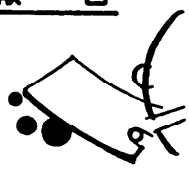
報　告



パネル討論会

CASE 環境の夢物語—革新的将来像—

「CASE 環境」シンポジウム† 報告

パネリスト
阿草　清滋[1]，　落水浩一郎[2]，　玉井　哲雄[3]
中田　修二[4]
司会　花田　収悦[5]

1. はじめに

司会(花田)　時間になりましたので今回のシンポジウムの最後の締めくくりとしまして，パネル討論を行いたいと思います．私は司会をいたしますNTT ソフトウェア研究所の花田でございます．

それでは簡単にこのパネル討論のやり方についてご紹介したいと思います．

「CASE 環境の夢物語」ということでパネリストの先生方にはかなり自由にお話していただきます．何を話すかを書きとめますと，論理があいまいだとかということで，多分皆さん少し遠慮されるんじゃないかと心配です．だから証拠が残らないように，事前に予稿集には載せておりません．

しかし，夢物語といいましても架空のものではしょうがありませんので，その辺は各先生方にお任せします．私ぐらいの年になりますと，夢を見るのさえ難しい．特に皆さん若い方々が多いわけですから存分に夢を見ていただきたいと思うんですが．まず4人のパネリストの方に私なら近い将来こうなるだろうという夢を見ていただきます．自分の夢を見るのは自分の責任なわけです．ですからこのパネルの最終的なターゲットは，皆さんが皆さんなりの夢をつくっていただくことです．そのケーススタディといいますかサンプルとして4人の先生方にパネリストになっていただきまして日ごろの夢を語ってもらう．

夢を見る順番を一応決めておりまして，京都大学の阿草先生には主として要求分析，定義技術の観点から，静岡大学の落水先生には開発の方法論あたりから，シグマ本部，ついこの間までは日本電気で活躍されていたんですが，中田さんにはシグマという環境も含めまして開発環境とかソフト流通のあたりから，最後三菱総研の玉井さんには，ご専門であります AI 技術の応用という観点からどういう夢を見るかお話いただきたいと思います．

各パネリストの簡単な経歴は，発表される直前に紹介したいと思います．だいたい1人15分から20分ぐらい，その後会場の皆さんと30分間ぐらい討論をいたしまして，できたらまとめたいなと思っておりますのでよろしくお願いします．

それでは，早速この順番でパネルを始めたいと思います．最初のパネリストは，京都大学の阿草先生でございます．昭和47年に工学部の修士課程を卒業されまして，ずっと京都大学にお勤めになっております．興味をもっているのはアップストリームといいますか，上流工程です．ずっと要求仕様の形式的処理の研究を続けておられます．先生の主張は非常にすっきりしておりまして，ソフトウェアの本質はカスタム化あるいは最適化であるととらえまして，その辺に的を当てましていろいろ取り組んでいる状況でございます．きょうは先ほど申しあげましたように，要求分析あるいは定義のあたりを中心に楽しい夢を語っていただきたいと思います．

それではよろしくお願いいたします．

† 日時　平成元年3月3日（金）15：00～17：00
場所　機械振興会館大ホール（地下2階）
1) 京大(現名大)，2) 静岡大，3) 三菱総研(現筑波大)，4) Σ本部（現日電），5) NTT

2. 上流工程

阿草 京都大学の阿草です．アイウエオ順というと最初に出されますし，アップストリームで上流工程をやっているといって最初に出されます．あとからしゃべるほうがしゃべりやすいときや，先のほうがしゃべりやすいときなどいろいろあるんですが．特にきょうはあとで落水さんが夢を話されるんで，僕は堅い話をすればいいんじゃないかなという感じで，最初のところは夢というほどではないお話になるかもしれません．

CASE ということで話せといわれたんですが，2年ほど前，ここでプロトタイプの話をしたことがあります．そのときも玉井さんと同じだったんですが，タイトルが変わっても同じ人が出てくるのはおかしいなと思いながら CASE ということで話させてもらいます．

CASE というのは DFD とかミニスペックであるとか，そういう今までの上流工程での分析ツールのコンピュータ化であるという定義を本で読んだことがありました．ですから自分のことを話しさえすれば CASE に関連するのかなと考えますが，必ずしも CASE という感じじゃないかも分かりません．とにかく，CASE というのはかなり要求仕様に近いところをやるのが定義かなという感じがしております．

ところが，CASE と対応されるものに CAD があると思いますが，僕はコンピュータ・エディッド・デザインじゃなくてドラフティングだという感じがしています．建築とか回路とかはドラフティングに非常に時間がかかっているので，そういうものを支援するものがある．同じように，DFD も絵をかくときの最初のツールとしては非常に便利になるんじゃないか．どなたかドキュメンテーションが大事だといわれましたが，ドキュメンテーションが煩雑なときに，試行錯誤的にいろんな絵をかき直したり，ドキュメントのフォーマットをそろえたりするときに，CASE が役に立つんではないかと．

その次に，じゃドキュメントはなぜ必要かというと，自由に考えなさい，自由な絵をかきなさいといっても，何をやっていいか，なかなか分からないときに，記述方法が決まれば思考がコントロールされたりリードされる，そういう意味で CASE は重要なんではないか．

DFD では細かなところよりも大まかなとらえ方を大事にしますが，結局最初のシステムはビジュアル化を目的とするんではないかと思います．シグマのプロジェクトの中でも要求仕様はビジュアル化が一番重要であるということで，そういう研究というか検討にも参加させていただいております．

要求仕様がどういうことかを，もう一つのとらえ方から考えてみます．そもそもソフトウェアはどういうものかを考えないと，ソフトウェア開発と要求仕様との関係が分からない．一般的にソフトウェアというのは，システム動作をいかに記述するか，いわゆるプログラム的な立場としてのオペレーションのセマンティックスをどう与えるかという方法である．もっと広く，システム運用のノウハウ，たとえばプラントの制御の運転方式とか，そういうことまで含めることもあります．

ソフトウェアには何か固定部がある．これはアプリケーションプログラムを買ってきても同じで，固定部はやはり一種のハードウェアなんです．ハードウェアとソフトを分けたのは，ある固定のサービスを提供するときに，ハードウェアでインプリメントしてもいいしソフトウェアでインプリメントしてもいい．たとえばアプリケーションのパッケージ1個買ってきても，自分のところに必ずしもうまく合わない．その部分をインタフェースするのがソフトである．ということは，ソフトウェアの作成というのは結局は自分の環境をいかに定義するか，いわゆる仕様化をどうするかということです．そもそもソフトウェアの作成は自分の環境をいかにとらえるかということであり，そんなものは自動的にもらえっこない．やはり仕様化の過程，すなわちソフトウェア開発の過程は最後まで残るんではないかという感じがしております．

夢物語として，ソフトウェアの要求分析がどういう具合にサポートされるかを考えます．CASE でサポートされるときには要求の分析と記述という，記述の過程の中に分析が入ったり，分析の過程の中に結果としての記述がどんどん入ってきたりします．たとえばプログラムも設計とプログラミングの区別はなかなかしにくいのと同様です．CASE の対象はどちらにあるかを考えると，今までの CASE でやっているのは結局開発経験のあるシステムをいかに効率よく作るかということの支援です．これは知識の再利用の支援

です．それから，既存システムに類似したシステムの作成．これも再利用といいますけれども，そのためにはどれが類似しているかをいかに見つけてくるかが問題になるわけです．ここらまではなんか CASE でサポートできるかも分からない．

ところがその次から非常に問題になる．人間には，なんとなくあの人が作りたいのはこんなシステムであるというのは分かる．でも類似のシステムはないと，そういうものをいかに分析するかもよく分からない．もう一つは，あの人やっぱりちょっとおかしいんじゃないか，というシステムを作りたいと言う人がおります．しかし，夢物語というのは夢を見ている人は楽しいかも分かりませんが，ほかの人は全然分からない．そういうシステムをいかに分析するかということは，できるとは思えないわけです．

サポートは可能なものとしては，たとえば要求仕様化における定型的な支援，資料の収集であるとか，まとめであるとか，連絡事項，清書，だいたいこういうものができるんではないか．そうすると，事務管理的な仕事という意味でのアップストリームのサポートは非常に意味がありますが，必ずしもそれはクリエイティブな仕事ではないが，コンピュータでサポートできるものです．事務管理的な仕事をわれわれの生活の中でみてみると，有能な秘書がいればいい．有能な秘書がいればいいというのはどういうことかといいますと，キーポイントを押さえて，常識をもっていて，いつものやり方をもっている．それとあとは学習機能，適応能力というものです．

一般的にソフトウェアの問題は，解決の仕方を書くことと問題そのものを書くことは，複雑さの程度は変わらない．問題が何かをしっかり書こうと思うと，解を与えたほうが早い．解を与えるということで結局問題を定義している．定義さえきっちりしてくれればプログラムはちゃんとつくれますというのは何かおかしいが，そういう場面がたくさんあります．

有能な秘書なら，どこどこにあることでこうしてくれというキーポイントを言えば，そのほかのことは全部やってくれる．キーポイントをいかにうまく押さえ，常識をもち，いつものやり方を学習を通じてとらえていくような CASE をつくれといわないと，要求仕様化はなかなか支援が難しいんではないかと思います．

これは IPA のところでの話で，そのときもまたアップストリームのところで議論させられたんですが，結局要求分析のときに何をつくればいいか，夢物語として何を入れればいいかというときに，結論としてはこういうことだったと思います．

要求の分析というのは結局はコミュニケーションとインタビュ，相手の気持ちを引き出すための技術である．そういうのは人間の世界ではコーディネータといわれていて，そのようなコーディネータのロボットをつくればいいんだ．コミュニケーションとかインタビュ技術が得意なロボットをつくればいい．それが一つの夢物語．

それから，アイデアを表現するとき，自分自身の思考がリジットに決まっているような思考を表現するときはかなり楽なんですが，新しいシステムをとらえるときには，自分自身の概念すら本当はもっていないことがある．すなわち自分自身がどういうものをつくりたいかが記述の中で変わってしまう．今はワープロですから簡単に文字が直せますが，昔手書きで原稿用紙を埋めてるときは，字を書き間違えると，書き間違えた字のままに文章がつながるように文章自身を考える．ですから，記述をする対象と記述のもとはダイナミックに変わる．

そういう場合には表現してみないと分からない．自分は何を考えていたんだろうということを表現したあとで初めてフィードバックがかかる．これも IPA のときの結論なんですが，結局紙とはさみをコンピュータの中につくればいいということになった．切ったりはったりコピーしたりして，われわれがアイデアを求めるのと同じスピードで処理ができるようなシステムを作ればいい．これは今ゼロックスでもやっていると思いますが，大規模な電子会議システムみたいなもので，みんながフリーにディスカッションしている間にドキュメントが自動的に組み上がっていくようなアドバンスな電子会議システムが作られればいいんじゃないかと思っています．

アメリカでこの前，幾つかの電子会議システムを見せてもらったんですが，かなりアメリカ的で，「はい，次の方どうぞ」とかいうトークを回していろんなドキュメントを組み上げていくとか，思想をまとめていくものです．日本ではもう少し，まあまあ，なあなあのことがうまくできるような電子会議システムができればいいのじゃないかと思っています．

記号論の話のときに，われわれがコミュニケーションできるのは，シンボルとそのシンボルに対するリーガルなオペレーションとしてコミュニケーションがで

きる．すなわち単語自身が理解されるというか，文の中で正しい単語の使われ方ということで理解されるとしますと，定義するときに前もって記号と意味，適合操作が定義できるのか，その文の中で定義されるのか．すなわちソフトウェアを定義する大もとの定義語が存在してソフトウェアが定義できるのか，あるソフトウェア対象を定義する中でその意味が決まってくるのかということが非常に難しい問題です．

あるアプリケーション・ドメインを決めれば，すなわち環境を決めればソフトが決まる．記号を決めると，そこに対するオペレーションが決まるけれども，もともとソフトウェアはそういうものであったかと．すなわちハードウェアがフィックスされていて環境に対してインタフェースするのがソフトである．一番最初に示したような定義にすると，環境に独立になっていないといけないという意味では，システムの動作の中で意味を定義しないといけない．そういうのはなかなか難しい問題なんです．

環境ごとに自動的に言語が決まってくる．すなわちその言語を決めるような体系がプログラムされていて自分たちがその中でコミュニケーションしていくうちに意味がはっきり決まってくる．そういうものを決めていかないといけないんじゃないか．

ところがその言語が与えられると，また言語を使って何か書くのはソフトウェアであって，そのソフトウェアはどこまで決めたら自動的にうまくいくかが非常に難しくて，要求仕様をうまく定義することイコールソフトウェアではないかと考えています．

ソフトウェアというのはでき上がってしまえばすでにハードウェア，すなわちフィックスされた機能である．それをある人がある環境のもとに自由に使えるためには，完全にカスタムマイズ可能なソフトウェアでないといけない．そうすると，完全にというのはあまり行きすぎると，ハードウェアだけ用意しましたから勝手にソフトをつくってくださいとなり，こう行きすぎるとだめなわけです．

そういう意味でどこまでプリミティブを落とすかはいろいろ問題があると思いますが，ソフトなソフトウェアをつくるべきで，われわれがやっていることはかなりハードなソフトをつくっているんじゃないか．こういう場面で使えるソフトですというのは，環境をバシッと決めて，その環境がちょっとでも違うと使えないようなソフトをつくっているんじゃないか．たとえば，知識システムでもエキスパートシステムでも，ある分野のある知識においてはうまい知識推論がなされるけど，それ以外では非常に危ないんじゃないかというのをつくっている．われわれソフトウェア屋さんがやらないといけないことは，非常にソフトなソフトウェアをやらないといけないんじゃないか．そういうことを今考えています．

日本語というか，できるだけナチュラルな文を入れたとき，そのナチュラルな文がどのように解釈されるかを計算機に出そうとしています．計算機を一つの標準の解釈系として，その文章をどう理解したかをビジュアライズすることで，ソフトウェアの要求分析をしようということを，シグマの上流工程のツールを考えてくれという依頼に対して，先進的ツールということも考慮して開発しているところです．

そういう先進的なツールをやるときでも，大学関係でわれわれの夢物語をやっていることとは別に，実際にソフトをつくってみないとそのアイデアが分からない．さっき言ったように，われわれは表現を通じて初めて自分たちの思考過程がうまく決まるんではないかと，現在そういうことを考えています．

以上です．

司会 ありがとうございました．後ほどいろいろ議論をしていただくことにしまして，続いて落水先生からどちらかというと開発方法論の立場からお願いします．

落水先生は，昭和49年に大阪大学の基礎工学研究科の博士課程を修了されまして，そのあと静岡大学に移って研究を続けられております．専門は，ご存じのように「開発方法論」とか「開発支援環境」ということで，きょうはその辺のご専門家の立場から夢を見ていただくようにお願いいたします．

3. 開発支援環境

落水 落水でございます．まず過去の夢を整理いたします．最初に，システムに変更がしやすくテストがしやすい構造をつくりこむ設計方法論開発の夢がありました．次に，ユーザ要求の表現手段を開発したり，大人数をうまく管理しつつ品質や生産性を上げていくプロジェクト管理の夢．最近は，アップストリームにおける動作解析を可能にするという夢があります．

これらの夢は一部実現されましたがまだ完全ではあ

りません．その積み残しを含めながら，最近，つくる人間も使う人間も素人が増えて困るなんとかしてくれという現実が世の中にあります．

これらの問題に対する現在の夢を4つ程あげてみましょう．CASE ツールはその一つです．しかし現状の CASE ツールはシステムの論理モデルを書くだけで，パフォーマンスの解析や，製造段階へのマッピングが十分ではありません．二つめは，プロセスモデルと統合環境です．これは，聞き方によっては，機械に主導されてプロジェクトが動き，プロダクトがどんどん出てくるような誤解をうけやすく，その意義効果を今後明確にしていく必要があります．

三つめは，ソフトウェア開発に関する人間要因をプロトコル解析によって明らかにすること．

最後は，知識工学の成果を応用して人間の知性を増巾させる機構を環境に組み込もうというものです．

夢にもいろいろ立場がありまして，研究の最前線，技術の最前線，最前線から遅れたところでえがく夢はそれぞれ違うのですが，きょうは，どうせ夢を見るんでしたら研究の最前線の夢を語りたいと思います．

以上の4つの夢を要素分解してみますと，CASE ツールをつくるための道具要素，つかう立場としての管理者や開発者の行動特性，そういうものを組み合わせてソフトウェア開発を進めるときのプロジェクト構成法や，ツールの統合法等の一つの方向がうかびあがってきます．これに応えようという動きがプロセスモデルに関する研究です．

その意味で，次の時代のソフトウェア開発を特徴づけるキーワードはプロセス記述言語です．アップストリームに仕様記述言語があって，設計時の動作解析を助ける．下流にプログラミング言語があって，アプリケーション分野に応じた言語を使い分ける．そういう言語回りにツールが張りついている．そのようなツールを使う人間がもっている開発や管理に関するコンセプト．そこら辺をうまく書きとめて集団の動きを調整したり，個人の動きを助けたりすることがプロセス記述言語にかかわる夢です．

プロセス記述言語という，新しいキーワードの背景にある考え方は次のとおりです．まずワークステーション，ネットワーク，ディスクサーバというマシン群と，その上のツール群がある．それを使って仕事をする開発者群がいる．その開発者群に対して指示をしたり結果を評価したりする管理者群がいる．というと，なにか非常に意地の悪い人みたいですから，全体のプロジェクトの達成と成功に使命感を燃やしている方々がいらっしゃる．

そうしますと，開発者群の活動と管理者の間のポリシをつなぐプロセスモデルが必要になります．プロジェクト管理者はリスク解析，リスク回避であるとか品質保証などの目標を持っています．一方開発者群は，与えられたスペックに従って，迅速，正確かつ楽に仕事を進め，自分の楽しむ時間をなるべくたくさんつくりたいという希望をもっています．プロセスモデルに対する明示的な記述手段を与えるのがプロセス記述言語です．

プロセスモデルやプロセス記述言語に関する研究は開発パラダイムの研究グループとプログラミング環境の研究グループが合流して，進められています．

開発方法論とプロセスモデルの関係は以下のとおりです．開発方法論はソフトウェアプロダクト，中間生成物をつくり出していくための一定の枠組です．従来の開発方法論はつくる人間にとっても，管理する人間にとってもかゆいところに手が届かないという問題がありました．中間生成物をつくり出す個人や組織の問題解決のプロセスがどのように行われているか，そこら辺に焦点を当て明示的に書きとめることにより，熟練者と初心者のスキルレベルの差の吸収，組織全体の構成法，あるプロジェクトが動いていくときの活動を円滑にサポートする支援環境の構築原理等を明らかにしようという機運が生まれてきました．ある組織がある開発方法論を採用して，あるツール群を利用しつつ，こういう人員構成でプロジェクトをスタートする．このほうがより全体的な見通しが立つということでソフトウェアプロセスモデルという概念が生まれました．

さてソフトウェアプロセスモデルの研究を進めるにあたって，今の設計方法論とかツールでは助けることのできない，ソフトウェア開発に関わるチームの活動面を三つほどあげてみましょう．

一つは，阿草先生のお話でも出てきましたが，チーム内交信の支援です．プロジェクトリーダがいてサブシステムの担当者がいる．その下にプログラマがぶらさがっているというワークブレークダウンストラクチャを考えていただきます．このようなハイアラーキカルな組織を組んでいるときには，命令を正しく伝達して，何をやって欲しいかを相手にきちんと委任して仕事をやってもらう，そういう種類のコミュニケーションをサポートしなければなりません．また一番最初にシステム構想を作りあげるところで，経験者とかアイ

デアをもっている人がお互いの経験を突き合わせながら，アプリケーションドメインや開発経験の知識を統合してシステムコンセプトを定めるという段階にもコミュニケーションのサポートが必要になります．

チーム内交信の機構は各自の作業がばらばらにならないように，緩い規約の下に仕事を進めていくというチーム内コーディネーション（調整）の機構とあわせて協調支援のために今後開発されるべき重要な技術です．

このような機構は，開発チームやその活動が時間分散，空間分散して存在しているという事実を考慮に入れて構築されるべきです．オブジェクトベースやネットワーク技術の発展がポイントになります．

2番目の問題はソフトウェア開発活動を構成する要素の再吟味とお互いの関連づけです．たとえば，仕様書とか，ソースとかの情報を考えますと，それを読む，書く，まとめるというオペレーションがあります．ある一つのデータに対する一群のオペレーションのまとまりを一つのアクティビティと呼ぶことにしますと，これは問題解決活動の一つの基本になります．

そのようなアクティビィテイが複数個あるとき，その組み合わせ方，だれに仕事をアサインするか，仕事の順番をどのように並べるか，それから，あるアクティビティにおいてつくり出されたものの品質をどうやって評価するか，等の活動を考える必要があります．

組織レベル，チームレベル，個人レベルにおける諸活動を支援しつつ全体をまとめあげるには，ソフトウェア開発活動の再吟味が必要です．

このような諸要素を最後にまとめあげるものが開発パラダイムです．世の中で代表的なパラダイムが二つあります．

まずお客さんのニーズをソースに変換してお客さんに返すというタイプ．

もう一つは，ニーズの顕在化しない形のソフトウェア開発．シーズをつくるといいますか．今つくればどれだけ売れるかという評価基準がポイントになるタイプのソフトウェア開発です．プロセスモデルに基づく統合環境がカスタマイズされスペシファイされて，それぞれの開発タイプを助けることになればいいなというのが私の夢です．

最後に，オブジェクトサーバの問題をとりあげたいと思います．ソフトウェア開発環境に適合するオブジェクト管理システムの開発は重要です．

従来のデータベース技術は，社内で管理すべきデータに対するコンセンサスをつくり，概念スキーマとして表現するという形で発展してきました．

一方，最近出現したハイパテキストの技術は，そのようなコンセンサスなしで必要なら入れていこうという接近法です．ソフトウェア開発を支援するオブジェクトサーバはその両者の機能を必要とします．

プロセスモデルやプロセスプログラムの話を最初にICSE で聞いたときには，シェルスクリプトやメーク，または，単なる管理の方針じゃないかという印象だったのですが，その後，いろいろ考えているうちに本日お話させていただいたような重要な新しい技術であると判断するにいたりました．

司会 どうもありがとうございました．続きまして中田先生にお願いしたいと思うんですが，中田さんは47年に日本電気に入られまして，データベース管理システムを最初手がけられたようです．その後55年，56年あたりにミシガン大学の ISDOS のプロジェクトに参画されまして，そのときに要求定義技術を研究されました．帰国されましてからは，今回の基調講演で藤野さんが話されました SEA/I のオフィスプロセッサ用のものの開発に従事されまして，昨年の9月からシグマ本部に出向しているような感じになっているということです．

それではよろしくお願いします．

4. 開発環境・ソフト流通

中田 中田でございます．

「開発環境・ソフト流通」ということでお題をいただきまして，夢を話させていただくということでございますが，革新的な夢を見るのは非常に難しいことを今回体験いたしました．Riddle 先生のお言葉といいますか，論文の中に「情報工学関係で1人の人が夢を見てからそれが世の中に非常に普及して使われるまで，だいたい平均20年ぐらいかかる」というお話を読んだことがございまして，そういう意味合いでいいますと，CASE というものはまだまだ夢の最初のところかなと思っております．世の中の認識がだんだん広がってきている．だけどまだまだ世の中の普段の生活で普及して使っているという状況ではない．

そうしますと，今の CASE は非常に初歩的なものですから，過去の夢をベースにつくられているんじゃないかと思っております．いうならば，既存技術，

ツール，ハードウェア，方法論という過去の材料を非常にうまく組み合わせてつくっている段階で，そういう意味では非常にエボルーショナルといいますか，まだ進化型の段階で非常にレボルーショナルなものをどう夢見ればいいのか，現在悩んでおります．

じゃどういうふうに夢を見ようかということで，まず CASE のユーザさん，CASE を提供しているベンダさん，こういう人たちはいったいどんな夢を見たいのかというところを考えてみました．

独断と偏見で考えてみますと，CASE のユーザさんは，よい開発環境を利用してソフトウェア開発に効果を上げたい．ここでよいというときにいろいろあると思うんですが，第一にいろいろなツールなりハードウェアの中から一番自分に合ったものを選択できる，最善のものを選べる環境というのが一つあると思います．第二には，それを自由に組み合わせて利用したいということがあると思います．

そういうことになりますと，市場といいますか，世の中にいろんないいツールがいっぱい出ていて，その中から自由に選んで自由に組み合わせて，自分に最適なものが環境として組み上げられる，そういうことがユーザの夢であろうかと思います．第三には，ソフト流通ということになりますと，自分と他の部門といいますか，他所にあるような部品を自由に使って開発ができる．仕様とか，部品の再利用とか，部品の共通性．たとえば，一つの部品を使って一つの目的環境向けのソフトしかできないというんではなくて，いろいろな目的環境向けのソフトができるような部品も欲しいという夢があろうかと思います．

それと似たことですけど，1回つくったソフトがいろんなところの実行環境でも使いたいということで，つくったソフトのポータビリティも無限にあって欲しいという夢もあろうかと思います．

まとめてみますと，どの程度の将来か分かりませんが，ユーザさんがだんだんレベルアップし，CASE も普及してきますと，ノウハウとか，知識とか，利用技術もだんだん充実してきますんで，要求も高度になる．そうすると，特定のベンダあるいは特定の製品群の中だけでそれを満足するのは大分難しくなるんではないかと思います．

最初に「最善のものを選択する」ということを申し上げましたが，いろいろなものが提供されている中で，お互いに性質の違うツールなりハードウェアもあるかもしれません．それらの中からユーザさんが一番自分に合ったものを選んで，ばらばらなものから自分にとっては統一的な開発環境を構築したいと，こういうちょっと難しい夢になってくるんじゃないかと思います．そのためには，オープンで非常に柔軟な環境の機構が欲しいと思っているんじゃないか．そういうのがユーザさんの一つの夢かなと思っております．

一方，そういうものを提供していきますベンダさんのほうを考えてみますと，ベンダさんにはハードウェアとか OS の提供者だとか，ツールの供給者さんがあると思うんですが，特にツールの供給者という立場でみますと，世の中で受け入れられるいろんないいツールが先ほど申しましたような環境の上で提供できるものであって，それによって CASE ツールを受け入れる人々もどんどん広がるのであれば，共通の環境を大いに歓迎したいということはあると思います．

共通の環境というのは，ハードウェア的にみますと，世の中で一番普及しているハードウェア上にツールなりを供給したいというのが一番身近な例かと思います．もう少し共通という点を長期的に考えてみますと，開発環境を標準化して，世の中が標準的な開発環境を認知するような状況になって，その上にツールをつくって，ツールが世の中に受け入れられるということであればいいということもあろうと思います．

ただし，そういう共通のことばかりやってて疎外されては困ることが一つあります．自分の供給するツールを製品差別化したいとか，いいところをいっぱい盛り込みたい，それを疎外するような共通環境じゃ困るかというところもあろうかと思います．

ちょっと独断と偏見的な夢を見てしまったかもしれませんけれども，ユーザさんなり CASE ベンダさんなりは，将来的に最適な環境を構築するためには，いろんなものから自分に合ったものを組み上げられるということが一つのポイントかなと思っております．そういう意味合いでの CASE 開発環境を次に話してみたいと思います．

「いろんな構成要素を使って，それらを柔軟に組み合わせて一定の統合化した環境をつくりたい」という点についてみてみます．現在，いろいろなツールが提供されています．それらがライフサイクルをどの程度カバーしているかということと，それから，いろいろなツールを自由に組み合わせて使える度合（ある CASE ベンダさんの提供する製品群にサードパーティのソフトなどを混ぜ合わせて利用できる度合）を考えてみます．まず，現状では，特定の工程なり特定の

目的に沿って一品一品ツールを提供なさっているベンダがあると思います．それから，メーンフレーマさんや大手のソフトハウスなんかで多いと思われますが，一貫したライフサイクルの支援システムを提供しているけれども，それは proprietary といいますか，一定の自分が前提としている環境の上での道具立てであるということがあろうかと思います．

CASE ユーザさんのレベルがどんどん上がるといいますが，要求がどんどん上がってきますと，先ほど申しあげましたように，最適な環境をつくるためにはいろいろなものを自由に組み合わせて，自分に合った環境をつくりたいということになろうかと思いますから，そのためにはライフサイクルを全部カバーしながらいろいろなものを混ぜ合わせて使うという意味の統合 CASE が一つの夢かなと思います．そのために個別ツールをうまく統合化する．そういう何か機構といいますか，ファシリティといいますか，そういうものも必要かなと．それから，特定の環境で一貫している proprietary なものに対してはもっと柔軟にいろんな環境で提供されたものが使えないかという意味合いの柔軟化も必要と思います．

異種のいろんな構成要素を使いながら，ユーザさんにとっては開発環境が一貫していて，ツールを利用する場合とか，ツールでつくった情報を管理する場合とか，開発管理，プロジェクト管理についても，いろんな要素を使いながらも一貫した環境が組める．そういうためには何か共通した基盤といいますか，インフラストラクチャ的なものが必要になってくると思います．

その辺について技術動向的にみますとまず個別ツールの間で，データ交換ができるようにしようという動きがなされています．そういうのをデータ交換のブリッジと呼んでいます．

それから，あらかじめそういうツールをいろいろ提供なさっているベンダさんがお話し合いなさって，一つの共通のデータ表現といいますか，データ交換の形式を話し合って，それに沿ってみんなが参加できるようにしましょうというやり方も進んでいると思います．

それから，単一の環境で統合的なツールをいろいろと提供なさっている方は，このツールがもう少し外の環境にも適合できるようにということは必要かと思います．

最終的には，共通的な開発基盤環境を想定して，その上だったらAさん，Bさん，Cさん，Dさんのいろんなツールが統合化されて，そして自由に一貫して使えると，そういう夢が開発環境の中であるんじゃないかなと思っております．

そういう話はどの程度難しい話かといいますと，実は夢は難しい話になるんじゃないかなと思いますが，そういうインフラみたいな上でいろんなツールを持ってくるための技術課題としては，第一にライフサイクル全体に関していろいろなマルチベンダさんのツールを継ぎ目なしに，シームレスネスといいますか，つないで使うことがあります．第二に，ユーザインタフェースも利用者からみて一貫したものにすることがあります．そのためにはなんらかの統合化とか標準化が必要になってきます．いろいろなレベルで必要になると思います．ユーザインタフェースとかドキュメンテーションといったものも，いろんなツールを持ってきても一貫してないと使いづらい．

第三に，ツールとこの基盤層とのインタフェース，ツールを埋め込むやり方も何か標準的なものがないとやりにくい．第四に，先ほど落水先生からありましたプロセスについても，個々のツールを使ってうまくやるための一貫したやり方が提供されてほしい．第五に，構成管理とか，作業フロー管理とか，プロジェクト管理でも，いろんなツールを持ってきているんだけれども一貫して環境としては構築したい．第六に，ハードウェア的あるいは OS 的なプラットホームのレベルでは，お互いにいろんなものを使っても，コンパティビリティ，相互接続性が欲しい．そして，第七にはこういうもの全体を使ってつくった設計情報といいますか，そういったものに関しては，情報の表現とか内容とか相互関連といったものがある程度標準化されてほしい．

こういう難しい話があろうかと思います．そのためには，夢を実現するためには，非常に広範な標準化活動がこういった目的のために焦点が合って，協調されて，少しは打開の方向にいくことが必要なんじゃないかなと思っております．

実際にそういう動きは少しあって，CASE のスタンダードといったものが幾つかの団体のところで議論されていると聞いております．

今開発環境のほうばかりいってたんで，ソフト流通のほうにも夢を見ようとしたんですが実は，なかなかうまい夢が見れませんで，ちょっとお話したいと思います．

今のソフト流通の中味をツールの流通と部品の再利用の流通に分けて考えてみました．冒頭阿草先生から，ツールに関して設計ツールか，清書ツールかというお話があったんですが，だいたい9対1ぐらいで今世の中に出ていってるのは清書ツールだと聞いております．

そういう意味ですと，流通ツールといっても流通するためには使い方が分かっていないと非常に難しいわけで，今後方法論の普及が前提となるようなツールがどの程度流通するかなというのが，非常に個人的には興味があります．

部品はさらに難しくなると思います．部品の再利用や流通を考えたときには，部品の共通性だとか，それを検索したり管理するのが非常に難しい．今のレベル，世の中に普及しているツールではなかなか解決できなくて，もっともっと人間側のインテリジェンシが必要で，なかなかうまくいかないのが現状かと思います．そういう意味では，ちょっと逃げてるみたいですけれども，標準化や共通基盤やツールのみでは部品の再利用や流通はちょっと限界があるかなという印象をもっています．

ちょっと変な夢を見てしまったかもしれませんが，「開発環境と流通」ということで，以上プレゼンテーションさせていただきました．ありがとうございました．

司会 どうもありがとうございました．

それでは最後に，玉井先生から一応AIの関連という形でお話してほしいということをお願いしてあります．玉井さんは45年に東京大学の工学部を卒業されまして，修士まで進みました．47年に(株)三菱総合研究所に入られ，現在に至っているわけです．よろしくお願いいたします．

5. AIの適用

玉井 玉井でございます．

私は「AIの適用」という立場からということで，夢を語るには一番やりやすい立場です．つまり，今AIでやろうとしていること，あるいは世の中でやられていること自身が夢みたいなもので，というのは先ほどの落水先生じゃありませんけれども，フィージビリティがないという意味での夢なのかもしれませんが，要するに現状は夢を追いかけてる．うまくいっている部分は少ないわけですから，現状を話せばそれが夢物語になってしまうということで，非常に楽なわけです．

そういう意味で，AIベースのソフトウェア開発支援システム，そういう意味での自動プログラミングを振り返ってみたい，あるいは分類してみたいということなんです．

これを三つのタイプに分けてみました．最初が形式的アプローチ，2番目が知識的アプローチ，3番目が実践的アプローチ．これは別にきちんと定義するつもりはなくて，キーワードを並べてだいたい想像していただくということなんですが，形式的アプローチというのは，たとえば演繹推論とか，プログラム変換法とか，帰納推論とか，そういうものを利用している，あるいは実行可能仕様という立場で研究しているのもある．こんなようなのを想定しているわけです．

2番目は知識的，いわゆる知識工学あるいはエキスパートシステムを作る立場のアプローチで，キーワードとしてあげたのは，ソフトウェアエンジニアリング知識ベースとか，プログラム理解システムだとか，自然言語のインタフェースとか，この三つはかなり側面が違うかもしれませんが，この辺のキーワードから想像していただく．

三つ目の実践的というのは，たとえば第四世代言語とか，部品から合成するとか，問題向き言語とか，プログラムジェネレータなど，ただし一応AIの中で考えていますので，ある程度AI的なものがこの中にあるような立場のアプローチというふうに考えてみます．あとでもう少しこれを個別にご紹介したいと思います．

「プログラマズ・アプレンティス」というMITでずいぶん長いことやられて，なかなかうまく宣伝もしているプロジェクトがありますが，それをやっているリッチとウォーターズという人がおもしろいサーベイを書いてます．「自動プログラミングの神話」というようなタイトルだったと思いますが，その中で，自動プログラミングというのは神話，要するに実現しない夢だとすると，これはこういうことだと．つまり自動プログラミングに三つの条件を要求している．一つはエンドユーザが使えるシステムであること．エンドユーザが仕様というか，やりたいことを書けばプログラムは自動的にできると．2番目は汎用的であること．どんな分野にも使えること．3番目は完全自動であること．しかしこの三つを同時に満たすなんてことは無理だと．多分無理でしょうね．この三つの条件のうちの

一つを緩めるという立場で幾つかアプローチがありますよというサーベイです.

たとえばエンドユーザ向けという条件を緩和する. 彼らの言葉では, これはボトムアップアプローチだというんですが, 私の分類でいうと, これはだいたい形式的アプローチに近いもので, この辺はエンドユーザどころか, かなりのソフトウェア実務者でもついていけない. というよりはやる気がしないかもしれませんけど, そういうものです.

2番目に, 汎用性ということをあきらめて, ある領域を限定するという形でのアプローチがある. 先ほど実践的として分類したアプローチには, その辺を目指しているのが多いかと思います.

3番目は, 完全自動をあきらめて, これを緩和する. プログラマズアプレンティストをやっている立場からは, これが本命だということを言いたいんだと思いますが, このアシスタントというのは前の三つの中だと知識的アプローチに近いという感じかと思います.

これだけではあまりにも漠然としています. それぞれの例をもう少しみてみますと, 形式的アプローチというのは, まず出発点として形式的な仕様が与えられる. そこからプログラムを合成しようとする立場だと言ってしまっていいかもしれない. もちろんその中には, 形式的仕様をどう書くか, どういう理論に基づいて形式的仕様をつくるかという研究自身も入っている.

これはほかのものに比べると, 理論的な基盤は割とはっきりしている. 特に2番目の知識的アプローチに比べると, かなりちゃんとした体系があることが普通ですが, 一方できるものは非常に小さな, いわゆるトーイプログラムしかできていない. できていませんけれども, しかしトーイプログラムに対してはある程度の能力は実証されているケースが結構あります. もちろんこれらは研究レベルであって, 研究室の外に出ている例はほとんどありません. ここにいろいろ例をあげましたが, どちらかというとこれはかなり古くて, 場合によっては1960年代から始まっているものもあるぐらいで, 広い意味でのAIの中で割とずっとやられてきたことです. あるいはコンピュータサイエンスの基礎論的な部分に興味のある人たちがやってきたと言えます.

2番目のジャンルは, 同じAIでもいわゆるナレッジエンジニアリングということが言われ出して, それがある程度ソフトウェア以外の分野で成功して, それに刺激されて, ソフトウェア開発のほうにもこういう知識的アプローチが使えるんじゃないかということから始まった立場だと思います. ですから比較的浅いというか, むしろごく最近.

特徴をあげるとすると, 要求分析とか設計とか比較的上流の知識を利用しようとしている. 私がここではそういうものを分類したといったほうが正確かもしれませんが. CASE という関係からいうと, どちらかというと上流の CASE に近い. それから, 先ほど紹介しましたように, 完全自動化というよりはむしろ補助的, 助手的な役割を狙っている.

知識工学的なアプローチをソフトウェア開発に使おうというのはだれでも期待することですが, 現状いろいろ出てきているものを実は今回のために多少つけ焼き刃的に手近な雑誌でにわか勉強してみたんですが, どうもみんな概念的といいますか, スペキュラティブといいますか, そういう印象です. さっきの形式的なアプローチのほうに限った問題についてはある程度結果が出ています. しかしこちらのほうは, プロトタイプをつくったという話はいろいろありますが, それはあくまでも概念を実証するためのものであって, ある例題をやってみたというだけ. そういう意味で有効性はまだ実証されていない.

例としてはプログラマズアプレンティス. これはもう10年以上やっていて, プロジェクトの中では紆余曲折はあるんですが, おもしろい結果も出てはいるようです. その他, IDeA とか ASPIS とか PSDL などはみんな新しいもので, まだまだ本当の結果が出ているわけじゃない. それぞれについても雑誌の論文で見ただけですので詳しいことは分からないんですが, たとえばデータフローモデルの表現で書いた設計のスキーマが後ろにしまってあって, その中から適当なものを取ってきて, それを一つは具体化, インスタンシエイトする. もう一つは詳細化, 階層を下げていく. そんなようなことをサポートしましょうといったものが多い.

富士通の杉山さんのものとか, 日立の千吉良さんたちのものは, どちらかというとこれらと違いまして, 杉山さんのは日本語で仕様を書いて, それからプログラムを生成しようというものですし, 千吉良さん達のは日本語のスペックを書いて, それをしまって, あるいは再利用してある程度のチェックをするというタイプのものですので, ちょっと違うかもしれませんが,

分析設計フェーズでの知識的なアプローチという例としてあげておきました.

ちゃんと定義しておりませんので, 先ほど形式的なものの例にあげたものでも知識的なほうに分類してもいいものもあるかもしれませんし, 今の例でも, 形式的なほう, あるいはこれから申しあげる実践的な方に分類してもいいものもあるかもしれません. その辺のバウンダリは非常にあいまいです.

実践的アプローチは, さっき言いましたように, 第四世代言語とか, プログラム生成系とか, 部品化, 再利用との親和性が高い. そういう意味では下流 CASE.

問題領域を比較的限定しているものが多い. だいたい事務処理とかシステムソフトの例が多いようです. 事務処理といってももう少し範囲を限定することもあるかと思います.

ここにあげたものの一部は開発支援的なタイプです. もう一つは自動生成. たとえばきょうお話があった原田さんのシステムは, どちらかというと自動生成タイプかと思います. これらの中には商用化, あるいはそれに近いレベルのものもあるんじゃないかと思うんです.

この中で Marvel というのはご存じの方もいらっしゃるかもしれませんが, 知識的アプローチということをかなりうたっていて, ソフトウェアエンジニアリングのデータベースに対してそれらの間のプロセスの記述をルールベースで書いている. またプロセスの結果の生成物はオブジェクトベースとしてしまって, その構造や関係を管理するという, 割とおもしろいアプローチかもしれません. また CASE/MVS というのは IBM の例です.

そんなようなことで, きょうの全体のテーマの CASE ということで考えてみますと, CASE を文字どおりコンピュータ・エイデッド・ソフト・エンジニアリングと額面どおりに受けとると非常に範囲が広くなってしまいますし, これまでやられてきたツールとかソフトウェア開発環境と何が違うのかということになる. そこで CASE の出てきた経緯あるいは力点の置き方をしいて限定的に言えば, 一つは, どちらかというと下流よりも要求分析とか設計を支援するんだと. もう一つは支援するのに図式表現, グラフィックスのようなものを使ってやる. こういう性格づけができるんだろうと思うんですが, そうだとすると, 今三つあげましたアプローチの中でこういう意味での CASE に割と近いものは, やはり知識的アプローチじゃないかと思います.

ところが, 今あげました三つのアプローチの中で, 知識的アプローチといっているのは, 先ほど言いましたように概念的なレベルの状況でありまして, 何ができるのかという見通しが, 今の段階で最も不透明. そういう意味では最も夢らしい夢なのかもしれませんが.

そういうだけではあまりにも不確かなので, この知識的なアプローチの中でやられているものの比較的共通な基盤を考えてみますと, これは今回のシンポジウムでもきのうの松本さんの話を初め多く出てきましたが, 一つは多様な生成物, 生成物というのは仕様とかプログラムとかテストデータとかその他のドキュメントとか, それらの単なるデータベースじゃなくて, もう少し構造をもった, オブジェクトベースという言い方がいいのかと思いますが, 生成物とそれらの間のいろんな関係を含めたライブラリのようなもの. 二つ目が, これらの間に作用するプロセスについてのいろいろな知識. たとえばさっきの Marvel の例では, これらをプロセスの前提条件と終了条件ということで, ルールで書くことをやっておりましたが, そういう形に限らず, もう少しダイナミックな知識ですね. こういうものの組み合わせでやるのが, 共通な技術基盤としてはあり得るだろうし, その辺に見込みがあるのかもしれません.

最後に, 夢という題ですので夢を語らなくちゃいけないんですが, 私自身の夢を言う前に今日あげたものの中を幾つか物色してみますと, 一つは先生タイプの CASE ですね. なんとか方法論を教えてくれるようなシステム. こういうものもありがたいかもしれませんが, それはある程度本を読めばいい. 本じゃ本当は身につかなくて, 経験して身につけるべきものかもしれませんが, それにはどういうシステムのサポートができるのか分からない部分もある.

二つ目のタイプは, プログラマズアプレンティスなんかそうですけれども, 助手としていろいろやってくれる. こういうのも多分役に立つでしょう. しかし AI という意味では先生でもなく助手でもなく, むしろ自分の分身, 要するに自分とともに成長していってくれるようなシステムが欲しい. あるいは何も分身といった人間のアナロジでいわなくても, ツールとして使い込めば使い込むほど使いよくなるようなものでもいいのです. AI では, 機械学習というのは永遠のなかなか進まない分野ですが, やはりだんだん賢くなっ

ていくというものが実現してほしいわけです.

そういうものとして，たとえば，終わったプロジェクトについて要領よく覚えておいてくれる．要領よくというのは，ほっておくと膨大なデータベースができちゃいますね，そうじゃなくて，ある程度めりはりをつけて，あるいはプライオリティを考えて覚えてくれる．それから，覚えているだけじゃなくて分析とか設計など，いろんな場面で関連する過去の経験を思い出してくれる．失敗した例についてはよいタイミングで，こういうときはこういうふうにやったら失敗したということを思い出してくれる.

これらだけですと，経験は個別的かもしれませんが，それらの経験をだんだん一般化して，少しずつでも体系化していってくれる．そういう意味では使い手とともに成長していってくれるようなシステム．これはあくまでも夢ですけれども，AI らしいテーマではないかと思います.

これではあまりにも個人的じゃないか．たとえば，先ほどからいろいろ出ているチームとしてのシステムとしては困るじゃないかということがあるかもしれませんが，もちろんこれでもだんだん体系化していったら，ほかの人が育てたものを使えるかもしれない.

もう一つは，別に過去の経験は個人的な経験だけじゃなくて，だれだれに聞いたとか，どの本を読めばうまくいくとか，どのツールを使えばいいとか，要するに世の中に対する知識も当然経験の中に含めて，あるいはあの人に聞くと失敗すると，そういうことも含めて言っているわけです.

以上が私の夢です.

司会 どうもありがとうございました．以上，4人の方の夢の一端をまずお聞きしたわけなんですが，お聞きになって分かりますように，やはり四人四色といいますか，いろんなタイプの夢を見ておられるわけなんですが，まず夢に対してだいたい自分のタイプはどの先生のタイプに近いかなと考えながら，だいたい夢というのは朝起きてもなんとなく矛盾しているところとか，どうも理論的に合わないところが私の夢にはよくあるわけですが，そういう意味で，4人のパネリストの方々の夢をもうちょっと確認しておきたい部分を中心に，質問などお願いしたいんですが，いかがでしょうか.

6. 質　　疑

堀川(三菱電機) 三菱電機の堀川と申します.

今4人の先生方の夢をお聞かせいただきましてどうもありがとうございました．私の夢は落水先生の夢と少し似てまして.

司会 落水タイプですね.

堀川 はいそうです．人間のコミュニケーションをなんとかしていこうというところが非常にいいなと思っております.

それで，人間のコミュニケーションとかインタビューといった人間を相手にした場合，人間というのは機械と違って，うそついたり，何も考えていないのにしゃべったり，いろいろなタイプの人間がいるわけで，コミュニケーションとかインタビューというのは非常に難しい問題だと思います.

そういった意味で，阿草先生にご質問したいんですが，もしコーディネイトロボットというものができてみんなが1人ずつコーディネイトロボットをもったとしたとき，コミュニケーションはロボットとロボットの間でしか成り立たなくて，それは機械ですからなんかごちょごちょやっている．人間はみんな孤立してしまって，みんな哲学的な世界に入ると考えてよろしいんでしょうか.

阿草 当然落水さんにふられたもんだと思って安心していたんですが.

コミュニケーションとかインタビューのところでのコーディネイトロボットは触媒の作用であって，本人は理解しないものであると．ですから，司会者というのは必ずしも理解していなくても，その場の雰囲気は盛り上げられるわけですから，そういう意味ではいろいろなアプリケーションにとって，その人にとってどういうきっかけを与えれば話し出すかを見つけ出すんであって，内容を必ずしも理解しなくてもいいので，コーディネイトロボットが存在しうる.

そういう意味では，そこで話された内容をもう一回まとめ上げてシステムにあげるのは別の人であって，コーディネイトロボットは単にそこでの触媒効果であると僕は理解しております.

堀川 どうもありがとうございました.

司会 よろしいですか．ほかにありませんでしょうか．夢ですから何を語ってもいいわけで，あまり堅苦しく考えずにご発言いただきたいと思います.

長谷川(オリンパス光学) オリンパス光学の長谷川です.

私どもの仕事の現状は，新しい製品をつくることが少なくて，過去にあるものを改良してチューンアップ

して行うとか，そういう形のものが多いんですが，そうしますと，与えられた新しい問題に対して過去のことと見比べて，自分はどういう仕事をしなければいけないか，どういうアプローチをしなければいけないかをいつも考えて，そこで非常にとまどっているのが現状なんですが，何か新しい問題を与えれば，ここではこういう問題であるからこういうことをしなさいよと教えてくれるようなシステムが欲しいなと，私たちは夢として描いています．

そうした場合にどういうふうなアプローチをしたらいいのか教えていただければ幸せなんですが，多分AI 的な知識アプローチになると思いますので，玉井さんにお聞かせいただきたいと思うんですけど．

司会 質問の趣旨をもう一度お願いします．過去と未来というのがよく分からなかったんですが．

長谷川 過去にある問題があって，解決したシステムが働いている，そして今度新しい問題，ある程度は似通っているけれども，でも問題解決は違う．

司会 完全には同じじゃないけれども，過去の遺産といいますか，いろんな経験が新しいシステムを作るときにうまく手軽に応用できるような仕掛けはないでしょうかということでしょうか．

長谷川 そうですね．

司会 じゃ，そういう観点から玉井さんお願いできますか．

玉井 私の夢の中でもそういう過去の経験をうまく活用するようなものが欲しいという話をしたんで，おっしゃることの気持ちはよく分かるわけです．ですけれどもあくまでも夢で，どうやったらいいかというのは私も分からないんです．今少しやられているのは，過去のものといいますか，たとえばこういうタイプの問題について類似なものを検索する．類似なものというのは，過去のものがごたごたに入ってたらもちろんだめで，ある程度抽象化したものがしまわれていて，それを検索して，それをもってきてその問題に合わせて加工する．そういう想定で研究している例もあるようですし，また可能性もあるかもしれませんが，やはりそこのスキーマというか，どういうものを抽象化しておいたら役に立ち得るのか，何をきっかけで検索したらいいかというのが非常に難しい．

今やられているような形であると，とても知能的とは言えないし，そういうもので一定の限られた例題についてはこんなにうまくいくじゃないかという例をみせられても，それはあくまでもデモであって，その例でしか使えないんじゃないかなと思っています．その辺が私自身も分からないといえば分からないところです．

ですから私が思うには，そんなに大きなことを期待しないで，ヒント，たとえば自分自身では忘れているようなことがあるかもしれませんし，もっとシンタクティクカルなことで検索するんでもいいから，そういう形で引っ張ってこれる．引っ張ってきて，見ているだけでも参考になるかもしれないし，あるいは無駄かもしれませんけれども，そういうようなシステムが当面考えられるかと思います．

西尾 西尾と申します．

私自身ここ数年間 AI の研究をしておりまして，一番感じることがあります．阿草先生にお伺いしたいんですが，シンタックスとセマンティックスというはざまですね．要するに，ソフトなソフトウェアというのは本当にどこまでできるのかというのが質問です．たとえば，一つ日本経済をゴリゴリな経済学者が書かれたら，分かる人は分かるんですが，精緻に書かれると間違っているところがはっきりしてくる．これを漫画日本経済でやりますと，分かっているところも分からないところも一緒になって分かったような気分になる．漫画的な表現をすると非常に危険な部分をもつと思うんです．あいまい性ということもあるんでしょうけれども，要するにシンタックスとセマンティックスというのははざまがあるんでしょうか，それを埋められるんでしょうか，その辺私自身すごく疑問に思っています．お答え願えますか．

阿草 非常に難しい問題だと思いますが，少なくともわれわれがコミュニケーションをするときには，まず言葉を定義してコミュニケーションすることがよくありますね．そうすると，そういう定義のメカニズムをもっていて，最後までその定義が相手に伝わったかどうかが分からずにコミュニケーションが始まって，定義に戻ることがよくある．そういう意味では，やはりシンタックスとセマンティックスというのはダイナミックにわれわれのコミュニケーションでは変えてる．それがコンピュータで変えられるかということが問題です．少なくとも今，ソフトウェアの環境がつくられるとき，一番最初にまずシンタックスを決めてこうしましょうと．いわゆるデータアーキテクチャを決めてこうしましょうというやり方は，そういう意味で問題ではないかと．ひっかかったときにいかに戻れるかという問題だと思います．

あとは，玉井さんのほうとも似てるかも分かりませんが，失敗の例をタイミングよく見つけるんでしたっけ．なんか適当な時期にうまく見つけると．僕は正しく成功した例はあまり役に立たないと思うんです．同じようなものだったらうまくいくけれども，別な例を前に成功したからと同じことをアプライしようとすると，前のセマンティックスとシンタックスで次の問題も解こうとするから問題が起こるんであると．そういう意味では，知識ベースの中に失敗の知識をうまく入れることがシンタックス・セマンティックスにもダイナミックに後戻り点が見つかる方法ではないかと思う．

ただ，じゃどうやるのかというと全然分かりませんが，感覚としてはやはり知識ベースとかなんとかというときに，いつもいいほうのことばかりやっている．たとえば，お金持になった人が「こうやればお金持になります」という本を書かれても，そのとおりやってもみんなお金持にならずに，本当は失敗した例だけを書いておくと，少なくともそういう失敗に陥らない．そういうことの蓄積をもう少し AI 屋さんがやってくれたら役に立つんじゃないかなと．きょう玉井さんの話でも，僕はあそこの項目が一番気に入ったんです．そんなので答えになりますか．

西尾 そういう解釈ですと，先ほど言われた玉井さんのプログラマズアプレンティスというのは私も研究しているんですが，要するに失敗事例も含めて提示していると，より人間は総合的に判断するという理解でしょうか．

阿草 (うなずく)．

西尾 どうもありがとうございます．

司会 ほかにございませんでしょうか．

久世(IBM) IBM の久世と申します．

落水先生にお伺いしたいんですが，このシンポジウムでも幾つかプロセスプログラミングに関する発表があったと思うんですが，これから先プロセスプログラミングが非常に有効になると思うんです．私は個人的に，プログラミングですからなんらかの記述する言語が必要になってくると思うんですが，その際にソフトウェアのプロセスはなかなか定式が難しいと思うんです．だから言語もデザインが難しいと思うんですが，落水先生がもっていられる夢のプロセスプログラミング用の言語というんですか，その辺に関してお伺いしたいと思います．

落水 その件ですが，今いろいろ調べてるとこです．先ほど阿草先生とか玉井先生に出た質問にも関係あります．

この１年間学生相手に設計方法論を三種の問題に適用するという設計プロセスを収集しました．それを基にプロセスフローチャートをつくったり制御構造を検討したり，データのまとめ方をを検討しておりますが，なかなか大変です．

手続き型がよいのか，オブジェクト指向がよいのかルールベースがよいのかなかなか結論がでません．

プロセス記述に関する共通の問題点はプロセスのダイナミックスの記述です．平たい用語で言いますと，臨機応変ですが，人間なら簡単にできる臨機応変をどのくらい書けるかが，結局こういう研究が単なる夢物語に終わるか，ある程度は部分的にも使えるようになるかの境目だと言われてます．

司会 よろしゅうございますか．それでは別の方．

鯵坂(京都大学) 京都大学の鯵坂と申します．

夢の矛盾を一つお聞きしたいんですけれども，ツールを自由に柔軟に組み合わせるという話が特に中田さんからあったと思うんですが，自由に柔軟に組み合わせるというと自由，柔軟という言葉に反しまして，ツール間のインタフェースはどうしても固定しにかからなければいけないわけです．そういたしますと，インタフェースを固定されたツールを使ってつくったソフトウェアというのは，やはりどこか型をはめられたものになってしまう．そうすると，冒頭阿草先生がおっしゃってたソフトらしいソフトじゃなくて，CASE というのは，まあソフトウェアの開発を工業化するというのはそういうことなのかもしれませんが，自由に柔軟にツールを組み合わせようとしているのに，インタフェースが固定されて，でき上がってくるソフトにたがをはめてしまう．そういう方向に進むものであるのかどうかをお聞きしたいわけです．

現在，たとえば UNIX ですと，ツール間インタフェースがパイトストリウムと１本のパイプで，これはすごくルーズなもんだから柔軟に使えると思うわけですが，それに対してツール，特にマルチベンダのツールを組み合わせてやっていくというのは，どういうふうな様子になっていくでしょうか．

司会 中田さんよろしいですか．自由で柔軟というのは矛盾じゃないかという……．

中田 ユーザインタフェースの件とツール間のデータのインタフェースの件，それぞれ自由につくったもの，あるいはいろいろなツールメーカさんといいます

か，そういうところで供給されたものを組み合わせて一定のものにすると，自由度がなくなり，ガチガチのものになるんじゃないかというお話ですけれども，こういうふうにまずご説明を補足させていただければいいかと思います．

たとえば，ユーザインタフェースですと，メニュー形式を好まれるところとか，グラフィックスによるポップアップメニューみたいのでやるとか，いろんなやり方があると思うんです．対処としては，ユーザ要求に合わせて CASE の中でフロントエンド的にたとえば，ツールのユーザインタフェース，メニューをつくる機能，そういったものが共通にあって，それと個々に買ってきたツールをつないで，ユーザは全部メニューインタフェースでツールが使えるというようにします．

たとえば，裸の UNIX に馴れてて使いやすいという方もいらっしゃいますが，ホストコンピュータに馴れている方ですと，ちょっとよく使えないということがあります．そうすると，ホストコンピュータ的な操作で使いたいというときに，そういったことを簡単に可能にするようなフロントエンドのユーザインタフェース提供といいますか，そういうことをやっているツールもありますし，そういったものと組み合わせて使えば，いろんなところで買ってきたツールがエンドユーザの一部でメニュー画面みたいなものから選択して使えると．そんな感じのインタフェースの統一が実際にある程度やられております．

そのユーザにとって一つの馴れている，使いやすいユーザインタフェースを使うことが統一という意味で，別のユーザにとって，それは必ずしもそのユーザが好むものではないかもしれません．そういうことですから，幾つかの用意されたユーザインタフェースの統一のツール群，機能がないと，当然そのユーザに適したものは組み合わせてつくれないと思います．

もう一つ，データインタフェースのほうなんですが，確かにご指摘のように，幾つかのツールのデータインタフェースを話し合ったり，あるいはコンソーシアムといわれているように幾つかのベンダさんがスタンダードを目指して一定のデータ表現フォーマットの標準を話し合って，それを普及しようとしたりしています．これはご指摘のように非常に苦労されています．どんな標準ができるかといっても，標準を決めることがなかなか難しい．それから，それをいわゆるデータフォーマット以上の段階でもっと抽象度を上げて，本当に標準らしい標準に定義していくことが今非常に難しい状況になっています．

CASE 関係の学会で 88 年に 9 つぐらいの団体がそれぞれスタンダードに関して議論されたんですけど，これからもまだ検討を続けようというぐらいのところです．それらのレポートを見ましても，データアクセスの共通ライブラリの仕様はこんなもんだねとか，実際の関数ライブラリの仕様書みたいなのがどっときたり，データフォーマットのプログラミングレベルの仕様書がどんときたりして，それがまだ現状でのデータ交換の議論の対象ということですから，なかなかいっぺんに夢が実現するとは思えない状況だと思います．

夢を見ている人，あるいは組織が世界で 10 個ぐらいあって，先ほど言いましたように，それは標準として実現されるかもしれないけれども，されないかもしれないという状況だと思います．

司会 よろしゅうございますか．

鯵坂 はい．

司会 ほかに質問あるいは私の夢とどこが違うのかと，そういうのでもよろしいんですが，いかがでしょうか．

小林(NTT) NTT の小林と申します．

CASE の定義そのものを少し広く考えての意見なんですけれども，今の発表の順から分かるように，アッパー CASE からずっとご説明されましたが，現在ソフトウェアは当然膨大な資産があるわけで，そういったところにこういう CASE 環境での開発形態が入ってくれば，なんらかのそういうブレークスルーがと思うんです．

そういうときに，一つ考えられるのは，リバースエンジニアリングに属する話かもしれませんけれども，たとえば現状のあるソフトウェアのコードからドキュメントですとか，ないしは設計概念みたいなものを逆に生成していくというアプローチが，上流からのシステムを導入する前段としてそういうものがないと，完全に移行していけないんじゃないかと考えているんですが，その辺は AI 的なアプローチになるかとも思いますので，玉井先生にお答え願えればと思います．

玉井 私のところにくると思ってなかったんで．

過去のソフトウェアの蓄積から設計の役に立つような情報を抽出して利用することを考えるべきじゃないかということですか．

確かにそうかもしれませんけれども，私の感じでは

プログラムという形での膨大な資産をプログラムレベルから抽象化し，あるいはパターン化して設計へ結びつけていくのは，かえって非常に難しいんじゃないかと思います．確かに元はそこにあるんですが，それは要するに，海の中に宝物を探しに行くような感じで，もともと設計段階でのアブストラクトした形で考えられたものがあるはずなのに，そういう構造が見えていないものからやっていくのは，私自身は難しいんじゃないかなと思ってます．

それと，これはたしか阿草さんがいつだったか言ってられたと思いますけど，過去のソフトウェアの資産というのはおかしくて，負債だといっていたことがあったと思うんですが，(笑)僕はなるほどなと思ったことがあるんです．

阿草 概念を表現するときに非常に多様性があると．多分プログラムも一つの目的のためにリアライゼーションは幾つかある．その幾つかに分かれてしまったものから元のを見つけることは，本当は何かおかしいという感じがするんです．多分われわれの時代に習ったプログラミングの考え方にのっとってプログラムができているものを，今若い人たちが，過去の悪いプログラミングはなぜそうしたのかを考えながら元の設計に戻すことは，多分労力のほうが大きいんではないかと．

プログラムの幾つかは，プログラムがあるからメンテしないといけないわけで，いつのまにかなくなったら結構使ってないソフトはあるんですね．だから，ものを整理するときはいったん捨てて，必要なものだけ拾いにいくのが多分一番いい整理で，ソフトウェアはディスクはどんどん大きくなるし計算機はどんどん大きくなるんで全部入れているんだけど，本当にどれだけ使っているかというのは必ずしもよく見えない．そうすると，そのリバースエンジニアリングかけるだけの能力をかければ，本当はもう一度新しいものを設計方法にのっとってドキュメント体系もきっちりしてつくれば，次のメンテナンスは非常にしやすい．

ということは，今までソフトウェアの資産であるといって，わけの分からないものをたくさんもっているのは，たとえばスペーシングファクター一つにしても効率の悪いことを一生懸命やっているんで，ソフトは財産じゃなくて負債じゃないかと．ですから，本当に使って初めて価値があるんで，本当に使っているものは書き直しても価値がある．使ってないものを一生懸命メンテナンスするためにそういう努力をすることは，お金を無駄に使っていることで，どこでお話したんですかね，なんかそういう話を一度したことは確かにあります．

小林 分かりました．

司会 ほかにございませんか．

原田(電力中研) 今リバースの話が出てきたんで私もちょっと．

今質問された方及び回答の方のご意見に私は半分同感なんですが，半分実は同感でない．というのは，私はできると思っているんですね．それはなぜかというと，リバースを全部リバースでやろうと思うと私はできないと思うんです．私が今考えているのは，午前中最後のほうで申しましたけど，今設計も自動化しようと考えていまして，その場合の前提は，ER モデルの話がきょうほかにもありましたけれども，私も ER モデル的な形でデータ間のリレーションシップだけを宣言的に入れてもらったら，スペースのコードが出てくるような方法を考えているんですが，それがもしできたとして，そうすればあと既存のプログラムの中にはデータ間のリレーションシップ，要するに代入文でかなりいろいろあるわけです．

私が今考えているのは，既存の COBOL プログラムからそういったデータ項目間のリレーションシップだけをサルベージしてきまして，それから今度トップダウンでみていく設計方法論にその関係書をくわせる．その方法で，既存のプログラムをスペースの仕様書に逆変換しようと，それは私にとっての今後の課題なんですが，考えようと思ってます．

だから，完全にリバースだと無理があると思うんですけど，幾つか上からみる，ガイドする機能があればよい，ソフトウェアをつくるときに一番重要なのは，要求を分析して各データ項目の関係を全部考えてくるわけです．それにはそれをそのまま捨てておくのはもったいないので，それをサルベージしてきて使おうと思っています．だから，セミリバースという感じでやるのが私は有効じゃないかと思っています．

落水 原田さんにご質問しますがよろしいですか．

大事だとかもったいないというのは分かるんですけれども，たとえばツールをつくる人間が抱えているジャンクにしても，そういうアプリケーションの中の部品にしても，コンテクストがあって初めて使えてるわけで，そのコンテクストを切り離した残りが何かのときに役に立つというのは，相当難しい技術を要するんじゃないですか．

原田 コンテクストはシステムを分類するわけです．つまり，これは制御変数なのか，実際の本当に優位なデータ項目を表す変数なのか．データ項目を表す変数に関しましては，ファイル側に定義されていますから，それ以外のものはその途中のロジックに対するものだとか，やはりそのコンテクストをある程度サルベージするときに分類しないと，ゴミまでサルベージしてきたんではしょうがありませんので，それをやろうと思っています．

阿草 あまり同じ問題で議論するのはよくないのかも分かりませんが，今の仮定もまともな方法論にのっとってつくられたソフトウェアなら，その途中経過がなくても元へ戻る．その可能性はあると思います．ところが，そういうソフトであるかどうかの判断もせずに一生懸命サルベージしてても，たとえば下から土砂をすくおうと思っても，下に土砂があることをいかに見つけるかという方法論がないかぎり難しいんじゃないかと．

だから，原田さんが過去につくって，しかもドキュメントが何かの過程でつくられなかったソフトウェアだったら多分うまく戻るでしょう．ところが，だれがつくったかよく分からないソフトであるとか，変数名一つにしても，たとえば制御変数でないものを使っていたのに，あいているからといって別の名前を使っちゃうとか，そういうことは昔はよくやったわけで，そのソフトから幾ら考えてもアイデアは出てこないと思います．だから，便易法で使ってしまったものを幾ら考えても，もとには戻らない．そういう問題もあるということ．

原田 これは多分ソフトウェアの種類が違うからそういう誤解が生じると思うんですが，業務用のソフトウェアの場合には，データ名はドキュメントにもしっかり残っていますように，対応がついているんです．割とその選別はできます．阿草先生のおっしゃったことに一つ同感するのは，全く自動化は無理だと思っているんですよ．私は支援系でもっていきたいと思っているんです．

なぜそういうことを考えているかというと，SPACEとかそういうものの人の説明書は，既存のシステムをどうしてくれるんだという意見が非常に多くて，幾つかを SPACE にリライトするときも，いちいち計算式を取ってきて，また同じような方法でやるのは非常にめんどう．それはなんとなくかなり自動化できるかなというのが，今言った話の発想の元なんですけど．

だから，大きなモジュール分割とか，その上の分割は人間がある程度みてやらないとできないと思います．

7. おわりに

司会 まだ議論はしたいと思うんですけど，ちょっと時間の関係で．では最後に，4人の先生方に，先ほどで結論も言われた先生もおられるようなんですが，もう一言，きょうは私はこれを言いたかったんだという，あるいは，実はこういうふうに言いたかったんだということのまとめをほんの数分ずつお願いできますか．

阿草先生からまた．

阿草 プロセスプログラムでプログラムを書けばいいと．プログラムを書いてもダイナミックに実行時に変えないといけないということは，結局ある意味で，何も書かなくてもダイナミックに考えなさいと書いておくだけですむわけです．

ですから，われわれが自動的にソフトウェアの CASE をうまくサポートしようとすれば，今の原田さんのご意見でもあるように，アプリケーションさえ決まればかなりの部分決まるんだと，そういう事実を知っているにもかかわらず，あまりにも一般化しすぎた問題にトライしすぎているんじゃないか．

そういう意味で，環境ごとになんらかの言語の体系をもって，それが自由に言語を定義できる，いわゆる自分たちのカルチャーをうまく写すように言語をいかに定義させるか．そうすると，その定義に応じていろんなソフトができてくると．そういう意味で，用意されたソフトウェアはソフトなソフトウェアで，ある部分自分が定義すれば，自分のアプリケーションにより身近に使えるようなソフトウェアを今からつくるべきではないかと．

ですから，たとえばアプリケーションスペスフィックな 4GL をつくったから，これであとは書きさえすればいいといっても，やはりそれでは不満足な人もいる．そういうのをいかに柔軟にするかを今から研究を進めていくべきでしょうというのが僕のきょうの主張というか，結論です．

司会 どうもありがとうございました．落水さんお願いします．

落水 実はこういうシステムを作りたいという OHP を用意してきたんですが，それよりも玉井さんがおっしゃったことの方がもっとうまくまとめてあります．

個人レベルの経験を分析してデータを集め，その事例がたくさんたまったときにそれをアブストラクションする．

それを形式的に言語で記述して他人が利用できるようにする．結局玉井さんの夢を実現する夢をもっているということになります(笑)，これが結論です．

司会 分かりました．中田さんお願いします．

中田 プレゼンテーションでこういうツール供給者，ベンダ，ユーザという話を申しあげまして，いろんなツールを自由に，場合によってはユーザさん自身がつくったインハウスのツールも組み込めるという話をちょっとしましたが，この中の議論で抜けてるなと思うのは，今あるツールを組み合わせてどれだけいいCASEができるのという話があるかと思います．今あるCASEというのは，阿草先生が最初におっしゃったように，清書ツールが非常に多いわけです．それがそれなりに非常にいい面をもっているわけです．そういうツールは，人間でいえば経験5年以内ぐらいの人の能力というか，スキルをサポートしているのかなと思いますが，実際にいろんなソフト開発の中では，エンジニアリング的に10年とか15年とかのノウハウがないとできないことがあります．

人間が成長していく，経験とともにソフトウェア関係の仕事で成長していくのに比べると，今提供されているツールをただ組み合わせただけで，どれだけいいCASEができるんだろうという夢の心配があろうかと思います．

きょうの話で一つ抜けていましたが，ツールがただ組み合わさる，あるいはいろんなツールを組み合わせてそれらが広がっても，もう一つ方法論をどうするのという話があろうかと思います．AIに非常に期待したい．あるいはプロセスプログラミングがどうなるのか分かりませんが，非常にうまくいった結果として単に今ある単品ツールがどんどんどんどん組み合わさったものというのではなくて，方法論からずっと上がってきて，それが全体のCASEになる．統一した方法論を組込みというとちょっと言い過ぎなのかもしれませんけれど，そういうところができるためにAIに私はぜひ期待したいと思っております．

玉井 私の夢は落水先生が実現してくださるそうですので，私としては全く言うことはございません．ただ，AIというのは常に夢だと思いますので，そういう夢は残しておいていただきたいということ．それは多分大丈夫だと思いますが．

以上です．

司会 ちょうど時間になりましたので，これで一応このパネル討論を終わりたいと思います．

特に結論はないんですが，最初に私が話しましたように，このパネルを企画した一番の背景は，夢をもたないというのは非常に悲しいことだということを，私みたいな年寄りはいいとしても，皆さんのように若い方は，夢をもたないことがいかに悲しいことかということを，そのうち実感として出てきますので，ぜひ夢をもっていただきたい．夢をもてばある程度情熱がわいてきますし，それなら一つやってやろうということにもなるわけで，ぜひまず夢を見る習慣をつけていただきたいということを最後に私のほうの企画側からお願いいたしまして，これでこのパネルを終わりたいと思います．

それでは，4人の先生方に感謝の意を込めまして，拍手でお礼を申しあげたいと思います．どうもありがとうございました．(拍手)