U.D.C. 681.32.06.002:[681.323.012.014:681.5]:519.68

HIDIC V90/50ソフトウェア開発支援システム

Industrial Realtime Software Development Support System for HIDIC V90/50

大島啓二* *Keiji Ôshima*
林　利弘* *Toshihiro Hayashi*
堀　雄太郎* *Yûtarô Hori*
中所武司** *Takeshi Chûsho*

計算機制御システムの開発ソフトウェア量は，制御規模の拡大と複雑化に伴って急増しており，ソフトウェアの生産性，信頼性及び保守性の向上は，解決を急がねばならない最も重要な課題の一つとなっている。

日立製作所では，従来からソフトウェアエンジニアリングの観点から各種ソフトウェアツールの開発に取り組んできたが，HIDIC V90/50では，更に一歩進めて，高性能ハードウェアの上に，「作りやすいソフトウェア」，「使いやすいソフトウェア」，「管理・保守しやすいソフトウェア」を目指して，計算機制御システムの設計段階から保守段階までを首尾一貫した思想で支援する強力なソフトウェアツール群を開発・整備し，体系化した。

1 緒　言

近年，計算機制御システムは，マルチコンピュータ化や機能分散ネットワーク化など，ハードウェアの技術革新に伴い，制御規模・範囲が急速に拡大しているが，これらを支えるためのソフトウェア人口の伸びは停滞している。一方で投資の早期回収の観点からシステムの早期稼動の要請が強い。

このような厳しい環境下にあって，与えられた期限と費用の中で所期の目的を達成するシステムを信頼性高く作り上げ，効率良く保守していくには，従来の手工業的なソフトウェア作りを近代的なソフトウェア作りに変革してゆくことが必須である。

HIDIC V90/50(以下，H-V90と略す。)では，この課題にこたえるために，

(1) 作りやすいソフトウェア
(2) 使いやすいソフトウェア
(3) 管理・保守しやすいソフトウェア

を目指し，高性能ハードウェアの上にソフトウェアエンジニアリング手法を駆使した強力な支援ツール群を開発し，ソフトウェア開発支援システムとして開発・整備し，体系化した。

2 ソフトウェア開発支援システムの体系

ソフトウェア開発支援システムは，**図1**に示すように，開発初期の設計支援から，開発完了前のテスト，更には保守に至るまでの全開発過程に対し幅広く体系的に支援するもので，これにより制御用アプリケーションソフトウェアの開発が，信頼性高くかつ効率良く行なえるようになっている。本システムでは，これらのねらいを達成するために，以下に述べる「バーチャルシステムアーキテクチャ」，「ユニコンセプトシステムアーキテクチャ」，「ビジブルソフトウェアアーキテクチャ」を三つの基本思想として採用し，その具現化を図った。

2.1 バーチャルシステムアーキテクチャ

ハードウェアデバイスの物理的なデータ構造や，制御方式を意識してプログラムを作成するのでは，プログラム自体が複雑なものとなり，作成効率及び出来上がったプログラムの信頼性も低いものになる。H-V90では，ソフトウェア開発支援システムの基本アーキテクチャとして，

(1) デバイスごとに異なるハードウェア特有の物理データ構造や制御方式から，アプリケーションプログラムを解放する。
(2) プログラムの構造や処理を変更することなく，プログラムの性能(応答性，処理性)の調整を可能とする。

ことを目指す「バーチャルシステムアーキテクチャ」の考えを大幅に採用し，「作りやすいソフトウェア」の実現を図った。

このための具体策として，**図2**に示すように「問題向きプログラミング言語」によるシステム環境仕様定義を，従来の「手続き向きプログラミング言語」によるプログラムの処理手続き記述から分離・独立させる方式をとった。

2.2 ユニコンセプトシステムアーキテクチャ

たとえツール自身の機能が豊富であっても，ツールごとにその操作の仕方がまちまちであっては，ツールを使う際，操作ミスを起こしやすく，またツールの機能も十分発揮されず効率も悪い。更にツールが有機的に関連づけられていないと，作成したプログラムやデータの有効利用が図れないことになる。

H-V90では，

(1) ツール共通の強力なコマンド体系
(2) ツール間ファイルの一元管理機能

の導入によってこの問題の解決を図った。

これらユニコンセプトシステムアーキテクチャの採用によって，

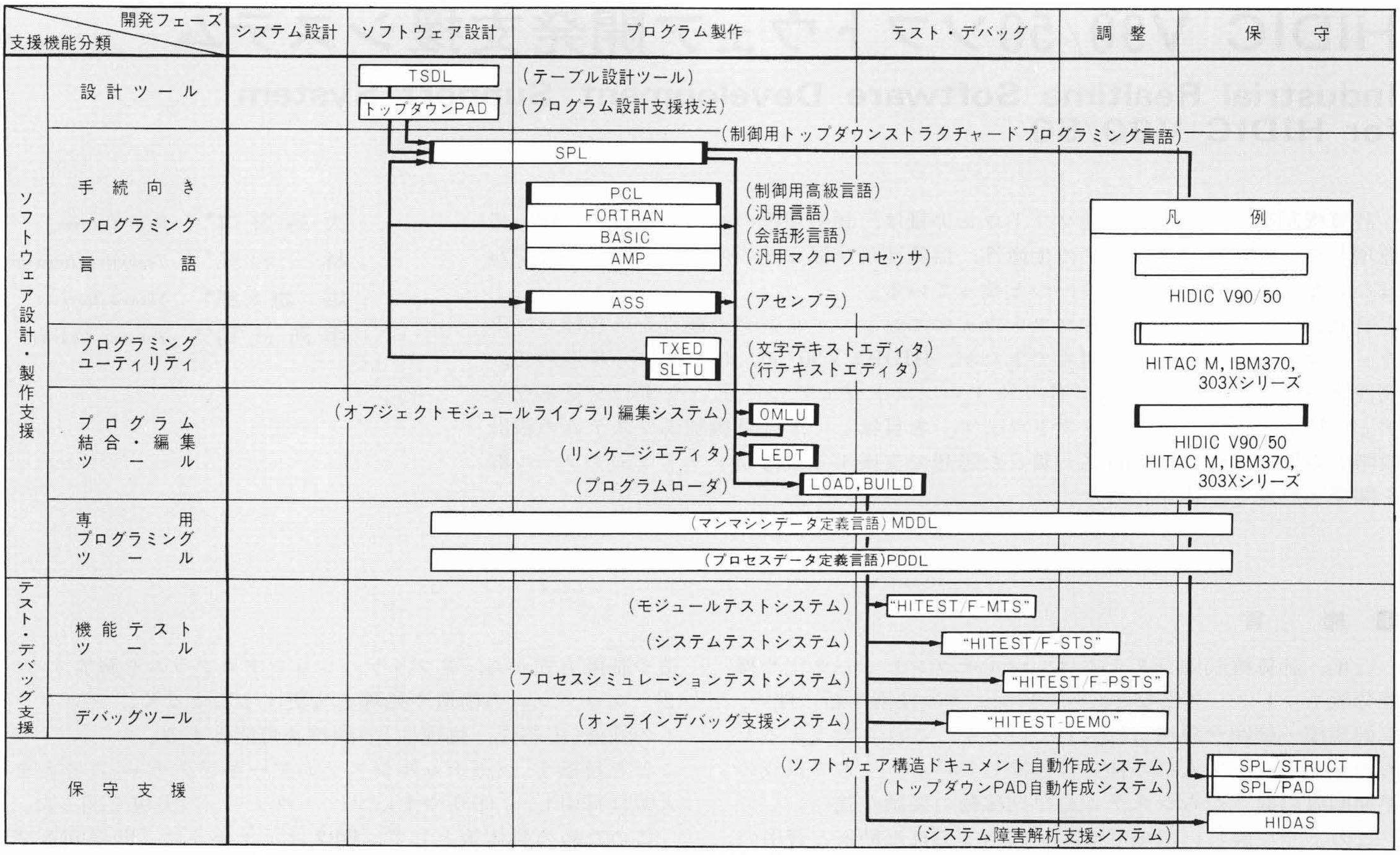
(1) 共通コマンドを用いて，だれにでも簡単に支援ツールが使いこなせる。
(2) 大量のプログラムを高速にコンパイル・テストするときは「バッチ処理」を，短いTAT(Turn Around Time)が必要な場合には「対話処理」を，それぞれ同一のコマンドインタフェースで使用できる。
(3) 作成したプログラムやデータをツール間共通ファイル経由で相互に利用し合うことが可能となるため，単に操作性が向上するだけでなく，複数ツールによる相乗効果が期待できる。

など，「使いやすいソフトウェア」の実現を可能とした。

2.3 ビジブルソフトウェアアーキテクチャ

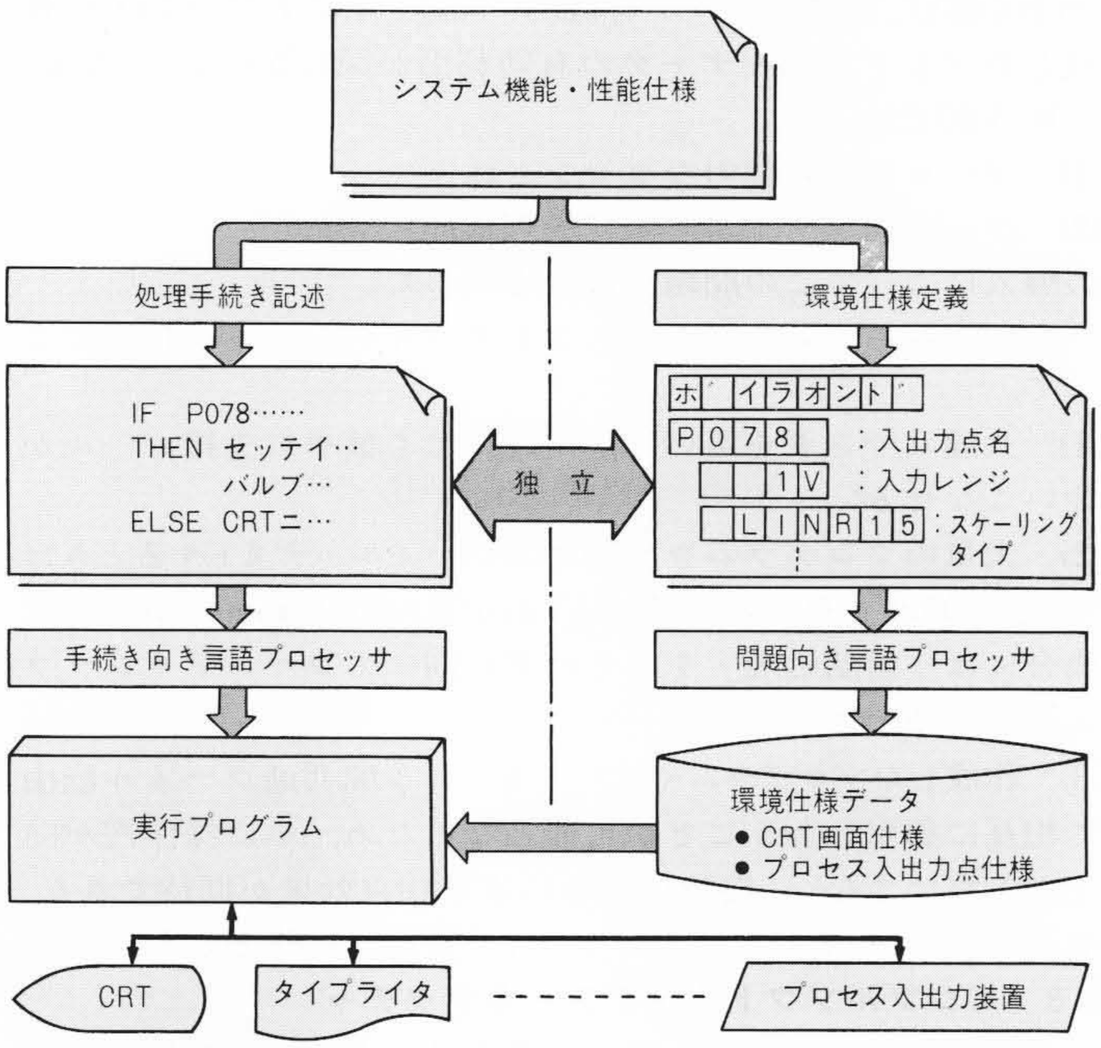
リアルタイムシステム建設の成否は，大勢の人々によって作られる膨大なソフトウェアを，予定された期間と費用でいかにうまく組み立てるかというプロジェクト管理にかかって

* 日立製作所大みか工場　** 日立製作所システム開発研究所



注：略語説明 TSDL (Table Specification Description Language)
PAD (Problem Analysis Diagram)
SPL (Software Production Language)
PCL (Process Control Language)
AMP (Advanced Macro Processor)
ASS (Assembler)
TXED (Text Editor)
SLTU (Source Library Tape Update)
OMLU (Object Module Library Update)
LEDT (Linkage Editor)
LOAD (Program Loader)
BUILD (System Builder)
MDDL (Man-machine Data Description Language)
PDDL (Process Data Description Language)
"HITEST" ("Hitachi Integrated Test System")
"HITEST/F" ("HITEST/Function")
MTS (Module Test System)
STS (System Test System)
PSTS (Process Simulation Test System)
DEMO (Determinate Evaluation, Monitor and Output System)
SPL/STRUCT (SPL/Software Structure Documentation System)
HIDAS (HIDIC Diagnosis Assistant System)

図1 HIDIC V90/50ソフトウェア開発支援システム体系 このシステムは，システム設計から保守までを一貫した形で支援できるように体系化されており，この体系に沿って各種ツールが開発・配備されている。



注：略語説明 CRT (Cathode Ray Tube)

図2 バーチャルシステムアーキテクチャによるプログラムの「処理手続き」と「環境仕様」の分離 バーチャルシステムアーキテクチャの採用により，個々のプログラムはハードウェアを意識せず論理レベルで入出力実行が行なえる。また，プログラムの処理手続きの変更なしに性能調整が可能となる。

いるといっても過言ではない。

このプロジェクト管理の難しさは「ソフトウェアそのものが目に見えない」ところにあり，この問題を解決するには，ソフトウェア開発支援システムの中に

(1) プロジェクトリーダから常にソフトウェア全体が見えるようにし，不具合点が早期に発見できるような設計・製作手法がとられていること。

(2) 前フェーズの作業結果が後フェーズの作業者に的確に伝わり，関係者による作成プログラムのレビューが容易に行なえるような支援機能が備わっていること。

が重要である。

またこうして出来上がったシステムは，設置後10年以上にわたって使用されるのが普通で，この間種々のソフトウェアの改造・拡張が行なわれる。特に，リアルタイムシステムは，タスクと呼ばれる数多くのプログラム群の有機的関連により機能しており，その保守を的確に行なうためには，更に支援システムとして，

(3) 個々のプログラムの処理内容だけではなく，むしろシステム全体の動きが正確に把握できる機能をもっていること。

が必須である。

H-V90では，ソフトウェアの設計から保守に至るまでのすべての支援ツールにこの思想を反映させ，「管理・保守しやすいソフトウェア」の実現を図ることとした。

3　ソフトウェア開発支援ツール

3.1　設計支援ツール

ソフトウェア設計作業では，システム設計で決定されたソフトウェア機能仕様書に基づき，共通テーブル及びプログラム仕様の設計を行なう。これらの作業では，設計者の思考の流れを自然な形でガイドし，かつこのソフトウェア設計の結果が次フェーズのプログラミング作業へ一対一にスムーズな形で展開してゆけることが重要である。

3.1.1　テーブル設計記述言語

計算機制御システムで被制御対象である設備環境情報は，テーブル仕様に反映される。TSDL（Table Specification Description Language：テーブル仕様記述言語）[1]では，この設計作業をFIF(Fill In the Form)シートにより標準化するとともに，テーブル仕様書の作成やテーブルに関するプログラミング作業を自動化し，テーブル設計作業の高信頼化と高効率化を可能にしている。**図3**にTSDLを用いた階層的テーブル設計と環境モジュールコードの自動生成の様子を示す。

3.1.2　プログラム設計支援技法

プログラム設計支援技法トップダウンPAD（Problem Analysis Diagram）[2]は，処理手続きであるプログラム設計の定型化を目的とした記法である。トップダウンPADを用いれば，**図4**に示すようにプログラミング言語の詳細な文法構造を知らなくても，階層的に構造化プログラムの設計ができ，設計結果が一対一の形で実プログラムに反映されるので，プログラムの生産性，信頼性及び保守性が向上する。

3.2　手続き向きプログラミング言語

手続き向きプログラミング言語としては，種々の使用局面を考慮して，**図1**に示す六つを用意している。ここで，アセ

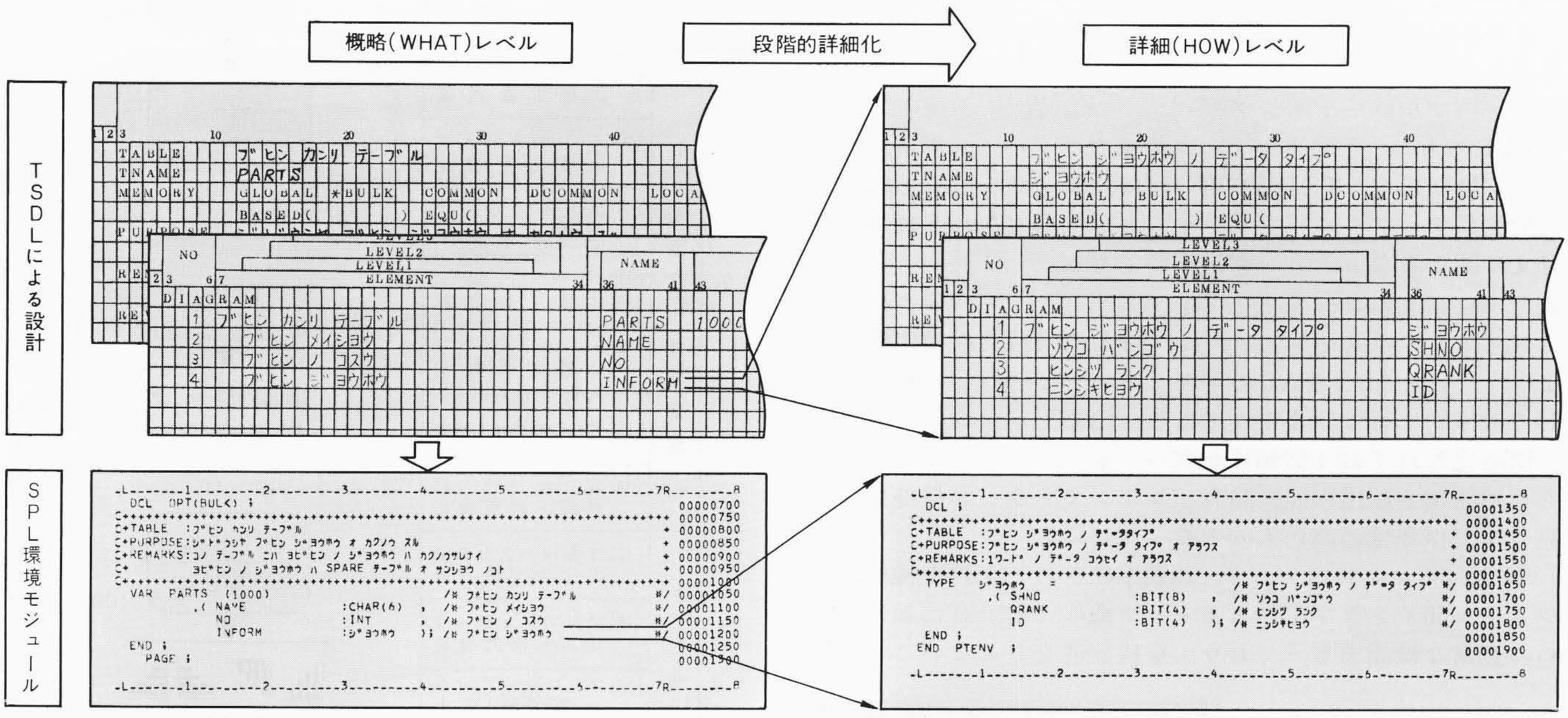


図3　TSDLによるテーブル設計とコード自動生成　　テーブル仕様はFIF(Fill In the Form)形式で記述できる。階層的なテーブル設計が可能であり，このシートからSPLの環境モジュールコードが自動生成される。

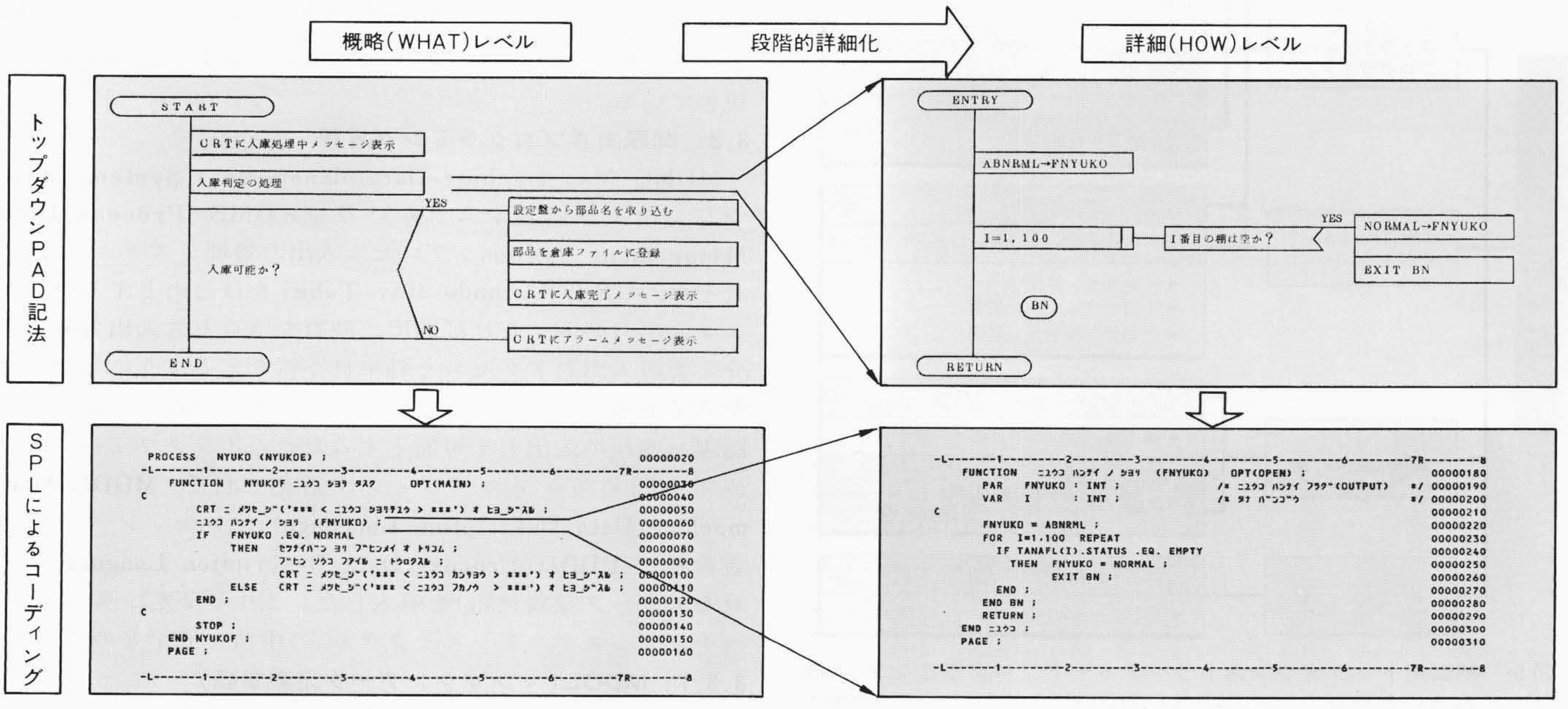


図4　トップダウンPAD記法によるプログラム設計とSPLのコーディング　　プログラムはトップダウンPAD記法により階層的に設計され，しかも，SPLプログラムは設計結果と一対一の形で作成できる。逆に，SPLプログラムから保守ドキュメントとしてのトップダウンPAD図作成も可能である。

ンブラはハードウェアに近い制御ソフトウェアや，特に性能が問題となるソフトウェアを作成する場合に，FORTRANは他機種との間でプログラムの移行性が問題となる場合に，BASICは性能は問題とならないが手軽に会話形でプログラミングを行ないたい場合に，それぞれ有用となる。制御用高級言語PCL(Process Control Language)は，FORTRANをベースに，制御用に必要な機能を付加し，高級言語ベースでアプリケーションソフトウェアを能率良く開発できるように設計，開発された言語である。

一方，制御用トップダウンストラクチャードプログラミング言語SPL(Software Production Language)[3]は，PCLのもつ各種リアルタイム制御用機能は包含しながら，更にソフトウェアの本質的高信頼性，高保守性及び標準化のしやすさといったソフトウェアエンジニアリング的な機能を大幅に強化した新しい思想の言語である。SPLのねらいと主な機能を**図5**に，コーディング例を先の**図3**及び**図4**に示す。

このSPLの機能によるプログラミング作業は，次のようになる。

(1) テーブルコーディング

SPLのもつ「手続きとデータの独立コンパイル機能」によって，先に述べたTSDLを用いれば，テーブル設計の結果が自動的にテーブル宣言プログラムとして出力されるので，宣言文に関するコーディングは不要である。

(2) プログラムコーディング

プログラムのコーディングは，SPLの段階的詳細化機能により，トップダウンPADのドキュメントと一対一に対応した形で機械的に行なうことができる。出来上がったプログラムは構造化されており信頼性が高い。また，プログラムリストも自然語ふう記述機能，自動インデンテーション機能により見やすく保守性の高いものとなっている。

SPLは更に，共通データ・共通手続きの一元管理機能，モジュール間インタフェースチェック機能など，高信頼化のための豊富な機能をもっており，重複記述をさせないこと，プログラム不良を後フェーズにもち込ませないこと，によってシステム開発全体としての大幅な信頼性・生産性の向上を実現している。

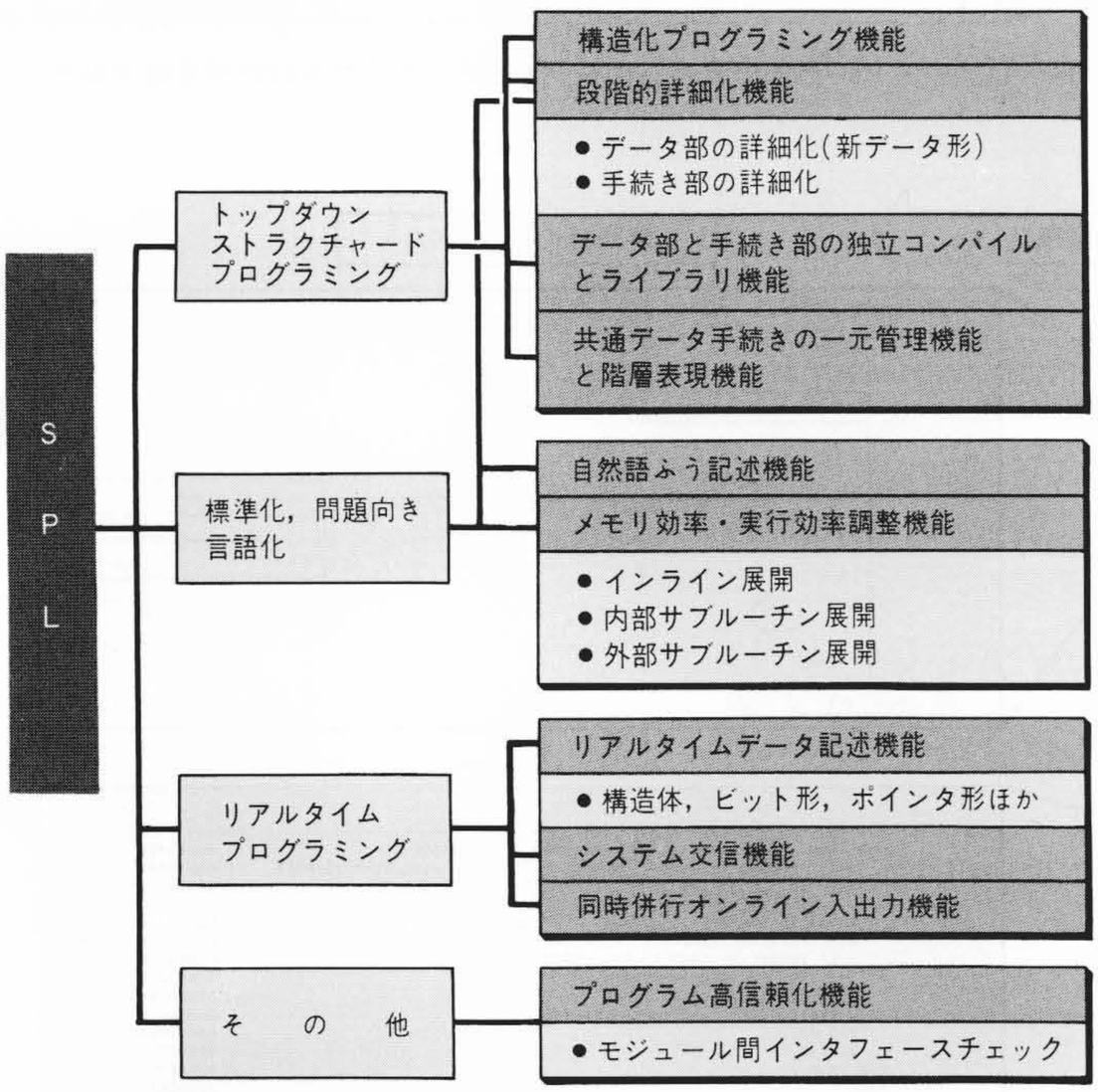
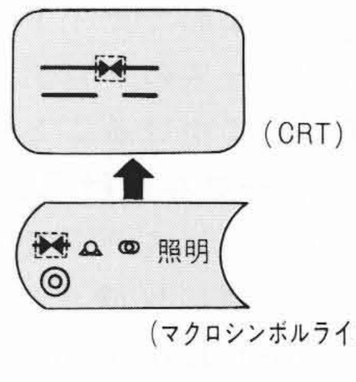
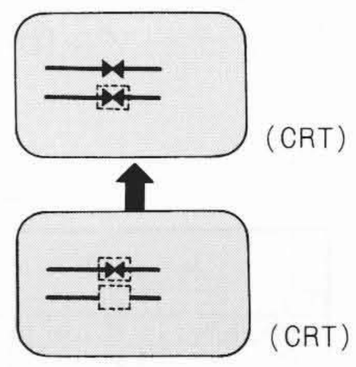
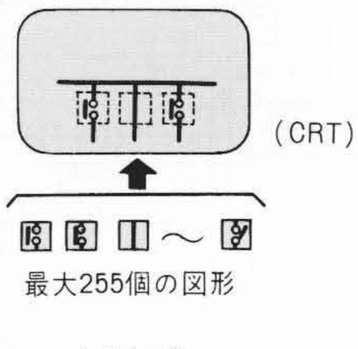
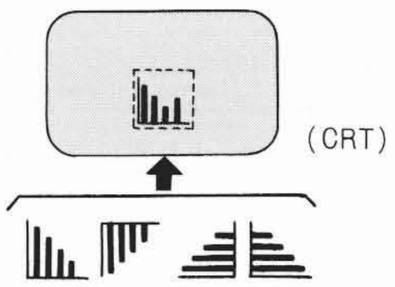


図5 制御用トップダウンストラクチャードプログラミング言語SPLのねらいと機能 SPLは段階的詳細化機能により，トップダウンプログラミング言語と標準化(問題向き)言語を表裏一体のものとして扱うことができる。

表1 マンマシンデータ定義言語MDDLの機能と具体例 これらの機能は，バッチ形・会話形共に用意されており，ユーザーは，マンマシンデータを効率良くかつ信頼性高く作成・修正できる。

No.	分類	機能	代表的な具体例
1	図形作成	(1) ドットパターン作成機能	① 図形呼出し機能(3) 
		(2) 文字埋込機能	
		(3) 図形呼出し機能	
		(4) 図形複写機能	② 図形複写機能(4) 
		(5) 図形変色機能	
		(6) 図形登録・削除機能	
		(7) BG画面登録・削除機能	
2	FG画面変化仕様定義	(8) 交替図形仕様定義機能	③ 交替図形表示(8) 
		(9) メッセージ入力仕様定義機能	
		(10) メッセージ出力仕様定義機能	
		(11) バーグラフ仕様定義機能	
		(12) トレンドグラフ仕様定義機能	
		(13) 変化仕様呼出し機能	
		(14) 変化仕様登録・削除機能	④ バーグラフ表示(11) 
3	その他	(15) 作画ソースリスト出力機能	
		(16) マンマシンデータリスト出力機能	
		(17) マンマシンデータオブジェクト入出力機能	

注：略語説明 BG(Background)，FG(Foreground)

3.3 問題向きプログラミング言語

MDMS(Man-machine Data Management System：マンマシンデータ管理システム)[4]及びPDMS(Process Data Management System：プロセス入出力管理システム)は，それぞれ，CRT(Cathode Ray Tube)をはじめとするマンマシン入出力装置，及び制御用に特有なプロセス入出力装置を介しての入出力アクセスを効率良く行なえるようにしたシステムである。これらのシステムでユーザープログラムからの論理レベルの入出力を可能とするためのインタフェースを記述する問題向きプログラミング言語として，MDDL(Man-machine Data Description Language：マンマシンデータ定義言語)，PDDL(Process Data Description Language：プロセスデータ定義言語)を導入した。これらは先に述べた「バーチャルシステムアーキテクチャ」の中核を成すものである。

3.3.1 MDDL(マンマシンデータ定義言語)

マンマシンデータにはCRT画面仕様やタイプライタ，ラインプリンタなどの帳票仕様のデータがある。従来，オンライ

ンでCRT画面や帳票を入出力する際は，CRTとそれ以外の入出力機器でもっているハードウェアの機能・性能が大きく異なるため，各々に応じた処理方式をとらねばならなかった。しかし，入出力する情報そのものは「マンマシン情報」という点で論理的には本質的な相違はない。MDDLではこの考え方をベースに，CRT画面と帳票をマンマシンデータとして同一に扱えるようにしている。

MDDLの特長は，次に述べるとおりである。

(1)　CRTへの画面入出力とタイプライタなどへの帳票入出力とに関して，共通のデータベースを使用し，設計から保守までを一貫し，かつ体系的に支援している。

(2)　複雑なCRT画面データを，バッチ的に正確かつ効率良く作成し，会話形コマンドにより柔軟に修正できる。

(3)　CRT画面をその性質から，システム構成図のように変化しない情報からなるBG(Background)画面と，プロセス状態数値のように変化する情報から成るFG(Foreground)画面とに分けて体系立てて作成できる。

MDDLの機能を**表1**に示す。

3.3.2　PDDL(プロセスデータ定義言語)

プラントから計算機へ取り込まれるプロセスデータは，そのままではレベルの低い生データであり，これをユーザープログラムで取り扱うのに便利な工学値などのプロセスイメージデータへ変換する必要がある。PDDLはPDMSのデータ生成・保守のサブシステムであり，工学変換・各種計算といった複雑・多様なプロセスデータ処理を統一的に管理するデータベースを生成・保守できるようにし，ユーザープログラムからはプロセス入出力装置や工業計器といったハードウェアを意識しない，論理レベルの入出力を行なえるように支援する問題向き言語である。

PDDLの特長は次に述べるとおりである。

(1)　データ生成・保守機能は

(a)　システム仕様定義

(b)　プロセス入出力装置構成仕様定義

(c)　入出力点仕様定義

に分かれており，システムエンジニアや，プラントエンジニアが各々の専門分野に応じて最適なデータ生成・保守を効率良く行なうことができる(**図6**)。

(2)　ホストコンピュータ側から分散配置された端末計算機でのプロセスデータ処理の定義データを，統一的に生成・保守できるので，機能分散システムでもプロセス入出力管理を信頼性高く行なえる。

(3)　FIF形式の簡単な言語であり，変換・計算仕様のデータを簡単かつ間違いなく生成・保守できる。

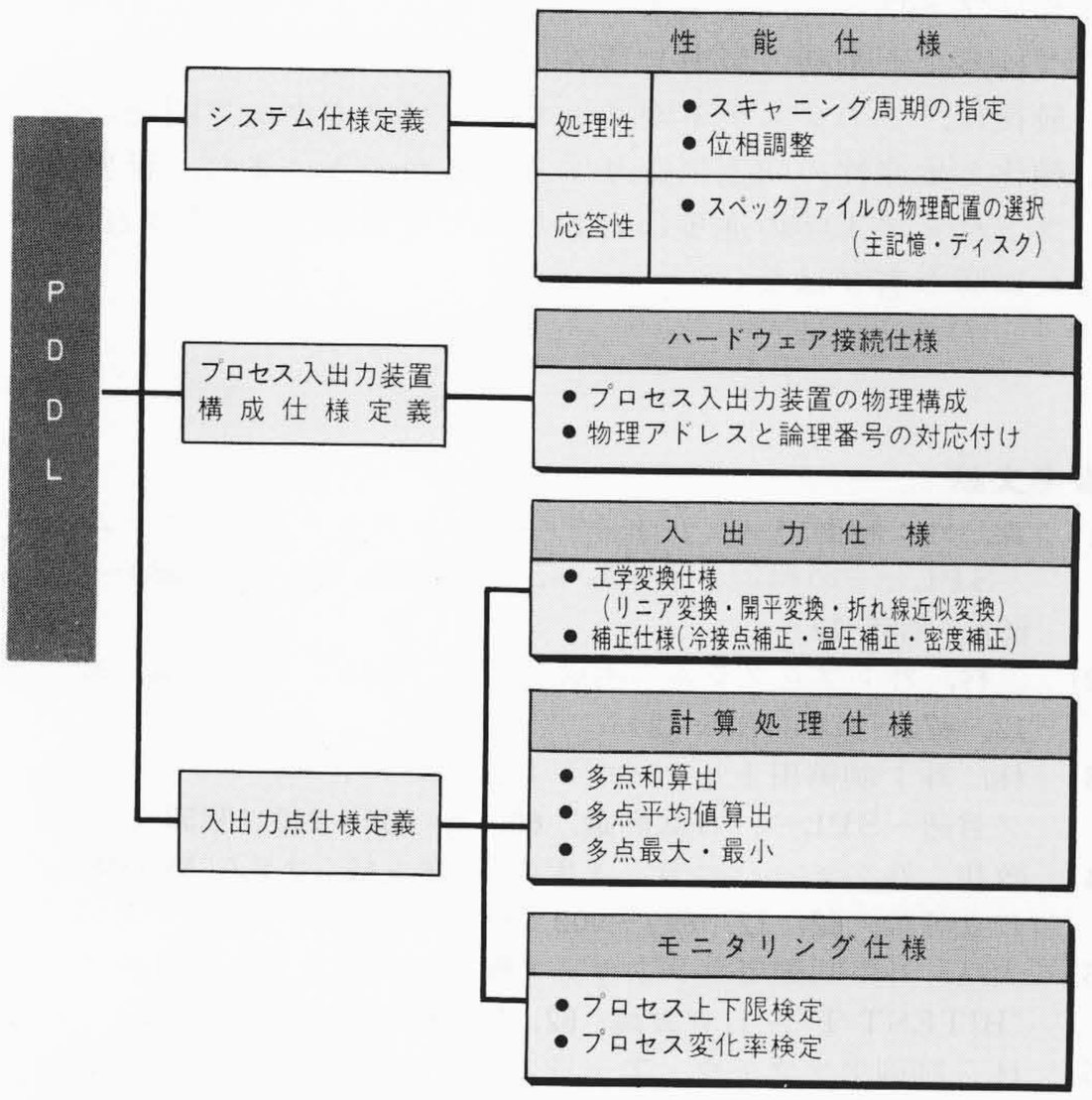


図6　PDDL(プロセスデータ定義言語)の機能　　FIF形式の簡単な言語で，プロセス入出力点ごとの処理仕様を正確かつ効率良く作成できる。また，システム仕様定義によってプロセス入出力処理の性能(処理性，応答性)を調整することもできる。

3.4　テストデバッグ支援ツール

ソフトウェア開発作業の半分以上が，機能上，性能上のバグ(不具合点)を発見するためのテスト作業と，発見されたバグの原因を追求し対策を行なうデバッグ作業とに費されている。

したがって，テストデバッグ作業の効率を上げることは，全体の生産性向上に大きく寄与するとともに，完成したソフトウェアの信頼性向上にも直結する。

このような観点から開発したのが，“HITEST”(“Hitachi Integrated Test System”：日立制御用ソフトウェア一貫テストシステム)[5]である(**図7**)。

3.4.1　機能テストツール

通常，機能テストはモジュール単体及び組合せ，タスク組合せ，総合といった過程に分けて実施される。機能テストシステム“HITEST/F”(“HITEST/Function”)のテスト支援機能部は，これらのテスト過程に対応して三つのサブシステム，すなわちMTS(Module Test System：モジュールテストシステム)，STS(System Test System：システムテス

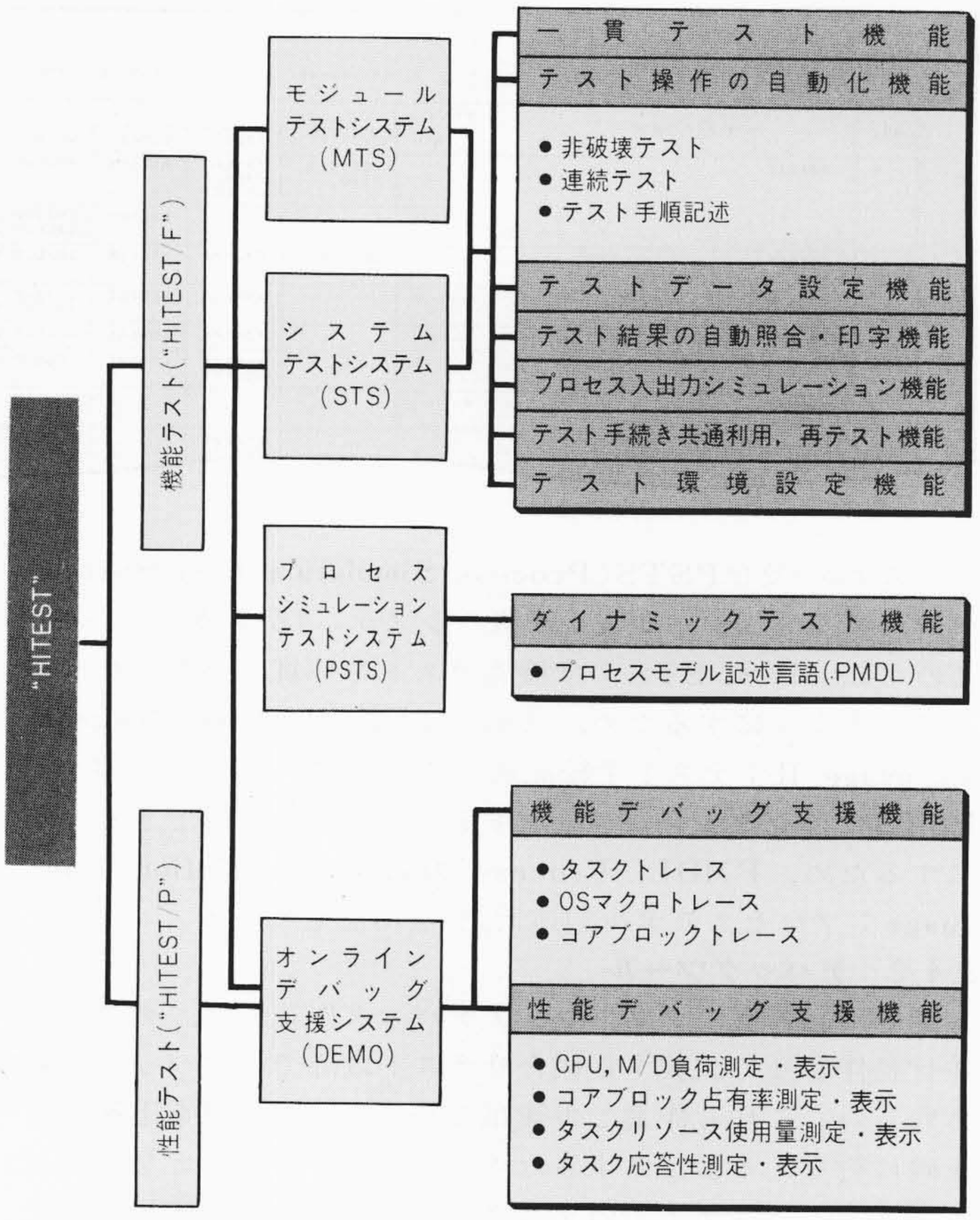


図7　制御用ソフトウェア一貫テストシステム“HITEST”の構成と機能　　“HITEST”は，性能テストと機能テストをそれぞれ一貫した形で支援するシステムで，特に，機能テスト手続きを記述するための専用言語TPLをもっている。

**********　タスク カン セイキ"ヨ カンケイズ"　**********

NO.	タスク メイ	セイキ"ヨ ノ ナカ"レ	システム コウシン フ"ン オヨビ タスク メイ						
			カ"イフ" キト"ウ	ABORT	RLEAS	QUEUE (キト"ウ ヨウイン)	SUSP	RSUM	TIMER (インシ"ケータ, キト"ウ ヨ
1	ANALGT	<——	RESTART PI/O	CRTDSP FDUMPT GFREEZ	MSGSDT	MSGSDT (FACT1)	EALARM	EALARM	
		——>			EALARM	CRTDSP (FACT1) EALARM (FACT2)	*ALL	*ALL	FDUMPT (3 , FACT
2	CRTDSP	<——	OPECON	FDUMPT	MTSCAN	ANALGT (FACT1)	ANALGT EALARM	ANALGT EALARM	FDUMPT (2 , FACT1
		——>		ANALGT	LCRTGT	LCRTGT (FACT) LCRTIT (FACT)			
3	EALARM	<——		FDUMPT	ANALGT	ANALGT (FACT2)	ANALGT	ANALGT	
		——>		MVDRUM RCOPYT	LOGGMT	LOGGMT (FACT)	*ALL	*ALL	
4	FDUMPT	<——					ANALGT EALARM	ANALGT EALARM	ANALGT (3 , FACT2)
		——>		ANALGT CRTDSP					CRTDSP (2 , FACT1)

図8　ソフトウェア構造ドキュメント自動生成システムSPL/STRUCTで生成したドキュメント例
システムを構成する全タスクに対して，それらの間の制御関係が一覧できる。また，各タスクの処理の概要が把握できる。

トシステム)及びPSTS(Process Simulation Test System: プロセスシミュレーションテストシステム)から成っている。このうち，MTS，STSには機能テストを一貫した形でサポートできるようにするため，共通のTPL/II(Test Program Language/II：テスト手続記述言語)を用意している。また，PSTSにはシミュレーションモデルを容易に作成できるようにするため，PMDL(Process Model Description Language：プロセスモデル記述言語)を用意している。

3.4.2　デバッグツール

制御用計算機のソフトウェアデバッグ作業では，タスクを並行動作させて試験する組合せテスト以降でのデバッグが最も難しい。これら作業での複雑なバグの原因究明を迅速かつ正確に行なえるようにするためのデバッグツールとして，リアルタイムモニタ下でのタスクの動きを総合的に監視しながら，各種情報を会話指定により収集し，見やすい形で出力するオンラインデバッグ支援システム“HITEST-DEMO”(“HITEST-Determinate Evaluation, Monitor and Output System”)がある。この“HITEST-DEMO”の機能には次のようなものがある。

(1)　機能デバッグ支援機能
  (a)　タスクの実行順序トレース
  (b)　タスク内のOSマクロ(入出力マクロ，タスク制御マクロなど)の発行状況のトレース
  (c)　コアブロック使用状況のトレース
(2)　性能デバッグ支援機能
  (a)　CPU(Central Processing Unit：中央処理装置)，M/D(Magnetic Disc：磁気ディスク)の負荷の測定・表示
  (b)　コアブロック占有率の測定・表示
  (c)　タスクごとのリソース使用状況，応答性の測定・表示

3.5　保守支援ツール

3.5.1　ソフトウェアドキュメント自動生成システム

システムの保守担当者は必ずしもそのシステムのソフトウェア開発担当者とは限らない。したがって，ソフトウェアの保守を信頼性高く，効率良く行なうためには，システム全体の構成とタスクごとの処理内容が100%正確に把握できる保守ドキュメントが必須である。H-V90では，そのためにSPL/STRUCT(SPL/Software Structure Documentation System：ソフトウェア構造ドキュメント自動生成システム)[1]とトップダウンPAD自動作成システムSPL/PADとを用意している。

SPL/STRUCT，SPL/PADで自動生成されるドキュメントには，次に述べるような特長がある。

(1)　タスク間，タスク～共通テーブル間のリンケージなど，システムを鳥観できる横断的保守情報が図表形式で**図8**に示すように見やすい形で得られる(SPL/STRUCT)。
(2)　タスクの処理が概略レベルから詳細レベルへと階層的にドキュメントされ，第三者に分かりやすい(SPL/PAD)。
(3)　ドキュメントは，すべてSPLソースプログラムをもとに作成されるため正確であり，ドキュメント間の矛盾がない。

3.5.2　システム障害解析支援システム

システム障害は，ソフトウェア，ハードウェアの要因が複雑に絡み合っていて原因の究明が困難なことが多い。システム障害解析支援システムHIDAS(HIDIC Diagnosis Assistant System)は，システム障害時，必要なデータを即時に収集した上で分かりやすく編集し，見やすい形でドキュメント出力することによって，障害原因の的確な究明と迅速な対策を可能とし，システム稼動率の大幅な向上を目指すものである。

4　結　　言

HIDIC V90/50で，従来のツールの適用経験を踏まえ[6]，「作りやすい」，「使いやすい」，そして「管理・保守しやすい」ソフトウェアの実現を目指して，設計から保守に至る全ソフトウェア開発プロセスに対し，一貫した思想で幅広い支援を行なうことのできる強力なツール群を，ソフトウェア開発支援システムとして開発・整備し，体系化した。

日立製作所は，今後，適用経験をフィードバックし，使いやすさのいっそうの向上を図るとともに，新しい技術を取り入れソフトウェア開発支援システムとしてより完全なものに近づけるため不断の努力を積み重ねてゆく考えである。

最後に，このシステムがユーザーのソフトウェア開発の高信頼化と生産性の向上に寄与することができ，また，世界のソフトウェア工学の進歩にいささかでも貢献できるとすれば，これに勝る喜びはない。

参考文献

1)　森，外：制御用ソフトウェア一貫プログラミングシステム—SPLとその周辺システム—，日立評論，**62**，12，889～892（昭55-12）
2)　二村，外：プログラムの木構造図面“PAD”，日立評論，**62**，12，871～874（昭55-12）
3)　林，外：制御用トップダウンストラクチャードプログラミング言語—SPL—，日立評論，**60**，3，235～240（昭53-3）
4)　政井，外：マンマシンデータ生成・保守支援システム“MDMS”，日立評論，**62**，12，899～902（昭55-12）
5)　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）
6)　林：制御用ソフトウェア一貫生産支援システムによる品質向上支援，日科技連「ソフトウェア生産における品質管理シンポジウム」（第1回），（昭56-7）
7)　中西，外：HIDIC V90/50基本制御ソフトウェア，日立評論，**63**，12，863～868（昭56-12）