U.D.C. 681.327.13.015:681.32.06.002

ソフトウェア開発指向のワークステーション

—ソフトウェア エンジニアリング ワークベンチ—

Workstation for Software Development —Software Engineering Workbench—

前沢裕行* *Hiroyuki Maezawa*
小林正和* *Masakazu Kobayashi*
富岡幹雄** *Mikio Tomioka*
城戸　宏** *Hiroshi Kido*

ソフトウェアの生産性・信頼性の向上のためには，開発の全フェーズにわたる開発技法・支援システムの充実に加えて，優れた開発設備の採用が必要である。ソフトウェア開発設備としては，(1) 計算機の負荷分散による応答性の向上，(2) 大形計算機リソースを共同利用するときの運用上の制約の解消，(3) 開発者の利用の仕方に応じたフレキシブルかつヒューマンフレンドリーな対話インタフェース構築の面から，ワークステーションとホストコンピュータとを結合し，この上でソフトウェア開発支援システムを一貫して利用できる統合開発環境が必要である。

本稿では新しいソフトウェア統合開発環境として，日立製作所が研究開発したSEWBについて紹介する。SEWBを採用することにより，ヒューマン フレンドリーなインタフェースを通して，ソフトウェアの一貫開発が可能になる。

1 緒　　言

ソフトウェアの生産性・信頼性の向上のために，開発に計算機を活用することが多くなってきている。従来のソフトウェア生産では，プログラミング，テストといった限られた作業にだけ計算機が活用されるにとどまっていた。現在では，これらの作業ばかりでなく，計画・設計からテスト後の保守，機能改良に至るまでの一連の作業全体を支援するトータルな計算機システムが要求されるようになってきている。

ソフトウェア生産に計算機システムが活用される度合いが高まるにつれて，開発技法とともに開発者が技法を十分に活用できる開発設備が重要視されている。優れた開発設備がないと十分に生産性を高めることができない。計算機ハードウェアが高価であった時代には，センタにある計算機をバッチ処理で利用するか，あるいはインテリジェンスのない端末をセンタにある計算機に数台つなぎ，計算機リソースを時分割(TSS：Time Sharing System)で利用する方式がとられてきた。これらの方式では，利用者が多くなると計算機への要求を出してから結果を得るまでの応答性(バッチ処理ではターンアラウンド時間)が悪いこと，センタ運用上の時間，使用可能なリソースの制約など，利用上の問題も多かった。

これに対しワークステーションは，(1) 手元に置いて計算機リソースを専有利用できること，(2) ネットワークなどを用いて，センタ計算機や他のワークステーションと情報交換ができることなどにより，理想的なソフトウェア開発設備になり得る可能性をもっている。

ソフトウェア開発に用いられるワークステーションは，ワークベンチ(作業台)とか，ソフトウェア開発環境(Software Development Environment)とか呼びならわされている[1)~5)]。

ここでは，日立製作所が研究開発したソフトウェア開発用の新しいワークステーションである SEWB (Software Engineering Workbench：ソフトウェア エンジニアリング ワークベンチ)[1),5)~8)]について紹介する。

2 ソフトウェア開発の統合環境

図1は，今後のソフトウェア開発システムの在り方を示している。開発作業を支援する計算機システムと，開発者とはワークベンチを通して，対話を進めることにより，ソフトウェア(仕様書，マニュアルなどのドキュメントとプログラムコード)を完成させる。

ワークベンチに重要な要件には，

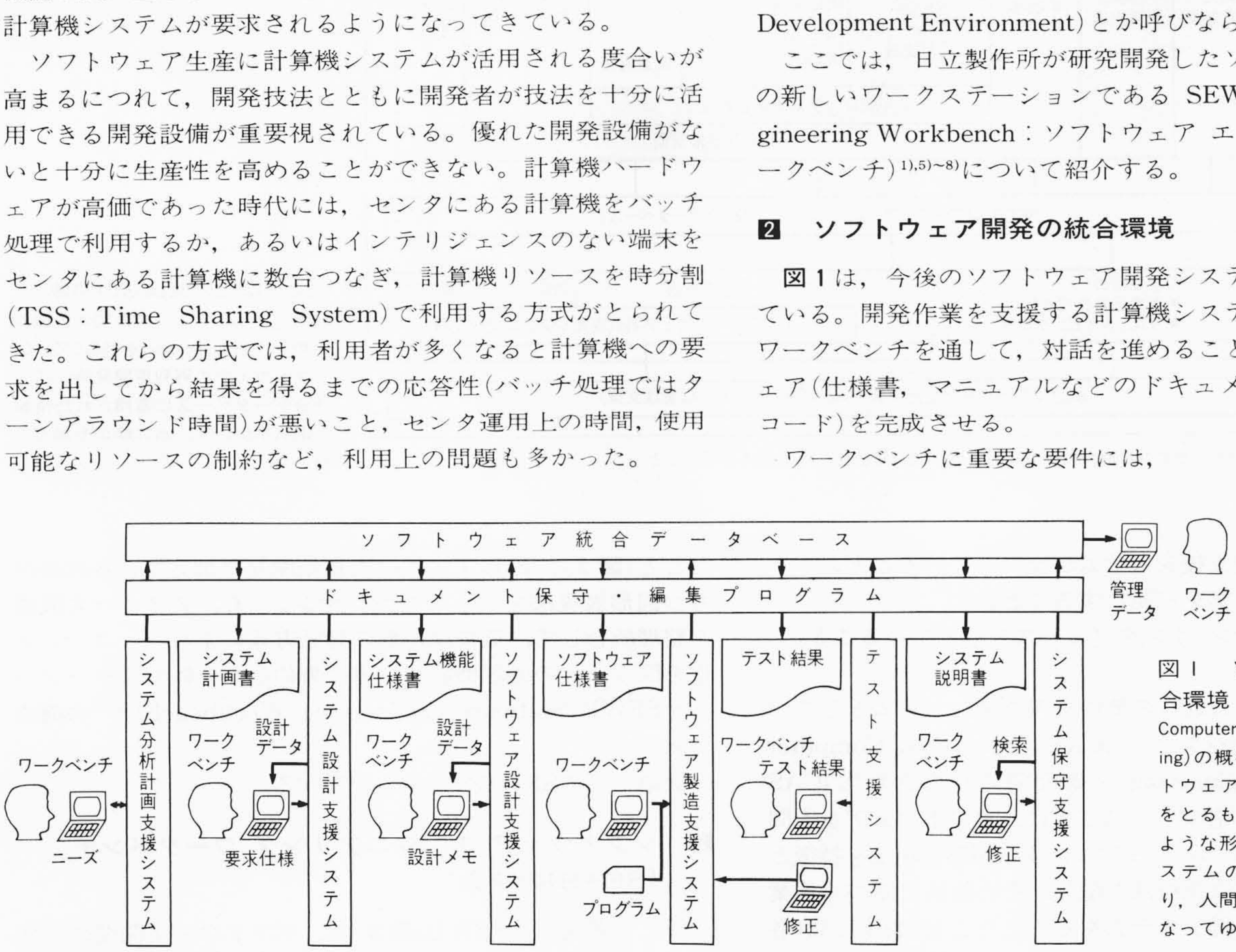


図1　ソフトウェア開発総合環境　図は，ICAS(Integrated Computer Aided Software Engneering)の概要を示すが，今後のソフトウェア開発は，このような形態をとるものと思われる。更に，このような形態を取りながら各支援システムの自動化の度合いが高まり，人間の介入の度合いが少なくなってゆく。

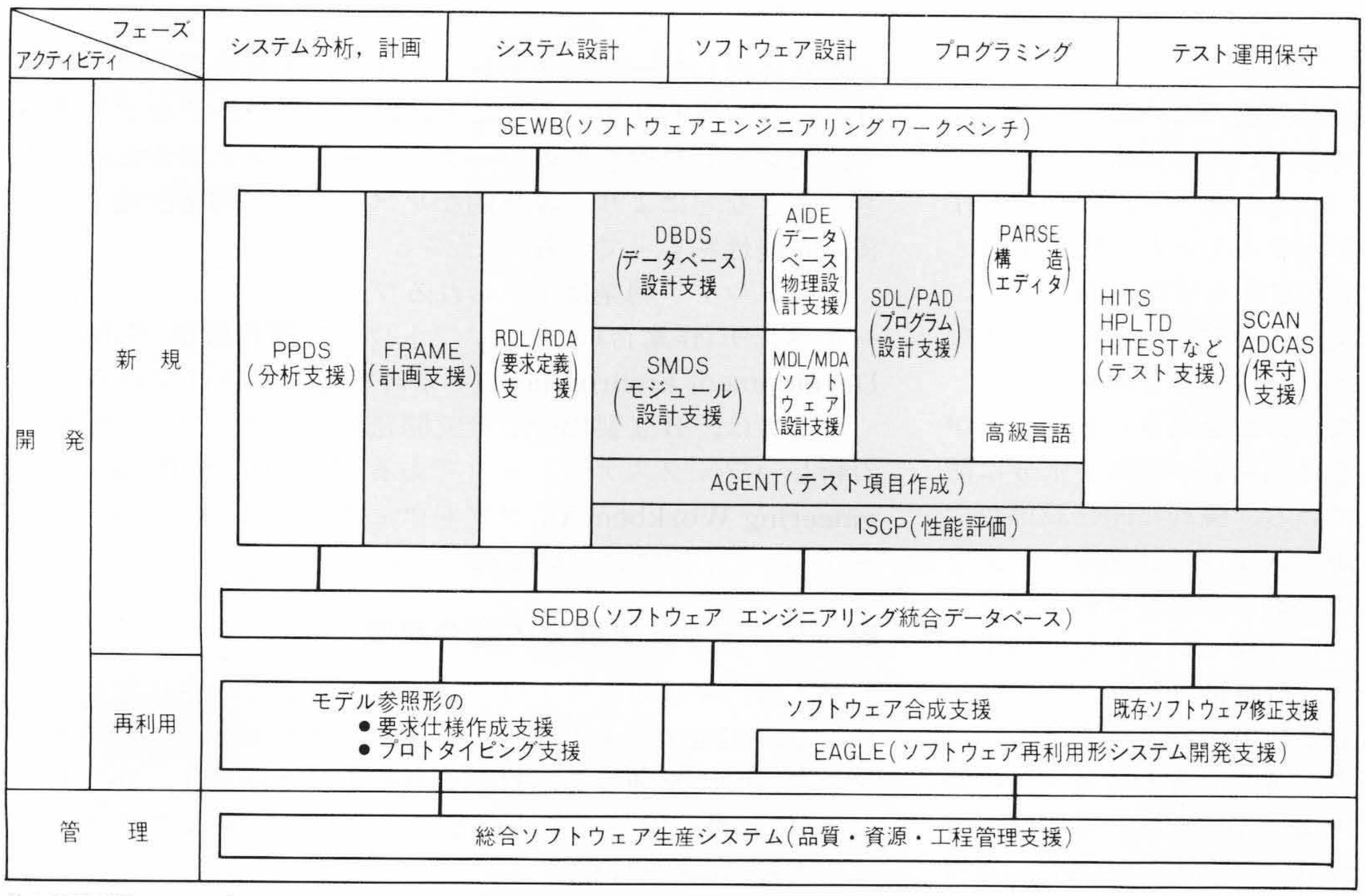
* 日立製作所システム開発研究所　** 日立製作所神奈川工場

表1 ICASの各フェーズの核となる「図形を中心としたソフトウェア構造化技法」 ICASでは，ライフサイクルを通して一貫した方法論，言語，図形表現，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作成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	●フローネット図 ●ER図	●ストラクチャードチャート ●IPO ●データ構造図	●PAD図 ●高級言語	機能図式
	支援システム	PPDS FRAME	RDA DBDS，SMDS	MDA DBDS，SMDS	SDL/PAD PARSE	AGENT

注：略語説明（アルファベット順）

ADCAS(Auto Documentation Aid System), AGENT(Automated Generation System for Test case), AIDE(Adaptable data manager Information Design Package), DBDS (Date Base Design System), EAGLE (Effective Approach to Achieving High Level Software Productivity), ER (Entity Relationship), FRAME (Formalized Requirements Analysis Method), HITEST (Hitachi Programming Test System), HITS (Highly Interactive Testing and Debugging System), HPLTD (Hitachi Programming Language Test and Debug), ICAS (Integrated Computer Aided Software engineering), IPO (Input Process Output), ISCP (Integrated Tools for System Configuration Planning), 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), SCAN (Static Code Analyzer System), SD (Structured Design), SDL (Software Design Language), SEDB (Software Engineering Data Base), SEWB (Software Engineering Workbench), SMDS (Software Module Design System), SP (Structured Programming), ST (Structured Test)



注：略語説明 SEWB(Software Engineering Workbench)，その他の略語は表1下の注を参照のこと。

図2 ICASの諸技法の関連

図中 ▭ は，フェーズの間の情報の変換システム(Bridge)を表わしている。ソフトウェアの再利用開発時には，統合データベースに蓄積された情報を選択抽出して，追加修正を加えて用いる。

(1) 利用できる開発支援システムが充実していること。

(2) 一貫した開発作業を可能にすること。

(3) 優れたマンマシン インタフェースをもっていること。

などが挙げられる。

日立製作所では，これらの要件を満たすシステムとしてソフトウェア一貫生産システム ICAS (Integrated Computer Aided Software engineering)を研究開発してきた[1]。ICASの特徴は，(1)開発の各フェーズごとに，ソフトウェアを人間に理解しやすくし，かつビジュアルにする図形表現法を特徴とするソフトウェア構造化技法を配し，その技法に基づく作業をガイドする開発支援システムをそろえたこと(**表1**)，(2)各フェーズの情報をデータベースによって一元管理するとともに，フェーズ間の情報を，計算機との対話を通して変換する橋渡し技術(Bridge技術)を確立し，一貫した開発を可能にしたこと(**図2**)，(3)以上に述べた技術開発に加えて，各技法がもつ図形表現法を主としたマンマシン インタフェース機能を整理統合して，直接操作形の対話方式により統一コマンド化を図り，その処理機能を端末に集約した分散形のワークベンチSEWB(Software Engineering Workbench)[5)~8)]の確立である。

次章でこのSEWBの特徴を紹介する。

3 ソフトウェア エンジニアリング ワークベンチ(SEWB)の特徴

新しく開発したSEWB(**図3**)は，ソフトウェア開発の各種支援システムの統合化，高機能対話インタフェース，分散処理方式の採用により，ヒューマンフレンドリーなソフトウェア開発統合環境(**図4**)を実現する。次にSEWBの特徴を説明する。

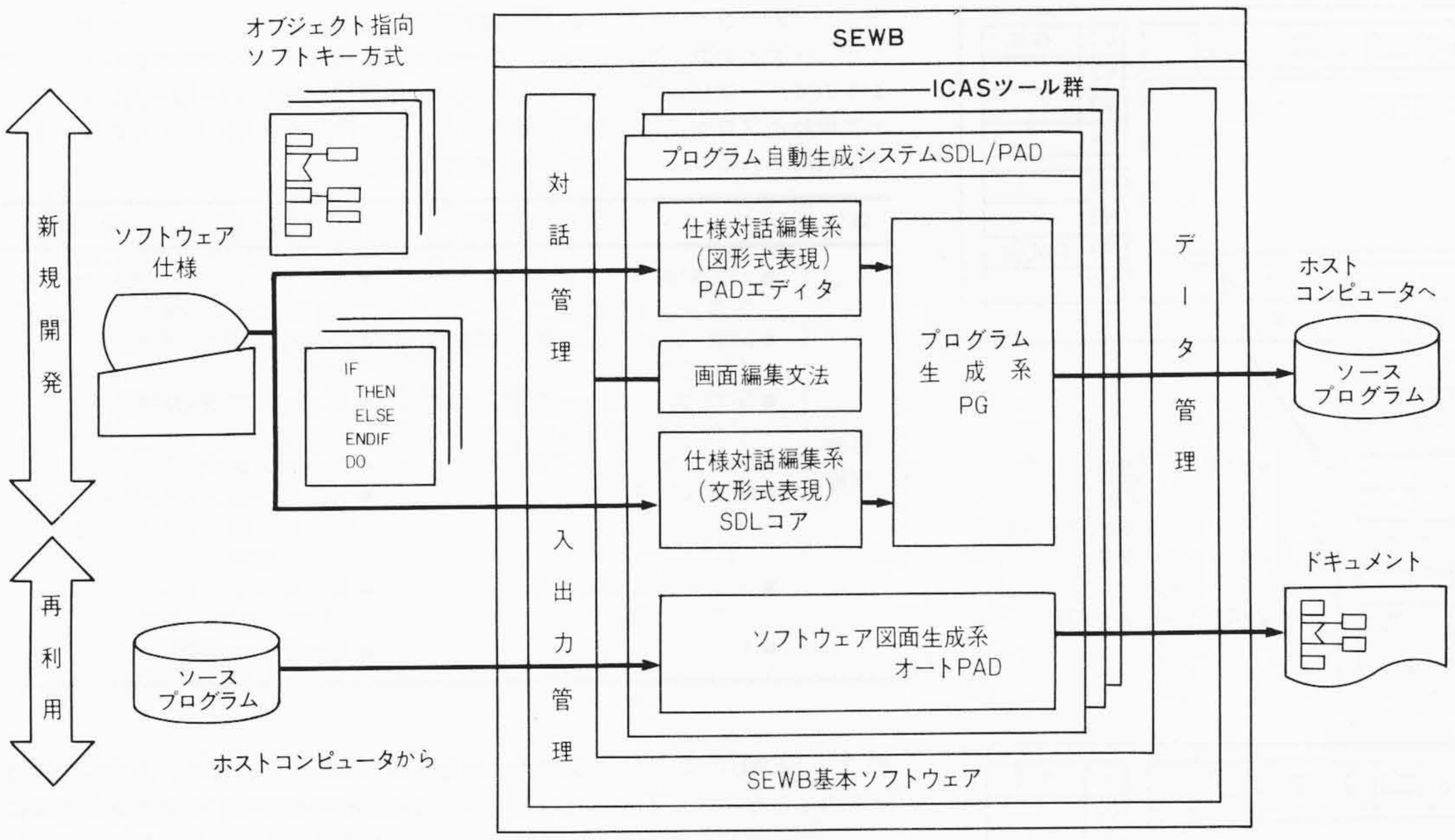


図3　SEWBのツール搭載例　SEWBへプログラム自動生成システムSDL/PADを搭載した例を示す。なお上図は，すべてワークステーション上の機能を示す。完成したソースプログラムは，ホストコンピュータに送られ，ホストコンピュータの最適化コンパイラを通して，効率の良い機械語に翻訳される。

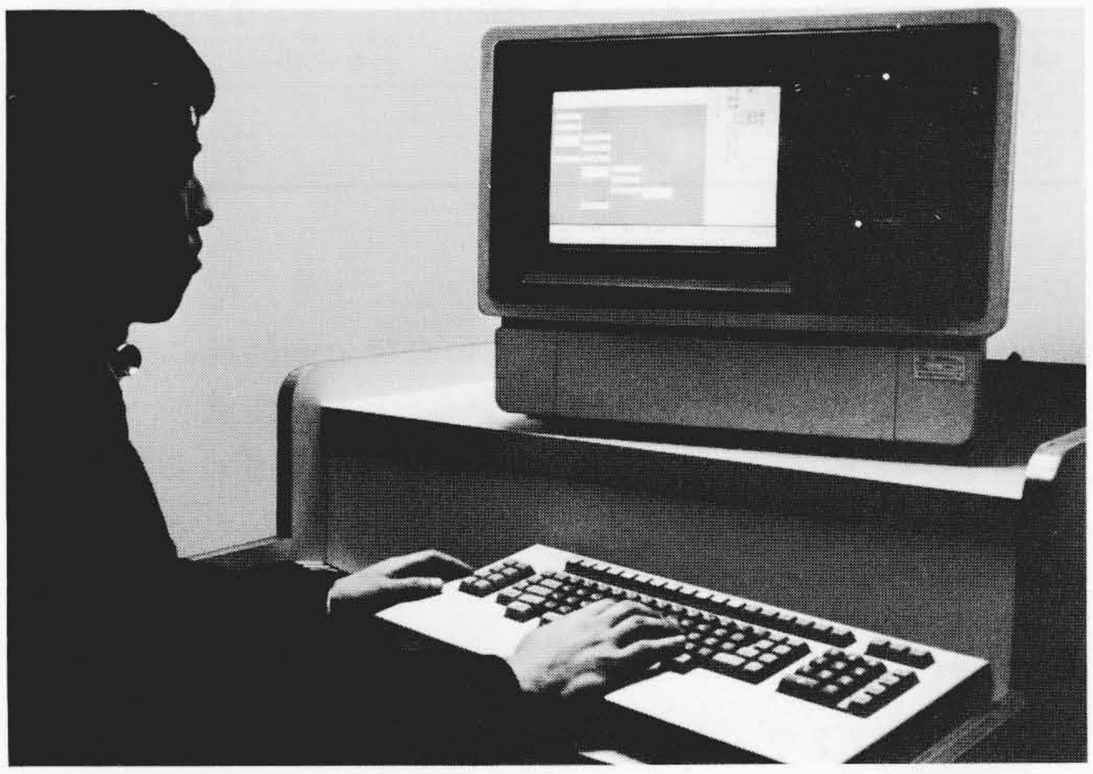


図4　SEWBを用いたソフトウェア開発　SEWBのヒューマンフレンドリーなマンマシンインタフェースを通して，プログラム自動生成システムSDL/PADを利用している光景を示す。開発者は，図の要素を積み木細工のように組み合わせさえすれば，プログラム コードは自動的に支援システムが作成してくれる。

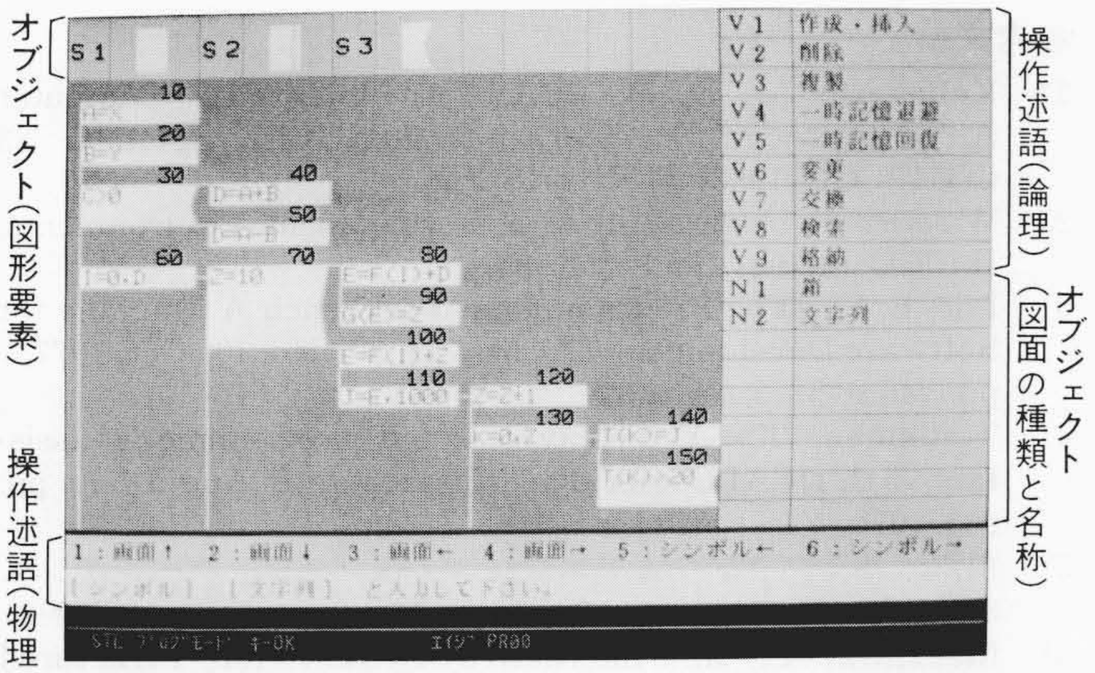


図5　オブジェクト指向ソフトキー方式　技法及びツールの利用法をオブジェクト(手順図，データ図などの図面の種類，及びそこで用いられる図形要素)と操作(挿入，削除など)とに分けたガイダンスが表示される(オブジェクト指向)。計算機からの誘導に従って，開発作業者は，対応する記号を選択指示する(ソフトキー方式)ことにより，「～を～する」という日本語と同じ順序の指示を計算機に与え，これを繰り返してソフトウェアの開発が進められる。

3.1　統合化

SEWBは，様々な開発技法(ウォータフォール形，プロトタイピング形，再利用形，再生形など)に基づいて開発された種々の開発支援システムを順次搭載することによって，ソフトウェア開発作業の全体を機械化する統合開発環境を構築する。ある支援システムの出力が，別の支援システムの入力となり，一連のツール群を連携させる一貫化，仕様情報やプログラム コードを統合データベースにより一元管理することによって支援システムを結合すると同時に，支援システム間に横断的な機能を共通ソフトウェア(カーネル)化することにより，新たな支援システムの追加開発を容易とする増殖機能をもっている。

3.2　高機能対話インタフェース

SEWBでは開発作業を，ソフトウェア情報を対象とした人間と支援システムとのコミュニケーションに基づく創造活動としてとらえている。このコミュニケーションを円滑にする施策として，(1) 図形の要素を積み木のように組み合わせる要領で，ソフトウェアを作成するための画面上で図形形式のコマンドを選択操作する。「オブジェクト指向ソフトキー方式」(**図5**)，(2) 図面を作る文法を内蔵させ，配置，結線，図形サイズなどを自動的に決定する「図面文法内蔵」による誘導形対話方式(**図6**)，(3) 図面の高速処理により対話応答性を良くするための高速図形発生機構※) などを開発しており，ユーザーに親しみやすい対話インタフェースを実現している。

3.3　分散処理

SEWBが，ワークステーションとホストコンピュータの結合(マイクロ・メイン フレーム リンケージ)による分散処理を採用している理由は，(1) 計算機の負荷分散による応答性の向上，(2) 大形計算機リソースを共同利用するときの運用上の制約の解消，(3) 開発者の好みに応じたフレキシブルな対話インタフェースの構築などである。(3) に関しては大形計算機の得意とするところは，画一化された処理は豊富なリソースを

※)　円，直線などの図形要素をドットに展開するマイクロプログラムと，バイポーラのビットスライス マイクロプロセッサから成る高速演算機構である。

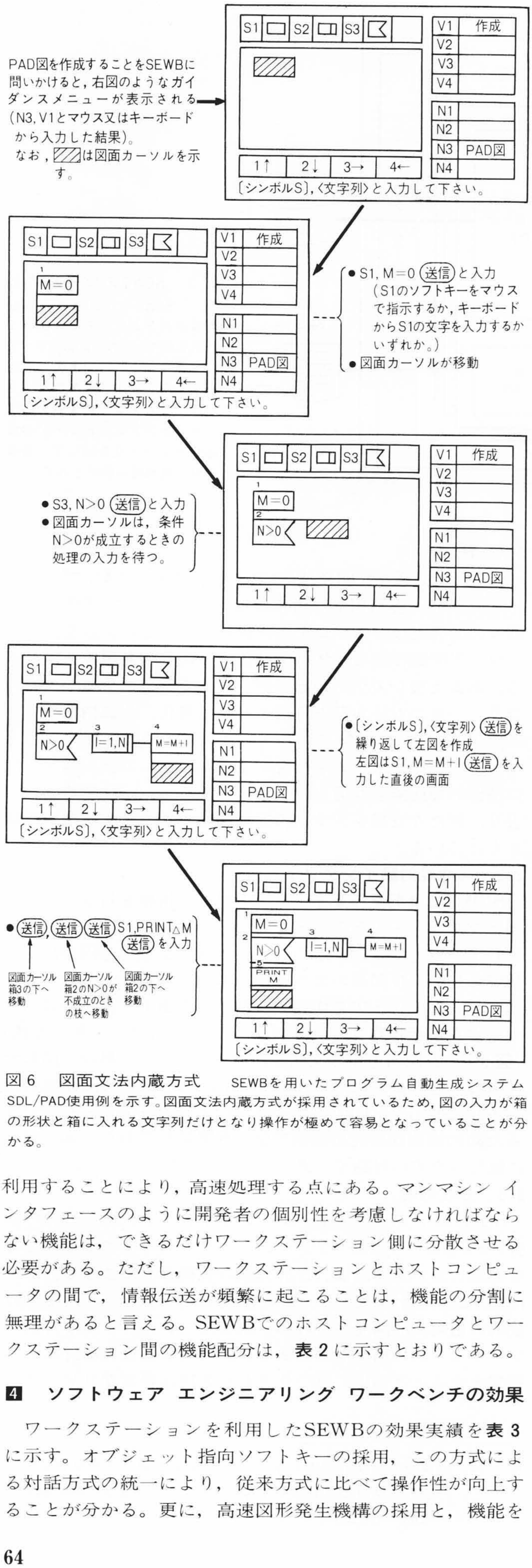


図 6 図面文法内蔵方式 SEWBを用いたプログラム自動生成システムSDL/PAD使用例を示す。図面文法内蔵方式が採用されているため，図の入力が箱の形状と箱に入れる文字列だけとなり操作が極めて容易となっていることが分かる。

利用することにより，高速処理する点にある。マンマシン インタフェースのように開発者の個別性を考慮しなければならない機能は，できるだけワークステーション側に分散させる必要がある。ただし，ワークステーションとホストコンピュータの間で，情報伝送が頻繁に起こることは，機能の分割に無理があると言える。SEWBでのホストコンピュータとワークステーション間の機能配分は，**表 2** に示すとおりである。

4 ソフトウェア エンジニアリング ワークベンチの効果

ワークステーションを利用したSEWBの効果実績を**表 3** に示す。オブジェット指向ソフトキーの採用，この方式による対話方式の統一により，従来方式に比べて操作性が向上することが分かる。更に，高速図形発生機構の採用と，機能をワークステーションへ分散させたことにより，応答性は従来のノンインテリジェント端末を用いた時分割処理に比べて1秒以内と，$\frac{1}{3}$から$\frac{1}{60}$に改善されている。入力操作量も，「図面文法内蔵方式」により，従来方式に比べ$\frac{1}{3}$に改善される。これらの効果は，ワークステーションとホストコンピュータの連携によるものである。

表 2 ワークベンチでの機能分散例 ワークステーションとホストコンピュータとの間の機能配分は，ワークベンチの利用者の作業内容によって決まるので，一概に決めることはできない。本表はSEWBでの分散例で，ソフトウェア設計とプログラム作成及び単体テストまでが，作業者に課せられている場合に利用されるワークベンチの典型例である。

媒体	ワークステーション側機能	ホストコンピュータ側機能
搭載機能	● 対話制御（メニューガイダンスコマンド解釈） ● 図面，言語入力処理(PAD図，SDL言語など) ● 図面，言語エディタ(構造エディタ，フルスクリーン エディタ) ● 図面，言語インタプリタ ● 図面言語テスタ ● 図面，言語相互変換(SDL/PAD) ● データ変化転送	● ソフトウェア統合データベース(作成，更新，検索) ● フェーズ間情報変換(Bridgeツール) ● フェーズ間情報解析 ● ドキュメント印刷 ● ソフトウェア コンフィギュレーションコントロール（バージョン管理） ● 各種管理システム(工程，資源，品質) ● データ変化更新

表 3 SEWBの効果 図面文法内蔵方式の採用などにより，入力操作量は従来の$\frac{1}{3}$となっている。また応答性は，高速図形発生機構，分散処理の効果により，従来の時分割処理に比較して$\frac{1}{3}$～$\frac{1}{60}$に改善されていることが分かる。

項 目	SEWB方式	従来方式
一 貫 性	諸ツールに横断的な統一対話方式	ツール個別の対話方式
操 作 性	オブジェクト指向ソフトキー	キーボードにより逐次入力
入力操作量	1 対 3	
応 答 性	1 秒以内	3 ～60秒

5 結 言

以上，ソフトウェア開発に用いられるワークステーションの典型例として，ソフトウェア一貫生産システムICASの一環として新しく開発したSEWBについて紹介した。

ソフトウェア需要の急速な増大に対処するためには，開発技法，支援システムの充実に加えて，開発設備の充実が重要である。今後，SEWBのようなソフトウェア開発設備の適用は拡大していくことが期待される。

参考文献

1) Kobayashi M. et al.: ICAS: An Integrated Computer Aided Software Engineering System IEEE SPRING '83 COMPCON 1983. 4 Silver Spring
2) Wasserman A. I.: Software Development Environments, IEEE Computer Society (1981)
3) Ivie, E. L.: The Programmer's Workbench A Machine for Software Development CACM, Vol. 20, No. 10, Oct. 1977, pp. 746～753
4) Teitelman, T.: A display-oriented programmer's assistant, 5 th IJCAI Cambridge, Massachusetts Aug. 1977, pp. 905～915
5) 前沢，外：ソフトウェアエンジニアリングワークベンチシステム—構成と基本技術，日立評論，**66**，3，181～184(昭59-3)
6) Maezawa, et al.: Interactive System for Structured Program Production, Proc. of 7 th International Conference on Software Engineering, 1984, 3 Orlando, Florida
7) 青山：ソフトウェア開発に用いられる図形表現法，情報処理学会誌，**25**，5，451～461(昭59-5)
8) 前沢，外：ICASにおけるソフトウェアエンジニアリングワークベンチ分散型SEWB，第30回情報処理全国大会(昭60-3)