ネットワークを介した協調活動の支援環境

川上直木, 川瀬宏一郎, 小寺康男
鈴木大輔, 萩原豊隆, 橋本昌嗣
落水浩一郎

1996 年 4 月 16 日
IS-RR-96-0012S

北陸先端科学技術大学院大学
情報科学研究科
〒 923-12 石川県能美郡辰口町旭台 1–1
knao@jaist.ac.jp
k-kawase@jaist.ac.jp
kodera@jaist.ac.jp
daisuke@jaist.ac.jp
hagiwara@jaist.ac.jp
masatugu@jaist.ac.jp
ochimizu@jaist.ac.jp

ISSN 0918-7553

要旨

ネットワークの普及期を迎え、人間の社会活動、仕事の形態は大きく変化しつつある。ニュースや電子メールによる情報の発信と獲得、内外の研究者とのディスカッション、会議による合意形成、意思決定、共同設計作業、研究成果のリアルタイムな発表などの人間の知的諸活動を、ネットワークを介して実施するための、新しい計算機環境の実現に関する基礎理論、基礎技術の開発が必要である。このような活動形態においては、「活動の場から、必要な情報に迅速にアクセスできること、さらには活動の場にそれを支える必要な情報が追随できること」が肝要である。

本報告では上記目的のための研究課題を以下の点にわたって考察する。

- インターネットに加えて、移動計算環境、偏在計算環境、移動体通信技術の充実による、計算機環境の携帯性の向上
- ネットワークに点在する膨大な量のデータから必要な情報のみを抽出・利用する情報フィルタリング技術（情報検索技術の革新）
- ごく限られた知識しか持ち合わせない人が、計算機の助けを借りて問題を解決する探求的学習の支援
- 共有情報の管理、決定事項の管理、プロセスやノウハウの共有を促進する、共同作業支援のための各種サーバ類の開発
- 上記ソフトウェア群をオブジェクト指向アーキテクチャに基づき効率よく開発するための、ソフトウェア開発環境

もくじ

第 1 章

はじめに

1.1 先端的ソフトウェア創出のための研究環境について

先端的ソフトウェア創出のためには以下の点を留意する必要がある。

- 広く内外の情報に接することにより、新しいアイディア、理論、製品に関する情報を発信／獲得／利用すること。
- 必要に応じて専門家達が集まり、ブレーンストーミングや評価などの活動を充実させること。
- 既存の先端技術をハードウェア、ソフトウェア面にわたって使いこなしつつ、問題点を発見し、解決への糸口をつかめること。

そのためには以下のような研究環境を整える必要がある。

- 情報獲得／共有／利用の支援機能
- コミュニケーション (CSCW) の支援機能
- 共同作業 (ソフト開発、執筆等) の支援機能

本論文では、1.1 に示すような視点から、研究環境の現状の分析、評価を行いつつ、紙コンピュータ、アクティブチャネル、グループウェアベース、プロセスサーバ、分散サーバ、バーチャルルーム等の新しい概念を提案し、現状の計算機環境上で上記アイディアを実験する手段を論ずる。

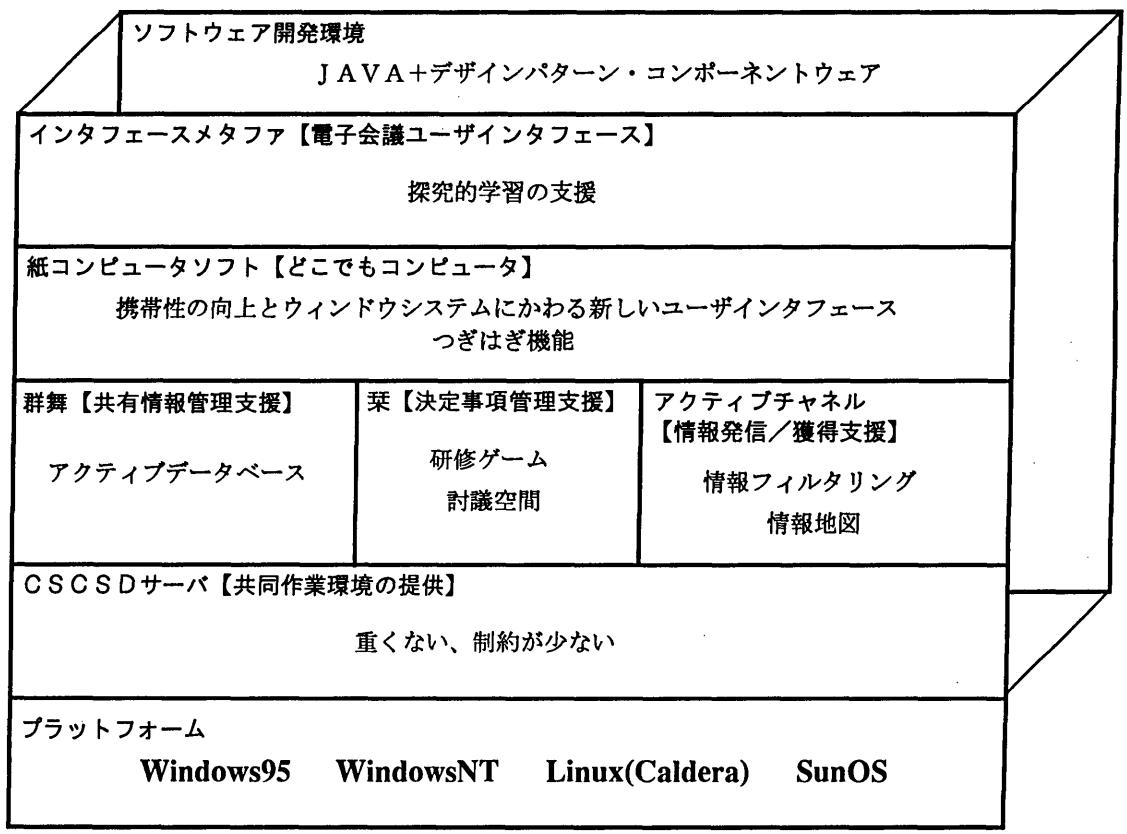
ソフトウェア開発環境
ＪＡＶＡ＋デザインパターン・コンポーネントウェア
インタフェースメタファ【電子会議ユーザインタフェース】
探究的学習の支援
紙コンピュータソフト【どこでもコンピュータ】
携帯性の向上とウィンドウシステムにかわる新しいユーザインタフェース
つぎはぎ機能
群舞【共有情報管理支援】
アクティブデータベース
栞【決定事項管理支援】
研修ゲーム
討議空間
アクティブチャネル
【情報発信／獲得支援】
情報フィルタリング
情報地図
ＣＳＣＳＤサーバ【共同作業環境の提供】
重くない、制約が少ない
プラットフォーム
Windows95 WindowsNT Linux(Caldera) SunOS

図 1.1:「自在＋＋」環境

1.2 現状の分析

落水研究室の現在の研究環境を題材にして、研究活動支援に関する対象を以下のような視点から構造的に分類する。

- 既設の計算機設備 (ハード、システム) を使いこなすための作業
- 既存の計算機による支援環境 (ソフトウェア) を使いこなすための作業
- 計算機に依存しない思考活動

上記を分類マトリックスの横軸とし、縦軸を以下の 3 つの項目とする。

- 要求される時間 (1.2)
- 要求される経験度 (1.3)
- 要求される独創性 (1.4)

図 1.2,1.3,1.4 中の時間、経験、独創性の要求度の高いものをなるべく軽減する、つまり、そのサポートをおこなってくれる環境が必要になると考えられる。よって、これを考えることで、現在の計算機環境に何が足りないかを発見する手がかりになるであろう。

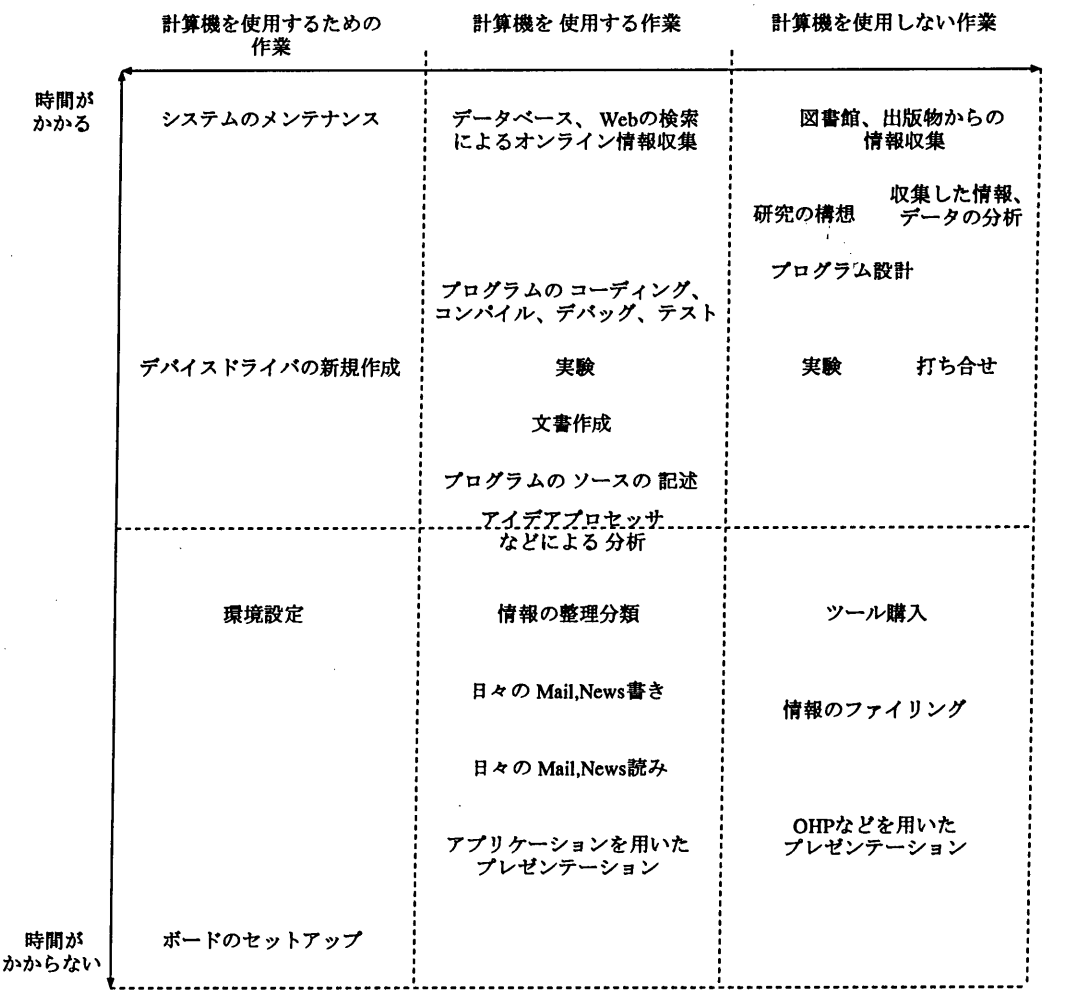

図 1.2: 効率化への指針としての分類

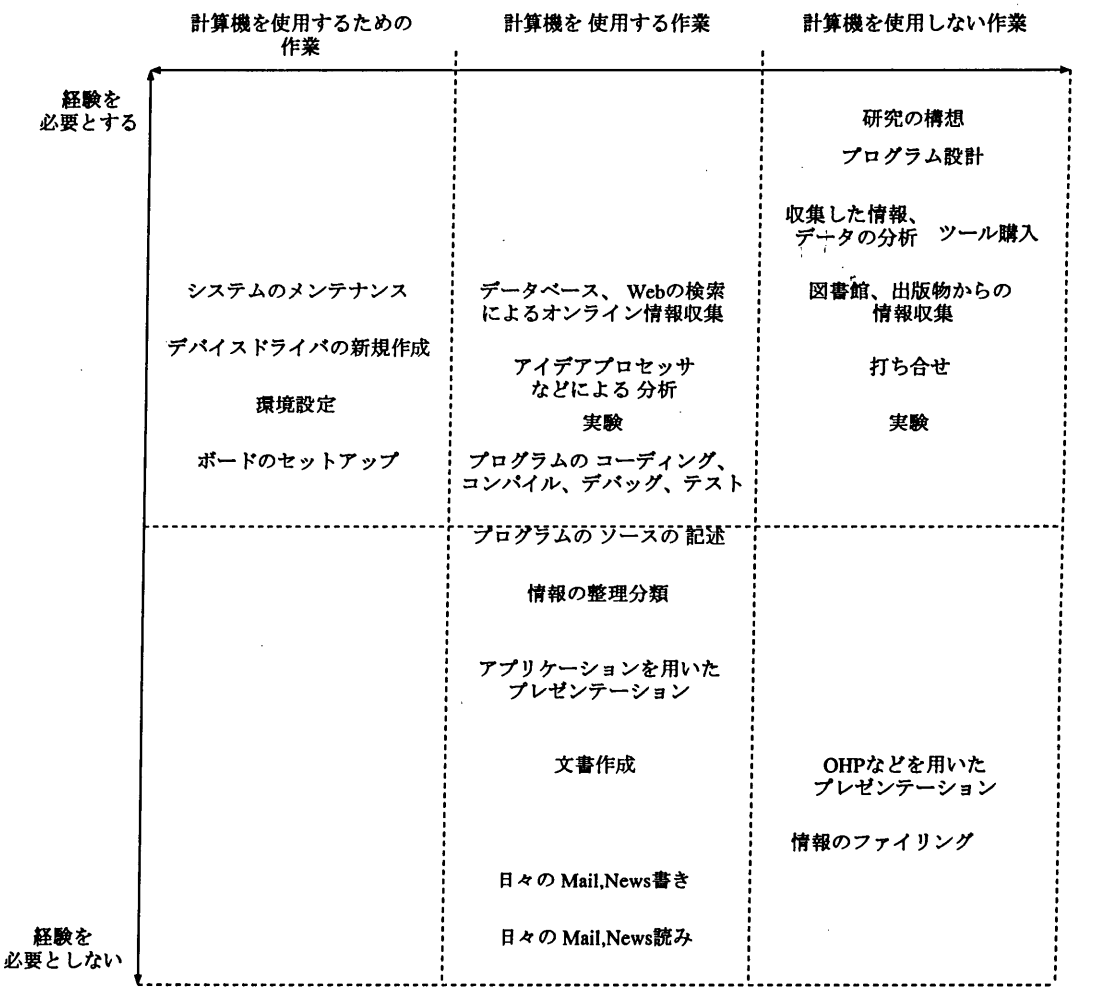

図 1.3: 経験の有無による負荷軽減への指針としての分類

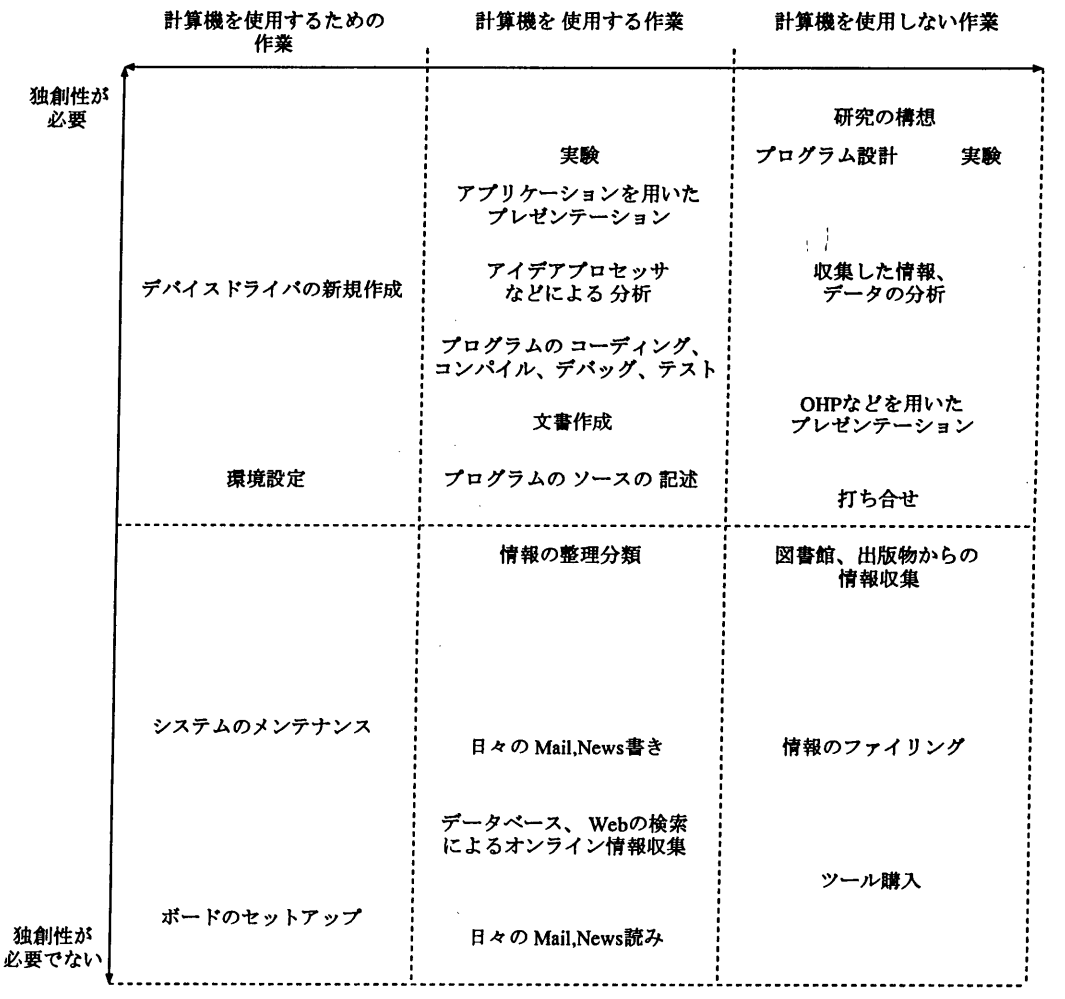

図 1.4: 独創性への指針としての分類

1.3 理想像

図 1.2,1.3,1.4 を分析することにより、情報獲得／共有／利用、コミュニケーション、共同作業の支援のためには、以下のような新しい考え方でその活動性を高める必要があると判断する。(1.3)

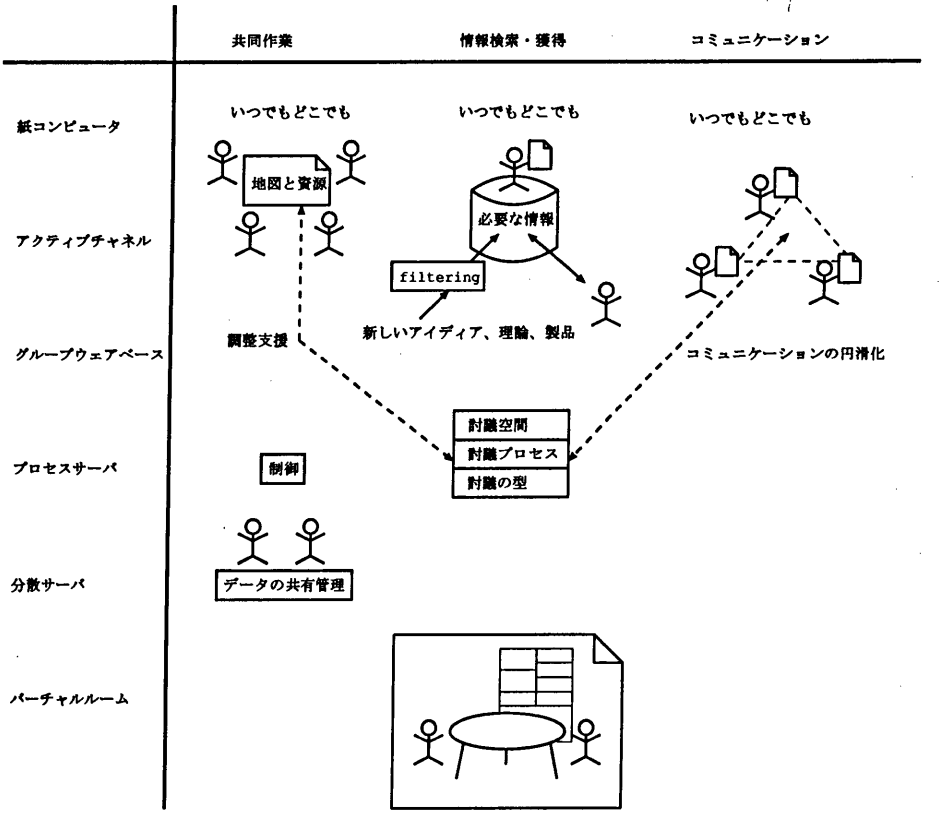

図 1.5: 新しい研究環境の構成要素

問題点発見の支援 .

(1) 人間系 …分散環境における共同活動の阻害要因 (人間系、ツール系、ネットワーク系) を、プロトコル解析により分析し形式化する。ネットワークのありなしに依存する作業の促進/阻害要因を明確にすることを最終目標とする

紙コンピュータの NotePC による試作 .

(1) 携帯性 …計算機環境の携帯性を向上する。

情報獲得／共有／利用の支援 .

(1) アクティブチャネル …研究活動の推進に必要な情報の管理、獲得、提供を支援するようなデータベースが望まれる。紙コンピュータに装着されるアクティブチャネルは、情報フィルタリング、グループウェアと連動しつつ、最新の情報を獲得提供する。

コミュニケーションの支援 .

(1) グループウェアベース (栞) …ソフトウェアプロセスの実行にともなって発生するコミュニケーション内容を、討議プロセス・討議空間・討議の型により構造的に記録し、討議内容の相互関係・進捗状況の把握、討議経過の把握を支援するまた、人間系における閉塞状態を検出し調整することを支援するような機能の実現も望まれる

(2) プロセスサーバとグループウェアベース …変換プロセスとコミュニケーションプロセスを融合することにより、ソフトウェア開発活動全体をカーバするソフトウェアプロセスモデルを定義し、作業の連続性を保持する手段を検討する

(3) グループウェアベースとオブジェクトベース …オブジェクトベースは成果物とその間の依存関係を管理し、グループウェアベースはプロセスの実行やコミュニケーションによって発生する「決定事項とその間の因果関係」を管理する。

共同ソフトウェア開発の支援 .

(1) プロセスサーバ (皆伝) …標準 (設計基準、品質基準)/作業ガイド (方法論/開発環境に関する情報提供)/ルール (決定事項の文書化、文書管理、変更管理、コミュニケーションプロトコル)/インタフェース (他者との仕事の関係) 等の情報を、プロジェクト構成員毎に提供しつつ、サーバ間の交信により、チーム活動状況 (プロジェクト進捗状況) を把握する。オブジェクト指向方法論に対するメトリクス体系を開発し、それに基づく標準化体系=作業標準 (ガイド)/ルール/ インタフェースの開発が必要である

(2) 人間系とプロセス系 (Vela) …事例推論とモデル推論を用いてノウハウを形式化し、再利用と技術移転を促進する (ドメインに依存する作業ノウハウの提供)

(3) 分散サーバ (群舞) …共有データの動的変更/アクセス制御/変更管理を WorkSpace Manager,Intelligent Meditator, Half Protocol により支援する。また、作業結果の伝達・整合に関するルールの動的管理を開発する

(4) 人間系と分散サーバ …データベース分野における諸成果 (並行制御、トランザクション管理、トリガ) を、人間系を含めたシステムに適用可能であるように拡張する

(5) **OOA/OOD/OOP** …デザインパターンとオブジェクト管理機構、メタオブジェクトによるオブジェクト指向アーキテクチャ、オブジェクトモデリングに関する事例研究とドメインモデリング。既存概念 is-a(整理),is-part-of(構造化) に加えて、新しい基本概念の発見。オブジェクト指向プログラミング以外の分野 (分析・設計) でのオブジェクト指向技術の有効性の証明

仮想現実感に基づくユーザインタフェース .

(1) **バーチャルルーム** …上記のシステムすべてにおいて、状況に応じた活動メタファを基に設計されたユーザインタフェースが必要である。ネットワークを介した、話し合い。作業、情報交換/検索に的した作業メタファを、仮想現実感の技術を背景にして、新たに設計する必要がある。また作業状況に応じたコンテキストを提供する機能も必要である。

計算機環境の整備 .

(1) **CSCSD サーバ (飛翔)** …Platform の異なるマシン・OS を土台にして、上記の機能群を円滑に作動させるには、OS、開発環境の連携、種々のデータ形式の変換・共有が効率よく、制限の少ない形で実現されなければならない。また、適切なパラメータ (適切なメディア、会話形態) (ドメインと方法論、計算機環境、ゴールとコスト、人間要因) に基づき、作業状況に応じた資源を提供する。

第 2 章

各論

2.1 紙コンピュータ

2.1.1 現状の整理

現在普及しているパーソナルコンピュータ (PC) のシステムは、基本的に 1 人のユーザが 1 台のディスプレイ装置を利用することを前提として構成されている。その理由の 1 つとして、デスクトップ型 PC のディスプレイ装置として広く利用されている CRT モニタの場合、必要な設置スペースが大きい上に重いので、机の上に複数台設置するのは物理的に困難であることが挙げられる。ただし、Note PC で利用されている液晶モニタであれば、CRT モニタに比べて必要な設置スペースが小さく軽いので、机の上に複数台設置するのは容易である。また、Note PC はより薄くより軽くという方向で進化していくと思われる。

2.1.2 紙コンピュータの定義

PC を構成する様々なデバイスの小型化・軽量化が進むことにより、将来以下に定義するような紙コンピュータが実現できると仮定する。

紙コンピュータとは、

- 薄い・軽い・安価である『紙』
- 計算能力を持ち、ネットワークに接続できる『コンピュータ』
- 動的な表示を行なえる『ディスプレイ装置』

という 3 種類のデバイスの特性を兼ね備えた 1 個のデバイスである。

2.1.3 紙コンピュータのイメージ

私が思い描いている紙コンピュータの具体的なイメージは以下の通りである。

- 外見は A4 サイズ程度のボール紙に近い。
- 表の全面は表示画面かつペン入力用のデジタイザであり、ペンコンピュータとして使用できる。

- キーボードやマウスが接続でき、PC として使用できる。
- 重量は片手で持ち歩いても気にならないほど軽い。
- 無線通信機能を持つ。

そのイメージを図 2.1 に示す。

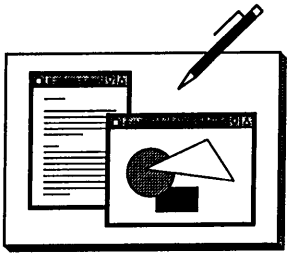

図 2.1: 1 枚の紙コンピュータ

2.1.4 紙コンピュータの使用形態

私が想定する紙コンピュータの使用形態について述べる。

1 枚の紙コンピュータの使用形態

この場合、現存の PC の使用形態と大きな違いはない。机上で作業するときには、デスクトップ型 PC 兼ペンコンピュータとして使用できる。つまり、キーボードやマウスを接続して入力したり、ペン入力を行なえる。携帯して作業するときには、Note PC 兼ペンコンピュータとして使用できる。

複数枚の紙コンピュータの使用形態

紙コンピュータの特性を活かした使用形態として、同時に複数枚の紙コンピュータを使用して作業する形態を想定している。

1 枚の紙コンピュータよりも広い面積を持つ紙コンピュータが必要な場合には、複数枚の紙コンピュータをタイル状に結合することにより、1 枚の大きな紙コンピュータを作ることができる。そして隣接する紙コンピュータの継目にまたがって表示やペン入力を行なえる。なお、結合と分離は作業中に動的に行なえる。そのイメージを図 2.2 に示す。

机上での作業においては、無線 LAN で接続された複数枚の紙コンピュータを 1 人で同時に使用できる。紙コンピュータの枚数は、作業状況に応じて動的に増減できる。紙コンピュータとして、前述の大きな紙コンピュータを混在できる。また、机上の紙コンピュータは必要な枚数を携帯できる。出先で作業したり情報を蓄積した後、持ち帰ってそのまま元の（あるいは別の）机上の作業環境にマージできる。そのイメージを図 2.3 に示す。

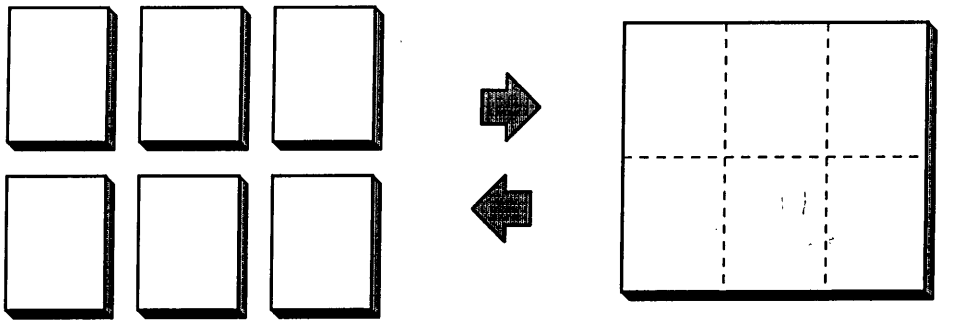

図 2.2: 複数枚の紙コンピュータと 1 枚の大きな紙コンピュータ

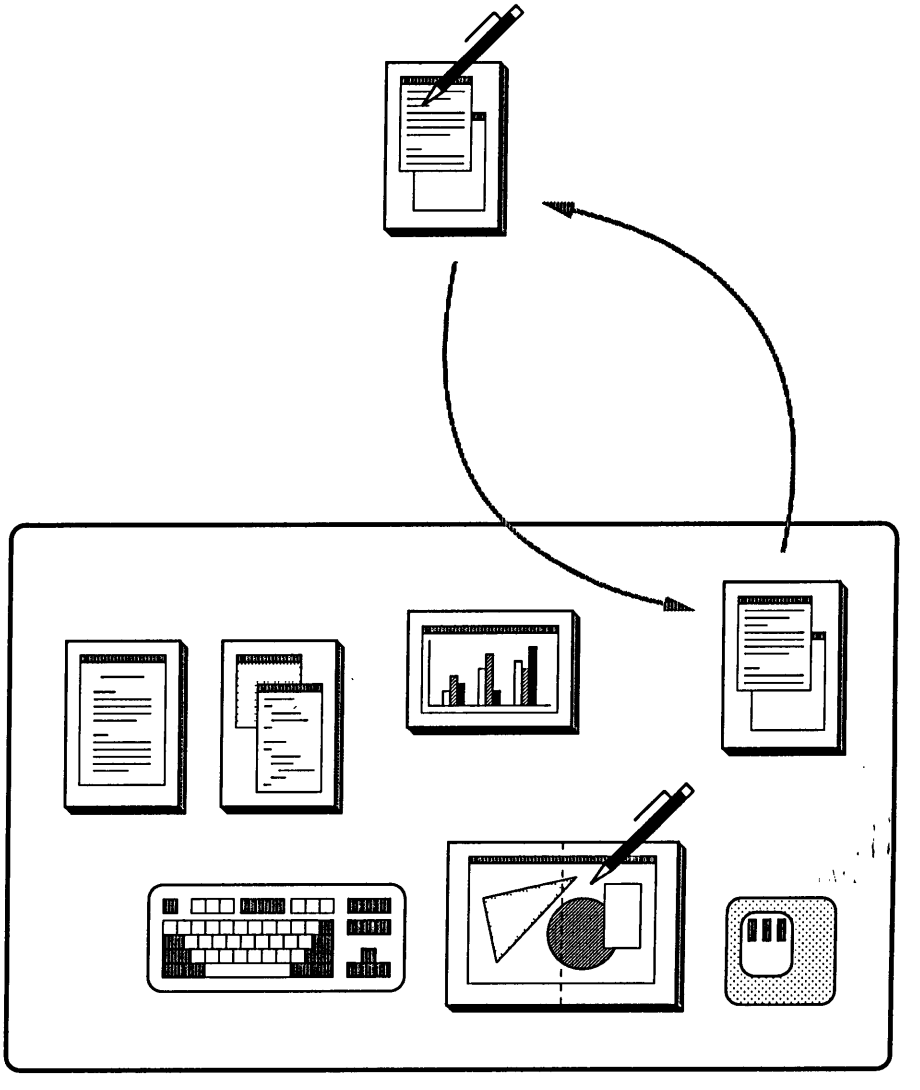

図 2.3: 机上および携帯時における紙コンピュータの使用形態

2.1.5 研究の目的

本研究の目的は、第 2.1.4 節で述べたような紙コンピュータならではの使用形態を実現するためのソフトウェアを開発することである。これにより、現実の机の上をインテリジェント化することを目指す。

2.1.6 研究計画・方法

本研究の計画・方法を以下に述べる。

疑似紙コンピュータの準備

紙コンピュータは現存しないデバイスであるから、とりあえずは現存のコンピュータ（Note PC やペンコンピュータ）で紙コンピュータをシミュレートすることにする。これを疑似紙コンピュータと呼ぶ。そして疑似紙コンピュータを複数枚用意して机上に無線 LAN を構築することにより、1 人で同時に複数枚の紙コンピュータを使用する作業形態をシミュレートする。

ソフトウェアの設計・実装

第 2.1.5 節で述べたソフトウェアを設計し、前述の無線 LAN 上に実装する。このソフトウェアは疑似 X サーバとして設計する予定である。この疑似 X サーバの仕事は、第 2.1.4 節で想定した使用形態において、各紙コンピュータにおける画面表示およびペン入力の受け付けを適切に行なうことである。

評価

実際に疑似紙コンピュータで構成した無線 LAN を使用し、前述のソフトウェアおよび私が想定した使用形態の評価を行なう。また、本研究室でその研究が進められている機能群と連係して使用し、全体の評価を行なう。

2.2 アクティブ・チャネル

2.2.1 必要な情報の選択による活動環境の支援

コンピュータ・ネットワークの著しい発達に伴い、コンピュータ環境下での研究活動においては、ネットワークを通して多種・大量の情報を手にすることが可能になっている。しかし一方では、その情報の多種・大量性は爆発的に増大しており、もはや個人がそれらの情報すべてを処理するのは、事実上不可能になってしまっている。したがって、ネットワークを流れている情報源から、自分にとって必要であると思われる情報を優先的に選択する何らかの手段を講じない限り、情報の収集・吟味に割く時間と労力は増加の一途を辿る羽目に陥ることになる。研究活動における情報収集は、ある最終的な目的へ至るための、不可欠ではあるが過渡的な作業であると言える。したがって、その作業の負担を軽減してやることにより、研究活動がより生産的なものになるように支援することが可能になるのである。

以上の点を踏まえて、活動環境、とりわけ情報を利用する環境を支援するためのシステムを今後検討して行きたい。

情報フィルタリングとアクティブ・チャネルによる、情報環境の支援のためのシステム

コンピュータ・ネットワークに流れている大量の情報に対して、ユーザのニーズ・興味・嗜好によるフィルタをかけることによって、ユーザが必要とする情報だけを優先的に選択し、提供するための技術として情報フィルタリングがある。

情報フィルタリングによって獲得した情報には情報提供者やそのコミュニティに関するアドレスも保持しておく。これらのアドレス情報は、情報源に向かうための、言わば地図である。その地図を基にして、関心のある情報については、その提供者と直接のコミュニケーション・チャネルを開く。

情報環境の支援のためには、

- ネットワークニュースと電子メールで取り交わされている膨大な情報の中から、個人そしてグループが必要とする情報を、情報フィルタリングによって効率良く的確に収集して分類・整理し、
- そうして形成された新しい情報を利用したコミュニケーション・チャネルのサポートもする、

そのようなシステムを構築することを考えてみる。

一般に情報フィルタリングは、ユーザが必要とする情報を優先的に収集するために、そのユーザのニーズ・興味・嗜好を反映させたプロファイルに基づいて行なわれる。しかし一方で、ユーザ個人のプロファイルのみに依存する場合、ユーザ個人の価値観によって限定された情報収集を繰り返すことになり、新しい知識に出くわす楽しみを味わったり、積極的な探求欲求を刺激したりするのを狭めてしまう危険性もある。この点を鑑みて、このシステムでは、ある個人のプロファイルと他人のプロファイルを組み合わせることによる柔軟なプロファイルの形成を提案する。ネットワーク上での知的生産活動を考えるとき、個人による概念の捉え方や発想方法の違いを積極的に利用しない手はない。

ただしその一方で、プロファイルは個人的な情報を格納することになるので、プライバシーの保護も重要な問題として扱われなければならない。

上記のような個人差をより直截的に利用していく機能も考える。それには、「電子メールや電子ニュースは発信者が誰であるかを明示している」という特徴に着目してみる。それによって、「こういうことが知りたいならば、この人に聞け」というように、情報とその提供者のマッピングが可能になる。このマッピングによって、獲得された情報は、外に開かれたチャネルを持つことになり、そのチャネルを通じて情報提供者とコミュニケーションをとることが可能になるわけである。さらに、そのマッピングをした結果生成される情報地図は、一過性のものではなく、将来も活動グループ内で利用できるようなデータベースとして保持される。このコミュニケーション・チャネルによって、よりユーザ本位の情報検索を進めていくことができるであろう。

システムの構築へ向けてのアプローチと課題

このシステムの研究のアプローチとしては、以下の2つがある。

1. 情報フィルタリングシステムにおいて核となる要素・機構について、それらをどのようなものにするのかを追求する。
2. その核となる各要素・機構に連結した形で機能する、何らかの機構について追求する。

1. のアプローチは、フィルタリングシステムの基本アーキテクチャ内でなされることになり、2. のアプローチは、その基本アーキテクチャを拡張させたシステムを考えることになる。この両アプローチについて、図を用いて説明しておこう。本研究で取り組むシステムのアーキテクチャの抽象図について、一般的なフィルタリングシステムの基本アーキテクチャを元に図示すると、以下のようになる。
この図に書かれている4つのバッファ（一時的に何らかの形で情報が貯められるところ）は、情報源からユーザへフィルタリングを介して情報が配送される過程で用いられるもので、それぞれが以下のような働きをする。

buffer-1 フィルタがかけられる前の情報を保持する。

buffer-2 フィルタリングによって選択された（まだユーザには配送されていない）情報に対して、何らかの処理を施すことができる。

buffer-3 ユーザの手元に配送されてはいるが、まだ読まれていない情報に対し、さまざまな追加処理を施すことができる。

buffer-4 ユーザからのフィードバックを受け取り処理するための、柔軟な処理ができる。

1. のアプローチ

フィルタリングシステムの核となる各要素・機構（上図の点線内の部分）について検討と構築を進めて行く。基本的な方針としては、情報環境支援のためのシステム全体の中での各機能間の連動に留意しながら、既存のフィルタリングエンジンを改変していくことを考えている。その際には、主に以下の課題に取り組むことになる。

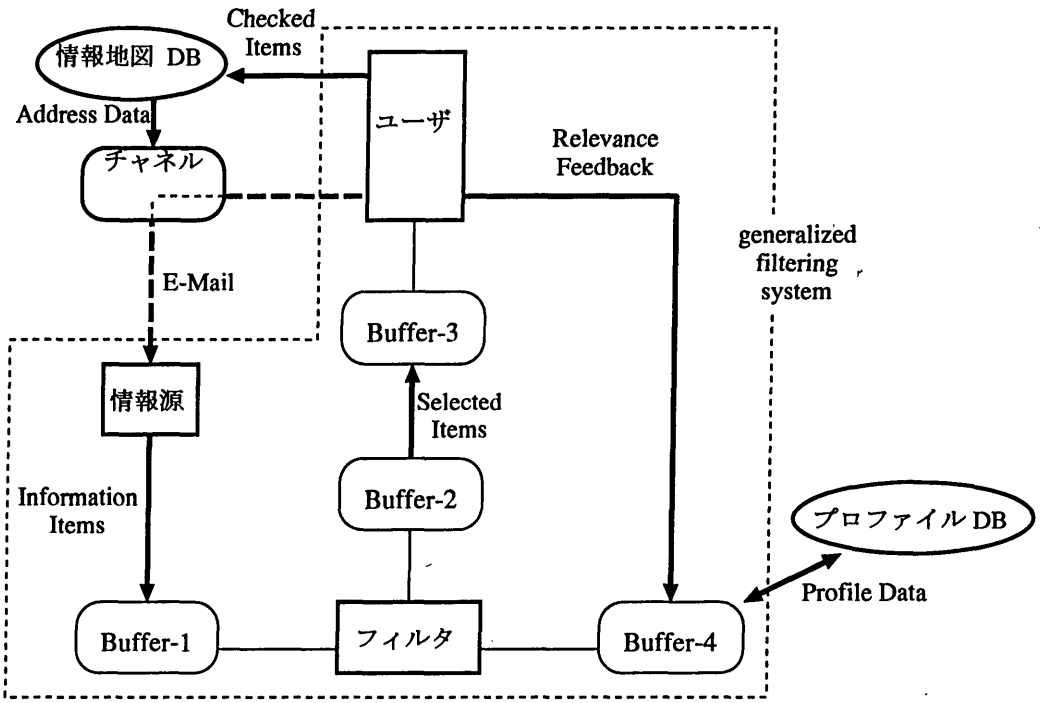

図 2.4: 情報環境支援システムのアーキテクチャ

「フィルタ」機構 — フィルタリングそのものにはどのような手段を用いるのか

プロファイルに基づいてフィルタリングを行なうとき、その方式は、boolean モデル、ベクトル空間モデル、統計的モデルといった手法に基づいたテキスト検索技術を用いることになる。

"buffer-4" — プロファイルの表現はどのようなものにするのか

一般には、プロファイルは、キーワードや規則によって記述される。

「ユーザ」⟶**"buffer-4"** の機構 — フィルタリング結果をプロファイルにフィードバックさせるにはどのような方法を用いるのか

ユーザが情報の価値を採点する、ユーザの振舞いから見出された何らかの規則性によってプロファイルを変更する、といった方法がある。

"buffer-2" — 情報をユーザにとって利用しやすい形に処理しておくにはどうすればよいか

フィルタリングによって選択した情報を、プロファイルとの関連性に基づいて順序付けておく、といった処理が考えられる。

2. のアプローチ

活動グループ内でのプロファイル参照機構や、コミュニケーション・チャネルの構築に取り組む。ここでの課題は、以下のようなものになる。

「プロファイル・データベース」機構 — 個人プロファイル情報を他人が参照できるようにするためには、どのようなデータ構造にするのか、また、プライバシーの保護は如何にするのか

データの構造については、プロファイルの表現方法と共に考える。プライバシーの保護については、ユーザが匿名で個人プロファイルを提供するといった方法が考えられる。

"buffer-4"←→「プロファイル・データベース」の機構 — 個人プロファイル情報を如何にしてやり取りするのか

「ユーザ」→「情報地図データベース」の機構 — ユーザがフィルタリングを介して得た情報から、情報地図を生成するにはどうすればよいのか

「こういうことが知りたいのなら、この人に聞け」という地図の生成には、ユーザの判断を用いることになる。なぜなら、発信者が問題解決のための力を有するかどうかは、記事の内容を理解しないと解らないからである。したがって、情報地図の生成の手順は、

記事がフィルタリングされて手元にやってくる→ 読む → ユーザが知りたかったことが書いてある記事があった → その記事を、「見出し（扱われている問題）・発信者のアドレス・記事本文」といった形でデータベース化する

ということになるが、最後のプロセスのための機構を考えなければならない。

「情報地図データベース」機構」— 情報地図 DB にはどのようなデータが保持されるべきなのか

将来もそのデータを参照できるようにしたい。また、誰がどのことをどのレベルで力になってくれそうかをブラウズする機能や、何度も同じことを同じ人に問い合わせることがないようにする機能も必要である。

「ユーザ」が「チャネル」を介して「情報源」にアクセスするための機構 — ユーザが情報地図 DB を参照にして直に情報源に問い合わせをする方法としては、どのようなデザインがなされるべきか

ユーザが情報地図 DB の中のデータを横目で見ながら、エディタ上でメールを書く、といった方法の場合、「コミュニケーション・チャネル」とはメタファとしての意味合いを持つ。一方、ユーザが情報地図 DB にアクセスしたときに、情報源への問い合わせ専用のウィンドウが開く機能があれば、「コミュニケーション・チャネル」はインタフェースとしての意味合いを持つ。

2.3 グループウェアベース

2.3.1 現状の整理

近年のネットワーク社会の広まりにより、電子メールは電話や手紙、ファクシミリなどに代わる情報媒体（メディア）になりつつある。瞬時に届く、簡単に複数の相手に送る事ができる、受け取ったデータを二次的に利用できるといった、従来のメディアが持っていなかった特徴を持つ事から、それらに置き換わるというよりも、むしろ新たな手段が加わったという表現の方が正確かもしれない。共同作業においては、情報の伝達、不確定な事項の確認、意識合わせといったようなコミュニケーションに利用されているが、先のような特質を活かして、メーリングリストによる電子会議を行なうといった利用形態もある。会議の参加者は時間や場所を同じにすることなく、意見の交換や方針の決定、情報伝達といった一般的な会議の目的を果たすことができる。

このようにして交わされるコミュニケーションを追跡することにより、共同作業にとって利益となる点を見い出そうという動きは以前から存在していた。The Coordinator では、ウィノグラードとフローレスの会話モデルに基づき、構造化電子メールにより利用者が発信するメッセージの種別を明示することで、議論の状況を把握しようと試みた。gIBIS は、議論のプロトコルに従って発言していくことで、問題、案、意見とその間の関係からなる議論ネットワークを利用者に提供するというグループウェアである。やはり議論の流れをつかむのに有効であるが、もともとの IBIS モデルは、ソフトウエア設計の上流工程で交わされるコミュニケーションを、グループ共有の知識として記録して再利用することを目的としていた。

2.3.2 問題提起

前に述べたような、特定の世界や作業手順を明示的にモデル化して設計されたグループウェアには、柔軟性を欠いて汎用的でないという問題がある。つまり、モデルが対象としている範囲にマッチする世界は、うまく扱うことができて効果があるものの、すこしでも対象の範囲を外れるとシステムは無力化してしまうのである。同じように、対象とする範囲内を扱っていても、モデルとして明示化されていない例外的な事象には対応が難しい。

対象世界をモデル化するということは、計算機が扱いやすいように利用者にある種の構造的制約を要求するということである。それらの制約が利用者に受け入れられなければ、システムは利用者を支援できない。また、現実世界の会話や作業は一つのパラダイムでモデル化できるほど単純なものでもなく、個人レベルでさえモデルの捉え方が異なると言える。さらには、会話や作業のプロトコルには国や世代によっては、モデル化の困難な文化やしきたりといった概念が存在する。例えば、ある確認を投げかけて時間が経っても返事がない場合、それは異論がないと見做すこともある。このようなプロトコルはモデル化するのは難しいだろう。

グループウェアベース「菜」は、プロトタイプとしてメーリングリストを利用した電

子会議内容の進捗把握を支援するシステムを実現している。「栞」は、個々のメールの内容に構造上の制約を与えず、管理者がそれらを討議プロセスの概念で分割、編集し、グループで共有する討議空間を構築することで、この問題に対応できるシステムを目指した。その結果、一応の効果を得られるものの、

- 管理者が討議プロセスの編集に労力を要する
- 討議空間の構築には、管理者の主観が反映されてしまう

という点で議論の余地がある。

これらの懸案事項を一層深刻にするトピックスとして、コミュニケーションの範囲がある。一般に共同作業で発生するコミュニケーションは、グループ全員で行なう必要のあるもの、特定のメンバだけが行なうものに大別でき、これはコミュニケーションの内容や、メンバのグループ内の権限や、作業の状態といった要素により変化する。ところがこれまでのシステムやモデルは、共同作業に参加しているメンバ全員がフラットに同じ情報（コミュニケーションの記録）を共有している世界を対象にしていると言える。例えば、共同作業のコミュニケーションにメーリングリストを利用した場合、コミュニケーションの範囲は固定的になり、その情報が全員に必要という場合以外は冗長な情報を消化／共有することになってしまう。さらに、もう少し強い意味では、特定の範囲以外には伝搬してほしくない情報も存在するだろう。このような場合、情報の共有を行ないたい範囲毎にメーリングリストを設けて、それぞれを管理する必要が生じてしまう。

それぞれのコミュニケーションは、それぞれの範囲で行なわれてきたコミュニケーションを前提に発生している。従って、膨大なコミュニケーションの記録の中から、行なおうとするコミュニケーションの範囲に必要なものだけを、そのコミュニケーションの範囲に含まれるメンバ間で共有して、その範囲で行なっている議論の状態を的確に判断できるような仕組みが必要であると考えられる。

2.3.3 問題へのアプローチ

このようなシステムを検討するにあたり、まず各個人の属するコミュニケーションの範囲が任意であることから、管理者が空間の構築（コミュニケーション記録の編集）を行なうという方法はふさわしくないと考える。また、前述の構造的アプローチの欠点も解消したい。すると、理想的には

- コミュニケーションには構造的な制約を伴わない
- 議論や会話の状態を表す空間を自動的に生成する
- グループで変化するコミュニケーションの範囲に柔軟に対応できる

という要件を満たすことが望ましい。

電子メールによるコミュニケーションに伴う制約を極力排除し、会話のプロトコルや議論モデルの導入も行なわないで、議論や会話の状況を把握する、あるいは有効な知識

情報として記録するにはどうしたらよいか?まず第一歩として、電子メールによるコミュニケーションに付随するパラメタから、議論や会話の性質、状態を推測できるのではないかという予測がある。例えば、サブジェクトは議題を、宛先は情報の共有範囲を表しているという直接的なものの他に

- 短い間隔でメールが交わされるのは議論が活発な状態を示す
- 小さいサイズのメールは、確認の要求や、それへの応答である
- 大きなサイズのメールは情報量が多い可能性がある
- 返答の要求に対し長い間反応がないのは、思考の最中であるか、他の負荷が重いためである

といった特性が確認できれば、それを利用者に何らかの形で視覚的に提供することで、議論や会話の状態を理解させることができるのではないかと考える。さらに各個人の持つ情報から空間を構築することで、各個人の視点で捉えた空間を演出できると考える。しかもそれぞれが属している、様々なコミュニケーションの範囲を対象にすることができそうである。

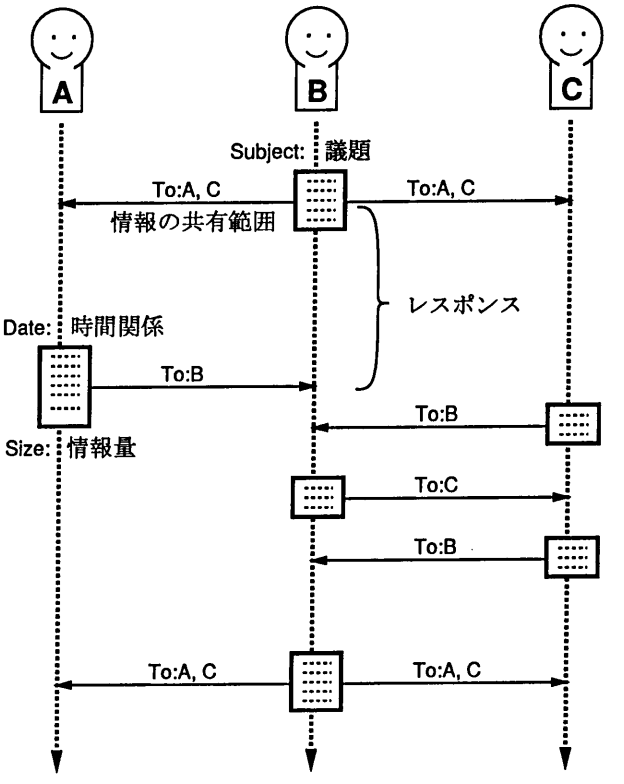

図 2.5: 電子メールによるコミュニケーションに付随するパラメタ

2.3.4 今後の課題

今後、課題として消化していかなければならない項目と、その論点をあげておく。

電子メールによるコミュニケーションと議論の状態の関係

構造的な制約を受けない電子メールによるコミュニケーションについて、それに付随する様々なパラメタと、議論や会話の状態との相関関係を探る。ここでは、特にコミュニケーションの範囲やベースとなる共同作業の内容には拘らずに、何通りかの実験を行なう。その記録を分析して、一つ一つのメールそのものに付随するパラメタ（サブジェクト、大きさ、宛先など）と、それらメール間に存在するパラメタ（レスポンス、大きさの比、宛先の変化など）が、その時の議論や会話の状態と特別な関係が認められるかに焦点を絞る。

空間演出方法（ユーザインタフェース）

電子メールによって進められる議論によって、ひとつの空間が構築されていると捉えて、これを利用者に視覚的に演出することを考える。この際、先の実験／分析結果を踏まえて、より的確な表現を実現するよう注意する。重要なのは利用者がそれを見て、議論、会話の状態を的確に判断できるものでなくてはならないという事である。

システムのアーキテクチャと機能仕様

電子メールは、我々が日々使用している e-mail を使用する予定であるが、そのままのデータでシステムが扱えるか、あるいは最低限の構造化が必要なのかといった見極めが必要となる。それと同時にシステムのアーキテクチャを設計し、必要な機能も見積もっておく。新たなコミュニケーションツールとの融合といった要求にも耐えうるような汎用性も考慮しなければならない。

システム評価のためのモデルの設定

システムの構築に成功した時には、その効果を検証しなくてはならないが、そのためのモデルを設定する。これに「研修ゲーム」と呼ばれる問題を利用できないか模索中である。「研修ゲーム」とは、いろいろなゲームを通して社員の能力開発を図る研修技法で、親近感や相互理解を促進するもの、チームワークの醸成に役立つもの、コミュニケーションを考えさせるものといった分類ができる。もちろん、多くの場合それらは複合している。このような目的はシステム評価のモデルとしては不要であるが、必要な特徴をいくつか兼ね備えていそうなものもある。モデルとして望まれる要素は以下の通りである。

- 権限の異なるメンバでグループが構成される
- コミュニケーションの範囲が複数存在していて、各メンバが複数の範囲に属している
- 事務的な作業ではなく、創造性を問う
- 作業内容が一定ではなく、変化に富む

できあがったモデルを使用して、共同作業の実験を行ない、システムの効果の検証、利点／欠点の抽出を行なう。

研修ゲームの例　以下にコミュニケーションの重要さと、役割分担を理解するための研修ゲームの一例を示す。

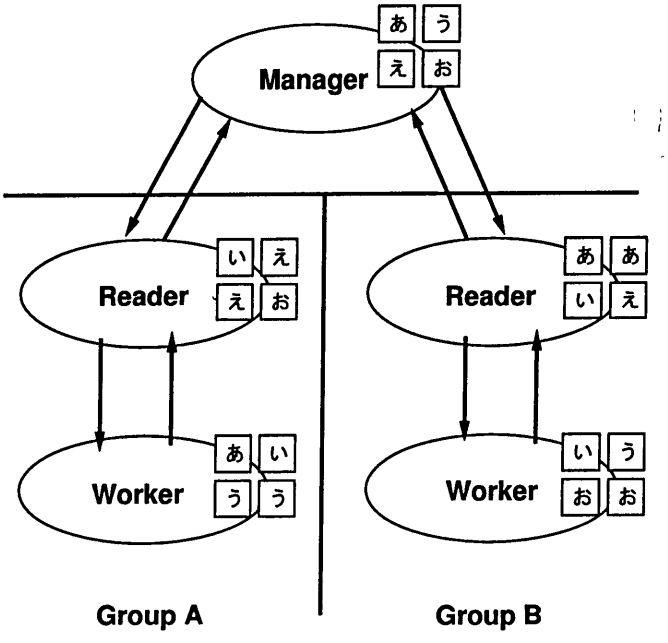

図 2.6: 研修ゲームの一例

ルール：

- このゲームはリーダーと作業員で構成される二つのグループと、それを管理するマネージャで行なう。
- 各員の手元には、文字を書いたカードが適当に 4 枚配られている。このカードは、同じ文字を書いたものが 4 枚存在する。
- カードを交換していき、各員の持っているカードがそれぞれ同じ文字で統一されている状態を目指す。最初にこの目的を知っているのはマネージャだけである。
- 作業員は同じグループのリーダーとしかコミュニケーションできず、各リーダーはマネージャとコミュニケーションできる。
- コミュニケーションの際、一度に交換できるカードは 1 枚のみ。
- リーダーやマネージャは、ある人とのコミュニケーションの内容を第三者に知らせてはいけない。

2.4 プロセスサーバ

開発工程全体を把握、分析、制御するものとしてプロセスサーバを構築し、その特性を評価する。本研究で対象とする開発環境は、DataRun 方法論に基づくケースツールである SilverRun を基礎におき、さらに「コミュニケーションツール」「テストツール」等を基本アクティブとして追加したものとする。研究の方針としては、従来から本研究室で研究されてきたデータ共有機構またコミュニケーションに関する機構等の知見を有効に利用できるような機構をプロセスサーバに組み込むこととする。

図に関し簡単な説明を加える。まず、ケースツールとして図下部にある「SilverRun」を採用した。さらに「SilverRun」に不足分の機能を「CASE ツール基本アクティブ群」として図の中段に描いている。これら基本アクティブ群は、従来、本研究室において開発されてきたツール類をもちい、また、相当する機能が市販品で存在する場合は市販品の購入も検討する。最後にそれらツール類を使用した開発における開発の進行等を把握、分析、制御するものとしてプロセスサーバを考える。プロセスサーバは開発に参加する複数のメンバーからアクセスされるものとし、各メンバーに自分の仕事に関する情報を提供する。さらに開発における管理者には進捗状況と開発工程設計に必要なデータ等も提供できるようにしたい。

なお、本研究では、今回使用する「SilverRun」等のケースツールを始め、MS-Windows 系の OS で動作するツール類が多く存在する。従来からの研究室環境は、Unix を中心とした環境であったため、今回、研究の推進と合わせて、従来ツールの Windows 環境への移行、Unix 環境と Windows 環境の統合等を行う必要がある。

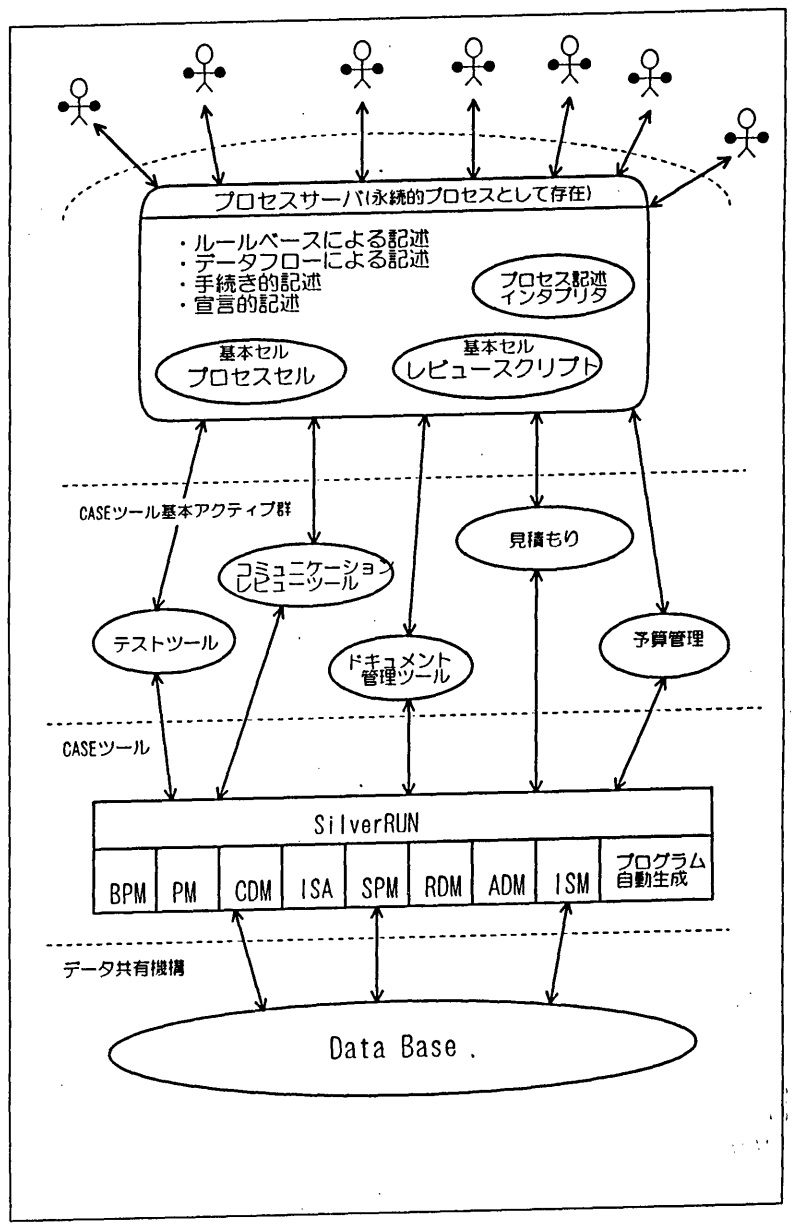

図 2.7: プロセスサーバの実現

2.5 分散サーバ

ソフトウェア開発に置ける共有情報を以下の方式に基づいて管理する。この管理方式は、オブジェクト指向モデルに基づきモデル化される。

1. ワークスペースマネージャオブジェクトを成果物オブジェクトにより、個人の作業の責任範囲を明示化し、また、共有情報のアクセスを制御する。
2. 自律メディエータオブジェクトにより、共有情報の変更に関連したネゴシエーションとそのコーディネーションを支援する。
3. これらのオブジェクトそれぞれにメタオブジェクトを設け、個人の作業の責任範囲や共有情報の内容や範囲の動的変更、作業状況に応じた処置の動的選択を可能にすることにより、作業方針の変更に関する対応を可能にする。

詳しくは、
堀、落水：ソフトウェア開発における自己反映オブジェクト指向モデルに基づく共有情報の管理法
コンピュータソフトウエア Vol.13, No.1 (1996), PP.37-54 参照

2.6 CSCW におけるインタフェースメタファの抽出とインタフェースの設計

2.6.1 目的

現実の会議においては、議長、書記、タイムキーパーと役割が割り振られているのに対し、ネットワークを介した電子会議では参加者は、すべて同じインタフェースを持ったツールを介して会議を行っている。現状では、ツールのインタフェースに人間が振り回されている感が拭いきれない。電子会議においても会議における役割を明確にし、人間の意志決定の助けとなるために、役割や権限に従った情報を効率的を引き出して、発信する必要がある。そのためには役割や権限に応じて最適化されたインタフェースを導入する必要がある。

本研究ではインタフェースメタファに注目し、階層分析法 (Analytic Hierarcy Process 以下 AHP) を用いて役割ごとの インタフェースの機能の重要性を評価し、CSCW における役割に最適化されたインタフェースメタファの抽出を行う。得られた評価結果から仮想現実の技術を背景としたシームレスなインタフェースを提案する。

2.6.2 背景・特色

本研究室では先端的ソフトウェア創出のための研究環境の提案をしており、そのためには知識獲得／共有／利用の支援を行うアクティブチャネル、コミュニケーション (CSCW) の支援を行うグループウェアベース (琹)、共同作業 (ソフト開発、執筆等) の支援を行う分散サーバ (群舞) 等が設計されている。

これらのシステムすべてにおいて、状況に応じた活動メタファを基に設計されたユーザインタフェースが必要である。ネットワークを介した、話合い、作業、情報交換/検索に適した作業メタファを、仮想現実の技術を背景にして、新たに設計する必要がある。また作業状況に応じたコンテキストを提供する機能も必要である。

また現在多く使用されている電子会議ツール、vat,vic,nv では、多くの人数が参加した場合、画面上には多くのウィンドゥが溢れ返り、そのオペレーションに多くの労力が必要とされ、それらの軽減も必要である。

2.6.3 研究計画・方法

電子会議におけるメタファの抽出とそれに基づくインタフェースの提案は以下の手順で行う。

1. インタフェースメタファ抽出実験

電子会議ツールの参加者の役割に応じた機能メタファを抽出するのに以下のような実験を行う。本実験の目的は電子会議においてタイムキーパー、議長用のインタフェースにどの程度の差異を与えればよいかのデータを得ることである。会議を開く事前打合せはメールで行うとする。以下に行う会議の議題は、ある問題に対して創造的な会議がなされるものとあるたたき台があってそれをチェックするものの 2 つを取り上げる。また時間コスト、ストレス、話題の展開、収束を比較するために通常の会議と既存の vat,vic,nv,InPerson 等の電子会議ツールを使用して会議を実施する。実験はタイムキーパー、議長の権限に

以下の 4 つの差異を設けて行う。

(1) 参加者に役割を決めず、会議を行う方法 (通常会議/電子会議)

(2) タイムテーブルに従って進める方法 (通常会議/電子会議)

(3)(2) の条件に、ブレーンストーミングの (自由にアイデアを出し合う) 手法、ノーマルグループ・テクニックの (参加者が解決策を書き出して発表する) 手法を選択して加える。(通常会議/電子会議)

(4)(3) の条件に会議のテンションを保つ手法や課題を明確にする手法を加える。(通常会議/電子会議)

2. 実験データの分析

インタフェースメタファの抽出には AHP を用いる。AHP の特徴は問題対象を問題、評価基準、代替案という階層構造で表現し、評価基準のウエイトを 1 対比較の手法を用いて決定し、代替案を検討する。結果の利用法としては、代替案の中からウエイト (優先度) をもとに多角的に評価することができる。また判断の整合性をチェックすることも可能である。今回の分析では役割ごとの機能の重要性を計る。

3. インタフェースの設計

ウィンドゥの数をできるだけ減らし、仮想現実の技術を用いたシームレスな先進的なインタフェースを提案する。上記の分析より限られた画面空間により AHP で得られたウエイトの数値を基準として、必要な機能を効果的に配置する。スムーズなインタラクティブサイバースペースを実現するために参加者間で交換される情報量をできるだけ抑えたい。また計算機の初心者ユーザにも使い易く、好奇心をそそるエンターテイメント性をも取り込みたい。電子会議のインタフェースは以下のモードから構成される。

(1) 議長・タイムキーパーモード

・タイムテーブル機能・コーディネーション機能

(2) 書記モード

・記録機能・議事録配布機能

(3) 参加者モード

・情報獲得発信機能

2.7 CSCSD サーバ

2.7.1 異種プラットホームが混在する共同作業環境の理想

複数人で計算機を用いた共同作業をするときには、以下のようなことを実現したいという要求がある。

- 各人の利用する計算機やOSなど、プラットフォームが異なっても共同作業したい。
- 各人が離れた場所にいながら共同作業したい。
- 各人が作業する時間が異なる場合にも共同作業したい。
- より効率的に共同作業したい。

ここまでで述べたように計算機環境を構築すると以下のような概念図になる。

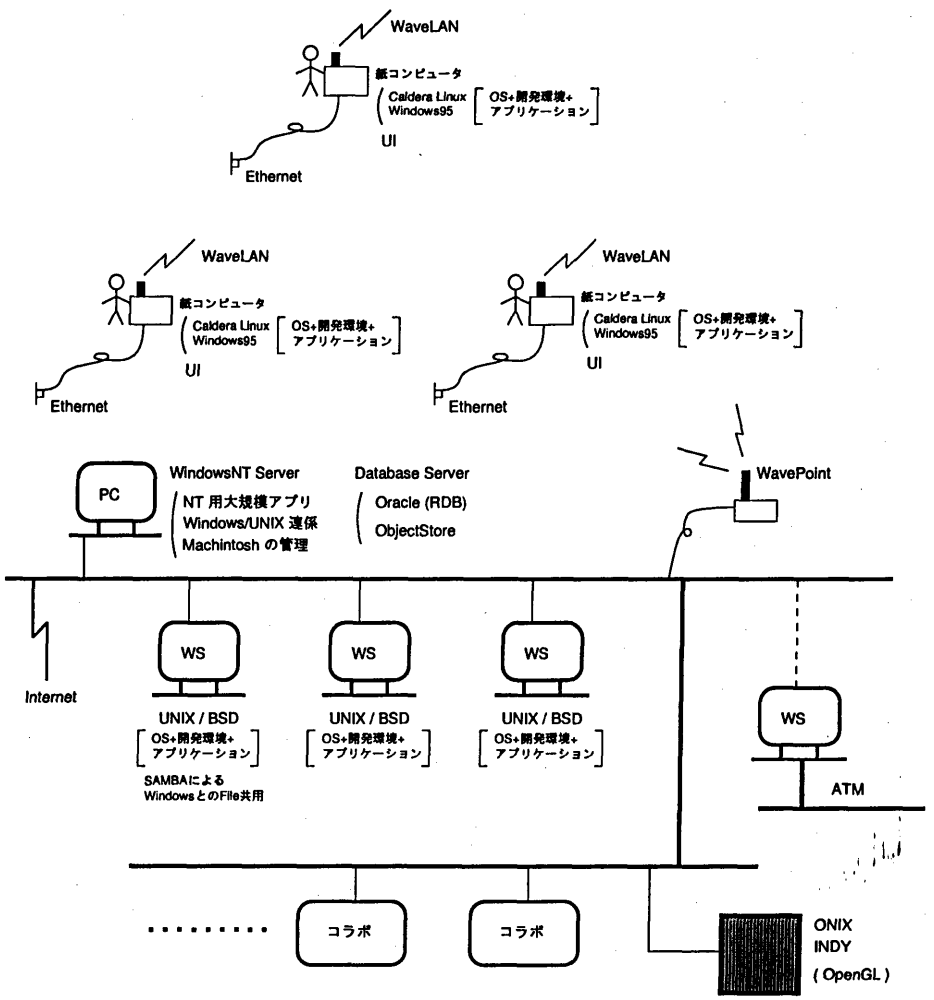

図 2.8: 落水研究室計算機環境の概念図

2.7.2 アプリケーション実行状態を示す仮想ファイルシステムを用いたサーバによる端末間移動の透過性の設計と実現

すでに、WABI,WINE, samba 等により UNIX 上での Windows アプリケーションの実行や、ファイルの互換性等の支援は実現されているが、CSCSD サーバはそれらの機能を前提として、作業環境についてもその連続性を保証するものである。

上で述べた環境において、違う端末間を自由に移行したり、その間の連続性を保つということをを実現するには、以下で述べるようなことが必要である。

1. CSCW 機能群のマッピング
   前述した機能群を現状で最適と思われるもので実現をする。ここでは、本研究にとくに関係の深いものについて述べる。
   ノート PC には、その OS として、Linux および Windows95 、WS 側には、UNIX を利用する。また、あらゆる場所で、ネットワークに接続できることを実現するために、無線 LAN の利用も考えられる。
   また、開発環境、コミュニケーション環境などは各々に応じたものを使用する。

2. 現況の分析
   ノート PC,WS などが混在し、それらを時と場合に応じて使い分ける環境となるため、いつでも、作業ベースの変更を行なえ、かつ、その間の連続性が保証されなければならない。そこで、既存のソフトウェアを用いることで、解決可能な機能を洗い出し、その上で不足している機能を CSCSD サーバとして実現する。現状では、データの受渡し、データ形式の変換等は容易に行なえるが、それを連続的にすることは難しい。

3. CSCSD サーバの設計
   既存のものの組合せでは、データの受渡し、データ形式の変換等は容易に行なえるが、それを連続的にすることは難しい。また、プロセスマイグレーションや仮想マシンの利用では重いという問題がある。そこで、CSCSD サーバでは、アプリケーション自体はそれぞれの環境で動くものを用いる。環境の移行に必要な情報はアプリケーションの種類、抱えているファイル名とその位置、状態である。ここで、状態とは例えば、エディタであれば、編集中の場所等がある。また、移動元、移動先の情報を得られるようにそのための情報も必要である。これらを一括して管理する CSCSD サーバ を各 WS/PC で動かし、各サーバ間の通信により移動をサポートする。

4. 上で設計した CSCSD サーバをデザインパターンに基づき、できる限り、拡張が容易な形で実装する。初期の段階では最低限の機能のみを用意し、各拡張機能はモジュールの形で実装できることが望ましい。

5. このサーバに必要な機能の確認の手段として、以下のような環境が実現できるかどうか、ということを判断の材料として利用する。

   - 文章/図作成
     (1) 作業中の環境を任意に ノート PC/WS へ相互に移行できる。

(2) 共同編集中にある参加者から全 (あるいは特定の) 参加者へ同じ画面を見せる。
(3) 共同作業中に作業ベース (マシン) が変わった時、その環境の自動転送。
(4) 別のマシンからあるマシン上の自分の環境を呼び出す。
(5) 異種データ形式の変換。

- プレゼンテーション環境
  (1) Windows で作成したプレゼンテーションのデータを UNIX で表示する。
  (2) 会議参加者の手元の WS / NotePC へ任意の画面を表示させる。

- 開発環境
  (1) Windows 環境で作成したものを、UNIX で使用する。
  (2) 異種プラットホーム間で共通のソースコードを使うためのツールキット。

- コミュニケーション環境
  (1) Windows 用グループウェアとの接続。

このサーバが実現されることによって、携帯型端末と固定型端末 (WS) 間、異なるプラットホーム間での作業環境の移行容易性を実現できる。

2.8 パターンを利用したオブジェクト管理機構の設計と実現

2.8.1 Java 言語の特性

Java 言語は 1991 年に Smalltalk,Objective-C 等をルーツとしてサン・マイクロシステム社により開発された。当初 Java は個人向け小型遠隔制御装置用のソフトウエア開発を目的に開発されたが、途中からターゲットを WWW に切り替えた。1995 年 5 月 Java インタプリタを内蔵した HotJava が公開され、動的な Web ページの記述により注目を集めた。また Java はプラットフォームに依存せずに実行可能なため、ネットワーク時代の開発基盤として注目を集めている。

大きな特徴として以下の項目を挙げることができる。

1. 機種非依存で動作する
   Java 環境では Java コンパイラ (javac) によりソースコードをプラットフォームに依存しないバイトコードに変換する。変換されたバイトコードは Java インタプリタが用意されたすべてのプラットフォーム上で実行することが可能となる。Java インタプリタはバイトコードを解釈し実行をする。

2. 標準のクラスライブラリが存在する
   Java には標準のクラスライブラリが存在するため、C++にあったような互換性のないクラスライブラリに関する混乱は生じない。

3. マルチスレッドが使用できる
   Java は言語レベルでマルチスレッドに対応しているため、ネットワークプログラミング等、用途によっては他言語を使用するよりも、より自然にソフトウエアの振る舞いを記述することができる。

4. ネットワーク上での使用を考慮
   World Wide Web ブラウザに組込まれた Java インタプリタは、アプレットといわれる Java バイトコードを実行する。アプレットは Web サーバからクライアント側に転送され、クライアントで実行される。このため、サーバとネットワークの負荷を削減することが可能となっている。

5. セキュリティについての考慮
   Java プログラムは、アプレットとして Web サーバからダウンロードされ、クライアント側で実行されることがあり、セキュリティ上の問題が発生することが考えられる。そのため、Java プログラミング言語には厳しい制約条件が課されている。Java 実行時インタプリタではバイトコードがどの言語要件にも違反していないことを検証している。とくに特に WWW ブラウザに組込まれた Java インタプリタについては、実行時に厳しいチェックが行われ、クライアントの側ファイルアクセス禁止、クラス名とアクセス権限の検証等を行っている。また将来のリリースでは公開鍵暗号化 (public key encryption) をサポートする予定である。

6. 構造の簡略化
   オブジェクト指向への自然な対応と、さらにセキュリティとロバストを考慮して

C++の構文を変更を加えている。主な変更項目を示すと次のようになる

- ポインタがない・多重継承ができない
- 計算子のオーバーロードができない
- struct,union,enum がない

これらはプログラマによるミスを軽減し、不正アドレス領域へのアクセスを制限する。

2.8.2 Java 開発環境の現状と問題点

1996 年 3 月までに公開された代表的な Java 開発 (支援) 環境について分析する。当初、開発環境は Sun Microsystem 社により供給された JDK のみしか存在しなかったが、最近では多様な開発環境が発表されはじめている。それぞれの環境は、その動作方式により次の 2 つに分類できる。

1. 各プラットフォームに依存するもの
   各プラットフォーム用に開発環境が発表されている。仮に Java 開発環境を新しく開発する場合、Java コンパイラやデバッカ等を含めるために、Sun Microsystem 社より Java のライセンスを取得し、各機種用のネイティブコードにより開発環境を構築することが必要になってくると考えられる。これら開発環境は各プラットフォームのコードで動作するため、速度や機能 (各プラットフォーム独自の機能が使用できるため) については有利に構築できる。しかしながら、すべてのプラットフォーム上で同様な環境を提供することは困難である。なお、Sun Microsystem 社が提供している JDK 自体は機種依存のため、このカテゴリに含まれる。

2. Java 言語を用い開発され、プラットフォームに依存しないもの
   Java 言語自身を使用し開発され、プラットフォームに依存せずに動作できる。そのため、Java 言語が動作するプラットフォームであれば、共通した開発環境を導入することが可能となる。Java を使用して開発された開発支援ツールの代表には、「Taikade」をあげることができる。なお、Java で開発されているため、プラットフォーム上のネイティブツールよりも処理速度上不利になる場合も存在する。

これらの開発環境の支援目的を分類すると以下のようになる。ここであげる開発環境は以下の目的のうち一つまたは複数をもつ。

コード再利用の促進　クラス階層の整理とブラウジングにより、利用目的に添った既存コードの再利用を促進する。クラスの継承構造から、または、クラスの分類構造からの検索をソフトウエアにより支援し、グラフィカルなユーザインターフェースを採用することにより、エディタのみの環境と比較して使いやすくなっている。また、Taikade 等では履歴管理機能を付加することによって、開発進行に対応した再利用・検索が可能となっている。

しかしながら、現在まで発表された java 開発環境は、管理手段が「クラスと継承」および「ユーザによる分類」のみにしか対応していないため、協調して振る舞うクラス群

等に対する管理ができない。また、「ユーザによる分類」は客観性に欠け、組織的な再利用を促進する上で問題がある。

統合的開発環境　コンパイラ、リンカ、デバッカ、エディタを統合することにより、一つ一つのツールがもつ能力の合計以上の能力を引き出そうとするアプローチである。各ツールが継ぎ目のない密な開発環境を構築するため、成功すれば各ツールの力を最大限に発揮できる。しかしながらその構造上、閉じた環境になりやすくユーザが新しいツール等を追加または置換することは難しい。

なお、この統合的開発環境は、しばしばグラフィカルな開発環境を伴う。

ビジュアルプログラミング　グラフィカルな概観をもつプログラムの作成を支援する。プログラマはCRT上でプログラムの外見をインタラクティブに作成でき、その部分に対し動作を記述できる。外見のデザインはプログラムコンポーネントを配置することによって行うことができる。ただし、現状のjava開発環境で完全にサポートしているものはない。

以下に、1996年3月までに発表された代表的なjava開発環境を示す。

JDK(Java Developeers Kit)

Java言語による開発に必要なツールをまとめたツールキットである。Sun Microsystem社よりリリースされた。このツールキットを使用することにより、アプレットやJavaアプリケーションを作成できる。

JDKには、コンパイラ、デバッカ(プロトタイプ)、アプレットビュワー、サンプルアプレットが含まれる。

JDKは以下のプラットフォームで動作する。
Solaris 2.x(SunOS 5.x),Linux,BSD Unix,FreeBSD,HP/UX,IRIX, Windows NT/Window 95,MacOS,AIX

JDKではエディタ等で作成したプログラムをコマンドライン上でコンパイルし、アプレットビュワーで実行する。グラフィカルな開発支援環境もなく、非常に基本的な構成となっている。JDK自体は各プラットフォーム上のものを用意する必要があるが、作成後コンパイルしたプログラムはプラットフォーム非依存で使用できる。

なお、コンパイラについてはJDK付属品以外にもサードパーティからさらに高速なものが出ている。

Taikade(元dejava,The environment informally known as dejava)

クラスブラウザを中心に置いたJava開発支援ツール。ソースの履歴管理機能を持つ。株式会社PFU(町田)により開発されたフリーウエア(ソースコード付き)。

クラスブラウジング機能を利用し、コードを検索、カットアンドペースト等を用い再利用できる。またクラスの継承構造(ハイアラーキー)を表示でき、クラスの再利用を支援する。作成したソースはTaikade上でコンパイル・実行でき、コンパイルエラーにつ

いてはソースの該当部へジャンプするなどの機能を持つ。ただしデバッカがないため、taikade の支援はコンパイルまでであり、実行時のデバッグ支援機能はない。

なお、Taikade の操作環境はある程度の視覚化が進んでいるが、プログラムの構成要素を視覚的に構築することはできない。

以下に Taikade 付属ドキュメントから Taikade の特徴を抜粋したものを示す。

- Java のソースファイルをプレーンテキストのまま参照および修正
- クラス／メソッドをベースとした Smalltalk ライクなブラウザ
- 特定のクラス／パッケージ／プロジェクト単位でのオンラインコンパイル
- main(String[]) を持つクラスのオンライン実行
- AppletViewer を用いた Applet の実行
- grep, 行番号でのジャンプといったファイルベースのツール群
- Smalltalk ライクな変更履歴管理機構
- プロジェクト／パッケージ単位のデフォルトヘッダ
- .properties ファイルによるプロパティー設定

Symantec Espresso(tm)

Windows95/NT 用の開発環境。Symantec C++ 7.2.1 へのパッチが download できる。

JAVA MAKER

Win32 上の Java 開発環境。クラスビューワ&エディタ

Cosmo

SGI 社による VRML, Java, JavaScript の統合開発環境。視覚的な開発環境をもつ。

- Java ランタイムインタプリタおよびコンパイラ
- グラフィカルソースデバッガ
- ビジュアルソースブラウザ
- Web ビジュアルビルダ
- Cosmo Motion Engine および MediaBase(TM) ライブラリ

Java WORLD

Borland 社の Java 開発環境。現在はデバッカが公開されている。Borland 社の Web ページ上でのアンケート調査の項目から推測すると、将来、同社の java 開発環境は以下のような項目を考慮に入れてくると考えられる。

- データベースとの連携（リレーショナルおよびオブジェクト指向データベース）
- コンポーネントを利用したプログラミング (OCX、アプレット)
- 同社のパスカルコンパイラ Delphi と同様な、プログラムを視覚的に構築する技術
- ビジュアルな開発環境

2.8.3 Visual 開発環境

Visual 開発環境と言った場合、以下の２つの意味がある。

1) 操作環境がビジュアル

2) 操作対象がビジュアル

1) の操作環境がビジュアルであるツールは、ウインドウシステムの普及により一般的になった。2) の操作対象がビジュアルであるプログラミング環境は最近になって商用システムで見られるようになってきており、商業的に成功しているプログラミング環境も存在する。

この操作対象がビジュアルであるプログラミング環境では、対象プログラム構成要素のオブジェクトを適切に配置あるいは適切なタイミングで生成消滅させる。またそれらオブジェクト同士を適切に協調作用させることによってシステムを動作させる。協調動作はシステム内の各オブジェクトが提供するサービスを組合わせることによって行う。

IvarJacobson の分類 [Jacobson1992] によると、システム内に存在するオブジェクトは以下の３種類に分けられる。

1. インターフェースオブジェクト
   システム外部との連係を取るオブジェクト。外部データベースエージェントや GUI 部品オブジェクト等が含まれる。
2. エンティティオブジェクト
   構築しようとしているもんだ磯のものの本質を表現しているオブジェクトであり、ユーザインタフェースや表層的なアプリケーションの都合によって影響されない部分である。
3. コントロールオブジェクト
   上記２層に挟まれエンティティオブジェクト上に定義されたサービスを利用し、アプリケーションが必要とするより高度な機能を実現する。

ビジュアルプログラミングでは上記のうち、おもにインターフェースオブジェクトについて部品化され、視覚化されることが多い (ボタン、リストボックスなど)。この部品化オブジェクトは外界に対し以下の３つのインターフェースを持つべきと考えられている。

- 自分自身が外部に提供するサービスのインタフェース
- 内部状態の変化を外部に与えるためのインタフェース
- 自分自身が正しく機能するために外部に要求するサービスのインタフェース

操作対象がビジュアルな開発環境では、現在以下に示す項目で相違のある２種類のビジュアル化戦略が存在する。

部品配置戦略

- 部品配置を行うためFormと呼ばれる部品配置しなければならない環境(VB,Delphi等)
- 部品編集用の場所を提供し、その上に多様な部品をくみ上げていく環境を提供する。このためウインドウを持たない(目にみえない)部品もビジュアルに構築できる(Visual Smalltalk等)

部品の関連に関する戦略

- 部品間の関連についてビジュアルな支援がないもの。部品の配置後それらの関係はコードによって記述される(VB,Delphi等)
- 部品間の関連をビジュアルに扱う。そのため部品間の関連は単に線のつなぎ替えで間に合う(Visual Smalltalk等)

Javaの開発環境を考えた場合、まだ操作対象がビジュアルであるような完全な開発環境は存在しない。ビジュアルな開発環境の完成が望まれる。しかし仮にビジュアルな開発環境が存在したとしても、ビジュアル開発環境では、エンティティオブジェクトとコントロールオブジェクトに対する再利用支援環境が十分とはいい難い。何らかの方法で再利用をサポートする必要を感じる。

2.8.4 デザインパターンを利用したJAVA開発環境

研究の目的

オブジェクト指向プログラミング(OOP)時の再利用性向上で問題となる(1)オブジェクト間の関連に関する理解、(2)クラスライブラリからの必要オブジェクト群検索の２点を支援するため、オブジェクト管理機構の設計と実現を行う。オブジェクト管理にパターン抽出を利用し、ソースコードレベルでの再利用性向上を目指す。

本研究では、対象開発領域をケースツールおよびグループウエアに絞り、対象領域用ソースコードに対して再利用性向上と実例収集を目指す。また対象言語として、マルチプラットフォーム開発の必要性から、Java言語を採用する。

抽出パターンとして本学の東田により提案されたpartyを採用する。さらに設計レベルパターンであるデザインパターンに関する成果を導入し、partyを拡張する予定である。

研究の背景・特色

ソースコードレベルでの再利用性を考えた場合、再利用のコストが問題となる。OOPで再利用上問題となるのは、(1) 再利用するための問題領域理解 (オブジェクト間の関連に対する理解)、(2) クラスライブラリからの必要オブジェクト群検索の２点である。これらを改善するため、OOP で典型的に現れるパターンを抽出し、それらをパターンとして再利用するアプローチが生まれた。

OOP で繰返し現れる設計は、デザインパターンとして体系的にまとめられた。プログラム設計者は、システム構築の際、開発メンバー間で共通する設計知識として、再利用可能なパターンを利用できる。しかしながらデザインパターンでは、その目的を設計の再利用においているため、ソースコードレベル再利用が困難であった。

そこでソースコードレベル再利用を検討する必要が生じた。ソースコードレベルのパターン利用による再利用性追求は、すでに本学の東田による研究が存在するが、デザインパターンに関する研究の進展、プログラム開発環境の変化により、さらに強力な再利用機構の検討が必要となってきている。

研究計画・方法

パターンに基づくオブジェクト管理機構の構築のために以下のフェーズに沿い、研究を推進する。

(1)Java クラスライブラリに対する Party 群抽出　本学の東田 (1994 年度卒業) により提案された Party に基づくオブジェクト管理機構を Java 言語環境に移行し、Java クラスライブラリソースコード (基本パッケージ、汎用機能パッケージ、GUI 部品パッケージ等、計 800Kbyte) に対し適用する。適用によって Java における party 抽出の有効性を検証し、さらにクラスライブラリ中の Party 群の is-a 関係を抽出する。Java 言語環境の性質を party より抽出された is-a 関係によるパターン連鎖という観点から検証する。

(2) ケース、グループウエア分野に対する Party 群抽出　Party に基づくオブジェクト管理機構を、ケース・グループウエア分野のプログラムに対し適用し、Party に関する実例を蓄積する。蓄積した実例により、この分野における Party の有効性評価と分野の特徴を導く。

なお、解析対象とするプログラムには、本年度研究室で新規に開発されるツールを含め、抽出した Party によるパターン連鎖に対する知見を、プログラム開発にフィードバックすることにより、開発時における Party の有効性について検討する。

(3) デザインパターンの実例収集および party との比較検討　本研究室の新規開発プログラムをデザインパターンに準拠し設計することにより、デザインパターンの適用事例を収集し、そのソースコードを分類する。さらに収集したソースコードから Party の抽出を行い、デザインパターンと、Party によるパターン連鎖との関連性を検討する。なお、必要に応じて Java クラスライブラリ等の各種 Java コードからもデザインパターンの抽出を行う。

プログラムの設計レベルにおける各要素の協調動作とソースに現れるオブジェクト群の協調動作を比較検討することにより、より抽象的なデザインパターンと具象的なPartyとの関連性を引き出し、partyによるパターン連鎖に設計レベルの情報を付加する方法を検討する。

(4) オブジェクト管理機構の改良および評価　フェーズ(1),(2)および(3)によって得られた知見により、オブジェクト管理機構を改良する。Partyに関する新しい知見が得られた場合には、それに基づきオブジェクト管理機構を改良し、クラス間の関係に関するより高い理解性と必要クラスのより高度な検索性を実現を図る。デザインパターンに対する知見が得られた場合は、それに基づきオブジェクト管理機構を拡張し、partyによるパターン連鎖の抽象性を高め、最終的に設計レベルの情報を導出することを目指す。また、開発用言語としてJavaを採用するため、Javaに特有な性質に対応したオブジェクト管理機構の実装を行う。

最後に、実装したオブジェクト管理機構により、研究室で開発したツールを分析し、次年度以降、新しく参加する開発メンバに対し、ツール開発者が直接引継ぎを行わなくて済むようなソースコード再利用支援環境を構築し、評価する。

付録 A

データ会議の標準規格の動向

A.1 はじめに

今回のレポートでは、テレビ会議やデスクトップ会議、データ会議と称される様々な形態の遠隔会議について、その標準規格の動向を調査した結果を報告する。これに先立ち、混乱を避けるために用語の説明を行なっておく。

テレビ会議 分散した会議室を専用線で結び、テレビカメラ、ドキュメントカメラ、モニタ、マイク、スピーカといった設備を使った、大がかりなシステムによる会議を指す。画像や音声の品質が高く、大人数の会議に適している一方、特別な会議室にシステムを設置するなどコストがかかる。また、地点間でデータの転送ができないといった欠点もある。

データ会議 ネットワークに接続された計算機を用いて、互いの映像、音声のやりとりに加えて、同じアプリケーションやホワイトボードの共有が行なえる。リアルタイムのコミュニケーション能力を兼ね備えたグループウェア・ソフトと言い表せる。画像や音声の品質は高くないが、パソコンで導入できるためコストが少ない。そのパソコンで扱うデータも会議で利用できることが、「データ会議」や「デスクトップ会議」とも呼ばれる所以である。

A.2 標準化の経緯と、関連する組織

A.2.1 IMTC の歩み

現在、データ会議に関係する標準規格はITU(International Telecommunication Union)で勧告作業が進められているが、その標準化はIMTC(International Multimedia Teleconferencing Consortium)が中心となって行なっている。このIMTCのメンバーには、AT&TやBritish Telecomなどの通信業界の企業に加え、Compaq,Intel,IBM,Microsoft,Novellといったコンピュータ業界の企業も数多く含まれているが、このような体制になるのには、次のような経緯があった。

1993 : マルチポイントデータ会議の標準規格として、ITU T.120 シリーズを作成するために、CATS(The Consortium of Audiographics Teleconferencing Standards,Inc.) が

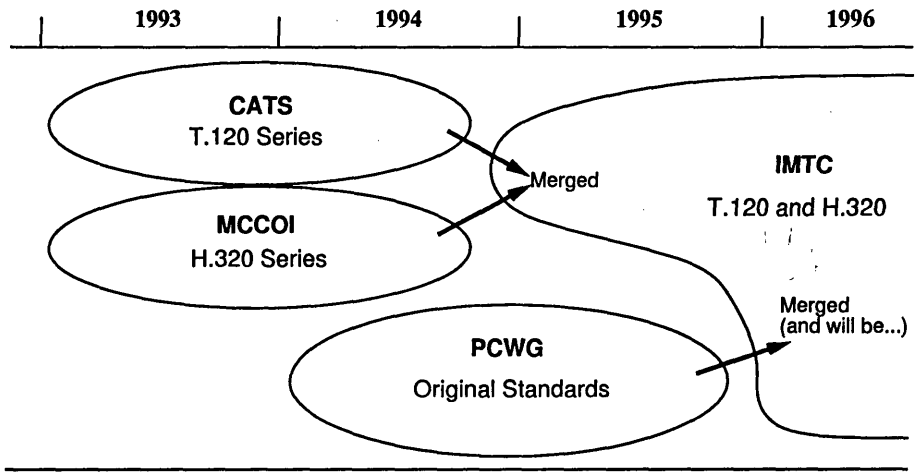

図 A.1: IMTC の経緯

形成された。約 30 の関連企業のサポートを結集することに成功した。

1993 : 同じ 1993 年に、マルチメディア協調作業アプリケーションの発展と、相互動作性のための標準規格を広めるために、MCCOI(The Multimedia Communications Community of Interest) が結成された。MCCOI は特に、画像データ圧縮方式の標準規格である ITU H.320 シリーズ[1]に焦点を合わせて活動してきた。世界各国 50 社以上の企業の参加を受け入れることができた。

1994(January) : 相互動作可能なデスクトップ会議システムの標準仕様作りを目的として、Intel が中心となって PCWG(Personal Conferencing Work Group) を組織した。それまで Intel は、独自仕様のシステムである ProShare1.0 を販売していたが、1994 年末に標準仕様の第一版を完成させ、これに準拠した ProShare2.0 を 1995 後半に出荷する見通しを立てた。ITU によるデスクトップ会議システムの標準仕様作りが遅れていたことから、145 社もの関連企業が PCWG に参加し、この仕様が業界の事実上の標準となる可能性が高まった。

PCWG の標準仕様の第二版では、既存のテレビ会議システムの画像符合化方式である、H.261 を仕様に盛り込むことになったが、これは PCWG に参加している通信機器メーカやテレビ会議システム専業メーカの強い要望があったためである。しかし第一版で符合化方式に採用していた、Indeo Video C3.0 も引続き仕様に含められるということだった。パソコン同士の接続には、こちらの方が適しているからである。

1994(September) : 二つの標準化団体、CATS と MCCOI が合併して IMTC(International Multimedia Teleconferencing Consortium,Inc.) となった。引続き ITU の標準規格である T.120 と H.320 シリーズの標準化を進めていくことになった。最低限この二つの規格

[1]従来のテレビ会議のデータ圧縮の標準仕様である H.261 を ISDN の帯域に拡張したもの。MPEG1 が採用されている。

を満たしていれば、異なるベンダーのアプリケーションが同じデータ会議で相互動作することを目指している。

1995(February) : ITU の国際会議において、従来の画像符合化方式である H.261 と、事実上の標準になりつつあった Intel 社の Indeo Video を一本化する方向で、H.320 の採用が決定された。これによりデスクトップ会議の普及に弾みがつくという期待の声が高まった。

1995(October) : Intel は、H.320 に対応した ProShare の拡張版の提供を開始した。これにより大規模なテレビ会議システムや、PictureTel [2]の製品との相互接続が可能となった。さらにオプションにより 3 地点以上の接続が可能になった。

1995(December) : IMTC と PCWG が合併して、一つの組織 (IMTC) になることを発表した。これによりエンドユーザに、互換性のあるデータ会議製品を供給するという共通の目標に向かって協力しあって行くことになる。

A.2.2 関連企業の動き

このような標準化の波に合わせて、関連する企業も以下のような動きを見せている。

PictureTel

テレビ会議システムの市場において 70%のシェアを持つといわれる最大手。ルーム型と呼ばれる大規模なシステムから、パソコン LAN 向けのシステムまで製品構成を広げてきているのに並行して、H.320 への対応も行なってきた。また、他企業と以下のような連係も展開している。

- Compaq は PictureTel のパソコン向けシステムを自社の製品 (デスクトップとノート) に組み込んで販売。
- NTT と提携し、N-ISDN に対応したボードを開発。Windows95 と CCD カメラ、データ会議システムをセットにして販売。
- Microsoft ととの、Windows 用データ会議 API の共同開発に着手。T.120 に準拠したものとなる可能性あり。

Intel

主にパソコンによるデータ会議システムをターゲットにしており、主力製品の ProShare も ISDN や LAN による、1 対 1 を基本とした製品である。画像の符合化には独自の Indeo を採用していたが、H.320 に対応した拡張版も提供している。

- ProShare の日本語版は NEC、富士通、東芝などが間接販売している。

[2]米のベンチャー企業でテレビ会議システムの最大手。

- AT&T,Lotus と提携し、AT&T の ISDN 回線と Lotus Notes を使ったデータ会議システムの開発を目指している。

その他の企業

Apple は Quick Time Conferenving Kit という製品により、自社の Mac 同士を接続したデータ会議システムを提供している。また、NTT と組んだシステムの販売も行なっている。Olivetti は British Telecom と、H.320 に準拠したデータ会議システムを共同開発し、これは世界各国の ISDN に対応可能なシステムとなっている。

A.3 T.120 と H.320 シリーズの概観

データ会議の国際標準規格の核をなす T.120 と H.320 について、その概観を以下に示す。ただし、ここでは主に T.120 シリーズに注目している。

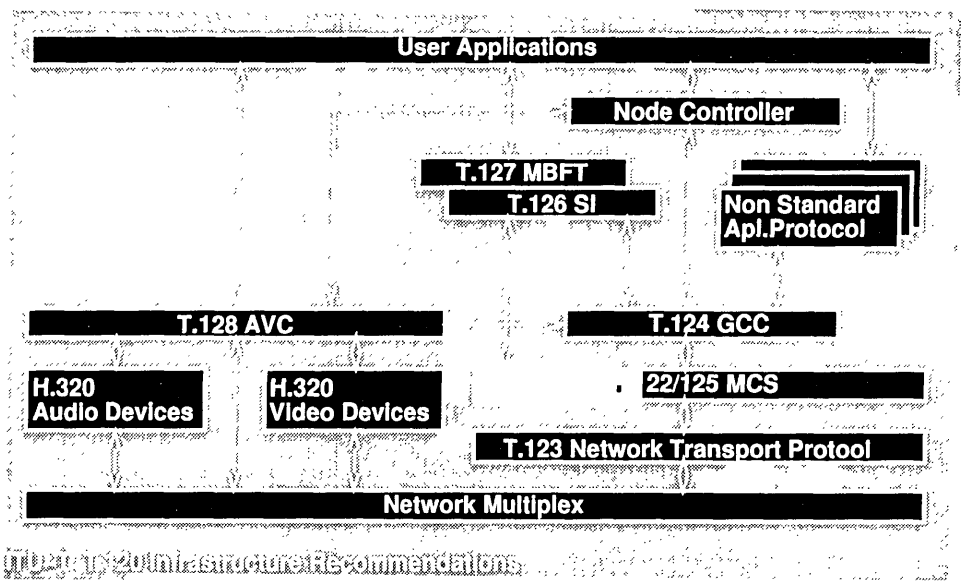

図 A.2: T.120/H.320 の概観

T.120 - マルチメディアデータの転送プロトコル:Working Draft

多地点間で、しかもその接続形態を限定しないでマルチメディアデータを利用した会議を実現するための、コミュニケーションプロトコルを定めている。

T.121 - アプリケーションテンプレート:Working Draft

一般的な T.120 のアプリケーションの雛型を提供し、アプリケーション作成者の負荷を軽減する。

T.122 - データ会議のためのマルチポイント接続サービス (MCS):Adopted

Multipoint Communication Service(MCS) は、多地点間で接続されたデータ会議のアプリケーションの、全二重、マルチポイントのコミュニケーションのサービスを定義する。各地点を接続しているネットワークの種類は、T.123 で特定するものが許される。

T.123 - データ会議のためのプロトコルスタック:Adopted

T.123 は、様々なネットワークの PDU(Protocol Data Units) をサポートし、トランスポート層として OSI 標準のサービス (X.214) とプロトコル (X.224) を、上位層の MCS に提供する。

T.124 - 会議コントロール:Adopted

会議の開催や終了といった、会議のコントロール機能を提供する。また、会議を構成しているノード（地点）や動作している会議アプリケーションの情報などを格納しているデータベースの管理も行なっている。

T.125 - マルチポイント接続プロトコル:Adopted

マルチポイントのデータ転送のプロトコルの定義であり、これによって T.122 のサービスが実現される。

T.126 - マルチポイントの静止画の共有とアノテーション:Adopted

多地点間で静止画（イメージ）データのやりとりや、アノテーションの付与について定めたプロトコル。特に異なるベンダー間のアプリケーションが同一の会議内でデータの共有や、相互操作が可能となるように配慮されている。いわゆるホワイトボードの共有機能がこれに当たる。

T.127 - マルチポイントのファイル転送:Adopted

会議内の任意の端末間、あるいは全体に対してのバイナリファイルの転送を定めたプロトコル。

T.128 - オーディオ・ビジュアルコントロール:Working Draft

マルチポイントにおける、双方向のオーディオ・ビジュアルサービスを制御、管理する。これらの機能はツールキットという形で提供される。

A.4 DataBeam 社のツールキットについて

IMTC のメンバーでもある DataBeam 社は、CCTS(Collaborative Computing Toolkit Series) という製品を提供しており、T.120 に準拠したアプリケーションの開発のひとつの解決策となっている。同社自身もこれを利用してデータ会議システム FarSite を開発、販売している。この他に、Apple,Microsoft, SONY,Xerox 等がツールキットのライセン

ス契約を結んでいる。現在、DataBeam 社が社外に提供しているツールキットは以下の四つである。

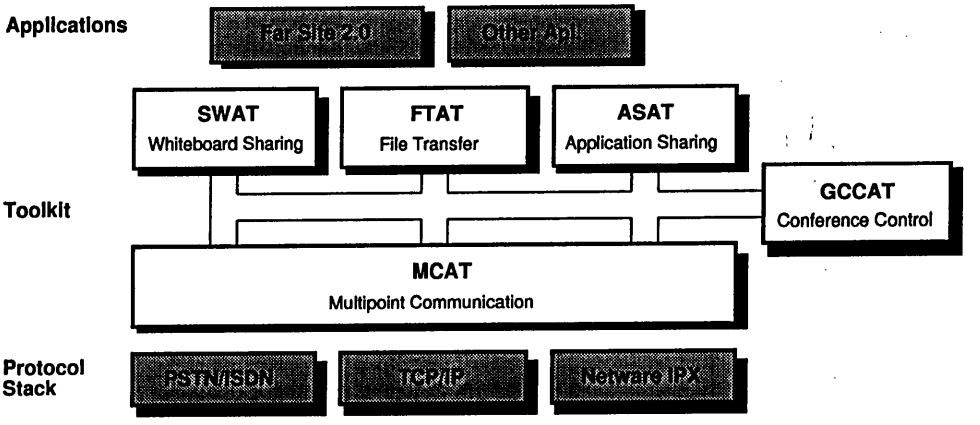

図 A.3: ツールキットの構成

- MCAT(Multipoint Communication Application Toolkit)
- GCCAT(Generic Conference Control Application Toolkit)
- SWAT(Shared Whiteboard Application Toolkit)
- FTAT(File Transfer Application Toolkit)
- ASAT はリリース準備中

これら全てのツールキットは、Microsoft Windows で動作する C 関数ライブラリ群として提供されている。ここでは、データ会議の概念を理解するのに重要となる MCAT と GCCAT に絞って説明を行なう。

A.4.1 MCAT の概略

MCAT は T.122/123/125 に準拠しており、多種多用なネットワークにおけるマルチポイント通信のための各種のサービスを提供する。そのサービスとは、ドメインの生成/破壊、ドメインへのアプリケーションのアタッチ/デタッチ、チャネルの確保、データ通信、トークンの制御等であるが、これらの用語について、以下の図を用いて説明する。またその他、トポロジーなどの主要なトピックスについても述べておく。

ドメイン— ネットワークにおけるコミュニケーションのグループ（複数台のコンピュータによって構成される）を、MCS ではドメインと呼ぶ。1 台のコンピュータが複数のドメインに参加することも可能である。図中、A という名前のドメインが存在している。

MCS プロバイダー— MCS プロバイダは 1 台のコンピュータに一つだけ存在し、MCS のクライアントアプリケーションと、下層のトランスポートスタックの仲介の役目を果たす。

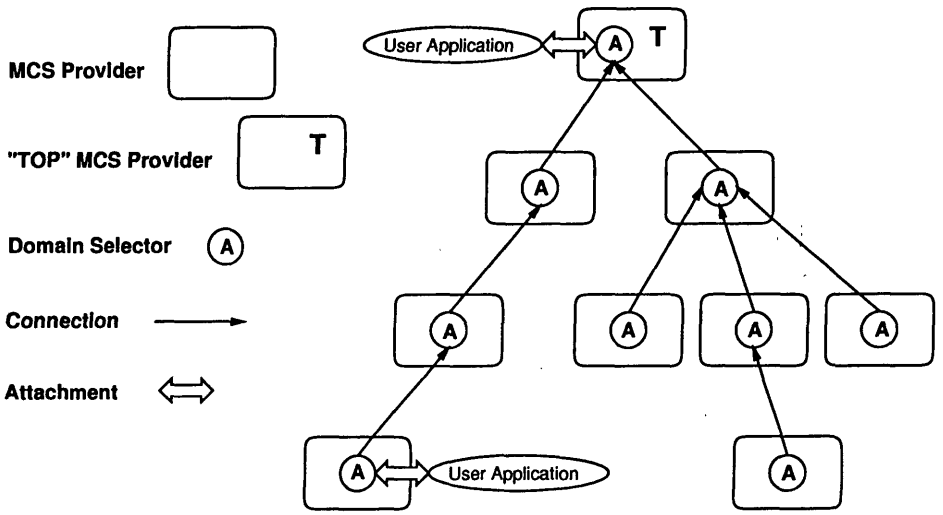

図 A.4: MCS のイメージ

トップ・プロバイダ— 一つのドメイン内に必ずユニークなトップ MCS プロバイダが存在する。トップ・プロバイダはドメイン内の全てのデータ配信、チャネルとトークンの割り当ての情報を管理している。

コネクション— MCS プロバイダ同士の物理的な接続をコネクションと呼ぶ。このコネクションには方向性があり、かならず一端が上位になる。このようにして階層的な接続によりドメインを構成するが、あるプロバイダの直接上位にあたるプロバイダは必ず一つだけである。自身より上位のプロバイダが存在しないプロバイダがドメイン内に唯一存在し、これがトップ・プロバイダとなる。

ドメインセレクタ— ドメインセレクタは、各 MCS プロバイダにローカルな概念であり、ドメイン全体として認識されるものではない。MCS プロバイダに接続するアプリケーションは、ドメインセレクタを通してドメインを参照することができる。

アタッチ/デタッチ— MCS クライアントのアプリケーションが、ドメインセレクタを通してドメインに接続することをアタッチと呼び、切り離すことをデタッチと呼ぶ。

チャネル— MCS における実際のデータ転送には、チャネルのメカニズムが利用されている。つまり、ドメイン内でユニークな ID によって識別されるチャネルを指定してデータの送受信を行なうのである。チャネルには、あらかじめ予約された ID を持ち制御用のデータ通信に使用されるものや、アプリケーションが要求して動的に割り当てられるものがある。MCS では、これらのチャネルを使用して以下のような通信方法を実現する。

- One-to-All(Broadcast)

- One-to-Many(Multi-cast)
- One-to-One(Point-to-Point)

これらの通信は同時に可能である。また、ドメイン内でチャネルIDはユニークであることが保証されており、逆にMCSが複数のドメインに参加している場合は同じチャネルIDであっても、それぞれ影響がないように扱われる。

トークン— トークンはドメイン内の資源や処理を管理／制御するために使用される、フラグのような役割を果たす。トークンも、やはりドメイン内にユニークなIDで識別され、Grab,Give,Inhibit,Please,Releaseといった要求を受けて、その状態を変化させていく。もちろん要求の競合が起こらないようMCSが制御する。

データ転送— データ転送には2種類の方法がある。Simple-Sendの場合は、トップ・プロバイダを経由するとは限らず、最も短いルートで転送される。また送信元は送ったデータが、きちんとチャネルにのったかを確認しない。一方、Uniformな送信では、データは必ずトップ・プロバイダを経由し、送信元は送り出したデータが指定のチャネルにのったことを確認する。さらにMCSは、データの転送に4段階の優先度を設定可能としている。この優先度はユーザアプリケーションで指定可能であるが、制御情報は高く、コマンドやテキストを中程、イメージやバイナリファイルの転送は低く設定すべきである。

ドメインのマージ— MCSでは、ドメインを安全にマージする方法も提供している。ドメイン内固有のIDなどがマージによって衝突しないよう調整が行なわれる。なおドメインのマージは、トップ・プロバイダが他のドメインにコネクションを張ることで実現する。

関数群の構成— ツールキットの関数群は、クライアント側が要求するファンクションと、MCSプロバイダからクライアント側に通知されるコールバックに分類できる。さらにファンクションには、アプリケーションがMCSのサービスを要求するRequestと、MCSの応答要求にクライアントが応えるResponseとがある。コールバックには、Requestに対してMCSから直接確認の通知を行なわれるConfirmと、他のMCSプロバイダからのRequestによって通知されるIndicationとがある。さらには、あるアプリケーションのRequestによって他のMCSプロバイダにアタッチしているアプリケーションにIndicationが通知され、それにResponseした結果、元のアプリケーションにConfirmが通知される場合もある。

トポロジー— MCATでは多様なネットワークトポロジーに対応している。図に示したようなcascadeや、単純なdaisy chain、star結合が可能であるが、トポロジーは通信の効率や性能に大きな影響を与えるため、注意が必要である。

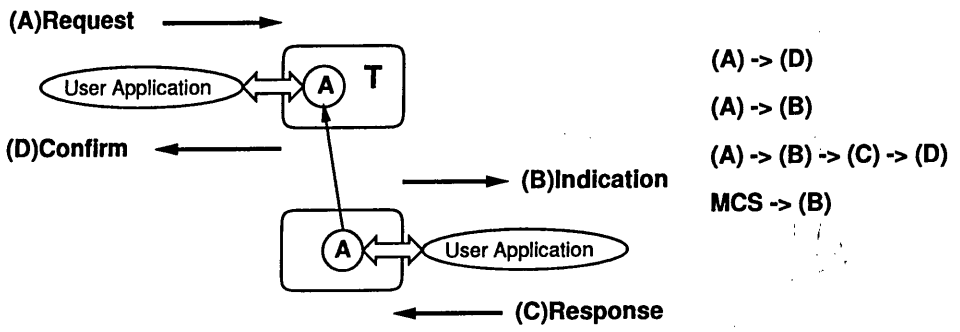

図 A.5: ファンクションとコールバックの詳細

A.4.2 GCCAT の概略

GCCAT では、T.124 に準拠して会議の管理、制御を行なうサービスを提供する。MCAT でドメインと称していた概念をカンファレンス (会議) と呼ぶなど、MCAT に比べ現実世界に近いイメージで理解できる。具体的には会議の生成／破壊、会議への参加／招集／登録／退出といった機能を司っている。ただし、実際のデータ通信を行なうわけではないため、アプリケーションはこのために MCAT の関数をコールする必要がある。GCCAT 固有の概念について以下のキーワードを説明していく。

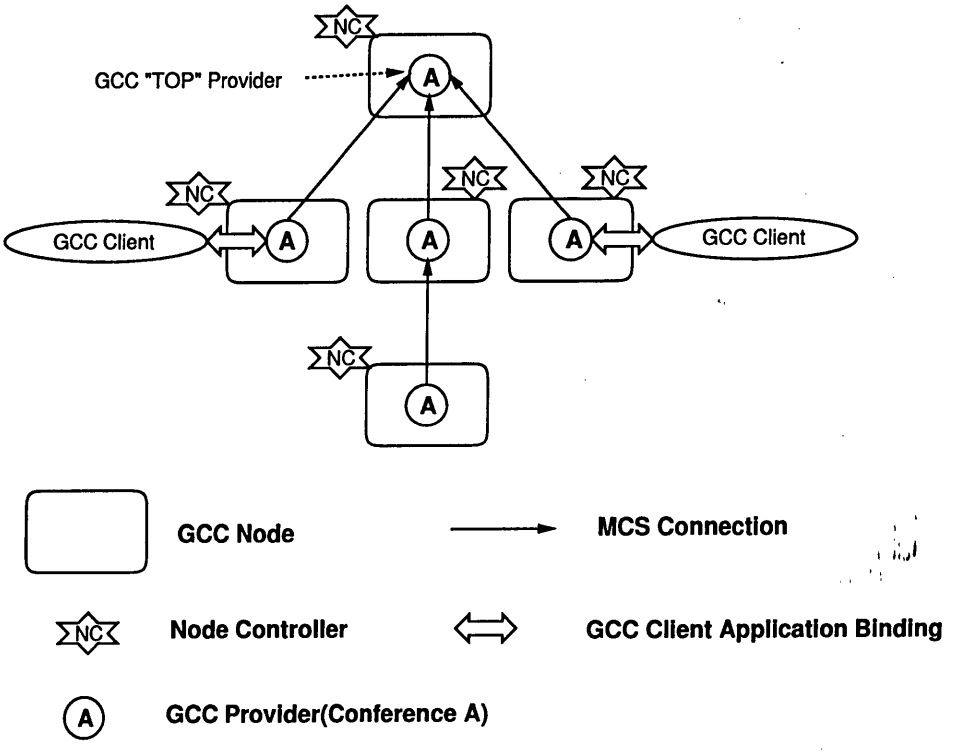

図 A.6: GCC のイメージ

GCC ノード — 会議に参加している装置を GCC ノードと呼び、MCU(Multipoint Con-

nection Unit) 装置や端末 (TERMINAL) がこれに該当する。MCU は複数のコネクションを持てるが、クライアントのアプリケーションを起動するのには使用されない。端末はコネクションを一つだけ持ち、ユーザがアプリケーションを起動して会議に参加できる。もちろん両方の機能を兼ね備えた装置も存在する。

GCC ノードコントローラ— ノードコントローラは T.120 シリーズで定義されているものではなく、GCCAT のクライアントの設計者が作成する。ノードコントローラは、会議中発生する様々なイベント（会議への参加や退出など）を受け取り、適切な対応をするように設計される。

GCC プロバイダ— GCC プロバイダはノードに一つだけ存在して、MCU や端末のクライアントアプリケーションにサービスを提供する。

トップ・GCC プロバイダ–トップ・プロバイダは会議に一つだけ存在して、会議の情報を一括管理している。GCC プロバイダは複数の会議に参加できるので、ある会議でトップ・プロバイダであったとしても、他の会議ではトップ・プロバイダである必要はない。

カンファレンス (会議) — 先にも述べたように、MCS でドメインと呼んでいた概念を、GCC では会議と呼ぶ。

会議 ID — 会議 ID は、各 GCC プロバイダにローカルな番号で、アプリケーションが参加している会議を識別するために使用する。

カンファレンス・ロースター (会議名簿) — GCC プロバイダの管理しているデータベースの一つで、会議に存在する全ノードについての情報を格納している。以下の内容が含まれている。

- 会議に接続しているノードの数
- 会議の名前
- 会議名簿の最終更新に関する情報
- ID や、タイプ (TERMINAL,MCU,or both)、使用しているプロトコルに従ったアドレス等の各ノードに関する情報

アプリケーション・ロースター— やはり GCC プロバイダによって管理されるデータベースで、会議内の各ノードにおけるアプリケーションの起動により内容が更新される。このデータベースには、以下のようなアプリケーション固有の情報が格納されている。

- 会議の ID
- アプリケーション毎にユニークなアプリケーション・キー
- アプリケーション間のセッションを識別するためのセッション ID

- アプリケーションが動作しているノードのID、ノードにおいてアプリケーションを識別するためのIDといった、ノード毎の情報のリスト

アプリケーション・レジストリ— アプリケーション・レジストリは、アプリケーションのMCSのチャネルやトークンの使用を管理する。アプリケーションはこの情報を参照して、データのやりとりを実行、制御できる。またこのレジストリは、アプリケーション・キーとセッションIDを管理するデータベースの役割もある。

アプリケーション・キー— GCCのクライアントアプリケーションは、それぞれ固有のアプリケーション・キーを持つ。これは国コード(2バイト)、ベンダーコード(2バイト)、ベンダー固有の情報域(サイズ指定可)で構成される。GCCATのクライアントアプリケーションを作成したベンダーは、ITUにデザインチェックリストを提出する。ITUはこれらのアプリケーションに固有のアプリケーション・キーを割り当て、一括管理している。

関数群の構成— ツールキットの関数群は、MCATと同じようにファンクションとコールバックに大別され、そのメカニズムもMCSとほぼ同様と考えて良い。

付録 B

Linux 環境の実現にあたって

Linux は Linus Torvals 氏が作成した Free な UNIX 互換 OS であるが、そのインストールパッケージや、アプリケーションとして以下に挙げる商用ソフトウェアを使用することにする。

B.1 Operating System および基本インストールパッケージ

- Caldera NETWORK DESKTOP v1.0
  - **PRICE** : $65 / 1 machine ()academic price)
  - **概要** : Linux Kernel , Base System , Caldera NETWORK DESKTOP , アプリケーション , 導入ガイド , すべてに source および binary が用意される。
  - **URL** : http://www.caldera.com
  - **EMAIL** : orders@caldera.com
  - **販売** : Caldera Inc.

元 NOVEL の会長が起こした会社 Caldera Inc. による RedHat LiNUX ベースの Linux パッケージ。主な特徴は RedHat ゆずりのわかりやすいインストーラと Caldera NETWORK DESKTOP と呼ばれる、デスクトップ環境である。
NOVEL が元ということで、Netware との接続もできるようになっている。

B.2 商用アプリケーションおよび拡張ソフトウェア

- Caldera's Network Desktop Application Bundle
  - **PRICE** : $379 / machine (x 11)
  - **概要**
    WordPerfect v6.0 for Linux (english only)
    表計算、など。
  - **URL** : http://www.caldera.com
  - **EMAIL** : orders@caldera.com

- 販売 : Caldera Inc.

NOVEL 系ということで、NOVEL が license を持つ Windows 用アプリケーションの移植と、Caldera 独自の商用アプリケーション集。ただし、すべてのアプリケーションが英語のみのためとくに必要ではないであろう。

- X INSIDE Accelarated X 1.2
  - **PRICE** : $99 / machine (x 11)
  - 概要 : X Window Server, 標準の XFree86 に比べ約 3 倍速い。
  - **URL** : http://www.xinside.com
  - **EMAIL** : info@xinside.com
  - 販売 : X INSIDE Inc.

  標準の XFree86 の X server と置き換えて使う。

- X INSIDE MWM + Development
  - **PRICE** : $149 / machine (x 11)
  - 概要 : Motif Deveopment Kit と MWM
  - **URL** : http://www.xinside.com
  - **EMAIL** : info@xinside.com
  - 販売 : X INSIDE Inc.

  MWM 開発環境。

- X INSIDE OpenGL
  - **PRICE** : $?
  - 概要 : OpenGL Tool Kit
  - **URL** : http://www.xinside.com
  - **EMAIL** : info@xinside.com
  - 販売 : X INSIDE Inc.
  - **NOTE** : Not Available now.

次に示すものは、上に挙げた X INSIDE 社のものと比較のため、導入をすることが考えられる。ただし、調べた限りでは、いまだに X11R5 のものしかないようである。

- Metro Link X 3.1
- Metro Link OpenGL
- Metro Link OSF/Motif 2.0
- Metro-Media

B.3 非商用アプリケーション

非商用 (Free) ソフトウェアで、別に手に入れなければならないものとしては、以下のものが挙げられる。

- Java Development Kit (JDK) 1.0 release
  - 作者 : SUN Micro Systems.
  - 移植者 : Java Linux Porting Project
  - **Files** : linux-jdk-1.0.try1.common.tar.gz
  - **Files** : linux-jdk-1.0.try4.static.tar.gz
  - **Files** : linux-jdk-1.0.try4.shared.tar.gz
  - 入手先
    http://java.blackdown.org/
    ftp://java.blackdown.org/
- teikade : The environment infomally known as dejava
  - 作者 : satoo@pfu.fujitsu.co.jp, shira@pfu.fujitsu.co.jp
  - **Version** : 1.6 for JDK 1.0
  - **Files** : teikade-1.61-src.tar.gz
  - 入手先 : do not ask.
- "vic" dirver for QuickCam
  - 作者 : Koji OKAMURA ¡oka@is.aist-nara.ac.jp¿
  - **Files** : vicbin-2.7a32-linux.tar.gz
  - **Files** : qcam-0.3.*
  - 現状 : 現在テスト中
- Linux driver for the AT&T GIS (formerly NCR) WaveLAN PCMCIA ethernet card
  - 作者 : Anthony Joseph ¡adj@pdos.lcs.mit.edu¿
  - **Files** : wavelan.tar.Z
  - 現状 : 現在αテスト中

これ以外の一般的な開発環境やほとんどすべてが Caldera Network Desktop (CND) にふくまれていますので、継ぎ足す必要は特にないと思われます。

X に関しては XFree, Acc-X, Metro-X どれでも動くと言う観点では Note においてはあまりかわらない。が、しかし、性能の点では Acc-X が一番動作実績が多く、速いと思われます。[1]

[1]現在、Hinote でテスト中の XFree3.1.2-SVGA の改造版でのみ Note の LCD で 16bpp が表示できます。(まだ、Hinote では動きません。) T610 は確実に動くようです。 ただし、1/3 位に遅くはなります。作者は筑波大学の伊藤希さん。

OpenGL に関しては現時点で X INSIDE のものはできていないようなので、MetroLink のものをつかうか、XINSIDE 以外のものを使う必要があるでしょう。

B.4 ハードウェア

- Toshiba Portage 610CT
- 3com 3c589C-TP
- Connectix QuickCam
- AT&T GIS (NCR) WaveLAN PCMCIA ethernet card
- Adaptec APA-1460 SlimSCSI PCMCIA card

付録 C

MS-Windows 系 OS 環境の実現にあたって

C.1 MS-Windows 系 OS の概要

現在使用されている MS-Windows 系 OS としては次の 4 種類がある。

- MS-Windows3.1
- MS-Windows95
- MS-WindowsNT3.51workstation
- MS-WindowsNT3.51server

これら 4 種類のうち、「MS-WindowsNT server」を除く、3 種類がクライアント OS として位置づけられている。また、そのクライアント OS のうち、「MS-WindowsNTworkstation」がビジネスユースなどの業務向け、また、「Windows3.1」および「Windows95」が個人ユーザ向けとして販売されている。ただし、「Windows3.1」に関しては標準ではネットワークに対応していないので、ネットワークを前提としたクライアント・サーバモデルの実現にはむいていない (Netware 等を導入する必要がある)。ただし、日本で発売されていない「Windows3.1workgroup」には標準でネットワーク機能 (MicroSoft LAN マネージャ) が内蔵されているため、米国などでは「Windows3.1workgroup」によるネットワーク化が進んでいる。

以下にそれぞれの OS の性格と主な用途について示す。

名称	用途	性格
Windows3.1	個人向けまたは スタンドアローン	不安定だが使用できるソフト が豊富
Windows95	個人向けまたは ネットワーククライアント	Windows3.1 に比べ安定性が 向上した
NTworkstation	業務向けまたは ネットワーククライアント	基幹業務に使用できるほどの 安定性と信頼性をもつ
NTserver	業務向けまたは ネットワークサーバ	複数の Windows 系 OS を乗せた コンピュータを管理できる

本研究室ではネットワークを介した協調作業等を行うため、検討対象とするOSをWindows95およびWindowsNTとした。

研究の性質上、プログラム開発を伴うため、そのために必要なOSの安定性等を考慮すると、制約がなければ、使用するOSとしてはWindowsNTがもっとも望ましいと考えられる。また、Windows系OSを導入した各マシンを統一的に管理するために、WinodwsNTserverを利用したドメイン管理を行うことが望ましい。なお、ここでいうドメインとは、WindowsNTserverによる管理単位であり、ドメイン中には複数のWindows系OSを搭載したコンピュータを収容できる。また、収容されたマシンのユーザアカウント等は、ドメインを管理するドメインサーバにより一元的に管理でき、アクセス権等が設定される。

しかしながら、ノートパソコンによる運用を考えた場合、残念ながら現在のところ、pcmciaカードの使用(NT3.51よりpcmciaカード対応となったが、NT起動時の認識しかできず95のようにplug&playに対応していない)や必要となる資源量(一般的にいってNTの方が95よりも必要とする資源が大きい)等に問題があるため、ノートパソコンおよびマシンパワーの低いディスクトップ機に限ってはWindows95の方が、WindowsNTよりも望ましいと考えられる。

以上のことを考慮した結果。本研究室で採用するMS-Windows系OSは以下のようになった。

- 各個人に配布するノートパソコンにおいては、使用OSをMS-Windows95とする。
- ディスクトップ機においてはMS-WindowsNT3.51workstationを採用する。
- 高速なディスクトップ機のうち一台をMS-WindowsNT3.51serverとし、ネットワーク上のMS-Windows系コンピュータの管理を行う。

なお、MS-WindowsNTserveは、リレーショナルデータベースのサーバ等にも利用することとする。

C.2 MS-WindwosNT

WindowsNTはマイクロソフトが開発した完全に新規開発のオペレーションシステムでありる。製品のターゲットとしては企業の基幹業務等の安定性・信頼性・高いセキュリティ等が問題となるような業務を対象としている。また、従来あるUNIXワークステーション並みまたはそれ以上の処理能力を発揮できることが必要であり、さらに、ネットワークサーバ等の機能を備えることが必要になる。そのため、WindowsNTはプロセスごとに独立した32ビットのフラットな仮想アドレス空間を持ち、スレッド単位のスケジューリングを実現している。

マイクロソフト社の既存システムとの互換性も確保されており、標準的なMS-DOSおよびWindows3.1アプリケーションが動作可能である。また、WindowsNT3.51よりWindows95アプリケーションへの対応が強化され、ほとんどのWindows95アプリケーションが動作可能となった(Windows95アプリケーションとしてマイクロソフトよりロゴマークの使用を許可されるためにはWindowsNT3.51以上で動作することが条件の一

つになっている)。さらに、国際標準である POSIX の API(Application Programming Interface) がサポートされているために、UNIX ベースのアプリケーションの移植も可能となっている。

ネットワーク機能としては、TCP/IP、Netware およびマイクロソフト社の NOS(Network Operating System) である LAN Manager の機能もシステムの一部として含まれている。また、WindowsNTserver においては Machintsh を完全にネットワークに組み込むことも可能であり、異なるネットワークプロトコルに対する柔軟性を持つ。

WindowsNT はそのデザイン・コンセプトにクライアント・サーバ・アーキテクチャを用いている。WindowsNT は、いくつかのオペレーティング・システムのパーソナリティ(環境) を実現しているが、このオペレーティングシステムのパーソナリティ自体もクライアント・サーバ・アーキテクチャに基づいて実装されており、それらのパーソナリティはサブシステムとして各パーソナリティに必要な要求を処理する。これにより、現在のユーザ・ニーズに答えるだけでなく、将来における機能拡張もオペレーティングシステムの基本部分を根本的に変更することなく対応できる。また、カーネル部はマイクロカーネルまたはカーネライズド・カーネルなどと呼ばれるコンパクトなモジュールに分離されているため、オペーレーティング・システム自身の移植が可能となっている。現在 INTEL 以外にも、MIPS-R4x00、AlphaAXP などに対応している。

C.3 Windows95

マイクロソフトが Windows3.1 の後継 OS として発表した。内部コード的には Windows3.1 時代の 16bit コードも数多く残っており、WindowsNT とは直接の関係はない。

特徴としては、Windows3.1 時代よりも使いやすくなったとされる 95 シェルと、周辺機器のプラグアンドプレイが挙げあられる。とくに、ノートパソコンにおいては、pcmcia の抜き差しに対応していることと、対応する pcmcia カード種類の多さにより、WindowsNT よりも適しているといえる。また、要求される資源量においても、概して WindowsNT より少量ですむため、非力なマシンでは Windows95 の利用価値はそれなりにある。

反面、Windows95 は Windows3.1 よりも安定したといっても、WindowsNT に及ばない面が数多くあり (一部画面描画 API においてリエントラントが可能でない 16bit コードが用いられているため、一部アプリケーションのハングアップ等により全体がハングアップする可能性がある)、また、セキュリティ関係が割愛されているため、業務に用いることを考えれば、非力な点が多い。マシンパワー等に余裕がある場合は、WindowsNT を導入した方がよいと考えられる。

C.4 Windows 系 OS の開発環境

本研究室では従来、メインの開発言語として c++を使用してきた。そこで今回、MS-Windows 系統の OS での開発を行うにあたって、Windows との整合性および総合的な開発能力を考慮し、c++としてマイクロソフト社の VisualC++を導入することにした。Unix 環境と同時にソフト開発を進めるため、サンマイクロシステム社の java を導入し、方針としては、メインの開発言語としては java を使用し、不足分を VisualC++で補う

こととする。また、さらに、マイクロソフトの VisualBasic を導入し、一部のソフト開発では、補助的に使用することとする。

付録 D

Unix と Windows 系開発環境の統合

Unix 環境との統合を図るため、次のソフトウエアを導入する。

- Xサーバ
- Telnetd,Rlogind
- Windows アプリケーションを X へ表示するソフトウェア
- SAMBA

Xサーバを導入することにより、Windows を搭載したコンピュータをX端末化することが可能となる。これにより、普段使用する Unix 環境を Windows95 を導入したノートパソコン上でも使用することが可能となる。また、Telnetd,Rlogind は、Windows 系 OS で標準で搭載されていない、仮想端末等のサーバ機能を実現すし、コマンドラインによる遠隔ログオンを可能とする。Windows アプリケーションを X へ表示するソフトウェアは、MS-WindowsNTserver の画面を X サーバ上に X プロトコルを用いて送信することができ、機能的には Windows のアプリケーションをXのクライアントと同等に扱えるようにする。この機能を導入すれば、通常使用している Unix 端末上から MS-Windows のアプリケーションを使用することが可能となる。

ファイル共用のために UNIX WS に SAMBA を導入し、Windows 系の File System を収容する。