SEAMAIL

Newsletter from Software Engineers Association

Volume 10, Number 2-3 August, 1996

目　次

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

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，700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30数社ちかくを数えます．支部は，東京以外に，関西，横浜，長野，名古屋，九州，広島，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．１０，Ｎｏ．２－３　　１９９６年８月１５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
T: 03-3356-1077　F: 03-3356-1072　sea@@sea.or.jp
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　１，０００円（禁無断転載）

編集部から

☆

前号をお届けしてから半年以上たってしまいました.

その間何人かの会員の方から,"どうなっているの?" というお問い合わせをいただきましたが, 先日の支部･分科会案内にも書いた通り, 編集部はみなさんからの原稿をお待ちしていたのです.

☆☆

巻頭の松原さんの "プロセス論議" に関する最新動向資料は, 昨年 9 月の第 1 回 SPIN Workshop および今年 2 月の Forum で使われた OHP のコピーです. SEA/SPIN は昨年結成されて以来すでに 3 回も Workshop を開催するなど, 活発に活動しています.

☆☆☆

昨年第 9 回目を迎えた教育 (SIGEDU)Workshop ですが, プログラム委員長の河村先生の御努力で, 初めて立派な報告書が完成しました. 河村先生からは, 技術が大きく変動しつつある時代の中での情報処理技術者試験のあり方についての提言もいただきました.

☆☆☆☆

システム分析･設計についての佐原さんの論文は, 昨年秋にスタートした設計Workshop の副産物です. 企画運営スタッフだった青山先生 & 伊藤さん, 公式レポートのほうもよろしく.

☆☆☆☆☆

代表幹事の山崎さんからは, 2 人の SEA 会員が執筆/翻訳した新刊書 (雑誌の発行が遅れたので, もはや旧刊か?) についての紹介をいただきました.

☆☆☆☆☆☆

次号はなるべく夏休み中に編集作業に入りたいと考えています. みなさん, ふるって原稿をお寄せください.

☆☆☆☆☆☆☆

1996 年度総会報告

今年度 SEA 年次総会は，さる 6 月 6 日 (木), Software Symposium 96 の第 2 日夜，広島国際会議場で開催され，新年度役員の選出，年度予決算の報告·承認が行なわれました．

1. 新年度役員

新年度の幹事会メンバーは次の通りです．

代表幹事は山崎利治さんに今年もお願いします．

昨年度の幹事のうち，井川裕基 (日本電子計算)，西武進 (SES)，増井和也 (東芝アドバンストシステム)，松本理恵 (菱信システム)，盛田政敏 (さくら KCS) の各氏は，今回退任されることになりました，どうもお疲れさまでした．

代わりに，端山毅 (NTT データ通信)，藤野誠治 (富士通) のお 2 人が新しく幹事会メンバに加わりました．よろしく．

代表幹事

山崎 利治 (フリー)

常任幹事

大場 充 (広島市大)
熊谷 章 (PFU)
坂本 啓司 (オムロン)
中野 秀男 (大阪市大)
深瀬 弘恭 (IIJ)

幹事

青山 幹雄 (新潟工大)
荒木 啓二郎 (NAIST)
市川 寛 (電算)
伊藤 昌夫 (MHI エアロスペース)
菊地 俊彰 (東北電力)
君島 浩 (富士通ラーニングメディア)
窪田 芳夫 (東京電力)
酒匂 寛 (SRA)
塩谷 和範 (SRA)
篠崎 直二郎 (NEC ソフトウェア)
杉田 義明 (日本 NCD)
高橋 光裕 (電力中央研究所)
武田 淳男 (安川電機)
田中 一夫 (山一情報システム)
玉井 哲雄 (東京大)
中来田 秀樹 (ネクストファウンデーション)
中谷 多哉子 (フリー)
西武 進 (SES)
野中 哲 (アップル テクノロジー)
野村 行憲 (岩手電子計算センター)
野呂 昌満 (南山大)
端山毅 (NTT データ通信)
平尾 一浩 (IIJ 九州)
藤野誠治 (富士通)
二木 厚吉 (JAIST)
堀江 進 (NEC ソフトウェア)
松原 友夫 (Office Peopleware)

会計監事

辻 淳二 (辻システム計画事務所)
吉村 成弘 (公認会士)

事務局長

岸田 孝一 (SRA)

2. 決算および予算

(1) 1995 年度収支

支出の部		収入の部	
人件費	11,520	新入会費	454,000
事務所費	3,057,327	更新会費	4,792,000
印刷費	2,830,983	賛助会費	3,499,279
通信費	2,359,263	イベント収入	1,167,900
会議費	213,740	雑収入	686,770
支部支援費	239,400		
国際活動費	104,171		
消耗品費	231,033		
雑費	12,856		
当期収支差額	1,539,656		
合計	10,599,949	合計	10,599,949

(2) 1995 年度予算

支出の部		収入の部	
人件費	120,000	新入会費	792,000
事務所費	3,000,000	更新会費	4,800,000
印刷費	3,600,000	賛助会費	3,600,000
通信費	3,000,000	EVENT 収入	1,500,000
会議費	300,000	雑収入	360,000
支部支援費	360,000		
国際活動費	240,000		
消耗品費	240,000		
雑費	18,000		
当期収支差額	174,000		
合計	11,052,000	合計	11,052,000

昨年度は機関誌の発行が滞ったため，印刷費支出が減って黒字になりましたが，依然として緊縮財政です．

プロセスをめぐる世界の動き

― なぜプロセスの議論か ―

Office Peopleware
松原 友夫

プロダクトの質はプロセスに依存する(1)

- このことはハードウェアでははっきりしている
  - 戦後の日本の産業成長の歴史は、プロセス改善の歴史でもある。
  - プロセスを改善するのに2つのアプローチがあった。
    - 三菱は小牧にいた精密加工に熟練した職工さんを温存した。それが、超精密加工を要する巨大パラボラアンテナの油圧制御機構の制作に生かされ、世界市場を占有するまでに発展した。（この分野での住友系の発展にも何か似た事情が隠されていると思うが、私は知らない）
    - これに反して、日立は長期にわたる5,555人の首切り闘争で、腕のよい職工さんをほとんど首にしてしまったので高精度加工の能力が著しく落ちた。その後いろいろな油圧機構を作ったが、最初は油漏れ事故が多発して散々な目に会い、そこから名人よりも誰でも高精度加工ができる、設備依存へのシフトが始まった。同じように、早くから設備依存のプロセス改善をはじめた典型的な例は、自動車産業であろう。
    - おそらく、三菱はもはや名人の腕前には依存していないであろうが、名人の存在がビジネスを変えた。
    - 私が機械工場にいた頃は、それ以外にも名人依存のプロセスがたくさんあった。例えば、
      - 車両の車体などの鋼板構造の歪み取り。名人は一発のハンマーで叩いて直す。普通の職人は、叩いてダメにする。
      - 鋳型や木型作り。名人が作ったものは不良が少ない。鋳物には、まだこうした名人依存の領域が残っている。それを嫌って、今は鋳物を使うところがどんどん減っている。大型ポンプでさえ、最近はケーシングを厚板を曲げて作る。
      - こういった名人依存プロセスは、本来標準時間（ST）が設定できないので、後づけ日課（作業票）と称してST欄は空欄になっていたが、私がいるころに「無理にでもSTを入れる」努力がなされ、これがさらに設備や作業環境の改善を促進した。
  - 多層プ　リント板や半導体では文化の違いがプロセスに大きな影響を与えた。
    - 当初、これらの生産に電気屋が携わったが、彼等は針金を切った後は捨ててしまう文化で育ったので、特に後始末については何も考えないし、材料の成分についても吟味しない。このため、捨てた半田の中に大量の金が混じっていたり、とんでもない廃液を知らずに捨てていたり、といったトラブルが続出した。化学出身者が生産を担当するようになり、空気や水の質と回収に注目するようになり、ようやくプロセスは正常になった。
    - 次の場面では、作られる半導体など精密加工製品の品質の日米の大きな違いである。日本では、当初からごみを極度に嫌う環境では素足で働くのに抵抗感がないが、アメリカの同じ環境では当時はどこでも靴ばきで作業していた。当初の日米の歩止まりの違いの原因は、主にこれであった。

プロダクトの質はプロセスに依存する(2)

- 無形のプロダクトでの極端な例、音楽では、
  - 音楽の質はプロセスそのもの（プロダクトとは演奏会？CD？）
    - 日本のクラシック音楽の質を西洋のレベルまで高めた斎藤秀雄の功績：独特の方法による徹底した基礎の教育
      - 指揮法の基礎：タタキとシャクイ
      - 弦楽器奏法の基礎：音を出さない運指、耳の訓練
      - そして音楽の心
      - 興味深いことに、彼は教育者としては卓越していたが、演奏者としては必ずしもそうではなかった。完璧主義者であったため、例えば演奏が一旦乱れると混乱を増幅させることがあった。
  - だが、音楽プロセスの教育法は、一般に定性的で民族伝承のレベルである。
    - 例えば、発声法は技術よりもイメージ教育に近い。教える人によってさまざまで、受け取る人の感性に依存する。
  - プロセス（演奏成果）をどう評価するか？
    - 単純な技術評価はできるが、それで評価できるのは初歩的な段階
    - ある程度の技術レベルに達してしまえば、プロダクト（演奏成果）の評価は主観が支配する。
    - 今年、マーラーの復活の演奏会を6月13日から16日まで連続4回（長崎2回、東京2回）行なったが、当然回を重ねる毎に「うまく」なる（これが録音で聞いたときの質の改善）のだが、現場に張り詰めた緊張感があって聴衆が最も興奮したのは初日であった。
  - 無形物をつくる場合、一般に質は問われるが生産性という概念はなきに等しい。
- 著述、翻訳などの著作物では？
  - 金物の場合よりも類似点が多い。

プロダクトの質はプロセスに依存する(3)

- 無形のソフトウェアにおけるプロセス
  - プロセスの質を上げればプロダクトがよくなることはたしか。
    - それも、個人のプロセスより、プロジェクト・チーム、さらに組織全体のプロセスが影響が大きいだろう。
    - だが、さてどうやって組織のプロセスを改善したらよいやら. . .
  - 共通する悩み：ポジション・ペーパーから
    - プロセスをどう定義したらよいか：どの程度の定義が必要で何が不要か。うまく定義できない。利用者に理解してもらえない。
    - プロセス基準があっても守られない。
    - プロセス改善案が出てこない。
    - どうやって自主改善をモチベートさせるか。
    - プロセス改善の成果が見えない。
    - 失敗の原因の多くは見積りプロセスにある。
    - 失敗事例がプロセス改善に活かされない。
  - 無形のプロダクトを対象とするプロセスの問題
    - 何を作っているのかが見えない。
      - 要求仕様、規模、機能性、性能、etc. したがって見積りも不正確。
    - 有形物が対応しないので、プロセスの区切り（中間プロダクト）が主観的になる。
      - 開発者が「できた」と思えばその中間プロダクトは完了する。
    - 出来映えが直観的に見えない。
      - 問題を先送りしやすい。
      - これが99%シンドロームの原因。
    - 知的プロセスである。
      - 人の能力、気分、やる気によって影響を受ける。
      - 経験は個人の頭の中にだけ蓄積される傾向がある。
  - よく見られるプロセスの矛盾
    - 火を噴いた修羅場プロジェクトほど見かけの生産性が上がる（ppから）。
    - 腕利きのプログラマーほど見かけのプロセス品質が悪い（例えばバグ出しワースト10に入る）。
    - 「動きさえすれば」何をやっても容認されるシンドローム。
    - ソフトウェアの開発スケジュールは、納期の直前までは順調に進む。*ソフトウェアの法則*、木下 恂

プロセスとは何だろう？(1)

― 同床異夢のプロセス ―

- いろいろなソフトウェア・プロセス定義
  - 議論するグループの数だけ少しずつ異なるプロセス定義ができる。
    - SEI CMM KPA (Key Process Area)
    - SLCP (IS 12207: Software Life Cycle Process)
    - ISO 9000-3
    - ISO/IEC Process Assessment Standard (旧SPICE)
    - Bootstrap
    - etc.
  - SEI CMM: Capability Maturity Model
    - 特徴：
      - 主目的はDoD業者査定
      - レベルは5段階
        1 初期レベル
        2 反復可能なレベル
        3 定義されたレベル
        4 管理されたレベル
        5 最適化されているレベル
      - レベル毎にKPSを定義
      - 各段階にKey Prosess Areaが定義されている。
    - ほとんどの組織はレベル1 - 2
  - SLCP: IS 12207 Software Life Cycle Process 共通フレームワーク
    - 特徴：
      - 主目的は2者間取り引きの円滑化のためのプロセスの国際的共通定義の確立
      - 一義的に決めたものではなく、プロセス要素を決めてプロジェクトの特性に合わせて定義する。つまりカストマイズ可能なライフサイクル。
      - 要素プロセスの呼び込みを許す。
    - アメリカではIEEE ソフトウェア工学標準化委員会(SESC)がこれをベース規格として強くこれを推奨している。
  - ISO 9000-3
    - 特徴：
      - ISO 9001をソフトウェアに適用する際のガイドライン（ISO 9001から引用したところ以外はShallを使っていない）。品質システムのためのプロセスを定義。
  - ISO/IEC Process Assessment Standard (旧SPICE)
    - 特徴（あとで詳しく述べる）
      - SEI CMMの国際規格版
      - Base Practice (Process定義)とCommon Featureとのマトリックス

プロセスとは何だろう？(2)

― 同床異夢のプロセス ―

- 評価モデルによって少しずつ異なるプロセス定義
  - SLCP vs. プロセスアセスメントのBP(Base Practice)
    - 購入と供給は、SLCPでは別、BPでは顧客／供給者としてまとめている。
    - BPにはサブルーチンコールの考えが入っていない。
    - SLCPではプロジェクトカテゴリが細分化されていない。
    - 肝心の開発プロセスさえ互いに対応しない。
    - SLCPのSupporting viewとBPのSupportの範囲が一致していない。
    - 互いに含まれないプロセスがかなりある。（ピア・レビューはBPのみ、マイグレーションはSLCPのみ、etc.）

      （注）この比較はSC7WG10の委員の作業の成果で、これをまとめて南アフリカ会議に提出した。
  - SEI CMM vs. ISO 9000-3（互いに完全に欠けているプロセスのみ）
    - ISOにあってCMMにないプロセス
      - 4.1.2: requires the appointment of customer's representative, empowered to cooperate with the organization
      - 5.2.1: requires the existence of a procedure for contract preparation and review
      - 5.2.2: requires the precise content of the contract - above all, with regard to the quality related criteria
      - 5.4.4.5: requires defining inputs and outputs of each phase
      - 5.4.6: requires verification of each phase
      - 5.4.1: requires preparing a Quality Plan
      - 5.4.1: requires preparing a Testing Plan and Integration Plan
      - 6.2: requires precise definition of procedures for management of documentation
      - 5.10.1-6: Maintenance
      - 5.10.7: Replication, delivery, installation
    - CMMにあってISOにないプロセス
      - Level 2 KPA: Project Planning - Activity-2, 15
      - Level 2 KPA: Project Tracking and Oversight - Activity-3, 4, 5, 11
      - Level 2 KPA: Subcontract Management - Activity-6
      - Level 2 KPA: Quality Assurance - Activity-6, 8
      - Level 3 KPA: Organization Process Focus - Activity-4
      - Level 3 KPA: Process Definition - Activity-4, 5
      - Level 3 KPA: Integrated Software Management - Activity-2, 5, 6, 7, 8
      - Level 3 KPA: Software Product Engineering - Activity-10
      - Level 3 KPA: Intergroup Coordination - Activity-2, 3, 5, 6
      - Level 4 KPA: Quantitative Process Management - Activity-5, 7
      - Level 5 KPA: Defect Prevention - All Activities
      - Level 5 KPA: Technology Change Management - Activity-2, 4, 6, 8
      - Level 5 KPA: Process Change Management - All Activities

プロセスとは何だろう？(3)

― 同床異夢のプロセス ―

- 評価モデルによって少しずつ異なるプロセス定義（続）
  - あまり建設的とは思えないが、この他にも比較論文がたくさんある。
    - 例えば、BootstrapとISO 9000, CMM、Bootstrap: Fine-Tuning Process Assessment, IEEE Software, July 1994
  - つまり、これらの規格がそれぞれ少しずつ異なるプロセスを定義している。
  - トップレベルのごく粗いプロセス定義でさえこのありさま、迷惑するのはこれらを使わされる産業界である。
    - 作った人たちは、視点が違う、と言うが、それなら規格毎に異なるプロセス定義ができてしまう。
- さかのぼってプロセスの原義を調べて見ると、

> Progressive forward movement from one point to another on the way to completion: the action of passing through continuing development from a beginning to a contemplated end: the action of continuously going along through each of a succession of acts, events, or developmental stages: the action of being progressively advanced or progressively done: continued onward movement
>
> *- Webster's 3rd New International Dictionary*

> Process is particularly appropriate when progress from a definite beginning to a definite end is implied and something thereby made, produced, or changed from one thing into another; the term usually suggests a division of the entire sequence of events into steps or stages
>
> *- Webster's New Dictionary of Synonyms*

これらの定義をまとめると、プロセスとは、何かを作る、または完成させる仕事全体を、いくつかのはっきりした始まりと終わりがある一連のアクション、イベント、または開発ステージに分けた、その各々をいうようだ。

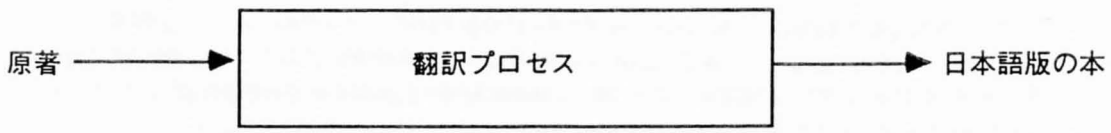
有形物のプロダクトが次第にプロセスによって姿を変えていくので共通の定義を決め易いが、無形のソフトウェアではこれは簡単でない。また、上の定義から、プロダクトとプロセスは組みとして考えるべきことを示唆している。

プロダクトとは何だろう？

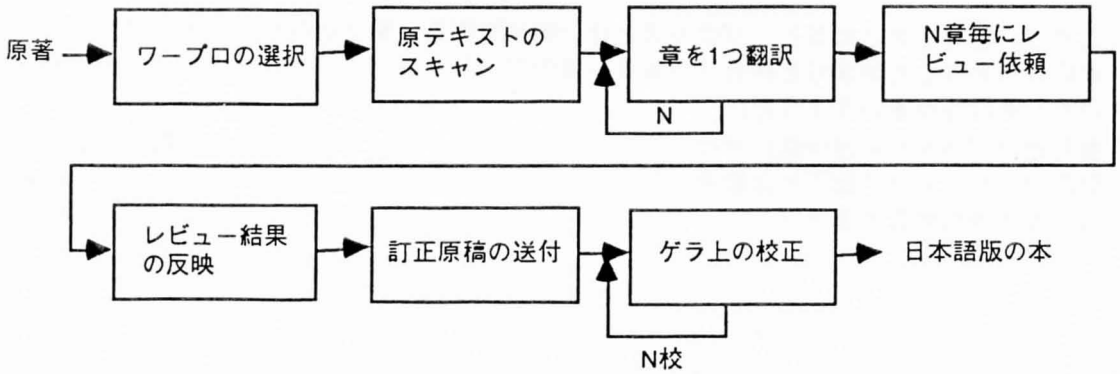
― 特に中間成果物をどう捕えるか ―

- 再び有形物では、
  - 機械部品
    - 切断した丸鋼（前のプロセスの中間プロダクト）を旋盤で削れば（プロセス）シャフトの原形（そのプロセスの結果としての中間プロダクト）ができる。
    - 見た目を通して共通の認識が得られる。
  - 化学製品
    - ある中間製品がタンクに入り、その中で反応して別の中間製品になる。つまり、タンクという物理的装置の前後でプロダクトの性質が変わる。
- 無形物では、
  - 最終成果物はプロダクトと呼べるものがあるが、中間成果物を客観的に識別するのは難しい。例えば、
  - 翻訳の例
    - 最終成果物は日本語に訳された本だが、プロセスの切れ目としてのプロダクトはは人によって状況によってまちまちである。

      マクロに書けば、



プロセス/プロダクトの一例：流れも切り方も人によってさまざま



    - 有形物と比べると、誰が見ても明らかな客観的な区切りがない。ソフトウェアでの中間プロダクトはかなり主観的な存在である。
- ソフトウェア・プロダクト評価の国際規格のプロダクトの定義

  開発プロセスの結果としてのコンピューター・プログラム、手続き、及び関連する文書とデーターで、ユーザーに提供されるためのもの。

ISO/IECソフトウェアプロセス評価(1)

- 現在の状況
  - SPICEからISO DTRへ
    - ISO/IEC JTC1のルールに則って再構成
    - 現在はWorking DocumentでDraft Technical Report にステージを上げるべくSC7内の投票にかけられている。
    - まだ、大幅な変更の可能性がある。
  - 現在の文書構成
    - 第1部：概念と構成ガイド（旧IG）
    - 第2部：プロセス管理のモデル（旧BPG） ― これにBase Practicesとして次の5つのプロセス・カテゴリー、及び能力レベルが規定されている。
      - Base Practices
        - Customer-Supplier process category (CUS)
        - Engineering process Category (ENG)
        - Project process category (PRO)
        - Support process category (SUP)
        - Organization process category (ORG)
      - 能力レベル(Capability Level) [CMMでは]
        1. Performed-Informally [Initial]
        2. Planned-and-Tracked [Repeatable]
        3. Well-Defined [Defined]
        4. Quantitatively-Controlled [Managed]
        5. Continuously-Improving [Optimizing]
    - 第3部：等級づけプロセス（旧PAG） ― 評価の段階について記述
      - 4段階評価
        - N: 不適切、P: 部分的に適切、L: かなり適切、F: 完全に適切
      - 2段階評価
        - N: 存在しない、Y: 存在する
    - 第4部：総合評価実施のためのガイド（旧PAG）
    - 第5部：評価のための仕組みとツールの構築、選択、及び利用（旧AI）
      - 主としてアセッサーが個々のプロセスの実施レベルの特性をみるためのプロセス・インディケーター及びプロセス・マネージメント・インディケーターについて記述
    - 第6部：アセッサーの資格条件と教育（旧ATQG）
    - 第7部：プロセス改善に使用するためのガイド（旧PIG）
    - 第8部：供給者のプロセス能力を決定するときに使用するためのガイド（旧PCDG）
    - 第9部：用語集（新たに追加）

ISO/IECソフトウェアプロセス評価(2)

- 日本の態度（WG10 南アフリカ国際会議報告より）
  - 日時： 1995年11月13～17日
  - 場所： Pilanesberg Kwa Maritane, South Africa
  - 参加国：16ヶ国　53名 今回ドイツなどの新参加国があり、過去最多の参加国数となった。
    - 日本からの参加者：青山、新谷、堀田、宮崎
  - 今回の会議での主要な変更点：

(1) 第2部で定義されるモデルに関する変更

以前から主張していたSLCPとの整合をとる件について、日本としては以下の２案を提案した。

1. 第2部のProcessの軸の定義を削除してSLCPを用いる。
2. 第2部を変更して、SLCPとより強く関係付ける。

日本としては、最初に1の案を主張したが、最終的に2の案で合意が得られた。この合意に基づき、第2部を中心とした再編成が新しい小グループで実施されることになり、日本としては、この作業に積極的に参加することとした。

この再構成では、SLCPとのマッピングの他に、以下のような大きな変更を行う。

- これまでNormative であった、BP、GPの定義をInformativeに変更する。
- これまでNormativeであった、Indicatorの定義をInformativeに変更する。
- これまでCapability Levelとして定義されていたものをより細かく分けてCapability とする。

なお、第2部の変更を強く主張した国は、日本の他に、米国、オランダ、イタリア、ノルウェー、イスラエル、カナダ等であった。

(2) Qualified Assesorについて

日本としては、Qualifiedという用語はCertificationを意図するように思われるため、Skilledという用語に変えることを提案したが、これは受入れられなかった。ただし、

- Qualification がアセスメントのスポンサーによって検証されること、
- Certificationを必ずしも意図しないこと、

を第9部（用語集）で明確に定義するということで合意が得られた。また、第4部では、Independent Assessment が第３者認証機関による認証を意図するように思われるため、Independent Assessmentの目的をより明確にするようにというコメントをした。これに関しては、Independent AsseessmentはProcess Capability Determinationのために必要であるという文章を追記することになった。

ISO/IECソフトウェアプロセス評価(3)

- 日本の態度（続）

(3) 第5部について

第5部については、まだ投票期間が終了していないため、正式にはコメント処理がされなかった。ただし、第5部のエディタに日本のコメントを説明し、Dispositionのドラフトを作成した。この結果、

- Indicatorのモデルをより明確にすること
- Indicatorの使用についての記述を充実させること

となった。

(4) 第3部について

日本としては、評価にNot Applicableが必要である、というコメントを出した。これについては、今のBP，GPの定義がInformativeになっても必要であると主張し、Dispositionの記録として残された。また、現在明確に記述されていないGPの目的を明確に記述すべきであるとのコメントを出した。これについては、(1) に示した内容でProcess Capabilityの見直しをすることになっているので、ペンディングとなった。

- まとめ
  - 依然としてソフトウェアの標準化テーマの中では最も関心が高い。
  - かなりの国の支持もあって、プロセスをSLCPに合わせよという日本の主張はほぼ通った。（日本は当然作業を分担することになる）
  - 認定に結び付けたくない、という点についてはWG10内にコンセンサスがある。

- ISO/IECソフトウェアプロセス評価の長所
  - 認定を目的としない（たしかに認定アレルギーが存在するようだ）。
    - ただし、供給者の査定に使うことは意図している。
  - レベル向上の道筋に自由度がある。
    - 例えば、レベル1からある程度のメトリックスを実施する、など。
  - 世界共通のものができる。

- ISO/IECソフトウェアプロセス評価の短所
  - 評価項目が多いので時間がかかる。

なぜプロセスの議論か？(1)

- 技術的な要因
  - 背景：
    - 奇妙な性質を持つソフトウェアを世界中が扱いあぐねている。 - L. Belady
  - ウォーターフォール型プロセス・モデルの否定と新たなモデルの出現
    - OOの普及と相俟って、シリアルな開発からサイクリックな開発に変化し、開発プロセスが複雑になった。
  - 統合開発環境の普及
    - 従来困難であったソフトウェア・プロセスがCASEツールに置き換わって明確になってきた。
    - CASEツールの適用範囲が開発プロセスから支援プロセス、管理プロセスへと拡がってきた。
    - 共通プラットフォーム、共通リポジトリーの利用が現実のものとなってきた。
      - PCTE
      - その他のコマーシャルde fact標準
  - ソフトウェア・メトリックスの普及
    - 主観的な（申告や推定に基づく）メトリックスから、客観的な（ツールが自動的に収集する）メトリックスに変わってきた。
  - これらを前提とした管理手法の進展
    - 構成管理はすでに必須の管理手法となった。
    - リスク管理もリスクに挑戦する重要な武器となった。
  - ソフトウェア・プロセスの研究の進展

なぜプロセスの議論か？(2)

- 社会的・政治的な要因（プロセス評価が騒がれだした背景）
  - 社会的な関心
    - ソフトウェアの責任が重くなり、ソフトウェアの欠陥が大きな損害や災害すらもたらすようになった。
    - アメリカでは、ソフトウェア技術者の資格制度の議論が始まった。
  - 産業界
    - ソフトウェアを制するものが生き残る。
    - ソフトウェアを制することができなければその国の産業全体がダメになる。L. Belady
    - BPRを支えるものはソフトウェア
    - Concurrent Engineeringを支えるものもソフトウェア
    - ビジネス・システムで優位に立つにもソフトウェア
    - 製品におけるソフトウェアの役割が増大
      - 製品の品質を左右する
      - 製品の成功失敗を左右する
      - 製品寿命を延ばすのもソフトウェア
      - 製品をシステム化するのもソフトウェア
    - 発注者の不安
      - 「ISO 9000はトランキライザーになりつつある」 - Motolora
  - ソフトウェア組織
    - 経済環境が厳しくなり、プロジェクトの失敗の余裕がなくなった。
      - 競合が激しい、または利益マージンが減っているため、失敗が致命傷になってきた。
    - 実質人件費売りの安易なビジネスからの変換を迫られている
      - 仕事量のパイが減り、利益マージンは下がり、スキルの低い技術者があまってきた。
    - 品質・生産性の改善のあゆみがのろい。
      - コンピューターや関連機器のコスト低下が著しいので、それがソフトウェア・コストを圧迫するようになった。
    - アウトソーシング
      - 日本ではまだあまり顕在化していないが、アメリカやヨーロッパでは、すでにかなりのソフトウェア開発が開発途上国に流れている。
    - 国際取り引き上の（政治的な）理由
      - 英国や米国とビジネスをする場合に要求される。
    - ビジネス上で優位に立ちたい
      - 例えば製品のベンチマーキングで

プロセス改善をどう進めたらよいか？(1)

― 私のアプローチを中心に ―

- 組織全体の動機づけ
  - はっきりした大義名分がなければ始まらない。
    - 実質的にプロダクトの成果で商売していなければ（例えば実質派遣や作業請負の場合は）、プロセス改善は意味がない。
    - 仕事が安定供給されていたり、短期的な仕事探しに狂奔している状況では、プロセス改善の動機づけはできない。
  - 組織の動機づけには状況を一目瞭然に示すチャートが有効（後で述べる）
- コミットメント
  - 技術者、第一線管理者レベルのコミットメント
    - 現状を何とか打開したいという思いとやる気のある人がかなりいなければ動き出さない。
  - 経営者のコミットメント
    - プロセス改善の必要性を理解し計画を承認し改善プロジェクトを支援する経営者がいないと、改善活動は長続きしない。
  - トップダウン・アプローチ vs ボトムアップ・アプローチ
    - Martin Thomas（トップダウン）とFrank McGarry（ボトムアップ）の論争 - IEEE Software, Top-Down vs. Bottom-Up Process Improvement, July 1994参照
    - 私は両方からのアプローチが成功を約束すると思う。
- 問題解決指向
  - ソフトウェアには多面性があるので、視点によってプロセスとプロダクトの定義が異なることを先に述べた。
    - ソフトウェアの共通プロセス定義は期待できない。
    - このため、その組織がやりやすいプロセスを決めなければならない。
  - そこで私がやってきたアプローチは問題解決のためのプロセス定義である。
    - つまり、まずその組織での解決すべき問題を、優先順位をつけて識別する。これには通常パレート図が使われるが、これを作るためにはメトリックス制度が確立していなければならない。
    - 問題には必ず何らかのプロダクトとプロセスがセットになっている。この観点からプロセスを定義すれば、関係者の理解が得られやすい。

プロセス改善をどう進めたらよいか？(2)

― 私のアプローチを中心に ―

- システマティックなアプローチ
  - 改善を永続させるメカニズムを作る。
    - これはSEI CMMではレベル5の要件であるが、私は最初から、原始的なレベルでよいから、組織に改善を促進する何らかのメカニズムを組み込む必要があると思っている。
    - 少なくとも、解決すべき問題の現状と目標との差異が誰の目にも明らかな形で示すこと、つまり、情報をタイムリーに知らせること、及び、それが問題の制御に結び付けられるようにしておくことが重要である。
  - この前提に立つと、状況判断と手段の選択が的確にできることが最も重要な教育・訓練のテーマになる。
    - 余談になるが、最近の子供の教育はこの逆で、母親は子供の世話を焼きすぎて、判断力を失わせている。結果は、学校への持ち物を忘れても、傘を忘れて濡れて帰ってきても、子供はすべて親のせいにする。必要なのは、「天気予報では雨が降ると言っていたよ」という情報だけであって、傘を揃えて置くことではない。
    - 同じことが、11月27日づけの朝日新聞朝刊に載っていた。

      「世話焼く親にこの子あり、宿泊学習をのぞいてみると」

      宿泊学習の途中、雨が降ってきた。ある子どもがぬれそぼっている。雨具を持ってこなかったのかと尋ねると、「わからない」と答えた。「自分の荷物でしょ」「だって、お母さんが入れたんだもん」小学生の場合、荷物は母親が詰めることが多いらしい。「七、八割は親がやっているようだ」と話す教師もいる．．．．
    - 職場でも手取り足取りをやっていないだろうか？
  - すべてコストに結び付ける
    - コストに結び付けると改善ドライブをかけやすい。
    - 会計原則から離れて、管理会計的な手法を用いる
  - プロセスもプロダクトも情報であることを最大限に利用する。
    - つまり、環境が捕える客観的な生でタイムリーな情報を、刻々プロセスにフィードバック可能な技術が実現しているのだから、これをプロセス改善にうまく活かさない手はない。

プロセス改善をどう進めたらよいか？(3)

― 私のアプローチを中心に ―

- 問題識別のための技法
  - Divide and Conquer
    - 問題の的を絞る。または対策しやすい程度に分割する。
    - 案外見過ごされているのは、適切な分類またはカテゴリー分けである。よく見られる悪例は、カテゴリーの不均等、及び異なる次元を一緒にしているものだ。
    - ドメイン別の問題識別も重要。
  - パレート図
    - すべての改善の基本ツール
    - 簡単な図だが、どういうわけかソフトウェア組織ではあまり見かけない。しかし、繰り返し教育していると習慣になる。80% - 20%の原則を目で確かめることができ、効果的な努力と資源の配分ができる。
    - 問題解決のためのメトリックスを促進する。
- 問題を制御可能な形で可視化する
  - 仮説の議論
    - 問題を制御するための第一歩は、問題原因の仮説の議論である。
  - 問題を表現するメトリックス
    - 次に、最ももっともらしいいくつかの仮説に対してメトリックスを考える。
  - 表現方法（チャート）の工夫
    - メトリックスは、変数だけでなく、捕捉する区分と、表現方法（つまりチャート）についても同時に考える。
    - しかも、ターゲットとの差異が一目見えるように、また、制御可能なタイミングで更新する仕組みを考える。
  - こうしたことを考えるのは管理者の役割

プロセスをめぐる世界の議論(1)

- インターネット上でのSEI CMMに関する議論
  - 昨年のSEAMAILの5月号(Vol.9, No.1)に、岸田さんが1Mバイトに及ぶアーカイブから要約編集された「プロセス成熟度モデルに関する議論」が載っている。それをさらに要約すると、
    - 賛成論
      - 時間の経過とともにどのように成熟していくべきかを極めてよく表わしている。
      - このアプローチはデミングが日本でやったのと同じものである。
      - やむなく大規模なプロジェクトの運営を強いられるが、その際CMMが助けになる。
    - 反対論
      - 組立工場化の道をたどるのか？プログラミングの楽しさは失われてしまうのでは？
      - 大切だと思うスキルや経験が軽視されている（多くの反対論に共通している）。
      - 権力の帝国を作り上げようと心の中で望んでいる管理者たちの絶好の餌である。
- ごく最近のNancy LevesonやDavid Parnasを巻き込んだ安全計画MLでの論争を紹介しよう。
  - 論争のきっかけは、11月18日のUC BerkleyのSchneidewingからの、ソフトウェア信頼性工学は、より安全なソフトウェアを作る、というメッセージ（indentされた部分）。これに安全性の権威者Nancy Levesonが噛みついた。

From leveson Sun Nov 19 03:33:02 1995

Subject: Re: Software-Safety Relationship

> 1. I recommend that we focus on the contribution that software reliability engineering (SRE) can make towards making software safeER.

Although this will keep software reliability analysts employed, it will not have much effect on safety. We know this from the history of engineering when it did not work (e.g., early aircraft), from empirical studies, and from accidents of the past.

> increasing software reliability cannot guarantee system safety but, then, no other method, like hazard analysis, can guarantee it either.

Nothing can be guaranteed. So what? We make decisions on what will be the most effective use of our resources. Some approaches will be much more effective in achieving our goals than others.

> Because software is an important sub-system in complex systems, increasing its reliability may and probably will improve system safety.

What evidence do you have of this? My fifteen years of intense study of this subject says that this statement is completely wrong. I would like to know what evidence you base this belief on.

- つまり、Nancy はソフトウェア信頼性工学などは安全性には役に立たない。信頼性が安全に寄与するというなら証拠を示せ、と反論している。

プロセスをめぐる世界の議論(2)

- 安全性論争（続）
  - これに、Norm が証拠で反論。

From: Norm Schneidewind, Wed Nov 22

Subject: Evidence of Effectiveness of SRE

The following is my response to Nancy Leveson's message about showing evidence that Software Reliability Engineering (SRE) can CONTRIBUTE to system safety:

The evidence I offer is that of the Space Shuttle flight software. There have been no Severity 1 post delivery failures (i.e., loss of life or mission) in SOFTWARE. My evidence comes from two sources:

1) the practitioners who manage the Shuttle software reliability program (Ted Keller, et al at Loral, Houston, former IBM Federal Systems Division), and

2) my own research on the Shuttle software as a consultant to Loral and NASA.

With regard to 1), I suggest that there is overwhelming evidence that the COMBINATION of a controlled and repeatable procees (the only CMM Level 5 organization in the U.S., I understand), highly formalized software control boards and review processes, rigorous inspections, reliability modelling, use of metrics as early indicators of "hot spots" in the code, fault trend analysis, IV&V, and a highly experience and dedicated staff, contributes to system safety.

Again, this process cannot GUARANTEE system safety but there is ample evidence that it CONTRIBUTES to safety if, by being safe, we mean that astronauts will not get killed and the Shuttle vehicle will not be lost. If we can use the above-named methods to identify and remove life-threating faults in the code, this makes the Shuttle system safer. With regard to 2) I have seen that it is possible to predict, using reliability modeling and metrics analysis, potential and actual reliability problems (I have studied software intensively too -- actually, more that 15 years). Now, no practitioner I know would rely on a reliability model exclusively for assessing the reliability of software; I certainly would not. But what such techniques provide, in combination with other parts of the process, is increased CONFIDENCE that the software has no faults that will cause loss of life or the mission. Another way to frame the issue is to ask the question: Is the Shuttle system safer with or without the described methodology? The track record suggests the former. Could more or different things be done, like hazard analysis? Sure, but this does not mean that SRE should be dismissed as "absolutely wrong" or as a scheme of reliability analysts to put bread on their tables!....

Respectfully,

Norm Schneidewind

- つまり、CMMレベル5の信頼性工学をしっかりやっている組織だから、スペースシャトルがソフトウェアの問題で問題をおこしていない。これが何よりの証拠だ、と反論している。

プロセスをめぐる世界の議論(3)

- 安全性論争（続）
  - そこで、Parnas大先生のお出まし。

From parnas Wed Nov 22 14:34:00 1995

Subject: Re: Evidence of Effectiveness of SRE

Norm,

I think you may be confusing anecdotes with evidence.

I have heard the same anecdotes about the success of Shuttle Software attributed not to CMM or SRE but to Mills style structured programming.

Moreover, I have heard lots of other anecdotes about Houston that do not paint such a favourable picture, Those anecdotes paint a picture of a system that is not sensitive to software quality, i.e. a system that tolerates a lot of error messages.

I have my own anecdote too, one based on a visit in which I saw some very bad practices.

None of this is evidence and it should not be called evidence.

On the other hand, years of experience have convinced me that anything that makes people take software and system design more seriously improves quality. If nothing else, there is a Hawthorne type effect.

Both the arguments that attention to software reliability is "completely wrong" and the arguments that there is "evidence that it works" leave me quite unconvinced.

Prof. David Lorge Parnas

McMaster University,

- Parnas：逸話と証拠とごちゃまぜにするな。これは証拠ではない。私の経験からは、ソフトウェアとシステムの設計を真面目にやっていれば品質はよくなるよ。それをやっていなければ、効果があるとしたらそれはホーソン効果だ。
- そこで、Norm は、これは逸話ではない、と反論。

From: Norm Schneidewind Thu Nov 23

Subject: Shuttle Response

Dave,

My information is more than anecdotal. Because I have been heavily involved in Shuttle software measurement research for the last few years, I have been privy to the details of the Shuttle development and maintenance process and I have had access to the project's documentation and measurement data....The following are just three examples of this maturity:

- Software failure frequency: （省略）
- "Escape Mechanism": （省略）
- Attitude about software measurement: （省略）

- この後、UCLAコンピューター・サイエンスのJohn Cosgroveから、中間的なポジションで反論があり、再びParnas登場。

プロセスをめぐる世界の議論(4)

- 安全性論争（続）

From parnas Fri Nov 24 1995

Subject: Re: Shuttle Response

Dear Mr. Cosgrove,

In my mind this is a discussion on the rules of evidence, not on Houston about which I have no up-to-date information and only anecdotes.

I sense a bit of circularity. The SEI ratings are Self Evaluative Information. It seems that you accept the data because of the SEI rating and the SEI rating because of the data.

There seems to be a broad assumption that SEI ratings and software quality correlate. I have seen data from Capers Jones that place that assumption in question. There are also many anecdotes and a lot of common sense that lead me to believe that organisations with low ratings can produce good software and that organisations with high ratings can produce bad software.

Perhaps we skeptics are a dying breed.

- Parnasいわく。Capers Jonesのデーターによれば、CMMのレベルと品質は相関がない。
- イギリスのUniversity of York, Professor John A McDermidから賛成論

I wholeheartedly agree. I too have seen similar data. I have also had bizarre conversations with SEI people (or people who have been brainwashed with the SEI orthodoxy). If you try to seek evidence from them that level 5 gets you better software (in one case I had a conversation where better = more secure) you get arguments that mean "its better because we define level 5 to be better and its level 5" although what they say is encoded so that you have to work hard to exrtract thew tautology....

John

- イギリスにもCMMのレベルを盲信している人がいるようだ。
- この後（Nov 25）、Nancy先生からNASAの実態についての長文のメッセージが3通。最初のメールの始めの部分だけ紹介し、あとは省略。

I agree with Dave Parnas' statements. Let me also add a few facts that I picked up while doing a study of the Space Shuttle software process for the NRC/NASA...

Nancy

- ここに引用した論争は安全性に関する議論なので、一般的なソフトウェア品質とは必ずしも直接の関連はないが、CMMの意義やソフトウェア信頼性の品質への効果について触れていて興味深いので紹介した。

プロセス改善の効用と限界

- プロセス評価モデルの効用には限界がある。
  - ISO 9000のパネルでの賛成論を要約すると、
    - 改善意識高揚のトリガとして有効
    - 実際にやるべきプロセスがきちんと定義され品質マニュアルが整備された。

    ということであろう。しかし、本当に品質が改善されたのか？
    - Parnasのメッセージにあるcapers Jonesの統計を見てみたいものだ。
  - 私の考えでは、おそらくどのプロセス評価モデルでもレベル3まではいくが、4, 5の解はこの延長にはないだろう。その理由は、
    - ハードウェアとちがって効果が直接的に現われない。
    - ソフトウェア的な要素が増えるほど文化や伝統の影響を受ける（半導体生産の例を思い出して欲しい）。
    - レベル4, 5はほとんど経験したことがない状態で評価モデルを書いている。つまり、これを規定するには、時期尚早である。
    - レベル4, 5の前提になっているのは、ハードウェア流の品質管理である。
- 恐ろしいのは、問題のすり替えである。
  - 認定を取ることが目的になってはならない。
    - 認定の取得はあくまでも方便と考えるべきである（認定を前提とした国際規格は作るべきでない。これは問題を歪めるだけだ）。
  - 悲しいことに、ソフトウェア・コミュニティーにはLemmingineeringが横行している。

    Alan Davis, "Software Lemmingineering", *IEEE Software*, Sept. 1993
- プロセス評価モデルを地図として使う。
  - あまりにも、モデルで定義されたプロセス定義にこだわるべきではない。
  - それより、真のソフトウェア品質・生産性の改善をまず考え、それをCMMなどの上にマッピングして、正しい道を歩んでいるかを確かめる、という使い方をすべきであろう。
  - 問題を次々に解決していって、気がついてみたら高いレベルに到達していた、というのが理想的なアプローチではないか。

おわり

プロセスをめぐる世界の動き

― なぜプロセスの議論か ―

補足資料

1996.2.19

松原友夫

昨年の11月30日のSPIN Workshop in 御殿場から、ソフトウェア・プロセスの標準化をめぐる情勢は変化したし、その後のネットワーク上での議論で、関心の焦点もはっきりしてきた。本来ならば、本日は、これらを反映させた資料を準備すべきであるが、本体はWorkshopでの資料をそのまま全体の話しの枠組みとして、この補足資料を追加、更新、または補足として使用する。

なぜプロセスの改善が必要か？(1)

- プロセス改善は本当に効き目があるのか？
  - 御殿場で紹介したParnasからのコメント、「プロセス改善が実際の製品品質に効果があるという仮説には疑問がある、というCaper Jonesのデーターを見たことがある」（資料pp.20）に基づいて、Capers Jonesに直接問い合わせたところ、その回答として彼から送ってきたレポートを紹介しよう。
  - SPRによる調査レポート
    "An Economic Analysis of the Software Engineering Process" Prepared for the Okulahoma City Air Logistics Center, Directorate of Aircraft Software Division (LAS) Tinker Air Force Base, Oklahoma, September 7, 1994
  - 1993年末における米国でのレベル分布
    - 参考として示されている168の組織を対象としたSEIのCMMレベル評価データー
      - レベル1：75%
      - レベル2：15%
      - レベル3： 5%
      - レベル4： 0%
      - レベル5： 0.5%（1組織）
  - LASにおけるプロセス改善の歴史
    - 過去8年間で2回評価が行なわれた。
      - 第1回：1990年、評価はレベル1
      - 第2回：1993年、評価はレベル2だが、かなりのプロセスがレベル3にランクされた。

なぜプロセスの改善が必要か？(2)

- 効果評価対象プロジェクト
  - 改善前
    - C-141 全天候着陸システム（AWLS） レベル1 45KS
    - B-1B Electrical Multiplexing System （EMUX） レベル1->2 161KS
  - 改善後
    - C-5B 航空管制システム（AFCS） レベル3 173KS
    - F110-GE-129 GE製F110 Digital Engine制御装置（DEC） レベル3-> 101KS
- グラフの見方
  - スケールは*log-log*
  - 2本の線（改善前、改善後）はSPRが集めた4,500を超えるソフトウェア・プロジェクトの実績値データーベースに基づく統計的理論値
  - C-141とB-1Bのプロットは細線、C-5BとF-110は太線と対比して見る。
  - 評価結果の見方は分かれる。
  - 肯定的（SPR, Capers Jones）
    - 8年間で$1.5 Millionのコスト低減、$13.2 Millionの保守費節減
    - ROIは8.8 : 1
  - 否定的（Parnas）
- レベルと成果の関係はかなり間接的である。
  - 火事はだらしのない家からでる。だから、整理整頓のレベルが上がれば火事の危険度は減るだろう。 ー という程度の関係

プロセスの改善の副作用

- たしかに効用はあるが、落とし穴はその副作用だ。
  - プロセス官僚主義
    - 先ず第一に、岸田さんが紹介したInternet Magazine, "Object Current, Feb. 1996.に載っているHumpheryのプロセス官僚主義がある。

      The process bureaucracy

      > Some years ago, I saw a perfect example of how a helpful process can become hopelessly bureaucratic. ...

    - 彼はこの論文の最後を「いったんスタッフのグループがプロセス改善そのものを目的として考え始めると、そのときから官僚主義的な行動が始まる。製品の品質、コスト、または納期のような顧客に直接関連する問題についてゴールを設定すれば、プロセス改善は効果的であり続けるだろう」という言葉で締めくくっている。
  - 手段が目的にすり替えられる。
    - 認定制度やレベル評価により、それ自体が目的になりやすい。
    - 日本では、デミング賞やISO 9000認定のための企業の対応などで、この傾向が顕著に見られる。
    - 数字の魔力はこの状態で起こる副作用の一つではないか？
  - 次に危険なのは、本当の品質改善努力が妨げられることである。
    - CMMの高い評価やISO 9000の認定は可能性を高めるだけで、個々の製品の品質を改善するのはもっとspecificなアプローチなのに、関心、投資、努力などのすべての面で、真面目な改善努力がスポイルされる可能性が高い。
  - 最も深刻な副作用は、創造的な仕事の軽視である。
    - 分析や設計に多く含まれる、天井の一角をにらんで沈思黙考する管理しにくい作業は、最も重要なのに、そのプロセス評価は第3者には本質的にできない。
    - Parnasもこのことを指摘している。

どのように動機づけしたらよいか？

- ここで議論すべき重要課題。だがここで一つだけ述べておきたいことがある。
- 日本のソフトウェア組織自体にプロセス改善の動機があるのだろうか？
  - おそらくソフトウェア搭載プロダクトがビジネスが中心である会社は、品質が自分の仕事に跳ね返ってくるから、動機づけはそれほど難しくないだろう。
    - いい加減なものを売ればあとでトラブルに巻き込まれる。
    - ハードウェア・プロダクトと一蓮托生
    - 販売後のトラブルで業績が悪化すれば収入にも影響が及ぶ。
  - しかし、日本のソフトウェア・インフラにおける商慣習では、多くのソフトウェア企業がプロセスを改善しても見返りが得られない。
    - 本質的にプロダクトでなくプログラミング工数を売っている。
    - つまり、工数の積み上げ見積もりに基づいて価格が叩かれた上、固定価格で契約する。
    - したがって、もともと余裕が少ない上、余計なことをすれば利益マージンが減るだけ。
    - 国際規格の普及のための機関が必要との議論があるが、これも動機づけがないためうまくいかないことは目に見えている。
    - 発注者が発注のやり方を変えない限り、事態はよくならない。
- 個人の動機づけは、「なにがやる気をなくすか」という個人の視点から議論するとよい。

結局は個人のプロセス改善

- People - CMM の構造

"The People-CMM", B. Curtis, W. Hefley, S. Miller, M. Konard, SEIより

  - レベル1 - 初期
    - 組織が一貫した開発環境を作業者に与えていない。
    - 人的資源に関する作業が官僚的になされている。
    - 管理者は作業者の扱いについて教育されていない。
    - 作業者はビジネスの目的について動機づけされていない。
    - 組織的な教育がなされていないため作業者のスキルが向上しない。
  - レベル2- 反復可能
    - 作業者の扱いに関する方針が示されている。
    - 第一線の管理者に作業者のスキル向上の責任がある。
    - 基本的な作業者関連の活動を組織化する。
      - 作業環境
      - コミュニケーション
      - 要員編成
      - 行動の管理
      - 教育・訓練
      - 見返り
  - レベル3 - 定義された
    - 知識及びスキルの分析
    - 要員計画
    - 能力開発
    - キャリア開発
    - 能力に応じた作業割り当て
    - 全員参加の文化
  - レベル4 - 管理された
    - 個人的指導者制度
    - チームの形成
    - チームワークによる作業
    - 組織全体の能力管理
    - 組織的な実施結果に基づく調整
  - レベル5 - 最適化されている
    - 個人の能力開発
    - コーチ制度
    - 継続的な作業者チームの能力改善

T. Matsubara

組織に変化をもたらすには

- これもここで討論すべき問題であろう。
- 私が最近観察したこと。　—　討論の材料として
  - 長い年月にほとんど変化が起こらない組織もある。
    - 幹部が目先の個々の引き合い受注案件の処理に追われている。
    - 長はワンマン、幹部は変化を起こすように育てられていない。
    - 変化を起こす計画は滅多に作られないが、計画があっても誰が責任をもって個々のアクション項目を実施するかが明確でない。
    - 計画を作り結果をフォローしつつ更新するのは誰かがわからない。
    - そのうちにジリ貧になって余裕のある資源割り当てができなくなった。
  - 変化が起こすには総合的な組織力がものを言う。
    - 人（上も下も）
    - 金
    - リスクを負える余裕
    - 戦略的思考
    - 組織的な制御力
    - etc.

どういう標準/ルールを設定し従うべきか

- プロセス改善はみんなが共通の標準/ルールにしたがうことが求められる。
  - プロセスのエントロピーを下げなければ製品の品質改善には結びつかない。
  - しかし、現実には成果が検証されない頭の中だけで考えた標準/ルールがたくさんまかり通っている。
    - モデルは検証される必要がない、という誤解さえある。
- まとめとして、確実に成果が得られる標準/ルールを作る際の心構えについてのDavid ParnasとNancy Levesonのコメントを引用する。

From parnas@triose.crl.McMaster.CA Mon Jan 29 12:33:01 1996
Date: Sun, 28 Jan 96 21:34:04 EST
From: parnas@triose.crl.McMaster.CA (Dave Parnas)
Subject: Re: Models for Software Safety Standards
To: LAWRENCE@addvax.llnl.gov, leveson@cs.washington.EDU

Nancy has said something very wise. I particularly liked the following:

"Computer scientists certainly should not be spending their time trying to reinvent engineering.

Standards and codes of practice are ways to pass down accumulated experience, not ways to propose untested hypotheses. Engineering standards that are not based on accumulated experience and in depth knowledge of the engineering domain (in this case, industrial and system safety engineering) will simply be dangerous. "

Dave

標準とルールは蓄積された経験を伝える方法であって、テストされていない仮説を提案する方法ではない。蓄積された経験や十分な知識に裏付けされていない技術標準は、明らかに危険である。

以上

T. Matsubara

第９回ＳＩＧＥＤＵワークショップの報告

ＳＥＡＳＩＧＥＤＵ主催
平成７年11月16日～18日

第９回ＳＥＡ教育ワークショップの概要

プログラム委員長
河村　一樹

１．はじめに

教育分科会（SIGEDU）では，毎年１回秋期にワークショップを開催している．第１回の八ヶ岳麓での会合を皮切りに，今回まで８回実施されている．ちなみに，開催地は，八ヶ岳・泉郷（第１回），山形・天童（第２回），熊本（第３回），韓国・ソウル（第４回），金沢・片山津（第５回），浜松（第６回），福山・鞆の浦（第７回），松山・奥道後（第８回）となっている．

今回のワークショップのメインテーマは，「教育を事業として成功させることができるか？－技術革新と人材育成の現状と改善案の検討－」である．このメインテーマをもとに，募集要項に次のような前書きが記載されている．

「技術革新の激しいコンピュータ業界では，新しい技術が次々と編み出され，ソフトウェア技術者は，いつも何かを吸収しつつ業務に励んでいます．すなわち，第一線で活躍する技術者たちの技術や意識を絶えず向上させるためには，質の高い教育が何よりも大切になります．

そこで，ＳＥＡの教育分科会（SIGEDU）では，過去８回のワークショップを振り返り，また６年後の21世紀をにらみ，『今ソフトウェア技術者に必要な教育とは何か？』『教育で何ができるか？』を絡めて，集中討論をするためのワークショップを開催します．

『ソフトウェアは人となり』とよくいわれますが，古くて新しい問題である教育について，秋の夜長を雪国の越後湯沢でじっくり語り合いたいと思います．

プログラム委員会では，できるだけ幅広くいろいろな立場の方々の参加を希望しています．日頃，ソフトウェア技術者の育成を考えておられる方の参加をお待ちしております．」

ワークショップの日程・場所・参加状況については，次の通りである．

【日程】　1995年11月16日（木）15:00～11月18日（土）13:00

【場所】　雄大な自然の里　越後湯沢　ナスパニューオオタニ

【参加状況】　定員25名のところ，13名（男性10名，女性3名）参加

以上のように，場所的にはスキー場が近く（ホテルの裏がすぐゲレンデ），温泉（それだけでなく，温

水プールまで）もあり，おいしい地酒が揃っているという好条件のところであった．また，ホテルそのものがニューオオタニの系列であることから，大変きれいで気持ちのよい宿泊施設であった．

次からは，３日間のワークショップの概要について説明する．

２．全体的なスケジュール

ここでは，３日間のスケジュールをもとに，ワークショップの概要について記載する．

【第１日（11/16）】

15:00～16:30　ガイダンス（進行係：篠崎）
　　参加者自己紹介

16:30～18:00　基調講演（進行係：篠崎）
　　青山幹雄氏（新潟工科大学　情報電子学科）
　　「技術変革期での教育のあり方」

18:00～21:00　懇親会

【第２日（11/17）】

9:00～12:00　第１セッション（進行係：牧野）
　　河村一樹（尚美学園短期大学　情報ｺﾐｭﾆｹｰｼｮﾝ学科）
　　「技術移転『手続き指向からオブジェクト指向へ』のための教育」

13:00～15:30　第２セッション（進行係：杉田）
　　君島浩氏（富士通ラーニングメディア）
　　「教育工学この一年」

19:30～22:00　オフレコセッション（進行係：中園）
　　「？…？」

【第３日（11/18）】

9:00～11:30　第３セッション（進行係：橋本）
　　平山順子氏（山一情報システム　オープンシステム第１部）
　　「中高年に対するＰＣ教育の現状」

11:30～12:00　総括セッション（進行係：橋本）
　　君島浩氏（富士通ラーニングメディア）
　　「本ワークショップのまとめ」

３．ワークショップの概要

ここでは，３日間のワークショップの概要（とその実態）について，進行順にまとめる．

３．１　初日

最初に，実行委員長である篠崎さんから，挨拶や本ワークショップのスケジュールについて説明があった．それから，恒例のＯＨＰを用いた自己紹介が行われた．それぞれ初めての人もいるわけで，多少緊張した雰囲気の中で進んだように思われる．

次に，ワークショップのオープニングを飾る基調講演として，青山さんにより「産学におけるソフトウェア技術者教育のあり方」についてというテーマで，１時間ほど講演が行われた．ただし，講演の間にも自由に質疑応答をはさむことができたため，多少時間が伸びることになった．詳細については，講演で用いたＯＨＰシートと質疑応答の状況を後ろに添付するので，参考にして頂きたい．

午後６時近くにようやく終了し，さっそく「どこだかの間」（宴会場）に全員で向かった．実行委員長の篠崎さんから，連絡事項が伝えられた後，宴会が始まった．今回は，私のゼミナールの学生２名（とりあえずギャル達）を同伴していたわけで，ＳＩＧＥＤＵとしてはいつもとは違った（より華やいだ）雰囲気になった．

しかし，残念だったのは，青山さんがどうしても都合がつかず，７時過ぎの新幹線に乗らなければならなかったことである．これ（酔った勢い）を機会に，いろいろとお話をしようと思っていたわけだが，そうはならなくなったのである．皆で酔ったせいもあり，青山さんを，盛大に（？）見送りさせていただいた．

その後も宴会は，順調に進んだ．越後湯沢という場所柄のせいか，地元の旨い日本酒があり，たちまちのうちに一升瓶は空になってしまった．宴会料理もなかなか大したものであり，次から次へときれいな器に盛られた料理が運ばれてきた．女中さんも礼儀正しく，なかなかの新潟美人だった（はずだ）．

そうこうしているうちに，宴会時間が過ぎてしまい，実行委員長の部屋で２次会となった．これまた，めずらしく全員が集まった．例年は，（私だけでなく！）数名が，どこか夜の街に繰り出すのだが．．．やはり，学生諸君のおかげかなあ．そこでは，「おじさん」の定義について話題になった．「おじさん」とは，一体何をもってそう呼ばれるのか．ジュンジュン（平山さんのあだ名）から「おじさん」年齢当てクイズが出されたり，「川辺おじさん」が幸せそうな顔のまま寝てしまったり，「和田おじさん」がわけのわからない奇怪な言葉を発したり，「杉田おじさん」が足の裏を布団から出したまま寝てしまったり，それはもう完全な酔っぱらい集団と化してしまったのである．

３．２　中日

２日目の午前は第１セッションとして，（めずらしく二日酔いにならなかった）私から「技術移転『手続き指向からオブジェクト指向へ』のための教育」というテーマで，１時間ほど講演を行った．その後，質疑応答が続いた．これについては，ポジションペーパーと質疑応答の状況を後ろに添付するので，参考にして頂きたい．

お昼は，昨夜の宴会場の隣の店で頂いた．昼食にしては，大変豪華な内容である．それを立証する写真があるので，次に載せておく．これは，君島さんが撮影したものである．

昼休みの時間に，ちょっと温泉につかった．広くて気持ちがいい．これだから，温泉場でワークショップをやりたくなるわけだ．なお，このとき，元気な中年おじさん「牧野おじさん」と「和田おじさん」は，（自分のものではないが）私の学生と一緒に温泉プールに入ったそうである．なんと元気なことか．さらには，プールに入っている姿を，遠くでうらやましそうにながめていたＸＸおじさんとＹＹおじさんがいたというのも驚き．

２日目の午後は第２セッションとして，君島さんにより「教育工学この一年」というテーマで，１時間ほど講演が行われた．これについても，講演で用いたＯＨＰシートと質疑応答の状況を後ろに添付するので，参考にして頂きたい．

夕方になり日が沈んだ頃，再び夕食の時間を迎えることになった．昨日と同じ宴会場で酔っぱらい集団となるべく，仕込みが始まった．ただし，今日の場合は，夕食後そのままオフレコセッションになるので，あまりむちゃくちゃに飲まないようと思いつつ，でも飲んでしまったが．．．

オフレコセッションでは，何が話題になったのか，全然覚えていない．結局，飲み会の延長のようなもので，まさしくオフレコそのものになった．ただ，途中で部屋の電気を消してワイワイ楽しんでいた時に，たまたま女中さんが入ってきてしまった．多分，異様な集団と思ったであろう．

２次会では，明日が和田さんの第？回目（ダブル二十．．．）の誕生日ということで，皆でお祝いをした．確か，あの有名なミッキーマウスの誕生日と一緒とかで．また，このとき，来年度以降のＳＩＧＥＤＵワークショップの運用についても話が出た．来年で第10回目を迎えるにあたり，そろそろ実行委員長（篠崎さん）とプログラム委員長（私だったのだ）を交代しようという提案である．考えてみれば，篠崎さんと私で結構続けていたような気がする．実際には，私はほとんど何もせずに，篠崎さんにおんぶにだっこだったが．ということで，第10回開催を機に，新しい人にバトンタッチをしようということになった．橋本さん，和田さん，がんばってください．

３．３　最終日

３日目の午前は第３セッションとして，平山さんから「中高年に対するＰＣ教育の現状」というテーマで，１時間ほど講演を行った．その後，質疑応答が続いた．これについては，ポジションペーパーと質疑応答の状況を後ろに添付するので，参考にして頂きたい．

それから，君島さんから３日間を通じての総括が発表された．第１セッションに関しては，生産工学と情報学および教育学の関連付けが重要であることなどが指摘された．また，教育と実務の関係は，次のようになっている点を明らかにした．

作業←ｽｷﾙ／ﾂｰﾙ←教材／教授設計

実務←実務教育←基礎教育

第２セッションに関しては，伝統対オブジェクト指向の関係において，変わらぬための批判（～が問題なのでやらない）とか説得のための説得（オブジェクト指向のここが良い）といったことではなく，「まずはオブジェクト指向にしよう」という姿勢を持つ必要があることなどが指摘された．

第３セッションに関しては，幹部の入力・作業・出力をワークフローやワークベンチシステムに切り替えることを指摘した．これによって，いままでのコピーや文書作りやお茶くみなどが不要になり，幹部自身が行うことになる．また，エクゼクティブに対する教育に関しては，その特性（積極性，恥じない，わからないことを精密に自覚，コンパクトな説明を好む）を把握しておく必要があることなどが指摘された．

こうして，本ワークショップが無事に終了した．ホテルでの豪華な昼食を再び食べてから，解散となった．全員で，越後湯沢駅まで送迎して頂いた．ここでちょっと気付いたことだが，この駅にはポンシュ館という店というか，会館があった．新潟の多くの地酒を扱っている所であり，その脇には五百円で１杯の利き酒ができるコーナーがあった．糸魚川の地酒「男山」（北海道の銘柄ではないのでご注意を）もあった．新幹線の時間に余裕があればここで一杯と思ったが，残念ながらそこまでの時間がなかった．

ということで，温泉とお酒につかりきった３日間が終わり，ポンシュ館の利き酒を思い浮かべながら車上の人となった．

参加者一覧

今回のワークショップに参加された各人の一覧を，以下に記載する．

1．篠崎　直二郎（実行委員長，ＳＥＡ会員）

ＮＥＣソフトウェア　クライアントサーバシステム事業部技術研修センター

2．河村　一樹（プログラム委員長，ＳＥＡ会員）

尚美学園短期大学　情報コミュニケーション学科

3．川辺　正明（プログラム委員，ＳＥＡ会員）

㈱スタット　社長室技術部

4．君島　浩（プログラム委員，ＳＥＡ会員）

富士通ラーニングメディア　第１研修部

5．杉田　義明（プログラム委員，ＳＥＡ会員）

日本ＮＣＤ

6．中園　順三（プログラム委員，ＳＥＡ会員）

㈱富士通ＢＳＣ　技術推進部

7．青山　幹雄（ＳＥＡ会員）

新潟工科大学　情報電子学科

8．牧野　憲一（ＳＥＡ会員）

オムロンソフトウェア㈱　企画室

9．和田　勉（ＳＥＡ会員）

長野大学　産業情報学部

10．橋本　勝（ＳＥＡ会員）

山一情報システム㈱　システム総務部

11．平山　順子（一般）

山一情報システム㈱　オープンシステム第一部

12．佐村　愛佳（事務局）

尚美学園短期大学　情報コミュニケーション学科（河村ゼミ）

13．川上　知子（事務局）

尚美学園短期大学　情報コミュニケーション学科（河村ゼミ）

また，本ワークショップでは，参加申し込みにともなう事前アンケート調査を実施している．アンケートの質問項目は，以下の６項目である．

① ソフトウェア技術者教育について，日頃思っていること．

② 本ワークショップに期待すること．

③ 技術者の育成において，日頃工夫していることや苦労していること．

④ 教育ワークショップでとくに取り上げてほしいテーマ

⑤ 今考えている（もしくは作成した）ポジションペーパーのテーマ

⑥ その他本ワークショップに対する要望や意見

それでは，ワークショップ参加者（杉田さんと事務局以外）各人のアンケート回答について，以下に記載（五十音順，敬称略）する．

【青山 幹雄】

① 産業界全体として教育が重視されていない．大学・産業界を問わず，教育方法，教育内容，テキストについて，もっと議論すべきである．例：Ｃ言語の良い教科書が非常に少ない．

② 産業界，大学などにおける教育のあり方，方法について突っ込んだ議論をすること．③ この９月からＣ言語を大学１年生に教えている．

・テキストの選定：約40冊の類書から選ぶ

・内容：毎回実習をする（物作りを重視する）

・環境：１人１台ワークステーション

④ ソフトウェア技術者教育のあり方（今さら？）

⑤ なし

⑥ よろしく

【川辺 正明】

① 企業として将来の発展およびリスク回避のため，必要不可欠の行動と思っている．しかし，現実には軽視されることが多く，このままでは企業はもとより，産業としての確率も危ういのではないか．

② 教育，教育と騒がなくても，良い会社，またはそのような状況を作り出すため，何らかの答えを見つけたい．

③ 目標を明確にすること

④ なし

⑤ 産業として確立すべき情報サービス産業と教育の役割

⑥ ひたすら楽しみにしています．

【河村 一樹】

① 学校教育だけでなく，企業教育においてもコンピュータサイエンスをベースにした教育体系を確立すべきと思っている．たんなる技能教育だけでなく，基礎理論も含めた実践的カリキュラム体型ができないだろうか．

② 今年こそは，教育ワークショップとして報告書を発刊したい．そのために，私のゼミの学生を参加させ，テープおこしからＤＴＰによる編集を含めて完成させたい．

③ 文科系学生に数理科学を基礎にしたコンピュータサイエンス教育を行うことのむずかしさに苦労している．数学嫌いの学生に，基礎理論を教授するためには，具体的実例をもとに抗議を展開すると教育効果が高まる．

④ オブジェクト指向パラダイムといった新しい方法論をどのように企業内で技術移転することができるか，その具体的アプローチについて．

⑤ 「手続き指向からオブジェクト指向へ」の技術移転のための教育

⑥ 今年度はぜひ報告書を作成しましょう.

【君島 浩】

① なし

② なし

③ なし

④ なし

⑤ 特集に敬意を表して，私および弊社の商談の事例などを紹介したい．あとは，９月の米国出張の見聞も.

⑥ なし

【篠崎 直二郎】

① インストラクタの質の向上と効率的な作業の取り組み

② マンネリを打開しよう.

③ 技術基礎の急激な変化と教育担当の保守化傾向

④ どこから教育工学に手をつけるか.

⑤ プロジェクト管理教育における参加型教育の事例紹介

⑥ そろそろ実行委員長を誰かさんに変わろうよ.

【中園 順三】

参加申し込み依頼文のみ．．．でした.

【橋本 勝】

① 「教育」と「躾」

② ソフトウェア技術者に対する考え方，アプローチの仕方を吸収

③ 保守業務に対するＣＡＳＥツール，リバースツールの導入

④ 意識を変えるためのアプローチ

⑤ リバースツール導入教育事例

⑥ 今年は肩の力を抜いて参加できそうです．よろしくお願いします.

【平山 順子】

① 実際にＰＣ教育を日頃業務として行っているが，いまだ「紹介」の域から脱していないよう感じています．今回このセミナーに参加することにより，このヒントを得られれば幸いです.

② なし

③ 身近な例をあげ，具体的にイメージできるよう説明する.

④ キーボードの使用，初心者に対し，キー入力を行えるように指導する方法

⑤ 中高年に対するＰＣ教育の現実

⑥ なし

【牧野 憲一】

① 本当に技術を保有している者は，どんどん新しい技術を取り入れようとする姿勢を見せるが，実は自信がない技術者は理由をつけて今のやり方を守ろうとする．この根底に流れる姿勢・考え方の違いにより，

教育効果も大きく異なる．後者の人物をいかにして活用するかが改題である．

② 直接利害関係がないメンバなので，お互い忌憚のない意見交換が可能である．

③ 研修を効果的に行うには，カリキュラムの充実のみならず，研究を取り巻く仕組み・仕掛けが重要な要因になる．別途PPに記載する．

④ パソコンを中心とする激しい変化の時代に対応できる人材を早期に育成するにはどうすればよいか．

⑤ 時間がないのであまり書けないと思いますが，上記③に関連して作成したいと思います．

⑥ とくにはありませんが，初参加の方にはからなずコーナーを設けて何でもいいから自己紹介以外に日頃考えていることを発言してもらいましょう．

【和田 勉】

参加申し込みだけ．．．でした．

講演・討議内容

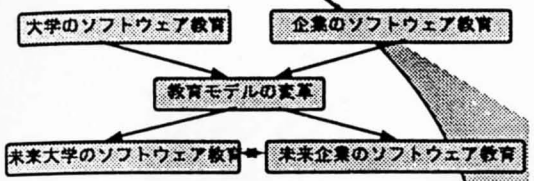
ここからは，招待講演を含め，各セッションの内容について記載する．ただし，具体的な内容については，発表者のポジションペーパーあるいはＯＨＰシートをそのまま載せることにする．本来ならば，講演内容すべてを詳細に記載したかったのだが，一部テープが聞き取れないところなどがあったため断念せざる得なかった．ただし，質疑応答についてだけは，テープ起こしがうまくいったので載せることができた．

１．招待講演

青山さんによる「技術変革期での教育のあり方」について講演が行われた．青山さんは，富士通に長く勤務されてから，最近新潟工科大学に移られたという経歴の方である．

以下に，講演で使われたＯＨＰシートを記載する．その次に，質疑応答（Ｑは質問，Ａは回答，Ｃはコメント）についてまとめる．

産学におけるソフトウェア技術者教育のあり方



青山 幹雄
新潟工科大学 情報電子工学科

大学のソフトウェア教育

C言語を教える：なぜCか？

1．実践への傾斜
　専門学校ではないが、就職してこまるのでは？

2．言語として
　Ｐａｓｃａｌ：１年前期
　Ｃ＋＋：１年生にはちょっと。。。
　やはり、Ｃか。。。

3．取り巻く環境
　大学のＵＮＩＸ文化・パソコンでも使えます

大学のソフトウェア教育

私の授業設計

1．目　標
　Ｃ言語のプログラムを読み，書ける
　プログラム言語の構成要素の考え方を理解する
　プログラミングの楽しさを知る
2．授業設計
　前半（４５分）：教科書に沿って講義
　後半（４５分）：例題演習（レポート）
　授業は一人一台ＷＳが利用出来る演習室で実施
3．途中経過
　内容の理解は十分とは言えない：実装の難しさ
　演習は，好き好き：時間不足

大学のソフトウェア教育

悩める大学：４年後のあなたの部下は？

1．入試の功罪：多様化する学生
　理科を知らない学生，高校の補修をする大学
2．カリキュラム改革の功罪：１年生の専門教育
　言語は使えれば良いか：計算メカニズムと言語
3．膨れ上がる技術：先端技術と教育内容のギャップ
　何をどこまで教えるべきか
4．大衆化の功罪
　生徒化する学生：学ばない学生，８０人教室
5．大学教員の教育技術：素人教育者集団
　教育方法論，コンピュータを知らない教官

大学のソフトウェア教育

Ｃ言語を教える：教科書探しの顛末

1．Ｃ言語の本：約５０種類
2．私の選択基準
　１）文法だけでなく概念も教える
　２）１年生のための入門書
　３）プログラミングできること
　４）プログラミングに
　　親しみ・楽しみを持てること
　５）高くない：２０００円程度
3．採用した教科書
　清水・首田著，Ｃ言語のススメ，サイエンス社

大学のソフトウェア教育

体験的ソフトウェア教育私論

1．文法主義からの脱却
2．講義室から工作室（ソフトウェア工房）へ
　物作りの楽しさ
　画一な『工場型教育システム』からの脱却
　演習：半日
3．読んで分かる教科書，文法書から脱却
　良い教科書には先人の知恵が詰まっている
4．学びたいと思ったときに学べる仕組み
　常時オープンな演習室：ソフトウェア遊戯室
　ネットワークの活用

企業のソフトウェア教育

企業内ソフトウェア技術者教育プログラム

1．基礎教育
　論理的に説明する，文章にまとめる
　ビジネスマナー
2．いわゆるソフトウェア教育
　1）新人教育
　・開発工程にそった設計方法の講義
　・プログラミング演習
　・ＯＪＴ／ＯｆｆＪＴ
　2）中堅社員教育
　3）管理職教育

企業のソフトウェア教育

企業内ソフトウェア技術者教育の課題（2）

3．ソフトウェア『専門教育』の欠如，ＯＪＴ依存
　・設計方法論
　・基本的アルゴリズム
　・試験技法
　・開発管理
4．中堅社員教育
　姿なき中堅技術者
5．国際化とソフトウェア開発技術の空洞化
　物作りの軽視
6．インターネット？

教育モデルの変革

未来型教育モデル

大量生産型教育システムからJust-In-Time教育へ

・権威型からエキスパート型へ
・全体（クラス）から個人（インスタンス）へ
・大量生産型からJust-In-Timeへ
　学びたい時に，学びたいものを，学びたいほど

Imagina a University without walls
– D.Perrin (ED Journal/IEEE Computer)

企業のソフトウェア教育

企業内ソフトウェア技術者教育の課題（1）

1．知識連鎖（キャリヤパス）の崩壊
　プログラミング＞＞設計＞＞ＳＥ
　・いわるる大企業：仕様書書き，発注管理
　・ソフトハウス：断片的設計，プログラミング
2．開発を取り巻く条件の多様化
　・アーキテクチャの変化
　　ＰＣアプリ，ウィンドウ（ＧＵＩ）アプリ
　　クライアント／サーバシステム
　・開発方法の変化：オブジェクト指向，ＲＡＤ
　・開発条件の変化：短期開発

教育モデルの変革

未来型教育モデルへのアプローチ

・大量生産型教育システムの限界

・ＩＴ技術による新しい教育システムの可能性

・ソフトウェア教育の必要性の広がり
　あらゆる人にソフトウェア教育を？
　企業：専門家教育
　大学：理系・文系
　家庭：非専門家教育
　　パソコン・カルチャースクール？

未来大学のソフトウェア教育

未来大学の実現

1．Just-In-Time教育
　学びたい時に，学びたいものを，学びたいだけ

2．いつでも学べる仕組み

3．オフ・キャンパス教育：自宅から，会社から

4．世界中から，世界中へ
　フィンランドの学生から指導を依頼された経験

5．経済性の実現：ＩＴ技術の活用

未来大学のソフトウェア教育

すぐできること

・学生個人が自由に物作りできる環境の実現
　ソフトウェア実験（遊戯）室
　異機種マシン（１０台），ファイルサーバ
　常駐アドバイザ（技官）
・ネットワークの活用
　電子メールによる直接アドバイス
　グループアドバイザ制度
・カリキュラムの内容の見直し
　何を教えるか，実験・実習の内容，教材
　『教』と『育』
・外部講師の活用：現場とのコミュニケーション

未来大学のソフトウェア教育

未来大学の姿と教官像

The University of the Future is a virtual learning environment, not a physical campus place.
D. Perrin, ibid.

未来大学の教官は，物理的実体としての大学とは独立の『仮想教官』である。

未来企業のソフトウェア教育

米国企業内での大学教育例

・専門教育：モトローラ大学，インテル大学
・大学院の企業内分校：モトローラなど
　夜間に企業内の会議室で授業
　ＣＡＴＶによる遠隔授業
　WWW／ビデオなどの教材による自習
・大学院派遣：修士，博士課程
　パートタイム，フルタイム
・大学の夜間時間における一般講義
・社内講師による教育との組み合わせ

未来大学のソフトウェア教育

劇するＭＢＡ達：あるＭＢＡクラス

・事例研究授業
３０分：担当チームによる調査企業の劇表現
　例：アメリカン航空：スチュワーデス物語風
３０分：調査対象企業の経営者による講演
６０分：経営者への質疑応答

・批評（１回／５事例）
幾つかの事例をまとめて，幾つかの観点から批評

・デベイト（２回／学期）
例：経営者と倫理，会社はだれのものか

未来企業のソフトウェア教育

大量生産型教育システムからJust-In-Time教育へ

１）新人教育のあり方

２）中堅社員教育のあり方
・個人の専門能力の追求
　個人が自ら学ぶための環境
・技術移転教育の必要性：オブジェクト指向？

３）シニア教育とシニアの活用
・シニアの動機付け
・技術の伝承

まとめにかえて

ソフトウェア工学大学教官の役割：*現場に身を置く自己弁護*

１．大学教官による現場のソフトウェア教育
　『専門教育』の実現：方法論の体系的教育

２．現場の空気を吸う
現場にいるだけでは現場を知ることにはならない
しかし。。。

３．ソフトウェア工学の実験場：企業内研究室
　Research-then-Transfer から
　Industry-as-Laboratory へ
　(C. Potts, IEEE Software, Sep. 1993)

Q. 篠崎「（採用した本について）関数概念をまず最初に話しているか？」
A. 青山「それ程説明はしていない．途中から出てくるが，まず文法からではなくて，その時は形で表して説明している．」

Q. 篠崎「演習問題は，全員同じ問題を出して各自に解かせるの？」
A. 青山「同じ問題で各自に解かせている．大抵の人はお互いに相談してやったり，よく出きる子がいるのでその人が教えたり，私と２人教える人がいる．わからない人は授業外に演習室に来てやり終える．」

Q. 和田「理解しようとする態度はあるのか？理解しないで丸写ししようという人もいるのでは？」
A. 青山「そういう人もいる．演習の時間はなるべく一緒にいるようにして，わからない人には手を挙げてもらう．」

Q. 和田「出来たけど電子メールの出し方がわからないから出さないというのは？」
A. 青山「一応，電子メールは教えて使えるようになっている．全員使えるようになっている．」

Q. 河村「出席率は？」
A. 青山「必修授業なので，出席率は良い方．」

Q. 平山「人数は？」
A. 青山「９０人．」

Q. 牧野「１回抜けると，次の週に出てきた時に出来ないということがあるのでは？」
A. 青山「ある．その場合説明し，理解してもらう．抜けてしまって，いやになってしまうと困る．」

Q. 和田「抜けたのだから分からなくて当たり前と思うのならまだいいが，自分には向いていないとか難しいとか思われて困った経験は？」
A. 青山「まだ，いない．もう少ししたらそういう人が出てくるかも．」

Q. 平山「レポートの採点等の基準はどのように？」
A. 青山「レポートをすべて提出すれば成績５割は確実．後の５割は，試験．差が大きいので，提出したことに対して付けている．」

Q. 河村「出席はカウントしているのか？」
A. 青山「今は取り合えずレポートだけで考えている．出席取るだけで時間を取ってしまう．」

Q. 君島「出席していないのにメールレポートだけ来たことは？」
A. 青山「無きにしも非ず．」

Q. 河村「前期にアルゴリズムとデータ構造とか，全くなしで突然Ｃをやったのか？」
A. 青山「Ｐａｓｃａｌはやったが，データ構造とか基本的なアルゴリズムとかはやっていない．」

Q. 河村「概念は２年生になってからやるのか？」
A. 青山「２年生でやる．」

Q. 中園「就職先としてはソフトウェアを作るところに大多数の学生が行きたがっている？」

A．青山「きちんと調査はしていないが，かなりの人はそう思っている．」

Q．和田「“学びたい時に，学びたいものを，学びたいほど”学びたくないものは見放す？」
A．青山「見放せない．学生は，下に合わせると上が退屈，そうするとクラスの雰囲気が悪くなる．上の人には次の課題を与えるようにするか，教えるのを手伝ってもらう．ギャップを埋めるようにする．」

Q．篠崎「教育は拡大するのと儲けは違う．儲かるのか？」
A．青山「創才学院がなぜ儲かっているのかは，よく分からないが仕組みとして非常によくできている．教材がよくできているだけではなくて，個人の特性を把握するような仕掛けを作っている．こういうことをやれば当然コストはかかるが，授業料はそれほど高くない．土台をきちんと作ればいいのでは．」

Q．河村「個人レベルの教育をするために，画一的な教育をどうやって切り崩していくのか？」
A．青山「企業には，入社試験を厳しくして欲しい．そうすれば，大学では自分で学ぶようになり，また，それなりに仕事が出来るようになるのでは．」
C．君島「アメリカのソフトウェア会社のＪ３ラーニング株式会社では，“学びたい時に，学びたいところで，学びたい分”だけ教育するような形式を取っている．」

2．第1セッション

私による「技術移転『手続き指向からオブジェクト指向へ』のための教育」について講演を行なった．私は，現在尚美学園短期大学の情報コミュニケーション学科で教鞭を取っている．担当科目は，コンピュータ科学論・データベース概論・データベース演習・ゼミナールなどである．また，情報処理学会の情報処理教育に関する委員会委員やＣＡＩＴの各種委員会委員を兼務している．

以下に，講演のためのポジションペーパーを記載する．その次に，質疑応答（Ｑは質問，Ａは回答，Ｃはコメント）についてまとめる．

（なぜか、宴会の写真です）

Postion Paper For SIGEDU WorkShop'95

技術移転「手続き指向からオブジェクト指向へ」のための教育

A Education for Technology Transfer about "from Process Oriented to Object Oriented"

尚美学園短期大学，川越市
Shobi Junior College,Kawagoe city
河村　一樹
Kazuki KAWAMURA

1．はじめに

今日のコンピュータ領域に関する技術革新は，著しい進展が見られる．それにともない，新しい技術の現場への普及と定着を目指して技術移転を強化する必要が生じている．

このような中で，最近とくに顕著なこととして，新しいソフトウェアパラダイムの台頭があげられる．ソフトウェアパラダイムとは，情報システムを構築するためのソフトウェア設計・開発に対する規範のことである．また，特徴的なこととしては，コンピュータシステムに実装するための操作環境に依存する部分が多い点である．具体的には，プログラム言語の種類があげられる．

プログラム言語には，手続き型言語をはじめに，非手続き型言語として関数型（Lisp）・論理型（Prolog）・オブジェクト指向型（Smalltalk-80）などがある．手続き型言語は，その歴史も古いことから，現在でも広く普及している．一方，手続き型言語では記述力が弱い領域（例えば，人工知能，ＧＵＩ等）に対して，それぞれ特化した機能を言語に装備した非手続き型言語が注目を集めつつある．その中で，再利用性を強化したオブジェクト指向型言語は，保守工数を大幅に軽減できる可能性が高い点から注目をあびつつある．以上のようなプログラム言語の推移は，新しいソフトウェアパラダイムを生み出す原動力になった．

本稿では，新しいソフトウェアパラダイムの一つであるオブジェクト指向パラダイムを取り上げ，既存の手続き指向パラダイムからの転換を教育面から支援するためのアプローチについて述べる．その中で，オブジェクト指向教育のためには，その方法論（概念）の教育，その技法（分析・設計・プログラミング）の教育，実装までの教育，開発支援環境の整備，などが有効であることを示す．それとともに，いままでの手続き指向教育の体系を一部適用することも効果的であることも報告する．

2．手続き指向パラダイムとその教育

2．1　手続き指向パラダイムとは

手続き指向によるソフトウェアパラダイムとしては，代表的なものに構造化パラダイムがある．実装のためのプログラム言語は，当然手続き型言語（COBOL，FORTRAN，PL/1，ALGOL，PASCAL，Ada，Cなど）が相当する．

構造化パラダイムは，その基盤になる方法論と一連の技法によって構成される．以下に，具体的に示す．

（1） 構造化パラダイムの方法論

これには，次のようなものがあげられる．

・フォワードプロセス（forward process）

ウォータフォールモデルを前提に，後戻りのない開発工程に準拠する．

・分割統治（divide and conquer）

ソフトウェアを部分に分割した上で統合するための指針に相当する．

・段階的詳細化（stepwise refinement）

大規模な問題を，構造的に分割するための指針に相当する．

・情報隠蔽化（information hiding）

構造化された部分に対して独立性をもたせることが要求され，そのための指針にもなり得る．

・手続きと制御の抽象化（abstraction）

構造化プログラミングの基本的な枠組みであり，構造的なプログラムを作成するための指針にもなる．

以上のように，いずれも構造化パラダイムを実践する上での基本的な考え方になるとともに，このような考え方にもとづいて，具体的な技法が展開されることになる．

（2） 構造化パラダイムの技法

これには，次のようなものがあげられる．

・構造化分析（Structured Analysis: SA）

デマルコ（T.DeMarco）により提案された技法［1］である．事務系の適用業務処理に対する構造化仕様書を作成する．なお，これ以外に，制御系のリアルタイム処理に拡張したもの［2］［3］もある．

・構造化設計（Structured Design: SD）

コンスタンティン（L.L.Constantine）により提案された技法［4］である．構造化分析の結果をもとにして，インタフェースを含めたモジュール構造を作成する．また，同類のものに，マイヤーズ（G.J.Myers）の複合設計（composite design）［5］がある．

・構造化プログラミング（Structured Programming: SP）

ダイクストラ（E.W.Dijkstra）により提案された技法［6］である．制御の抽象化をプログラミングに取り込んだものである．

・構造化チャート（structured chart）

構造化プログラミングを図式によって記述できるようにした技法［7］である．すでに標準として規格化されたものとして，PSD（オランダ，西ドイツ規格），DSD（イギリス規格），SPD（日電），HCP（ＮＴＴ），PAD（日立），LCP（フランス規格）があげられる．

・構造化コーディング（structured coding）

手続き型言語を用いて，見やすくわかりやすいプログラムリストの作成をめざすための技法［8］である．

２．２ 手続き指向パラダイムの教育

1970年代以降に提案された構造化パラダイムに対しては，米国はもとより我が国においても，実務現場あるいは教育機関においても定着した観がある．前者については，例えば情報処理技術者試験のための標準カリキュラム［9］［10］において，全面的に取り入れられている．後者については，大学等の情報系専門学科におけるコンピュータサイエンス教育カリキュラム［11］の中で，具体的に指摘されている．このように，構造化パラダイムの教育は，実際のカリキュラムとして体系化されていることがわかる．

ただし，このようなパラダイム教育においては，次のような留意すべき事項がある．

① 方法論から技法への展開

具体的な技法の教育に入る前に，構造化パラダイムの方法論を教育する必要がある［12］．これによって，構造化パラダイムが持つ基本的な考え

方が把握できることから，具体的な技法についても理解しやすくなるという効果が見られる．

② 事例研究の強化

構造化パラダイムそのものは，数理科学にもとづく理論的なバックボーンはなく，経験的な成果を集大成したといってよい．このため，属人性に依存する部分も多い．そこで，過去の成功例を事例として教材に用いると，より効果的である．これによって，どうやればうまくいくかという指標を得ることができる．

③ デモンストレーション（分析・設計・実装）の実施

構造化パラダイムにもとづき，小規模な情報システムを構築する場面を一通り体験することによって，一貫した技法として体系化することができる．

④ 開発環境の整備

構造化パラダイムを支援するＣＡＳＥを利用することによって，実装部分をある程度自動化することが可能になる．これによって，分析・設計内容のフィードバックが容易に実施でき，評価の期間を短縮することができる．

３．オブジェクト指向パラダイム

３．１ オブジェクト指向パラダイムの台頭

1972年に作成されたSmalltalk-72をもとに，米国ゼロックス社パロアルト研究所のAltoマシンで稼動する言語処理系として実現されたSmalltalk-80によって，オブジェクト指向型言語が登場した．それまでの手続き型言語とは異なるいくつかの新しい試みが取り入れられた斬新なプログラム言語になった．

また，もともとSmalltalk-80は高性能ワークステーション用言語として開発されたという経緯をもつ．それとともに，著しいワークステーションの普及によって，Smalltalk-80に注目が集まるようになった．

以上のようなSmalltalk-80の登場は，オブジェクト指向パラダイムの土壌を築く土台になったといってよい．その背景として，既存のプログラム言語に大きな影響を与えたという経緯があげられる．例えば，次のような新しい拡張が行われている．

・Cから，C++あるいはObjective-Cへ

・PASCALから，Object PascalあるいはTurbo Pascal(Ver5.5)へ

・LISPから，CLOSあるいはFlaovorsへ

・Prologから，ESPへ

・COBOLから，OO-COBOL(？）へ

このようなプログラム言語のオブジェクト指向化にともない，下流工程にある言語の世界だけではうまくプログラミングができないという問題が生じてきた．そこで，より上流工程である分析や設計の世界から，オブジェクト指向を取り入れる試みが進められた．その結果，オブジェクト指向分析やオブジェクト指向設計が提案されるようになった．それだけでなく，オブジェクト指向は，オペレーティングシステム領域やデータベース領域およびＧＵＩ領域にも適用できることが明らかになった．以上のような活動から，一連のオブジェクト指向パラダイムが体系化されたわけである．

３．２ オブジェクト指向パラダイムとは

オブジェクト指向パラダイムも，その基盤になる方法論と一連の技法によって構成される．以下に，具体的に示す．

（１） オブジェクト指向パラダイムの方法論

これには，次のようなものがあげられる．

・インクリメンタルプロセス（incremental process)

らせんモデルのような繰り返しのある開発工程に準拠する.

・データ抽象化 (data abstraction)

機能ではなく，データの状態とその振る舞いに着目するという指針に相当する.

・カプセル化 (capsulation)

対象（オブジェクト）の状態と振る舞いを一緒にするという指針に相当する.

・情報隠蔽化 (infomation hiding)

カプセル化によって，オブジェクトの状態は手続きからでしかアクセスできないという指針に相当する.

・継承 (inheritance)

共通する性質（クラス）を階層的な構成によって引き継ぐという指針に相当する.

以上の中では，手続き指向パラダイムと一部共通するものがある．これより，オブジェクト指向パラダイムは，手続き指向パラダイムから派生した部分があるといってもよい.

これに対して，手続き指向パラダイムとは，本質的に異なっている部分（新しい拡張）もある．その一つが，カプセル化の概念である．手続き指向パラダイムにおけるプログラムの考え方は，手続きとデータを分離して捉える傾向にある．この結果，データベースの世界では，データ独立性 (data independecy) というアプローチを確立した．それにともない，一方ではインピーダンスミスマッチ（プログラム言語とデータベース言語の非関連性）という問題が生じている．これに対して，オブジェクト指向パラダイムでは，カプセル化によりデータと手続きを一体化する．また，データベースに関しては，オブジェクトの永続性という視点から言語とデータベースの区別はほとんどない.

もう一つは，継承の概念である．これは，手続き型言語には存在しない機能である．継承によって，プログラムの再利用性が高まり，クラスライブラリの整備による生産性向上が期待できることになる.

（2） オブジェクト指向パラダイムの技法

これには，次のようなものがあげられる.

・オブジェクト指向分析 (Object-Oriented Analysis: OOA)

問題領域に対して，クラス／オブジェクトの構造と性質，あるいは，メッセージによる相互作用などを決定するための技法である.

・オブジェクト指向設計 (Object-Oriented Design: OOD)

オブジェクト指向分析の結果に対して，コンピュータによる実装を前提にした設計のための最適化を行う技法である.

・オブジェクト指向プログラミング (Object-Oriented Programming: OOP)

オブジェクト指向型言語を用いてプログラミングを行うことである．言語処理系としては，次のような機能が装備されていること [13] が条件になる.

クラスライブラリ (class library)，

継承 (inheritance)，

メソッド (method)，

データ抽象 (data abstraction)，

動的束縛 (dynamic binding)，

隠蔽ぺい (encapusulation)，

ガベージコレクション (garbage collection)，

ポリモアフィズム (polymorphism)，

メタクラス／メタオブジェクト (meta-class/meta-object)，

ブラウザ (browser)

・オブジェクト指向オペレーティングシステム (Object-Oriented operating system)

並列制御や分散処理を実現するために，オブジェクト指向の考え方をオペレーティングシステ

ムの機能として実現したものである.

・オブジェクト指向データベース（Object-Oriented database）

オブジェクトの永続性を実現するために，オブジェクト指向の考え方をデータベースの機能として実現したものである.

・オブジェクト指向GUI（Object-Oriented Graphical User Interface）

アイコンとマウスをもちいたイベントドリブン方式の処理系を実現するために，オブジェクト指向の考え方をGUIの機能として実現したものである.

以上の中で，オブジェクト指向分析・設計に関しては若干問題が残っているといえる．それは，現時点において標準化されたものがなく，提案者それぞれ（例えば，［14］［15］［16］［17］［18］など）によって内容に差があることである．このため，現場ではどれを採用すべきかについて迷うようなケースが生じている.

表１．手続き指向とオブジェクト指向の比較

	手続き指向ﾊﾟﾗﾀﾞｲﾑ	ｵﾌﾞｼﾞｪｸﾄ指向ﾊﾟﾗﾀﾞｲﾑ
基盤になる理論	ｺﾝﾋﾟｭｰﾀｻｲｴﾝｽの基礎理論（ｵｰﾄﾏﾄﾝ論，検証理論，計算量理論，等）	とくになし
開発ﾌﾟﾛｾｽﾓﾃﾞﾙ	ｳｫｰﾀﾌｫｰﾙﾓﾃﾞﾙ	ｲﾝｸﾘﾒﾝﾀﾙﾓﾃﾞﾙ
方法論	・構造化分析 ・構造化設計 ・構造化ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ ・構造化ﾁｬｰﾄ ・構造化ｺｰﾃﾞｨﾝｸﾞ	・ｵﾌﾞｼﾞｪｸﾄ指向分析 ・ｵﾌﾞｼﾞｪｸﾄ指向設計 ・ｵﾌﾞｼﾞｪｸﾄ指向ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ
技法	・段階的詳細化 ・情報隠蔽化 ・手続きと制御の抽象化 ・分割統治	・段階的詳細化 ・情報隠蔽化 ・ﾃﾞｰﾀ抽象化 ・ｶﾌﾟｾﾙ化 ・継承
ﾌﾟﾛｸﾞﾗﾑ言語	手続き型ﾌﾟﾛｸﾞﾗﾑ言語（COBOL, FORTRAN, PL/1, PASCAL, C, 等）	ｵﾌﾞｼﾞｪｸﾄ指向型ﾌﾟﾛｸﾞﾗﾑ言語（Smalltalk-80, C++, ESP, CLOS, 等）
ﾌﾟﾛｸﾞﾗﾑの特徴	関係ﾃﾞｰﾀﾍﾞｰｽ	部品化＋再利用
ﾃﾞｰﾀﾍﾞｰｽ	ﾃﾞｰﾀ構造＋ｱﾙｺﾞﾘｽﾞﾑ	ｵﾌﾞｼﾞｪｸﾄ指向ﾃﾞｰﾀﾍﾞｰｽ

３．３　手続き指向からオブジェクト指向パラダイムへの転換

手続き指向パラダイムとオブジェクト指向パラダイムを比較すると，表１のようになる.

その中で，二つのパラダイムの最も顕著な相違点は，実装部分にあるといってよい．つまり，プログラムの再利用性を考慮してないか，あるいは，しているかという点である．ここに，オブジェクト指向パラダイムが手続き指向パラダイムと比べて優位性があるといえる.

90年代にはいって，実践的な場でのソフトウェア開発にも，オブジェクト指向パラダイムが採用され始めている［19］［20］．具体的には，小規模なプロジェクトで，いわばプロトタイプとしてオブジェクト指向を用いた開発する形で進められている．これより，現在は，手続き指向パラダイムから，オブジェクト指向パラダイムの転換期に位置しているといってよい.

手続き指向パラダイムからオブジェクト指向パラダイムへの移行は，表２のようなケースが考えられる．このうち，転換期という意味で，手続き指向とオブジェクト指向を一緒にしたようなケースが見られる．つまり，上流工程でオブジェクト指向パラダイムを，下流工程で手続き指向パラダイムを用いて，システムの実現をはかるというやり方である．ただし，いずれも開発工数にかかる負担は大きくなる.

いずれは，オブジェクト指向パラダイムを支援

表２．手続き指向とｵﾌﾞｼﾞｪｸﾄ指向の組合せ

分析・設計	実装	ﾃﾞｰﾀ管理	負担度合
ＳＡ／ＳＤ	ＳＰ	ＲＤＢ	大
ＯＯＡ／ＯＯＤ	ＳＰ	ファイル	大
ＯＯＡ／ＯＯＤ	ＳＰ	ＲＤＢ	大
ＯＯＡ／ＯＯＤ	ＯＯＰ	ファイル	大
ＯＯＡ／ＯＯＤ	ＯＯＰ	ＲＤＢ	中
ＯＯＡ／ＯＯＤ	ＯＯＰ	ＯＯＤＢ	小

SA:構造化設計
SD:構造化設計
SP:構造化ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ（手続き指向）
OOA:ｵﾌﾞｼﾞｪｸﾄ指向分析
OOD:ｵﾌﾞｼﾞｪｸﾄ指向設計
OOP:ｵﾌﾞｼﾞｪｸﾄ指向ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ
RDB:関係ﾃﾞｰﾀﾍﾞｰｽ
OODB:ｵﾌﾞｼﾞｪｸﾄ指向ﾃﾞｰﾀﾍﾞｰｽ

する開発環境（例えば，オブジェクト指向ＣＡＳＥなど）が整備されるにしたがって，すべてのプロセスにおいてオブジェクト指向が採用されることになるといえる．

４．オブジェクト指向パラダイムの教育

以上のような経過の中で，今後手続き指向パラダムよりも，オブジェクト指向パラダイムについての技術移転に対して関心が高まるといえる．そのためには，オブジェクト指向パラダイム教育のための在り方や具体的なカリキュラム指針について明らかにする必要がある．そこで，次のような提案を行う．

４．１　教育体系の在り方

（1）　事前の習得条件

オブジェクト指向パラダイムの教育を受ける側として，あらかじめ習得しておいた方がよい知識と技能がある．具体的には，次のようなものがあげられる．

①　知識関連

・事象駆動システム（ペトリネット，プロダクションシステムなど）

・抽象データ型（Abstraction Data Type）

②　技能関連

・問題領域のモデル化表現（例えば，Entity-Rerationship Diagram，Data Flow Diagram, State Transition Diagramなど）

・ＧＵＩ（Windows, Macintosh, X Windowsなど）

・プログラミング（Visual BASIC, HyperTalk, SIMSCIPTなど）

（2）　アプローチの方法

技術移転を行う教育には，二通りのアプローチがある．一つは，トップダウン方式であり，基礎理論や概念（おもに講義）から始めて応用技術（おもに演習）へ至るアプローチである．抽象化から具体化への展開である．もう一つは，ボトムアップ方式であり．応用技術から始めて基礎理論や概念へ至るアプローチである．具体化から抽象化への展開である．

以上のことをソフトウェアパラダイムの教育にあてはめてみると，「方法論からプログラミングへ」か「プログラミングから方法論へ」という二つのアプローチに相当する．オブジェクト指向パラダイムの教育という視点からは，前者の方が効果的である．というのも，オブジェクト指向では，問題領域のモデル化やそこでのオブジェクトの切り出しといった作業を行わない限り，プログラミングができないことがあげられる．また，実際のプログラム開発では，新規にプログラミングすることはほとんどなく，既存のクラスライブイラリの修正や追加（差分プログラミング）が中心になるからである．

以上より，理論や概念といった方法論を講義によって実施してから，事例をおり混ぜた演習を行うといったアプローチを提案する．

４．２　カリキュラム構成

４．１にもとづく具体的なカリキュラムの構成

について提案する.

（1） 講義

① オブジェクト指向の概念

取り上げるべき事項は，次の通りである.

・抽象化と体系化
・情報隠蔽化
・カプセル化
・データ抽象と抽象データ型

② オブジェクト指向の基本用語

取り上げるべき事項は，次の通りである.

・クラスと抽象クラス
・オブジェクトとインスタンス
・属性（状態）
・メソッド（振る舞い）
・メッセージ
・動的結合（束縛）と静的結合
・is-a関係（特化と汎化）
・part-of関係（分解と集約）
・継承（単一継承と多重継承）
・多相性

③ オブジェクト指向分析・設計

ここでは，要求モデルを三つの視点から明らかにすることを目指す.

・対象の静的構造
・対象の動的構造
・対象が持つべき機能

これより，問題領域をどのようにオブジェクトおよびクラスの構造としてモデル化するかがポイントになる．また，モデル化した対象を，コンピュータシステムに実装するための最適化についての手順を明らかにする必要がある．ただし，この段階までは，プログラム言語にどう落とすかといったことを考慮する必要はない.

OMT法の場合は，オブジェクトモデル，動的モデル，機能モデルの三つを作成する.

ブーチ法の場合は，論理設計としてオブジェクト図・クラス図・状態遷移図・オブジェクト交信図を，物理設計としてモジュール図・プロセス図を，それぞれ作成する.

コード・ヨードン法の場合は，多重レイヤーモデル，多重コンポーネントモデルの二つを作成する.

シュレィアー・メラー法の場合は，情報モデル，状態モデル，プロセスモデルの三つを作成する.

④ オブジェクト指向プログラミング

ここでは，オブジェクト指向分析・設計で導いた仕様をどのようにプログラムに具現化するか，その過程を明らかにする必要がある．実装部分を理解しておくことは，プログラミングに従事しない設計者にとっても必要なことである．このことは，手続き指向パラダイムの場合でも然りである.

なお，技術移転という立場から見ると，既存言語を拡張したもの（例えば，C++やObject Pascalあるいは00-COBOLなど）が有効といえる．また，再利用性という面からクラスライブラリの管理についても考慮する必要がある.

（2） 演習

① 事例研究

ソフトウェアパラダイムの教育は，非常に抽象的で概念的である．このため，具体的な事例を参照しながら教育を進めていく方が効果的といえる．そこで，オブジェクト指向分析・設計といった一連の開発手順を，実践的な事例（例えば，［23］をもとに演習を行うといったアプローチが考えられる.

② プログラミング演習

講義で取り上げた内容をもとに，オブジェクト指向プログラム言語を用いてプログラミングを行う．この場合，①で扱ったテーマの一部をプログラミングするようなプロトタイプを作成する．そのための準備を教材として準備する必要がある.

ただし，ここではプログラム言語の詳細な構文

を習得させるのではなく，実装の過程を理解させることを目標にすればよい．

③ 開発環境の整備

２．２で取り上げた開発環境の整備と同様であり，オブジェクト指向パラダイムを支援するＣＡＳＥの利用を行う．その中で，ソースコードテンプレートをもとに，オブジェクト指向プログラムに変換することが可能になる．これによって，分析・設計の検証を実施することができる．

５．おわりに

以上，手続き指向パラダイムをベースに，オブジェクト指向パラダイムに転換するための技術移転の在り方について述べてきた．

その結果，次のような事項について提案した．

・方法論や概念の重要性

・技術移転のやり方としては，講義から演習，つまり，「方法論からプログラミング」へのアプローチ

・実装への連携の必要性

・開発環境の支援

今後の課題としては，具体的なカリキュラムをベースにした教育の実施とその評価，カリキュラムに則したテキストの執筆などがあげられる．

参考文献

[1] DeMarco,T.: Structured Analysis and Specification, Prentice Hall(1979)

[2] Ward.P.T.: The Transformation Schema, An Extension of the Data Flow Diagram to Represent Control and Timing, IEEE, Transactions on Software Engineering, Vol.SE-12, No.2, pp.198-210(1986)

[3] Hatley,D.J.: The use of structured methods in the development of large software-based avionics systems, American Institute of Aeronautics and Astronautics Inc., Paper No.84-2595(1986)

[4] Stevens,W., Myers,G,J. Constantine,L.L.: Structured Design, IBM System Journal, Vol.13, No.2, pp.115-139(1974)

[5] Myers,G.J.: Reliable Software through Composite Design: Petrocelli/Charter(1975)

[6] Dahl,O.J.,Dijkstra,E.W., Hoare,C.A.: Structured Programming, Academic Press(1972)

[7] ＪＩＳハンドブック情報処理ソフトウェア編：プログラム構成要素及びその表記法X0128-1988, 日本規格協会 (1992)

[8] 国友義久：効果的プログラム開発技法，近代科学社(1983)

[9] 高度情報化人材育成標準カリキュラム：第二種共通カリキュラム，中央情報教育研究所(1993)

[10] 高度情報化人材育成標準カリキュラム：第一種共通カリキュラム，中央情報教育研究所(1994)

[11] 大学等における情報処理教育検討委員会：大学等における情報処理教育のための調査研究報告書，情報処理学会(1991)

[12] 河村一樹：構造化およびオブジェクト指向によるソフトウェア設計教育の検討，情報処理研究報告，Vol.92,No.57(1992)

[13] 土居範久編：オブジェクト指向のおはなし，日本規格協会(1995)

[14] Rumbaugh,J., Blaha,M., Premerlani,W., Eddy,F., Lorensem,W.: Object-oriented modeling and design, Prentice-Hall(1991)

[15] Booch,G.: Object-Oriented Design with Applications, Benjamin/Cummings(1991)

[16] Shlaer,S., Mellor,S.J.: Object-Oriented System Analysis:Modeling the Worlds in States, Prentice-Hall(1992)

[17] Coad,P. Yourdon,E.:Object-Oriented Analysis 2nd edition, Prentice-Hall(1991)

[18] Coad,P. Yourdon,E.:Object-Oriented Design, Prentice-Hall(1991)

[19] オブジェクト指向'95シンポジウム論文集, 情報処理学会(1995)

[20] 本位田真一, 青山幹雄, 深澤良彰, 中谷多哉子：オブジェクト指向分析・設計, 共立出版(1995)

[21] 本位田真一, 山城明宏：オブジェクト指向システム開発, 日経ＢＰ社(1993)

[22] 落水浩一郎, 東田雅宏：オブジェクトモデリング, ジャストシステム(1995)

[23] 日経コンピュータ編：最新オブジェクト指向実践ガイド, 日経ＢＰ社(1995)

[24] 河村一樹：入門情報科学シリーズ第６巻ソフトウェア工学、ソフトバンク社(1995)

[25] 河村一樹：第一種共通テキスト第７部ソフトウェア工学、中央情報教育研究所(1994)

[26] 河村一樹：ソフトウェア工学入門、近代科学社(1994)

Q. 篠崎「河村先生としては，ソフトウェア工学をどの程度教えているか？オブジェクトオリエンティッドにまつわる話題は？」
A. 河村「していない．企業に対してはやっているが，学校の方では教えていない．」

Q. 篠崎「聞いていて難しい話しがあったとか，学校で習っていない等の感想を」
A. 佐村「ほとんど知らないようなことばかり．授業では少し触りという感じでやった．普通高校の人はわからないという人が多かった．」
A. 川上「分かる部分もあるが，外国語を聞いてるような感じでわからない．」

Q. 和田「どの程度のレベルなら教えるか？」
A. 河村「ＣＳ学科なら教える．ソフトウェア工学は，提言されているのは大学院レベルである．学部レベルでも，本当のソフトウェア工学は無理だろうという指摘がされている．」
C. 君島「あんまりいらない．管理的なことは会社の先輩が知っているから．」
C. 篠崎「みんながみんな理論を知っている必要はない．」
C. 牧野「ソフトウェア工学は実務経験がないと難しい．」
C. 河村「アメリカはＣＳの学科を卒業した人がそのまま業界に入ってきて，日本のように文科系の人は入れない．ルートがない．」
C. 君島「新人教育は会社の用務を教えればよい．」
C. 河村「マニュアルも揃っている．日本みたいに新入社員教育をやる必要がない．」
C. 中園「なぜ手続型指向はだめなのかということをはっきりしないと企業側が進まない．富士通系の中で研究会があり，新日鉄のソフトウェア工学研究グループにオブジェクト指向について話しをしてもらうと，ソフトウェア生産モデルという話しが出てきて，ウォータフォールモデルがベースにあった．先程の“メーカー側の論理でウォータフォールモデルをだけで来ているから拒否反応を示している．”というのがあったが，なぜウォータフォードモデルはなぜだめなのかを出さないと先に進まない．

現在，ウォータフォールモデルで問題でになっているのは，まず‘短期開発’．従来，第三次銀行オンラインなどは４，５年かかって作っていた．今は半年，１年で作れといわれている．それから‘コスト’．いわゆるシステム開発自体のコストが非常に押さえられている．また，‘仕様確定’．ウォータフォールモデルは基本的に仕様確定しないと進められない．仕様確定がオープン時代にできるのかといったらまるきりできない．これらに対応するために，どうしたらいいのかという答えとして，オブジェクト指向がある．ソフトを作る側としては，いかに高付加価値のソフトを開発していかなくてはならないのかという話しがある．このまま従来のウォータフォールモデルで請け負って，開発していこうというソフトハウスは，何か特徴を持たないと完全に負けてしまう．その特徴として，短期開発で作ってしまう．機能とかよりも期間．期間限定の中で何を作るか．仕様未決定状態から作れるという柔軟性がないとだめである．途中の仕様変更に対応できるソフトハウスでないと生きていけない．

スパイラル開発に必要な技術は何か．仕様未決定の部分をどうやって作っていくのかというところで，プロトダイビング技術は必要．再利用技術がないと短期開発はできない．その中にオブジェクト指向技術というのがある．プロジェクト管理的なところでいろいろなオブジェクトが発生する

のでそれらをまとめる構成管理技術，共同作業を支援する技術，プロセス管理技術，以上６つの観点が必要になる．位置付け的にはスパイラル開発に必要な技術の中の一要素としてオブジェクト指向があるというのは，きれいに整理できている．

オブジェクト指向技術者に必要な技術ということで３つ挙げられていた．それは，抽象化能力，設計の一流プロ，プログラミング能力である．このうち，’抽象化能力’のない人は始めからやらない方がいい．しかし，一方では，実利的な抽象化能力を高めるものはない．」

C．河村「実利的なものではなく抽象化能力を高めるには，算数をやればよい．」

Q．和田「小学生とかに算数をやれといっても，取り組みにくいのではないか？」

A．河村「算数そのものをやれというわけではなくて，文章題を式に落とすということはどういうことなのかということを理解すればよい．」

C．中園「設計は一流のプロでないとだめ，適当な設計はむちゃくちゃになる．また，プログラミング能力も必要となる．オブジェクト指向をやる限りは，設計と実操作が一体になっていなくてはいけないので，一流プログラマーでないとやってはいけない．当然プログラム言語を含めて全部やらないといけない．

新日鉄でショッキングだったのは，２１世紀はオブジェクト指向だということで，最初６ヶ月ぐらいＣＳ関係をしっかりとやり，１年か２年は実務に付かせない．一流大学出て，それだけの教育をやっても，最終的にできる人はほんの数人しかいない．最初から向いていない人はやらせない方がいい．

企業側から言うと，コストがある．ＣＳを含めて教育するのにコストがいくらかかるか．それがないと企業をなかなか説得できない．技術者をどれだけの期間投入しないと全部クリアできないのか．単にオブジェクト指向を説明しても”関係ない”ということになってしまう．」

C．君島「富士通では，ＳＤＥＭという作業分解構造を提案している．これを，プロトタイピンプ型のスピーディなモデルに拡張する必要がある．オブジェクト思考の技術移転に関しては，まず手続き派とオブジェクト指向派のそれぞれの欠点をお互いに挙げてみることから始めればよい．ここから，本来の技術移転が始まるといってよい．また，今のオブジェクト指向についての欠陥部分（テストのしにくさ，性能の不十分性，作業標準の欠落）について，認識している必要がある．以上のことを前提に，オブジェクト指向短期講座を次のようにして開講することができる．それは，半日コースでいいが，旧パラダイム（例えば，ＰＬ／Ｓなど）の習得者を対象にする．つまり，オブジェクト指向に関する前提知識は不要である．ここでは，業務例の分析を中心に，各任務の設計・実現・評価をインクリメンタルな形で繰り返すような講座にすればよい．」

3．第2セッション

君島さんによる「教育工学この一年」について講演が行われた．君島さんは，富士通沼津工場に勤務されてから，最近富士通ラーニングメディアに移られている．このため，新幹線通勤をされているとか．私事になるが，いくつかの委員会や，著書の共著として，一緒にお付き合いさせて頂いている．

以下に，講演で使われたＯＨＰシートを記載する．その次に，質疑応答（Ｑは質問，Ａは回答，Ｃはコメント）についてまとめる．

教育工学のこの１年

君島　浩
教育工学担当部長
㈱富士通ラーニングメディア

教育事業でもうけるには：子会社化の１年
教育工学のこの１年：私個人のこの１年
第２セション向け：ＩＳＤから教科研究へ

担当講座：
- 教育の企画・教材開発・実施
- マルチメディア生産工学
- 文章技術とワープロ
- 発表技術とＤＴＰｒ
- ソフトウェア工学
- ソフトウェア品質管理
- ソフトウェア事業と教育
- 経営と教育

最近の講演・講義・出版の顧客・主催者

１月　富士通ラーニングメディア自身（以下略）
　　ソフトウェア技術者協会
　　沖縄富士通，沖縄国際センター
　　マネジメント社
２月　日本テクノセンター
　　旭化成
　　ＮＴＴラーニングシステムズ
３月　岩手県商工研修センター
　　大分統計談話会
　　富士通南九州システムエンジニアリング
４月　ニフティ
　　オムロン
５月　情報処理学会コンピュータと教育研究会
　　富士通）ＬＳ研究会
　　富士通北海道システムエンジニアリング
　　札幌品質管理大会
　　ソフトバンク
６月　日本能率協会
　　日本能力協会名古屋
　　日本テクノセンター
７月　林野庁森林総合研修所
　　島根県松江教育センター
８月　京セラ
　　米国研修ツアー事前講義
９月　米国研修ツアー主催
　　静岡大学教育学部
　　日本労働研究機構
　　ニフティ
　　富士ゼロックス総合教育研究所
１０月　ニフティ
　　東京商科学院
　　中央情報教育研究所
１１月　日本教育工学会
　　富士通神戸エンジニアリング
　　ソフトウェア技術者協会
　　新日鉄情報通信システム
１２月　専門教育出版
　　共立出版

教育事業：ＦＬＭの分社化まで

１９７０年代　ＳＥ部門教育事業部
　　ＩＢＭが手本．重役が重視．
　　新人から専任（上司は異動）
　　ＩＳＤの導入→ＣＤＥＭ
１９７７年　　ＦＩＥ創立．執筆・翻訳等
１９８０年代　ＴＯＷＮＳでＭＭ教材
１９８６年　　ＦＤＬへ分離（計算機系）
１９９０年代　ＳＥ拠点子会社に教育課
　　ＳＥはＯＳ部門の５倍の人数
　　１本部制でがんばったが限界

私のＦＬＭ出向まで

１９８５年　沼津ＯＳ部門教育課長　８人
　　ＯＳ子会社に教育専任者設置
１９９１年　教育部長　１０人
１９９３年　担当部長　３０人
　　講師が少なく，物足りない．
　　東京の仕事が増えた．
１９９４年　教育事業部 ——→ ＦＬＭ
　　ＦＤＬ ——┘
　　君島 ——┘

分社化１年後の現在のＦＬＭ

従業員　３７７名
売上高　９４下：５３億円⇨９５上：６０億

ＳＥ会社４１社中　売上高４位
　　利益高１位

商談が増加→外注・パートなどで対応
親会社ＳＥ教育比率の低下，しかし巨大収入
　教育事業部のときから収支黒字
ＳＥ以外・社外・情報技術外の増加
第２次パソコンブームと互換性
経済不況→教育のアウトソーシングとＣＤ化

教育会社の仕事

ＳＥ子会社の受講が減り気味
　委員会で親会社※重役が受講指令
予算手続き⇨商談状況報告⇨決※算
商談⇨台帳登録⇨企画書・見積り
　⇨開発⇨実施⇨請求書⇨反省・商談拡大
子会社化で諸手続き，管理が簡単になった
決起大会・研究発表会（学会活動は低調）
他社・国外との提携・契約交渉，講師派遣
富士通内国際教育共同体の連係
競合他社との情報交換・研究会活動
マニュアル・翻訳は品質・納期・価格がよい
　教育部門は工学化が遅れている
君島　ＦＬＭ社内教育・助言・講師引受け

奥道後からの私の１年

関西研修部でＭＭ商談の科目と作業法を取材
ＣＡＩＴのＭＭ教材開発テキスト
　パラグラフ概念とソニーカタログの影響
省庁系ヒアリングや商談で政策・経営へ視点
「新時代の研修技法」出版
兵庫県南部地震→東海地震ボランティア開始
「プログラム言語」で主題文方式へ転換
　１年１作品から多作へ
木下是雄氏講演で分類学・辞書学に関心
ＮＴＴ－ＬＳ，ＦＸＬＩ等競合他社で講演
ＣＡＩＴの調査研究委員会
（続く）

私の１年（続き）

大分統計談話会で論文作成技法を洗練
顧客で文章技法＋ワープロの統合講座が成功
ＤＢサーチャー教育を見学して発奮
リピータ顧客の誕生
日本能率協会セミナーなどで一般業種へ展開
ＭＭ教育などのカラー発表資材の充実
　カメラを購入（Ｋｉｓｓ，ティアラ）
ＩＳＤ・ＭＭと既存学問との統合
米国出張・国際作業標準作り
採用活動
大学が教育エンジニア学科設立へ
カラーノートＰＣ（ＣＤ付）を購入依頼

ＩＳＤ一般論から教科研究へ

日本語・語り・フィードバックなどの徹底へ

講師がマニュアルコンパイラから転換しない
　エンジニアリングからテクノロジへ
　表現は平易に．内容は高度に．
　用語定義・理論付け
　意外な共通部分の統合
　ツール改版追随から汎用計算機との共通性

「文章技法」共著，専門学校出版
　専門学校生が修得できるマニュアル技術
　従来：最終文面を分かりやすく
　今回：見出しや用語説明は辞書を見て書け

会社入門講座：日程管理と会話技術

学生は世間知らず→ビジネスマナー教育×
先輩は会社知らず→会社の仕事とは何か
日程管理とＯＪＴ（管理・指導される立場）
　任務管理の基礎
　ＯＪＴの基礎
　人事査定の基礎
　経理・販売の基礎
　会話技術（テクニカルカンバセーション）
　　まず単語．無駄のなさ．→会議・電話
　　→文章技術・発表技術の基礎
　　→会話型コンピュータの基礎
　→ハードウェアの基礎
　→オペレーティングシステムの基礎
　→プログラムの基礎
　→教育・作業マニュアルの基礎
　→経営の基礎
演習
　何かの文章を各自が書く
　速さ，質の個人差の測定と指導

文章技術とワープロ

文献検索（分類学，命名学，キー，集合論）
　表題，キーワード，著者などの必要性
　ライブラリと本
分析・重み付け
　→アウトライン
表題，見出し，用語説明
　表題にも構文規則がある．辞書から引用．
「まえがき」は範囲（ｓｃｏｐｅ）
　規則，職務分掌，リード文と同じ考え
章・節構造とハイパーテキスト，ＳＧＭＬ
文段と段落と主題文
単文・重文・複文
用語の慎重な選択と定義
　言葉のあいまいさ「分かるとは」
係り受けと記憶労力
箇条書きの構文規則，長所・短所
視覚媒体の控え目な活用
　モチーフ（キーワード，視覚キー）
構図の科学的説明
文書種類と目次構成の定石
ライブラリと目次の標準（雛形）
評価方法

演習講座の反応とフィードバック

発表技術とＤＴＰｒ

分析　視聴者行動分析
　　商品分析・既存作品分析
　　情報構造図
設計　標準体系・標準設計から設計
　　構成設計（構造的，階層的，網）
　　絵コンテ（構図定石，媒体選択）
実現　取材，執筆，編集
　　文面・色彩・写真・音声・動画
評価
講師準備　語り原稿，練習，動作，評価
　　質疑応答
実施
発表演習と反応・フィードバック

表計算講座

数値の前に名前が重要である
　集合・分類・順序，命名・改名
　マトリクス：分類Ａと分類Ｂ，名前と属性
名前と数値
　領域，集合，状態遷移
作業分析（情報の流れ，保管，所要時間）
コンピュータ化と人間に残る作業
だれがどうやって作るのか
だれが何の作業に使うのか
標題の付け方
表の集合の体系
情報入手（だれが，どこから）
表の解釈の仕方
　⇨エキスパートシステム（マクロ）
表のデザイン
　罫線，間隔，色彩

「ソフトウェア工学の欠損部分と拡張部分」

－日程管理とレビュー
－個別品質管理
－設計書を捨てない
＋複数顧客分析
＋会話型マルチメディア設計
＋使用性レビュー・使用性テスト

Q. 篠崎「（東海地震ボランティアの時）ワークステーションは？」
A. 君島「パソコン，ワークステーションで，１割壊れた．」
C. 篠崎「ＮＥＣソフトウェア工学の事例で，５階建の屋上の貯水槽が割れて５階のフロアが水浸しになり５階のフロアでは，回線をすべて取り替えた．」

Q. 中園「（儲かるのはＣＤに作業マニュアルを圧縮しているのもある）関係ないのでは？」
A. 君島「教育とか作業マニュアルにお金，人を賭けているところもあるけれども，検索が出来ない．コピーすると高くつく．場所，倉庫が大変だ．」

Q. 和田「オンラインマニュアルではやっていない？」
A. 君島「我が社の教育というのはマニュアル部隊でやっている．」

Q. 中園「お客様でオンラインマニュアルをしたいという場合は？」
A. 君島「作業マニュアルは教育部門でやっている．」

Q. 牧野「分散化する前は他の会社がやっていた？需要がなかった？」
A. 君島「営業に歩いていなかった．分散化して営業マン営業ウーマン１０人ぐらいおいた．」

Q. 篠崎「“何かの文章”の‘何か’とは何？」
A. 君島「自分の仕事の説明とか，新人だったら趣味，中堅社員だったら今の仕事の紹介．」

Q. 中園「そこに企業の目的は入らないのか？」
A. 君島「入る．ここで教えるのは，企業が社員にどう貢献していくのか．給料，教育，作業環境．企業の目的は弊社側が考えることで，社員はいかに給料を上げるかということを考える．また，企業の目的は収益を上げることではなくて，物価を下げることだと教える．」

Q. 中園「時間配分はどういう基準でやっているのか？」
A. 君島「書きながら，教科書の量を見ながら，考える．演習はそれなりに時間かかる．」

Q. 篠崎「見積もりと実際の時間のずれは？」
A. 君島「わからない．最後にうまくつじつまを合わせる．」

Q. 河村「講座の設計で，時刻が詳しく入ったものは実際にやってみる前に作られたものか？」
A. 君島「教科書があるので，１ページ５分と見積もっている．」

Q. 和田「仮にかなりずれ出したら，こっちに合わせて強制的に終わっていなくても切るか，それとも現実の方を重視してずらすか？」
A. 君島「ずらす．休み時間に相談して．」

Q. 平山「ワープロ講座ではなくて，文章技術入門講座というのは，書く内容というのはみんなが同じものではなくて個人のものなのか？」
A. 君島「新入社員なので，自分の特技とか学生時代の生活，女性だったら化粧品をどう買ってどう使うかとか，ラクビーの人がどういう部活をしているか，これらを１ページ書いてもらう．」

Q．平山「普通のワープロ講習だとみんなが揃って同じものを書くのだが，内容を各自に任せるということで，時間的にも技術的にも個人的に満足できるものが出来るのか？」
A．君島「早い人は直していたり，それなりに時間を潰している．かなりスピードに差はある．」

Q．和田「ワープロを使って，文章を考えて作って，生徒にそれをやらせながらワープロを学ぶ，あるいは，文章を書くのを学ぶ．そういう理解になる？」
A．君島「アウトラインプロセッサは今のは使い物にならないので，それがよくなったら設計ツールを使いたい．文献検索のところはできたらパソコン通信でやりたい．そこまではまだいっていない．中堅社員の時のテーマは，お客様との苦情その事例を持ってこさせたり，自分の仕事を説明しなさい，というふうにした．」

Q．篠崎「日本の大学教育で一番欠けているのは日本語講座だと思う．日本語を大学でもちゃんと教えるべきではないか？」
A．君島「テクニカルライティングというのを教えられるのは文科系にはいない．理工系にしかいない．その人の数が少ない．さらに緻密に教える人がいない．」
C．平山「大学で教わった文章理論というのは「いかに単語を難しく表現させるか」だった．テクニカルライティングは「いかに単語を優しく分かり易く」というものだった．」
C．篠崎「文学の論文は，いかにボリュームを付けるかというもの．理工系はいかにコンパクトかというもの．」

Q．河村「横書き文字に，カンマ・ピリオドは使わないのか？縦だと白抜きなのでは？みんなごちゃごちゃ使っている．」
A．君島「パンフレット全体のレイアウトもその意識になっていない．思い付きで書いている．」

Q．河村「レイアウト学みたいのはアメリカの方では確立されているのか？」
A．君島「あると思う．ソニーの場合，作業標準があると思う．アメリカ向けと日本向けは配置が違う．アメリカ人は図が嫌いなので，文章を先にする．日本人は文章が嫌いなので，図を先にする．アメリカ人でもエリートは文章が中心，エリートではない人は日本人と同じ図が多い．」

Q．和田「君島さんがやっているような講座は，口頭ではなければ実現出来ない，しにくいと考えるか．それとも話すのと同じ内容のテキストを作って，それを読んでもらえば実現できるか？」
A．君島「８割がたはできる．」

Q．和田「考えながら話しているアナウンサーは，考えている時間は黙っている方が一番いいのか？」
A．君島「その方がいい．ただ，何か談話をしている時に”あのですね”というのがあった方がいい場合もあるのかなと思う．これから聞きますよという意味で”あのー”とか」．
C．篠崎「訓練の場でストップウォッチを持って，１分間に非言語を何回言うのか測った．だいたい，１０回を越えると耳障りになる．ちなみに６回ぐらいまでだったら問題にならない．また自分が考えているのではなくて，相手に考えさせる間に非言語を使う人もいる．」
C．和田「全く何にも話さないと，こっちが話そうと準備していても，相手が話してしまう．これを防ぎたいという流れもこの場合あるのでは．」
C．篠崎「話すスピードは，昔はゆっくり落ち着

いて低い声でというものだった．今はテンポよく話すというのがベースになっている．活気を付ける．」

Q．平山「プレゼンのビデオとかやった本人に見せているのか？」
A．君島「やっている．」
C．篠崎「実際，トレーニング現場で非言語をなくす特効薬というのは‘気付く’ということである．」
C．平山「自覚がないと直しようがない．」
C．篠崎「気付いた人はほとんど0になる．今まで一番多かった人で1分間に「え」というのを29回言った人がいる．それから3日後には完全に0になった．12，3回だった人は最終日でも12，3回のまま．」

Q．篠崎「君島さんから見て，教育現場のインストラクターは若いのがいいか，熟練がいいか？熟練とはどの辺で熟練とみなすか？」
A．君島「3年目ぐらいからはマニュアルコンパイラから離脱して欲しい．上級者に望まれるのは，入門講座．インストラクション10年ぐらいやった人がいろんな講座を体験して，それを組み合わせて，しかもそのエッセンスを収容する．入門こそ大ベテラン講師がやるべき．今は逆になっている．」
C．篠崎「新人教育の教材とか入門は若手で出来るだろうとなっている．」
C．君島「若手の方が，自分が理解に苦しんだ体験生々しいから出来るだろうというのが一般的．」
C．篠崎「アメリカでは入門講座とか，手ほどきというのは一流の人が書いている．」
C．牧野「（講師を呼ぶ場合）メリハリのある講師．1～2時間の間で大事なところを明確に示してくれる人．」
C．中園「理論だけはだめ．事例をちりばめた先生というのがインパクトは大きい．生徒もこういうことをやるとこういう失敗につながると分かる．生々しい事例があるといい先生だなと思う．」
C．篠崎「今実践しているのは，参加型教育．君島さんの中に働きかけるというのがあったが，それが出来ないインストラクターはマニュアルコンパイラの域を出れないのではないか．発問出来ないインストラクターは，マニュアルそのものになる．小演習，演習，できたら個人演習の後，グループ演習を入れるべきである．1つの事例として，プロジェクト管理2日コースは丸2日講師が話しっ放し，それも何の働きかけもなく分厚い本を読んでまわるだけ．受講者は参加していかない限り理解していこうという気にはならない．」
C．君島「それも管理できる．講義中心だったら，始めに設計しておく．今のインストラクターはやらないから品質管理ができない．」
Q．中園「現場の事例が必要な講座にはどうやって対処しているのか？」
A．君島「事例はデータベースにあるが，今のところは上級のところとかは技術部の人に頼んだりしている．専門家は下らないところを飛ばしたりするから，教育が取材してきたもののほうがよい．」

4．第3セッション

平山さんによる「中高年に対するＰＣ教育の現状」について講演が行われた．平山さんは，山一情報情報システムでいくつかの研修コースの講師として活躍されているバリバリのＯＬである．エクゼクティブの研修もあるらしく，いろいろと面白い話が飛び出した．

以下に，講演のためのポジションペーパーを記載する．その次に，質疑応答（Ｑは質問，Ａは回答，Ｃはコメント）についてまとめる．

ポジションペーパー

山一情報システム株式会社
平山　順子
TEL:0474-37-3383
FAX:0474-37-3366
E-mail:junko.hirayama@yis03.yis.co.jp

中高年に対するＰＣ教育の現状

ＰＣ教室の企画・運営・実施を担当してから、はや３年目となりましたが、この間、約２０００名の方々が各講習会に出席しています。ここで、実際に、私が講習会の現場で受けた印象、またお客様からいただいたアンケートの声をまとめ、中高年の方々のＰＣに対する現状をまとめてみたいと思います。なお、私個人の解決作も添付いたしましたが、できれば、この研修中にとりあげていただき、皆様のご意見を得られたら幸いです。よろしくお願いいたします。

問題点

① ＰＣに対し、「難しい」という先入観がある
② マウス操作ができない
③ キーボード入力ができない
④ カタカナ用語に慣れない
⑤ すぐ忘れてしまう
⑥ プライドが先行し、質問する機会を失ってしまう

問題点の現状

①ＰＣに対し、「難しい」という先入観がある
「今から始めても遅い」という思いこみが邪魔し、「やる気」さえも失ってしまう。いかに、講習のはじめに、この先入観を取り外し、「やってみよう」という意欲を興すことが大事なように思う。

②マウス操作ができない

ＧＵＩの理念がそのまま形になった、このマウスだが、実際に講習に立ち会ってみると、このマウスでつまづいてしまう人が多い。クリック、ドラッグといった操作はなんとか行えるが、ダブルクリックが以外に難しいようだ。どうやら、２度目にボタンを押すタイミングがうまくつかめないらしく、ほとんどの人が「クリック２回」といった操作になってしまっている。

③キーボード入力ができない

実際に個々のアプリケーションの実習になると、文字入力が必須となる。しかし、キーボードから、目的のキーを探すのが大変である。頭の中に、打ち出したい文字が形になっているのに、なかなか画面上に表示出来ないギャップで、いらいらする人が多い。また、ローマ字入力の場合は、特殊文字（「を」「っ」など）の打ち出し方がわからず、そこで止まってしまう。

④カタカナ用語に慣れない

ＰＣ教育の場合、普段聞き慣れないカタカナ用語が連発してしまう。よって、講習中、それがどの場所なのか、どのような操作なのかわからず、ついていけなくなってしまう人がいる。

⑤すぐ忘れてしまう

演習形式の講習だと、講師と一緒に作成すれば、なんとかついていけるのだが、自習形式の演習問題の時間になると、ほんの１、２時間前に行った操作でさえ、思い出すことができない。

⑥プライドが先行し、質問する機会を失ってしまう

教室契約の講習会の場合、同じ会社の人が集団で受講するといった形になるが、この場合、なかなか素直に質問をする機会が少なくなってしまうよう感じる。しかし、管理者コースなど、同じ役職の方が集まっている場合は、かなり積極的に質問を行っていたようである。

解決策（案）

①ＰＣに対し、「難しい」という先入観がある

先述したが、講習のはじめに、この先入観を取り外し、「やってみよう」という意欲を興すことが必要。また、身近な例をとりあげ、親近感を抱いてもらうよう、講師

側の努力が必要。

②マウス操作ができない

コントロールパネル等で、ダブルクリックの間隔を調節。必要があれば、win.iniからも設定を行う。（実施）また、現在、３つボタンマウス（１つのボタンがダブルクリック専用ボタンとなっている）が販売されているので、それを使用するという打開策もあり。だが、汎用機ではないので、考慮が必要。

③キーボード入力ができない

簡単な例文を繰り返し一緒に打つ。とにかく慣れが必要。また、ローマ字入力の場合、ローマ字表を別紙として作成し、参考にしてもらう。なお、現在５０音順キーボードが市販されており、こちらを使用するという打開策もある。しかし、前述した３つボタンマウスと同様、汎用機ではないので、考慮が必要。

④カタカナ用語に慣れない

単語表を作成し、別紙として配布。また、何度も繰り返し説明を行い、耳に慣れてきたようなら、その単語を使用する。（ex:「マウスの左ボタンを１度押してください ークリックです」）

⑤すぐ忘れてしまう

講習の合間に小演習を取り入れ、紹介後すぐ各自復習できるようなカリキュラムにする。また、演習問題は、なるべく、講義の時に作成したものと同じ形式にし、まったく同じ手順で作成できるよう（テキストを順に見ていけば１人で完成できるよう）考慮する。よって、テキストの内容も吟味しなければならない。

⑥プライドが先行し、質問する機会を失ってしまう

先述したが、管理者コースなど、同じ役職の方をなるべく集めるようにし、うちとけた雰囲気で講習に参加出来るよう、コース設定を考える。

受講者の声（アンケートより抜粋）

- ◆ レベルが違う人（高い）がいたため、若干迷惑である。今回は最低レベルで参加。（部長）
- ◆ パソコンに本格的に触るのは初めてなので楽しかった。思っていたよりも出来そう

（年の割には）なので、今後とも慣れ親しみたい。初心者用にカリキュラムが組まれていたので有り難かった。（部長）

- 経験不足と能力不足との年の為、質問回数が参加者の中出一番多く迷惑をかけました。だが、非常に楽しかったので有り難うございました。（部長）
- 今後とも時間外でも良いから続きをやってくださると嬉しいと思います。（部長）
- 自分自身でも日頃から勉強して行きたいと思います。（部長）
- 用語がなかなか頭に入らない。（部長）
- 演習が難しい。説明資料と同一パターンなら資料をひもとけるが・・・（部長）
- 昨日も同じような事を述べたと思いますが、講師のお話しにつれて操作することは何とかなりますが、ひとたび自分だけで操作するとなると全くダメになってしまうしまうのが現状です。（部長）
- ミス操作をしたときの元の状況への戻りかたが分かりづらい。（次長）
- キーボードの使い方がよくわからないので苦労した。（部長）
- まだまだ簡単にはできないのか？いってることがわからないところもある。（部長）
- パソコン教室は初めてであり、大変、有意義でありました。インストラクターもわかりやすい説明に努力されていたと思います。（部長）
- 「たいへん勉強になった」の一言につきる（専務）
- かねてよりＰＣを使えるようになりたいと念願していたが、どうアプローチしてよいかわからず困っていた。今回のイントロダクションセミナーは大変分かりやすく、かつ実務的で、なんとか続けていける自信を得ることができました。感謝しています。（社長）
- ワープロのプログラム、技術の進んでいるのに驚くと同時にたいへん面白く、今後活用したいとの意識をもつことができた。本セミナーは今後のＰＣ活用に向けて考え方を固めるのにたいへん役にたった。講師のお嬢さん方の熱心な仕事ぶりに感心。（社長）
- 初心者は専門用語になれるまでは時間がかかるので、できるだけ平素な用語を使って欲しい。（常務）
- ワープロ経験があるのでわかりやすかった。機能の素晴らしさにびっくりした。（常務）
- 初めてなだけに「こういうものか」という程度の理解だった。自分でくり返し練習してみたい。（常務）
- 自分で何回も練習してみる必要があると思います。（常務）

Q．河村「ワードとエクセルは何故選んだのか？」
A．平山「こちらの方は，弊社の方でＷＩＮＤＯＷＳ３．１が導入されて，今までは一太郎や１－２－３を使っていた．だが，ＷＩＮＤＯＷＳの導入ということで，じゃアプリケーションの方もマイクロソフトの方のエクセルとワードを使いましょうということで，弊社の方の社内標準で．」

Q．河村「一太郎のＷＩＮＤＯＷＳ版というのは入っていないんですか？」
A．平山「入ってないですね．弊社の方は，Ｙ社のシステム部門担当という形になっているので．グループ会社にパソコンを導入しましょうとすると，ではうちの社内標準のＷＩＮＤＯＷＳとエクセルとワードを入れましたからサポートをしますといっても，アプリケーションはあるＡ社では一太郎，Ｂ社では１－２－３という関連会社関係の中で，分かれていますとサポートの方のうちが大変になりますので．これから導入しようというような会社，これは関連会社ですが，うちと同じような設計書を導入している．ですから，まあコースの方にもエクセルとワードといったものでやっている．外部の方に向けてだとそのようなことはいっていられないのですが．」

Q．河村「コースは何人分ぐらいですか？」
A．平山「教室の方はパソコンが１２台用意してあり，基本的には１０人です．ＭＡＸ１２人ですね．」

Q．中園「その講座の履行者の条件は？」
A．平山「それはまあ，状況によって変わるかと思うのですが．Ｙ社の方は，自分の希望で来ていらっしゃると思います．逆に，普通の外部や関連会社系のパソコン教育の場合，春だと新人教育等が多くなるので，これはほとんど行ってこいと言われて来ていると思う．」

Q．中園「基本的には管理職の人ですか？」
A．平山「いいえ，若手の方がやはり多いです．人数の割合は１回１０人の中で８～１０人がまだ若手．しかし，最近になって管理職の人の参加率が増えてきました．エグゼクティブコースというような形でやってほしい．マンツーマンでやってほしいなど最近需要があるんだなあと思います．」

Q．篠崎「全然関係ないことなんですけど，Ｙ社またはＹＪ社さんで管理職の人は，コピーを自分でとる人と部下にコピーを依頼する人とどう振り分けられているか？」
A．平山「ＹＪ社は，私はあまり頼まれたことがないですね．百部，千部するといったときは別ですけど．」
C．橋本「単純なコピー，お茶くみなどは，管理職と平との区別はないですね．」
C．篠崎「何故それを聞いたかというと，最近コピー機高機能，ホッチキスがついたり，拡大，縮小など，実はあのオペレーションが難しいと思っているという人はパソコン難しいかなっと思って．で，自分で触りに行こうとしなくなるんですよね．」

Q．河村「ＯＡ基礎コースの方にタッチタイピングを導入したらどうですか？」
C．君島「題材を見るってことは大変によくないことで，題材見てキー探して，変換ていうのは．あの，文章にまず題材入れちゃって，その下に同じ文入れさせるとずいぶん楽です．画面とキー

ボードだけで，一番いいのは学習ソフトは，キーボードの図まで画面にでる．全部そこで目も動かさないで出来る．」

Q．中園「このような講習によって現場へのＰＣ導入が早まると思うか？」
A．平山「はい」

Q．牧野「マッキントッシュ系の講座はないのか？また，Ｗｉｎｄｏｗｓ95の対応は？」
A．平山「ＤＯＳ／Ｖが中心になっている．Ｗｉｎｄｏｗｓ95については，これからの段階にある．」

Q．君島「教育部門は事業化すべきではないか？」
A．平山「その方向性はある．」
C．君島「管理者を戦略的な視点から教育することによって，プレッシャを与えよ．また，同業他社の事例を取り上げ，マーケティング戦略の面から考えさせる必要がある．」

参加の感想

最後に，本ワークショップを参加した感想についてまとめる．ただし，順不同である．

【橋本　勝】

①固い感想

吹き抜きる風が晩秋を感じさせる越後湯沢での，今年のワークショップ．話題も多方面に渡り，有意義な３日間でした．ここで，討議された話題について，私自身が共通部分と感じた内容について簡単にまとめます．

３日間の話題における共通項は「どのような教育カリキュラムが適切であるか」ということであった．その共通解の１つとして，どのようなニーズをもった人々が参加している（参加する）かという「現状分析」が重要であると感じた．大学教育，新技術移転，中高年教育・・・，いずれの講座を行うにあたっても，講座に参加する人の保有知識，置かれている立場，講座参加の背景など，様々な要因が重なりあって講座に対するニーズが構築されるからである．教育もニーズありきの代物である．

しかし，実際の業務上においてこれらの活動を行うことは難しい．特に大学教育や，一般募集型の事業教育においては，不特定多数の聴講者を対象とするのでなおさらである．「教育」が古くて新しい問題として扱われているのはこの点も関与しているのではないだろうか．

一方，前出の局面と比較すると，企業内教育は参加者のニーズを事前にまとめることは幾分容易である．しかし，私自身はカリキュラム構成の検討そのものに気を奪われている．講座に参加する人々の現状についての調査／分析が不十分であったことは否定できない．企業内教育を担当している私にとってこの点を認識できたことは，本ワークショップでの大きな収穫である．

昨年は「教祖」（昨年の参加者しかわからないが）という単語をワークショップのお土産として持ち帰ったが，今年は「現状分析」を持ち帰りたいと思う．

最後になりますが，ワークショップ参加の諸先輩方にはいろいろご助言を頂き，ありがとうございました．今後ともよろしくお願いします．

②柔らかい感想

社外の色々な立場の方のお話しが聴けるのは，本当に勉強になります．何がと聴かれると？？？となってしまいますが，自分が勉強しなければいけないこと，強化していけばよいことなどが，浮き彫りにされてくる感じがします．

昨年は初参加だったので緊張した３日間でしたが，今年はリラックスして参加できました．このまま来年以降も，１参加者として参加できればよかったのですが．

【君島　浩】

事務局を含めて，若い人が参加したのが良かった．

ホテルは天童と１，２位を争う．美観はだいたい天童より上．風呂へのアプローチは天童．料理の量，質，美観は天童と同等．山場（メインディッシュ）が曖昧で平坦なのが難．

風呂　タオル持参が不便．湯船は天童より上．

青山先生のは博識なソフト工学研究者だったのでグレイドが高い．教育工学を勉強してくれるとなお良い．

河村先生のは相変わらず情熱的でためになる．参考文献一覧もためになった．　平山さんのはおそらく発表されたのが初めてのテーマなので価値が高い．いろいろなテーマを若い人が発表してほしい．

【牧野　憲一】

☆序章

初めて湯沢の地に降り立つ．紅葉の山々の合間に雪山が見える．秋本番か．同じ列車だったと思われる河村先生はタクシーでホテルへ．私と山一情報のお二人は歩いてホテルへ．だらだらした登り坂は距離以上に息を切らせた．しかし，初参加の平山さんと楽しく会話しながらだったのが救いである．

ワークショップでの自己紹介において，前述の平山さんの豹変振りに驚く．インストラクタの経験からか彼女の性格か，人前に立つと話し方が堂々としている．ただただ，すごい．

☆本章

青山先生のセッションでは，元企業経験者らしく大学・企業の観点に加え，現状・未来の姿に分類され明快で理解しやすい．Just-In-Time(学びたい時に，学びたいものを，学びたいほど)が印象的である．

河村先生のセッションでは，オブジェクト指向推進者の立場として，従来の手続き型との違いを短時間に明快に説明されていた．くしくも，中園さんも企業の立場として，オープン化の流れで激変する状況の中で，納期（Delivery)やコスト(Cost)に対する顧客の要求に応えるためには，既存のリソースを最大限に活用
することであり，そのためには従来の部品再利用とは違ったオブジェクト指向が効果的であると，企業の立場と一致した．しかし，浸透が遅れているのは事実であり，導入を前提にどうすれば導入できるかを真剣に検討していないからであるとの君島さんの意見など，メンバの立場は違えどなぜか？一致していた．私も同
感である．開発標準などの整備を行い，積極的に導入の姿勢を示していきたい．

君島さんのセッションでは，ＩＳＤ普及活動，教育は商売になるか，文章技術など盛沢山であった．それにしても，君島さんの着実な活動には頭が下がる．以前「教育担当者の言うだけで，行動力がない」と言われたのを今でも覚えており，毎年何か成果を出そうと努力しているが，君島さんのコメントは常に励みにな
る．

最後のセッションは平山さん．２年半のインストラクタ経験からのコメントはさすがに生々しい．リテラシー教育が如何に大変かよくわかった．一度平山さんに習ってみたいと思ったのは私だけだろうか？

☆番外編

今年はＳＥＡＭＡＩＬに掲載しょうとの意気込みから，河村先生が学生を連れてこられた．私も含めておじさん達の集まりで，何かと大変だったかと思われるが，テープ起こしに期待したい．頑張ってください．

☆終りに

ワークショップでは各セッションも大切であるが，それ以外の時間すべてが大切である．私の特訓の成果も披露できたし... メンバ間の情報交換はかけがえがない．私にとっては今年も収穫があったが，来年は１０回目と区切りの年，新実行委員長の基でどのように開催されるか，興味津々です．

【篠崎　直二郎】

１．全体について

・参加者が，１３名（常時１２名）と少数で議論しやすかった．

・記録係の学生さんや若手の方が参加して，ちょっとは若返った．

・堅い話ばかりでなく，有意義な意見交換ができた．

・会場施設は立派であった．経費は大丈夫だろうか？

・産業界だけでなく，大学関係や学生もいて参

加者の幅があった.

・青山（先生）さんの参加（滞在）時間が短くて残念.

・実行委員長とプログラム委員長の代替わり（若返り？）ができた.

・紅葉も雪も見れて良かった.

・駅前にヘギ蕎麦屋，駅構内に日本酒ミュージアムがあり満足.

2．施設会場について

・費用はともかく，グレイドの高い会場である.満足.

・従業員のしつけ（教育）も行き届いており気持ちがいい.

・仲居さんの対応も良かった．商品知識に欠けていた.

・フィットネスおよび温泉プール（有料）があり，リフレッシュできる.

・駅からの歩き１５分は上り坂で良い運動になる．ちょっときついかな？

・客室の冷蔵庫がカラッポだったので戸惑った.

3．討議内容

・青山さんの，Just in Time と未来の教育のあり方は興味がもてた.

・河村先生の，ＯＯについては「とりあえずやったえ」が印象深い.

・君島さんの，ご活躍には頭が下がります．今度は反論でもしてみたいです.

・平山さんは，定常メンバになってくれたら楽しいだろうな.

・今回は，発表者に対する集中討論という形を取った.

・人数的にも，ほとんどの人が討議に参加し，盛り上がった.

4．その他

・最終日が，ミッキーマウスの誕生日という事でショートケーキを食べた.

・ついでに，11/18は，和田ベン先生もお誕生日だそうです.

・若手インストラクタの平山さんが，「すてきなオジサマ方？」の集まりとの感想を述べ，複雑な心境です.

・今回は，酒どころ新潟ということもあり，ちょっと多めのお酒かな？

・全くお酒の飲めなかった某オムロン系列の人が，訓練をすれば上達することを証明してくださった.

・来年が１０周年に当たり，新体制で頑張って欲しい.

【和田　勉】

すでに，数回この教育ワークショップには参加していますが，各回のテーマとそれに関する議論もさることながら，毎回おなじみの人達や，各回初めて参加する方々と定期的に（夜まで）話し合い，連係をとりあうことの意味が大きいです.

追記　私の誕生日を祝って頂いて有り難うございました.

【川辺　正明】

今年も，やはり愉しいワークショップでした.水着を持参せず，悔しい想いも楽しみのうち．議論では，君島氏の「ぐずぐず言わずにやること」と河村先生の「オブジェクト指向」が特に記憶に残ります.

また，全体を通すと「おじさん」を自覚したことに尽きます．来年は，教育ワークショップ１０周年，新たな企画と，多くの参加者を楽しみにしています.

【中園　順三】

感想文はもう少しお待ちください．（←そのまま来ませんでした．．．．！）

【河村　一樹】

ようやく念願のワークショップ報告書を作成し終えることができた．今回は，私のゼミ生２名を，

この報告書を作成させる目的でワークショップに参加させた．彼女達のおかげで，ようやくこの作業を終えることができた．といっても，肝心の講演内容までテープ起こしをすることができなく，ちょっぴり残念．まあ，何となくＳＩＧＥＤＵのワークショップの雰囲気を味わう程度のレポートになってしまった．

それはそうと，今回は，美女と野獣のごとく（?）「ギャル」（現役バリバリのＯＬ１名＋現役女子大生２名）対「おじさん」（その他大勢）という関係の中で，いままでのＳＩＧＥＤＵにはない雰囲気に包まれたワークショップになった．実は，今回学生を参加させるにあたり，私としてもいろいろとその人選に悩んだ．我がゼミの中でも最も飛んでいるピチピチギャル（ボディコン＋ヒール＋ワンレン．．．）を参加させてみようか，あるいは，まじめで勤勉な女学生（ジーパン＋シューズ＋ショートヘア．．．）を参加させようかなど．結局，参加者にあまり刺激が強すぎてもまずいと思い，比較的社交的で普通のギャルを選んだわけである．それでも，結構インパクトがあったようで，あの温水プールでの出来事に象徴されるような状況になった．

それはそうと，ワークショップ自体は，昨年と同様に結構盛り上がったのではないだろうか．教育というテーマは，誰でもが参加して討論できるものであり，そういった意味ではもっと参加者が多くてもよかったかもしれない．今回のワーキショップのテーマは，「教育を事業として成功させることができるか？」という大風呂敷的なテーマだったが，これについて討論するよりも，身近なテーマについて話し合ったような気がする．私は，オブジェクト指向パラダイムに関する技術移転の教育に関心があるので，このテーマで講演を行った．反響はどうだったのだろうか．

これで，ようやく我々の仕事を終えることができた．実行委員長とプログラム委員長の交代が行われることになり，ほっと一安心といったところか．でも，それほど仕事はしなかったけれども．なにはともあれ，篠崎さん，実行委員長ご苦労様．また，来年からは新しい委員長のもとで，新しい形でワークショップが開催されることになるでしょう．来年度は，ぜひ皆様のご参加をお待ち致しております．一緒に飲み明かしましょう．

終わり（13 February，1996）

情報処理技術者試験のあり方を問う

河村 一樹
(尚美学園短期大学，情報コミュニケーション学科)

1. はじめに

昭和44年度から実施が開始された情報処理技術者試験は，いくつかの試験区分を追加しながら拡張され発展してきた．それにともない，情報処理技術者試験を受験する人数も増加の一途をたどってきた．とくに，第2種の受験者数の累計は，現在までに約400万人(そのうち合格者は約40万人)になっている．このような結果，情報処理技術者試験は，通産省認定資格として社会的に広く認知されることになった．

企業では，社員の資格取得を支援したり，社内の給与体系に反映させる制度を実施し始めた．それとともに，多くの新入社員教育には，第2種に準拠したカリキュラムが組まれるようになった．また，学校では，専門学校が中心になって，資格取得のための学科やカリキュラムを作り始めたり，合格者数を競うようになってきた．

一方，コンピュータ業界の技術革新は激しく，情報技術の変革は著しい勢いで進められている．このため，情報処理技術者として求められる人材像にも変化が生じてきた．そこで，通産省の産業構造審議会情報産業部会の中に，情報化人材対策小委員会が新たに発足した．その中で，情報処理技術者試験の見直しが進められ，平成6年秋期から新制度のもとに試験が実施された．

筆者は，情報化人材対策小委員会のカリキュラム委員会の一つである「第一種情報処理技術者(当初は，中級情報処理技術者)専門部会」に，委員として参加する機会を得た．このような経緯から，今回の新制度の情報処理技術者試験に対しては，いろいろと考えるべき点が多い．そこで，本稿では，新制度試験のあり方という視点から，今後への提言を含めて述べることにする．

2. 新制度確立までの経緯

産業構造審議会情報産業部会情報化人材対策小委員会は，平成4年12月に中間報告を，平成5年5月に最終報告を，それぞれ発表した．ここでは，それらの報告書[1][2]をもとに，新制度確立までの経緯を明らかにする．それとともに，筆者の参加した第一種カリキュラム部会での活動状況について述べる．

2.1. 中間報告

中間報告は，次の四項目によってまとめられている．

・情報化の新たな段階
・新情報革命の担い手として求められる人材
・情報化人材育成策の基本的方策

ここでは，それらのうち，情報処理技術者試験に関連した事項についての提言をまとめる．

2.2.1. 人材像の類型

いままでの大まかな職種分けによる人材像(システム監査人，システムアナリストに相当する特種情報処理技術者，オンラインエンジニア，システムエンジニアに相当する第1種情報処理技術者，プログラマに相当する第2種情報処理技術者)ではなく，専門分野に特化した技術者を類型することを目指している．それによると，次のような区分による類型がなされている．

・情報システムの企画，設計，開発，運用及び評価に関連する人材

システムアナリスト(システムコンサルタント及びシステム監査人)，プロジェクトマネージャ，アプリケーションエンジニア，プロダクションエンジニア，テクニカルスペシャリスト，システム運用管理エンジニア

・技術者教育，利用者教育に関連する人材

教育エンジニア

・システムソフト及びマイコン応用システムの開発に関連する人材

デベロップメントエンジニア

・利用者側で情報化をリードする人材

システムアドミニストレータ

2.1.2. キャリアパスとカリキュラムの連動

それぞれの人材類型に対応したキャリアパスを明示することによって，情報処理技術者としての技術目標を確立する必要性を主張した．それとともに，各キャリアパスに対応したカリキュラム編成を提案することも唱えている．

2.1.3. 試験制度の見直し

提示されるキャリアパスを踏まえた情報処理技術者試験に改訂すべきことを提言している．また，その中には，試験における一部免除の導入の検討まで含まれている．

2.2. 最終報告

2.1 の中間報告をもとに，最終的な報告書が作成された．それは，次の四項目によってまとめられている．

- ·情報化人材育成システムの確立のための標準カリキュラムの策定
- ·各教育機関における教育の役割と充実のための施策
- ·一貫した育成システムの形成に向けた試験制度の見直し
- ·情報処理技術者育成における国際化への対応

ここでも，それらのうち，情報処理技術者試験に関連した事項についての提言をまとめる．

2.2.1. 標準カリキュラムの体系化

中間報告で提唱されたキャリアパスに対応したカリキュラムを，ここでは「標準カリキュラム」と命名するとともに，その体系化を明らかにした．

キャリアパスの連携から見ると，入社１年～３年程度の技術レベルを想定した第二種共通カリキュラムに，入社後３～５年程度の技術レベルを想定した第一種共通カリキュラムに，その上に専門分野別の高度情報処理技術者別カリキュラムを，それぞれ設定している．また，これらのカリキュラムとは別枠に，新しく初級レベルと上級レベルを分けたシステムアドミニストレータ育成カリキュラムを設定している．

これより，標準カリキュラムでは，コンピュータ開発者とコンピュータ利用者を明確に分けている点 (完全な二極化) が強調されている．また，いままでの第２種や第１種や特種といった職種としての情報処理技術者の区分けを廃止している．職種としては高度情報処理技術者だけを認定しているのであり，そこに到達するためのキャリアパスの段階として第一種レベルと第二種レベルがあるという意味づけにかわっている．

2.2.2. 実務能力の評価を考慮した試験

旧試験では，(特種の論述式以外は) どちらかというと知識偏重型の出題に片寄っていたといえる．このため，真の実務能力を問うことが難しく，第２種情報処理技術者試験では中学生や小学生が合格するという珍事が起こったりした．このような実態が，試験に合格しても現場で使いものにならないという陰口や，給与アップのための手段としか見られないといった問題を生じさせた．

このような問題の抜本的な解決のために，実務能力を評価できるような試験制度に変更する試みが取り入られている．例えば，それぞれの午後の試験に関しては大幅に論述式や記述式の問題が増やされたり，高度情報処理技術者試験については受験者の業務経歴を考慮した採点方式が導入される．

2.2.3. 一部免除制度の導入

中間報告でも提言されていたが，新制度試験では一部免除を制度化している．これは，第一種情報処理技術者試験に適用される．具体的には，CAIT(中央情報教育研究所) や地域ソフトウェアセンターで実施されているカリキュラム研修に参加し講師による合格証を受領されることによって，午前の問題が免除される．

2.3. 第一種カリキュラム部会

ここでは，情報化人材育成カリキュラム委員会「第一種 (当初は，中級) 情報処理技術者専門部会」で活動した内容から，第一種共通カリキュラムが作成されるまでの経緯について述べる．

第 1 回 (1993/02/17)

参加者は，企業関係 (14 名)，学校関係 (6 名)，外郭団体 (6 名)，通産省職員 (数名)，CAIT 事務局 (数名) であった．この時点では，まだ「第一種」ではなく「中級」と呼ばれていた．

そこでは，中級情報処理技術者は，上級情報処理技術者に共通する技術レベルのうち中級段階を習得した人材であり，中級情報処理技術者そのものは存在しないという指摘がなされた．これより，中級情報処理技術者は，あくまでも高度情報処理技術者のキャリアパスに向けての中間点に位置づけられることになった．

また，コンピュータサイエンスの基礎教育の必要論 (我々学校側) と不要論 (企業側) の意見不一致が，顕著に表れた．

第 2 回 (1993/03/02)

共通カリキュラムの原案 (全 15 科目) が提示され，共通知識·選択知識·関連知識·実務能力の区分けがなされた．また，中級レベルに対する具体的な人材像は設定せずに，経験年数 (入社後３年～５年) だけで表すことになった．

ここでも，コンピュータサイエンスが必要かそうでないかで，議論が白熱した．

第 3 回 (1993/03/16)

第２回で区分けられた分野それぞれについての共通カリキュラム概要が提示された．そのうち，関連知識については，いままで入っていた数学や英語が除外された．

第 4 回 (1993/04/02)

上級の各部会に対して，中級の共通カリキュラム案を提示し，アンケート調査を行った．その結果について検討した．ここでは，メインフレームだけにとらわれないオープンなカリキュラム作りが提唱された．

第5回 (1993/04/20)

ここで，次のような名称の変更が，通産省職員から提示された．

・「上級情報処理技術者」は「高度情報処理技術者」に
・「中級情報処理技術者」は技術者像としての名前はなく「第一種レベル」に
・「初級情報処理技術者」は技術者像としての名前はなく「第二種レベル」に

このような変更になったのは，おそらく初級という名前に対する印象度の悪さがあったのではないかと思われる．また，旧試験との区別が不明確になるので，「第１種」を「第一種」に，「第２種」を「第二種」に，それぞれ変更した．

第6回 (1993/04/28)

実務能力分野を，応用システム開発 (アプリケーションエンジニア，プロダクションエンジニア向け)･基本システム開発 (デベロップエンジニア向け)･システム評価 (テクニカルエンジニア向け) に分けることになった．

また，この回から第 12 回まで，共通カリキュラムを構成する各科目一つ一つに対して

，カリキュラムレビューを始めた．そこで，内容の再確認や重複･不足事項について検討を行った．

第7回 (1993/05/12)

マイコン部会より，マイコン試験を第一種に追加せよという依頼があった．これによって，マイコン技術者も第一種に包括することが決まるとともに，システムアドミニストレータだけが別の試験区分に分けられることになった．

第8回 (1993/05/21)

第二種共通カリキュラム目次案との比較を検討した．

第9回 (1993/06/01)

高度情報処理技術者カリキュラム目次案との比較を検討した．

第10回 (1993/06/14)

マイコンの試験追加に関して検討した．

第11回 (1993/06/23)

第一種共通カリキュラムと第二種共通カリキュラムの最終的な調整を行った．

第12回 (1993/07/12)

カリキュラムの第３者レビューの仕方と，カリキュラム執筆要項について検討した．

第13回 (1993/08/03)

最終的なカリキュラム構成を決定した．これによって，第一種共通カリキュラムがすべて完成した．それとともに，教育時間の算定を行った．その結果，基本的には，３年間にわたり OFFJT として教育する．その内訳は，次のようになる．

知識: 平均 3 日 × 8(科目) = 24 日
(共通 5 科目，選択 3 科目とする)
応用能力 (共通: 5 日 × 2(科目) = 10 日
応用能力 (選択): 10 日
合計: 44 日 (年間 15 日間)

第14回 (1993/08/11)

カリキュラムレビューのスケジュールおよびカリキュラム作成のスケジュールについて検討した．

以上のような経緯により，第一種共通カリキュラムが作成されたわけであるが，ここで痛切に感じたことは，コンピュータサイエンス教育に対する企業の認知度の低さである．

第一種カリキュラム委員会は，企業側 (おもに，メーカーやディーラ) が圧倒的に多く，学校側 (おもに大学) の参加が少なかった．それでも，非アカデミック対アカデミックという勢力対峙が見られた．その中で，企業側の方の多くは部長クラスであるのだが，コンピュータサイエンスという言葉すら知らずに意見を述べていた．ここに現状のコンピュータ業界の問題があるわけで，現場で叩き上げた技術だけに固持している方がいるという事実である．そのような方によって，このように影響の大きな施策が決定される可能性があることは，大きな問題である．

もちろん，大学のカリキュラムをそのまま持ち込む必要はないが，少なくともコンピュータサイエンスの基礎的な科目は設置すべきである．他の専門分野においても基礎となる学問領域が確立されている．それと同様に，コンピュータ分野においてもその基盤となる学問として，コンピュータサイエンスが存在するわけである．

以前のコンピュータ業界では，需要対供給のバランスが完全に崩れ，異常なほど需要が高くなった．このため，多くの会社では人材の確保に奔走した．採用条件欄には「学部学科問わず」というようたい文句が氾濫し，ネコも杓子も採用したわけである．そして，形だけの社内教育を実施し，OJT という言葉のもとに人材派遣を行った．これによって，コンピュータ教育を専門に受けない人材 (いずれコンピュータサイエンスという言葉すら知らない部長になるのだが) が大挙して，コンピュータ業界に入り込み，バグを量産することになった．

幸いなことに，バブルの崩壊により，需要が減った．これによって，コンピュータ業界の人材に対する視点は，量から質への転換へと変わりつつある．そこで，情報処理技

術者試験の改正のをうまく利用して，コンピュータサイエンス教育をきちんとやりましょうという提唱をするための機会にしようと考えた．以上のような理念のもとに，第一種の委員会に参加したわけである．

委員会では，上述したような意見の不一致があったが，無理矢理「コンピュータ科学基礎」を共通知識科目に設定することを主張した．もちろん，企業側の委員の中でも，賛同して頂した方もあり，ようやく実現されることになった．ただし，本来１科目だけに集約できるものでもなく，各科目がコンピュータサイエンスの基礎的な部分を，エッセンスとして取り込む必要があった．しかし，現実問題として，大学側の委員が少なかったことから，これについてはほとんど不可能であった．

また，気になったこととして，旧制度の試験では入っていた数学や英語がすべて除外されたことがあげられる．その理由は，さだかではない．

コンピュータサイエンスの基盤には，数学があげられる．米国のコンピュータサイエンス教育のカリキュラムの中[3] ～ [6] にも，数学のカリキュラム体系 (微分積分学入門，解析学，確率，線形代数，離散構造，統計など) が明示されている．コンピュータを科学としてとらえると，人工科学として位置づけられる．人間が考え出した基礎理論の上に知識や技術が集積されており，それらを応用的に利用するという構図になっている．その基礎理論は，数学分野 (とくに，離散数学) から演繹されたものが多い．

英語については，英文マニュアルの講読やインターネットの普及にともない，ますます必要とされてくる傾向にある．コンピュータ用語の多くは英語であり，最近では日本語に訳さないでそのまま使用していることも多くなっている．このように，英語についても，試験に入れる意義は十分にあると思われる．

以上のようなことから，本来第二種でも第一種でも，数学や英語は必修の基礎知識として出題すべきかと思われる．そこで，前者に関しては，「コンピュータ科学基礎」において，コンピュータに関係する数学 (集合，関係代数，命題論理，述語論理，順列と組合せ，グラフ理論) を一部分取り込むことにした．

3. 新制度実施の評価

２で述べてきたような各部会が，それぞれカリキュラムを作成するとともに，共通カリキュラムに対応した共通テキストの執筆部会も発足した．これらは，CAIT が中心になって進めた．また一方では，情報処理試験センターにより，共通カリキュラムに準拠した試験問題の作成も並行して進められた．こうして，平成６年度の秋期から新制度のもとに，情報処理技術者試験が実施されるに至ったわけである．

今回の新制度実施にあたって，評価すべき点としては，次のような事項があげられる．

3.1. キャリアパスの明確化

旧試験では，数種類の職種しか設定されていないばかりか，キャリアパスという見方はほとんど含まれていなかった．職種が少なかったために，専門領域をすべて網羅することができなかったり，突如として新しい職種 (オンラインシステム技術者など) が追加されたりするだけだった．

それに対して，それぞれの専門分野に対応した高度情報処理技術者に向けてのキャリアパスを明らかにしたことは，現場で仕事に従事している技術者にとって，自分の技術レベルの位置づけを見出したり，将来の方向性を考える上でも有効である．一方，企業にとっても，社員の評価に対して，客観的な判断ができる材料になる．

3.2. 専門職種の確立

高度情報処理技術者で設定された各職種は，情報処理分野の中でより専門性の高い位置づけになっている．これによって，より専門性の高い職種として，社会的に認知される機会になった．それとともに，今回からの新しい職種である教育エンジニアやシステムアドミニストレータといったものが，社会的に認知されるきっかけになったことに意義がある．

3.3. 標準カリキュラムによる出題範囲の明示

旧試験では，CASL の仕様などしか公開されていなかった．このため，受験者にとっても，具体的な出題範囲がわからず，学習目標が立てにくいといった問題があった．また，出題レベルについては過去の問題から推測するしかなく，とにかく過去問をより多く解くという実践的な (悪く言えば，場当たり的な) 勉強方法しかなかった．

新制度試験では，標準カリキュラムがそれぞれに作成された，これによって，試験の出題範囲がはっきりするとともに，系統的に学習することができるようになった．その中でも専門学校などでは，標準カリキュラムに準拠した科目を設置したり，受験対策講座なそを開講するような動きも見られる．

3.4. 教材や指導方法の支援

CAIT では，標準カリキュラムに準拠した共通テキストをそれぞれ作成し発刊している．それとともに，各出版社も，受験対策用テキストを刊行しつつある．これによって，標準カリキュラムの内容が広く紹介されるようになるだけなく，社内教育用の教科書としてそのまま利用できることになる．

また，CAIT の標準カリキュラムには，教育目標 (教育者側の教えるための目標)，講義時間と演習時間，学習目標 (受

講者側の学ぶための目標), 取り上げるべき内容 (具体的な演習課題も含む), 指導方法及び指導上の留意点, 参考文献が列挙されている. これより, 教育ノウハウの少ない部門でも教育のための指針を得ることができる.

3.5. 柔軟な試験施策を実施

旧試験は, 当日の筆記試験だけの形式で一律的に行われた.

これに対して, 新制度では, 試験 (現在のところ, 第一種だけ) の一部免除制度を導入した. これによって, いままでの画一的な試験方式が変更され, (社会人にとっては) 受験しやすくなったといえる. 午前の知識偏重的な選択問題の試験が免除され, 午後の応用技能問題の試験にだけ集中できるからである.

4. 新制度実施後の問題

新制度が施行されてよくなった反面, 次のような問題や批判もいくつか生じている.

4.1. 応募者数の状況

平成6年秋期から開始された新制度における応募者数の推移を見てみると, 表1のようになる.

以上のように, 旧制度から新制度に引き継がれた試験区分の多くは, 前年度比がマイナスになっている. とくに, 第二種に関しては, 20 %減になっている. このような状況になったのは, 新制度になったためとりあえず見合わせた人が多かったことや, 新規に設置された試験区分の方に変更した人が多かったことなどが理由と思われる. いままで第二種を受験した人 (専門学校生など) の多くが, システムアドミニストレータに変更したことも考えられる.

情報処理技術者試験全体としては, 平成6年度受験者合計が526,332人だったのに対して, 平成7年度受験者合計は502,039人 (前年度比-4.6 %) に減少している. これより, 今回の新制度移行によって, 残念ながら受験者が減少したという事実が見出される. となると, 主催者側が思っていたほど情報処理技術者試験に対する社会的な協賛が得られていないという問題が残る.

4.2. メインフレーム指向の試験内容

それぞれのカリキュラム部会で作成されたカリキュラムは, すべてCAITから刊行されている. それだけでなく, 各カリキュラムに準拠した共通テキストも, 同じくCAITから順次刊行されている. これらの内容を査読してみると, 意外にメインフレーム時代の技術を中心に記載している記述が目立つ. このため, 試験内容がメインフレーム指向になりすぎるという批判が聞かれる.

このような状況になったのは, 多分に各部会に参加している委員自身がメインフレームの技術を背景にしているということがあげられる. それだけでなく, パソコンやワークステーションに関連した技術のうちいくつかが十分確立できていないため, 現状では評価のしようがないという問題もある. 例えば, システム設計·開発技術におけるクライアント／サーバーシステムの分散環境を構築するための方

表1

旧から新へ継承	平成6年度計	平成7年度計	前年度比 (%)
第二種	299,234	237,393	-20.7
第一種	103,586	87,151	-15.9
アプリケーション-E(旧特種)	31,674	26,656	-15.8
システム監査	7,464	6,000	-19.6
ネットワークSL(旧オンライン)	35,902	37,932	5.7
新規に設置			
システムアナリスト	8,156	7,444	
プロジェクトマネージャ		11,321	
システム運用管理		4,593	
プロダクションE		12,809	
データベースSL		7,979	
システムアドミニストレータ	40,316	62,755	

法論や技法がまだはっきりしていないこと，一連のオブジェクト指向技術の標準化が確立されていないこと，などがあげられる．このため，既存技術として確立しているメインフレームを中心にせざる得ないという問題も生じる．

以上のことは，(技術の規格化と同じ様な問題になるが) 情報処理技術者試験そのものが最新の技術に対応できていないという事実を表している．また，もしそれらに追従しようとすると，数年毎に標準カリキュラムと共通テキストを改訂していかなければならなくなる．しかし，それは，現実的には予算の都合や人材の確保などから難しいといえる．

4.3. 実務能力の評価への疑問

新制度からは，実務能力の評価をできるように試験内容を見直している．例えば，第二種の午後のプログラミングに関する問題について見てみる．いままでは，仕様が与えられてそれをもとにソースプログラムの穴埋めを行う形式だけだった．このため，プログラム言語文法を問う問題が中心だった．一方，新制度試験では，プログラムの改訂や保守に関する問題が出題されるようになっている．具体的には，ある変更仕様に対してプログラムのどの部分を修正したらよいか，あるいは，まちがったコーディングをした場合にどのような実行結果が生じるか，といった問題になっている．

以上のことから，確かに実務を意識した問題に変えようとしていることはわかるが，残念ながらこのレベルでは真の実務能力を判定することはむずかしい．プログラミング能力を正確に判断するためには，その場で仕様を与えてプログラミングを行い，その実行結果まで提出させる手順が必要になる．その上で，書法·正当性·計算量といった尺度基準をもとにプログラムの評価を行う必要がある．また，より上流工程における分析·設計に関する能力を評価しようとすると，もっと困難な問題に直面することになる．

5. 今後への提言

現在，情報処理関連分野における資格認定制度として，次のようなものがあげられる．

- ·情報処理技術者試験 (通産省·日本情報処理開発協会)
- ·マルチメディアソフト制作認定試験 (通産省·マルチメディアソフト振興協会)
- ·情報能力活用検定試験－通称 J 検－ (文部省·専修学校教育振興会)

いずれも資格としてではなく，認定として制度化されている．このため，資格を持っていてもいなくても，コンピュータ関連の職種に就くことができる．今後もこのような形態になることを前提に，情報処理技術者試験に関する今後への提言について，以下にまとめる．

5.1. 基礎的な学力を問う形に変えては？

現行の試験では，知識と応用能力に分けて出題されている．しかし，4 のやで述べたように，最新の知識あるいは個々人の技能を判定するための試験を実施することは難しい．そこで，情報処理技術者試験の内容を，(あまり時流に左右されることのない) 基礎知識だけに絞りこむように全面的に変更することを提言する．ただし，基礎知識といってもコンピュータサイエンスをベースにした範囲を対象にする．

コンピュータ分野における基幹的な領域分野には，コンピュータサイエンスがあげられる．これは，米国のカリキュラムだけでなく，我国の高等教育機関でのカリキュラム [7] ～ [10] にもきちんと明記されている．つまり，現在大学等で行われている情報処理教育は，いずれもコンピュータサイエンスをベースにした体系になっている．

コンピュータサイエンスを理解しているということは，学問としての基礎を勉学するという目的だけでなく，実務におけるさまざまな問題を解決するための考え方やアイデアを提供してくれるといった機能を持つ．コンピュータに関する応用技術の多くは，コンピュータサイエンスで提唱された基礎的な理論やモデル化された概念を，その基盤に持つ．例えば，表 2 に示すような関連 [11] があげられる．

これらのことを学んでいるかいないかで，実務現場での各種作業における品質や生産性が大きく左右される．このことを認識していない技術者が以外に多い．単に (大学だけの閉ざされた) 学問としてだけ捉えるのではなく，実務作業にこそこのような基礎的な理論やモデル概念の理解が必要になることを強調したい．

ただし，現状大学等で行われているコンピュータサイエンス教育をそのまま適用することは問題が生じるであろう．これらの基盤となる学問領域が，いかに実務技術に関連しているかを明らかにするようなカリキュラムが必要になる．そのための実践的な活動を，大学側も行わなければならない．一方，企業側でも，社内教育の一つにコンピュータサイエンスの基礎を教授するような科目を設置することも必要になる．そのための教育要員を育成しなければならない．これには，大学等での社会人教育などを利用するとよい．

以上のようなことを前提にした上で，情報処理技術者試験の問題そのものを，コンピュータサイエンス分野から出題するように変更する．これによって，試験内容が，時代の変化に流されることもなく，基礎的な知識としての学問領域だけにしぼることができる．それとともに，業界全体にコンピュータサイエンスの重要性が認識されるようになる．人材の確保にしても，「学部学科を問わず」といった無謀な採用条件は少なくなり，情報系学科が今以上に注目されることになるだろう．このことは，学生達にも影響を与え，優秀が学生が情報系学科を目指すことになり，業界全

表2

＜基礎理論＞	＜モデル概念＞	＜応用技術＞
命題論理	スイッチング回路	プロセッサ
デジタル回路	〃	〃
符号化理論	コンピュータアーキテクチャ	コンピュータシステム
(2進化)		
符号化理論	ネットワークアーキテクチャ	データ通信システム
(情報源･通信路)		
形式言語理論	プログラム言語	基本プログラム
ラムダ算法	〃	〃
述語論理	〃	〃
コンパイラ技法	〃	〃
オートマトン	アルゴリズム	応用プログラム
計算量理論	〃	〃
検証理論	〃	〃
確率統計論	〃	〃
数値解析論	〃	〃
グラフ理論	データ構造	データベースシステム
集合論	〃	〃

体として人材の質的向上が期待される．

5.2. 専門技能の試験は各種ベンダーに任せては？

情報処理技術者試験は知識を問うことが主体になり，技能面が除外される．そこで，技術に関しては，ソフトウェアベンダーが主催している試験に任せてしまうことを提言する．

とくに最新技術に関する部分は，標準規格 (ISO や JIS) にならない限り，どうしてもその技術を生み出した企業に特化したものになりやすい．このため，通産省という立場で，試験を実施することも難しい (現に，標準カリキュラムが作成された後のレビューの一つとして，登録商標されたりその企業が特定できるような用語をすべて除去するという項目があった．例えば，「一太郎」「LOTUS1-2-3」など.) わけである．逆に言うと，標準規格にならない限り，試験に出題できないことになる．このことが，最新技術をフォローできない現状を招いている．

このような状況から，一層のこと応用的な技術分野において技能を問う試験はやめてしまってかまわないといえる．そのかわりに，各企業では，それらの技術に直結したベンダー資格試験を評価すればよい．

ソフトウェアベンダーが認定する資格試験は，その認知度が高まりつつある．具体的には，次のような資格試験 [12] が実施されている．

･マイクロソフト社

プロダクトスペシャリスト，ソリューションデベロッパー，システムエンジニア

･ノベル社

認定技術者 (CNE-J)，認定アドミニストレータ (CNA-J)

･日本オラクル

データベースエンジニア，データベースデベロッパ

･ロータス

ノーツコンサルタント (LCNC)，ノーツスペシャリスト (LCNS)

これらの多くは，情報処理技術者試験の各試験区分に対応する．これより，通産省もこれらの試験を認定するような体制にすることも考えられる．

6. おわりに

以上，新制度になった情報処理技術者試験のあり方について，いくつかの視点から述べてきた．

最近のコンピュータ技術の発展動向を見ると，明らかに二極化しているように思える．それは，コンピュータ開発

者とコンピュータ利用者という二つの立場における区分けの明確化である．情報処理技術者試験でも，コンピュータ利用者に対してはシステムアドミニストレータという新しい職種を設け，そのための資格試験を実施している．とくに，コンピュータを利用する場合，プログラムを作る必要がなくなり出来合いのソフトウェアを流用すれば済むような形になっている．それだけでなく，多くのソフトウェアが，その内部をブラックボックス化させ，極力簡単な操作で済むような仕組みになりつつある．車と同様に，マニュアルの時代からオートマティックの時代に変わりつつある．このようなことから，本論では述べなかったが，コンピュータ利用者に関しては，現状の情報処理技術者試験から分離独立させ，新しい観点からの試験を実施する必要があるのかもしれない．

一方，情報処理学会でも，ようやく「社会人リフレッシュ教育に関する調査分科会」の設置が検討されている．今回の情報処理技術者試験の新制度改革に関して，情報処理学会から参加した人は少なかった (第一種部会では，私と数名の方だけ) といってよい

．リフレッシュ教育の一貫に，情報処理技術者試験を位置づけることもできるだろう．これによって，情報処理学会委員からの意見を反映することも可能になる．

いずれにせよ，情報処理技術者試験のあり方は，多くの技術者に (良い面あるいは悪い面でも) 影響を与えることになる．そのためにも，技術者にとって適切な試験でなければならない．そのために多くの人が，意見を述べることは有効と思われる．本稿がその一端になれば幸いである．

なお，本稿は，平成 6 年度の SEGEDU 月例会「コンピュータ教育雑感」で講演した内容に，いくつか追加する形でまとめたものである．

参考文献

[1] 通商産業省機械情報産業局編：新情報革命を支える人材像－ソフト新時代をめざして－，コンピュータエージ社 (1993)

[2] 通商産業省機械情報産業局編：ソフト新時代と人材育成，通産資料調査会 (1993)

[3] ACM Curriculum Committee on Computer Science: Curriculum 68 Recommendati ons for Academic Programs in Computer Science, Comm.ACM, Vol.11, No.3(1968)

[4] ACM Curriculum Committee on Computer Science: Curriculum 78 Recommendati ons for the Undergraduate Program in Computer Science, Comm.ACM, Vol.22, No.3(1978)

[5] Denning.P.J. 他: Report of the ACM task force on the core of computer science, acm Press(1988)

[6] Tucker.A.B. 他 : Computing Curricula 1991-Report of the ACM/IEEE-CS Joint Curriculum Task Force acm Press(1991)

[7] 大学等における情報処理教育検討委員会：大学等における情報処理教育のための調査研究報告書，情報処理学会 (1991)

[8] 大学等における情報システム学の教育の実態に関する調査研究委員会：大学等における情報システム学の教育の実態に関する調査研究，情報処理学会 (1992)

[9] 大学等における一般情報処理教育の在り方に関する調査研究委員会：大学等における一般情報処理教育の在方に関する調査研究，情報処理学会 (1993)

[10] 短期高等教育における情報処理教育の実態に関する調査研究委員会：短期高等教育におる情報処理教育の実態に関する調査研究，情報処理学会 (1994)

[11] 河村一樹：ソフトウェア技術者に対するコンピュータサイエンス教育カリキュラムの提案，情報処理学会新しい時代の情報処理教育カリキュラム．シンポジウム論文集，pp.95-102(1994)

[12] 青木慎一：人気高まるベンダー資格試験，日経コンピュータ 1995.11.27 号，pp

囲碁的分析・設計手法

佐原伸
URL: http://www.sra.co.jp/people/sahara
E-Mail: sahara@sra.co.jp

1章 はじめに

前々から、ソフトウェアの設計は囲碁のプレーに似ていると感じていた。囲碁の定石[1]や手筋は、ソフトウェアの設計で言えば「データ構造とアルゴリズム」などに相当し、囲碁のルールやヨセの一部が数学的に厳密に定義されている[2]のは、ソフトウェアの設計で言えば「形式仕様で問題の一部が記述されている」ことに相当しよう。

プロの棋士は、形式に定義されたルールや手筋、あるいは非形式に定義された定石や手筋を場面に合わせて使い、既存の定石や手筋がないところは自分で考えて碁を打つ。もちろん、形式に定義されたルールや手筋の中身を完全に知っているわけではない。形式定義から導かれた実践的な定理を使って打っているにすぎない。

ところが、現在の開発現場の設計者の大部分は、「ルールを知らず、定石を知らず、手筋を知らず」あるいは「間違ったルールと定石と手筋を使って」ソフトウェアの設計を進めている。

そこで、本稿では形式仕様とデータ構造とアルゴリズムなどの定石を使い、「囲碁的パラダイム」で実際にプログラムが開発できることを検証してみようと思う。もちろん、形式仕様の研究では、実用的な手法やツールが開発され「実際に使えること」は分かっているのだが、それと「開発現場で使えること」とは次元が異なるので、開発現場からの視点で検証してみようというわけである。

対象のプログラムは、日付計算にしてみた。有価証券評価システムをSmalltalkで作っているチームが私の隣にいるので、少しでも手伝おうと言う趣旨である。また、問題がさほど大きくならないので、本稿で説明するにはちょうど良い大きさであろうという判断もある。

2章 問題の説明

さて、まずは設計に先立つ分析であるが、ここでは事務処理に必須な「２つの日付の間の、日曜日の回数を求める」を行うことを考えてみよう。この機能は、「休日を除く日数の加算や減算」を行うことや「第２土曜日や第４金曜日を求める」場合に必須の機能である。この時、休日を考慮しない日付間の加減算や曜日を求める関数は存在するものとする。これらの機能も形式仕様で分析・設計・実現したのだが、天文計算のドメイン知識を調べるなどの面倒はあるものの、最終的にはプログラムというよりさほど複雑でない計算式で実現されるので、今回の説明からは除いた。ともかく上記のような関数を使えば、毎年年

[1]定石は約２万個あると言われている。

[2]エルウィン・バーリカンプ／デビッド・ウルフ著、吉川／小林／石原訳「囲碁の算法　ヨセの研究」トッパン, 1994, ISBN=4-8101-8929-5

末に日本の各地で繰り返される「不完全な日付処理」によるシステム・トラブルが少しは減るだろうし、昭和が平成に変わったときの混乱は防げたであろうし、無事2000年を迎えることができるだろう。

2.1 要求仕様記述言語

要求仕様は、形式仕様記述言語であるRSL[3]（RAISE Specification Language）を使ってみよう。この言語は、ロボットの仕様記述や、中国の鉄道システムの要求仕様記述言語として用いられた実績があるから、日付計算くらいは簡単に記述できると考えたからである。

2.2 行き当たりばったり仕様

まず「ソフトウェア開発の現場」の伝統に則って「行き当たりばったり」法で仕様を考えてみよう。

```
日曜日回数: Date×Date→Int
/* fromとtoの間の日曜日の発生回数を返す。 */
variable days : Int

日曜日回数 (from, to) ≡
  days := subtractDate(to, from);
  if isSunday(from)
    then days / 7
    else (7 - dayNumber(from) + days) / 7
  end
```

仕様1　行き当たりばったり仕様

「日曜日回数」が関数名で、２つのDate型の引数（from, to）を持ち、整数を答えとして返す。'/'は整数の商を計算する演算子である。subtractDateは日付の引き算を行う関数であり、isSundayは日曜日か否かを返す関数であり、dayNumberは曜日を0..6の整数（0が日曜日）で返す関数で、いずれも、あらかじめ定義されているものとする。

この仕様は、すぐにfrom日付が日曜日であると答えがおかしいことに気が付く。そこで、以下のように修正した。

---

[3]The RAISE Language Group著、「The RAISE Specification Language」、Prentice Hall、1992, ISBN=0-13-752833-7

```
日曜日回数: Date×Date→Int
/* fromとtoの間の日曜日の発生回数を返す。 */
variable days : Int

日曜日回数 (from, to) ≡
  if from = to then
    if isSunday(to) then 1 else 0 end
  end;
  days := subtractDate(to, from);
  if isSunday(from) then
    if days > 0 then days / 7 else abs(days) / 7 end
  elsif days > 0
    then (7 - dayNumber(from) + days)  / 7
    else dayNumber(from) - days / 7
  end
```

仕様2　行き当たりばったり仕様修正版

この仕様でも欠陥があるのだが、それは読者への宿題としよう。いずれにせよ、これらの仕様には以下の欠点がある。

⑴プログラミングとテストの工数が増大する
⑵プログラミングがいつ終わるか分からない
⑶正しいことの保証がない
　これが、最大の問題なのだが、「行き当たりばったり」法で作ってしまったプログラムの正しさを証明するのはきわめて困難である。実行時にいくらテストしてみても、「すべての組み合わせをテストした」と胸を張って言える場合以外、正しいことの保証はない。そして、「すべての組み合わせをテストする」のは、実用的なプログラムの場合、ほとんど不可能である。

2.3目的指向の段階的詳細化

「ではどうするか？」ということになるが、形式手法では「目的指向型」で開発する。具体的には、プログラムの作成と証明を同時に行い、段階的に詳細化していくという方法で、以下の手順で仕様からプログラムへと変換していく。

⑴問題は何であるかを把握する
　ちょっと面倒なことになるので、後述する
⑵事後条件・事前条件を見つける
　プログラムは事前条件を満たすどんな状態の下で実行しても、有限時間内に実行が終わり、事後条件を満たす状態をもたらす
⑶再帰または繰り返しにより事前条件を事後条件に近づけていく
　ここからは設計工程になる。実際には、事後条件を緩めて、事前条件P_0から事後条件へ

至る途中の「条件」（P_1, P_2など）を導出する

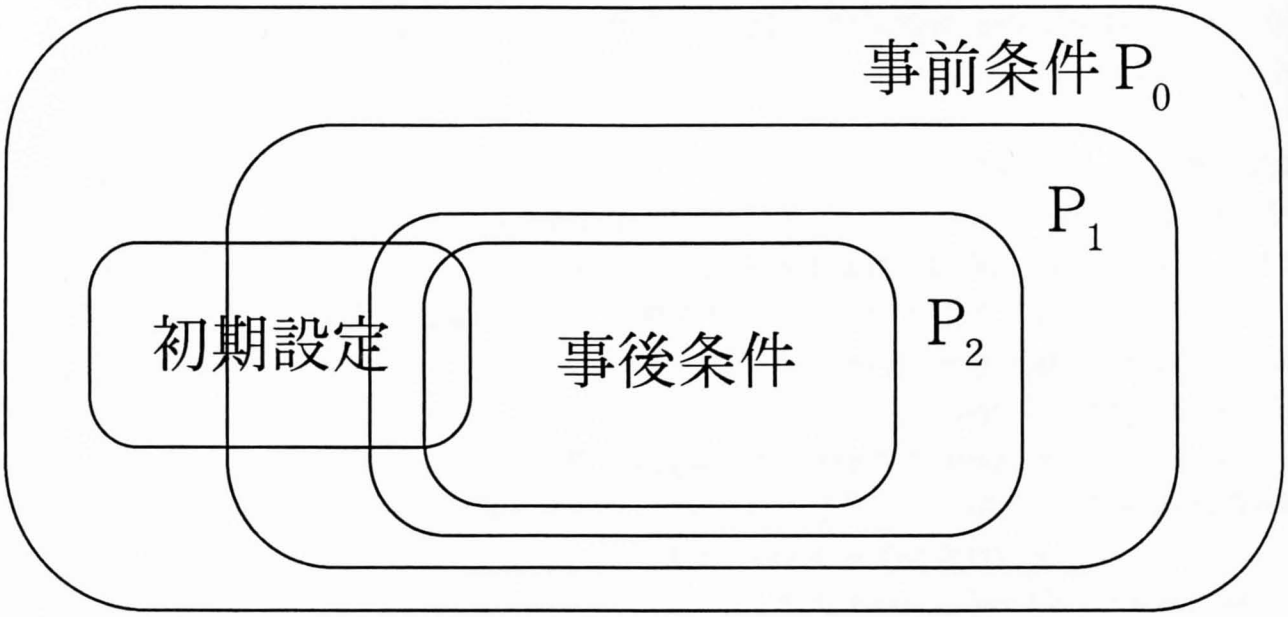


図 1　事後条件・事前条件・初期設定の関係

(4) 最後に、効率の考慮

効率の問題は、本稿ではあまり触れない

2.3.1 事後条件を求める

ここでは、「問題は何であるか」をひとまず考えないことにして、事後条件の洗い出しへ進もう。事後条件は以下のようになる。

```
曜日回数（dayOfTheWeek, from, to）≡
/* fromとtoの間の曜日dayOfTheWeekの発生回数を返す。 */

post
  ∃DW:曜日-set, ∀d∈DW・from ≦ d ≦to∧dayNumber(d) = dayOfTheWeek⇒
  曜日回数（dayOfTheWeek, from, to）≡ card(DW)
```

仕様3　事後条件

ここでは、事後条件を考えるついでに、「２つの日付の間の、日曜日の回数を求める」のではなく、「２つの日付の間の、指定された曜日の回数を求める」ように仕様を拡張した。日曜日だけを求める特殊な機能ではなく、問題を一般化したわけである。そのようなことは「最初から指定せよ」とお考えかも知れないが、実際の開発現場では、このように「要求」は「真の要求でない」形で出てくることが多い。ユーザーの言うとおりに要求を実現すれば「毎日が日曜日」ということなりかねない。

この事後条件の定義は「ある曜日の集合DWがあって、その任意の要素dがfrom ≦ d ≦ toかつdayNumber(d) = dayOfTheWeekなる制約条件を満たしているならば、求める曜日回数は集合DWの要素の個数（cardinality）である、とまことに素気ない。曜日-setは未定

義のまま記述しているが、例えば、９６年７月８日（月）から７月３１日（水）までの月曜日の回数は、{７月８日,７月１５日,７月２２日,７月２９日}という集合の要素数５である、ということである。

「その要素の個数を求めるのが問題なんだろう」とおっしゃるだろうが、「そうです」と答えるしかない。「行き当たりばったり仕様」では「問題の解き方」を書いていたが、形式手法では「まず、問題を記述する」のが目的なのである。解くのは設計工程よりあとの仕事というわけである。

2.3.2 事前条件を求める

次に事前条件を求める。事前条件は「初期設定」で簡単に設定できるか、あらかじめ問題が満たしている条件を考えることになる。ここでは、一応、以下のような事前条件を考える。

```
pre
∃D:日-set, ∀d∈D・ from ≦ d ≦to⇒DW⊆D
```

仕様4　事前条件

ここでは、日の集合Dがあって、その要素dがfrom ≦ d ≦toならば、事後条件で出てきた集合DWがDの部分集合になると言っている。

2.3.3 事後条件を弱める方法

事後条件を緩めて、事前条件との中間になる条件を求めるには、以下の方法がある。

⑴ 積項を取り除く

項A, B, Cがあって、A∧B∧Cのような形をしているとき「積項」と呼ぶ。そのうちのひとつの項を取り除いて条件を緩めようと言うのだ。ここでは、事後条件からd ≦toを取り除いてfrom ≦d ∧dayNumber(d)=dayOfTheWeekとしてみよう。

⑵ 定数を変数に置き換える

例えば、1≦d≦10というような条件があったら、1≦d≦i∧1≦i≦10に置き換えてみるという方法である。

⑶ 変数の領域を広げる

例えば、1≦d≦10という条件があったら、0≦d≦100に置き換えてみるという方法である。

⑷ 和項を追加する

これは、Aという項があったら、A∨Bを考えてみようという方法であるが、試行錯誤になってしまい、行き当たりばったりのプログラムと同様になりかねないので避ける。

2.3.4 不変条件と限度関数の開発

さて、dをfromからtoまで反復させてプログラムを作ろうと決心すると、反復の上限を

決める「蓋」の条件は、先ほど取り除いた積項の否定を使えばよいことが分かっている。今の場合、d > toである。また、反復の最中変わらない不変条件は、残りの積項すなわちfrom ≦d ∧dayNumber(d)=dayOfTheWeekとなる。反復が終了することを保証する「限度関数」tはt : to − dとすれば、tは常に正であり、かつdは増加するのだから、段々0に近づいていくことが分かり、反復が終了することを保証できる。

プログラムの大筋は以下のようになるだろう。

```
d := from;
do
  ?
  d := d + 1;
until  d > to end
```

プログラム1　大筋プログラム

上のプログラムで？の部分は、dの曜日が指定された曜日と等しいことを保証するコードということになる。すなわち以下のようになる。

```
曜日回数 (dayOfTheWeek, from, to) ≡
/* 事前条件 : ... */
/* 事後条件 : ... */
variable sum : Int := 0
d := from;
/* 不変条件 : ∃DW:曜日-set, ∀d∈DW・
from ≦d ∧dayNumber(d) = dayOfTheWeek */
/* 限度関数 t : to - d */
do
  if dayNumber(d) = dayOfTheWeek then
    sum := sum + 1
  end
  d := d + 1;
until  d > to end;
sum
```

プログラム2　一応完成したプログラム

2.4問題は何であるかを把握する

さて、一応正しいプログラムは完成したわけであるが、どう見てもエレガントなプログラムとは言えない。原因を考えてみると、どうも「問題は何であるか」を考えて、問題に内在する性質をうまく使っていないことにその原因があるようだ。

例えば、有名な「ケーニヒスベルグ[4]の７つの橋」の問題は、仕様５で示した事後条件（頂点n_1とn_2が繋がっていて、任意の頂点の弧が偶数個である）を満たしていれば一筆書き可

[4]現在は、ロシアのカリーニングラードである

能であることを示している。この場合、事後条件を発見した時点で、問題の解答も見つけていることになる。

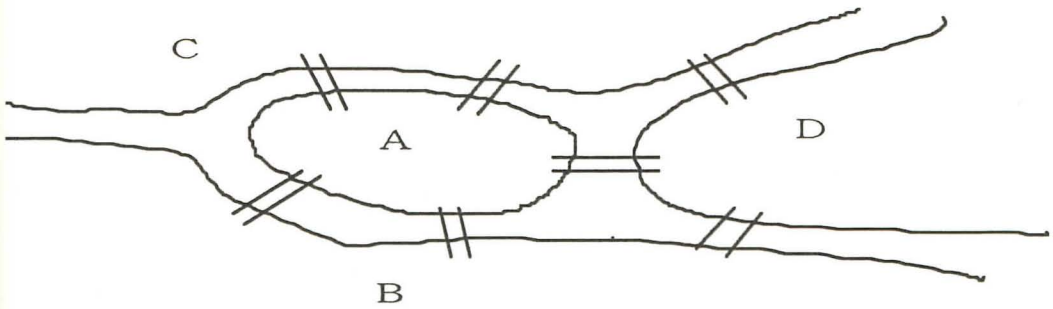


図 2 ケーニヒスベルグの７つの橋

$\forall n, n_1, n_2 \in Node$・
$card(arc(n)) \backslash 2 = 0 \wedge n_1 \leftrightarrows n_2$
⇒一筆書き可能

仕様5 ケーニヒスベルグの７つの橋の答え

2.4.1 そもそも、問題は何であったか？

曜日問題の場合、ケーニヒスベルグの７つの橋のようにうまくいくであろうか？
事後条件のfromを0に、toをnに対応させ、曜日を0..6に対応させてみると...
ある性質（７で割った余りが指定されたものと同じ）を持った自然数を数えることと同じになる。ならば、漸化式で問題を定義し、数学的帰納法で証明することができるはずで、仕様ないしプログラムを再帰的に書くことができる。

2.4.2 漸化式

漸化式は以下のようになるだろう。

```
d0 = 1（同じ性質のとき） | 0（同じ性質でないとき）
dk+1 = dk+1 （同じ性質のとき） | dk（同じ性質でないとき）
```

仕様6 漸化式による定義

従って、仕様は再帰的に以下のように書ける。

```
曜日回数: dayOfTheWeek×Date×Date→Int
/* fromとtoの間の曜日dayOfTheWeekの発生回数を返す。*/
曜日回数（dayOfTheWeek, from, to）≡
 let
  isIt(d) ≡  if dayNumber(d) = dayOfTheWeek
   then 1 else 0 end
 in
  case to - from of
```

```
    0 → isIt(from)
    _ → isIt(from) + 曜日回数 (dayOfTheWeek, from + 1, to)
  end
end
pre from ≦to
```

仕様7　再帰的な仕様

ここで、「_」は0以外の場合を表す。

2.4.3 数学的帰納法

仕様７は、数学的帰納法を使って容易に[5]証明できる。

なお、脇道にそれるが、数学的帰納法による証明には納得行かない向きもあろう。特に、高校や大学の教養課程では、以下に示すペアノの公理を示さずに数学的帰納法だけを教えるので、厳密に言えば「数学的帰納法で証明できた」というのはインチキである。

ペアノの公理で自然数の定義を以下のように「帰納的」に行っているので、自然数に対応させることのできる問題に数学的帰納法が使えるのだ。

```
value
      zero : N,
      succ : N → N /*  「次」を返す関数  */
axiom ∀n, n1, n2 : N・
      [first_is_zero]  /* ゼロの次はゼロでない  */
        ～ (succ(n) ≡ zero) ,
      [linear_order]  /*  「次」同士が等しければ、前も等しい  */
         (succ(n1)≡ succ(n2) ) ⇒ (n1 ≡ n2) ,
      [induction]  /*  「帰納法の定義そのもの  */
        ∀p : N → Bool・
         (p(zero) ∧ (∀n : N・p(n) ⇒p(succ(n))) ) ⇒ (∀n : N・p(n))
```

仕様8　ペアノの公理

2.4.4 再帰から反復への変換

末尾再帰のプログラムは、スタックを必要としない反復型のプログラムに変換できる。仕様７で表すプログラムは末尾再帰ではないが、末尾関数が結合則を満たせば、同様の変換ができることが分かっている[6]。従って、以下のプログラムが得られる。

```
variable sum: Int := 0
```

[5]通常、こう書く場合証明はさほど簡単ではない :-) が、「力仕事でできる」というニュアンスがある。

[6]なお、一般の再帰プログラムも、スタックに途中の結果を積めば反復型のプログラムに変換できる。

```
n := to - from;
while n≠0 do
      sum := sum + isIt(n);
      n := n - 1
end;
sum
```

プログラム3　反復型のプログラム

なお、一般の再帰プログラムも、スタックに途中の結果を積めば反復型のプログラムに変換できる。

2.5問題をもう少し考えると

さて、プログラム3は、プログラム２よりも簡単に証明しつつプログラムに落とせたものの、本質的にプログラム２と同じである。「日付特有の知識」を使っていないし、エレガントではないし、**n = to - from**とすれば計算量が○（n）であり、キリストが生まれた日から今日まで金曜日が何回あったかなどを計算しようとすると、言語処理系によっては難儀する。

問題をもう少し考えてみよう。実は忘れていた不変条件があった。

∀d0, d1, di∈**曜日**-set・
dayNumber(d0)=dayNumber(d1) ∧ from≦d0≦d1≦to⇒
∃d : Int・d1 - d0≡7×d ∧ max(di) - min(di)≡7× (card(**曜日**-set)−1)

仕様9　忘れていた不変条件

要するに、指定された曜日の日付同士の差は７の倍数であり、指定された曜日の日付で最も**to**側に近いものと、最も**from**側に近いものとの差は7×（card(曜日-set)−1）である。

変換すると

card(**曜日**-set)≡(max(di) - min(di)) / 7+1

となり、求めたい答えはもう不変条件の中に出ている！

しかも、max(di)と min(di)が見つかれば、途中の反復計算は必要なくなるので、計算量は○**(1)**になる。実は、min(di)さえ見つかれば、max(di)はtoで代用してよい。

to - max(di) < 7∧d1 - d0≡7×d **なので**
(max(di) - min(di)) / 7≡(to - min(di)) / 7 **なのである**

さらに、実はmin(di)はもう見つかっている。プログラム３で、実行時にdoの後の最初のif文を満たすdがmin(di)である。従って、プログラムは以下のようになる。

```
曜日回数（dayOfTheWeek, from, to）≡
d := from;
while dayNumber(d)≠dayOfTheWeek do
  d := d + 1
end
(to - d) / 7＋1
```

プログラム4　効率の良いプログラム

Smalltalkで実現したプログラムは以下のようになる。

```
occurrencesOfDayOfTheWeek: aDayNumber Between: aDate

  | date fromDate toDate |
  fromDate := self min: aDate.
  toDate := self max: aDate.
  date := fromDate.
  [date dayNumber = aDayNumber]
    whileFalse: [date := date addDays: 1].
  ^(toDate subtractDate: date) // 7 + 1
```

プログラム5　形式手法を使ったSmalltalk版完成プログラム

実は、行き当たりばったりプログラムにも、最終版があり、以下のようになった。

```
occurrencesOfDayNumber: aDayNumber Between: aDate

| days |
days := self subtractDate: aDate.
self dayNumber = aDayNumber ifTrue: [^7 + days abs // 7].
self dayNumber > aDayNumber
  ifTrue: [days > 0
    ifTrue: [^7 - (aDayNumber - self dayNumber) abs + days abs // 7]
    ifFalse: [^(aDayNumber - self dayNumber) abs + days abs // 7]]
  ifFalse: [days > 0
    ifTrue: [^(aDayNumber - self dayNumber) abs + days abs // 7]
    ifFalse: [^7 - (aDayNumber - self dayNumber) abs + days abs // 7]]
```

プログラム6　行き当たりばったりSmalltalkプログラム最終版

2.5.1 2つのプログラムの比較

形式手法と行き当たりばったりの2つのプログラム（プログラム5と6）を比較してみ

よう。

正常データのテストは両方とも一応正しい答えを返すが、異常データに対して形式手法版は無限ループに陥ることにより異常を検出した。一方、行き当たりばったり版は、ときどき間違った答えを返す。もちろん、商品としてのソフトウェアはパラメータをチェックして無限ループに陥らないようにした方がよいが、少なくとも間違った答えを返すよりは良い。

正しさを主張できるか？という観点で見ると、形式手法版は正しさを主張できるが、行き当たりばったり版は、今となってはなぜ動くかも分からない。

効率はどちらも○(1)で互角であろう。

2.6 またまた、問題を考える

さて、ここまでで一応形式手法に軍配が上がったと思うが、プログラム４（あるいは５）はまだまだエレガントではない、と思える。「反復を消せないのか」と誰でも直感的に感じるであろう。筆者は怠け者なので、「計算量○(1)だし、まあ良いか」と思っていたのだが、SEA代表幹事の山崎さんから、以下のようなエレガントな解を頂いた。以下、筆者の「翻訳ミス」があるかも知れないが、紹介する。

2.6.1 今度こそ最後のプログラム？

まず、問題を良く考えると事前条件・事後条件は以下のようになる。

```
pre
type R = {|rng [n →n / 7 | n∈Int]} /* 7で割った商の集合 */
f, t∈Int, w∈R, 0≦f≦t,
h: Int → R /* 環準同型（ring homomorphism） */

post
S = dom h(w) ∩ {f..t}・A ≡ card(S) /* Aが答え （dom h(w)≡h^-1(w)） */
```

仕様10　事前条件・事後条件最終版

上の条件を見ると、もはや問題は日付問題ではなくなった。すなわち整数系を環（ring）と見て、その商環（quotient ring）への準同型写像があり、その代数系上で事後条件を満たすプログラムを作るということになる[7]。

n / 7のところに少し日付問題の面影が残っているが、ここも任意の整数に変えて問題を拡張できる。が、とりあえず、以下の文中では「日付問題」の用語を使うし7という定数も

[7]かどうかは、この原稿を書き上げてからチェックする予定である。こうなってくると代数仕様記述言語CafeOBJとその背景にある理論（多ソート代数、カテゴリー）あたりに出動して頂くのが良いかも知れない。が、筆者の手には余るので、IPAのCafeOBJホームページ（http://www.ipa.go.jp/STC/CafeP/cafeproject.html）からマニュアルとソフトを手に入れて頂きたい。

また、そこまで行かなくとも、今はやりの秋山仁氏らが訳した「C.L.リュー著。成嶋弘、秋山仁訳。コンピュータサイエンスのための離散数学入門。マグロウヒル、1986年、ISBN=4-89501-087-2」あたりでも確認できるだろう。

使う。

さて、次に事後条件をより詳細に分析していく。

```
I = {f..t}
d = t - f + 1 /* = card(I) */
q = d / 7
r = d \ 7 /* 7で割った余り */
```

とすると、

```
q ≦ A ≦ q+1
```

が成り立つ。なぜなら、

(1) 任意の連続する7日間には、必ずw曜日がちょうど1日存在する。
(2) card(I) = 7×q + r (0≦r<7)であるから、Iには少なくともq個の連続する7日間が存在するが、q+1個は存在しない。
(3) 余りのr日間にw曜日が存在するかも知れない。

次に、

```
x ++ y = (x + y) \ 7
x ⊥ y = max(x - y, 0)
```

として、

```
T = {h(f)..h(f) ++ (r ⊥ 1)}
```

を考える。Tは余りr日間の曜日に対応する（card(T) = r）。
すると、

```
A ≡ if w∈T then q + 1 els q end
```

ここで、

```
x -- y = if x ≧ y then x - y els x - y + 7 end
```

とすれば、

```
w∈T ⇔ (w -- h(f)) + 1 ≦ r
```

である。なぜならば

```
w∈T ⇔ {0..(r ⊥ 1)}∋w' = w -- h(f)
  ⇔ r ⊥ 1 ≧ w'
  ⇔ r ≧ (w -- h(f)) + 1
```

従って、プログラムは以下のようになる。

```
A(f, t w)≡
     let
          d ≡ t - f + 1
          q ≡ d / 7
          r ≡ d \ 7
          delta ≡ if (w -- h(f)) + 1 ≦ r then 1 els 0 end
          x -- y ≡ if x ≧ y then x - y els x - y + 7 end
     in
          q + delta
     end
```

プログラム7　形式仕様によるプログラム最終版

補助関数はあるにせよ、反復は消滅し、プログラム６までよりはエレガントなプログラムに見える。

3章 おわりに

プログラム７の中の定数７を一般の整数に拡張した場合、何に役立つのかはまだ見えてこないが、どうも問題はこれで終わりそうにない予感がしている。しばし、代数の勉強にいそしみ、後日報告したい。

いずれにせよ、開発の現場に転がっていた問題をプログラムするのに意外に手間取った。が、手間取らなければ何が起こるか？数年後に欠陥が発生し、そのころには開発したメンバーは居なくなっているか、居ても、もう問題も解き方も忘れていて、かなりの大トラブルになるであろう。実際に、昔、証券業務プログラムの開発でこのような経験を何度もしている。たかが10行ほどのプログラムでも、場合によっては数百万円の損失を招く。まして、数百万行のプログラムでは...

それに比べ、形式手法による開発は高校程度の数学知識をベースにして、最初に考えなければならないことがやや多いにしても、開発工程の後の方の手間と手戻りが少なくて済むと言えるだろう。

今回対象にした日付問題は、別に「特に難しい」問題を選んだわけではない。実際の開発現場では「もっと難しい問題」がごろごろと転がっている。そのような問題に徒手空拳で挑むのは、囲碁の名人に目をつぶって挑むようなものである。

囲碁の定石は２万個ほどあり、アマチュア初段でもこれをほぼ使いこなしている。それでもアマチュア４段には歯が立たない。しかし、アマチュア初段と４段の差などは、プロから見れば誤差の内である[8]。

[8]小林元名人の筆者への忠告から。

形式手法の「定石」はRAISEの場合で約3000である。ソフトウェアの開発は囲碁よりは少し難しいとしても、これくらいを使いこなしてアマチュア初段格ということになるだろうか？筆者自身、まだこの程度も使いこなせないので、アマチュア５級というところだと考えているが、それでも、何とかプログラムの開発の一部に適用することができた[9]。「従って、開発現場でも形式仕様は使える」という結論が導けるが、やや強引だろうか？ :-)

日付計算プログラム全体[10]は９５年１月１日から開発が始まり９６年６月末に一応完成したが、全体の工数は0.5人月ほどである。Smalltalkの行数にして717行であり、天文計算やビジネスに使える日付処理を実現している。Smalltalkにあらかじめ備わっているDateクラスより、プログラムの適用性・保守性・再利用性がかなり高いという評価結果が出ている。[11]

なお、本稿は９５年９月に直江津で行われたSEAソフトウェア・デザイン・ワークショップの夜の成果を元に、９６年７月に行われた南山大学講義およびSEA名古屋支部セミナーで話した内容を基にしている。

[9]筆者は、大学時代「代数」で追試を受け、教官のお情けでなんとか単位を取った。

[10]http://www.sra.co.jp/people/sahara/softwareから入手できる。

[11]IPAによるオブジェクト指向評価プロジェクトの成果として、SRAのftpサーバーでβ版を無償公開しているツールOOMによって評価した。

新 刊 紹 介

ー働くソフトウェア技術者のための必読書2冊ー

山崎 利治
(Free)

最近SRAの酒匂さんと佐原伸さんが，それぞれ訳書と著書を出版された．会社勤めでこのような仕事をされるのは偉大な努力家の証拠で，まず敬意を表したい．この2冊の本はいずれも《働くソフトウェア技術者》にとっての必読書だと思うので，SEAMAILに紹介しておきたい．

****

Bertrand Meyer 著，
酒匂寛訳，
プログラミング言語理論への招待
ー 正しいソフトウェアを書くために ー
アスキー出版局，1995.
Introduction to
the Theory of
Programming Languages.
Prentice Hall, 1991.

****

これは大判で活字も大きく，老眼にはありがたいが，そのトレードオフとして，寝ころんで読むには重くて疲れる．

一瞥ではプログラム言語の意味論の本だと思う．たしかにその通りで，これはまた大切な主題である．というのも，実用プログラミングのために1つの言語の修得を急ぐあまり，プログラム言語の持つ基本概念や一般原理について認識不足のソフトウェア技術者が多いと思うからである．このような技術者は基礎ができていないので，新言語に向うと戸惑ってしまい，素早い習熟が難しいのである．そうならないためには，このような学習が必要である．

Meyerはこの本で，命令型プログラム言語の古典的部分(たとえばPascal subset)に対して，表示的意味論と公理的意味論を展開し，処理系などの実現に係わる詳細や例示に頼ったりすることなく，厳格にまた洗練した流儀で，言語の意味を与えている．これをそのまま受けとればいい．

そうではあるが，働くソフトウェア技術者としては，この本を「仕様記述入門」,「検証入門」さらにいえば「正しいプログラム作成のためのプログラミング入門」と受けとめて，読みたい．この気持ちは訳者の酒匂さんにもあるようでそれが訳書の副題(正しいソフトウェアを書くために)に表われていると思う．どうしてそう思うか，またどんな内容かをも，もう少し説明しなければならない．

乱暴な話だが，プログラム言語のよい理解方法に，その仕様を書いてみることが挙げられる．そのためには，言語を構文と意味に分けて記述するのが伝統的な便法である．構文面についてはAlgol 60報告書で用いたバッカス記法が十分満足できる形式的記述方法であり，よく普及もしている．しかし，意味面に関しては問題が多かった．意味記述について厳密さ，つまり，形式性が深刻に求められたのはPL/Iの出現によってである．

統一のない多彩さというよりは乱雑な言語であったPL/Iの自然語による意味記述は，処理系作成者たちに共通の解釈をもたらさないほどであったのである．そこで，IBMウィーン研究所でこの形式的記述のためにVienna Definition Language（VDL）が生まれた．これは言語を解釈実行する抽象機械を与え，言語の意味を抽象機械の基本操作の系列として定義するもので，操作的意味論(Operational Semantics)の嚆矢である．

一方，Algol 60の後継言語Algol 68の検討のなかで四苦八苦七転八倒している間にも，さまざまな意味記述方法が考えられてきた．いずれもMeyerが克明に説明しているものだが，ここでそれらにちょっと触れておきたい．

属性文法（Attribute Grammar): Ironや Knuthによる，バッカス記法で書ける文脈自由文法の意味を定義する枠組．文法に現われる記号に属性を与え，それぞれの生成規則にそれらの属性値を計算する規則を付随させたものである．

公理的意味論（Axiomatic Semantics): Hoareによる．プログラム中の変数全体を記憶とか状態とかいう．プログラム文P，状態上の2つの述語 Preと Postの組 {Pre} P {Post}を弱い表明という．これは述語 Preが成立している状態でプログラム文Pを実行し，もしPが終了すれば，そのとき述語 Postが成立するというものである．このような弱い表明に関する推論を行うために，代入文，条件文，反復文，連

接文に対する，また述語間の含意などに対する推論規則を用意して1つの形式論理 (Hoare 論理) を構成し，それを，プログラム言語の意味の定義とともに，プログラムの正当性などの証明に利用するものである．

表示的意味論 (Denotational Semantics): プログラムの意味を集合と関数によって与える外延的方法である．プログラムが計算する関数はプログラムの最小不動点関数としてえられるので，プログラムの意味をこの最小不動点によって与える．Strachey による考えで，彼は洗練された記法を与えたが，意味領域や意味関数を再帰的に定義するときの難点を Scott が克服し，広く利用される方法となった．VDL も，その抽象機械の詳細が非本質でそれを嫌い，表示的意味記述の Vienna Development Method（VDM）へと移行した．

構造的操作意味論: 表示的意味論が意味関数を直接明示的に与えるのに対し，それを言語構造に即した評価規則を与えて平明に意味記述する枠組である．Standard ML の仕様記述に使用しているが Meyer はこれに触れていない ([3] 参照)．

さて，本題である本の内容に入ろう．

第1章は前口上である．

第2章は以降の形式的記述言語の説明である．言語は VDM 風で，集合，関係，関数，列 (リスト) とその上の演算の記法を示す．

第3章は意味記述のための構文法である抽象構文を説明する．

第4章ではポケット電卓を裸にしたような言語 Lullaby の仕様を，属性文法，変換意味論，操作的意味論，表示的意味論，公理的意味論のそれぞれの枠組によって VDM 風言語によって書いてみせている．まずは手短かな形式的仕様記述の味見である．

第5章は仕切り直しで，表示的意味論展開のためのラムダ記法とラムダ計算を説明する．再帰的定義の問題は第8章に先送りする．

第6章は命令型言語 Graal の表示的意味記述である．懇切丁寧であって，懐手して読める．これで VDM 流の仕様化技術は一応卒業である．

第7章では現行言語の機能を若干追加してみせる．

第8章はこの本のもっとも数学的な部分である．第5章で先送りした再帰的定義の有効性－矛盾なく実際に領域や関数を定義していること－の保証である．フランス人らしく Berry の安定領域の流儀で不動的意味論を展開する．直観に訴えた説明であるが，厳密好みのひとには少し軟弱に映るかも知れない．そういう人は気が済むまで形式化して欲しい．

第9章は Hoare 論理と Dijkstra の最弱前件意味論であるが，ここでさまざまなプログラムの正当性証明（検証）の練習をする．

第10章は Hoare 論理と表示的意味論が相補的なものであることを説明する．つまり Hoare 論理は形式的体系でいう理論であって，表示的意味論はその一つのモデルを与える (充足関係) ものにすぎないというわけである．

ソフトウェアの形式的開発法は，形式的に仕様を書いて，正当性証明を与えながらその仕様を実現するものである．仕様記述の形式的方法はプログラム言語で生まれ育ってきた．

この本では，プログラム言語を題材に仕様記述を学び，また正当性証明を学ぶ．これはまさに形式的開発方法の核心を体得するものであり，ここに，この本を必読書とする所以がある．大方の愛読を期待したい．

最後に蛇足を加えたい．訳書の文献表には邦訳も記され親切である．いまでは入手困難であるが Manna の本も邦訳がある．

五十嵐滋訳，プログラムの理論，日本コンピュータ協会，1975.

ついでに邦語の関連文献を付けさせていただく．

[1] 林晋，プログラム検証論．共立出版，1995.
[2] 井田哲雄，計算モデルの基礎理論．岩波書店，1991.
[3] M.Hennesy 著，荒木啓二郎·程京徳訳，プログラミング言語の意味論入門．サイエンス社，1993. The Semantics of Programming Languages. Wiley, 1990.
[4] 中島玲二，数理情報学入門．朝倉書店，1982.
[5] 小野寛晰，プログラムの基礎理論．サイエンス社，1975.
[6] 武市正人，プログラム言語．岩波書店，1994.
[7] 高橋正子，計算論．近代科学社，1991.
[8] 横内寛文，プログラム意味論．共立出版，1994.

****

佐原伸著,

オブジェクト指向システム分析/設計 Q&A,

ソフト･リサーチ･センター, 1995.

****

こちらは酒匂さんの本と違って, ハンディで持ち運びやすい. したがって, 友人を待つ間の飲み屋でも, 電車の中でも読める. しかも, 質疑応答の形式は, このような断片的な読み方を助ける.

「オブジェクト指向システム分析/設計に関する教科書を読んで, 実際のプロジェクトに適用しようと思っているが, いまひとつ分らないという読者のため」に書いたと, 佐原さんはいうが, 教科書を読んでいない人も管理者も読んで欲しい本である.

そして, 実際この本は「オブジェクト指向技術の衣を着たソフトウェア工学」で, 普通のオブジェクト指向 (以降 OO と略す) の教科書が直接に言及しない計算科学やソフトウェア工学一般を視野にいれた解説が, かえって OO を浮き彫りにしてわかりやすくするのだ. 形式的方法をとる人びとが OO への傾斜をみせるなかで, 佐原さんは OO の実践に形式的方法を取込んでしまう人である.

第 1 章 "前口上"
第 2 章 "方法論"
第 3 章 "分析"
第 4 章 "設計"
第 5 章 "実現 (プログラミング)
第 6 章 "見積り･計画･環境･管理",
第 7 章 "教育"

という章立てのもとの102 個の質問は, 佐原さんのコンサルテーション活動やセミナー経験が躍動しており, 正鵠を射た回答は直載的で, 佐原さんの肉声を聞くようである.

たとえば,「Q6-3: OO 技術をなかなか覚えられないのですが?」に対して次のような答がある.

> "英語をなかなか覚えられないがと軌を一にしていて, ある言語を使ってなんとか生きていける能力水準に達するための学習時間の最低がだいたい定まっている. 英語の場合 400 時間, 日本の中学校では 280 時間である. あなたはそれだけ勉強したか?"

という調子である (以上は紹介者のうろ覚えによる文言である. ぜひ佐原さんの文言で読んで下さい).

分析/設計/実現の部分は, OO を実践してなにかうまくいかなかった人にとっての福音である. それは, 教科書に対する虎の巻になっている. しかし, それだけではない. 分析結果を評価するための Metrics が紹介されている. 設計の項では, 計算量, 機能仕様の記述方法, その正しい実現のための Refinement Calculus に触れ, 設計作業を根元的に理解させようとしている.

管理面についての質疑はユーモアにあふれているが, 日本の現状を鋭く衝くもので, いわゆる管理者に読んでほしい部分である.

教育の部分は, なにしろ中央情報教育研究所で作成された高度情報処理技術者育成指針のプロダクション･エンジニア編のまとめに盡力された佐原さんだから, その識見が自然に迸り出ている. プロダクション･エンジニアとは聞きなれない用語だが, ソフトウェア技術者のことである. 仕事をするソフトウェア技術者としては体得していなければならないものであるのは疑いようがない.

この本も多くの仲間の目にとまることを願うものである.

第10回SEA　教育ワークショップ
ー技術革新に対応する教育方法論･技術のあり方についてー
参加者募集

主催：　ソフトウェア技術者協会･教育分科会 (SIG-EDU)

ソフトウエア技術者協会教育分科会 (SIG-EDU) のワークショップも，早いもので，今年で第10回目を迎えることになりました．これまで多くの方のご参加をいただき，学校教育や企業内教育のあり方，また教育事業の可能性などについて，多面的な議論をして参りました．

今回の記念開催では，「効果的な教育」をメインテーマにあげ，日進月歩の技術革新に対応した教育テーマ，教育方法，教育カリキュラムなどについて議論し，２１世紀に向けた次世代教育像を追求してみたいと考えております．

南の地「八丈」で，夏の残像を楽しみながら活発な議論をしてみませんか? プログラム委員会では，できるだけ幅広く様々な立場の方々の参加を希望しています．日頃ソフトウエア技術者の育成を考えておられる方々の積極的なご参加をお待ちしています．

開催要領

1. 期日：　1996年10月24日 (木) 午後～ 10月26日 (土) 正午 (2泊3日)【現地集合，現地解散】

2. 場　所：　ホテル　プリシアリゾート八丈　(〒100-15　東京都八丈島八丈町三根886　　T:04996-2-5330)

3. 定　員：　20名（定員を越えた場合は，ポジション･ペーパで審査をします）

4. 参　加　費：　65,000円 (SEA会員)　75,000円 (一般)
※期間中の宿泊，会場費，食費，資料代，運営費を含みます

5. 主要テーマ：「効果的な教育」
日々ダイナミックな変動の続く業界動向に対応すべく，如何に効果的/効率的な教育を実践していくかという観点で，次世代教育のあるべき姿を模索します．

6. スタッフ：

実行委員長：　橋本勝 (山一情報システム)
プログラム委員長：　和田勉 (長野大学)
プログラム委員：　川辺正明 (ソフトサイエンス)，河村一樹 (尚美学園短大)，
君島浩 (富士通LM)，篠崎直二郎 (NECソフトウエア)，
杉田義明 (SRA)，中園順三 (富士通BSC)，
牧野憲一 (オムロンソフトウェア)

7. 申込み方法：

申込み締切：　1996年9月15日（日）必着
申込み方法：　所定の申込み書およびアンケートに必要事項を記入して，FAX，郵送，E-mailのいずれかでお送りください．
申込み宛先：　山一情報システム株式会社　システム総括部　プロダクト推進グループ　橋本勝
〒273　千葉県船橋市浜町２－２－２
TEL：　0474-37-3188　　FAX：　0474-37-3360
E-mail：hassy@benten.yis.co.jp

8. その他：　参加者には，全員ポジション･ペーパを提出していただきます．参加申込みをされた方は，９月２６日（木）までに申込先同様，実行委員長（橋本）宛にFAX，郵送，E-mailのいづれかにて送付してください．９月末にプログラム委員会を開催し，参加者へは１０月上旬にその結果をお知らせし，参加案内と資料を送付します．

********** **第10回 SEA 教育ワークショップ 参加申込用紙(締め切り9月15日)** **********

【送付先】

山一情報システム株式会社 システム総括部 プロダクト推進グループ 橋本 勝 宛

〒273 千葉県船橋市浜町２－２－２ FAX：0474-37-3360 E-mail：hassy@benten.yis.co.jp

1. 氏名：_____ 年齢：_____ 性別： □ 男 □ 女

   種別： □ SEA会員（会員番号：_____） □ 一般

2. 所属団体名：_____

   部門：_____ 役職：_____

   住所：〒( ) _____

   TEL ：_____

   FAX ：_____

   E-mail：_____

3. アンケート項目

   (1) ソフトウエア技術者教育について，日頃思っていること

   (2) 当ワークショップに期待すること

   (3) 技術者の育成について，日頃工夫していることや苦労していること

   (4) 当ワークショップで特に取り上げて欲しいテーマ

   (5) 今考えている（もしくは作成ずみの）ポジション･ペーパのテーマ

   (6) その他当ワークショップに対する要望や意見

SEA 1995 年～ 1997 年の主なイベント（実績と予定）

1995	6/13 ～ 16	ソフトウェア･シンポジウム'95	滋賀
	7/20	全国縦断 Forum in 松本 (コンポーネントウェア)	松本
	9/8	SEA 古典セミナー･シリーズ #1(以後毎月 1 回)	東京
	9/19 ～ 22	第 1 回 ソフトウェア･デザイン･ワークショップ	新潟
	9/29	全国縦断 Forum in 広島 (コンポーネントウェア)	広島
	10/9 ～ 11	第 3 回 IWSED (ソフトウェア･データに関する国際 WS)	米国･ Maryland
	10/26	第 1 回 SEA SPIN Meeting	東京
	10/25 ～ 28	Changsha Int'l CASE Symposium(CICS'95)	中国･長沙
	11/16 ～ 18	第 9 回 教育ワークショップ	新潟
	11/30 ～ 12/2	第 1 回 SEA SPIN ワークショップ	御殿場
1996	1/26	Forum (ビジネスを成功させるマルチメディア)	東京
	2/6	TISA Seminar (インターネットとマルチメディア)	
	2/8 ～ 9	ソフトウェア メトリクス特別セミナー	奈良
	2/10	第 16 回 ソフトウェア信頼性シンポジウム	京都
	2/16	全国縦断 Forum in 博多 (コンポーネントウェア)	福岡
	3/1	Workshop with MCC Tour Team	東京
	4/24	Forum (ソフトウェア プロセスの改善)	博多
	4/25 ～ 27	7th TM & 2nd SPIN 合同ワークショップ	対馬
	6/5 ～ 7	ソフトウェア･シンポジウム'96	広島
	6/7 ～ 8	Mini-workshop: "Open 化時代における企業間連携"	広島
	6/27 ～ 29	第 3 回 SPIN ワークショップ	浜松
		**********	
	9/17 ～ 18	Workshop at MCC	米国･ Austin
	10/9 ～ 11	Int'l Symposium on Future Software Technology '96	西安
	10/24 ～ 26	第 10 回 教育ワークショップ	八丈島
	11/28 ～ 30	第 2 回 ソフトウェア･デザイン･ワークショップ	御殿場
	12/16 ～ 17	Int'l Symposium on Conceptual Foundation of SE	東京
1997	6/18 ～ 20	ソフトウェア･シンポジウム'97	博多

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 8,000 円）　96-07

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＿＿＿＿）FAX：＿＿＿＿-＿＿＿＿-＿＿＿＿

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

自宅 TEL：＿＿＿＿-＿＿＿＿-＿＿＿＿

E-mail：＿＿＿＿＿＿＿＿

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）　96-07

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