ICOT Technical Memorandum: TM-0079

TM-0079

逐次型推論マシンのOS

服部　隆

November, 1984

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

逐次型推論マシンのOS

（財）新世代コンピュータ技術開発機構
研究所　第三研究室
服部　隆

1. はじめに

1982年から開始された第五世代コンピュータ・プロジェクトもその前期3年間の最終年度となり、パーソナル逐次型推論マシン（PSI）がその主要成果として発表された。本稿ではこのPSIのために開発されているオペレーティング・システム（SIMPOS）について述べる。

PSI（Personal Sequential Inference Machine）は論理型プログラミングの研究のためのソフトウェア開発用ツールとしてICOTで開発された個人用ワークステーションである。このため、PSIのプロセサやメインメモリは、核言語第0版（KL0）と呼ばれるPROLOGベースの高水準な機械語を直接に効率良く実行できるように設計されている。また、ワークステーションとして十分に機能するように、ハード・ディスク、フレキシブル・ディスク、ビット・マップ・ディスプレイ、オプティカル・マウス、キーボード、およびローカル・エリア・ネットワーク（LAN）インタフェースなどを入出力装置として装備している。

SIMPOS（Sequential Inference Machine Programming and Operating System）は、PSIの機能や能力を有効利用して優れたプログラミング環境を提供することに主目標を置いている。また、その基本的設計理念として次の点が上げられる。

・論理型プログラミング
　システム全体を論理型プログラミングという一定の枠組の中でとらえて設計する。

・高度対話型システム
　マルチ・ウィンドウを利用した高度な対話機能を備えた"スーパー"パーソナル・コンピュータとなるべく設計する。

・データベース機能
　PROLOGは関係データベース・システムに融合しやすいデータベース機能を含んでいる。この機能を十分に活用すべく設計する。

・日本語処理
　コンピュータを誰にでも使えるようにするためには母国語での対話が望まれる。日本人として日本語でコンピュータを利用できるべく設計する。

これらの設計理念が現時点で全て実現されているわけではなく、それに向けての努力がなされている。

2. SIMPOSの概要

SIMPOSは、プログラミング・システム（PS）と（狭義の）オペレーティング・システム（OS）とから構成されている。

プログラミング・システムは、
- プログラムやテキストの編集機能
- プログラムの実行・デバッグ機能
- プログラムのライブラリ機能

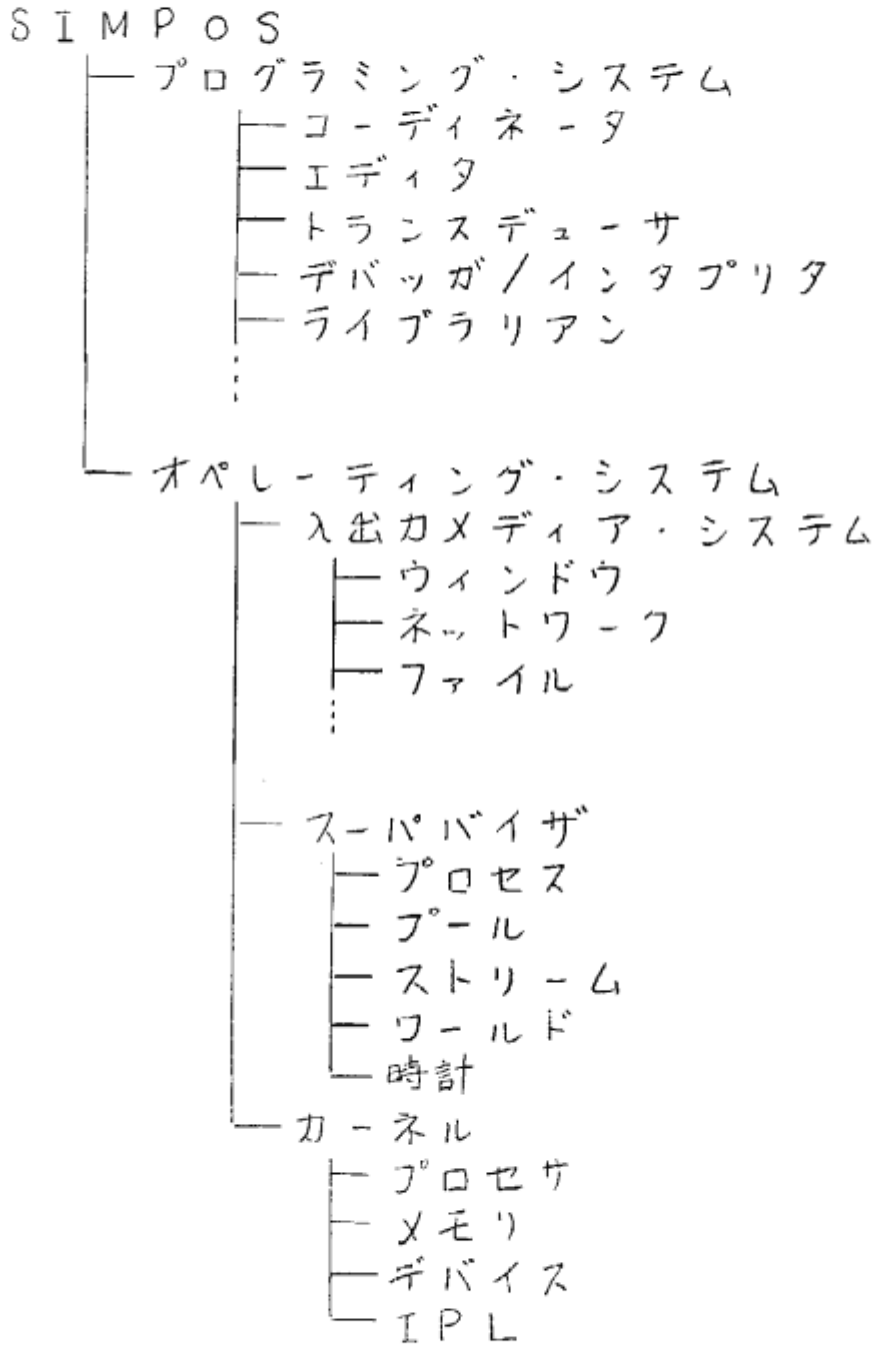
などを統合的にサポートする。PSの構成要素（サブシステム）はエキスパートと呼ばれ、プログラミング作業はこれらエキスパートとプログラマとの共同作業としてとらえられている。エキスパートとしては例えば、エディタ、デバッガなどが上げられる。

オペレーティング・システムはPSの下にあって、
- マルチ・プログラミング機能
- マルチ・ウィンドウ機能
- ファイリング機能

ネットワーク通信機能

といった諸機能を提供する。

SIMPOSの構成を示すと下の通りとなる。各要素については後述する。



3. システム記述言語ESP

オペレーティング・システムの具体的な設計にとりかかる前に，その設計に一定の枠組を与えるような基本的な構築概念が必要となる．SIMPOSの場合には，それは論理型プログラミングとそれに加味されたオブジェクト指向プログラミングであり，それを記述する手段としてのESP（Extended Self-contained Prolog）である。

SIMPOSの開発に際しては，限られたマンパワー（20～30名）とスケジュール（2～3年）ではすべての要求される機能を実現しつくすことは困難であるとの判断や，論理型プログラミングにより初めて作成されるオペレーティング・システムであることなどの理由により，システムの変更や拡張を容易にするような枠組が必須であった．我々のとったアプローチはオブジェクト指向であり，これはPROLOGに欠けており，かつオペレーティング・システムの記述のために必要な2つの機能，すなわち，

- 副作用による状態の記述
- モジュール化

を含んでいる。システムを記述する手段としての有効性はすでにSmalltalkやLISPマシンのFlavorシステムで実証されていた。

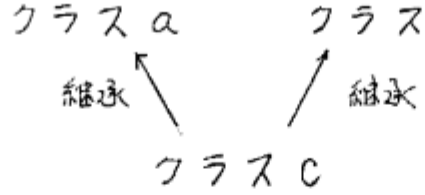
3.1 オブジェクトの概念

SIMPOSにおけるオブジェクトは，外部的にはそれに許される操作の集まりにより定義され，内部的にはクローズとスロットで定義される．ここで，クローズは操作手順を記述し，スロットは状態や内部構造を保持するために値や他のオブジェクトを格納するものである．オブジェクトを論理型プログラミングの立場から解釈すると，ひとつの公理集合を表わしていると言える．オブジェクトの状態を変えるような操作（例えば，スロットの値を書き換えるような）は，Pro-

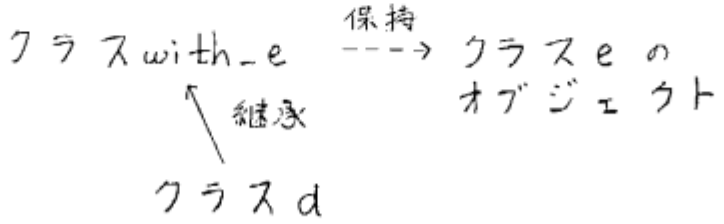
log の dsiable や retrace もの機能に対応させられる。

同じ定義のオブジェクトの集合をクラスという。クラスの考え方によれば、個々のオブジェクトを定義するのではなく、同じクラスに属すべきオブジェクトの原型（テンプレート）を与え、各オブジェクトをこのテンプレートから生成することができる。

クラスの考え方をさらに有効にするのが継承（インヘリタンス）である。継承により、すでに定義されたクラスから新しいクラスを容易に定義できる。例えばクラスaとクラスbを継承したクラスcを定義すると、このクラスcは両方のクラスの操作（言い換えれば性質）をもつようになる。さらに必要ならば、クラスc独自の操作を追加することもできる。内部的にはクラスcはこれらのクラスで与えられたフローズがOR結合され、一方スロットがすべて集められることになる。論理プログラミングの観点からは、継承されるクラスの公理集合が、継承するクラスに導入されるものと見なされる。継承により、クラス間に"is_a"の階層が形成される。



一方、クラス間の"has_a"階層は"is_a"関係を利用して形成することにした。例えば、クラスdがクラスe（正確には、クラスeのオブジェクト）をもつべき場合には、クラスeのオブジェクトをスロットにもつようなクラスwith_eをまず定義し、それをクラスdに継承させる。



たのクラスの構成要素となるようなクラスeとともにクラスdが操作された時に、それに伴なった操作をすべき場合が多い。もうん、そのような操作はクラスdの定義で記述すればよいわけだが、クラスeを持っているクラスにすべて共通な操作である場合にそれぞれの本体クラスに記述することはわずらわしい。むしろクラスwith_eにその操作を定義し、一般にそれを継承するだけにしたい。そのために機構がデーモンである。これは本体に行なわれる操作を監視しており、特定の操作の前後に決められた処理を実行するものである。内部的には、デーモン述語の呼び出しが本体の述語の前後に実行されるようにＡＮＤ結合される。

2.2　ＥＳＰの機能

ＥＳＰはＰＲＯＬＯＧをベースにし、前述のオブジェクト指向機能とマクロ機能を追加したシステム記述言語である。

ＥＳＰは次のようなシンタックスをもつ。

```
class クラス名
  〔マクロ・パンク定義〕
has
  〔継承定義；〕
  {クラス・スロット定義；}
  {クラス・クローズ定義；}
〔instance
  {インスタンス・スロット定義；}
  {インスタンス・クローズ定義；}〕
〔local
  {ローカル・クローズ定義；}〕
end.
```

このクラス定義は次の５つの部分に分けて考えることができる。

・マクロ部
　マクロ・パック定義より構成される。マクロはある構造をもつプログラムを別のプログラムに翻訳するためのメタ・プログラムであると考えられる。

・継承部
　継承定義として、このクラスが継承するクラスを自クラスも含めて列挙する。これらのクラスで定義されたクローズは指定された順にOR結合される。

・クラス部
　クラスに共通の性質をクラス・スロット定義とクラス・クローズ定義で与える。

・インスタンス部
　インスタンス・スロット定義とインスタンス・クローズ定義でオブジェクトのテンプレートを与える。

・ローカル部
　このクラス定義内だけで呼び出せるPROLOG述語を定義する。

なお、クローズの定義は通常のPROLOGの形式にほぼ準拠している。相異点は、オブジェクト指向呼び出し（クラス・クローズとインスタンス・クローズの述語）が、

:述語名（オブジェクト{，パラメタ}）

となっている点と、クローズ定義の終わりが"."でなく";"である点である。また、クローズの実行はPROLOGの場合と同じである。

2.3 オブジェクトの表現

SIMPOSのオブジェクトはKLOのデータ構造のひとつであるベクタにより表現されている。そのベクタのロケーションがオブジェクト・ポインタとして論理変数にバインドされる。このオブジェクト・ポインタの値は不変であり、PR_OLOGの実行メカニズムに違犯しない。

3. オペレーティング・システム

一般のオペレーティング・システムがマクロ命令や手続きとして提供している機能は、SIMPOSではクラス定義で記述され、そのクラスのインスタンス（オブジェクト）への操作として実現されている。したがって、ユーザが必要に応じて機能の追加や変更を容易に行なえる。

SIMPOSのオペレーティング・システムは前述のように三つの層で構成されている。

3.1 カーネル

カーネルは、ハードウェア資源を管理する部分で、最もハードウェアに近い層である。カーネルには次のものが含まれる。

- プロセサ管理
  多重プロセス環境を実現する。

- メモリ管理
  メモリの割当てと解放、さらにカーベッジ・コレクションを行なう。ただし、ファームウェアで大半のサポートが実現されている。

- デバイス管理
  入出力装置を制御する。

・IPL
　SIMPOSのブートストラップを行なう。

3.2　スーパバイザ

スーパバイザはSIMPOSにおける実行モデルを規定し、それを実現する機能を提供する部分であり、次のものから構成される。

・プロセス管理
　プロセスはプログラムを実行する。プロセス管理はプロセスの生成・終結、およびスケジュリングなどを受けもつ。
　プログラムはあるプログラム・クラスのインスタンスであり、そのメインゴールはインスタンス述語として定義される。

・プール管理
　プールは任意のオブジェクトの集まりを格納する、いわば容器である。プール中のオブジェクトは、その格納位置あるいは対応づけられたキーで参照できる。
　また、プール中のオブジェクトを順に参照するために、タップが用意されている。

・ストリーム管理
　ストリームはオブジェクトを流すパイプである。その一端に入れられたオブジェクトを他端で受取ることができる。もしストリームが空のときに受取ろうとすると、オブジェクトが入れられるまで待たされる。ストリームはプロセス間の同期および通信に利用される。
　ストリームを使って、プロセス間のメッセージ通信機能も提供される。これをチャネルという。さらにチャネルを使った双方向通信用のメッセージ・ボックスとしてポートも用意されている。

・ワールド管理

オブジェクトに名前をつけて管理できるようにディレクトリが用意されている。このディレクトリはさらに別のディレクトリを含んでもよく、それによりディレクトリの木が構成される。この木の中でオブジェクトはルートからの順路名で識別される。

ワールドは一連のディレクトリであり、プロセスがデフォルトで参照できる名前の空間を与えている。また、ユニバースはシステム唯一のディレクトリの木であり、すべてのプロセスで共有されている。

3.3 入出力メディア・システム

入出力メディア・システムは外界とのインタフェースを司る。現在これには、ウィンドウ、ファイル、ネットワークの三つのサブシステムが含まれる。

3.3.1 ウィンドウ・サブシステム

ウィンドウ・サブシステムは、ディスプレイ、キーボード、およびマウスを駆使したマルチ・ウィンドウ機能による高水準マンマシン・インタフェースを実現する。

ウィンドウを通してプロセスとユーザは交信することができる。つまり、プロセスはディスプレイに情報を表示出力し、ユーザはキーボードやマウスから指令やデータを入力する。マルチ・ウィンドウ機能は実際には一台しかない端末装置上にいくつものウィンドウを作り出すことで、ユーザとシステムとの間に並行した会話セッションを可能にするものである。

SIMPOSのウィンドウ・サブシステムでは、各ウィンドウはスクリーン上に長方形として表示される。ウィンドウは重なり合ってもよい。また、ウ

ィンドウはそのサブウィンドウを内部に含むこともできる。

さらにウィンドウには、ラベルをもつ、スクロールするなどという性質を持たせることができる。このためウィンドウ・サブシステムは基本的なウィンドウ機能をもつクラスと、それぞれの付加機能を与える補助クラスを提供しており、プログラマはそれらを選択的に継承して必要な機能をもつウィンドウ・クラスを定義できる。なお、特殊なウィンドウとして、テンポラリ・ウィンドウやメニュ・ウィンドウなども用意されている。

ウィンドウの表示、消去、移動、変形といった操作は、プログラムでそのウィンドウを操作することでできるだけでなく、ウィンドウ・マニピュレータを使ってユーザが直接指示することでもできる。

3.3.2. ファイル・サブシステム

ファイル・サブシステムはディスク・ボリューム上にデータおよびオブジェクトの格納場所を提供する。

データ（レコード）はファイルに格納される。ファイルのタイプとして、バイナリ・ファイル、固定長レコード（テーブル）ファイル、および可変長レコード（ヒープ）ファイルが用意されおり、アクセス法としてダイレクトおよびシーケンシャル・アクセスがサポートされている。また、ハッシュ・インデックス・ファイルも提供されている。

オブジェクトはインスタンス・ファイルにインスタンス・レコードとして格納される。さらにディレクトリ・ファイルの機能を使って、このインスタンス・レコードに名前を付与できる。ＳＩＭＰＯＳではファイルもオブジェクトであり、ＶＴＯＣを通称されるファイル管理領域も、インスタンス・ファイルのひとつとして実現されている。なお、ディレクトリ・ファイルはスーパバイザで提供されるディレ

クトリ木の一部として組込まれ、一様な操作でファイル中のオブジェクトも操作できるようになっている。

ファイル・サブシステムの拡張として、

- ファイル・マニピュレータ：ウィンドウ・ベースの統合化されたファイル・ユーティリティ
- バインダ：データ・ベースをサポートするためのファイルの結合機能
- リモート・ファイル・アクセス：他のＰＳＩ中のファイルをアクセスする機能

などを予定している。

3.3.3 ネットワーク・サブシステム

ネットワーク・サブシステムはネットワークで接続された他のマシンと通信する機能を次の３レベルで提供する。

まずマシン間通信機能ではＰＳＩが別のＰＳＩあるいは他種マシンと通信できる。この通信はデータの入出力として扱われる。ノード、ソケット、プラグ、およびケーブルといったクラスが用意されている。

次にプロセス間通信機能では要なるＰＳＩ上のプロセス間でメッセージ通信が行なえる。これはスーパバイザのチャネルをネットワークにまで拡張したものである。

さらにこのネットワーク・サブシステムの特徴としてリモート・オブジェクト操作の機能が提供されている。リモート・オブジェクトは他サイト上のオブジェクトを代理するもので、それを操作することで実オブジェクトを操作できる。プロセス間通信機能も実はリモート・チャネルとして実現される。

なお、ネットワークを管理するためのユーティリティとしてネットワーク・マニピュレータも用意される予定である。

4. プログラミング・システム

SIMPOSのプログラミング・システムは、エキスパートと呼ばれるサブシステム（プロセス）で構成され、各エキスパートはプログラミングの各フェーズに対応した専門知識をもち、プログラマの作業を補助するものである。これはプログラミング・システムをソフトウェア・ツールの集まりであるという見方から一歩踏み出したものといえる。

各エキスパートはユーザとの対話のためのウィンドウ（特にe_ウィンドウと呼ばれる）をもつ。ユーザはスクリーン上に表示されたウィンドウを通じてエクスパートを制御できる。したがって、ユーザの誤操作により作業を台無しにすることが少ない。

4.1 コーディネータ

SIMPOSにはトップレベルのコマンド・インタプリタのような明示的な監視プロセスはないが、代わりにコーディネータと呼ばれるバックグラウンド・プロセスがある。コーディネータはエキスパートの集合を管理し、かつ次のような機能を提供する。

- エキスパートを生成、削除する。
- キー・コマンドをエキスパートに送る。
- エキスパートからの特殊コマンドを処理する。
- エキスパート間の通信を助ける。

エキスパートの制御は、メニュ・ウィンドウやマウスクリックを使って指示できる。また、エキスパート名とそのためのプログラム名を対応づける表をユーザ毎に設定できる。

エキスパート間の通信はe_ポートと呼ばれるメッセージ・ボックスおよびホワイトボードと呼ばれるメッセージ・ボードで行なうことができる。e_ポートはあらかじめ通信し合うことがわかっているエキスパート間で利用される。一方、ホワイトボードは適宜発生するエキスパート間通信で使われる。例えば、ユーザが

あるエキスパートにメッセージをポスト・トゥ・ボードに書込むことを依頼し、次にもう一つのエキスパートにそのメッセージを読取ることを依頼することで通信が行なわれる。

4.2 デバッガ／インタプリタ

デバッガ／インタプリタはプログラムを解釈実行し、さらにその実行を制御するエキスパートである。デバッガ／インタプリタの基本的機能はDEC-10 PROLOGのデバッグ機能に準拠しているが、次のような新しい機能が付加されている。

- procedure/clause box control flow model
  DEC-10 PROLOGのデバッガは、box control flow modelに基いており、述語をデバッグ単位としている。しかし、これでは各クローズの実行はブラック・ボックス化し、ユニフィケーションが失敗しても、それがヘッドであるかボディであるかがわからない。ここで採用したモデルではそれも知ることができる。

- コンパイル済コードの呼出し
  PSIでは、インタプリティブ（解釈実行用）コードとコンパイル済コードが互いに呼出し合うことができる。しかし、デバッガはコンパイル済コードを追跡できず、組込み述語であるかのごとく実行するが、その中からインタプリティブ・コードが呼出されると再び追跡を開始できる。

デバッガ／インタプリタは、そのウィンドウの中にいくつかのサブウィンドウを持ち、追跡情報、オブジェクトのスロット値、あるいはクラス定義のソース・プログラムなどを見ることができるようになっている。

4.3 エディタ

エディタはテキストあるいはテキストで表現されるデータを編集できる汎用構造エディタとして設計されているが、特にESPプログラムの編集に便利なようになっている。このエディタの特徴として、

- 憶えやすいコマンド体系
- マクロ定義による追加コマンド
- 誤操作に対する配慮

などが上げられる。

編集対象となるテキストの構文をユーザが定義できる。汎用構造エディタではBNFを利用するものが多いが、このエディタではユーザ定義の括弧を許す演算子順位文法を採用している。これはBNFよりも単純で、かつテキスト表現により良く対応すると考えられるからである。

テキスト中のトークンは、

- アトム
- prefix演算子
- infix 演算子
- postfix 演算子
- 左括弧
- 右括弧

に分類され、各演算子は優先順位をもっている。ESPプログラムでは2～3レベルの優先順位で必要十分であり、

- 論理記号（":-", ",", ";"等）のレベル
- 関数記号（"+", "-", "*", "/"等）のレベル
- 述語記号（"<", ">", "="等）のレベル

に分けられる。

なお、文法用のパーサやプリティ・プリンタは、他のプログラムでも利用できるように、エディタとは別にトランスデューサとして作成されている。

4.4 ライブラリ

ライブラリはクラスを管理し、クラスの登録、クラス定義のロードとコンパイル、およびクラス・オブジェクトの生成などを制御する。

各クラスは、

- クラス・ソース・ファイル
- クラス・テンプレート（・ファイル）
- クラス・オブジェクト（・ファイル）

をもつ。

クラス・ソース・ファイルはプログラムのテキスト・ファイルであり、ただひとつのクラス定義を含む。

クラス・テンプレートはクラス・ソース・ファイルから作成され、継承から得られる情報を除いて、そのクラスの全情報を保持する。そのクラスの述語は、テンプレート作成時にインタプリティブ・コードとして保持されるが、ユーザの依頼によりコンパイルされ、そのコンパイル済コードも保持される。クラス・テンプレートは、クラス・テンプレート・ファイルに格納できる。

クラス・オブジェクトは、継承に従いいくつかのクラス・テンプレートから作成され、実行可能なイメージを与える。

ライブラリのもうひとつの機能は述語の管理である。述語単位での参照やコンパイルができ、オブジェクト指向呼出しやコンパイル済コードとインタプリティブ・コードの相互呼出しをサポートしている。

エキスパートとしてのライブラリアンは、このライブラリの機能を使って、ユーザのプログラム管理を支援するものとなる予定である。

5. おわりに

ＳＩＭＰＯＳは論理型プログラミングおよびオブジェクト指向プログラミングという新しい枠組の下に設計・開発された新しいオペレーティング・システムで

ある。１９８３年から本格的な設計に始め、現在（１９８４年１０月）オペレーティング・システムがほぼ稼動し、その上でプログラミング・システムのデバッグが鋭意進められている。また、１１月のＦＧＣＳ’８４ではＰＳＩ／ＳＩＭＰＯＳのデモンストレーションが予定されている。開発チームがＩＣＯＴと参加５メーカからの混成チーム（約４０名）であるのにもかかわらず比較的短期間でＳＩＭＰＯＳを開発できたのは、我々の取った新しいプログラミングの枠組が有効であったことを示していると考えられる。ＳＩＭＰＯＳの第１版リリースは１９８４年度末に予定されているが、今後広く利用され、そこからのフィードバックにより、さらに良いものとするための改良・拡張がなされるはずである。現在のＳＩＭＰＯＳはそのための基盤を与え得ると信じている。

参考文献

- 高木他，ＳＩＭＰＯＳ（Sequential Inference Machine Programming and Operating System）の概要，Proc. of the Logic Programming Conf. '84（1984）.
- 服部他，逐次型推論マシンＰＳＩのオペレーティング・システム，情報処理学会オペレーティング・システム研究会23-1（1984）.
- 服部他，ＳＩＭＰＯＳのオペレーティング・システム概要，情報処理学会第29回全国大会講演論文集4E-1（1984）.
- 横井他，論理型言語を機械語として実行する逐次型推論マシンとそのＯＳ，日経エレクトロニクス1984.11.5.