U.D.C. 681.32.06.002:〔658.512.2.011.56:681.322〕

ソフトウェア一貫生産システム“ICAS”基盤技術の確立

Integrated Computer Aided Software Engineering System “ICAS”

小林正和*　*Masakazu Kobayashi*
野木兼六*　*Kenroku Nogi*
片岡雅憲**　*Masanori Kataoka*
林　利弘***　*Toshihiro Hayashi*

ソフトウェアのライフサイクルを通しての生産性，信頼性及び保守性の向上を目的として，ICASの基盤技術を開発した。

主な内容は，(1)ソフトウェア ライフサイクルの各フェーズに，図形表記法を特徴とするソフトウェア構造化技法を開発したこと，(2)各フェーズ間の情報をデータベースで橋渡しするとともに，フェーズ間の情報の変換を計算機化する技術(Bridge技術)を開発したこと，(3)各支援システムの一貫利用を可能にする分散形ワークベンチの方式を確立したこと，である。

本成果を適用した場合には，机上評価によるとその適用前に比べ，生産性を2倍に向上できる見通しを得ている。なお本報では，本号に掲載されている他の関連論文との関係についても明らかにしている。

1 緒　　言

計算機ソフトウェアの需要は，年々増大している。しかも，ソフトウェアの機能は，高度化，複雑化，大規模化しており，その故障の与える影響も社会的に広範囲に及ぶようになってきている。しかし，ソフトウェアの信頼性，生産性は，ハードウェアのそれに比べて，まだ低いのが現状である(**表1**)。このような状況を打開するためには，従来のような人手に頼った個人的，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，生産方式を発展させる必要がある。

高品質のソフトウェアを効率良く開発するためには，(1)開発技法及びツール，(2)開発設備，(3)生産管理，(4)教育などの総合的な対策が必要である。このような，総合的かつ近代的なソフトウェア生産技術は，「ソフトウェア エンジニアリング技術」と総称される。

本号では，上記(1)と(2)について，日立製作所の最新の技術成果について紹介する。その中にあって，本報では，計算機ソフトウェアの誕生から，その使命を終わるまでのライフサイクルを通して，一貫して，ソフトウェアの生産性，信頼性を含めた品質の向上を図るための基本技術について紹介する。更に，本号掲載の諸技法との関連を述べる。

表1　ハードウェアとソフトウェアの比較　　ソフトウェアは，ハードウェアとは産物の特性が異なり，生産性，品質を高めにくい側面をもっている。しかし，従来，ハードウェアに比べて，生産技術の研究があまりにも遅れていたことも否めない。

比較項目＼対象	ハードウェア	ソフトウェア
産物の性格	● 有　形 （目に見える。）	●無　形 （目に見えない。）
産物の対象となる世界	●2値(ON-OFF) ●信号の世界	●多値に及ぶ。 ●意味情報の世界
産物を律する原理	●物理法則中心 （研究が進んでいる。）	●論理，人間行動に及ぶ。 （未解明の部分が多い。）
産物に期待される利用環境変化への適応	●あきらめられている。 （買替えが定着）	●期待されている。(改良，修正，拡張を強いられる。)
生産形態	●ある程度まとまった生産 ●試作と量産製造と分離	●個別生産，多種少量生産 ●試作と製造との区別不明確
生産性，品質の評価尺度	●ST(標準時間)，性能仕様など定量的尺度存在	●尺度自体研究課題
生産の標準化，合理化	●進んでいる。 （属人性小）	●目下の急務 （属人性大）
生産効率向上 （過去10年間）	50～100倍	2～3倍

2 ソフトウェア生産技術における従来の問題点とその対策

ソフトウェアのライフサイクルは，(1)システム分析・計画，(2)システム設計，(3)ソフトウェア設計，(4)ソフトウェア製造(プログラミング)，(5)システムテスト，(6)運用・保守の六つのフェーズに分けることができる(**表2**)。この中で従来の生産技術は，そのほとんどが(4)のソフトウェア製造フェーズに限られていた。また，(4)を含めて，他のフェーズに技法が存在したとしても，各々が独立しており，情報の受渡しがないなど，孤立していた。このため，期待されたほど生産性及び品質の向上に効果が発揮されなかった。

ソフトウェア一貫生産システムICAS(Integrated Computer Aided Software Engineering)[1),2)]は，これらの問題の解決を目的とした新しい体系的なソフトウェア生産システムである。

ICASの概要は，**図1**に示すとおりである。今回確立した基盤技術は，次のような特徴を備えている。

(1)　各フェーズごとに，図形表記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支援する計算機利用システムを開発したこと。

(2)　各フェーズ間の情報を，データベースで橋渡しするとともに，フェーズ間の情報を計算機との対話を通して変換する技術(Bridge技術)を確立したこと。

(3)　以上に述べた技法の開発に合わせて，各技法が個別にもつマンマシン インタフェース機能を整理統合して，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処理形ワークベンチ(SEWB：Software Engineering Work Bench)技術を確立し，一貫開発設備環境の充実化を図ったこと。

* 日立製作所システム開発研究所　** 日立製作所ソフトウェア工場　*** 日立製作所大みか工場

表2 ソフトウェア ライフサイクルの各フェーズとその作業概要
ソフトウェアの誕生から，その使命を終わるまでの過程は，表に示すように六つの段階(フェーズ)に分けられる。各フェーズの目標が異なるため，必要となる技術も異なる。

ライフサイクル フェーズ	作業の目標
1. システム分析 計画	●システム化の目的と解決手段の明確化 ●プロジェクトの到達点及び目標の設定
2. システム設計 → ハードウェア開発	●ユーザー要求の過不足のない仕様化 ●システムのハードウェア，ソフトウェア構成要素の決定
3. ソフトウェア 設計	●ソフトウェア機能要求を実現するソフトウェア方式の決定 ●設計の経済的な実現 (モジュール化，モジュール間インタフェース，モジュール内論理)
4. ソフトウェア 製造	●ソフトウェア設計仕様の計算機言語への翻訳 (コーディング) ●計算機言語への翻訳誤りの除去(デバッグ)
ハードウェア → 5. システムテスト	●作成されたシステム全体の要求との合致を検査 (機能，品質)
6. 運用・保守	●完成システムの使用と誤りの補修 ●使用環境変化に応じた修正・改良

次章に，以上の(1)～(3)を詳細に説明する。

3 ICASが提供する基盤技術

3.1 各フェーズの核となる「図形表記法を特徴とするソフトウェア構造化技法」

ICASは，対話形の生産システムであり，各フェーズごと，一つのフェーズから他のフェーズへの移行時には，必ず人間によるレビューを行なう方式になっている。これは，ソフトウェアの品質を高める上で重要なことである。この場合に，ソフトウェアが，(1)マクロな表現からミクロな表現へと段階的に構成されていること(これを「構造化」と呼ぶ。)，(2)記述の内容構成が，少数の定義が明確な図形記号を用いて表現されていること，が有効である。ICASでは，ソフトウェアのライフサイクルの各フェーズごとに，そのフェーズの生産物を「構造化」するための，図形表記法をもつ技法を配備した(**表3**)。

次にフェーズごとに，これらの技法の主なものの概略について紹介する。

3.1.1 システム分析・計画の技法—PPDS[3]

システム化の目的・目標は，以後の開発のすべてを規定するため，関与者の総意となっている必要がある。PPDS(Planning Procedure to Develop System)は，システムの関与者のランダムな意図，要求項目などを，計算機の助けを借りて整理し，大目的から副次的目的へと，階層化された目的樹木を作成する技法である。なお，システム分析・計画支援システムとしては，本号別掲載の システムエンジニア作業支援ツール「システムデザインオートメーション」[20]も有効である。

3.1.2 システム設計の技法—RDL/RDA[4]

システムの要求仕様を明確にすることは，要求に合致したシステムを作成する上で非常に重要なことである。要求仕様は，システムの利用者と開発者との合意に基づく内容でなければならない。したがって，理解しやすい記述と厳密な記述とが求められる。ICASでは，要求定義用に図形表現形式をもち，しかも形式性も高い言語RDL(Requirement Definition Language)を提供する。図形表現形式をフローネットと呼ぶ。フローネットの特徴は，ペトリネットの動作規則，抽象化機構，正規表現をそのモデルに導入したことにあり，これによって実行可能で，モジュール化され，かつ環境とのインタフェース記述を伴った要求定義が可能になる(**図2**)。更にICASでは，RDLによる仕様の記述を入力として受け付け，仕様の不完全性などのチェック及び仕様ドキュメント編集の機能をもつ計算機システムRDA(Requirement Definition Analyzer)を合わせて提供している。

3.1.3 ソフトウェア設計の技法—ADDS[5]

ソフトウェアを保守しやすくするためには，(1)全体のソフトウェアを単一機能で，しかも互いに関連の少ない独立性の高い要素(モジュール)に分け，これらの構成要素を組織的に組み合わせた形態とすること，(2)各々のモジュールを，その機能と機能の実現方式とに分け，モジュールの外部からは，機能だけが利用できる形態とすること，が有効である。ICASが提供するADDS(Automated Design Documentation System)は，上記(1)及び(2)の形態のソフトウェアを，要求仕様から組織的に作る技法と，その技法を用いて構造化されたモジュール構造を設計する作業を支援する計算機システムを提供する。ADDSの詳細は，本号の別掲載の論文[5]で説明してある。テスト技法については種々の会話形テスト技法[16),17)]があり，本号にはソフトウェア項目作成支援システム[9]のほか，2種のテス

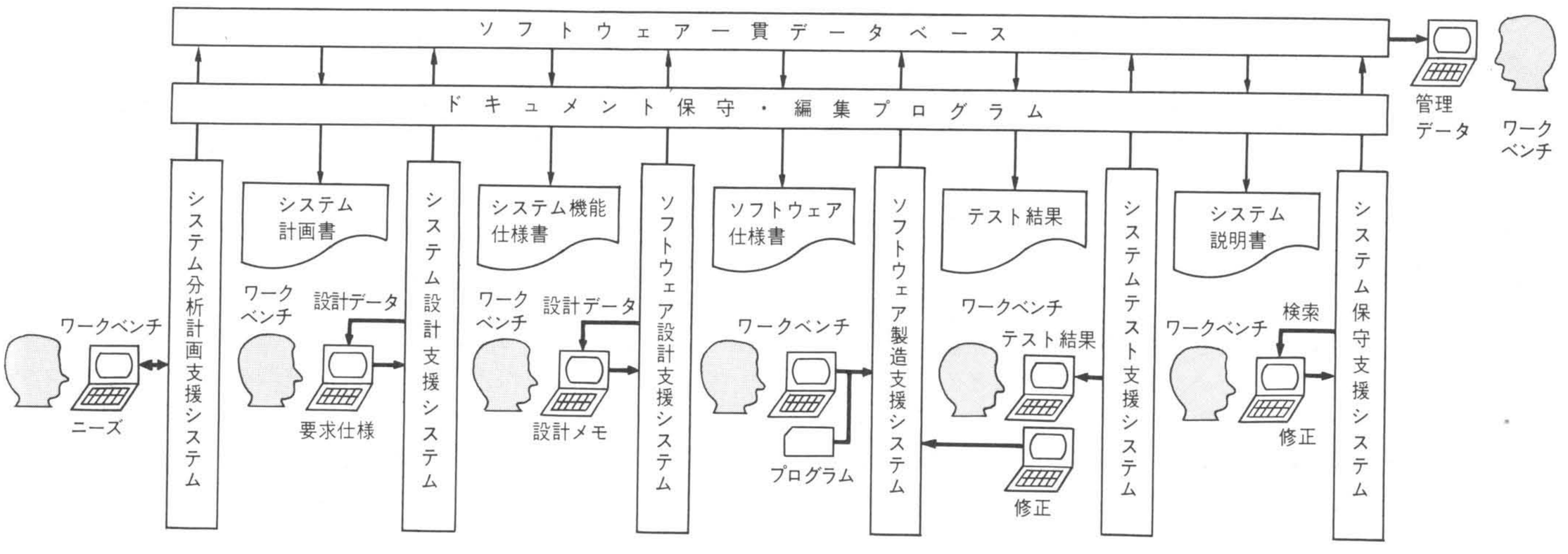


図1 ICASの概要 ICAS(Integrated Computer Aided Software Engineering)の諸技法を用いることにより，対話形で計算機システムの開発及び保守を，一貫した流れ作業で行なうことができる。

表3　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て方法論，言語，図形記号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，高品質のソフトウェアが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	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)	ソフトウェア設計言語(SDL)		—
	図形記号表現	●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	機能図式
	支援システム	PPDS FRAME	RDA DBDS	ADDS DBDS	SDL/PAD	AGENT

注：略語説明
PPDS(Planning Procedure to Develop System)
RDA(Requirement Definition Analyzer)
ADDS(Automated Design and Documentation System)
SDL/PAD(Software Design Language/Problem Analysis Diagram)
AGENT(Automated Generation System for Test case)
FRAME(Formalized Requirements Analysis Method)
DBDS(Data Base Design System)
RDL(Requirement Definition Language)
SDL(Software Design Language)
ER(Entity Relationship)
IPO(Input Process Output)
SA(Structured Analysis)
SD(Structured Design)
SP(Structured Programming)
ST(Structured Test)

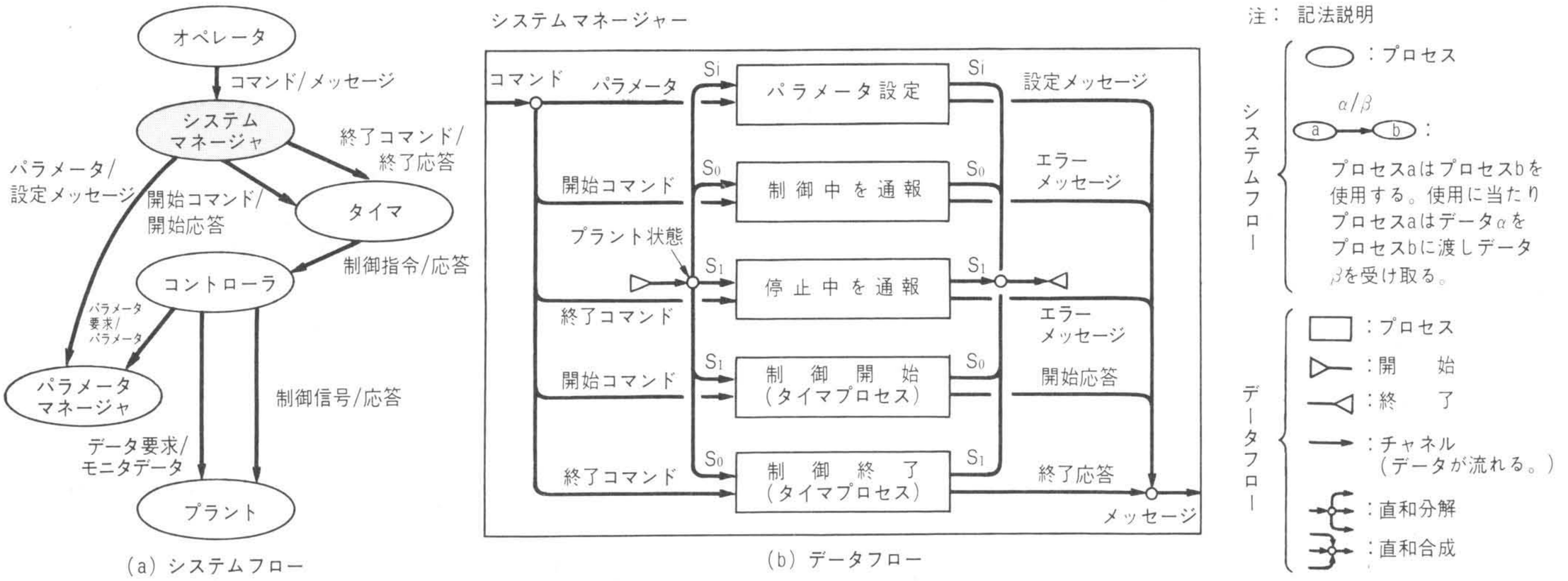


図2　フローネット記述例　フローネットによる要求仕様の記述は，(1)システムフロー図2(a)と(2)データフロー同図(b)とで構成される。システムフローは，システムの構成成分間の使用・被使用関係をプロセスの階層として記述するものであり，データフローは，プロセスの動作を，入力データから出力データへの変換として記述するものである。

ト技法[18),19)]合計3論文が掲載されているので，ここでは説明を省略する。

3.2　フェーズ間の橋渡し技術

3.2.1　一貫データベース方式

ICASでは，ライフサイクルの各フェーズ間にわたって伝達される必要のある情報を，すべてデータベース化し，フェーズ間に最新情報が伝達される方式SEDB(Software Engineering Data Base)を開発した(**図3**)。フェーズ間情報のデータベース化に当たっては，全情報を，データ構造のモデル化技法の一つであるER技法[7)](Entity Relationship Model)を用いて，統一モデルを作成した。

更に，フェーズ間の情報の保全性(Integrity)を制御する機構，変更履歴を管理する機構，変更作業手順管理機構を設けて，データベースの内容の一貫性を管理する方式を開発した。

3.2.2　対話形フェーズ間情報変換技法

(1)　システム分析・計画とシステム設計との間の変換—FRAME[6)](Formalized Requirements Analysis Method)

これは，システム分析・計画フェーズで作成された目的樹木に対して，目的実現のための機能，機能間の情報の流れ，機能の実現手段及び情報の伝達手段を与えて，業務処理時系列フローを作成する技法である。これにより，費用投資効果分析などが容易に行なえる。

(2)　システム設計とソフトウェア設計間の変換

(a)　要求仕様から処理モジュールへの変換—SMDS(Software Module Design System)

これは，フローネットで記述された要求仕様を一定の手続きで，計算機の処理モジュールと記憶モジュールから成るモジュール構造に，対話形で変換する技法である。モジュールの処理優先度，主メモリの常駐・非常駐，タスク構成などの決定を対話を通して行なえるようになっている。

(b)　要求仕様からデータベース設計への変換—DBDS[7)](Data Base Design System)

この技法は，(i)ソフトウェアに対する要求仕様から，データとプロセスとを分離して，(ii)データについて，その意味に着目して，事象(Entity)と事象間の関係(Relationship)から成るデータモデル(ERモデル)を対話形式で構築し，(c)　その

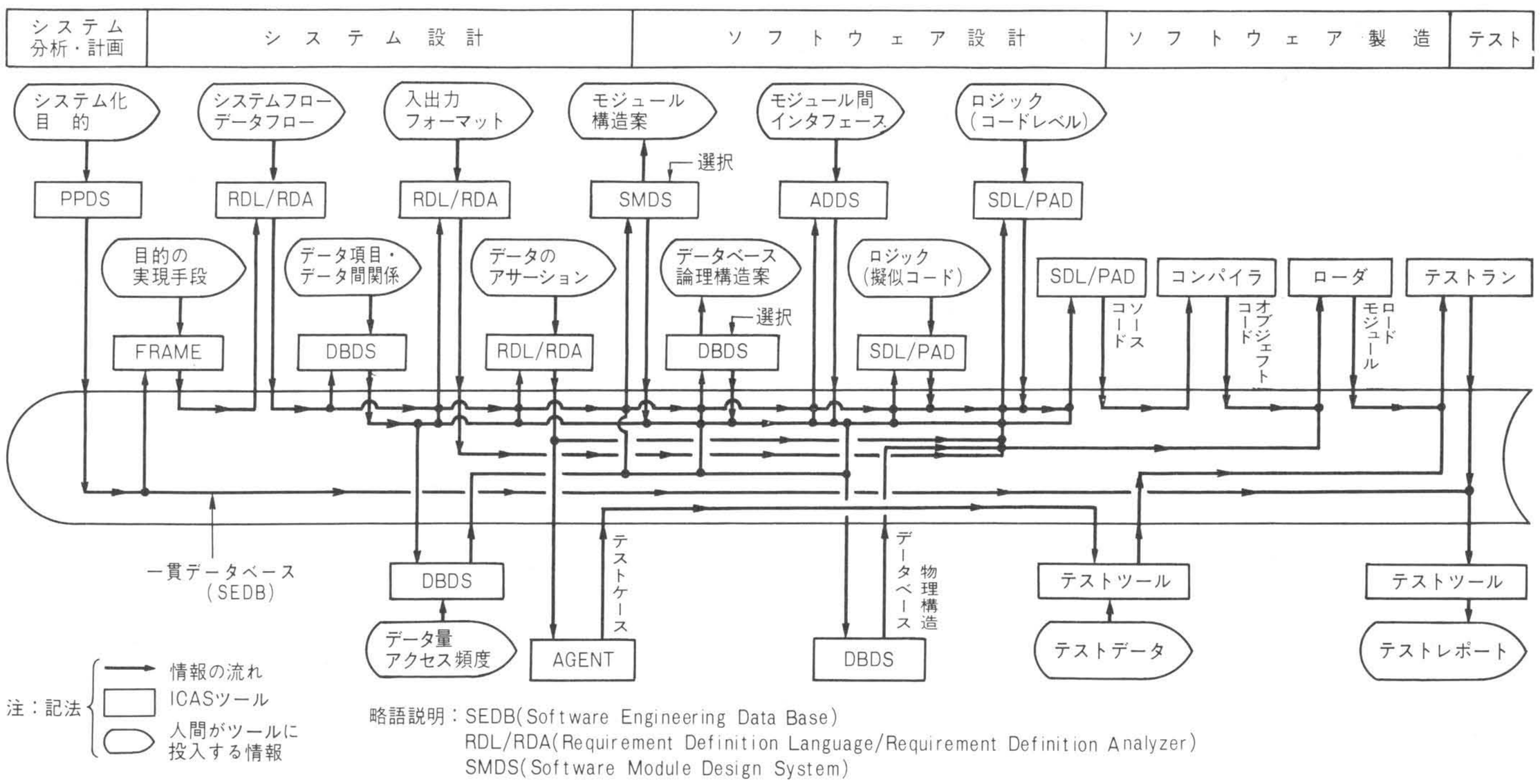


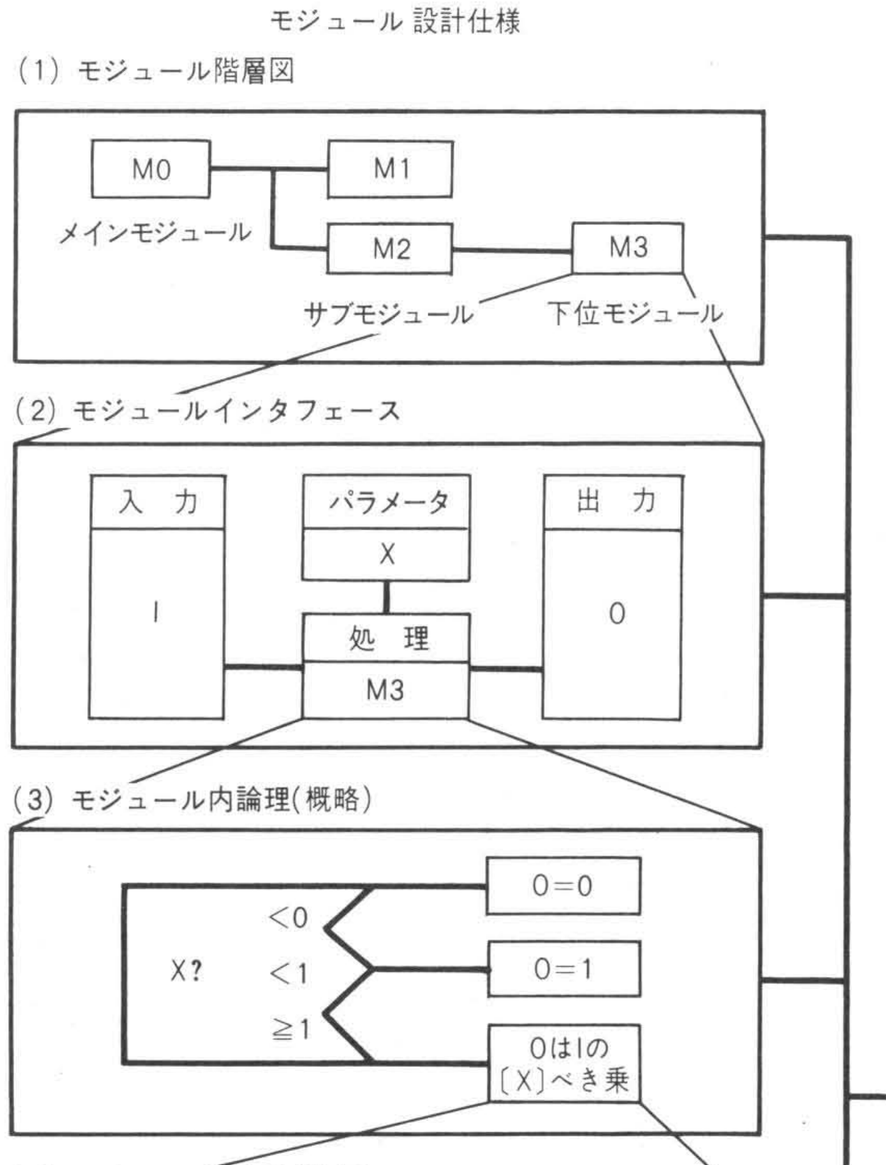
図3 一貫データベース方式の概要 一貫データベース内の全情報はERモデル(Entity-Relationship Model)によりモデル化されており，各フェーズの各システムは，ERモデルインタフェースを通して情報の授受ができるようになっている。

結果とデータ量，アクセス頻度などのデータ利用上の定量値から，指定されたDBMS(Data Base Management System)に合致したデータベース構造を作成するものである。

(3) ソフトウェア設計とソフトウェア製造の間の変換—SDL/PAD[8)]

この技法は，モジュール仕様を入力とし，高級言語プログラムと図形表現による仕様書を自動作成する(**図4**)。入力となる仕様の記述に自然言語ふうの言語SDL(Software Design Language)を用い，図形表現に理解性に優れたPAD(Problem Analysis Diagram)を用いて，両者の長所を計算機を媒介として統合化した点に特徴がある。また，高級言語プログラムを入力とし，これを解析し，PAD図による仕様書を作成する機能をもち，仕様書とプログラムとの一体化が図れるようになっている。

(4) システム設計とシステムテストの間の変換—テスト項目自動作成技法



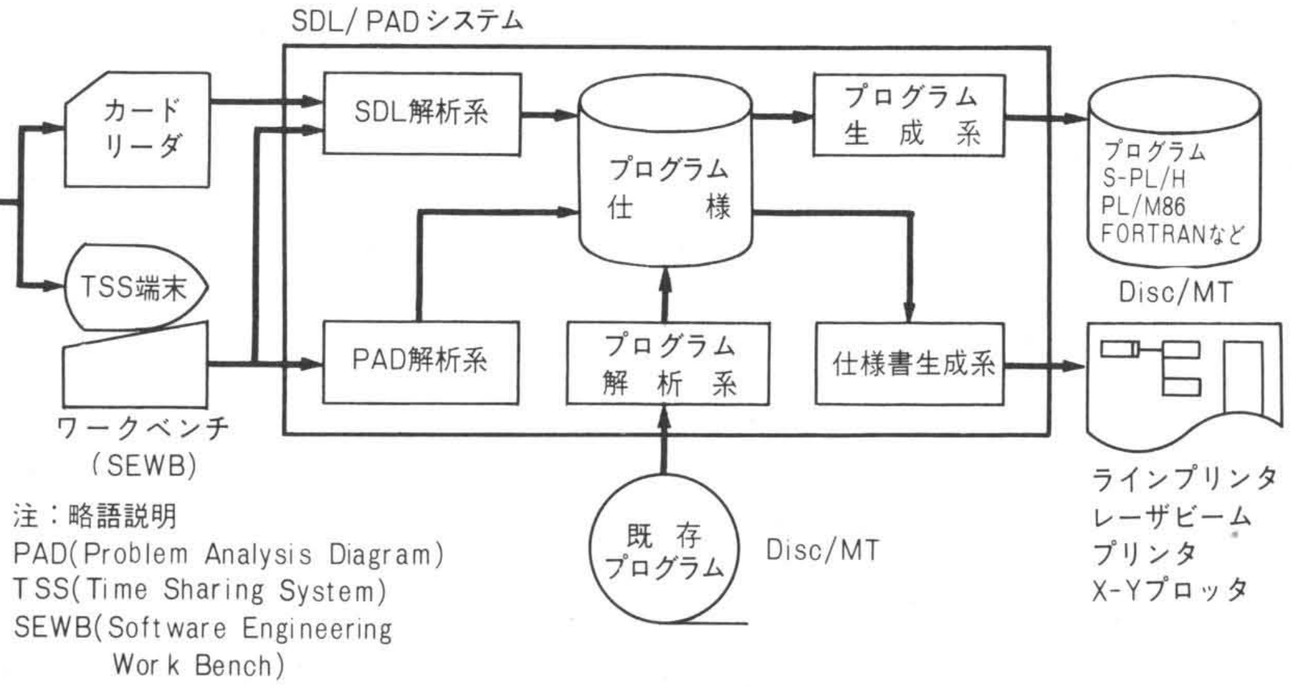
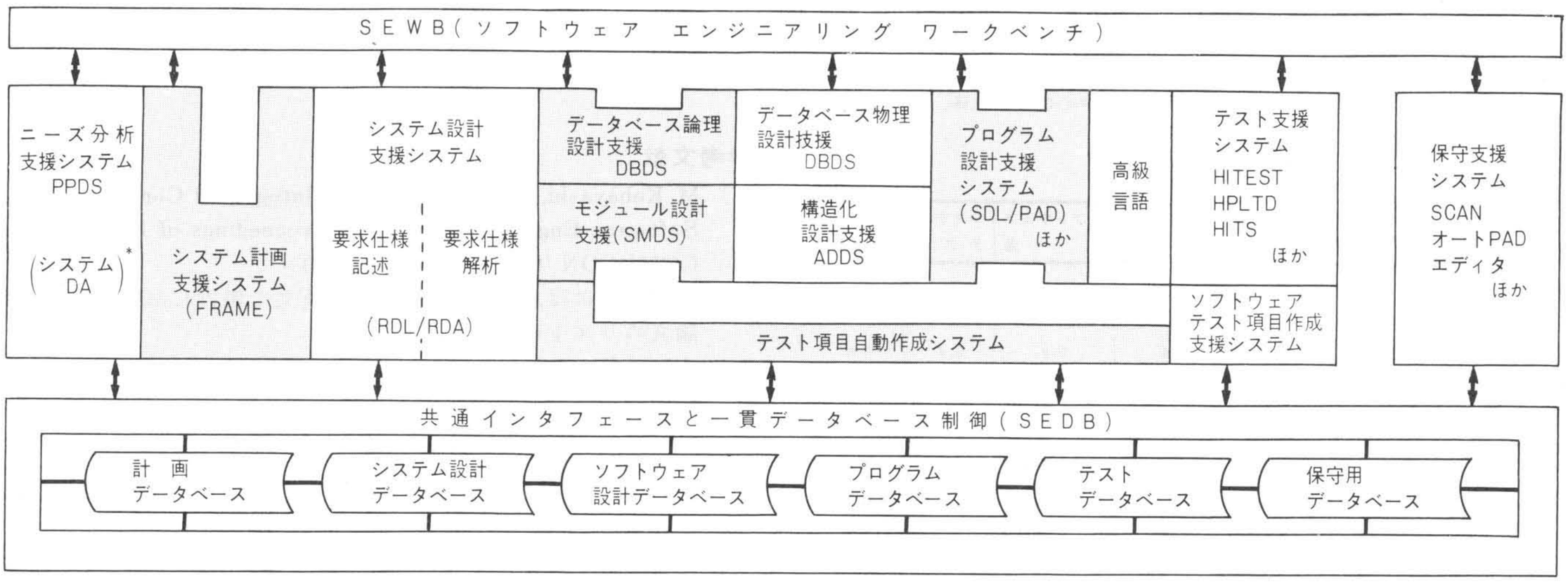


図4 プログラム及び図形化仕様書自動作成システムSDL/PADの概要 SDL/PADは，モジュール設計仕様を入力とし，計算機言語によるソースプログラムを自動作成するシステムである。更にソースプログラムを入力として，仕様書を編集する機能も合わせてもっており，仕様書とソースプログラムの一体化を実現する。



注：＊印は将来搭載予定のものを示す。

図5　ICASの諸技法の関連　図中で凹形，凹形はフェーズ間の情報変換システムを表わしており，縦の切り込みが深いほど人手による作業が多いことを示し，切り込みの幅が広いほど情報のギャップが大きいことを示している。また，各ツールの間ではデータベース制御部を介して，情報の受渡しが行なわれる。

この技法は，フローネット技法(**3.1.2**)で記述された要求仕様から，必要最小限のテストケースを作成する。概略の機能は，(a)フローネットのフローを解析して，ソフトウェアの入出力の論理関係と順序関係とを抽出する，(b)有向グラフのカバレージ技法及びハードウェア回路のテストパターン作成技術を応用って，状態遷移グラフを作成する，(c)この状態遷移グラフの初期状態から出発して，グラフの各弧を少なくとも1回通って，初期状態に戻る遷移を抽出する，(d)この遷移に対応して冗長性がなく，高い効用をもつテスト項目を抽出する，で構成されている。(d)の部分は本号に詳細な記述がある[9]。

図5は，ICASの諸ツール間の関係を示したものであるが，図中凹形，凹形などで表わされているのが，フェーズ間の変換システムである。

3.3　ソフトウェア一貫生産のための設備環境

ICASは，計算機との対話を主体とした生産システムである。この対話のための設備として，端末にマイクロコンピュータを搭載し，ホストコンピュータとの間で，機能分散を図れる新しい端末システムSEWB[10]を開発した(**図6**)。SEWBによって一貫作業ができるためには，各ツールの操作が統一化されている必要がある。SEWB開発に当たっては，各フェーズごとのシステムが必要とするマンマシン操作をすべてリストアップし，それらを動詞と目的語に分類し，コマンドの構成を統一した。更に，こうして統一したコマンドの入力を，画面上のシンボルメニューのピックアップ方式とし，更に，各フェーズごとのシステムの言語や図面の文法を内蔵化し操作ガイダンスの充実とともに，操作の自動化の度合いを高め，操作性向上を図った。

以上，本章でICASの概要について説明したが，本システムに対する海外からの関心も高く，IEEE(米国電気電子学会)主催のCOMPSAC'82[22](計算機のソフトウェアとその応用に関する国際会議)などでは，類似29システムのなかにあって，非常に高い評価を得ている。

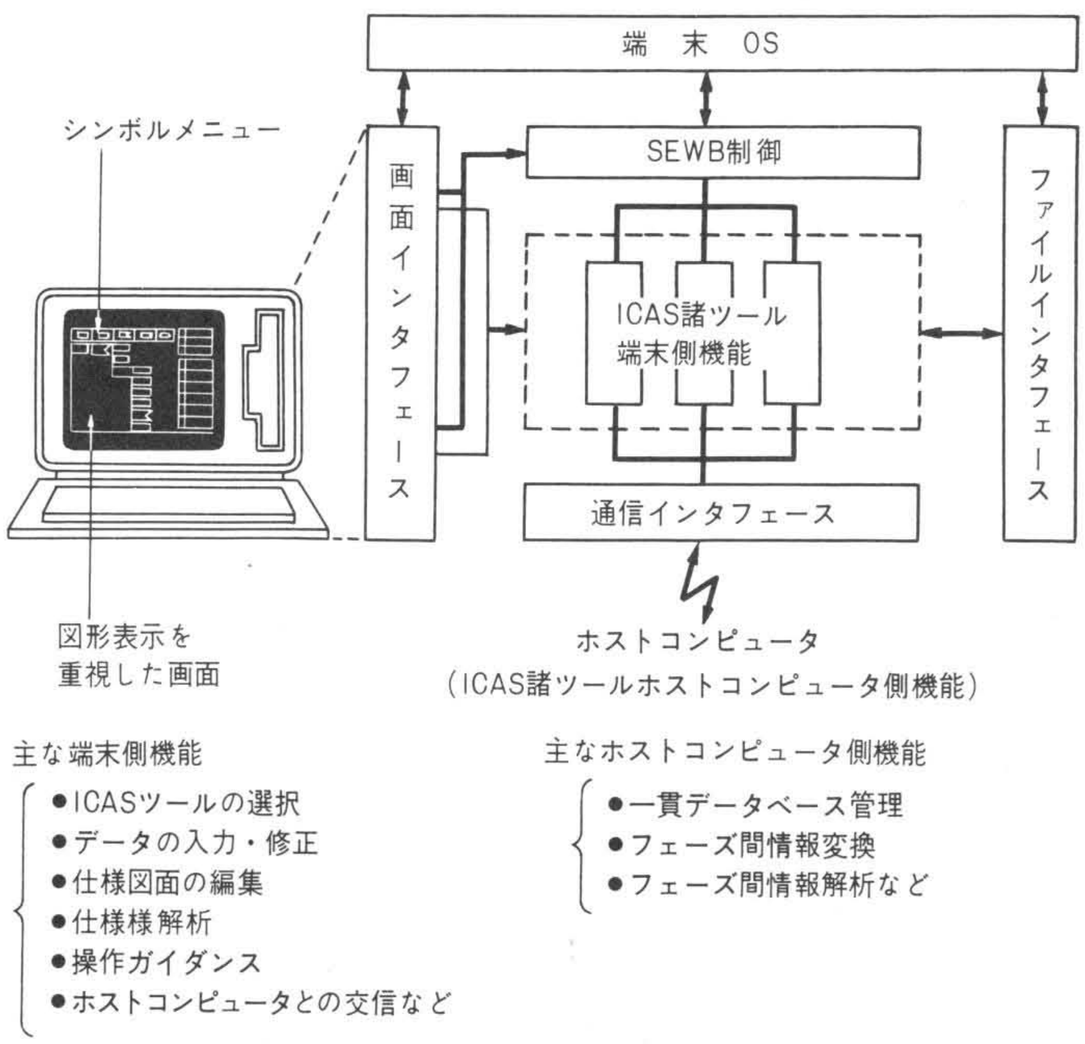


図6　分散処理形ワークベンチ(SEWB)　図形表示を中心とした統一したマンマシンインタフェースをもつ高インテリジェントなソフトウェア開発用専用端末であり，ホストコンピュータの能力に左右されない一定したクイックレスポンスが確保できる。

4　ICASの一貫性の評価

ICASでは，そフトウェアの一貫生産を目的として，基盤となる技術を開発してきた。ここでは，ICASを適用した場合と，その適用前とを比較した効果を検討してみる。

一貫性の評価のために，次のような一貫化率とドキュメント化率という指標を導入する。

一貫化率αの定義を次のようにする。

$$\alpha=(a+0.5b)/(a+b+c) \cdots\cdots (1)$$

ここに　a：前フェーズから自動的に伝達される情報の量
b：前フェーズから伝達された情報を，人手を介して修正・加工し利用する情報の量(係数0.5は，マクロにみて機械化が50%であることを示す。)
c：前フェーズからは伝達されない情報の量

ドキュメント化率βの定義を次のようにする。

$$\beta=(a'+0.5b')/(a'+b'+c') \cdots\cdots (2)$$

ここに　a'：ICASツールから得られるドキュメントの量
b'：ICASツールから得られるドキュメントを人手

表4 ICASの一貫性評価 この表の数値は，机上評価によるもので主観的なものではあるが，傾向は表わしていると考えられる。この数値によると，一貫化率，ドキュメント化率共にICAS適用前に比べほぼ2倍に向上しているといえる。今後はシステム設計，システムテストを中心に一貫化率，ドキュメント化率を高める努力が必要といえよう。

(a) 一貫化率

フェーズ / 比較	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	システムテスト	トータル
ICAS 適用前	—	0.2	0.2	0.5	0.2	0.3
ICAS 適用後	—	0.4	0.8	0.8	0.4	0.6

(b) ドキュメント化率

フェーズ / 比較	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	システムテスト	トータル
ICAS 適用前	0.1	0.3	0.4	0.2	0.2	0.2
ICAS 適用後	0.1	0.6	0.8	0.8	0.4	0.5

により修正・加筆して得られるドキュメントの量
(係数の0.5の意味は，一貫化の場合と同じである。)

c'：ICASツールからは得られないドキュメントの量

一貫化率，ドキュメント化率の評価は，実際には非常に困難であるが，机上で算定した結果は，**表4** (a), (b)に示したとおりである。この数値は主観がかなり入っているが，傾向は示していると考えられる。この評価によれば，一貫化率，ドキュメント化率ともに，ICAS適用前に比べ，適用後は，ほぼ2倍向上することになる。したがって，生産性も2倍向上することが期待できる。

5 結 言

以上，ソフトウェアのライフサイクルを通して，ソフトウェアの生産性，品質の向上を目的とした一貫生産システムICASの基盤技術について紹介した。

ICAS技術を実際の生産現場で活用するに当たっては，生産の対象となるソフトウェアの性格，生産体制，ソフトウェア開発，ノウハウの蓄積状態など，それぞれの事情を踏まえた最適化を行なう必要がある。

日立製作所では，過去長年にわたる技術の蓄積を基礎とし，ICASの技術にのっとり，特定分野別のソフトウェア一貫生産システムが実用化されている。OSなどの基本ソフトウェアCASD[16] (Computer Aided Software Development System)，プロセスなどの制御ソフトウェア用の一貫システム[11]，電子交換機ソフトウェア用のCROSS[12] (Composite Real-time software Oriented Production Support System)，ビジネス ソフトウェア用のHIPACE/EAGLE[13] (Hitachi Phased Approach for High Productive Computer System's Engineering/Effective Approach to Achieving High Level Software Productivity)，SKIPS[14] (Hitachi SK Improved Production System)，HISPOT[21] (Hitachi Interactive application Software Production Optimizing Tools)，CANDO[15] (Computer Aided New facilities for system Development and its fixation to the Organization)，マイクロコンピュータ ソフトウェア用のμ-ICASなどである。これらのシステムは，今後も継続して機能強化を図りより良いシステムを目指す計画である。

参考文献

1) M. Kobayashi, et al.：ICAS：An Integrated Computer Aided Software Engineering System, Proceedings of IEEE COMPCON '83 pp. 238～244(1983)
この論文には，1983年3月までに英文で報告したICAS関連の論文のリストがある。

2) 小林，外：ソフトウェア一貫生産支援システム(ICAS)の概要 情報処理学会第25回全国大会講演論文集，451～452(1982)
この論文には，1982年9月までに情報処理学会で発表したICAS関連の論文リストがある。

3) 中尾，外：システム計画のためのシステム要求分析手法"PPDS"の開発，日立評論，**62**，12，867～870(昭55-12)

4) 西尾，外：フローネット技法による要求定義について，情報処理学会第25回全国大会講演論文集，507～508(1982)

5) 鈴木，外：ソフトウェア構造設計支援システム"ADDS"，日立評論，**66**，3，185～188(昭59-3)

6) 中尾，外：アプリケーションシステム要求仕様分析手法，日立評論，**63**，8，573～578(昭56-8)

7) 近藤，外：データベースの論理設計技法，日立評論 **64**，5，323～328(昭57-5)

8) H. Maezawa, et al.：Interactive System for Structured Program Production, Proceedings of 7th International Conference on Software Engineering, to be published, 1984

9) 野木，外：ソフトウェアテスト項目作成支援システム，日立評論，**66**，3，199～202(昭59-3)

10) 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)

11) 森，外：制御用ソフトウェア一貫プログラミングシステム，日立評論，**62**，12，889～892(昭55-12)
このシステムは現在大幅に機能の充実を図っている。

12) 黒崎，外：通信用ソフトウェア生産技術の高度化，日立評論，**64**，3，171～174(昭57-3)

13) 葉木，外：システム開発支援ソフトウェア"EAGLE"，日立評論，**66**，3，189～194(昭59-3)

14) 下田，外：ソフトウェア一貫生産システム-SKIPS，bit臨時増刊ソフトウェアツール，pp. 63～85，共立出版(1982-2)

15) 小野，外：構造化ソフトウェア開発システム"CANDO"，日立評論，**66**，3，195～198(昭59-3)

16) 片岡，外：ソフトウェア開発支援システム(CASDシステム)，日立評論 **62**，12，879～882(昭55-12)

17) 大島，外：制御用ソフトウェア機能一貫テストシステム"HITEST/F"，日立評論，**62**，12，893～898(昭55-12)

18) 黒崎，外：マイクロコンピュータ用会話形テスト支援システム"HITS"，日立評論，**66**，3，203～206(昭59-3)

19) 大島，外：オンラインデバッグ支援システム"HITEST-DEMO"，日立評論，**66**，3，207～210(昭59-3)

20) 山中，外：システムエンジニア作業支援ツール「システムデザインオートメーション」，日立評論，**66**，3，211～214(昭59-3)

21) 白石，外：オンラインデータ エントリシステム開発支援ツール"HISPOT"，日立評論，**66**，3，215～220(昭59-3)

22) H. L. Hausen, et al.：Software Engineering Environments：State of the Art, Problems and Perspectives：Proceedings of IEEE COMPSAC '82 pp. 326～355 (1982)