SEAMAIL

Newsletter from Software Engineers Association

Volume 8, Number 2 July, 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．２　　１９９３年７月３０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

☆

Vol.8, No.2 をお届けします！あまりの快調な発行で，事務局としては嬉しい悲鳴をあげています．編集長が精を出している原因は黄色と黒の縞々チームが不振のせいで，今期の勝敗をあきらめているためでしょう．

☆☆

巻頭にある何やらむずかしい原稿は，先日の幹事会にオブザーバとして出席した佐原さんが，お弁当をタダで食べては悪いから.... といって．寄稿してくださったものです．

☆☆☆

編集長から，新年度の幹事会メンバーに依頼の e-mail/fax が出されていて，着々と原稿が集まりつつあります．この号には，その第１弾として，６人の方々からのエッセイを載せました．

☆☆☆☆

一般会員の方々も，ぜひ，論文／技術メモ／随筆 etc 何でも結構ですので，原稿をお寄せください．sea@sea.or.jp 宛に e-mail で plain text を送っていただくのが一番便利だと，編集部がいっています (最近は Nifty からでも送れます!).

☆☆☆☆☆

関数型言語 ML による仕様記述の試み

佐原 伸
(SRA)
sahara@sra.co.jp

0. はじめに

オブジェクト指向分析 (OOA) や構造化分析 (SA) を教えていると，受講生は必ずといってよいほど，仕様記述のところでつまずく．これは OOA や SA で仕様記述言語が決っておらず，「前条件や後条件などを使って，何をするのか，適当に仕様を書け」と教えざるを得ないためである．もちろん，前条件や後条件の書き方は教えるのだが，すべてを厳密な文法で規定しているわけではないので，どこかで必ず曖昧さが生じる．ここを適当に日本語を交えながら，しかもできるだけ曖昧さがないように記述するには，たしかに OOA や SA 以外にいろいろな知識がいる．そこで，仕様記述言語を一応定めて，小さな仕様を書いて実験してみた．

1. 仕様記述言語 ML

OOA や SA の仕様は「操作仕様」とか「プロセス仕様」とか呼ばれているが，結局何らかの機能を果たす関数の仕様と解釈できるので，関数型言語 ML を仕様記述言語として採用した．分析段階では"How ではなく What を書く」ことが重要なので，手続き的言語でなく，宣言的プログラミングができることもありがたい．フリーの ML 処理系 (Standard ML) が Unix や Macintosh 上で動くことも，ML を採用した重要な動機である．

2. 例題

とりあえず，日曜日の夜のやっつけ仕事で片付けられるように社内の「出張費の精算」を例題に選んだ．詳細は，仕様記述と共に説明していく．

3. 仕様

さて，ML は漢字がいまのところ処理できないのであるが，仕様記述にはやっぱり『漢字』が使いたい．で，仕様記述は漢字混じり仕様とし，漢字をローマ字に直して ML のプログラムとし，プロトタイプを作って動かしてみることにする．

まず，出張費の定義は次のようになる．

```
fun 出張費 (宿泊費, 交通費, 日当, 雑費) =
    合計 (宿泊費) + 合計 (交通費) + 合計 (日当) + 合計 (雑費);
```

出張費は，宿泊費･交通費･日当･雑費を引数とした関数で，「合計」という下働きをする関数を呼び出している．合計の定義は次のとおり．

```
fun 合計 [] = 0
  | 合計 (n::ns) = n + (合計 ns);
```

「合計」はリストの各要素を合計する関数である．まだ分析の段階で「リスト」などという設計や実現に絡む用語が出てくるのは気に入らない．が，何日間か出張していたとして，その日付分の宿泊費などを「合計する」ことを厳密に書くとすれば，データの持ち方を決めないわけにもいかない．「合計」の定義は，リストが空だったら 0 を返し，整数 n に続いてリスト ns が来るパターンの時は，再帰的に「合計」を呼び出し，n にその合計を加える．(合計 ns) は合計 (ns) と書いてもよい．このパターンマッチが ML の強力なところである．合計の動きを引数が [1,2,3] の場合について書き下すと，以下のようになる．

```
合計 ([1,2,3]) = 1 + 合計 ([2,3])
    = 1 + ( 2 + 合計 ([3]))
    = 1 + (2 + (3 + 合計 ([])))
    = 1 + (2 + (3 + (0)))
    = 1 + (2 + (3))
    = 1 + (5)
    = 6
```

ML はデータの型を類推し，決定する．情報が足りなければ文句をいう．いまの場合，「合計」は，合計 [] = 0 の定義から，整数のリストを引数とし整数を返す関数であることが分かるので怒られない．したがって，「出張費」関数も整数のリストを 4 つもらって，整数を返す関数になる．

宿泊費は，肩書と宿泊した都市と領収書の有無を引数とした関数である．定義は次のとおり．

```
fun 規定宿泊費 (肩書, 都市, 領収書) =
    if not 領収書 then 規定.宿泊費 (肩書) div 2
    else if 規定.大都市 (都市) then 規定.宿泊費 (肩書) + 1000
    else 規定.宿泊費 (肩書);

fun 宿泊費 (肩書) =
    if 肩書 = "主幹" then 9000
    else if 肩書 = "主席" then 8500
    else 8000;

fun 大都市 (都市) =
    if 都市 = "東京" orelse 都市 = "横浜" orelse
    都市 = "名古屋" orelse 都市 = "京都" orelse 都市 = "大阪"
    orelse 都市 = "北九州" then true
    else false;
```

ふつうのコトバでいうと，関数「規定宿泊費」の定義は，「領収書がなければ規定の宿泊費の半分で，宿泊した都市が大都市なら 1000 円割増しで，それ以外は規定通り」という曖昧なものになる．規定.宿泊費 (肩書) という関数呼び出しが出てくるが，これは『規定』というモジュールで定義されている「宿泊費」という関数を示している．モジュールの定義は，あとで全体を示すときに説明する．関数「宿泊費」は，肩書を引数として宿泊費を返しているだけである．関数「大都市」は，都市が大都市なら true を，そうでなければ false を返す．

日当の定義は次ぎのようになる．

```
fun 日当 (肩書) =
    if 肩書 = "主幹" then 3500
    else if 肩書 = "主席" then 3000
    else 2500;

fun 出発日日当 (肩書, 出発) =
    if 出発 > 1200 then 日当 (肩書) div 2
    else 日当 (肩書);

fun 到着日日当 (肩書, 到着) =
    if 到着 < 1200 then 日当 (肩書) div 2
    else 日当 (肩書);
```

「出発日日当」は，出発日の日当を計算する関数で，「出発」という引数は出発時刻を表す．時刻はここでは整数で表してしまっている．本来は，「時刻」というモジュールで表現方法とか制約のチェックをした方がよいのだが，もう月曜日の午前 2 時近いので，手を抜くことにした．ふつうのコトバでいうと，「出発が午後か，到着が午前中だったら，日当は半分ね」ということになる．本来，「日当」関数は出発日時と到着日時をもらって，

その間の日当を計算する方が汎用的だが，これも朝が近付いているので手を抜く．

出張精算の仕様の全体は以下のようになる．

```
signature 規定 SIG =
    sig
      val 規定宿泊費 : string * string * bool -> int
      val 宿泊費 : string -> int
      val 大都市 : string -> bool
      val 日当 : string -> int
      val 出発日日当 : string * int -> int
      val 到着日日当 : string * int -> int
    end;

structure 規定 : 規定 SIG =
    struct

    fun 規定宿泊費 (肩書, 都市, 領収書) =
        if not 領収書 then 規定.宿泊費 (肩書) div 2
        else if 規定.大都市 (都市) then 規定.宿泊費 (肩書) + 1000
        else 規定.宿泊費 (肩書);

    fun 宿泊費 (肩書) =
        if 肩書 = "主幹" then 9000
        else if 肩書 = "主席" then 8500
        else 8000;

    fun 大都市 (都市) =
        if 都市 = "東京" orelse 都市 = "横浜" orelse
        都市 = "名古屋" orelse 都市 = "京都" orelse 都市 = "大阪"
        orelse 都市 = "北九州" then true
        else false;

    fun 日当 (肩書) =
        if 肩書 = "主幹" then 3500
        else if 肩書 = "主席" then 3000
        else 2500;

    fun 出発日日当 (肩書, 出発) =
        if 出発 > 1200 then 日当 (肩書) div 2
        else 日当 (肩書);

    fun 到着日日当 (肩書, 到着) =
        if 到着 < 1200 then 日当 (肩書) div 2
        else 日当 (肩書);
    end;

    fun 合計 [] = 0
        | 合計 (n::ns) = n + (合計 ns);

    fun 出張費 (宿泊費, 交通費, 日当, 雑費) =
        合計 (宿泊費) + 合計 (交通費) + 合計 (日当) + 合計 (雑費);
```

signature「規定 SIG」は，後の方で定義されているモジュール定義のための structure「規定」の型検査の情報からなり，型や値を定義する．いまの場合，たとえば「規定宿泊費」は，1 番目と 2 番目の引数が文字列，3 番目の引数が論理値で，整数が返ってくる値として定義されている．

structure「規定」は，規定に関する定義がひとまとめにされたモジュールである．

4. プロトタイプ

上の仕様を，ほとんどローマ字に直しただけのものが，以下のML プログラムである．(*で始まり,*) で終わる部分は注釈であり，そこに各関数の呼び出し例が書いてある．結果は,>の後に示す．

```
signature KITEI =
 sig
  val kiteiSyukuhakuhi : string * string * bool -> int
  val daitosi : string -> bool
  val nittou : string -> int
  val syukuhakuhi : string -> int
  val syuppatubiNittou : string * int -> int
  val toutyakubiNittou : string * int -> int
 end;

structure Kitei : KITEI =
        struct

        fun kiteiSyukuhakuhi(katagaki, tosi, ryousyusyo) =
          if not ryousyusyo then Kitei.syukuhakuhi(katagaki) div 2
          else if Kitei.daitosi(tosi) then Kitei.syukuhakuhi(katagaki) + 1000
          else Kitei.syukuhakuhi(katagaki);

        fun syukuhakuhi(katagaki) =
                if katagaki = "syukan" then 9000
                else if katagaki = "syuseki" then 8500
                else 8000;

        fun daitosi(tosi) =
                if tosi = "tokyo" orelse tosi = "yokohama" orelse
                tosi = "nagoya" orelse tosi = "kyoto" orelse tosi = "osaka"
                orelse tosi = "kitakyusyu" then true
                else false;

        fun nittou(katagaki) =
                if katagaki = "syukan" then 3500
                else if katagaki = "syuseki" then 3000
                else 2500;

        fun syuppatubiNittou(katagaki, syuppatu) =
                if syuppatu > 1200 then nittou(katagaki) div 2
                else nittou(katagaki);

        fun toutyakubiNittou(katagaki, toutyaku) =
                if toutyaku < 1200 then nittou(katagaki) div 2
                else nittou(katagaki);

end;
(*
syuppatubiNittou("syukan", 1159); syuppatubiNittou("syukan", 1201);
>val it = 3500 : int
>val it = 1750 : int

syuppatubiNittou("syuseki", 1159); syuppatubiNittou("syuseki", 1201);
>val it = 3000 : int
>val it = 1500 : int

syuppatubiNittou("syain", 1159); syuppatubiNittou("syain", 1201);
```

```
>val it = 2500 : int
>val it = 1250 : int

toutyakubiNittou("syukan", 1159); toutyakubiNittou("syukan", 1201);
>val it = 1750 : int
>val it = 3500 : int

toutyakubiNittou("syuseki", 1159); toutyakubiNittou("syuseki", 1201);
>val it = 1500 : int
>val it = 3000 : int

toutyakubiNittou("syain", 1159); toutyakubiNittou("syain", 1201);
>val it = 1250 : int
>val it = 2500 : int

syukuhakuhi("syukan");
>val it = 9000 : int

kiteiSyukuhakuhi("syukan", "yokohama", true);
kiteiSyukuhakuhi("syukan", "kyoto", true);
>val it = 10000 : int>val it = 10000 : int

kiteiSyukuhakuhi("syukan", "nagano", true);
kiteiSyukuhakuhi("syuseki", "nagano", true);
>val it = 9000 : int
>val it = 8500 : int

[kiteiSyukuhakuhi("syukan","yokohama",true),
kiteiSyukuhakuhi("syukan","nagano",true),
kiteiSyukuhakuhi("syukan","yokohama",false)];
>val it = [10000,9000,4500] : int list

nittou("syukan"); nittou("syuseki"); nittou("syain");
>val it = 3500 : int
>val it = 3000 : int
>val it = 2500 : int
*)

fun sum[] = 0
  | sum (n::ns) = n + (sum ns);
(*
sum([1,2,3,4,5,6,7,8,9,10]);
>val it = 55 : int
*)

fun syuttyouhi (s, k, n, z) =
      sum(s) + sum(k) + sum(n) + sum(z);
(*
syuttyouhi([10000,1000,11000], [12000,400,12000],
[3500,3500,3500], [120,240,100]);
>val it = 57360 : int

syuttyouhi([kiteiSyukuhakuhi("syukan","yokohama",true),
            kiteiSyukuhakuhi("syukan","nagano",true),
            kiteiSyukuhakuhi("syukan","yokohama",false)],
            [10000,2000,11000],
            [syuppatubiNittou("syukan",1159),
            nittou("syukan"),
            toutyakubiNittou("syukan", 1159)],
```

```
        [120,240,0]);
>val it = 55610 : int

syuttyouhi([kiteiSyukuhakuhi("syukan","yokohama",true),
        kiteiSyukuhakuhi("syukan","nagano",true),
        kiteiSyukuhakuhi("syukan","yokohama",false)],
        [10000,2000,11000],
        [syuppatubiNittou("syukan",1201),
        nittou("syukan"),
        toutyakubiNittou("syukan", 1159)],
        [120,240,0]);
>val it = 53860 : int
*)
```

5.. まとめ

出張費の精算といった事務処理問題の分析モデルの仕様記述には，関数型言語 ML をベースとした仕様記述はとりあえず使えそうである．例題中に示した日本語の「曖昧な仕様」よりは随分とましであろう．ただ，分析モデルの仕様記述でありながら，リスト構造といったデータ構造を決めなければならないあたりが，やむを得ないとはいえ煩わしい．そのかわり，書いた仕様が実行可能であるという御利益がある．

ここでは述べなかったが，ML は，functor という structure から structure へのマッピングと，signature と structure とを用いて，オブジェクト指向の「継承」をうまく表すこともできる．したがって，SA だけでなく OOA の仕様記述にも向いている．また，ML は手続き型プログラミングもできるので，構造化設計やオブジェクト指向設計といった，How を記述する必要がある仕様記述言語としても流用できる．

半面，例題にも必要だったが，日付モジュールやその他の事務処理に必要なライブラリーが揃っていないため，すぐにこの分野に適用するのは難しい．

なお，この文章とプログラムは 4 時間ほどで作り，ML の経験自体も教科書を読みはじめてから 2 カ月程なので，考慮が足りない面もあるだろうがお許しいただきたい．

6. 参考文献

L.C. Paulson. ML for the Working Programmer. Cambridge University Press, 1991.

7. ML の入手方法

ここで使用した Standard ML of New Jersey の処理系は，以下のインターネットの FTP サイトから入手できる．

PRINCETON.EDU (128.112.128.1)
RESEARCH.ATT.COM (192.20.225.2)

ログイン名は anonymous で，あなたのメールアドレスをパスワードに入れる．ディレクトリーは pub/ml (princeton.edu の場合) または]dist/ml (research.att.com の場合) である．f tp は binary mode ("binary") にする．これ以上の情報は，README と release-notes.ps ファイルを参照して欲しい．

SEAのリストラクチャリングについて

(その1)

大場 充
(日本IBM)

IBMの大場です．今年1月に米国から帰国して，今年度，新しく幹事を引受ける羽目になりました．シンマイで，右も左も分かりません．ご指導のほど，よろしくお願いします．

とはいうものの，何せ海外帰国子女，ではなくて海外帰国サラリーマン，なものですから，いいたいことは文脈自由に何でも素直に口に出します．ご気分を害されることもあるとは思いますが，他意はありません．そんな時は，大目に見て下さい．それでも，バイリンガルの友だち(外人さん)にいわせますと，「あなたは，日本語で話す時の方が，ずっと謙虚(ていねい)だ」そうです．これは，英語でいわれました．私の英語は，よっぽど「ぶしつけ」かつ，日本語の時よりももっと「率直」なのでしょう．ただ，みなさんからのご批判も，素直に拝聴するつもりでおりますので，悪しからず．

米国に滞在中は，ちょうど，「米国大企業のダウンサイジング＆リストラクチャリング」の真っ最中でした．私の勤務していたIBM社も例外ではなく，帰国時には，着任時の4分の3の大きさになっていました．この点については，みなさんも先刻お聞き及びのことと思います．しかし，外から傍観しているのと，その中にあって実際に体験しているのとは，自ずから違いがあります．よい勉強をさせていただいたと感謝しています．はっきりいって，価値観が変わってしまいました．

リストラの神様は，私がお気に入りのご様子で，帰国して再び，今度は日本で「リストラクチャリング」を経験することになりました．大変ありがたいことです．アメリカ人から，「日本ではこんなことは経験できないだろう」と，冗談にいわれていたことを，帰国早々経験できるとは，夢にも思いませんでした．私自身の回りでも，「人員整理」の波は，ヒタヒタと音もなく，日々少しずつ近づいてきているようです．「そろそろ潮時か...」と思う毎日です．

よく考えてみると，これがふつうなのかもしれません．昨日までが異常だったともいえるでしょう．玉井先生が4月のテクニカル･マネジメント･ワークショップでいわれたように，「ソフトウェア業のバブル」もはじけたのです．いいかえれば，われわれの業界も「スリム化」しなければならなくなっているのではないでしょうか．だとすれば，SEAも当然，「スリム」にならなくてはいけません．

世の中のトレンドを見ると，「高付加価」「サービス」「学際的」がキーワードのようです．世界一高い人件費のかかる日本人が，アメリカ人やヨーロッパ人でもできるソフトウェアづくりをしているのは，ほとんど罪悪です．われわれは，かれらがどうがんばってもできないものを作らなければ，このさき，生きてゆけなくなります．数年来のリストラクチャリングで，アメリカから工場労働者や工場の技術者･管理者がいなくなったように，日本から「今までのようなソフトウェア技術者」がいなくなるのは，そう遠い日の話ではないでしょう．

「新しいソフトウェア技術者」像とは，どんなものでしょう．「高付加価」「サービス」「学際的」がキーワードだとすれば，それは高度に専門化された技術を持ったプロフェショナルといったところでしょうか．ここで，「技術」といっているのは，大学に行けば勉強できるようなものではありません．大学の講義から勉強できるのであれば，それは付加価値を生みません．ここでいう「技術」とは，問題解決のノウハウといったところでしょうか．これは，だれも教えてくれません．自分自身で，自分の経験から，積極的に「学ぶ」ことが必要です．

そのような「新しいソフトウェア技術者」のための「ソフトウェア技術者協会」とは，どのようなものなのでしょう．まず，ソフトウェア技術者の自己開発を助けるものであるべきでしょう．新しい技術の評価や，新しいものの見方･考え方，についての議論の場

です．つぎに，自分の専門とは異なる分野の専門家から，何かを学ぶことを助けるものであるべきでしょう．自分に経験のないアプリケーションの専門家や，ほかの固有技術の専門家との意見交換の場です．最後に，技術者が直面している問題の解決を助けてくれるものであるべきでしょう．技術者が自分にない経験をもつ技術者を探し，相談を受けるための場です．

これらをもう少し具体的にいうと，「シンポジウム」などの会議の主催と後援，「セミナー」や「チュートリアル」の開催，そして「ギルド」の形成ということになります．「シンポジウム」や「セミナー」などは，SEA ではもうすでにやってきています．私のここでのポイントは，その頻度と，開催場所です．これをもっと，「ユーザ･ドリブン」にしようということです．「ギルド」の形成も，SEA にはいくつか SIG(研究会) があります．「ギルド」には，研究会にはない意味が含まれています．それは，「ギルド」が技術者を「認証」する権限をもっていることです．「ギルド」の一員になることは，すなわち一人前の「親方」として「弟子」をもち，教育することが許されることです．

ニューヨーク滞在中に，ちょとしたきっかけで，そのような「ギルド」の一時的な会員になりました．早い話が，コンサルタント協会です．「ギルド」のメンバーになると，「ギルド」のパーティーに招待されます．パーティーでは，新しいメンバーが，どのような技術をもっているかを古いメンバーに説明します．ギルドのあるメンバーが，ある会社 (クライアント) と契約を結ぶ時に，自分にはない技術やスキルをギルドの他のメンバーから得ます．つまり，下請けに出すのです．大きなプロジェクトでは，そのようにギルドの複数のメンバーが共同で (一時的に) 働きます．私も，そのような仕事をしました．その時，こんなシステムが，日本にもあった方がよいだろうなあと思いました．

ギルドのメンバーには，コンサルタントを専門 (職業) としている人もいれば，私のようにある会社に勤務している技術者もいれば，ある会社でコンサルタントのような仕事をしている人もいました．仕事を探してきて，プロジェクト･チームを組織するのは，会社のコンサルタントや，独立したコンサルタント達です．プロジェクト･チームのメンバーになるのは，主として会社に勤務している技術者や独立コンサルタント達です．ギルドのメンバーには，大学の先生もいます．そんな「ソフトウェア技術者ギルド」を作ってみたらどうでしょう．雑用も増えますが，雑収入も増えます．SEA の財政もうるおいます？　ギルドのメンバーになり，プロジェクトに参加することによって，経験も多くなり，技術も向上します．

まとめます．ここでの私の提案は，今後予期される業界のリストラクチャリングに備えて，SEA の事業を次のようにリストラクチャリングすることです．

1. 「シンポジウム」「ワークショップ」の主催と後援の枠を拡大する．
2. 「セミナー」「チュートリアル」「フォーラム」の開催をユーザ主導にする．
3. 「ソフトウェア技術者ギルド」を設立して技術者の技術レベルを認定する．

次回は，このようなリストラクチャリングに備えて，SEA を「どうダウンサイジング」するかを考えてみたいと思います :-)

ちかごろ，SIGENV がらみの日常

渡邊 雄一
(アスキー)

1. はじめに

SEAMAIL の岸田編集長から，環境分科会 (SIGENV) での話題を書くようにとの依頼がありました．でも，SIGENV の活動状況は，昨年の岡崎の報告書の SEAMAIL (Vol.7, No.4-7, pp.257-258) に書いているので，同じような内容の文章をくりかえし書いてもヒンシュクを買うだけでしょう．

岸田さんからは「なるべくテクニカルな内容のものを！」とのリクエストですが，それはもう少し待ってもらわないとちょっと辛い..... で，さんざん悩んだあげく，この数ヶ月の間での SIGENV に関連した私事を日記風に記してみることにしました．

他の支部や分科会と同様に SIGENV でも，メンバ間の連絡の手段としてメイリング･リストを活用しはじめています．

UNIX の電子メイル (以下単にメイル) のメカニズムに疎い方のために簡単に説明しておくと，電子メイルの宛先はふつう，xxx@yyy.zzz の形になっています．これは，zzz 社の yyy というマシンに xxx という名前でアカウントを持っている人へのメイルです．メイリング･リストの場合も，字面上はそれとまったく同じように，xxx@yyy.zzz 宛にメイルを送るのですが，実はその名前 xxx が個人のアカウントではなく，実際にメイルを送付すべき人たちの名前 (e-mail address) が登録してあるリストの名前になっていて，届いたメイルは，自動的にそこに登録されている人たち全員に転送されるという仕掛けです．

FAX の同報通信を思っていただければ話が早いでしょう．しかし，FAX の場合とちがって，ここでは，個々のマシンにそうした設定が必要なわけではなく，zzz 社の yyy というコンピュータにだけリスト xxx を用意しておけば，みんなは，ただ xxx@yyy.zzz にメイルを送ればよい．そうすれば，目的の人たち全員に同文のメイル (のコピー) が届けられるようになっているわけです．

さて SIGENV のメイリング･リストは，分科会メンバの個々の利用環境に差があるので，常に積極的なトラフィックがあるわけではありません．が，WIDE/junet と NIFTY とのメイルの送受信実験サービスが始まったので，例会の 1 週間前後くらいには最低でも 4 ～ 5 通の連絡を兼ねたメイルが流れるようになってきました．

SIGENV では昨年来，

> Timothy A. Budd, "An Introduction to Object Oriented Programming", Addison-Wesley, 1991 (ISBN 0-201-54709-0)

を読んでいる (いた) のですが，昨年の秋から冬にかけては，上記の若手の会の報告書のとりまとめで忙しく，その後はメンバ個々人の年末/年始/年度末/年度始めの忙しさも重って，勉強会がおろそかになっていました．

それぞれの忙しさもなんとか峠を越したようなので，ぼちぼち活動を起動に乗せるように気合いを入れ直したのが 4 月からなのですが....

2. 前途多難の日々の始まり？

4 月 21 日 (水)

[PM 1:00] 会社の仕事で見事にハマってしまった．う～ん，今日の ENV の宿題で自分の担当分がぜんぜんできていない．どうしよう．これから南青山までネットワーク作業で出かけなければならないし，そこでの作業が 7 時まではかかるだろうし..... とりあえずハマっているという状況だけはみんなに知らせておかないと..... ENV のメイリング･リストに，いいわけを入れておこう．

[PM 5:00] ぼちぼち南青山に出かけないと作業開始に間に合わないなぁ．あれ，メイルが来たぞ．田中さん@SRA からだ．げっ，はまっているのはかまわわないが，必ず ENV の席には顔を出せと塩谷さん@SRA がいっているって！

[PM 8:30] なんとか仕事を切り上げて四谷での

ミーティングに出席．今日は，久々に福良さん@JMAS桜井さん@池上通信機と再会．ごめんなさい，ちっとも宿題やってなくて．

3. 今回はちょっと気合いを入れて....

5月19日(水)

[AM 9:30] 会社についてloginしてメイルを読む．あれ，桜井さんからメイルが来てるぞ〜．えっ，今日のENVは忙しいから出席できないのか〜，残念．

[PM 6:10] 先月ほどひどくはないけれど，今月も大した準備ができてないなぁ〜．でもいいや，できたところまでレジメをメイルで送っちゃえ〜．終業時間もとっくに過ぎたし…，ぼちぼち出かけるかな．

[PM 7:00] 今日は塩谷さんも出張が入ったということで，田中さんと福良さんの3人だけ．最近，後藤君@シスプランから連絡はないけれど元気してるかなぁ？

この日はBuddの本の4.1 Message-passing syntaxを読んだのですが記載されている内容をその前の章や付録の例にいちいち確認していると，記述上の矛盾と思えるような箇所が2〜3見つかり，思いのほか進みませんでした．

4. 後始末もまた楽しからずや

5月21日(金)

[PM 3:00] やっと仕事がひと段落したので，自宅のパソコンで書いてきた一昨日のメモをup_loadして，ざっとチェックする．さしあたって大きな問題はなさそうなので，ENVのメイリング･リストに送ってみんなにチェックをお願いする．そういえば昔(3年位前)は，毎回ENVの議事録をコンスタントに作っていたっけ．久々だなぁ〜．

[PM 5:00] 田中さんから早速チェックの入ったreplyが届く．ふむふむ．う〜ん，でもちょっと忙しいから後回し．

[PM 8:50] さて，ぼちぼち仕事も片付いたし，メイルの整理でもするか．おっと，田中さんからreplyを読まなきゃ．たしかに指摘されていることにうなずくばかり．えっ，これ以外に言語の本を並行して読まないとダメかどうかって？．そうだ，福良さんに以前に教えてもらって積んである本：

> Masini ほか, "Object Oriented Languages", Academic Press, 1991 (ISBN 0-12-477390-7)

は，たしか言語の本だよなあぁ．ちょっとこれで問題のObject Pascalを調べてみるか．ちっとも詳しく出ていないぞ〜．う〜ん，これ以上の深入りは危険な気がするなぁ〜．ここだけreplyしよっと．

[PM 9:10] さてっと，そろそろ..... あれ，メイルが来たぞ．おっ，久々に塩谷さんからだ．やっぱり勉強するならSmalltalkとC++ですか，ふむふむ．Objective-Cはよいと以前から塩谷さん，いっていたなぁ．えっ，Object Pascalは日本人で使っている人がいるだろうか？だって．なんとなくそんな気もするなぁ．Pascalで作られたシステムが自分の回りで動いているというのを聞いたことがないし，それのObject指向版なんて論外なのかな？

5. さすがアメリカは進んでいるなぁ

5月26日(水)

電子ニュースのcomp.objectをながめていたらBudd先生がpostしてるのを見つける．

何々，いま読んでいる本の講義用のOHPを頒けてくれるって！．どうもここに書いてあるアドレスにお願いすればよいらしいなぁ〜．どうしよう．とりあえずENVのみんなに，このニュースをforward しておこう．

5月28日(金)

[AM 9:30] 会社についてloginしてメイルを読む．あれ，福良さんからメイルだ．おっ，5/21のメイルのreplyだ．前回の補足として，Smalltalkの演算子の優先順位(キーワードメッセージ<2項メッセージ<単項メッセージ)の具体例を上げてくれている．

6月2日(水)

田中さんから，5月26日に僕がforwardしたおいてBudd先生のpostの件でreplyが来る．話のネタになれば面白いから，OHPを入手してみようってか？暇なときにでも恐る恐るメイルをしてみようかなぁ．

6月3日(木)

[PM 6:58] 舌足らずの英語でオレゴンにメイルを書いてみる．

[PM 7:00] あれ，もうreplyが来たぞ．僕の英文が解釈できないって！？．おおっ，これはauto replyのシステムになっていて，特定のstatementをmail bodyに書いて送らなければいけないのね．下手な英語の作文で悩むんじゃなかった．納得して，今度はカタログを要求する旨のメイルを送る．

[PM 7:06] おっと，これもあっという間に返って来る．早い！．えっ，正誤表があるって！．何々「間違いを見つけたら，直接Budd先生にメイルを出すこと」だって．ふむふむ「ページと行番号，それに可能だったら訂正内容を送ること」と書いてあるぞ．それに「まずこの正誤表をチェックしてから重複がないか調べてからエラーレポートを送って欲しい」と書いてあるなぁ～．よし，正誤表をもらっちゃおう～．

[PM 7:10] おかしい．待てど暮らせど返事が来ないぞ～！．僕のメイルが原因でWIDEの国際リンクを切るようなトラフィックを出しているなんてないだろうなぁ～，心配だ！

[PM 8:30] やっと正誤表が届く．こんな量(ヘッダ部も含めて全部で270行程)で国際リンクを落したこともなかろう．でも，それでも予想以上にあるなぁ．あれっこれは，前回のENVで散々悩んだところじゃないか～．お～，みんなで考えた通りの内容に修正されているぞ．すでに訂正が入っているのはちょっぴり残念だけれど，うれしいなぁ～，自分たちの議論していたことが正しくて！．とりあえずENVのメイリング･リストにforwordしよう．今日は疲れたので，OHPはrequestだけ送っておいて，ここらへんで切り上げて帰ろう．

6月7日(月)

[AM 9:30] loginしてメイルを読み出す．あれ，福良さんからメイルだ．やっぱり，あの正誤表には驚かされたようだなぁ～．そういえば，オレゴンからメイルが戻ってこないぞ．どうしたんだろう？．

[PM 8:40] やっと仕事がかたづいたので，あらためてオレゴン宛のメイルをチェックする．なさけない，アドレスを間違えている！．もう一度OHPのrequestメイルを送ってみると，1分ほどで1350行ほどのpostscriptファイルが2つのメイルに分けて送られてくる．さっそく1つのファイルにしてプリンタへ出力してみる．内容は，もちろん本の通りだが，何がポイントなのか箇条書きになっているのがありがたい．

6. おわりに

以上，

SIGENV
- → Budd先生の本の輪読会
- → オブジェクト指向への関心
- → ニュース･グループcomp.objectの存在を知った
- → Budd先生の投稿を見つける
- → Budd先生の本に関する種々の情報の入手

と，SIGENVの活動を継続していなければめぐりあえなかったことの不思議を，そして，ネットワークの威力に驚き，痛いほどその恩恵も感じられたいう，お粗末な一席でした．まあ，SIGENVではこんな調子で活動を続けています．もうちょっとペースを上げて，年内にはこのBuddの本は切りをつけたいと思っていますが，果たしてどうなりますか？．

行方の気になる人は，是非ともSIGENVの例会に顔を出してみて下さい．今年度の例会は，毎月第3水曜日PM 7:00から四谷のSRA小会議室で開催の予定です．

なお，今後の例会でのトピックス的な内容は，機会と気力があったら報告するように心がけてゆきますので，今回はこれでお許しを!!!

フツーの OL_SE からのお便り

大塚 理恵
(菱信システム)

こんにちは，大塚です．１年振りですがお元気でしたか？　今年度もお世話になります（昨年の投稿は上司も黙認してくれたのですが，今回も見逃して！）．

さて相変わらず普通の OL_SE を続けているので病院にでも入院しない限り，あるいは結婚でもしない限り話せるトピックは発生しないので，ここ１年の SEA について，また仕事について書くことにします（入院はしませんが SEA によるストレスで風邪をひきやすくなり，困ってます．ゴホゴホ．まだ修行が足りない？）．

まず幹事会を通して知り得た SEA とは．思っていたより年齢層が幅広く，20 代，30 代なんて若い若い．30 代で少しワガママいえて（君ってまだ若い，頑張ってね，という感じ），40 代でようやく一人前（大目にみるよ），そしてまだまだ頑張る 50 代で主役，60 代で悠々穏やか（いろいろ相談させていただきたいな！）というところでしょうか．分科会，支部会等でも，みなさんのワガママや頑固さに，またなんてアクティブなの！　と驚かされることもしばしばですが，一瞬，ああこの人たちって家庭ではいいお父さんなのね，と感じることがあり，ほのぼのしてしまいます（優しい娘心！）．

ところでディベートについて調べていたら，次のような引用を見つけました．

－ 吉川英治は，『宮本武蔵』をこうしめくくる．

波騒は世の常である．
泳ぎ上手な雑魚は，歌い，踊る．
けれど，誰が知ろう百尺下の海の下を．
海の深さを．
波は術である．海は道である．

著者はそこで日本人の「ディベート道」を説いていたのですが，とするとソフトウェア技「術」者協会＝SEA　は道に通じるものか？（私は溺れて百尺下に迷いこんだのかも．SEA の mermaid にも会えたし．）

次に昨年 ('92) 秋までにやっていたことの内容ですが，AIX CASE について述べます（私には仕事外の仕事となったものです．固有名詞が出過ぎでしょうか？）．

昨年日本アイ･ビー･エムのガイドシェアで，EWS についてのチームに参加し，AIX CASE の製品であるソフトウェア開発環境 SDE Workbench/6000 の評価を試みました（当時は RS/6000 で動いてました）．SEA フォーラムやソフトウェア･シンポジウム等で PCTE や統合ソフトウェア開発環境について聞く機会が多く，研究テーマが SDE に決定したときは内心ご機嫌でした（しかし社内では EWS にさわれる立場でないので，一度も触れぬまま，もっぱら事務局）．

結果的には，とりあえず動かしてみるというだけに終わり，評価というよころまでには至りませんでしたが，ユーザー･インターフェイスの操作性が統一されていること，SDE へ組込むアプリケーションの構築が容易であることなどが，メンバーには評判がよかったようです．内容は SIGNET で紹介させていただきましたが，発表をお願いしたメンバーに，隣にいらしたのが S?A 社の S さんであることを教えてくれなかったのはいじわる？

アプリケーションの事例研究など面白いでしょうし，研究テーマは継続されることを望みしたが，私たちの SDE チームは１年で解散．評価版とはいえ日本初のユーザーとしては残念．その後発表された分散処理もしくは分散開発環境に人気をとられたとも思えませんが，SDE の行方を知りませんか？　そして統合予定と表明されたあのソフトたちは，いまどこに....

最近は，ある大規模事務システムの開発プロジェクトにもぐりこみ，開発プロセスや標準化，CASE 適用について，一緒に検討しつつ，業務の分析も行うというカオス体験中（カオスのままではいけないのだが，いろいろむずかしくって....）．

今年の「若手の会」参加者探しのためもありますが，若手エンジニアの発掘！を行ってます．職場（社外）の若者に「ねぇねぇ，こんなイベント企画しているのだけど来てみませんか？」と勧誘したり，お世話になったソフトウェア･メーカーの部長様に（恐れ多くも）ご案内させていただいたり，話しかける口実にも事欠かず楽しんでいるようです．しかしなかなかよい返事がいただけず，勧誘のノウハウを知りたいところです（主婦の家庭用品販売に学ぼうか？）．

コンピュータもストレスを感じる?!

中来田 秀樹
(ネクスト ファウンデーション)

「コンピュータもストレスを感じます」などと書くと,「機械がそんなもの感じるわけがない！」と怒られるかもしれませんが，私の回りのコンピュータは，ほとんどがストレスを感じて，なんらかの症状が出てきています.「ほとんど」と書いたのは，私が見た限りではたった1台だけ，おそらく使い方のせいでしょうが，ストレス症状に陥っていないマシンがありました．もしかしたら，症状が出るほどに賢くはない(!)のかもしれません.

この問題は，まだまだ，研究段階でして，論文として発表できる所まで到達するには，もう少し時間が必要なのですが，あまり1人で研究していると，自分までストレスを感じてきそうなので，SEAMAIL 編集部から原稿依頼が来たのを幸いに，一部を発表させてもらうことにしました.

前提として，ストレスとは，歪みがたまることを指し，そのストレスがある一定以上たまったときに，なんらかの症状があらわれてくるるものと仮定します.

では，コンピュータがストレスを感じたときに出てくる症状を分類して，リストアップしてみましょう.

肩こり型ストレス：

「正常な状態と較べてなぜかおかしい！」というように，雰囲気的にしか，外からは分からない症状．しかし，そのま放置しておくと，次の胃潰瘍型に発展する可能性もある.

頭痛型ストレス：

これは，常時ストレスを感じるのではなく，特定の場合にのみ，発作的に軽い症状があらわれる.

胃潰瘍型ストレス：

これはかなり重傷で，悪い部分を取り除くか，現状とことなる環境に持って行き，ゆっくり療養をさせないと，社会復帰が見込めない.

自己破滅型ストレス：

もう，ここまでストレスが進んでいる場合は，ほとんどの場合，手の打ちようがありません．こちらはただ，祈るばかりです.

心臓発作型ストレス：

いきなり死んでくれるので，ストレスの原因追求もままなりません．運がよければ，死ぬ直前にメッセージを残してくれますが，運が悪いと，メッセージどころか，体も破壊してしまう場合があります.

以上にそれぞれの症状は列挙しましたので，こんどは，各症状ことに，そうしたストレスが発生する原因が判明しているものをあげてみたいと思います.

肩こり型ストレス：

- たとえば親（つまり私たち）が貧乏なために，メモリを沢山買えず，コンピュータが実メモリだけでは動けなくて，スワップを多発しているというのが，ストレスの原因です．動きがガクガクして，見てる方も肩がこってきそう....
- これがMacの場合だと，もともとスワップが遅すぎるので，ストレスにならないっていう話もあります.

頭痛型ストレス

- これはUnixマシンでは，newsがexpireした時とか，巨大なシステムをmakeしている最中に感じます．別にメモリー･スワップをしているわけではないのだが，自分の処理能力の上限で稼働している場合によく起こり，動きが一瞬止まってしまう時があります．頭の使い過ぎですね．やっぱり，無理をさせてはいけないですよ....
- Macの場合だと，アプリケーションを複数起動した場合に，アプリケーション同士で喧嘩をはじめて，いずれかのアプリケーションだけが異常終了

する場合が，これにあたります．これは，アプリケーションの起動時によく起こるので，再起動すれば動く可能性があります．

胃潰瘍型ストレス

- 古いディスク（主に５インチ･フルハイト）に対してアクセスをしていると，機種によっては，胃が痛くなるような音が発生する場合があります．音が出ているってことは，どこか無理があるのですから，そうしたストレスが蓄積されて行くと，いずれは心臓発作で倒れるか，自己破壊して死ぬことになります．

自己破滅型ストレス

- Mac の場合，リソースを食い尽くすようなアプリケーションがあると，それが自分の首を締め，最終的には，システムにまで影響し，突然すべてが停止してしまいます．
- Unix 等の場合，デバイス･ドライバの作りが悪いと，いきなりエラー･メッセージが画面に表示され，何もできなくなる場合がこれにあたります．自己破壊ですから，再起動させた場合，fsck で救済できない場合は最悪の死に方と判断します．対処としては，バックアップという消極的な方法が最善の策だといわれています．

心臓発作型ストレス

- 長時間の稼働という疲労から，ディスプレイ部分からがブン！といって画面表示が消えてしまう症状．
- 同じく長時間の疲労から，電源と呼ばれる患部からから煙が上がって CPU が止まってしまう症状．

以上のようなストレスを，私の周りのコンピュータたちは感じているようです．

しかし，冒頭でも書きましたが，まったくストレスの感じない馬鹿な（いいすぎかな？）コンピュータもあります．一応，その名前だけははっきりさせておかないといけないでしょう．

そのマシンの名前は PC-9801 DA といい，MS-DOS が動いています．私が判断するに，この場合，コンピュータのほうはストレスを感じていませんが，それを使う人間に対して，他のどのマシンよりも強いストレスを感じさせているように思います．つまり，PC-9801 と MS-DOS の組み合わせは，人間のストレスをエネルギーとして動くコンピュータだと考えられます．

まだまだ研究が足りないのですが，SEA の会員の方で，どこかでストレス症状に陥っているコンピュータを見かけたときは，naka@next.co.jp まで e-mail で報告していただけないでしょうか？

教育工学の理想と現実

君島 浩
(富士通)

NIFTY につながった WIDE から，編集長の執筆依頼が来た．幹事会メンバ交代を期にして，近況をお書きなさいとのことである．私が関係している分野は，教育分科会，静岡県，ソフトウェア工学 OB，メインフレーム･メーカである．教育分科会は教育以外の分野や経営トップに転身した OB が増えたが，私はいまだに教材を作っている現役である．最近は，教育システム開発方法論 (ISD) のジョブ分析技法の関係で，生産管理やプロセス･プログラミングなど，ソフトウェア工学との関係も再び強くなった．

1. 構造的教育方法論 (職人芸･素人仕事から工学へ)

私は生来，研究しては人を教えるのが大好きである．下手くそな人を見ると，自分だけはうまくやろうと思わないで，つい教えたくなる．7 年前にソフトウェア工学推進担当から教育部門に配置転換になっても，性癖は直らなかった．管理職は自分で講師をやる機会が少ない．すると研究･教育の対象は，部下である教育要員そのものになる．教育のやり方が下手くそで見ていられなくなるのである．

そういう時に米国出張があって，教育システム開発方法論 ISD の存在を知った．1970 年代前半に，構造的プログラミング技法を知って，導入した時のことを思い出させる．しかし，ISD は構造的プログラミング技法と違って，さっぱり広まらない．理想と現実にはギャップがある．

2. ISD の普及活動

ISD を知って以来，どんな活動をしたかを，詳細に書き出して見よう．御用とお急ぎのない人は見ていただきたい．

1992 年
- 2/2: 教育会社 TTI 社と ISD 訪米調査団打合せ
- 2/12: SEA 仙台セミナー「教育工学」
- 2/14: 社内教育事業部から 1979 年に導入済みの ISD のその後を取材
- 2/20-21: SRC セミナー「ソフトウェア技術者の教育要員育成技法」
- 3/6: (株) 関西 PFU 講演「教育システム工学」
- 3/9: 全日空整備士訓練所見学
- 3/16: (株)PFU 講演「教育システム工学」
- 3/26: 電子情報通信学会春季大会発表
- 4/16: 松下電器品川ショールーム見学
- 4/22: SIGEDU で日本航空客室乗務員訓練所見学
- 4/25: SEA 札幌セミナー「教育工学」
- 5/29: 日本能率協会に売り込み
- 6/3: (株) 富山富士通講演「教育システム工学」
- 6/4: 富山大学教育学部情報教育コース山西助教授 (当時) 訪問
- 6/10: ソフトウェア･シンポジウム「教育パネル」
- 6/23: 日本能率協会横浜事業所主催セミナー「教育システム工学」
- 7/9: 部内 ISD 教育
- 7/31:「企業内教育システムハンドブック」出版
- － 昨年の SEAMAIL での幹事挨拶
- 8/26-27: SRC セミナー「ソフトウェア技術者の教育要員育成技法」
- 9/17: SRC セミナー「UNIX 技術者教育の事例」
- 9/22: SEA 東京セミナー「教育工学」
- 9/25: SEA 大阪セミナー「教育工学」
- 10 月: 日本能率協会「人材教育」に投稿掲載
- 10/8: 社内 ISD 教育
- 10/15: 産能大小林薫教授に電話相談
- 10/20: NTT ソフト (株) に講演の押売
- 10/22-24: SEA 教育ワークショップ
- 10/29: CAIT 企画調査課佐藤課長と教育エンジニア構想について情報交換
- 11/4: 産能大に売り込み
- 11/5: CSK 研修所見学
- 11/12: SEA 九州支部月例会「教育工学」
- 11/13: (株) 富士通九州システムエンジニアリング講演「教育システム工学」
- 12/2: 産能大へ売り込み
- 12/9: PHP 研究所へ売り込み

1993 年
- 1/28: SIGEDU フォーラム
- 2/9: (株) 富士通インターナショナルエンジニアリングと情報交換
- 2/15: 社内 ISD 教育
- 2/16: 産業構造審議会教育エンジニア育成カリキュラム部会に参加開始
- 3/9: 電子情報通信学会教育工学研究会発表
- 3/10: 株) 両備システムズ講演「教育システム工学」
- 3/11-12: AIT 主催教育工学活用セミナー
- 3/13: トスギ塾 (沼津市の学習塾) と教育工学勉強会を開始
- 3/22: (株) 富士通関西通信システム講演「教育システム工学」
- 3/29: 部内 ISD 教育
- 3/15: 社内 ISD 教育
- 4/21: JSTD コンファレンス聴講 (小林薫教授と名刺交換)
- 5/12: 日本産業訓練協会に売り込み
- 5/28: 産業労働調査所に売り込み
- 5/29-30: 私学教育研究所教育情報工学研究会で NEC 佐藤氏の指導を受けた．
- 6 月: 共立出版「bit」に投稿掲載
- 6/7: 職業能力開発大学校に売り込み
- － 業務マニュアル製作会社 DOM 訪問．情報交換

今後の予定
- － 産能大「JSTD ジャーナル」に投稿掲載予定

- 産能大講演会講師
- 日産訓「産業訓練」に投稿掲載予定
- 産労調「企業と人材」に投稿掲載予定
- 労働省「職業能力開発ジャーナル」に投稿掲載予定
- 日経ビジネス投稿交渉中
- 日科技連より「社内教育」出版予定
- ダイヤモンド社出版提案中
- 静岡大学教育学部で計算機言語教育論の講義(審議中)
- 沖電気の藤岡氏に会って酒を飲む.
- 中央職業能力開発協会に売り込み.

7/1-2: SRC セミナー「ソフトウェア技術者の教育要員育成技法」
7/6: (株) オムロンソフトウェアで「教育体制の構築法」懇談会
7/16: SEA 関西セミナー「UNIX 技術者の教育」
7/29-31: 私学教育研究所構造学習サマー・セミナーを受講. NEC 佐藤氏が指導.
8/25: SRC セミナー「UNIX 技術者の教育」
9/17: NTT ソフトウェア公開講演「情報教育と教育工学の動向」
10/21-22: 職業能力開発研究発表講演会発表 (職業能力開発大学校主催)
10/28-30: SIGEDU 教育ワークショップ

これらの活動は, 依頼があって動いたものもあるが, 大半はこちらから売り込んだものである. 我ながら図々しい. 講演や投稿という段階までは行くのだが, だれかが ISD を採用したいという段階まで行かない. こちらの期待しているのは, 一つには ISD を一緒に推進する講師格の仲間が出てくることと, 「ISD 導入の事例」というのが他社から論文発表されることである. 今のところそういう反応はない. ISD について講師をやれる日本人は, 日本 IBM の鎌田氏と私の 2 人しかいないらしい. 日本 DEC や日本 ITU(国際通信連合)は米国から講師を呼んでいる.

百軒回って一つ売れればよいという気持ちで, 普及活動をしている. 商品の営業というのもこんなものだろう. 「企業内教育システムハンドブック」の販売量は 800 冊を越えたところである. 残念ながら構造的プログラミング技法や製品マニュアルのテクニカル・ライティングのようにワーッと盛り上がることにはならない. なぜ ISD は普及しないのだろうか.

3. なぜ ISD は広まらないか?

構造的プログラミング技法や製品マニュアル制作技法と違って, ISD が広まらないのには次のような原因があると思われる.

(1) マニュアルやソフトウェア障害については顧客が苦情をいってくれた. 教材は社員が相手なので苦情をいわれない.

(2) マニュアルを理解できないと機械を操作できないが, 教室で教育内容を理解できなくても, 実務までは間がある.

(3) 教材を理解できなくても, 講義で講師が補ってくれるし, 質問することができる.

(4) 教材が理解できなくても, 実務ではマニュアルを読んだり, 先輩が OJT をしてくれたりしてカバーできる.

(5) 品質管理がされていない. 障害分析の結果, 数割が顧客の使用法誤解であり, その原因はマニュアルの理解しにくさである, という統計的分析があって, マニュアル制作の工学化が進んだ. 教材の理解しにくさを批判する品質管理部門はない. 教育の最後に試験をしても, その回答状況を分析していない. 誤答があれば, 生徒が悪かったのだと考えて, 教材に障害があったという考え方はしない.

(6) 教材開発者の絶対数が少ない. もともとソフトウェア産業の教育要員は少ないので事務作業で忙殺される. マニュアルは専門ライタが書くか, ソフトウェア技術者が片手間で書くかは別にして, とにかく工数をかけて書かざるを得ない.

(7) 教材開発者が学会やセミナーに縁が薄い. ソフトウェア技術者には学会活動に関心の高い人がいるが, 事務作業の多い教育要員は技術や論文と縁が薄い.

(8) マニュアルには, 「理科系の作文技術」や「なぜマニュアルは分かりにくいか」などの参考書があり, 木下是雄先生という有名な啓蒙者が存在した. 教材設計には, 坂元昂先生や NEC の佐藤隆博氏などの権威がいて, 本も出版しているが, 木下先生のほどポピュラーではなく, かつ学校教材の設計が中心である. 社内教育の教材設計には啓蒙者, 参考書が乏しい.

(9) 年功序列賃金なので, 教材が悪いために成績が伸びなくても, 給料が上がらないわけではない. 情報処理技術者の資格によって給料が上がるとなると, 教科書にも文句の一つも付けるようになるのだが. 教材の作りが業績に直結するのは学習塾や予備校や私立進学校であり, 実際彼らは日本教育工学会などで教材設計の研究をしている.

教育要員の数といえば, 通産省の「新情報革命を支える人材像」の情報処理技術者の需要予測には, 教育エンジニアは 1995 年には 3.2 万人, 2.5% となっている. 絶対数はともかく比率は実績から算出した根拠のあるものだろう. 私の事業所では約 2000 人の社員に対して教育要員は 20 人だから, 1% しかいない. ソフトウェア産業の平均が 2.5% とはとても思えない. 教育に力を入れているといわれる CSK でも, 約 8000 人の社員に対して, 社内教育に 200 人の要員を置いているとは思えない. この 2.5% という数字を水戸黄門の印籠代わりにして, 教育要員を増員したらどうだろうか.

4. ISD による教材開発の近況

私自身の本業は，社内教材開発である．過去の作品の一つは「研究発表講座」である．研究の発案から，研究実施，論文執筆，OHP フォイル作成，リハーサル，本番までを，ISD で分析･設計してマニュアル的に書いた．作業期間や発表時間配分の見積り式まで入っている．論文の書き方には澤田昭夫氏の優れた市販本「論文の書き方」がある．私のは「OHP フォイルを運ぶ時には，ビニール･ファイルにはさむとよい」,「前夜は深酒，夜更かしをするな」とまで書いてあるから，更に懇切丁寧である．

現在開発中なのはプログラミング入門 (仮題) である．教育分科会の月例会に参加した人にはご披露している．入門とはいっても，C 言語の解説ではなく，生産管理とアルゴリズムを骨子とする学術的な入門書である．高度であって，しかも理解しやすく，実務スキルに結びつくことを目指している．アルゴリズムの話題を ISD によって作業工程に分解し，設計/コーディング/テストという作業手順の形にした．プログラミングとアルゴリズムとソフトウェア工学を同時に学べるというのがミソである．

次に開発する教材は，プロジェクト管理か構成設計になるであろう．

5. ISD の補強

実際に ISD を使って教材を開発してみると，ISD だけでは足りない部分が出てくる．これらの部分をノウハウ化して，ISD を補強するのが今後の課題である．教育工学の世界的権威の一人である NEC の佐藤隆博主席研究員の門をたたいて，教育工学のノウハウを勉強している．こういうことには，会社間の垣根は関係ない．

第 1 の補強点は，知識構造による教材構造の設計である．これは ISD の方法論と重複するので，構成設計･詳細設計のあたりの比較的ミクロな部分にだけ適用したい．

第 2 の補強点は，構造的テキスト･デザインである．作業の構造，知識の構造に加えて，紙面の構造にこだわって，教材をデザインする技法である．製品マニュアルでも研究されているが，教材独特のノウハウを盛り込みたい．

第 3 の補強点は，試験問題の実施による教材の反省である．これには佐藤氏の S-P 表の技法が使える．大勢の受講者の正誤状況を見て，教材にもまずい点があったのではないかと，問題一つ一つ個別に分析するのである．ソフトウェアの個別品質管理 (根本原因分析) にも相当する技法である．

第 4 の補強点は，作業指向分析の ISD に対して，オブジェクト指向分析または製品技術 (product technology) 分析を加味することである．現在の ISD では，作業手順と関係のないアルゴリズム講座や部品講座の設計を導き出すことができない．作業手順に現れない静的な教育項目を，だれでも系統的に洗い出せるようにするのが，この第四の補強である．

第 5 の補強点は，国際通信連合 ITU 版の ISD ノウハウの取り込みである．これは非常に複雑過ぎて実用化しづらいという評価を一部から受けているが，それなりの優れた点もあるはずである．

第 6 の補強点は，全社的教育体系の企画･改善の部分である．ISD は単一講座の開発に重点を置いているので，全社的な教育体系の設計の部分が弱い．これはイー･エス･イー総研の江村潤朗氏 (元日本 IBM) が CAIT で教えてこられた方法論を勉強して補強することになる．

第 7 の補強点は，製品マニュアル制作技法からのノウハウ導入である．こちらは実践の歴史が古く，研究者も多いので，一日の長がある．

もう一つの研究テーマは，教育と経営と生産管理を結ぶ学際研究である．こうなると私が以前にソフトウェア工学を担当していたことが生きる．プロセス･プログラミングをはじめとする生産管理 (作業管理) は教育と密接な関係ができつつある．

こういった研究･実践は仲間で競争･分担してやると進む．ぜひ，ISD の導入と改善について，論文発表･投稿をしてほしい．実際に教材を開発･保守している若手･中堅の教育担当者の人が教育分科会に参加して活躍してほしい．私は 7 年前に教育工学に開眼して，現在のレベルに到達した．昨年，幹事の挨拶を SEAMAIL に載せた時から，1 年間でまた伸びたと我ながら思う．もしも若い人が初めから工学的な教育方法論を勉強･実践したら，どんなに伸びるかと思う．と，教育分科会 (SIGEDU) の宣伝をしたところで，幹事としての挨拶･近況報告を終わる．

21世紀における技術者像

— Computer Engineer in the Next Century As Intelligent Meister —

熊谷 章
(PFU 上海)

本稿では，21世紀における技術者像を，「21世紀という時代」と「技術者とは何か？」という2つの観点から考察し，論及することにしたい．制約条件としては，一応，コンピュータに関係する文化と文明だけを対象範囲とする．

筆者の基本的態度は次のように表明できよう．すなわち，コンピュータで代表される電子媒体は，われわれ人類のデスクトップとして最良のものであり，ここ1000余年以上使い慣れてきた紙と鉛筆のメタファに取って代わるものである．このコンピュータというメタファが，今後どのような形で文化や文明に係わってくるか？が重要な要因になる．そのことの本質を把握するためには，連綿と続いている時代の流れを汲み取ることが必要だ．21世紀は，感性·芸術，設計，東洋的·高度情報化といったキーワードで象徴される社会になるだろうと予測される．その中で，コンピュータは，マルチメディア処理，超高速ネットワークの構築と運用，さまざまなシステムの設計，人工現実感等の応用に用いられるだろう．いわば，現代が味もそっけもない乾いた工業社会だとすれば，来るべき21世紀には，人間復興の新たなルネッサンスが展開される．それは，ウェットな形でわれわれの感性に直接作用し，われわれの生き方そのものを主テーマとするようになる．簡単にいえば，価値観の大きな変化が起こるものと予想される．

この変化にともなって，コンピュータ技術者も変化しなければならない．その変化は，工場における部品から一個の人間として生まれ変わり，各個人のレベルで，全体と個の関係を意識しながら，作品を創造するということであろう．そのとき必要になってくるのは，単なるテクノロジーではなく，それを超えた価値観あるいはセンスのようなものである．いわば，テクノロジーは必要条件の一つであって，十分条件がその新しい意識にあたる．昔から，アーキテクトという言葉があるが，それは，「アーキ」という原理を表わす言葉と，「テクト」という技を表わす言葉の合成語である．ものごとを設計する仕事は，先に述べたように，対象の原理と技の両方を体得したアーキテクトでなければできないのだ．21世紀の技術者にとっては，そうしたアーキテクトになることが，まさに最大の目的になるだろうと信ずる．

1. 21世紀とコンピュータ

コンピュータは，つねに時代の要請にもとづづいて変化し，成長を続けてきた．それにともなって，コンピュータ技術者の主テーマも変遷し，技術者自身も変化してきた．試みに，ここ当分のコンピュータを中心とした変化を考えてみよう．それは，たとえば次のような形に要約できるだろう．

昨日 (Yesterday)

代表的なマシン： 汎用機，ミニコン，PC
典型的なアプリケーション： 数値計算，システム制御，事務処理，電子メール
時代の風潮： 工業社会，生産性，品質，信頼性，西洋的

今日 (Today)

代表的なマシン： WS，PC，スーパーコンピュータ
典型的なアプリケーション： インタネット，文書処理，マルチメディア，CG，ゲーム
時代の風潮： 地球を大切に，アメニティ，グローバリズム，東洋的

明日 (Tomorrow)

代表的なマシン： マルチメディアマシン，並列マシン，超高速マシン
典型的なアプリケーション： 知的支援システム，設計，人工現実感，超高速インタネットワーク
時代の風潮： 高度情報社会，感性社会，設計，芸術，東洋的

この要約表の意味を少し鳥瞰してみよう．

昨日で代表される時代は，いわゆる工業社会である．時代の主テーマは，生産性や品質，信頼性で代表される大量生産の産業哲学であった．いまから約200年前に達成された産業革命の成果が，全世界に普及した時代である．物理学を筆頭とした科学的精神と理性が最も重要な価値観として定着し，アダム·スミスの経済理論が社会基盤になり，近代産業は次々と文明の利器を産み出した．コンピュータはこの時代の末期に考案された．大きく分けて，汎用機，ミニコン，PCといった範疇の製品が存在し，主として，

数値計算やシステム制御，事務処理，電子メールなどに使用された．時代の精神は，西洋科学と西洋哲学とであった．

一方，今日の現代はどう変化しただろうか？

時代の傾向は，工業社会から明らかな変化を見せた．地球規模のエコロジー，グローバルな思考，アメニティなどが主流となり，昨日までの工業社会が見直されるようになった．昨日までの時代が人間不在であったのに対し，今日では人間中心の考え方に移行しつつある．いや，むしろ人間を含めた動植物の生態系を，その環境を含めて考える傾向が見られる．

コンピュータ技術の分野でどんな変化があったかといえば，まず，新しい機種としてWSやスーパーコンピュータが登場し，他方ではPCが飛躍的な技術革新を通して新しい機械に変身した．それらの利用分野は，文書処理，マルチメディア情報の作成･編集･表示･処理，コンピュータ･グラフィクス，インターネットワークの制御，そして，画期的なことに，各種のゲームにまで使われている．昨日までの時代では到底考えられないような分野でコンピュータが使われるようになり，ある意味では，文化復興と呼んでもいいような傾向が現われた．

木村尚三郎氏によれば，文化とは「自分の手で耕し栽培し，その成果を生活の中で楽しむこと」だと定義できるそうである．工業社会の価値観では，効率のよさ，便利さ，安さ，速さ等が重要であったが，今日では，自らの手で栽培し，その成果をみんなと一緒に楽しむことが最も重要だというように，価値観が変化してきた．

コンピュータ技術者も例外ではない．技術開発にさいして，植物を栽培するような感覚で土地(技術基盤)を整備し，土壌(開発環境)を作り，種を蒔き肥料を与え(研究開発)，いろいろな世話(実用化･商品化)をし，最後に実りの収穫をする．この一連の作業を，一貫した考えと心を持って行い，成果物に接する必要がある．そうした考え方は工業社会では受け入れられなかった．なぜなら，大量生産のメカニズムは，文化という考えとは対極にあるからだ．

いずれにせよ，時代を支える思想も変化している．西洋風の科学･哲学を中心とすることから離れて，いくらか東洋科学･哲学寄りの考え方が注目されるようになった．

それでは，明日はどういう時代がくるだろうか？

今の時代の流れから，高度情報社会，設計中心，感性社会，芸術社会といった言葉で標榜されるような社会になるものと予想される．いわゆる物理的な国境はなくなり，各民族間でのマルチメディアによる情報交換が一般的になる．人間社会の構造は，つねに，交通の手段と交通量つまり情報の質と量によって大きく変化してきた．コンピュータの登場によって，この交通手段と交通量が革新的に変化することは，いまやほとんど自明である．その結果，人類がまだ経験したことがない新しい世界が開かれようとしている．

これからの時代における技術者の主テーマは，新しい何らかの対象物(それは当然システムを含む)を設計することである．設計という仕事では，設計者の価値観や感性が直接結果にあらわれる．逆にいえば，価値観や感性を持っていない人間には，ものごとを設計することができない．価値観や感性は，美的センスや芸術的センスと同義語であり，設計者の生きる姿勢そのものだといってもよいだろう．

つまり，これからは，技術を超えた価値観が大切になる．それに対応したコンピュータとしては，マルチメディア･マシン，並列マシン，超高速マシンなどが考えられる．それらの応用分野としては，マルチメディアを用いた表現，情報伝達，高速ネットワーク，さまざまなものの設計，バーチャル･リアリティなどがある．平たくいえば，ノウハウの表現と伝達，美しいものの表現と伝達，考え方や思想の表現と伝達，といったことが重要視されるようになる．

2. コンピュータ技術と応用分野の変化

コンピュータ利用の決め手はメタファにある，といって過言ではない．

ここでいうメタファとは，コンピュータを利用する上で利用分野にぴったりあった表現を指している．コンピュータがある特定の利用分野で利用されるには，このメタファの他に，それを支えるコンピュータ･アーキテクチャが必要だ．応用分野，メタファ，アーキテクチャの3者がそろったとき，それに合致し

メタファ	応用分野	アーキテクチャ
Fortran	数値計算	浮動小数点計算機構
Cobol/DB	事務処理	帳票発行機構，通信機構
スプレッドシート	表計算処理	パソコン
?	文書処理	ビットマップ，レーザプリンタ
Hypertext, Media	知的活動の支援	マルチメディア，ビジュアライズ機構
AI	知的活動の代行	推論，思考機構

表 1. 技術と応用分野の変化

たコンピュータが社会的に活躍する．

メタファとアーキテクチャは，正に技術そのものである．コンピュータ技術のこれまでのメタファとその応用分野を表 1 に示す．

Fortran，Cobol/DB，スプレッドシートまでは，衆目の一致するところであろう．文書処理や人工知能という応用分野においては，ピッタリしたメタファがまだ見つかっていない．したがって，この両分野では，比較的大きな努力が払われている割りには，大した成果が上がっていないのである．これに対して，最近注目されはじめているのが，知的活動支援という応用分野である．これは，人間の知的活動支援をコンピュータを用いて行おうという試みである．それに対するメタファとしては，Hypertext や Hypermedia があり，対応するアーキテクチャには，マルチメディア·マシンとビジュアライズ機構がある．

これからの新しいコンピュータの応用を考えるには，上の表に上げたメタファ·応用分野·アーキテクチャの三位一体を，われわれ技術者が考案し，作りだして行かなければならない．なかでも，よいメタファを考えだすことが，われわれにとってきわめて重要な仕事だと考えられる．特に，AI，文書処理，知的活動支援についての解を得ることが，当面の大きな課題である．

3. コンピュータ技術の変化

現在，コンピュータ利用に関して起こっている問題点の中で未解決のことがらは，次の 3 つのカテゴリに集約できるだろう．

すなわち，第 1 はノウハウの表現，第 2 はその伝達，そして第 3 が複雑なシステムを扱う方法である．前の 2 つは人工知能と認知科学の分野であり，ここ 10 年ほどさまざまな研究開発が行われたが，いまだに解決されていない．第 3 の問題は，主としてソフトウェア工学の分野で研究され，構造化プログラミング，トップダウン開発，情報隠蔽，抽象型データ，データ中心の開発技法，オブジェクト指向の方法論等々の解が，一応提案されている．したがって，未解決の問題は，いまや，ノウハウの表現と伝達に絞られる．

これを解決するアプローチとしては，次のようなものが考えられる．

- 技術的なアプローチ (いかにして作るか？)
- 科学的なアプローチ (その原理は何で，メカニズムはどうなっているか？)
- 工学的なアプローチ (いかにして量産品を産み出すか？)
- 認知科学的アプローチ(学習することとは何か？)
- 哲学的なアプローチ (知ることとは何か？)

私の直感によれば，上記の問題すなわちノウハウをうまく取り扱うためには，従来にない新しい哲学的かつ認知科学的なアプローチが必要だと思われる．それは，別のいい方をすれば，形而上学的アプローチとでも表現できよう．ベルグソンの言葉を借りるなら，従来の分析学的方法とはまったく異なる方法である．

まず，ノウハウ，つまり知ることを中心として考察したい．ものごとを知るには，次の 5 つの状態が存在する．最初は，抱卵状態 (Incubation)，次に来るのが，発見的推論状態 (Abduction)，演繹的方法状態

	発見的推論 (Abduction)	
抱卵 (Incubation)	演繹 (Deduction)	知る (Knowing)
	帰納 (Induction)	

表 2. 知ることの 5 つの状態

(Deduction), および帰納的方法状態 (Induction) の 3 つ, そして最後が, 知る状態(Knowing) である.

われわれが未知のものごとに出会ったときには, まず最初に抱卵状態に入る. つまり, 頭の中に何かの卵ができそれを温める状態になる.

しかし, 過去の経験から推して, 演繹的にそれが何であるか推論できる場合は, 既知の知識から演繹してその未知の物を知ることができる. いや, 知った積もりになることができる. また, 未知のものごとを何度も同じように経験すれば, そうした経験から帰納的に推論し, やはり, それが何であるか知った積もりになることができる.

上の 2 つの方法すなわち帰納と演繹はよく知られているが, これらのアプローチでは, 新しい発見や考案に到達することはむずかしい. なぜなら, 思考形態が常識や既に獲得された知識で固定化されているために, 柔軟な発想に飛躍することができず, 安直に手近な解にたどりついてしまうからである. つまり, 卵が十分に温められる時間がなく, 新しい雛が誕生するチャンスが失われてしまうのである.

発見的推論は, これらの方法とは決定的に異なる. 頭の中に作られた卵が温められながら, ある程度以上の時間が経過する. そうした時間の流れの中で, ある瞬間に, その未知のものごとのすべてが直観的に理解されるような事態が, 突然に起こりうるのである. 卵が孵って雛になったのだ!

この「抱卵→発見的推論」の細い糸を繋ぎ, それを維持させるものは何だろうか? われわれが世界を見るとき, 絶対に真実だといえるようなことが何もないというのは, 周知の事実である. だとすれば, 未知のものごとに出会ったとき, すぐさま帰納と演繹を用いて分かった風に解釈するよりも, 何かにこだわりを持ち, 疑問 (卵) を抱きながら, いつの日にかそれを解決しようという姿勢をとることの方が, 楽しいように感じられるし, また正しいやり方だと考えられる.

そうした姿勢こそ, 古来, 東洋の科学者および哲学者たちがとってきた姿勢であった. 少なくともそれは, われわれがここ 200 年のあいだに教えられ, 親しんできた西洋の科学あるいは哲学のアプローチではない. その実践には, ものごとに驚くこと, 楽しむこと, 興味を持つこと, そして, 自分の頭で考えることが, まず第一歩として必要である.

次に, 形而上学的方法について述べる.

一般に, 認識の方法には 2 通りある. 代表的なものが分析的方法 (Analytic Method) であり, もう 1 つが形而上学的方法 (Metaphysical Method) である.

分析的方法は, 観察者の視点によって特徴づけられる. すなわち, この方法では, 観察者は, 観察対象に対してつねに第三者であり, その対象の系から独立した系から眺めている. その際, 重要な要素は, 視点 (座標系) とシンボルとである. つまり, 現象を座標とシンボルで表現し, それを理解し解釈し操作しようというのである. 実際には, ほぼ無限に近い座標系が存在すること, シンボルは実物の射影に過ぎないこと, したがって, この 2 つを用いた理解は, つねに相対的な認識でしかありえないことが, このアプローチの特徴である.

この分析的方法は, 近代および現代の科学や工学にとって, きわめて有効で強力な道具であった. 物

理学を筆頭に，あらゆる自然科学，工学，人文科学等が，この認識方法を土台として発展してきた．しかし，最近では，それらの分野の最前線において，そうした分析的方法と演繹とを組み合わせた論理的思考の限界が指摘され，新しいパラダイムの模索がはじめられつつある．

一方，形而上学的方法は，それとはまったくく逆のアプローチである．まず，現象を当事者の視点で見る．分析的方法では，現象を外部からある種の計測装置を通して見るのに対して，この方法では，内部からビジュアルに見るのである．その際，重要なことは，座標系もシンボルも使用しないということである．直観 (Intuition) を用いて絶対的な認識を行なうことがポイントである．そんなことがはたして可能であろうか？ ,sp ,5

形而上学的方法は，次のようにして実践される．最初は，対象物に対して，何らかの共感を抱く．次に，われわれの意識を対象物のなかへ投入する．そして，自分がついには対象物そのものになりきる．いったん，そのものになりきり，完全に理解したところで，そのことの意味を他人に分かる形で表現する．その時の表現には，シンボル表現とマルチメディア表現の２通りが考えられる．

以上きわめて簡単に解説した形而上学的方法はオブジェクト指向アプローチを超える可能性の１つだと考えられる．そして，このアプローチによって，現在われわれが抱えている解決不可能な問題が解決できる可能性がある．短絡的にいえば，それは，イメージをベースとした新しいコンピュータで実現でき，われわれは対象物そのものを，自身の直観によってより正しく認識できるようになる．このアプローチに必要な機能として，ハイパー･シミュレーション，バーチャル･リアリティ，ビジュアル化技術などが挙げられる．それらの手法により，コンピュータの新しい利用が始まり，新しい技術と，それを実現する新しいコンピュータ技術者が台頭するものと予想される．

4. 21 世紀の技術者像

「国境を超え，会社を超え，技術を超える」というのが，21 世紀における技術者像にとっての必要条件だと思う．会社の閉じた社会を超えること，また，自らの民族意識を重視しながらも他民族を理解できる包容力を持つことも必要である．コンピュータ技術が必要なことはいうまでもないが，本稿でこれまで述べてきたように，技術を超えた洞察力･思考･そして魂が重要である．

端的にいえば，昨日までの技術者にとっては，ステレオタイプ（量産品）を作成することが主目標だった．今日の技術者は，プロトタイプ（模型）をいかに速く作成するかを目指して仕事をしている．明日の世界では，おそらく，アーキタイプ（芸術品のような原型）を創造することが目標になるであろう．それは，まさに時代の傾向と軌を一にしているように思われる．所詮，技術者も人間である以上，それぞれの時代の制約を受け，その時代の常識のなかでその時代の価値観を基準として生きて行くしかない．しかし，絶えず変化しているこの世界を，どのように，そしてどの方向に変化させるか，または本来あるがままの方向に変化させることができるかどうかが，その時代の人びとに課せられた課題になる．技術者･工学者･科学者･哲学者･思想家･経営者･労働者･芸術家，みんながそうした課題を背負っている．

最近の 200 年は分業と大量生産の時代であった．勝負は製品のコスト，品質それに生産量であった．したがって，一つの生産システムは閉じた系つまりトリー構造の組織形態を持っていた．それぞれの組織に所属している人びとは，自己の人間性を殺して，企業やの組織の目的のために生きてきた．企業戦士という表現はこの人びとを指していわれたものである．そこでは，所定の成果を上げるために，人間としては窒息しそうな管理体制や，ルール，あるいは懲罰制度等が導入された．

また，企業を超えた領域では，それぞれの国の民族意識が強かった．残念ながら，日本は世界からみれば異端児であり，理解不可能な国だと認識されている場合が多い．これは，ひとえにわれわれの組織が閉じた系を作り，外部との折衝を好まず，周囲のことを余り考えないで，自己中心的に生きてきたからだと思われる．そして，日本人は，生来の物真似上手の才能を生かして，品質のよい商品を安く世界市場に提供することに成功した．

しかし，これからの時代は，そうした企業組織･国

家／民族･技術といった概念をを超えて，より上位のメタクラスを作りだし，それを手掛かりに生きることが技術者の必要条件になると，私は信じる．

では十分条件は何だろうか？ それは，前節で述べた形而上学的アプローチのような方法を考案し実践できる能力だと思う．自分の頭と手と，そしてコンピュータを駆使し，必要に応じて従来になかった概念や方法論を作りだす姿勢である．それには，人生や世界に対するエネルギーが必要だ．つまり，文化的なアプローチによって技術を創り出すということである．従来の技術文明的なやり方から，文化的な技術への変化である．

5. まとめ

われわれが知的活動をするさいに，デスクトップのメタファはきわめて重要な役割を果たしている．21 世紀には，このデスクトップ･メタファがコンピュータの基礎になることがほぼ間違いないと思われる．本稿では，そうなった場合の技術テーマと，それを実現する技術者について本稿で述べた積もりである．要約すれば，技術文明の時代が終焉し，それに代わって文化中心の時代がやって来るだろう，ということだ．文化中心とは，自分の手と頭を使って，栽培というコンセプトにもとづいて物作りをし，その成果をみんなと一緒に日常生活の中で楽しむということである．

このようなスタイルで技術開発を行なうには，従来の技術専門家ではなく，技術を超えた全的人間としての技術者になる必要がある．そのためには，東洋科学や東洋哲学等に代表される形而上学的アプローチがキーになるであろう．そして，このアプローチにより，新しいコンピュータ技術とその応用分野が開拓され，新しい技術者が台頭し，新しい社会が形成されて行くものと考える．

[参照]

1. A. Kumagai,"Philosophical Issues of CASE", Oct. 6 1992, Urmuqi International CASE Symposium'92
2. A. Kumagai, "Beyond the Object Oriented Approach", Oct. 1991, Beijing International CASE Symposium '91
3. A. Bergson, "形而上学入門", 世界の名著.

SEA これからのイベント開催予定　1993年7～10月

7/6	第1回 ソフトウェア メインテナンス シンポジウム (協賛)	東京：総評会館 (参加者募集中)
7/16	技術者教育に関するセミナー & Forum	大阪： SIII 会議室 (参加者募集中)
9/1～4	第11回 夏のプログラミング ワークショップ	金沢：石川県教育会館 (参加者募集中)
10/13～15	TICS(泰安国際 CASE Symposium)'93	中国：山東省·泰安市 (論文募集中)
10/28～30	第7回 教育ワークショップ	福山市：鞆シーサイドホテル (参加者募集中)
11/25～26	第14回 ソフトウェア信頼性シンポジウム	奈良：先端技術大学院 (企画中)

****

第14回目を迎える来年の **Software Symposium'94** は

1994年6月15日 (水) ～ 17日 (金) の3日間，北海道·函館市·金森ホール

で開催の予定です.

実行委員長：杉田義明 (日本 NCD)

プログラム委員長：玉井哲雄 (筑波大学)　渡邉雄一 (アスキー)

****

入会申し込み先

〒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（TEL 03-3356-1077，FAX 03-3356-1072）

SEA 入会申込書（正会員：入会金3,000円，年会費7,000円）　93-07

氏　名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿ (ふりがな：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿)

生年月日：19＿＿＿＿年＿＿＿＿月＿＿＿＿日　　性別（男　女）血液型（A　O　B　AB）

勤務先名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

所属·役職：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

勤務先住所（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

勤務先 TEL：＿＿＿＿＿-＿＿＿＿＿-＿＿＿＿＿（内線＿＿＿＿＿）

勤務先 FAX：＿＿＿＿＿-＿＿＿＿＿-＿＿＿＿＿

自宅住所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

自宅 TEL：＿＿＿＿＿-＿＿＿＿＿-＿＿＿＿＿

資料送付先 & 連絡先（どちらかにチェック）□勤務先　□自宅

.....

SEA 入会申込書（賛助会員：年会費1口100,000円，何口でも可）　93-07

会社·団体名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

代表者氏名：＿＿＿＿＿＿＿＿＿＿＿＿＿＿ (ふりがな：＿＿＿＿＿＿＿＿＿＿＿＿＿＿)

連絡担当者：＿＿＿＿＿＿＿＿＿＿＿＿＿＿ (ふりがな：＿＿＿＿＿＿＿＿＿＿＿＿＿＿)

所属·役職：＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

住　所：（〒　　）＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿＿

TEL：＿＿＿＿＿-＿＿＿＿＿-＿＿＿＿＿（内線＿＿＿＿＿）　FAX：＿＿＿＿＿-＿＿＿＿＿-＿＿＿＿＿

申込口数：＿＿＿＿＿口

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