U.D.C. 681.32.06.002:658.5.011.5

ソフトウェア生産技術の最近の動向

Current Topics on Software Engineering

小林正和*　*Masakazu Kobayashi*
青山義彦*　*Yoshihiko Aoyama*

ソフトウェアの生産性，品質向上を目的としたソフトウェア生産技術の最新の動向と，本特集掲載の諸論文に紹介されている技術を中心に，日立制作所で実用化されている技法及び支援システムの位置づけを行なった。本論文で紹介した技法及び支援システムは，(1) ソフトウェア構造化技法及び支援システム，(2) ソフトウェア品質レビュー支援システム，(3) ソフトウェア知識化とソフトウェア再利用システム，(4) ソフトウェア開発統合環境としてのソフトウェア エンジニアリング ワークベンチである。本論文で特に強調した点は，ソフトウェアを良構造化することが，究極としてはソフトウェアの生産性向上につながる点と，既存の良構造のソフトウェアをできるかぎり再利用活用することが，生産性及び品質向上にとって重要である点である。

1 緒　　言

本論文では，ソフトウェアの生産性，品質向上を目的としたソフトウェア生産技術の最新の動向について紹介し，合わせて本特集に掲載されている諸技法・ツールの位置づけを行なう。

表1に，J.Martin[1]による計算機システムの利用形態の分類を示す。最近，エンド ユーザーが，計算機システムの利用ばかりでなく，利用する計算機システムのソフトウェアの作成もみずから行なうことが多くなってきている。そのためのエンド ユーザー向けの簡易言語も数多く実用化されている。しかし，エンド ユーザー自身が本来の業務をもつかぎり，ソフトウェア作成にはおのずから限界がある。専門の開発者が作成したソフトウェアを，エンド ユーザーが利用する形態(同表のタイプ1，2)での利用は，相対的な比率が低くなるとしても，量的には増大するものと考えられる。

専門の開発者が開発するソフトウェアでは，高品質化がますます重要になってきている。ソフトウェアの品質としては，過去，信頼性(誤りがないこと)が特に重視され，高信頼化技術[2]が開発されてきた。ソフトウェアの高信頼化が重要であることは，今後とも変わらない。しかし，ソフトウェアの品質は，**図1**に示すように多面的にとらえることが必要になってきている。ソフトウェアは使用性，保守性，再利用性の3種の尺度から評価されなければならない。

使用性は，ソフトウェアがエンド ユーザーに受け入れられるかどうかの尺度である。保守性は，ソフトウェアの改良・

表1　エンドユーザーのタイプ　J. Martin[1] によれば，エンド ユーザーのコンピュータ利用法は，四つのタイプに分けられるという。1，2のタイプに比べて3，4のタイプの比重は増しているが全体の総量が増えているので，表中1，2のタイプのユーザーも年々増大している。上記の表は，文献[1] のp.103をもとに作成したものである。

ソフトウェアの形態	ソフトウェアの開発者	エンドユーザーの役割
オフライン	開発専門家 (開発者と利用者区別)	タイプ1 エンドユーザーは，計算機から出力されたリストを使用。
オンライン		タイプ2 開発者が作成したオンラインシステム(例えば，みどりの窓口)を対話利用。
	エンドユーザー (開発者と利用者区別なし)	タイプ3 問題向き言語(アプリムーションゼネレータ)を用いてオンラインシステムを作成し，作成したシステムを使用。
		タイプ4 簡易汎用言語を用いてオンラインシステムを作成し，作成したシステムを使用。

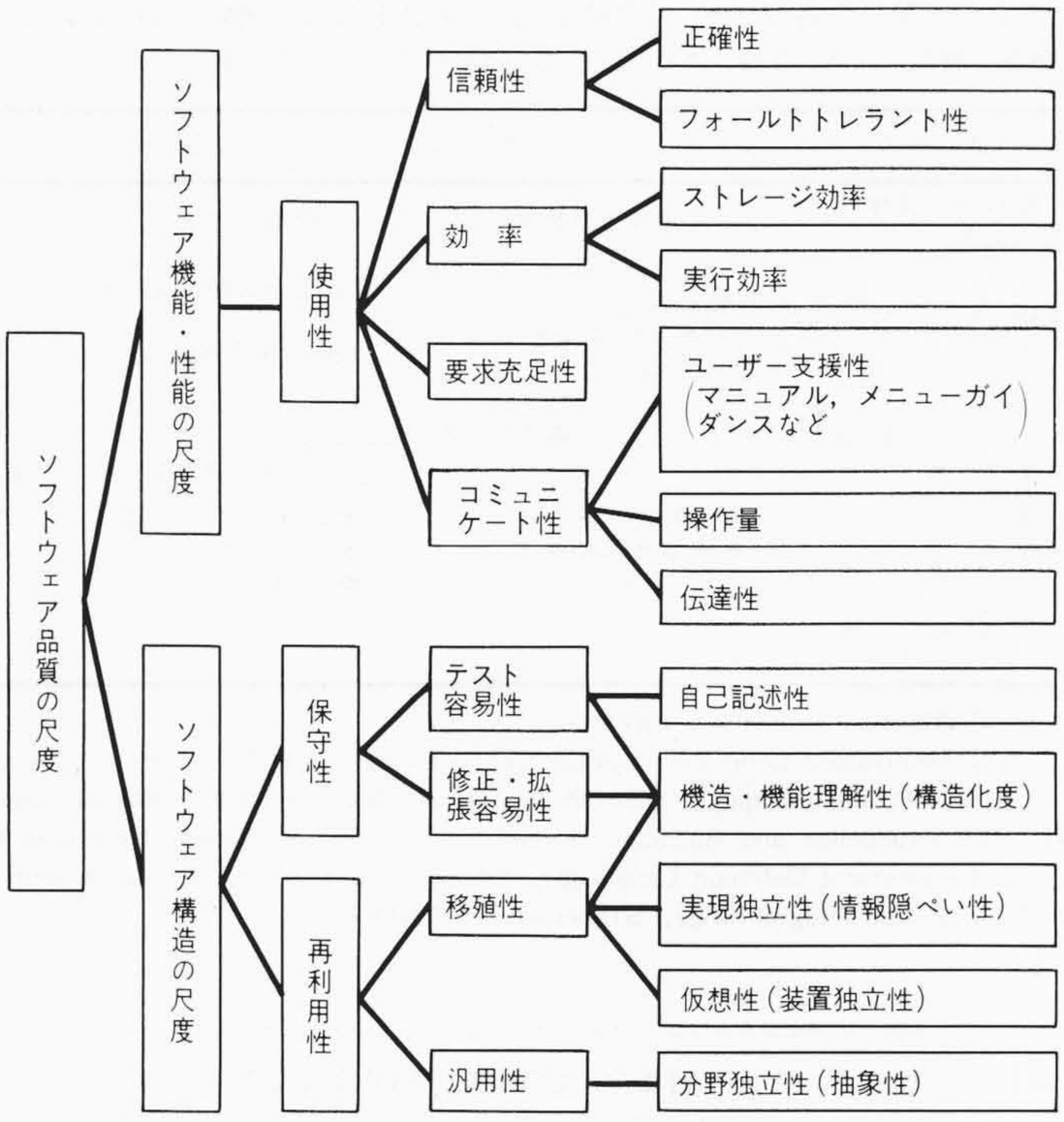

図1　ソフトウェア品質尺度　ソフトウェアの品質は，多面的にとらえる必要がある。上記の尺度はB. W. Boehm及びJ. A. McCallなどの尺度を参考にして作成したものである〔文献3)〕。

* 日立製作所システム開発研究所

拡張が容易かどうかの尺度である。再利用性は，一度作成されたソフトウェアが，他のソフトウェアの開発に当たって転用活用できるかどうかの尺度である。以上3種の尺度のうち，使用性は，ソフトウェアの機能及び性能についての尺度である。保守性，再利用性は，ソフトウェアの作り(構造)についての尺度である。ソフトウェアの生産では使用性はむろんのこと，保守性，再利用性がますます重視されてきている。

2章以下に，使用性，保守性及び再利用性の高いソフトウェアを，効率的に作成する技術について述べる。

2 ソフトウェア構造化技法及び支援システム

ソフトウェアの保守性，再利用性を高めるためには，ソフトウェアを優れた構造にする必要がある。ソフトウェアを優れた構造にするための技法として，次のような方法が有効である。

(1) ソフトウェアを，単機能で，しかも互いに独立なコンポーネント(モジュール)に分け，ソフトウェア全体を，モジュールの組織的な組み合わせによる形態とする(モジュール化と階層化)。

(2) 最も一般的な形で作りたいソフトウェアの機能を表現し，その各構成部分を定められた実現手段に合わせて詳細な動作に至るまで記述してゆき，最後にプログラム※1)を完成させる(情報隠ぺいと段階的詳細化)。

(3) プログラムの実行手順を定める文(制御構造と呼ぶ。)を連接文，反復文，選択文の3種の基本文だけにし，プログラムの流れを複雑にするGOTO文の使用を制限する(構造化定理の採用とGOTOレス化)。

以上の技法は，ソフトウェア構造化技法[4]と呼ばれ，ソフトウェアの高品質化への有効性が既に認められており，実務的にも広く使用されている。

日立製作所のソフトウェア一貫生産システムICAS※2) (Integrated Computer Aided Software engineering)[5]は，**表2**に示すように，PAD(Problem Analysis Diagram)図[6)~10)]などのソフトウェア構造化技法及び支援システムを備えている。プログラム自動生成システムSDL/PAD(Software Design Language/Problem Analysis Diagram)[7),11)]及び，構造化エディタPARSE(Production and Reduction Screen Editor)[12)]などの支援システムには，ソフトウェアを構造化するロジック(知識)が内蔵されている。SDL/PAD，PARSEなどの利用者は，支援システムからのガイダンスにこたえてゆけば，自然に構造化されたソフトウェアが作成できるようになっている。しかし，ソフトウェア構造化技法・支援システムを有効に活用できるようになるためには，ある程度の教育が必要である。日立製作所の教育については，文献[9]に紹介がある。

3 ソフトウェア品質のレビューの重要性と，レビュー支援システム

高品質のソフトウェアを作成するためには，開発の各フェーズで，種々の視点から，産物をレビュー[4]する必要がある(**図2，3**)。

システム設計のあとに行なわれる外部仕様(機能の仕様を利用面からまとめたもの)のレビュー(要求レビュー)では，主として使用性をレビューする〔**図2**(a)〕。

ソフトウェア設計(モジュール構成設計とモジュール内部設計)では，設計書の記述について保守性，再利用性の立場からレビューする〔**図2**(b)〕。アルゴリズムのレビューでは，場合

表2 ICASの各フェーズの核となるソフトウェア構造化技法及び支援システム ICASでは，ソフトウェア開発の全フェーズを通して一貫した方法論，言語，図法，支援システムから成る総合的な技法を提供することにより，高品質なソフトウェアを効率的に開発できる環境を提供する。

フェーズ		システム分析・計画	システム設計	ソフトウェア設計	ソフトウェア製造	テスト
構造化，抽象化の対象		システム化の目的	機能，情報	処理，制御，データ	プログラム	テストケース
構造化，抽象化の方針		1.目的の階層化 2.目的と実現手段との明確な対応づけ	1.問題とその解法との分離 2.機能，情報の段階的詳細化	1.問題を小問題（モジュール）に系統的に分割 2.外側から内側へと解法を構成	1.マクロな論理からミクロな論理へ段階的に詳細化 2.制御の流れの規格化	1.機能から系統的にテストケースを抽出 2.必要最小限のテストケースの抽出
ICASで提供する技法	方法論	構造化分析技法(SA)		構造化設計技法(SD)	構造化プログラミング(SP)	構造化テスト技法(ST)
	言語	—	要求定義言語(RDL)	モジュール設計言語(MDL)	ソフトウェア設計言語(SDL)	—
	図形表現	●目的樹木図 ●機能情報関連図(SDF図)	●ジョブフロー図 ●フローネット図 ●ER図	●ストラクチャードチャート ●IPO図 ●論理データ構造図	●PAD図 ●データ構造図	●機能図式 ●制御フローグラフ
	支援システム	PPDS	RDA	MDA	SDL/PAD PARSE	AGENT

注：略語説明(アルファベット順)
AGENT(Automated Generation System for Test case)，ER(Entity Relationship)，ICAS(Integrated Computer Aided Software engineering)，IPO(Input Process Output)，MDA(Module Design Analyzer)，MDL(Module Design Language)，PAD(Problem Analysis Diagram)，PARSE(Production and Reduction Screen Editor)，PPDS(Planning Procedure to Develop System)，RDA(Requirement Definition Analyzer)，RDL(Requirement Definition Language)，SA(Structured Analysis)，SD(Structured Design)，SDF(Structured Data Flow)，SDL(Software Design Language)，SP(Structured Programming)，ST(Structured Test)

※1) プログラムという言葉は，COBOL，FORTRAN，PASCAL，C，BASICなどの高級言語か，あるいはアセンブラ言語で記述された，いわゆるソースプログラムの意味で用いる。これに対してソフトウェアという言葉は，コンピュータシステムの開発時に作成される計画書，仕様書，設計書，テストに関する文書，各種マニュアル及びプログラムを総称するのに用いる。

※2) ソフトウェア一貫生産システムは，開発の対象となるソフトウェアの違いによって特化されるべきものである。日立製作所では，ICASという基盤技術をもとに，対象ソフトウェア別に各種のソフトウェア一貫生産システムが実用化されている[5]。例えば，本特集に掲載されているEAGLEシステム[21)~24)]は，ICASの一貫思想に基づいた事務処理用ソフトウェア一貫生産システムであり，既に製品化されているものである。事務処理ソフトウェア生産システムとしてはEAGLEだけであり，基盤技術であるICASの技法及び支援システムは，実用性・有効性の認められたものから順に，EAGLEに取り入れられていっている。

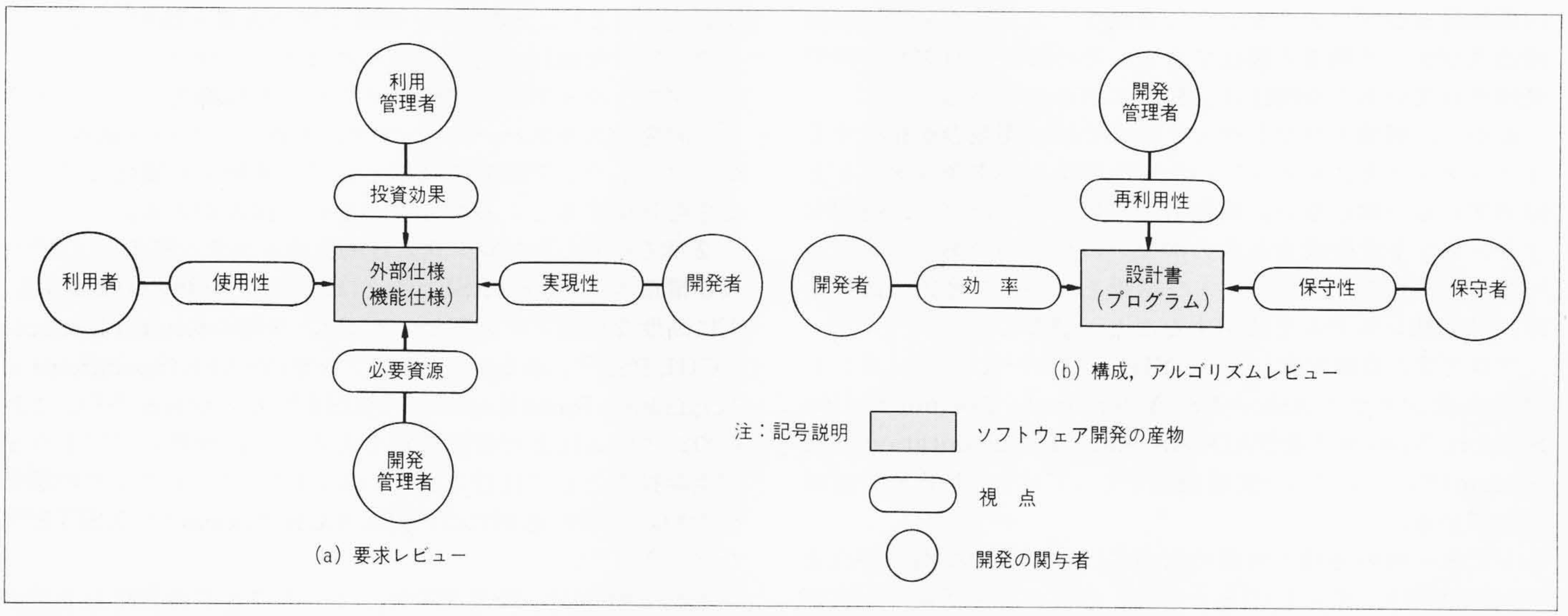

図2　ソフトウェア開発の関与者とレビューの視点　ソフトウェア開発の産物である機能仕様設計書，プログラムなどは，開発の過程で十分に種々の視点からレビューがなされる必要がある[4]。

によってはプログラム テキストをレビューする必要もある〔同図(b)〕。

要求レビューと設計レビュー(モジュール構成レビューとアルゴリズム レビュー)とでは，レビューの視点の重点が異なり，参加者も異なる(**図2**)。要求レビューでは，記述された外部仕様では限界があり，デモシステム，機能・性能のシミュレーションなどを行なって，仕様を確認することが有効である。このような技術は，ラピッド プロトタイプングと総称され，最近注目されている技術の一つである[14]。

現状の技術では，ソフトウェアの設計を完全に自動化することは難しく，人手による作業が必要である。人手による作業が存在するかぎり，レビューが重要になる。現実の開発では，開発者はとにかく動くソフトウェアを作ることに専念する。開発者が作る立場を離れて，自ら作成した産物を，例えば保守性，再利用性などの別の観点からレビューすることは，なかなか難しいことである。やはり他人の目を通してレビューすることが大切である。

レビューのためには，ソフトウェアが理解しやすく表現される必要がある。ソフトウェアは，従来，記述形式で表現されることが多かった。例えば，高級言語よる表現では，モジュールの包含関係(ブロック構造)，ステートメントの実行順序(制御構造)，アルゴリズムがすべて言語で記述される。このように，異質な情報は，表現を分けることが理解性を増す。アルゴリズムのように，論理性が要求される部分は記述形式が優れているが，構造にかかわる部分は，図示することが望ましい[15),16)]。本特集で紹介されているPAD図[6]は，モジュール

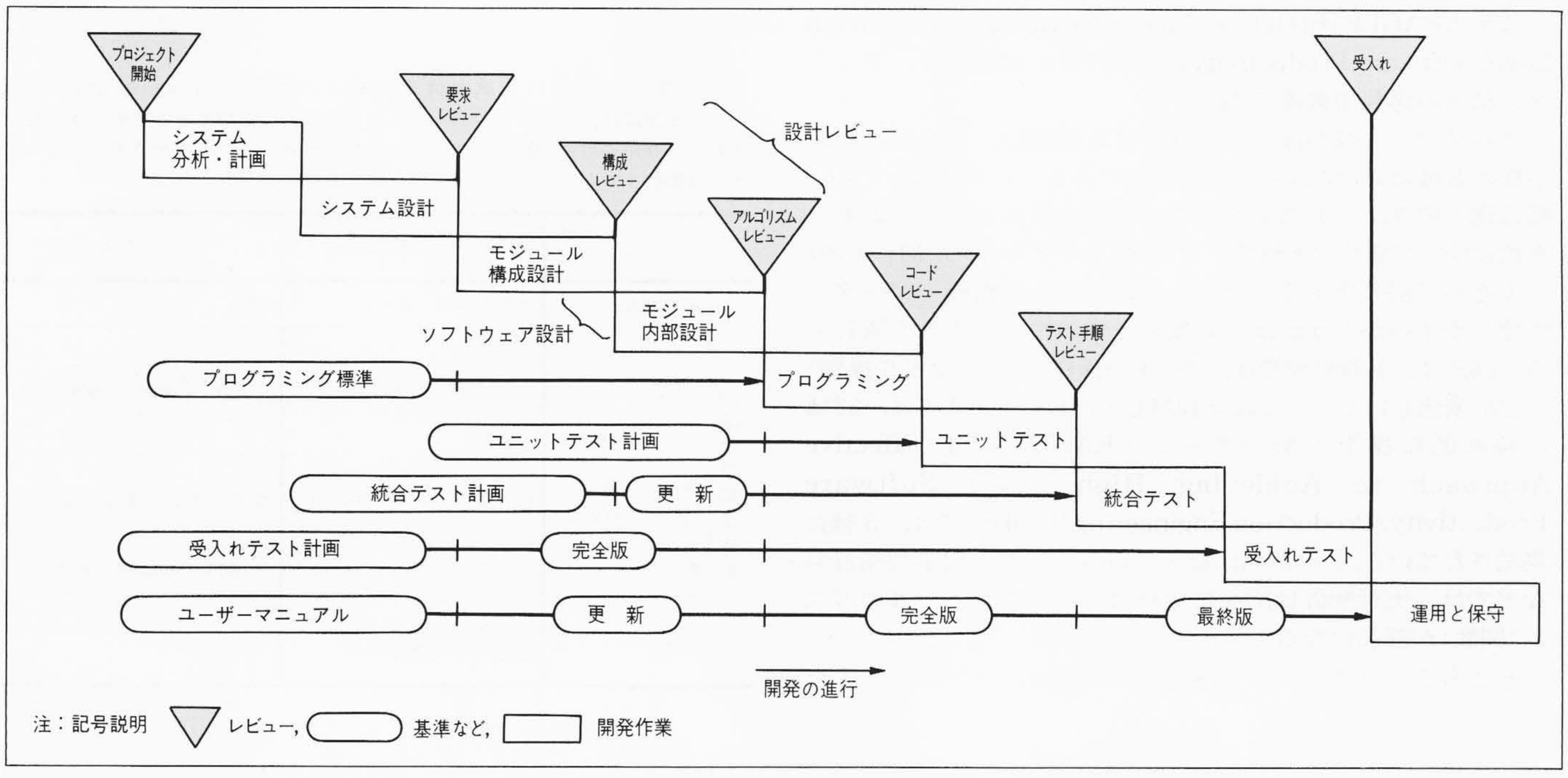

図3　ソフトウェア開発のフェーズとレビュー　開発のフェーズの区切りでは，十分なレビューを励行すべきである。レビューを行なうことによって，むだな作業を省くことができると同時に，ソフトウェアの品質も向上する。なお，上図はB.W.Boehmの教科書[13]を参考にして作成したものである。

内部設計及びプログラミングに有効な，ソフトウェアの制御構造及びデータ構造の図法である。そのほか，日立製作所で使用されている主な図法は，**表2**に示されている[5]。

しかし，現実のソフトウェア開発では，開発者が作成するドキュメントと，レビューのために望ましいドキュメントとは必ずしも一致しない。開発者はレビューのために，特別なドキュメントを作成する余力がないのが普通である。レビューのための情報作成，レビュー結果を反映した産物の修正には，計算機システムを使用することが望ましい。

プログラム自動生成システムSDL/PAD[7),11)]，ドキュメント自動生成システムAuto－DS(Automated Documentation Support System)[8)]及びADCAS(Auto Documentation Aid System)[10)]は，レビュー支援機能をもっており，社内外で活用されている。

レビュー時の各種の判断のためには，定量的な指標が役立つ。文献[17)]は，テスト フェーズで，内在バクを予測する実務的な技法を紹介している。今後は，設計時に，ソフトウェアの品質を評価するための定量指標が重要になる。日立製作所でも幾つかの試みはある。社会状況としても，モジュール化の良し悪しの定量的評価技法は，まだ研究段階といってよいであろう。

4 ソフトウェア再利用システムとソフトウェアの知識化

ソフトウェアの品質及び生産性を大幅に向上する手段は，既存ソフトウェアを活用することにより，同一機能のソフトウェアを重複作成しないことであると言われている[20)]。しかし，ソフトウェア開発の現状では，類似機能のソフトウェアを作成する際に，既存ソフトウェアを修正して利用するよりも新しく作り直したほうが，品質も生産性も高まる場合が多い。このような状況を打開するには，**2**章及び**3**章で述べてきたように，ソフトウェアを構造化し，良質のソフトウェア部品を作成する必要がある。ソフトウェアは，仕様書，設計書，プログラム，テストケース及びデータから成るが，プログラムについての部品化は，幾つかの実用システムが世に出ている。本特集に紹介している事務処理用ソフトウェア一貫生産システムEAGLE(Effective Approach to Achieving High Level Software Productivity)[21)~24)]の中心的機能は，プログラム部品の再利用機能である。

プログラムの部品は，アルゴリズム(関数)，データ，制御手順の3種に分けられるのが一般的である。いずれにしても，部品化の要点は，(1) 変化が少ない要素を部品とする，(2) ある変化に伴って変化する機能・データ(ライフスパンが同じもの)をまとめて部品化する，という2点にある。機能・データが変化しやすいかどうかは，対象の分野の性質に大きく依存する。例えば，事務処理では，データの操作手順(ファイル操作)などが変化しにくい。これを利用して，EAGLE[21)]では，22種の標準的な操作手順パターン〔EAGLE/PET (Effective Approach to Achieving High Level Software Productivity/Production Engineering Tool)[24)]では，5種に集約されている。〕が部品になっている。他方，科学技術計算などでは，実行制御は部品になりにくく，むしろアルゴリズム(関数)が部品になる。

ところで，ソフトウェア再利用の究極は，ソフトウェアを知識化することにある[25),26)]。知識工学の成果を活用して，ソフトウェアの知識化を行なう試みには大きく分けて，

(1) ソフトウェア開発過程のノウハウを知識化し，ソフトウェア開発のエキスパート システムを作ろうという試み

(2) ソフトウェア開発の結果としての産物を知識化して，これを再利用するしくみを作ろうという試みがある。

2章で紹介したプログラム自動生成システムSDL/PAD[7),11)]及び構造エディタPARSE[12)]などは，上記の分類の(1)に属する。更に高度な自動プログラミングには，米国のKestrel InstituteのCHI, PSI[※3)]，南カリフォルニア大学のSAFE(Specification Acquisition Form Experts)，GIST[※3)]などがある[26),27)]。これらのシステムはまだ研究段階であるが，次世代のソフトウェア生産技術として注目されている。また，ソフトウェアの解析への知識工学の応用には，米国DEC社のSpear[※3)]，XSITE[※3)]などがある[27)]。

上記分類の(2)に属するのが，ソフトウェア再利用システムである。従来のソフトウェア再利用方式では，(a) 部品化の規範が不明確であるため，部品の追加，除去，置換などの管理が困難であったこと，(b) 部品の機能・性能を利用者が熟知して部品を利用する方式のため，部品が利用できるにもかかわらず利用されないことも多かったこと，(c) 部品の修正が利用者に自由に行なえる方式のため，本来品質の良い部品であっても品質低下を招くこと，などの問題があった。本特集の裏表紙，明日を創るシステムに紹介のあるソフトウェア再利用方式(ICAS-REUSE：Integrated Computer Aided Software engineering-Reusable Software Engineering)では，(a) 情報とか，「もの」とか，具体的に把握できる実体を部品化の規範とした部品の体系化を図ったこと〔対象指向部品体系化方式(**表3**)〕，(b) 知識ベース，ルールベースを用いて，日本文で与えられた記述から部品を自動検索・修正・組合せ機構を設けたこと(**図4**)，などにより従来の問題の解決を図っている。

今後，知識工学を活用したソフトウェア再利用方式などは，広く実用化されると考える。

表3 新しい部品化方式と従来方式との比較 ICAS-REUSEで採用している部品は，情報とか「もの」といった具体的に把握できる基準によっているため，利用の可否判断がつきやすい。部品規模が小であるが検索，組合せなどを自動化するため，利用上に問題は生じない。

項目＼方式		新しい部品化方式	従来方式
部品及び部品体系	部品単位	対象(情報，もの)	機能
	部品例	台帳 帳票 表示装置 …	入力 検索 編集 …
	部品の汎用性(データ型変化)	該当部品内で修正閉じる。	関連全部品へ修正波及
	部品規模	小(20～50ステートメント)	大(300～500ステートメント)
	部品体系	業務依存部品から汎用部品までの階層的体系	なし

※3) CHI，PSIは，ギリシア文字の英語読みであり，Kestrel InstituteのC.Greenらの研究プロジェクトの愛称である。SpearはDECのシステムの愛称である。GIST，XSITEについては何かのAcronymと思われるが，公開文献[27)]には明示されていない。

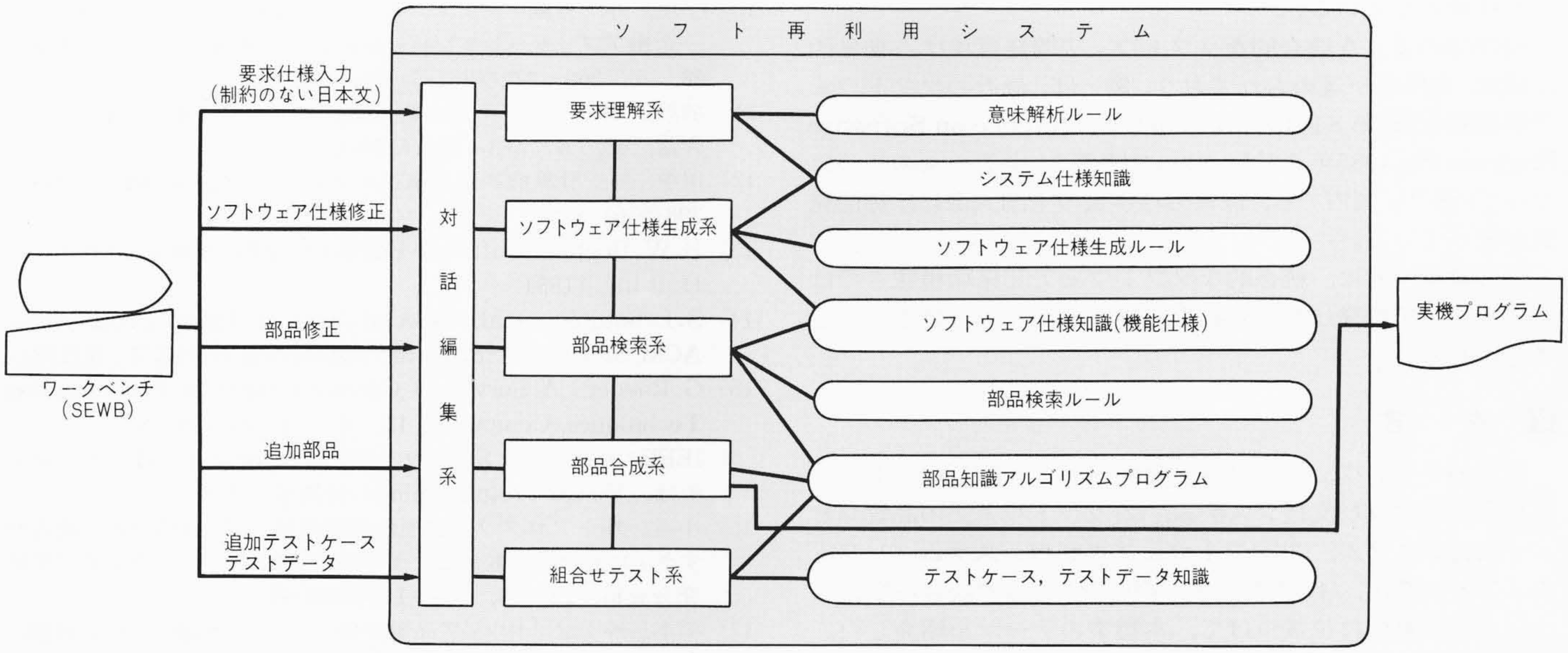

図4　新しいソフトウェア再利用システム(ICAS-REUSE)　従来の再利用システムでは，ソフトウェア部品の仕様を設計者が熟知し，利用するのが普通であった。そのような方式では部品が多くなると利用が困難となるため，少数の複合機能から成る部品を用いることになり，部品の汎用性に制約が出るなどの問題があった。本方式では，要求をシステムに与えて，部品の検索を自動化することによって，問題解決を計った。この方式が可能となったのは，ソフトウェアを知識化したこと，開発プロセスをルール化したことによっている。

5　ソフトウェア開発のための統合環境

以上2～4章で述べてきた諸技法，諸ツールを有効に利用するためには，優れたマンマシン コミュニケーション インタフェースをもつワークステーションが必要である。例えば，日立クリエーティブワークステーション2050で利用できるSEWB-Software Engineering Work Bench[30]はこの目的にかなうものであると考えている(**図5**)。

SEWBは，(1) 開発支援ツールの一貫した統合利用を可能にしていること，(2) 各種ソフトウェア図面を重視したフレンドリーなコミュニケーション インタフェースをもっていること，(3) 開発者の個人の仕事はワークステーションだけで遂行でき，チームについての情報管理(ソフトウェア統合データベースなど)はホストコンピュータで行なう分散処理が行なえることな

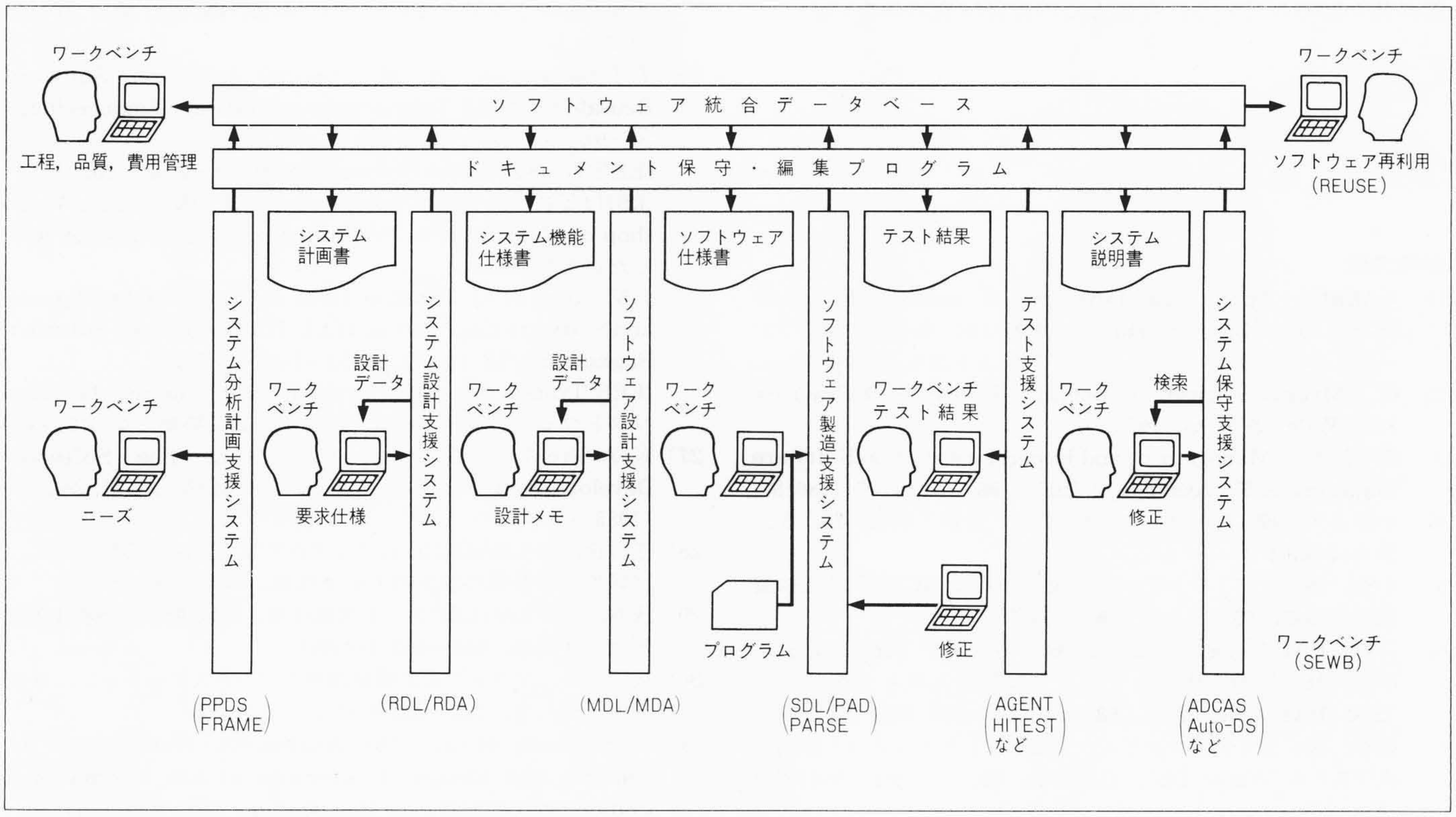

図5　ソフトウェア一貫生産システム(ICAS)　ソフトウェア一貫生産システムが普及するためには，ワークベンチ化が必要である。ここ数年の間にワークベンチによるソフトウェア一貫生産が急速に定着化してゆくと考えられる。

どを特徴としている。

SEWBのような統合的なソフトウェア開発環境は，世界的に研究，実用化が進められており，例えば，最近のソフトウェア学国際会議(第8回International Conference on Software Engineering 1985年8月Londonで開催)の主テーマの一つになっている[31]。国内でも，情報処理学会の全国大会には毎回発表が続いている。

ここ数年の間に，統合的なソフトウェア開発環境抜きではソフトウェア開発は行なわれないようになるのではないかと考えられる。

6 結　言

最近のソフトウェア工学の主要テーマである(1)ソフトウェア構造化技法及び支援システム，(2)ソフトウェアの品質レビュー支援システム，(3)ソフトウェア再利用システムとソフトウェアの知識化，(4)ソフトウェア開発のための統合環境，について最新動向に位置づけて，本特集のテーマを紹介した。

最後に，ソフトウェア生産技術が実用化されるためには，長い年月と関係者の多大な協力が必要である。例えば，本特集で紹介したPAD技法については，提案が1978年であり，社内では1983年までに全社的教育がなされるようになり，ほぼ1984年ごろに社内定着化が図られた。開発ツールについては，更に長い年月と実用化への努力が必要である。EAGLEについては，アイディアの萌芽は，1970年に発表されたHELP (Hitachi Effective Library for Programming)にまでさかのぼる。HELPは実に2,700社あまりの実用実績をもっている。この背景を踏まえて，ソフトウェア一貫生産システムICASのフィロソフィーのもとに，現EAGLEは1983年に製品化が行なわれたものである。日立製作所では，研究所，工場が一体となってソフトウェア生産技術の研究開発を進めており，新しく開発された技法・ツールはまず社内で使用して厳しい評価を行なっている。効果があると十分に自信のもてるものから製品化が図られている。今後もこの方針は変わらない。

参考文献

1) J.Martin：Application Development without Programmers, 1982, Prentice-Hall,(山中義昭訳・監修：プログラマーなしのアプリケーション開発，アシスト出版局，1984)
2) G.J.Myers：Software Reliability—Principles & Pracices, John Wiley & Sons (1976)
3) W.Curtis：Management and Experimentation in Software Engineering, Proceedings of IEEE, **68**, 9, 1147(1980-9)
4) 小林：大規模ソフトウェアの開発技術，計測と制御，**25**, 3, 34～41(昭61-3)
5) 小林，外：ソフトウェア一貫生産システム“ICAS”基盤技盤技術の確立，**66**, 3, 1～6 (昭59-3)
6) 二村：PADの開発，日立評論，**68**, 5, 351～356(昭61-5)
7) 前澤，外：図面を用いたプログラム開発方式と支援ツール“SDL/PAD”，日立評論，**68**, 5, 357～360(昭61-5)
8) 橋本，外：マイクロコンピュータ用ソフトドキュメント自動生成システム“Auto—DS”，日立評論，**68**, 5, 361～364(昭61-5)
9) 飯塚，外：PAD普及教育の一例，日立評論，**68**, 5, 365～368(昭61-5)
10) 平川，外：自動ドキュメンテーション支援システム“ADCAS”—金融アプリケーションパッケージへの適用例—，日立評論，**68**, 5, 369～372(昭61-5)
11) 前沢，外：プログラム自動生成システム“SDL/PAD”，日立評論，**66**, 6, 463～468(昭59-6)
12) 田中，外：計算機誘導形構造エディタ，日立評論，**68**, 5, 393～398(昭61-5)
13) B.W.Boehm：Software Engineering Economics, Prentice-Hall Inc.(1981)
14) S.L.Squires, et al.：Special Issue on Rapid Prototyping, ACM SIGSOFT Software Engineering Notes, **7**, 5 (1982)
15) G.Raeder：A Survey of Current Graphical Programming Techniques, Computer, **18**, 8, 11～24(1985-8)
IEEEのComputer Societyの機関誌Computerの**18**, 8 (1985-8)は，Visual Programmingの特集号である。
16) 小滝，外：プログラム論理仕様記述法PDLとPADの「読みやすさ」における比較評価，情報処理学会，ソフトウェア工学研究会資料，**28**, 19, 109～113(昭58-9)
17) 橋本，外：ソフトウェア品質評価システム“SQE”，日立評論，**68**, 5, 399～402(昭61-5)
18) D.N.Card, et al.：Criteria for Software Modularization, Proceedings of 8th International Conference on Software Engineeing, 372～377(1985)
19) D.Kafura, et al.：A Validation of Software Metrics Using Many Metrics and Two Resources, Proceedings of 8th International Conference on Software Engineering, 378～385(1985)
20) 小林，外：プログラムの自動作成・再利用技術，昭和60年電気・情報関連学会連合大会予稿集，5-35～5-28(1985)
21) 葉木，外：システム開発支援ソフトウェア“EAGLE”，日立評論，**68**, 5, 373～378(昭61-5)
22) 岡本，外：新日本製鐵株式会社における“EAGLE”の適用，日立評論，**68**, 5, 379～382(昭61-5)
23) 大野，外：EAGLEにおけるシステム設計支援システムの開発，日立評論，**68**, 5, 383～386(昭61-5)
24) 嘉藤，外：販売会社向けオフィスコンピュータ用システム開発支援ツール“EAGLE/PET”，日立評論，**68**, 5, 387～392(昭61-5)
25) T.J.Biggerstaff, et al.：Special Issue on Software Reusability, IEEE Transactions on Software Engineering, SE-10, 5, 473～609(1984-9)
IEEE Transaction on Software Engineering SE－10, 5は，米国ITT社主催で1983年9月7日～9日に開催されたWorkshop on Reusability in Programmingの代表的な論文を掲載した特集号である。
26) J.Mostow, et al.：Special Issue on Artificial Intelligence and Software Engineering, IEEE Transactions on Software Engineering SE-11, 11, 1253～1408(1985-11)
IEEE Transaction on Software Engineering SE－11, 11はソフトウェア工学への知識工学応用の論文特集になっている。
27) K.A.Frenkel：Toward Automating The Software Development Cycle, Communications of the ACM, 28, 6 (1985-6)
28) 千吉良，外：EAGLEにおけるプログラム部品合成実現方式，情報処理学会第31回全国大会予稿集，459～460(1985秋)
29) 永井，外：EAGLEにおける部品体系，情報処理学会第31回全国大会予稿集，461～462(1985秋)
30) 前沢，外：ソフトウェア開発指向のワークステーション，日立評論，**67**, 3, 235～238(昭60-3)
31) M.Stephens, et al.：The Analyst－A Workstation for Analysis and Design, Proceedings of 8th International Conference on Software Engineering, 364～369(1985)
このProceedingsには他に4件のソフトウェア統合開発環境の論文が掲載されている(pp. 2 ～33)。