U.D.C. 681.32.06.002:658.5.011.4/.5

ソフトウェア生産の諸問題とその生産技術の動向

Problems and Trends in Software Development Technology

青山義彦* *Yoshihiko Aoyama*
立川昭三** *Shôzô Tatekawa*

従来ソフトウェアは，ハードウェアに付属した個人的産物(財産)の意義しか与えられていない傾向が強かったが，ソフトウェア自身市場価値をもつ時代になってくるとともに，ソフトウェア開発への需要量が増大してきた。そのためには，近代的生産技術を活用したソフトウェアでなければならないし，それにこたえる生産技術の開発が重要性をもってくる。

ソフトウェア開発の諸問題と，過去の経験あるいは各種の論文を参考にし，ソフトウェア生産技術の体系化を行なった。ソフトウェア生産技術は多面的であり，まだ客観的に評価が定まったものばかりとはいえない状況にあるが，個々には有効な技法・ツールが開発され効果を挙げてきている。まだ研究開発に期待される部分が多いが，本稿ではそれらの点を明らかにした。

1 緒　言

ハードウェア技術の急速な進歩による性能向上，あるいはコストパーフォーマンスの飛躍的改善に比べると，ソフトウェアの生産性向上は，あまりにも低すぎると言われてきた。

システムの多様化，高度化及び大規模化に加えて，マイクロコンピュータの出現により，ソフトウェアはあらゆる産業分野の製品に入り込み，量的にも，質的にも，その重要性が高まりつつある。コンピュータメーカーにとっては，ハードウェアコストよりも，ソフトウェア開発コストのほうが，実際に大きな割合を占めつつある[1)~3)]。またユーザーでも，コンピュータの利用領域を多様化し，拡大するためには，どうしても新しいソフトウェアを開発していくことが必要であり，また過去開発したソフトウェアを維持管理していかなければならないため，ソフトウェア開発に対する投資を増大せざるを得ない状況にある[4)]。

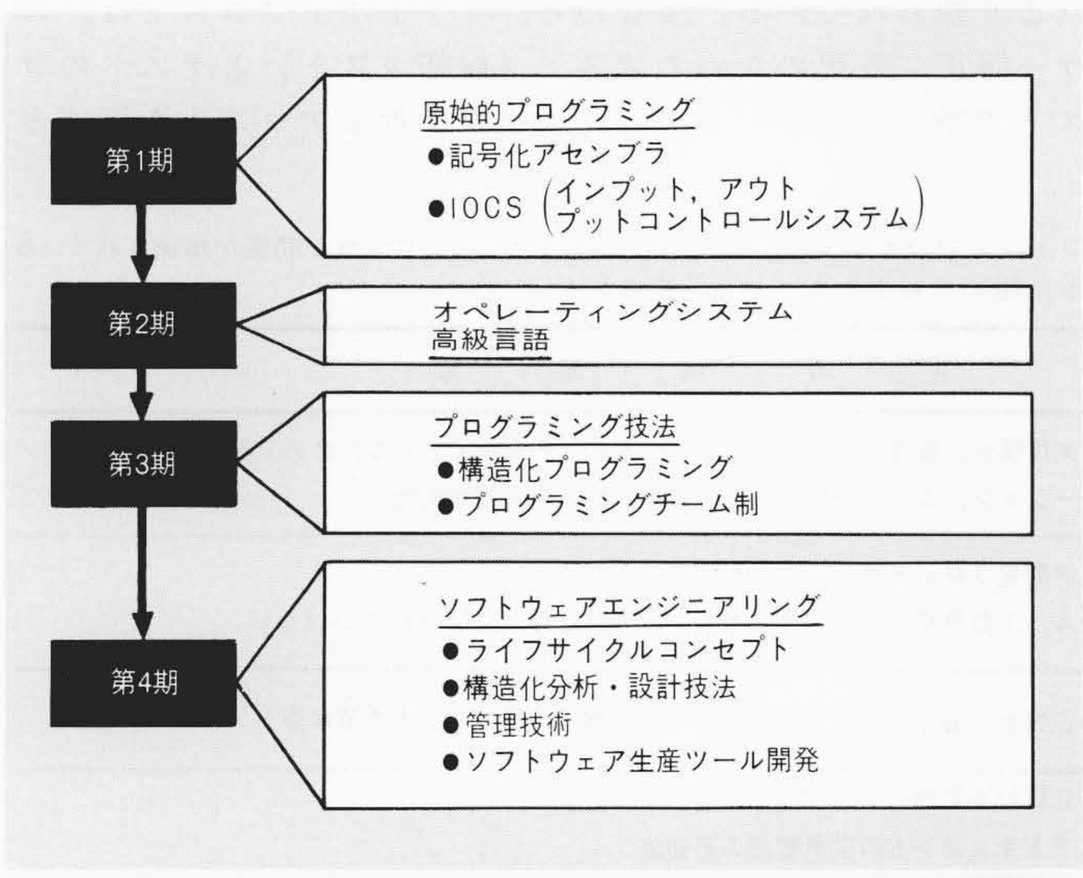

図1　ソフトウェア生産の環境の移り変わり　ソフトウェア生産が，時代とともにどのように改善されてきたかを概観したものである。現在は第4期の段階に入ってきている。

このような状況に対応するためには，ソフトウェア生産技術が，個人的，技巧的な生産方式から脱却し，より合理的で近代的な生産方式へと発展しなければならない段階にきている。高品質で低価格なソフトウェア生産技術が強く望まれるゆえんである。

従来ソフトウェアは，紙と鉛筆，それに言語プロセッサといった原始的道具を使って，必ずしも正規の体系的エンジニアリング教育を受けていない個人の技量に全面的に依存して開発されてきたといっても過言ではない。ソフトウェア生産性向上のためには，こうした家内工業的生産手段では限界があり，高度な生産手段がどうしても必要である。ソフトウェアの生産手段をいかに高度化していくか，そのための研究開発が必要であることが「ソフトウェアエンジニアリング」という言葉に込められていると考える[1)]。

本論文では，ソフトウェア生産の諸問題を概観し，それに対応する技術の状況・動向について整理するとともに，本特集に掲載紹介される各個別論文への橋渡しの役割をもつものである。

2 ソフトウェア開発の環境変化と生産技術

ソフトウェア開発の環境の移り変わりをみることによって，ソフトウェア生産技術の現状を示したのが**図1**である。これについては詳しく説明する必要はないと考えるが，ソフトウェア生産技術の観点からみると，過去10年間はプログラミングフェーズを中心としてのプログラミング技法，及び支援システムの開発と実用化が中心であったと考えてよい(**図2**)。

3 ソフトウェアライフサイクルと問題の所在

ソフトウェアの開発過程は，**表1**に示すように「ニーズ分析」に始まって「テスト」，「運用・保守」の過程を経ることはよく知られている。すなわち，いわゆる「ソフトウェアライフサイクル」と呼ばれるものである。このソフトウェア開発過程での主な問題点を**表1**，**図3**に示したが，指摘される問題を総括してみると「生産性」，「信頼性」，「保守性」及び「管理性」の四つの観点からみることができる。

* 日立製作所システム開発研究所　** 日立製作所生産技術部

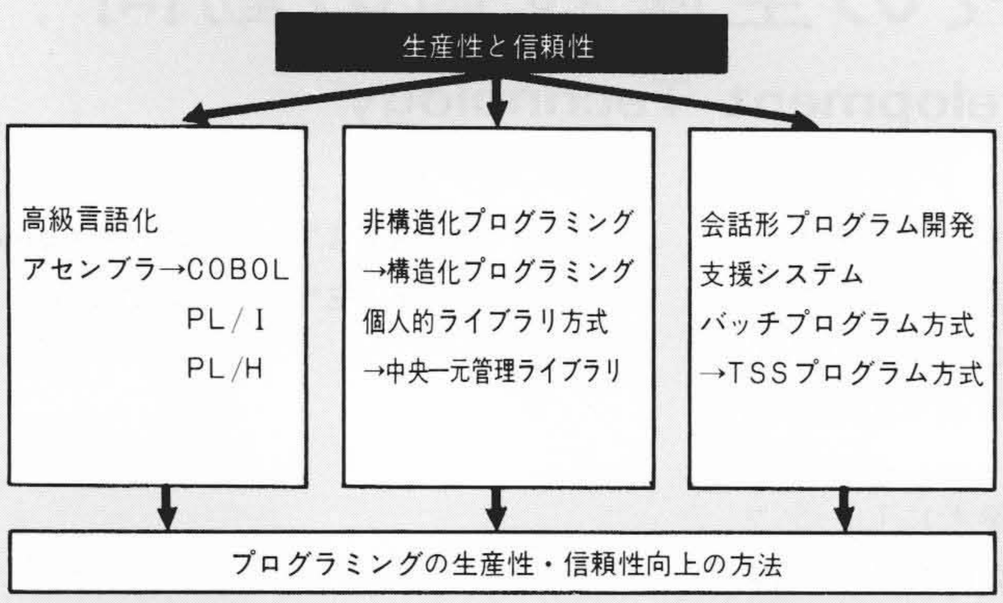

図2 プログラミングフェーズの生産性・信頼性向上の方法
過去のソフトウェア生産のための技術は，ソフトウェアライフサイクルの中でプログラミングフェーズに焦点があった。その代表的なものを図式化したものである。

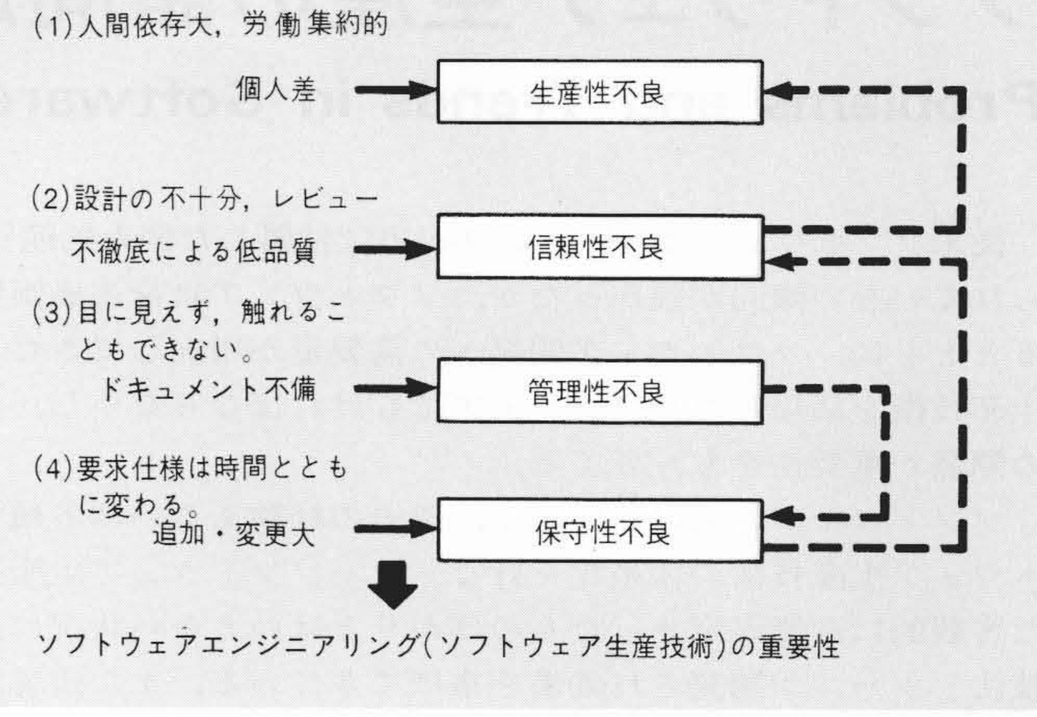

図3 ソフトウェア生産の問題点 ソフトウェア生産の問題点は多くあるが，まとめてみると四つの観点に集約できることを示した。これらの問題を解決する技術が，ソフトウェアエンジニアリングという言葉に集約されているといえる。

3.1 生産性

ソフトウェアの開発は，ペーパービジネスであるといわれるぐらい，紙と鉛筆を中心に，ソフトウェア技術者の労力・知力に負うところが大であり，実に人間への依存度が高く，労働集約的で家内工業生産的である。すなわち，コストは人件費が大半を占め（おそらく70%以上と想定される。）生産性が低いと言われるゆえんである。

(1) 人間に依存するだけに，ソフトウェア技術者の個人能力差が生産性に大きく影響せざるを得ない。サックマン，エリックソン及びグラントの調査[5]によると，「優れた仕事をする人と劣った仕事をする人では，ソフトウェア生産量で約10：1，プログラムの処理速度とメモリサイズで5：1の比率になる。」という。ソフトウェアは，人の問題に帰着するといっても過言ではないだけに，教育，適正管理及び生産運用方式が基本的に重要性をもっている。

(2) また過去，ソフトウェアの生産性は，それほど向上していないと言われている。C. ワイズマンによると[6]，「過去10年間のソフトウェアの生産性は数字の上では，2倍しか上昇していないのに対し，ハードウェアの生産性は，約50倍近くも上昇している。」として，ハードウェアの生産性に比べて，ソフトウェアの生産性の向上が重要な課題であることを指摘している。

(3) **図4**は，ソフトウェア開発の中で占める作業要素の比率を示したものである[7]。生産性向上のために，何をなすべきかを知ることができるが，一つの方向は，文書化を中心とした機械化による支援システムの開発[8]であろう。

3.2 信頼性

第2の問題は，ソフトウェアコストの割に信頼性が低いことである。マイヤーズによれば，[9]「ソフトウェアの信頼性は，過去は冗談の対象であったが，今や生きるか，死ぬかの問題になってきた。」と信頼性の重要性を強調している。システム及びソフトウェアの大規模化により，ソフトウェアのエラー，事故はその影響範囲を拡大し，人間生活の生死にかかわる問題と直結している。

(1) この信頼性は，先の生産性と表裏一体の関係にあることはよく知られている。すなわち，ソフトウェアコストは，エラー修正，変更のためのコスト（保守コスト）とテストのコストで多くの割合が占められ，ソフトウェアコストを下げる

表1 ソフトウェアライフサイクルとソフトウェア生産の問題点 ソフトウェアライフサイクルでの各フェーズで，どのような問題が指摘されているかをまとめたものである。それぞれの問題は互いに関係しているので，問題点の関係を見極めて対応することが必要である。

ライフサイクル(フェーズ)	機能	主な問題点
ニーズ分析	ユーザーが何を望んでいるか。 (目的と手段の明確化)	●システムの大規模化，複雑化に伴うシステムの真の目的を明確化するための工数大。 ●コミュニケーション，コンセンサス不十分のまま終わらせる傾向大。
システム計画	プロジェクトの到達点・目標設定 (スケジュール・工数見積など)	●ソフトウェア開発プロジェクトの計画が貧弱である。 ●スケジュール，工数見積が不正確で実現までの追跡管理へ結び付けていけない。
システム設計	システム全体の動作の記述 (適切な機能の設定)	●要求仕様の正確かつ安全な記述法が未確立で，仕様の一貫性がなく矛盾が多くテスト不可能。
ソフトウェア設計	実現すべきソフトウェアの方式設計 (性能確保)	●モジュール化設計を簡略化する傾向がある。 ●ソフトウェアドキュメントの変更管理の不徹底
プログラミング	計算機言語への翻訳 (コーディング・デバッグ)	●プログラムの「良しあし」の実際的計測法がない。 ●個人差が大
テスト	単体テスト・総合テスト (品質の検証)	●ソフトウェアのテスト手順及びツールの不足 ●ソフトウェアの信頼性を測定，予測する方法がない。
運用・保守	システムの改修・改良	●保守しやすいソフトウェアを開発する技術の未確立 ●保守のためのドキュメント未整備

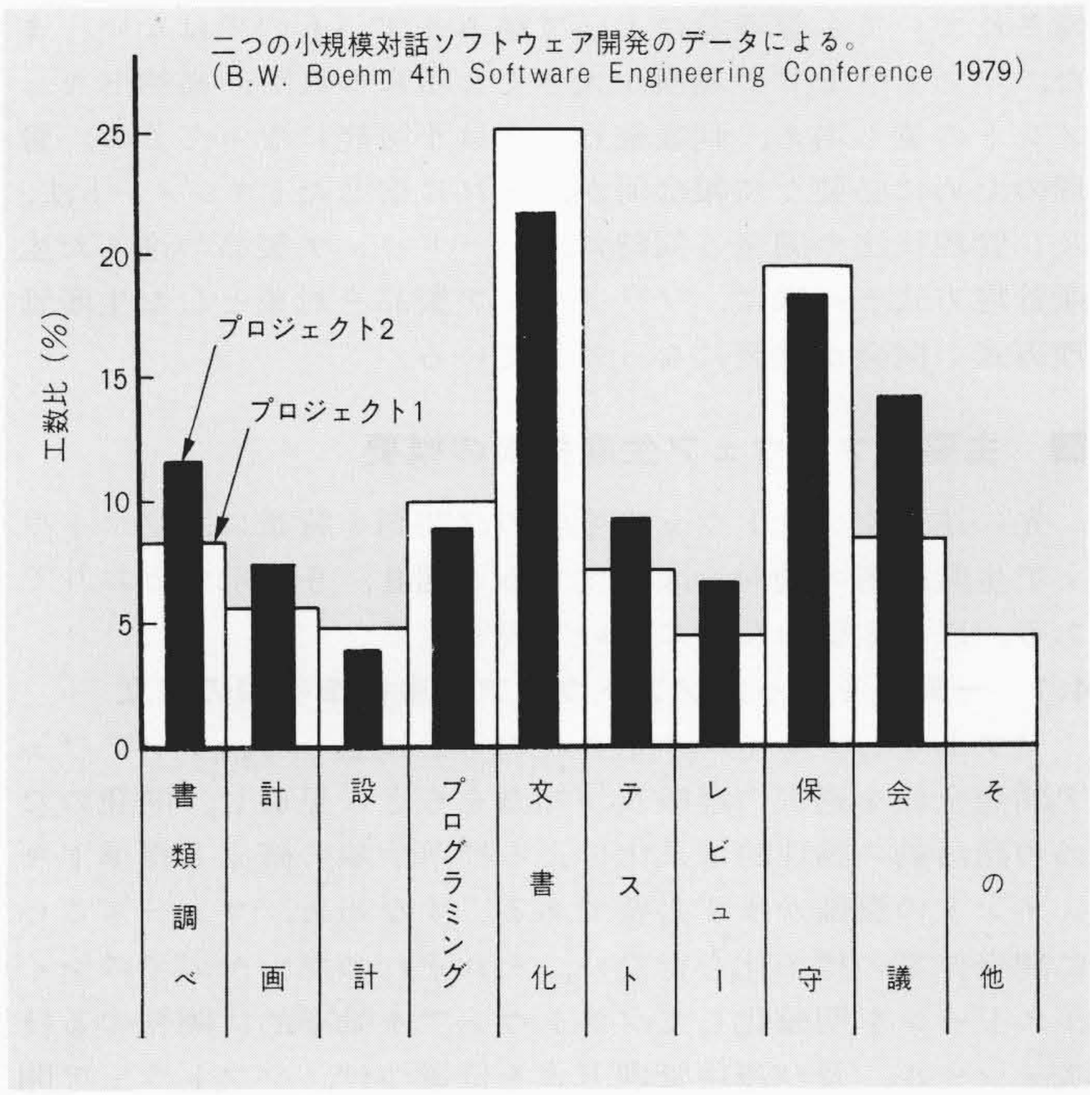

図4　ソフトウェア生産における作業要素　ソフトウェアを生産する場合の作業を，要素別にその工数比率をみたもので，文書化作業が大きな割合を占めていることが分かる。生産性向上のために合理化すべき部分はどこかがよく分かる。

ためには，保守コストとテストコストを下げることが大切である[2),3)]。

(2) **図5**は，ソフトウェアのエラーが埋め込まれる時期と発見される時期の関係を示したものである。ソフトウェアのエラーの大多数が，ソフトウェア開発の初期段階で埋め込まれてしまう。ソフトウェアのエラーの種類として，プログラムエラー，管理上のエラーなどがあるが，最も重要なエラーは，ニーズ分析，設計過程で生ずるエラーが大きいことを示している（**図6**）。そして後工程にいけばいくほど発見が困難で，修正のためのコストは大きくなっていく[10)]。

要求仕様化技術，ソフトウェア設計技術が未確立であったことと同時に，このフェーズがプロジェクトメンバー間の意見の食い違いを残したまま後工程へと進んでしまう傾向を本質的にもっていることが大きな原因と言える。

(3) ソフトウェアのエラーを考える場合，重要なことは，ソフトウェア自体に内在する問題と，ユーザーが期待しているとおりにソフトウェアが実現されていないことから生ずる二つの側面をもっていることである[10)]。前者は，正しく精密に設計できる技術，手段の開発の問題であり，設計結果，プログラミング結果を十分検証してテストを行なう方法（テスト技術）を確立することであるが[2),3)]，後者の問題は，ユーザーが正しく適切に問題を表現し，同時にユーザーの要求をソフトウェア技術者が正しく理解しなければならないという難しい問題を含んでおり，要求を正しく分析[11)]し，記述する技術及びユーザーと設計者のコミュニケーションの方法（仕様検証技術）の確立[12)]が望まれることになる。

3.3　保守性

(1) 過去のソフトウェア開発では，短納期の中で，とりあえず機能するものの開発をということに重きがあったため，ドキュメントの不備，テスト期間の不足，あるいはプログラム構造化がなされてないことから，開発されたソフトウェアは読みにくく，理解しにくいことが多かった。そのために手直

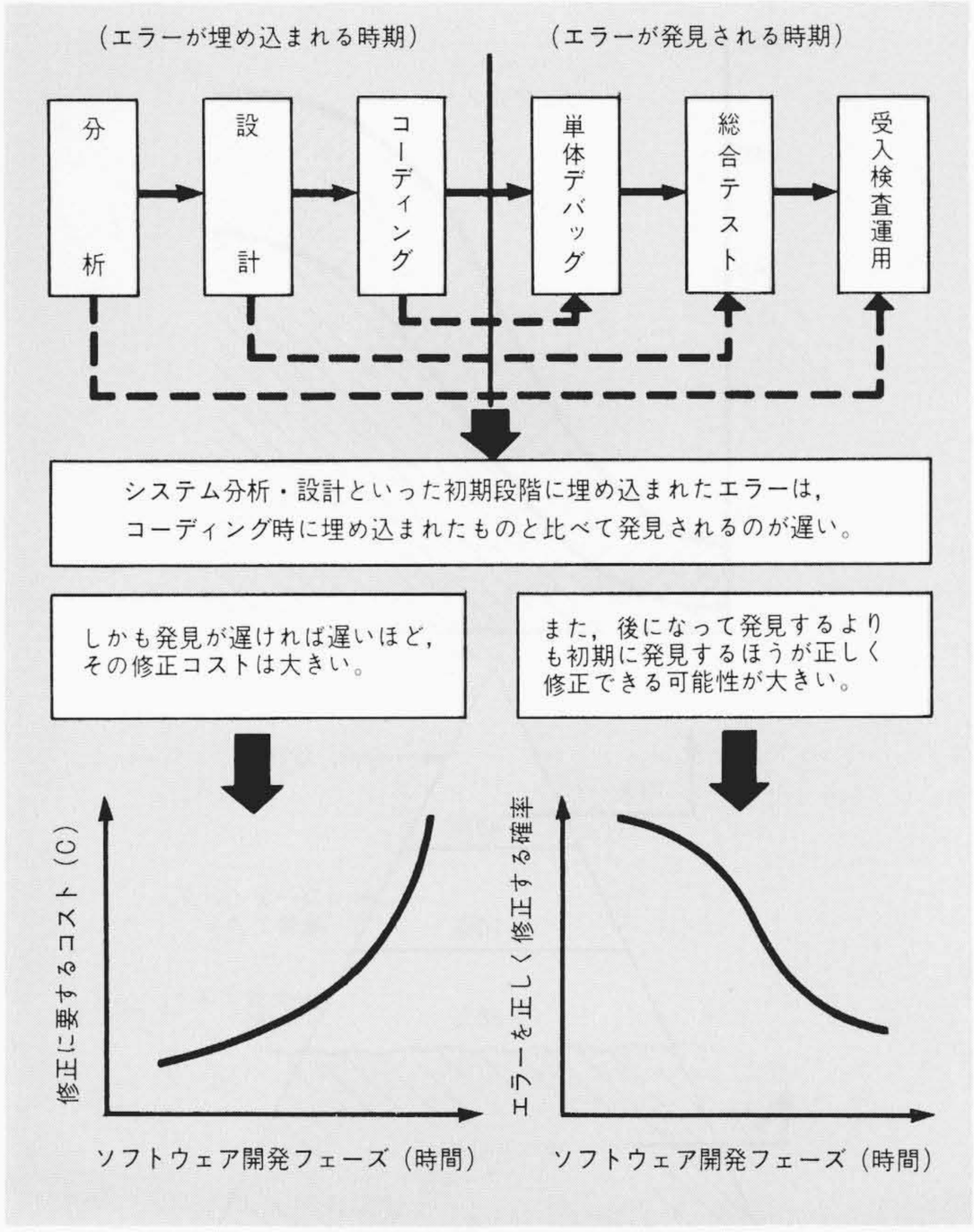

図5　ソフトウェアのエラーが入る時期と発見される時期の関係及びコストへの影響　システム分析・設計段階で多くのソフトウェアが入り込み，それを発見するには大きなコストがかかる。そのために，初期の段階でエラーが入ることを防止することが大切であることを示している。

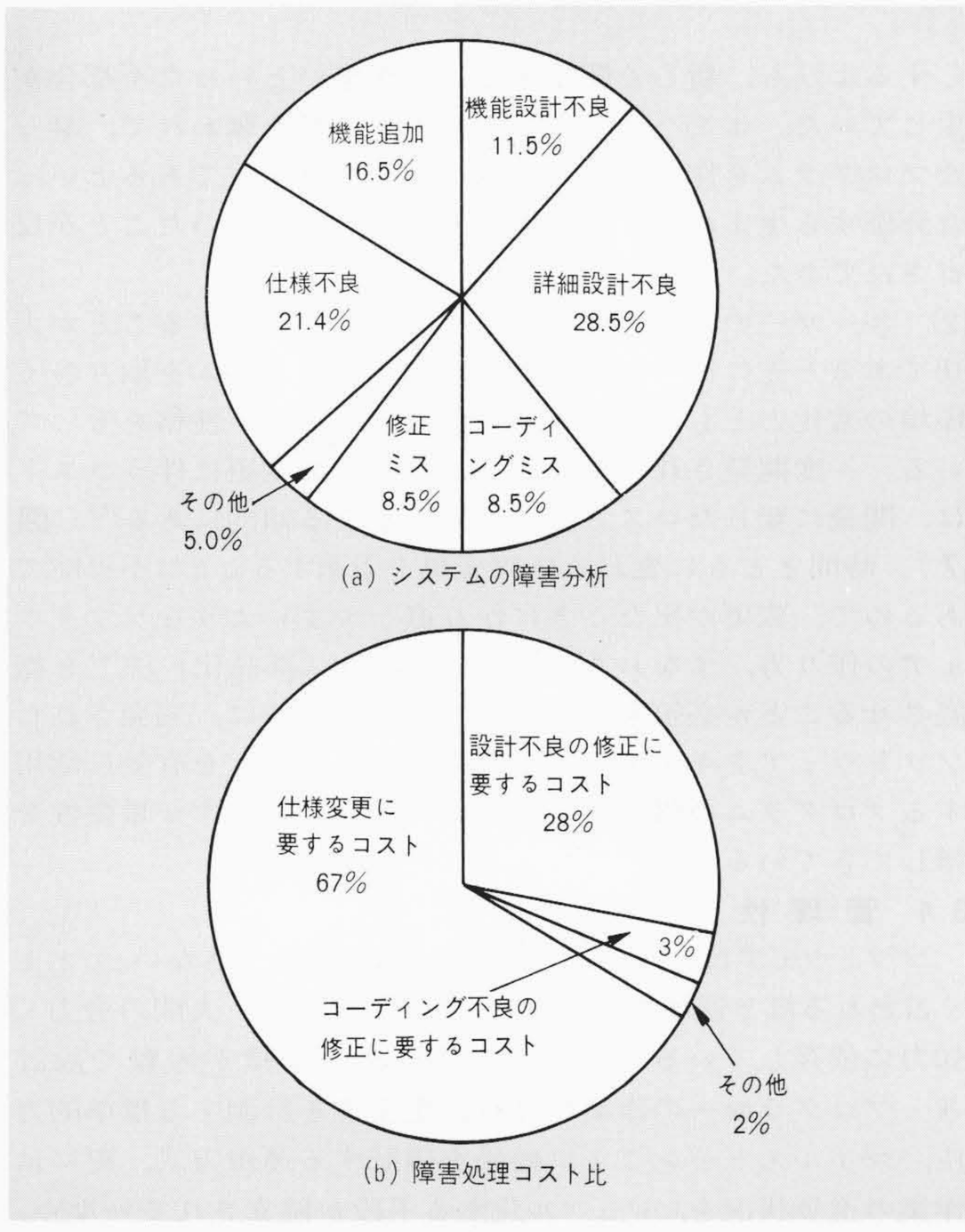

図6　システム障害原因と障害処理コストの関係　システム分析・設計段階での仕様検討の不備が大きなコストを占めていることを示している。

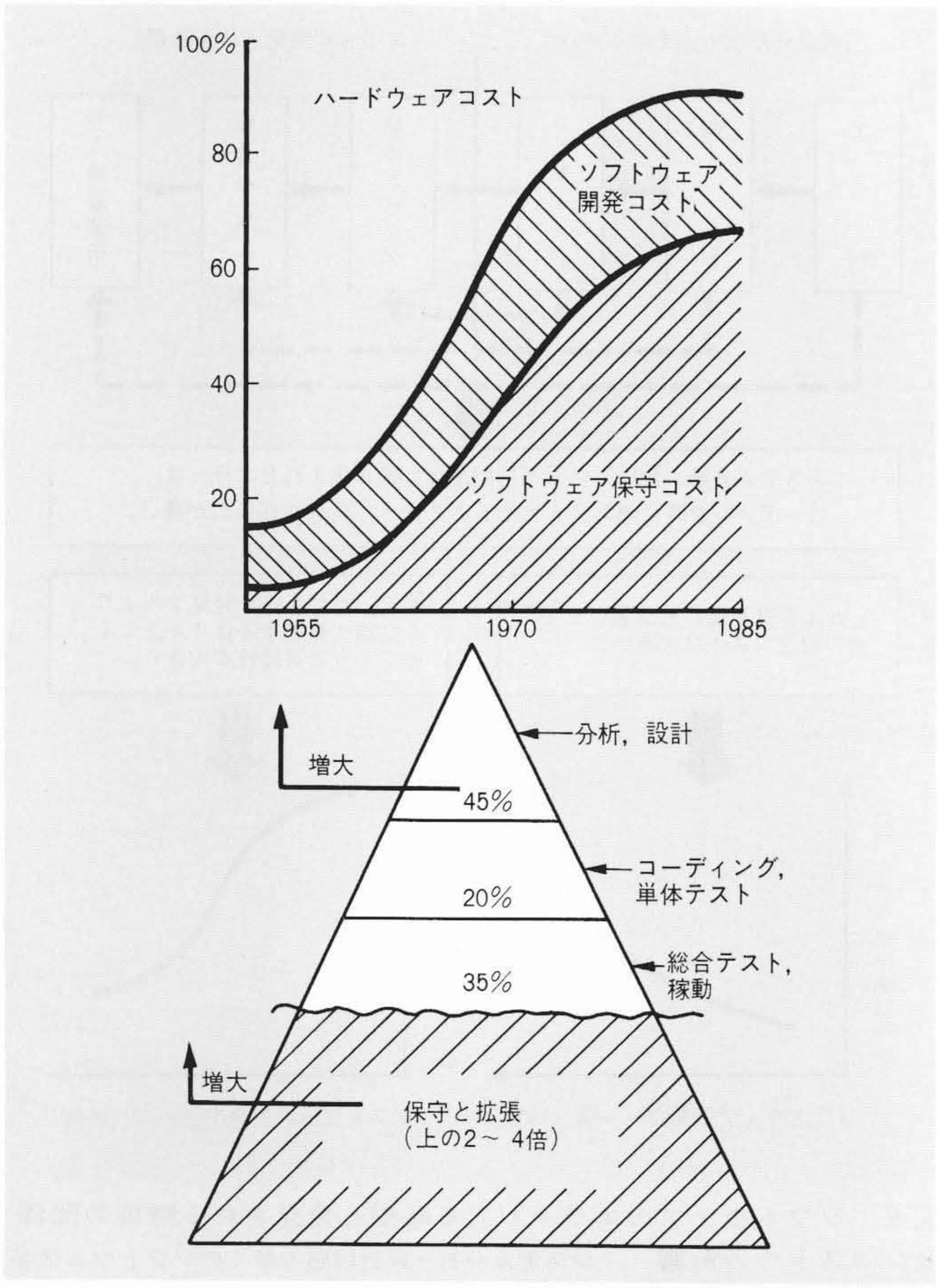

図7 ソフトウェア保守コストの重要性 有名なBoehmの図を示すもので，ソフトウェア保守コストが大きな割合を占めるようになってくるので，それへの対応が今後大切であることを指摘している。

しするよりも，新しく開発するほうが早いといった不都合が生じていた。またハードウェアコストに目を奪われて，凝ったプログラムを作ることが優秀なプログラマーであるといった評価すら生まれて，保守性を悪いものにしていたことが反省されてきた。

(2) ユーザーの要求仕様を正しく分析し，記述することが大切であると先に述べたが，要求仕様は，システムを取り巻く環境の変化とともに変わるという避けられない性格をもっている。一度開発されたソフトウェアの仕様変更に伴うコストは，開発に要したコストよりも大きくなる傾向にある[13]（**図7**）。時間とともに変わる仕様変更を予測することは不可能であるので，変更が出たときに作り直しやすいようなソフトウェアの作り方，すなわちモジュール設計（部品化）法[14]を徹底させることが必須条件となってきたと同時に，開発されたソフトウェアドキュメント，既存のプログラムを有効に活用するプログラムの維持管理支援システムの開発[15]が重要性を増してきている。

3.4 管理性

ソフトウェアは「目に見えず，触れることもできない。」とよく言われるほど管理性に難しい面をもっている。人間の労力・知力に依存しているだけに，人間資源の見積が困難で設計者・プログラマーの作業の品質，生産性を計測する標準的方法，マイルストーンごとに結果を確認する運用方式，更には作業の進歩状況をビジュアル化する手段が確立されていない。頼るとすればドキュメントであるが，このドキュメントが現実には未整備のまま開発が進められることが多く，たとえ整備されていても管理者にとって読みやすいものではない。また，ソフトウェアの規模が大きくなるにつれて，当然ドキュメントの量も増え，到底読むことは不可能になってくる。管理のために必要な情報は何か，それに応じたドキュメント法，及び管理技法の開発と同時に，ハードウェア製品の進んだ生産管理方式と同様に，ソフトウェア製品を対象とした生産管理方式の開発が重要になってきている。

4 主要ソフトウェア生産技術の概要

先に述べたソフトウェア生産の諸問題を背景に，ソフトウェア生産技術の動向を図式化すると**図8，9**に示すとおりである。以下主要な技術について説明する。

4.1 一貫性をもったソフトウェア生産標準手順の確立

ソフトウェアライフサイクルの考えに立って，ソフトウェア開発全般を過去の経験及び知識をもとに見直し，開発のための諸活動の論理的体系化による標準手順の確立と標準ドキュメントの整備がまず必要である。すなわち，フェーズごとに開発作業の標準化を行ない，それぞれのフェーズでのマイルストーンを明確化してソフトウェアを効率的に開発する技法・ツール，及び運用管理方式を位置づけ，ソフトウェア開発作業の全体を把握することからまず始めるべきで，過去このようなアプローチは必ずしも十分ではなかったと言える。本特集の別論文で発表の「アプリケーションシステムの効率的設計技法“HIPACE”」は，これに当たるものである。

4.2 構造化技法の開発

プログラミングフェーズでの構造化プログラミングはよく知られているが，上流フェーズであるシステム分析，システム設計，ソフトウェア設計フェーズでも同様に，構造化技法を導入することが大きなポイントになってきている。ニーズ分析での問題の構造化による目的，問題点の明確化技法，システム設計での要求仕様の構造化記述技法，ソフトウェア設計でのソフトウェアのモジュール構造化設計技法などである。

構造化技法については，幾つかの方法論が提案されている

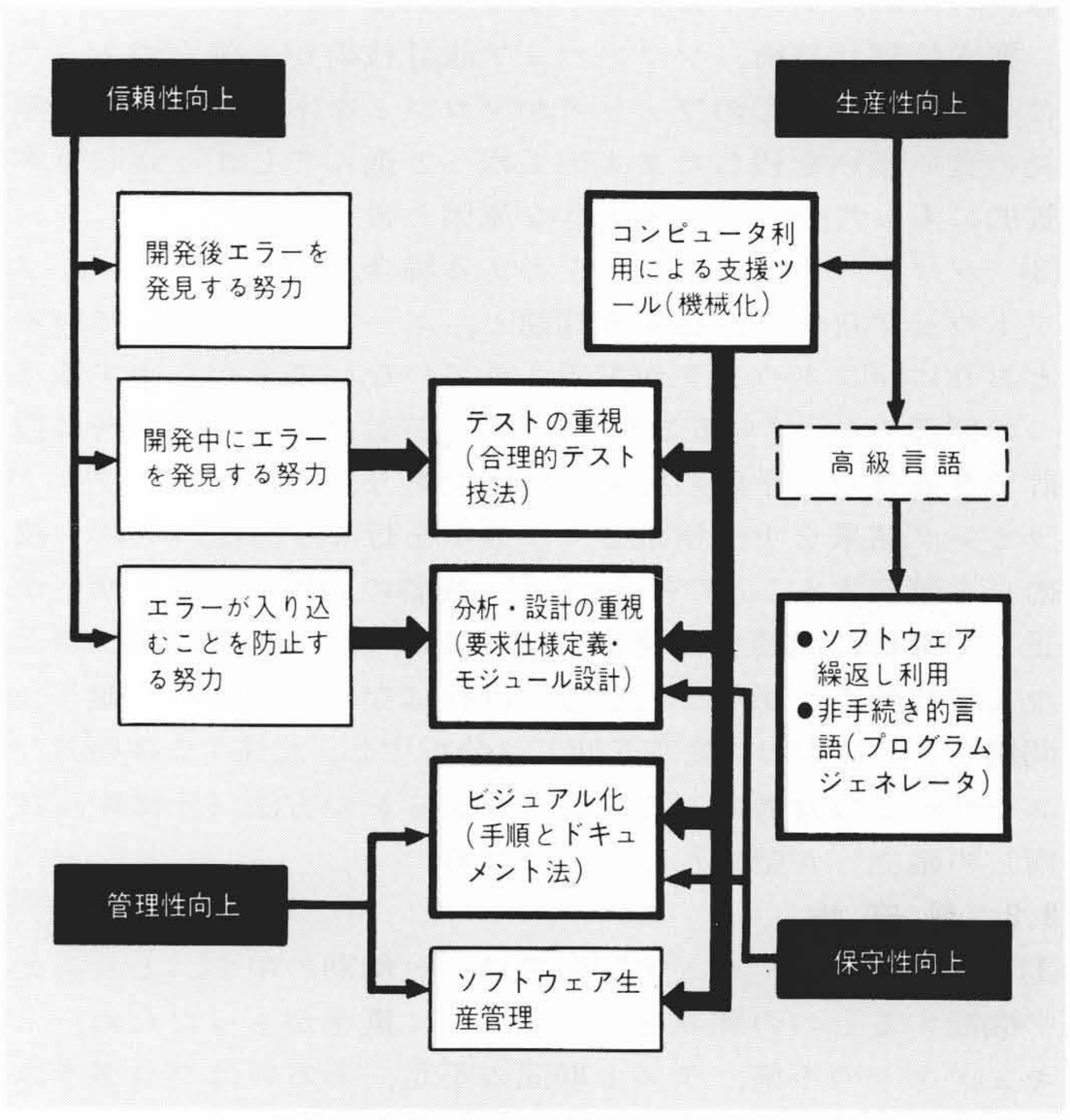

図8 ソフトウェア生産技術の重点開発分野 今後，重点的に研究開発が行なわれると考えられる分野を図式化したものである。

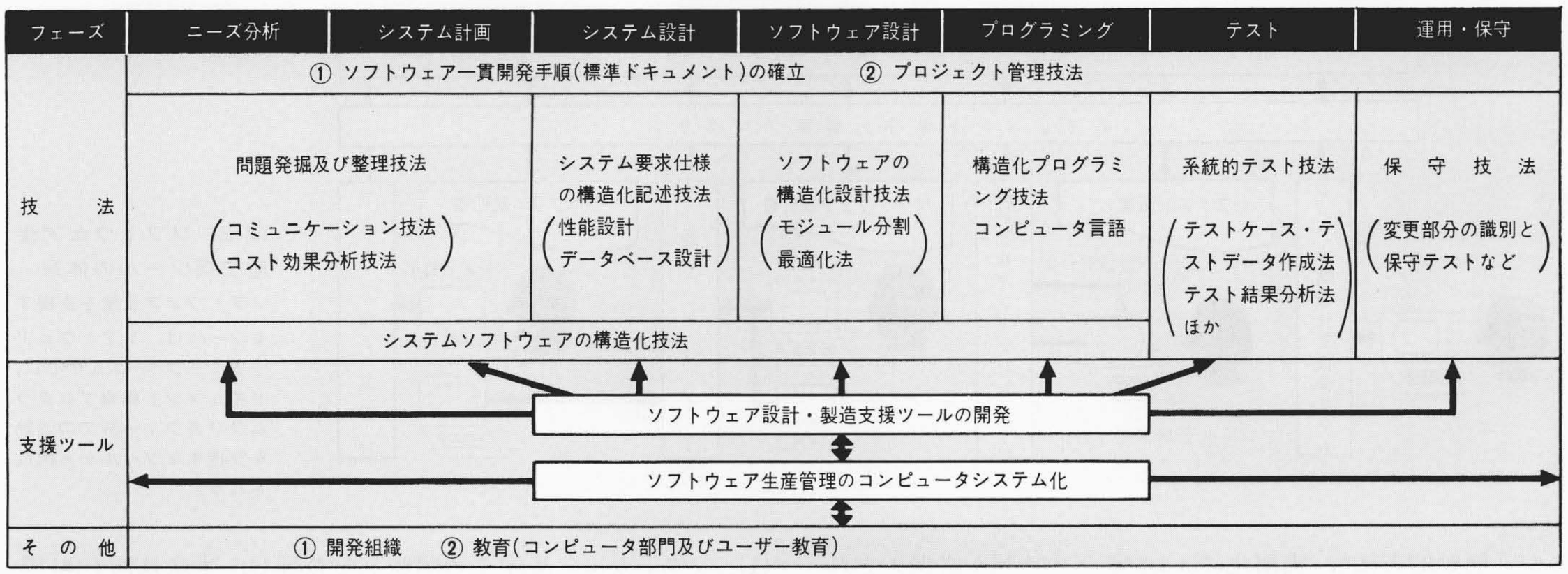

図9　ソフトウェア生産技術の体系概念図　　ソフトウェア生産を取り巻く技術は多面性をもっていて，体系的な整備，開発が必要である。

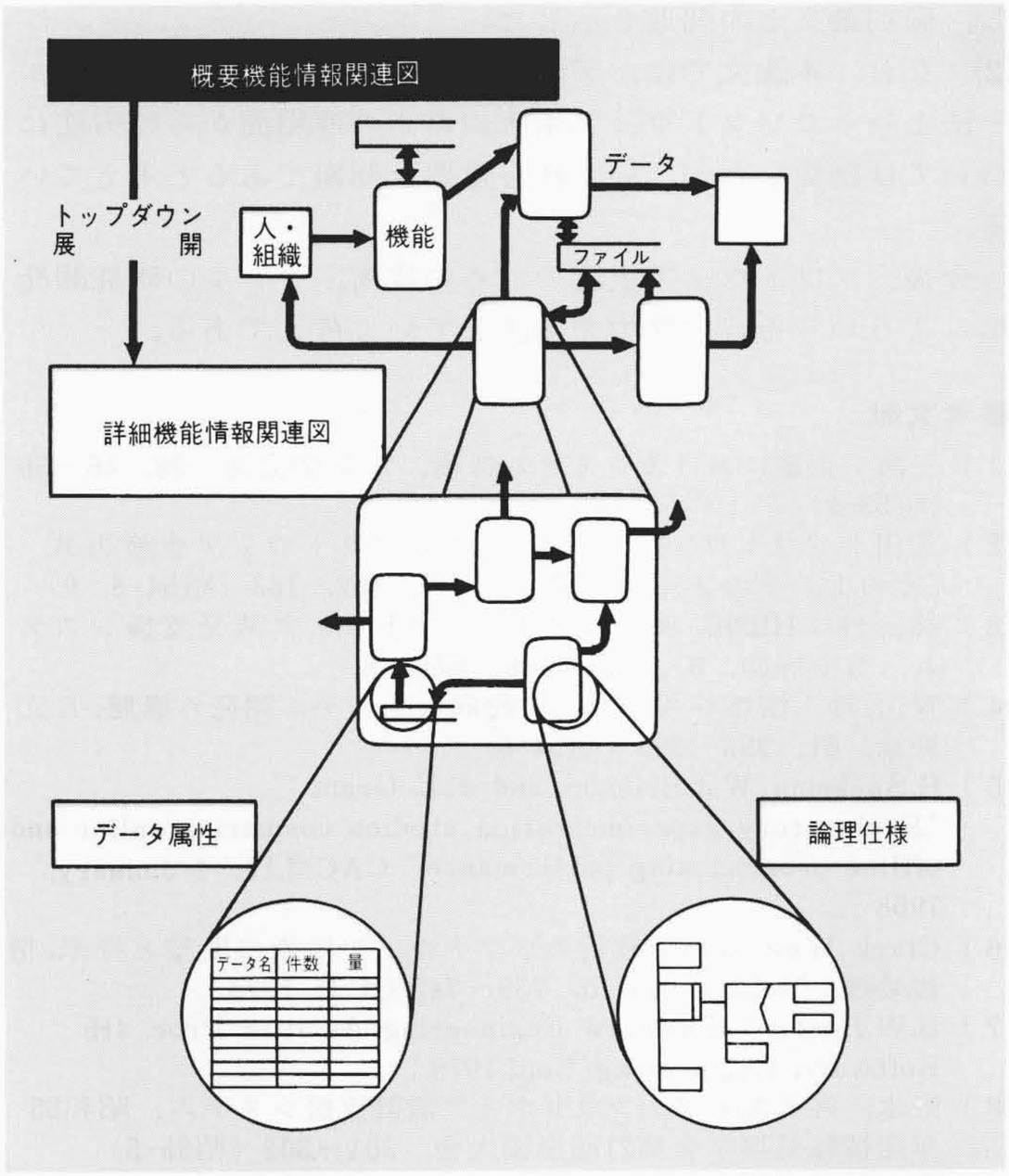

図10　構造化分析・設計法の概念図　　この方法は，ストラクチャードアナリシス及びデザイン法と称され，よく知られているものである。トップダウンによる仕様の詳細化を行ない，その結果は機能情報関連図，データ属性，論理仕様の3種のドキュメントにまとめられシステム仕様数が作られる。

が，その中心は，トップダウンアプローチ，図形化言語の採用によるビジュアル性，コミュニケーション性，エラー混入防止性などをねらっている点に大きな特長があると言える。**図10**に，システム設計での構造化技法の概念を示しておいた[16]。本特集でニーズ分析フェーズを対象に「システム計画のためのシステム要求分析手法“PPDS”の開発」，ソフトウェア設計フェーズを対象として「ソフトウェア構造設計技法」，また適用体験について「構造化設計技法の事務管理部門における適用」がそれぞれ掲載紹介されている。

4.3　プログラミング技法の開発

ソフトウェア生産のためには，簡単かつ理解が容易で，変更保守を容易にするためのプログラミング技法の開発が必要である。プログラミング言語，プログラミング言語を考慮したプログラミングスタイル，あるいはプログラム記述の統一性をもたせるためのコーディング規約の確立などがその主な内容である。本特集に掲載紹介された「業務アプリケーション向け構造化プログラミングパッケージ“UDDT”」，「制御用ソフトウェア一貫プログラミングシステム」，「電子交換機用仕様記述言語“SDL”」，「プログラムの木構造化図面“PAD”」及び「プログラミング言語の最近の動向」は，いずれもこの範疇に入るものである。

4.4　系統的テスト技法の開発

ソフトウェアの正しさを確認する方法を体系づけたのが**図11**である[17]。しかし，ソフトウェアの生産が人間の手によって行なわれる割合が大きい限り，ソフトウェアのエラーを完全に追い出すことは先に述べた各種構造化技法，プログラミング技法をもってしても不可能な状況にある。そこで，エラーの存在をなるべく多く明示し，少しでも完全なソフトウェアへ近づけるにはどうしたらよいかのアプローチ（受動的方法）が，このテスト技術の課題となってきている。

プログラムを実行させずにエラーの検出を行なおうとする静的テスト法，実際にプログラムを動かしてエラーの検出を行なおうとする動的テスト法，更には，テスト実施に当たってのテスト計画法とその進捗状況の管理法などが重要なテーマとなってきている。

一方，プログラムが正しいということを論理的に証明しようとするプログラムの正当性証明の努力も行なわれてきてい

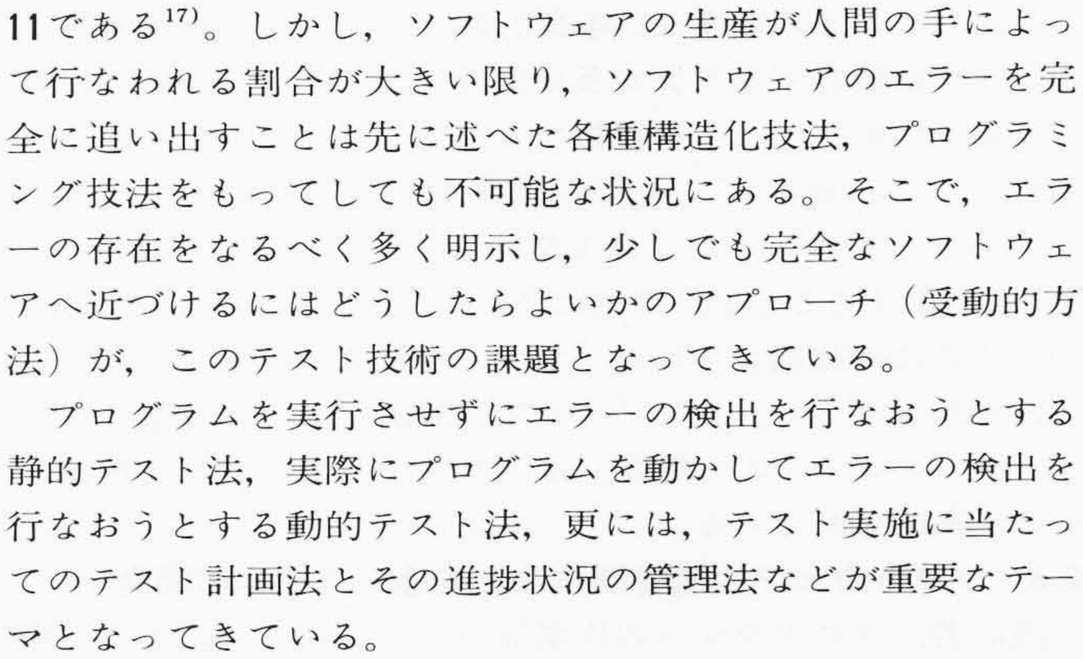

図11　ソフトウェアの正しさを確認する方法の体系　　ソフトウェアが正しくできていることを確認する方法を，体系化して示したものである。

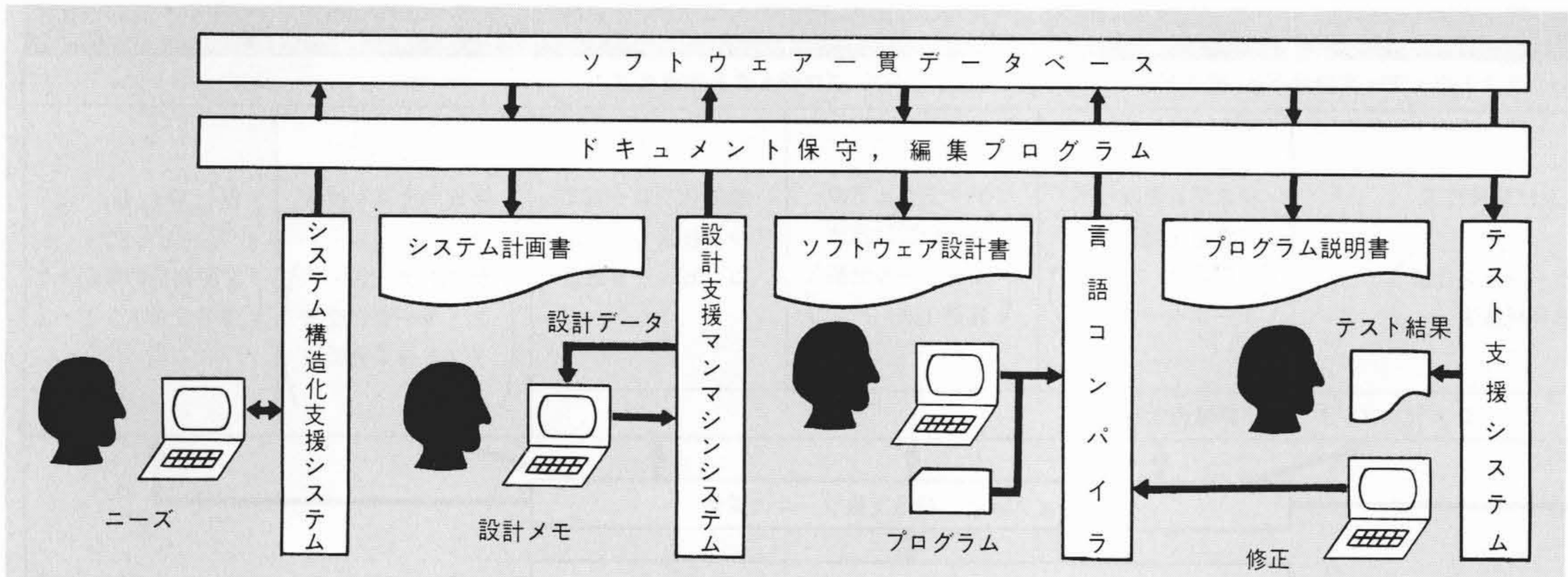

図12 ソフトウェア生産支援ツールの体系
ソフトウェア生産を支援するツールは，ソフトウェア一貫データベースを中心に，ドキュメント編集プログラム及び各フェーズでの活動を支援するツールから構成される。

るが（能動的方法），実用化にはまだ若干の時間が必要とされる。本特集に掲載紹介されている「制御用ソフトウェア機能一貫テストシステム“HITEST/F”」がこれである。また，「ソフトウェア開発支援システム(CASDシステム)」の中で各種のテスト技術について触れている。

4.5 ソフトウェア生産支援ツールの開発

ソフトウェア生産への人間の依存度をできる限り少なくすること，更にはソフトウェア技術者の知的な仕事を支援するためのコンピュータを利用した支援ツールの開発が，生産性の向上，信頼性の向上に有効である。構造化分析・設計を支援するツール，テストを支援するツール，プログラミングを支援するツール及び文書化支援ツールといったソフトウェアライフサイクル全体を通しての機械化システムの開発が進んでいくことが予想できる（**図12**）。

ソフトウェアの生産が，本質的にソフトウェア技術者の知的な仕事に依存する割合が多いだけに，人手を全面的に削除した完全自動化ツールを目指すものではなく人が介入することを認めた，言わば「人間の思考過程を助ける」すなわち「支援するツール」という軽工業的ツール[18]の性格をもつことをよく認識することが，開発に当たって重要であると考える。

本特集に掲載紹介された「ソフトウェア開発支援システム(CASDシステム)」及び「クロスCORALによる分散アプリケーションシステムの開発」は，ソフトウェア生産支援ツールの代表的な例である。

4.6 ソフトウェア生産管理技法と支援システムの開発

設計者，プログラマーの作業品質や生産性を計測する技術及びプロジェクト見積技法・作業進捗状況をビジュアル化する技術と同時に，プロジェクトデータ収集によるソフトウェア生産管理のコンピュータ化が望まれる。本特集に掲載紹介された，「総合ソフトウェア生産管理システム“CAPS”」がこれである。

5 結　言

ソフトウェアの生産技術には，客観的評価が十分でないものがまだ多く，今後の研究に待つところが大であるが，本特集で紹介したような技法を実地に適用，改善していく努力も大切である。一方，今後ますます高度化するシステム，大規模化するソフトウェアに対応するためには，これからの価値観の変化と思考の質的な高まりを洞察できる視野の広い人材の育成が大切であり，またソフトウェアの生産過程をより理解しやすく徹底的にビジュアル化することによって，より多くの技術者がソフトウェアの開発に参画できるように，いっそうの努力を傾注することが任務と考えている。

(1) ソフトウェア生産での問題点を整理し，生産技術を(a)構造化技法・プログラミング技法・テスト技法，(b)支援ツール，(c)生産管理システム，(d)標準手順の確立の四つに分けて説明し，個別論文との関連を示した。

(2) なお，本論文では，デザインレビュー法，ウォークスルー法といったソフトウェア生産のための運用面からの方法については割愛したが，いずれも重要な問題であると考えている。

今後，ソフトウェア生産のための技術，ツールの研究開発に，よりいっそうの努力を傾注していく考えである。

参考文献

1）三浦：企業におけるシステム技術，電気学会誌，98，46～50（昭53-9）
2）芝田：ソフトウェア工場におけるソフトウェア生産方式（その1，その2），HITACユーザ，182，183（昭54-8，9）
3）林，外：HIDIC 80シリーズ　ソフトウェア開発支援システム，日立評論，61，603～606（昭54-8）
4）青山，外：情報システム化の動向とシステム開発の課題，日立評論，61，393～398（昭54-6）
5）H.Sackman, W.J.Erikson and E.E. Grant：“Exploratory experimentation studies comparing online and offline programming performance,” CACM,11, 1 January, 1968
6）Clark Weissman：最近のソフトウェア技術の展望と将来，情報処理，Vol.14, No. 10, 739～745 OCT. 1973
7）B.W.Boehm：Software Engineering-As it is, Proc. 4th Software Engineering Conf.1979
8）野木，外：ADDS・ソフトウェア設計支援システム，昭和55年度情報処理学会第21回全国大会，301～302（昭55-5）
9）Glenford J.Myers：Composite/Structured Design, Litton Educational Publishing（1978）
10）Glenford J.Myers,Software Reliability：Principles and Practices New York：Wiley-Interscience（1976）
11）N.Komoda et al.：Accessibility and Maintenability in Man-Machine Interactive Structural Modeling, Conference on Cybernetics and Systems, 1978
12）野木：要求定義技術の動向，情報処理，20，487～494（昭54～6）
13）B.W Boehm：Software Engineering ; IEEE Transactions on Computer,Vol.C-25, No.12（1976-12）
14）青山，外：アプリケーション・プログラム標準化法とソフトウェア工学，昭和54年電気四学会連合大会
15）小林，外：ソフトウェア部品のテーブルによる統合化方式，昭和54年電気学会全国大会1776～1777（昭54-3）
16）Chris Gane and Trish Sarson：Structured System Analysis：tools & techniques, Prentice-Hall,Inc,1979.
17）日本電子工業振興協会（ソフトウェアエンジニアリング専門委員会報告書）：ソフトウェアエンジニアリングに関する調査（要求定義技術～品質評価技術）
18）木村：Software Toolとは何か？情報処理，Vol.20 No.8，673～680 Aug.1979