ソフトウェア工学 49－4
（１９８６．７．３０）

ソフトウェア開発環境用マンマシンインタフェース（ＨａｎｄＳ）

桑名栄二　小野雄二　中村雄三　長野宏宣
ＮＴＴ　電気通信研究所

パソコン（ＰＣ）やワークステーション（ＷＳ）の世界ではマルチウインドウやマウス等の新しいマンマシンインタフェース（ＭＭＩ）が開発されソフトウェア開発に活用されてきている．大規模共同利用環境でのいわゆる大型汎用機を用いた開発環境のＭＭＩにもＰＣやＷＳの手触りの良さを導入して環境を改善したい．

本論文ではユーザエンド側にＰＣやＷＳを用いてそれらの高機能ＭＭＩを利用し，バックエンド側で大型汎用機のツール起動する垂直分散型の環境におけるＭＭＩ構築の考え方について述べる．さらに，実際にＵＮＩＸワークステーション上に構築したソフトウェア開発支援用ＭＭＩ（ＨａｎｄＳ）について述べる．

Man-Machine Interface for Software Development Environment: HandS

Eiji Kuwana　Yuji Ono　Yuzo Nakamura　Hironobu Nagano
Electorical Communication Lab., NTT
1-2356 Take Yokosuka-shi Kanagawa 238-03 Japan
(英文住所とも含む)

In recent years, personal workstations with high resolution bit-map screen, multi-window, mouse and so on are utilized.
In this study, to provide software production environment with a rich and powerful man-machine interface, the following subjects were investigated. Also, in this paper, the man-machine interface design of vertical distributed software development environment was presented.

(1) Accumulation and reuse mechanism of software development process and know-how.
(2) Construction of user customized software development environment.
(3) Functions of increasing overall efficiency through decreasing miscellaneous software development tasks.

１．はじめに

パソコンやワークステーションの世界では，高精度ビットマップディスプレイやマルチウィンドウ等，高機能マンマシン・インタフェースが採用され，ユーザのレベルに応じた環境の構築が可能な使い勝手の良い環境が提供されている．

これに対して，大規模ソフトウェアの開発環境では，キャラクタディスプレイのスクリーンエディタの提供レベルに留まっている．

大規模共同利用環境での，いわゆる大型汎用機を用いた開発環境のマンマシン・インタフェースにも，最近のパソコン・ワークステーションの手触りの良さを導入してより良い環境にしていく必要がある．また，開発ソフトウェアの大規模化，ワークステーションの高機能化・低廉化，ソフトウェア開発の地方分散化，共同利用システムの使い難さ等に伴い，負荷分散，機能分散を図る分散ソフト開発環境構築が必要となってきている．

本研究において構築しようとしているソフト開発環境は，ユーザエンド側にワークステーションやパソコンを用いて，それらの高機能マンマシンインタフェースを利用し，バックエンド側で大型汎用機上のツールを起動するという，垂直分散型環境である．

本論文では大規模ソフト開発を前提としたソフトウェア生産環境用マンマシンインタフェースの設計条件を明らかにし，実際に構築したソフトウェア生産支援用マンマシンインタフェースの事例について述べる．

第２章で，ソフトウェア開発作業の分析によるマンマシン・インタフェースの基本設計方針について述べる．

第３，４章では，第２章で示す基本方針をもとに具現化した（プロトタイプした）ソフトウェア生産支援用マンマシン・インタフェースシステム（ＨａｎｄＳ：Humanized and Simple software development environment)の概要を述べる．

さらに，ソフト開発作業の手順構築の支援，実際の開発作業手順の蓄積，再利用を図る機構(HandSシナリオ機構)について述べる．HandSの動作環境（ハードウェア構成，ソフトウェア構成）について説明する．

第５章で，分散ソフトウェア環境でのHandSの利用イメージについて述べる．

２．新しいﾏﾝﾏｼﾝｲﾝﾀﾌｪｰｽ機構の活用

ここでは，ソフトウェア開発作業の分析による，ソフトウェア生産性向上の要因を示す．

さらに，ソフトウェア生産性向上の要因を支援するマンマシン・インタフェースの基本設計方針について示す．

２．１　ｿﾌﾄｳｪｱ生産性向上の要因とﾏﾝﾏｼﾝｲﾝﾀﾌｪｰｽ

１９７０年代に，UNIX/PWB(Programmer's Work Bench)において，開発環境とソフトウェアの実行環境の分離が，以下に示す概念のもとに実現された[1]．

①ソフトウェア開発，試験，運用は，異なった作業であり，その作業に適合した環境が選択されるべきである．

②開発システムは，統一されたインタフェースを利用者に与えるべきである．

③ターゲット・マシンとは独立に，開発システムは，中・小規模の計算機上で構築できる．

ワークベンチの目的は，自システムのソフトウェア開発を支援するのではなく，ターゲット・マシン上のソフト開発を支援することである．

このUNIX/PWBの考えをもとに，リモート・ジョブ・エントリー，リモート・ログイン機能，ソースコード管理（版管理，構成管理），テキスト処理／文書編集ツール，ホストのシミュレータ等の開発がなされてきた[2,3,4]．

このように，各ライフサイクル毎のソフトウェア開発作業に適合した環境に生産環境を分離し，支援ツールを提供することで，ソフトウェア生産性向上を図ることは，ある程度実現可能となった．しかしながら，改善すべき点はまだまだたくさんある．

以下に例を挙げる．

- 一時にたくさんの情報を見ることができない．
- 起動したツール間及び，各環境間での情報のやりとりが難しい．
- ソフトウェア開発作業手順（ノウハウ）の蓄積，再利用ができない．
- システムの提供するソフトウェア開発（試験）環境を，自分自身で再構成できない．
- 過去の画面情報の再利用ができない．
- ソフト開発における帳票類（例えばバグ票，コメント票）の管理等の付帯業務に時間が浪費されている．

これらの問題点は，ソフトウェア生産性向上の要因として，次のようにまとめることができ，新しいマンマシンインタフェース機構（ウインドウ，メニュー（例えばポップアップメニュー），マウス，アイコンや手続き言語を駆使したシステム）の活用の可能性がこれらである．

- 作業の並列化
- 情報・作業手順の再利用
- 作業の効率化，視覚化，省略

（１）作業の並列化

これは，同種又は，異種の作業を並列化して実施できる環境を意味する．

高精度ビットマップディスプレイに複数の窓（ウィンドウ）を開いて，同時に同種（又は異種）の仕事を指示する．さらに，複数の窓に表示される情報を互いに参照しながら作業を進める．例えば，自システム上のエディタで，プログラムテキストの編集を行いながら，別のウィンドウでホスト上のツールを起動したり，通信したりする．

（２）情報・作業手順の再利用

ソフトウェアの世界での再利用には，プログラムの再利用（部品化），テスト・データの再利用，作業手順の再利用，設計文書，さらに計算機出力メッセージの再利用等がある．

プログラムの部品化／再利用，データの再利用，設計文書の再利用機構は．一部の環境[例えば5,6]で実用化されている．

ここで取り上げる再利用とは，作業手順の再利用である．大規模ソフトウェア開発においては，複数のプロジェクトで作業が進行する．そして．その開発は，ある規範に従ったもので，手順．ドキュメント等管理すべき項目は多い．

また，プロジェクト内には熟練者から初級のプログラマまでいる．熟練者の作業手順ノウハウを蓄積し，プロジェクト内外の他の開発者に継承する機構があれば，ソフトウェア開発者の作業負荷，記憶負荷を減少することが可能である．これは，ソフトウェア開発作業手順（例えば，ソーステキストの編集，コンパイル，リンク作業の繰り返し作業）を蓄積し，再利用できる環境を意味する．

ソフトウェア開発には，種々の知識が要求される．例えば，要求定義においては，対象とする業務の知識，設計／製造においては，計算機操作の知識，プログラム言語の知識，ツール操作の知識等，多くの知識が要求される．知識獲得のために，要求されるソフトウェア開発者の学習負荷．同じ作業を繰り返し行う作業負荷を軽減し，生産性向上を図る必要がある．

ソフトウェア開発手順は，広い意味で考えると，ユーザの作業環境である．熟練者の手順（環境）を流用する場合，自分の作業手順や開発ソフトウェアの種類等に合わせて．環境を再構成するする必要のある場合もある．従って，単に手順の蓄積を行うのみでなく，ユーザ自身で再構成できる（ユーザカスタマイズできる）機構が必要である．

（3）作業の効率化，視覚化，省略

作業の並列化，手順の再利用も，作業の効率化に含めることは可能だが，ここでは並列化，再利用以外の作業の効率化，省略について考える．

ここでの効率化，省略としては，ソフトウェア開発における付帯業務の軽減を挙げることができる．付帯業務とは，帳票類（例えばバグ票，コメント票，連絡票）の管理や，ソフトウェア試験環境の設定（例えばＶＭ環境の設定）等が挙げられる．

現状の開発環境では，帳票類は手作業で管理されている．データ・ベースと開発環境の電子メール機構を組み合わせ，帳票類の管理（例えばバグ票の送付，未解決帳票のリストアップ，項目の分類，内容の検索等）を自動化し，必要な時に端末を通して参照・変更することを可能とする．

作業の視覚化とは，アイコン，メニューによるツール起動，複数の窓，アイコンを用いたディスクトップ等による計算機処理の直観的な表現を意味する．

作業の視覚化は，ユーザの行う作業の種別を分かり易くするだけでなく，従来のコマンド投入によるユーザの入力ミスの防止，キーストローク時間の短縮の効果がある．

２．２　基本設計方針

２．１節で述べた作業の並列化，再利用，効率化，視覚化を行うために，以下に示すＷＳ等で提供されている機能（以降，マンマシン・インタフェース構成要素と呼ぶ）をフル活用する．

・マルチウィンドウ，マルチタスク
・ポップアップメニュー
・マウス，アイコン
・高機能スクリーン・エディタ
・シナリオ機構

ここでのシナリオとは，ソフト開発における作業手順を記述したものを意味する．

マンマシン・インタフェース構成要素と，ソフトウェア開発支援項目（マンマシン・インタフェース基本設計方針）の対応を表－１に示す．

［表－１］マンマシンインタフェース構成要素と
マンマシン・インタフェース基本設計方針

基本設計方針	マンマシンインタフェース構成要素
作業の並列化 ・大量の異なる種類の情報を同時に表示 ・異なる種類の作業を同時に行う	マルチウインドウ マルチタスク ポップアップメニュー アイコン マウス
作業の視覚化	
情報の再利用 ・スクリーンに表示されている全ての情報の修正を可能とする	高機能スクリーンエディタ バーチャルペーパ
作業の機械化，再利用，効率化 ・作業手順の蓄積・再利用 ・ソフトウェア開発作業者の知識，ノウハウの継承	シナリオ シナリオ定義
環境定義 ・動作環境のユーザカスタマイズ化	
作業のガイダンス 付帯業務の軽減 ・ソフトウェア開発における帳票管理の電子化	電子メール ＯＡツール

３．ＨａｎｄＳマンマシンインタフェースの概要

［表－１］に示すマンマシン・インタフェース構成要素を用いてソフトウェア開発作業の支援を行うが，ここでは，従来のワークステーションで提供されてきたマンマシン・インタフェースとの差異と，本システムで提供するマンマシン・インタフェースの特徴を述べる．

３．１　ＨａｎｄＳの動作環境

図－１に示すように，ＨａｎｄＳは，ＵＮＩＸ上で稼動するシステムである．
ＵＮＩＸ環境上に構築した理由を以下に示す．

- ＵＮＩＸには，応用プログラム，生産支援システム構築を支援するツール群が既に存在する．
- ＵＮＩＸは，既に多くの環境で利用されており，新しいシステムに展開することは比較的容易である．
- ＵＮＩＸは今後，よりユーザ・エンド側の小型マシンでも利用可能となる．

また，ＵＮＩＸそのものを，生産支援システムとして利用しない理由を以下に示す．

- ＵＮＩＸの利用と操作は，初心者も含めた一般のプログラム開発者向きでない．
- ＵＮＩＸ上に，マルチウィンドウ，マウス等のマンマシン・インタフェースを採用したシステムはあるが，まだまだ高価であり一般的でない．

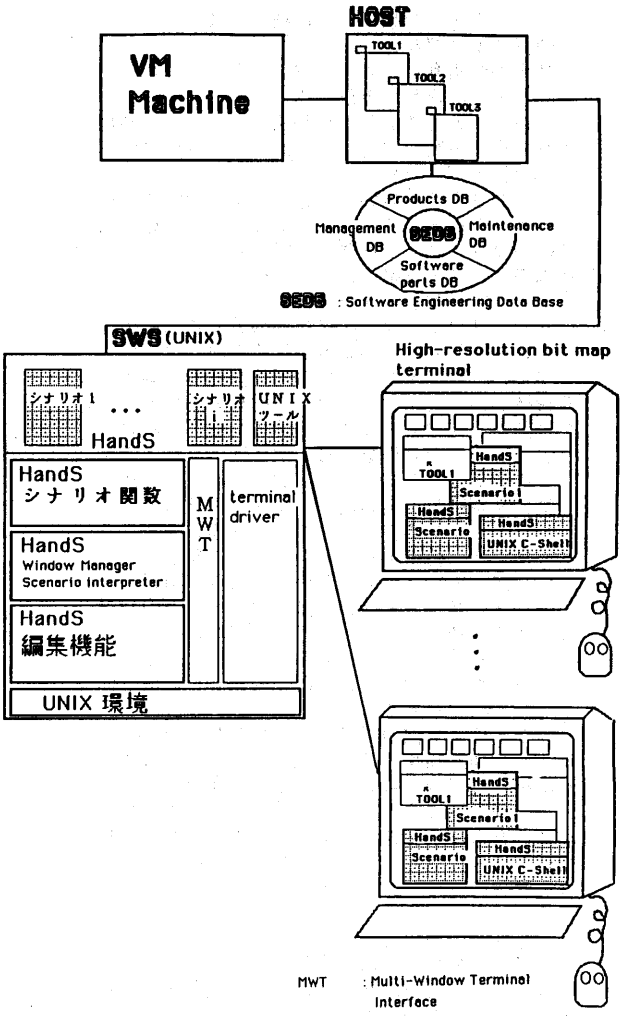
このような理由から，ＨａｎｄＳはＵＮＩＸ上にマルチウィンドウ，マウス等を持ったソフトウェア生産用マンマシン・インタフェースを新たに構築することにした．この動作環境上の特徴の一つは，１台のＵＮＩＸマシン上で，複数人が端末を通し，高機能マンマシン・インタフェースを利用できることである．
これは，従来のＴＴＹ型の端末に代わり，ウィンドウ表示制御，リージョン制御，マウス制御，フォント・アイコン，メニュー制御機能を持った高精度ビットマップ端末（ＨａｎｄＳではＤＣＬ社のＧＴＲＯ１，パソコン等）を用い，ビットマップ端末とＵＮＩＸワークステーション間で，垂直型の機能分散を図ったことによる．本システムでの端末と，ＵＮＩＸマシン上のＨａｎｄＳ間の機能分担の基本的考え方を以下に示す．

- 端末側
  - ①画面，キーボード，ウィンドウ，マウス制御
  - ②ＵＮＩＸマシンとの通信
- ＨａｎｄＳ側
  - ①ＷＳに対するマルチウィンドウ制御
  - ②エディタ
  - ③応用プログラム（含：大型ホスト通信）
  - ④作業手順の蓄積，再利用（シナリオ）
  - ⑤ＵＮＩＸ各種ツールによるソフト開発支援

３．２　ＨａｎｄＳマンマシンインタフェースの特徴

以下に，ＨａｎｄＳマンマシン・インタフェースの特徴を簡単に説明する．

- 複数人が同時に高機能マンマシンインタフェースを利用できる



［図－１］HandSソフト開発環境の構成

近年のワークステーションにおいて（例えばSUNワークステーション，Apple社のMacintosh，Apollo社のDOMAIN等）も，マルチウインドウ，マウス，ポップアップメニュー（またはプルダウンメニュー），アイコン等が取り入れられ操作性が格段にアップしている．

しかし，これらのワークステーションで本当にソフトウェア開発に利用できるものはまだまだ高価である．さらにマルチウインドウ等のマンマシンインタフェースの利用は本体と一体型で提供されているため，一人のみである．

これに対して，図－１，前節で示したように，ＨａｎｄＳは端末として利用する高機能ビットマップディスプレイとＵＮＩＸワークステーションの間での機能分担により，複数人に高機能マンマシンインタフェースの提供を行っている．

従って，１台のＵＮＩＸワークステーションのもとで，　複数人の開発者が同時にＨａｎｄＳマンマシンインタフェース機能を利用しソフトウェア開発を進めることができる．

・エディタが中心

ＨａｎｄＳマンマシンインタフェースのもう一つの特徴はエディタである．
ＨａｎｄＳはエディタがベースであり，ユーザがコマンド投入，テキスト入力できるウインドウは全てスクリーンエディット可能である．
また，全てのウインドウが無限ロールペーパ（またはバーチャルペーパと呼ぶ）構成となっており，過去に投入したコマンドやその結果を参照したり，再利用したりすることができる．
ウインドウ内，ウインドウ間のテキスト情報のやり取りはＨａｎｄＳエディタのＹａｎｋバッファを用いる．
また，画面操作，テキスト入力のインタフェースをツール間で統一するために，ＨａｎｄＳエディタを入力フロント・エンドとしてツールに組み込むことを可能とする．

・シナリオ機構

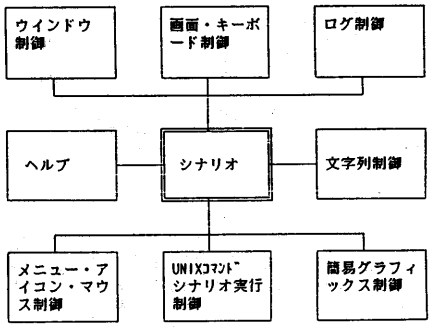
ＨａｎｄＳの提供するマンマシンインタフェースのもう一つの特徴は，ソフトウェア開発作業の流れとそのとき使用するツールを記述するシナリオ機構（手順を記述したものをシナリオと呼ぶ）である．
シナリオ機構は作業手順の記述の他にツールの選択や各ツールのウインドウ等の利用が可能である．これはＨａｎｄＳ提供のウインドウ，ポップアップメニュー，マウスの利用の仕方，さらにヘルプや文字列制御，ツールの起動環境等をシナリオの中に記述することができる．
シナリオ機構については，詳しく４章で説明する．

４．ＨａｎｄＳシナリオ

大規模ソフト開発を前提とした場合，ソフトウェア生産性向上には作業手順統一が必須である．これは以下に示すソフトウェア開発形態の問題等から必要となる．

・作業手順は作業環境，開発するソフトウェアの種類，要員の質により異なり，技術，ノウハウの継承がない．
・要員の中には非熟練者，新人が多い．
・バッチスタイルの開発形態が残っている．

これらの現在でも見られる非生産的スタイルから脱却し，ソフトウェア開発作業の自動化，作業環境の設定，開発手順の再利用（エキスパートの蓄えている知識の移転）を行うのがＨａｎｄＳシナリオである．
ＨａｎｄＳは２．１節に示した作業の自動化等を行うために，ＵＮＩＸのＣシェル風のコマンド記述言語と図－２に示すシナリオ関数を持っている．シナリオ関数はマンマシンインタフェース向上を目指すものと，ＵＮＩＸコマンドやシナリオ実行制御を行うものに大別できる．



［図－２］シナリオ関数

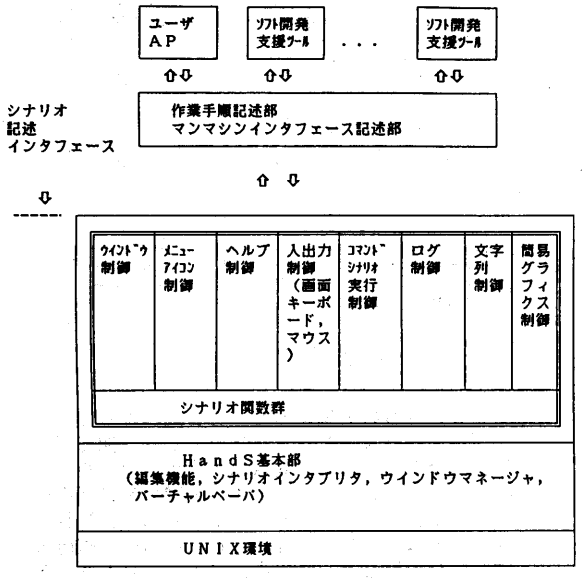
ＨａｎｄＳの実現にあたって，高精度ビットマップ端末とＵＮＩＸワークステーション間で垂直型機能分散を図った．同様に大規模ソフトウェア開発環境においてもソフトウェア開発者に高機能マンマシンインタフェースを提供すべく大型機のフロントエンドとしてＨａｎｄＳを用い，垂直分散型開発形態をユーザに提供する．　ＨａｎｄＳ上でソーステキストの編集作業，その後のｺﾝﾊﾟｲﾙ・ﾘﾝｸ・試験作業等を一貫した形でＨａｎｄＳシナリオ機構を用いて支援する．大型ホスト上のツールの起動選択，パラメータ投入等をＨａｎｄＳの持つウインドウ，メニュー制御用シナリオ関数を用いて視覚化した形で手順化し作業支援を行う．利用形態の一例については５章で述べる．

４．１　ﾏﾝﾏｼﾝｲﾝﾀﾌｪｰｽの向上，手順制御

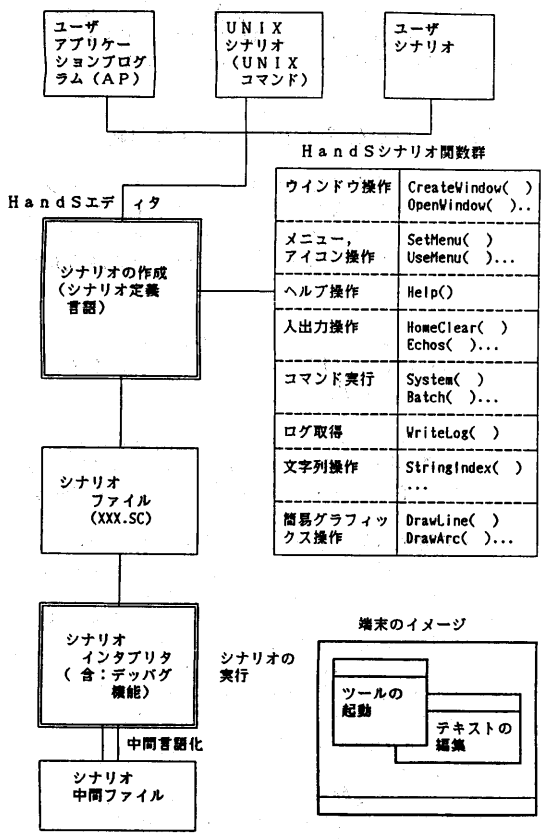
従来，ＵＮＩＸ環境のもとでもアプリケーションプログラム（ＡＰ）記述段階でウインドウ，マウス，ポップアップメニュー等とのインタフェースを提供したものもあったが，以下の問題点があった．

①マンマシンインタフェースがＡＰ依存
②特定の言語（例えばＣ言語）に対してインタフェースの提供

ＨａｎｄＳのもとではユーザは図－３に示すように作業手順記述段階（シナリオ記述段階）でマンマシンインタフェースの操作を記述することができる．
ユーザ側から見たシナリオ作成の概要を図－４に示す．ユーザはＨａｎｄＳがシナリオ関数として提供するウインドウ等の制御関数をＵＮＩＸコマンド（ツール）と同様にシナリオ中に記述する．
シナリオはインタプリタ形式で実行されるが，シナリオ作成においてはシナリオインタプリタのデバッグ機能（トレース機能，ブレイクポイント設定機能等）を用いて効率化を図っている．
シナリオの例を図－５に示す．図－５はウィンドウをオープンし，そのウィンドウでシナリオインタプリタを起動するシナリオである．図－５で下線でマークしたものがシナリオ関数である．以下に，ＨａｎｄＳで現在提供しているシナリオ関数の内容を示す．



［図－3］シナリオ関数の位置付け



［図－4］シナリオの作成と実行

（1）ウィンドウ制御機能

ウィンドウ制御機能として，以下のものを持つ．

- ウィンドウの生成，表示，消去，削除，切り換え，稼動
- ウィンドウコンテキストの変更
- ウィンドウのポップ，プッシュ
- キーボードのアタッチ
- ウィンドウ状態の問い合わせ

（2）入出力制御機能

- 画面消去，カーソルのホームポジション移動
- フォントローディング
- 文字列出力，メッセージ出力
- キーボードから文字入力

（3）コマンド制御

シナリオ・コマンド制御として，以下のものを持つ．ここでの特徴は，複数の作業の同期実行，並列実行を支援するための Execute/Wait 関数を持つことである．

- シナリオやＵＮＩＸコマンドのフォアグラウンド実行
- バッチ実行，時間指定実行
- シナリオの同期，非同期実行

（4）メニュー・アイコン制御

- ポップアップメニューの登録・表示・削除，表示取り消し
- アイコンの登録・表示（同期，非同期）・削除，表示取り消し，反転表示

（5）ヘルプ制御

- ヘルプメッセージの登録

（6）ログ制御

- ログ情報のファイル出力

（7）文字列操作

文字列操作は，ＰＷＢにおけるホスト計算機の出力メッセージの解析を行い，ツール起動を制御するために，以下のものを持つ．

- 位置の検索
- 文字列の長さ
- 文字列の取り出し
- 正規表／セパレータによる文字列検索

（8）簡易グラフィックス制御

- 直線，円弧，ボックスの描画
- 閉領域の塗りつぶし

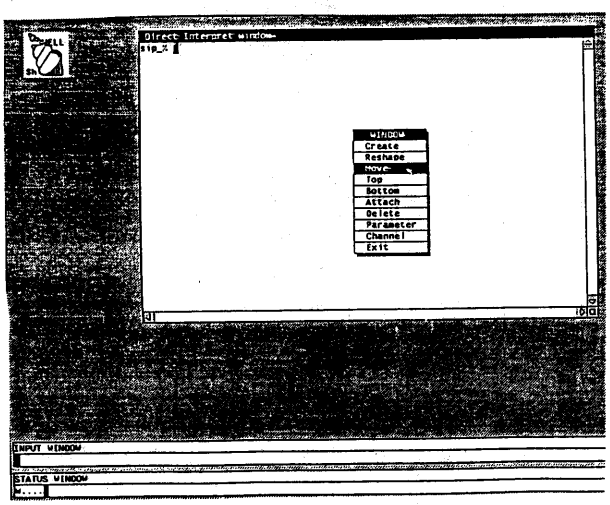
```
#
#       Direct Interpret Scenario
#

Echos( "\n Direct interpret ..." )
set title = " Direct Interpret Window"
include ./Make_Window                   # Return variable 'Window_no'

#Echos( "\n Window status = " )
#set return = StatusWindow( $Window_no )
#if ( $return == "-1" ) then
#       Echos ( " .. StatusWindow Fail .. " )
#       Exit()
#endif
#Echos( "$return" )


System( "sip" )
set return = CloseWindow( $Window_no, 20, 100 )
if ( $return == "-1" ) then
        Echos ( " .. CloseWindow Fail .. " )
        Exit()
endif

Return:
```



［図－5］シナリオファイルの例

４．２　環境定義

ＨａｎｄＳシナリオはツール起動順序やマンマシンインタフェースの制御の他にツールやシナリオの起動環境（またはアクセスできるファイル等の記述も含む）を定義することができる．例えばあるプロジェクトに属するあるユーザにはある特定のファイルに対してＲｅａｄ権のみを与えるというように環境を定義できる．　従来ＵＮＩＸ環境ではファイルのアクセス権をowner／group／worldで設定できた．しかし，大規模プロジェクトのもとであるユーザに対してはＲｅａｄ権のみという設定の必要性もでてくる．このような要求に対処するためにシナリオ内での環境設定を可能としている．
また，ＵＮＩＸのシェルと同様にシナリオ変数という形でホームディレクトリやパスを設定できる．

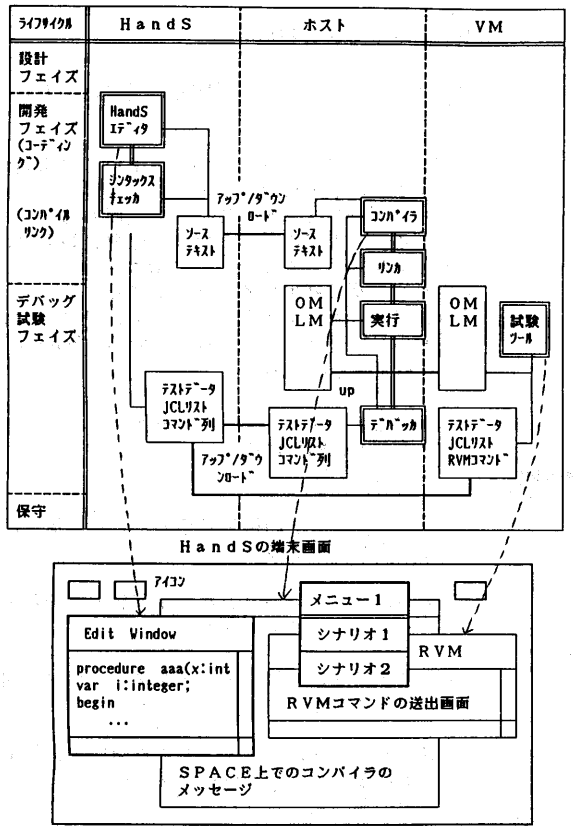
５．ＨａｎｄＳの利用

５．１　大型ソフトウェア開発のフロントエンド形態

本節では大型ホスト上でのソフトウェア開発のフロントエンド形態としてのＨａｎｄＳ利用イメージについて説明する．
大型ホスト上でのソフトウェア開発の効率化，負荷分散を図るために，以下に示すＨａｎｄＳの機能をユーザは利用することができる．
これはＨａｎｄＳを大型ホスト上でのソフトウェア開発のＰＷＢ（Programmer's Work Bench）として利用するものである．

- 高機能スクリーンエディタを利用したソーステキストの編集作業
  （ＭＭＩの向上）
- ソーステキストの編集効率向上，及び構文誤り混入を防止する構文エディタ
- 大型ホスト上でのコンパイル回数削減のためのシンタックスチェッカ
- 大型ホストとＵＮＩＸワークステーション間のファイル転送
- ＨａｎｄＳマルチウインドウを用いた大型ホスト作業とＵＮＩＸワークステーション上の作業の並列化，視覚化
- ＨａｎｄＳシナリオ機能とホストOS-UNIX間の通信機能を用いたホストコマンド自動送出（ツール実行制御），メッセージ解析
- ＨａｎｄＳからのＲＶＭ（Remote-VM）制御

ＰＷＢの概念は文書処理と等しいプログラムテキストの編集作業を手元の端末（ワークステーション）で行いながら，ホストに大型の仕事を依頼するというものである．ＨａｎｄＳではＵＮＩＸワークステーションとホストＯＳの提供する通信機能，ファイル転送機能とＨａｎｄＳの提供するマルチタスク・マルチウインドウ機能等のビジュアルなソフト開発支援環境によってＰＷＢを実現している．
ユーザはＨａｎｄＳで開発したソフトウェアをホスト（ターゲットマシン）や，ＲＶＭ上で試験する場合はリモート・ログイン機能を使う．
例えば大型ホストにＨａｎｄＳからログインし，ＨａｎｄＳの一つのウインドウを仮想端末としてオープンし，ホストでのプログラム実行状況を監視する．ＨａｎｄＳを用いた場合の作業イメージを図－６に示す．この時，４章で示したＨａｎｄＳシナリオがユーザのＵＮＩＸワークステーション上での作業（例えば，編集作業とシンタックスチェック），大型ホストでの作業（ファイル転送，コンパイル，リンク，試験）をサポートする．



[図－6] ＤＩＰＳソフトウェア開発のフロントエンド例

一連の定型作業はＨａｎｄＳのシステムシナリオ（予めＨａｎｄＳで用意したシナリオ）として提供するが，ユーザは自分の作業形態に合わせてシステムシナリオを書き直し，自分自身の環境で作業を進めることもできる．

５．２　スタンドアローン形態

本節ではＨａｎｄＳのスタンドアローン形態の利用について説明する．
ＨａｎｄＳは大型ホストの存在に関係なく，ＵＮＩＸ上の高機能ソフトウェア開発環境として機能する．
従来のＵＮＩＸソフトウェア開発環境上でもマルチウインドウ，マウス等のマンマシンインタフェースを提供したものは存在した．しかし，３．２節のＨａｎｄＳシナリオのところで述べたようにＨａｎｄＳ環境では複数人が以下の機能を用いてＵＮＩＸ上のソフトウェア開発を効率良く進めることができる．

・ＵＮＩＸシェルレベルでウインドウ等のマンマシンインタフェース制御が可能
・既存のＵＮＩＸツールをそのまま利用できる

ＨａｎｄＳとしてＵＮＩＸソフトウェア開発支援のために，現在，特別にツールを用意していないが，［表－２］に示すツールの中で共通と示してあるものは，フロントエンド，スタンドアローン形態に関係なく利用できる．
ここに示したツールで，構文チェッカ，構文エディタ等を除いたものは，ユーティリティーとして位置付けることができる．

［表－２］ＨａｎｄＳ提供のツール

ツール名	機能概要	利用形態
シンタックスチェッカ	ﾎｽﾄでのｺﾝﾊﾟｲﾙ回数削減のためのｼﾝﾀｯｸｽﾁｪｯｶ	front-end
簡易構文エディタ	ｿｰｽの作成，修正支援	front-end
帳票管理システム	ソフト開発における帳票管理，帳票の自動発信／受信	共通
ホスト・ＲＶＭ制御	UNIXからﾎｽﾄ，ＲＶＭへのﾌｧｲﾙ転送，ｺﾏﾝﾄﾞ列送受信	front-end
アイコン・エディタ	アイコン作成支援ツール	共通
ディレクトリエディタ	ファイル，ディレクトリ操作支援ツール	stand-alone
メール（含：電子掲示板）	ＵＮＩＸﾕｰｻﾞ間の電子ﾒｰﾙ電子掲示板	共通
ノート・パッド	メモ用紙	共通
電卓	電卓	共通
簡易イメージエディタ	お絵書きツール	stand-alone

６．おわりに

本論文では，大型ソフトウェア開発支援用マンマシンインタフェースの一例として，高精度ビットマップ端末，ＵＮＩＸワークステーション，大型ホスト間で垂直機能分散を行った環境を示した．
従来のマルチウインドウ等のマンマシンインタフェースの利用できるワークステーション環境と異なり，ＨａｎｄＳシステムのもとでは，ソフトウェア開発作業手順の再利用，機械化し，統一された作業手順，マンマシンインタフェースをユーザに提供できると考える．
今後，機能の充実，性能の改善を図り，実際の大規模ソフトウェア開発に適用していく考えである．

謝辞

今回，本研究の機会を与えてくださったソフトウェア生産技術研究所生産システム研究室伊東室長，ならびにシステム構築にご協力頂いたＤＣＬ社下村氏，三橋氏に記して感謝します．

[参考文献]

[1]T.A.Dolotta et al. "UNIX' Time-Sharing System: The Programmer's Work Bench," Bell System Tech. J., 57, No.6(1978)

[2]B.W.kernighan et al. "UNIX Time-Sharing System: Document Preparation," Bell System Tech. J., 57, No.6(1978)

[3]M.J.Rochkind "The Source Code Control System," IEEE Trans. on Software Eng., SE-1, 4(1975)

[4]伊東 他 "総合ソフトウェア生産システムの実用化," ＮＴＴ通研実報, Vol.33, No.12 (1984)

[5]L.P.Deutsch "Reusability in the Smalltalk -80 Programming System," Workshop on Reusability in Programming(1983)

[6]B.W.Kernighan "The UNIX System and Software Reusability," Workshop on Reusability in Programming(1983)