TM-0181

SIMPOSにおける
プログラミング環境

石橋弘義，近山　隆，吉田かおる，
佐藤裕幸，内田俊一

July, 1986

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

ＳＩＭＰＯＳにおける
プログラミング環境

石橋弘義，近山隆，吉田かおる，佐藤裕幸，内田俊一
（財）新世代コンピュータ技術開発機構

1．まえがき

ＳＩＭＰＯＳ（Sequential Inference Machine Programming and Operating System）は、ＩＣＯＴで開発した逐次型推論マシンＰＳＩ（Personal Sequential Inference Machine, ψ）用のオペレーティング・システムである。ＳＩＭＰＯＳはすべてＥＳＰという論理型のプログラミング言語で記述されている。ＰＳＩ／ＳＩＭＰＯＳ／ＥＳＰは論理型プログラミングのための良好な開発環境を提供し、現在、ＩＣＯＴ関連の多くの研究分野で利用されている。

本報告ではＳＩＭＰＯＳの記述言語ＥＳＰ、およびＳＩＭＰＯＳのユーザ・インタフェイスについて述べる。

2．SIMPOS 概要

PSI は論理型プログラムのための良好な開発環境を提供するいわゆるスーパー・パーソナル・コンピュータ・システムである。その処理速度は汎用大型コンピュータ上のプロローグの処理系（DEC10-Prolog）と同等の約３０Ｋ LIPS(Logical Inference Per Second) を実現し、実メモリ容量は６４個の１６Ｍ語の実装が可能である。これにより、実用規模の大きなプログラムの実行にも耐えうる計算機システムとなっている。

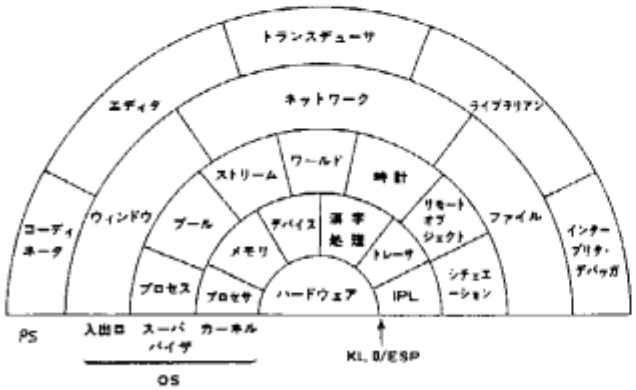
(1) 設計方針

SIMPOS は以下の基本設計方針により作られている。

①個人用のプログラム作成ワークステーションとして、良好なプログラム開発環境を支援する
②マルチ・ウインドウ環境を実現し、高度なマンマシン・インタフェイスを持つ
③コンピュータ・ネットワーク機能により、資源や情報の共有を容易にする
④論理型プログラミング言語ＥＳＰに基づく単一言語システムを提供する
⑤新しい概念（オブジェクト指向）に基づき、簡潔で一貫性をもち柔軟性の高いシステムの構築を計る

(2) システム構成

SIMPOS は、単一ユーザ，複数プロセス，対話的処理を基本とするオペレーティング・システム部（ＯＳ）と、その上で使いやすいユーザ・インタフェイスを提供するプログラミング・システム部（ＰＳ）とから成る。



OS： カーネル（資源管理プログラム）はハードウェア資源（プロセサ，メモリ，入出力装置など）の管理およびシステムの起動を司る。スーパバイザ（実行管理プログラム）はプログラムを実行するために必要な機能や環境を提供する。入出力メディア・システムは PSIと外界との間の通信路を提供する。

PS： プログラミング・システムとはユーザが直接操作するプログラム開発支援ツール（エキスパート）のことで、 OS の各種機能を提供するマニピュレータもこれに含まれる。

3．記述言語ＥＳＰ

(1) ＥＳＰとＫＬＯ

SIMPOS はすべて ESPで記述されている。 ESPは SIMPOS のシステム記述言語のみならず応用プログラムのためのユーザ言語としても広く使われている。ESP は論理型言語 Prolog をベースにして、オブジェクト指向の考えを導入することにより、大規模なプログラムの開発に欠かせないモジュール化の機能を実現している。さらに、ESP は強力なマクロ展開機能を備えており、プログラムの読み易さ／書き易さを向上させている。

ESPで書かれたプログラムは PSIの機械語である KL0にコンパイルされる。KL0 は幾つかの機能拡張を施した Prolog 系の論理型言語である。KL0 の全ての機能は ESPでも直接利用できる。

ESP （Extended Self-contained Prolog）
- ＝ 論理型プログラミング言語 （Prolog）
- ＋ オブジェクト指向型言語 （Smalltalk, Flavor）
- ＋ マクロ展開機構
- ＋ KL0 言語機能

KL0 （Kernel Language version 0 ）
- ＝ 論理型プログラミング言語 （Prolog）
- ＋ 拡張データ型 （stack-vector, heap-vector, string）
- ＋ 拡張制御構造 （remote-cut, on-backtrack）
- ＋ ハードウェア操作用組込述語 （set-word, set-register）

(2) オブジェクトとクラス

オブジェクト指向とは、抽象データ型の考え方を発展させたものであり、手続き（操作）とデータを一体化させたオブジェクトを基本とした考え方である。

クラスは、良く似たオブジェクトをひとまとめにして定義したもので、プログラムの一つのモジュール単位となる。また、クラスを定義する際に他のクラスの性質（定義内容）を自分自身の中に取り込む機能を継承と呼ぶ。この継承機能により、クラス間に共通する性質をできる限りまとめて記述し、異なる部分だけを個別に記述することが可能になる。

(3) メソッドとスロット

ESP の述語には、クラスの外部から参照することのできるメソッドと、定義されたクラス内でのみ使用されるローカル述語の二種類がある。つまり、メソッドはオブジェクトの外部インタフェイスであり、ローカル述語は情報隠蔽のために使われる内部述語である。また、メソッドは継承により取り込まれるのに対して、ローカル述語はたとえ継承されても取り込まれない。

スロットは、オブジェクトの動的に変化する状態を保持するための状態変数である。論理型言語の通常の論理変数とは異なり、代入により上書きが可能であり、また、バックトラックしても値は元には戻らない。

(4) オブジェクト指向プログラミング例　－多重継承－

ESP の大きな特徴の一つに、複数クラスを継承できる多重継承機能がある。この機能を用いれば、ユーザは SIMPOS の提供する機能を自由に組み合せ、さらに自分の必要とする機能を付加することにより、自分用にカスタマイズできる。この時、 SIMPOS 自体もこのような使用法を念頭に置いた設計になっており、標準的な機能を組み合せたレディメイドのクラスを提供するとともに、機能単位の部品クラスも各種提供している。

例えば、標準入出力のクラス standard-io-window の場合、そのクラス定義には 10 個の継承クラスを定義しているだけであるが、実際の全継承クラス数は 40 個にもなる。さらに、それらのクラスから継承されるスロットは 121個（内、クラス・スロット 1），メソッドは 527個（内、クラス・メソッド 14 ）にものぼる。

(5) オブジェクト指向プログラミング例　－オブジェクト－

オブジェクト指向プログラミングにおいては、実行の主体はオブジェクト自身である。つまり、あるメッセージにがオブジェクトに渡された時、具体的に何をすべきかと言うことはオブジェクト自身が知っていると言う考え方である。　例えば、出力ルーチン（部品クラス）を作る時は、その出力先が何であるかと言うことを気にせずに、渡された出力用オブジェクトに対して、出力メッセージ（メソッド）を送ってやるだけで良い。 SIMPOS 自身このような方針に従って、ウインドウ／ファイル／バッファ（メモリ）などを標準入出力機能として提供している。さらに、 SIMPOS ではこの概念を入出力のみならずあらゆる部分に適用させたシステムとなっている。

(6) オブジェクト指向による特徴

長所

①モジュール化がすすみ、プログラムの共有，既存のプログラムの流用が容易になる．

②システムの拡張が容易である．
　新たな処理に対しては、手続きの記述の "変更" ではなく、新しく "追加" するだけでよい．

③概念や思考の整理がしやすい．

短所

①メソッド呼出しに対してどのクラスのメソッドを実行するのかは実行時に動的に決まるために、メソッドはローカル述語に比べて実行時のオーバーヘッドが大きい。このため、ローカル述語のみで記述できる内部処理は、メソッドを使わないのが ESPのプログラミング・スタイルである。

②親クラスで継承に係わる変更があると、子クラスも解析しなおさなくてはならない。これは、実行時のオーバーヘッドを少なくするために、継承解析を登録時に静的に行っているためである。

③継承を使って新しくクラスを作る場合、親クラスで定義されているスロットやメソッドを誤って再定義してしまう危険がある。これに対しては、登録時に警告を出すなどの対策が考えられる。

(7) SIMPOSの記述

SIMPOS は最下層のデバイス・ハンドラやメモリ，プロセス管理ルーチンから、最上層のプログラミング・システムに至るまで、すべてESP で記述されている。下層では充分な処理速度が、上層では記述性の高さが要求される OS のような大規模なシステムを単一の言語で統一的に、しかも短期間で開発できたことは、ESP の言語としての優秀性に負うところが少なくない。

SIMPOS 2.1版のサイズは以下の通りである。

ソース・プログラム（注釈行，空行を含む）	約200，000行
クラス総数	1，448個
定義述語総数	約 17，600個
内、ローカル述語	約 8，380個

4．ユーザ・インタフェイス

以下で、 SIMPOS についてユーザ・インタフェイスを中心に説明する。

(1) ウインドウ

ウインドウは SIMPOS のユーザ・インタフェイスの中核をなすものである。 SIMPOS では、ビットマップ・ディスプレイという一つの物理端末上に、ウインドウという多数の論理端末を構築している。これにより、ユーザは各ウインドウで別々の仕事をすることができ、複数プロセスの処理状況を同時に見ることもできる。また、一つのプロセスであっても、性質の異なる表示を別ウインドウに出すことにより、より見易い表示を行うことができる。さらに、ウインドウとマウス操作を組合せることで、メニュー選択という便利な操作法も提供されている。 SIMPOS では、種々のタイプのメニューが用意されており、ユーザ・インタフェイスの向上に役立っている。

以下で説明するプログラミング・システムやマニピュレータは、ウインドウ・システムの機能を利用して優れたユーザ・インタフェイスを実現している。

(2) ログイン

ユーザ名とパスワードを入力すると、そのユーザ用の初期化が行われる。初期化操作には以下のものがある。

・システム・メニュー項目の設定
　例えば、自分で作ったプログラムを呼出すためのメニュー項目 "my program" をシステム・メニューに加えることが出来る。

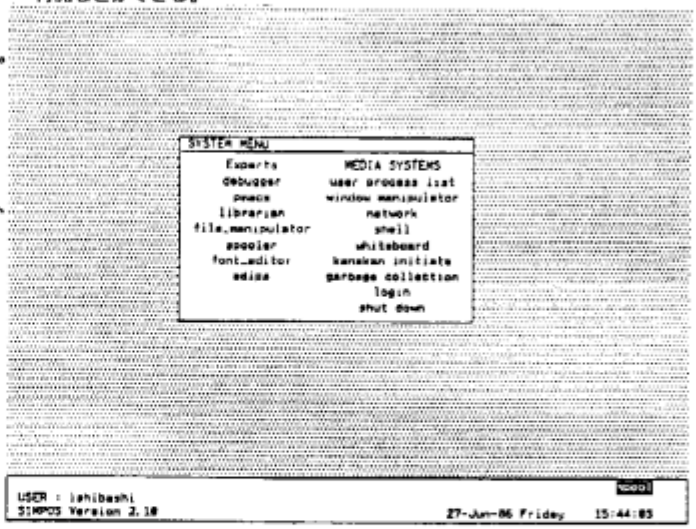
・ディレクトリに対する論理名の設定
　デフォールトとして、 me: (>sys>user> ログイン名) がある。

・初期化プログラム
　ログイン時に指定のプログラムを走らせることにより、その他自由に前処理ができる。

(3) システム・メニュー

システム・メニューは SIMPOS のコマンド・プロセッサの役割を果すもので、いつでも右2回のマウス・クリックにより呼出すことができる。

メニューには EXPERTSと MEDIA SYSTEM の欄がある。 EXPERTSは各々複数個の別プロセスとして使用できるが、MEDIA SYSTEMは各々1つしか作れない。また、これらの各プロセスはマウス操作のみで起動／終了／中断／再開などができる。



(4) プログラムの作成／実行手順

ESP プログラムの作成／実行／デバッグは、基本的には以下の操作の繰返しである。

①エディタで、ソース・プログラムを作成／修正する。

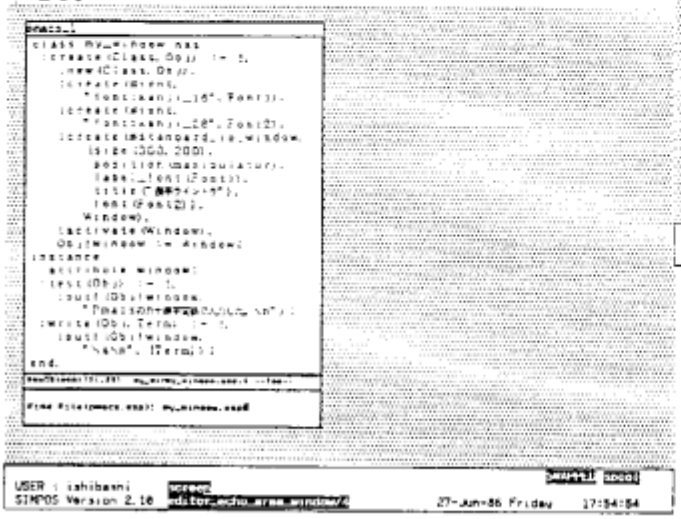
②ライブラリアン（コンパイラ）で、実行可能なコードに変換する。

③デバッガ（コマンド・インタプリタ）で、プログラムを実行またはデバッグする。

(5) エディタ

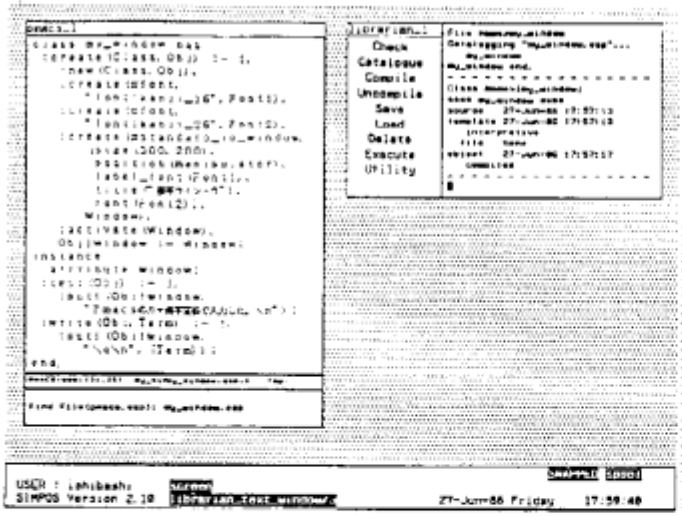
エディタは、プログラムやテキストの作成／編集、ファイルへの格納などを行う。

エディタには Pmacsと呼ばれる Emacs風のエディタと、構造編集可能な Edipsの二種類用意されている。 Pmacsはカナ漢字変換機能を有しており、日本語の入力も容易に行なえる。さらに、 Pmacsはバッファと呼ばれる編集エリアを持っており、編集可能な入出力ウインドウとして使うこともできる。



(6) ライブラリアン

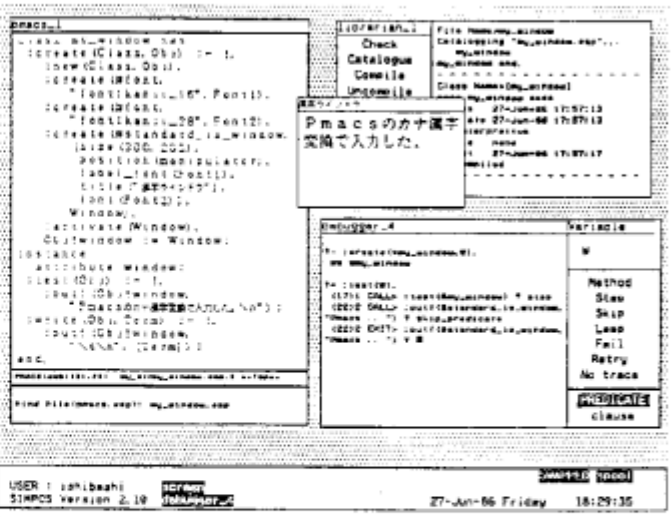
ライブラリアンは ESPソース・プログラムから実行可能なコードを生成し、管理する。実行可能なコードには、解釈実行コードと直接実行コードの２種類ある。デバッグ中のクラスは解釈実行コードを、デバッグの完了したクラスは直接実行コードを用いることにより、効率良くデバッグや実行ができる。さらに、ライブラリアンは OS やユーザ定義のクラスに関する情報の検索機能も提供している



(7) デバッガ／インタプリタ

デバッガはプログラムのデバッグを目的とするものであるが、対話的なプログラムの実行をおこなう ESPリスナの働きもする。

デバッガのトップレベルからは、既にライブラリアンに登録されているメソッドや ESPの組込述語からなるゴール列が実行できる。メソッドの呼出し方法は解釈実行コードであっても直接実行コードであっても同じである。もちろん、トレースや実行制御ができるのは解釈実行・コードに限られる。



ESP におけるデバッグの方法として、トレースなど述語の実行順序を確かめる方法の他に、オブジェクトの内容（スロット値）を確かめる方法がある。後者を行うのがインスペクタであり、トレース中のオブジェクトの状態を見たりすることができる。

(8) ウインドウ・マニピュレータ

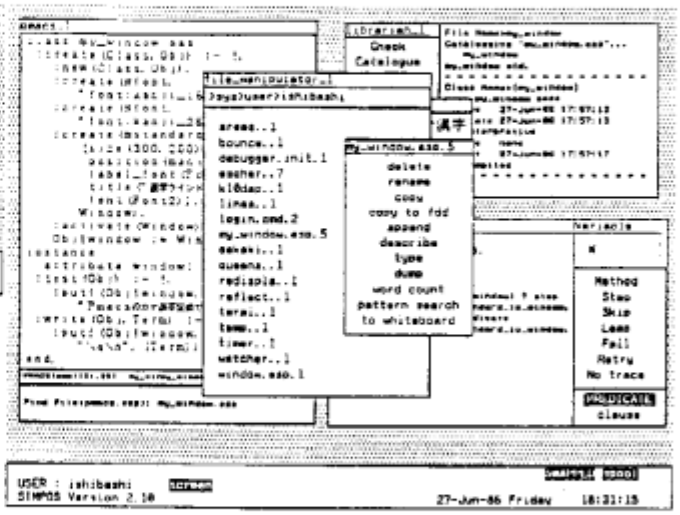
ディスプレイ上に表示されているウインドウの位置やサイズなどをメニューにより、変更することができる。その他、ウインドウの状態（shown/hidden）や属性（表示フォントなど）も変えることができる。さらに、ウインドウひとつひとつやスクリーンの一部分のハードコピーを取ることもできる。

(9) ファイル・マニピュレータ

SIMPOS のファイル・システムは UNIX と同様な木構造をしている。各ファイルはその階層に従って、 ">sys>user>me>file-name" 等のパス名でアクセスされる。また、マルチプロセス環境での共用／排他制御機能も備わっている。

以下に示す各機能はメニュー方式で提供されている。

- パス名に対する論理名の設定
- ディレクトリの生成
- ファイル名一覧の表示
- ファイルの消去、コピー
- ディスクとフロッピ間のファイル・コピー



(10)ネットワーク

PSI はイーサネットによるローカル・エリア・ネットワーク (LAN) をサポートしている。これにより、 PSI-PSI間あるいは PSI-VAX間でファイル転送が可能である。また、ゲートウェイを介して NTTの DDX網と接続でき、遠隔地にある LANと通信が可能である。

主な機能を次に挙げる。

・ファイル転送 (PSI-PSI間, PSI-VAX間)
・トーク (実時間通信機能)
・電子メール
・リモート・ファイル・アクセス
・リモート・プリンタ出力

これにより、大容量ディスク装置をもった PSIをファイル・サーバとして利用したり、レーザ・プリンタなどの高価なプリンタをもつ PSIをプリント・サーバとして活用できる。

1)日本語入出力

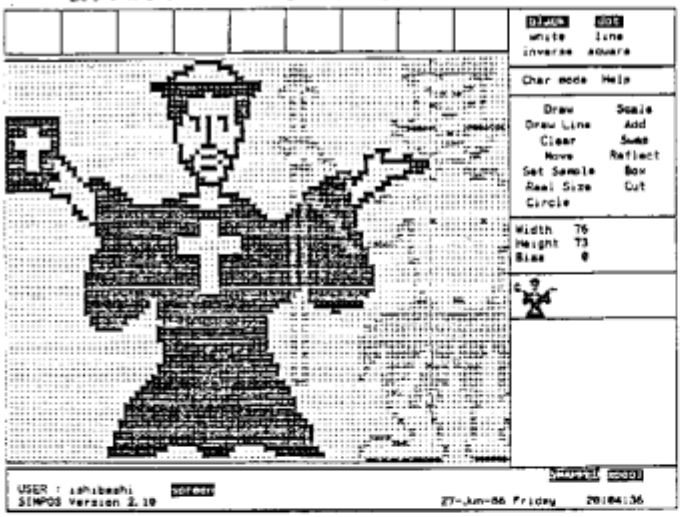
PSI は JIS 16 ビット・コードを内部コードとして採用しており、英数字と漢字, 仮名を統一的に扱うことができる。

漢字入力はカナ漢字変換機能によって行われる。SIMPOSでは、 Emacsにおけるカナ漢字変換機能以外にキーボードに直結したカナ漢字変換機能がある。これにより、 SIMPOS のあらゆるプログラムに対してカナ漢字データをキーボードから直接入力することが可能である。

(12)フォント・エディタ

フォント・エディタは表示用フォントのドット・パターンを編集するためのものである。 現在システムで用意しているフォントには、以下のものがある。

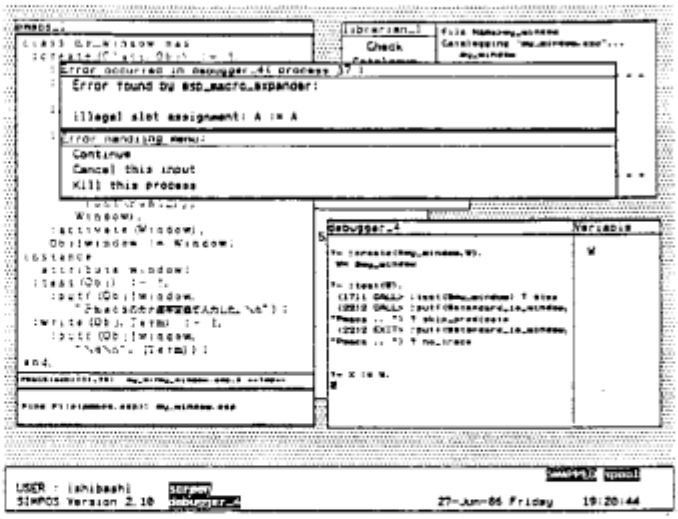
英数記号フォント	font-13, font-11, font-10, font-7
漢字フォント	kanji-16, kanji-28



(13)シチュエーション

エラー処理やヘルプ機能を例外事象として、システム・プログラムからユーザ・プログラムに至るまで統一的な枠組みを提供するのがシチュエーションである。

これにより、同じ例外事象であっても発生する状況 (シチュエーション) により、その処理を変えることができる。通常エラーが起こると、起こった状況に応じて対処可能な回復方法がメニュー表示されるので、ユーザはメニュー選択だけでエラー回復できる。



5. むすび

現在、PSI は ICOT , メーカー, 大学などを含め 60 台以上が稼動中であり、それらは LANでつながりつつある。また、 SIMPOS は第3版を作成中である。今後、より使い易いシステムにしていくためには、ユーザからのフィードバックも含め、一層の改良/拡張が必要である。

6. 参考文献

近山他：逐次型 Prolog マシンのシステム記述言語 ESP
The Logic Programming Conference'86 チュートリアル資料, 1986.6

服部他： SIMPOS のオペレーティング・システム
情報処理学会第29回全国大会, pp.285-304, 1984.9

黒川他： SIMPOS のプログラミング・システム
情報処理学会第30回全国大会, pp.295-316, 1985.3

近山他： SIMPOS のプログラミング環境
情報処理学会第33回全国大会, 1986.10 (予定)

石橋他： SIMPOS - ユーザ・インタフェイスと開発過程 -
情報処理学会コンピュータ・システム・シンポジウム,
pp.107-113, 1985.12