報　告

パネル討論会

明日のソフトウェア工学†

パネリスト
落水浩一郎[1]，菅野　文友[2]，木村　　泉[3]
紫合　　治[4]，竹内　郁雄[5]
司会　鳥居　宏次[6]

鳥居(司会)　本日ここに並んでいただいた先生方は，興味の対象やご専門が少しずつ違っているようで，なかなか一堂に会していただけないのではないかと思います．少し土俵を考えながら議論を進めないと，話題が発散する懸念がありますので御協力をお願いいたします．さっそくパネリストを，ご紹介させていただきます．

1番向こうから東京工業大学理学部の木村泉先生です．あらためてご紹介するのはむずかしいですね．最近先生はソフトウェア工学という点からは，表現が悪いですが少し日和っていらっしゃるのではないかと思っております．

2番めは日本電気の紫合さんです．若手だった紫合さんも今や中堅で広くご活躍されています．きょうの資料をご覧いただきましてもわかりますように，ソフトウェア工学に対する非常な熱意が感じられます．

3番めは，静岡大学工学部の落水先生です．実は今回のこの講習会の仕掛人だったようで，仕掛けたからには，少しは責任をとっていただかなければいけないだろうということです．5人の中の真中で締めていただこうと考えております．

4番目はNTT通信研究所の竹内さんです．竹内さんとソフトウェア工学とのマッチングはとれるのか，いわゆる竹内流というフィロソフィがあるのか，なども含めてきょうは是非本音をお聞きできればと思っております．

最後を締めていただきますのは東京理科大学工学部の菅野あやとも先生です．菅野先生と言えば，ソフトウェアの品質保証の中でもQC活動など現場をよくご覧になったうえでの多角的なお考えをお持ちだと存じております．

きょうのパネルは前半と後半に分け，前半ではソフトウェア工学というキーワードを与えた時に，昨日から今日，すなわち現状に対してどのようなお考えかということを，お聞きします．

後半の第2ラウンドで，いよいよ主題の，「明日のソフトウェア工学」に関して忌憚のないご意見をお伺いしようと思います．その後フロアからのご意見をいただいて一緒に議論したいと考えておりますのでご協力ください．

今日までのソフトウェア工学

それではまず第1ラウンド，「現状に関して」のお考えを5人の方々からお聞きいたします．

木村　日和っているとのご紹介をいただきました木村でございます．ソフトウェア工学という言葉はソフトウェア・エンジニアリングという言葉が先にありまして，その日本語訳としてできたのだと思います．実態が言葉のあとから付いてきたような気がします．

この言葉が最初に使われたのは，NATOの昔々の国際会議でのことではないかと思います．誰ともなく言い出されたんじゃないでしょうか．やがて例えば日本でも，ソフトウェア工学の国際会議が開かれるようなことになっていたわけですが，その日本での国際会議の頃，しきりに聞かれた苦情が2通りございます．一つは「ソフトウェア工学と言う言葉はけしからん．ソフトウェアに関係することは全部入ってしまうではないか，それでは何が何かわからない．」というご意見，もう一つは「ソフトウェア工学とは大げさだ．テクノロジならいいけれども工学なんてけしからん．」というものでした．ソフトウェア工学はこういう二つ

† 日時　昭和61年12月12日(金)，14：40～17：00
場所　機械振興会館大ホール
1) 静大，2) 東理大，3) 東工大，4) 日電，5) NTT
6) 阪大

のけしからん論にとり囲まれていたわけです．

今考えてみますと第1点は当時としてはある程度もっともだったように思われます．政治的な流れもいくぶんはあったようで，例えばデータベース分野が全部なだれ込んできた時期がありました．またOSの構造に関する細かい話までソフトウェア工学になりかねない雰囲気でした．そういう誤解は現在ではほぼ消えたといってよいでしょう．現状ではソフトウェアを作ることに関係のないことはソフトウェア工学じゃないといい切っても，多分どこからもお叱りは出ないんじゃないかと思っています．

で，第2点の方はといいますと，パネルというものは大体八百長の喧嘩に決まっておりますので，その八百長の喧嘩の題材としてはこれなんかは絶好じゃないかと思うのですが，そういう，「ソフトウェア工学とは誇大宣伝である．」というようなご意見は，大体において電気工学とか，機械工学とかの大御所の方がおっしゃいます．そういう方々は工学という時に本当に神々しいものを思い浮かべていらっしゃった．事実たとえば電気工学はすばらしく神々しいわけでして，例えば電磁気学というものが背後にばんとあって，それで全部わかる建て前になっています．しかもその上にものすごいノウハウが乗っています．そしてもっとすごいことに，脈々とした人脈が存在している．そういうのと比べられたら，やはりソフトウェア工学はちょっと見劣りがします．

しかし今やソフトウェア工学も，それなりの蓄積をもってます．ですから，そういう方々のおっしゃったことは神棚に上げておいて，いつの間にか成長してきた現在のソフトウェア工学を祝いたいと思っています．

私の見るところ，現状でのソフトウェア工学は狭義と広義にわかれるように思います．広義と言うのは，ソフトウェアを作ることに関係することなら一応何でもよろしい，という立場ですが，現在世間で主流をなしているのはむしろ狭義の方で，ライフサイクル・モデルできちんとやって，こんなドキュメントを書いてと，そういう話です．その部分では結構ノウハウがたまっている．この狭義の方は確かにいまや工学だといっていいと思う．とっさにいい例が思い浮びませんけれどもたとえば原子炉工学のような個別的工学も，世間では工学と呼ばれています．そういうものと比較して見れば，ソフトウェア工学も，もう立派に工学なのではないかと思います．

では広義のソフトウェア工学はどうか．これはまだビジョンなのではないでしょうか．で，これからこれを本当の工学にして行きたいと，私は思います．広義の方がもう立派にできている，等と言うものですから，先ほど司会者が日和っているとおっしゃったのじゃないかと思います．私は人を大勢集めて，わっと人海戦術でやるという感じに見える部分が個人的にはあまり好きでなくて，あの中には入りたくないなという気持ちがあっていろいろぶつぶつ言っていた時期があります．それを言わなくなったから，あいつは日和っているというのが司会者のご意見ではないかと思うのですが，その話は置いておいてともかくこのお席では，めでたいめでたい，これからもっとやろうよ，ということで，めでたく打ちあげておきたいと思います．

紫合　昔若かった紫合でございます．今，木村先生から，さすが，以前ソフトウェア工学研究会の主査でいらっしゃったので，非常な励ましの言葉を受けましたが，今言われたようにソフトウェア工学に対してはいろいろな見方があると思います．

パルナスは例の「SDI批判」の論文*の中で，ソフトウェア工学というのは，ソフトウェア開発において設計者が一時に覚えておく必要のある事柄をできるだけ削減していく技術である，と述べています．この見方に添って，設計法，仕様記述，ツール，管理等も考えられるのではないか．特に比較的複雑なソフトに関してはパルナスの見方は役立つと思います．例えば設計法では，一時に覚えておくべきことを少なくするための手段として，情報の局所化，モジュール化があります．モジュール化にも，トップダウン，データ抽象化，プロセス化等いろいろな方式がありますが，それぞれの分野ごとに，もしくはそこでの成熟度によって，中央集権制（トップダウン）がよかったり，地方分権制（データ抽象化）がよかったり，連邦制（プロセス化）がよかったりすると思います．しかし，一時に覚えておくべきことを少なくするという観点からは，私としては，これからは連邦型と言いますか，プロセス化のような技法の発展に期待をもっています．

落水　落水でございます．私は明日のソフトウェア工学のために今日なすべきことは，とい

* D. L. Parnas, Software Aspects of Strategic Defence Systems, CACM, Dec. 1985（高橋訳：SDIは不可能である．世界（岩波）86年6月号）．

う立場でお話をさせていただきます．結論をまず述べさせていただきますと，実際の要求分析定義・設計・保守等で行われている一つ一つの活動や，そこで利用されているデータをもっと詳細に観察してモデル化する必要があると思います．モデル化にあたっては鋭い洞察が必要ですが，その前に，まず，事実に正直でなければなりません．これは非常にコストがかかり，かつ，忍耐のいる仕事ですが，その作業が前提となって初めて，抽象したり，モデル化するのに値する豊かな知識が手にはいるわけで，ソフトウェア工学の発展のためにはさけてとおれない部分であると考えております．そのような知識を潜在的に保有しているのは実務担当者です．実務担当者に内在している活動プロセスや情報構造を抽出し，明示化する必要があると思います．ソフトウェア作りの長い歴史の中でなぜそのような知識が誰もが利用しうる形で整理されなかったのかという問題を考えてみます．まず，個人の世界，すなわちプログラミングの世界の特徴は与えられた問題をよく理解し，また実現しようとする計算機の性能・機能をよく把握してその両者をつなぐ手段（デザイン・コンセプト）を見つけるという作業パターンになります．このとき，最も肝心な設計上の意思決定過程は個人の中に埋没してしまいます．一方，ソフトウェア工学が真に必要な大規模ソフトウェアが対象になりますと状況はもっと悪化します．まず，業務の世界がきれいに定義できないから問題が形にならない．また，業務世界の規模からして実現しようとする計算機の世界と問題の世界をそう簡単にはつなげない．しかも，その遂行の担い手は複数の人間であり全体の作業をきれいに分離できないということもありまして，結局，ベース，ライン制御という大まかな手段にたよらざるをえなくなります．こういう状況下でソフトウェア作りに関する有用な知識が集積されるとは思われません．本日は，大規模ソフトウェア開発プロセスのモデル化にあたって，その最も大きな障害となる問題世界の定義や理解のプロセスに関わる知識を収集・整理するためのアプローチを紹介させていただきます．そのような努力こそ明日のソフトウェア工学を築くためには，まず必要であるというのが私の立場でございます．

竹内　竹内です．３番目の落水先生で中締めをして最後の菅野先生で締める，ということはその中間におります私は，締めを締めらしくするために話をだらしなくしなければいけないと思っております．私とソフトウェア工学のかかわりに若干の疑問があったようですけれども，それは正解でしょう．一昨年まで私はソフトウェア工学研究会に入っておりました．去年まではいやいや入っておりました．資料だけは送ってもらっていました．今年はやめました．私とソフトウェア工学の関係はそんなところです．さて実は私，きょうここにくる電車の中で途中ずっとマイクロプログラムの虫とりをしてきました．こういう人間には今日についても明日についても語る資格はなさそうです．ですから私の題だけは，明日のソフトウェア工学じゃなくて，明後日と言いますか，あさってのソフトウェア工学にしました．当然，話の内容もあさってです．それは次のラウンドでやることにします．

菅野　四捨五入して60歳の菅野でございます．1971年4月12日と言いますと，ちょうど15年半前です．『日経エレクトロニクス』の創刊号の中でソフトウェアは作れなくなるんじゃないかという話が10数ページに亘ってルポ形式で出ておりました．今が1.5メガステップで，２メガになるともう効率ゼロだ．どうしようか，ということで，企業のメンバがいろんな意見を出した．その中で今残ってるのは一体何だろうかと見ますと，高級言語を使ったらいいだろう，あるいは芸術的センスでやったらいいだろうというのは大体なくなりまして，私が誤解と偏見で見るところ段取り万全．作戦単純，それしか残ってないんじゃないかと思います．これはつまり定石の多角的活用です．昨日，今日とお話がありましたのは，全部すばらしい定石です．その多角的活用というのは，すべて ABC にある．ABC というのは簡単に言いますと，あたりまえ(A)のことを，ぼんやり(B)せずに，ちゃんと(C)するのが ABC です．ところが問題は何があたりまえで，何があたりまえでないかという価値判断が全部違うわけです．またぼんやりせずにとおっしゃいますけれども，ぼんやりしている本人は気が付かない．最終的に気が付くのは，そういうものを売り込まれたお客さんです．１人１人はぼんやりしても，みんなが集まればぼんやりしない仕掛けは何かないだろうか．ちゃんとしたかどうかというのは自分ではわからないんです．ちゃんとしたかどうかは，他人が冷たく見なきゃわからない．その辺のところが，今の大変大きな問題じゃないかと思います．実学の立場

から見ますと,.日本でのソフトウェア開発の費用の総額は，大体100億円弱というのがユーザ・レベルでの話で，その中で9割以上がオーダ・メイド，つまり自社開発及び外注です．ところが工数経費で見ますと，自社開発を10として，外注は5，ソフトウェア・パッケージを利用すると1ないし2です．つまり，自分が作らなかったソフトウェアを使って大事な仕事をするという世の中です．これがうまくいくためには，やはり世の中の信用を失わないようにしなければいけない．信用というのは論理じゃないんです．感覚です．感覚的な信用が失われたら，それを論理で再構成するのは非常にむずかしい．今やソフトウェアの危機ではないかというわけです．そのためにどうすればいいかというと，日常茶飯事やはりいろんな経験を積み重ねております．その経験の体系化で共通的な標準をはかる．と同時に四捨五入して60歳というのは，失敗の歴史です．失敗を何とか活用しよう，こうすればだめだよということを，恥をさらして反面教師とする．これもまた存在価値はあるだろうということです．成功例よりも失敗例に学ぶべきことがまだまだいっぱいある．花田先生がいみじくも昨日お話いただきましたように，データを出し合おうじゃないか．今こそまさしくそういう時期ではないかと思います．そして実績に立脚し，氏素姓を硝子張りに確保していこう，こういう考え方は，森口先生が，ソフトウェアの品質管理ということでお話いただきました路線じゃないかと思います．現状は日暮れて道遠し，正に暗澹たる気持ちです．

司会 ありがとうございました．やはり予測してたように，話題がかなり広がりそうです．竹内さんの話ではもう少し荒れるかなという懸念はあったんですけれども，少し自制をしていただいているようです．フロアから何か意見があるでしょうか．…．やはりないということは，昨日から今日までの現状の認識には，さほど差はないと理解させていただきます．

明日のソフトウェア工学へのかけ橋

それでは第2ラウンドに進めたいのですが，ただし，ここではまだ夢を語らないでください．夢は最後に残しておいていただいて，現状の技術，および今後開発されるであろう技術を踏まえて，「あさって」ではなく「明日」のソフトウェア工学へのイメージなりビジョンをお話いただけないかと思います．できましたら，可能な限り立脚できる技術的な話題を，具体的に話していただければ，パネルの共通土俵としてわかり易くなるんじゃないかと思っています．木村先生からお願いします．

木村 私は夢のない男なものですから，夢は語るなと言われても，全然困らないんです．とにかくとりあえずやれるといいなと思うのは，いま菅野先生から巨大なソフトのお話がありましたが，巨大でないソフトについてもよく考えて，巨大なのとそうでないのを両方包むような形で工学化ができないか，体系的にきちんと押さえてもう少し世の中の風通しをよくできないか，ということです．と言いますのは，これ私あちこちで言って歩いているんですが，ワープロソフトというのがありまして，ものすごい競争であちこちで作られて，すごく売れてる．売上げが何10億となってるわけですね．そしてユーザに愛されて世の中を確かによくしている．小さいソフトウェア・ハウスさんで作っておられて，当然そういうのは巨大でないわけです．中には真偽のほどはわかりませんが，「タコ部屋方式」と噂されているところもある．そういう場面でも役に立つようになって行かないと，やはりソフトウェア工学というのは誇大宣伝になるんじゃないか．今度は「工学」のほうが誇大宣伝になるんじゃなくて，「ソフトウェア」のほうが誇大宣伝になるんじゃないか，こういう気がしております．

そのためにはどうしたらいいかということですが，ちょうど電気工学で電磁気学があるように，バックの理論というようなものがなければいけないんじゃないかと思う．つまり，こうやるとうまくいくよという話だけではなくて，なぜそれでうまくいくのか，なぜそれ以外ではだめなのかという，そういうサイエンティフィックな立場がこれからは大事だと思います．

私としては，これは今たまたま非常に興味を持ってる方面なものですから，多少我田引水的に申しあげますが，人の頭の中の仕掛けを理解する，という方向が大事だと思います．例えば創造工学という話に近いのではないかと思いますが，プログラミングとか，システム設計とか，システム分析とかをやっているとき，人の頭の中はどうなっているか，どうやってアイディアをためていって，どうやってそれを，どこかに飛んでいってしまわないようにまとめるか，そういう部分をもっとちゃんと調べる必要があると思います．先ほど引用されたパルナスの話は，考えなければならないことをどう押さえ込もうか，という話でしたが，それは人間の頭の中にはそういっぺんにはたくさんのものが押

し込めないので，問題の方を押さえ込まなければいけないわけで，もとへ戻ると，どうしてたくさん押し込めないのか頭の中の構造から出発して理解するということが必要じゃないかと思うのです．それはプログラマ，分析家，設計家，みんな含めての話だと思います．

近頃はソフトウェア心理学という話がありまして，端末のやりとりのあたりですとかなりいろんなことがわかってきています．それを見ますと，人間の頭の中のこともかなりモデル化できて，それをもとに将来は実際的なソフトウェア技術のバックグラウンドにあるサイエンスをだんだん作っていくことができるじゃないかと，これは夢ではなくむしろ見通しとして考えています．菅野先生と昔はずい分違うことを考えていると思っていたのですが，ぐるっと回ってこの頃は同じことを考えてるようで，私も進歩したんじゃないかと思います．一応ここまでにさせていただきます．

司会 今，木村先生のコメントに対して，フロアも含めてすぐに何か言いたいという方がいらっしゃいましたら，どうぞ．

荒武(管理工学研究所) 私どものところでワープロ・ソフトを作っていますが，ワープロ・ソフトとソフトウェア工学ということでお考えをおききいたします．

木村 管理工学さんは多分きちんとやっていらっしゃると思います．しかし無責任な噂かも知れませんが「あそこはタコ部屋じゃ」と噂されているところもある．びしびしと徹夜をして作るんだそうです．そういう古風な話が聞こえる一方で，いい製品が出てるんですね．とすればいまのソフトウェア工学にはカバーし切れていないところがあるわけです．そういうことを言いたかっただけです．

司会 どうも答えにくいのを，ありがとうございました．フロアからほかになければ，次は紫合さんお願いします．

紫合 それでは先ほどに続きまして，方法論の辺をもう1度振り返ってみることにしたいと思います．まず設計法としては，データ抽象化技法がいいとよく言われます．データ抽象化ではオペレーションの集合とその間の規則によって抽象的なデータが決定すると言えます．抽象化のために一つのモジュールに複数個のオペレーションをまとめて実現しますが，そのオペレーション集合だけを見て抽象データのイメージがわくというわけです．これをよく考えますと，オペレーション集合のある部分集合だけに注目しますと，別の抽象データのイメージになるはずです．例えばスタックのプッシュだけを見ますと，ライトオンリのファイルが，ポップだけだとリードオンリのファイルという，スタックとはちがった抽象的なイメージが浮びます．すなわち抽象データのイメージはモジュールに対応するのではなくて，注目すべきオペレーションの集合に対応するという話です．

次に，プロセス化，またはデータフローの技法と抽象化との関連について述べたいと思います．最近思いますのは，プロセスも結局データ，または抽象装置のイメージが思い浮ぶということです．プロセスの間を信号が行き来する，例えばAランプ・オン，ランプ・オン OK というような信号のやりとりがありますと，そこから「ランプ」という抽象装置の概念が出てくる．すなわち，信号の集合と信号のやりとりの規則(プロトコル)によって抽象装置という概念が出てくるわけです．ここでも抽象装置の概念はプロセス自身の属性ではなくて，信号の出入りするポートの属性と考える方が自然です．この概念を電気の例でいいますと，例えば100ボルトのコンセントでは，その向こうに原子力発電がつながろうが，水力発電がつながろうが関係ないわけで，とにかく100ボルトの交流電源という抽象装置がイメージされます．一方その裏側から見ますと，そこに電気掃除機がつながろうが，パソコンがつながろうが，要はインピータンスという抽象装置がつながると考えます．コンセントの表裏の両方側から抽象的な装置のイメージが見えてくる．こういう考え方がモジュール化にとっては重要じゃないかと思っています．

プロセス化の考えはマイヤーズの構造化設計にもあります．そこでは，入力から出力への変換プロセスの系列を考えて，その中から中央変換プロセスを選びます．そこで，中央変換の入力側が最大抽象入力，出力側が最大抽象出力ということになります．けれども，どこが中央変換になるかを決めるのは難しいし，プロセス化の方法論ではあまり重要ではないといえます．変換のそれぞれのプロセスの立場でみると，その出力側が最大抽象出力で，入力側が最大抽象入力になるわけです．いわば，「相対的抽象化」といえます．

一般に，いくつかのプロセスが信号をやりとりしてるというモデルでは，その信号集合とその間の規則によって抽象装置のイメージができてきます．そうすると，各プロセスの環境となる抽象装置がどういう性質で，どういう変化をしていくべきかということを理解

することによって，そのプロセスの処理内容が理解できるわけです．すなわち，相手方プロセスの中身を理解することは必要ありませんが，単にインタフェースとなる信号がどういうふうにくるかということだけではだめで，そこに抽象装置という概念がないと理解できないのではないかと思います．

この考え方でいきますと，プロセスの仕様は，その環境の変化をみて判断し，その結果どのように環境を変化させていくかということで述べることができます．これは，環境の中での登場人物である抽象装置が変化していく様子を書くわけですから，nコママンガみたいになる．そこでは，ある場面と次の場面の差からその間の動作が想定されます．具体例としては，交換機の状態遷移図として実際に使われてる仕様記述(CCITT の SDL) があります．状態遷移図中の安定状態を示すノードの中に，電話やベル，回線，課金装置等の抽象装置，すなわち，登場人物の絵を書きます．例えば，電話のベルが鳴っている状態で，相手が受話器を上げると通話状態に移りますが，それを2台の電話，ベルを鳴らす装置，課金装置，回線，タイマ装置等の登場人物の変化の絵で書かれています(**図-1**)．この仕様から，リンギング状態から通話状態に移るときはベルを止めるとか，課金をスタートさせるというような処理が必要であることが自動的にわかるわけです．このような，マンガによるプログラミングという方式が，今後はもっと汎用的にできていくのではないかと思っています．

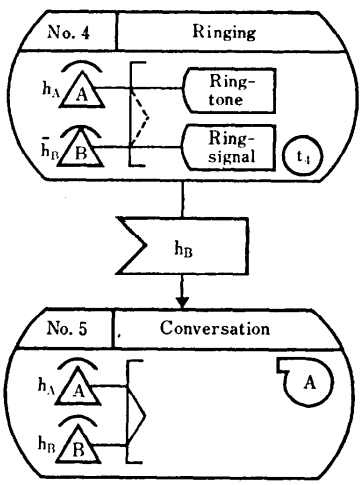

図-1 CCITT/SDL による
交換機の状態遷移図 (一部)

CCITT RED BOOK VI-VI. 11 (1984)
Functional Specification and Description Language (SDL)
Annexes to Recommendations Z 100-Z 104 より転載

司会　どうもありがとうございました．概念的な話が入って参りましたが，フロアからどなたかご意見がありますでしょうか．

直田(富士通研究所)　今抽象化とプロセス化の二つに分けられましたが，それを一つにまとめてしまえば結局オブジェクト・モデルと言うか，昨日ご講演いただいた東芝の松本先生がおっしゃってた平行オブジェクト・モデルとか，そういったようなものになるんじゃないかと思いますが，何か違いはあるんでしょうか．

紫合　オブジェクト・モデルの場合，メソッドを送るというのは，信号を送るというより，オペレーションをコールしているという感じが強いと思います．メソッドの受け取り側も，オブジェクト指向の場合は，各メソッドの実現が並んでいるわけですが，プロセスだと信号の読み込み命令がプログラム中にばらまかれることになります．ですからプロセス指向では，信号を送るほうと受けるほうは全く対等であり，オブジェクト指向では，送るほうが受けるほうを制御する，すなわち上位にあるという違いがあると思います．また，プロセス指向では抽象装置（抽象データ型）がポートの属性であるのに対して，オブジェクト指向ではそれがモジュール（クラス等）の属性になるといえます．

司会　今，プロセスという言葉をお使いになったのは，資料にもお書きいただいてますように，ジャクソン法に出てくるようなプロセスだと思います．もともとプロセスという言葉は，「OS ではプロセスがいくつも走りますよ」という考え易いのがありますね．

紫合　はい．私がここで述べたプロセスと言うのは，1番考え易いのは，シーケンシャル・ファイルを介してソートやマージをしてレポートを出力するというような処理のジョブステップというか，実行単位ですね．また，並列性を考えますと，例えば新しいほうのジャクソン法(JSD)で言うプロセスとか，CSP でいうプロセスとかを考えていただければよいと思います．

司会　それでは次に落水先生お願いします．

落水　従来型の開発パラダイムで，要求定義から設計につながる部分の技術精度を高めていくにしろ，昨日から色々お話いただいています自動化パラダイムの世界，すなわち実行可能仕様と自動変換の世界で今後やっていくにしろ，応用分野の知識を整理し，システムの構造に変換していく世界でおこなわれていること

に対して，具体的な事実やデータが整理されていないということは今後の技術発展に非常に悪い影響を与えると思います．1人で完全にわかってプログラムを作っている時のような楽な感じにはいつまでたってもならないような気がします．この問題にどのようにアプローチしていけばよいかということに対する一つの考え方を紹介させていただきます．流行をおいかける前に，現実をていねいに観測して，その中で成り立つ一般性を見つけることを（設計方法論が初期に作られた時に行われたようなことを），もう1度やり直す必要があるのではないかという意見です．「そんなことをいくらいってもきりがない，提案ばかりしてもしかたがない，自分自身で努力し，その結果を示しなさい」と，鳥居先生に叱られたことがあります．その当時は，ちょうど努力の最中で「やってます」とは言えない状態だったのですが，今日は少し話ができます．

実は5年ほど前にデータベースを中心にしたCAD環境を作成した時に，最終的に行き詰まった点があります．それはデータベースの中に何を入れるのかという問題です．開発時のドキュメントを少しぐらい整理して入れたところで，開発時の変更支援にも，保守支援にもぜんぜん役に立たない．そういう反省を持ったことがあります．それがきっかけになりもう少し地道に努力をしなければと思いました．実際に始めたことは，対象を事務処理システム，観測するプロセスをメンテナンス，観測の媒体としてはドキュメント，等をもとにしまして，そのドキュメントを手掛りに開発者や，保守SEから，彼等の中に内在するプロセスや，そのプロセスで利用されている情報構造をモニタリングするという作業でした．事例としては固定資産システムを選び，まとめるのに3年かかりました．ただし1年目は，実務レベルの方との話し方を覚えることだけでした．「要求定義から設計に移る時は横のものを縦にするんだよ」というお話がありまして，それがわからないというような状況が続きました．これは大学人固有の労力です．この固定資産システムとは，開発の初期から付き合わせていただきまして，今ちょうど巣立ちコレクティブ・メンテナンス受けてるという状況なんですが，その開発時文書とか，保守に使われてる文書をいただきまして，それを見ながらその文書の中に含まれてる情報は何か，それがどのような構造を持っていて，どのように作られていくのか，開発時やメンテナンス時に，どのように利用されるのかというところを一生懸命調査したのが2年目です．と言いましても実際にそれをやって，モデルを作ってくれたのは学生さんなんですが．3年目には，こんなふうな形でいいんじゃないかというところまでいきましたので，小規模な評価実験をやってみましたところ，かなりうまくいくという感触をつかめました．つまりユーザの保守要求から，プログラム中の変更箇所を特定していくプロセスに関してその明示化の見通しがやっと立ったところです．開発時文書は保守には役に立たない．それを計算機支援環境中のデータベースに入れてもやはり役に立たない．保守SEはどのような情報をどのように利用して保守をやっているのだろうかという疑問に対して，われわれとすれば，やっとすっきりした状態です．内容の詳細については，研究会等で，実際やってもらった学生さんに発表してもらってますので省略させていただきます．

さて，最後に，このような事例の積み上げの結果として我々が手に入れることができるものは何かということに対する私の考えをまとめさせていただきます．ソフトウェア工学の具体的な成果としては，計算機ハードウェアの動作の世界を仮想化することにより得られた，言語と環境に関する成果があります．それに比べると要求定義の成果も，設計方法論の成果もきわめて不十分です．その理由としては，業務世界を抽象することの困難さがあげられます．この問題に対して，例え個別的であり，応用分野に依存しても，一つ一つソフトウェア開発・保守に関わる事実を明らかにしていけば，何かがその次に見えてくるはずです．ところで観測とか，モデリングというのは非常にコストがかかる作業です．当然そのような作業を助ける環境を作る必要があるでしょう．実はそのようにして作られた環境こそ，集積されつつある知識を有効に活用してユーザ，保守SE，開発SE等の作業を大幅に軽減するものになると思います．計算機の世界を使いこなす環境と業務世界を制御するための環境を使いこなして，その両者を接続することの満足感と楽しみの世界を実現するのが私のソフトウェア工学の目標です．

司会 どうもありがとうございました．今の落水先生の例題は現実に動くものなんですね．

落水 はい．

司会 どれくらいの規模のものなんですか．

落水 ステップ数は正確に聞いてないんですが，プログラム本数は百数十本程度で，開発時文書は24種類ぐらいです．ただ，実際に整理した結果としては，その内のほんの4〜5種類が保守のためには重要であ

るということがわかりました．

司会　ありがとうございました．竹内さんお待たせしました，お願いします．

竹内　もう少し乱暴なことを話せという趣旨の発言があったように思いますが，ご希望にそえるかどうかわかりません．しかし，これからお話しすることは半ばジョーク半ば本気ではなく全部本気なので取り扱いにご注意ください．

プログラムには虫があってはならないというような脅迫観念が今だに多くの人にあるわけですが，まずこれがなくなります．なくなった以上は，あとは保険の問題になりますから，プログラムバグ保険というものが登場します．この保険料率の計算に対して工学的思考が使えるかどうかということになるわけです．ところが，今までのソフトウェア工学の研究者は保険のことは全然知らないので，全然コントリビューションができない．そこで××生命の人が劇的な論文を出して，解決を見ることになります．要するに今の生命保険とか，火災保険と同じなんじゃないかと思います．このプログラムの保険料率はいくらかと言われた時に，過去1年間に重大なバグを起こしましたとか，システム・クラッシュを何回起こしましたかという問診があって契約が行われるという感じでしょうかね．結局あそこのメーカが作ったんだから，あそこのソフト・ハウスが作ったんだから，これくらいでしょうという純粋に統計的な話になっちゃって，信頼性評価技術とか何とか言ってるのは，多分役に立たないだろうと思います．

次ですが，これは既にその兆候が表れてるという説があるんですが，プログラムを今日ほどたくさん作る必要がなくなります．なぜそうなるか言わないといけないんですけれども，今のソフトウェアのプログラムの作り方を見ていますと，ちょうど20年前ぐらいのディスクリードな3本足のトランジスタやコンデンサを組み合わせて，電気回路作ってた時代と非常に近い．ところがLSIができて，そういう回路屋さんが昔に比べるとどんどん減ってきました．それと同じようなことが，いわゆるプログラマという職業についても起こります．例えば，今よりは昔のほうがOS屋さんという人が多かった．なぜかと言うと，OSというのはもう何か言ってもあれの系統ですね，という感じになって，いい悪いの問題とは別に，世の中の固定化しちゃった．つまり，ああUNIXですか，と言って，UNIXに毛がはえたり，なくなったりするレベルで話がすむようになってしまった．さらに言いますと，世の中のニーズのほうが，既にあるソフトウェアのほうに合ってしまう．1番典型的なものが，給与計算システムのようなものでしょう．さっきのお話に出た固定資産システムは，おおかたの場合最初から作る必要はないわけで，うちの固定資産システムは，あのソフトに合わせてしまおうというふうに世の中なっていく．そういうことがまさにさっきの回路技術のほうに起こってるわけですね．このLSIはTIのほうに合わせてしまおうという話になってしまうわけです．だからプログラマが減り始める．要するにスクラッチからプログラムを作る必要が減少する．

もう一つ，私は要求定義技術とか，プログラム自動合成技術とかが何のことかよく知らないから勝手なことを言うんですが，これらは何か自動的に作っているという雰囲気をかもし出すために冗長なトートロジを加えるだけのものです．だからそのうちそういうふうには言われなくなる．つまりただのプログラミングであると再認識されることになります．それは歴史的な事実が物語っているとおりです．昔アセンブラはオートコードと，オートという言葉が付いてました．それからFORTRANが自動プログラミングでしたね．と言うわけで，今の自動合成というものそのうち言葉としてはなくなってしまう．常に研究者は自動という言葉を追い求めるものですけれども，実は単に冗長度を加えていくだけなのですね．だから冗長度をどれくらい加えればいいかというような理論が多分出てくるに違いない．シャノンの情報理論によりますと，英語のアルファベットの冗長度というのは2.5ぐらいですか．自動プログラムにしてもあまり冗長にしたら書くほうが大変です，だから最適なレベルが何かあるに違いない．そこで天才的な学者が1人現れまして，最適冗長度はこれだというものを2016年12月2日に発表するわけです．

それからもう一つ，ソフトウェア工学というのは，どうもおもしろくない．研究会に入るのもやめたくなる．つまりどうもソフトウェア工学で作っているソフトウェアは人間味がないという話が出てくるわけです．そこで，こういう方法論に厳しい批判が加えられます．批判をするような人たちはプログラムを作る腕としては1級のものを持ってますので，分派活動を開始します．われわれのプログラムは芸術だと言い始めるわけです．ところが日本芸術院からは，そんなものは芸術ではないと断定されてアウトサイダの道を歩み

出すことになります．そういった連中はしょうがないので，ちまちまと学会誌とか論文誌などにものを発表するわけですが，いわばプレタポルテであって，ほとんど世の中にはインパクトを与えない．某社の某ワープロシステムが世の中を席巻するがごとく，いいか悪いかは別として，なにかが一旦デファクトスタンダードになってしまうと最後までそれで突っ走ってしまうという固定化現象が顕在化して，結局ソフトウェアの進歩は1992年から止まるというのが私の予測であります．

司会 どうもありがとうございました．やっと竹内さんの片鱗が聞こえてきたようでありまして．今すぐ反論したい方いらっしゃいますか．いなければ菅野先生にお願いします．

菅野 私の基本的な考え方は，竹内さんのお話と120% 合致しております．姿・形・清潔感が違いましてもこれだけ人生感が一致するというのは非常に珍しい真実です．ただし一つだけ私は追加したい．それはPL の問題です．プロダクト・ライアビリティ，つまりソフトウェアの法律的な責任です．おそらく竹内さんは，そのパイオニアとして，これから華々しく活躍されるだろうということです．さて皆さん方『bit』誌にワインバーグさんのイーグル村通信ということで文章が出てることをご存じでしょう．あれは木村先生が翻訳しておられるんですが，あれは嘘です．本当は木村先生が最初に，あのすばらしい日本語でお書きになったものを，あとでワインバーグさんが英語に翻訳されている．そういう感じのすばらしいものです．そのワインバーグさんというお名前で，木村先生のお話が出ております内容は，その随所に“人間は”という感覚で，すでにお亡くなりになりました碩学の高橋秀俊先生がお話なさってる内容とぴたり一致しているわけです（図-2）．これは決して私がモデルになってるわけではございません．すべてこういうことです．そういう人間性は，時の古今，洋の東西を問わず，同じ感じがするわけです．目の色が違い，皮膚の色が違っても，同じ感じ方をするわけです．ところが，だんだん社会的に近付いていきますと，どうしても文化の差が出てくる．ということで，資料の中の表-1 には，農耕民族と狩猟民族の差がどうしても出てくるわけです．マイコンの信頼性向上のポイントにも，民族的な文化の差というのが，出てくるということがあります．これは，日本の現在のメーカで，非常にうまくやってるところはこういうふうにやっているというノウハウの開陳です．まず図法による文書化があります．そうしますと HCP がいいですか，PAD がいいか決めなさい，という方がよくありますが，何でもいいわけです．例えば，機能仕様書には HIPO を使う，設計仕様書には HCP を使う，製造仕様書は PAD を使う，というように使っても，さしつかえないわけです．この図法での文書化ということの次に，機能モジュール化があります．つまり，作る立場でのモジュールではなくて，ユーザ・ニーズでのモジュール化をして，その再利用を克明にやることです．要するに，あたりまえのことを，ばかにしないでちゃんとやってるわけです．次にデザインレビュですが，これが日本的な感覚です．もともとは，設計審査という名前があるわけですが，そこのところをぼやかして，デザインレビュというハイカラな言葉を使うわけです．前もって源流を清め，弁当箱が流れてくるのを川下で拾うよりも，川上で流さないほうがいい．先憂後楽一先に憂いて，後で楽しむということで一生懸命やろう，ということです．次は，皆さん方の一部の方が大嫌いな QC サークルというものがあります．これは，職場小集団活動で，お互いに提案制度をがっちりやっていこうということにも通じます．近藤次郎先生（日本学術会議会長）のお書きになりました本で．「巨大システムの安全性-事故はなぜ起きるか」があります．これを見ますと，やはり QC サークル的なアプローチの重要性を説いておられます．モチベーション，やる気，があるかどうかではっきり違う．モラル・アン

表-1 農耕民族と狩猟民族

項番	農耕民族	狩猟民族
1	話合い	契約
2	多元論理（玉虫色）	二元論理
3	残す	残さない
4	集団	個人
5	エントロピ減少	エントロピ増大

怠けものである
不注意である
気紛れである
単調さを嫌う
ノロマである
忘却度が不分明である
何をするか分からない

図-2 人間とは

ド・モラールが良くも悪くも，われわれ民族の特色です．それから，稼動実績データベースが大切です．要するに氏素姓をちゃんと硝子張りに知られるということです．次に，生まれ悪ければ育ち悪し，ということがあります．信頼性（こわれにくさ），保全性（直し易さ），拡張性というのは，最初の段階で考えなければいけない．だからデザインレビュの段階で徹底して考えることになります．次に直行率の向上，これが実は大変なパロメータでして，途中でもたもたしているとだめです．東京理科大学にも大分留年してるのがいるわけですが，とにかく直行率の向上で親が安心するわけです．次に，調整箇所縮減によるトランケーションに注意することが大事です．例えば昔の無線機では，あちこちにずい分いじるところがありました．その理由が一見理路整然としていて，電波というのは不規則に減衰が起きる，あるいは電離層というのはいつも変わるんだと言うわけです．近頃の電波機器を見ますと，全然そういういじるところが付いてない．じゃ電波は変動しなくなったのか，フェージングは起きないのかというと，昔どおりあるというんです．やっぱり技術が未熟の間は調整箇所がいっぱいあり，その調整箇所をいじることによって無理やり規格に入れ，あるいは手直し調整をして出す，ということだったわけです．トランケートした場合には，分布形では必ず規格ぎりぎりのところで納まります．したがって，規格がちょっとずれますと，すぐマージン（余裕）はなくなるわけです．次が，仕損費率です．どれだけ損したかを，標準（あるいは目標）原価に対してどのくらいでできたかを，きちんと目標管理することです．次に，要求仕様を作った人は，これには間違いがないと思ってはいけない，まだまだ間違いがいっぱいあると考える．だから謙虚なゆとりを持たなければいけない，ということです．

さて，これからは高学歴，高年令化時代です．高学歴というのは，決して頭のいい人がいっぱいふえることじゃないということは，自分のことを考えてよくわかるわけです．どういう人がふえるかと言うと，屁理屈こねて仕事をしないという人が増えるわけです．高年齢化というのはどういうことか．ジイサンと，バアサンが両方出るわけです．そういうジイサン/バアサンも一緒に，ソフトウェア産業の中に入って働いていただきたい．そういう方々に何をやっていただけばいいか．新しいアイディアに対する柔軟性はとても望むべくもありません．どうすればいいか．やはり酸いも甘いもかみ分けた包容力，そしてあのジイ様とよくも我慢して一緒にいるというすばらしい忍耐力が重要です．それからいろんな立場で，相手によって言い方を変えるという，一本気でない，そういう老練さが発揮されます．そういうことを考えますと，先ほどのデザインレビュは，ある程度円熟しないと，とてもできません．そういうことがたくさん出てくるということです．したがって，これから先は，大いに長生きして頑張らねばいかんということになります．もちろんこれには，ツール，道具というのが一杯必要なわけです．

司会 非常におもしろくお話いただきまして，ありがとうございました．

今までお話しいただきました中で，いろいろキーワードが出て参っております．それを思い出していただく意味でも，少しおさらいをしたほうがいいかと思います．木村先生のお話から参りますと，場合によっては，ソフトウェアが誇大宣伝になるんじゃないだろうか．又，先生ご自身は人の頭の中の仕掛けを解明したい，ソフトウェア心理学とか，創造工学というような言葉が出ておりました．さらにレビュの技術，やはり単なる技術じゃなくて，それなりに確立したものを望んでいらっしゃるようでした．

紫合さんのお話は従来のトップダウンと言うような一徹なやり方だけではなく，データ・アブストラクションのほかにプロセスの方からも考えたいということだと思います．

落水先生のお話では，具体的に，50万ステップでしょうか，データを出し合いましょうということの典型的な具体例になってるかと思います．開発時の文書とか保守用の文書を媒体としたプロセス・モニタリングしモデル化をやってみました．やはりソフトウェア工学というものをそれなりにきちんとやっておれば，うまくできてきたんじゃないかということだろうと思います．

竹内さんのお話では，社会のほうがパッケージに適合していってしまう．また，スタンダード化が進んでいくんじゃないだろうかとか，それから要求定義技術に対しては，菅野先生も一体どういう意味かよくわかないけれどもというお2人のコメント付きのキーワードのディファレンスがあったように思います．最後に竹内さんのほうからは，工学の話をしてるんですけれども，芸術というところにもどっていくんだと，1992年とおっしゃいましたが，これは容易ならざることでありまして，あと数年しかございませんので，できれ

ばその根拠をお聞きしたいですね.

最後に菅野先生からは，非常に私ども身につまされる教訓をたれていただいたわけで，仮にマイコン・ソフトとはおっしゃいましたけれどもマイコンに限られた話ではないと思います．図法による文書化というのは，ビジュアルアプローチ，デザインに関してはもう昔から問題になっている話題です．それからQCに関しては独壇上のお話でしたし，最後に人生同様お年寄りの話で締めていただきました.

そういうことで，もう少し，単にソフトウェア工学と言うか，ソフトウェアだけが1人歩きできるわけじゃなくて，ハードとか人間とのインタフェース，さらには人間の心理的要因等々，かなり学際的な要素も考慮しなくてはならないのでしょう．その辺も含めまして，フロアからご意見ありましたらお願いいたします.

河村(日本電子専門学校)　明日のソフトウェア工学というテーマなんですが，その中で今回出てこなかった問題として技術移転，テクノロジ・トランスファがあるんじゃないかと思います．今後ソフトウェア工学の分野でもいろいろな新しいイノベーション，技術革新が出てくると思いますが，それをいかに一般に普及していくかということが非常に大きな問題になるんじゃないかと思います．今日，学際的なレベルと，実務的なレベルに非常に大きなギャップがあると言われてる一つの大きな原因は，やはりその技術移転があまりうまくいってないのではないかという気がするわけです．例えば，30歳，40歳のSE，あるいはプログラマたちが，自分はSEでいろんな経験を積んできたと言っていますが，そういう人たちがAIとか新しい技術に対応できていくかと言うと，やはり自分が長年積み上げてきた経験というものがありますから，なかなか次にいけないという問題があるわけです．このため，いかに技術移転をやっていくかということが大きなテーマだと思います．技術移転がうまくできていくとするならば，ソフトウェア工学というのが学際レベルから，実務レベルにどんどん降りてくるような気がします．そのために一体どうすべきなのか．大学の先生もいらっしゃいますので，その辺を踏まえてお聞きしたいと思います.

司会　これは大変な問題だと思いますが，パネリストの皆さん方にお聞きしたほうがいいのかと思いますが.

菅野　今のお話は大事な問題です．世の中に実例がなかったかと言うと，実はいっぱいあるわけです．というのは日進月歩ではなくて，むしろ秒進分歩です．学校で教えられなかったことをやらなければ飯が食っていけない．しかも年齢は，先ほどのお話にありましたように，専門の仕事をやってる間にどんどん高くなる．そこでどうするかという話です．ある意味では簡単で，ある意味では難しい．簡単なほうから申しますと，自分が持っている技術は，技術移転をどんどんするということで，自分がまた新しい技術に取り組めるのは当然です．しかし，その技術移転のためのツール，ノウハウということを知っていないために，自分が持っている技術が移らず，いらいらする．まわりからも，あの野郎，墓場に持って行く気か，と言われる．ではどうすればいいのかと申しますと，経験が大事です．経験を積んだ人は勘を持っているわけです．そしてわれわれは度胸で仕事をしている．その経験と勘と度胸というものは，すべからく技術移転すべし，これが私の持論です．で，経験ということはどうやって技術移転するか．経験には三つあります．どんな現象を今まで自分が経験してきたか．どのような原因で，悪くなったり良くなったりしてきたか．そしてどのような対策を自分が打ってきたか．この三つになるのではないでしょうか．そうしますと，その現象の確認，原因の追究，そして対策の吟味，こういったものを気軽に表現し，皆でそれをディスカッションして，後輩がまた先輩から聞き出す．こういうことをやっていったのが，今の新日鉄とか，建設会社の技術移転なわけです．その道具が，特性要因図（魚の骨，フィッシュ・ボーン）です．まずそういうことで相当程度クリアになる．衆知結集の一つの方法です．もう一つは勘です．勘というのは論理的には全く関係ない筈のものを結びつけることです．つまり，代用特性値で，真の特性を推定できることです．これには，散布図が利用できます．例えば二次元の散布図を書きますと，必ずそこに相関関係の有無が出ます．もちろんリニアな関係ですが，ノンリニアの場合には縦軸，横軸の変換をすればいいわけです．そしてその相関系数が出ますと，その2乗は，寄与率です．勘が非常によく当たるというのは，寄与率が高い，勘が当たらないというのは寄与率が低い．データを集積して散布図を書くことが大事です．最後の度胸です．石橋を叩いてなおかつ渡らずというのは度胸がないわけです．非常に慎重な方はたくさんの条件を考えてやる．それが度胸の有無です．したがってこれは，かの有名なイギリスの経済学者パ

レートのパレート図，つまり重要項目の2割を押えれば，それで行動の8割の影響がセーブできる．それを2割で考えるか，4割で考えるかで各人が度胸を付けた根拠をパレート図で書く，ということです．これはオーバなようですが，現在の QC サークル活動で 18 歳のお嬢さんと，68歳のジイ様が一緒になって仕事して，どうやって話がわかり合うのかというと，こういったツールによるわけです．これが一例です．

落水 技術移転ですが障害がいくつかあると思います．つまり技術移転をする価値のあるよいソフトウェアを作るのは，竹内さんの分類でいきますと，下から2番めのカテゴリに入ってる方々でありまして，この人たちが本当に十分考えぬいてソフトを作った時に，技術移転の対象になるような，現実をちゃんととらえた上でしかも今まで解決できなかった問題を解決するソフトが生まれてくると思います．こういう方は何も1992年までまたなくても今もたくさんいらっしゃるわけですが，そういう方々がなぜ頑張ってソフトウェアを作らないかというと，二つの理由があると思います．一つは，そのような方々を評価する体制が日本にないということと，もう一つは何を作れば良いかということを考えるためには現実を知る必要があるわけですが，その現実を知ってる世界の方々とお付き合いするのが中々難しいということです．例えばすぐお金の話になり束縛されるとか企業秘密とかでものすごい枷をはめられるとかいうようなことです．次の時代をリードするよいプログラムが，アメリカの大学あたりではどんどん出てくる，そういう意味での技術移転はこの二つの障害をうまくとりのぞかない限り日本では望めないのではないかと思います．

木村 私は技術移転を受け入れる側の問題ということを一つだけ付け加えさせていただきます．ほかの点はお2人のお話で尽きてるように思いますので．偉い人ほど死ななきゃ直らないということがあるんですね．大先生は自分の分野が非常に大事で，それ以外の分野のことをなかなか受け入れられない．たとえば昔，新しい絶縁物がなかなか信用してもらえなくて困るというグチを，若い人から聞かされた記憶があります．そういう先生が引退されますと新しい絶縁体が使えるようになる．偉い人ほどいけないんですが，日本には偉い人が割に多いみたいでありまして，例えばこんな例があるんですね．ある職場に非常に優秀な SE さんがいたんですね．この人は IBM 系のオペレーティング・システムをいじるのが，ものすごくうまい．ところがその職場で，少しやりかたを変えよう，UNIX を入れよう，とこういう話になったところ，その人が猛然と抵抗しだしたんですね．で結局その人は UNIX を使わずにすむ職場に移ったといいます．この人は優秀な人だったんですけど，優秀であればあるだけ，自分の優秀さをじっと守ってしまって技術移転を受け入れられなかったということで，それはやはり悲しいことです．じゃあどうしたらいいか，やはり偉い人も偉くない人と同じように，何か新しいものを受け入れられるようにする必要があるんじゃないかと思うんですが，これは見方を変える，今まで自分が持っていたものをいっぺん捨てて，新しい見方でものを見るということで，いわばお風呂から一度出るようなものです．やはり一時寒いわけです．そこで具体的にどうしたらいいかですが，時々強制的に環境を変える仕組を，われわれの職場のいたるところに持ち込む必要があるんじゃないだろうか．例えばアメリカの大学では，5年くらい同じ講義をすると必ず講義の題目を強制的に変える，という話を聞いたことがあります．そんなふうなことをしなければいけないんじゃないか，せめて外部の得体の知れない講習会に行くとか，そのくらいのことはできるんじゃないかと私は思います．そこで大事なことは，そういう今まで考えてきたことをいっぺんぶち投げるための時間的余裕というものを，マンパワーの計算の中にちゃんと入れておくということじゃないでしょうか．技術移転の問題に関連してご提案しておきたいと思います．

蓄野 先ほど申しあげなかったのは，残念ながら次のような手法を，ソフトウェアの面では，あまりお使いいただいてなかったということです．例えば N7 (新 QC 7つ道具) は，ソフトウェアのために開発されたものではありません．しかし，私がこれを見ました時に，これこそソフトウェア屋，しかもソフトウェアのデザイン，そのために具合がいい，と思ったわけです．ところが私が黙っておりますうちに，日本電気さんや富士通さんは，どんどん活用し，いろんな席上で発表されているわけです．立派な事例がたくさん出ています．先ほどの学術会議会長近藤次郎先生が開発されたのが PDPC 法です．川喜多二郎先生の KJ 法というのは，親和図法という名前で出てます．それから昨日，保田さんがお話されました品質機能転開というのは，この新 QC 7つ道具を組み合わせた形のもの，ということでも理解いただけると思います．そのほかに，日本電気さんが使っていらっしゃいますのは，ソフト

ウェアの関係でU管理図を使って成果をあげておられます．また，富士通さんが使っていらっしゃるのは，直交表によって，検査の項目を効率よく構成したということを報告しておられます．多変量解析につきましては，ユーザ・ニーズの分析ということで，いろんなものがいっぱい出ているわけです．それから品質保証の面では，故障解析法というのがたくさん出ています．ずい分たくさんの事例があります．少しずつ，つまり草の根品質活動というかっこうで実績をあげているということです．どんどん皆さん方がお使いになって，まずかったらクレームをどんどんつけて改善していただきたいと思います．要するに，これから先のソフトウェアの技術者は，幅を広くしていただきたい，いうことです．「I型」というように，専門の幅が狭くて，行きづまるのは全然だめです．バツ（×）です．もう少しいいのは，専門の幅が広くて，専門もどこで終わるかちょっとはっきりしないのが「T型」です．これは三角（△）です．ではマル（○）は何かといいますと，専門が少なくとも二つ持ってる「Π（π）型」です．しかし本来，私以外の今この壇上の方々には，二重マル（◎）が付いています．どういう二重マルかといいますと，専門の幅はずっと広く，しかも専門の足は必要に応じて随時出る，というかっこうです．これを日本語で言いますとムカデ(百足)型人間，あちらの言葉で言うとマルチ人間という方々が，皆さん並んでいらっしゃるわけです．これからは，それでなければいけない．そのためにも先ほど，木村先生からお話いただきました創造工学というのは大事だと思います．その時にもやはりN7などは，お役に立つんじゃないかと思います．やはり積極的に自分で勉強することです．自分の才能というのを，製品の付加価値の中に広げていくことです．先ほどの芸術というのは正しくそれじゃないですか．Σ計画に期待しますのは，このソフトウェア・ハウスの悪い意味でのタコ部屋制度の排除ということでの質的転換です．そして，協力企業と一体化したグローバルな意味でのTQCの推進をはかっていただきたいわけです．そのためには，個性の尊重が大切です．個性の尊重というのは，はっきり申しあげますと，だめな人間は早くはずせということです．だめな人が仲間に入っている，こんなばかばかしいことはないんです．ソフトウェアではだめでも，ほかでは立派にやれる人ですから，もっと能力を発揮するほうへ行っていただけばいいんです．じゃあ最後に，どういう人をお前は推薦するかと聞かれれば，まず陽性です．ツベルクリン反応と人柄はやはり陽性なほうがいい．次に，やわらかに考える，柔軟思考です．さっきのマルチ人間のことです．そして好奇心を持って，執念深く，人に隠れてでも勉強するという，自己啓発があります．さらに，わかったことは人に教えたいという相互啓発が大切です．そして，企画力を明確に発揮することです．論点をはっきりさせ，根拠を適確にし，表現を適切に行い，実行力を確実に示すことです．有言実行です．意志決定迅速，と言いますと，これは神様じゃなきゃできないんじゃないかということかもしれませんが，これをチームとしてやり遂げればいいわけです．一人一人はアンバランスでも，チームになりますと解決できる，というのがわれわれ日本人の一番得意とするところじゃないかと思います．

伊土(NTT 情報研)　ソフトウェア工学とは何かという話で，私なりに解釈してみますと，ソフトウェアを開発する時に，できるだけ知的作業に専念できるようにするための諸技術ではないかと．例えば，私のところでもデータベースを作っていますが，アイディアを考えて，それをソフトに実現するとなると，1年間に10億円以上の金をかけている．それをよくチェックしてみると，アイディアの実現に直接かかっているのはほんのわずかで，ほとんどのコストはデバッグにかかっているんです．できればそういうことに金は払いたくない．そこで，私なりに考えますのはできるだけプログラムを作らないのがよいということです．アイディアを実現するための最低限のプログラムだけを作る．最近モジュール化や，再利用などの研究が盛んですが，それがいつ頃実現されるのか，その実現されたもののバックボーンになる技術は何かということを教えていただきたいと思います．

竹内　プログラムを作らないのが1番いいというのは，まさにそうです．作らなくても実はよくなるというのが私の話だったわけです．再利用と言うのと，部品化と言うのもちょっと違うと思います．再利用というのは，ごみ箱から拾ってくる感じですけれども，部品化というのは再利用の意志を持って作るということだと思います．その部品化をうまくやる方法として，少なくともプログラミング言語の枠組みの中では，オブジェクト指向言語が優れています．これは概念的に昔からいい，いいと言われていたんですけれども，実際私たちのところでやっているプロジェクトの中で，本当にありがたいなと最近感じ始めています．

もっと広い意味での部品化まで話がいつ頃進むかわかりませんけれども，少なくとも一つの言語の枠内での部品化は緒についてきたんじゃないかと思っています．

落水 今おっしゃった意見と全く同じで，期待はしたいと思います．ただ少し将来に対する見方が違いまして，本質的に効果のある再利用なんてできないのではないかと思います．再利用を実現するために必要な，システム作りに関する知識が何も分析されていない状況ですから．ただ言語のメカニズムを改良して，現在の状況を少し改善することはできるだろうと思っています．例えば Small talk の世界です．その辺で終わりだろうと思っています．

紫合 部品化についてはオブジェクト指向がよいという話でしたが，私はオブジェクト指向だけではまだ不十分という感じがしています．オブジェクト指向で書かれたプログラムを見ましてもシステム全体が何となくわかりにくいという感じがする．そのため，先ほどのプロセス間のインタフェースと言いますか，その辺が部品化を発展させるキーになるんじゃないかと思ってます．再利用という面では，まずソースプログラムが簡単に見えるという，そういう環境があると非常に便利です．メーカでは，非常に申し訳ないんですが，ソースプログラムを外部には出せないということが多いんですが，少なくともある範囲内で，例えば社内とかで，ソースを自由に見られる環境を作るとよいと思います．最近ネットワークや，ワークステーションの進歩で，そういう環境が構築できるようになってきました．ソースプログラムが簡単に見えれば，ちゃんとしたコメントを書いて再利用を考えて作っておかなくても，案外再利用が促進されると思います．すなわち，再利用促進のために，まず簡単にできることは，ともかく誰でもソースプログラムを簡単に見られるような環境を提供することだと思います．現にそれで効果をあげてるところが多いようです．

蕾野 私は20いくつかのプロジェクトのマネージャを経験しています．その時のことを申しあげますと，マネージャの腕が非常にものをいうと思います．お客様が理解されるというのは，二つの要素の掛け算です．こちらの説得力とあちらの納得力の積が理解力になります．この二つは独立ではなく，誰が説得力を発揮して誰に納得力を発揮してもらうかが問題です．そして再利用，つまりなるべくプログラムを作らないという真実に，少しでも近付くためには，スペックを切るということです．花田先生から昨日フリーズするというお話がありました．凍結，正しくそのとおりです．いかにタイミングよく，どのスペックをフリーズするかです．とにかくまず，ものを出してしまう．出してしまいますと，それで泡沫のニーズは消え去るのが現実です．そして本質的に残るものがある．これはやはり逐次バージョン・アップ（予防保全）でやらなければいけない．これは汎用製品につきましても，いわゆる SE（システム・エンジニア）の仕事です．信頼性工学のバスタブ曲線に対応すると，最初，デバックをして，バグがゼロにならないままで出荷するのが第Ⅰ期です．出荷してバグがちょこちょこ出てる間は，リビジョン・アップ（事後保全）をする．それが第Ⅱ期です．最終的にユーザが現状不満になり他社との比較などでクレームが上昇してくるのが第Ⅲ期です．つまり，不良率が上昇していく．その上昇していく時期がユーザで同じであれば，同じバージョンですむわけです．ばらばらになっちゃうと，ユーザごとに全部違うバージョンになる．全く同じ時期にすることはできないまでも，ばらつきを少なくできる．進んでいるお客さんには，もうちょっと待ってくださいとお願いし，ゆっくりしているお客さんには今のうちにこれを使っていただきたいとお願いする．これがシステム・エンジニアの仕事です．そして，それを包括するのがマネージャです．人間の判断は，人間が決心をしなければいけない．

木村 いつになったらプログラムを作らないですむかというお話があったんですけれども，それはいつになってもすむようにはならないだろうと思います．ただ現時点でも，プログラムを作らなければならない割合は少しずつ減ってきている．それは，確かに事実だと思います．

例えば UNIX の環境というのは，あれは部品化と再利用がし易いからはやっているのだと思います．それからもう一つ，あれにはソースプログラムが一緒にくっ付いてくるから直し易いという，先ほど紫合さんが指摘された要因もあるのですけれども，ともかく新しいものを作る時，もちろんちょっとしたプログラミングはいるんですが，既にあるものをちょっと組み合わせるための仕組が発達してるためにそのちょっとしたプログラミングがやりやすい，それは UNIX ではやっているのであって，それは部化と再品利用の重要性を示しているのだと思います．

落水 ちょっと先ほどの話を補足させてください．

再利用ができないというちょっとショッキングな言い方をしたのは，再利用ができるかできないかという議論の立て方自体が意味がないということを申しあげたかったわけでして，再利用は程度の問題です．同じ文は2度と書かないようにしたいと思ったら，それはできないわけですが，一方，ライブラリという形の再利用は計算機ができた時代から可能なわけです．ですから再利用の程度という問題設定の上で議論しなければ，意味がないと思います．私が先ほど申しあげたかったのは，絶対直さなければいけないところしか直さないとか，90% とか，そういう話になると，それは無理である．一つのシステムを作る時に，以前に作ったもので使えるものはかなりの割合で使えるというような技術は出てこないのでないかと申しあげたかったのです．

伊土 私も何年か前からモジュール化をやっていますが，数10ステップ，数100ステップまでだと，比較的他人が使えるようなものができるのですが，数キロを越えると他人が使えるようなうまいモジュール化ができない．まだ，何か再利用についての分析がたりないんじゃないかと思うんです．そういうことがいつ頃できるのかを知りたかったんです．

落水 木村先生がおっしゃった UNIX 環境というのも前提に入れての話でしょうか．あの UNIX 環境というのは，ほしいものがかなり入っているから捜すだけという雰囲気が若干あるんですが，いかがでしょうか．

伊土 UNIX も我々は使ってますが，やはりまだ不十分だと思ってます．

司会 よろしいでしょうか．こういう話はかなり規模や分野のよるところが多く，かなりケース・バイ・ケースになるわけですね．最終的にはソフト作りは，人間の最高の知的能力を要求しているわけだから，要は工学としてどの程度にやるかというトレード・オフの問題になり，したがって今の議論は尽きないような気がします．そこで話題を変えさせていただきます．先ほどから菅野先生にあおられていますが，QC や TQC はこれは絶対必要なんですよという点で誰も否定はしないと思います．しかし，QC なり TQC というのは，工学というより現場から上がってくるボトム・アップ的なアプローチのような理解を私はしてるわけです．それに対して仕様をきちんと書きましょう．設計はある方法論に基づいてやりましょう，といういわゆるフィロソフィカルなアプローチというのは，トップ・ダウン的な感じがしておりました．しかし，先ほどからの雰囲気では少しボトム・アップ的なアプローチに席巻されかけてるんじゃないかという気がしますので，落水さんか，紫合さんあたりに QC や TQC に任せておいていいのという質問をしたいんですが．

紫合 TQC は日本電気でもかなりやっています．まあ感じとしましては，何万人という人たちが何千チームも作ってやるわけですから，それなりに非常に効果があると思います．たしかにモラルもあがっているようです．ただ少し気になるのは，QC サークル活動では細かい身近かなことに尽きてしまうことが多いんですね．それはそれで非常に大切なことではありますが，もう少し長期的なこと，例えばソフトウェア工学研究会の資料を見るとか，研究会に出てみるとか，そういう活動が現場で QC 運動と合わせてもっとあってもいいのではないかと．QC サークルのグループ長ぐらいの人は，できればソフトウェア工学研究会に入って，論文も読むという態度がほしい．そうなればもう少し変わってくるのではないかと思います．

落水 鳥居先生の質問のご意図が正確につかめませんが，何を申しあげればよろしいんでしょうか．TQC をどう思うかと言われましても，ちょっと私には答えられませんが．

司会 もう少し TQC に対して，ソフトウェア工学のオーソドックスな側からの応援演説はないかということです．

落水 それは，今の開発体制からすれば，絶対必要なものであろうとは思いますが，私自身はあまり関心が….

菅野 お隣りどうしで意見が違うと具合が悪いんですが，QC サークルと TQC は違うわけです．これはご存じでそうおっしゃっているから安心して申しあげるわけですが，TQC というのは，方針管理，具体的に言いますと機能別管理と部門別管理，この二つの面でやはりトップ・ダウンですね．ただそれが具合よくいくためには，同時併行的にモチベーションがちゃんとしてなければだめです．QC サークルというのは，今武蔵工大学長の石川馨先生，それから森口先生，これはソフトウェアの関係ですが，ご両所が同じ絵を書いておられます．QC サークルとは TQC と半分オーバ・ラップしているんですね．あとの半分は，いわゆる職場から離れたものなんです．ですから，そのオーバ・ラップしている半分のところが TQC を支えてく

れる．TQC というのは，いわゆる "companywide" つまり全社的，あるいは総合的という意味です．やはりトップがしっかりしてないところはだめです．社長がその気になって，副社長がその気になって，責任者は少なくとも副社長，専務クラスが推進本部長になってない会社は全部失敗です．米国，フランス等で失敗しておりますのは，全部 QC サークルに任せておけ，というかっこうで完全に失敗しているのです．私は TQC が万能だと言ってるんじゃないんです．いろんなツールをどんどん使ってください，固有技術が非常に大事です，ということを踏まえているわけです．ずばり言いますと，機能も性能も出ない，信頼性だけあげたい，これはやっぱりできないわけです．結婚して子ども作るというのが順序なんで，子ども作ってから結婚したんじゃやっぱり具合が悪い．TQC というものをやろうぜということで，トップが考え方を決めて，それで QC サークルの育成が行われる．大変教科書的で申し訳ございませんけれども，現在日本電気さんがやっていらっしゃるものも富士通さんがやっておられるのも，また日立製作所グループのほうでやっておられます MI (management improvement) というのも，基本はやっぱりそれです．"Top down" と "Bottom up" が両々相まっており，五徳みたいなものですから，1本の足が欠けると具合が悪いと思います．

司会 Tというのはトータルということで，かなり広い意味を含んでいることがわかりました．ほかにフロアからの意見はございませんか．

森沢（日本ユニバック）「明日のソフトウェア工学」という観点でパネラの方々にお聞きしたいんですけれども，昨日からの話で出てます言葉はほとんど輸入語ばかりなものですから，少し日本的な発想法で，新しい何かが，何かおもしろそうなものが出てくるかどうか，皆さん方の見解をお聞きしたいのですが．

司会 時間が迫って参りましたので，最後に夢も含めて，その辺のことはお伺いすることにさせていただきます．もう一方フロアからのご意見を先にお伺いいたします．

坂口（NTT ソフトウェア） 最後にまた出てくるのかとも思いますが，「明日のソフトウェア工学」の展望がまだはっきりしない．例えば，10年で2倍から2.5倍ぐらい生産性があがってるという分析がありますが，本当にあがっているか，それはどういう技術であがっているかというような分析をもう少ししてほしいと思います．確かにトップ・ダウンのプログラミング，構造化設計，情報隠ぺい，ウオークスルー等という技法はあるんですけれども，私は18年ほどソフトをやっていますが，言葉が整理されただけで，昔やっていたのと今も大して変わらないわけです．新人にプログラムを作らせたとき，仕事が忙しく十分な管理ができてないままに放っておくと，新人が目茶苦茶なプログラムを作っている．規模が非常に大きいのです．私の勘からいってこんなに大きくなるはずはないので，リーダ・クラスを入れてレビュすると半分くらいになりました．管理をきちんとやって，それなりの人間が入って，レビュ等をちゃんとやれば適正規模になると思いますが，それを放ったらかしていると，できあがったものは大した機能はないのに，規模だけふえています．見かけ上，生産性はあがっているんですけれども実体のない話になっているわけですね．現在のソフトウェア開発は人間に頼らざるを得ないわけです．そうすると開発組織に参加する人たちがやる気になり，かつ無駄のないインタフェースで仕事ができるような技術だとか，あるいはモチベーション付けなどが効果が大きいと思います．そういう意味で先ほどの菅野先生の QC のお話しはよくわかりましたが，さらに何か新しい考え方，例えば心理学的な考えを取り入れていくような話を多少期待していたのですけれども．「明日のソフトウェア工学」の一つとして，その辺のお話も聞かせていただければありがたいと思います．

司会 最後になって，だんだん質問が厳しくなってきました．

明日のソフトウェア工学—まとめ—

お2人のフロアからのご質問のことも踏まえまして，それぞれパネリストの方から，まとめを兼ねましてお話ししていただき終わりたいと思います．最初から聞こうかと思ってたんですが，例えばソフトウェア・エンジニアリングはどう定義するんですかと，いうようなことをやり出しますと，ある意味で収拾がつかない可能性があるわけですね．そこで単刀直入に，「明日のソフトウェアに向って何をどうすればいいんでしょう」という点をお考えいただいたうえで，順番に締めくくっていただきたいと思います．

木村 ソフトウェアの生産性が上がってるというのは本当だと思います．20年くらい前になりますか，あるメーカの SE さんが見えまして，ちょっとしたア

プリケーション・プログラムを作ってくださることになりました．毎日々々遅くまでいらっしゃいまして，なかなかできないんです．何やってるかと言うと，アセンブラで書いてあるのを1〜2行直して，ぼんと放り込んで，ダンプを見て，うんうんうなってるんですね．最後には曲がりなりにも何かできたことはできたんですけれども，今そんなことやる人は誰もいないだろうと思うんですね．もう少しきちんとしたやり方で，せめて先に仕様書を書いて，ということになるんだろうと思います．当時と比べれば生産性があがってることは明らかだと思います．ただ，そういう無駄は，まだ形を変えて残っていると思います．そしてそういうものはまだこれから減っていくのだろうと思います．

一つむずかしいのは客先の要求というものです．お客さんの要求だからカードで仕事するとか，能率をあげなければいけないなんて言って，今だにアセンブラで書くとか，そういう話はまだまだあるようです．それが今後なくなっていけば，生産性はまだまだあがるんじゃないでしょうか．

実際自分でもはっと思う時があります．学生にこういうプログラム作ってよ，なんて言いますと，あっと言う間にできてくる．どうしたのと言ったら，LISPで書きましたなんて言うんですね．こっちはLISPなんて予期してなかった場面でです．ですから気を付けて，身の回りの細かいごみを見落さずに，自分の思い込みをだんだん削っていけば，今後もよくなると思います．

紫合 私はソフトウェアの生産性は，昔と比べて大分あがってると思っています．昔は例えばコンパイラを書くなんてのは，すごい作業だったのですが，最近は簡単なものなら学生さんの卒論ででも作れるぐらいだと思います．それは，一つはコンパイラの作り方の本ができたとか，そういうノウハウの蓄積が社会全体，場合によっては部門ごとに進められてきたこと．より具体的なのは，ライブラリなどの部品の蓄積です．昔はグラフィックスなんか大変だったのが，非常にいいグラフィック・ライブラリができてきて簡単にできるようになってきました．ウィンドウ処理なんて非常に大変だったのが簡単になってきた．実際に書いたステップ数という見方からすると，それほど生産性があがっている訳じゃないかもしれないですけれども，部品を組み合わせることの効果は非常に大きいと思います．竹内さんのおっしゃった大きい部品がどんどんたまっていくというのが期待できるのではないかと．徐徐にそういう方向に向かっていくんじゃないかと思います．

夢ということですが，昨日の講演で大野先生が，ホロニック・プロセッシングという言葉をちょっと言われましたが，その時に感じたのですが，モジュールという部品とシステムという全体が，まだ何かしっくりいっていないという気がしました．それで，ホロニックなモジュール化というような感じの新しい方法ができないかという夢が私にはあります．先ほど述べたプロセスとその間のインタフェースの概念をもう少し深く考えることで，この夢につながっていかないかなというふうに思っています．

落水 生産性に直接に寄与してきたのは言語と環境だと思います．今後もワークステーションとかネットワーク技術の発展をとりこんで放っておいても発展すると思います．私が申しあげたかったのは，要求定義や設計活動を支援する環境もつくらないと，問題と実現の世界をつなぐ部分が楽にならないということです．その素材は備いつつあると思います．例えばProlog，X-window，知識ベース技術，日本語処理等です．ユーザが日本語で表現してくる要求を分析しその意味合いを知識ベースに蓄える．設計時や保守時に問い合わせをやると必要な情報がスクリーンの前にぱっと出てくる．問題は容れ物の中身です．ただやみくもに入れても，中身がともなわなければ全然役に立たない．その中身を「観測」という手段によって組織的に集める必要があります．

竹内 ソフトウェアの話をすると私は最後にいつも開き直ってしまうんですけれども，要するにプログラムというのは，やはり言葉だから，普通の文章を書くのとほとんど同じであるというのが私の基本的な考え方なんです．たとえば小説工学というのがあったか．いかにして新聞の連載推理小説をこなすかについて工学的手法を発見して荒稼ぎをしようという小説家がいたか．いたような気もしますけれども，あんまりいなかったんじゃないかという気がします．プログラミング言語も人間の言葉と基本的には同じだから工学の俎上に本当に乗るのかという感じがしております．自然言語に，司法書士が書く登記書のような文章から，いわゆる純文学の世界に至るまで，非常に広いスペクトラムがあるように，プログラムにも給与計算システムから，いかにもかっこいいプログラムまで非常に広いスペクトラムがあります．ソフトウェア工学と言った

時に，それが全部一緒くたに入るわけがないわけです．ところで，先ほど輸入言葉がどうのこうの，アメリカではどうのこうの，という話がありましたけれども，そのことにちょっと触れておきます．言葉，小説，文学の世界には徒弟制度ふうのところがあって，そういう中でものが伝わっている．欧米のソフトウェアが進んでると日本人が思うのは，彼らが本質的にロゴスの世界の人間で，なおかつ言葉に出してコミュニケートするという習慣が強いところからきています．ドキュメンテーションが自然にできちゃうんですね．たとえば，アメリカから日本にきた男が，隣りに座ってる男にメールを送るんですね，話しかけずに．話したという証拠が残るということなんでしょうけれども，とにかくドキュメントが残ってしまうわけですね．ですからARPA ネットなんかを通じて，プログラミング言語のスペックなんかのディスカッションが，非常にオープンでなおかつ大量にドキュメントが残りつつ行われるわけです．逆に日本人の特性を生かすにはどうすればいいか．さっきから批判の対象になってるタコ部屋も呼び方を変えればアトリエでして，やはり日本人独得のコミニケーション，顔と顔を合わせて以心伝心というものがある．それがソフトウェアの分野に役立つかどうか確信は持てないんですけれども，私の経験を申しあげましょう．私の今関係しているプロジェクトのソフトウェアは，全体量で UNIX の全ソースを上回ってると思うんですけれども，その中心部分は約3人で正にタコ部屋ならぬアトリエで作りました．この輪を広げるために，さっきも出ましたテクノロジ・トランスファをやって，それにかかわる人をふやさなければいけなかったわけです．そのとき文書によるトランスファはとても暇がなくてできなかった．しょうがないので乗り込んでいったんですね．乗り込んでいってまた新たなタコ部屋的アトリエ環境を作ったわけです．うなり声一つを聞きとめて，そこはそうだと言いながら半年ぐらいでやって結構うまくいきました．いかにも日本人的やり方で，今だにドキュメントは不確かなんですけれども，ひょっとしたらこういうやり方で日本人の特質が生かせるのじゃないかなどと思っております．

菅野 タコ部屋よりも若衆宿という粋な感覚でおっしゃっていただいたら更によかったと思いますけれども，私も全面的に賛成です．確かによくなってきています．つまり，無理，無駄，むらが減ってきた．それが何でわかるかといいますと，手戻りが減ってきた．それを何で定量的に測るかというと，仕損費率が各社とも確実に減っています．なぜそれが減ってきたか？これが指針になると思いますが，やはりタコ部屋あるいは若衆宿の効果的活用により「べし・べからず」の蓄積が非常に効果があると思います．そして，「データに語らせ，データに聞く」という習慣付けが大切です．また各種のデータの層別と標準化というようにして，データがどんどん蓄積していく．これから先はどうかといいますと，創造的頭脳労働であるという本質に，増々みがきをかけて，芸術的なセンスを燃やすものは燃やさなければならない．そのためには，ワークステーションといったような装備率というところに金をかけてなければいけない．私の仕事の一つが信頼性工学ですが，信頼性工学の基本は信頼性を高める設備に情熱と金を惜しんではいけない，という鉄則があります．装備率を効果的にあげ，そしてそれを結ぶ，ワークステーションによる LAN というお話がありましたけれども，問題はそれを上手に使うような人間の教育を何とかしなければいけないと思います．文章作法には，『プログラム書法』，『プログラム作法』という日本語での大変な名著がございますけれども，そういうところから教えられることは非常に多いわけです．やはり短文の積み重ねということが先ほど出ました．あれは日本語でも同じですね．私どもも，ドキュメントを書きます時に，400字詰原稿用紙の1行半に一つはコンマを入れ，3行に一つはピリオドを入れろ，ということをよく言ってるわけですが，そういう文章作法がちゃんとできれば，マニアルももっとよくなるだろうと思います．まあ，新人類の将来に大いに期待したいと思います．

司会 どうもありがとうございました．今日と明日のソフトウェアの工学の違いと，一見ディスクリートに感じるんですが，これは時間的にそんなディスクリートなはずがないわけでして，継続的に改良中だというような理解で今後も見守っていかなくちゃいけないんだろうと思います．お忙しい中，きょうはパネリストの方々にお集まりいただきまして，本当にありがとうございました．それから学会と研究会のほうでパネルのセットをしていただいたにもかかわらず，どの程度皆さん方に満足していただけたかわかりませんけれども，これを糧に今後とも議論の場を広げていっていただきたいと思います．フロアからも多くの方々が積極的に発言していただき珍しく盛りあがったと思います．どうも皆さま方ありがとうございました．