「ダウンサイジング」における プロジェクト管理問題（下）

伊 東 暁 人

2．プロジェクト管理問題から見た「ダウンサイジング」の意味

前節では、「ダウンサイジング」の本質がネットワークを介した分散処理にあることを述べたが、本節ではそのような分散処理の環境で稼働するソフトウェアの開発がどのようなものであり、汎用大型機による集中処理環境のシステムを開発する場合と何が異なるのかを指摘し、それがプロジェクト管理から見てどういう意味を持つのかを明らかにしたい。なお、ここでいう「ソフトウェア」とは、アプリケーションソフトウェア、中でも受注に応じ個別に生産される、いわゆるカスタムソフトウェアを示し、パッケージソフトウェアやＯＳなどの

図1－(1). 情報システムにおけるソフトウェアの位置づけ

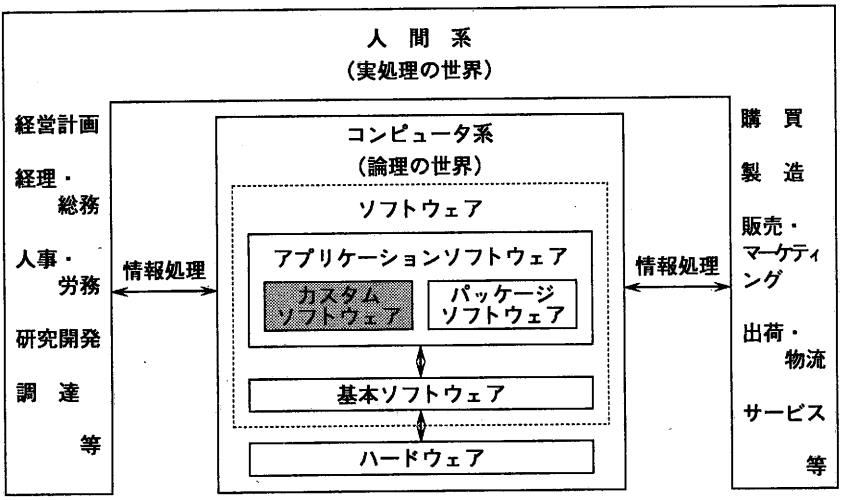


図1－(2). ソフトウェアによる要件把握のちがい

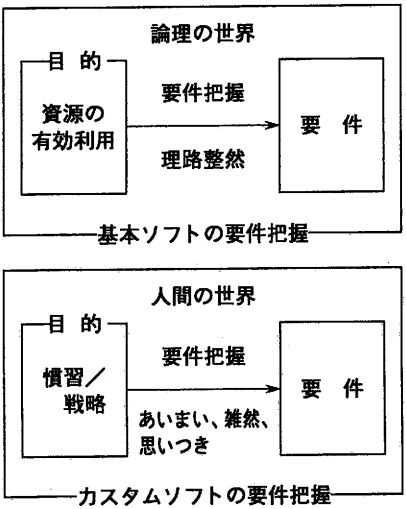


図1－(3). ソフトウェアによる生産形態のちがい

生産方式／受注方式	
	個別システム開発
受注生産	カスタムソフトウェア製品
見込み生産	パッケージソフトウェア製品
	基本ソフトウェア製品

出典：『ソフトウェア生産工学ハンドブック』(1991)

基本ソフトウェアは、その要件把握・生産形態において違いがあるため、ここでは除外する。（図1－(1)〜(3)）

(1)プロジェクト管理問題の位置づけ

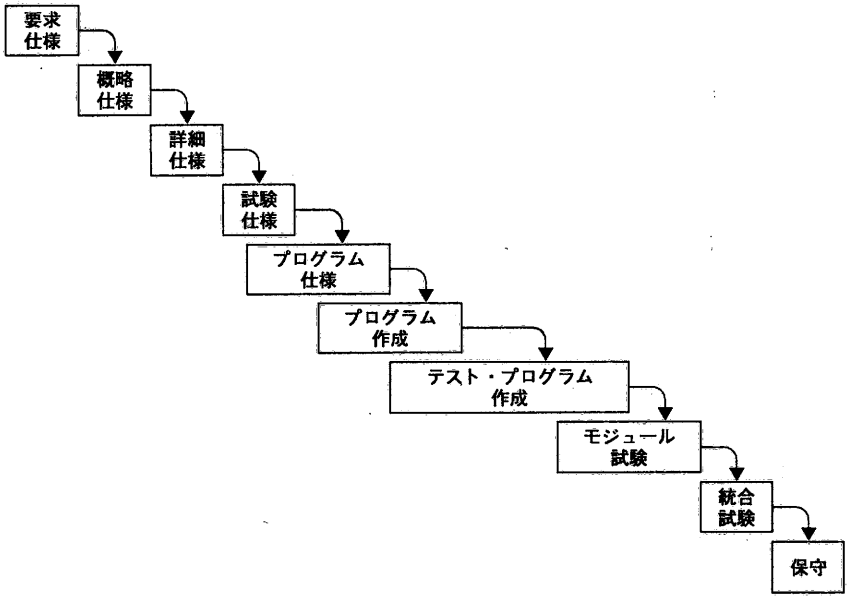
社会の情報化が進行するなかで、ソフトウェア開発の現場では、依然として低い生産性と労働集約的な構造が問題となっている。ソフトウェア開発の生産性が問われるようになってから、はやくも20年以上になる[1]が、その間それなりの成果は見られたもののシステムのニーズにエンジニアリングが追いついておらず、ハードウェアが格段の進歩を果たしたのと比し、極めて進歩が遅いといわざるを得ない。そのような状況のままで、ソフトウェアの複雑化、大規模化、高度化が一層進行しており、問題がより複雑さを増しているにもかかわらず、抜本的な解決は行われるに至っていないのが現状である。ソフトウェア開発の生産性が低い原因としては、様々な要因が挙げられるが、近年の実証研究の成果として、ソフトウェアのプログラム生産そのものよりも開発におけるプロジェクト管理にその主因があることが多いことが明らかとなってきた。[2]

「プロジェクト管理」と一言でいっても、その内容は多岐にわたる。一般には、工程管理、予算・費用管理、品質管理、要員管理、外注管理、開発環境管理などを含む全般または、一部を言うが、なかでもここで問題とされねばならないのは、開発プロセスの管理に関わるプロジェクト管理についてである。その理由は、以下のように概括できる。すなわち、アプリケーションソフトウェアの多くは個別の受発注契約で開発されるが、その場合、一般的な工業製品と異なり試作から量産へといった形態をとらず、開発即製品という形態をとる。そのことは、必然的にソフトウェア開発において開発のプロセスそのものが問題とされることになる。また、その開発工程も自動化・機械化がなされていない、きわめて人間の「能力」に直結したエンジニアリングであるためである。

ソフトウェア開発における「プロセス」とは、引き合い（受注）から稼働（引渡）に至る一連の工程の流れを指すが、なかでもソフトウェア開発・運用の流れをライフサイクル（＝生涯）にたとえ、計画から設計、プログラミング、テスト、導入、運用、廃棄にいたるフェーズごとにわけて把握する考えかたとして、ライフサイクルモデル（Life Cycle Model）がある。特に、システムの仕様に関する情報においては、上流工程（計画・設計）から下流工程（プログラミング）へと流されるべきであり、逆流すべきではないという「ウォータ

フォール（Water Fall）型のライフサイクルモデル」（図2）が、長くソフトウェア開発のプロジェクト管理において主流を占めてきた。というのは、このフェーズドアプローチは、それぞれの工程が明確に分離しているため、開発に必要とされる要員や機器などの諸資源の割当、作業の工程管理、開発工費の見積りなどを実施する際に積算が行いやすいためである。さらにこのアプローチを前提とした各種の構造化手法(3)、段階的詳細化(4)などの技法が展開され、生産性向上とソフトウェアの品質向上に一定の役割を果たしてきた。このアプローチはまた、各工程ごとに分業体制がとられている現在主流の開発体制とも適合することから、契約管理・要員管理などの事務的・実務的な観点から見ると、それなりに利便性が高いことが多い。

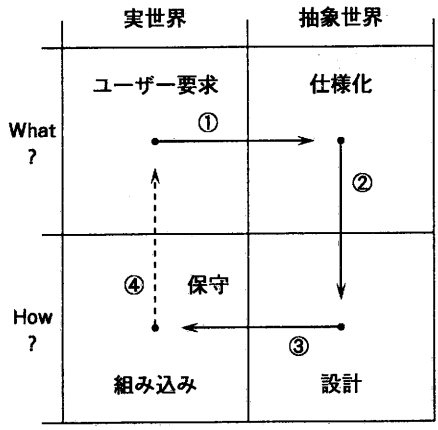
図2．ウォータフォール型ライフサイクルモデル



出典：黒川利明『ソフトウェアの話』（岩波新書，1992）

ところが、このウォータフォール型の管理モデルに対して、その利点とともにソフトウェアの生産管理面で多くの弊害が指摘されてきたのも事実である。このウォータフォール型の開発手法では、上流工程のフェーズにおいて仕様に関わる要求分析、定義を行い、設計フェーズ以降でその実現方法を決めていくため、「何を（What）」と「どのようにして（How）」が分離されて考えられる。しかし、現実問題として、これら「What」と「How」の分離は、大変に困難である。なぜなら、ユーザの仕様要求である「What」をプロジェクトの最初から厳密に定義するためには、ユーザ自身が自らのシステム要求を完全に把握している必要があり、さらに、たとえ仮に「What」をプロジェクトの最初から厳密に定義することができえたとしても、その実現方法「How」の検討の結果いかんでは、その結果を再度「What」に「逆流」（フィードバック）せざるをえないことが多いからである。（図３）

図３．Zelkowitzによるソフトウェア開発手順



出典：M.V.Zelkowitz: Principles of Software Engineering and Design, Prentice-Hall, 1979. に加筆

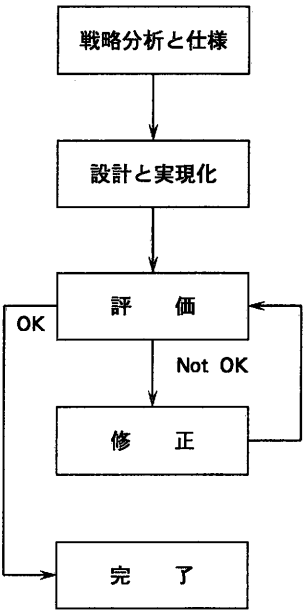
このことは、ウォータフォール型の開発手法が、システム仕様の変更への対応コストの面から－それは、ソフトウェアの仕様がもともと不確実なものであることに起因するのだが－多くの問題を持つものとして指摘されてきた。実際、ソフトウェアの修正費用は、設計時に修正する費用を１とすると、ソフトウェ

アのテスト時で15倍、保守・運用時になると30倍になると言われており、また、開発費用の約30%～50%が開発管理が不備なために後になって、修正することに支出されている。（うち、約40～50%が仕様設計時のエラーである。）

こうして、70年代に追求されてきた「要求仕様（What）とその実現方法（How）は分離されるべきもの」という方法論は、80年代以降、その分離の困難さが明らかになるにつれ、逆に両者を接近させることで仕様を早期に実行可能とし、現実への適合性を評価することを重視する方法論へと移ってきた。

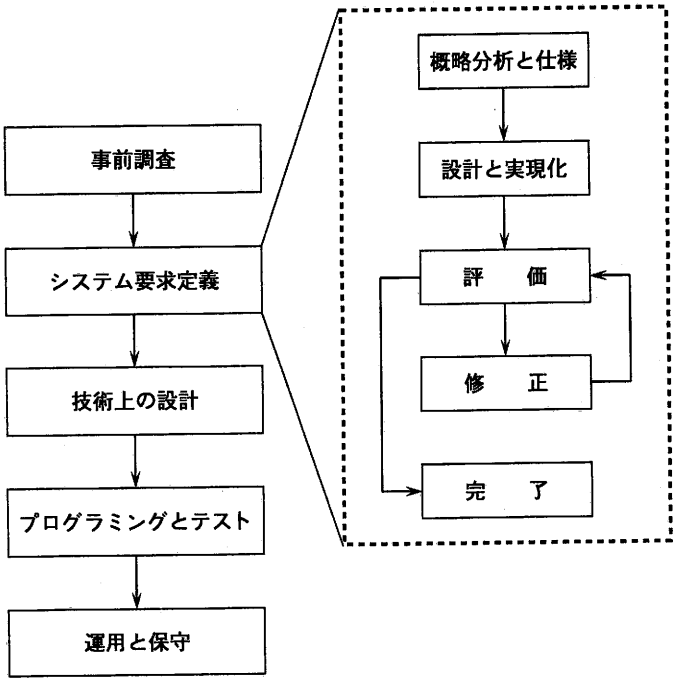
ウォータフォール型モデルが要求仕様の不確実性への対応において硬直的であるとの批判を受けて提起されてきたのが、「プロトタイピング（Prototyping）」（図4－(1), (2)）による開発方法論である。プロトタイピングとは、「最終システムを作る前に、利用者の要求を明確にするために、あるいは設計の実用性を検証するために、実験的な、しかし実際に動く原型（プロトタイプ）を作る」[6]方法である。プロトタイピングが前記の目的を達成するためには、i）実際にある程度、実システムに近い動作が可能で仕様が確認できる環境であること、ii）プロトタイプを開発する期間が短期間であり、かつコスト的に安価であること、iii）作成したプロトタイプの評価、改良が容易におこないうる環境にあること、などの条件が用意されている必要がある。

図4－(1). プロトタイピング・アプローチにおける要求定義段階



しかし、この手法を従来からの大型汎用機を中心としたシステム開発に適用することは、仕様の不確実性を吸収するには一定の効果を持つものの、i）実システムに近い動作環境を設定することが困難である、ii）時間とコストがかかる、iii）プロジェクト管理が困難になる、などの問題があり、充分な普及には至ってこなかったのが現状である。[6]

図４－(2). 開発工程におけるプロトタイピングの位置づけ



出典：R.Vonk: Prototyping, Prentice Hall international (UK), 1990.

以上、歴史的に概観したように、ソフトウェア開発におけるプロジェクト管理の問題(とりわけライフサイクルモデルにおけるプロセス管理の問題)は、長年その問題性が指摘されながらも、大型汎用機をベースとした開発環境では様々な制約があり、根本的な解決を見出すには至っていないのである。これに対し、「ダウンサイジング」により実現される新たな分散環境での開発は、プロジェクト管理にも新たな道をひらく可能性を持っていると思われる。しかし、そうしたハードウェアとしての開発環境の変化により、従来型の開発方法（技法、管理方式）との不適合が生じつつあり、これらも変えなければならないにもかかわらず、現在のところ、その必要性は必ずしも充分に認識されているとは言いがたい。(管理とテクノロジー、組織とテクノロジーの不適合（アンマッチ))

では、「ダウンサイジング」によって実現される環境において、上記のプロジェクト管理問題はどのように展開されるのであろうか？

(2)「ダウンサイジング」下のシステム開発

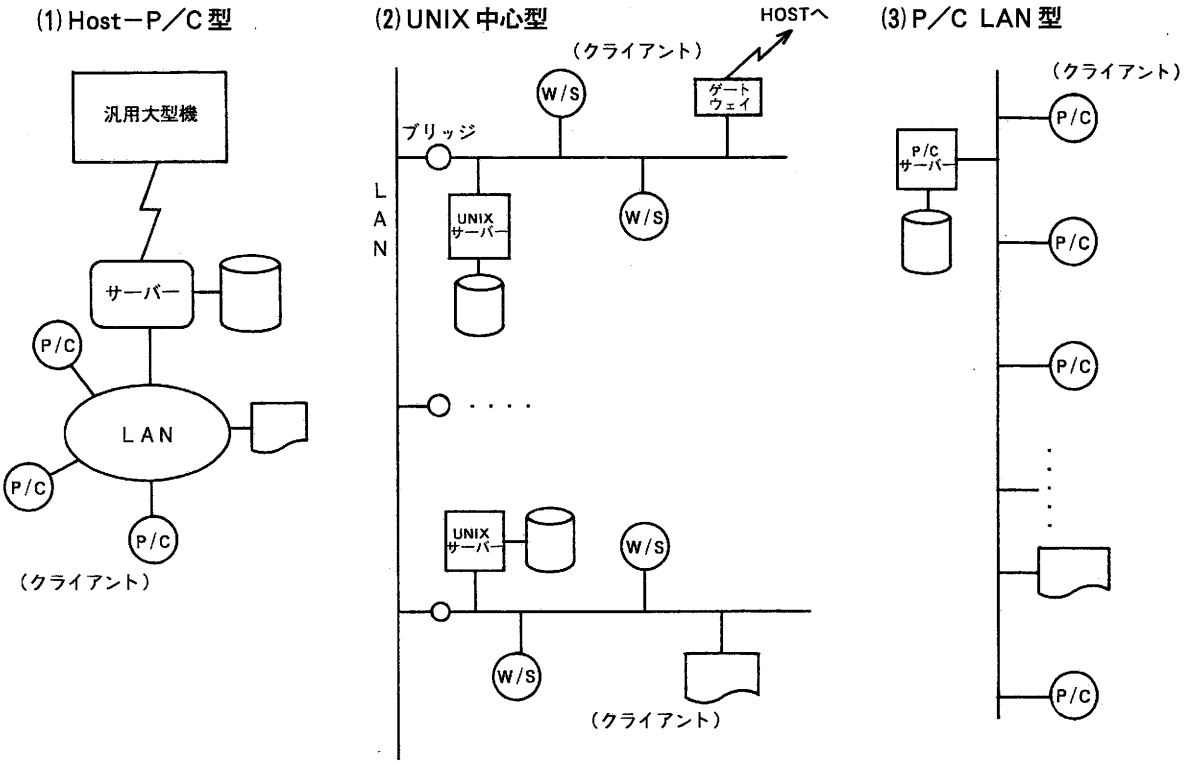
「ダウンサイジング」下のシステム環境の特徴は、i）分散である、ii）ネットワーク化している、ことにあることを前節で述べた。ネットワーキングによる分散環境の代表的な形態は、クライアント－サーバーモデル（Client-Server Model: 以下「C－Sモデル」と略す）である。C－Sモデルによるシステムは、W／S、サーバーとなるマシン、それらを繋ぐネットワークの3つから構成されるのが基本であるが、実際には、どのようなコンピュータをクライアント、サーバーに用いるのか、ネットワークとして何を用いるのかにより、様々な形態が考えられる。（図5）

C－Sモデルの特徴の一つは、ユーザのW／Sがネットワーク（LAN）を通じてサーバー機に特定の処理を依頼し、サーバーはその結果をクライアントであるW／Sに返すという形態をとることであるが、これは、一つ一つのプロセスが協調し、連携をとりながら全体の仕事を進めていくという点で、人間が集団で進める協調的な作業スタイルにより近いものといえよう。

このような「ダウンサイジング」されたシステム環境－それは、前述のようにW／Sとサーバー、ネットワークからなるC－Sモデルであるが－における開発は、従来の汎用大型機をベースとする開発が、まがりなりにも20年以上の経験を持つのに対し、実用面では数年の経験しか蓄積しえていない。そのため、評価の固まった、実証的な開発プロセスがまだ存在しないのが現状である。しかしながら、大型汎用機による環境がプロトタイピングの実現にあたって、多くの制約を持っていたのに対し、C－Sモデルによる分散環境は、プロトタイピングを実現する要素（＝条件）をより多くもっているものと思われる。

プロトタイピングは、見方を変えると従来型のウォータフォール型の開発と比較し、利用部門主導型の開発とも言える。利用部門は本来、システム化の対象となる業務には精通しているものの、それをシステム化にあたって要件として的確に定義できることは稀である。すなわち、利用部門は漠然とした利用要求は持っていても、それがシステムにとってどういう形で説明したらよいかわからないことが多いのであり、システムができて初めて細かな要求に気付き、修正の必要性を認識するのである。その意味で、i）修正を迅速かつ頻繁に行

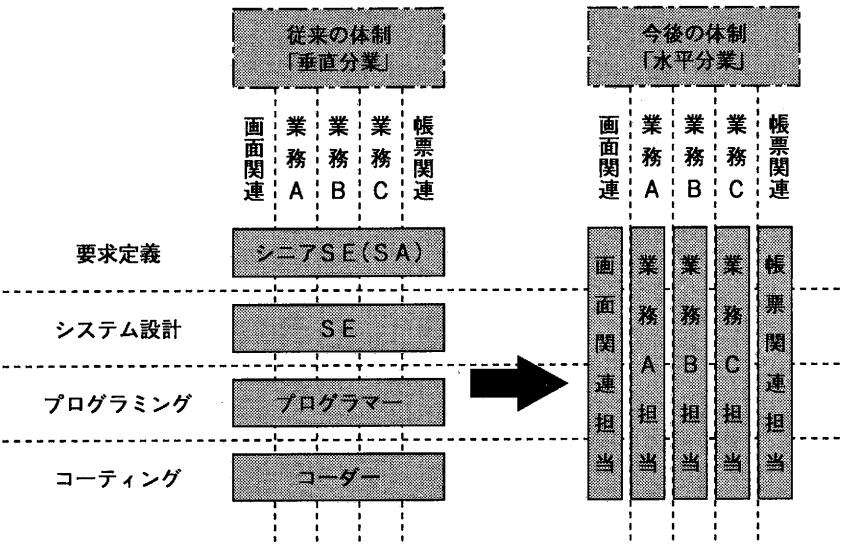
図5．C－Sモデルの諸形態



うことが可能であり、ii）場合によっては利用者自身が修正することが可能である、W／Sによる分散開発環境は、より現実に則した開発環境であるといえる。プロトタイピングを中心とする利用部門主導型の開発を容易にしうる環境が、「ダウンサイジング」によってもたらされつつある。しかし、利用者と開発者の意思疎通が活発になる反面で、一時の思いつきや「誤差」を含んだ仕様情報が氾濫し、開発管理に新たな混乱を持ち込む危険性も孕んでいる。過去の開発例においても、プロトタイピングが、要求仕様の不確実性を克服しつつソフトウェアを開発するのにある程度有効ではあるが、一方で新たな仕様変更を誘発したり、際限のない開発になり、開発管理上の問題を拡大する可能性があることが指摘されている。(7)

次に、「ダウンサイジング」によってもたらされる開発環境について、組織の側面から考察する。一般に、情報ネットワークの利用は、これまで階層的に分断管理されてきた情報を組織的に共有することで、従来の組織におけるミドルマネジメントを排除し、上下階層の直結とタスクフォース化を進めると言われてきた。しかしこれらの組織論からのアプローチの多くは、開発された後の

図6．開発体制の変化



出典：角田好志「ダウンサイジングとパラダイム革新」（『事務と経営』1991．6）に加筆。

実稼働システムを利用する組織が、どのように変化するのかを論じたものであり、そのシステムを開発する組織に及ぼす影響については充分には論じられてこなかった。ソフトウェア開発においても、ネットワークの利用が組織内情報の共有化をはかり、管理上、よりフラットな組織を形成する基盤をあたえる可能性を持つであろう。（C－Sモデルでは、ネットワークを通じシステム仕様に関する各種の情報が組織的に共有され、活用される条件をもつ。）このことは、従来ではフェーズごとに、SA（システムズアナリスト）、SE（システムズエンジニア）、プログラマ、コーダーなどのいわば「上下関係」的な職種階層別にとられてきた分業体制（「垂直分業」）を、担当業務別に対等なプロジェクト参加者として工程全般に責任を持つという新たな分業体制（「水平分業」）へと変えて行き（図6）、将来的には、システム開発の組織をより柔軟かつ自立的なネットワーク組織へ（階層型組織からヒューマンネットワーキングへ）と発展する可能性を与える。[8] しかし、こうしたシステム環境による組織環境の変化は上下の枠を崩しえず、逆に、組織の横の繋がり（組織階層、職種）だけを強化し、より自閉的になるとの指摘もあり、この問題については、CSCW（Computer-Supported Cooperative Work＝コンピュータ支援による共同作業）を実現するグループウェア[9]の利用を含め、今後、実証的に検討していく必要があろう。

(3)「ダウンサイジング」下のシステム開発－事例分析－

最後に、C－Sモデルの環境下でシステム開発を行っている実例を分析することで、前述の記述について検証を行いたい。

この事例を分析対象として取り上げた理由の一つは、この開発を行ったチームが、従来、ホスト主体の大規模なデータベースシステムの開発を主に行ってきたチームであり、過去にも事例研究で取り上げたことがあり[10]、開発環境の違いによる比較が行い易いためである。調査は、1992年夏、書面（質問表）による質疑応答を実施した後に、電話・面談により補足調査を行った。

〔事　例〕

- 開発業務：大手生命保険会社の資産運用システムの開発
- 開発環境（図7）：

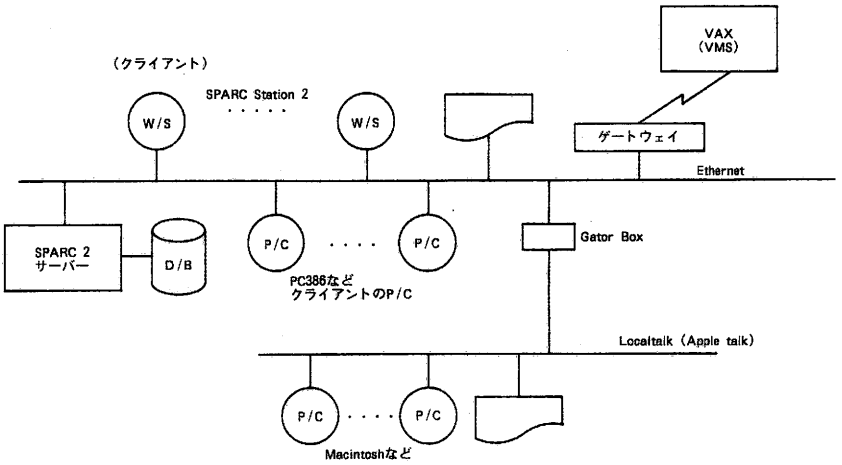
〔ハードウエア〕	SPARC Station 2	7台
	Sun Workstation (diskless)	4台
	X-Terminal (DEC Station)	1台

Macintosh（Quadra700, ci, fx, SE）　5台
Epson PC386　4台

〔ソフトウエア〕 OS：UNIX, Motif, NFS（Network File System）
DBMS：Ingress
その他：Wingz
GUIビルダー（Teleuse）

- 開発規模：言　語　C, SQL, AWK　で約60万ステップ
  工　数　約200人月
  費　用　約3億円
- 開発期間：概　念・基本設計　10か月
  詳細設計・プログラミング　8か月
  総合テスト　2か月
  ユーザテスト　3か月

図7．事例における開発環境



本システムは、生命保険会社が保有・運用している多数の資産の実績管理や予測値を計算することを目的として開発された。システムの持つ機能としては、①資産の持つ価値のポジション把握、②リスク分析、③パフォーマンス評価、④各種シミュレーションなどであり、資産運用に関わる意思決定を総合的に支

援する情報系のシステムと位置づけることができよう。

開発にあたって、下記の組織編成がとられた。

(1)ユーザ（委託側）

- システム企画部門：概念／基本設計段階の理論モデル構築　5人
- システム運用部門：勘定系データのシステム間パス、開発後の運用の視点からの要求とりまとめ　20人
- 資産運用部門　：エンドユーザ要求（外部仕様）の定義　6人

(2)開発者（受託側）

- システムズアナリスト：プロジェクト管理、概念／基本設計　3人
- システムズエンジニア：基本／詳細設計、総合テスト　4人
- プログラマ　：プログラム開発、単体テスト　15人
- テスタ　：テストオペレーションと結果の検証作業　5人

プロジェクト管理の形態は、従来型のウォータフォール型のフェーズド・アプローチを基本としてはいるが、その設計段階では、何度もプロトタイピングが実施されている。段階的にシステムを完成させていくスパイラルモデルなどの様な一貫したプロトタイピング技法ではないが、下記のヒアリングの結果を見ても、プロトタイピングが大きな役割をはたしていることがわかる。

さらに詳細に、このプロジェクトをみると、実際にインプリメント（プログラムメイキング）する初期段階において、データベース、ＧＵＩ（グラフィックユーザインタフェース）、通信などの各技術ごとで分割したチームを編成し、モジュールインターフェイスの標準化したルーチンを作成している。これにより、開発の中期段階以降には、この標準ルーチンとＵＮＩＸ、Ｃ言語の一般的な知識さえあれば開発のメンバーとして投入することができるようになっている。開発チームは、これら標準ルーチンの作成を含めプログラムスケルトン生成ツールとプロジェクト管理ツールを自前で開発して生産性向上を図っている。

本開発業務を実務面で統括したプロジェクトマネージャーにヒアリングした大型機による開発との違いは下記の通りである。

〔問〕汎用機を中心とする開発と比較して開発の方法論に変化はあるか？それは具体的にはどういう変化か?

〔答〕
- オブジェクト指向による開発方法論など、新しい方法論を導入する場合にそれを支援する各種環境があるので取り込みやすい。
- ユーザインターフェイスについて、委託者側の要求水準が高くなり、

プロトタイピング手法を取り入れて、要求水準を満たしているか確認する場合が多くなる。

〔問〕汎用機を中心とする開発と比較して、プロジェクト管理は容易になったか？それとも難しくなったか？（進捗管理、工程管理、予算管理、開発環境管理、外注管理、品質管理、要員管理など各々についてどうか）

〔答〕
- 予算管理…マシン使用に関わる費用が汎用機の場合は、複数の部門による共同利用であったことから使用ＣＰＵ時間に対して課金されていたものが、部門の専有であるW／Ｓになったことで、マシン費用が固定費として扱えるようになり、その点では予算の管理は容易になった。
- 開発環境管理…汎用機の場合は、別の組織であるマシン環境管理部門（コンピュータ管理部）が管理し、開発部門はそれを利用させてもらう形態だったが、W／Ｓになったことで、その管理機能を開発プロジェクトの内部に持つ必要が出てきた。プロジェクトの都合に合わせて柔軟に管理ができる利点がある（従来は、汎用機を稼働するための専任オペレータや同一マシン上で稼働している他のシステムの影響を受け、様々な制約があった）一方で、機器メンテナンスなどの保守管理業務が新たに発生し負荷が増えている面もある。
- 品質管理…ＤＢＭＳなど基本ソフトウェアのパフォーマンス、信頼性に依存する部分が多くなるのでそれに関する設計上の考慮、チューニングの負荷が大きくなる。
- そ の 他…あまり変わらない。

〔問〕汎用機を中心とする開発と比較して、ＳＥ、プログラマなどの職務範囲に変化はあるか？それは具体的にはどういう変化か？

〔答〕ＳＥが従来より広く工程全般を管理する必要が出てきたために、ＳＥにシステム管理者としての機能と能力が要求される。さらに、オブジェクト指向の開発方法論をとる場合には、ＳＥがプログラム設計にも、より深く関与しなければならなくなる。

〔問〕汎用機を中心とする開発と比較して、ユーザとの仕様確認に変化はあるか？それは具体的にはどういう変化か？

〔答〕ＧＵＩ（グラフィカルユーザインタフェース）に対する要求水準が高くなるので、プロトタイプなどにより仕様確認を充分行う必要がある。

同時に、アプリケーションがエンドユーザ主導の設計になることが多いので、仕様確認段階でのユーザとのコンタクトが増える。エンドユーザ側のリーダシップをとる人間（意思決定者）によって、仕様確定までの時間が大きく左右される。

〔問〕汎用機を中心とする開発と比較して、作業の「手戻り」に変化はあるか？それは具体的にはどういう変化か？

〔答〕計画時に想定していたデータ量に見積りミスがあったり、それによりＤＢＭＳの限界能力に達するような場合（データ量に起因する障害としては、Ｗ／Ｓのメモリ、ＣＰＵパフォーマンスも同様）、汎用機以上に設計の根幹に関わる重大な変更が生じる可能性が高い。

〔問〕汎用機を中心とする開発と比較して、仕事を進める上で発生するストレスに変化はあるか？それは具体的にはどういう変化か？

〔答〕Ｗ／Ｓの開発環境－特にWindowを中心とした画面環境－はプログラマにとては、ストレスを減少させる要因となっている。開発中のトラブルによりマシンダウンを引き起こしたり、プログラムが無限ループに入った場合など、開発環境が汎用機と異なりプロジェクトに限定されているため、影響が他に及ばないのでそういう意味でのプレッシャは減少した。(11)

〔問〕その他、汎用機を中心とする開発と比較して、プロジェクトを進める上で変化があるか？

〔答〕汎用機を中心とした開発と比較すると、まだ開発ツール・開発方法論が固まっておらず、良質、優秀な人材の工夫によって開発が進められているのが現状である。その意味では、汎用機主体の開発より、人材の質（個人の能力）に頼る割合がより高くなっている。

〔問〕いわゆる「ダウンサイジング」について開発側としての考えがあるか？

〔答〕「ダウンサイジング」は必然の方向だと思われる。しかし、今後は開発だけのコスト評価ではなく、保守・運用を含めたシステム全体のライフサイクルコストをいかにして低減させられるかが課題となろう。

上記のヒアリングの回答は、「ダウンサイジング」によってもたらされる開発環境が、汎用機を中心とした開発では困難であったプロトタイピングによる開発により、柔軟な仕様変更への対応を技術的に可能とすることを示している。しかし、その一方で、様々な管理業務とプロトタイピングを行うことによって

発生する新たな仕様（それは、利用者の「学習」によって発生するものである[12]が）への対応は、ＳＥの職務範囲を拡大し、かえって個人の能力への依存率を高めていることを示している。従来からただでさえ多忙な職種の代表といわれてきたＳＥに、一方で開発環境管理まで含んだより広い職務範囲の管理者能力を要求し、さらに一方ではプログラム開発にまで深く関与せざるを得ない状況を与えるようになってきているのである。こうした従来の「ＳＥ」という概念では範囲外であった業務にまで職務範囲が拡大するという変化は単に、プロトタイピングや分散開発環境だけから進行している現象ではない。「ダウンサイジング」の進展と平行して普及しつつあるＣＡＳＥツール[13]の登場は、その能力においてまだまだ不十分ながらもプロトタイピングをサポートすることで従来の開発工程を大きく変えるとともに、ＳＥ、プログラマといった分業構造をも変えつつある。

さらに、こうした「ダウンサイジング」の進展は従来のシステム開発における業界全体の分業構造を大きく変える可能性をもっている。いわば「垂直分業」から「水平分業」への構造変革は、単に一プロジェクトチーム・一企業の枠組を超えて、中央：地方での地域間分業構造を変える可能性をもっている。すなわち、従来、中央（東京・大阪などの大都市圏）で発生するソフトウェア開発の多くは、都市圏に所在する「元請」会社が一括受注する形態をとるものの、実態としては顧客との頻繁な折衝が要求される営業やシステム分析、基本設計など上流工程部分のみを「元請」会社が実施し、それ以降のフェーズである詳細設計、プログラミングなどは地方に所在する「下請」会社が実施するといった、地域間分業によって作成されてきた。（図８）この分業構造の枠組みの前提である、フェーズドアプローチが「ダウンサイジング」によって大きく変わるのである。

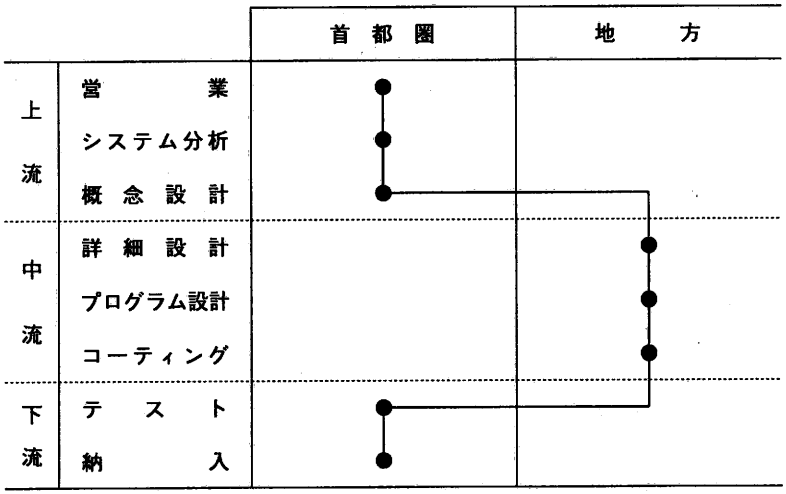
以上の論点に加えて、「ダウンサイジング」によってもたらされるシステム開発環境においては、一般に下記の問題点が指摘されている。

①新しい技術（Ｗ／Ｓ、ＤＢ／ＤＣ、ＵＮＩＸ、Ｃ－Ｓシステムなど）に精通した技術者が不足している。

②新たな分散開発環境に対応した管理者を設置する必要があり、同時に、そのような管理能力を持った管理者を育成する必要がある。

③新たな開発環境に適合したプロジェクト管理技術が未確立であり、各種管理技法が技術進歩へ追いつかない。

図8．都市圏と地方の分業関係



出典：今野浩一郎『ソフトウェア産業と経営』（東洋経済新報社・1990）

④開発のライフサイクルの変更により、開発期間が短期間になり、かつ各種開発業務がその期間に集中化する。

⑤プロトタイピングの実施とユーザの設計作業関与により、仕様の不確実性が増大する。[14]

⑥マルチベンダー化にともなう調整作業が増大する。オープン化にともない複数社のH／Wを使用することになるが、それら機器の互換性の作り込みと検証にかかる負担が増加している。[15] これは、通信プロトコルも同様である。

上記の論点および問題点をまとめると、「ダウンサイジング」がソフトウェア開発に従事する多くの者にとって、あるべき方向なのかということは、単純に利用者の利便性やコストなどで考えられてきた従来の論議とは別の次元で論議される必要があることが明らかとなる。今後、これらの問題点をどう克服していくかが、「ダウンサイジング」下でのソフトウェア開発の成否を決めるであろう。

（1） ソフトウェアの生産性が論議されるようになった最初は、1968年にＮＡＴＯが当時の西ドイツ・ガルミッシュで開催したソフトウェア工学会議と言われている。この会議の中で、「ソフトウェア危機」という言葉が生まれた。

（2） 例えば、ソフトウェア開発における残業発生要因の上位は開発管理関係（なかでも仕様の不確実性の問題）が占める。（『情報サービス業における労働時間短縮の現状と今後の方向』労働省・1989年）また、別の調査では、「ソフトウェア開発プロジェクトの主要な問題がどこに起因しているか？」との問いに対し、日本の57.1%、欧米の45.0%が「プロジェクト管理」と回答している。（「ソフトウェア開発プロセスの実態調査」，ソフトウェア技術者協会，1992）

（3） 「構造化手法」…ソフトウェアの規模限界説に起因する危機感から、ソフトウェアプログラムの構造化が提起された。具体的には、Dijkstara らのGO TO LESSプログラムや国際規格ISO 8631 Program construct and convensions for their representation となって用いられている。

（4） 「段階的詳細化」…問題を解決する理想的な仮想システムをまず想定し、仮想システムの命令とデータ型の分解を排他的に実施しつつ、プログラム動作の具体化を進めていく方法。（N.Wirth:Program Development by Stepwise Refinement, CACM, vol.14, No.4, pp.221-227, ACM, 1971）

（5） 玉井哲雄「ソフトウェア開発におけるプロトタイピング」（『bit』vol15, no13, p11, 1983.12）

（6） 拙稿「大規模システム開発におけるプロジェクト管理問題」（『ソフトウェアシンポジウム'91』pp.D15-D16）

（7） ibid.pp.D13-D14

（8） ネットワーキングが組織に与える影響については、今井賢一編著『ソフトウェア進化論』（ＮＴＴ出版、1989）、C.M.Savage: "FIFTH GENERATION MANAGEMENT（DEC, 1990）などで議論が展開されている。

（9） 「グループウェア」…グループによる共同作業を支援するために作成されるコンピュータシステムの総称。ＣＳＣＷ（Computer-supported Cooperative Work:コンピュータによる協調作業支援）の概念を具体的に実現するためのシステム（なかでもソフトウェア）をいう。グループウェアの機能として、「ＷＹＳＩＷＩＳ（What you see is what I see）：あなたが見ているものを私もみている」という言葉に代表されるような、共同作業のコミュニケーション支援機能、共同作業記録機能などを持つ。具体例としては、開発中のシステムのソフトウェアや仕様書を画面に提示し、それに対して通信回線で結ばれた各開発者のＷ／Ｓから意見を提示したり修正したりしながら（つまりあたかも机を共有するような作業環境で）会議が進めることができるような環境が既に

一部で実現している。（『日経産業新聞』1992.8.31）グループウェアはその形態と機能によって、①電子メール系（グループ内における電子メール、伝言板、スケジュール管理などのソフトウェア）、②テレビ会議系（動画のリアルタイム相互伝送の機能拡張）、③ハイパーメディア系（オブジェクト指向ＤＢでの図表・画像・音声データの関連付けによるノウハウの共有化）などに大別される。

(10) 拙稿「大規模システム開発におけるプロジェクト管理問題」(『ソフトウェアシンポジウム'91』pp.D12.D15)

(11) なお、プロトタイピングが技術者のストレスを軽減することには役立たないという説もある。(藤垣裕子『ソフトウェア技術者の職業性ストレス』P.P49-50, 労働科学研究所出版部, 1992)

(12) 拙稿「大規模システム開発におけるプロジェクト管理問題」(『ソフトウェアシンポジウム'91』pp.D15-D16)

(13) ＣＡＳＥ…Computer Aided Software Engineering の略。コンピュータによりシステム開発のライフサイクル全体あるいは一部を支援・自動化すること。システム開発に構造化分析技法（Structured Analysis）などソフトウェア工学の手法を持ち込み、その概念にもとづくツールCASE Tool）を利用することで、開発の迅速化、低コスト化を図る。近年、ソフトウェア開発を上流工程から下流工程までサポートするツールとして、様々なＣＡＳＥツールの開発と導入が進められてきているが、評価は定まっていない。Ｃ－Ｓ型アプリケーションのソフトウェア開発を効率良く進めるためには、ＣとＳに分割されたアプリケーション設計から生成を行うＣＡＳＥツールが必要となる。さらに、発展すればいわゆる分散協調型アプリケーションに対応したＣＡＳＥツールが必要であるが製品としてはまだ少なく、不十分である。

(14) ユーザーが設計段階から積極的に参画しながらＳＥとともに仕様を記述する場合、どうしても機能中心のフローとなり客観的なレビューが困難となりエラーが見過ごされやすくなる。

(15) 『日経エレクトロニクス』(1992.4.27号) など。

おわりに　－総括と今後の課題－

以上、本論では「ダウンサイジング」の本質がどこにあるかを定義し、それによってもたらされるシステム開発環境がどのようなものであり、システム開発労働がソフトウェア開発管理から見てどのように変わる可能性を持つのかを論じてきた。その結論を総括すると下記の３点になる。

①「ダウンサイジング」を単に「小型化」として捉えることは誤りである。「ダウンサイジング」の本質は、ネットワークを介した分散処理への移行にある。

②「ダウンサイジング」により実現されるシステム環境はソフトウェア開発のあり方を大きく変える。「ダウンサイジング」の進展に合わせて、ソフトウェアの開発面においても本格的なプロジェクト管理のパラダイム変革が必要であり、分散開発環境下でのプロトタイピングを中心としたライフサイクルへの移行がその中心となる。

③分散開発環境下でのプロトタイピングを中心としたライフサイクルは、ソフトウェア開発における従来の分業構造を変える。また、それにともない、ＳＥの職務範囲が拡大するが、開発環境に適合した管理方法が不在な現状では、新たな混乱を招く恐れがある。

継続的かつ急速な技術革新により、ハードウェアのコストパフォーマンスは飛躍的に向上したが、その一方で、商品としての非差別化が進行している。このことは、コンピュータシステムにおいて、ハードウェアがもつ付加価値の相対的な減少（逆に言えば、ソフトウェアに対する相対的な増加）を意味し、広い意味では情報産業全体の付加価値構造の変化を意味する。従来では、（ハードウェア、ソフトウェアを共に含む広義の意味での）情報産業における付加価値の源泉は、圧倒的にハードウェアにあったが、「ダウンサイジング」の進行は、それをソフトウェア（サービスを含む）に移行することを促している。また、特に近年のＧＵＩやマルチメディア技術の急速な発展は、ユーザーインターフェイス部分の設計の高度化を求めてきているが、従来型のメインフレームを中心とするＣＡＳＥやプログラムジェネレータでは限界が指摘されている。これらの分野は、今後のソフトウェアエンジニアリングの中で、最も人間的な創造性が期待されうる部分である。今後はそうしたソフトウェアの高付加価値化といった意味からも、より一層ソフトウェア開発のあり方が重要となってくるであろう。

究極のソフトウェアエンジニアリングは、「使いたい人間が自分でシステムを作る」というＥＵＣ（エンドユーザコンピューティング）、ＥＵＤ（エンドユーザデベロップメント）かもしれない。装置技術とソフトウェア開発の技術が一層進展することにより、エンドユーザ自身の手によるソフトウェア開発が一般化する可能性がある。もちろん、現実にはまだそうした状況が急展開され

るほどのシステム開発支援環境は望めないし、また、近年の事例では、ＥＵＣ、ＥＵＤを追求した結果、システム管理の体系を大きく逸脱したアナーキーな開発が横行し、組織の情報システム全体に対してだれも把握しておらず、誰も責任をとっていないという、一昔前の状況が再現されつつあることが指摘されている。(1)

しかし、大きな流れとして進むＥＵＣ、ＥＵＤの潮流は、やがては従来からソフトウェア開発に従事していた労働者を、高度な技術と業務知識をもったコンサルタントと運用・保守要員に二極分化し、従来型のプログラマ、ＳＥを淘汰する可能性をもつ。実際、昨今の景気停滞に伴い申請された「雇用調整助成金」の支給申請のうち、実に64.6%（95事業所）がソフトウェア業によって占められた(2)が、今後、景気が回復したからといってソフトウェア業界の経営環境が以前のようになる（大手ユーザー企業の大規模な投資の再開）とは、考えられにくい。エンドユーザが直接システム構築を行いうる環境ができつつある時に、従来からの「とりあえずプログラミングができる」だけの人材がいつまで必要とされるであろうか。「ダウンサイジング」の進行は究極的には、ソフトウェア開発のための単純な労働力を提供することしかできない会社を淘汰するであろう。(3)実際、すでにプログラマの余剰感が高まっているのである。(4)

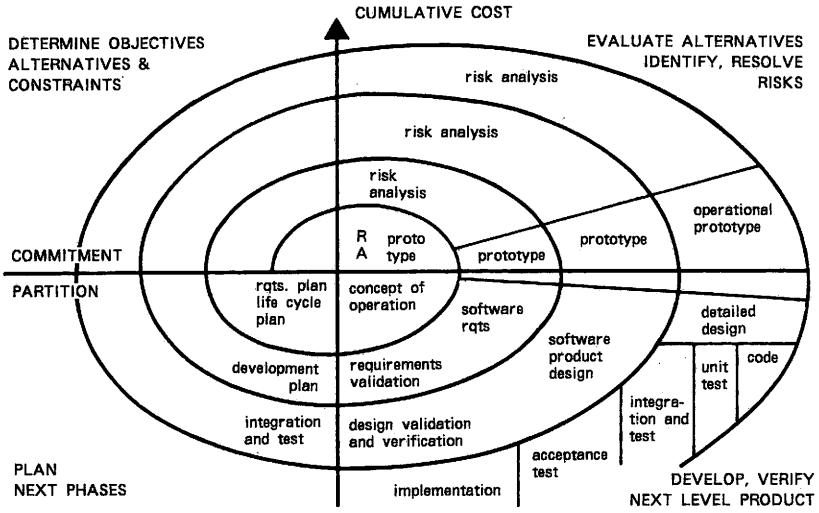
このように旧来どおりのソフトウェア開発を業務とする企業にとって、現在の経営環境は厳しいものがあり、「ダウンサイジング」は決して楽観的な見通しを提示するものではない。しかし、困難な時期である今こそ、「ダウンサイジング」の真の意味を見据えた経営戦略の転換とそれに見合った人材育成戦略を確立することによって、ソフトウェア業界の構造改革を実施することが急務の課題として提起されているのである。

（なお、本稿の事例分析にあたっては、三井情報開発株式会社の佐枝三郎部長をはじめとする多くの方々のご協力を頂きました。ここに記して厚く御礼申し上げます。）

（1） 分散化したシステムは、いずれどこかでその方向性や効率をチェックし、統合する必然があろう。これからのシステムは分散→統合→分散の繰り返しとなる部分もありうる。例として、中堅ゼネコン会社の日本国土開発において構築中のトー

タルマネジメントシステムがあげられる。このシステムはC－Sシステムで、ソフトウェアの開発にあたっては「ソフトウェアの開発の途中で逐次、使う側の意見を採り入れていくプロトタイプ方式の開発」を念頭にして構築されている。このプロトタイプを各支店で使い勝手のよいものにするためソフトウェアの遠隔メンテナンスを実施したうえで、成長した各プロトタイプを再統合する予定となっている。(『日経産業新聞』1992.9.20) 将来のソフトウェア開発は、こうした長いライフサイクルで考えたステップバイステップ、スパイラルな開発モデルを前提とすることになるろう。(図9)

図9．スパイラルモデル（Spiral Model）



出典：B.W.Boehm: A Spiral Model of Software Development and Enhancement, (Proceedings of International Workshop on the Software Process and Software Environments, 1985).

（2）『日本経済新聞』(1992.11.17)

（3）すでにかなりの数の倒産が発生している。92年1月～5月で1000万円以上の負債で倒産したソフトウェア開発の企業が30件を超えている。(帝国データバンク調査、『朝日新聞』1992.6.16)

（4）『情報サービス産業動態統計月報』、『日経コンピュータ』1992.7.13号など。