SEAMAIL

Newsletter from Software Engineers Association

Volume 5, Number 3 1990

目　次

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，1650 名を越えるメンバーを擁するにいたりました．法人賛助会員も約 60 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．５，Ｎｏ．３　　平成２年３月３０日発行
編集人　岸田孝一
発行人　ソフトウェア技術者協会（ＳＥＡ）
〒１０２　東京都千代田区隼町２－１２　藤和半蔵門コープビル５０５
印刷所　サンビルト印刷株式会社　　〒１６２　東京都新宿区築地町８番地
定価　５００円　（禁転載）

事務局から

☆

３月２０日，正午．いま，Vol.5 No.1-2 合併号を郵便局に手渡したばかりのところです．今年の SEAMAIL はスタートでちょっともたつきましたが，これからは順調に行けそうな予感がします ?????

☆☆

上のパラグラフの末尾の疑問符が取れるかどうかは，会員のみなさんの御協力如何にかかっています．

☆☆☆

まず，去年の夏から秋にかけて開かれたワークショップ関係者のみなさん，レポートのとりまとめを至急お願いします．だれですか，まだ自分の担当分の原稿を提出していないと頭をかいているのは！

☆☆☆☆

このところ月例 SEA Forum の企画は快調で，１月そして２月と連続して大入り満員の盛況でした(両方とも CASE がテーマ)．参加者の名簿を見ていると，ほぼ毎月参加しておられる常連の方が何人か目につきます．そこで，これらの方々へお願い！　出たくても出られない地方会員のために，ぜひ，Forum での討論を聞いて考えたこと etc をまとめて，事務局あてお送りください．よろしく．

☆☆☆☆☆

この馬鹿人気の CASE Forum については，各地の支部や関連地方団体の協力を得て，「日本縦断ツアー」をやろうかという半分冗談まじりの企画が出て，すぐ悪ノリする SEA のこと，先日の幹事会で，とりあえず仙台（５月末）および長野（６月末）での開催を目指して動き出しました．他の地方でも，ローカル・アレンジメント（会場の手配，参加者募集，etc）を引き受けてもいいという方がおられたら，ぜひ御連絡ください．

☆☆☆☆☆☆

さて，この号の巻頭の記事は，２月に行った会員アンケートの自由回答部分の集大成です．このアンケートは，なぜか回答率がよく，全部で１７０通以上集まりました．そこで，２匹目のドジョウをねらって，やはり「これからのソフトウェア開発」をテーマにしたアンケート用紙を同封しました．今回も大勢の方からの回答をお待ちしています．

☆☆☆☆☆☆☆

1990 年代のソフトウェア開発について

— SEA 会員アンケート —

２月に行われた会員アンケートでは，1990 年代のソフトウェア開発を象徴するキーワード，および，ソフトウェア技術／ソフトウェア産業のこれからについての予測をおうかがいしました．以下に，みなさんからの回答を御紹介します（敬称略，到着順）．

野村　敏次（日本電子計算）

◇ マルチメディア
◇ オブジェクト指向
◇ 分散開発環境における集中化

高速，大容量の RISC マシンがかなり普及し，現在の開発環境とは異なった形の分散環境が出現する．しかし，技法の普及の遅れ，環境構築技術の未熟さなどのゆえに，ソフトウェア開発の分散化は，まだ本格化はしない．メインフレーマとの融合化が進み，WS 対汎用機といった概念的な区分は，なくなるであろう．

林　香（SRA）

◇ Smalltalk
◇ CASE
◇ Hyper*

Smalltalk による革命的な変化がおこる．1980 年代の Unix の比ではない．

西郷　正宏（日本システムウエア）

◇ エンドユーザ・コンピューティング
◇ グラフィック・ユーザ・インタフェイス（GUI）
◇ マルチメディア指向

基本的には，SEA の常任幹事である臼井氏の「さようなら戦艦，さようなら英雄」のシナリオ（SEAMAIL Vol.3, No.10）に賛同します．また，ソフト技術自体も，臼井氏のいう Hard of Soft から Soft of Soft の世界へ進んで行って欲しいと思います．つまり，生産物としてのソフトウェアから，著作物としてのソフトウェアへ移行して行かないかぎり，われわれの産業に未来はないでしょう．その意味で，ソフトの生産面のみを扱おうとしたΣ計画は，少し夢が足りなかったのではないでしょうか？

小林　俊明（ソフトウェア技術）

◇ ネットワーク（広域ネットワーク）
◇ オブジェクト指向（Smalltalk）
◇ 分散処理

前半は，Unix ワークステーションと LAN による企業内業務処理の合理化が進むだろう．メインフレームも LAN に接続され，メインフレーム内の DB をワークステーション側で利用するようなアプリケーションが増加するだろう（サーバ・クライアント・モデルの全盛期）．

後半は，Unix が限界に達し，分散処理に適した OS，言語，ハードが出現してくる．それは，分散型 Smalltalk とでもいえるような，分散型オブジェクト指向マシンだろう．この分散型オブジェクト指向マシンは，ネットワーク上の CPU パワーの共有，プログラム資源の共有を実現し，ネットワーク上のどこからでも，そのネットワーク上にあるすべての CPU パワーとプログラム資源を利用できるようになる．

また，プログラム開発の生産性はオブジェクト指向のインヘリタンス機能を活用することにより，飛躍的に増大するものと思われる（分散型オブジェクト指向モデルの発展期）．

窪田　芳夫（東京電力）

◇ エンドユーザ自身の手によるシステム開発
　（第４世代言語）
◇ 分散型共通開発環境
　（ターゲット・マシンからの独立）
◇ ノイマン型コンピュータからの脱皮

ソフトウェア開発環境は飛躍的に改善され，プログラム開発だけを業とするようなソフト会社は急速に衰退する．商品として，あるいはプロダクトとして，どんなシステムを作り出すかが勝負のポイントになる．知恵とアイデアへの傾斜が高まり，腕力競争の時代は早晩終わるものと考える．

浦野　昇千（ソフトウェア興業）

◇ ハイパーテキスト／メディア
◇ NeXT
◇ 光

ソフトウェアは工業製品に近くなる．プログラム言語と呼ばれるものは，ほとんどなくなる（意識しないですむようになる）．TRON は大失敗に終わる．

岡芥　晃（東芝エンジニアリング）

◇ LAN
◇ オブジェクト指向
◇ ＧＵＩ

コンピュータがさらに社会の末端（個人レベル）へ拡大して行く．これにともない，GUIのよくないシステムは取り残されていく．GUIのすぐれたシステムを作るためには，オブジェクト指向の考え方が必要になってくる．また，今後の情報処理においては，情報をさまざまな断面から抽出・分析できるようなDBシステムや思考支援ツールというものが重要になるものと思われる．

篠崎　直二郎（日本電気ソフトウェア）

◇ CASE
◇ 教育（CDP）
◇ Unix & PC

２極化：　大手メーカ系ソフトウェアの充実と業務別ノウハウに強い専門ソフトウェアの成長．

人材交流　－　産学だけでなく企業間で，出向提携，組織的な移籍などのリストラクチャリング．

パッケージ化　－　自動カスタマイズが可能なパッケージの開発．

村重　邦男（日本ビクター）

◇ ヒューマン・インタフェイス
◇ プログラムレスのソフトウェア
◇ 並列処理

アプリケーション（ニーズ）の整理・統合が進み，言語を意識させないような使い方が拡大して行く．仕事に応じて，第４世代／第５世代言語またはツールを使い分けることにより，可能なジョブがふえる．CASEツールは，ルーチンワーク的なソフト開発（特にソフトハウスでの）の分野で，急速に普及すると思う．リリースの標準化(ハード)が進み，ソフトの部品化，プロトタイピング，および仕様の完全化が，コストダウンをもたらし，ソフトウェア産業を支える．それを取り込めない企業は，時代に乗り遅れる．

いずれにしても，得意業務における使いやすいプロトタイピング・ツールを，ノウハウとして整備し，人に依存して移動して行く知的財産を，何とかシステム化することが必須となる．

中村　宏（ハイマック システムズ）

◇ オブジェクト指向
◇ 人手不足が加速
◇ プログラムの知識が必要（いっぱい勉強しなくちゃいけない）

90年代は，意外につまらない10年になるかもしれない．マイクロ・エレクトロニクスの進歩は続くが，ソフトウェアの伸びはゆるやかで，オブジェクト指向やCASEツールの普及も，10年はかかる．いわば，一種の端境期で，次の時代のツールにとっての成長期であるともいえる．ネットワークも接続・管理・コストの問題等があって意外に進まない．しかし，デスクトップWSの性能が100 Mipsをこえるとき，１つの大きなうねりが生まれるだろう．そこそこに期待している．

江沢　智（日商エレクトロニクス）

◇ ダウンサイズ・コンピューティング
◇ ネットワーク・コンピューティング
◇ 分散型コンピューティング

ソフトウェア技術は，個人レベルで，いろいろなツールが選択できるようになる．生産性や品質を向上させたかったら，ツールの選択が１つのキーとなってくる．ソフトウェア産業は，エンジン屋，ツール屋，アプリケーション屋といった形で，専業化が進む．

藤井　論（松下通信工業）

◇ CASE
◇ 分散開発環境
◇ 在宅勤務

ソフトウェア開発は，ようやく労働集約型から脱皮し，CASEを中心とした自動化が進む．ソフトウェア産業は，人手不足というよりは，教育を含めた要員の質の向上が1990年代の課題．そのための道具としてCASEツールが大いに活用される．ソフト技術者の地位が社会的に向上する．

山本　邦雄（横河ジョンソンコントロールズ）

◇ ボトムアップ開発からトップダウン開発へ，および両者の融合
◇ 現場主義（開発担当者が利益を実感できる）に徹した生産技術
◇ 自動化から自働化へ（人と機械が調和したフレキシブルな生産技術）

プロジェクトおよび品質の管理技術，設計技術，開発技術，テスト技術，保守技術などの技術格差をなくすための支援ツールがサポートされる．ハードの発展に伴ない，高機能ソフトウェアが要求され，第２のソフトウェア・クライシ

スが到来する．それらを解決すべく，ソフトウェアの高機能部分をハードウェア化するなど，ソフトとハードの協調(融合)が促進される．新技術は，どしどしハードウェア・コンポーネントとして部品化され，システム化技術が要求されるようになる．

川辺　正明（スタット）

◇ CASE
◇ WS
◇ ハイパーメディア

ソフトウェア産業界内部での淘汰が始まり，上下の２極分化が進む．

中来田　秀樹（ニコンシステム）

◇ SEA
◇ IBM
◇ AT&T

このまま当分のあいだは，混沌の世界が続くでしょうね．

伊藤　朗（日鉄電設工業）

◇ AI
◇ 自然言語理解
◇ オート・コーディング

オート・コーディングおよび各分野の Problem Oriented Language に期待する．OS や言語などの標準化が進まなければ，ソフトの再利用性や可搬性は確保できないが，それぞれのメーカやソフトウハウスは差別化を目指すため，そうした統一は不可能であろう．強力なユーザが，それぞれの分野でまとまり，技術の統一や標準化(国際的にも）を図る必要がある．

大竹　克利（トッパン・ムーア・ラーニング）

◇ CASE
◇ 品質管理
◇ AI

石丸　径美（タクトシステムズ）

◇ オブジェクト指向
◇ マルチメディア対応
◇ CASE

ソフトウェア技術の分野では，前半は特に目新しいものがないが，後半とんでもないものが出てくる（内容は不明）．キーボード入力は主流ではなくなる．ソフトウェア産業は，業界の再編成が進み，ソフト会社の数は今の半分位になる．勤務時間が第１次産業と実質的に変わらないため，人離れが進む．

宇佐美　宣由（ミウラ）

◇ CASE
◇ オブジェクト指向
◇ EWS または分散環境

ハードウェアは価格的にも購入しやすくなるが，結局のところ，中小のソフトハウスにとって，新しい開発技術を導入することは難しい．したがって，現在の開発スタイルがそのまま踏襲され，CASE やオブジェクト指向等の新技術は，大規模組織のみで試行されるだけにすぎない．中小の会社は，相変わらず，力作業に明け暮れている．

福田　充利（スペースプライ）

◇ エンドユーザ・プログラミング
◇ オブジェクト指向
◇ アホにプログラムを作らせるな!!「アホでもできるプログラミング」の否定

名前負けする（名前だけの）ソフトウェア技術者の大量無差別導入により，業界の平均レベルが下がり，金は入ってくるが人材は腐って行くという悪循環が繰り返されるが，突如として大異変が起き，一挙に厳しい淘汰が始まるでしょう．

乾　成里（日本大学）

◇ １人１台の WS
◇ ネットワーク／分散環境
◇ ソフトウェア危機

メインフレームから WS へ，共同利用から１人１台へという流れの中で，ソフトウェア技術は混迷の度を増す．

青木　久夫（長野県情報技術試験場）

◇ オブジェクト指向
◇ チーム・ネットワーク
◇ CASE

ソフトウェア技術の分野では，開発したソフトウェアの保守のための DB 化技術が，最も求められるでしょう．そして，開発での再利用へと向かうのでは....？　ソフトウェア産業は，ソフト販売からシステム販売へ向かうでしょう．

中居　博（昭和電工コンピュータサービス）

◇ オブジェクト指向
◇ Unix
◇ 4GL

4GL や各種ツールの普及，ハードの価格の低下等により，

個人レベルにまでWSやPCが普及し，それにともなって，エンドユーザ·コンピューティングが進む．その結果，ソフトウェア会社の果たすべき機能が，現在のプログラム開発から，データベースの提供という形に変化する．

斎藤　末広（テスク）

◇ CASE
◇ オープン·アーキテクチャ
◇ マルチメディア

SEAとして，再度Σプロジェクトに協力することに意義があると思う．

梶原　秀明（沖電気工業）

◇ スパイラル·モデル
◇ グループウェア
◇ ピープルウェア

ソフトウェア工学などの進歩があったとしても，それらを開発現場に取り入れる企業幹部の考え方や，工業生産一般に関する現場の考え方が変わらないと，80年代と本質的に変わらないだろう．Unixの普及やウィンドゥ·システムの充実などの延長線上での進歩が，今後もアメリカを中心に進むが，開発現場にそれらを導入することは難しく，一部のソフトウェア·ハウスを除いて，日本のソフトウェア生産環境は旧態依然であろう．そうした状況下で，新技術の取り入れに熱意をもって取り組む新しいメーカが登場して，市場をかき回すにちがいない．

大浦　洋一（シーイーシー）

◇ CASE
◇ オブジェクト指向
◇ プレゼンテーション

CASEと開発環境の整備により，アプリケーションの開発効率が大幅に改善される．要求定義と設計の重要性が増し，アプリケーション開発におけるプログラミング作業はほとんど自動化され，必要なくなる．一方，基本ソフトや汎用ソフトの部門では，高い能力のプログラマが要求され，ソフト開発は大きく２極分化する．

加藤　重信（凸版印刷）

◇ 利用者インタフェイス（開発インタフェイス）の改善
◇ マルチメディア対応
◇ フォーマル·アプローチ

今のままの状態で自己改善努力がなければ，ソフトウェア産業は壊滅的に破綻する．

渋谷　誠（アルプス電気）

◇ オブジェクト指向
◇ GUI
◇ LAN

SEAを始めとして，企業にとらわれないソフト技術者間の交流がさかんになり，この業界では転職も日常茶飯事である．いままでの日本の社会のワク組みを越えた新しい業界が形成されてくると思う．それは，たとえば，今の音楽産業や映画産業のように，アーティスト風の人間を中心とした個人の能力が物をいう業界，組織力がそれほど意味をもたない業界となるであろう．そうなることを強く期待する．

石原由紀夫（相互印刷紙器）

◇ バグ
◇ 事故
◇ システム·ダウン

長　昌克（日本コンピュータ研究所）

◇ CG
◇ CASE
◇ パソコン通信

共通OS（Unixなど）の上で作成されたソース·レベルで互換性を持つソフトが，多種類のマシンに移植される．ただし，一般的な（あたりまえの機能を有する）ソフトは，次々にPDS化され，より高度なソフトの開発が業界に要望される．

菅井　浩太郎（住友金属工業）

◇ オブジェクト指向
◇ ハイパーメディア
◇ ISDN

Unixは，技術者と称する人々が集う世界から離れて，ビジネス界で広く応用されるようになる．ホーム·オートメーションの市場を作りだすために，さまざまな試みがなされる．リアルタイム·マルチプロセシングのCPUを有効に活用する分野が注目されるだろう．ソフト開発では，オブジェクト指向が一般的になり，一部の技術者たちは，ニューロ·コンピュータ関連の話題に夢中になるだろう．

北　英彦（沖電気工業）

◇ 分散開発
◇ 外注比率の増加
◇ グループウェア

藤井　猶介（明電舎）

◇ （先進的な）AI
◇ （標準化された）マルチメディア DB
◇ CAI（高度技術教育のスピードアップ）

ソフトウェアの生産負荷はますます深刻化し，ソフトウェア産業に従事する技術者の質のバラツキも深刻な問題になる．むしろ，質が悪くても使わざるを得ないという状況面が生じる．これに対する策が講じられるであろう．

つまり，WS 上にすべての生産支援／管理支援ツールが結合化され，ネットワークを介して，マルチメディア DB を用いた開発管理が行われるようになる．管理データは，DB 上の成果物から自動的に計測され，進捗／生産性／品質などに関するデータが，自由にいつでも見られるようになる．CASE は AI の先端技術をとり入れ，設計や仕様確定などの知的活動を支援する．ドキュメントはすべて機械化，非常に美しくなる．テスト工程も自動化され，仮想マシンやシミュレーション的発想のツールも豊富になる．標準化についても，何かが考えられ実現される．CAI により技術者育成のスピードアップが図られる．

堀田　裕彦（住友電工ハイテックス）

◇ RISC 地獄
◇ Unix 地獄
◇ マルチプロセッサ地獄

大型汎用機はしぶとく生きのこるが，WS は PC に喰われて，衰退して行くのではないか（恐々）．

北村　幸雄（総合システムサービス）

◇ 創造力
◇ オブジェクト指向
◇ CASE

地球環境保存のため，経済活動は減速を余技なくされる．それがどのような形でわれわれにはねかえってくるのか（同じように減速となるのか，あるいは，よりクリーンで効率のよい生産のためにコンピュータが使われ，われわれは忙しくなるのか）が，よくわからないところである．

だれにでもソフトが作れるのではなく，それ相当の教育や技術力が必要になる．つまり，CASE 等を使いこなすためには，他分野のエンジニアのように，設計方法論をきちんと系統立てて身につけなければならない．その上で，初めて創造性が求められるのである．プログラマは，いまのような訳のわからない人種ではなく，技術者と芸術家の性格を合わせ持った新しい人種にならねばならない．

飯富　章博（システムイイトミ）

◇ ハイパーテキスト／メディア
◇ ピープルウェア
◇ 統合型 DB システム

現在，ソフトウェア開発作業の大半は，非能率なドキュメント作成に当てられている．本来重要であるべき，クリエイティブなアルゴリズム開発や MMI の向上等に，充分な時間がさかれていない．ソフトウェア業界のみならず，それを取り巻くハードメーカやユーザがそのことに気づけば，ソフトウェア業界自身も変革をせまられるだろう．ちょうど，東欧の民主化により，ソ連のペレストロイカが促進されたように....？

目黒　敏樹（シンコー）

先端技術を使ったアプリケーション開発が開始される，大規模な先端的プロジェクトと，旧態依然としたアプリケーション開発とが混在することになると思う．

花野　三郎（大和）

◇ CASE
◇ WS
◇ オブジェクト指向言語

中西　和男（日新電機）

◇ リアルタイム・システム
◇ CASE
◇ 分散システム

Unix が事実上の標準 OS になり，WS からメインフレームまでを支配する．一方，これと歩調をあわせて，LAN による分散処理が進むとともに，分散処理システムの開発分野において，そのソフトウェア技術に混沌が生まれ拡大する．さらには，並列処理マシンの市場拡大が始まり，ソフトウェア業界以外の人がプログラミングに苦しめられ始める．

平岡　信之（アルプス技研）

◇ リモート・エンジニアリング（要求発生地と開発地が離れている）
◇ 広域分散システム
◇ インタファミリ・ネットワーク（家庭間の情報通信網）

現在，「気」や「サイ」などと称されている人間の潜在能力が，一部の人に利用されて着目されるようになる．現在のような形態の端末や WS 等は，「低気能者」によるデータ処理や通信を目的とするものであり，そうした高度な情報処理には向いていないので，新しいマン・マシン・システムが

開発されなければならない．

しかし，半分以上の人たちは，やはり旧態依然としたDOSだのC++だのの「文字言語」を使い，EWSみたいな「物理端末」の上で，デジタル・プログラミングしてるんでしょうネ．

佐原　伸（SRA）

◇ オブジェクト指向
◇ CASE
◇ 在宅勤務

ソフトウェア工学および科学の重要性が見直されるだろう．逆にいえば，これらの重要性に気がつかないソフトウェア企業は滅びる．キーワードとしては，オブジェクト指向の全盛になるだろう．

久保　宏志（富士通）

◇ Interoperability
◇ Standardized Platform
◇ Object Oriented Approach

日本のソフトウェア業界で，産業の再編成が起こる．ソフトウェア工学は，ソフトウェア技術の進歩に寄与しない．IBMの地位は下がる．SUN，Appleの地位が上がる．

田中　慎一郎（SRA）

◇ 協調支援
◇ OOSD（オブジェクト指向設計）
◇ 開発プロセスのモデル化

SEAの会員が10,000人になる．

藤野　晃延（富士ゼロックス）

◇ CSCW
◇ Human Interface
◇ エイジェント

ネットワーク上に，実世界をそっくり仮想的に表現したVirtual Reality Worldができる．したがって，すべての新規開発ソフトウェアは，このVRWで実験された後に，実世界で適用されるようになる．ソフトウェア開発は，完全に万人のものとなり，VRWでは，ソフトウェアとアートは同一視され，作成者はアーチストとして遇されるようになるだろう．

岸田　孝一（SRA）

◇ プロセス
◇ オブジェクト指向
◇ CSCW

10年前に"Unix!"と大きな声で叫んだとき，多くの人たちから気狂い扱いされた．しかし，今みんなが"Unix!"と叫んでいるのを見ると，何となく気色が悪い．そこで，気まぐれに"Smalltalk!"と叫んでみた（実は，何回かの環境ワークショップでFXISの青木さんのプレゼンテーションを聞いて，だんだん応援したくなってきたのだ）．さて，5年後，10年後に世の中はどう変わっているかな？

西岡　健自（横河電機）

◇ マルチメディア
◇ ソフトウェア・データベース／レポジトリ
◇ プロセス・プログラミング

1990年代の前半に，ソフトウェア開発における産業革命が進展し，ソフト作りの徹底した機械化が浸透する．したがって，多くの職人的ソフトウェア・エンジニアが潜在的失業者となるが，恐らく解雇されることはない．なぜなら，ソフトウェアの需要が爆発的に増大するためである．

ただし，その仕事の内容は，ソフト作りから，何を作るかの仕様発案作りに移行し，1990年代後半から20世紀初頭にかけて，現在では思いもおよばないような，奇妙奇天烈なソフトウェア製品が，りっぱに商品としてまかりとおるようになる．

小林　伸二（小林ソフトウェアエンジニアリング）

◇ Windows
◇ EWS
◇ 通信（図形を含んだ伝送）

一層の機械化が進むので，設備投資ができない会社はアブナイ．新しい考え方が次から次へと出てくるので，それにふり回されないようにしないといけない．大規模プロジェクトがもっと多くなる．技術者の資質が問われ，ソフトウェアだけ知っていればよいという時代ではなくなる（現在もそうだが，そうした傾向が強まる）．1人前の産業として，（会社，技術者の両方に）より一層の努力が求められる．

加藤　明（フジミック）

◇ CASE
◇ Network
◇ SI

ソフトウェア技術・産業，ともに2極分化する．大規模な人海戦術的開発を必要とする従来型ソフトウェアと，フレキシブルな創造的ソフトウェアとは，市場，価格，開発技術などすべての面で，別の産業であるかのごとくに発展して

行く．企業経営のやり方も，労働力の供給も，それぞれに専門化する．ハードウェア面でもメーカ間の壁が厚く，両者の間の技術移転は小規模にしか行われない．共通基盤ができるのは，2000年を迎えてからであろう．

中野　秀男（大阪大学）

◇ ハイパーテキスト／メディア
◇ Smalltalk
◇ 分散環境

ソフトウェア産業は，より道具作りにはげむようになる．従来のようなアプリケーションは，ユーザ･サイドで作るようになる．「ソフトウェア技術は，Smalltalk 80/90のような言語／環境を使えば飛躍的に生産性が上がる」という人もいるので，しばらくは，のせられてみようと思う．

中山　聖章（富士通ソーシアルサイエンスラボラトリ）

◇ Unixワークステーション
◇ ネットワーク
◇ 知的財産権

企業間の生産性格差が拡大し，生産性の低い企業は生き残れなくなる．総合的な開発環境の差が，生産性の差となって表れてくる．

岩田　康（SRA）

◇ ワークステーション
◇ ネットワーク
◇ オブジェクト指向

岸本　剛一（東芝）

◇ 分散処理
◇ De Fact Standard
◇ Interoperability

ソフトウェア技術の最大の問題は，あまりに低い機能を組み合わせて，非常に高い機能を実現できることである．ハードウェアでは，寸法，重量，強度，発熱などの条件から必ず起きている「モジュール化 → 技術の固定 → 技術の共有財産化」がなかなかできない（できても永続きしない）．ようやくUnixとその上のツール群が出てきたが，フィルタでコマンドをパイプでつないだ形のシステムで，どのくらいのアプリケーションがカバーできるだろうか？　しかし，これこそ今後の一般ソフトのあるべき姿だと思う．

飯沢　恒（三菱電機東部コンピュータシステム）

◇ EOA
◇ CASE
◇ UI

ソフトウェアの生産技術が特にクローズアップされてくる．現在は，ひとくちにソフトウェアといっても，ゲーム･ソフトから巨大なアプリケーション･システムまで，あまりにも幅が広いが，将来は工業製品（規格品）と工芸品（芸術品）とに分化する．産業もそれに伴い，いくつかのグループに分化する．

尾上　俊明（オムロンソフトウェア）

◇ ハイパーメディア･システム
◇ オブジェクト指向データベース
◇ CASE

能見　巧（PFU）

◇ 心の中の開発環境
◇ 超高速･日本的コンピュータの夜明け
◇ もう1人の自分

1990年から始まる10年間は，コンピュータがコンピュータであることを否定し，人間が現時点での自分をコンピュータに求める時代が来て，コンピュータ世界でコンピュータの中に主役となった自分を見る時代がくる．感応ソフトウェア時代の幕明けとなるか？

福島　秀明（PFU）

◇ 効率化･生産性
◇ 信頼性
◇ 高度化にともなう一層の激しい労働環境

標準化が一段と進展する．日本社会の閉鎖的な体質から抜けだすことが，企業生き残りの大前提である．共通プラットフォームの環境で，知恵やアイデアを花咲かせることが肝要になる．旧人の発想ではなく，ゲーム感覚の世代の感性が，社会生活の変革をもたらすだろう．

太田　晃二（ソフテックインターナショナル）

◇ 分散処理
◇ 標準化
◇ 自動化

ユーザ･インタフェイスのよいWSによる分散処理へと，時代の流れは向かって行くように思われます．開発フェーズにおいては，一層の自動化が進み，システム･コンサルティングを中心として，ソフトウェア業界の社会的地位は向上して行くでしょう（そう希望しています）．

藤田　昌弘（関西情報センター）

◇ ポータビリティ

◇ インタフェイス
◇ マルチディメンジョン

ソフトウェア規模はますます大きくなる．従来の技術を変える力は，新人にしかない（人はだれでも保守的である）．すでにソフトウェア人口にカウントされている人には，新しい技術は使えない．新人たちは WS の周辺でのソフト開発に従事し，大規模プロジェクトは，旧人類が四苦八苦して何とかやって行くしかない．そうした世代による２極分化が進むだろう．

折田　豊（テーエスデー）
◇ オブジェクト指向
◇ エンドユーザ･プログラミング
◇ プロトタイピング

オブジェクト指向，イベント駆動型のプログラミングが流行する．似ているけどちょっと違う規格がたくさんできて，混乱する．

岡本　吉晴（三菱総合研究所）
◇ ヒューマン･インタフェイス
◇ インテリジェンス
◇ 超機能分散

1980 年代は，情報通信ネットワーク技術と大規模統合データベース技術にもとづく「高度情報化」の時代であった．1990 年代に入ると，情報システムが大衆のレベルにまで接点が拡大され，そのために「ヒューマン･インタフェイスの高度化」と「柔軟に対応するためのインテリジェント化」が要求されてくる．すなわち，1990 年代は「インタフェイス＆インテリジェンスの時代」であるといえる．そして，「企業の，企業による，企業のための情報技術の発展」から，「大衆の，大衆による，大衆のための情報技術の発展」へと重心がシフトして行くことになろう．

伏見　論（情報数理研究所）
◇ モジュラリティ
◇ 開発環境データベース
◇ 高度ネットワーク

SE は，システム･アナリストまたはインテグレータになり，プログラマは，高度な技術を持つシステム･プログラマと，利用者環境カスタマイザ（人数はいらない）とに分かれる．いわゆる情報（ソフトウェア）産業がリーディング･インダストリーになるというのは誤りだと判明する．

北條　正顕（シスプラン）
◇ OOP（オブジェクト指向プログラミング）
◇ GUI の考慮
◇ サーバ／クライアント･モデル

90 年代後半は，ソフトウェアの自動作成技術が完成し，ほとんどすべてのアプリケーションは簡単にできあがってしまう．そんな中で「手作り実演ソフト」なるものに，ひそかな人気が出てくる．グルメ地図のように，「あそこのソフト屋の製品には年期を感じさせる旨味がある」とかいった情報が，ひそかに流される．そんなソフト屋に私はなりたい！

新妻　正夫（シスプラン）
◇ アート
◇ アマチュア化
◇ 分離と融合

後藤　浩（シスプラン）
◇ Smalltalk
◇ COBOL
◇ 私

稲垣　浩一（丸互）
◇ ネットワーク
◇ オブジェクト指向
◇ バックログ

ISDN の普及に合わせてネットワークが整備され，また，分散化が一層進む．そのため，ネットワーク関係のハード･ソフトを含めた SI の仕事ができる企業が生き延びて行くだろう．

小田　七生（西部ガス情報システム）
◇ オブジェクト指向
◇ LAN
◇ CASE

プログラマ，システム･エンジニアといったソフトウェア技術者は，今後，高度な開発支援ツールや環境を整備していく専門集団と，それらのツールを利用して，アプリケーションを開発する一般集団とに，２極化する．そして，後者の一般集団は，ソフトウェア技術者のみならず，最終利用者の層へと降りてくる．

そのために，90 年代には，専門家ではなく一般の人々がシステムを構築できるように，エキスパート･システムや AI 技術を応用したアプリケーション･ジェネレータ等が開発され，普及していく．データベースも，オブジェクト･データベース等，一般利用者に理解しやすいものが研究され，順次商品化されて行くだろう．

植田　憲巳（構造計画研究所）

◇ WSの一般化

一般的なソフトウェア開発では，CASEツールなどの発展にともない，ソフトウェアの専門家は不要になる．ソフトウェアハウスは，単純作業向けのマンパワーを提供するものと，ある専門分野に特化した知識集団的なものとに，2極分化する．

渡辺　雄一（電力計算センター）

◇ エキスパート・システム

◇ マルチパラダイム

◇ 並列処理の実用化

90年代後半には，馬鹿をやめさせるための圧力を社内教育のカリキュラムとして組み込む企業があらわれる一方で，そうした馬鹿を集めて土方仕事に専念させる会社ができる．当然，過去のソフトウェア，Unix System Vなどは，かれらによってROM化されることで命を長らえる．

川嶋　將男（富士通）

◇ Large Scale System Integration

◇ CASE-Tool (Including X-window)

◇ Hyper-Media Workstation

社会の安定を維持するためには，さまざまな業種においてNonstop Large Integrated Systemの開発に対するニーズが増加し，その開発のためのSEおよびintegratorの教育やCASE（上流～下流まで）への期待が高まる．HYPER-Media技術はWSとともに普及する．それにしても，他の産業界をも含めて，人材不足が激化するであろう．

武田　知久（日本システムサイエンス）

◇ 自動化

◇ 分業／分担開発（地方分散／海外との協力）

◇ 技術集約化

ボディ・ショップ（派遣業）は壊滅的に減少する．高い技術にもとづく発想が要求されるようになり，いまのように，だれでもプログラマにはなれなくなる．

寺井　良樹（ジャコス）

◇ （慢性的な，いや末期症状の）人手不足

◇ （特に，ハイレベル技術者の）老齢化

◇ 開発費用の増大

OSIの一般への普及は，メーカ同志のエゴのために21世紀にずれこむだろう．技術の飛躍的な向上はなく，頭打ち．相変わらず，紙と鉛筆で稼ぐ時代が続くだろう．ただし，リレーショナルDBの新製品は続々発表されると思う．

野口　典佳（ジャパンシステム）

◇ ネットワーク

◇ ユーザ・インタフェイス

◇ AI技術の一般化

ISDNなどにより社会のネットワーク化が進むが，アメリカと違って，ARPANETのように国レベルでの統一されたネットワークがないため，混乱状態に陥る．ソフトウェア開発の現場では，機種や，動作環境（OS, DB, 言語など）の壁を越えた統一的な開発支援環境（CASEその他）が求められる．

尾形　修一（大日本スクリーン製造）

◇ サテライト・オフィス

◇ ワークステーション

◇ 衛星通信

すぐれた開発環境，開発方法を実際にうまく運用する企業と，なかなか昔ながらのやり方から抜けられない企業とに色分けされるだろう．前者は，面白い仕事ができるが，後者はつらい仕事を続けることになる．ソフトウェアの限界に社会が気づき，ソフトウェアに過大な期待をしなくなるだろう．したがって，現在の予想に反し，ソフトウェア開発の仕事はそれほど増えない．90年代の終わり頃に情報化社会は破綻し，まったく別の価値観で社会が動きだすだろう．

大橋　三郎（リコー）

◇ CASEツール

◇ DB

◇ 4GL

ソフト開発環境にはAI技術も取り入れられ，機械化は一層進展する．ソフト産業全体は膨脹の一途．ただしソフトハウスは，マル金とマルビに2極化する．メインフレーム，オフコン，WS, パソコンなどのターゲット・マシンとは無関係に，ソフト開発のやり方は統一される．

久富　忠雄（CSK）

◇ オブジェクト指向

◇ ネットワーク

◇ 並列処理

ソフトウェア資産，データ資産があるため，メインフレーム上のアプリケーションは引続き残る．しかし，一方では，ワークステーションが多数導入され，それを前提としたアプリケーションがどんどん作られる．ツールの能力も向上するので，メインフレーム上の小規模なアプリケーションは，スクラップ & ビルドの形で，ワークステーションに移

される．ネットワーク環境を前提とした技術の研究が進む．

船切　誠（日本電子）

◇ OOP
◇ CASEの（本当の）活用
◇ 分業

「重要だ重要だ」といわれながら，ソフトウェア技術者の待遇は，（多くの場合）改善されない．「エセ·プログラマ」，「エセ·システム·エンジニア」と「本当の」人間の区別がつきにくいまま，しばらくは経過する．そのうち「本物」を抱える会社が他を引き離し，ソフト産業のリーダーとなる（神かホトケか....）．ただし，メーカ内の「ソフト部隊」は，あいかわらず虐げられる（暗い...!）．分散環境上での開発が認められ，さらに「分散されたシステム」をセールス·ポイントにした巨大システムが現れる．「生き物」に近い構成を持つシステムの予感（？）．

林　敏広（日本コンピュータ研究所）

◇ オブジェクト指向
◇ CASEツール
◇ 知識ベース

1990年代は1980年代より変動に富んだものとなるだろう．当然，分散開発(処理)がメインになる．人工知能(ファジィ，ニューロを含む)技術がある程度実用化されるにともなって，エキスパート·システムがあちこちで利用されるようになるだろう．ソフトウェア会社の数は，現在の１／３位になるのでは？　ソフトの生産性向上のために，CASEが大々的に導入される．それにともない，デマルコ etcの構造化分析／設計手法が日本でも普及し，５倍程度の生産性向上をもたらす．

筏井　幸夫（SRA）

◇ オブジェクト指向
◇ WS + LAN
◇ CASEツール／環境

３期に分けられると思う．第１期（1990～1993）は，「机の飾り」的にWSが普及する．第２期（1994～1996）は，開発方法論の見直しとその成果にもとづくCASEツール第２世代が出現する．第３期（1997～）にいたって，ソフトウェア技術／産業が劇的に変化する．メインフレーマの作るコンピュータから「汎用」の２文字が消え，パソコン－WS－ネットワークが，コンピュータ産業の中心になり，したがって，そのソフトを開発すること／活用することがソフトウェア産業の主流になる．

栗原　正利（SRA）

◇ ネットワーク
◇ マルチメディア

人月という工数カウント上の概念がなくなる．コンピュータが家庭に入り込み，ソフトウェアが一般化することにより，アホなエンジニアはすべて失業し，アホな会社はつぶれる．Σに代わって，こりもせずMac/NeXT/TRON等を持ち出して，Λ（ラムダ）プロジェクトができる．東京にオフィス税が適用され，ソフトウェア産業は，次々に東京都から脱出する．韓国／中国の労働力を利用した開発が増加し，ソフトウェアにも“Made in Japan”が明記されるようになり，外国で製造したソフトウェアとの価格差が発生するようになる．

山崎　康弘（蝶理情報システム）

◇ CASE
◇ LAN
◇ WS

1990年代後半から2000年にかけては，メーカよりもソフトハウスの方が，システム·ベンダーとしての主導権をとるのではないかと思う．メーカは自社のハードウェア中心の販売が中心であるが，ユーザが望んでいるのはマルチベンダのシステムであり，この際ユーザ視点に立って最も適格な提案やサービスができるのは，メーカに依存しない独立系ソフトハウスだと考えられるからである．

瀧本　隆一（両備システムズ）

◇ CASEツール
◇ 高性能ワークステーション
◇ LAN（Micro-Mainframe Linkを含む）

CASEツール，WS，LANをキーワードとして上げはしたが，以下のような問題がある．まず，CASEだが，米国製のツールが多く紹介されているが，われわれ日本人の開発方法（考え方）にはフィットしづらい，もっと日本的な発想のツールができるかどうかがキーとなる．次に，WSについては，互換性（過去の資産との），コスト／パフォーマンスが問題である．５０万円以下でハード，ソフトが（最近ソフトが高い）提供されることが望ましい．LANは，網間接続，設置の簡易化（コードレス），ソフト対応がポイントである．

白杵　敏雄（システムエイジ）

◇ 効率
◇ 品質

◇ 再利用

後半に入り，やっと労働集約からの脱皮のきびしさが見える．

椎熊 敏朗（日本データ通信）

◇ 総合的 CASE 環境（理想）

◇ 分散型チームによる開発（理想）

◇ いつまでもある古いもの（現実）

90 年代には，実用的なオブジェクト指向環境の研究／開発が進み，知識ベース技術，CASE，ネットワーク技術などと組み合わされて，これらが適用可能な分野（あるいは部門）では，従来とはかなり変わった開発サイクルが実現できる．しかしながら，ソフトウェア産業全体にそれらが普及するのは，21 世紀になってからであろう．この業界の技術移転の悪さをどうにかしないかぎり，ソフトウェア産業に大きな変化は期待できないでしょう．

礒村 克彦（三菱電機メカトロニクスソフトウェア）

◇ OS

◇ 言語

◇ ネットワーク

OS，言語，ネットワークといったいわゆる基本ソフトの分野では，アメリカの優位性はゆるぎないものだと思います．日本のソフトウェア技術者は，ほとんど基本ソフトのユーザであり，その上でせっせと生産性や品質の向上に励んでいるという構図は，変わらないのではないでしょうか．ただ，神奈川のあるソフト会社が発表した Lisp-C トランスレータのように，ときにはキラリと光るものも見られ，そう悲観することもないかな？ と考えたりします．

岡田 典久（住友電工）

◇ CASE

◇ 部品合成

技術的には大きくかわらない．ただ，新人が入ってきて CASE など導入しやすくなる．

横山 博司（SRA）

◇ ネットウェア

◇ フリー･エンジニア

◇ ハンドメイド･ソフト

現在のソフトウェア開発は，WS や PC に示されるように，以前に比べてエンジニア一人当たりのマシン･タイムを飛躍的に増大させ，しかもネットワークによる通信が発展してきたことで，人が時間をかけて移動するのではなく，データを瞬時に移動できるようになってきた．このことは，生産性向上の上で大いに役立ったことでしょう．

ただ，ソフトウェア開発が人の手から完全に離れた訳ではなく，あくまで道具が発展してきたとみるのが妥当でしょう．以前は，エンジニアがシステムに合わせて作ってきた環境が，いまではすでに用意されているわけで，それを使いこなせるか否かが，大きなポイントとなります．

しかし，現在のコンピュータはノイマン型であるため，人間とは疎外関係にあることは否定できません．なかには，取っ付けない人がいるでしょう．そこで，登場するのがファジィ･コンピュータです．より人間的な，曖昧な情報から，的確な回答を出してくれるわけですから，これ以上に頼もしいものはない．1990 年代は，ファジィ･コンピュータの時代です．

松村 好高（SRA）

◇ OOSD

◇ ピープルウェア

◇ ソフトウェア･メインテナンス

1990～1995 年を想定すると，1985～1990 年における金融第３次オンラインのような，全体主義的（ある一つの産業全体での）システム開発は少ないものと思われる．しかしながら，この種のソフトウェアの地方銀行への流布および WS ／ LAN のアプリケーション･システムの出現により，実在するソフトウェア･システムのメインテナンス負荷が増大し，第２次ソフトウェア危機が出現するものと思う．

山口 繁幸（エナジーサポート）

◇ CASE

◇ SIS

◇ ネットワーク

本当にどうなるんでしょうね．CASE とか分散環境とかいわれていますが，どういう形で具体化してくるのか，興味はつきません．ただ，70 年代に世に出たマイコンが 80 年代には，脚光を浴びその地位を不動のものとしたように，90 年代には，いま脚光を浴びつつある CASE や OOP なんてのが盛んになるんでしょうか？

佐藤 雄一（SRA）

◇ ワークステーション

◇ Unix

◇ 巨大化

私は，ソフトハウスやメーカ系ソフト会社の規模に興味をもってます．どんどん大きくなってきて，いまでは，何千人という単位はちっともめずらしくなくなりました．数を

頼みとする傾向は今後も続くでしょう．でも，そのうち必ずネットワーク社会が訪れると思います．そうしたら，独立した小さい組織がまだ増えるでしょう．今でも，優秀な人はどんどん独立していますが，その傾向に拍車がかかるでしょう．大規模ソフトハウスはどうなるのかな？（はきだめ化する？）というのが興味あるところです．

岡崎　博樹（武市コンサルティング・オフィス）

◇ OOP
◇ GNU プロジェクト
◇ Unix

分散型の開発環境が急速に普及して，個々の技術者にとっては仕事がやりやすくなる．

長井　修治（大和ソフトウェアリサーチ）

◇ オブジェクト指向
◇ ユーザ・インタフェイス
◇ 分散処理

ハードウェア面では，小型化，高速化，大容量化が一層進む（PC と WS の垣根がなくなる）．しかし，ソフトウェア面は混迷化し，新技術／技法／考え方の相つぐ登場に，ブーム現象が増大する（救世主の出現を望む？）．

島岡　宏光（NTT データ通信）

◇ オブジェクト指向
◇ CASE
◇ 4GL

わが社では，マルチベンダ・インテグレーションを唱えている．ソフトウェアの生産性向上の鍵は，「いかに作らないか，すなわち部品化／共通化をどう進めるか」にあると見るが，ターゲット・マシンを問わない部品化の実現に，非常な難しさを感じている．開発の流れのどこかでソフトウェア資源を蓄えなければならないと思うが，まったくの手探り段階だ．Σ計画は，全国規模でソフトウェア資源を流通させることを狙ったものだと思うが，資本主義の競争原理に逆行しているので，成果は絶対に普及するはずがない（全然予測になりませんでしたね）．

岡本　淳（CSK 総合研究所）

◇ GUI
◇ OOP
◇ CASE

CASE は「型」として普及する．すなわち「従弟制度」ではさばき切れない大量の新人が，最初におぼえる「型」として．これを破って独自の「型」を創れるのは，ごく少数のエンジニアしかいない．これは従来と変わらない．お絵描きツールの普及は，プレゼン（の資料作り）のうまいエンジニアを増やすだろう．そしてかれらが，CASE のトレーニングを担当するだろう．

松永　浩（日本銅管）

◇ 統合化 CASE
◇ 自動化
◇ AI − CASE

製作作業の空調化が進む．

大野　誠一（高崎共同計算センター）

◇ ユーザ・インタフェイス
◇ AI
◇ ネットワーク

開発環境が充実してくる．たとえば，サテライトオフィス，在宅勤務，地方への拠点展開（豪華な住環境を整備した），ツールの拡充，ハードの携帯小型化，etc,

蓜積　芳子（日本ビクター）

◇ ユーザ・インタフェイス
◇ オブジェクト指向
◇ ハイパーテキスト

趙　東一（シーイーシー）

◇ CASE（広い意味で）
◇ 国際分業
◇ 分散化

ソフトウェア産業は，３流の産業と見なされて，外国人労働者に依存する．また，中国やインドが下流の仕事を下請する．日本人はもはやプログラミングをしなくなる（大量の肉体派 SE やプログラマは失業する）．エンド・ユーザや業務担当者が直接にシステムを組み立てられるような開発環境や方法論ができる．

大橋　壮一（テックシステムズ）

◇ CASE
◇ ワークステーション
◇ オブジェクト指向

新しいツールへの適応性，積極性，技術力において差が開く，遅れた企業は競争に勝てない．開発／保守コストの消滅等，技術革新のできないソフトハウスは無用である．

岩渕　洋一（キャノン）

◇ 分散環境
◇ 上流工程

◇ オブジェクト指向

島村　直幸（大林組）

◇ ネットワーク
◇ CASE
◇ オブジェクト指向

製造業における自動化などの生産技術と同様な考え方が，ソフトウェア生産においても適用·浸透してゆき，ソフトウェア生産の工場化が図られる（べきである）．また，ユーザ·インタフェイスの標準化が行われ，ユーザ·サイドでは開発よりもアプリケーションの購入利用が進む．

小泉　毅（キャノン）

◇ 生産性
◇ 技術者教育
◇ 人事制度（生産分業，体系化）

CASE，オブジェクト指向など技術的な新技術は，それなりに発展して行くが，その導入技術，要員教育等は立ち遅れる．特に，日本の社会には，従弟制度的感覚が根強く残っており，ソフト産業のように方法論の変革が激しい業界には適していない．この部分がネックとなって，世にいうソフトウェア·クライシスが，日本にふたたび訪れる．

匿　名

◇ Object Oriented Software Development
◇ Multimedia & Hyper Text
◇ Visual Language

匿　名

◇ オブジェクト指向
◇ 協調型計算
◇ 分散環境

花石　昌文（富士通長野システムエンジニアリング）

◇ Unix
◇ ワークステーション
◇ ネットワーク

照井　拓（新日鉄情報通信システム）

◇ オブジェクト指向
◇ ネットワーク
◇ 分散処理

M&A 等による統廃合が進み，上位企業による寡占状態になる．

深瀬　弘恭（アスキー）

◇ Object Orientation
◇ Look & Feel
◇ Metropolitan Area Network

建設業界のような機能分化が起こるだろう．つまり，設計事務所，ゼネラル·コントラクタ，下請工務店等に細分化されて行く．特に，大多数のソフトウェア会社は，少数のゼネコンの下請となるだろう．

関根　英和（トヨコムシステムズ）

◇ CASE ツール
◇ 自動プログラミング
◇ 技術者不足

加藤　弘治（シーイーシー）

◇ 分散処理
◇ 自動化

ホスト集中化から WS 使用による EDP システムの分散化が進むであろう．当然，プログラマの仕事も現状とは変わり，ふつうのプログラムは，エンド·ユーザが自分の手で楽にできるようになると思われる．

黒田　清隆（三菱電機）

◇ 流用（再利用）
◇ プロトタイピング
◇ 技術の伝達

ソフトウェアの標準化が進行し，ソフトウェア企業の棲み分けが進む．エンドユーザ·コンピューティングの進行も目立つ．ただし，使用頻度がある程度あれば（性能上の問題から）現状と同様に開発されるので，開発の需要は減らない．プロトタイピング·シミュレーション技術の進展．

井上　浩人（富士ファコムソフトウェア）

◇ スキル
◇ 体制
◇ 創造

宇佐美　学（アンリツ）

◇ データベース
◇ ネットワーク
◇ オブジェクト指向

浦　直樹（横河電機）

◇ オブジェクト指向デザイン
◇ CASE ツール
◇ パーソナル WS

桜井　実（大塚システム研究所）

◇ Unix

◇ CASE
◇ OOSD

川下　敬之（横河電機）
◇ 分散化
◇ 信頼性向上
◇ ユーザ・インタフェイス

小高　信人（富士通エフ・アイ・ピー）
◇ オブジェクト指向
◇ CSCW
◇ CASE

ハードウェアおよびメーカ提供の開発支援ツールの急激な進歩により，できることの範囲は拡大するが，ソフトウェア産業の構造的問題（長時間労働，低賃金等）はますます深刻になる．

中村　昭雄（山武ハネウェル）
◇ CASE
◇ ネットワーク
◇ Unix

CASE を中心にソフトウェアの開発が行われるようになる．仕様記述言語が発達し，仕様を書けばプロトタイピングができる．あとは簡単な修正ですむ世界になると思う．

佐々木　敏夫（日本海事検定協会）
◇ CASE
◇ Σ計画
◇ 部品化

井戸　昌幸（中央コンピュータシステム）
◇ SIS
◇ CASE
◇ 情報システム

一般的にはあまり変わらない．しかし，ユーザとメーカが業務システム開発には手を結ぶので，ソフトウェア分野は高度な技術が必要となり，要員不足のために，業界全体としては，あまり進展しないと感じる．

林　　豊（中央コンピュータシステム）
◇ AI（エキスパート・システム）
◇ CG（コンピュータ・グラフィクス）
◇ 分散（在宅勤務）

独立系ソフトウェア会社が主導権を取る．

倉品　順誠（富士ファコムソフトウェア開発）
◇ ネットワーク
◇ オブジェクト指向
◇ CIM

高橋　敏彦（アンリツ）

今よりももっと分業・差別化，使いわけが進む．

中坊　徹（堀場製作所）
◇ 統合 CASE ツール
◇ チームウェア
◇ ハイパーメディア

ソフトウェア技術は，一層オブジェクト指向へ向かうと思う．ソフトウェア産業は，より人間の感性の領域に入り込んだものとなると思う．

石川　健一（富士ファコム制御）
◇ オブジェクト志向
◇ 並列処理
◇ 分散

アプリケーション・インタフェイスの標準化が進み，プログラマは楽になるようにも思えるが，実は楽にはならない．標準仕様がふくらんでゆき COBOL 化する．Unix はすそ野が広がるが，その分だけつまらなくなって行く．

相田　雅隆（ジェーエムエーシステムズ）
◇ オブジェクト指向
◇ ネットワーク
◇ ハイパーテキスト

大林　誠（シーイーシー）
◇ 生産性向上
◇ 人手不足

刀根　利光（シーイーシー）
◇ オートメーション
◇ オブジェクト指向
◇ ネットワーク

産業革命が必要と思われる．いつまでもマニュファクチャでは限界があるであろう．そのためにも CASE には期待しているが，本格的に使えるものはまだ先のような気がする．とりあえず楽ができるようになりたいものです．

松尾　好博（メルクス）
◇ ネットワーク
◇ CASE
◇ WS

ネットワーク利用の複雑化が進む．このため，ネットワーク管理技術およびセキュリティの重要性が高まると考

えられる．一方，ハードウェアの中に組み込まれたソフトウェア機能の高度化／多様化が進む（ロボット etc）と考えられる．

佐藤　大作（シーイーシー）
- ◇ CASE
- ◇ SI
- ◇ SDS

プログラミング工程の自動化に拍車がかかり，工業製品化が一層はっきりしてくると思われる．それとともに，ソフトウェア関連の企業形態も，保守を専門に行なう会社，設計を専門とする会社，開発を専門とする会社等と，分化する方向に行くのではないか？ネットワークについては，異機種互換という意味で，ハードウェア上の制約がなくなり，日本というエリアをこえて広く活用されて行くと思われる．

深瀬　満男（富士電機）
- ◇ CASE
- ◇ ADA

言語としては，ADA が主流となり，プログラミングの自動化が進んで，上流工程における分析・設計が，SE の主な仕事になる．また，その結果すなわち生産物も目に見える形になる．

安田　眞房（日本 IBM）
- ◇ Cross Life Cycle
- ◇ Information Model
- ◇ Medularity

プログラム仕様書の最も理解しやすく，誤解されにくい（？）形は，それが日本語（自然言語）で記述されていることだと思う．開発工程の上流工程から，下流工程まで「日本語ドキュメント」を正式なものとし，それからプログラムをジェネレートできるようになれば，大きな進歩だと思う．それには，上流工程における分析／設計の方法論，モデル化の方法論がきちんと整備され，提供されることが必要であろう．

濱窪　ひとし（日本ハイソフト）
- ◇ CASE
- ◇ SE 不足
- ◇ 需要とコストのアンマッチ

三神　敬（明電舎）
- ◇ オブジェクト指向
- ◇ 統合化 CASE
- ◇ AI

神戸　尚志（シャープ）
- ◇ CASE
- ◇ ユーザ・インタフェイス
- ◇ マルチメディア

マルチプロセッサ環境下でのソフトウェア開発が，科学技術部門を中心に最も重要になる．相変わらず，アメリカ主導でソフトウェア技術は進歩する．シグマが成功するかどうかは，政府が今後どれだけ力を入れるかによる．

森松　耕一（明電舎）
- ◇ CASE
- ◇ 業界標準
- ◇ 分散環境

UpperCASE と LowerCASE の両方がそろって進歩し，ライフサイクル全体を一貫して支援して行く方向になる．そのため，各種 CASE ツールの動作環境の標準化（欧米ではさかんに検討が進んでいるらしい）が重要となる．Σは，ツールのレパートリはあるが，統合化されていない（データ＝ DB の共通化がなされていない）ため，いまのままではメリットがあまりない．

吉岡　智司（横河ヒューレットパッカード）
- ◇ オブジェクト指向
- ◇ グループワーク
- ◇ 在宅勤務

アイデアをいかに表現するかという，上流工程におけるクリエイティブな課題を解決することが人間の仕事になる．下流工程については，Know How を共有化し，重複作業を避けるような仕掛けが確立され，テスティングに必要な環境も苦なく実現できるようになるでしょう．そうなると，エンジニアにとって一番時間がかかる仕事は，ミーティングになるハズです．

居　素直（ソニー）
- ◇ HyperCASE (CASE Tool, but something NEW)
- ◇ OOP
- ◇ 見境のない応用領域の拡大

真のダイナブック，つまりナレッジ・ナビゲータに近いパーソナルコンピュータが登場する．組み込み型マイコンの利用が，見境なく広がって行く（家電機器レベルのネットワークの実用化）．ソフトウェア工学の恩恵にあずかりにくい低レベルのアプリケーションが増大する．究極のコストダウン追求が求められるので，メモリー資源の小さいマ

イコンが相変わらずなくならない．

山田　光一（富士写真フィルム）
- ◇ CASE
- ◇ オブジェクト指向
- ◇ エキスパート・システム

ソフトウェア開発の定量化に関して，上流工程をも含めたメトリクスが定着する．方法論と CASE ツールを土台に，新しい徒弟制度が生まれる．

伊藤　宏二（松下システム）
- ◇ 分野別ソフトウェア・モデリング
- ◇ メッセージ・パッシング
- ◇ WS と開発環境

数字的な理論上の基礎を確立した手法が本命となるだろう（モデリング，上流検証，下流検証）．

斉藤　隆（アイネス）
- ◇ オブジェクト指向
- ◇ AI

井上　公夫（横河ヒューレット・パッカード）
- ◇ オブジェクト指向
- ◇ ワークステーション
- ◇ データベース

西尾　敏彦（シー・イー・シー）
- ◇ CASE
- ◇ OOSD
- ◇ ネットワーク

下流工程の自動化手段としては 4GL が普及し，それを用いて作成されたモジュールが流通する．将来のソフト開発作業は，実現方法を考えるまでが主な仕事で，以降は既存のモジュールの結合で何とかする．どうしてもないものだけを作成すればよい．

嶋　久登（ソニー）
- ◇ オブジェクト指向
- ◇ データ駆動
- ◇ ネットワーク

田中　正則（SRA）
- ◇ CASE（I－CASE）
- ◇ OOSD
- ◇ 統合化環境

人見　庸（ジェーエムエーシステムズ）
- ◇ ネットワーク
- ◇ 分散開発境界
- ◇ グループウェア

産学がより近づいて，どうなっていくのかなぁ？

吉沢　浩宣（平田機工）
- ◇ CASE
- ◇ AI
- ◇ EWS

90 年代は，オブジェクト指向××と名付けられたツールや方法論が一種のハヤリとして普及するだろう．また WS の普及にともない，（ある程度効率をギセイにしても）自動プログラム・ジェネレータが使われるようになるであろう．

岡田　秋良（平田機工）
- ◇ オブジェクト指向
- ◇ CASE ツール
- ◇ データベース

ソフト開発はペーパーレスに近づくと思うが，ドロくさい作業はまだまだ残るでしょう．

中川　徹（平田機工）
- ◇ トポロジー
- ◇ プログレス
- ◇ エクステンション

中島　英一（昭和電工コンピュータサービス）
- ◇ CIM
- ◇ UpperCASE
- ◇ 分散環境

分散環境（WS／ミニコン／オフコン ＋ ネットワーク）による分散型のアプリケーション，およびエンドユーザ・コンピューティングが本格化する．しかし，ソフトウェア開発の現場では，分散化があまり進まない．

上川　清士（CSK）
- ◇ AI
- ◇ ファジィ
- ◇ 通信

コンピュータがもっと人間に近くなってくる．たとえば，現在はソフトウェア開発に欠かせないキーボードから，プログラマやオペレータが開放され，言語（音声）による入出力が主流になり，五感を持ったコンピュータが登場してくる．第１ステップとしては，そういったコンピュータ向けのソフトウェア技術が芽を出し，第２段階で，応用技術（ソ

フトウェア）が爆発的に市場に出てくるものと，予想する．

野口　悟（ジェーエムエーシステムズ）

◇ CASE
◇ SI の中断
◇ CIM

産業構造に変化が起こり，製造程過のドーナツ化が推進され，開発の下請けとして東南アジア圏を巻き込むことになる．コンピュータ革命の兆しとして，ファジィ・コンピュータの実用化が，部分的にスタートする．

田村　整（小松ソフトウェア開発）

◇ RISC
◇ Smalltalk

Smalltalk 等のオブジェクト指向言語が普及し，RISC マシンにより大幅なコストダウンが達成される．WS の価格は，５年後に現在（80 年代後半）の半分以下（1/4）となり，本当の意味で１人１台（２台も？）の環境が実現するはず！

腰原　貞利（富士通エフ・アイ・ピー）

◇ CASE
◇ 分散処理（Unix，LAN，ISDN）
◇ リバース・エンジニアリング

WS の高性能化，価格低下による１人１台の実現，LAN や ISDN などネットワークの充実，高生産性ツールの普及などといった開発環境整備の促進により，90 年後半には，開発の方法が大きく変わると思う．すなわち，プログラムやドキュメントなどの再利用（流用）が一般的になり，また，分散開発が大きく進展する．一方，リバース・エンジニアリングが進み，保守の負荷は大幅に軽減される．SE の分業化がより明確になり，プランニングや分析を専門とする現在のコンサルティング会社のような企業がかなりふえる．

栗田　憲芳（東芝）

◇ WS
◇ CASE
◇ 分散コンピューティング環境

より強力で小型化されかつ安価な WS が登場・普及し，１人１台の時代が到来する．つまり，分散コンピューティング環境が中心になるでしょう．また，若者がソフトウェア関係の職種に就職したがらなくなり（現在でもすでに人気がない），ソフトウェア・エンジニア不足はもっときびしくなるでしょう．そして，開発の生産向上のために，より強力な CASE ツールが登場し，ある種のソフトウェアは，技術者なしでも，そうした CASE ツールを駆使して設計・作成がなされるようになるでしょう....?

山田　達哉（ワイズシステム研究所）

◇ Unix
◇ オブジェクト指向
◇ AI

ソフトウェアの開発ツールや環境がかなり整備され，とくに AI 的な技術が，開発を支援するためのツールに利用されるだろう．これによって，ソフトウェアによるソフトウェアの自動生成ということが，ある意味で現実になると思われる．したがって，そのツールを持てば，だれでもプログラマになれるわけで，プログラマという職業はなくなってしまうかも知れない．あるいは，ナレッジ・エンジニアをプログラマと呼ぶことになるだろう．いずれにせよ，ソフトウェア開発は，本来の知識集約型産業に変わって行く．

楊　啓廷（SRA）

◇ オブジェクト指向
◇ RISC
◇ ネットワーク

日本でも大規模なネットワークが整備される．高速な RISC WS の普及によって，開発環境における貧富の差が一層広がる．強力なツールが使えるようになるが，生産性はそれほど向上しない．ソフトウェア産業は成熟期に入り，業界の再編成が行われる．

上田　敏雄（中日電子）

◇ オブジェクト指向
◇ ネットワーク
◇ パーソナル Unix WS

ソフトウェア開発のプラットホームとしては，Unix が標準となり，MS-DOS からの移行がかなり進む．オブジェクト指向プログラミングが常識となる．ネットワークに弱い OS は，だんだん使われなくなる．Unix 上でも，グラフィックなユーザ・インタフェイスが普及する．C++ や Lisp が普及する．アセンブラの利用はだんだん減って行く．とにかく，90 年代に，ソフトウェア技術はかなりの速度で変化すると思う．

神原　貞夫（富士電機）

◇ ヒューマン・マネジメント
◇ 個人契約
◇ 開発の総合環境

上級 SE や優秀なマネージャのヘッド・ハンティングが盛んになる．ソフトウェア技術者相手のストレス・マネジ

メントのビジネスができる．ソフトウェア産業は，大きな転換点を迎える．従来の量的拡大路線ではなく，質的変換が要求される．国際的な提携（特に東南アジア）が活発となる．

木全　洋（ビクターデータシステムズ）

◇ オブジェクト指向
◇ CASE
◇ ネットワーク

ソフトウェア開発の形態は，CASE ツールを利用してオブジェクト指向プログラミングを実践するといったように，いままでとは変わってくる．ワークステーション中心，ソフトウェア重視の企業が伸びる．

咲間　繁（三菱電機）

◇ CASE
◇ データベース
◇ イーサネット

199? 年某社の情報システム部の状況： ソフトウェアを開発（コーディング，テスト）することをすっかり忘れ，どこの会社のどのパッケージをどう利用して満足のゆくシステムを組み立てるかが，SE の技術力になっている（ドキュメントの項目に「利用ソフト一覧」が追加されている）．一方，ソフト会社の方では，パッケージのライセンス料収入が，全売上の大半を占めている．

横田　秀明（富士総合研究所）

◇ Multimedia
◇ Hyper Text
◇ CG (3D Realtime)

LAN， WAN による分散環境の実現．E-Mail の普及．スーパーコンはネットワーク内の高速演算サーバになりさがっている．出版物の CD-ROM 化．ソフトウェアの本質的な生産性は，さほど向上しないのではないかと思う（文芸的な意味で，工業的な意味では話が違うが）．

前田明道（住友金属システム開発）

◇ 人
◇ 金
◇ 技

成沢　知子（住友金属工業）

◇ 美楽（美しく楽しく）： ビジュアル化が進み，見て美しくそして楽しく仕事ができる．
◇ 悠（ゆったり）： コンパクトで高性能の WS が出回って，ゆったり仕事ができる．
◇ 易（やさしく）： プログラムが簡単に組み立てられる．またはプログラムなんか作らない．

少なくとも 80 年代は，「過去の資産はどうしましょう？捨てる訳にはいかないし，かといって作り直すのも大変」という時代だった．90 年代は，もっとよいものを手っ取り早く作ることができるようになって，古いシステムは惜しげもなくゴミ箱へ捨て去ることができるようになっている．

ディスプレイがブラウン管でなく，カラー発光ダイオードを使った板状のものになり，いまの EWS はノート型の PC 並みの大きさになる．パッケージ･ソフトがかなり流通するようになる．すべての分野において，（著作権の関係で）1 からソフトを開発できるところは益々減る方向に....

藤間　真（日本ビジネスオートメーション）

◇ Unix
◇ GUI
◇ 下流工程の機械化（コーダーが不要となる）

開発環境としての Unix はますます広がってゆくが，GUI にはついて行けなくなる．オフコンはパソコンに駆逐され，パソコン上で専用ソフトを走らせるような形態になって行く．パソコン･ソフトなど，一般の利用者向けのソフトは，マッキントッシュ風の GUI を重視したものになってゆき，MS-DOS ではついて来れなくなる．新しい銀の弾丸は，少なくとも 90 年代前半には発見されないが，これまでに考えられた各種の弾丸を乱射する機関銃はできるので，下流工程の機械化が進む．

新田　稔（SRA）

◇ ナレッジ･ナビゲーション
◇ ファジィ
◇ フォーマル･アプローチ

ソフトウェアハウスは人手不足から，かなりひどい技術の低下をおこす．ユーザ企業は，蓄積したノウハウでソフトウェア業界に進出して来るが，認識不足から痛い目に合う．ソフトウェア工学は，よくわからないが「トレンディ」な言葉を４つぐらいは絞り出す．ハードウェアに関して，「より速く，大きく（メモリ空間が），安価になる」程度の予測しか立てられない人たちは死んでしまう．

斎藤　末広（テスク）

◇ CASE
◇ オープン･アーキテクチャ
◇ マルチメディア

SEAとして，再度Σに協力することに意義があると思う．

駒井　孝志（三井銀ソフトウェアサービス）

◇ オブジェクト指向
◇ モデリング
◇ バイオロジー／エコロジー（生態系）

コンピュータ・アプリケーションは，WSの普及にともない，ますます分散型に移行していくと思うが，それにともなって，ますますシステムの管理，メンテナンスの負担が増大して行くと思われる．一方，システム開発に関する技術やノウハウは，まだきっちり知識ベース化されておらず，依然として職人芸の域を出ていない．また，プロジェクト管理のための統一的な手法も，いまだに確立されておらず，業界自体は，今後５年はさらに「乱世」の様相を呈して行くと思われる．

その中で５年～１０年という，この業界にとっては長期的なレンジで，ソフトウェアの生産管理，品質管理を追求して行く企業だけが生き残って行くと思われる．そうしたソフトウェアの真の工業化のカギは，オブジェクト指向さらには自然生態系の中に隠されている．

杉田　義明（SRA）

◇ 分散開発環境
◇ ユーザ・インタフェイス
◇ CASE

CASEがプログラマのためのワープロとして定着し，すべてのプロダクトがワークステーション上のCASEを用いて作られるようになるだろう．しかしながら，依然として人材不足の状態が続き，要員養成のために，街にはCASE教室が従来のマイコン／ワープロ教室に代わって目につくようになる．

青島　茂（日本サンマイクロシステムズ）

◇ CSCW (Groupware)
◇ Networked Software Environment
◇ CASE

Programming by Example, Programming through Picture, Programming with Teamということになったらいいなぁ．

池田　誠（日本ネットワーク研究所）

◇ Unix
◇ オブジェクト指向
◇ 国際化

ソフトウェア技術はハードウェア技術と比べてあまり変わりばえせず，労働集約的・非人間的な状態が続くのではないか．

落水　浩一郎（静岡大学）

◇ ラピッド・プロトタイピングの普及
◇ CSCW
◇ ソフトウェア・プロセス

Dirty Workのイメージから脱却した企業のみが生き残る．あとは，しだいに人が集まらなくなり，死ぬ．

佐藤　千明（長野県協同電算）

◇ 分散データベース
◇ エンドユーザ・コンピューティング
◇ CASE

SIサービスを行なう会社とコーディング受託会社に２極分化して行く．ハードおよびソフト技術全般に長けた技術者，または，特定業界のアプリケーションに長けた技術者のみが，生き延びられる．メインフレーム中心の考え方から，ネットワーク中心の考え方に変化する．

鵜谷　章（PFU & 富士通）

◇ オブジェクト指向技術（OOA, OOD, OOPL）
◇ HyperMedia
◇ 知的支援（Intelligence Augmentation）システム

新しいアプリレーションがソフトウェア産業の最重要課題となる．したがって，

メーカ→ソフトハウス→ユーザ

というこれまでの仕事の流れが逆になる．そこで重要になる技術は，ドメインに依存したモデル記述，知的活動の支援ツール，プロトタイピング，ビジュアル化，マルチメデュアなどである．コンピュータに関するメタフォアまたはパラダイムが革命的に変わる時代になる．

SEA 秋のセミナーウィーク '89
Session C4

KJ 法を活用した要求分析支援環境

講　師：大岩　元（豊橋技術科学大学）
報告者：逸見　研一（CSK 総合研究所）

1. はじめに

今日は，ソフトウェアの要求分析や基本設計について，私の考えていることをお話しします．話の途中でも御意見や質問がありましたら，随時割り込んでください．

設計支援のための CASE 環境が最近話題になっています．設計作業には，仕様書など各種ドキュメントの作成がつきものであり，こうした作業をコンピュータで支援することが重要だというわけです．ふつう，これらのドキュメントはワープロで作られていますが，ワープロは清書です．つまり，設計それ自体ではなくて，その後始末の仕事を支援していることになります．

私どもの大学では，そういった後始末ではなく，設計作業それ自体にどのようにコンピュータが関与できるかという課題に，取り組んできました．

2. 要求分析が困難な理由

設計工程のさらに上流には，要求分析という工程があり，ここが非常にむずかしい．ソフトウェアの生産性が悪いとよくいわれますが，その大半の原因は，この要求分析にあるように思います．解法を考える以前に，解決すべき問題が何かを明らかにすることが，むずかしいのです．

その理由として，SE には業務上の知識がなく，一方ユーザ側にはコンピュータのことがよく分かっていないために，コミュニケーションがうまく行かないのだということが，よくいわれます．近頃では，さらに，ソフトウェア技術者の側にコンピュータの知識が不足している場合があるようにも，見受けられます（笑い）．

いま，とにかく SE が不足していて，それをどう育成するかが大問題です．一方，プログラマはあまっているといわれています．そのため，SE 教育のカリキュラムでは，プログラミング技法とか，コンピュータのアーキテクチャとかは教える必要がない．とにかく早く SE が育てばよいということで，促成栽培が行われています．こうして，コンピュータのことをよく知らない SE が量産されつつあるのです．

私は 47 才になりますが，私の年代の SE は，昔はソフトの値段がタダだったせいか，システム分析や設計だけでなく，コーディングまで引き受けて，コンピュータ室の中に入り込んで仕事をしていました．しかし，ソフトとハードの価格分離がなされてからは，SE がコードまで書いていたらソフトの値段が高くなってしまうので，プログラミングなどという下賤な仕事はしない（笑い）というのが，当たり前になってしまいました．

それはそれでいいのですが，プログラミングの経験がまったくないというのは，問題です．そういう SE は，要求分析技法やツールの使い方は知っていても，自分が書いた仕様書がどうやって具体化されて行くかというプロセスに関して何の知識も持っていない．そういう人間が，これまた専門外の業務の話を聞いて，何かデッチあげるのですから，いい仕様ができたら不思議というのが，ソフトウェア・ビジネスの現状なのです．

3. 要求分析はボトムアップ

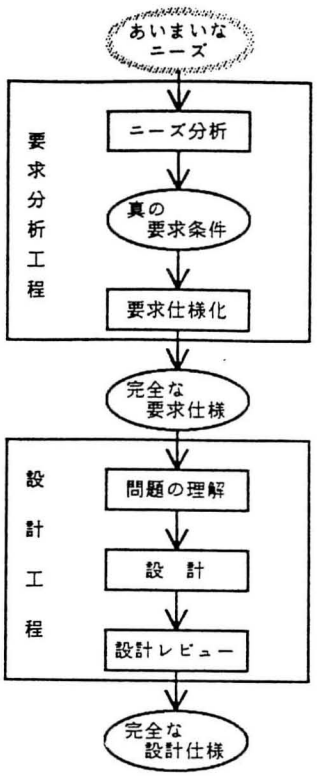
要求分析フェイズの出発点には，必ずしもコンピュータに明るいとは限らないユーザーの，これからコンピュータにこんなことをさせてみたい，というバラ色の夢があります．そして，その夢の中味は，絶対実現しなければならないという強い要求と，実現できたらうれしい程度の弱い希望とが，ごちゃまぜになっています．

一方，開発者側からすれば，実現にあたってどんな制約があるのか，利用可能な技術として何があるのか，などといった要因をいろいろ考えなければなりません．

要求分析の過程では，このようなニーズや制約をしっかり分析して，ほんとうに何を作るのかが決められるのです．この作業は，まずはじめに，個々の要求項目の相互関連はひとまずおいて，とにかくみんなが持っている要求をすべて吐き出し，次にそれをまとめる，といった具合にボトムアップに行います．

要求分析がちゃんと完了すると，問題がちゃんと把握されたことになりますから，今度はトップダウンに設計を行います．設計が完了すれば，次にそのレビューをして，レビューが完了すれば，完全な設計仕様が得られ，コーディング工程に渡せるわけです．

この最初の要求分析を，何だかよくわからない形で適当にすませて突っ走ると，そのうち問題が露見して，また初めに戻らざるを得なくなる，ということがよく起こります．そして，これがソフトウェアの生産性悪化の大きな原因になっています．



ソフトウェア基本設計の工程

4. KJ 法について

KJ 法では，思いついたことをカードに書いて机の上に並べます．このとき１枚のカードには１つのことしか書きません．

次に，こうして作ったカードを，相互に関連のあるものを近傍に集めることによって，グループ分けします．一応のグループ化ができたら，それぞれのグループに対して，そのグループを代表するカードを書きます．次は，このカードのグループを相互の関連によってまとめる．それを繰り返して行けば展開図が得られるので，これをチャートにします．

この操作を A 型図解化といいます．この図解化によってカード相互の関係が視覚化されます．こうして得られた２次元表現は非常に分かりやすいものになります．こうした点が KJ 法の意義のあるところなのです．

この次にあるのが B 型文章化というプロセスでして，先ほどのチャートを文章化します．こうするとチャートのおかしなところに気がついたりして，A 型図解化，B 型文章化のプロセスを行ったりきたりということになります．

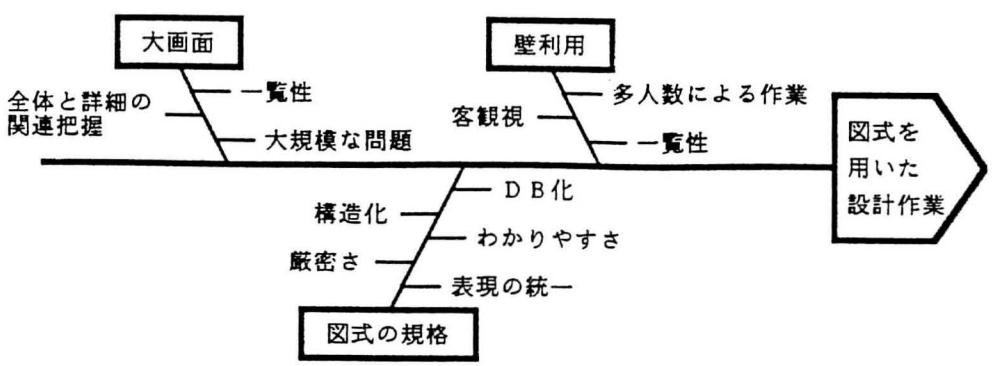
ここの，１つの事柄を２つの表現で見てみると対象が見えてくる，というところがよいのです．ソフトウェアの場合には，テキストを書くというところが，仕様書を作成することに対応するのですが，ここのところは HCP チャートに置き換えてもいいわけです．

5. HCP チャートで処理の目的を記述

数ある設計図法のなかから HCP チャートを選んだことには，理由が２つあります．

１つには，KJ 法との親和性がいいように思えたからです．

また，チャート形式の設計図法には，PAD とか SPD とか他にもいろいろあるわけですけれども，これらはみな，処



図式を用いた設計作業の特性要因

理内容を２次元的に分かりやすく表現するというところから出発しています．

しかし，HCP チャートはこれらと違って，いま何をしたいのかという処理の目的をまず書いて，ここから，その目的を実現するためには何をするか，というぐあいに分解して行きます．この分解をドンドン繰り返して，最後にはコードにもっていくのです．

ところで，他の設計図法では，こういう情報は，処理内容を人間の言葉で書き換えただけですから，本来同じ性質の情報なのです．しかも，コンピュータのコードと人間の言葉で書いたものとでは，人間の言葉で書いたものの方が不完全です．

一方，処理の目的とプログラムとは，性質が全然異なります．

たとえば，保守のことを考えてみると，プログラムの目的があらかじめ理解できていれば，そのコードを読む作業は非常に楽になりますが，逆に，コードのみが与えられてそこから目的を読みとるのは，簡単な仕事ではありません．要するに，目的さえちゃんと書かれていれば事足りるのです．

このように，トップダウンに目標を把握し設計できる点が，HCP チャートが他の設計図法と違うところです．だが，これを作成することは非常にむずかしく，そのためになかなか普及しないようです．私のところでも HCP チャートを学生に書かせてみたのですけれど，なかなか書けない，なぜかというと，どうも自分の作っているプログラムの目的を把握して，それを展開するということが大変むずかしいようです．

そこで，どうやって目的を分析するか，というところから考えなければならなくなりました．

7. KJ 法エディタ

では，具体的に KJ 法をどう応用するかをお話しましょう．

まず最初に思いつくことをカードにします．そしてできあがったカードを机の上に広げます．

私どもの作ったエディタでは，こんな具合にカードができて，カードを並べることができます．最初はこうして並べまして，それを次々にグループにまとめていきます．

グループ化をどうするかといいますと，同じようなものを同じような場所に置きます，重ねることもできます．重ねたカードの下側は見えなくなりますから，下のカードを上に出したりとか，カードを動かすとかは，みんなマウス操作でできるようになっています．

カードがまとまったところで，その回りをぐるっと囲んで線を引いて，システムにこれはグループだよと教えます．さらにこのようなグループに対して，囲んだ線をマウスでピックアップして，グループ全体を動かすことができるようになっています．

B 型文章化のフェィズでは，このチャートを見ながら文章を書くために，編集機能のついたウインドゥを用意しました．このウインドゥを画面上に切って，その中でチャートを見ながら，文章を作成して行くことができるようになっています．

＜聴講者よりの質問＞

Q： 大画面のワークステーションをもってしても，一度に表示できるカードというのは，たかが知れたものだと思います．そういうことは問題とはならないのでしょうか．

A： まさにご指摘の点が大問題でありまして，どういう具合に，小さな画面で，机の上全体のようなものを実現するか，ということをこれからお話し致します．

7. ディスプレイの広さの制約への対応

私のところは PC-9800 ベースで作業をしております．ディスプレイは 640*400 ぐらいの分解能しかなく，広さの制約は EWS より一層きびしい．そこで，私どもは次のような枠組みでこれに対処しました．

まず２種類の画面を用意します．１つはユニバーサル画面と称しまして，これはいわば論理的な大きさ 200＊150 キャラクタの机でして，この上にカードを配置します．どこにどの大きさのカードを作るかということを，マウス操作で実行することができます．この画面では，個々のカードに何が書かれているかは示されず，カードの配置が全体でどうなっているかいうことだけが，判読できるようになっています．

また，これとは別にローカル画面という，論理的な大きさが 40*20 キャラクタの画面を用意します．この画面にはディスプレイ上に細部までを表示します．もちろん，ローカル画面の領域は，ユニバーサル画面の中で，上下左右に動かすことができるようにしています．

さて，われわれは，机上にカードが並んでいるのを見るとき，同時にすべてのカードが見えていますが，同時にすべてが読めるのではありません．ちょうど自分の視線の先しか読めないのです．この見える視野に対応するのがユニバーサル画面，読める視野に対応するのがローカル画面ということになります．われわれのシステムでは，ディスプレイ

にこの2つの画面を重ねて表示するようにしています．

この枠組みのもとでは，カードをマウスで選択してローカル画面の外に出すようなときに，マウスの動きに応じてローカル画面が動かなければいけません．このためには，いろいろな方法が考えられます．

よくあるように，ウインドゥにスクロール·バーをつけて対処する方法では，画面の制御と中のカードの制御が分離されてしまいますので，操作性がよくありません．そこで，われわれは，カードをピックアップした状態で，マウス·カーソルがローカル画面の端を切ったら，それにつれてローカル画面の領域が移動するようにしました．

これだと，ローカル画面の領域を超えてカードを動かす時に，カードをピックアップしてカードを動かすと，カードにつられてローカル画面が移動するということになります．

私どものシステムでは，この操作を高速に行えるように設計を工夫して，操作性の大変よいものを実現しました．

8. KJ法エディタはほんとうに役に立つか

次に，これがほんとうに役に立つか，ということを議論します．結論を先に申しますと，残念ながら，やっぱり机の上で作業したほうがいい，ということになります．

そこで，こういうカードを動かすとか並べ変えるとかいう機械的作業にかかる時間を，机の上でするのと，われわれのエディタの上でするのとを測定して，比較してみました．理屈の上では，人間が視線を動かすのと同じような感覚でローカル画面が動くハズなので，同程度の作業効率を期待していたのですけれども，ローカル画面を動かさずにすむ作業はよかったのですが，ローカル画面を動かす作業では，作業効率が6倍程悪くなることが判明しました．

これはなぜかといえば，ローカル画面を動かすと，その上にあるカードのイメージが非常に速く移動します．これはちょうど，動いている電車の車窓からすぐ近くの広告をみるようなものです．われわれの目は，このように視野の中のものが非常に高速に動くことに，耐えられないのです．残像のためにちゃんと見えないし，不快感があったりするのです．

では，どうして机の上でちゃんと視線を動かせるのかというと，実は人間の目には，サイケディック·サプレッションという機能があって，目玉を動かしたり首を動かしたりして視線を高速に移動させたときには，始点と終点ではちゃんと見えているんですけれども，途中の過程では盲になるのです．この機構のために，われわれは，残像に悩まされることなく，机の上をランダム·スキャンすることができるのですが，そういうことがわれわれのエディタの上ではできない．あるいは，マウスを動かしているときだけ目をつぶる，という訓練をすればよいのかも知れませんが....(笑い)．

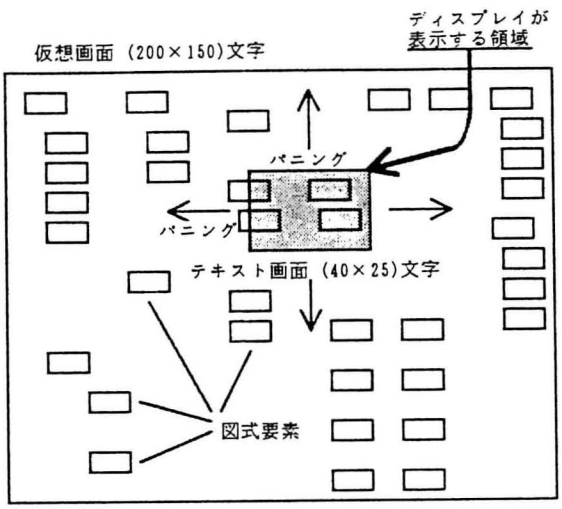
このような理由で，スキーマ自身はよさそうに思えたのですけれど，実際やってみると，あまり役に立ちませんでした．ただし，今のところカードを並べてその配置を変えられるエディタは，他に世の中にないので，そういう点では役に立つものだといえましょう．

先の話をしますと，創造的な仕事のための道具としては問題がありますが，きれいな設計図面を作ることができるので，そういう意味では役に立つだろうと思いまして，これをもう少し本格的なワークステーション上で，やり直しているところです．

＜聴講者よりの質問＞

Q： KJ法の先ほどの御説明のなかで，個人レベルで「思いだし」をして，集団レベルでブレーン·ストーミングでまとめあげていくよ，というお話でした．が，実用レベルでの，たとえばわれわれのシステム開発における要求分析フェイズでは，関連部門の人が集まって，ワイワイガヤガヤやってまとめて行くのがふつうです．そういう集団レベルでの使い方は，どういう感じになるのでしょうか．

A： 個人で行なう場合は，思いついたことをメモしていって，それがまとまったところですればいいのですが，集団で行なう場合には，会議をして，このとき，書記をひとり決めておいて，書記にはカード作りに専念してもらい，同時に録音も取っておく，というぐあいにします．そして，あとで書記が録音を聞きながらカードを整理します．



仮想画面とテキスト画面

カードの作り方なのですが，はじめからこういう KJ 法エディタの上でカードを作るのは能率が悪いので，ふつうのエディタの上で文章にします．私どものシステムでは，エディタ上の文章をカードに流し込む機能が備わっています．また，物理的なカードを印刷する機能がありますから，印刷して，物理的なカードを作って，ふつうの机の上で KJ 法を行なうことができます．

さて，１時間も本気で議論すれば，すぐ 500 枚位のカードができてしまいます．しかし，500 枚のカードでまじめに KJ 法を行なうのはとても大変です．これを簡略化する技法がいろいろ知られていて，大体，30～100 枚ぐらいにカードを減らしてから行うのが実際的です．

集団の場合には，カードを物理的に作って，壁にペタペタと張りながら，みんなで眺めて作業するのが一番いい方法だと思います．大体の大枠ができたところで，それを KJ 法エディタにほうりこめば，最後の詰めがちゃんとできるでしょう．しかし，残念ながら，そのような場合でも，100 枚を越すカードの全部を見ながら作業をするのは，このフレームワークでは，非常に効率が悪いように思われます．

話は変わりますが，このごろ米国ではグループウェアというのがはやっているようですが，それとわれわれが考えているものは非常に違うと思います．アメリカの仕事のやり方は日本と違って，組織としてもトップダウンなのです．アメリカでは，チーフ･プログラマーというのがいて，チームのメンバー各人のやる仕事を非常に明確に決めてしまう．そこのところをどうやって能率よくやるかが，この頃アメリカで流行っているグループウェアというものなのです．

日本ではこのプロセスが逆なのです．たとえば KJ 法がそうなのだけれど，「衆知を集める」というか，みんなでワイワイやりながら決めていこうという，ボトムアップなプロセスになるのです．

このボトムアップのプロセスにトップダウンのツールを使ったりすると，きっと悲劇的なことになるでしょう．だから，アメリカで作られた CASE ツールをそのまま日本で使うのは，危険なことだと思います．

9. 図化の御利益

さて，図にしたときに，何がありがたいかといえば，たとえばこの HCP 図は，A3 版１枚で全体が即座に見渡せます．しかし，これを文章表現にしたら，恐らく厚さ数 cm の量になってしまうでしょう．チャートならば，見る人が見れば，それがダメなチャートかどうかは即座にわかります．文章では，こういうわけにはいきません．

これが２次元情報と１次元情報の本質的な差なのです．２次元情報は，全体像の把握に役に立ちます．そして，全体像を把握することは，ソフトウェアを作るさいに重要です．そういう点で，KJ 法の Ａ型図解化は役に立ちます．KJ 法は，１次元の文字情報と２次元的な図形情報を非常にうまく組み合わせる技術で，このような技術はこれから重要になってくるだろうと思われます．KJ 法は大変日本的な手段なのですけれど，多分，人類の英知として世界中に広まって行くのではないかと思います．

10. フレンド２１に対するドン･ノーマンの見解

通産省は，マンマシンの問題が今後コンピュータの中心課題になるだろうと考えています．こうして，第５世代コンピュータの次のプロジェクトとして去年から６年間１３０億円かけて２１世紀のマン･マシン･インタフェイスを考える「フレンド２１」という研究が組織されました．

今月の初めに，このフレンド２１が，京王プラザホテルで，世界中のヒューマン･インタフェイスの研究者を集めて，プレゼンテーションを行いました．フレンド２１では，「メタウェア」(これは，メタフォアつまり比喩から作った新造語です) を中心概念に据えて，だれでも，いつでも，どこでも使えるコンピュータを，２１世紀に向けて作るんだという目標がぶちあげられました．

このとき，認知工学の提唱者である UCSD のドン･ノーマンが，これに対して，「ヒューマン･インタフェイスを研究するといっているけれど，ここには１人も認知科学の研究者が入っていないじゃないか，これはひどい」といういい方をしていました．たしかに，日本では，技術者のヒューマン･インタフェイスに対する関心が薄く，機能さえあればそれでよし，それがほんとうに使えるかどうか，という人間側からの議論が軽視されるようです．

ドン･ノーマンはまたこうもいいました．「フレンド２１では，ヒューマン･インタフェイスについて，まちがった問題を解いている，しかし，かれらは日本人だから完全にそれを解決するだろう」(笑い)．

たしかに，日本人は要求分析をきちんとしないまま物を作っています．そこのところが，ソフトウェアの場合にいろいろの問題を引き起こしているのです．

目的がはっきりしていて，はじめて，それに対してコンピュータを使うべきか，あるいは従来の手作業のほうがいいのか，とかいった判断ができるのです．ところが，そういう目的なしに物を作ることしか考えない人は，「これまでスタンド･アロンでやってきたのだから，今度はネットワーク

にしましょう」という発想になってしまう．

そうではなくて，目的をはっきりさせて初めて，そこはコンピュータを使うべきなのか，どうなのか，どういう風にすれば，全体としての作業が快適になるのか，そういうことがわかるようになってくるのです．そのあたりが，日本の産業界が遅れている所で，今後は問題になってくると思います．

ドン・ノーマンは，「日本のヒューマン・インタフェイス設計技術はひどいものだ．少なくとも，ヨーロッパやアメリカではもうチョットまじめに研究しているから，まあ１０年ぐらいは安心していられるなあ」という毒舌を吐いて，帰って行きました．

われわれは，今後，人間の問題をもっといろいろ研究しなければならない，と思います．

11．生産性向上はブラインド・タッチから

ソフトウェア・ツールを利用するにあたっての問題は，金を出せばツールは手に入いりますが，それを活用することには，まだかなり距離があります．ツールのメリットを出すためには，入力装置，ことにキーボードを使いこなすことが必要でありましょう．この中でブラインド・タッチのできる方はどのくらいいらっしゃいますか？　そうですか，たった３人ですか....

キーボードは１００年程前アメリカで作られ，当初は字をきれいに印字するための清書の道具だったのです．ところが，指を９本使って手を見ずに打てることが次第にわかってきて，今世紀に入ったころには，大体プロのタイピストにはブラインド・タッチが当り前になりました．手を見ないで打つようになると，手のコントロールを意識せずに仕事ができるようになり，スピードのほうも，喋るのに近い早さでできるようになります．

そこで，問題は，ブラインド・タッチをおぼえるのがどのくらい大変か，ということになります．アメリカのタイピストは学校で３ヶ月のトレーニングを受けます．喋るのと同じようにタイプを打つには，これに加えて大体千時間程度の実務経験が必要だといわれています．なにごとも，無意識にできるようになるには，それくらいの時間はかかるでしょう．

では，一応タイプが打てるようになるには，どのくらいかかると思われますか．自動車の運転ならば，免許を交付され，一応運転できるようになるまで，実技３０時間ぐらいでしょうか，キーボードならどれくらいだと思われますか？

答えを申しますと，３～５時間，せいぜい１０時間以内で，自動車学校卒業程度にはなります．使わなければペーパ・ドライバと同じで忘れてしまいますが，引続き使っていれば，半年ないし１年で，あまり意識しないで打てるようになるはずです．

ソフトウェア技術者の場合には，当然毎日，プログラムを書くにしても，仕様書をワープロで打つにしても，キーボードから逃れられない運命にあるのです．ソフトウェアの現場で，その基礎訓練がたった数時間でできるという事実がほとんど知られていない．これは実に困ったことだと思います．みなさんは，どう思われますか？

＜聴講者よりの意見＞

Q： 現状では，入力装置やソフトウェアが，ブラインド・タッチを考慮して設計されていないので，あまりブラインド・タッチの意味があるとは思えないのですが．

A： 現状ではブラインド・タッチができない人が大部分ですから，そういう人たちを標準として入力装置やソフトウェアが設計されているので，おっしゃるようになっています．しかし，欧米では，たとえば Emacs などがいい例なのですが，ブラインド・タッチにいいように，どうすればホーム・ポシションから手を動かさずに作業ができるか，という発想でソフトが設計されています．結果として，ブラインド・タッチ技術が普及するかどうかで，ヒューマン・インタフェイスの設計がまったく変わってしまうでしょう．

しかも，ブラインドタッチは，３時間程度の基礎訓練で，健康で正常な人間ならば，だれでもできることなのです．ところで，プログラミング教育を受ける人間の中で，プログラムがちゃんと書けるようになるのはせいぜい半分，設計がちゃんとできる人は１割に満たないでしょう．こういった知的な仕事には，人間の能力差が非常に大きくあらわれますが，キーボードを叩くことなら，少なくとも指がちゃんとついていて，日本語がちゃんと読めるひとであれば，必ずできるようになるのです．

CASE 環境とかいうけれども，ここから始めなければ，それ以降のツールの設計方針がまるっきり変わってきますし，ブラインドを前提としたシステムとそうでないシステムとでは，作業効率が大きく変わってくるでしょう．

12．ソフトウェア・エンジニアの質について

CASE 環境の裏には，それを支える設計技法があります．それを理解しないと CASE は使いこなせません．したがって，CASE の導入にあたっては，まず設計技法の教育を行なう必要があります．しかし，簡単で効果が歴然とし

ているキーボード技術の訓練さえ，なかなか普及しないのですから，CASE 導入のためのトレーニングをどうするかは，これから大きな問題になるでしょう．

アメリカでは，少なくとも，設計に携わるようなエンジニアには，大学でコンピュータ·サイエンスを専攻した人間を雇うのがふつうです．しかし，日本には，まず，コンピュータ·サイエンスを教えている大学がほとんどないのです．情報工学科はあちこちにありますが，その実態といえば，大部分が電気（または電子）工学科で，その中に情報工学がマイノリティで混ざっているという状況なのです．

そして，ソフトウェア業界は，その情報工学科の学生にさえなかなか来てもらえない，そこで，ソフトウェアが専門でない人を集めて仕事をしているのが実情です．これでは，よいソフトができるほうが不思議です．

いまは，買い手のが，ソフトのことがわかっていないから，それらしきものがあれば売れています．そして，何でもいい売れればよしという商売を続けた結果，バックログをかかえてフーフーいっているのが現状です．そんなひどい業界には，まともな人間は来てくれません．ロクな人間はいないのに仕事だけはあるという現在の状況が，このまま続いたら，そのうち日本の工業は駄目になるんじゃないでしょうか．

ツールや方法論を導入するのは簡単ですが，ほんとうにそれで生産性を上げようと思ったら，それらの道具使う人間を，それに適合するように教育しなければなりません．私にいわせれば，その第１歩がキーボードなのです．キーボードがブラインドで叩けないような程度の知性しかない人間に CASE 環境を与えたところで，ロクなソフトはできません．よいソフトを作ろうと思ったら，あるいは，ソフトでちゃんと利潤を出そうと思ったら，技術者の能力をどう上げるかが，まずなによりも重要だと思います．

そのためには，まずはキーボードをブラインドで叩くことから始めなさい，次に，コンピュータ·サイエンスをちゃんと勉強しなさい，ということをいいたいのです．

そうした努力をしなくても，ソフトウェアは何となくできてしまうので，みんな勉強しないのでしょうけれど，やはり，技術にちゃんと初期投資をして，その代わり他の人の何倍かの能力を発揮するのが，ほんとうのプロというものではないでしょうか．

現在の日本のソフトウェア業界は，まだ素人集団にすぎないので，客が馬鹿だからそれですんでいるだけのことです．プロとしての能力を教育することをサボっていることに，大きな問題があります．だから，ソフトウェアの技術者の能力差が１０倍以上あるなどという状況になるのです．１０倍なんてもんじゃないという意見もあります．

一方，製造業にいって聞いてみますと，たとえば自動車工業では，技術者の能力差は２倍までしか許せないということだそうです．１０倍以上の能力差が許されるソフトウェア業界は，大変異常なのです．この異常な能力差を２倍以内にもっていく，このためには能力のない人にはヤメてもらわないといけない．そうでもしないと，ソフトの生産性は決して向上しないでしょう．

そういう努力をした会社は生産性が上がって競争力ができ，努力しないところは脱落していく，そうなったとき初めて，ソフトウェア業界というものが，社会的に認知されたプロフェッショナルな業界になるのではないでしょうか．

そうするための第一歩がブラインド·タッチの訓練です．そして，ちゃんとした能力のある人に仕事をしてもらうというプロフェッショナル集団を目指すべきです．そういうことのほうが，CASE 環境のハードウェア構成をいろいろ工夫することなんかより，はるかに大事なことなのです．

私も CASE 環境を研究しています．そして，日本の社会風土に適した CASE，たとえば KJ 法や HCP チャートをベースにした CASE 環境を試作していますが，そういうものを導入する以前の問題のほうがはるかに大きいと感じています．そして，ぜひこのことを本気で考えていただきたいというのが，今日のお話を通じて私のいいたかったことです（拍手）．

SEA 秋のセミナーウィーク '89

Session C2

マルチメディアを活用したオブジェクト管理システム

講　師: 熊谷　章 (PFU & 富士通)

報告者: 長井　修治 (大和ソフトウェアリサーチ)

1. はじめに

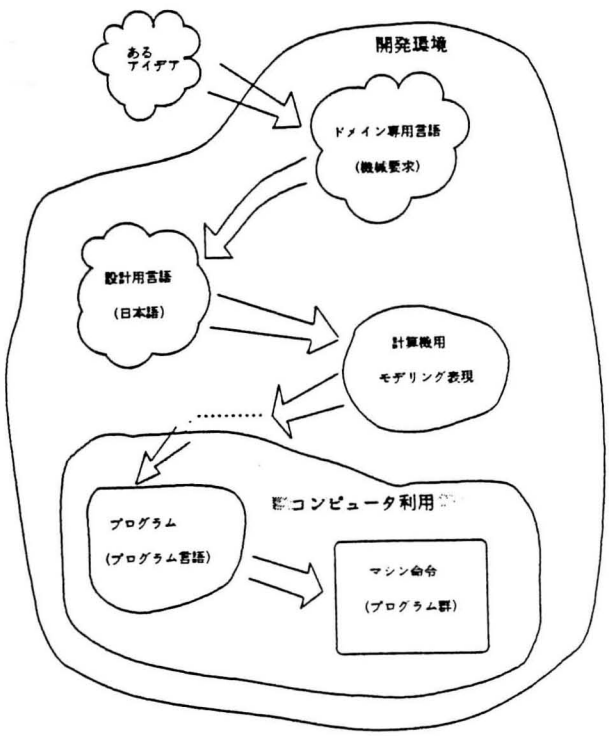
当日のセッションは、概ね下記の11項目に沿って進められた。ただ講師の含蓄の有る言い回し（将に口をついて出る言葉自身がオブジェクト指向的に働き掛けてくる様に思えた。）を直截に伝える事は難しくＯＨＰの抜粋を中心に報告する。

尚、オブジェクト管理システムを以下、ＯＭＳ（Ｏｂｊｅｃｔｅｃ　Ｍａｎａｇｅｍｅｎｔ　Ｓｙｓｔｅｍ）と表記する。

1）ＯＭＳの必要性
2）ＯＭＳとＣＡＳＥとの関係
3）ＯＭＳの概念
4）ＯＭＳのアーキテクチャ
5）データモデル
6）オブジェクトのタイピング
7）Ｏｂｊｅｃｔ　Ｔｙｐｅ　Ｅｖｏｌｕｔｉｏｎ　の問題
8）ＯＭＳにおけるツールの統合化
9）ＯＭＳの一提案
10）ＯＭＳとＨｙｐｅｒ　Ｔｅｘｔの関係
11）ＯＭＳの今後の方向

2. ＯＭＳの必要性

ウォーターフォール型の開発に於けるＳＥの活動工程に目を向けると、現在の開発環境の中でどうしても表現の変換を担うツールが必要となってくる。



従来のシステム開発の方法
　グループ作業と組織的管理運営（ｉｎｃｏｒｐｏｒａｔｅ　ｄｉｒｅｃｔｉｏｎ）
　定型用紙ベース（ｆｏｒｍ）
　標準仕様／標準作業の順守（ａｃｃｅｐｔａｎｃｅ　ｓｔａｎｄａｒｄ）

欠点
　論理的な開発方法の欠如（ｌａｃｋ　ｏｆ　ｆｏｒｍａｌｉｔｙ）
　保守性の欠如（ｌａｃｋ　ｏｆ　ｍａｉｎｔａｂｉｌｉｔｙ）
　環境と道具が人間の思考の速度に合わない

これからのシステム開発の方法
　人間とコンピュータの協調システム
　ｆｏｒｍａｌな開発方法を低経費で実現
　システム開発の全ライフサイクルをサポート
　人間の思考速度に合った環境と道具（電子媒体）

3．ＯＭＳとＣＡＳＥとの関係

システム開発の全ライフサイクルをサポートすると言う意味では、ＣＡＳＥ　ｔｏｏｌも同様に考えられるが、ＯＭＳとＣＡＳＥとの関係は、次のようである。

1）キーワード

統合化（Ｉｎｔｅｇｒａｔｉｏｎ）
　マルチプルツールの統合　　ｔａｉｌｏｒａｂｌｅ＿ｆｒａｍｅｗｏｒｋ

データベース（Ｄａｔａｂａｓｅ）
　統合化された情報の格納庫
　ＥＲＡ技術とオブジェクト指向ＤＢ（変革の起爆剤）
　Ｈｙｐｅｒｔｅｘｔ　ｓｙｓｔｅｍ
　Ｍｅｔａｍｏｄｅｌｉｎｇ

標準（Ｓｔａｎｄａｒｄ）
　ツールの結合のキーポイント

進化のための機能（Ｅｖｏｌｕｔｉｏｎ）
　全ライフサイクルをカバーする自動化への進化
　第一世代ＣＡＳＥから第二世代ＣＡＳＥへ

エキスパートシステム（ＳＥ＿ＫＢ）
　設計支援と自動プログラミング

品質保障（Ｑｕａｌｉｔｙ　ａｓｓｕｒａｎｃｅ）
　分析、テスト、信頼性への新しい可能性
　操作と性能の品質保障
　シュミレーションとプロトタイピング

2）第一世代のＣＡＳＥの機能

方法論の推進とトレーニング
　ツールを介して実現

システム分析ダイアグラムのサポート
　要求と設計を対話型のエディタで可能にした

Ｓｉｎｇｌｅ－ｅｎｔｒｙ ｓｐｅｃｉｆｉｃａｔｉｏｎ ｂｏｏｋｋｅｅｐｉｎｇ
同一情報格納庫に総ての情報を格納しマルチビュウ

Ｒｅｍｉｎｄｅｒ ａｎｄ ｃｏｎｓｉｓｔｅｎｃｙ ｃｈｅｃｋ
データの無矛盾性
他のコンポーネントとの整合性のチェック

3）第二世代のＣＡＳＥ

第二世代のＣＡＳＥの機能要求
方法論の適用（ｍｅｔｈｏｄｏｌｏｇｙ ａｄａｐｔａｔｉｏｎ）
管理運用機能
ドキュメントの出力、知的なダイアグラム作成支援機能

ＣＡＳＥを用いることから実現される信頼性
検査と分析
品質保障とテスト工程の標準
カバレジテストの実現
動的な情報格納庫
実用的な百科事典（ソフトウェア設計のための）
ドキュメントの修正が容易に
システム変更のための活性源（ｅｎｃｏｕｒａｇｅ ｔｏ ｍｏｄｉｆｙ）

4．ＯＭＳの概念

概略的に把握する意味で、現在開発中の機能要求を掲げる。

ビジュアルなマンマシンインターフェース
複数オブジェクトの同時アクセス
ユーザインターフェースのカスタマイズ機能（ＴＯＯＬＢＯＸ）
システム開発のすべてのプロダクトをオブジェクトとして格納
ＯＭＳと他コンポーネントとのインターフェース
①プログラム呼出し
②コマンドレベル呼出し
オブジェクト間のリレーションの関係付けと操作
オブジェクトのブラウジング機能
リレーションの変換（フィルタリング）
思いどおりに素早く操作できること

5．ＯＭＳのアーキテクチャ

基本的な設計思想になっているオブジェクトを中心に進める。

1）ソフトウェア開発におけるオブジェクト

プロダクト：　要求仕様書、設計仕様書、マニュアル、プログラム、設計書、試験書
資源情報　：　金額、人員、コンピュータ資源
計　　画　：　開発計画、詳細な開発計画書
自動化装置：　プロセスプログラム、各種ツール
測定データ：　分析データ、評価データ

2）ＯＭＳの中心テーマ

ＯＭＳの問題　：＝　ｆ（ソフトウェア工学、ＤＢ研究）
①データモデル（ｄａｔａ　ｍｏｄｅｌ）
②統合化の管理（ｉｎｔｅｇｒｉｔｙ　ｍａｎａｇｅｍｅｎｎｔ）

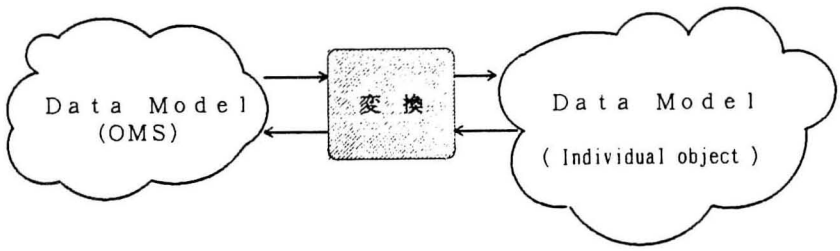
ＯＭＳ　：＝オブジェクトとその情報の管理
次の情報を作成、修正、削除、検索する
①特性（ｐｒｏｐｅｒｔｉｅｓ）
②オブジェクト間の関係（ｉｎｔｅｒｒｅｌａｔｉｏｎｓ）

3）オブジェクトの性質とその問題点
オブジェクトとしてどんなまとまりにするのか
複合オブジェクトは必要か
オブジェクトへのアクセス制御として何が必要か
どんなオペレーションに意味があるか
オブジェクトの性質は動的に変化するか
オブジェクトは物理的にグルーピングする必要があるか

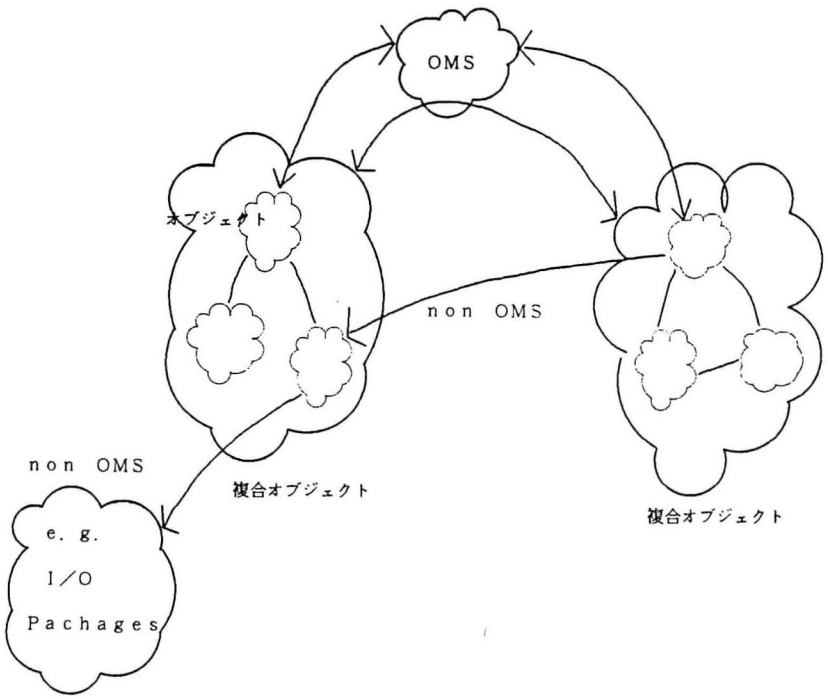
4）ＯＭＳの適用範囲
一つのＯＭＳで多くのプロジェクトをサポート
アクセス制御、Ｔｙｐｉｎｇ機能でプロジェクト毎の対応
複数個のＯＭＳで複数個のプロジェクトをサポート
Ｐｒｏｊｅｃｔ－ｓｐｅｃｉｆｉｅｄ　ＯＭＳ
異なったプロジェクト間のオブジェクトの交換
ｅｘｐｏｒｔ、ｉｍｐｏｒｔ、ｒｅｆｅｒｅｎｃｅ

6．データモデル
1）Ｏｂｊｅｃｔ　Ｇｒａｎｕａｌａｒｉｔｙ（ＯＭＳが取扱うオブジェクトのまとまりの単位）
ＯＭＳのトレードオフ
①ゼネリックなＯＭＳはプリミティブなデータタイプをサポートする
②より複雑な情報の効率的操作を可能にするように、オブジェクトの内部構造までＯＭＳがサポートする

Ｇｒａｎｕａｌａｒｉｔｙの選択



2) Sub−granualarとConsistency check



3) OMSのObject管理機能

ObjectのGranualarityの設定

Sub−granualarの参照が生じないようにする

Object Compositionのパラダイム

様々なcompositionのやり方を可能にする

OMSの適用範囲

半永久的に複数のプロジェクトに跨がってOMSは適用される

OMSはプロジェクトの境界に従ってtailorableにする

7. オブジェクトのタイピング（Typing of Object）

1) タイピング

OMSにおけるタイピング

①オブジェクト中の情報　②属性の特性　③他のオブジェクトとの関係

OMSにおけるタイピングモデルの目的

①特性とtypeのインスタンスの関係を表現

②あるオペレーションとtypeのインスタンスの関係を表現

③typeのインスタンスの制約

④OMSの一貫性の保守のための束縛条件を実施

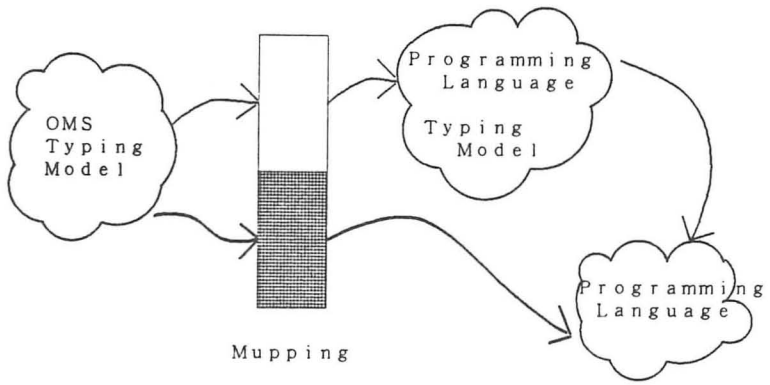
⑤ユーザによる特性のビジュアルなviewを作ったり、ネーミングの矛盾発生を解決するをview作る

2) オブジェクトタイピングの問題点

Type Expectation

コンパイル時に全てのオブジェクトは不明 ⟶ expectation

Type Validation



8. Object Type Evolution の問題

1) Typing Evolutionのために必要なメカニズム

タイプ定義の部分の変更
タイプ定義の前のバージョンのオブジェクトも継続して存在
そのタイプの新旧両方のオブジェクトを操作できるツールの存続

2) 三種類の一般的な Type Evolution Model

特殊化／一般化モデル
Type Versioning Model
In-place Modification Model

3) Type Predicates

Type Evolution には、Type Predicates がよい
Type Predicate Model の留意点
混乱した Type Evolution を避けるために、高度な Formalismが必要

4) Multiple Type Model

レベル1 : Data Model、Family of Type Model
例 ERA

レベル2 : OMSに組み込まれた特殊データモデル
例 CAIS ERA

レベル3 : DDL (Data Description Language)

レベル4 : Schema Definition in DDL

レベル5 : Instance of Type

5) 普遍か特殊か) (Type Model)

一個のゼネラルなTyping Model で充分か?
複数個のTyping Model が必要か?
Multiple Typing Modelにおける困難さ
Consistency Constraint を Formulateすることが極めて困難
Multiple Type Modelのインプリメンテーションが大変
Single Typing Model への期待
表現力の弱さを克服すればOMSの多くの問題が解決できる途

9．OMSにおけるツールの統合化

ここ数年の状況　：　ツールが生成したデータのセマンチックスレベルでの移住

最近の状況　　：　オブジェクトの統合のための制約条件がOMSの対象

統合化管理のためのアーキテクチャ

10．OMSの一提案（HyperBook）

1）PFUにおける試作事例を基に、OMSのデータの特性と操作について

比較項目	OMS	DBMS
データの抽象度	低い	高い
データの意味	利用者に依存	データ構造が表現
データ構造	動的	静的
データの種類	マルチメディア	文字と数値
データのサイズ	～数MB	～1KB
ユーザビュー	可変	固定
スキーマの設計	データ収集後	データ収集前

2）概要

プロダクト（オブジェクト）の定義

名前、アイコン、属性、他オブジェクトとの関係

プロダクトの産出

クラスから産出した名前、属性などの情報を書き込む

プロダクト間の関係

関係の定義をマニュアルで行う

関係の種類・・・参照、等値、部分、分解、組立、抽象、包含、先者、後者

プロダクト間の操作

関係を対象とした操作

注視の対象を動的に変化させそのリレーションを取り出す

プロダクトの種類

常識プロダクト・・・プロジェクト、キャビネット、本、シート、プログラム、データ

非常識プロダクト・・ユーザ定義のプロダクト

プロダクトの操作

各プロダクトの操作はオブジェクト指向型とじてプロダクトに付加

操作をするためにはその操作を指定

シートの操作と編集
　既存シートへのアクセス、新規シートの作成

プロダクトの参照と操作
　ビジュアルにかつ高速に探す方法 } 現在インプリメント済
　キーワード、印で検索する方法
　属性の内容で検索する方法

インプリメンテーションの見通し

要求される機能	実現するメカニズム
動的構造	オブジェクト指向、ｌｉｓｐ
ビジュアルインタフェース	ＳＴ－８０、ＥＲＡエディタ
カスタマイズ	インヘリタンス、メタモデル
カプセル化	オブジェクト指向、Ａｄａ
リレーション操作	Ｈｙｐｅｒｔｅｘｔ、ＥＲＡ、ＭＭＤＢ
ネットワーク機能	ＵＮＩＸ、ＭＡＣＨ

11. OMSと　Ｈｙｐｅｒ　Ｔｅｘｔ　の関係

1）Ｈｙｐｅｒｍｅｄｉａの定義

情報はあるユニットの束である
ユニットの情報はウィンドウに表示される
ユニットはリンクによって関係付けられる
ユニットの創成、編集、関係付けによって様々な情報を表現
Ｈｙｐｅｒｍｅｄｉａは共有されたマルチアクセスが可能である
＊　Ｋｅｙ　Ｉｓｓｕｅｓ　：　Ｄａｔａ　Ｍｏｄｅｌ、Ｄｉｒｅｃｔ　ｍａｎｉｐｕｌａｔｉｏｎ

2）ユーザインタフェースの問題

①どんなスタイルのユーザインタフェースを使用すべきか
　Ｄｉｒｅｃｔ　ｍａｎｉｐｕｌａｔｉｏｎ　Ｐａｒａｄｉｇｍ
②ノードはディスプレイ上でどのように表示されるべきか
③リンクをディスプレイ上でどのように表現するか
④リンクを辿った時システムはどれ位の速度でレスポンスすべきか
　最も重要な技術　：　＜０．２５秒／ノード　（因みに現状は、１秒間に４シート）
⑤システムはブラウジングをどのようにサポートすべきか
⑥データベースのグラフィカルなビューは必要か
⑦いかにして”ｌｏｓｔ　ｓｐａｃｅ　ｐｒｏｂｌｅｍ”を解決するか
　階層スケルトン、ナビゲーションコマンド、高速レスポンス
⑧ブラウジングの他の検索の方法として何が必要か
⑨ユーザインタフェースをどのようにしてカスタマイズするか

3）Ｈｙｐｅｒｍｅｄｉａの今後の課題

システムの拡張性

Ｐｒｏｇｒａｍｍｉｎｇ　Ｌａｎｇｕａｇｅが必要（例、Ｈｙｐｅｒｔａｌｋ）

巨大なデータベース作成が難しい

電子情報からの紙情報へのコンバート

Ｈｙｐｅｒｍｅｄｉａの自由度の大きさは利用者には両刃の剣

採用したデータモデルがそのシステムのすべてを決める

人間とコンピュータの交信の問題

ソフトウェアシステムの複雑さは半永久的に増進する

Ｈｙｐｅｒｍｅｄｉａではデータモデルの標準を研究開発することができる

Ｄｅｓｋｔｏｐ　Ｍｅｔａｐｈｏｒの革新が可能である

12. ＯＭＳの今後の方向

ＯＭＳ自体の発展は、これまでの中でその方向性を既に示唆しているので、ここではソフトウェア開発全般の展開をまとめて終わりとする。

1）ＡＩとソフトウェア工学

Ｇ．Ｆｉｓｃｈｅｒ（コロラド大学）によるとソフトウェア開発の流れは３つのフェーズでみることができる。

ソフトウェア開発：３つのフェーズ

フェーズ１：直観（Ｉｎｔｕｉｔｉｏｎ）

フェーズ２：仕様の確定

フェーズ３：作成

１９８５年までのソフトウェア工学の主テーマ

対象フェーズ：フェーズ２、フェーズ３

技術的テーマ：構造的プログラミング、検証、ｆｏｒｍａｌ　ｓｐｅｃｉｆｉｃａｔｉｏｎ

ターゲット　：この段階の問題は構造化された問題であり最適の解答を求めること

１９８６年以降のソフトウェア工学の主テーマ

対象フェーズ：フェーズ１、フェーズ２

技術的テーマ：ラピッドプロトタイピング、視覚化、エラーの削減

ターゲット　：この段階の問題はあまり構造化されない問題であり最適解ではなく、満足する解を見出すこと

2）理論的とＡＩ＆ＩＡアプローチの比較

比較項目	理論的アプローチ	ＡＩ＆ＩＡアプローチ
計算機科学	formal、数学的	実験的
基本姿勢	明晰な思考	よりよい道具
主なツール	形式的仕様定義	実験的プログラム
プログラム方法論	作成前の検証	誤り修正が設計
問題の性質	well-structured	ill-structured
解の性質	最適解	満足解

3）ＡＩ＆ＩＡのソフトウェア工学への寄与

ソフトウェア設計者のための知識ベース支援システム

①設計者の問題解決能力を増幅させること
②設計者の作業環境を改善すること
③設計者の生産能力を増大させること

複雑な問題に対して深くてよい理解を可能にすること

設計対象の性質を次のように変えることから可能になる

ｗｅｌｌ－ｓｔｒｕｃｔｕｒｅｄ ⟶ ｉｌｌ－ｓｔｒｕｃｔｕｒｅｄ
最適解 ⟶ 満足解 ⟶
プロダクト中心 ⟶ プロセス中心
アルゴリズム ⟶ ヒューリスティック

── 以 上 ──

システム サービス プログラム

横山 博司
SRA 関西支社

1. システムサービスプログラム

1-1. 基本的な考え方

ソフトウェア開発手法の一つとして，システムサービスプログラムおよびツールについて考えてみたい．その基本的な考え方は，以下の通りである：

(1) アプリケーション・プログラム開発者（ユーザ）が，OSのシステムコールの使い方を学習することなく，容易にシステムのリソースを使用できること．

(2) メッセージのログを提供し，かつ，タスクの生成やメールボックスの生成が容易に指定でき，使いやすいデバッグ環境を提供すること．

(3) ソフトウェア・システムから見てモジュール化された１つのロジックであり，構造化設計の助けとなること．

1-2. 機能

システムサービスプログラムは，マルチ CPU，マルチタスクシステム向けのものであり，以下の機能をサポートしている（もちろん，最初の２つについては OS 自体の機能が必要）：

(1) メッセージの送受信（CPU の内外を問わない）

(2) 共通テーブルの排他制御

(3) システムダウン時の処理

1-3. 効能

ユーザは，OS のシステムコールに代わって，システムサービスプログラムを使用する．それは，特定システム向けにアレンジされてた専用のプログラムである．したがって，汎用的に作られた OS の不必要なパラメータを意識する必要はなくなる．

一連の処理の流れを１つのシステムサービスプログラムで行なうため，複数のシステムコールを使う必要がなくなり，繁雑なロジックから解放される．

OS のシステムに関係するところは，あらかじめ動作確認した上で提供されているので，OS（特にシステムコールの使い方）を意識することなく，本来のアプリケーションのデバッグに専念することができる．

もちろん，CPU 間やタスク間のインタフェイス，資源（システム共通テーブル）の占有などの機能，および開発システムの仕様については，十分な理解が必要である．

提供されるメッセージ・ログは，デバッグ中の CPU またはタスク間の通知状態を知る上で，大きな手助けとなる．特に，ダイナミック・デバッガ（タスクのスイッチング状態，メッセージ内容などをリアルタイムで見ることができるもの）が提供されていない場合には，効果は絶大である．

また，タスクの生成やメールボックスの生成の指定が容易にできることは，単体テストから順次タスクを結合させて行く過程では，大きな武器となる．

1-4. 実際

システムサービスプログラムは，最初の開発時はもちろんのこと，改造に際しても非常に効果を発揮する．

実際にあった例だが，MS-DOS 上で作り上げたシステムが，大きくなりすぎて，メモリーに載らなくなってしまったことがある．対策の１つとして，常駐の必要がないタスクをディスクに追いやり，必要に応じて呼び出して使えるように，オーバーレイすることにした．

具体的には，アプリケーションプログラムとのインタフェイスにはほとんど手をつけず，システムサービスプログラムで，タスクへの起動要求が発行されるごとに，オーバーレイ対象のタスクか否かを判断して処理をしたのである．そのお蔭で，アプリケーションの改造はきわめて少なくてすんだ．

1-5. 実行効率

OS のまわりにこのようなサービスプログラムをかぶせると，実行ステップがふえて実行速度が遅くなるのではないかと疑問を持たれる向きもあるかも知れない．

しかし，この考え方を採用しなくても，開発時に何らかのモジュール分割を行なう（つまり，サブルーチンとして作る）わけだから，実行ステップにはあまり変わりはなく，速度には影響は出てこない．

2. システムサービスツール

前章では，実機（ターゲットマシン）におけるシステムサービスプログラムの機能や効能について考えてみたが，本章では視点を少し変えて，開発マシン上での支援ツールとしてとらえてみよう．この種のツールのことを，システムサービスツールと呼ぶ．

2-1. 必要性

制御系のシステム開発では，ソフトウェアとハードウェアが並行して開発されることが多い．たとえハードウェアが先に稼働していても，まだソフトウェア開発用には使えないといったこともよくある．

こんな場合，実機と開発マシンとではプログラムの動作環境がちがってくる．たとえば，実機にはリアルタイムモニタが搭載されるにもかかわらず，開発には Unix や MS-DOS を使う場合がある．となると，開発マシン上でテスト環境を構築する必要があり，しかも信頼性の高いテストを行なうためには，OS のシステムコールを模擬する必要がある．

つまり，リアルタイムモニタ(OS)のシミュレーションになるわけだが，あまり大規模な仕掛けを考えるのは得策ではないので，ここでは，マルチタスクとしてではなく，あくまでシングルタスクの動作環境を提供することを考える．

2-2. 機能

一般に OS のシミュレーションは楽ではない．なぜなら，特にリアルタイムモニタなどの場合は，それがハードウェア(CPU)に依存しているので，まるで OS を移植するのと同じことになる．この際，基本的な部分を取り出してきて，タスク間メッセージの通信，テーブルの排他制御，定周期起動，そして，1回休み(一時停止)などとしてとらえれば，話は簡単になる．つまり，ハードウェアに依存しないシステムサービスプログラムにしてしまえばよいのでそうすれば，シミュレーション環境も簡単に作れる．

話を整理すると，システムサービスツールとは，以下の目的を持ったものになる：

(1) 高度なテストをするため，アプリケーションプログラムを変更せずに，開発マシンから実機への移植が出きること（デバッグのためにソースに修正を入れていたのでは，デバッグ終了後の正規の形に戻す過程でのミスが原因でバグを作りこむことになる）．

(2) あくまでテスト用ツールなので，アプリケーションプログラムは，該当タスクだけをとらえた場合，実機上と変わらない状況で動作し，しかも，テスト データの設定と実行結果の収拾ができること．

ここで，テストデータの設定と実行結果の収拾とは，

(a) システムサービスプログラムからの出力データが任意に設定できること

(b) システムサービスプログラム呼び出し時の入力データを見ることができること

をいう．

たとえば，C 言語でシステムサービスツールを作るならば，設定値は getc() で取り込むようにすればよいし，実行結果は printf() で出力させればよい．Unix や MS-DOS であれば，リダイレクションを利用すれば，簡単にファイルとのアクセスに替えることができる．

3. 最後に

システムサービスツールを用いてテストされたプログラムは，実機の上で，システムサービスプログラムを用いて結合テストがなされる．つまり，単体テストから結合テストまでの環境が統一されるわけである．

こうした開発のやり方を実現するためには，少なくとも詳細設計の時点までに，方針をまとめておく必要がある．でないと，作業に余分な手戻りが生じることになる．すでにコーディングまで終わってから，あらためてプログラムに手を加えようなどというのは，とんでもないことで，とても不可能な大改造になってしまう．

行き当りばったりのやり方ではなく，上流工程から下流工程まで，きっちりと全体の構成を踏まえた形で，細部に渡ってスムーズに展開して行くことが，美しいプログラム[*1] を作るための1つの条件である．骨組み（基礎）さえしっかりしていれば，少々の仕様変更にもうろたえることはない．

身近かなところで，こうした創意工夫をどんどん取り入れて，お互い，もっと楽しくプログラムしようではありませんか．

We love beautiful software.

[*1] 美しいプログラムとは，すっきりした構造で，むだなステップがなくて，いかにもメインテナンス（仕様追加，変更が）しやすそうで，信頼性が高そうで（バグが少なそうで），実行速度の早そうな .... つまりは見ていて惚れぼれするような，気持ちのよい(仕事のやりやすそうな／楽ができそうな）プログラムのことをいう．

それとは正反対のダサいプログラムは，そのうち，TV の低俗番組と同じように，追放運動のターゲットにされること請け合いである．しかし，その手のものに限って，しぶとく生き残るのよネ，きっと ?!

SEA 1990 － 91 年の主要イベント（実績と予定：1990.3.20 現在）

	***** これまで（1990年 1月～ 1990年 3月中旬） *****	
1/11-13	第 2 回テクニカル マネジメント ワークショップ	北海道：函館市（参加者 14名）
1/27	SIGENV-Forum「現場にとってソフトウェア開発環境とは？」	岩手：盛岡市（参加者 30名）
1/31	SEA Forum「CASE ツールの現状と動向」	東京：機械振興会館（参加者 89名）
2/1-3	第 2 回 ソフトウェア プロセス ワークショップ	静岡：伊東市（参加者 23名）
2/27	SEA Forum「CASE ツールの導入と活用」	東京：機械振興会館（参加者 93名）
3/2	九州支部設立 2 周年記念イベント	熊本：熊本市（参加者 57名）
3/6-7	春の集中セミナー'90	東京：青年会議所会館（参加者延べ 165名）
	***** これから（1990年 3月～ 1991年 3月） *****	
3/21-4/3	第 12 回 ICSE 研修ツアー	Nice（France）（参加予定者 17名）
4/13	SEA Forum「ダウンサイジング !?」	東京：機械振興会館（参加者募集中）
		－現時点で申込 47名－
5/15（午後）	SEA Forum「ファジィ !?」	東京：機械振興会館（企画中）
5/15（夜）	SEA 1990 年度総会	東京：機械振興会館（企画中）
6/5	第 3 回 日中ソフトウェア・シンポジウム	大阪：千里国際情報事業財団（企画中）
6/6	ソフトウェア シンポジウム '90 併設チュートリアル	京都：京都リサーチパーク（企画中）
6/7-8	ソフトウェア シンポジウム '90	京都：京都リサーチパーク（参加者募集中）
9/5-8	第 8 回 夏のプログラミング ワークショップ（若手の会）	
	「CASE ツールの実際的応用」	岩手県：盛岡市（企画中）
9	第 4 回 教育ワークショップ	開催場所未定
9	SEA 秋のセミナー ウィーク '90	東京：青年会議所会館（予定）
10	第 11 回 ソフトウェア信頼性シンポジウム	大阪（予定）
10/29-31	6th International Software Process Workshop	北海道：函館市（Paper 募集中）
11/20-22	第 6 回 実践的開発環境ワークショップ	
	「開発環境革新のケース・スタディ」	愛知県：名古屋市（企画中）
12	4th SDE Symposium 研修ツアー	Irvine（USA）（Paper 募集中）
1	第 3 回テクニカル マネジメント ワークショップ	北海道（予定）
1/31-2/1	第 3 回 ソフトウェア プロセス ワークショップ	静岡：伊東市（予定）
3	SEA 春のセミナー ウィーク '91	東京：青年会議所会館（予定）

[1] 東京でのイベントを中心にリストアップしました．

[2] 90 年 6 月以降の SEA Forum は未定です．

[3] 5 月以降，仙台・長野などで 巡回 CASE Forum を開催すべく，
現在企画中です．

13th International Conference on SOFTWARE ENGINEERING

CALL FOR PAPERS

SYSTEM DESIGN: RESEARCH & PRACTICE

Austin, Texas May 13-16, 1991

CHAIR:
Laszlo A. Belady
ICSE13 Chair
MCC
P. O. Box 200195
Austin, Texas 78759 USA
belady@mcc.com
512/338-3504
FAX: 512/338-3899

PROGRAM CO-CHAIRS:
David Barstow
ICSE13 Program
Schlumberger Laboratory for Computer Science
P. O. Box 200015
Austin, Texas 78720-0015 USA
barstow@slcs.slb.com
512/331-3728
FAX: 512/331-3760

Koji Torii
ICSE13 Program
Dept. of Information & Computer Sciences
Osaka University
Machikaneyama 1-1
Toyonaka City
Osaka, JAPAN
torii%tori2.ics.osakau.ac.jp@relay.cs.net

TOOLS FAIR CHAIR:
Laurie or John Werth
Department of Computer Science
University of Texas at Austin
Austin, Texas 78712 USA
lwerth@cs.utexas.edu
jwerth@cs.utexas.edu
512/471-7316
FAX: 512/471-8885

TUTORIAL CHAIR:
Herb Krasner
Lockheed
Software Technology Center
O9610/B30E
2100 East St. Elmo
Austin, Texas 78744 USA
krasner@stc.lockheed.com
512/448-5738
FAX: 512/448-5728

LOCAL ARRANGEMENTS CHAIR:
Bryan Fugate
ICSE13 Local Arrangements
MCC
P.O. Box 200195
Austin, Texas 78759 USA
fugate@mcc.com
512/338-3330
FAX: 512/338-3899

ell into its third decade, the engineering of software is becoming the engineering of computerized applications. Increasingly, new systems arise through the adaptation and integration of existing applications, with software as the glue that holds the pieces together (in much the same way that CAD and CAM have been integrated as Computer Integrated Engineering). Creating these systems requires the expertise and cooperation of a wide array of specialists — specialists in the application domain, in real time performance, and in reliability. The software engineer becomes a system designer who must understand the application domain in order to select not only the system's software but also its hardware. Perhaps most importantly, the system designer must be able to work well in teams with others.

These complexities are giving rise to new, often interdisciplinary software engineering research subjects. Already, a wealth of experience and insight has emerged from research in such disciplines as computer supported cooperative work, distributed systems design, reverse engineering, integration technology, and computer aided education and training. And the more established research subjects — quantitative methods, languages and representations, project management, and others — are expanding in breadth and excitement. From this work, a new generation of tools and techniques for improving the precision, quality, and predictability of system building projects is arising. Without superb technology and continued research, companies and nations will lack the productivity to be competitive.

ICSE invites researchers and practitioners to share their recent ideas and experiences, particularly those which aim to improve the design of complex computer applications. The program committee will review all submissions for quality and for relevance to future work. Papers, panel proposals, and reports must be received in writing by September 14, 1990.

SUBMISSIONS: Eight (8) copies in English of papers, panel proposals, or experience reports should be submitted to the address below by September 14, 1990:

David Barstow
Schlumberger Laboratory for Computer Science
P. O. Box 200015
Austin, Texas 78720-0015

Papers should be limited to 6000 words, full-page figures being counted as 300 words. Each paper must include a short abstract, a list of keywords, and the lead author's address.

Panel proposals should include title, proposed chair, tentative panelists (including a short vita), a two or three paragraph description of the subject, format of the presentation, and rationale for the panel.

Experience reports should describe practical experience in some aspect of software engineering (using a tool or method, applying a metric, following a project management discipline, reusing work products, etc.). The contributor(s) should submit a 5 to 10-page written description of the experience and a one-page outline for a five-minute presentation.

TOOLS FAIR: A software tools fair will be held during the conference, so that attendees can see and learn about current experimental and commercial software tools. Those who want to exhibit or demonstrate a tool, and especially those interested in presenting a descriptive paper, should contact:

Laurie or John Werth
Department of Computer Science
University of Texas at Austin
Austin, Texas 78712

FURTHER INFORMATION: For further information and/or copy of the advance program when available write either to ICSE13, MCC, P. O. Box 200195, Austin, Texas 78720 USA, or to ICSE13, IEEE Computer Society, 1730 Massachusetts Ave., NW, Washington, DC, 20036 USA.

IMPORTANT DATES: Submission deadline: 14 September 1990; Acceptance notification: 14 December 1990; Final versions due: 8 February 1991.

Sponsored by ACM Special Interest Group on Software Engineering
IEEE Computer Society Technical Committee on Software Engineering

acm

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC
IEEE

sea
ソフトウェア技術者協会
〒102 東京都千代田区隼町2-12 藤和半蔵門コーフビル505
TEL.03-234-9455　FAX.03-234-9454