SEAMAIL

Newsletter from Software Engineers Association

Volume 9, Number 5 October, 1994

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，800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40社を数えます．支部は，東京以外に，関西，横浜，長野，名古屋，九州，東北の各地区で設立されており，その他の地域でも設立準備をしています．分科会は，東京，関西，名古屋で，それぞれいくつかが活動しており，その他の支部でも，月例会やフォーラムが定期的に開催されています．

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

SEAMAIL Vol. 9, No. 5　　1994年10月31日発行
編集人　岸田孝一
発行人　ソフトウェア技術者協会（SEA）
〒160　東京都新宿区四谷3－12　丸正ビル5F
TEL: 03-3356-1077　FAX: 03-3356-1072
印刷所　サンビルド印刷株式会社　〒162　東京都新宿区築地町8番地
定価　500円（禁無断転載）

事務局から

☆

ごぶさたしました．Vol.9, No.5 をお届けします．

☆☆

この夏，事務局兼編集部は，KICS'94，全国縦断 Forum，若手の会，etc とさまざまな活動に巻き込まれて，1つしか物理的肉体のない悲しさ，しばらく忘れていた SEAMAIL 船便シンドロームを再び味あわされることとなりました．

☆☆☆

この号は，ほんらい1ケ月前に出す予定だったのですが，結果的には，いま，渡辺雄一プログラム委員長がまとめている SS'94 in 函館のライブ･セッション･レポート特集号 (Vol.9，No.6-7) とほぼ同時印刷ということになってしまいました．(もしかして，史上初めての2冊同時発送か？)

☆☆☆☆

さて，内容ですが，大場さんからは，「外から見た IBM」と題するタイムリーなコンピュータ産業論を寄稿していただきました．したがって連載の地方情報産業論はお休みです．

☆☆☆☆☆

君島さんの Short Note は，テクニカル･プレゼンテーションの心得に関する具体的なガイダンスです．来年の SS'95 を含めて，これからの SEA まわりのイベントでの若手技術者の方々の発表が改善されることが期待されます．

☆☆☆☆☆☆

山崎さんの講義ノートは，つい数日前のワークショップの会場で入手 (強奪?) しました．形式的仕様記述への関心は次第に現場でも高まりつつありますが，なかなか取り付きにくい感があります．その意味で貴重なチュトリアルだと思います．じっくりとお読みください．

☆☆☆☆☆☆☆

最後の哲学的脱線記は，一部すでにいくつかの ML で流れたものをベースに，一連の哲学騒ぎの記録をコンパイルしました．まあ，秋の夜長の酒のツマミにでもしていただければ.....

☆☆☆☆☆☆☆☆

外から見たIBM

大場 充
(広島市立大学)

暗闇の中に，街灯に照らされた古い農家の納屋のような建物が，うっすらと見えてくる．馬の匂いが残っていそうなたたずまいである．その建物を右に見て右折，ボロ車を駐車場に入れる．駐車場は，いまにも崩れ落ちそうな，サビた車で一杯である．車を降りてみると，かすかに音楽が聞こえてくる．ジャズともウェスタンとも聞こえるピアノのメロディが印象的だ．

街灯で照らし出された歩道を進むと，小さなドアがあった．その木製のドアを開けると，足元から音楽が響いてくる．ボーイにコートを預け，擦り減った木の床を踏み締めながら真っすぐに進む．床がギシギシと音を立てる．幅1メートルほどの暗い廊下を10メートルも行かないうちに，また木のドアの前に立つ．ドアを通して，大勢の男たちの話し声が，音楽と一緒に耳に入ってくる．まるでリップ･バン･ウィンクルの，あの小人たちの酒盛りのようだ．

ドアを開けると，ハシゴのような急な階段が階下に向かって降りている．ゆっくりと階段を降りてゆくと，タバコの匂いが鼻をつく．タバコの煙で部屋の全貌はほとんど見えない．ピアノの上の電灯が若いピアニストの顔を映し出す．金髪の，あまり背の高くない若者である．ほとんど無心でピアノに向かっている．聞かせるためにではなく，自分のために弾いているように見える．スコアはない．即興である．

階段を降りきった正面に，バーのカウンターがある．「バドワイザー」というと，ジョッキの上半分を切り落としたようなグラスに，ビールをついで渡してくれる．クォーターをカウンターに置いてビールを一気に飲み込む．泡が唇に残る．その泡を指でふく．見回すと，直径50センチぐらいの小さな丸テーブルを囲むようにして，いくつかの人の輪ができている．男たちの熱気で，部屋全体が暑くなっている．

はげかかった丸顔のメガネの男の回りに何人かの男たちが集まり，ビールを飲みながら大声で話している．南部訛り，ニューヨーク訛り，ニューイングランド訛りがまざりあっている．「明日の午後，ボブの所へ呼ばれているんだ．何といえばいい，アール」と，南部訛りのはげかかった男が聞いている．「ノーアイディア．何とでもいってくれ」と，背の高い金髪の端正な顔の男が答える．「なんてこった．それがお前さんの答えとは.... ボブはオレを打ち殺すぜ」といい放って，フレッドは席を立った．ビールを取りにゆくらしい．

カウンターへ向かう途中，背の低いイタリア系らしい男がフレッドに声をかける．「やあフレッド，調子はどうだい，明日ボブに呼ばれてるそうじゃないか?」「チクショウ，お前さんの告げ口で，とんだことになっちまった．話が早くなって感謝してるぜ．銀のタマでもブチこんでやりたい心境さ」「それはゴメンだね．何事もビジネスさ．幸運を祈ってるぜ」「オレはお前さんをのろい殺すぜ」「それは相手が違う．呪うんだったらボブにしてくれ．あの石頭(ハードウェア)にな」「負けたよ....」

カウンターに来て，フレッド．「ミラーをくれ．ところでいま何時だ，ジョー」「11時を過ぎたところです」「もうそんな時間か.... オレは帰らなくちゃいかんな．じゃ，ビールはいらん．みんなの分も今夜はオレが払う．いくらだ，ジョー」「4ドル50です」「それじゃ5ドルだ，ほら」「いつもありがとうございます」「おやすみ，ジョー」「気をつけて帰ってください」「帰る前に，フィリップに声をかけて行こう」と，自分にいい聞かせている．

フレッドは，ピアニストの方へ歩いてゆく．「やあ，フィリップ，いつもながらいい音楽を聞かせてくれるナ．腕は，プロ並だ」「どうも，フレッド．これしか楽しみがないから．ストレス解消です」「ところで，コンパイラの方はどうだ．大丈夫か」「遅いですが，着実に進んでいます．多分，大丈夫でしょう」「それはいいニュースだ．期待してるぜ．明日，ボブに会うが，その話をしよう．たまにはいい話でもしないと，打ち殺されちまうからナ」「軍隊からもらった防弾チョッキ貸しましょうか」「アハハハ，大きな穴があいているやつじゃないか．ノーサンキュー，フィリップ．おやすみ」

外へ出ると，空気が冷たい．吐く息が白い．いい気持ちだ．遠くに列車(アムトラック)の音が聞こえる．店のカンバンを見上る．「トレジャーチェスト(宝の箱)」と書いてある．店の前を走っているのは，ルート9．ニューヨークからハドソン川に沿ってオーバニーへ向かう国道である．駐車場を出て左折し，北へ向かって50メートルほど進むと，交差点がある．交差点の向こう斜め前方に，真っ黒な巨大な影が見える．工場である．工場の前の駐車場には，真夜中であるにもかかわらず，ボロ車があふれている．さらに工場のはるか彼方に，キャッツキル山脈の雄大な影が見える．

フレッドの運転する車が，ウィンカーを点滅している．左折だ．赤信号が変わって，青い左折の矢印になる．ブーンと大きな音を立てて，フレッドの車は巨大な黒い影を目がけて，暗闇の中に消えて行った．明日の会議の準備のためである．かのオフィスには，すでに秘書が待機している．原稿を

タイプし，コピーするためである．大きなオフィス･ビルの中は，煌々と照明が輝き，まるで昼間のようだ．フレッドだけではない．何人ものマネージャや技術者が，コーヒーをデスクの上に置いて働いている．1963年10月，ポキプシーにあるIBMのメイン･ビルディングの様子はそんな風であった．

トレジャーチェストは，今もある．いや，あると信じたい．OS/360発祥の地だからである．しかし，時代は変わった．ポキプシーはもはやIBMの頭脳でも心臓でもない．かつての若者たちの誰ひとりとして，いまもIBMに残っているものはない．社内には，かれらの名前をおぼえている技術者は，ほとんどいない．古くは鉄道王バンダービルトの夏の宮殿の町，フランクリン･ルーズベルトの生まれ住んだ町として栄え，戦後はIBMの繁栄とともに栄えたポキプシーの町は，IBMの衰退とともに廃墟になって行く．メインフレーム時代の終焉である．

数年前，当時のジョン･エイカーズ会長は，なんとかしてIBMのメイン･フレームを守ろうとしていた．かれの右腕は，テリー･ローテンバックIBM-US社長であった．テリーは，ハードウェアとしてのメイン･フレームを信じていた．ジョンとテリーが，考え出した「IBM生き残りのシナリオ」は，巨大で動きの鈍い恐竜IBMを，小さな企業の集合体に変えることであった．

大きなIBMを小さな事業体に分割し，それぞれを専門化させ，独立に経営することによって，将来のサービス化経済に備えようとした．それ自体，理論的には間違っていない．ただし，大きなIBMのメリット，組織的なノウハウの蓄積は，小さなIBMでは困難になる．個々の事業体の独立性が，組織的なノウハウの蓄積を阻害するからである．テリーがこの問題を真剣に考えていたのは，事実である．

ある会議の席上，テリーはこう発言した．IBMは，いろいろな意味で知識の宝庫である．われわれは，IBMがこれまでに獲得した知識を整理できてもいないし，新しく得ている知識をお互いに交換することも十分にはしていない．ある報告書によれば，わが国にあるソフトウェア開発組織だけでなく，世界に点在するIBMのソフトウェア開発組織で，似たようなテストの問題があるそうだが，これは組織間の知識の交換が十分でないことの端的な証拠である．

ジョンとテリーの戦略は，最終的には失敗に終わった．一番の理由は，メインフレームの開発にかかるコストである．もはや，メインフレームは，一つの企業の力で設計し，生産できるようなものではなくなっている．このため，メインフレームを開発し，生産しても，それ自体では利益がない．この傾向は，ほぼ10年前から始まった．つまり，その時以来，IBMは，過去の遺産とソフトウェアからの利益を食いつぶして来たのである．

ジョンとテリーのもう一つの失敗は，小さなIBMへの転換を緩やかに進めようとしたことである．小さなIBMには，従来ほどの従業員規模は必要としない．むしろ，少ない従業員で運営しなければならない．つまり，IBMが創業以来守って来た「レイオフをしない会社」のポリシーに反する事態に直面し，希望退職制度に固執したことである．このポリシーが時代の要請に合わないことは，明白である．創業者ワトソンの時代には正しかったことが，ジョンの時代にも正しい保証はない．

国家のマクロ経済から見れば，雇用の保障ができなくなることは，決して喜ばしいことではない．しかし，個々の企業の実態を考えれば，過剰労働力を抱えることは，その企業の生産性を低下させ，競争力を低下させる原因になる．その意味で，レイオフをしないことで，IBMへの忠誠は示せたが，企業としての体力を必要以上に減退させてしまったのである．

ジョンに代わってIBMを再生させる使命を負ったのが，ルー･ガースナー会長である．かれは，ジョンとテリーの戦略を根本からひっくりかえした戦略を取ろうとしている．すなわち，IBMの強みを大きなIBMに見ている．ノウハウの蓄積と知識の共有に関して，大きなIBMの方が有利だからである．しかし，この戦略は，まったく逆の弱みをもつ．それは，組織の効率と専門化に逆行することである．

ルーは，そのような戦略上の弱点をカバーするために，たとえ高収益を上げている部門であっても，IBM本来の(コンピュータの製造と販売)事業から離れている部門を売却することによって，大きくても専門化したIBMを再構築することに全力を投入している．たとえば，かつてのFSDを売却したものそのためであろう．また，一層の人員の削減にも努力している．

ルーとジョンの戦略の違いはそれだけではない．もう一つの重要な違いは，メインフレームに対する考え方の違いである．ルーは，明らかにメインフレーム時代は終焉したと見ているようである．メインフレーム部隊の指揮官ニック･ドノーフリオを，ルーの直属の部下にしていないことから，このことが推察される．ルーの就任直後，ルーは，ニックを彼のオフィスに呼んで，メインフレーム･ビジネスの総括的報告を聞いた．

ニックとかれのスタッフたちが練りに練ったOHPの資料をニックがプロジェクターに乗せ，スクリーンに映し出した瞬間，間髪を入れずルーがいった，「チャートは要らない．言葉で説明してくれ」と．一瞬，オフィスに緊張が走った．ニックは，明らかに動揺した，というより色をなした．ニックは，さもルーの言葉が聞こえなかったかのように，無視をしてプレゼンテーションを進めた．今度は，ルーが色をなし

た．もう一度，今度は命令口調でいった．「言葉で説明しなさい」．さすがのニックも，今度は無視できなかった．「私の話を理解していただくための資料です．これは，必要なんです」．これが，二人の宿命的な出会いだった．

その後ルーがポキプシーを訪れたとき，社員の一人が質問した．「あなたは，IBMの大型システムについてどのようなお考えをお持ちですか」．ルーは，躊躇せずに答えた．「今日，IBMが存在し続けられるのは，大型システムの資産があるからてす．現実に，IBMの利益の大部分は，大型システムが稼ぎ出しています」．しかしこの答えとは違って，ルーは45歳の若いエリート，ニックを遠ざけた．それは，大型システムに固執する限り，IBMはローコスト･プロデューサーにはなれない．そして，これからはローコスト･プロデューサーしか生き残れない，という読みがあるからであろう．

ルーの経営の重心は，ワークステーションやPCにある．ルーが断行した一連の改革によって，IBMの小型システム事業部は，ローコスト･プロデューサーとして蘇りつつある．開発コストは，従来の半分になり，製造コストも2/3ぐらいにはなるのではないか．当然，利益は従来の1/5ぐらいにまで縮小するであろう．それでも，世界一安くて品質の良い製品を生産できれば，IBMは蘇るに違いないとルーは考えているのだろう．このことが実現するかしないかは，ルーの手腕にかかっている．

ルーの戦略は，間違いではない．開発費用が一膨大で，リスクの大きな大型システムを開発するよりも，開発費用のほとんど要らない小型システムの方が，確実に業績を上げられる．本来，付加価値の高い大型システムの開発の経験が長いIBMには，小型システムの開発のノウハウが不足している．ムダが多いのである．このムダを取り除くことは，あまり難しい課題ではない．原価管理を徹底すればよいからである．

小型システムにシフトしたにもかかわらず，ルーはIBMの研究部門を身売りせずに置いている．付加価値の高い大型システムの開発のためには，素子からソフトウェアまで，すべての分野に関する高度な知識が必要になる．しかし，価格が勝負のワークステーションやPCを開発するのに，基礎研究は要らない．では，なぜルーは金食い虫の研究部門を手放さないのだろうか．

多分，ルーの小型システムへのシフトは，現実的な「その場しのぎ」の戦略なのではないだろうか．ルーは，一度手放した技術は二度と戻らないことを知っている．そのような意味のことをかれ自身がどこかでいっている．ポキプシーを手放し，フィッシュキル（半導体の工場があるポキプシーの隣町）を手放し，研究部門を手放したら，IBMに残る技術は何もなくなる．ルーは，それを知っている．

大型システムは，かつてのように繁栄することはなかろう．これは，だれでも予想していることである．しかし，恐竜は絶滅しても爬虫類が残ったように，大型システムと小型システムの「棲み分け」が起こるだろう．また，鳥類が恐竜から進化したように，新しい技術が大型システムの技術から生まれるかも知れない．そう考えると，今日まで培って来たIBMの技術やノウハウを，あっさりと捨て去るわけには行かない．

ルーのプログラムは，次のようなものであろう．1980年代に拡大したIBMの戦線をまず縮小する．これによって，無駄な損失を最小限に止める．一時的に，小さな戦力である程度の成果を期待できる「小型システム戦線」に全勢力を投入する．他の戦線で残ったものは，最小の兵力で防御に徹する．小型システム戦線で戦闘が続いている間に，新しい技術を発見し，育てる．小型システム戦線での戦局が決定したら，新しい技術を投入して新しい戦線を開く．

ジョンとルーの違いは，キャッシュ･カウ(金のなる木)と呼ばれる大型システムをどう見て，自分の戦略にどう組み込むかの違いである．ジョンは，IBMを分社化によって解体し，そのいくつかが自然淘汰の末，残ればよいと考えた．市場原理の導入は，困難ではあったが，21世紀へ向けての壮大な実験であった．IBMの内部は，どんな社会主義国よりもうまく管理された計画経済にもとづいて運営されて来たのである．それを，市場経済システムに移行させようとしたのが，ジョンの野望であった．

ルーは，ジョンに比べればずっと保守的な方法でIBMを変えようとしている．経済システムはそのままにして，軍備を中心とした歳出を徹底的に押さえ，国境の防御線を後退させ，国民の生活レベルを下げることによって，経済危機を乗り越えようとしている社会主義国に似ている．ジョンは，市場経済の自然淘汰という大自然の英知に全てを任せようとした．反対にルーは，すべてを自分の知恵で完全に管理しようとしている．

この2人の対照的な経営者のどちらの選択が正しいかの結論を得るには，少なくとも15年はかかるのではないだろうか．一時的に，ルーの方が成功するのは単なる確率の問題である．長期的に，2人の選択がどのような結果を生むかは，今後の世界がどう変わって行くかにかかっている．21世紀が大方の予想に反して20世紀の延長であれば，ルーが正しかったことになる．そして，IBMは21世紀でも世界の一流企業として繁栄しているだろう．

逆に，21世紀の経済が本当に資本でなく知識を基礎としたものになれば，ジョンに先見の明があったことになろう．そして，IBMは存続していたとしても，歴史だけの企業になっているだろう．世の中の多くの企業は，20世紀の伝統

的な計画経済的な企業経営を止め，新しい市場経済的な企業経営を導入しなければ生き残れなくなっているだろう．さらに言えば，もしそうだとすれば，20世紀的な意味での大企業は，もはや存在し得なくなっているだろう．

トレジャーチェストでの，あの若いソフトウェア技術者や管理者たちの会話は，昭和40年代の日本の発展を担ったハードウェア技術者や管理者の，赤提灯での会話にどこか似ている．そんなバイタリティーは，いまの日本やアメリカにはない．そして多分，日本とアメリカ以外のどの国にもなかったし，これからもありえないのではないか．ソフトウェアという新しい産業が生まれる瞬間や，TQCという新しい管理手法が生まれる瞬間の，技術者と管理者のエネルギーの蓄積と爆発だったのではないか．

あのOS/360から20数年の歳月の後，当時の若いピアニスト，フィリップは，私の前に立ってこういった．「IBMを去るに当たって，君たちが私の最後の日に立つべき場所を，この場所を与えてくれたことを感謝します．われわれは，ここに集い，時間を忘れて議論した．あの瞬間瞬間が，われわれの人生のすべてだったといってもいい．いまも忘れない、ブルックスがいったこと．ウィーラーの発言．ピエトラサンタのジョーク．ありがとう」．すでに，あのIBMは歴史である．

先日，日経の朝刊を見ていると，「日本IBMが分社化した子会社を再統合する」との記事が目に入った．妻は，「日本IBMも悪くないみたい」と，分かったようなことをいう．多くの社員にとっては，短期的にはよいことかも知れない．しかしどう見ても，これはIBMの戦略とは関係がない．社長の交代での人事のゴタゴタを一時的に解消しようと苦し紛れにやった「のれん分け」を，社長の権力が安定したので元の鞘へ収めただけではないか．いや，より大きな権力が欲しくなったのかも知れない．何と次元の低いことか．戦略も何もない．

IBMでも実施しなかった分社化をし，軌道に乗らないうちに今度はそれを統合する．この分社化と統合で発生するコストは，誰が支払うのか．従業員か，ユーザか，その両方か．いずれにせよ，経営者本人たちではあるまい．世界一労働コストの高い日本で，高々1億の小さな市場を標的にした日本語パソコンを開発したり，大型システムを製造したり，理屈では考えられないことをしているだけではもの足りずに，今度は分社した子会社を統合すれば，ただでさえコスト高の製品をますますコスト高にする．

これが世界に冠たる経済を動かしている日本の大企業の経営者の常識なのだろうか．だとすれば不思議な話である．いま，日本の経済が一流であることに疑問の余地はない．しかし，こんなことが許されるのならば，日本の企業経営はどう見ても三流以下である．だとしたら誰が一流の経済を引っ張っているのか．技術者か．そうかも知れない．いや，そんな者はいないのかもしれない．経営者も，管理者も，技術者も，現場の工員さんも，パートの人も，みんなそれなりに，その日その日を精一杯生きているだけなのかもしれない．

別に大層な戦略や深い考えがある訳ではなく，日々の問題を，たとえそれが単なる権力闘争の問題であっても，地道に解決し続けていたら，いつのまにか自分たちの知らない間に経済だけが一流になっていたのかも知れない．この幸運な偶然がいつまで続くのだろうか．別に能力がある訳でもない人が，何かの偶然でIBMの子会社の社長になっていた．親会社がよければ，その七光りで，ほんとうの能力とは関係なく子会社も一時は繁栄する．いや，目先の利く子供なら成功している親を積極的に真似るかもしれない．単なる猿真似である．別に自分なりの考えがあった訳ではない．

年金生活者になった親(アメリカ)には，かつての莫大な財産もヒラメキもない．本来ならば，世代交代が必要なのである．親に代わって子供が自分で考え，自分で新しい道を切り開いて行かなければならないのである．かつて大英帝国に代わって，独立した若いアメリカが自分の力で新しい産業社会を建設したようにである．NCRの文化を継承したIBMは，大型コンピュータという新しい分野で成功した．その文化を継承した日本IBMは，親の衰亡を横目見ながらも，まだそのスネをかじってモラトリアムを決め込んでいる．

もしかすると，末期ガンで苦しみながらもまだ現役で頑張ろうと最期の努力している親に向かって，ベンツを買うから頭金を出してくれとせがんでいるバカ息子のように，何もわかっていないのかも知れない．親が倒れて，医者から「あと1カ月の生命です」と宣告されて初めて，ことの重大さに気づくのである．親の体重の変化に注意し，顔色を見ていれば分かることであり，1年前に「病院へ行って見てもらった方がよい」と勧めていれば，最悪の事態を避けられたにもかかわらず，親が一代で築き上げた財産(知識)を，すべて手放す結果になる．これは日本IBMだけの問題ではない．

登壇発表の指針

君島 浩
(富士通ラーニングメディア)

論文の作り方について，先日，共立出版のbit誌に投稿した．今回ここでは，函館のソフトウェア・シンポジウム'94の体験をもとにして，登壇発表に重点を置いて指針を述べる．時間オーバーや内容のレベルが低いという問題を，どうしたら防ぐことができるだろうか．

1. 材料の洗い出し

論文は面白いのに，発表した内容のレベルが低いという問題がある．これは材料の洗い出しと時間配分の問題である．発表準備作業を，インプット(材料の洗い出し)，プロセス(構成の設計)，アウトプット(OHPシートの作成)に分けていうと，慣れない人はいきなりアウトプットに取りつく．重要でないことを延々と書き出して，重要な材料を落としてしまう．

したがって，まず第一のポイントは，自分が発明したことを克明に洗い出して，重要度の低い項目を削っていくというインプット作業を行なうべきである．25分間の発表は短いので，あれもこれもと欲を出さずに，ばっさりと削るべきである．

2. 時間を見積もる

長時間の発表の場合には，OHP 1枚を5分と見積もる．25分間の発表の場合には，0HP 1枚を3分と見積もって，8枚+表紙1枚にする．

3. ストーリーを設計する

プロセス(構成の設計)では何をするか．古典的な論文作成法の本には，ストーリーを起承転結にせよ，と書いてある．起承転結は漢詩の定石であって，長い報告には向かない．起承結と一貫したテーマを話している山場に，「転」が入ってよいわけがない．起承結という3部構成で進めるのが正しい．しかし，起(前置き)が長すぎてはいけない．私の函館での発表は，前置きは5分間で終わった．OHP 2枚分である．

発表に慣れていない人は，聴講者がその専門的な発表について前提知識がないことを心配して，イントロダクションを長々とやる．25分間の発表で，15分間ぐらい前置きに費やしてしまう．中には会社の紹介から始める人もいる．これが素人の浅はかさであって，本論の方がもっと専門的で難解なのだから，本論にたっぷり時間をかけるべきなのである．前置きを長々とやれば，本論を説明する時間がなくなり，発表全体のレベルが低いと評価される一因にもなる．

米国人の中にはわざと前置きにだけ時間を使って，内容に触れないという宣伝型の発表をする人がいる．これは技術発表では邪道である．前置き(前提条件や方針)ではなく，本論(自分がやった業績)を発表するのが，視聴者へのエチケットであろう．

起承結の指針は，必要ではあるが十分ではない．承(本論)の部分が長いのだから，ここの構成設計の指針が必要である．インプット(洗い出し)のところで材料を選抜して，それを種類別に体系化する．その体系をOHPのストーリーにマッピングするだけでよい．構成はインプットで決まるのである．現状分析，問題点指摘，改善方針立案などといった時間的ストーリーを用いるのは，前置きが長くなる原因である．技術をその体系のまま，淡々と並べればよい．

4. OHPシートを作る

OHPシートは映写してみて，見えなければ意味はない．最低，通常の文書の活字の縦横2倍(6mm x 6mm)は必要である．これは練習で映写して確認すれば分かることだ．10メートル離れて字が明確に見えるかどうかを確認する．通常の文書の活字をそのまま使う人が少なくないが，いったいどういう神経を持っているのか，理解に苦しむ．相手に対する思いやり感覚が欠けているとしかいいようがない．

通常の大きさの活字を使うと，OHPの中に多数の字句を盛り込む結果になる．とても3分間では済まなくなってしまう．これも時間オーバーの原因である．

5. 語りを作って練習する

発表の初心者は，語りの原稿を作るべきである．初心者が語りも作らず，練習もしないというのでは，時間オーバーになるのは当たり前であろう．私も初体験の発表の練習では2倍の時間がかかってしまった．語りの速度はおよそ1分間300字だから，OHP 1枚＝3分間用で900字である．

練習すれば，発表にかかる時間は確実にわかる．時間をオーバーしたら，語りの原稿を切り詰める形で，文書上で直すべきである．原稿と練習を併用すれば，時間オーバーする心配はない．

定石を知ったら，あとは経験で上達するのみである．失敗を恐れずに，場数を踏んで欲しい．

仕様記述入門ノート

山崎 利昭
(Free)

北陸先端科学技術大学院大学の二木厚吉先生を中心に，実行可能な仕様記述言語とその支援環境の開発を目指してIPA技術センターで進められているCafeOBJプロジェクトでは，その成果を広く発表するとともに，欧米に比べて日本ではやや遅れている形式的技法の普及·実用化を意図して，去る10月21日に第1回のCafeOBJワークショップを開催しました．

このワークショップでは，まず，山崎利治さんから，形式的仕様記述言語全般についてのお話があり，続いて疋田輝雄先生(明治大)によるZの，そして二木先生によるCafeOBJの，それぞれチュトリアルを受けて，柳生孝昭(日本ユニシス)，玉井哲雄(東京大)，大槻繁(日立)の各氏によるパネル討論が行われました．あいにくの悪天候にもかかわらず，会場の定員60名ぎりぎりの盛況でした．

ここに収録したのは，山崎利治さんが，とくに今回の講義のためにまとめられたノートです．御本人は「殴り書きなので，書きなおさなければ....」といっておられたのですが，何しろ遅筆で有名な方ですから，そんなわがままを聞いていたらいつ完成原稿がいただけるかわかりません．無理矢理強奪して掲載する次第です．

(編集部)

目 次

1 はじめに

正しく安全なソフトウェアを作成する切り札として形式的方法が注目されている。ソフトウェアが社会生活の利便に寄与すると共に恐ろしい公害源にもなってきている。最近ではこの見地からソフトウェアに法的規制を加える例も見られる。トロント郊外の原子力発電所の緊急停止システム、パリ地下鉄の運転システムや米国の中距離航空機に搭載を義務付ける衝突回避システムなど安全第一 (safety-critical) システムがそうである。これらのすべてがその安全検証のために形式的方法を採用している。

ソフトウェア開発の出発点が仕様作成である。形式的方法はあらゆる議論を厳密な証明付きで行なおうとするから仕様記述に対しても特別の工夫と記法を用意している。このノートは形式的方法での仕様記述とそのための言語の手短かな紹介を意図している。仕様言語 (specification description language) はだいたい抽象データ型を記述単位としてその記法と、いくつかの単位をまとめて大きな仕様とするための機構を持っている。プログラム言語の場合と同じように、仕様言語も仕様自体と記述方法についての知見の蓄積とともに数多くが検討・作成されている。複雑な通信プロトコルの記述のために考案された言語 LOTOS が国際規格に制定されて以来、VDM や Z などの仕様言語や方法の規格化も進んでいる。

2 形式的方法・仕様・実現

形式的方法の目的は正しいプログラムを作成することである。正しいプログラムとは仕様があってプログラムがその仕様を満たすときにいう。仕様とはプログラムに対して成立しなければならない性質の言明をいい、プログラムを直接に明示し規定するものである。また同時にプログラムを必要とした課題対象自体をプログラムを離れて議論するための、いわば、応用領域理論 (application domain theory) の提示でもある。この意味で、プログラムが関数を表示すると考えるなら、仕様は理論あるいはそのモデルを表示するといえる。

形式的方法は “形式的” と “方法” という二つの用語を含んでいる。さまざまな要素を統一的な連関や構造に結び付けるものを形式というが、ここではプログラムの仕様作成とプログラムとして仕様を実現するために必要な分析・推論に計算や理論を利用することと考えていい。また方法とは一時的な術や方便ではない客観的な認識と論理による目標達成の道程をいう。プログラム開発の形式的方法は、したがって、そのための多くの原理・理論を持ち、この原理の適用手段である技術が存在し、さらにその適用を円滑にする道具を用意してそれらを選択利用する手順を提供する。形式化は一般に対象についての理論展開、つまり、対象が持つ性質の発見・解析・検証をやさしくし、そのような作業の計算機による支援を可能にする。計算機はそこで得られた知識や成果の再利用にも貢献する。

さてプログラムのほかに仕様が必要である大きな理由に実用プログラムが Cobol、Fortran、C、Ada などの命令型プログラム言語で書かれている事実がある。代入文

や制御文を持つ命令型言語はいわゆる参照が不透明で、プログラムの正しさなどを証明しようとするときホーア論理などによらざるを得ない難しさがあるからである。そうであればはじめからプログラムの満たすべき条件を仕様として書き仕様を満たすことを証明しながらプログラムを作成すればいい。この考えはさまざまに具体化できるがその詳細には触れない。正しいプログラムを作成しなければならない意を込めて、プログラムの作成を仕様の実現といったりすることもある。

このノートでは、まず、仕様の記述例を二つ紹介する。一つは対象をそれが持つ性質の面から記述し、その持つ構造を明示しない方法で、性質指向記述である。もう一つは対象の構造をはじめから明示して、その構造に基く演算によって対象を記述する、構成指向ないしはモデル指向記述である。

3 仕様記述 — 性質指向

まず簡単なプログラム作成問題を考える。

> 1 から n までの自然数のうち、その 3 乗が回文になるものをすべて求めよ。回文とは文字列で左から読んでも右から読んでも同一列になるものをいい、自然数は 10 進表現する。

たとえば、$1^3 = 1$、$2^3 = 8$、$7^3 = 343$、$11^3 = 1331$、...、$2201^3 = 10662526601$、... などが求める自然数である。こんな例題なら課題はすぐに判り、プログラムも直接書けるが、プログラムの正しさを証明せよといわれ、執拗に厳密にプログラムを書こうというのである。まず仕様を書かなければならない。目には目をで、数学の便利な記法を流用しよう。

課題の核心は「10 進数字列が回文である」を判断することである。考えるべき対象は自然数とその集合、10 進数字、その有限列である。これらの集合を Nat、$Nat\text{-}set$、D、$D\text{-}seq$ と記そう。自然数を 10 進数字列に変換する関数 $conv : Nat \to D\text{-}seq$ と、数字列上の述語「数字列が回文である」$palin : D\text{-}seq \to Bool$ を定義すれば課題の仕様は次のように書ける (図 1)。

仕様　回文 =
　入力　$n : Nat$
　出力　$ans : Nat\text{-}set$
　　$= \{ x \in Nat \mid x \leq n \wedge palin(\ conv(\ x^3\ )\ )\ \}$

図 1

ここで、$\{ x \in A \mid p(\ x\ )\ \}$ は集合の内包表記で、「性質 $p(\ x\ )$ を満足する A の要素 x の集合」を表す。$p \wedge q$ は述語 p と q との連言 (論理積) で、「p かつ $qrfloor$ を意

味する。つまり、上の ans は、「自然数 x の集合で、$x \leq n$ かつ x^3 の 10 進表現が回文であるもの」と読める。

つぎに関数 $conv$ と述語 $palin$ を定義しなければならないが実は少し困る。というのは数字列についての「何か」がなければこれらの定義の仕方がないわけである。$conv$ と $palin$ の定義作業を一旦中断して数字列についての仕様を書こう。構造的プログラミングを通してこの扱いをよく知っている。数字列を抽象データ型 (abstract data type) と考えるのである。抽象データ型とは集合団とその上で定義した演算群のことで、演算群はある法則を満たすものである。このような対象を数学で一般に代数 (algebra) というので、抽象データ型は対象を代数と考えることである。このように考える利点は情報隠蔽として有名である。

さて、数字列は数字の並びである。数字を含まない空列 [] は数字列であり、数字 d を数字列 s に繋いだもの $d \triangleright s$ は数字列であると帰納的に定義できる。数字列 s と数字列 s' とを繋ぐ演算 $s \circ s'$ も用意し、数字 d 自体を一字から構成された数字列とみなす演算 $[\, d \,]$ も導入しておこう。以上を形式的に書いた例が図 2 である。

仕様 $D\text{-}seq =$

種 $D, D\text{-}seq$

演算 $[] :\rightarrow D\text{-}seq$

$[\_] : D \rightarrow D\text{-}seq$

$\_ \circ \_ : D\text{-}seq \times D\text{-}seq \rightarrow D\text{-}seq$

$\_ \triangleright \_ : D \times D\text{-}seq \rightarrow D\text{-}seq$

公理 $\forall d \in D \bullet$

$\forall s, s', s'' \in D\text{-}seq \bullet$

$[] \circ s = s$

$s \circ [] = s$

$(\, s \circ s' \,) \circ s'' = s \circ (\, s' \circ s'' \,)$

$d \triangleright s = [\, d \,] \circ s$

図 2

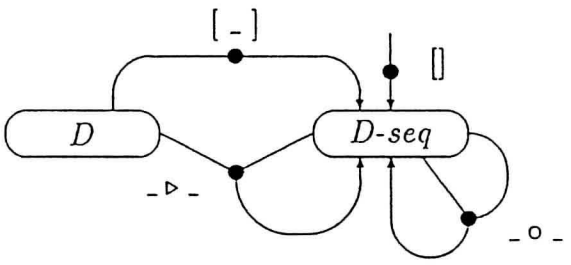

図 3

種以下にこの抽象データ型が必要とする集合を書く。種 (sort) とは集合のことである。

演算以下にはこの抽象データ型が持つ演算を列記する。ここで演算の定義域と値域を付記する。矢印→の左辺が定義域、右辺が値域である。$[] :\rightarrow D\text{-}seq$は、演算 [] が定義域を持たず値域を $D\text{-}seq$とする、つまり $D\text{-}seq$の定数であることを示している。演算子で下線_の位置にその引数を書く。

公理以下には演算が満たすべき関係を等式の形で列挙する。$\forall d \in D \bullet$ は「集合 D の任意の要素 d に対して」、$\forall s, s's'' \in D\text{-}seq \bullet$ は「集合 $D\text{-}seq$の任意の要素 $s, s's''$ に対して」と読む。$s \circ [] = s$ は任意の s に [] を○演算したものは s に等しいと読め、$d \triangleright s = [\ d\ ] \circ s$ は任意の数字 d に任意の数字列 s を▷ 演算したものは 1 字からなる数字列 $[\ d\ ]$ に数字列 s を○演算した結果に等しいことを示している。この間の事情を図示したものが図 3 のポリグラフである。長円は種を表し、矢印の根は定義域を、矢先は値域を示している。

図 2 のような記述方法は代数を書いていることから代数的記述 (algebraic description) という。対象の構造を直接表現しないので歪のない実現が期待でき、この意味から性質指向 (property-oriented) 記述ともいう。

さて、もとの例題に戻ろう。関数 $conv$と述語 $palin$ は抽象データ型 $D\text{-}seq$ によっていまや直ちに書ける。図 4 である。

$$
\begin{array}{l}
palin : D\text{-}seq \rightarrow Bool \\
\quad palin(\ s\ ) = (\ s = rev\ s\ ) \\
\\
rev : D\text{-}seq \rightarrow D\text{-}seq \\
\quad rev\ [] = [] \\
\quad rev\ d \triangleright s = (\ rev\ s\ ) \circ [\ d\ ] \\
\\
conv : Nat \rightarrow D\text{-}seq \\
\quad conv\ 0 = [\ 0\ ] \\
\quad conv\ x = con(\ x \div 10\ ) \circ [\ x \bmod 10\ ] \\
\\
con : Nat -> D\text{-}seq \\
\quad con\ 0 = [] \\
\quad con\ x = conv\ x
\end{array}
$$

図 4

述語 $palin$ は「数字列が回文である」であったが、数字列を逆順に並べ換える関数 revを用意して、$s = rev\ s$ と定義できる。関数 revは引数を場合分けして定義す

る。引数の数字列が [] ならば []、$d \triangleright s$ ならば $rev\ s \circ [\ d\ ]$ とする。このような引数のパターン照合による場合分けを用いた関数定義は関数型プログラミングの常套手段である。ここではまた関数をその引数を含む式によって直接定義した。その定義式のなかに定義している関数を引用している。このような帰納的定義は関数型プログラミングでは本質的である。

定義域が有限集合であれば、引数の値と対応する関数値を組にしたグラフとしての定義方法も利用できる。わかりやすく強力な定義方法として次の例が示すような間接的な方法もある。

$$\sqrt{\ }: Real \rightarrow Real$$
$\sqrt{x}$を yとして：
前件　$x \geq 0$
後件　$|\ x - y^2\ | < 10^{-6}$

これは非負の実数の平方根を求める関数$\sqrt{\ }$の間接的な仕様記述である。前件 (precondition) とは関数適用前に成立しているべき条件 (述語) であり、後件 (postcondition) とは適用後に成立する条件である。

プログラム作成の形式的な方法は以上のような仕様から出発する。そこから抽象データ型の実現に向かう。抽象データ型を具体的に表現し、演算をその表現に対して実現し、そこで公理を成立させることを証明する。この過程を段階的に実施し実用プログラムを作成するわけであるが詳細は省略する。

抽象データ型の代数的仕様記述についてここでまとめておこう。この記述は図式的に次のように書ける。

仕様 = 標 (種、演算) + 公理 (+論理系)

仕様記述に必要な集合を種といい、演算は種の有限列と種との組を添え字として持っている。この種集合と演算集合を組として標 (signature) という。ここで仕様化対象自体とその性質を記述する基本語彙、つまり、定数、関数、述語が用意できたわけである。対象とその性質を記述する文 (論理式) はここでの論理的推論を展開するための論理体系に依存するが、代数的記述ではこの論理体系として等式論理 (equational logic) を採用したわけである。ここでは演算によって構成される項の上の等式を公理として与えた。抽象データ型は公理を満足する演算を持つ集合団、つまり、多種代数 (many-sorted algebra) のことであり、この集合団をモデルという。

ところで、図2に書いた抽象データ型とは何だろう。このモデルはいくらでもある。

例1 種 s の項を次のように定義する。

1. 種 s の集合の要素は種 s の項である。

2. t_iを種 S_iの項とし ($i = 1, 2, \ldots, n$)、演算$\omega : S_1 \times S_2 \times \cdots \times S_n \rightarrow S$ があるとき、$\omega(\ t_1, t_2, \ldots, t_n\ )$ は種 s の項である。

抽象データ型を定義するすべての種の項全体を考える。項に対して演算が作用でき、結果がまた項になる。つまり、項全体が代数になり、これを項代数 (term algebra) という。また等式系が項代数の上の同値関係を与えるから、同値類に属する項は等しいとみる商代数 (quotient algebra) が得られる。これは図 2 の仕様のモデルになる。

例 2 $D = \{\ 0, 1, \ldots, 9\ \}$ であるとする。すべての $d \in D$に対して $[\ d\ ] = []$ とする。$D\text{-}seq = \{\ []\ \}$ は公理群を満たし、これも仕様のモデルになる。

例 3 $D = \{\ 0, 1, \ldots, 9\ \}$ とする。$D\text{-}seq = Nat$ とし、$[\ d\ ]$ は自然数とみた d、$[]$ は自然数 0、$s \circ s' = s + s'$(自然数の加算)、$d \triangleright s = [\ d\ ] + s$ は $D\text{-}seq$の公理群を満たし、やはりモデルになる。

例 1 が考えた抽象データ型 $D\text{-}seq$であるとするとき、これを始代数意味論 (initial algebra semantics) という。これは二つの項 t、t'に対して、公理から $t = t'$が証明できないなら、$t \neq t'$と考えるという態度である。

例 2 が考えた意味であるとするとき、終代数意味論 (terminal algebra semantics) という。これは二つの項 t、t'に対して、公理から $t \neq t'$が証明できない限り $t = t'$と考えるもので、たとえば、$d \neq d'$であっても $[\ d\ ] \neq [\ d'\ ]$ が証明できないので $[\ d\ ] = [\ d'\ ]$ と考えるのである。

例 3 は緩意味論 (loose semantics) が意味するモデルの一例である。つまり公理を満たす演算を持つ集合団であれば何でもいいとする態度である。同一演算を持ち同一公理群を満足する代数全体を代数族 (variety of algebra) ということがある。

仕様が意味するものについてはあとで再度議論する。

4 仕様記述—モデル指向

計算機を利用したシステムを、その外部および内部で発生した事象に応答する状態機械と考えていい場合が多い。このような例に対してモデル指向の仕様記述を行なってみよう。モデル指向 (model-oriented) は、性質指向が対象の構造を明示しないで間接的に定義するのに対し、具体的に対象の構造を明示して陽に定義する。この場合、その構造はプログラム言語が持っているようなデータ型を用意して、それらを用いて記述する。次の課題を考えよう。

スーパーマーケットが客に信用カードを発行して、それによって掛売りをしようとしている。これを支援するシステムを作れ。

状態変数を持つ状態機械を念頭に抽象データ型を定義しよう。図 5 がそれである。

仕様　スーパーマーケット =
　型　客、商品
　　価格、金額、売掛金 = Int
　　請求書 = 客 × 売掛金
　変数　$ptable$: 商品 $\xrightarrow[m]{}$ 価格
　　$cfile$: 客 $\xrightarrow[m]{}$ 売掛金
　演算　表作成 :→ **write** $ptable$
　　帳作成 :→ **write** $cfile$
　　カード発行 : 客 → **write** $cfile$
　　価変更 : 商品 × 価格 → **write** $ptable$
　　販売 : 客 × 商品 → **read** $ptable, cfile$ **write** $cfile$
　　請求 : 客 → **read** $cfile$, 請求書
　　入金 : 客 × 金額 → **read** $cfile$ **write** $cfile$
　　カード中止 : 客 → **write** $cfile$
　公理　$\forall c$: 客, m : 商品, p : 価格, a : 金額 •
　　表作成 ≡ $ptable := [\,]$
　　帳作成 ≡ $cfile := [\,]$
　　カード発行 (c) ≡ $cfile := cfile\dagger[\, c \mapsto 0\,]$
　　価変更 (m, p) ≡ $ptable := ptable\dagger[\, m \mapsto p\,]$
　　販売 (c, m) ≡ cfile :=
　　　let $p = ptable(\, m\,)$ in
　　　let $a = cfile(\, c\,)$ in
　　　$cfile\dagger[\, c \mapsto a + p\,]$
　　　前件 $c \in$ dom $cfile$
　　請求 (c) ≡ ($c, cfile(\, c\,)$)
　　　前件 $c \in$ dom $cfile$
　　入金 (c, a) ≡ cfile :=
　　　let $b = cfile(\, c\,)$ in
　　　$cfile\dagger[\, c \mapsto b - a\,]$
　　　前件 $c \in$ dom $cfile$
　　カード中止 (c) ≡ $cfile := cfile \setminus \{\, c\,\}$
　　　前件 $c \in$ dom $cfile$

図5

型以下にここに出現する対象の型を定義する。型はプログラム言語でいう型と考えていい。型の明細を書かなくてもいい。客、商品はこの抽象水準では明示しないというつもりである。さらに詳細化した段階で補えばいい。

変数以下は状態変数の宣言である。ここに型を書いてもいい。状態変数 $ptable$ は

商品 $\underset{m}{-}$ 価格型でこれはあとで説明する有限写像型である。変数はその型の値を保持し、代入演算:=を持っている。

演算以下に演算とその型を列挙する。変数を持つ場合、それを参照・更新する演算には矢印→の右に**read** 変数、**write** 変数と書いて、演算の持つ副作用を注意する。

公理以下には代数的記述の場合と同様にそれを書く。

ここで有限写像型について説明する。A、Bを集合、$A \times B$をその直積集合とする。$A \times B$の部分集合rをAとBの関係という。関係rに対してその定義域 dom と値域 rng を次のように定義する。

$$\mathrm{dom}\ r = \{\ a \in A \mid \exists b \in B \bullet (\ a, b\ ) \in r\ \} \subset A$$
$$\mathrm{rng}\ r = \{\ b \in B \mid \exists a \in A \bullet (\ a, b\ ) \in r\ \} \subset B$$

AとBの関係rが次の (*) を満たすとき、始域 A 終域 Bの写像といい、$r : A \to B$と書く。

(*) $\forall a \in \mathrm{dom}\ r \bullet card\ \{\ b \in B \mid (\ a, b\ ) \in r\ \} = 1$
($card\ s$ は集合 s の濃度 (要素数))

さらに、$card\ \mathrm{dom}\ r < \infty$、つまり、その定義域が有限集合であるとき、$r$を始域 A 終域 Bの有限写像 (finite mapping) といい、$r : A \underset{m}{-} B$と書く。また $(\ a, b\ ) \in r$に対して、$b = r(\ a\ )$、$[\ a \mapsto b\ ]$ などと書く。$[\ a \mapsto b\ ]$ を写像片 (maplet) ということがある。$r = \phi \subset A \times B$のとき、空写像といい、$[\ ]$ と書く。有限写像に対して外延表記や内包表記;

$$[\ a_1 \mapsto b_1, \ldots, a_n \mapsto b_n\ ]$$
$$[\ a \mapsto b \mid a \in A, b \in B \bullet p(\ a, b\ )\ ]$$

を許す。有限写像に対して次の演算を設ける。

$$\_ \setminus \_, \_/\_ : (\ A \underset{m}{-} B\ ) \times A\text{-}set \to (\ A \underset{m}{-} B\ )$$
$$r \setminus s = [\ a \mapsto r(\ a\ ) \mid a \in (\ \mathrm{dom}\ r\ ) - s\ ]$$
$$r/s = [\ a \mapsto r(\ a\ ) \mid a \in (\ \mathrm{dom}\ r\ ) \cap s\ ]$$
($A\text{-}set$ は集合 A の有限部分集合の全体、$s - s' = \{\ e \in s \mid e \notin s'\ \}$)
$$\_ \dagger \_ : (\ A \underset{m}{-} B\ ) \times (\ A \underset{m}{-} B\ ) \to (\ A \underset{m}{-} B\ )$$
$$f \dagger g = [\ a \mapsto b \mid a \in \mathrm{dom}\ f \cup \mathrm{dom}\ g \bullet$$
$$b = \mathrm{if}\ a \in \mathrm{dom}\ g\ \mathrm{then}\ g(\ a\ )\ \mathrm{else}\ f(\ a\ )\ \mathrm{fi}\ ]$$

図 6 は有限写像についてのポリグラフである。

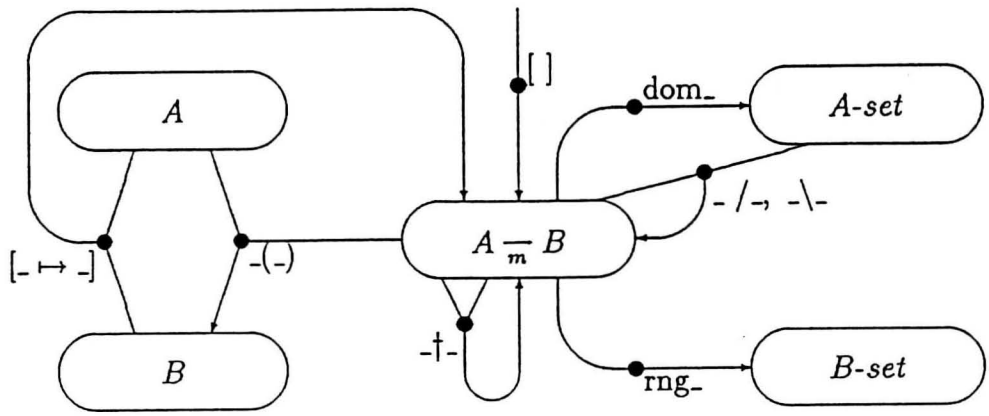

図 6

5 仕様言語について

プログラム言語同様仕様言語も数多く存在する。記述対象や記述側面に依存して特殊言語があり、広範囲の応用に役立ち仕様化から実現までのプログラム開発のすべての段階を支援する汎用言語がある。通信プロトコル記述を目的としたLOTOS、SDL、PSF などは前者であり、VDM、Z、RAISE、OBJ3、CIP-L、PA^{nn}dA-S、ACT ONE、ACT TWO、PLUSS、Extended ML、Larch などは後者である。基礎とする論理体系は何か、モデル指向か性質指向か、経時の挙動など動的側面を重視するか、静的側面で十分か、記述自体が何を意味するのか、引数付き仕様やモジュール仕様その他仕様構成演算にどんなものを持っているか、また実現へ向かっての詳細化をどう支援するかなどによって仕様言語はさまざまである。断片的にすぎないがこれらの諸点について補足しておきたい。本来は仕様言語原論として仕様記述のように厳密に展開しなければならない内容である。

5.1 論理体系

仕様に関する論述は厳密でなければならないから論理を自然形式化することになる。仕様言語にもその基礎とする論理の形式的体系が必要である。数学理論の展開には通常古典的な一階述語論理を用いるといわれているが、仕様記述のためにはさまざまな論理が選択できる。3 節で抽象データ型数字列の仕様を書いたが、それは等式論理を採用していた。そのほかに、高階論理、直観主義高階論理も考えられるし、時制・時相論理、ホーア論理などの様相論理やホーン節論理も用いられる。

実際、一階論理において、否定を許さないとき、ホーン節論理によって記述していることになり、さらに選言を許さないと代数仕様になり、またさらに等式の形を制限すれば関数型仕様が得られる。連言も許さないなら逐次型のプログラムを書かざるを得なくなる。直観主義論理のある体系のもとでは、仕様を定理として述べその証明を与えるとそこから自動的にプログラムが導出できるようにもなる。

5.2 記述枠組

仕様記述の枠組として次が一般に存在する。

1. 種集合と演算集合の組である標。これが対象を定義する定数、関数、述語などの基本語彙を構成する。
2. 標を基に構成した文である一群の公理。
3. 標によって定義される多種代数族。
4. 多種代数が公理を満足することを示す多種代数族と文との間の充足 (satisfaction) 関係。

1,2 によって書かれた仕様をどう解釈するかについて、3,4 によって定めるわけであるが、始代数意味論、終代数意味論、緩意味論などがあると 3 節で述べた。これらにはそれぞれ長短があり、緩意味論の立場でそれに必要があれば制約を加えて所期の解釈を定めようとする傾向もある。たとえば自明でない部分代数を持たないという極小性を課したり、代数 A がモデルなら、それを見掛け上差が見られない代数 B もモデルとするという観察同値性を加味したりする。

5.3 構造化機構

1 基本演算

二つの仕様を融合させたり、一つの仕様に種や演算または公理を追加して複雑な仕様を構成したり、一つの仕様から種や演算を隠したり、名前を付け換えて目的の仕様に調整する仕様に対する演算があると便利である。このような演算を持った先駆的な仕様言語 CLEAR を例にし説明しよう。

$$
\begin{array}{l}
\underline{\text{const}}\ \mathit{Rationals} = \\
\quad \underline{\text{let}}\ R = \underline{\text{enrich}}\ \mathit{Integers}\ \underline{\text{by}} \\
\quad \underline{\text{data}}\ \underline{\text{sorts}}\ r \\
\qquad \underline{\text{opns}}\ <\ \_,\_\ >: \mathit{int}, \mathit{int} \rightarrow r \\
\qquad \underline{\text{err-opns}}\ \mathit{ZERODEN} : r \\
\qquad \underline{\text{eqns}}\ \underline{\text{all}}\ m, n, k : \mathit{int}. \\
\qquad\quad <\ k \times m, k \times n\ >=<\ m, n\ >\ \ \mathit{if}\ \ k \neq 0\ \ \mathit{and}\ \ n \neq 0 \\
\qquad\quad <\ m, 0\ >= \mathit{ZERODEN}\ \ \underline{\text{enden}} \\
\quad \underline{\text{enriched}}\ \underline{\text{by}} \\
\qquad \underline{\text{opns}}\ [\ \_\ ] : \mathit{int} \rightarrow r \\
\qquad \underline{\text{eqns}}\ \underline{\text{all}}\ m : \mathit{int}.\ [\ m\ ] =<\ m, 1\ >\ \ \underline{\text{enden}}
\end{array}
$$

```
enriched by
    opns 0,1 : r
         +,−,×,÷ : r,r → r
    eqns all m,n,m',n' : int'.
         0 = [ 0 ]
         1 = [ 1 ]
         < m,n > + < m',n' > =< m × n' + m' × n, n × n' >
         < m,n > − < m',n' > =< m × n' − m' × n, n × n' >
         < m,n > × < m',n' > =< m × m', n × n' >
         < m,n > ÷ < m',n' > =< m × n', m' × n >   enden
in derive sorts rat
        opns +,−,×,÷ : rat,rat → rat
             itr : int → rat
    using Integers from R by
        rat is r
        itr is i   endde
```

図 7

図 7 は抽象データ型有理数 *Rationals* を CLEAR によって記述したものである。まず抽象データ型整数 *Integer*の仕様があり、これに種 r、その定数 0,1、r上の四則演算$+,-,\times,\div$、整数を有理数に埋め込む演算 i を加え、公理を置いて仕様を強化 (enrichment) し、それから、必要な演算だけを抽出調整 (derivation) したものである。以下に CLEAR の構文を説明する。

- const $S_n = S$は仕様本体 Sに名前 S_nを付ける宣言である。
- theory S endthは Sが仕様であることを表し、Sには種sorts ...、演算opns ...、公理eqns ... を書く。sortsの直前にdataを付ければ、Sのモデルが始代数と同型であることを意味する。そこでは、

  1. 余分な要素を持たない、すなわち、代数の要素は必ず項に対応する
  2. 混同しない、すなわち、代数の二要素による等式は、要素に対応する二項による等式が公理群に存在するとき、そのときに限り満足する

  という性質が成立する。
- $S = S' + S''$は仕様 Sが仕様 S'と S''との融合仕様 (combination) であることを意味する。これは S'と S''の種、演算、公理を合併したときに得られるモデルを表示するものである。

- enrich S by X enden、S enriched by X endenは仕様Sに種、演算、公理Xを合併し強化した仕様を意味する。

- derive Y using S' from S by Z endde
  ここで、
  Yはsorts...、opns...
  Zは $news$ is $olds$ ...、 $newo$ is $oldo$ ...
  の形をしている。これは仕様SからYで示す種と演算以外を隠した仕様を意味する。このときZで示すように仕様Sの種と演算の名前を付け換える。using S'は結果得られる仕様がその部分仕様としてS'を含むことを示す。

- let $S = X$ in Yは局所仕様を導入するための文で、Yにおいて仕様XをSと名付けて引用することを示すものである。

以上で図7は読めるだろう。

(2) 引数付き仕様

仕様を引数として仕様が書けると便利である。このような引数付き仕様 (parameterized specification) をどう考えるかはさまざまである。CLEAR の場合は仕様から仕様を作り出す手続きを考える。

```
meta D = theory sorts d endth

procedure Seq( X : D ) =

    X enriched by data
      sorts seq
      opns [] : seq
           [ _ ] : d → seq
           _ o _ : seq, seq → seq
      eqns all s,t,u : seq.
           [] o s = s
           s o [] = s
           s o ( t o u ) = ( s o t ) o u   enden

const Digit =

theory data sorts digit
      opns 0,1,2,3,4,5,6,7,8,9 : digit   endth

const D-seq = Seq( Digit [ d is digit ] )
```

図8

meta $D = X$は仮引数の仕様で、一般の**const**仕様とは区別する。仮引数の仕様は実引数の仕様に対して、それが満たすべき制約になる。手続き引用に際しては、仮引数仕様の制約を満たす**const**仕様を添え、仮実の種と演算の対応を s **is** s' ... の形で書く。

(3) モジュール仕様

プログラム・モジュール同様、仕様に対しても輸出入の界面を用意することが考えられる。これは引数付き仕様に輸出入部の仕様を追加したものであるが説明は省略する。

仕様言語について話すべき内容は多い。代数仕様に関連して項書換系や、実現に向けての段階的詳細化とそれに伴う証明事項や証明系、また仕様作成のための支援環境などの重要事項に全く触れていない。これは Z と CafeOBJ のチュートリアルの前口上 45 分の話題の要約にすぎない。

参考文献

[1] D.Anderws and D.Ince, Practical Formal Methods with VDM. McGraw Hill, 1991.

[2] F.L.Bauer and H.Wössner, Algorithmic Language and Program Development. Springer-Verlag, 1982.

[3] F.Belina et al., SDL with Applications from Protocol Specification. Prentice Hall, 1991.

[4] J.C.Bicarregui et al., Proof in VDM: A Practitioner's Guide. Springer-Verlag, 1994.

[5] M.Bidoit et al., Algebraic System Specification and Development A Survey and Annotated Bibliography. LNCS 501, Springer-Verlag, 1991.

[6] D.Bjørner and C.B.Jones(eds.), The Vienna Development Method. LNCS 61, Springer-Verlag, 1978.

[7] ibid., Formal Specification and Software Development. Prentice Hall, 1982.

[8] I.Claßen et al., Algebraic Specification Techniques and Tools for Software Development The ACT Approach. World Scientific, 1993.

[9] J.C.Cleaveland, An Introduction to Data Types. Addison-Wesley, 1986. 小林光夫訳, データ型序説. 共立出版, 1990.

[10] J.Dawes, The VDM-SL Reference Guide. Pitman, 1991.

[11] G.Dromey, Program Derivation The Development of Program from Specifications. Addison-Wesley, 1989.

[12] 榎本肇編, ソフトウェア工学ハンドブック. オーム社, 1986.

[13] N.Gehani and A.D.McGettrick(eds.), Software Specification Techniques. Addison-Wesley, 1986.

[14] C.George et al., The RAISE Specification Language. Prentice Hall, 1992.

[15] S.Hekmatpour and D.Ince, Software Prototyping, Formal Methods and VDM. Addison-Wesley, 1988.

[16] 稲垣康善他, 抽象データタイプの代数的仕様記述法の基礎 (1)-(4). 情報処理, 1984.

[17] C.B.Jones, Software Development: A Rigorous Approach. Prentice Hall, 1980.

[18] ibid., Systematic Software Development using VDM, 2nd ed. Prentice Hall, 1990.

[19] C.B.Jones and R.C.Shaw(eds.), Case Studies in Systematic Software Development. Prentice Hall, 1990.

[20] C.B.Jones et al., mural: A Formal Development Support System. Springer-Verlag, 1991.

[21] J.T.Latham et al., The Programming Process. An Introduction using VDM and Pascal. Addison-Wesley, 1990.

[22] 松本吉弘, ソフトウェア工学. 丸善, 1992.

[23] S.Mauw and G.T.Veltink, Algebraic Specification of Communication Protocols. Cambridge Univ. Press, 1993.

[24] C.Morgan, Programming from Specifications. Prentice Hall, 1990.

[25] C.L.N.Ruggles(ed.), Formal Methods in Standards. Springer-Verlag, 1990.

[26] P.Thomas et al., Abstract Data Types, their Specification, Representation, and Use. Oxford Univ. Press, 1988.

1994 年夏 ― 哲学的暴走･脱線顛末記

青木 淳･岸田 孝一･熊谷 章
(SEA 哲学分科会: あれ，そんなのあったっけ?)

この夏から秋にかけて，SEA のいくつかの集まりで，ソフトウェア開発の哲学的基礎 (!?) に関する議論が断続的に行われました．最後は，まるで冗談みたいに，たまたま公募があったお上の補助金プロジェクトに応募してみようじゃないかということになり，結果はみごと予想通りに落選しましたが，「どう見ても，採択された提案よりこっちのが勝ってるよね」とは，周りで見ていた野次馬たちの判定．その当否はとこかく，その間の経緯を，手元に残った資料でまとめてみました (岸田).

1. YDOC 例会 (1994.5.27) ［青木淳］

5 月の Ydoc で，私が手書きの OHP シートを使って話したことを，コンピュータの中に情報化するために入力してみました．もしも，この原稿が SEAMAIL のお役に立つようでしたら幸いです．

「モモ」「大乗起信論」「死体を探せ！」と「(オブジェクト指向) ソフトウェア開発」の関係が，いまとっても面白いです．暇な時にまとめてみますので，できあがりしだい，投稿しようと思っています．乞う御期待!

密教とオブジェクト指向
― ソフトウェアの匠をめざして ―

OHP No.1 (道)

「道はさまざまであるが，それらの道は不放逸をもって根本とする．諸行は滅の性質のものである．汝等は不放逸にして勤めよ」(相応部経: Samyutta-Nikaya)

「不放逸」とは，集中と持続に重点をおいた努力．

高い山に登るのに，登山口がたくさんあるけれど，頂上に向かうにしたがって道が近づくように，とにかく登らなければならない．ふもとでは差だけが目立ってしまう．

ソフトウェアを開発する方法は数多あるけれど，一つの方法に集中して，それを持続してみることが大切．

OHP No.2 (般若)

「私は人気のない林を彷徨って，ふと古い道を見つけました．その道を辿って行くと，昔の人が住んでいた古い城がありました」(相応部経: Samyutta-Nikaya)

お釈迦様自身は過去の仏陀たちが辿った道 (古い道) を発見したに過ぎない．このことが，法身 (大日如来) や久遠の仏 (古い城) を生み出し，密教へとつながっていく．

「パーラミターとは彼岸に到ること．修行の末に彼岸に到ると思ってはいけません．なぜなら，彼岸に修行があるからです．修行すること自体が到彼岸に他ならないのですから」(正法眼蔵: 道元)

道元禅師は不放逸の真髄をうまく述べている．「ガテーガテーパーラガテーパーラサンガテーボーディスヴァーハー (往ける者よ，往ける者よ，彼岸に往ける者よ，彼岸に全く往ける者よ，さとりよ，幸あれ)」の真言の得心が弘法大師と同レベルにある．

「無苦集滅道 (苦しみも，苦しみの原因も，苦しみを制してなくすことも，苦しみを制してなくす道もない)」(般若心経: Prajna-Paramita-Hrdaya-Sutra)

ただ不放逸に徹してソフトウェア開発を行うことがパーラミター．

OHP No.3 (犬とプログラマ)

イソップ物語:「骨をくわえた犬が，水に映った自分の姿を見て，骨をくわえた犬がもう一匹いると思い込み，その骨を奪い取ろうとして吠えたとたんに，自分がくわえていた骨を水の中に落してしまった」

ちょっと前までは構造化，いまはオブジェクト指向，そして，あしたはエージェント指向．Smalltalk がよい，いや Objective-C もなかなか，でも C++ がメジャーみたい．まさにイソップの犬．

私は，まだ 10 年強しか，オブジェクト指向 (Smalltalk) をやっていない．せめて 50 年ぐらいはプログラムを作り続けよう．お釈迦様が 29 歳で出家し，30 代中頃で正覚して (悟りを開いて) から，80 歳になるまで説法を行った (古い道を辿り続けた) ように．

OHP No.4 (ものを造るのは学者ではない)

1 年間に (Smalltalk で) 約 1 万ステップ (C++ にすれば約 10 万ステップ) 程度のプログラムを組む人でありたい．

理解 (りげ) と行解 (ぎょうげ) のバランスが大切．

ものを造るのは職人．達人 (匠) は職人から生まれる．学者のいうことは様式論にすぎない．自分の手でプログラムを実際に組んで様式論を述べている学者は稀有．

飛鳥様式だとか，白鳳様式だとかいったところで，法隆寺や東大寺が建てられないのと同じこと．職人がしてきた仕事を分類し整理し系統立てたのが，学者の好きな学問 (様式論) に他ならない．

様式を知って，その実装の技を持っている人が，理解 (りげ) と行解 (ぎょうげ) の中道を行く (バランスのとれた) 覚者 (匠) だと思う．

OHP No.5 (図面だけでは分からない)

端や隅がいちばん難しい．たとえば，プログラム制御構造の出入口，オブジェクトを囲う殻．

博士さんや修士さん，学士さんは，分析図や設計図を書くことができるかもしれないが，実際の施工ができないんだ．(ある大工さんの言)

私は，なぜか，ソフトウェアに関する論文などを読んでも，心が動かされたり，心が引きつけられたりすることがない．では，どんなものを読むと，心が動かされたり，心が引きつけられたりするのだろうか．それは建築の本や仏教の本などである．もの造り数千年の歴史を持つ建築，考え方のありとあらゆる宝庫の仏典，これらと比較すると，ソフトウェアに関する論文がたわいないものに見えてくるから不思議．私たちの脳は，人間が誕生してから此の方，変化がないのだから当然のことか．

OHP No.6 (手でした仕事がすべて)

とやかく効能書きを並べるのは職人のすべきことではない．手で伝える．手でやってみて教える．手でやってもらって学ぶ．

書物 (論文など) でもなく，口 (講義など) でもなく，手 (プログラム自身) で教える．やってみせて「直に」教える．結縁灌頂を授け，これはと思う者に伝法灌頂を授ける．一子相伝．奥義を伝える者が現われないことは，自分の不徳の致すところ．

「盲人に語るのでなければ言葉で伝えようとするな．言葉は自明で本質的なことを語るのみにせよ」(Leonardo da Vinci)

「筆授 (学問) だけでは伝わらないものがある」(空海)

OHP No.7 (手で直に教えると何が伝わるか)

プログラム自身ではなく，プログラムを作らしめたものが伝わる．

密教がお釈迦様だけでなく，お釈迦様をして悟らせたもの (法身：大日如来) を大切にするのに似ている．

顕教と密教．すなわち，筆授の好きな学者と相伝の好きな職人．

OHP No.8 (三蔵法師と鳩摩羅什)

共に多くの経典を漢訳した人．

＜訳し方＞： 三蔵訳は，文字の正確な訳をめざしている．スタティック (顕)．羅什訳は，要点をつかむ訳をめざしている．ダイナミック (密)．

＜生き方＞： 三蔵は，経律論 (三蔵) を身につけ，規律正しく生きて経典を訳す．羅什は，亀茲 (きじ) の国 (西域) の人で，女と寝ながら経典を訳す．

私は三蔵よりも羅什が好き．

OHP No.9 (Copyright)

私はプログラムに Copyright Notice など入れたくない．そもそも自分が作ったなんていえない．自分が著作権を所持していると宣言するなんて烏滸がましい．多くの師，多くの輩，多くのプログラム，多くの書物.....「私が作った」「自分が造った」などというのは，自らの「宿業」を知らない愚か者のいうことだと思う．

私自身は，GNU の GPL に則ったフリーソフトウェアを作成し配布しているが，フリーソフトウェアであっても，Copyright を宣言しなくてはならない．あの世の中 (彼岸) と違って，この世の中 (此岸) はままならぬ．

OHP No.10 (目がつんでいる方を暗い方へ)

これは，大工仕事が好きな父の言葉．木の木目 (年輪) を見て，目がつんでいる方は北側に向いていたのだから暗い方へ，目がつんでいない方は南側に向いていたのだから明るい方へ，この木の癖を間違えると，その木はしなってゆがんでしまう．

仕事とは，事 (こと) に仕 (つか) えること．事は，関係 (癖)．癖に仕えるのが職人．私はプログラムの癖に仕えるプログラマでありたい．

仕事は給料のためにやるのではない．法喜や法悦のためにプログラムを作る．自分がプログラムを作ったのではないことを知るために，ただひたすらに作る．つまり，理趣経の説くところの「全力を尽して何もしない」．

OHP No.11 (プログラムの匂い)

バタくさいプログラム，醤油のようなプログラム，関西のプログラム，関東のプログラム..... 青木くさいプログラムとは? ("お線香の匂い!" from 野次馬)．

あるスーパ･プログラマのプログラムは，頭の中から湧きだしてくるプログラムに，キーボードを打つ手が追いついていかないような，もどかしいプログラムの匂いがする．

プログラムは，選んで，寄せて，立てる．癖に仕えることが大切．

OHP No.12 (オブジェクト)

「一一 (いちいち) の字 (じ) は即 (すなわ) ち法 (ほう) なり．此 (こ) の一一 (いちいち) の名 (な) は皆 (みな) 世間 (せけん) の浅名 (せんみょう) を以 (もっ) て，法性 (ほっしょう) の深号 (じんごう) を表 (あらわ) す．即 (すなわ) ち是 (こ) れ喩 (ゆ) なり」(般若心経秘鍵: 空海)

オブジェクト (抽象データ型) を観る目．

記述されたプログラム (オブジェクト) は浅名．オブジェクトは＜データ＋手続き＞．

プログラム (オブジェクト) を書かせしめたは深号．オブジェクトとは＜自他を分かつこと＞．

腑に落ちるとは深号を得心した時．浅名だけでは臓腑に落ちない．

(オブジェクト指向の) ソフトウェア開発において，ほんとうに伝えなければならないのは，そのソフトウェアの背後に横たわっていて，そのソフトウェアを作らしめた「共同幻想(シソーラス：ラング)」．

OHP No.13 (もの造るを人は無常を得る)

「志に至らざることは，無常を思わざる故なり」(阿含経: Agama)

ソフトウェア技術者は，脳の中の「共同幻想」の鏡像である「仮想現実」をコンピュータの中に作りあげ，それを現実だと多くの人に思い込ませるための職人．おとぎの国の仕掛け人．

オブジェクト指向は「おとぎの国」を構築する有力な方法の一つ．

「共同幻想」も「仮想現実」も共に「無常」であることを得心すると「妙(ス)」に至るらしい．

50年ぐらいプログラムを作り続けてみよう．それが私の不放逸．

2. SEA 関西第1回哲学分科会 (1994.6.9)

2.1. ソフトウェア唯名論 [岸田孝一]

今回のテーマは，熊谷さんがソフトウェア「実在論」の立場でしゃべるということなので，対抗上，私の立場は(イスラム風?)「唯名論」ということにしました．

実在論(リアリズム) vs 唯名論(ノミナリズム)の区分は，ヨーロッパでは中世の神学論争に由来していて，普遍(この特殊ケースでは"神")の存在を信じるのが前者，信じないのが後者の立場です．東洋(中国)では，それよりはるか以前(紀元前数世紀)のいわゆる戦国の諸子百家の時代に，有名な「名実論争」が展開されました．

このとき，実在論の立場に立って，たとえば「道」という普遍的概念の重要性を説き，名と実との一致を求めたのが，孔子をはじめとする儒家の人々でした．「必ずや名を正さんか」という孔子の有名なことばは，そのまま今日のソフトウェア・プロジェクト管理のガイドラインとして使えると思います．

熊さんごヒイキの道家(老子＆荘子)は，儒家のアンチテーゼですが，「道」という普遍概念の存在を信じているかぎりは，やはり実在論派に属します．

諸子百家の中で，普遍的概念の存在を認めず，「名」はただの名でしかないという立場から議論を展開したのは，いわゆる名家や墨家の人々でした．たとえば，名家の代表的論客・公孫竜の有名なテーゼ「白馬は馬にあらず」は，世の中ではただの詭弁だと誤解されていますが，じつはきわめて現代的な認知意味論(Cognitive Semantics)と相通ずるようなニュアンスを含んでいます．

つまり，唯名論の立場からすれば，「白」という普遍概念は存在しない．「白馬」は，「白」という色彩を認識するという主体の意識を仮に「馬」というメディアを借りて表現しただけのもの，一方，単に「馬」といった場合は，形としての「馬」の認識である．色の認識と形の認識とは，主体の意識において明らかに異なるということです．

われわれが作るソフトウェアが，対象を記号化したモデルつまりは「名」である以上，それが対象すなわち「実」を正確に表現しているか否かが，つねに問題になります．その意味で，2千年以上前に隣の大陸で展開された哲学論争は，ソフトウェアに関わるわれわれの仕事や技術を考える上で，大きな意味を持っているように思われます．

諸子百家時代の哲学論争については，たくさんの参考文献がありますが，私のおすすめは：司馬遷「史記列伝・第七十章"太史公自序"」(岩波文庫)，白川静「孔子伝」，「中国の古代文学」(いずれも中公文庫)，加地伸行「儒教とは何か」「中国人の論理学」(いずれも中公新書)，大室幹雄「正名と狂言」(せりか書房)といったところです．

孔子の哲学の解釈は，朱子学や陽明学などさまざまですが，私見では，もっとも正統的なのは，わが荻生徂徠のそれだと思います．そのことについては次の本が詳しい：野口武彦，「荻生徂徠－江戸のドン・キホーテ」(中公新書)．

ところで，「哲学」分科会という今夜の集まりの趣旨に戻ると，哲学とは一体われわれにとって何なのでしょうか？

私の好きなスペインの思想家オルテガ・イ・ガセットのことばを引用すれば，それは「人間が生きる上での基本的なオリエンテーションの技術である」ということです．人生において，われわれは，その出発点から道に迷っています．たとえば，「私はなぜプログラマであるのか?」，「どのようにこのシステムをとらえればよいのか?」，「オブジェクト指向設計はどうすればうまく行くのか?」といった問いを，われわれは常に自らに向かって発するのですが，その答えは簡単には得られません．

人生におけるこうした基本的オリエンテーション問題に対するオルテガのアプローチは「私とは私とその環境である」というキャッチフレーズに集約されます．われわれ人間は孤独な存在ではなく，さまざまな人や事物から構成されたこの世界の中に生きているので，「私が私自身を見出す」のはいつもどこかの環境の中でのできごとです．つまり，「私にとって生きるとは私自身の外部に存在すること」(別のいい方をすれば，絶えざる自己疎外こそが生きることの本質)なのです．

しかし，私を取り囲むこの「環境」も，認識主体としての私と無関係に存在することはできません(素朴客観主義的実在論の否定)．そうした環境あるいは他者との関わりにおい

て，どのように私自身を発見し，成長させて行くかが，オルテガのすべての著書のライト･モチーフになっています．

オルテガは1930年代の初めに，今日の大衆化社会の状況を予言した名著「大衆の反逆」(白水社)を書いたことであまりにも有名ですが，われわれソフトウェア技術者にとっては，その遺作「個人と社会」(白水社)もまた，プロジェクト管理やグループウェアなどの問題を考える上で，きわめて示唆に富んでいます．「私とは私とその環境である」というその主テーゼの解説としては，次の講義録が一番わかりやすいと思うのですが，残念ながら日本語には訳されていません．

Jose Ortega y Gasset: "Some Lessons in Metaphysics", W.W.Norton & Co.

ところで，私(主体)とその環境(客体)との対比についていえば，すぐに思い出されるのは，禅の世界における巨人･臨済義玄禅師が提示した悟りの意識における相互に交換可能な4つのモード「人境俱奪･奪境不奪人･奪人不奪境･人境俱不奪」ですが，その解説は，臨済自身の手になる美しい詩的メタファにまかせることにして(たとえば，岩波文庫版「臨済録」参照)，ここでは，環境すなわち客体に対するわれわれ自身の意識について，少し考えてみましょう．

井筒俊彦さんが，名著「意識と本質－精神的東洋を求めて」(岩波文庫)で述べているように，およそ，意識はつねに「何か(X)への意識」です．つまり，意識はそれが成立する以前に，本能的な対象(X)の本質把握を必要としているのです．たとえば，ここに一輪の花があるとして，われわれがそれを花として意識するとき，われわれの心の中には，すでに花とは何か(花の本質)についての理解が形成されている．そうした本質把握は，じつは，われわれが使う言語の意味分節機能に大きく依存しています．

われわれが使っているこの言語というツールの持つ魔術的な働きとその危険性について鋭い警告を発したのは，今世紀前半に生きたアマチュア言語学者B.L.ウォーフでした．

「言語はそれを使う人間の非言語的行動にも影響を与えそれを制約する」というかれの問題提起は，物理学におけるアインシュタインの相対性理論とならんで，言語学的相対論仮説と呼ばれ，20世紀における最大の知的発見の1つといわれています(池上嘉彦･訳「言語･思考･現実」講談社学術文庫)．ダウンサイジングやオープン･システム化の嵐の中で，いまだにメインフレーム文化を捨て切れずにもがき苦しんでている大勢のCOBOL プログラマたちの姿は，まさにこの仮説の正しさを実証するひとつの典型的な例だといえるでしょう．

一方，わが東洋では，すでに千年以上も前から，言語の持つこの魔術的な意味分節化機能を虚妄であるとする考え方が広く行き渡っていました．たとえば，数ある仏教教典の中でもっとも形而上学的なニュアンスが強いといわれる「大乗起信論」には，次のように説かれています：

◇ 一切の言説は仮名にして，実なく，ただ妄念に随えるのみ....(すべての言語表現はただ仮に立てられた名前にすぎないのであって，別にそれぞれの名前に対応する「実」すなわち本質があるわけではない．ただ妄念の働きによって，いろいろなことばが浮かんでは消えているだけなのだ....)

ならば一体，ことばを主たるツールとして成り立っているわれわれソフトウェア技術者の立場はどうなるのか？ いかなる分析･設計技法を用いようと，われわれが描くシステム･モデルは所詮ただの妄念の所産でしかありえないのか？ といった質問を投げつけてみたい誘惑に駆られますが，しかし，「一切空」の立場を固持する仏教哲学のサイドからは，直接的な答えは返ってこないでしょう．井筒さんが好んで引用された青原惟信禅師の述懐：

◇ 未だ禅の道に入る以前は，山を見れば是れ山，水を見れば是れ水であった．その後修業を積み，いささか悟るところあって，山を見ても是れ山にあらず，水をても是れ水にあらずという段階に達した．そして，さらに修業を深め，安心の境地に落ち着いたいまでは，山を見るにただ是れ山，水を見るにただ是れ水である．

このメタファは，まさしく，オブジェクト指向方法論が持つウィークポイントを鋭く突いているように感じられます．

言語表現とは，それ自体は何の意味も持たない記号のつながりにすぎません．そうした記号列に，それでは，どうしたら意味を与えることができるか？ 常識的には，それぞれの記号と現実世界に存在するモノとの対応づけによって，ことばの意味というものが考えられているわけですが，しかし，恐ろしいことに，あるアメリカ人哲学者の手によって，そうした意味づけは無効であることが，最近(記号論理学的に)証明されてしまったのです．

たとえば，ひとつの命題を表す文(ステートメント)を考えたとして，それを構成する各要素と現実との対応づけを変えたとしても，命題の真偽が変化しないような反例をあげることが可能だというのが，その証明(パトナムの定理)のポイントです．いいかえれば，ひとつの文章の中の単語をとりかえても，文章全体の意味が変化しないという場合がいくらでも考えられるのです(詳しい説明は，たとえば，G.レイコフ「認知意味論」紀伊国屋書店の第15章「パトナムの定理」参照)．

これは，数学における非ユークリッド幾何学の発見に匹敵するくらいの事件です．言語による世界の記号モデル化を意図しているわれわれにとっては，特に，きわめてショッキングなできごとだといえましょう．

それでは，意味分節化のツールとしての言語を放棄して，素手で世界に立ち向かった時，われわれの目の前に出現するであろう(いわば言語以前の無分節的な)本質とは何でしょうか？ それは，すべての属性を剥ぎ取られたスーパー･メ

タ·クラスとしての純粋オブジェクトのようなものでしょうか？ もし，あなたが，そうした純粋本質，純粋存在，あるいは普遍者の存在を信じるなら，プログラマとしては実在論派に属するといってよいでしょう．

ところで，哲学の世界では，「本質」には，２つの種類があり，それらはお互いにはっきり区別されなければならないとされています．

第１種の本質は，「特殊的意味での本質 (イスラム哲学の用語では "マーヒーヤ" と呼ばれる」) で，これは，「それは何か？」という問いへの答えです．つまり，X を X たらしめているもの，X という概念に相当します．

これに対して，第２種の本質は「一般的意味での本質 (フウィーヤ)」であって，その意味するところは，「これであること」，つまり個々の具体的な X の即物的リアリティのことです．

これら２種類の「本質」に対してどのような態度をとるかによって，哲学の諸流派の立場は鮮明に分かれます．前述の書「意識と本質」で，井筒さんは，このモノサシを用いたシステマティクな分類を試みています．それによれば：

立場-A： まず，フウィーヤがある．われわれの意識がそれに向かうとき，それがマーヒーヤとして捉えられるだけのことにすぎない．多くの芸術家，たとえば，芭蕉やリルケのアプローチはこれにあたる．

立場-B： いや，マーヒーヤは実在する．それはフウィーヤを存立させる土台である．では，どこに存在するのか？

B-1：「われわれの深層意識に」というのが，宋儒 (朱子学) の格物窮理の考え方．

B-2：「深層意識に，ある種の原型イメージとして」のが，密教マンダラや，ユダヤまたはイスラム神秘主義，の考え方．

B-3：「表層意識にある」というのが，孔子の正名論．

もちろん，ここで A は唯名論的，B は実在論的な考え方にあたります．

かつて，構造論的プログラミングの旗印を掲げて現役を張っていたころの私自身のひそかな目標は，一切の意味を剥ぎ取った純粋プログラムの構造を追求することでしたが，それはどちらかといえば，上の立場-A に近かったのではないかと思います．

イスラム哲学の歴史の中でもっとも過激な問題提起を残したことで知られるアヴィセンナ (イブン·スウィナー) の次のような唯名論的なテーゼが (もちろん唯一神アッラーを信じていたかれが唯名論者であるはずはありませんが)，プログラマとしての私のアプローチをきわめてよく表しているように感じられます：

◇ 何かがあることと，何かが何かであることとは，別である．初めに存在があり，それが自己を限定し特殊化し個別化して，具体的なものとして立ちあらわれる．本質とは，その際の自己分節化の単なる指標であるにすぎない．ところで，ものがものでありながら，しかも現実には存在していない状態を指す「タビーア (本性) という概念が考えられる．それは，いわば純粋本質である．それを追求しよう．

..... というあたりで，どうやら「落ち」がついたようです．

2.2. ソフトウェア実在論 ［熊谷章］

<私のモットー>

－ Seeing is believing (百聞不如一見)

－ Knowledge is structured with image, without symbols.

<私の座右銘>

不易流行 (松尾芭蕉，老荘思想): 流行も不易も極まれば相通じるものだ．流行を分からないものには不易が分からない（その逆も真である)．

マイノリティ: マイナーなものには，人の心を動かす何かがある．

人間性中心主義: 何でも人間中心に考えねばならない．

<今日のテーマ>

設計，名前，現象，時間，科学．

A. 名前とフェティシズム (物神崇拝, 呪物崇拝)

物の価値は，そのもの (entity) になく，名前にある．

スニーカ (もの) － リーボック (名前)

かばん (もの) － ルイビトン (名前)

われわれはものを買うときにブランド名で選び，ブランドを買う．

B. 思考は言語に縛られている

虹は何色か？ 日本では７色，アメリカでは６色，インディアンは５色，その他の言語では２～３色．

ものを認識する基準は言語に依存している．言葉が違うと非言語的行動までもが異なる (言語相対論)．証明: 青·緑·青緑のカードを見せて，一番異なるもの選ばせる実験で，言葉に青と緑の区別がある民族とない民族とでは明らかにに行動が異なった。

C. 名前と現象と時間を生成する認識

人 (entity) の名前は一定だが，人相，風貌など (attribute) は，子どもが成人し，結婚したりするにつれ，時間とともに変化する．しかし，何十年か経過した後でも，小学校の同窓会で同級生に会うとすぐにそれが誰かわかるのはなぜか．

一般名詞についても同じことがいえる．たとえば，「熊」であるが，一度実物の熊を見たことのある人は，以後，異なった種類の熊を見ても，それが熊であることをただちに認識できる．一方，百科辞典などろ通じて言葉だけで「熊」の特徴を学習した人が，ほんものの熊を一見してそれが熊だと

認識できる確率はきわめて低い.

このことから, われわれの頭の中には, 名前と結びついた形で時間を生成するイメージがあると思われる. ある現象が生じたとき, それに対応するイメージが想起され, それに時間函数が働き, 具体的なイメージを生成する. そして, そのイメージと目前の現象を比較し認識するのではないだろうか. それは, 構造主義でいわれたような変形を原型にもたらして生み出されると思われる.

D. まとめ

従来の科学やものの見方は, 時間を止めた静的な状態を対象にしていた. そして, その対象を分析的に解体し, 定量的に把握することによって, その全体を表わすことができると考えられていた.

つまり, 次のようなキーワードで示される:

> 共時的 客観的, 定量的, 分析的, シンボル, リピータブル, 数学, 物理学.

一方, これに対抗する考え方は次のようなキーワードで表わせる: 通時的,]

> 主観的, 定性的, 直観的, Structured Image, Generic, 一度だけ, アート.

これは時間を生成し, 直観的で, 主観ををベースとして定性的にものごとを認識する方法である. 今後はこの方法にもとづいた考え方とものの見方が重要だと考えている. 上にあげた名前が時間を生成する例は, この方法の簡単な例である.

E. おまけ (老子について)

老子: 五千言 (固有名詞を一切使っていない). いわば, クラスで詩として語っている. それに対して, **荘子**は, 隠諭, つまりインスタンスで語っている.

道: 存在として道と当為としての道の２種類がある.

徳: 万物が所有する道を体得しそれを実践できること.

自然: 主体としての人と客体としての道を融合すること.

老子の考え方は, 無為自然という言葉ですべてが語られているが, それは何もしないということではなく, 道を知り, 徳を実践することである. したがって, この考えは, 次に示すキーワードに深く結びついている:

> 無為, 無私, 無欲, 無知, 自然, 対立と統一, 復帰, 政治思想.

これらの言葉が示しているものは, 最初にあげた不易流行と同じ考え方で, 二項対立を止揚し自然に生きる方法を示している.

私が老子を好きな理由はそれが詩であり美しいからだ. そして, 生成的だと感じている.

3. YDOC 哲学 & 鰻の会拾遺 (1994.8.26) [岸田孝一]

先日の YDOC 鰻の会で聞いたところ, プログラマのみなさんは現代アメリカ哲学にあまり興味を持っておられないようなので(それなのに, なぜ「ゲーデル･エッシャー･バッハ」の訳本が売れたのかな？), ちょっとコマーシャルをしておきたいと思います. とはいえ, そういう私自身も身を入れて読みはじめたのはここ１年ばかりのことなので, 正確な道案内をする自信はありませんが....

ことの始まりは, 昨年暮れに行われた SDA プロジェクトの最終ミーティングでした. ソフトウェア･デザインの基礎に関するスピーチをしたいと材料探しをしているうちに, 田中克彦さんの「言語学とはなにか」(岩波新書) で B.L. ウォーフの言語相対論仮説を知りました. さっそく講談社学術文庫で翻訳が出ている「言語･思考･現実」を読んでみたら, これがめっぽう面白い. 田中さんが 20 世紀最大の知的発見の１つと称賛しているこの仮説のその後の評価はどうなのだろうと, 巻末の参考文献リストを眺めていると, 意外にも, なつかしい本の名前が見つかりました.

それはジョーシ･レイコフの "Women, Fire, and Dangerous Things". UC バークレイ校を代表する認知意味論の大家が著したこの本は, 初版が出たとき, サンフランシスコの下町の書店で平積みになっていて, 題名に引かれて手にとってはみたものの, ちょっと部厚いからと敬遠した記憶があったのです. すぐ, 神保町の北沢書店へ出かけて入手しました (あとで邦訳が紀伊国屋から出ているのを知って, それも買ったのですが, 原書のほうが読みやすい. 言語学関係の本は, そうしたケースが多いのはなぜでしょうか!?).

さて, それからが蟻地獄.

レイコフ先生の手になるウォーフ理論の解説はそれなりによく書けていましたが, それ以上にショッキングだったのは, 形而上的実在論の基礎をみごとに打ち砕いたパトナムの定理の紹介でした.

「ほんらい無意味な記号のつながりである言語について, 文を構成するそれぞれの要素 (単語) を適当に現実世界のモノや関係と対応させることによって "意味" を定義しようとしても, 決して意味は一意には定まらない」ということを論理的に証明してみせたこの定理は, 何らかの言語を用いて現実世界の有意味なモデルを構築しようとしているわれわれソフトウェア技術者にとって, 致命的なインパクトを持っているように感じられました.

そういえば, 池袋西武リブロの哲学書コーナーで, ヒラリー･パトナムの名前は目にしたことはあったが, そんな人とは知らなかった.

ということで読みはじめた「実在論と理性」(勁草書房) ですが, これが, 面白いけれど難物. 第１章 "モデルと実在" の冒頭にいきなりレーヴェンハイム＝スコーレムのパラドックスなんてのが出てきたりして, おもわず机の脇の本棚に学

生時代(30年前)から棚さらしにしたままであった教科書「記号論理学入門」(岩波書店)を引っぱりだしてしまいました．そういえば，このテキストブックの著者W.v.O. クワインこそが，論理実証主義の誤りを指摘し，ヴィトゲンシュタインを祖とするウィーン学派の伝統を引きついだ現代アメリカ分析哲学の親分だと気がついたのは，しばらくたってからの話．

「心と世界の両方が協同して，心と世界とを作り上げる」というのが，いわゆる転向後のパトナム先生が掲げる "内的実在論" のキャッチフレーズで，前述のレイコフも認知意味論の立場から，それに同調している節があります．詳しくは，前述の本，またはようやく法政大学出版局から翻訳が出たパトナムの主著「理性·真理·歴史」をお読みください．ただし，唯名論に組する私には，ちょっと物足りないところがあります．そこで，他にだれかいないの？ と物色しはじめたところ，最初に目にとまったのがネルソン·グッドマンでした．

「この本は決して平坦な道を走らない．この本は狩りをする．そして，ときどき別の木に棲む同じアライグマを攻撃したり，同じ木に棲む別のアライグマを攻撃したりする．かと思えば，攻撃したものが木に棲むアライグマなどではまったくないことがわかることさえある..... そして，この本が数え上げるのは，獲物ではなく，狩りの道筋で踏査した領域から学んだことなのである」

スタンフォード大学での講義をきっかけとして書かれた「世界制作の方法」(みすず書房)は，上のような魅力的なまえがきではじまっています．私が気にいっているのは，グッドマンが，科学と芸術とを区別せずに，両方の分野を同時にカバーするような思想を構築しようとしていることです．それは，ユニークな技術論と芸術論，そして大衆社会論を並行して展開した故·オルテガイガセットに一脈相通じるところがあります．

「世界には複数のバージョンがある」というのがグッドマンの主張ですが，そこには，やや逆説的なニュアンスが含まれています．ユニークな唯名論者であるかれは，もちろん，それらの相異なる複数のバージョン(記号化された世界の記述)の背後に，記号化されていない1つの「正しい」実在世界が存在するなどとは，認めません．それぞれの記号体系は世界そのもののバージョンであり，複数のバージョンがあるということは，世界自体が複数存在することを意味します! 「セカイのツクリカタ」という書名はここに由来しているのです．

ソフトウェア開発に関するトラブルの最大の理由として，エンドユーザ，アナリスト，デザイナ，プログラマ等，開発関係者のあいだでのコミュニケーションの不足があげられます．つまり，対象システムの記述に関して，複数のそれぞれ不完全なバージョンが勝手に作られ，それがさまざまな誤解の原因になっているというわけです．そうした議論の根底には，それらのバージョンの背後に正しい「真の」システムの実体が存在するという素朴実在論的な信仰があります．各種の分析·設計技法の優劣も，一般に，それらの技法が，対象システムの真の姿をどれだけ正確なバージョンとしてとらえられるかという観点から，あれこれ論議されているように見えます．

グッドマン流のバージョンの解釈はそれとは違うのです．それぞれのシステム記述は，すべてが正しいバージョンです．つまり，そこには，バージョンの数に対応して複数の世界(群)が存在する．それらの異次元の世界(群)が衝突しあい，やがては1つに融合して行くプロセスこそがシステム開発だと考えられるのです．ある意味で，それはコペルニクス的な発想の転換をもたらすものだといってよいでしょう．人間の意識から独立した実在としての対象システムなどは存在しない．そうした非実在論の考え方からすれば，「神」の視点に立った絶対的なシステム·イメージはありえないのです．量子物理学における観測者の理論のアナロジーを，システム開発の場に応用することを想定してみてもよいかも知れません．

グッドマンの著書は，もう1冊翻訳されています．「事実·虚構·予言」(勁草書房，原題はFact, Fiction,and Forcastと頭韻を踏んでいる)がそれです．反事実的条件文，素質，帰納法など，非現実的なものを経験主義の立場からどう扱うべきかを丁寧に分析したこの本も，論理好きのプログラマ諸氏にとっては，なかなかに魅力的な語りくちを持った名著だといえましょう．

さて，人間がことばを使う動物であり，世界についてのわれわれの認識や記述が，ことばという道具に依存している以上，哲学にとって，ことばをどうとらえるかは，きわめて重要なことがらです．前述のパトナムの定理もそのことに関連していますし，さらに源流にさかのぼれば，ヴィトゲンシュタインの有名な「言語ゲーム論」があります．

そういった観点で，哲学における言語の問題を真正面からとりあげている人物といえば，パトナムやグッドマンと同世代のドナルド·デイヴィドソンを無視するわけには行かないでしょう．「ことばがそれの意味するものを意味するとはどういうことなのか?」をテーマとするかれの論文集「真理と解釈」(勁草書房)は，軟弱な意味論(セマンティクス)などではなく，もっとハードボイルドな意味の理論(Theory of Meaning)の構築を目指した手強い書物です．私自身まだ最初の1章しか読んでいないので，それ以上の紹介は差し控えておきましょう．腰巻きにうたわれた宣伝文句は，「論争の旋風を巻き起こし，反批判の砲撃を続けながら前進する意味理論の重戦車」と，まるで三流週刊誌の中吊り広告みたいですが，どこかで池田晶子女史がからかっていたように，哲学書のコマーシャルは，「どうせ売れないのだから...」とヤケ気味になったハチャメチャ調が多いのが特徴です．

ところで，クワイン大先生をはじめとして，以上に紹介した人々はいずれも今世紀初頭生まれの老大家ばかりで，いかにその論旨が興味深くても，なんとなく横町の御隠居さんの御高説を拝聴しているようで，こちらの気が休まりません．もう少し若手(自分と同世代)の人はいないのかしら，と探してみたら，いました，恐ろしいのが!

リチャード･ローティ．1931年ニューヨーク生まれ，シカゴ大卒業後，イェール大でPhD取得．1979年に出版されたその処女作「哲学と自然の鏡」は，アメリカだけでなく国際的な反響を巻き起こしたといわれています．日本語版(産業図書，1993年)のカバーに書かれた内容紹介によれば：

> ー 17世紀このかた，心や知識や哲学をめぐる議論は，表象という概念によって支配されてきた．心は，実在を映し出す1枚の鏡になぞらえられる．知識は，これらの映像の正確さに関わっている．そして，こうした知識を獲得するための戦略(鏡を検査し，修繕し，磨き上げる戦略)こそが，哲学に属しているのである．このような一連のイメージ系を探査し，広範囲にわたる批判を展開するにあたって，ローティは，それが哲学(とりわけ近現代の分析的思考)に与えた影響を概観し，かつそれを脱構築することを提唱している．
>
> デューイ，ヴィトゲンシュタイン，およびハイデガーによって提起された伝統批判に依拠することを通じて，ローティは，かれらの「啓発的」哲学を，ロック，デカルト，カント，ラッセル，フッサールらの「体系的」哲学と対比する．そして，後者を拒絶することによって，かれは，思考ないし言語と世界とのあいだの対応を発見するという考えが放棄されなければならないこと，それとともに，表象の理論を中心とした哲学という概念もまた放棄されなければならないことを主張するのである．

要するに「これまでの西欧哲学はオシマイよ!」と宣告しているわけで，そんな本が論争の的にならないわけはありません．20世紀の思想史も，終わりに近づいてきて，ようやく面白くなってきたという感じがします．なお，ローティの著書としては，その後に書かれた論文を集めたものも，「哲学の脱構築ープラグティズムの帰結」(お茶の水書房)として翻訳されています．もしかしたら，こちらのほうが読みやすいかも知れません．

以上，駆け足の現代アメリカ哲学読書案内でした．

4. IPAへの落選提案 [岸田孝一]

この夏，IPA技術センターが「独創的情報技術育成育成事業に係わる研究テーマ公募」を行ったことは，みなさん御承知の通りです．どうせ，お役所の考える「ドクソー性」とわれわれのイメージ(臼井さん@JIPのいう「毒草性」)とはまったく異なるだろうと予想しながら，青木-岸田組は次のような提案を試みました．審査員のみなさんにとっては，さぞいい迷惑だったことでしょうね:-)．

東洋哲学にもとづく
新しいソフトウェア開発パラダイムの研究と
その支援環境の試作

コンピュータが近代西洋科学の成果の1つとして誕生した関係上，これまで，ソフトウェア開発の方法論についても，「構造化」や「形式仕様」，あるいは「オブジェクト指向」など，科学的客観主義の哲学にもとづく技法がいくつも提案され試行されてきた．しかし，1960年代末に意識されたいわゆる「ソフトウェア危機」は，いまだに本質的な解決を見ず，「銀の弾丸」を求めての空しい探索が続いている．

一方，認知意味論などの研究が進んだ結果，たとえば「言語相対論仮説」の実験的検証や「パトナムの定理」の理論的証明などによって，科学的客観主義の限界が明らかになってきた．

この研究は，そうした状況を踏まえて，東洋哲学の成果をアナロジカルに反映した新しいソフトウェア開発パラダイムの可能性を探り，その具体的支援環境のイメージをプロトタイピングすることを目的とする．

本研究のテーマは，SEAがこれまでに主催したいくつかの国際会議における断続的な討論の中から浮び上がってきたものであり，今回の提案作成にあたっても，mailing listの上で，その内容に関する若干の検討が行われた．

したがって，研究の推進母体として，SEA内部に，提案者を中心とするワーキング･グループを作り，国際および国内のワークショップの企画･運営にあたることとする．

これらのワークショップの目的は，関連諸分野の専門家を交えて，まず，広くさまざまな可能性を探ること，そして，とりあげられた具体的テーマの中から，プロトタイプの試作およびそれを用いた試行実験のプランを立て，実行することである．

ソフトウェア開発に東洋哲学の成果を応用することのの可能性は，すでに科学的客観主義のアプローチの限界が明らかになっている開発上流工程(システム分析/定義/概念設計)において，特に大きい．

一口に東洋哲学といっても，その幅も広く深さもさまざまであるが，故･井筒俊彦氏の名著「意識と本質ー精神的東洋を求めて」は，共時的構造化の手法により，中東から極東にいたる広大なアジア文化圏に古代以降展開されてきた哲学的思惟を，言語の意味文節機能と人間意識の階層構造との関連において体系化したという点で，本研究開発にとって適切なガイドマップとして利用できるであろう．

本研究においては，具体的な試作開発のテーマを検討するワークショップ(国際および国内)と，プロトタイピング活動とを並行して行なうことを考えている．国際ワークショップ

の狙いは，(1)広く各専門分野の識者から研究のアイデアを提供して貰い，また成果についてレビューを受けること，および(2)本研究の国際的なVisibilityを高めることにある．国内ワークショップでは，プロトタイピング計画の詳細な検討，およびその成果の評価，試用実験等を主たる目的とする．

試作するプロトタイプの詳細なイメージは，実際にプロジェクトが始まり，最初のワークショップでの討論を終えなければ確定できないが，これまで，提案作成に際してe-mail上で話題に上がったいくつかの候補をアトランダムに例示すれば，以下の通りである：

◆ 禅の方法論を応用した要求分析支援システム

「言無展事(ことばではものごとの本質は表現できない)」という禅の基本的なパラダイムにしたがって，システム開発者とエンド･ユーザとの言語/文化の違いにもとづくコミュニケイションの難しさを，相互の意識のズレを拡大して見せることにより，逆に意志疎通を改善させる試み．故･井筒俊彦氏風の表現を用いれば，いわば，非対話(Beyond-Dialogue)型のグループウェア．

◆ ドメイン指向型アラヤ識レポジトリ

表層意識のレベルで構築される通常のオブジェクト･ライブラリではなく，唯識哲学でいうアラヤ識レベルに存在する原型イメージのレポジトリをいくつかのアプリケーション･ドメインを対象に構築し，その有用性を探る．レポジトリ･メカニズムとしてはPCTEやORBなどの標準機構を用い，その前面に分散型Smalltalkを置いた形で実現される．

◆ 九会曼荼羅による情報伝達システム

大(図像)曼荼羅，法(一字)曼荼羅，三昧耶(象徴：所持物や印相)曼荼羅，羯磨(行為や作法)曼荼羅などを駆使して，八近傍の概念を用いて情報相互間の遠近(距離)や抽具(取捨)などを表現する．

◆ 比喩(修辞)によるシステム分析/設計

全体と部分を表現する換喩(Metonymy)，抽象と具象を表現する提喩(Synecdoche)，相似や近似を表現する隠喩(Metaphor)などを縦横に駆使できるような新しいモデリング支援環境により，システムの仕様定義やシステムの拡張や再利用などの示唆を，より容易に行なうことを目指す．

◆ 二十四詩品風メタ･プロセス･モデルの試作

晩唐期の詩人･司空図の代表的作品である「二十四詩品」は，詩のモードを24種類に分類し，それぞれのモードを24篇の詩として表現したメタ･ポエムとして，そのユニークさが高く評価されている．このコンセプトを応用して，典型的なソフトウェア･プロセスをいくつかにカテゴライズし，それぞれの特徴をメタ･プロセス･モデルとして記述することを試みる．

◆ 分散型情報システムのための概念的社会モデル

1988～91年に，文部省科研費重点領域研究として実施された「比較の手法によるイスラームの都市性の総合的研究」は，農耕文明やその発展型としての工業文明におけるものとは異なるネットワーク型の商業文明社会における都市計画パラダイムの特徴を明らかにした(事典･イスラームの都市性，亜紀書房)．その成果をコンピュータ情報システムの分野に応用して，来るべき情報化社会におけるシステム･アーキテクチャの基本的な概念モデルを構築する．

先日fj.comp.misc他で公開された次の11篇の採用テーマ(いずれも大変ドクソー的でリッパに見えますね!)と比べると，どうやら，われわれの提案は何ビットか桁がずれていたみたいです(反省!)．

－ 高効率な分散協調アプリケーションの構築を支援する単一仮想記憶空間上の分散永続オブジェクト管理機構の実現

－ 並列オブジェクト指向言語の汎用高並列計算機向け処理系の開発とその応用実証プログラムによる評価

－ M3K：超分散超並列環境のためのオペレーティングシステムカーネル

－ エージェント指向情報共有ベース

－ On Line Universityのための知識共有環境システムの開発

－ 3次元データベースシステムの研究

－ カード操作ツールを用いた要求仕様作成支援の研究

－ 移動計算機環境におけるアプリケーションの構築法に関する研究

－ プロセッサの設計記述からのコンパイラの自動生成に関する研究

－構成的プログラミング･システムの開発に関する研究

－汎用超並列オペレーティングシステムカーネルSSS-COREの研究

しかし，日本を代表する有名大学のコンピュータ･サイエンスやソフトウェア･エンジニアリングの先生方が知恵を絞ってこの程度のアイデアしか出てこなかったというのは，ちょっと淋しいような気がしませんか？

論 文 募 集

ソフトウェア･シンポジウム'95

1995 年 6 月 14 日 (水) ～ 16 日 (金)　於: ラフォーレ琵琶湖 (滋賀県守山市)

主 催: ソフトウェア技術者協会 (SEA)

協 賛 (予定) 日本ソフトウェア科学会　情報処理学会　情報サービス産業協会

ソフトウェア･シンポジウムは、ソフトウェアハウス、コンピュータおよび組込み機器メーカ、エンドユーザ、大学、研究所などさまざまな場で活躍している技術者、管理者および研究者が一堂に会し、ソフトウェア技術に関する多面的な経験や知識を交流するユニークで貴重な場を提供してきました。一つの区切りとも言える第１５回を迎える'９５年のシンポジウムは、北上を続けてきた開催地を一転し日本の真ん中、琵琶湖のほとりで開催します。さらに今回はツール展示を併設し、技術交流をより充実したものとします。

毎年、現場に立脚した質の高い論文が集まるこのシンポジウムのレベルは、回を重ねる毎に着実に上がっております。今回も第１４回と同様に単一トラックでセッションを構成しますが、そのため採録可能な論文は１５編程度と少なくなる見通しで、高レベルの論文採用基準になると予想されます。皆様の積極的な御応募を期待いたします。

なお、従来通り最も優秀な論文に送られる最優秀論文賞は、シンポジウム開催時に選定します。論文は日本語の他、英語による投稿も可としますが、発表は日本語に限ります。応募論文テーマとしては、たとえば

- 分析／設計技法　・ CASE　・ 開発環境　・ ネットワーク　・ CSCW/グループウェア
- プロジェクト管理　・ 品質管理　・ プロセス改善　・ メトリックス　・ ソフトウェア産業の新パラダイム
- 標準化対応　・ 人間的要因　・ 技術者教育　・ マルティメディア　・ ドメインエンジニアリング

などさまざまなものが考えられますが、必ずしもこれらにとらわれる必要はなく、ソフトウェアの領域で他の人に参考となるテーマであれば何でも構いません。先進的な研究テーマの発表はもちろん歓迎しますが、開発現場での苦労･努力をまとめた経験報告も大歓迎です。奮って御応募下さい。

応募要領

応募論文は未発表のものに限ります。また、他への二重投稿はご遠慮下さい。今回もプログラム委員会での審査は本論文にて行ないますが、応募予定の方は論文概要を記入したカバーシートにより **1994 年 12 月 20 日**までにエントリをしてください。カバーシートは必要事項を記入の上、極力電子メールで、電子メールが使えない場合は FAX または郵送で 2 人のプログラム委員長のいずれか宛にお送り下さい。本論文の様式は自由ですが、A4 サイズで 5 ～ 10 ページ程度を目安とし、7 部を郵送で **1995 年 1 月 31 日**までにプログラム委員長宛にお送り下さい。プログラム委員会で内容の審査を行い、結果を 3 月下旬までに応募者全員に通知します。その他不明な点がありましたらプログラム委員長までお問い合わせ下さい。

主要スケジュール:	1994 年 12 月 20 日 (火)	論文概要によるエントリ締切り
	1995 年 1 月 31 日 (火)	本論文、及びツール展示応募締切り
	1995 年 3 月下旬	採否通知送付、及びツール展示依頼送付
	1995 年 4 月下旬	カメラレディ原稿締切り

シンポジウムスタッフ

実行委員長	中野 秀男 (大阪大学)		
プログラム委員長	佐伯 元司 (東京工業大学)	坂本 啓司 (オムロン)	
プログラム委員	青山 幹雄 (富士通)	鯵坂 恒夫 (京都大学)	荒木 啓二郎 (奈良先端大)
伊藤 昌夫 (MASC)	江谷 典子 (FXIS)	大場 充 (広島市立大学)	尾本 林貞 (ダイキン)
岸田 孝一 (SRA)	楠本 真二 (大阪大学)	熊谷 章 (PFU)	小泉 毅 (キヤノン)
小林 允 (日本ユニシス)	酒匂 寛 (SRA)	砂塚 利彦 (NEC)	高橋 光裕 (電力中研)
田中 敏幸 (シャープ)	谷津 行穂 (日本 IBM)	玉井 哲雄 (東京大学)	中来田 秀樹 (Next Foundation)
中谷 多哉子 (FXIS)	布川 博士 (宮城教育大学)	野中 哲 (ATJ)	平山 伸一 (さくら KCS)
二木 厚吉 (北陸先端大)	古川 善吾 (九州大学)	古山 恒夫 (日本電信電話)	増子 泰弘 (松下通信)
増井 和也 (東芝 AS)	松原 友夫 (Peopleware)	松本 健一 (奈良先端大)	
ツール展示委員	佐藤 正道 (オムロン)		
ローカル･アレンジメント	田中 敏文 (オムロン)		

ソフトウェア·シンポジウム'95

論文応募カバー·シート

氏名(筆頭著者)：＿＿＿＿＿＿＿＿ (ふりがな) ＿＿＿＿＿＿＿＿

所属(会社·大学·組織)：＿＿＿＿＿＿＿＿

部門(学部)：＿＿＿＿＿＿＿＿ 役　職：＿＿＿＿＿＿＿＿

住　所：(〒　　)＿＿＿＿＿＿＿＿

TEL：　　　(内線　　)　FAX:

E-mail：＿＿＿＿＿＿＿＿

論文タイトル：＿＿＿＿＿＿＿＿

(ツール名)：＿＿＿＿＿＿＿＿

共同執筆者：(もしいれば。ツールの場合不要)

氏名:＿＿＿＿＿＿＿＿ 所属:＿＿＿＿＿＿＿＿

氏名:＿＿＿＿＿＿＿＿ 所属:＿＿＿＿＿＿＿＿

氏名:＿＿＿＿＿＿＿＿ 所属:＿＿＿＿＿＿＿＿

論文概要(ツール概要):

応募カテゴリ：(いずれかにチェック)　□：経験報告　□：技術論文　□：ツール展示

キーワード：(最大5つのキーワード挙げてください。)

1：＿＿＿＿＿＿＿＿ 2：＿＿＿＿＿＿＿＿

3：＿＿＿＿＿＿＿＿ 4：＿＿＿＿＿＿＿＿

5：＿＿＿＿＿＿＿＿

------------------- 論文応募送付先 & 応募締切り：1994年12月20日(火) -------------------

〒152 東京都目黒区大岡山2-12-1
東京工業大学工学部情報理工学科 計算工学専攻
佐伯 元司
E-mail: saeki@cs.titech.ac.jp
FAX: 03 - 3729 - 1399 TEL: 03 - 3726 - 1111 ext. 2192

または

〒525 滋賀県草津市西草津2丁目2-1
オムロン(株)EFTS統轄事業部開発センタ第2開発室
坂本 啓司
E-mail: sakamoto@eftses.krc.omron.co.jp
FAX: 0775 - 65 - 5579 TEL: 0775 - 65 - 5048

------------------- ツール展示応募送付先 & 応募締切り：1995年1月31日(火) -------------------

〒617 京都府長岡京市下海印寺
オムロン(株)技術本部開発支援センタSPI推進課
佐藤 正道 E-mail: sato@lsa.ncl.omron.co.jp FAX: 075 - 953 - 7604 TEL: 075 - 951 - 5111 ext. 3407

第1回アジア太平洋ソフトウェア工学国際会議 (APSEC'94) 開催について
The First Asia-Pacific Software Engineering Conference (APSEC'94)

日程 1994年12月7日(水)～9日(金)

場所 早稲田大学国際会議場 (〒169-50 東京都新宿区西早稲田1-6-1)

主催 情報処理学会ソフトウェア工学研究会

協賛 The Technical Board for Computer Systems and Software Engineering of the Australian Computer Society, The Chinese Institute of Electrical Engineers, The Special Interest Group on Software Engineering of the Korea Information Science Society (KISS/SIGSE), IEEE Computer Chapter Singapore Section, IEEE Tokyo Section Computer Chapter, 電子情報通信学会ソフトウェアサイエンス研究会, 電子情報通信学会知能ソフトウェア工学研究会, 日本ソフトウェア科学会ソフトウェアプロセス研究会, ソフトウェア技術者協会

主旨 ソフトウェア工学に関する国際会議は、従来から種々開催されてきておりますが、そのほとんどが欧米を中心とするものでした。本国際会議 APSEC'94 は主としてアジア太平洋地域の国々を中心としたソフトウェア工学に関する国際会議として開催するものです。プログラムを次ページに示します。

参加申し込み 参加希望者は下記の registration form の要領でお申し込み下さい。

First Asia-Pacific Software Engineering Conference (APSEC'94) Registration Form
December 7-9, 1994, Tokyo, Japan

Please mail or fax to: Akira Kagaya (Registration Chair, APSEC '94)
Systems and Software Engineering Laboratory
Toshiba Corporation
70 Yanagi-cho, Saiwai-ku, Kawasaki 210, Japan
Phone: +81-44-548-5633 E-Mail: kagaya@ssel.toshiba.co.jp
Fax: +81-44-522-5198 (Until Oct.23), +81-44-520-5855 (After Oct.23)

Remittance should be a bank transfer or a bank check. Account Name: APSEC 94
Account No. 588-6383376
Account Type: Deposit Account
Sakura Bank, Yokohama Branch

Please Type or Print:
Last/Family Name:_____ First Name:_____
Affiliation (for name tag):_____
Address:_____
City/State/Zip/Country:_____
Phone: _____ Fax:_____
E-mail: _____
IPSJ or Cooperating Society Membership Number:_____

Please mark the box and fill in the _____ if necessary.

Registration Fees	Advance(by Nov. 10)	Late(after Nov. 10)
Member of SIGSE	☐¥20,000	☐¥25,000
Member of IPSJ or Cooperating Societies	☐¥25,000	☐¥30,000
Non-Member	☐¥30,000	☐¥35,000
Student	☐¥5,000	☐¥6,000

Remittance: ☐ Bank Transfer ☐ Bank Check

Do you need a bill and/or others for remittance from you ? ☐ I need a bill. ☐ I need_____

Registration fees include conference attendance, refreshment breaks, conference receptions and one copy of the conference proceedings. Request for cancellation must be received through mail or fax by the registration chair no later than November 21. A handling charge will be deduced from the registration fee.

Advance Program
First Asia-Pacific Software Engineering Conference (APSEC'94)
December 7-9, 1994, Tokyo, Japan

Wednesday December 7, 1994

9:30 - 10:00 **Welcome and Opening Remarks**

10:00 - 11:30 **Keynote Address**

Analysis in Software Engineering

Koji Torii (Nara Institute of Science & Technology and Osaka University)

11:30 - 13:00 **Lunch (Not Provided)**

13:00 - 17:00 **Special Session**

Software Engineering in Asia and Pacific — Looking Back and Looking Forward

Six or seven 30 minutes' presentations from each country of Asia and Pacific.
Each speaker talks retrospective and perspective of software engineering in his/her country.
Speakers : Karl Reed (Autralia), Danny Poo (Singapore), Yue-Sun Kuo (Taiwan), TBD

Thursday December 8, 1994

9:30 - 10:40 **Invited Talk**

Requirements Engineering: A Review And Research Agenda

Anthony Finkelstein (City University, UK)

10:40 - 11:00 **Break**

11:00 - 12:00 **Parallel Session 1A : Applied Methods**

Applying Object-Oriented Construction to Fault Tolerant Systems

James Miller, Murray Wood, Andrew Brooks, Marc Roper
(University of Strathclyde, Scotland)

An Adaptive User Navigation Mechanism and its Evaluation

Jeongwon Baeg, Atsushi Hirahara, Yoshiaki Fukazawa (Waseda University, Japan)

Parallel Session 1B : Development Environment

Developing Distributed Applications by Semantics-based Automatic Replication

Sumin Huang (Freie Universitat Berlin, Germany)

A TAHG Model Based Software Generator System

Mikifumi Shikida, Yasuhide Yamamoto, Yoshimasa Kimura, Takehiro Tokuda
(Tokyo Institute of Technology, Japan)

Parallel Session 1C : Re-Engineering & Maintenance

Proprietary vs "Open Systems" Options in the Construction of Knowledge-Based Software Re-Engineering Environments

Paul Bailes, Steven Atkinson, Murray Chapman, Dan Johnston, Ian Peake
(University of Queensland, Australia)

A Software Maintenance Survey

Stephen Yip (Hong Kong Polytechnic),
Thomas Lam (Provisional Airport Authority, Hong Kong)
Stephen Chan (Hong Kong Polytechnic)

12:00 - 13:30 **Lunch (Not Provided)**

13:30 - 15:00 **Parallel Session 2A : Software Process I**

Guiding the Requirements Engineering Process

Colette Rolland (Universite de Paris Sorbonne, France)

Naveen Prakash (Delhi Institute of Technology, India)

Constraint-Centered Descriptions for Automated Tool Invocation
Kazuto Tominaga, Takehiro Tokuda (Tokyo Institute of Technology, Japan)

Software Documents, Their Relationships and Properties
Jun Han (Monash University, Australia)

Parallel Session 2B : Domain Modelling

Supporting User-Analyst Interaction in Functional Requirements Elicitation
Alessandro Cucchiarelli, Maurizio Panti, Salvatore Valenti (University of Ancona, Italy)

A Model-Based MICOM Application Software Development Method
Kyo-Chul Kang, Sey-Chan Jang (Pohang Institute of Science & Technology, Korea)

Object and Domain Policies Specification
Danny C.C. Poo, Shwu-Yi Lee (National University of Singapore Singapore)

Parallel Session 2C : Testing and Debugging I

A Knowledge-Based Approach to Regression Testing
Taewoong Jeon (Goldstar Industrial Systems, Korea),
Anneliese von Mayrhauser (Colorado State University, USA)

Integrating Data Flow and Domain Testing
Bingchiang Jeng (National Sun Yat-Sen University, Taiwan)

Automated Class Testing; Methods and Experience
Daniel Hoffman (University of Victoria, Canada),
Jonathan Smillie, Paul Strooper (University of Queensland, Australia)

15:00 - 15:30 **Break**

15:30 - 17:00 **Parallel Session 3A : Software Process II**

From the Software Process to Software Quality: BOOTSTRAP and ISO 9000
Hans-Jurgen Kugler (European Software Institute, Spain),
Richard Messnarz (Technical University Graz, Ireland)

BOOTSTRAP: A Software Process Assessment and Improvement Methodology
Jouni Simila, Pasi Kuvaja, Lech Krzanik (University of Oulu, Finland)

Software Process Representation to Support Multiple Views of the Enacted Process
David Jacobs, Chris Marlin (Flinders University, Australia)

Parallel Session 3B : Applying Specifications

Dynamic Evolution of Distributed Systems Specifications using Reflective Language
Issam A. Hamid (Tohoku University of Art & Design, Japan)

A Mapping System from Object-Z to C++
Masakazu Fukagawa, Teruo Hikita, Hiroshi Yamazaki (Meiji University, Japan)

Reversing Concurrent Systems Into Formal Specifications
Karl R.P.H. Leung, Clement F.S. Yim (Hong Kong Polytechnic)

Parallel Session 3C : Testing and Debugging II

Ordered Sequence Testing Criteria for Concurrent Programs and the Supporting Tool
Eisuke Itoh, Yutaka Kawaguchi, Zengo Furukawa, Kazuo Ushijima
(Kyushu University, Japan)

Reachability Testing: An Approach to Testing Concurrent Software
Gwán-Hwan Hwang (National Tsing-Hua University, Taiwan),
Kuo-Chung Tai (North Carolina State University, USA),
Tin-Lu Huang (National Chiao-Tung University, Taiwan)

Fall-in C : A Software Tool for Pitfall Detection in C Programs
Tetsuro Kakeshita (Saga University, Japan),
Mariko Oda (Kurume Institute of Technology, Japan),
Yoshihiro Imamura (Sumikin System Development, Japan)

Friday December 9, 1994

9:30 - 11:30 **Parallel Session 4A : Process Modelling**

Producing and Managing Software Objects in the Process Programming Environment OPM
Yasuhiro Sugiyama (Nihon University, Japan)

Analysis and Enactment of the AttNet Model: A Distributed Software Process Model
Woo Jin Lee (KAIST, Korea), In Sang Chung (Hallym University, Korea), Yong Rae Kwon (KAIST, Korea)

Process-Sensitive Software Engineering Environments: An Object-Oriented View
Min Kang, Doug Grant (Swinburne University of Technology, Australia)

Parallel Session 4B : CASE

Generating Data Access Programs from PCTE Schemas with Constraints
Atsushi Sawada, Naruki Mitsuda, Tsuneo Ajisaka, Yoshihiro Matsumoto (Kyoto University, Japan)

An Analysis of the Effects and Evaluation of Upper CASE Tools for Embedded Microprocessors in Japan and the US
Naomi Fujimura (Kyushu Institute of Design, Japan)

Scalability for Graph Based CASE Tools
Mark Sifer (University of Technology, Sydney, Australia), John Potter (Microsoft Institute, Australia)

Parallel Session 4C : Formal Methods

PARTS: A Temporal Logic-Based Real-Time Software Requirements Analysis Method Supporting Multiple Viewpoints
Kwang-Il Ko, Kyo-Chul Kang (Pohang Institute of Science & Technology, Korea)

Formal Definitions of Behavioural Compatibility for Active and Passive Objects
Graeme Smith (University of Queensland, Australia)

The Cogito Methodology and System
Anthony Bloesch, Ed Kazmierczak, Peter Kearney, Owen Traynor (University of Queensland, Australia)

The Cogito Repository Manager
Owen Traynor (University of Queensland, Australia)

11:30 - 13:00 **Lunch (Not Provided)**

13:00 - 14:30 **Parallel Session 5A : Object-Oriented Analysis and Design**

A Sentential Function Mapping Method for Object-Oriented Analysis and Design
HyeonKon Kim, Michael Bjorn, Hui Yao, Ryosuke Hotaka (University of Tsukuba, Japan)

When to Inherit and When Not to
Yue-Sun Kuo (Academia Sinica, Taiwan)

Object-Oriented Analysis and Design Support System using Algebraic Specification Techniques
Junichi Yamamoto, Akihiko Ohsuga, Shinichi Honiden (Toshiba, Japan)

Parallel Session 5B : Software Information Management

A Hybrid Program Knowledge Base System for Static Program Analyzers
Stan Jarzabek, Han Shen, Hock Chuan Chan (National University of Singapore)

The GOODSTEP Project: General Object-Oriented Database for Software Engineering Processes
The Goodstep Team

Software Information Management System based on the Entity-Relation Model
Moon Hae Kim (Konkuk University, Korea), Young-Chul Shim (Hongik University, Korea)

Parallel Session 5C : Quality Assurance and Reliability

Prescriptive Metrics for Software Quality Assurance
Chin-Feng Fan (Yuan-Ze Institute of Technology, Taiwan)
Swu Yih (Institute of Nuclear Energy Research, Taiwan)

An Approach to Predict Software Maintenance Cost Based on Ripple Complexity
Toyohiko Hirota, Masaharu Tohki, C. Michael Overstreet,
Masaaki Hashimoto, Robert Cherinka (Kyushu Institute of Technology, Japan)

Optimal Release Policies for Hyper-Geometric Distribution
Software Reliability Growth Model with Scheduled Delivery Time
Rong-Huei Hou, Sy-Yen Kuo (National Taiwan University),
Yi-Ping Chang (National Central University, Taiwan)

14:30 - 15:00 **Break**

15:00 - 17:00 **Panel Session**

To What Degree is Reverse Engineering Possible? - Expectations and Reality -
Coordinator : Haruki Ueno (Tokyo Denki University)
Panelists :
Paul Bailes (University of Queensland, Australia)
Sanya Uehara (Fujitsu, Japan)
TBD

17:00 - 17:15 **Closing Remarks**

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