがはじけちゃったのよね

2年半前には，「平凡で何が悪い!」という開き直りが立派に通用した**「ソフト屋さんたちの世界」**は，バブルが崩壊した後には，だんだん通用しなくなってきた．少なくとも，開き直っている限りは，仕事が来なくなった．そして，さらに悪いことには，**「ソフト屋さんたちの世界」**は**「ソフト屋さんたちの文化」**をしっかり形成しており，いまさら，**「専門家集団を目指せ!」**とか，**「リストラだ!」**とかけ声をかけても，おいそれとは変れないのである．

たとえば，数年前までは**「ソフトウエア開発には，バカもリコウも要らない．やれといわれたことをそれなりにできる人間がいればよい」**と平然といいきっていたマネージャも結構いた．そうしたマネージャたちは，技術力はあっても自己主張の強い人間をたたき，あるいは，追い出したので，その文化の中で長く生き残った者は，その文化をしっかり身にまとって離せない．**「危機感を持て! 自分で判断し，積極的にチャレンジせよ!」**と声を張り上げただけでは，急に意欲的になれる道理はない．

「ソフト屋さんたちの世界」が変化を余儀なくされた理由の1つに，発注者側(市場)の変化が上げられるだろう．これまで，EDP部門など，ソフトウェア開発の経験情報や必要な情報を準備でき，自らも開発経験と洗練された(これは，相手による)開発プロセスを持

ち，主導的に開発全体をマネージングできるところが直接の発注者だったのが，それらを一切持たない，エンドユーザからの直接依頼が多くなった．つまり，

- ノウハウその他の有益な情報をどこから入手するか?
- 開発プロセス全体をどうマネージングしていったらよいか?
- 下請け型開発文化から提案·主導型開発文化への意識およびプロセス変換は?

などといった問題にぶつかり出した．これは，実は潜在化していた問題があるきっかけにより浮かび上がって来たものだともいえる．さらに，問題はこれだけでにとどまらない．

- 技術者が日々の作業に追われて，新しい技術を学び/実践する機会が少ない．
- 技術支援の明確な窓口がない/見えない．
- プロジェクトがクローズされていて，技術を含むほとんどの情報は外部に伝達されない．
- 新しい技術を取り込む意欲/余裕が技術者に欠如してきている．
- 開発計画そのものがほとんど受身で，技術的な戦略が乏しい．

などなど，根本的な問題 (と私は思っている) もすでに山積みしている．

専門家としての分析能力と開発力を求められ出した「ソフト屋さんたち」は，バブル崩壊とともに「平凡で何が悪い!」を，あたふたと引っ込めてはみたものの，いったいどう変るべきか，どうしたら変れるのか，いま迷いに迷っているようだ．

3. 終わりに，なんだかいいわけ

凡人ソフト屋さんが奮起し，あるいは堕落人ハッカーが悔い改めたとして，どんな「ソフトウエア遺産」なるものを残せばよいか? 誰にどんな形で残して，どうやって相続するのか? このあたりまで，実は議論を持っていきたかったのだが....，どうやらこのあたりで根がつきてしまった．なんだか全体的に脈絡の無い雑文になってしまった感がある．

ここまで書いてきて読み返してみると，他人が昔しゃべったことを理不尽にもいまごろ書き起こし，それに尾ひれをつけてお茶を濁してしまったようで，一沫の後ろめたさも感じてしまったりする．

「ソフト屋さん」革命への，私の個人的な提案や試みについても，機会があれば，何かいってみたいと思っている．

ともあれ，何か書けたということへの自己満足だけで，この原稿を締めくくることにする (これもまた1つの堕落!?)．

中国四国地域ネットワークについて

西武 進
(システム･エンジニァリング･サービス)

今度，中野先生からのご推挙ということで，微力ながら，SEA の幹事をやらせていただくことになりました．

四国の松山で，通信関連のソフトハウスを営んでいます．その関係もあって，ネットワークを張ることに生き甲斐を感じている?! みたいに頑張っています．汎用機 (SNA) から OSI そして現在は主に TCP/IP 関連の GW ソフトの自社開発と受託開発を行っています．以上が簡単な私のバックグラウンドです．

現在，仕事上の関連性もあって，主に大学の先生方と中国･四国地域の地域ネットワークとして「中国･四国インターネット協議会」(略称： CSI) を今年の 4 月に設立しました．この協議会では他の地域ネットと同様のインターネットの普及促進をめざしています．具体的には，1993 年 7 月現在，参加サイト数は，ac 系が 12，その他が 5，合計 17 サイトです．

ここで，この CSI が抱えているさまざまな問題点を挙げてみましょう．当方のようなインターネット後進地域での普及の方策を考えるにあたっての，1 つの問題提起になれば幸いです．

1. 普及を妨げているバックグラウンド

中央と地方とのあいだの情報格差が叫ばれて久しいのですが，本当に情報を必要とする人たちが，その必要性を感じていないということが，ネットワークが普及しない理由だと思われます．そのいくつかを列挙してみましょう．

- **情報に対する評価**

  東京からの情報が，主流と考えている．
  ＝＞ 水平分散ではなく，まだまだツリー構造である．
  ＝＞ 役所が総じて中央省庁指向である．
  ＝＞ この種の企ては，中央がするものと決め付けている．

- **情報インフラの不足**

  情報機器の設置台数不足
  ＝＞ パソコンの導入でさえ不十分な環境である．
  回線費用が高い
  ＝＞ 海外との比較では無く，首都圏と地方との情報入手経費に格差．
  隣接サイトとの距離が遠い．

- **人的資源の不足**

  ＝＞ ポストマスターレベルの技術者が足りない (中央でも足りないのに....)．

- **資金不足**

  ＝＞ これはどこでも同じ．あるところにはあるのですが..
  ＝＞ スポンサーとなるべき企業が決定的に少ない

といったところで，ヒト･モノ･カネそして情報の，どの要素を取って見てもきびしい現実があります．

2. 普及への遠い道のり

では，このような情勢の中で，普及への方策があるのでしょうか?

少し時間を要しますが，堅実な方法は，次の通りです：

- **大学での普及と整備**

  大学等での，普及と整備が先決と思われます．が，ここでも，問題はインターネットの必要性を認めている学生等の就職先が東京･大阪などの大都市圏に限定されるという人的課題へ戻ります．

- **地域振興を訴える**

  地域ネットでは，大部分のネットワークが，地域振興を唱えています．

＝＞ はたしてどこまで，この旗印が有効か，見きわめられていません．

・ 楽しさの宣伝

インターネットを通して得られるさまざまな利益･楽しさを宣伝する．

＝＞ 個別「おたく」ではなく，少なくともネットワーク「おたく」に

＝＞ ネットワークに費用が必要なことの認識も同時に得られるか?!

3. 普及への遠いが意外に近い道のり

しかし，時代の変化が激しいこの時節にのんびりしたことはいっておられません．何らかの方策を考えねばなりませんが，いくつかの方策を上げたいと思います．

・ 自治体トップダウン方式

大分県の知事主導型と同様な各自治体の知事や市長への働きかけ．

＝＞ 知事の資質に大きく依存する．高知県の橋本知事は望みあり?

・ 自治体の広報活動を支援

原爆を体験した広島が平和へ向けた平和情報の世界への提供，広報活動を支援するインフラとして利用を促す．

＝＞ 自治体の方向性に大きく依存する．広島市は望みあり?

4. 組織の方向性

普及活動にも必然的に経費も必要であり，組織としての資金的な裏付けが必要です．そのためには，地域ネットを運営する組織をどのような形態にするかが，大切な課題です．

CSI での方向性は，以下の通りです：

(1) 非営利で，普及活動を行う
(2) 技術調整等を行う (ローカルな JPNIC または JEPG 的調整)
(3) 各自治体レベルでの，より小さい地域ネットの組織作りの先駆け

この中で，一番の問題点は (3) です．

より地域に密着した形での組織作りというと，結果的には自治体との共同作業になります．その１つの例が上記した大分県知事のバックアップによるパソコン通信「コアラ」です．最近ではパソコン通信というと割合と通りもよくなり，この手の話しは多くなっていますが，インターネットとなると和歌山県が積極的に動いているという程度だと思います．話として上がっているものはかなりあるとは思いますが，まだまだの状況です．

内容の是非はともかくとして，経済基盤の弱い地域での方向性としては，この自治体との連携というのが，もっとも可能性が高く，かつ，最も根気の必要な方法論だと思います．われわれのネットワークもこれから具体的な折衝を開始しようとしていますが，まだまだ手探りの状況です．

5. 最後に

インターネット後進地区といわれながらもなんとか地域ネットワークを旗揚げすることができましたが，ほんとうは，これからが勝負と思っています．さまざまな地区での各種の試みを参考に今後もインターネットの普及と発展に何らかの貢献をしていきたいと思います．みなさんのご指導･ご助力をよろしくお願いいたします．

E-mail: nishitak@ses.co.jp Fax:(0899)23-3709
NIFTY: NAB02662 MIX: ses CIS: 72011,2662
中国･四国インターネット協議会事務局
E-mail: sec@csi.ad.jp Fax:(0899)23-3709

プログラマについて考える
― ある技術交流会の報告 ―

伊藤 昌夫
(NHI エアロスペースシステムズ)

1. はじめに

これまで何度か，社内の仲間たちと，技術交流会という形で，ソフトウェア開発にまつわる問題について議論してきました．ここに報告するのは，昨年の10月9日に「プログラマについて考える」という題で，名古屋の熱田神宮文化会館で行った第6回目の会合の記録です．

実は，この会はまったくのボランティア活動で(そういう意味ではSEAと同じなのですが)，スピーカの方々(SRAの岸田さん，塩谷さん，YHPの土屋さん)にも，やはり完全にボランティア･ベースで参加していただきました(皆さんに交通費すらお渡ししていない)．仲間ともども深く感謝しています．

ところで，われわれの会には2つの特徴があります．1つは，常識的にすぐに答の出そうなタイトルをつけること．すぐにというのが，たいていは問題で，われわれは問題の本質を自分の力で見きわめようとせず，人のことばに頼ろうとする生き物ですから，そういうすぐに答の出そうなタイトルについて，ほんとうにそうかどうかを見直そうというわけです．

参加者には，かならずポジションペーパを書いてもらっているのですが，今回のタイトルについても，いつもと同じくさまざまな反応がありました．ふざけている人，むずしいと考え込む人，いろいろでした．しかし，もともと深遠な真理が生まれるようなテーマを考えているわけではなく，なるべく議論が百出し，それを通して何らかのヒントが得られればと，軽く考えています．そういう意味で，パネラの皆さんにはずいぶんご迷惑をかけたと思いますが，今回の討論を通して，プログラマとは一体どういう人間であり，われわれはどこを目指していけばよいのかが，多少なりとも明確になったのではないかと思います．

2番目の特徴は，明日から自分たちにできることを見つけようということです．これは前者と矛盾しているようですが，会での議論を机上の空論で終わらせたくない，自分の身になって考えようということですから，問題について真剣に考えるという点では同じであると思います．

今回の会合は次のようなスケジュールで行われました：

(1) テーマの説明
(2) スピーチ1 ― 岸田さん
(3) スピーチ2 ― 塩谷さん
(4) 昼食及び宝物殿見学
(5) スピーチ3 ― 土屋さん
(6) パネルディスカッション
- ･プログラムはアートかサイエンスか？
- ･職業プログラマとして何を学んでいくべきか？
- ･明日から何ができるか？

この報告では，以上のうち(2)と(6)をテープから採録しました．(3)の塩谷さんのお話は，これまでどういったプログラム作りをしてきたか，その中で自分がプログラムと一体となるという経験をしてきた．一体になった時，何か会議で変更要求があると，関連箇所のリストが頭の中でスクロールするかのようであったということと，NeXTを例にとって，どのような環境がわれわれに取って望ましいのかというお話でした．

土屋さんのお話は，プログラミングにおいては予測可能性(predictability)が非常に重要である，つまり，プログラマは常に先のことを予測しながら仕事をして行かなければならない．それができてこそプロだという主旨でした．

2. 岸田さんのスピーチ

岸田さんの話は，まずプログラマとしての御自身の経歴からはじまりました：

- ･翻訳のアルバイト
- ･フリーランスプログラマ
- ･組織の中のチーフプログラマ
- ･ソフトウェアハウスのマネージャ
- ･ボランティア(金子郁容さんのいうネットワーカ)

今回来ていただいたのも，まさにこのボランティア

としての立場でした.

次に, ライトモチーフとして「プログラマとは何者か?」について, いくつかの定義を挙げていただきましたが, その中で, 最も私自身が興味深かったのは, 「プログラマは本来何者でもないので, 何者にもなりうる」という指摘でした. プログラミングという仕事の真髄は, まさにこのあたりにあるのではないかという気がします.

そして, 翻訳者としての岸田さんが手がけられた「構造化プログラミング」(なんとマシン・ランゲージを用いた1960年代初めの構造化設計技法!?) の紹介がありました.

このあたりからのお話を, 以下にテープ録音を頼りに再現してみたいと思います.

2.1. プログラミングの芸術的側面について

.... といったことが1960年ごろに考えられていました. その考え方を発展させて行けば, データも同じように構造化しようということになり, 構造化設計から抽象データ型, さらには今日のオブジェクト指向につながってくるわけですが, そうした動きはすでに60年代に始まっているわけです.

こうした状況が, 私がプログラマの道を歩きだしたころの状況です. ただし, 私が翻訳したこのテキストブックは, 発注元のセミナ屋さんが途中で潰れてしまったので, 残念ながら日本では出版されませんでした.

さて, 私はこうして, なかば偶然のきっかけから, 翻訳者をへてプログラマになったのですが, それ以前にやっていた絵を描いたり, 詩を書いたりといった「芸術少年」としての活動がプログラミングとどんな関連があるのか, どう役に立ったかということを次にお話しします. 実は私は自分の芸術活動の遺産で, 最初の5年ぐらいのプログラマとしての生活を維持していたのです.

私の絵の先生 (といっても勝手に本を読んで勉強していただけのことですが) は, パウル・クレエという抽象画家です. スイス生まれのユダヤ人で, ドイツで活躍した人ですが, 第一時世界大戦後, ドイツにできたバウハウスという有名なデザイン学校で教えていました. その講義録 (もちろんドイツ語) が私の教科書でした. かれの有名なことばに,「グラッフィクアートの目的は, 目に見えるものを再現することではなく, 目に見えないものを見えるようにすることだ!」というのがあります.

さきほどお話したように, プログラマの仕事は, 対象とするシステムのモデルを作ることです. ところで, あらゆるシステムは, もともと客観的な存在ではなく, われわれがそのシステムの中にいくつかのコンポーネントを認識し, それらの要素間に何らかの関係を見いだしたとき, 初めて見えてくるような抽象的実在です. それはまさに, 目に見えないフォルムを目に見えるようにし, それらの間にある種の関係をつけて行くという抽象画家のやり方とまったく同じだと思われます.

クレエのもう1つの有名な考え方で,「分析」という概念が, 自然科学と芸術では大きく異なる, という指摘があります. これは, かれが1921年11月14日に, バウハウスでの連続講義の前置きのセッションで話したことです:

> — 化学分析について考えてみよう. たとえば, ある薬品がとても効き目がいい. よく売れている. そこで他のメーカが, その製品を自社の研究所で分析する. 秘密はなんと成分XとYをある比率で混合したことであるらしいことがわかる. あるいは, あるものを食べた人が死んでしまった. その食べ物を分析した結果, ある有害な成分が明らかになる. いずれの場合も, 初めに訳のわからないシステム全体があって, それをコンポーネントに分解して行くことによって, 謎が解かれる.
>
> では, そうした分析手法を芸術作品に対して適用するとどうなるだろうか?
>
> 1枚のきわめてすぐれた絵画がある. それをコンポーネントに分解し, 個々のフォルムや色彩, あるいはそれらの組合せを研究する. しかし, そうしたアプローチでは, せいぜい贋作しかできないであろう.
>
> 芸術における分析は, プロダクトの分析ではない. すぐれた作品の前に立って, 作者がいったいそれをどうやって創造したのかを考え, そのプロセスを想像して, 自分の足でたどってみることなのである. そうすることにより, 自分自身が未知の荒野に歩みだそうとしたときに, どうすればよいかがわかるのだ.

この, プロダクト分析とプロセス分析との違いは, プログラマにとっても重要な意味を持っていると思い

ます．なぜかといえば，すべてのプログラムは，何らかのプロセスを実現するための手段です．われわれがプログラムを作る時には，ユーザが求めているプロセスをどうやってモデル化するかを第一義に考えなければなりません．私が現役プログラマだった時代に，同僚の仕事ぶりを見ていて，自分の方が勝っていると感じたのは，この２つの分析方法論の違いについての認識の有無でした．そういう意味で，コンピュータ･プログラミングはアートによく似ていると思います．アートの勉強から得られることが多い．だから，Fortran を勉強するるより，絵の勉強をした方がよいというのが私のオススメです (笑い)．

2.2. プログラマの誕生と成長について

さて，以上お話したのは，人はいかにしてプログラマになるか? についての，私自身の特殊ケースです．

一般的にいってわれわれは，どうやってプログラマになるのか? 別にプログラマでなくても，何でもいい．人はいかにして何者かになるか？

これについては，数多くの思想家がいろいろなことをいっていますが，私が一番気に入っているのは，スペインの哲学者ホセ･オルテガ･イ･ガセットのモデルです．かれの最後の著書「個人と社会」(これは日本語にも翻訳されています) のなかで，オルテガは，個人と社会の関係について，次のように述べています．

「個人 vs 社会」というふうに問題提起した場合，ふつう，われわれはまず，個人について考え，次に社会を考えます．そして両者の間にある種の葛藤が存在するとうイメージを抱きます．そうした考え方が実は間違いだというのが，オルテガの逆説的な主張です．

人間は，オギャーと生まれた瞬間にプログラマになるわけではない．いや，それ以前に，われわれはオギャーと生まれた瞬間にはまだ人間ですらない (!)．たとえば，生まれての赤ん坊が森の中に置き去りにされる場合が，たまたま起こったとします．その赤ん坊は森の中で狼に育てられ，狼少年に成長する．それをインディ･ジョーンズのような探険家が発見し，かわいそうだと連れ帰って，ニューヨークあたりのアパートで一生懸命に教育しても，狼少年は絶対ふつうの人間には戻らない．

つまり，人間は人間として生まれてくるのではなく，人間に育てられ，人間の社会の中で成長することで，人間になるのだというのが，オルテガの主張なのです．私という存在は，生まれた瞬間にはまだ存在しない．ただ，周囲に見知らぬ人間たちがいて，かれらには顔も名前もない．つまり，生まれたばかりの人間の自己認識は，「私は，この回りにいる何だか訳のわからない生き物 (People) と同じ存在らしい」という程度でしかありません．

やがて，時間の経過とともに，回りにいる人びとのうち何人かについて，顔や名前がわかってくる．両親，兄弟，友人，上司，その他，名前と顔のついた人間たちが登場する．つまり，People の中の何人かが You というかたちで認識される．この時点になっても，しかし，まだ私自身は存在せず，あなたたちと同じだという程度の自己認識しかありません．

ところがある日突然に，まるで禅の悟りのような稲妻が閃く．なかには，一生閃かないで死んでしまう人もいるようですが (笑い)，ある日突然に「私はあなた (たち) とは違う」ということに気づく．そのとき，これまでは自分の外部ばかり意識していたのだけれど，そうして閃いた瞬間に周囲との交渉をいっさい絶って，自分の中に沈潜するという行動をとる．この自己沈潜の能力こそが，人間を他の動物と区別する特徴だとオルテガはいっています．仕事をほうり投げてぼーっと宙の一点を見つめていたり，トイレに閉じ篭ったり，喫茶店でお茶を飲んでいたりとか，物理的にはいろいろなスタイルが考えられますが，とにかく論理的に外部との交渉を絶って自分の心のなかに沈潜する．

とにかく，自分が他の人間たちと違う存在だということはわかった．しかし，ロビンソン･クルーソーではあるまいし，これから先，人間としては，自分１人だけで生きて行くわけにはいかないので，どんなふうにこれから周囲とつきあって行くか，そのための具体的戦略を思案する．そして，ある戦略を手にして，また現実世界に戻って周囲の人間たちとの交渉を始めるのです．そうした自己沈潜の繰り返しを通じて人間は人間として成長して行くのだというのが，オルテガの個人-社会モデルです．

われわれはプログラマとして生まれてくるわけではなく，プログラマになるのです．ということは，オルテガのモデルで示されたこの成長プロセスを通るのです．たまたま大学でコンピュータ･サイエンスを勉強したためにプログラマになってしまった (笑い) とか，銀行に入ったつもりがコンピュータ部門に配属されてしまったとか，いろいろなケースがあるでしょうが，とにかく，赤ん坊が生まれるのと同じような意味

合いで，最初は，プログラマとしての自分はまだ確立されておらず，回りに無名のプログラマたちが大勢いるという状況から出発するわけです．

そのうちプロジェクトとかいう場所に配属されると，そこのメンバたちとかリーダとかとの関係がはじまる．つまり，無名の Programming People のうちの何人かが You-Programmer として出現するわけです．そうして仕事をしているうちに，突然「俺はちょっと違う!」と閃く瞬間がやってくる．

そのときどうするか? それは，人によって異なるでしょう．会社を辞めることもあるかも知れないし，同じ組織の中で違った自分を実現するという方策を講じることもあるでしょう．そういう場面を何回も経過しながら，プログラマとしての自分を成長させて行く．そのうちにいろいろなことがわかってくる....

で，結果として確立されてきた自己をどう表現し，実現するか? これもまた，社会との関わりの中で行われる営みですから，周囲からはただ，特定のプロジェクトの中での動きとか，実際にかれが作ったプログラム，あるいはドキュメントや論文，などを通じてしか，理解する方法はありません．

一般論として，真のプログラマのあり方を論ずることは，だから，きわめてむずかしいわけですが，たとえば，ソフトウェアの世界で流行しているいくつかのキーワードや概念について，かれがどう考えているかを見れば，間接的に推論することはできるでしょう．

2.3. 独断的 Unix 論

たとえば，Unix というキーワードがあります．これは今後もしばらくのあいだ，われわれにとって重要な意味を持ち続けると思います．たまたま先月，SEA 関西で，「Unix のあとに来るもの?」というフォーラムがありました．私もそこで，Unix と自分との関係について，以下のようなことをお話ししたのですが，1 つのケース·スタディとして，聞いてください．

Unix は複合的な性格を持っています．いまの時代の一種のファッションあるいはトレンドとしての Unix，プロダクトとしての Unix，技術としての Unix，プログラミング文化としての Unix，この 4 つの主な顔があるように思われます．もちろん，これは私というプログラマ (正確にいえば，元プログラマ) の見方で，みなさんは，あるいは別の意見をお持ちかも知れません．Unix ということばを媒体として，1 プログラマとしての私がどのようにものを考えているかを，参考までにお聞きください．

流行としての Unix ブームは，もちろん，みなさんご存知のように，オープン·システムとか，ダウンサイジングとかいった現象につながって行きます．そうした時代の波に巻き込まれることなく，その背景に何があるかを考えることが大切だと思います．私見では，ソフトウェアとハードウェアとの主導権争いがこれまでずっと続いていて，その一環として Unix を捉えるのが正しいと考えています．

私が現役のプログラマだった時代は，ハードウェアがあまりにもチャチで，ソフトウェア主導型の時代でした．次に，1970 年代に入ると，第 3 世代メインフレームの全盛期がやって来ました．それは，完全にハードウェア主導の時代でした．そして，どうもそうした傾向は気に入らないと考えたプログラマたちが起こしたのが，Unix 革命だったのです．

現在の状況は，ふたたび WS や PC などのハードウェア技術が優勢で，それを何とかせねばという次のソフトウェア革命の準備期ではないかと思います．流行語に象徴される時代の流れを，1 プログラマとしての私は，このように捉えています．

プロダクトとしての Unix とは何かと考えると，やはり，オープンなソフトウェアというのが，最大の特徴でしょう．

システムとしてのソフトウェアは，他のあらゆるシステムと同じく，時間の経過とともに進化して行く運命にあります．システムの一番自然な進化は，ユーザがシステムと一緒に進化することです．ユーザが一歩進めば，システムもそれに併せて形を変え，システムがちょっと先に進めば，ユーザの立居振舞いも，それに合わせて変わって行くというのが最も自然です．そういう観点から見ると，Unix が作られ，発展してきたこれまでの歴史は，非常に素直なシステムの進化のプロセスをたどっているように思われます．

Unix は，もともと，ベル研にいた一握りの研究者たちの手で，自分たちのためのシステムとして誕生した．その後，バークレーの学生ハッカー·グループを中心として，それを囲むユーザたちの手で育てられてきた．今日 AT&T が巨額なライセンス料収入を儲けているというのは，そうしたシステムの自然な進化の道から，少しはずれているように思われます．もうちょっと自然な姿に戻そうと，R. ストールマンたちがやっている GNU の運動は，だから，多くのプログラマたちに好感をもって迎えられている．

次に，テクノロジとしてのUnixを考えてみましょう．それは，われわれ自身がコンピュータの上で利用するソフトウェア·ツールの集まり，あるいはそうしたツール群を載せる開発環境の基盤としてのUnixというイメージです．この研究会の前回のテーマは「開発環境」だったそうですが，開発環境というコンピュータ·アプリケーションをシステマテックに考える，つまり1つの抽象芸術作品のようなシステムとしての開発環境を考えるという動きのはしりが，Unixであったと思います．そして，次に来るものは，いま私たちが普及活動を始めたPCTEのような次世代インフラ技術だろうと予想されます．

最後に文化としてのUnixですが，冗談めかしていえば，「ハッカの天国，マネージャの地獄，ソフトウェア·エンジニアの頭痛の種」ということになる思います(笑い)．

インタネットワーク上のニュースグループなどをのぞいていると，まさに気狂いじみたものが多いのですが，しかし，それなりに面白い．今日ハードコピーでお配りした英文の資料は，そうしたネットワーク上を流れていた"Real Programmer Joke"の集大成版です．このジョークのスタイルを借りていえば，「ほんとうのプログラマはプログラムを書かない!?」というのが，Unixの世界のパラダイムでしょう．

これまでCobolやFortranの世界で生きてきた人たちがUnixの世界に入って最初にとまどうのは，ベテランのCプログラマたちがほとんど自らの手でCのコーディングをしないことです．御存じのように，Unix環境のツール·ボックスには，数百種類のコマンドが用意されており，それらを組み合わせれば，たいていのプログラム仕様は簡単にプロトタイピングできます．そして，もしパフォーマンスその他の理由から，どうしてもC言語でコーディングする必要が生じた場合には，必要な機能を持つコマンドのソース·コードを検索し，適当に切り貼りし，手を加えればよいわけです．

それは，よくいえば既存の知識の再利用であり，悪くいえば著作権無視の盗作です．別のいい方をすれば，これまでの製造(Manufacturing)パラダイムによるソフトウェア開発から，商業取引または情報交換(Trading)パラダイムへの転換を意味しているとも考えられるでしょう．

これまでの開発環境のアーキテクチャは，ハードウェアの製造パラダイム，つまり工業生産や，農業生産のモデルにもとづいて考えられてきましたが，商業取引のパラダイムをベースにものを考えると，全然別な形のものができ上がるのではないかと，私は考えています．

ソフトウェア開発環境は，一種の社会システムみたいなものですが，ふつうにわれわれが考える社会や組織のモデルは，基本的に農耕文明およびその発展型としての工業文明を支えている管理型の都市社会のモデルです．しかし，人類の歴史を振り返ってみると，そうした管理型都市モデルだけが，唯一の社会モデルではない．たとえば，マホメッドが活躍した中世アラブの都市(メッカやメディナ)は，まったく別のパラダイムにもとづいてプランニングされ建設されている．それは，当時のアラビアの文明が，ヨーロッパやアジアのそれとは違った商業文明だったということ，それから唯一神アッラーの前には万人が平等であるというイスラム教のフラットな観念的社会モデル(これはわれわれが見なれているピラミッド型の階層的管理組織のモデルとは根本的に異なる)と関連があるのではないかと思われます．

とまあ，偉そうな歴史的展望をお話ししましたが，実はこれは他人のアイデアの受け売りでして，数年前に文部省の補助金を使って，数多くの歴史学や社会学の先生たちが行なった「イスラームの都市性」に関する協同研究の成果の一端を御紹介しただけのことです．興味を持たれた方は，たとえば後藤明「メッカ」(中公新書)や，事典「イスラームの都市性」(亜紀書房)をお読みになってください．開発環境の人間的側面における諸問題を考えるには，そうした社会的な視点がどうしても必要になると私は思います．

以上，流行·製品·技術·文化という4つの観点から，Unixについての勝手な私見をお話ししました．おそらく，世の中で語られているUnix論とはからに違ったニュアンスを含んでいると思います．だとすれば，それは，プログラマとしての私が，名もなく顔もないただのプログラマとは異なる自分を確立しようとして，それなりの努力をしてきた証拠だと思うのですが，どうでしょうか?

2.4. プログラミングの科学について

ここで，少し観点を変えて，科学としてのプログラミングについて，少しお話ししましょう．

プログラミングはアートであると同時に，サイエンスとしての側面も持っています．ただし，どういう科学かということが大きな問題でしょう．この点につい

ては，10年ほどまえに亡くなったアメリカの哲学者エリック·ホッファ (子どものときに失明，18歳くらいになって視力が回復してから独学で勉強し，サンフランシスコで沖仲仕をしながら，バークレーで哲学を教えていたという変な人．わたしはそういう変な人が好きです) が，その遺著 'Between Devil and Dragon' で述べていることが，たいへん参考になると思います．

かれがいう Dragon は，中国の龍とは違って，西洋風のドラゴン，つまり自然界におけるすべての悪の集合を指します．それをどうやって退治するかが，自然科学のモチベーションだというわけです．自然を神がわれわれに与えたメッセージだと考え，その暗号を解読しようというのが，自然科学の目的です．一方，Devil(悪魔) とは何か? それは，われわれ人間の心の中にあるすべての悪を集めたものであり，非自然的な悪の象徴である．それをどうやって退治するかを目的として生まれたのが社会科学だというのが，ホッファの基本的な主張です．

人間がいかに非自然的な存在であるか．ホッファは，そのことをいろいろな例をあげて実証しています．1970年代のいわゆるロック世代，フラワー·チルドレンの全盛期に，ホッファはそのアンチテーゼとして論陣を張り，「自然に帰ろう!」というのを時代のキャッチフレーズに対抗して，「冗談じゃない，人間にとっては都会の環境が一番自然なのだ」という逆説的な発言をしたりしていますが，その論旨は，ノーベル賞を貰ったハーバート·サイモンの「人工の科学」と一脈相通じるニュアンスを含んでいるように思われます．

プログラミングに関する科学といえば，すぐに，応用数学の1分野としてのコンピュータ·サイエンスが思い浮かべられますが，しかし，これまで20年以上にわたるソフトウェア工学の研究が明らかにしてきたように，プログラミングに関わる諸問題には，人間に絡む要素が非常に多い．最近のホット·トピックの1つであるソフトウェア·プロセスの問題にしても，連立方程式を解くとか，化学反応を制御するとかいった非人間的なプロセスを相手にするのではなくて，人間的要因を中心的コンポーネントとして含むようなプロセスをどうモデル化し，起動し，マネジメントするかが，最大の課題になっています．そうした意味からすれば，われわれプログラマは，数学や自然科学よりも，むしろ，社会学や心理学をもっと勉強すべきではないでしょうか?

問題提起としてはこんなところで私の話は終わりにして，あとは午後のパネル討論にゆずりたいと思います．

3. パネル討論

今回の集まりの最後は，岸田·塩谷·土屋という3人のゲスト·スピーカに，DEC名古屋の松尾さんを加えたパネル討論でした．司会は私 (伊藤)．以下に，その内容をなるべく忠実に記録してみましょう．

3.1. プログラムはアートか，サイエンスか?

司会： まず，プログラミングは芸術か科学かについて，一言ずつ御意見をお願いします．

土屋： むずかしい注文ですね．ちなみに私の祖父はプロの画家でした．そして，私自身は学生時代，芸術と科学のはざまにある哲学をやっていました．あとの議論に関係すると思いますが，あるプログラマの仕事上の成果について，それが芸術作品か，それとも科学的活動の結果かを評価するさい，評価者自身もプログラマである (いやそうでなければ評価できない) というケースが多いと思います．

派手なGUIのプログラムなら，それが芸術か否かをだれでも判定できるかも知れないが，実行結果が目に見えないもっと地味なプログラムの場合はどうか? それでも，これはアートだと考えているプログラマもいるわけですが，もし，プログラミングがアートなら，そうした特定のプログラマだけに通じる世界は打破されなければならない．芸術は，より強い力につき動かされて行われる活動なのですから....

プログラミングの場合にも，同じような創造のエネルギーがあるのかも知れませんが，しかし，非常に制約が多い．たとえば，C++ で書かれたのプログラムをCobol プログラマが見たら，たとえどんなに芸術的なコーディングであったとしても，絶対に理解できないでしょう．

つまり，私は，プログラミングはアートではないという立場なのですが，いまいったことは，結果として生じることで，一義的ではない．作品としてのプログラムはあくまで結果である．芸術的なプログラムを作るのが目的ではなく，結果としてスマートなプログラムができる．そのスマートさの中に，何らかの芸術性が見出されるということはあるかも知れませんが....

岸田： 近代芸術の主流は自己表現，つまり芸術作品は作者自身の精神的な分身であるという考え方です．プログラミングがもし，そういった意味でアートであ

るとすると，困ったことが起こる．G. ワインバーグが 'Psychology of Computer Programming' という本の中で指摘していることですが，プログラムにもしバグがあると，それはプログラマの精神に何らかの欠陥があるという意味になってしまう (笑い).

しかし，認知的不協和という心理学上の原則からいって，人間は自分自身に欠陥があるという認識の上では，生きて行くことができない存在である．そうすると，何が起こるかといえば，プログラマがバグの存在を認めたがらない．でも，いくらかれが認めたがらないからといって，バグはやはり存在するわけですから，今度は，その原因を自分以外のものに押しつけようとする．「今日はマシンの調子が悪い」とか，「だれか他の人が作ったモジュールのせいだ」とかいいはじめるわけです．

ワインバーグは，こうした弊害を避けるために Egoless Programming というアプローチを提唱しています．一言でいえば「バグを憎んで人を憎まず!」という態度です．その意味で，プログラミングは近代芸術とは異なる．

プログラミングがサイエンスかどうかについての私の考えは，午前中にお話しした通りです．プログラミングは，自然科学というよりもむしろ社会科学の範疇に属している．われわれのほとんどの仕事は，人間的要素を含んだ社会ロセスのモデルを作ることだと思います．

塩谷： アートとしてのプログラムは単なる趣味でしかない．それは個人が自分のために作るホビーのプログラムです．プロとしてのプログラムはアートではなく，サイエンスの領域に属する．そうあるべきだと思います．

プログラマが自分の書いたプログラムに対してある種の美意識を持つ．その美意識にしたがって，きれいにプログマム書く．それはいいことです．そして，たしかにアートに似ているかも知れない．しかし，土屋さんのお話にもありましたが，チームワークの中で自分が１つのモジュールを担当している場合には，そうした個人の美意識を制限して，読みやすさとか，他のモジュールとの連結の容易さとかを考えたり，あるいはチームの作業スケジュールを配慮して，より早く仕上げるとか，プレディクティブに (予見性を高く) 作るといった工学的･科学的なアプローチのほうが，より重要です．

司会： 土屋さんがおっしゃった予見性とは具体的にはどのようなことでしょうか?

土屋： 科学工学的な意味で予見性を考えるというのは，閉じられた環境の中で，条件を同じにすれば，同じ結果が出るようにしたい．つまり，完全に制御された実験と同じように，再現性を高めて行くと努力することです．たとえば，個人差とかいったバラツキをなくして平均的にする．そうすると，乱暴にいえば，5人で10ヶ月かかるなら，10人でやれば5ヶ月でできるというような予測が成り立つ．仕事を非個性化し再現性を高めることによって，予見性が出てくる．

松尾： アートかサイエンスかということですが，アートというのはよくわかりません．むしろ，クラフト (職人芸) かサイエンスかという対比のほうがピッタリくる感じです．

いずれにせよ，この問題を考えるとき，成果物 (作品) について判断する場合と，開発の手法あるいはその背後にある考え方 (哲学) を論じる場合とでは，判断の基準が異なるように思います．

たとえば，作品としてのプログラムを考えた場合，きれいにネスティングされていて，わかりやすいコメントがついている．そうしたリストを見ると，あまりの美しさにある種のショックを受けたりするのですが，実際にランさせるとコンパイルエラーで落ちてしまう (笑い)．それがはたしてアートか?

また，ロジック (アルゴリズム) の美しさがアートか否かという議論もあります．さらに GUI．たしかに NeXT の GUI ビルダはアートと呼んでもいいできばえだと思いますが，それを使って作られるすべての GUI がアートであるかどうかはわからない．

手法や哲学はアートかサイエンスといえば，いろいろ議論はあると思いますが，かつては物理学もアートみたいな代物だった．いまわれわれがサイエンスだと思っているものも，その出所をだとると，実は昔はアートみたいなところから出発している．たとえば，AI．チェスのプログラムは，かつては AI の重要な研究テーマだったのですが，いまのファミコンで動くチェスのソフトはだれも AI とは呼ばない．そういう意味で，定義や分類は時代と共に変わって行くのではないでしょうか?

プログラミングという仕事は，コストという要因を考えないでよいのならば，アート的な性格を大いに持っていると思います．芸術作品には完成はない．作者本人が納得するまで制作活動を続けるというのが

アートでしょう．プログラミングにおいて，仮にバグがゼロになるを完成だと定義すると，みなさん御存じのように，なかなかそこには行きつけない (笑い).

しかし，われわれ職業プログラマの仕事上のポイントは，コストと品質のバランスをとることにあり，あるところで納得せざるをえない．だとすると，僕自身がこれまで10数年間やって来たことは，サイエンスとか工学とかいう分野に属すると思います．

司会： ありがとうございました．一言で言ってしまうと，パネラの大方の意見は，プログラムはアートではないということにつきるように思います．一方，フロアの参加者の意見は，ポジション·ペーパから見るかぎりでは，アート派のほう方が多かったと思いますが，何か意見をお持ちの方はいらっしゃいませんか?

フロア： 自分の書いたプログラムを１つの作品だと考えている人は，実際たくさんいるのではないでしょうか?

塩谷： ゲーム·プログラムのように，個人が自分の好みを強く出して作りたいものを作るという場合には，それを自分の作品として見てほしいとか，動作や画面をアートとして評価してもらいたいという場合もあるでしょう．しかし，会社の仕事として，あらかじめ決められた仕様にしたがってプログラムを作る場合は，アートというより，やはりサイエンスでしょう?

土屋： 逆の考え方もあるのではないかしら? つまり，自分のプログラムをアートにしたいと思っている．岸田さんなんかは，そうではないですか?(笑い)

岸田： 自分が作りたくて作るのは，あくまでアマチュアのプログラムです．プロのプログラムは注文にもとづいて作られる．

その時，半人前のプログラマは野路通り注文に合わせて仕事をする．私は，それは間違っている (プロとして未熟だ) と思います．いま与えられた注文 (仕様) は単なる１つの例題であって，真のプロなら，次に別のお客さんから類似の注文が来た時のことを考えなければいけない．いま手元に与えられた給与計算またはゲームソフトの表面的な仕様の裏にあるものを考えて，次の仕事に応用できるようにする．それは，設計技法かも知れないし，コーディング·スタイルかも知れない，そんなふうにものを考えるというのは，芸術という意味でのアートではなくて，松尾さんがいわれたように職人芸 (クラフト) に近い．サイエンスとはちょっと違う．

松尾： ホビーには，アート的な側面はあるのでしょうか?

岸田： 絵の世界のアナロジーでいうと，広告宣伝の絵を描く人が職業プログラマで，いわゆる絵描きさんはアマチュアのプログラマに近い．

塩谷： 非常に単純に考えて，役に立つか立たないかで判別したらどうでしょう? アートは，役に立たない (笑い)．しかし，そこには自由がある．われわれの仕事は，できれば世の中の役に立っていると思いたいので，そういう意味で，アートではない (笑い).

土屋： プログラマに対する評価基準は，時代とともに変わってきているように思います．特に，最近では，オブジェクト指向の影響が大きい．HP の場合には，プログラマという職種はなくて，ワーカと呼ばれています．その評価は，これまでは，どれだけのステップ数が組めるかということがポイントでした．ある程度の量のコーディング能力を前提として，その後で，仕事の信頼性とかが云々されていた．

しかし，最近では，できるだけ書かない人を評価するようになってきた．さきほど岸田さんがいわれたように，いまの仕事だけではなくて，再利用を考えている人を評価する．

司会： それはどのように評価されるのでしょうか?

土屋： コンテストをやるのです．ある仕様を与えて，再利用による開発のコンテストをする．また，再利用のための開発という観点から見るものもあります．

その一方で，あくまで自分の手でコーディングしたいという欲求が，あちこちに根強く残っている．他人の作ったプログラムは信用できない．しかし，コーディングしなければバグはでないのです (笑い)．仕事は，すればするほど品質は下がる．アートもクラフトもサイエンスも，何かを作りたいという衝動があるという点では同じことでしょう．何らかの理由を作って，人に見てもらいたいという欲求になる．人間というのは，実に悲しい動物ですね (笑い).

松尾： DEC の話を御紹介しましょう．ダウンサイジングの嵐の中で，IBM 帝国同様に DEC 帝国も没落しかかっていますが，最近またアルファとかを出して，帝国の逆襲を狙っています (笑い).

これまでの DEC は，技術者が作りたいものを作って成功してきた会社です．PDP も VAX もそうでした．DEC で自分の思い通りのものを作れなかった人

たちは，DECを出て，新しい会社を始めました．データ·ジェネラルがそうです．最近ではWindows NTのアーキテクチャもそうです．会社が小さい間は，みんなが好きなものを作り，作れない時はスピンアウトしてきたわけです．トップが交替して，それが変わりました．いろいろな部署で同じようなものを作っていたのですが，それをやめて合理化を心がけています．

そういう意味からすると，作りたいものを作っていた間はプロではなかったのかな，と思ったりするのです．アートはほんとうに金にならないのかな，と思います．NeXTみたいな製品は，サイエンス路線からは出てこないのではないのでしょうか?

塩谷： それはこだわりがきっかけでしょう．現実のNeXTは既存の新技術をアセンブルしただけのものだと思います．ハードウェア自体も，またソフトウェアも....

3.2. 学習と適性をめぐって

司会： では，この辺で話題を変えて，専門プログラマとしては何を学ぶべきかについて，また，パネリストの方々の御意見をうかがいましょう．今度は，松尾さんから....

松尾： 何を? といわれても，あまりにも対象が多すぎて簡単には答えられません．逆に，どういう姿勢で学んで行くべきかについてお話したいと思います．

ソフトウェア技術者というのは，世界でもっとも保守的な人間ではないでしょうか? 新しいOSは使わない，新しい言語は使わない，第1バージョンのソフトは使わない，自分の覚えた手法以外は使わない，自分が作ったサブルーチン以外は使わない等々，きわめて保守的です．新しいツールを与えても使おうとしません．パソコンでプログラムおぼえた人は，そのエディタをいまだに使い続け，結果をわざわざファイル転送してUnixでコンパイルしたりしている(笑い)．

それは，よい悪いの問題ではなく，過去に何度か痛い目にあっているからです．では，保守的なままでよいか? たしかに，そのほうが安全確実で，仕事を少ない工数で仕上げられるかもしれません．しかし，そのままでは，ただ自分に知識や経験の範囲内で，ちまちまと効率化しているだけにすぎません．この技術進歩の激しい時代では，新しい試みによって，画期的に生産性を上げることが可能なのです．

ソフトウェア技術者にとって重要なのは，そうした職業病的な保守性を打ち破り，自分が知らないこと，新しいことを積極的に取り入れていこうとする態度だと思います．

司会： だれかのポジション·ステートメントにあったのですが，生まれつきプログラマに向いている人と，向いていない人がいる．適性がない人がいくら学んでもムダではないかという意見については，どう思われますか?

松尾： ソフトウェア開発の仕事全般に関しては，別に向き不向きはないと思います．ただし，工程によってはあるでしょう．一般に，ソフトウェア開発工程の上流に行けば行くほど，技術が高級になり，それをやる人間は偉いという迷信があるようですが(笑い)，これはおかしい．もし，人によって工程ごとに適性があるとすれば，ある工程の仕事に関して，道を究めていくという生き方もあるのではないでしょうか?

塩谷： 最初の問いに対する答えは，なんでも勉強しなさいということですね．

ソフトウェア技術者の適性に関しては，私は，向き不向きが存在すると思います．論理的な文章が書けない人は，プログラマになってはいけないとはいいませんが，もしなったとしても，よいプログラムは書けないでしょう．オブジェクト指向プログラミングの場合はどうかわかりませんが，ふつうの手続き型プログラミングや，Prologのような論理型プログラミングの仕事は，そういう(非論理的な)人には向いていないと思います．

もう一言．さきほど何でもといいましたが，あえて挙げれば，プログラミング専門書を読んだり，勉強したりするのは止めて，プログラミングに関係ない書物をもっと読んだほうがよいということです．それはなぜかといえば，土屋さんのお話にもありましたが，プログラマに対して仕様を出す分析者とか，分析者に対して要求を出すお客さんがいて，われわれはそのお客さんの要求を最終的にプログラミングしているわけです．そのとき，いわば一種の通訳としてのプログラマにとって，プログラミングおよびコンピュータに関する知識だけでは不十分で，アプリケーションのドメインに関する知識がどうしても必要になってくるからです．

私がいつも思っているのは，アメリカでは，学校でコンピュータ．サイエンスを勉強した学生だけを雇ってソフトウェアを開発していますが，それはまちがいだ．経営学とか哲学とかいろいろな分野の勉強ををしてきた人たちがプログラミングにたずさわったほう

が，それぞれのドメインについての知識が得られてよい．その意味では，日本のやり方のほうがよいと思います．

司会： プログラマに向いている人の比率はどのくらいでしょう？ Knuth 先生は，たしか 4% だといっていますが....

塩谷： 全体から見れば，そのくらいの割合かもしれません．

岸田： 職業プログラマとしては，当然，ソフトウェア·エンジニアリングは学ばなければならないでしょうね．

技術以外の分野では，もし会社の中で出世したり成功したいと思えば，「論語」が推薦図書です．

> 名正しからざれば，言順 (したが) わず．
> 言順わざれば，事成らず．
> 事成らざらば，礼楽興らず．

これは，論語·子路篇に出ている有名なことばですが，要するに，基本的なコンセプトにもとづいて開発プロセスのモデル (名) を確立し，開発方法論 (言) を整備しなければ，プロジェクト (事) はうまく行かず，支援ツールや技法 (礼楽) もうまく育たない，というように読み替えることができる．つまり，ソフトウェア·エンジニアリングの精神にもとづくテクニカル·マネジメントのあり方を説いているわけです (笑い)．

最近流行のオブジェクト指向についても，その中心的な原理の１つであるクラス·ハイアラーキーつまり概念の階層的分類については，やはり孔子の後継者で春秋戦国時代の思想家·旬子が，その正名論において，詳細な説明を展開しています．つまり，ソフトウェア·エンジニアリングの枠組みは，2000 年以上前にほぼ固まっているわけで (笑い)，そうした古代東洋の知恵を身につければ，少なくとも，テクニカル·マネージャとしての成功は保証されるでしょう．

それから，さきほど塩谷さんがいわれたように，他者との円滑なコミュニケーションが，プログラマにとってきわめて重要ですが，この分野では，応用言語学の１分野である一般意味論の勉強が役に立ちます．たとえば，S.I. ハヤカワの「思想と行動における言語」(邦訳は岩波書店刊) は，わかりやすいテキストブックとして定評があります．表面的なシンタクス上の意味と，深層的な真の意味との違いを理解することは，ソフトウェア技術者どうし，あるいはユーザとの対話を進める上で，大いに役立つでしょう．

司会： 私は，あまり出世したくないのですが，そういう人間は何を勉強したらよいのでしょうか?

岸田： 仏教などいかがですか? 一切空とかいっています (笑い)．

土屋： 岸田さんの意見にほぼ全面的に賛成です．私自身，現役プログラマ時代は，あまり他人と口をききたくなかった．しかし，プロとしては，好きでないこともやらなければならない．えてして，機械相手の仕事をする人間は孤独になりやすい．危機的な状況に陥っていたとしても，そのことがわからないままどんどん進んでしまう．他人と相談するよりは自分でこつこつやりたいとガンバルのだが，そのうちに，自分の心のなかで無限ループしてしまう．そんな時，隣にも悩んでいる人がいるんだとわかれば，かなり助けになるでしょう．それが，私自身学びたいと考えていることです．ともすれば閉じてしまいがつな職場なので，空間的にも時間的にも，発想を変えるよう心がけなければならないと思います．

司会： 適性·不適性の問題については，お２人はどうお考えですか?

岸田： 野球選手の場合と，足の早い人と遅い人を比べると早い方が向いている．バスケットの場合だと，背が高いほうがよい．そういう意味いえばプログラマの場合には，たとえば記憶力のいい人の方が向いているでしょう．パーセンテージはそんなに多くない．全人口の１割ぐらいかな?

土屋： それには，需要と供給の関係がある．世の中に野球というスポーツしか存在しまければ，猫も杓子野球をするでしょう．プログラマについては，２つの問題がある．１つは粗製乱造になっていること．もう１つは，コーディング能力とか JCL の使い方とか，仕事の入り口の部分で足切りを行われることが多い．つまり，単純な記憶力のよし悪しによる判別です．これだと，そうした単純記憶には弱いが，コミュニケーションやプレゼンテーション能力がすぐれている人は，プログラマにはなれずに，別の間接部門に配属されてしまいます．

しかし，最近の進んだ開発環境,, たとえば当社の SoftBench などでは，別に Unix コマンドを覚えなくても十分に仕事ができるようになってきている．これは，まじめに考えないといけないことだと思います．プログラミングというのは，きわめて多様な精神活動ですから，入り口でカットされてしまうと，ほんとうに才能のある人たちが入れなくなってしまう．

岸田： いわゆる２－６－２の原則というのがありますよね．どんな分野でも，２割はよくよく仕事できる．６割はそこそこ，あとの２割はどうしようもない(笑い)という形で正規分布している．だから，土屋さんがいわれるように，全体の２割くらいは，初級のコーディング･ジョブに向いていないことは十分考えられるが，だからといって，かれらがシステム分析やプロジェクト管理に向いていないとはいえない．

フロア： 職業プログラマの定義はどのように考えたらよいのでしょうか?

土屋： プロにとって大切なのは，お金をもらっていかに効率的に仕事を進めるかということでしょう．既存のライブラリの品質が悪いという場合，あえてそれを使わずに自分で作るということもある．それは，作りたいから作るというのではなく，ローコストでよいものができるから，そうするということです．アマチュアは，お金の感覚なしに，勝手に自分の好きなことをする/できる(笑い)．

フロア： ということは，職業プログラマといわれている人たちの中にも，アマチュアが混ざっているということですね?

土屋： そうだと思います．ただし，何かを作りたいという衝動はとても大切なことで，それをあえて否定するつもりはありません．何か作りたいという時に，それをやみくもにプログラムするのではなく，きちんとしたビジネスプランを作って行動するのが真のプロでしょう．ただプログラミングが好きだということではなくて，プロジェクトがうまく進むように新しいプロセス･モデルを組み立てる．そうしたことも十分な創造力があって初めて可能ことで，何でも既成の枠にはめて管理し規制することがプロではない．

塩谷： 一定の仕様を与えられて，目的の製品を作る．それは，企業の中での生産活動ですから，効率性や経済性が要求される．職業プログラマはそれでお金をもらっているのですから，プロとしての意識がなくてはいけない．そうした意識のもとで仕事をしなくてはいけない．

アマチュアというのは，それが別に職業プログラマであってもよいわけです．ある時点で，その仕事に関してペイしてもらっていなければ，それはアマチュアとしての仕事になる．それゆえに創造性が発揮できるという場合もあります．創作意欲をムリに押し殺すと，よいものはできません．これまでに多くのすぐれたソフトウェア･システムが(たとえばUnix)，組織のなかで不遇であった人たちの手で作られてきた．プロのプログラマたちが，仕事とは別に，自分の欲求があって作ったものが，結果として世の中に広まっている．それは否定できません．

ただし，プロのプログラマである以上は，与えられた仕事はきちんとこなさなければならない．

岸田： 新しい創造的な仕事が，単純に創作意欲だけから出てくるわけではなくて，私の解釈では，組織の中で命令された仕事をやって行くというプロセスの中からでは，なかなか新しいものは出てこない．なぜかといえば，われわれの文明においては，組織は，まるで中世の城壁都市のように，がっちりした城壁に囲まれていて，そこには美しいピラミッド型の管理階層ができあがっている．中国･朝鮮･日本など儒教文化が支配的な地域では，特にその傾向が強い．この管理のピラミッドは見た目はきれいですが，内部はかなり腐っている(いつの時代にも絶えない汚職がその証拠です)．そこでは，人間の創造的エネルギーは圧殺されてしまう．

では創造性はどこから来るといえば，それは，城壁の外の荒野からです．たとえば，中国古代(春秋戦国時代)のいわゆる諸子百家といわれる思想家たちがそうでした．孔子にしても，孟子にしても，何十台かの牛車を連ねて，都市から都市へ遍歴し，王様たちに自分のアイデアをセールスしてまわっていたと，歴史の本には書いてあります．現代でも同じことで，コンピュータ･サイエンスやソフトウェア･エンジニアリングの研究者たちは，やはり企業という城壁の外側に住み，派手(?)なシンポジウムや国際会議でアドバルーンを打ち上げている(笑い)

ところで，城壁の中の見方からすると，ときどき城門を開いて，外からの新鮮な創造エネルギーを取り入れないと，組織の活性化がむずかしい．秦の始皇帝がどうして中国を統一できたかというと，あの国には，総理大臣から参謀総長まで，政治および軍事に関する重要な知識やノウハウを全部外国から輸入するという立派な(?)伝統があったからです．

したがって，いま議論されているプロフェッショナルの定義も，組織の中でうまく立ち回るということもさることながら，そうした精神の城壁をいかに崩して，何をするかということを考えなければいけないと思います．

司会： 岸田さんは以前，企業の中に遊園地が必要だというお話をされていましたよね．そのこととの関係

は?

岸田： 壁の外にある新しいものを取り入れるために，正規の城門以外に別の入口を開けておいて，そこからどんどん新しい情報が入ってくるようにしておいたほうがよいのではないか，ということです．

土屋： いまの話との関連で一言．これまで，私はいくつかの会社を変わってきました．いまの会社でも，いろいろな部門を経験して，最近ソフトウェア営業本部に移りました．ここは，社内ベンチャみたいな形で，ファクトリもセールスもサポートも持っている．ただ規模は小さい．つまり，外からではなくて中から壁をぶち抜いていく．そういうことを試みようとしているのです．いま，孔子や孟子の話が出ましたが，レオナルド·ダ·ビンチなんかもやはり，キャラバンを組んでいて諸国を遍歴している．

しかし，いまプログラマが個人で　組織の壁をぶち抜くいうのは，非常にむずかしい気がします．会社の中でトライするとしたら，何人かの人が集まってベンチャ的な形で，知恵を働かせてやっていくしかないのではないでしょうか?

フロア： それは，いまのの会社組織では，創造力を発揮するのは困難であるということですか?

土屋： 組織には方針がある．方針は上意下達です．上の方針と下のうやり方が矛盾してはいけないのです．方針展開というのは，１つの価値観の展開で，これがある種の目に見えない城壁になっています．別の観点からそれをぶち抜かないと，創造性は出てこない．それをやるためには，上の方針に依存しないでも行動できるような受け皿(基盤)を持ってなくてはならない．それを個人が持つことはむずかしい．

岸田： 孔子の場合は，子貢という弟子がいて，うまく金を儲ける．その金を使って遊説キャラバンをやっていたようです．

土屋： 何を学んでいくかということに戻ると，私は「われわれは孤独ではないんだよ」ということを学んでいくべきではないかと思います．

松尾： 城壁には，中にいるものを守ってくれるという別の機能があります．ただ，城壁はいつ崩れるかわからない．オブジェクト指向は城壁なのか，それとも荒野からきた破壊エネルギーなのか?　将来は，また別のものがあらわれるでしょう．門を開けて待つよりも，いつ崩れるかわからないのだから，崩壊にそなえて体を鍛えておくということが大切でしょう．いずれにせよ，城壁が永遠に存在するということを前提にしてはいけない．

3.3. 明日に向かって

司会： ２番目のテーマについては，とりあえずこれで終わりにしたいと思います．

最後に，これ迄の議論を踏まえて，「個人ではできることは少ない」という話があったあとで聞きにくいのですが，この交流会では，いつも最後に「明日から何ができるか?」ということを考えています．この点に関してコメントしていただけますでしょうか?

岸田： チームとか組織とかを考える場合，われわれの頭の中に既存のモデルがありますよね．そのモデルの正しさを，ときどき疑ってみる必要があるのではないかと，最近思っています．午前中にお話しした後藤先生の「メッカ」という本を読んで，イスラームの都市モデルは西欧やアジアの都市モデルと根本的に異なるという話にかなりショックを受けました．明日からできるかどうかはわかりませんが，これから，アラビア語とコーランの勉強をしようかと考えています(笑い)．

それと，少し関連しますが，先日 NHK でやったモンゴル帝国の連続特集はおもしろかった．あの帝国は，他の王朝と違って，周囲を城壁で囲まなかった．ユーラシア大陸全土をカバーする道路網を張りめぐらして，そのインフラストラクチャに依存して，政治·経済のシステムを維持しようと試みた．まさに，これからのコンピュータ·インタネット時代におけるソフトウェア·ビジネスの概念的プロトタイプとして，参考になりそうな気がする．これも，モンゴル帝国の参謀役だったアラブ系の人たちのアイデアです．そういった意味でイスラームにいま興味を持っています．われわれが全員イスラームに改宗すると，プログラマ革命ができるのかも知れません(笑い)．

塩谷： ちょうど１種間前に NeXUS (NeXT ユーザ会) のシンポジウムがあって，そのパネルで，いま手がけている PCTE や，トースタ型環境モデル，あるいは環境レポジトリのスキーマ設計の話を紹介しました．明日に役立つ技術を考える場合，もちろん保守的ではいけなくて，いろいろな新技術を積極的に吸収して行く必要があるのですが，そのとき，表面的な現象だけを追いかけるのではなく，背後にあるコンセプトをしっかり掴まなければならない．それには，ER をはじめとする各種モデル化技法をしっかり勉強しないといけないでしょう．

土屋： すぐ明日に役に立つというのは，体力ですね(笑い)．知的な体力がなければ，途中で考えるのをあきらめてしまう．たとえば，デバッグだって，単純にステップを追いかけるのではなく，別の視点から検討しようとするときには強い体力が必要です．そうした知的パワーの養成には，スパーリングが第１です．たとえばこういった集まりで，真剣に討論に参加する．あるいは，岸田さんがいわれたように，歴史書や哲学書を読む．そういうことが重要ではないかと思います．

松尾： ２つあります．１つは流行にだまされないこと．ソフトウェア業界は，ともすればファッションを追いかける傾向がある．一昔前の MIS は，いまの SIS と本質的に何ら変わらない．

２つ目ですが．私自身，以前ソフトウェアハウスにいた時，いろいろな会社に派遣され，さまざまな環境で仕事をしなければなりませんでした．そんな状況下では，何が役立つかといえば，さきほど塩谷さんからモデリングの話が出ましたが，まさにその通りで，普遍的に応用可能技術を学んで置かなければならない．ただし，そうした技術は普遍的であるがゆえに特定のものがない．そういう場合は先人の跡をたどるのが一番です．ER もそうだし，データフロー分析もそうです．

しかし，１番目にお話ししたことも忘れないでください．大切なのは，マルや四角を描くことではないし，ツールの使い方でもない．分析方法の本質を学ぶことが大切なのです．そういうものを身につけることによって，一匹狼として，どこでもやっていけるという体力をつけることが重要だと思います．

司会： どうもありがとうございました．これで，パネルを終わります(拍手)．

4. 青空の彼方から　ープログラマ考ー

最後にまとめとして，私自身がプログラマについてどのように考えているかを述べたいと思います．

4.1. プログラマについての幻想

以下に挙げるいくつかの例は，われわれプログラマに対する(意識的に仕組まれた)幻想の代表である：

- プログラマが不足する．
- これからは職人プログラマは不要で，工業プログラマというべき個性のないプログラマのみが必要となる．
- プログラマは実はもういらない．SE が不足している．
- CASE はプログラム作りに有効である．

短期的には，たとえば流れ作業方式のソフトウェア工場のアイデアは，金銭的には正しいのかも知れないが，少なくとも私には(サラリーマンとしてではなく，１人の人間として)興味がない．

ヴィトゲンシュタインのことばを借りれば，「科学の問題に私は興味を持つが，ほんとうに心を惹かれるのは，概念と感性の問題だけである」ということであろうか．したがって，以下に展開する私の議論も明日の生活費を稼ぐためのものではなく，そういう意味では，もしかしたら，この記事自体が多くの人から怒りを買うかも知れない．

しかし，私には，日頃心の中にわだかまっていることを書きたいという自分自身の気持ちをどうしても押さえることができない．もし，この小論を読んで不愉快になる人がいたら，個人的な意見ということで容赦願いたい．

4.2. プログラマとは何か?

今回の会合で，岸田さんから，プログラマについてのニュース記事を見せていただいた("Real Programmers Don't Use PASCAL"，Ed Post)．このユーモア記事の面白いところは，それがかつてのプログラマをステレオタイプに，そして諧謔的に描いているようで，実は大きい声ではいえないこと，あるいはまともに説明するには時間がかかことを，もう一度，すなわち２重に変換することで強烈に表現している点である．

たとえば，真のプログラマはコメントを必要としないとか，Goto の利用を恐れないといった表現に対して，ふつうの人ならただ笑うだけであろう．私自身，自分のプログラムに Goto を見つけた人間から素人呼ばわりされたことがある．およそ一度でもプログラムを書いたことがある人間なら，わざわざ Dijkstra の CACM Letter を持ちだすまでもなく，Goto の弊害を知っている．しかし，そのことと，実際に利害のトレードオフを行った上でGoto を使うこととは，明らかに別である．

コメントに対しても同様のことがいえる．だれもが，コメントをつけなさいという．しかし，だれもコメントの有害さについては教えない．コメントはメンテナンスの対象になりにくい．プログラムと違ったコメントが残るは有害である．これは，コメントが実行

されないというまさにその性質ゆえである.

つまり, 何がいいたいかといえば, プログラムは, <最終的に機械で実行されるという意味において単純であるが, その単純さの総和は複雑であり, 全てを単純さへと還元することはできない>ということである.

この命題の意味することを, いくつかの例を挙げて説明しよう.

(1) 数値解析のプログラミング

何年か, 社内の新人に数値解析の教育を行ったことがある (新人のうち1割程度は情報科出身であるが, 残りは, さまざまな理工学系出身の人たちであった). 一般的には, オイラー法やニュートン法といったところから入って, 偏微分の解析方法としてガウス・ザイデル法や最適化手法へと, 計算機を使ってやっていくのだと思う.

しかし, 私は一切計算機を使わなかった. 正確にいえば, ふつうの関数電卓しか使わなかった. なぜかというと, そのほうが, 計算機で計算をすることのムズカシサがよくわかってもらえるからである. たとえば, たいていの受講生は, 電卓で計算するほうが, 単精度でプログラム計算するより精度がよいということに驚く (IEEE754 準拠だとその有効桁数は 7.2 桁でしかない).

そうすることの利点は他にもあって, アルゴリズムの差異 (計算時間, 精度) といったものを実感できることである.

たとえば, 最初にオイラー法で何か微分方程式を解いてみる. 徐々に刻みを小さくしながら, 誤差がどのように減っていくかを真値と比較しながら見ていく. 次に, 4次のルンゲクッタ法を用いて, 電卓で計算する. ある精度を出すのに対する手間が少ないことに, 指の痛さを通して, すぐに気がつく (といっても数分間は電卓をたたく必要があるが). あるいは, 途中でメモすることになる中間値の数から, 必要な中間記憶の量が, 鉛筆の芯の量として実感される. こういった過程を通して, まさに身体で, 表現誤差, 離散化誤差, 計算誤差といったさまざまな誤差や計算時間に対するセンスを, 身につけることができるようになる.

もちろん, 応用プログラマとしては, これだけで現場に立てるわけではない. 多くの問題解決アルゴリズムを学んで行かなければならない. しかし, 数値解析の応用プログラマとしての最も基本的素養は, どのFortran 演習問題にもあるニュートン法のプログラムを書く以上に役に立つ.

これは,「プログラムを (直接に) 教えないことにより, プログラム技術が向上する」ということの1例である.

(2) 協調活動としてのプログラミング

ある程度以上の規模のプログラム (システム) は, 複数人で開発することになる. この時の協調はどのように行われるのだろうか?

たとえば、CSCW の世界を考える時に, 一般的には時間と空間の直積をとった4つの視点から考えられる. また, 合わせてマルチメディアが問題になる. なぜか?

これは, 同時／同空間がとりあえず望ましいという合意があるからで, そのために, あたかもそこに人がいるかのようなビデオ・音声が必要だということであろう. しかし, いわゆる, 現実の同時／同空間の会議がむなしいことは多々ある.

協調活動を考える上での危うさは, 人間をあまりにも機械的なモノ (すなわちこういう入力を与えるとこういう出力を出す) と考えているところにあるのではないだろうか? いや, そうではない. フレーム問題を通して, 入力＋環境により出力が定まる. さらに時間的な経過も含めれば完全だといわれるかも知れない. しかし, そんなことが可能であろうか? もし可能だとしても, それにどれだけの意味があるのだろうか?

つまり, 人との協調を考える上で, 人間を反応論的にモデル化する必要は必ずしもないし, おそらく不可能である. では, 実体論的に (つまり人間はこう考えるものであるからと) モデル化できるのであろうか?

以前何かの学会で, 計算機に感情を持たせるという議論を聞いた. 計算機に感情を持たせることで, ユーザフレンドリなコンピュータができるというものである. その方法は (本人がどれだけマジメかは知らないが), たとえば, 悲しみ (10) －喜び (3) ＝悲しみ (7) といった算法にしたがう.

ベルクソンの思想 (たとえば「意識に直接与えられたものについての試論」) を持ちだすまでもなく, こういった単純な算式が成り立たないこと, つまり, われわれの感情は数量化できないことは明かであろう (私はこういった単純化がソフトウェア工学の世界で多いことに驚く). これは, すなわち実体論的に, 人の思

考を，少なくとも現時点では (いずれ遠い将来にその可能性があるとしても)，考えることが不可能だということを示している．

話がプログラマから離れてしまったが，人間は複雑な生き物である．したがって，平均的といったプログラマのモデル化ができない以上，そこにわれわれは種々の異なったプログラマを前提とすべきである．これは，「均一のプログラマを作ることはできない」，「安易なプログラマのモデル化は危険である」という例である．

つまり，この問題に対するわれわれの立場は，いかにして問題を１人の人間が一時に扱える単位にするかということであり，こういった謙虚な立場でしか解決策はないということを意味している．

(3) プログラマの能力

プログラマとは一体何をする人であろうか? 明らかに，プログラマはプログラムを書く人である．

そうである以上，基本的な能力として，プログラムを正しく書く能力を持っていなくてはななならない．このことは，字面以上の重要な意味を含んでいる．プログラムそのものにどれだけ集中できるか? この意味で，プログラム中のコメントやドキュメント等は不要である．なぜなら，これらは実際に実行されはしない．

プログラマが正しくプログラムを書かないという前提に立つ技術 (たとえば形式的方法) の有効性は認める．だからといって，標準的な環境において，正しくプログラムを書く能力の有用性が減るわけではない．

ここまでが能力の欠くべからざる第１段階である．しかし，少なくとも私がいっている意味でのプログラマには，それとは別の能力もまた必要とされる．

プログラマは，作るべきプログラムが何のために作られるのかを理解してなくてはならないし，プログラムが正しく動作することを保証できなくてはならない．このことが，プログラマに要求される第２の能力である．

(4) プログラマの関与する範囲

別の観点からこのことについて考えてみる．いままでのソフトウェア工学の大きな目標の１つに，いかにして多人数が協調して働けるかということがあった．そして，そのことは (2) 項で見たように，容易ではない．

しかし，「どうしたら少人数で (大規模な) ソフトウェアが作れるか?」を考える方が自然ではないのか? つまり，逆の視点から，われわれのいままでの知見や道具を利用しつつ，個々のプログラマの能力を拡充できるならば，問題に対する見通しが，より明るくなるのではないか?

少人数であれば，人数の２乗に比例するような膨大な人間同士のインタラクションを気にする必要もないし，プログラムも，レベルはどうあれ均質化する．

たしかに，現実は，それぞれの組織の開発能力に応じて，ソフトウェアの規模も増加傾向にあることはわかる．しかし，個々のプログラマの能力をいかに拡充し発揮させるかという視点が，現状ではあまりにも欠けているのではないだろうか?

こういうふうに考えると，「そういう人間は特別優秀であって，それを前提にした議論はできない」という反論がすぐ聞こえてきそうである．

しかし，機械設計，建築設計等，どんな設計でもそうではないのか? 職業としての設計分野で，だれにでもできるという世界があるのだろうか?

次に来る反論は，「プログラムは設計ではない」というものであろう．

ここで，本項のテーマとつながるのであるが，私自身の考えているプログラミングの仕事は，少なくとも，いわゆる設計から，コーディング，検証までを指している．さらに，これらは不可分であると考えている．

この範囲でのプログラマ作りを考えることで，さきに述べたプログラマの能力についての評価も可能となる．つまり，評価は，完成したひとまとまりのプログラムが，要求者の期待しているものにどれだけ近いかという点において，なされるのである．そのことは，プログラムのメトリクス (たとえば，循環複雑度や結合度といったシンタックスの評価) から，プログラムの意図する意味の評価へ，ということである．

4.3. プログラマが必要とする道具

乱暴にいえば，何も必須なものはないのだろうが，もう少し常識的な範疇で考えてみよう．

コンパイラ，エディタ，デバッガ．それから，ドキュメント用の文書整形ツールあたり．さらにプログラムの差分を取ったり，shell のように自動実行するためのユーティリティの類．これらは，必須ではないが，あったほうが便利である．

CASE はどうだろうか? このレベルまで来ると，

あったほうが便利という比重も低くなる．それと同時に，いまのCASEが生みだす幻想は，良心的なプログラマを混乱させる．

CASEについてよくいわれることで，「これを使うことで，だれでもプログラミングできるようになります!」という宣伝文句がある．はたして，このような道具をわれわれは欲しているのだろうか?

たとえば，次のようなことを考えてみよう．遊園地にあるゴーカートはだれにでも乗れる．子供から老人までだれにでも乗れる．たとえば，もし，私とF1ドライバのプロストがこのゴーカートで競争したら，どうなるだろうか．もちろん，プロストが勝つだろうが，その差はせいぜい2倍もないであろう．では，レーシングカーで競争したらどうなるだろう? たちどころに数倍の差がつくにちがいないし，もしかすると私はスタートすら切れないかも知れない．

本来プロの道具とは，こうあるべきである．職業としてプログラムを作る人間が，いかにしたら大きな能力向上を得ることができるかが，まず考えられるべきである．

自分自身も世の中でCASEと呼ばれる類のものを作っていて，つねに思うのはこのことである．お絵描きツールが重要なのではない．チェック機構が重要なのではない．設計方法論すら，このコンテキストの中では不要である．なぜなら，描画エディタは考える力に，チェック機構は注意深さに，設計方法論は経験に，<決して>勝てないからである．

CASEが，考える力や注意深さや経験を必要としないで使えるものだとしたら，それはゴーカートみたいなものでしかない．CASEに価値があるとしたら，充分な考える力と注意深さと経験を有している人間が，その力を数十倍にも発揮できるものでなくてはならない．

さらに，そのプロの道具としてのCASEは，F1カーと同様に，使う人や環境によりカスタマイズされる必要があり，そうあるべきである．そのカスタマイズに耐え得るようなメタなCASEが作れるかどうかが，本来的な道具としてのCASEが生き残れるか否かの条件であろう．

4.4. 結論，そしてプログラマの将来

ソフトウェアはこれまで，作業の機械化による業務の合理化という側面から多く作られてきた．その意味において，一通り機械化できる分野におけるプログラムは，充分とはいえないまでも，ある程度行きわたった．今後作られるソフトウェアは，それらの改良か，あるいはいままで機械化しにくかった部分のプログラム作りという，いわば将棋崩しの最後に残ったむずかしい駒の部分に移行して行くのは明らかである．そして，いままでのような(ある意味で見境のない)プログラミング需要はなくなるはずで，いわば産業としてしかるべき位置に来たということであろう．

そうした過渡期的な状況において，現時点でのプログラマに対する枠組み(いくつかは"幻想"として挙げた)では，問題は解決しない．

まず第一に，よいプログラマとは何かを見直すことが重要であり，そうすることでソフトウェア開発をめぐるさまざまな問題についての見通しをよくすることができる．

いままでの議論を通して，将来プログラマに要求されるのは，概念をたくみに操る能力であり，すぐれた感性であるということがわかる．また，そうしたプログラマのみが，今後生き残れるのではないかと思う．

そうして，プログラミングはアートへ変身する．少なくとも，私自身はそういう世界で仕事をしたいと考えている．

The United Nations University

UNU/IIST

International Institute for Software Technology

Current Topics in Programming Methodology

An IFIP WG2.3 — UNU/IIST Seminar

Jan. 10–21, 1994, Macau

This two-week seminar is aimed at young University Lecturers, Academy Researchers and leading edge Software Industry Technologists who are either scientifically or technologically active in the area of Programming Methodology — and who are from Far or South-East Asia, or the Pacific.

The seminar is organized by UNU/IIST and is otherwise sponsored by IFIP TC2 WG2.3: the Programming Methodology Working Group 2.3 of Technical Committee 2 of the International Federation for Information Processing.

The emphasis of the two week seminar is that of Programming Methodology: techniques (supplemented by underlying foundations) for the construction of elegant and correct software.

Programme & Lecturers:

1. Ralph-Johan Back, Seminar Programme Chairman, Prof., Åbo Academy, Finland
   REFINEMENT CALCULUS, SELECTED TOPICS: rules for mechanical reasoning and the use of the calculus in the stepwise refinement of parallel and reactive programs.
2. Dines Bjørner, Seminar Organizer, Prof., Director, UNU/IIST, Macau
   REQUIREMENTS AND SOFTWARE ARCHITECTURES: techniques for abstract specification of large scale application domains and software systems.
3. Eric Hehner, Prof., Univ. of Toronto, Canada
   A PRACTICAL THEORY OF PROGRAMMING: covers sequential and parallel programming, communicating processes, and nonterminating computations.
4. Michael Jackson, Independent consultant, London, England
   REQUIREMENTS AND SPECIFICATION METHODOLOGY: development is viewed as the construction of descriptions: What descriptions are needed? How are we to construct them? How do they fit together? How are they to be understood? What languages should be used?
5. Jayadev Misra, Prof., Univ. of Texas, Austin, USA
   CONCURRENT-PROGRAM STRUCTURING: we propose a design theory for concurrent programs by extending the UNITY-logic: data objects, module-by-module design of concurrent systems, and reasoning about the interactions between a process and its environment.
6. Carroll Morgan, Univ. Lecturer, PRG, Oxford Univ., England
   THE REFINEMENT CALCULUS: The refinement calculus is a method of deriving and presenting imperative program developments using refinement laws, and mixed programs and specifications.
7. Zhou ChaoChen, Prof., Principal Research Fellow, UNU/IIST, Macau
   THE DURATION CALCULUS: a design calculus for specifying and reasoning about safety criticality and real-time.

Each lecturer will give 4 double lecture of 2 times 45 minutes. Each of 10 lecture days will feature 3 such double lectures plus one discussion session.

Contact us for detailed information: lecture abstracts, publication references and lecturer bio-data. We will then send you a 16 page folder.

Here's How to Apply: If you wish to attend this IFIP TC2 WG2.3 Seminar then mail or fax us an application letter containing the following information:

1. Your full name
2. Complete mailing address
3. Telephone, fax and e-mail numbers
4. Your educational background: M.Sc., Ph.D. and granting institution
5. Your research interests — 10-20 lines at most
6. Your publications, incl. perhaps sample of your recent reports
7. Important: Proof of proficiency in English
8. Important: Recommendations

Costs: You should expect the following, approximate cost cost of attending the seminar:

1. For some: International return air travel	Approx. US$ 250–750	
2. For some: Hong Kong – Macau return travel		US$ 40
3. Thirteen nights in hotel	Approx. US$ 75/night:	US$ 975
4. Thirteen days of meals	Approx. US$ 25/day:	US$ 325
5. Seminar fee		US$ 500

The organizers will secure single hotel rooms provided you pay a full (US$ 975) deposit in advance to UNU/IIST.

Please do not send any monies now. Wait till we have accepted your application. We will then instruct you on how to proceed.

Here's How to Reach Us: Our postal, electronic communication and visiting addresses are:

Postal address:	Electronic addresses:	Visiting address:
Miss Chiu Chi On (Anna)	E-mail: cco@iist.unu.edu	UNU/IIST
Programme & Project Clerk	Phone: +853-712.930	Edf. Banco Luso Intl. 18/F
UNU/IIST	Fax: +853-712.940	1-3 Rua Dr. Pedro Jose Lobo
P.O.Box 3058, Macau		Macau

Scholarships: A small number of scholarships are offered by UNU/IIST. They cover either of:

- International travel
- Lodging and meals and/or
- Registration fee

Should you need financial support then please write us urgently stating reasons for and amount of support needed.

Deadlines:

- Send in your application to reach us by October 1, 1993
- We will notify you of admittance (or not) to reach you by October 15, 1993
- We will send you detailed instructions from Macau November 1, 1993

Visa Formalities: Should you require visas (to Hong Kong and/or Macau), then please notify us now (urgently), so that we can send you a formal, tentative letter of invitation with which you can then start applying for a visa. Our sending you such a letter is no guarantee that you will eventually be admitted to the seminar.

論 文 募 集

ソフトウェア･シンポジウム '94

1994 年 6 月 15 日 (水) ～ 17 日 (金)
於： 金森ホール (北海道函館市)

主 催：
ソフトウェア技術者協会 (SEA)
協 賛（予定）
日本ソフトウェア科学会　情報処理学会
情報サービス産業協会　北海道ソフトウェア協会

ソフトウェア･シンポジウムは，ソフトウェアハウス，コンピュータメーカ，計算センタ，エンドユーザ，大学，研究所など，さまざまな場で活躍している技術者，管理者および研究者が一堂に会し，ソフトウェア技術に関する多面的な経験や知識を交流するユニークで貴重な場を提供してきました．迎えて第14 回となる来年のシンポジウムは，初めて本州から離れ，会場を北海道の函館に移して開催します．

毎年，現場に立脚した質の高い論文が集まるこのシンポジウムのレベルは，回を重ねる毎に着実に上がっております．今回は，会場の関係もあり，単一トラックでセッションを構成します．そのため採録可能な論文は 15 編程度と少なくなる見通しで，例年にも増して高いレベルの論文採否が予想されます．そこで，従来からの当シンポジウムならではの現場に立脚した論文を尊重する意味で，採録の審査を，(1) 経験報告と (2) 技術論文に分けて行うことにしました．

なお，従来通り最も優秀な論文に贈られます最優秀論文賞 (服部賞) は，シンポジウム開催時に選定します．その際は，経験報告，技術論文の区分によらず，全論文を一律に選考審査の対象とします．経験報告，技術論文とも，日本語の他，英語による投稿も可とします．ただし，発表は日本語に限ります．

応募論文テーマとしては，たとえば;

・ 分析／設計技法　・ 品質管理　・ オブジェクト指向　・ 開発環境　・ ネットワーク
・ プロセス管理　・ CASE　・ 新時代のパラダイム　・ プロジェクト管理　・ 人間的要因

などさまざまなものが考えられますが，必ずしもこれらにとらわれる必要はありません．

応募要領

応募論文は未発表のものに限ります．これまで，概要のみによる応募も可とすることがありましたが，今回は本論文による投稿のみを認めますので，御注意下さい．また，他への二重投稿は御遠慮下さい．様式は自由ですが，A4 サイズで 5 ～ 10 ページ程度を目安とします．応募論文は，別掲のカバーシートに必要事項を記入の上，**1993 年 12 月 17 日**までに 2 人のプログラム委員長のいずれか宛に，郵送 (この場合は 5 部) ないし電子メールでお送り下さい．電子メールの場合は，roff, TeX, LaTeX, PostScript, 単テキストのいずれの形式でも構いません．プログラム委員会で内容の審査を行い，結果を 2 月下旬までに，応募者全員に通知します．その他不明な点がありましたら，プログラム委員長まで御問い合わせ下さい．

主要スケジュール (Important Date) :　1993 年 12 月 17 日　論文応募〆切
1994 年 2 月 25 日　論文採否通知送付

シンポジウム･スタッフ

実行委員長
杉田義明 (日本 NCD)
プログラム委員長
玉井哲雄 (筑波大)
渡邊 雄一 (アスキー)
プログラム委員
青山幹雄 (富士通)
荒木啓二郎 (奈良先端大)
伊藤昌夫 (MASC)
大塚理恵 (RSK)
大場充 (日本 IBM)
兼子毅 (東京大)
岸田孝一 (SRA)
久野靖 (筑波大)
熊谷章 (PFU 上海)
元治景朝 (さくら KCS)
佐伯元司 (東工大)
坂本啓司 (オムロン)
佐藤千明 (長野県協同電算)
塩谷和範 (SRA)
武田淳男 (安川電機)
高橋光裕 (電力中研)
中野秀男 (大阪大)
布川博士 (東北大)
野口正浩 (新日鉄)
野呂昌満 (南山大)
藤野晃延 (FXIS)
二木厚吉 (北陸先端大)
細谷僚一 (NTT)
堀江進 (NES)
松本健一 (奈良先端大)
山崎利治 (日本ユニシス)

ソフトウェア･シンポジウム '94

論文応募カバー･シート

氏名 (筆頭著者)： ________________ （ふりがな） ________________

所属 (会社, 大学, 組織)： ________________

部門（学部）： ________________ **役　職：** ________________

住　所：（〒　　　）________________

TEL: [　　　－　　　－　　　　内線　　　] **FAX:** [　　　－　　　－　　　]

E-Mail: ________________

論文タイトル: ________________

共同執筆者（もしいれば):

氏名: ________________ **所属:** ________________

氏名: ________________ **所属:** ________________

氏名: ________________ **所属:** ________________

氏名: ________________ **所属:** ________________

論文概要:

____

____

____

____

____

____

____

____

____

応募カテゴリ:(いずれかにチェック)

□：経験報告　　□：技術論文

キーワード:(最低２つ、最大５つのキーワード挙げて下さい)

1 : ________________

2 : ________________

3 : ________________

4 : ________________

5 : ________________

------------------------ 送　付　先 ------------------------

〒112 東京都文京区大塚 3-29-1
筑波大学大学院 経営システム科学 助教授
玉井 哲雄
TEL: 03-3942-6869
E-MAIL: tamai@gssm.otsuka.tsukuba.ac.jp

または

〒215 神奈川県川崎市麻生区南黒川 12-1
(株) アスキー 管理本部 ＯＡ推進部
渡邊 雄一
TEL: 044-989-9435
E-MAIL: yuu-w@ascii.co.jp

********** **応募締め切り： 1993 年 12 月 17 日（金）** **********

SEA Seminar & Forum (September 1993)
参加者募集

コンピュータ先進国でありながら，ことネットワークに関してはなぜか後進国でありつづけたわが国の奇妙な状況がようやく変わろうとしています．昨年の後半から，北海道·東北·関西·中四国·九州などの各地で，地域ネットワーク構築の動きがいっせいに活発化し，また，これまで junet や WIDE などで実験·研究されてきた技術を実用化して全国規模で商用ネットワーク·サービスを提供することを目的とする会社も，ついにキックオフされました．

そこで、来月の SEA Seminar & Forum では，そうした本格的なコンピュータ·インタネット時代の到来が，ソフトウェア産業の基盤構造やビジネスの形態にどのような変化をもたらし，われわれソフトウェア技術者のライフスタイルにどんな影響を及ぼすかを，さまざまな視点から検討してみようと考えました．多数の方々のご参加をお待ちします．

開催および申込要領

1. 日時： **1993 年 9 月 14 日 (火)　9:30 ～ 16:30**

2. 場所： **文祥堂イベントホール B1F** (東京都中央区銀座 3-4-12)

3. プログラム：

Seminar (9:30 ～ 12:30) with Demonstration

ネットワーク新時代　ー　これからのビジネス基盤としてのインタネット

講師：深瀬弘恭，吉村伸 (Internet Initiative)，佐野晋 (NEC) ほか

いま計画されている商用コンピュータ·インタネットは，われわれにどんなサービスを提供してくれるのか，またそれは，これからのソフトウェア·ビジネスに対していったいどんなインパクトをもたらすのかを，IIJ のスタッフの方々を招いて，サービス·ベンダの立場から，デモを交えて具体的にお話しいただきます．

Forum (13:30 ～ 16;30)

インタネットを利用した分散ビジネス/分散開発

コーディネータ：中野秀男 (大阪大学)

パネリスト：　Seminar 講師各氏に加えて，杉田義明 (日本 NCD)，西武進 (SES)，他 2 ～ 3 名 (交渉中)

ビジネスあるいは研究開発，またはコミュニケイションの場としてのインタネットの発展は，これからのソフトウェア·コミュニティのライフスタイルをどう変えて行くのでしょうか？　この興味ある話題を，フロアからの質疑や意見，コメントも交えて多角的に議論してみようと考えています．

4. 参加費：

SEA 正会員	15,000 円
SEA 賛助会員	20,000 円
一般	30,000 円

5. 申込み方法：　下の申込用紙に必要事項を御記入の上，SEA 事務局まで Fax，郵便または E-Mail でお申込みください．折り返し受講票をお送りします．今回は，いつもとちがって Seminar & Forum 一括受講のみとします (いずれか一方だけの参加は受け付けません)．なお，参加費は当日会場受付にてお支払いください (領収書を差し上げます)．申込受付後のキャンセルは原則としてお断りします．

申込み宛先：　〒160 東京都新宿区四谷 3-12 丸正ビル 5F　ソフトウェア技術者協会 (SEA)

Tel:　03-3356-1077　　Fax:　03-3356-1072　　E-Mail:　sea@sea.or.jp

SEA Seminar & Forum (Sep. 1993) 参加申込票

氏名 (ふりがな)：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿(　　)

会社名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

部門·役職：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

住所：(〒　　)＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

Tel:＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿　Fax:＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

種別 (該当欄にチェック):　☐　正会員 (No.　　)　☐　賛助会員 (No.　　)　☐　一般

ソフトウェア技術者協会
〒160 東京都新宿区四谷3-12 丸正ビル5F
TEL.03-3356-1077　FAX.03-3356-1072