U.D.C. 681.327.13.06:681.32.06.002:658.5.012

新日本製鐵株式会社での分散形プログラム開発維持支援システム“NS-OBJET”

—SEWB基本ソフトを活用したはん(汎)用ツールの開発—

Distributed System for Software Development & Maintenance NS-OBJET for Nippon Steel Corporation —Development of Software Tools Applying SEWB Kernel—

加来丈雄* Takeo Kaku
水谷正保** Masayasu Mizutani
阿部芳弘** Yoshihiro Abe
山川　寿*** Hisashi Yamakawa

システム開発生産性と品質の向上は，システム部門にとって永続する課題である。このテーマに対する回答はこれまで種々提出されてきたが，高性能ワークステーションの出現はそれらの回答の幾つかを現実化する強力な武器となり得る。新日本製鐵株式会社では，ワークステーション2050を駆使した分散形ソフトウェア開発維持支援環境を開発しており，そこにはホスト計算機によるソフトウェア開発維持支援システムの利用を通じて得られた知見と工夫が最大限盛り込まれている。本稿では，同システムの特長，機能，機構及び適用効果について概略紹介する。

1 緒　言

「経済の国際化，ソフト化」に象徴されるように，我が国の経済・産業構造は現在大きな変動に直面している。この中で企業は新しい経営環境に対応するため，計算機システムや情報通信ネットワークを活用することによって，企業競争力を強化するとともに新たな事業機会の発掘に努めている。これに伴って，経営の情報システム構築にかける期待と要求も従来になく強まっており，システム部門の企業経営に果たす役割についても再認識するに至っている。

こうした状況下で，「ニーズに適合したシステムをいかに早く，正確に構築し得るか」が，あらためてシステム部門の経営に対する貢献度を測るものさしとしてクローズアップされている。新日本製鐵株式会社では，システム開発生産性と品質向上のテーマに長年取り組んでおり，ソフトウェアの再利用をパラダイムとするTSS(Time Sharing System)環境下での開発維持支援システムの適用も意欲的に進めてきた。今回，新日本製鐵株式会社は，更にいっそうの開発生産性と品質の向上を実現することを期して，分散形開発支援環境の開発に着手した。このシステムは，ソフトウェア工学の近年の成果を取り入れ，高機能ワークステーションを徹底的に活用することによって，(1)図式表現と日本語表記によるソフトウェアの可読性向上，(2)プログラム仕様書作成と一体化したプログラム製作，(3)論理レベルでのプログラム・テストの実現，(4)標準的なシステム開発工程・手順を反映した操作環境の提供，を図ったものである。同システムは，分散化の実現によりホスト計算機のボトルネック化というTSSにまつわる問題も併せて克服している。この結果，開発生産性と品質の向上という所期の目標は十分実現可能であると確信している。

2 NS-OBJETの概要

2.1 NS-OBJET開発の背景

近年，新日本製鐵株式会社ではシステム開発維持効率を向上させるため，標準化活動や個別支援ツールの開発及びTSSを活用したプログラム開発維持支援システムを開発・適用して，相当の効果を挙げている[1]。しかし，このTSSを主体とした方式は次のような問題があり，抜本的な改善が必要となっている。

(1) 一定の応答が得られにくい。

(2) TSS端末の増強がホスト計算機の増強につながりやすい。

同時に，次のような問題もある。

(1) 設計ドキュメントとプログラム・ソースコードのかい(乖)離が著しく，プログラムの維持効率が悪い。

(2) COBOL，PL/Iなどのソースコードによる製作作業のため長期間の教育が必要なうえ，細かい言語仕様の習得を要求するため，生産性が上がらず保守性も悪い。

(3) ソフトウェアの再利用技術が一般化していない。

(4) 機械化率が低く大半が手作業である。

2.2 NS-OBJETの開発方針

NS-OBJETの目標は，システム開発の生産性・保守性の効率向上及びソフトウェアの品質向上である。この目標を達成するためには，規格化・標準化に立脚したソフトウェアの自動生産を志向する「ソフトウェア一貫生産管理システムの構築」

* 新日本製鐵株式会社エレクトロニクス・情報通信事業本部　** 新日本製鐵株式会社大分製鐵所システム部
*** 日立製作所大森ソフトウェア工場

が不可欠である。この観点から次のような開発方針とした。

(1) プログラミング支援機能の先行

設計フェーズについては，各種の開発方法論が提案されているが，いずれが最適解であるか直ちに結論を下しにくい状態である。これに対しプログラミングフェーズについては，機械化がフィージブルであり，適用効果が明確である。

(2) ワークステーションへの機能分散

ホスト計算機はライブラリ管理と最終的なテストを分担し，プログラミング支援機能は大部分をワークステーションが分担する構造とする。また，ワークステーション単独でも利用可能な機能・構造とする。

(3) ドキュメント主体のプログラミングへの転換

ソースコード中心のプログラミングに伴う生産性・保守性の悪さ，品質の悪さを克服すべく，可読性に優れたプログラム仕様書を作業対象として，プログラム作成・保守作業を進める方式に改める。

2.3 NS-OBJETの特長

NS-OBJETの機能面から見た特徴は以下のとおりである。

(1) ソフトウェアのビジュアル化とドキュメント

ソフトウェアの論理を階層的に表現し，かつ図形・日本語などで記述することにより，ソフトウェアの可読性を飛躍的に高めている。論理図などの設計・製作情報はワークステーションで作成され，また，ワークステーションで清書・印刷して本の形にして入手できる。

(2) 再利用支援機能とデータ移動命令の自動生成

プログラムパターン，プログラム部品の登録，及びプログラムパターン，プログラム部品によるプログラム合成機能を準備している。また，データの移動命令のような同一形式の命令が反復する場合に自動生成することが可能である。

(3) プログラムの自動生成

プログラム仕様(使用するプログラムパターン，プログラム部品，データ定義，論理図など)を記述することにより，COBOL，PL/I，Cなどによるソースプログラムを自動生成する。機構面から見た特徴としては，次の二つが挙げられる。

(4) ワークステーションによるプログラム開発・維持

図1に示すように，ホスト計算機に全面的に依存した従来のプログラム開発・維持方式を改め，開発・維持作業の大半をワークステーション上で行えるようにした。

(5) ワークステーションの機能を利用した優れた操作性

ワークステーションが持つマウス，アイコン，マルチウインドウなどの機能を利用して，的確に利用者を誘導・支援する機能を備え，だれでも簡単に操作ができる。

2.4 NS-OBJETの構造

日立製作所の製品であるSEWB(Software Engineering Workbench)では，各種ソフトウェア開発支援ツールを統合する方式としてSEWB基本部を中核に据えたカーネル化構造形方式を採用し，支援ツール間のインタフェースは論理レベルで統一されている。

NS-OBJETは基本的に，このSEWBの方式を踏襲しており，SEWB基本部から見るとその上位機能に位置づけられる。NS-OBJETはSEWB基本部の提供する論理図作成機能，論理図テスト機能などを，標準の関数インタフェースを通じて利用することによって，低位の図形処理，ファイル・アクセス処理などを直接意識することなく，各種のプログラム作成・維持支援機能を実現している。同時に，主として論理図以外のドキュメント作成機能については，SEWB基本部のサポート範囲を越えるため，NS-OBJETとHI-UXとの直接のインタフェースによって，マルチウインドウ，マルチタスクをはじめとしたハードウェア及びソフトウェア機能の活用を図っている。このように，NS-OBJETの構造は，SEWB基本部とインタフェースを持つ部分と，HI-UXと直接インタフェースを持つ部分に分けられるが，その全体構造を**図2**に示す。

3 NS-OBJETの機能

図3にNS-OBJETの機能構成を，**図4**にNS-OBJETの機能階層構造図を示す。以下，本章ではNS-OBJETの機能について説明するが，NS-OBJETでは**図5**に示すメニュー画面で実行したい作業を示す図形シンボルをマウスで選択すること

在来方式(ホスト依存形)

	工程名	担当部門		マシン	
		SE	Prog	ホスト	ワークステーション
1	機能設計	○	—	▲	×
2	機構設計	○	—	▲	×
3	プログラム仕様書作成	○	—	×	×
4	製作	—	○	▲	×
5	一貫テスト	○	—	▲	×
6	保守	○	—	▲	×

ワークステーションに機能移管

分散方式(ワークステーション+ホスト)

工程名	担当部門		マシン	
	SE	JSE	ホスト	ワークステーション
機能設計	○	—	▲	▲
機構設計	○	△	▲	▲
プログラム仕様書作成(含む製作)	△	○	×	▲
一貫テスト	○		▲	×
保守	○		▲	×

注：略語説明など
○(主として担当)
△(従として担当)
▲(利用)
×(利用しない)
—(関係せず)
SE(システムエンジニア)
JSE(下級システムエンジニア)
Prog(プログラム・コーダ)
ホスト(ホスト計算機)

図1 システム開発工程と分担 NS-OBJETはホストコンピュータ非依存形プログラム開発方式である。また，ソースプログラムコーダは不要になる。

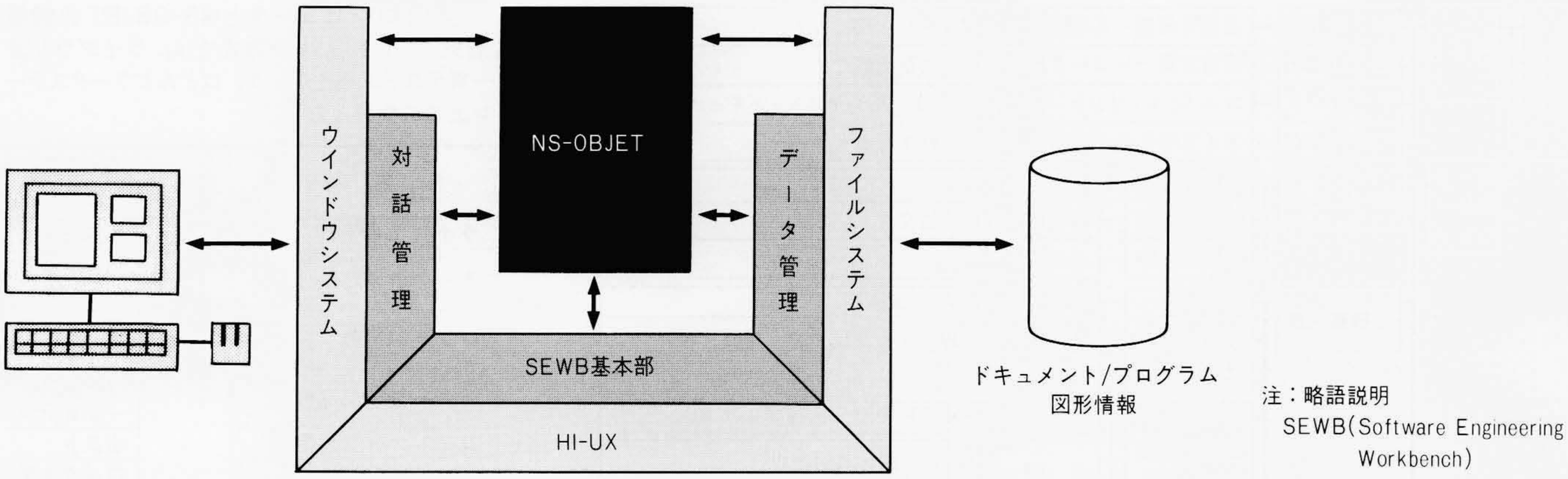


図2　NS-OBJETの全体構造　NS-OBJETは，SEWB基本部とインタフェースを持つ部分と，HI-UXと直接インタフェースを持つ部分とで構成される。

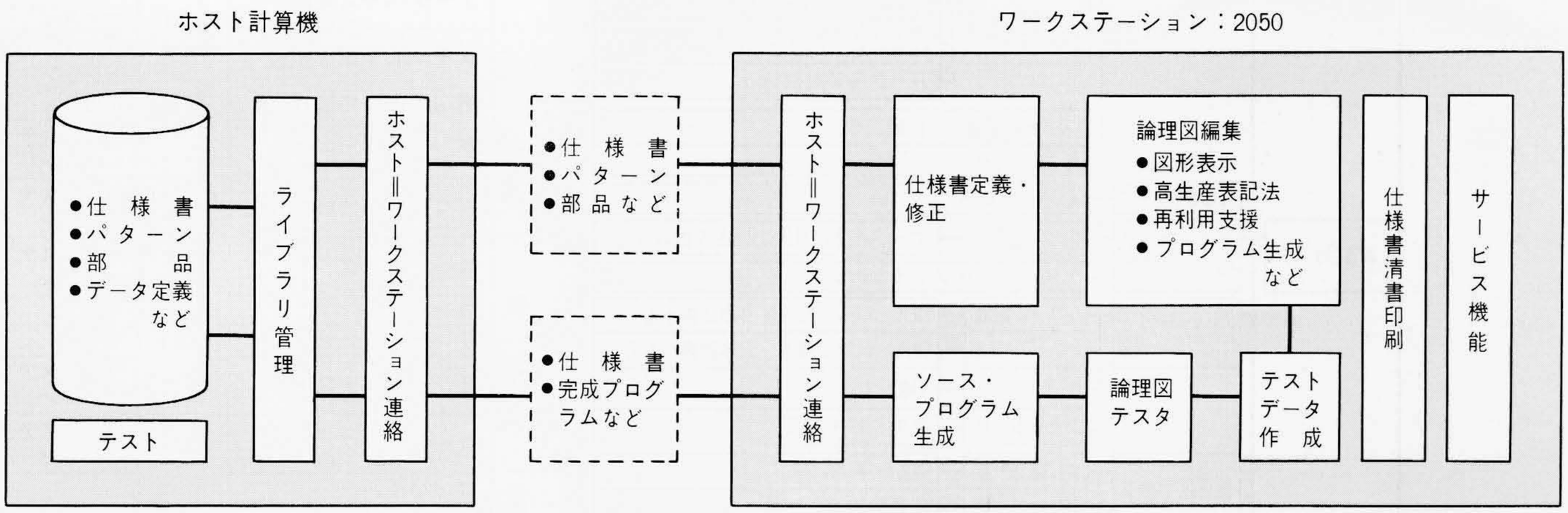


図3　NS-OBJET機能構成　ホスト計算機は，ライブラリ管理・テストを分担し，ワークステーションはプログラム開発作業を受け持つ。

により，当該作業の開始に移れる。

3.1　ホスト計算機との機能分担

表1にNS-OBJETとホスト計算機との機能分散形態を示す。この表から明らかなように，ホスト計算機は膨大な量のプログラム仕様書などのライブラリを管理する機能と，最終的なテスト機能を分担し，プログラム製作作業の主体はワークステーションが受け持つ。

3.2　ホスト計算機とNS-OBJETの連絡

ホスト計算機とNS-OBJETの結合は，NS-OBJETと種々のホスト計算機との接続を可能とするため，ファイル転送レベルで実現している。具体的な転送対象ライブラリを**表2**に示す。ホスト計算機でのライブラリからの抽出及び格納処理を行うユーティリティもNS-OBJETに付帯して用意されている。

3.3　プログラム仕様定義

NS-OBJETは設計仕様書を起点にプログラムを自動生成する方式を採用している。このため，プログラムの主要要素(論理図・データ定義・プログラム間連絡パラメータなど)を各々文書化して登録する。各要素を定義登録するための画面を**図6**に，またその概要を**表3**に示す。

プログラム仕様定義の中で重要な位置を占める論理図定義機能の仕様は以下のとおりである。

(1)　論理図記号と論理図内表記法

論理図記号は10種類である。この論理図内への表記は，次のような特徴を持つ(**図7**)。

(a)　自動生成対象言語(COBOL，PL/I，C)から独立している。

(b)　算術表現，日本語表記が主体となっており，ソフトウェアの可読性を高めている。

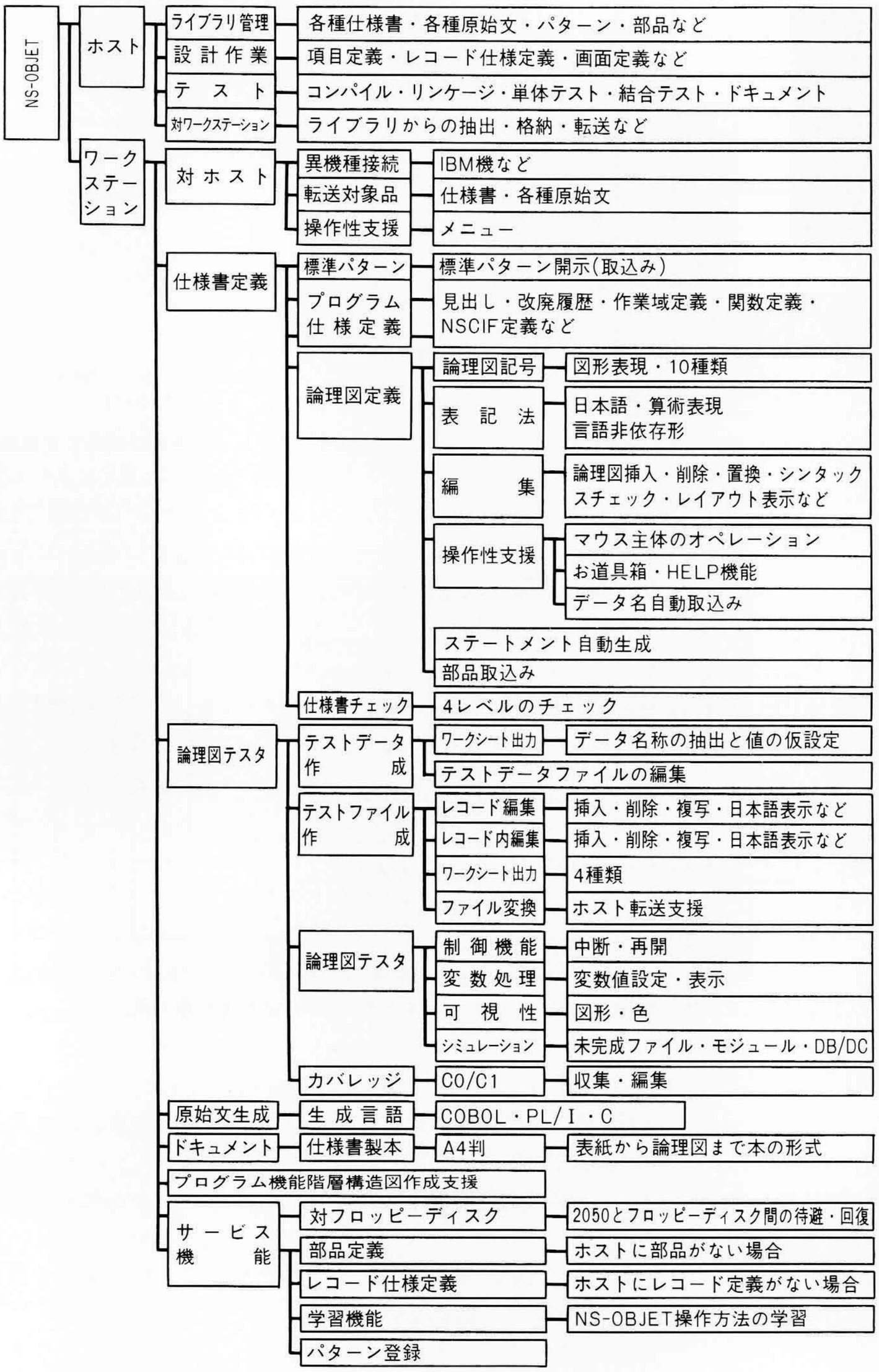
(c)　日本語表記やデータ名の記述は，メニューからマウスで選択する方式を持っており記述量が少ない(**図8**)。

(2)　ソフトウェア再利用支援機能

プログラム仕様書見出し定義画面で，事前に登録してあるプログラム・パターンを指定すると，当該パターンがプログラム仕様書の原型として取り込まれる。NS-OBJETの利用者は表示される各定義画面にプログラム・パターンの修正情報を入力することにより，プログラム仕様書を完成させる。また，プログラム・パターンと同様に蓄積されているプログラム部品を選択し，論理図中に展開することによって既存ソフトウェア資産の活用も図れる。

(3)　データ移動命令の自動生成

プログラム中では，ファイルからファイル，あるいはファイルから画面・帳票などへのデータの移動が頻繁に行われ，同一パターンのデータ移動命令を反復記述する場合が発生する。こうした一連のデータ移動命令をレコード仕様書に基づ



注：略語説明 NSCIF(Nippon Steel Corporation Interface)
DB/DC(Data Base/Data Communication)

図4 NS-OBJET機能階層構造 対ホスト機能として異機種接続もサポートしている。

表1 ホストコンピュータとNS-OBJETの機能分散形態 NS-OBJET方式では，ライブラリ管理，一貫テスト工程を除いて，ほとんどワークステーション上での作業となる。

工程	作業項目	在来形	NS-OBJET方式
	ライブラリ管理	ホスト	ホスト
設計作業	データ項目定義	ホスト	ワークステーション*
	レコード定義		ワークステーション
	パターン・部品作成		ワークステーション
	コードテーブル定義		ホスト
	画面・帳表定義(端末)		ワークステーション*
	帳表定義(中央)		ワークステーション*
	データベース定義		ワークステーション*
	ステートメント生成		ワークステーション*
	用語辞書維持管理		不要
	デシジョンテーブル		ワークステーション*
	プログラム仕様書作成	紙鉛筆	ワークステーション
製作	コーディング	ホスト	ワークステーション
	コンパイル		不要
	リンケージ		不要
	モジュールテスト		ワークステーション
一貫テスト	コーディング	ホスト	ワークステーション
	コンパイル		ホスト
	リンケージ		
	単体テスト		
	結合テスト		
	運用テスト		
維持	プログラム仕様書	紙・鉛筆	ワークステーション
	コーディング	ホスト	ワークステーション
	コンパイル		ホスト
	リンケージ		ホスト
	ドキュメント		——
	日本語ソースリスト		不要
	プログラム関連図		ワークステーション
	PAD図		ワークステーション
	日本語コンパイルリスト		ホスト

注：*は今後ワークステーション化を検討する機能を示す。

表2 転送対象ライブラリ NS-OBJETはプログラム仕様書で維持作業を行うことを前提としているため，ホスト計算機からワークステーションへの原始文のダウンロード機能はない。

転送対象ライブラリ	ホスト ⇒ ワークステーション	ワークステーション ⇒ ホスト
レコード仕様書	◎	◎
プログラム・パターン	◎	◎
プログラム部品	◎	◎
原始文	×	◎
プログラム仕様書	◎	◎

注：◎(対象とする)，×(対象としない)

表3 プログラム仕様定義機能一覧(PL/I例) プログラム仕様書定義画面の種類とその定義情報の概要を示す。

機能名称	機能概要
見出し定義	プログラムのタイトル・機能概要などの見出し情報を記述
改廃履歴	プログラム仕様書の改廃履歴情報を記述
手続き間連絡データ定義	モジュール間のパラメータ情報を記述
レコード仕様書定義	使用するレコード仕様書名を記述
作業域定義	使用するワークエリアを記述
外部参照モジュール定義	外部参照するモジュール名を記述
ファイル定義	使用するファイル情報を記述
関数定義	使用する関数を記述
NSCIF定義	(社内ユース用)
&&COPY定義	(社内ユース用)
論理図定義	PAD図で論理を記述

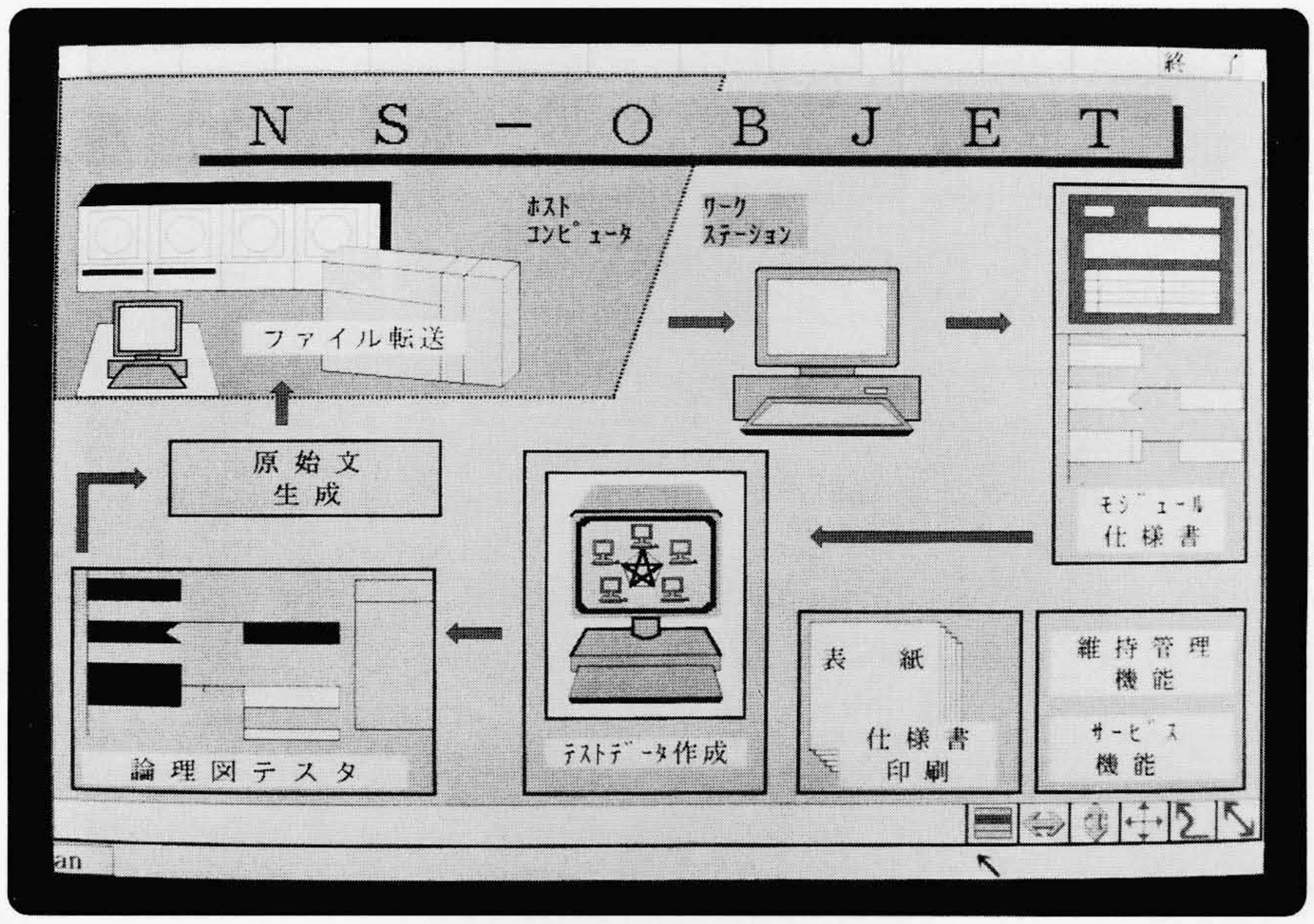


図5　NS-OBJETのメニュー画面　図柄はアイコン表示となっている。例えばテストを行う場合は，論理図テスタの図柄をマウスでピックする。

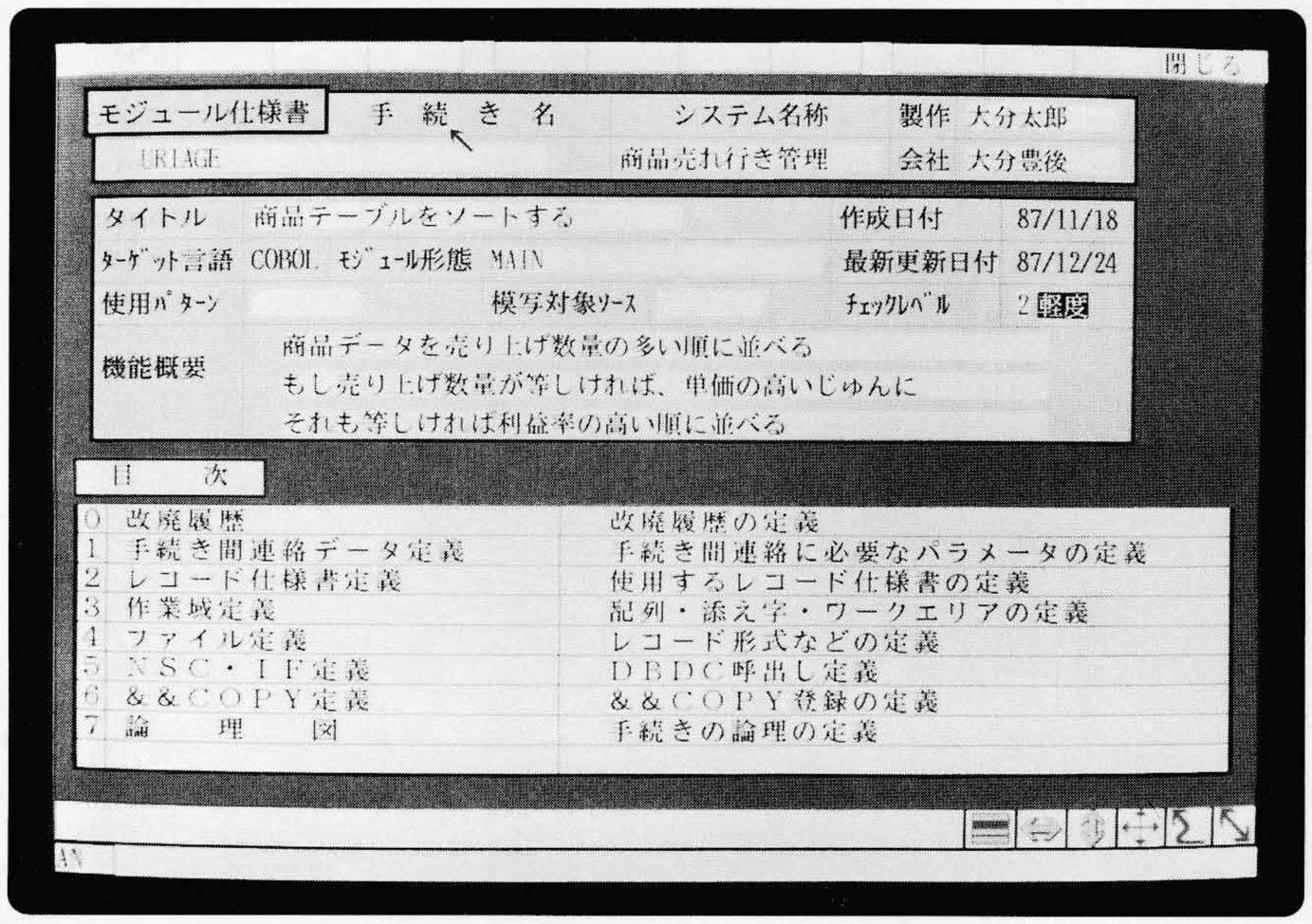


図6　仕様書定義画面
見出し定義と目次の例を示す。

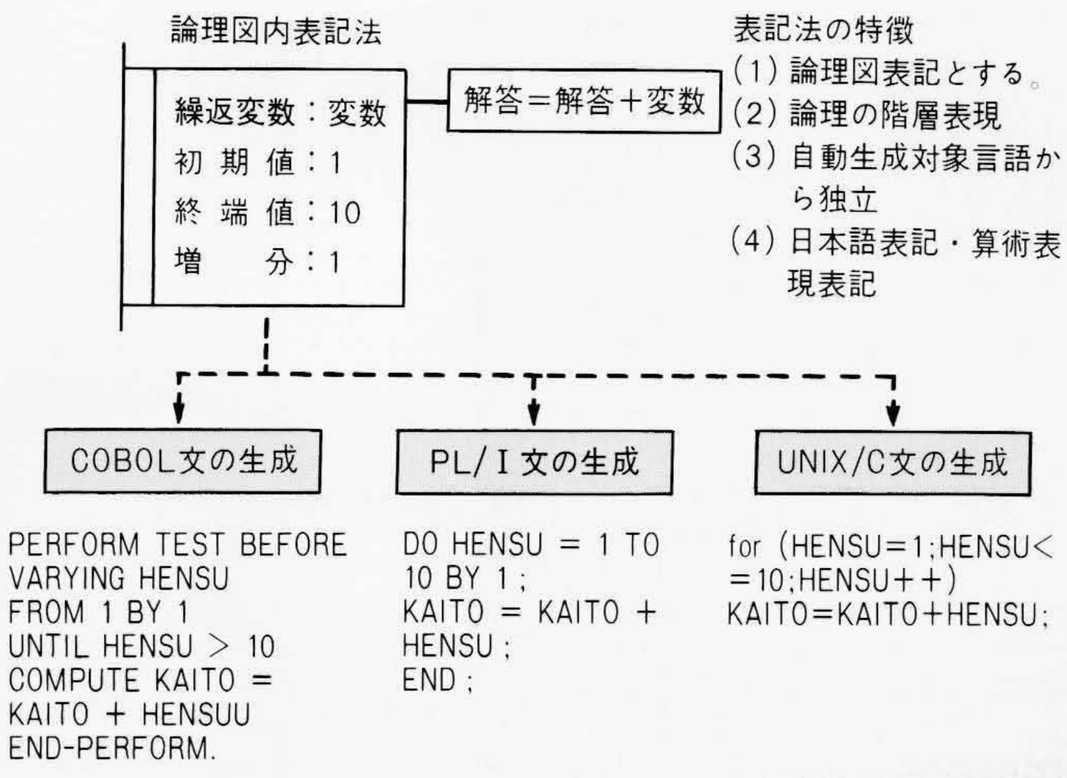


図7　論理図内表記法の例　1から10までの足し算を行う例を示す。

いて自動的に生成することにより，プログラミング作業の負荷軽減が実現する。

自動生成された命令の編集処理機能も合わせ持っている（**図9，10**）。

3.4　仕様書チェック機能

3.3節で述べたプログラム・モジュール定義情報に記述上のミス，矛盾がないかどうかをチェックする機能であり，**表4**に示す4種類のチェックレベルがある。チェックレベル3を指定すると，最終的に得られる原始文を翻訳したとき文法的にエラーが発生しないほどの厳しいチェックを行う。

3.5　プログラム・テスト機能

プログラム仕様書定義の論理図定義画面で定義した情報をもとに，プログラム・テストを行う。このテスト機能は，大別して以下に示す三つの機能から成る。

(1)　テストデータ作成機能

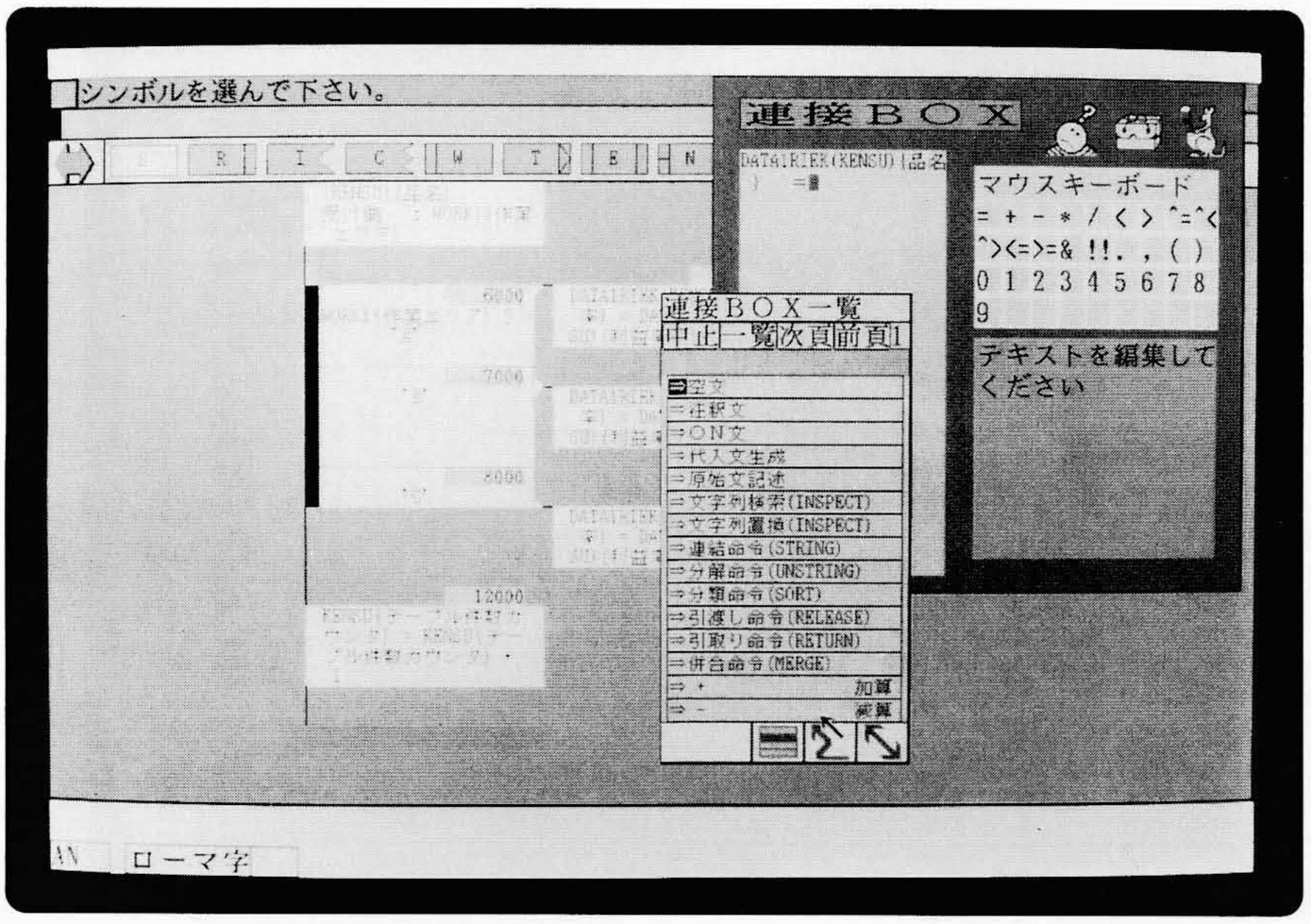


図8 データ名称の自動取込み画面 日本語表記やデータ名称の記述は，メニュー画面からマウス指定によって自動的に取り込むことができる。

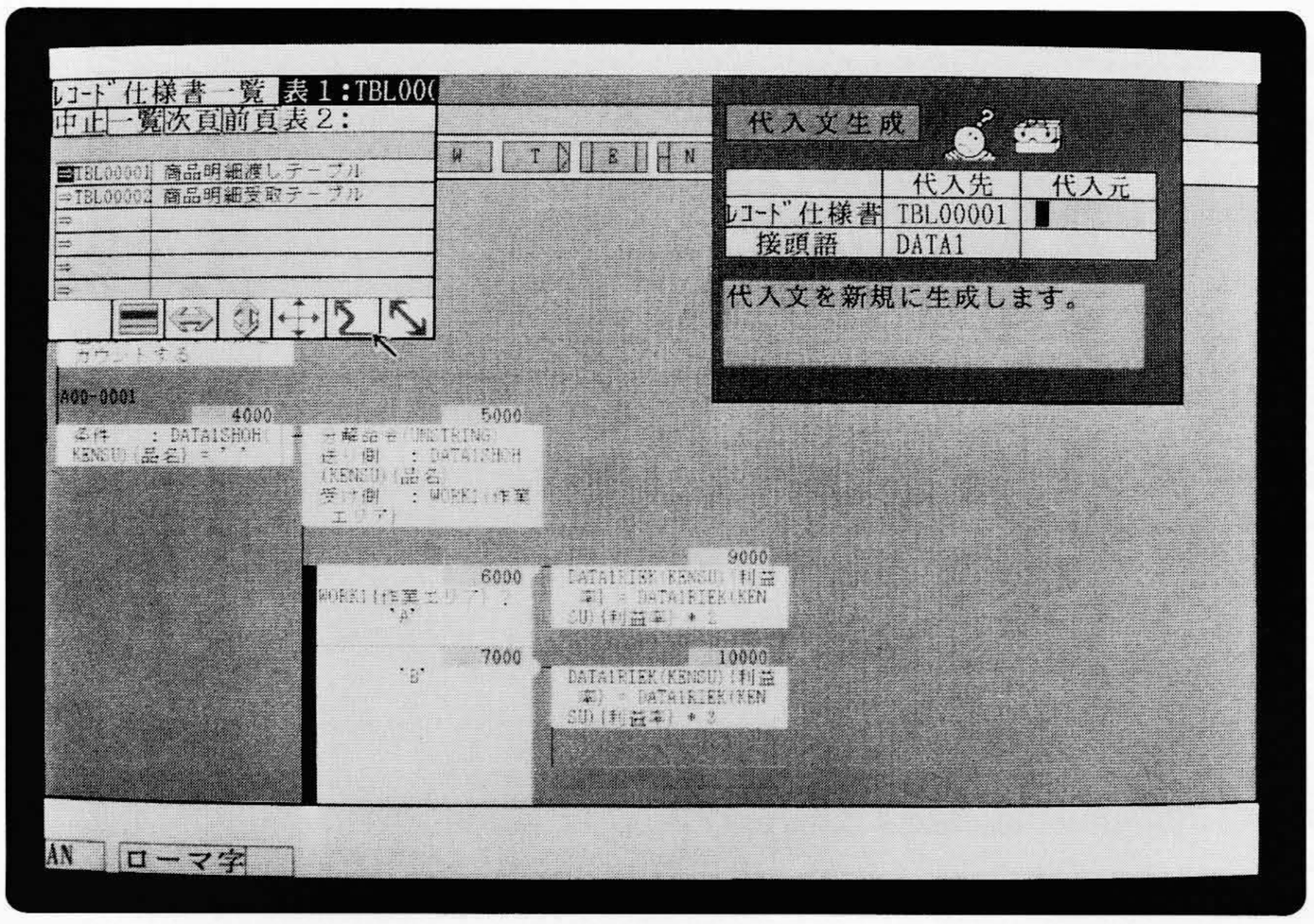


図9 データ移動命令の自動生成画面 データ移動処理命令を，レコード仕様書をキーとして自動的に作り出すことができる。

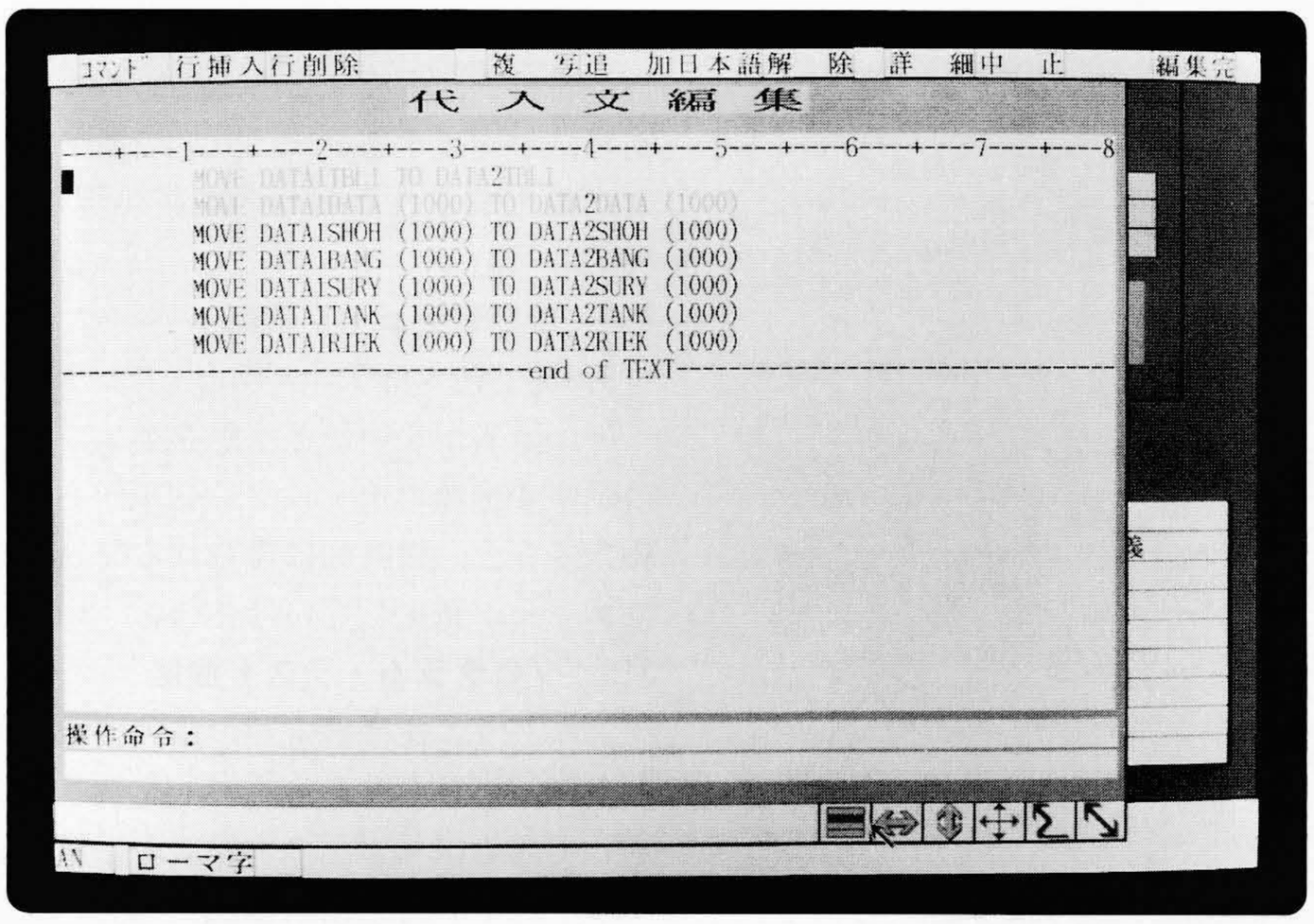


図10 データ移動命令の自動生成結果画面 属性，長さなど不正の命令は赤色表示される。

表4　構文チェックレベル　レベル3を選択すると，ホスト計算機で翻訳時，コンパイルエラーが発生しないほどの厳しいチェックを行う。

レベル	内　　容
0	仕様書チェックをしない。
1	論理図記号単位にシンタックスチェックを行う。
2	分岐先などの整合性チェックを行う。
3	エリア定義との整合性までチェックを行う。

論理図に表記されたデータ名称からその属性・長さ・用途を解析して，設定すべきデータ名称の自動抽出と設定データの仮設定を行う。これによって，人手で行っていたテストデータ作成作業の品質と効率が著しく向上する。

(2)　テストファイル作成機能

プログラム中で使用するファイル情報が比較的容易に設定・更新できる機能であり，次の四つの機能をうまく組み合わせて使うことにより，テスト環境設定負荷が大幅に削減され，同時にテスト結果の検証作業も容易に行える。なお，本機能はNS-OBJETから独立しても機能するはん(汎)用プログラムとなっている。

(a)　データ名称によるファイル編集

データ名称(英・数字，日本語)をキーとして，データ属性(けた数，データタイプ)に従ってデータの設定・表示が可能である。

(b)　ワークシート編集出力

テストデータを設定するためのワークシート出力や設定済みのデータを用途に応じた4種類の編集フォーマットでワークステーションのプリンタに編集出力する。これによって，テストデータ記述作業やファイルの検証作業が容易になる。

(c)　ファイル・フォーマット変換

ホスト計算機とファイル転送を行うとき，転送サイズに制限がある。この問題を解決するため，指定ファイルを任意のサイズに分割する，あるいは任意のサイズで連結する機能である。これによって，どのようなファイルでも転送できる。

(3)　論理図テスタ

ワークステーション上で定義した論理図によるテストであり，以下に示す特徴がある。

(a)　ビジュアルテスト

(i)　論理図に走行ルートを図示

(ii)　論理図内走行回数を自動表示

(iii)　任意のデータ名称のデータ内容を表示

(b)　任意のデータ名称にデータ内容を設定する。

(c)　中断点の設定とテスト再開始処理

(d)　シミュレーション

(i)　未完成プログラムのシミュレーション

(ii)　DB/DCシミュレーション

(iii)　ファイル・アクセスのシミュレーション

(e)　テストカバレッジ

(i)　プログラム内命令走行ルートの収集と図示

3.6　原始文生成機能

プログラム仕様書見出し定義に記述された自動生成対象言語の指定に従い，プログラム仕様書定義情報などからCOBOL，PL/I，Cの原始文を自動的に生成する。生成された原始文はホスト計算機に転送のうえ翻訳され，ターゲット計算機のオブジェクト コードによるテストが行われる(**図11**)。

3.7　仕様書清書印刷機能

NS-OBJETを用いて記述したプログラム定義情報を清書印刷する機能である。本機能を使用する場合は，ワークステーションにLBP(レーザビームプリンタ)を接続する必要がある。

3.8　サービス機能

ホスト計算機上から独立してワークステーションだけでNS-OBJETを利用したい場合や，NS-OBJETの運用性向上を目的として**表5**に示すサービス機能を用意している。

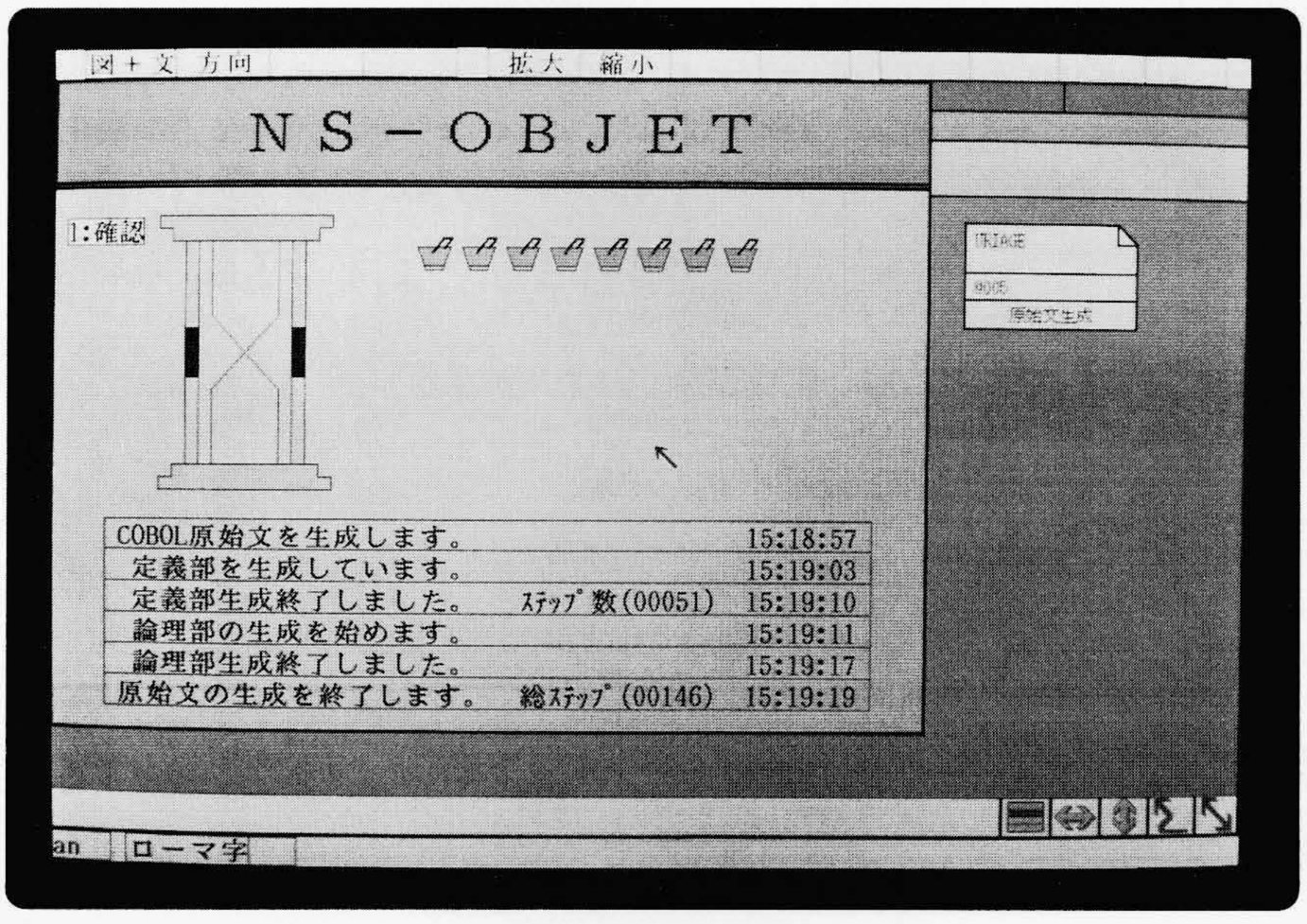


図11　原始文自動生成画面
原始文生成過程は，砂時計及びバケツの図柄で時々刻々表示される。

表5 サービス機能一覧表 ホスト計算機に開発維持支援システムを持たない箇所でも，NS-OBJETが利用できるように配慮している。

機能名称	機能概要
部品定義機能	プログラムに組み入れられる部品の定義・編集
レコード仕様書定義機能	データ名称(英語・日本語・属性・けた数など)の定義・編集
NS-OBJET↔フロッピーディスク機能	NS-OBJETで作成した各種情報をフロッピーディスクに退避したり，フロッピーディスクから回復する機能
パターン登録機能	パターン名称の登録

4 NS-OBJETによる効果

4.1 稼動環境と適用分野

NS-OBJETは，ワークステーション2050の上で動作しホスト計算機と接続するオンライン構成とスタンドアロン構成のどちらでも利用可能である。NS-OBJETは，計算機の種別，規模，バッチ/オンラインなどの利用形態を問わず大形計算機からワークステーションに至るプログラムの開発に適用できる。

4.2 適用効果

NS-OBJETを使用することで以下に示す効果が見込まれる。

(1) 生産性向上・保守性向上と品質向上

次に示す効果により，2～3倍の生産性向上が期待できる。

(a) ホスト計算機に影響されない開発環境

(b) プログラム仕様書作成とソースコード記述の二重作業の解消

(c) 仕様書とプログラムのかい離防止

(d) 図形，算術記述，日本語表記を主体とする分かりやすい論理図表記法と，論理のビジュアル化の実現

(e) 従来の仕様書作成及びソースコード記述に比べ圧倒的に少ない記述量

(f) テストデータやテストファイルの設定工数削減

(g) 論理図テスタによるデバッグ効率向上と品質の改善

(h) 文書化の容易な実現

(2) ホスト計算機の負荷軽減

ワークステーションに機能を移すことによって，次の効果が見込まれる。

(a) ホスト計算機リソースの余力創出

(b) ホスト計算機リソースの膨張抑止

(3) システム開発工期の短縮

ホスト計算機の能力に関係なく開発ピークに対応してワークステーションが設置でき，システム開発要員の効果的な投入が可能となる。この結果，開発工期の短縮も可能となる。

(4) 分散開発の実現

ワークステーションへの機能分散により，従来のTSSに比べて通信コストが大幅に削減できる。この結果，遠隔地でのプログラム製作が容易になる。

5 結 言

NS-OBJETは，システム部門が現在直面しているシステム開発上の問題のうち，プログラム製作作業と保守作業の合理化，及びホスト計算機に全面的に依存した開発方式から生じる問題など，システム開発作業の下流工程の工業化を今回実現した。

NS-OBJETの究極のねらいは，システム開発作業全体を工業化された一貫した生産システムに改革することであり，今後ともこの方針の下で，より上流工程(計画・設計作業)の工業化を推進していきたいと考える。

参考文献

1) 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)

2) 二村：プログラム設計法PAD/PAM，情報処理，25，11，1237～1246(昭59-11)