TM-0143

SIMPOS（逐次型推論マシン PSI の OS）
ユーザ・インタフェイスと開発過程――

石橋弘義、近山　隆、高木茂行
吉田かおる、佐藤裕幸、佐藤正俊
金枝土敦史、内田俊一

November, 1985

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191～5
Telex ICOT J32964

ＳＩＭＰＯＳ（逐次型推論マシンPSI のOS）
－　ユーザ・インタフェイスと開発過程　－

石畑弘義，近山隆，高木浪行，吉田かおる，佐藤裕幸，
佐藤正茂，金坂上敦史，内田洋一
（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.の）用のオペレーティング・システムである。これは、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・システム部とから成る。

本報告では SIMPOS の概要，ESP を用いたオブジェクト指向プログラミング，およびその開発過程について述べる。

2．SIMPOS 概要

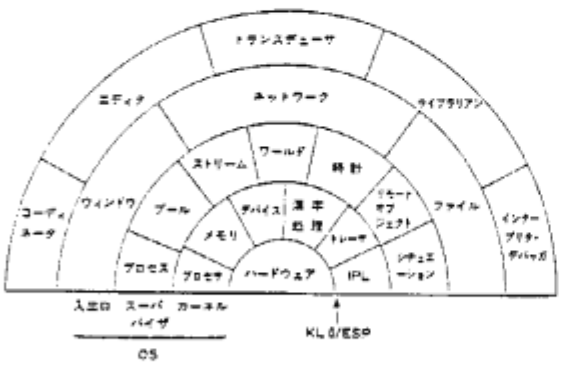
PSI は論理型プログラムのための良好な開発環境を提供するいわゆるスーパー・パーソナル・コンピュータ・システムである。その処理速度は汎用大型コンピュータ上のプロローグの処理系（DEC10-Prolog）と同等の約３０K LIPS(Logical Inference Per Second)を実現し、実メモリ容量は６４個の１６Ｍ語の実装が可能である。これにより、実用規模の大きなプログラムの実行にも耐えうる計算機システムとなっている。

(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実現する
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）に基き、簡潔で一貫性をもち柔軟性の高いシステム構築を計る

(2) システム構成



①資源管理プログラム（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。
プロセサ管理：マルチプロセス環境を実現する機能を提供する
メモリ管理：メモリの割当てやアドレス空間の管理などを行う
デバイス管理：接続される各入出力装置を管理，制御する
ＩＰＬ(Initial Program Loader)：システム本体のローディングや初期化を行う

②実行管理プログラム（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。
プロセス管理：プログラムの実行主体であるプロセスを管理，制御する機能を提供する
アール管理：オブジェクトを格納する種々の容器としての機能を提供する
ストリーム管理：プロセス間通信の機能を提供する
ワールド管理：プロセスの実行環境としてディレクトリ機能を提供する
時計管理：日付，時計，アラーム，ストップ・ウォッチなどの機能を提供する
リモート・オブジェクト管理：LANで接続された他のPSI 上のオブジェクトを操作する機能を提供する
漢字処理機能：カナ漢字変換による漢字の入力機能を提供する
システム・トレーサ：システム開発専用のデバッグ機能を提供する
シチュエーション：エラーなどの後を情報によって柔軟に後処する機構を提供する

③入出力メディア・システム
　PSIと外界との間の通信路を提供する。
ウインドウ・システム：マウスやマルチ・ウインドウの機能などを提供する
ファイル・システム：外部記憶装置への格納・取出し機能を提供する
ネットワーク・システム：他のマシンとのデータ転送機能を提供する

④プログラミング・システム
　ユーザが直接操作するプログラム開発支援ツール
コーディネータ：利用者との対話的なインタフェイスを支援する
エディタ：プログラムやデータの作成・修正を行う
トランスデューサ：プログラムやデータの文字列表現と構造表現との間の変換を行う
ライブラリアン：プログラムをコンパイルしたり、コードやオブジェクトの管理を行う
インタプリタ／デバッガ：プログラムの実行やデバッグのための各種機能を提供する

⑤マニピュレータ
　SIMPOS 各種ユーティリティの機能をメニュー選択という簡便な方法で提供する。

ウインドウ・マニピュレータ： ウインドウの位置やサイズの変更などのユーザ・インタフェイスを提供する
ファイル・マニピュレータ： ファイル名一覧の表示，ファイルのコピーや削除などのユーザ・インタフェイスを提供する
ネットワーク・マニピュレータ： ファイル転送や電子メールなどのユーザ・インタフェイスを提供する
ホワイトボード・マニピュレータ： コーディネータが提供するホワイトボードの参照，消去，コピーなどのユーザ・インタフェイスを提供する

3．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。

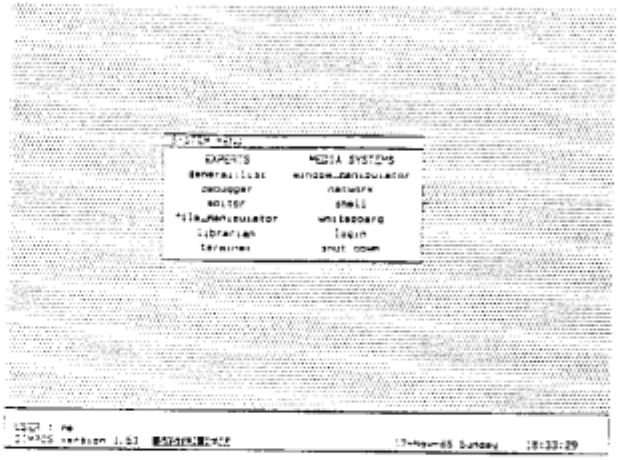
初期化には以下のものがある。

・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
　デフォールトに，ne:（>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，その他自由なことができる。

(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2種がある。EXPERTSは各々複数個並行して使用でき，MEDIA SYSTEMは各々1つしか作れない。これら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，実行の手順

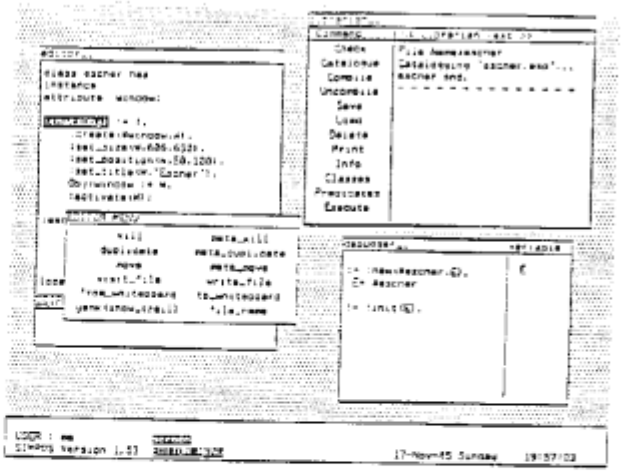
ESP プログラムの作成，実行，デバッグは，基本的には以下のことの繰返してある。

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。

編集には，文字単位で編集する通常の文字列編集モードと，編集対象の構文を理解して編集する構造編集モードとがある。トランスデューサの機能により構文規則はユーザが定義でき，ESP プログラム以外の構造編集にも利用できる。また，ウインドウ上でのテキストの表示場所や編集操作の対象範囲（ターゲット）はマウスで指示できる。



(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，インタプリティブ・コードとマシン・コードの2種類ある。デバッグ中のクラスはインタプリティブ・コードを，デバッグの完了したクラスはマシン・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。

SIMPOS で扱う全てのクラスはライブラリが管理する。このユーザ・インタフェイスとしてライブラリアンがある。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。

ライブラリは各クラスの3つの形式を管理する。

①ソース
　ESPプログラムのソース・テキスト

②テンプレート
　継承するものを含めないクラス単体に関する情報
　具体的には、スロット情報，メソッドやローカル述語のコード、および継承クラス名，参照クラス名である。メソッドやローカル述語の2種類のコードはここで管理している。

③オブジェクト
　継承関係の解決を終えた実行可能な形式

あるクラスのオブジェクトを作る時のライブラリの動作としては、次の手順となる。

・まず、そのクラスのテンプレートを作る
・次に、そのクラスが継承しているクラスのテンプレートを用いて継承関係の解決（継承解析）をする
・最後に、実行可能なオブジェクトを作る

☆関連クラスのオート・ロード機能

一つのクラスを実行するためには、そのクラスのみではなく、そのクラスが直接／間接に使用するクラス，つまり、そのクラスの継承クラスと参照クラスを全てロードしなくてはならない。これらは、ライブラリアンで自動的に行われる。

☆継承に係わる変更

オブジェクト指向呼出しの高速化のため、クラスの継承解析は実行時に静的に行う。これにより、以下のようなクラスの修正をしたときは継承情報が変るので、そのクラスを継承している全てのクラスを作り直さなければならない。

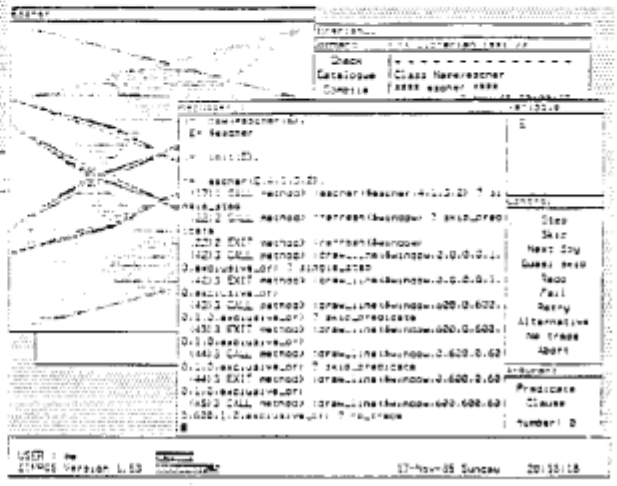
・メソッド（デモンも含む），スロットの追加／削除
・メソッドの内容の（ある種の）変更
・スロットの初期値の変更

例えば、ローカル述語に関する修正の場合は上の条件に当てはまらないので、そのクラスのみを登録し直せばよい。また、変更したクラスを参照だけしているクラスは、どの様な変更に対しても登録し直す必要はない。

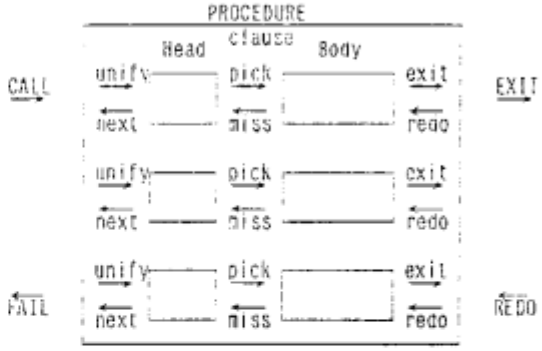
(7) デバッガ／インタプリタ

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リスナの動きもする。

デバッガのトップレベルからは、ライブラリに登録されているメソッドや ESPの構造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インタプリティブ・コードであってもマシン・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インタプリティブ・コードに限られる。



SIMPOS における ESPのデバッグのための実行モデルとして、述語単位の操作に加えてクローズ単位の操作のできる P&Cモデル（Procedure and Clause Box Control Flow Model ）を使用している。例えば、従来述語呼出しが失敗したということしかわからなかったものが、このモデルを使用することにより、ヘッドのユニフィケーションで失敗したのか、あるいはボディの実行で失敗したのかをクローズも含めてわかるようになった。



Procedure & Clause Box Control Flow Model

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。これ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

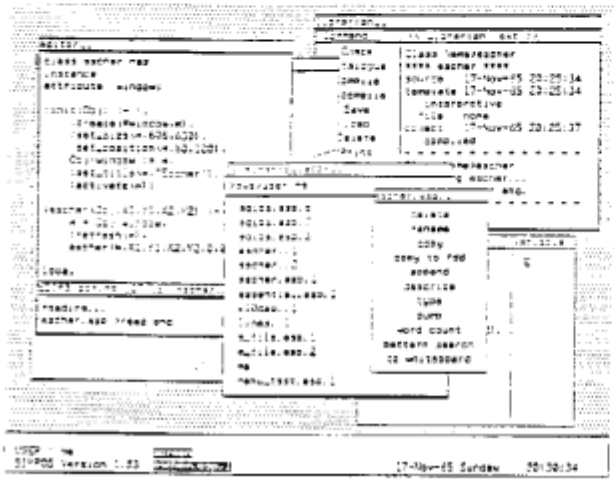
ディスプレイ上に表示されているウインドウの位置やサイズなどを変更することができる。その他、ウインドウの状態や属性も変えることができる。

(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。ファイルはワールドと呼ばれるディレクトリに格納され、パス名でアクセスされる。

主な機能を次に挙げる。

・ディレクトリの生成
・ファイル名一覧を表示する
・ファイルの消去，コピーを行う
・ディスクとフロッピ間のファイル・コピーを行う



(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同あるいは PSI-VAX間と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。

現在まだ実現していないが、以下の機能を予定している。

・ファイル転送（ PSI-PSI間， PSI-VAX間）
・漢字メール
・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・プリンタなどの高品位プリンタをもつ PSIをプリント・サーバとして活用できる。また、電子メール用に必ず PSIを1台メール・サーバにする必要がある。

(11)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、カナ漢字変換機能はエディタとは別になっており、キーボードに（ソフト的に）内蔵する方式となっている。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。

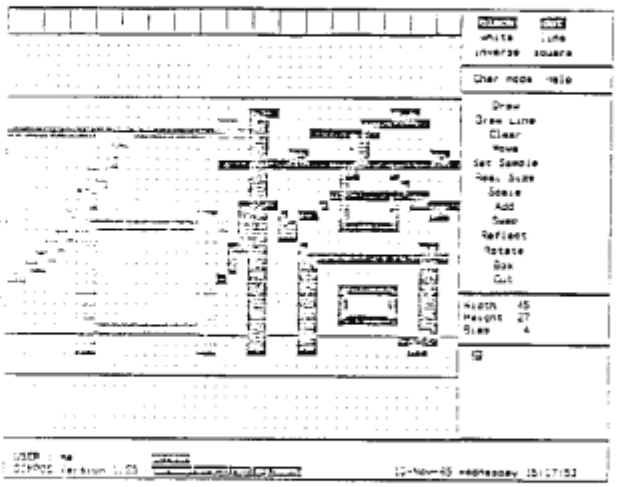
カナ漢字データをウインドウに表示する際は、ウインドウのフォントを漢字フォントに変えればよい。

(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
漢字フォント　kanji-16, kanji-28



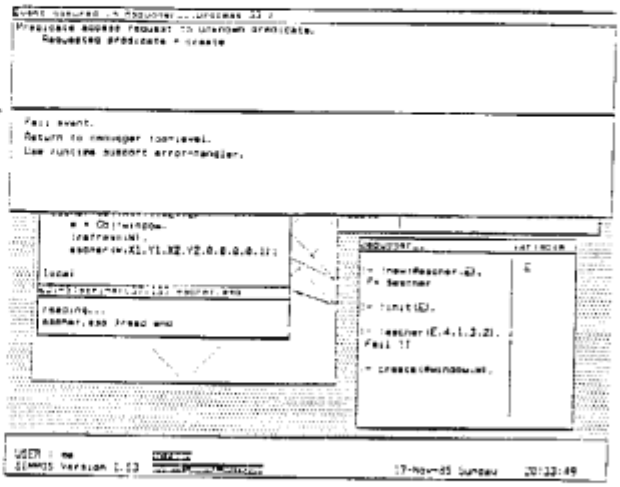
(13)シチュエーション

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。

例外事象は ESPのクラスにより分類される。その継承の頂点にくるのが eventで、そのサブクラスとしてエラーやヘルプなどを用意する。さらに、その他細かく継承により階層づけられて実現される。警告，通常エラー，致命的エラーなども多重継承により分類される。

```
event
  — malfunction
      — error  (library-error, interpreter-error....)
      — internal-error (バグ)
  — help
      throw
```

シチュエーションは、例外事象(event) とその処理 （ handler ）のペアを要素とするスタックである。プログラムのある部分を実行する時にシチュエーションのスタックを積み、終わった時（ success, failのどちらでも）にシチュエーションを元に戻す。これにより、同じ例外事象でも発生する環境（シチュエーション）により、その処理が異なってくる。



4. ESP と SIMPOS

(1) 記述言語 ESP

SIMPOS はすべて ESPというユーザ・プログラミング言語で記述されている。ESP は 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。 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。

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）
- ＋ その他 （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）
- ＋ 拡張制御機構 （remote-cut, on-backtrack）
- ＋ ハードウェア操作用組込述語 （set-word, set-register）

(2) 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唯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 1.5版のサイズは以下の通りである。

ソース・プログラム	約145，000行
クラス総数	960個
述語総数	約 15，000個

	class	instance	local	runtime
principal	4405	5349	5349	104
before demon	6	60		
after demon	33	200		
method predicate	285	3380		
slot initiation	0	336		

SIMPOS の述語数（一部ユーザ定義を含む）

(3) SIMPOSにおけるオブジェクト指向プログラミング例

ESP の大きな特徴の一つに、親クラスを複数許す多重継承機構がある。この機能を用いれば、ユーザはシステム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、システム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、各種の機能単位の部品クラスも提供している。

以下でウインドウ・システムにおける多重継承の例を挙げる。

①基本ウインドウ・クラス（必ず必要）
- temporary-window （ マウスが外にでると消える）
- inferior-window （ プログラムで指示すると消える）

②属性クラス
- with-border, with-label, with-two-labels, with-margin

③各種機能クラス
- as-output （キャラクタ出力機能）
- as-scroll （出力字のスクロール機能）
- as-graphic （図形出力機能）
- as-input （キーボード入力機能）
- as-mouse-input （マウス・クリック入力機能）

④トランスデューサ機能クラス
- as-asp-transducer , as-standard-transducer

⑤メニュー基本機能クラス
- as-menu, as-multiple-column-menu, as-scrolling-menu

⑥メニューの部品機能クラス
- as-single-select, as-multiple-select, as-on-off

ウインドウでは、以下のような部品クラスを提供している。例えば、ウインドウなら②から④の中から，メニューなら①②③④を適当に組み合せることにより、簡単に専用のウインドウを作成することができる。また、システムでは標準品のクラス window や menu も用意している。

(4) オブジェクト指向による特徴

長所
①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インプリメント上、実行時のオーバーヘッドが大きい．
②親クラスで継承に係わる変更があると、子クラスも作り直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処理が考えられる。

5. 開発過程

(1) 開発環境

SIMPOSは、新しいマシンのための新しいソフトウェア・システムであり、開発のためのプログラミング環境が極めて重要であった。SIMPOS の開発には、汎用大型コンピュータ DEC-2060 とミニコン DEC LSI-11 が使われた。

DEC 2060 --クロス・システム

↓ ファイル転送

DEC LSI-11(SVP)

↓ フロッピ経由

PSI (CSP) --実機

開発手順は以下の通りである。

①汎用機 DEC-2060 上のエディタで ESPソース・プログラムを作成、編集する。

②汎用機上で、クロス・コンパイルおよびリンクして、LSI-11にファイル転送する。

③LSI-11上でフロッピに出力して、フロッピ経由で PSIにロードする。

④実機(PSI) 上でのデバッグは、コンソール (CSP)を介して、ESPトレーサやシステム・トレーサなどを利用して行う。

(2) ESP 処理系

ESP の処理系として作成されたものは大別して以下の三系統がある。

①ESP クロス・シミュレータ
　汎用機上でコンパイル及び実行の全てを行う

②ESP クロス・コンパイラ
　汎用機上でコンパイルすることにより実機用のコードを作成し、実機上に転送して実行する

③ESP セルフ・コンパイラ
　実機上でコンパイル及び実行の全てを行う

これらの処理系のうち、クロス上のものは Prolog で (一部 Pascal) 、実機上のものは ESPで書かれている。クロス・シミュレータは ESPプログラムを Prolog プログラムに変換し、 Prolog 処理系の機能を使って動作する。

クロス・シミュレータ

- 継承機能　…イ
- オブジェクト指向呼出し　…ロ
- マクロ展開　…ハ
- Prolog による KL0組込述語の実現
- Prolog の機能

このうち、イ，ロ，ハは基本的にクロス・コンパイラやセルフ・コンパイラでも同じインプリメンテーションであり、クロス・シミュレータやクロス・コンパイラを Prolog で書くことにより、多くの部分が流用できた。

以下でそれぞれの長短を挙げる。

①クロス・シミュレータ
- ○ 実機を使わずにデバッグが可能
- ○ 汎用機上のツール（エディタなど）が利用可能
- × メモリ・サイズが非常に小さい
- × 実機とコンパチではない

②クロス・コンパイラ
- ○ 汎用機上のツールが利用可能
- ○ ハードウェア依存部分もデバッグ可能
- × 実機のハードウェア，ファームウェアが必要
- × ファイル転送などが面倒

③セルフ・コンパイラ（ライブラリ）
- ○ プログラムの修正，再実行などが迅速
- × 実機上の環境がかなり整わないと利用できない
- × OS 部分などのデバッグには利用できない
  （例えばウインドウ・マネージャ）

これらは、開発の初期段階（ハードウェア・ファームウェアが不完全な状態）ではクロス・シミュレータを用いてクラスの単体テストを行ない、その後は主としてクロス・コンパイラを用いて実機上でデバッグを行った。セルフ・コンパイラは SIMPOS がほぼ完成段階に至って動作し得たので、主としてシステム全体の統合的な動作試験や評価に用いた。また最近では、プログラミング・システムの開発にも利用されている。

(3) デバッグ・ツール

実機上でのデバッグ・ツールとしては以下のものがある。

①CSP のデバッグ機能
　コンソール・プロセサには、メモリやレジスタの表示，変更，ブレーク・ポイントの設定など多くのコマンドが用意されている。これらはファーム・デバッグを目的としたものであるが、ソフトウェアのデバッグにもかなり利用した。

②ESP トレーサ
　ESP のメソッドの呼出しのみがトレースでき、ローカル述語の呼出しなどのトレースはできない。

③システム・トレーサ
　主に機械語レベルでのトレースに使う。これにより、コンパイル・コードのローカル述語や組込述語のトレースもできる。

④デバッガ，インタプリタ
　ライブラリ（セルフ・コンパイラ）で作成したインタプリティブ・コードのトレース等に使う。

③のデバッガはクロス・コンパイルを必要とするシステム・プログラマには不向きである。このため、クロス・コンパイラで作成したコードのデバッグには ESPトレーサやシステム・トレーサを用いる。

(4) 開発期間

昭和58年 5月に SIMPOS グループを結成してから、60年 4月に SIMPOS 第1版をリリースするまでの間、 ICOT と外注メーカーを合わせて為 30 人程度が従事した。現在も、 SIMPOS 第2版に向けて約 40 人のメンバーが作業している。

(5) リリース経過

58/ 5	SIMPOSグループ結成	…設計，コーディング開始
/12	（PSI 1 号機納入）	
/ 3	ファーム完成	…基本部分実機デバッグ開始
59/ 5	OS 立上げ版	… OS 部分実機デバッグ開始
/ 9	OS 部分	… PS 部分実機デバッグ開始
/10	国際会議デモ版	(PS 部分なし)
/12	SIMPOS 0.7版	(バグ多い)
60/ 2	0.8版	(使えるバージョン)
/ 4	1.0版	(一応の完成版)
/ 9	1.5版	(ユーザ・インタフェイス向上)
61/ 3	2.0版	(信頼性向上，機能アップ)

6. 評価，改良

SIMPOS の評価の基準はプログラミング開発ツールとしての有用性に置くべきであるが，現状においては，まずシステムの高速化が重要と考えられ，処理速度などを測定した。この結果 SIMPOS は個人用ワークステーションとして，一応の能力を達成できたと評価される。

(1) 高速化

ウインドウは，SIMPOS のユーザ・インタフェイスの基本機能であり，その実行速度はユーザが受け止めるシステム全体の速度に大きく影響する。このため，早期より実行速度の評価を開始し，その結果に基づいて改良を繰り返した。

ウインドウ・システムの実行速度は，それ自身のプログラム効率のみではなく，カーネル，スーパーバイザなど使用しているサブシステム，さらに，オブジェクト指向呼出し実行のためのランタイム・サポートや，ファームウェア全体の速度に大きく依存する。

以下は，改良前後の実行時間である。

	ソフトの改良		ファームの改良
一文字入力	2 msec	→ 120 msec	→ 21 msec
一文字出力	3 msec	→ 14 msec	→ 7 msec

この間の主な改良項目およびその効果は次の通りである。

ソフトウェアの改良

- ESP ランタイム・サポートの改良　3 倍
- ビットマップ・ハンドラの割込み処理のポーリング化
  （プロセス切替のオーバヘッドの削減）　1.5-2倍
- プロセス切替の高速化
  ハンドラの改良　3.5-4.5倍
  （ビットマップ・ハンドラとウインドウ・マネジャの一体化，
  キーボード・ハンドラとユーザ・プロセスの直結）

ファームウェアの改良

- ESP の実行サポート　3- 5倍
- ファーム自身の高速化　1.5倍

例えば，プロセス切替は当初 21 msecかかっていたのが，これらの改良で 2 msec になった。また，ESP の実行時サポートをファームウェア化することにより，メソッドの実行は 5倍程度，スロット・アクセスに関しては 10 倍もの速度を得ることができた。

以上の改良により，一文字のキーボード入力から表示終了までに要する時間は 28 msecとなった。これは，毎秒 35 文字に相当し，キーボードのエコーバック速度としては十分な値を得られた。

(2) その他

SIMPOS ではこのほか，述語毎に呼出し回数を計るコール・カウンタと呼ばれるツールがある。これを使うと，全体のうちどこに処理の比重がかかっているかを見ることができる。

実行速度のほかに，メモリ消費速度も重要な要素である。PSIではバックトラックしない限りグローバル・スタックを縮めることはできない。このため，システム・プログラマはバックトラックを意識してスタックを解放する必要がある。また，オブジェクトやストリングなどで消費されるヒープ・エリアは GC によってしか回収されないので，むやみに使ってはならない。これらのメモリ消費量の測定は必要に応じて CSPの機能を用いておこなった。

(3) ファームウェアへのフィードバック

SIMPOS が完成するに従って，ファームウェア仕様の見直しや追加も行なわれた。以下に主なものを挙げる。

- ESPの実行時サポート
- ストリング処理（サーチ，シフト）の追加　…エディタ
- on-backtrack の見直し　…シチュエーション
- 多倍長整数演算（ add-extended など）の支援…ユーティリティ

その他，SIMPOS からファームウェアへの種々のフィードバックがあった。

7. むすび

現在，PSI は ICOT，メーカー，大学などを含め 40 台近くが稼動中であり，SIMPOS は第2版を作成中である。今後，本当に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。

8. 参考文献

近山：ESP reference manual，ICOT TR-044,1984.2

渕一他：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.295-304，1984.9

渕一他：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