別システムからプログラムソースファイルを受信する．次に各種シェルコマンドで，受信したソースファイルの文型検索，文型置換等を行う．そしてコンパイル，デバッグ等を行い，修正したソースファイルを別システムに送信する．マルチウインドウ処理により，これら複数の処理が同時に，1つの画面上で行えることは言うまでもない．これらシェルコマンドとマルチウインドウの相乗効果は一言では表わせないが，従来のシステムと比べて，ソフトウェアの生産効率が，飛躍的に高まることは間違いない．

また，OS の各モジュールをプログラムから呼出すために，約300種類のシステムコールがある[5]．これらのシステムコールはその機能から，プロセス（計算処理環境）の制御，画面処理，ネットワークを含めたI/O 処理の3つに分類できる．各々の詳細な説明は省略するが，これらを有機的に組合わせることにより，アポロを特定分野向けのワークステーションとして活用できる[6]．

7. おわりに

以上，アポロについて，そのローカルネットワーク機能を中心に解説してきた．しかし，これもアポロ全体から見ればほんの1部の機能でしかない．例えば，ウインドウを含めた画面処理，プロセス間およびノード間通信を行うメールボックス，分散型データベース等，注目すべき機能は多い．これらは紙面の都合で詳しくふれることができなかった．

さて，これらの機能に加えて，今後のアポロ社の動向が気にかかるところである．

まずソフトウェアに関して言えば，現在，D3M 分散型データベースに最も開発費がかけられている．これまでの経過を見ても，アポロ社は，ローカルネットワークを組合わせた分散型データベースマシンの構築をめざしているという気がする．

ハードウェアに関しては，2～3年の期間を置いてコストパフォーマンスを倍にしていくとのことである．これは，ユーザに専用の CPU という方針からすればきわめて当然である．価格が安ければ安いほど，それを実現しやすくなる．

最後に，本稿を作成するにあたり，日本アポロコンピュータ(株)から各種資料を提示していただいた．ここに感謝して結びとする．

参考文献

1) 山村紀夫：スーパーパソコンアポロ DOMAIN，BIT，Vol. 15，No. 2，pp. 4-13 (1983)．
2) Jim Nelson：802：A Progress Report，DATAMATION，Vol. 29，No. 9，pp. 136-152 (1983)．
3) 石田晴久：ローカルエリアネットワークの応用，情報処理，Vol. 23，No. 12，pp. 1161-1168 (1982)．
4) 木村　泉：Software Tool とは何か？，情報処理，Vol. 20，No. 8，pp. 673-680 (1979)．
5) System Programmer's Reference Manual，Apollo Computer Inc.，p. 640 (1983)．
6) 田中善一郎，橋爪誠一：既存コンピュータの手が届かない分野をカバーするワークステーション，日経エレクトロニクス，pp. 149-172 (1983年5月23日号)．
7) DOMAIN System Command Reference Manual，Apollo Computer Inc.，p. 467 (1983)．
8) 石田晴久：UNIX，共立出版，p. 242 (1983)．
9) 木村　泉訳：ソフトウェア作法，共立出版，p. 516 (1981)．
10) APOLLO DOMAIN ARCHITECTURE，Apollo Computer Inc.，p. 18 (1982)．

（昭和58年10月14日受付）

製品例

オフィスプロフェッショナルワークステーション (JStar) と統合プログラミング環境ワークステーション (1100 SIP)†

上林憲行†† 伊東　健†† 上田良寛††

† Office Professional Workstation (JStar) and Integrated Programming Environment Workstation (1100 SIP) by Noriyuki KAMIBAYASHI, Ken ITOH and Yoshihiro UEDA (Fuji Xerox Co. Ltd.).

†† 富士ゼロックス(株)

1. ゼロックスの高機能ワークステーションとそのネットワークインテグレーション

現在，富士ゼロックスでは，オフィススタッフ用ワークステーション JStar と，プログラム開発用ワークス

テーションである 1100 SIP (Scientific Information Processor) 上で稼働する統合プログラミング環境である，Interlisp-D と Smalltalk-80 を，発表している．これらのシステムは，先進的なユーザインタフェースを完備したパーソナルコンピューティング環境を基盤として，それぞれの目的のタスクに対して適した強力な機能を提供しているのが特徴である．

また，ゼロックスのこれらのワークステーション群は，パーソナルコンピューティング環境の枠を越えて，ローカルエリアネットワーク (LAN) を介して，各種のサービスを受けられるばかりでなく，情報・リソースの共有を図ることが可能である．

1.1 ネットワークサービス

ゼロックスの提供する各種のワークステーションは空間的な広がりを意識させない，ローカルエリアネットワークシステムの基盤の上に展開される．以下に，XINS (Xerox Information Network System) のコンセプト（図-1 参照）において，ワークステーション群が享受することができる，多彩なネットワークサービスについて簡単に触れる．

◇ファイルサービス：ファイルサーバ（サービス）の役割は，ワークステーションのローカルディスクの容量を補い，バックアップ機能としても働くがユーザ間でファイルを共有することが第一義な役割である．

◇メイルサービス：メイルサーバは，ネットワーク上のメイル（情報の種類，サイズの制約はない）を受け取り，そのメイルを所定のユーザごとに振り分けをする電子ポストオフィスを実現している．このサービスは，従来の通信システムが提供していた物理的な site to site のサービスから，直接的に，User to User のサービスを実現した意味で，画期的なものである．

◇プリントサービス：プリントサーバ（サービス）は，印刷用のプロトコルとして送られてくる情報を，ラスタイメージに変換し，レーザプリンタに印刷するものである．テキストとグラフィックスが混在したドキュメントを印刷する能力はもちろん，ハーフトーン，マルチフォント，A4（縦，横），B4（縦，横）などの用紙サイズに，多彩な表現力をもった高品質な印刷が可能である．

◇クリアリングハウスサービス：クリアリングハウスサーバは，システムを構成する各種論理リソース名をネットワークユニークアドレスに写像する機能を果たすクリアリングハウス (Clearinghouse) サービスを提供するもので，電話の 104 機能に相当するものである．

◇コミュニケーションサービス：コミュニケーションサーバは，地域的に離れた（たとえば，本社と支社）Ethernet 間を接続する機能を提供するインタネットワークルーティング (Internetwork Routing) サービス，さらに，メインフレームホストとの伝送制御手順，各種 WP や PC とのコミュニケーション機能を提供するゲートウエイ (Gateway) サービスなどに責任を持つ．

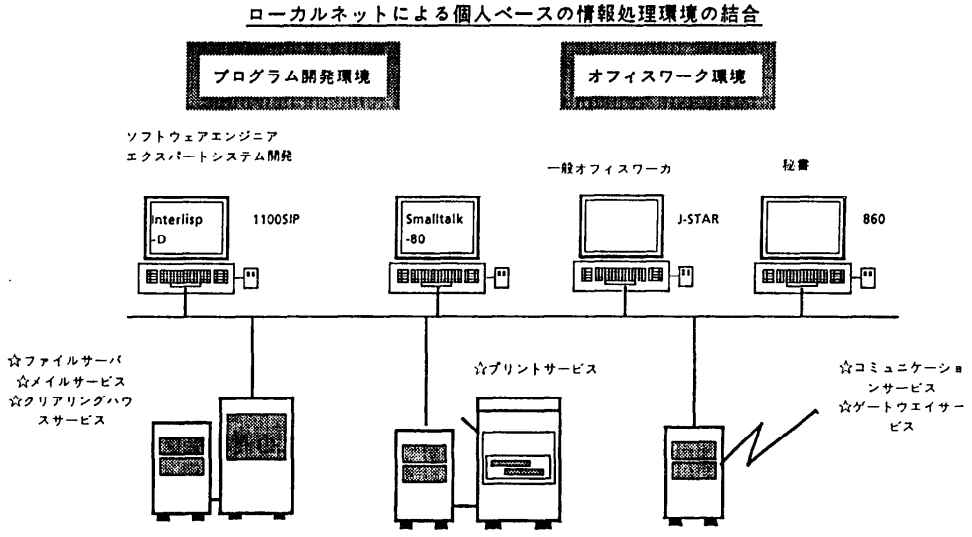

図-1 ゼロックスインフォメーションネットワークシステム (XINS)

1.2 イーサネットとゼロックスネットワークシステムズ

Ethernet は，10 Mbps の通信容量を持ち，受動素子からのみ構成されるバス方式の LAN の代表格である．Ethernet の特長は分散システムからの要件である，高信頼性，高保守性の特性と，OA からの要件である，動的チャネルアロケーションの特性を合わせもった，最も実績と評価の高い LAN である．この Ethernet は，米国の IEEE (802 委員会) 及び，欧州の ECMA 等の各標準化推進機関によって，LAN の方式の標準方式として採用が決まっている．

さらに，ゼロックスの高機能ワークステーション群を支える強力なバックボーンが，ゼロックスネットワークシステムズ (XNS: Xerox Network Systems) と命名されている，LAN 指向のネットワーク・アーキテクチャである．XNS はゼロックス分散システムのアーキテクチャ基盤を明らかにし，相互に交信可能な互換性のあるシステムを構築するのに充分な通信規約を規定するものである．XNS は，すでに一部が公開されており，Ethernet との整合性を考えると，Ethernet と同様に，デファクトスタンダードの地位を確保すると予想される．

以下に，オフィススタッフ用ワークステーションである JStar と，プログラム開発用ワークステーションである 1100 SIP 上で稼働するプログラミング環境である Interlisp-D と Smalltalk-80 の概要を述べる．

2. オフィスプロフェッショナルワークステーション—8012-J Star—

2.1 開発のバックグラウンド

8012-J Star (以下 JStar と略称する) は富士ゼロックスと米国ゼロックスとの長年にわたる共同開発の産物である (図-2 を参照)．

結果として JStar はゼロックスの Star に日本語処理機能を付加した形に見えるが，内容的には決して単純な付加ではなく，Star そのものがマルチナショナルなワークステーション体裁を備えている．その一環として日本語入出力・編集処理を有しているのが JStar である．Star の開発と並行して日本語入出力処理および編集処理の研究開発がやはり日米ゼロックスの共同で進行し，そこで得られた成果を基に，製品開発としての Star プロジェクトと一体化する形で JStar 開発が進行した．

Star の開発当初は多言語処理機能は特に考えられていなかったが，JStar プロジェクトが本格化するにつれ，Star の基本仕様やソフトウェアそのものが陰に陽に日本語処理に必要な諸々の機能の影響を受け，最終的には広範な各国言語を処理するに適したフレームワークが進んで現在のようなスペックとソフトウェアの体系になった．

現在，日本，米国，欧州各国で販売されている (J)Star は，データファイル (フォント等) の内容に相違はあるものの，通信機能を含めたすべてのソフトウェアは相互に 100% の互換性が保たれている．この意味

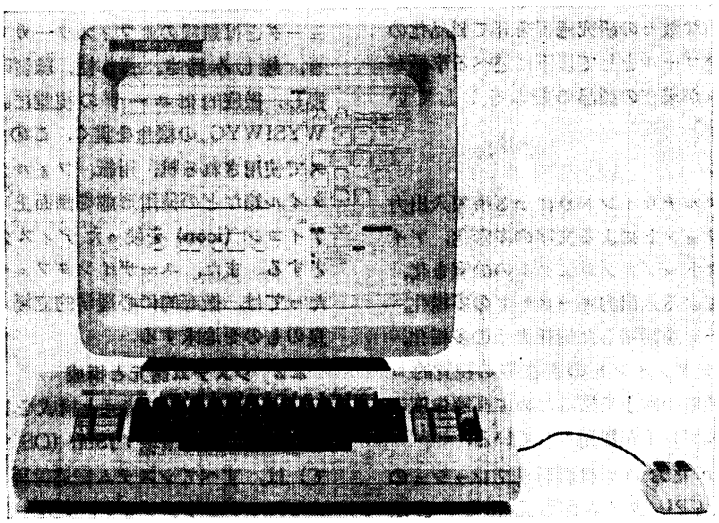

図-2 JStar 画面の例とキーボード，マウス

するところは大きく，Ethernet による OA の構築が単に一構内の一言語システムにとどまらず，外国も含めた広範囲な地域と多言語システムとが渾然一体となった OA 化ができることである．

ゼロックス社のパロアルト研究所（PARC）では，1973年に独自開発した Alto と呼ばれるビットマップディスプレイ方式のパーソナルコンピュータにマウスと呼ばれる革新的なポインティングデバイスを使用し，実験的な Ethernet に接続して一人一人の研究者に配分してあらゆる角度から将来の OIS (Office Information System) に必要な基礎技術と応用技術の大規模な研究が開始された．

特に現在の JStar を中心にした 8000 INS に密接に関係するものとしては，各種のサイズと多様なタイプフェイスを有した文字を自由に扱えるワードプロセッシング，頁ディスプレイ方式による“What you see is what you get (WYSIWYG)”の概念の具現化，グラフィックスやイメージの入出力および編集処理，マルチウインドウやコマンドメニュ方式によるより良いユーザインタフェースの心理学的立場からの追求，個人間あるいはグループ間電子メイル方式の高機能化，ユーザ間で情報をシェアできるファイリング機能の高機能化，あらゆるイメージのプリンティングを可能にするレーザプリンティング方式，頁上の文字情報やイメージ情報をプリンタインディペンデントに表現するためのデータ形式，Ethernet 内通信や Ethernet 間通信などの低位から高位に至るまでのあらゆるプロトコル等々がある．

以上述べた基礎的な数々の研究過程を経て製品化の模索が始まり，開発ゴールとして以下に述べる事柄が含まれており，それが現在の製品の根本をなしている．

[ハードウェア]

◇ディスプレイ：マルチウインドウによる情報入出力の多角化，マルチフォントによる文字の多彩化，アイコンによるディスクトップオブジェクトの絵解き化，仮想キーボードによる入出力キーボードの多様化，ハーフトーンイメージを活用した情報表示の多様化，グラフィックスとドキュメントの混在等の視覚的コミュニケーション機能の向上を図るために高解像度のビットマップディスプレイが採用されている．

◇プロセッサ：Alto においては画面リフレッシュのために最大 60% の CPU タイムが取られていたが JStarにおいてはアーキテクチャ上の工夫により CPU への負担を大幅に減少させる．また，Alto と同様に書き換え可能なマイクロコード (Mesa エミュレータ) によって制御する．内蔵するディスクは信頼性と容量を考慮して固定ディスクとし，ユーザデータのオフラインによる保存や転送を可能にするためフロッピディスクを搭載する．

◇マウス：マウスは繊細かつスピーディなカーソルコントロールとマウス上のボタン（スイッチ）操作によって極めて扱い易いユーザインタフェースの具現化ができるポインティングデバイスである．JStar のマウスの動作原理は Alto のマウスと同じであるが，信頼性と操作性を向上させるために再設計し，数々の実験結果から，ユーザの心理的負担を軽減しユーザインタフェースを向上させるために，ボタンの数は Alto での三つから二つに減らす．

◇LAN：国際標準の 10 Mbits/sec の Ethernet と XNS に準拠する高水準プロトコルを採用．

[ソフトウェア]

◇体系：JStar の特長は，ワードプロセッシング，数式処理，テーブル処理，グラフィックス，レコードプロセッシング，ターミナルエミュレーション，ファイリング，メイリング，プリンティング等々の諸々の機能を一つのパッケージに納めたことである．こうすることにより，一つの機能を実行中に他の機能も並行して実行させたり，面倒なソフトウェアの再ロードやマニュアルによる切り替えを行うことなくデータの相互利用を実行できるようにする．

◇ユーザインタフェース：ユーザがあくまでコンピュータとは無縁のオフィスワーカである事を念頭に置き，親しみ易さ，操作性，練習時間の最小化を重要視し，徹底的にユーザの視覚に訴えるとともに，WYSIWYG の概念を貫く．このため，一般のオフィスで使用される机，用紙，フォルダ，キャビネット，メイル箱などの使用形態を画面上にシミュレートし，アイコン (icon) を使ったディスクトップ思想を基本とする．また，ユーザインタフェース細部の決定にあたっては，徹底的に心理学的立場から実験を行い，最良のものを追求する．

2.2 システム諸元と構成

JStar のシステム諸元と構成を表-1 に記す．

◇システム開発環境：JStar (OS である Pilot を含めて）は，すべてシステム記述言語 Mesa (協調型のプロセスインタラクション，コルーチン，モニタ，エクセプションハンドリング等のメカニズムが内包）で

表-1 JStar のシステム諸元

システム諸元	仕　　様
キーボード	JIS 準拠カナキーボード+マルチ機能キー (24 種)
スクリーン	17 インチビットマップディスプレイ (1024×808 ドット)
ポインティングデバイス	2 ボタンマウス
プロセッサ (仮想チャネルプロセッサ)	CPU (AM 2901 A×4), マイクロプログラム制御, 16 ビット／ワード, シンクロナス・マルチタスキング方式, I/O プロセッサ (i 8085 A), バイトコード & フレームアーキテクチャ
メインメモリ&メモリ管理	384 k ワード, 仮想記憶方式 (8 メガバイト空間)
2 次記憶システム	固定ディスクウインチェスタ型 (29 メガバイト), 8 インチフロッピディスク 1 セット, ファイルサーバ
オペレーティングシステム	Pilot (パーソナルコンピューティング環境用 OS カーネル)
ローカルネットワーク	Ethernet & Xerox Network Systems

コーディングがなされている. JStar は, Mesa プログラムよりなるが, Mesa ではプログラム間インタフェース定義機能 (インプリメンテーションボディとの明確な分離) があり, JStar ソフトウェアの生産性, 信頼性, 保守性に大きく寄与している. 8000 INS の開発には強力なプログラミングツールを搭載した Mesa ソフトウェア開発ワークステーションと LAN によるシステム開発環境により, ソフトウェアの生産性を大幅に向上させた.

◇基本アーキテクチャとその能力：プロセッサまわりのアーキテクチャについては, プロセッサの能力のバランスのとれた配分を目的としたマイクロタスキング方式以外は特別な工夫がない. Mesa エミュレータアーキテクチャの特長は, バイトコード (コード／データのコンパクト化), 評価スタックアーキテクチャ, フレームベースコントロールトランスファ等である. 基本的には, ベースマシンレベルでは汎用性と高速性を重視し, ある機能を特殊用途のハードウェア機構で実現するよりは, マイクロプログラムまたはソフトウェアによって様々な機能をソフトコントロールしている. JStar のベースマシンの能力は浮動少数点演算を除けば, VAX 11/780 の約 1/2 の能力があると報告されている.

2.3 機能と能力の特徴

JStar の機能の概要を表-2 に記す.

以上のように JStar の個々の機能は多岐にわたるが, 全体の特徴としては,

◇タスクの連続性と並行性：各機能が他の機能の継続をターミネートすることなく実行できる. たとえば, ワードプロセッシングやグラフィックエディティングを実行しているウインドウを開いたまま, 電子メイルを受信して内容を読み, 返信の中に現在編集中の文章やグラフィックスを転記して送ったり, ホストコンピュータの計算結果をエミュレータウインドウを使って読み出し, 編集中のウインドウに転記したりできる. また, 作業途中でファイルドロワを開け, 他のドキュメントを捜してディスクトップに置き, それを開いて内容の一部を編集中のドキュメントに転記したりできる. こういった作業の流れは日常のオフィスでよくある流れであり, オフィスの机の上をシミュレートしたディスクトップを画面に表示する方式やアイコンに代表される図形的な表示方式とあいまって, 計算機とは無縁の人達が違和感なく溶け込めると同時に, 効率的な作業ができる.

◇モードレス・インタラクションの実現：日本語ワードプロセッシング, 英文ワードプロセッシング, その他欧州言語ワードプロセッシング, 数式処理, グラフィックス処理, テーブル処理などがそれぞれに最適化された形でソフトウェアが統合化されており, 一つあるいは複数のウインドウ内でそれらを自由に組み合わせながら仕上がりが印刷並みに美しいドキュメント作りができる.

◇ユーザコンセプショナルモデルの提示：WISYWIG の概念に基づき, 画面にはプリントされる時と同一のレイアウトで表示されるため, 従来のシステムで意外と時間のかかるレイアウトの修正と最終化が効率よく実行できる. このレイアウト修正には, 行間隔やパラグラフ上下左右マージンの微調機能, それにページ上下の見出しなどの機能も欠かせぬ要素となっている.

◇オブジェクト指向ユーザインタフェース：文字サイズや種類の変更など, 各種パラメータの変更はプロパティシートと呼んでいる新しい概念のウインドウを必要時にのみ表示させることにより, 機能の豊富さが逆にユーザにとって煩雑で使いづらいシステムにならないようにしてある. 高度なシステムにありがちな弊害として, 機能のすべてを一通り習得しないと簡単な作業さえもできなくなることがあるが, JStar では各種の機能を必要な時に必要なだけ習得すればよいようにユーザインタフェースが設計されている.

表-2 JStar の機能の一覧

基本機能	機 能 の 概 要
ユーザインタフェース	ディスクトップ，アイコン，最大6個のマルチウインドウ，上下左右ウインドウスクロール，ウインドウサイズ変更，プロパティシート，コマンドメニュ，仮想キーボード
文字種	JIS 第1第2水準漢字，ひらがな，カタカナ，数字，英文字，フランス文字，ドイツ文字，ラテン文字，ギリシャ文字，数学記号，論理記号，特殊文字
文字字体	日本語は明朝とゴシックの2種類，アルファベットはクラシック（明朝体に準拠）とモダン（ゴシック体に準拠）および14種類のプリントホイール字体の計16種類についてノーマル，ボールド，イタリック，ボールドイタリックの4種類のタイプフェース
文字サイズ	8, 10, 12, 14, 18, 24 ポイント（一部制限あり）
文字表示位置	肩文字，足文字，肩肩文字，肩足文字，足足文字，足肩文字
仮名入力	ひらがなモードまたはカタカナモードにおける仮名直接入力，ローマ字仮名変換入力
漢字入力	かな漢字変換方式，文法処理，仮想鍵盤による同音語最大8個の同時表示，JIS コード入力可
辞書	一般単語，人名，地名，会社名，その他専門用語等で合計約8万9千語，ユーザ単語登録（約2千語）
入力モード	ひらがな，カタカナ，英数（以上鍵盤からの直接モード切り替え），特殊，事務，記号，数学，ギリシャ語，仏語，独語，露語，欧州，論理（以上仮想鍵盤による間接モード切り替え）
グラフィックス	点，直線，曲線，円，楕円，三角形，四角形，棒グラフ等．いずれも，サイズ，線幅，タイプ，シェーディングが可変．任意の図形エレメントの合成，分割，拡大，縮小機能
テーブル	自動作表，編集，棒グラフ作成，演算，分割行／列作成
書式指定	用紙サイズ指定（A4, B4, B5 各縦横サイズ，その他），ページマージン指定，複数コラム指定，上下見出し，ページ番号付け
編集機能	選択，選択調整，挿入，削除，転記，移動，センタリング，アンダライン，文字サイズ変更，字体変更，探索置換，タブ，インデント，行間隔指定，ジャスティファイ，ページジャンプ，図形合成／分割／拡大／縮小等
ファイル管理	多重ファイル構造，変更日／変更者自動記録，ディスクトップ機密保持機能，リファレンスアイコン
プリント	レーザプリンティング（12本/mm），キューイング機能等
電子メイル	複数個人宛同報メイル，受信人チェック，返信／転送機能等
その他	フィールド処理／演算，レコードプロセッシング，IBM 3270 エミュレーション，TTY エミュレーション等

2.4 応用と効果

JStar の応用は，ワードプロセッシングやグラフィックス処理やテーブル処理・数式処理の組み合わせによる美しいドキュメント作成に始まって多岐にわたるが，大きな効果はやはり Ethernet による LAN の中に位置付けられていることによるところが大きい．スタンドアローン型の機器と異なり，遠隔地の Ethernet 同志が Internetwork Routing の機能によって結合されることもあって，地域的な広がりを有した組織的なドキュメントプロセッシング業務を非常に効率よく実現できる．

つまり，電子メイルやリモートファイリング，ドキュメントの共有化，リモートプリンティングの機能によって地域ギャップが解消できる．特に電子メイルは1対多数の同時コミュニケーションを可能とし，極めて効果的な時間利用を組織的に図れる．加えて，ホスト端末のエミュレーション機能や TTY エミュレーション機能を使用するとオフィスオートメーションのかなりの領域がカバーされる．スタンドアローン型のワードプロセッサの普及により，すでにホワイトカラーの文書作成・編集効率が上がってきているが，さらに機能を大幅に高め，それらの機器をネットワーク化する事によってオフィス業務の効率の向上を具現化したのが JStar システムと言える．

米国においてはすでに数千台がユーザ先で稼働しており，Star-Ethernet による OA 化がどんどん進行している．最近では一つの官庁だけで数千台の Star ワークステーション導入が決定しており，Ethernet ベースのオフィス革命が官庁民間を問わず浸透してきた．日本においてもすでに数十のネットワークをベースに多数の JStar ワークステーションが官庁民間のユーザ先で OA 化に貢献している．ヨーロッパにおいても同様な状況にある．我が国における JStar-Ethernet に基づいたオフィス革命も急速に進行するであろう．

3. 統合プログラミング環境ワークステーション
1100 SIP—Interlisp-D と Smalltalk-80—

統合プログラミング環境ワークステーション 1100 SIP には，人工知能システム開発用プログラミング環境である Interlisp-D (Interlisp-D マシン) とオブジェクト指向型言語である Smalltalk-80 プログラミング環境 (Smalltalk-80 マシン) の2つの先進的なパーソナル & パワフルコンピューティング環境が動作し，各環境ごとに，マイクロプログラムレベルで最適化が

なされている.

3.1 1100 SIP のシステム構成と諸元

1100 SIP は，プロセッサ本体（メモリ，固定ディスク，I/O インタフェース内蔵），ディスプレイ，キーボード，マウスの4点から構成される（**図-3** 参照）．1100 SIP のベースマシンとしてのシステム諸元を**表-3** に示す.

3.2 人工知能システム開発用プログラミング環境 —Interlisp-D—

3.2.1 Interlisp-D の開発の背景

記号処理用言語 LISP は，人工知能，知識ベース・システムの分野では不可欠なものとなっている．Bolt, Beranek & Newman 社 (BBN) とゼロックスのパロアルト研究所 (PARC) によって開発された Interlisp は，LISP のなかでも特に強力なものの1つであり，欧米の大学・研究所において大きな Interlisp コミュニティが成立している.

Interlisp は，強力かつ大規模なシステムで，動かすためには大きなメモリ空間を必要とする．また，対話的に使われる必要があるため，大型計算機の TSS 環境で発展してきた．しかし，ユーザが増えて負荷が重くなってくると，応答が遅くなりユーザに対して十分な応答性を提供できなくなってきた.

こうした問題を根本的に解決するために，TSS ベースシステムから決別してパーソナル・マシンにすること，および，それに LISP 専用のマシン・アーキテクチャをもたせて実行効率の向上を図ることが必要になった．このような背景のもとに開発されたのが 1100 SIP/Interlisp-D マシンである.

Interlisp-D は，1100 SIP のようなユーザインタフェースに優れたパーソナル環境に合うように，Interlisp にさらに改良を加えたもので，Interlisp に対しては上位互換性がある.

Interlisp-D は次のような特徴をもっている.

1) LISP 向きアーキテクチャによる処理効率の向上 2) ビットマップ・ディスプレイとマウスを用いた洗練化されたユーザインタフェースの完備 3) ローカルエリアネットワーク（イーサネット）のサポート 4) 強力なプログラミング環境

以下，それぞれの特徴について述べる.

1) Interlisp-D アーキテクチャ

Interlisp-D では，処理能力向上のために以下に述べるアーキテクチャ的工夫がなされている.

◇バイト・コード命令セット：各命令長は原則1バイト長にコンパクト化する．この方式を採用することによって，プログラムのコードの量を極力減らし，2次記憶とのスワッピング・オーバヘッドと命令のデコード時間の両者を大幅に削減することができる.

◇CDR エンコーディング：CDR 部が次のアドレスにあるセルの場合，単に次のセルが CDR 部であることを示す“タグ”だけを書いておく方法．これは，データのコンパクト化を図るテクニックであるが，仮想記憶空間においてはデータの局所性を高め，スワップの

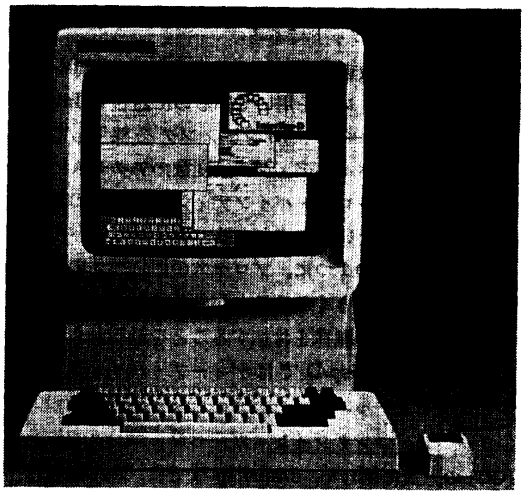

図-3 富士ゼロックス 1100 サイエンティフィック・インフォメーション・プロセッサ

表-3 1100 SIP のシステム（ハードウェア）諸元

システム諸元	仕　　様
キーボード	Alto タイプ ASCII キーボード
スクリーン	17 インチビットマップディスプレイ (1024×808 ドット)
ポインティングデバイス	3 ボタンマウス
プロセッサ（仮想チャネルプロセッサ）	ビットスライス型マイクロプログラム制御プロセッサ，16 ビット／ワード，ダイナミックマイクロタスキング方式，WCS，200 ナノ秒のマイクロ命令サイクル時間
メインメモリ＆メモリ管理	1.15—2.0 メガバイトメインメモリ，仮想記憶方式（4 メガバイト空間）
2 次記憶システム	固定ディスクウインチェスタ型 (29 メガバイト)，ディスクパーティション機能
オペレーティングシステム	各環境にそれぞれ独自の OS に相当するもの内蔵 (Boot 時，Alto Exective)
ローカルネットワーク	イーサネット＆Xerox Network Systems/Pup Protocols
外部インタフェース	RS-232-C ポート，カラーモニタコントロールボード

回数を少なくするという効果を持つ．なお，CONSセルのサイズは32ビットである．

◇変数束縛：変数束縛に関してInterlisp-10ではシャロウ・バインディングを用いているのに対してInterlisp-Dではディープ・バインディングを採用している．ディープ・バインディングは変数のアクセスに時間がかかり，シャロウ・バインディングは変数の束縛に時間がかかるという欠点がある．シャロウ・バインディングの欠点の1つに，プロセスの切り換えの際のオーバヘッドが大きくなることがある．プロセス（マルチプロセス）の機構を付加することを最初から考慮に入れて，Interlisp-Dではディープ・バインディングを採用している．

◇ガーベジ・コレクション：ガーベジ・コレクション(GC)の方式に関して，Interlisp-Dでは，DeutschとBobrowによって考案された“リファレンス・カウント方式”を採用している．この方式では，GCに要する時間は，回収されるガーベジの量に比例する時間しかかからない．また，Interlisp-Dでは，一定時間以上入力待ちをしているときに自動的にGCがかかるようになっており，実質的にGCのオーバヘッドが表層にでないようにしている．これは，今まで多くのシステムで採用されていた“マーキング方式（アドレス空間に比例するGCの時間が必要）”に比べて1100 SIPのように大きな仮想空間を持つマシンでは効率が良い．

2) 対話型グラフィックスとユーザ・インタフェース

Interlisp-Dでは，ビットマップ・ディスプレイとマウスのハードウェアを利用したグラフィックスを用いることによって，より良いユーザ・インタフェースを構築できる．

◇メニュベースインタラクション：マウスの典型的な使用例としてメニュがある．メニュにより，タイプすることなくコマンドを選択できる．今までの計算機では，タイピングの量を減らしたいためになるべく短いコマンド名を使うが，メニュを用いれば，タイプする必要がなくなるため分かりやすい長いコマンド名を用いてもよくなり，インタラクションにおけるユーザの負担が軽減される．

◇マルチウインドウ・システム：Interlisp-Dは，ビットマップ・ディスプレイを用いたマルチウインドウ・システムが基本になっている．ウインドウの管理はシステムが行うので，ユーザは，他のウインドウの存在や，ウインドウの表示されている位置を気にせずプログラミングすることができる．ウインドウはメニュにより，移動，拡大・縮小，消去，アイコン化等のオペレーションを対話的におこなうことができる．もちろん，これらのオペレーションは，プログラムによっても行うことができる．

3) ローカルエリアネットワーク

Interlisp-Dでは，イーサネットの下位レベルはもとより種々の上位レベル・プロトコルまでサポートしている．特に，ファイルの操作は，ファイル・サーバや他のワークステーション上のリモート・ファイルもローカル・ファイルと全く同じ扱いをすることができる．

4) 強力なプログラミング環境

Interlispの特徴はその強力なプログラミング環境である．Interlisp-Dのプログラミング環境は，その環境を受け継ぎ，対話型グラフィックスを用いることによってユーザ・インタフェースが飛躍的に改善され，さらに強力なものになっている．Interlisp(-D)のプログラミング環境の特徴は，それが単なる独立したツールの寄せ集めではないということである．個々のツール群は統合化されて1つの環境を形成している．

以下では主なプログラミング・ツールの説明を行う．

◇ファイル・パッケージ：Interlispではプログラムの編集・デバッグ・実行といったセッションは存在しない．プログラムの修正は，Interlispの環境の中で直接データ構造を変更することによってなされる．このようなシステムでは，内部的なデータ構造をファイルに書き出す機能が，バックアップを取るという意味でも，プログラムを他の環境に移すという意味でも重要になってくる．Interlispでは，プログラムとファイルとの1対1の関係は存在しないので，どの関数がどのファイルに入っているかを管理するのは重要なことになってくる．これを行ってくれるのがファイル・パッケージである．ファイル・パッケージは，エディタやDWIM(後述)などによる関数の定義の変更も知っており，ユーザに対してどのファイルが再セーブ，再コンパイルされなければならないかを教えてくれる．

◇エディタ：Interlispのエディタは，既述のとおり内部のデータ構造（リスト構造）を直接変更する構造エディタである．リスト構造の修正には大きな威力を発揮する．このエディタはInterlisp-Dでは対話型グラフィックスを用いた，ディスプレイ・オリエンテッ

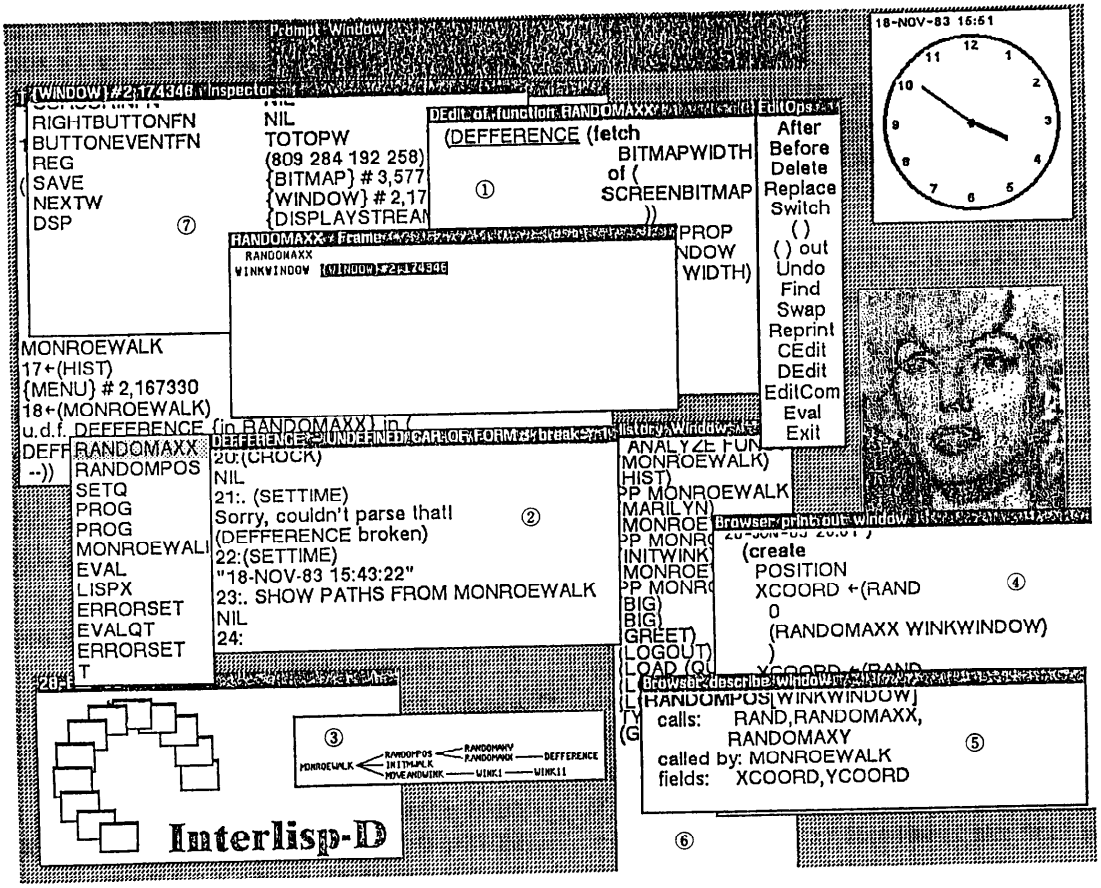

図-4 Interlisp-D の視覚的プログラミングツール

ド・エディタとなっている（図-4①）．メニュを用いて編集を進めていくので，使いやすいし，覚えやすい．結果はすぐにプリティ・プリントされるので操作ミスも直ぐに発見される（間違っている場合にはUndoコマンドを用いて操作を取り消せばよい→プログラマーズ・アシスタントの項参照）．一度に複数の関数を修正することもできるため，1つの関数の一部を取り出して他の関数の中にコピーすることも可能である．また，コンパイルされている関数を修正しようとした場合，ファイル・パッケージの働きにより，自動的にソース・コードが読み込まれる．

◇ブレーク・パッケージ：Interlispではプログラムの中にエラーがあった場合ブレーク・パッケージ（図-4②）というデバッガに自動的に入る．ブレーク・パッケージは，エラーが起こった時点でのシステムの状態をそのまま保存しており，ユーザはバック・トレース・コマンドやインスペクタ（後述）によりこれらの情報を参照できる．ここからエディタを呼び出してエラーを起こした関数などを修正することができる．修正がすんだら，その関数の前まで制御を移して，そこから実行を再開すればよい．

◇DWIM：DWIM（“Do What I Mean” の略）はプログラムの自動修正機構でその1つの機能としてスペリング・コレクションがある．プログラムを起動するときにミスタイプがあっても，自動的に修正されてそのまま実行してくれる．また関数定義のなかのミスタイプについては，システムはユーザに間違いを訂正してよいか問い合わせをしてくる．

◇マスタスコープ：プログラムのサイズが大きくなってくると，プログラムの一部分の変更が他の部分にどのような影響を及ぼすかがわかりにくくなってくる．この影響を調べるために開発されたプログラムの解析ツールがマスタスコープである．そのコマンドの1つに，関数の呼び出し関係をツリー状にプリントするSHOW PATHS というコマンドがある．Interlisp-Dでは，このコマンドを使うと新たにウインドウが作ら

れ，呼び出し関係はグラフィカルに表現される（図-4③）．それだけでなく，マウスを使うことによって，その関数の定義を見たり（図-4④），関数の引数や使っている変数およびデータ構造を調べたり（図-4⑤），さらには，エディタを呼び出してその関数を修正したりできるように拡張されている．これはブラウザと呼ばれている．

◇プログラマーズ・アシスタント：ユーザの入力，その副作用，およびその結果の履歴は，システムに貯えられている．プログラマーズ・アシスタントは，この履歴を操作するコマンド群を提供している（図-4⑥）．REDO コマンドである操作を繰り返すことができる．FIX コマンドでユーザの入力の一部を修正して再実行させることができる．また，UNDO コマンドを使うことによってある操作の効果を取り消すことも可能である．履歴は，アプリケーション・プログラム中でも使用可能である．

◇CLISP：Lisp のシンタックスは単純ではあるが，括弧が多いため人間にとっては読みづらいものとなっている．Interlisp に ALGOL 的なシンタックス（+，−，*，/等のインフィックス・オペレータ，IF-THEN-ELSE 構文，FOR や WHILE-DO 等の繰り返し制御文など）をとりいれたものが Clisp である．Clisp では LISP の構文に新しい構文が追加されたものとなっており，Clisp の構文と通常の構文の混在も可能になっている．

◇インスペクタ：Interlisp-D では，S 式だけではなく複雑なデータ構造を扱うことができる．たとえば，ウインドウやメニュ，プロセスなどである．これらのデータ構造は，プリント関数などではプリントアウトできず，データのタイプとそのアドレスだけが表示されることになる．この内容を調べたり，変更をするときにインスペクタが用いられる．インスペクタはあるオブジェクト（データ）のすべてのフィールド名とその値の対をインスペクト・ウインドウ（図-4⑦）に表示する．値はマウスを用いてさらにインスペクトすることができる．インスペクタは値のデータタイプをみており，エディタで修正できるデータタイプであればエディタを呼び出すことができるようになっているし，値がビットマップというデータ構造ならば，ビットマップ・エディタが呼び出される．

そのほかにも，Interlisp(-D) ではいろいろなツールを用意している．たとえば，パターン・マッチングの機能を使いやすい形で提供しているパターン・マッチ・コンパイラ，他の LISP で書かれたプログラムを Interlisp に変換する Transor，ソース・ファイルにクロス・リファレンスをつける機能，ディスプレイ・ベースのテキスト・エディタなどである．

Interlisp-D ではマルチウインドウの機能により，ツール群が統合化されていることの利点が増幅されている（図-4 参照）．マルチウインドウ・システムでは，あるツールから別のツールへ移ること，および，再び元のツールに戻ることが，モードを切り替えることなくスムースにできる．それまで使っていたツールで得た情報はそのままそのウインドウに残っていることも重要な点である．複数の情報を一度に参照しながら開発を進めていくことができるので，さらに開発の効率は上がる．

3.2.2 Interlisp-D の応用

Interlisp はこれまで PDP-10，DEC-20 などで使われてきて，様々なアプリケーションができている．その中には，MYCIN や MORGEN などのエキスパート・システム，KRL，KL-ONE，UNIT などの知識表現言語，LUNAR などの自然言語処理システム等がある．Interlisp-D によって，これらの資産がパーソナル・マシン上で利用できるようになったことの意義は大きい．しかしそれ以上に重要なことは，Interlisp-D によって，これらのシステムにさらに強力な機能（使いやすいユーザ・インタフェースやコミュニケーションなど）をもたせることができる点である．

Interlisp-D の機能をフルに用いて開発されたシステムには次のようなものがある．エキスパート・システムでは，Schlumberger 社で開発された石油探索のデータを解析するシステムなど．PARC では LOOPS というエキスパート・システム作成環境が開発されている．また，KEE という同様のシステムが Intelligenetics 社によって開発されている．自然言語処理の分野では，新しい文法のフレームワーク LFG の文法記述システムが PARC で開発されている．

そのほかにも Interlisp-D の上には種々の資産が開発されてきているが，ユーザ・コミュニティの広がりとともにこれからもその資産は増え続けていくことだろう．

3.3 統合プログラミング環境ワークステーション 1100 SIP—Smalltalk-80—

1100 SIP 上には，Interlisp-D プログラミング環境の他に，Smalltalk-80 プログラミング環境が稼働する．

3.3.1 Smalltalk のバックグランドと意義

最近，Smalltalk-80 に対する関心が高まっているが，実際の Smalltalk-80 の姿は，恐らく関心を持たれている人々の抱いているイメージよりは遥かに革新的であり，かつ深遠である．Smalltalk-80 の目指すところは，開発のリーダである Adele, Goldberg 女史の言葉を借りれば，『人間の思考モデルとコンピュータのモデルの間の橋渡しを表現する“言語（環境）”および人間とコンピュータの相互の意志の疎通を表現する“言語（環境）”の実現』である．

Smalltalk のパイオニア達は，コンピュータやソフトウェアの役割，意義の本質を，ある目的のツール，モデル（環境）を現実世界にある制約を受けずに，自在に創作・作成するベースマシン（加工道具）としての能力またはそのツールの機能を即座にシミュレーションする能力と認識している．つまり，こうしたコンピュータの能力を引き出すための，望ましいモデル表現・記述の言語を模索した中から生まれたものが，Smalltalk である．人々が必要に応じて，簡単にかつ自在に目的のツールを作成・シミュレーションする事が可能となれば，今までのコンピュータ歴史の中で示されたコンピュータの有用性の次元とは，異なる新しい次元の可能性と恩恵をもたらすと考えられている．こうした壮大なビジョンが，DynaBook (personal dynamic media) というコンセプトに集約されている．Smalltalk-80 はこの DynaBook のソフトウェアコンセプトの延長上にあると考えるのが自然である．

また Smalltalk-80 を別の側面から眺めてみよう．最近，ソフトウェアの生産性についての関心が高まっているが，Smalltalk-80 は，ソフトウェアライフサイクル設計手法に対する反省から，最近，注目を浴びているソフトウェアプロトタイピングの格好のツールとしても，最適であると評価されている．

3.3.2 Smalltalk-80 の言語フレームと理念

◇Smalltalk-80 の基本概念：Smalltalk の前提モデルは Communicating Object モデルとして知られているものである．これはこの世界の環境，知識，意志をモデル化し表現するのに，まず個々のリソース（認識の対象，機能性）を objects（対象）として捉え，objects 間のコミュニケーション（意志伝達）の方法は Message として定義される．Smalltalk で規定していることは，そのモデルにおけるインタラクションの表明，つまり message の記述方法のフレームを定めていると理解される．基本的には，この message の記述方法は，以下の形式となる．

受 信	キーワード	アーギュメント
object名	セレクタ：	object名

ここで，重要なことは，こうしたオブジェクトおよびセレクタは Smalltalk-80 の仮想イメージの世界で，定義（システムおよびユーザの区別なし）されるものであり，Smalltalk において，規定されているのではない．Smalltalk はその意味では一種の拡張可変言語の体系を備えている．こうした意味から，Smalltalk 言語においては，制御構造のメカニズムも規定されていない．

Smalltalk-80 の言語環境を形成している，重要な概念を簡単に表-4 に示す．

◇Smalltalk-80 アーキテクチャ：Smalltalk-80 のシステムは，Smalltalk-80 仮想イメージと Smalltalk-80 仮想マシンの2つから構成される．前者は，Communicating object モデルですべて表現されているもので，System Browser を介して，その内容を参照することができる．すべての Smalltalk-80 仮想イメージで規定されている method は，コンパイラによって，Smalltalk-80 仮想マシン用のコード（バイトコード）列に展開される．Smalltalk-80 仮想マシンは，このコード列を解釈・実行する．この意味で，1100

表-4 Smalltalk-80 の基本概念

概念項目	説明
Everything is an Object 概念	Smalltalk-80の世界では，データもプロシージャも基本的には，独立して存在しえない．すべて，objectが基本単位である．Smalltalk-80 における object の定義は，『プライベートなメモリスペースとオペレーションの手続きの集合によって表現される Smalltalk-80 の構成要素』である．
Message passing 概念	ある object にたいして，そこで規定されているオペレーションの一つを実行することの要求の意志を伝達するのが message である．Message passing の考え方は，交信する object の所在に依存しないために，普遍的なモデル化能力をもつ．
Class and Instance 概念	Smalltalk-80 の世界では，類似した objects をひとつのグループとして記述したものを class と定義している．また，class によって，記述された objects のひとつの具体例を instance と定義している．この考え方は対象モデル（プログラミング）の表現能力を飛躍的に向上させる．
Subclassing 概念	subclass は，『すでに存在する class から，method と variable を相続した class』と定義される．また superclass は method と variable の相続権を提供している class と定義される．Smalltalk-80 では，ある class の subclass として新しい class を生成することが，プログラミングの基本である．

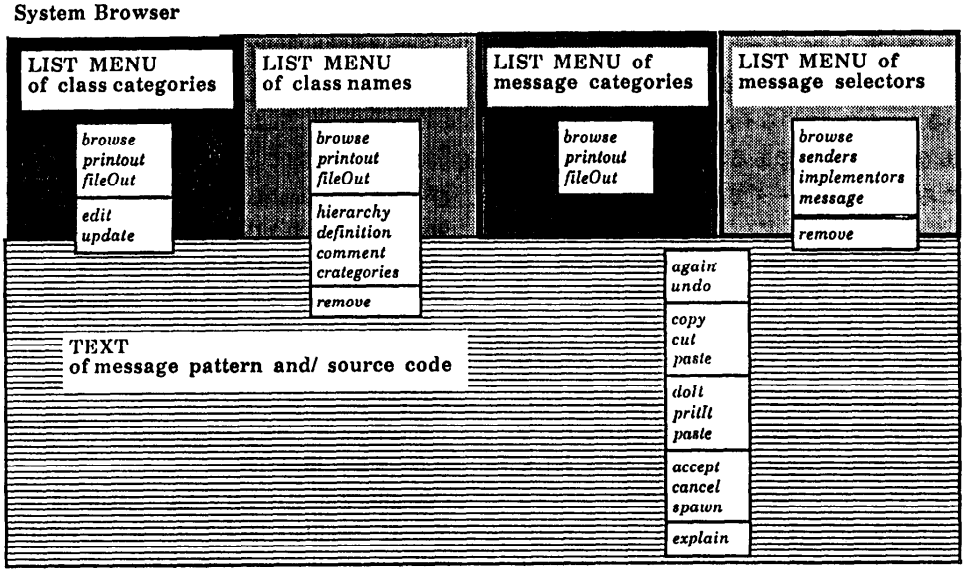

図-5 Smalltalk-80 システムブラウザ（表示情報とポップアップメニュ）

SIP は，Smalltalk-80 の環境がローディングされている状態では，マイクロプログラムレベルで最適化されているので，Smalltalk-80 マシンとして機能する．仮想マシンアーキテクチャの特徴は，評価スタック & ダイナミックセレクタサーチを基本とするバイトコードインタプリタとオブジェクトを基本単位とするオブジェクト指向型メモリ管理方式である．

3.3.3 プログラミング環境としての Smalltalk-80

Smalltalk-80 は，従来の概念の延長上にある単なるプログラミング言語ではなく，まさしく統合化プログラミング環境 (Unified Programming Environment) であり，それ以外の言葉によっては Smalltalk-80 のもつ機能性，能力は説明が不可能である．ここで，述べるプログラミング環境とは，プログラムの設計，記述，作成，テスト，管理などの，一連のソフトウェアの作業工程に応じたツールが完備した作業空間を意味する．また，そのプログラミング環境が統合化されているという意味は，それらのツールが単に独立してその機能性を提供するのでなく，それらのツールが連綿とした関係をもち，ユーザにとってプログラミング作業を進める上で必要となる情報とメカニズムを自在に提供するように整備されているということである．

さらに，Smalltalk-80 プログラミング環境では，System Browser を介したインタラクションでフォロされた情報は体系的データ・ベースとして組織化され，要求に応じて有効な情報を提示することが可能である．このことも，プログラミング環境の統合化の重要な意味である．ここでは，Smalltalk-80 プログラミングスタイルの考え方が集約されている System Browser の機能の概要を述べる．

◇System Browser ウインドウ：端的に言えば，Smalltalk が想定しているプログラミング作業の本質をサポートする基本概念を具体化して，洗練化を図ったのがこの System Browser である（図-5 参照）．Smalltalk-80 を構成している，約 160 の object (class)，それに付随する 4000 あまりの method は，約 4 万行（通常のプログラミング言語に換算するとこの約 5 から 7 倍のサイズに）のソースプログラムからなり，この電子プログラム図書館に体系的に収まっている．この System Browser のウインドウの様子を図-6 に示す．ユーザは，プログラム（たとえば，Class や Method）の修正・変更，追加・削除などすべてのプログラミング作業をこの System Browser を介して行う．つまり，プログラミング作業を知らず知らずナビゲートしてくれるようになっている．さらに System Browser に登録管理されているライブラリに関しては，object の名前およびその定義，各種の変数名，セレクタの定義，class 階層等の情報がデータベースとして体系化されている．プログラム作成・解読・デバッグに必要となるすべての情報をクイックリファレンスすることが可能である．

プログラミングツールとしては，以上述べたウイン

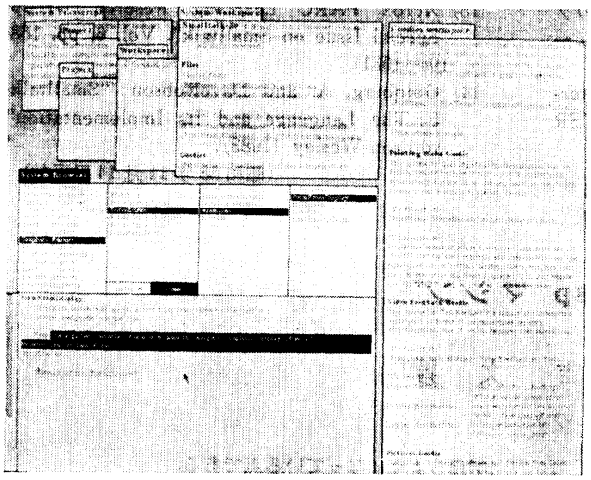

図-6 Smalltalk-80 の画面

ドウとして表層に出てくるもの以外に，コンパイラ，デバッガ，ディコンパイラ，インスペクタ，各種のディクショナリ等のプログラミングツールが完備している．特に，デバッガ等は，陽にその機能を指定しなくても，messageが解釈実行出来なくなると，自動的にデバッカのモードに移り，その実行時の環境をすべてリファレンスすることができる強力な機能を備えている．

3.3.4 Smalltalk-80 の先進性

実際に，Smalltalk-80を使用した率直な感想を以下に述べこの稿を閉じる．Smalltalk-80 環境においては，ソフトウェア設計にたずさわった人であれば，今まで頭を悩ましてきたことが一連の Smalltalk-80 の提示するプログラミングパラダイムによって，すべて前向きに解決される．さらに，Smalltalk-80に陰に陽に組み込まれている言語概念は，1970年代ソフトウェア危機の中から生まれたソフトウェア工学やプログラムの方法論の成果が集大成されたもので，学術的にも優れた評価がなされているといえる．Smalltalk のプログラミングの方法論は，基本的には，優れたプログラマが行ってきていた方法論そのものであると考えることもできる．そのため，Smalltalk-80のナビゲーションによって，比較的経験のないプログラマでも，優れたプログラマと同等の質のプログラムを作成することが期待される．Smalltalk-80においては，プログラマに対して，雑事を行わせないという立場から，強力なプログラミング環境を提供しかつエレガントなナビゲーションをしているが，その反面，意欲的な創造性の芽を摘むような制約を極力なくし，創造性の源である，おおいなるフリーダムを与えている．また，Smalltalk-80では，コンピュータのハードウェアの特性が，陰に陽に，プログラミング言語に入り込むことを徹底的に排除して人間との親和性を非常に大事にしている．

このように，Smalltalk-80は，コンピュータ（ソフトウェア）の可能性を引き出す壮大で昇華された一大ビジョンであることは確かであり，まだまだ奥行きがある．日本でのSmalltalk-80の本当の評価はこれからであるが，Smalltalk-80は，ソフトウェア設計およびプログラミングの本質を見据えた大いなる秩序をもった知的創造性活動の泉である．今後の発展に期待し，本稿を閉じる．

参考文献

[*Xerox Network Systems*]

1) Digital Equipment Corporation; Intel Corporation; Xerox Corporation: "The Ethernet, A Local Area Network: Data Link Layer and Physical Layer Specifications," (Sep. 30, 1980) Version 1.0 (富士ゼロックスから邦訳あり).
2) Xerox Corporation: "Internet Transport Protocols" XSIS 028112, (Dec. 1981) (富士ゼロックスから邦訳あり).
3) Xerox Corporation: "Courier: The Remote Procedure Call Protocol" XSIS 038112, (Dec. 1981) (富士ゼロックスから邦訳あり).
4) 上林，池田，龍田："ローカル・エリアネットワークの現状と課題"，電子写真学会誌，No. 2 (1983).

[*JStar*]

5) Smith, D. C., Irby, C. H. and Kimball, R. B.: "The Star User Interface: An Overview." AFIPS Proceedings of National Computer Conference, Vol. 51, pp. 515-528 (1982).
6) 伊東："JStarにおける日本語入力方式"，情報処理学会日本文入力研究会資料 (1983).
7) 上林："ワークステーションの実際"，サイエンス別冊（ニュービジネス特集号）(1983).

[*Interlisp-D*]

8) Teitelman, W.: "Interlisp Reference Manual", Xerox (1978年10月，1983年8月).
9) Xerox: "Interlisp-D Users Guide" (1982年9月).
10) Xerox: "Interlisp-D The Chorus Release" (1983年4月).
11) Seil, B.: "Power Tools for Programmers"

DATAMATION 1983年2月号 (邦訳:「プログラマ待望の強力なツールが登場」, 日経コンピュータ (1984年4月18日号).
12) Teitelman, W. and Masinter, L.: "The Interlisp Programming Environment", COMPUTER pp. 25-33 (1981年4月号).
[*Smalltalk*-80]
13) Xerox PARC Learning Reserch Group: "Special Issue on Smalltalk", Vol. 6, pp. 168-194 (1981).
14) Goldberg, A. and DaveRobson: "Smalltalk-80: The Language and Its Implementation", Addison-Wesley (1983).

(昭和58年11月14日受付)

製品例

Lisp マシン†

元吉　文男††

1. はじめに

Lisp マシンは, いわゆるワークステーションとは開発目的が若干異なっているが, 製品として見た場合には, ビットマップディスプレイによるマルチウインドウシステムやネットワークサポートなどの機能があり, ワークステーションの仲間に入れても差し支えがないようである.

記号処理用言語 Lisp は計算機に数値計算だけでなく, 構造を持ったデータを扱うように作成された言語であり, 人工知能等の研究に古くから使用されてきたが, メモリを大量に使用する, 処理速度が遅い等の理由で普及が遅れていた. そこで, この Lisp を高速に実行するパーソナル計算機が開発されることになり, まず CONS と呼ばれる Lisp マシンが MIT で開発された. 続いて CADR と呼ばれるマシンも同所で開発され, 商業化の動きもでてきた. このような中で LMI 社と Symbolics 社が Lisp マシンを製品化して販売することになった. どちらのマシンも CADR の流れを汲むもので, 似かよったものになっている.

近年, 知識情報処理の分野が注目を集めているが, この方面の記述には Lisp が適しており, Lisp を高速に実行するパーソナル Lisp マシンに対する需要は大きくなるものと思われる.

ここでは Symbolics 社のマシンについて詳しく見てみることにする. LMI 社の製品との大きな違いは, LMI 社のマシンは MC 68000 とのデュアル CPU 構成を前面に出しており, MC 68000 を使用した Unix が走るようになっている点である.

2. ハードウェア

全体の構成図を図-1 に示し, 以下の説明はこの図に添って行うことにする.

2.1 CPU 等本体

このマシンの CPU は Lisp を高速に実行させるために, 市販の CPU ではなく TTL を使って組み上げたマイクロプログラム制御方式である. マイクロ制御記憶は 112 ビットで 8 K 語でありマイクロマシンサイクルは 200 ns となっている.

主記憶は 1 語が 36 ビットで 256 M 語の仮想アドレス空間をもっており, 物理的には 7.5 M 語までの, エラー訂正機能付きのものをもつことができ, Lisp で扱う大きなアドレス空間をサポートしている. データの表現には 36 ビットのうち 8 ビットをデータタイプを示すタグとして使用しており, マイクロプログラムでデータタイプのチェックを行うことによって, 従来の計算機上で実行する場合に処理時間のかかっていたものが高速に処理できるようになっている.

入出力については MC 68000 を使用したフロントエンドプロセッサが使用されており, ディスク・ビットマップディスプレイ・キーボード等の管理を行っている. CPU 等が故障した場合には, このフロントエンドプロセッサが診断システムとして動作するようになっており, レジスタ・バス・主記憶・ディスク等にアクセスできる. また, ネットワークを介して, 他の Lisp マシンから診断することもできる. マイクロコードのデバッグの機能もこのフロントエンドプロセッサが持っており, ステップ実行やブレークポイントの設定が行えるようになっている.

† Lisp Machine by Fumio MOTOYOSHI (Information Science Div., Electrotechnical Laboratory).
†† 電子技術総合研究所