人 工 知 能 77−11
（1991. 7. 18）

Michele: 分散環境に対応した協調作業支援システム構築ツール

中内 靖　伊藤 嘉邦　安西 祐一郎

慶應義塾大学 理工学部

Michele は非同期通信に基づいたグループウェアを構築するために開発されたマルチエージェントモデルに基づくグループウェア構築ツールである。本論文ではユーザ間での処理の流れを表すことのできる Michele のマルチエージェントモデル、エージェントの振舞いを記述するための言語 MDL、Michele の分散環境に対応した実装について説明する。

Michele: A Groupware Toolkit for Distributed Environment

Yasushi Nakauchi　Yoshikuni Itoh　Yuichiro Anzai

Faculty of Science and Technology, Keio University

3-14-1, Hiyoshi, Kohoku-ku, Yokohama 223, Japan

Michele is a groupware toolkit based on multiagent model. Michele is developed to construct groupware based on asynchronous communication. In this paper we describe that multiagent model of Michele can describe flow of processing in cooperative work, multiagent model description language MDL, and its implementation in the distributed computational environment.

1 はじめに

近年、オフィスやグループなど組織の協調作業を支援するための研究としてCSCW (Computer-Supported Cooperative Work) の研究が盛んに行なわれている。またそれにともない、協調作業を支援する計算機システム（グループウェア）の開発が活発に行なわれてきている [3] [5]。また近年、オフィスや研究機関ではネットワークに接続されたパーソナル・ワークステーションが普及するのにともない、電子メールに代表されるLANによる通信技術を個人が容易に利用できるようになってきている [4]。このような分散環境に対応したグループウェア例として、電子スケジューラ・システム、オフィスオートメーション・システム、グループの意思決定支援システムなどが挙げられる。これらのグループウェアが単一目的であるのに対して、また一方では様々なグループウェアを構築する枠組としてLIZA [2]、Object Lens [6] といったグループウェア構築ツールが提案されている。Object Lensはユーザ間での情報の整理を目的としており、オフィスの定型業務の処理のようにユーザ間の処理の流れを規定することはできない。また、LIZAはリアルタイム型のシステムであるため、オフィスの定型業務のような非同期通信に基づくグループウェアを扱うことはできない。

そこで我々は非同期通信に基づき、しかもワーカ間での処理の流れを扱うことのできるグループウェアを構築することのできる枠組として、グループウェア構築ツールMichele (Multi-agent Interface with Communication by Hectic ELEments) [9] の提案を行なう。非同期通信を分散環境において実現するためにはユーザや他の構成モジュールをうまく表現することのできるモデルが必要である。我々はこの目的を達成するためにマルチエージェントモデルを選択した。このモデルを利用することにより、分散環境において協調作業を行なうユーザ、協調作業に必要とされる知識、ユーザ間で受け渡される書類など、協調作業を構成する任意の要素をエージェントとしてモデル化することができる。Micheleにより提供される幅広い枠組を利用することにより、アプリケーションに依存しない様々なグループウェアの構築を可能にする。Micheleは我々が既に開発したグループウェア構築ツールMulti-Cooperator [8] を、分散環境においてより幅広く対応できるようにマシン依存性を軽減し、さらにシステムが学習することによりユーザを支援できるよう、知識獲得の機能を追加して改良したものである。

本論文では以下の第2節でMicheleが提供するマルチエージェントモデルについて説明し、第3節ではエージェントの振舞いを記述するために開発された言語MDL (Multi-agent Description Language) について説明する。そして、第4節では分散環境に対応した実装について説明する。そして、最後にMicheleの将来の展望について述べる。

2 協調作業のモデル

2.1 分散環境における協調作業

グループウェアでは人間の作業と計算機システムによる支援を融合することにより、より良い作業環境をユーザに提供できることが重要である。ここではまず、分散環境における非同期通信に基づくグループウェアのモデル化において、そのモデルがどのような要件を満たさなくてはならないかについて考察する。

ワークステーションが疎結合された分散環境では、各ユーザが直接アクセス可能な計算機資源の範囲には限りがある。そこで、システムの構成要素はモジュール化され、ユーザは必要なモジュールだけを参照すればシステムを利用できることが望まれる（独立性）。また、グループウェア構築ツールとしてシステムの拡張を容易にするためには、既に構成されたモジュールを他のモジュールのサブモジュールとして利用できることが望まれる（構築性）。このような特性を提供するために、我々はマルチエージェントモデルを利用することにする。マルチエージェントモデルでは一般に系はエージェントの集合とエージェント間のインタラクションから構成される。また、個々のエージェ

ントは自己完結であり、計算機システムの構成要素であるデータ・手続き・プロセスの概念を統合的に扱うことが可能である [1]。

マルチエージェントモデルを用いて系を表現した場合、ユーザ、ユーザ間でやりとりされる書類、LIZA [2] のようにユーザの代わりをしてくれる代理人など、系を構成するすべての要素をエージェントとして表現することが可能である。ところが、計算機システムで実現されるエージェントと実際のユーザとではその振舞いには相違があり、ユーザにとって利用しやすいシステムを構築するためには両者の相違を考慮して、両者を統合した環境を提供することが重要である。協調作業を構成する系をユーザ／計算機システムのそれぞれの立ち場からお互いに他方の振舞いを見た場合に、以下に示すような差分が見えてくる。

計算機システムを中心にしてユーザの振舞いを見た場合、ユーザは常にログインしているとは限らない。したがって、ユーザに対するタスクの依頼の方法は非同期通信[1]に基づいたものでなくてはならない（非同期性）。またタスクの実行では、同時に複数のタスクがあるユーザに割り当てられている場合、これらの処理は到着順に実行される保証はない。各ユーザはデッドラインや優先順位に基づきスケジューリングを行ないながら処理を行なっていると考えられる（スケジュール可能性）。さらに分散環境では、ユーザがアクセス可能な環境は局所的であり、ユーザに対してタスクを依頼する場合、ユーザの位置を意識しなくてはならない（局所性）。

一方、ユーザを中心にして計算機システムの振舞いを見た場合、通信に用いられるメッセージの表現として、システムとユーザで同じ表現を用いているのではユーザにとって理解するにはあまりにも煩雑である（表現可換性）。また、エージェントは以下に示すように状況に依存した振舞いをとることが望まれる。例えば、オフィスの定型業務処理において、ある書類（エージェント）がワーカ間を移動しながら処理されていく場合、現在どのワーカがその書類を処理しているのか、また、現在までに何人のワーカの処理を経てきたかにより実行される手続きは異なる。このようにエージェントが保持する複数の手続きのうち、ユーザがある時点で実行可能な手続きはタスクの進捗状況に依存しており、ある時点において実行可能な手続きに状況に応じた制限を設ける必要がある（状況依存性）。

非同期通信に基づく協調作業をモデル化する枠組は、様々な協調作業の形態を表現するのに十分な枠組を持つだけでなく、上述したようなユーザ／計算機システム間の差分を考慮したものであることが望まれる。

[1]これは戻り値をとらない手続き呼び出しのことを意味しており、グループウェアが非同期型であることとは異なる。

2.2 Michele のマルチエージェントモデル

2.1に述べたように、システムが独立性と構築性といった特性を満足するために我々はユーザも含めてシステムを構成する要素すべてをエージェントと呼ばれるモジュールとして表現する。ここでエージェントとは Actor モデル [1] での Actor のようなものであり、固有の内部状態を持ち、そのエージェントに関する手続きをメソッドとして持つものである。そして、エージェント間の通信はすべてメソッドコールとして実行される。また、これらのメソッドの呼び出しでは、状況に依存して選択的に実行されるようにガードを付加することができる。エージェント内のアクティビティは一つであり、あるメッセージに対するメソッドの手続きの実行中に他のメッセージが届いた場合、そのメッセージは現在実行中のメソッドの処理が終了するまで待たされる。また、エージェントの活動状態には ready・active・dead の3種類があり、エージェントは生成されると最初は ready 状態になる。そして、他のエージェントによりメソッドの手続きが呼び出されるとそのエージェントは active 状態になる。また、エージェントは dead 状態になることにより消滅される。

例えばエージェントは、「部長」や「秘書」といったワーカ、アンケート調査での「アンケート

用紙」やオフィスワークでの「請求書」といった書類、または「アンケート用紙を回収し集計を行なう」といったある機能単位を一つにまとめたものを表現することができる。書類に相当するエージェントでは、その所有者が現在その書類に関する仕事に責任があることを表している。したがって、書類の所有者を明確にし、処理の流れを明示的にする必要がある。既存のマルチエージェントモデル [1] ではユーザを処理の対象として考慮されておらず、ユーザ間での処理の流れを表現することはできない。そこで我々は、2.1に述べたユーザの環境の局所性を利用して、ユーザ間の処理の流れを明示的にモデル化できるようにするためにユーザ環境と呼ばれる概念を導入した。

ユーザ環境とは協調作業に参加するユーザに1つずつ存在するもので、そのユーザ環境内にあるエージェントは対応するユーザの所有物となり、他のユーザは直接アクセスすることはできない。また、エージェントは一度にはどれか一つのユーザ環境内にしか存在できず、エージェントがユーザ環境間を移動することによりその所有権が移譲される。

したがってエージェント間の通信には、同一ユーザ環境内の場合と異なるユーザ環境間との場合の二通りがあり、それぞれの通信は以下のように行なわれる。同じユーザ環境内に存在するエージェントに対して通信を行なう場合、エージェントはメソッドコールにより直接通信を行なうことができる。このときの呼び出しは遠隔手続き呼び出し (Remote Procedure Call) [11] と同様であり、呼び出した側のエージェントの処理は呼び出されたエージェントの処理が終了するまで待たされる。一方、異なるユーザ環境に存在するエージェントに対して通信を行なう場合、通信元のエージェントが通信先のエージェントの存在するユーザ環境に移動することにより通信が行なわれる。このとき、通信元のエージェントの処理のために新たにプロセスが生成される。すなわち、通信元のエージェントを呼び出していたエージェントが存在した場合、呼び出していたエージェントの処理と通信元のエージェントの処理は並行実行される。また

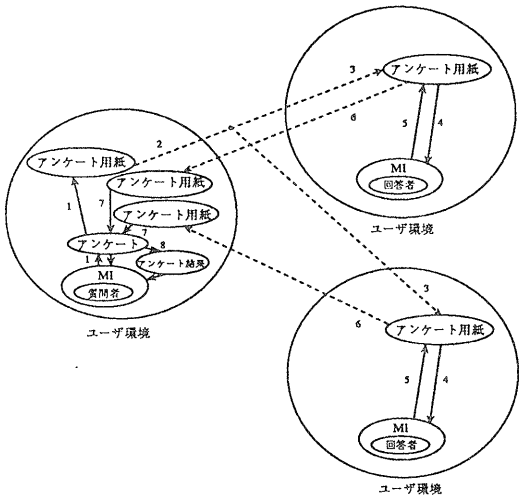

図 1: アンケート調査のマルチエージェントモデル

たこのとき、通信元のエージェントの所有権はユーザ環境間を移動することにより移動先のユーザに移譲される。ユーザ環境間の通信はエージェントがパケットとして渡される非同期通信となっており、2.1で述べた非同期性を考慮したものになっている。

システムで用意されたエージェント（以下システム・エージェントと呼ぶ）とユーザ間の通信では 2.1で述べたように表現可換性の問題がある。そこで我々はシステム・エージェントとユーザ間でのメッセージの表現の差異を埋めるために MI (Message Interpretor) を導入する。MI はウィンドシステムによって実現される。MI によりユーザとシステム・エージェントの間でのメソッドコールは、ユーザとシステム・エージェントの双方から見て理解し易い表現に翻訳される。ここで、ユーザに MI をかぶせたものを特にユーザ・エージェントと呼び、システム・エージェントと区別する。また MI では状況依存性を考慮しており、システム・エージェントの持つ手続き群のうち、現在実行可能な手続きだけが選択的にユーザに提示されるようにフィルタが掛けられる。図 1にアンケート調査のマルチエージェントモデルを示す。

1. 質問者がアンケート用紙を作成する。
2. 質問者がアンケート用紙を回答者に送付する。
3. 回答者がアンケート用紙を受理する。
4. 回答者がアンケートの内容を評価する。
5. 回答者がアンケートの回答を作成する。
6. 回答者がアンケートの回答を質問者に送付する。
7. 質問者がアンケートの回答を受理する。
8. 質問者がアンケートの回答の統計値を計算 (評価) する。

図 2: アンケート調査の手続き

3 エージェント記述言語

Michele のマルチエージェントモデルによりモデル化された協調作業を計算機に支援させるためには、マルチエージェントモデルにより概念レベルで表現されたエージェントの振舞いを計算機に理解できる表現に素直に変換できる言語が必要である。本節で Michele のマルチエージェントモデルを記述するために開発されたエージェント記述言語 MDL (Multi-agent Description Language) について説明する。ここではまず MDL の言語仕様を説明するために、非同期通信に基づく協調作業の簡単な例としてアンケート調査の手続きを紹介する。そして、個々の手続きが MDL を用いてどのように記述されるかについて説明する。

アンケート調査は一般に図 2 に示すような手続きにしたがって実行される。これらの手続きは書類に対する手続きを中心に考えると、書類に対する作成・送付・受理・評価の 4 種類の手続きから構成されていると考えられる。また、7 のアンケートの回答の受理では、回答者に配布した複数の書類を待ち合わせる機能が必要である。

MDL のプログラムはエージェントの宣言の並びから記述される。エージェントの宣言は図 3 に示すように、エージェントの内部状態を表わすフィールドの並びと他のエージェントから呼び出すことのできるメソッドの並びから構成される。フィールドの記述では初期値をとることもできる。そしてメソッドの記述では、引数だけでなく、キーワード when と共に状況に基づく条件式を記述することもできる。また、メソッドの手続きの記述のなかで最後の評価式がそのメソッドの戻り値となる。ここで、“エージェントの説明” と “メソッドの説明” はそれぞれエージェント名とメソッド名の別名であり、図 4 に示すようにユーザが MI を通してエージェントを見た時のそれぞれエージェントの名前と実行可能な手続きの名前として現れる。このようにエージェント記述では、エージェントの振舞いについての記述だけでなく MI の仕様についても記述される。

```
(DEFAGENT ≪エージェント名≫ "エージェントの説明"
    (FIELD ≪フィールド名:≫ [≪初期値≫])
           ⋮
    (METHOD ≪メソッド名≫ (≪引数の並び≫)
            ≪when (条件式)≫ "メソッドの説明"
              ≪メソッドの手続≫)
           ⋮
)
```

図 3: エージェントの宣言

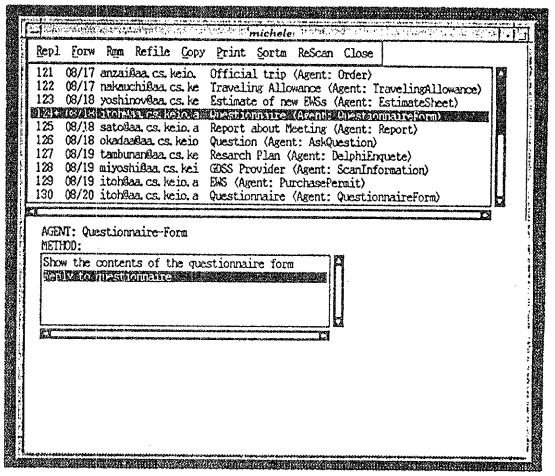

図 4: MI を通して見たシステム・エージェント

エージェントのフィールドにはエージェントの状態を表すためにフィールド “activity:” がデフォールトで付けられる。このフィールドの値を “dead” とすることによりエージェントを消滅させることができる[2]。メソッドコールは一般に

[2]正確には、あるメソッドの手続き実行終了時に “activity:” フィールドの値が “dead” となっているとそのエージェントは消滅させられる。

(≪メソッド名≫ ≪IP[3]≫ ≪引数の並び≫)
と記述される。2.2にも述べたように並行動作するエージェントの間の通信の同期は遠隔手続き呼び出しと同様で、起動したエージェントから起動されたエージェントに制御権が移り、その実行終了を伝える確認情報（戻り値）が返されるまで起動側のエージェントは待つことになる。

アンケート調査の例で示したような書類に関する基本的な手続きを実現するために、各エージェントには基本的なメソッドがシステムであらかじめ用意されている。書類の作成は用紙の用意と書き込みによって行なわれる。これらの操作はそれぞれ、エージェントのインスタンスの生成と、インスタンスのフィールドに値をセットによって実行される。

エージェントのインスタンス生成は関数 New を用いて

(New ≪エージェント名≫)

と記述され、戻り値として生成されたインスタンス・ポインタが返される。また、C++ [12] と同様にエージェント名と同じ名前のメソッドは構築子と呼び、インスタンス生成と同時に呼び出される。さらに、Michele ではこの構築子を持つエージェントだけがユーザから直接インスタンスを生成できるエージェントと定義し、これを基本エージェントと呼ぶ。したがって、基本エージェントだけがユーザが最初に起動できるサービスとして MI に登録される。

そして、フィールドの値の操作はフィールドの値を読み出すメソッド Get とフィールドに値をセットするメソッド Put によって実行される。

メソッドコール Get と Put はそれぞれ

(Put ≪IP≫ ≪フィールド名:≫ ≪値≫)
(Get ≪IP≫ ≪フィールド名:≫)

と記述される。

エージェント間の通信は、同じユーザ環境内では上に示したメソッドコールとして実行されるが、ユーザ環境間の通信の場合は以下に示す関数 Migrate によって実行される。

(Migrate ≪ユーザ環境名≫ ≪手続きの並び≫)

[3]インスタンス・ポインタ (Instance Pointer)

エージェントは関数 Migrate を実行することにより、第一引数で指定されたユーザ環境[4]に移動する。そして、第二引数以降に記述された手続きが移動先のユーザ環境内で実行される。関数 Migrate の呼び出しでは新たにプロセスが生成され、そのメソッドを呼び出したエージェントと関数 Migrate の処理は並行実行される。したがって、関数 Migrate は戻り値をとることはできない。

関数 Migrate により書類の送付と受理が記述される。書類の送付は対象となる書類が関数 Migrate を実行することにより実行される。また、書類の受理は到着した書類が受理される相手に対してメソッドコールすることにより実行され、メソッドコールされたエージェントは書類の受理にともなう手続きを実行する。そして、複数の書類の選択的な受理はエージェントの内部状態および届いた書類の内容にしたがって処理を分岐することによって記述される。また、複雑な手続きも記述できるように、メソッドの手続きの記述には Lisp の豊富な関数群を利用することができる。そして、書類の評価はユーザが定義するメソッドに手続きの並びとして記述される。

2.2で述べたように、ユーザ・エージェントはユーザに MI をかぶせたものである。したがって、ユーザ・エージェントとシステム・エージェント間のメソッドコールでは、システム・エージェントからは MI に対するメソッドコールとして、また、ユーザからは MI により示されるメニューによる手続きの選択として実行される。システム・エージェントとユーザ・エージェント間のインタラクションは主にユーザに対する情報の提示と問い合わせの２通りである。そこで、MI には質問を受け付け、これに答えるための２種類のメソッド Show と Ask を用意した。メソッド Show はユーザへの情報の提示を、そしてメソッド Ask はエディタまたはメニューを用いてユーザに情報の問い合わせを行う。メソッド Show と Ask はそれぞれ、

(Show ≪IP≫ “フォーマット文字列” ≪引数の並び≫)
(Ask ≪IP≫ [:editor | :menu “メニューの並び”])

[4]ユーザ環境名はネットワーク内でのユーザの識別名、すなわちメールアドレスが指定される。

と記述される。

4 処理系

4.1 電子メールの利用

Michele の実装ではエージェントの内部状態を保持する方法とエージェント間の通信を分散環境において実現する方法が重要である。以下に、それぞれの実現方法について説明する。

エージェントの内部状態は複数のフィールドとその値から構成されており、アンケート調査などの協調作業では一連の仕事が終了するまで存在し続けなくてはならない。そこで、Michele ではエージェントの内部状態をフィールドとその値から構成される半構造化メッセージ [6] [7] として表現する。半構造化メッセージはメイルのヘッダの構造と同じ形をしており、メイルシステムによるメッセージの転送を利用し易い。

エージェント間の通信には同一ユーザ環境内の場合と異なるユーザ環境間の場合の2通りがある。同一ユーザ環境内でのエージェント間の通信はメソッドコールによる手続きの呼び出しであり、これはメソッドに対応した関数を呼び出すことにより実現する。また、同一ユーザ環境内に存在するエージェントは実際にはあるユーザのホーム・ディレクトリ内ファイルとして存在させることにより、ワークステーション間の通信機構を必要とはしない。

一方、異なるユーザ環境間での通信は通信先のエージェントが相手先のユーザ環境に移動し、移動先のユーザ環境においてユーザ環境内のメソッドコールを実行する。エージェントの移動はエージェントの内部状態を保持した半構造化メッセージをメールシステムを利用して転送することにより実現する。したがって、エージェント間の通信はメールの届く環境であれば実行可能であり、Michele は TCP/IP で通信できる環境よりもさらに広い環境において利用することができる。

4.2 ユーザ環境間の通信機構

異なるユーザ環境間の通信では他の環境からのメッセージが到着すると、関数 Migrate の第二引数以降に記述された手続きを起動し実行する必要がある。この働きをするのが ICS (Inter-user-environment Communication Support system) である。Michele ではエージェントの移動にメールシステムを利用するため、到着するメッセージが普通のメールであるかエージェントであるかを判別しなくてはならない。ICS はユーザ環境に到着するメッセージを監視しており、それがエージェントの内部状態を持つものであった場合、関数 Migrate に第二引数があるかどうかを判断して対応する手続きを実行する。また、普通のメールが到着した場合には、ICS は Information Lens [7] と同様にユーザによって記述されたルールにしたがい、フォルダに分類して格納する。

4.3 メソッドの手続きの実行

Michele のマルチエージェントモデルではエージェントが複数のユーザ環境間を移動するため、一つのエージェントの手続きが異なるユーザ環境で実行されることがある。分散環境では異機種の計算機が存在するため、エージェントのメソッドの手続きはどの計算機上でも実行可能な形になっている必要がある。Multi-Cooperator [8] ではエージェントの手続きは実行前にあらかじめコンパイルして計算機の機種毎に実行オブジェクトを作成しておき、これを各ユーザ環境から参照できるようにしていた。そのため、分散環境において計算機の機種毎に異なるオブジェクト管理することは非常に繁雑であった。そこで Michele ではエージェントの手続きを Lisp のコードにコンパイルしておき、Lisp の処理系の上で実行できるようにした。また、エージェントの実行コードの管理についてはエージェントの内部状態を保持する半構造化メッセージに附属させ、エージェントの移動と同時に実行コードも移動されるようにした。

エージェントのメソッドの呼び出しには3通りの場合がある。それらはユーザが MI を介して呼

び出す場合、ICS がエージェントの到着を検知して呼び出す場合、エージェントがメソッドの手続きの中で他のエージェントを呼び出す場合である。上述したメソッドの手続き実行の手法を用いることにより、3通りいずれの場合においても Lisp の処理系が実装されている計算機上であれば計算機の機種に依存することなくメソッドの手続きを実行できるようになった。また、分散環境において実行オブジェクトの管理が容易になった。現在、Michele の処理系は複数機種のワークステーション (SONY NEWS、SUN-3、OMRON LUNA) 上で稼働中である。

5 おわりに

本論文では非同期通信に基づくグループウェア構築ツール Michele を提案した。Michele のマルチエージェントモデルでは、分散環境において協調作業を行なうユーザ、協調作業に必要とされる知識、ユーザ間で受け渡される書類など、協調作業を構成する任意の要素をエージェントとしてモデル化することができる。ユーザ環境間をエージェントが移動することにより、ユーザ間での処理の流れを表すことができる。本マルチエージェントモデルでは2.1で述べたユーザと計算機システムの差分について考慮したわけであるが、スケジュール可能性については MI を介してスケジュール可能なタスクの一覧を表示するに留まっており、今後の課題として残っている。また我々はさらに Michele を発展させるべく、扱えるデータのマルチメディア化・学習機構の導入 [10]・ユーザに利用しやすいプログラミング環境の開発を試みている。

参考文献

[1] Agha, G. : *ACTORS: A Model of Concurrent Computation in Distributed Systems*, MIT Press, 1986.

[2] Gibbs, S. J. : Liza: An Extensible Groupware Toolkit, *CHI '89*, pp.29-35, 1989.

[3] 石井 裕: グループウェア技術の研究動向, 情報処理学会, Vol.30, No.12, pp.1502-1509, 1989.

[4] 板倉 節男: LAN とワークステーションを用いた電子メール通信, 情報処理学会, Vol.28, No.8, pp.1030-1037, 1987.

[5] Kraemer, K. L. and King, J. L. : Computer-Based Systems for Cooperative Work and Group Decision Making, *Computer Survey*, Vol.20, No.2, pp.115-146, 1988.

[6] Lai, K. Y. and Malone, T. W. : Object Lens: A "Spreadsheet" for Cooperative Work, *CSCW '88*, pp.115-124, 1988.

[7] Malone, T. W., Grant, K. R., Lai, K. Y., Rao, R. and Rosenblitt, D. : Semi-Structured Messages are Surprisingly Useful for Computer- Supported Coodination, *CSCW '86*, pp.102-114, 1986.

[8] Nakauchi, Y., Itoh, Y., Sato, M. and Anzai, Y. : Modeling and Implementation of Multiagent Interface System for Computer-Supported Cooperative Work, *Ergonomics*, 1991. (in press)

[9] Nakauchi, Y., Itoh, Y., Sato, M. and Anzai, Y. : Michele: A Multi-agent Interface Architecture for Distributed Open Environments, *Tools'91*, pp.61-69, 1991.

[10] Nakauchi, Y., Okada, T. and Anzai, Y. : Groupware that Learns, *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pp.688-691, 1991.

[11] Nelson, B. J. : Remote Procedure Call, Ph.D.dessertation, Carnegie-Mellon University, 1981.

[12] Stroustrup, B. : *The C++ Programming Language*, Addison-Wesley, 1986.