ヒューマンインタフェース 35−20
（1991. 3. 8）

PilotMail:グループ内会話モデルに基づいたメイルシステム

市村 哲　平岩 真一　松下 温

慶應義塾大学理工学部

我々の提唱する「チームウェア」は主に蓄積型分散環境でなされる協同作業を支援対象とするコンピュータシステムであるが、特にここでは、そのような環境下でのグループ内コミュニケーションを促進させることを目的とした提案を行う。

本稿では、オブジェクト指向モデルをグループ内通信に適用し、チームのメンバー間で行うメイルのやり取りをオブジェクト間のメッセージパッシングと捉えることにより、グループの多様な会話モデルに動的に対応できるメイルシステムPilotMailを提案する。このメイルシステムの主要目的は人と人がコミュニケーションする際にしばしば発生する「誤解」を抑え、相互コミュニケーションを親密にし協同作業を促進させることにある。

PilotMail: A Group Conversation Based Mail System

Satoshi Ichimura　Shin-ichi Hiraiwa　Yutaka Matsushita

Faculty of Science and Technology, Keio University

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

In this manuscript, we propose the mail system named "PilotMail" which supports various group conversation models through applying the Object-Oriented model to group communications and regarding exchanges of electronic mails among members of a team as message passings among objects. The principal aim of this mail system is to prevent occurrence of a "misunderstanding" which is frequently produced through group communications and to realize the communication environments in which users can communicate with one another as if they were in the same place. For this purpose, we followed and examined the technique which prevents occurrence of a misunderstanding and also applied the forwarding-mechanism of an information processing procedure and also applied the structured-mail technologies to exsisting electronic mail systems.

1 はじめに

従来、コンピュータによる作業支援の形態は個人作業環境の支援にのみ重点がおかれていた。しかし、複数人による協同作業を支援することの必要性が高まるにつれ、グループウェアに対する関心が高まり、研究が盛んに行われるようになりつつある[1]。我々の提唱する「チームウェア」は主に蓄積型分散環境でなされる協同作業を支援対象とするコンピュータシステムであるが[2]、特にここでは、そのような環境下でのグループ内コミュニケーションを促進させることを目的とした提案を行う。

本稿では、オブジェクト指向モデルをグループ内通信に適用し、チームのメンバー間で行うメイルのやり取りをオブジェクト間のメッセージパッシングと捉えることにより、グループの多様な会話モデルに動的に対応できるメイルシステム PilotMail を提案する。このメイルシステムの主要目的は人と人がコミュニケーションする際にしばしば発生する「誤解」を抑え、相互コミュニケーションを親密にし協同作業を促進させることにある[3]。

2 チーム内コミュニケーション

実際の社会組織ではグループによる協調作業が活動の基礎となっており、その中では人と人あるいは集団と集団の間のコミュニケーションが無数に行われる。我々の提唱する「チームウェア」では、グループ内コミュニケーションのうち特に蓄積型分散環境でなされるものに対して支援をしている（図1参照）[4][5]。我々は「チーム」を、「各自が達成目標を確実に把握しつつ協調して実作業を進めるエキスパートの集団」と定義しているが、グループワークでは、当初予定していたスケジュール通りには作業が進行せずそのための連絡を互いにとりあったり、個人が作業をしている際に発生した問題点を打ち合わせによって解決しなければならない状況に遭遇する。すなわち、他の人と進行を合わせたり、作業を依頼したり、必要な情報を交換しながらチームの作業は進むのである。

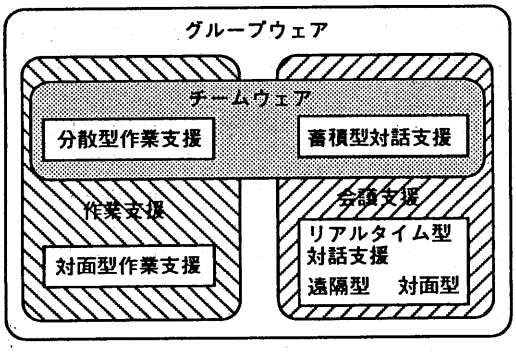

図 1: チームウェアの位置づけ

グループウェアを従来のシステムと区別する重要な用件の一つとして、Winograd は人と人との協調構造に基づいて設計されたシステムであることを挙げているが、これはグループウェアが人間のインタラクション構造に基づいて設計されるべきであることを指している[6]。すなわち、人間の行う会話を単なるメッセージの集積ではなく、構造を持った情報処理機構として捉える必要があり、その中では、会話の行方（将来とり得る遷移状態や結果）が意識されなければならないことを示している。この必要性ゆえ、最近、会話モデルを基盤としたグループウェアが注目されるべき研究対象と成りつつある[7]。

電子メイルは、紙を用いず、しかもワープロソフトを使って作成した原稿や書類あるいはコンピュータプログラム等を相手に送るといった使い方ができることから、特にコンピュータエンジニアの間で重宝されるようになった。ところが、電子メイルは今や重要な通信手段に成長したにもかかわらず、人間のインタラクション構造に基づいて作成されておらず、グループの会話モデルを考慮していない。このために、コミュニケーションのためのツールでありながら、パーソナルユースのためだけに設計されていると言わざるを得ない。

3 誤解を抑えるメイルシステム

コミュニケーションの際に発生する「誤解」を抑えるためには送信者の意図をできるだけ正確に受信者に伝える必要があった。このために我々は、誤解を予防する手法について検討し、それに基づいて、受信者への情報処理手続きの転送機能とメイルの構造化を電子メイルに施した。

3.1 誤解のメカニズム

人と人のコミュニケーションには、程度の差はあるが、思い違いや誤解がつきものである。しかし、誤解があったことを知るのはその発生時からかなり時間が経過してからになるため、ひとたび重大な誤解が生じるとそれを解消するためコストは膨大なものになる。それゆえ、発生からそれに気付く間に行った作業が無駄になるばかりでなく、他の人と進行を合わせられなくなり、その結果チームにとって多大な損失がもたらされる。すなわち、誤解が発生した後に回復しようとしても手遅れであり、誤解を予防することこそが必要となるのである。

誤解の発生を招く原因としては種々考えられるが、我々がPilotMailを考案するにあたって着目したいくつかの要素を以下に記す。

受信者の行動が予測できない: 従来の非同期分散型のコミュニケーションの場合、送信者はメッセージの受取人の行動を予測することができない。そのため、確実に受信者がメッセージを受け取ったのかさえもわからず、信頼して作業を依頼することができない場合が多い。そこで、メッセージを受け取った受信者がどのように振る舞えばよいのかを送信者がナビゲートしてやることが相互の誤解を抑える手法となり得る。

メッセージの伝達ミス: 日常会話においても、話し手の言葉を聞きもらしたり、あるいは話し手が要点を伝え忘れることが多分にある。リアルタイムのコミュニケーションではお互いにその場で指摘しあってその脱落を補足してゆくが、そのような補足が難しい非同期コミュニケーションでは送信者は要点の書き忘れがないようにしなければならないし、受信者は文面から要点を的確に読みとらなければならない。そこで、情報の要点が欠如することなく送信者から受信者に伝えられる仕組みが必要となる。

情報整理の難しさ: 正確に伝達されたメッセージであっても、そのメッセージを整理するときに失敗してしまうと「二次的な誤解」を発生させるおそれがある。ここでいう二次的な誤解とは、メッセージ受諾時には正確に伝えられたメッセージであっても、後からそのメッセージにアクセスしたときには必ずしも当初のメッセージに対する正確な認識と同一に認識するとは限らないということを指しており、広くは、メッセージを忘れ去ったり紛失したりしてメッセージに対する十分な認識ができなくなるといった状態をも含む。たとえば、重要な会議が数日後に開催されるという通知がきても、それをスケジュールに組み込むのを忘れたならば、また、組み込んだとしても通知の内容を把握できなければその通知は無駄におわってしまうのである。すなわち、伝達される情報の形式は整理され易いものであるのが好ましいといえる。

3.2 受信者への情報処理手続きの転送

メッセージを受け取った受信者がどのように振る舞えばよいのかを送信者がナビゲートするための手法として、情報処理手続きを転送することを提案する。これにより送信者の意図に従って受信者が忠実に振る舞うことが保証される。すなわち、送信者が会話の行方や受信者の行動を予測できるようになり、信頼して処理を受信者に任せられるようになり、誤解の予防ができる。

しかしながら、他人からの要求を鵜呑みにして履行することは煩わしいだけでなく危険をもはらんでおり、他人から依頼された要求であっても自分なりに解釈して実行できる機構が必要となる。そこで、メッセージに対する最低限の共通の認識が存在しさえすれば、オブジェクト内部の具体的な処理を詳細に知る必要がないというオブジェクト指向知識表現をメイルシステムに採用した。このようにオブジェクト指向知識表現の計算モデルはチームの作業の進め方に非常に近いものを持っており、複数の制御権を持ったもの同士の関係が自然な形で表現できるという特長を持っている。つまり、オブジェクトという独立して実行される実体をそのままチームにおける個々のメンバーにあてはめ、メッセージパッシングをメンバー間のインタラクションにあてはめて考えることができるのである。

3.3 メイルの構造化

Information Lens (Object Lens) システムのようにメイルを構造化し、オブジェクト指向デー

タモデルの採用により、メイルの使用目的に応じたクラス階層化を施して管理した[8][9]。クラスが階層化していることで、クラスは親クラスの機能・属性を受け継ぐことができ、たとえば、新たにメイルのクラスを定義するときには親クラスとの差分のみの機能・属性を付加するだけでよくなる。また、メイルが情報の内容に適合した構造（フィールド群）を持っているため、受信者はそのメイルの内容を正確に把握し易く、送信者にとっても要点の書き忘れが無くなる。

さらに、チームで共通の構造を定義しておけば共通の情報形式でメイルが伝わるため、その後の情報の整理および返答がしやすくなる[10]。Lens システムでは、新しいメイルの到着、pre-specified time 、ユーザによる選択などによって引き起こされる半自律的なエージェントをシステム内にルールベースとして構築することができる。エージェントが引き起こされると、もしメイルの特定のフィールドがルールのある条件を満足したならばあらかじめ指定したアクションが動作する。メイルの情報フィルタリングなどにこの機能が応用されている。

しかしながら、メイルを仕事の種類に応じて構造化してしまうと、例外的な仕事や処理を他人に依頼する場合、結局プレーンなテキストとしてその依頼を記述しなくてはならない。そこで我々は、任意のメイルに任意のフィールドを追加して、そのフィールドにデータ処理手続きを記述できるようにし、非定型業務および複雑な処理の依頼への対応性を高めた。

4 PilotMail システム

4.1 PilotMail システムの使用例

つぎに、簡単な具体的使用例を挙げて我々のメイルシステムを概説する。ここに挙げる例は、ユーザ Ichimura がグループミーティングを招集しようとしてそのミーティングの参加者に向けて通知を行う例である（斜体字は新たに入力した文字）。ただしこの例では、PilotMail システムと個人のスケジュールマネージャーシステムの間にインタフェースが存在することを仮定している。

クラス階層化されて管理されている構造化メイルの中から MeetingAnouncement 構造を選択すると図 2のデフォルトテンプレートが得られる。図の左部はメイルのデフォルトフィールド名であり、それぞれの右部に、対応するデフォルトプロシジャが現れる。図中の DATE: TIME: フィールド、PLACE:フィールドはそれぞれミーティングの開催日時、開催場所が記入されるべきフィールドであり、また、PARTICIPANTS: TOPIC:フィールドはそれぞれミーティングへの参加者、ミーティングの議題が記されるフィールドである。

図でわかるようにプロシジャの記述言語は Smalltalk-80 を基礎としているが、図中の AgentSet は Agent（個人 ID やグループ ID）のみが入る Set である。また、input は新たなデータ入力を促すためのクラスメソッドであり、例えば Date クラスのエディタがカレンダーのようなものであっても良いし、Place クラスの入力エディタが地図・人員配置図のようなものであってもよい。それは個人の好みによって選択されるべきものであり、デフォルトプロシジャ自体をも個人の好みで設定できる。

もしデフォルトのテンプレートを変更したい場合は、図 3のように自由に変更できる。図では、ミーティングの開催日を 3 日後に設定し、更に、NOTIFY:フィールドを追加してミーティングの 1 時間前に受信者のスケジュールマネージャーのノーティフィケーション機能が起動されるように設定している。図のように Agent および AgentSet クラスの doIt:メソッドが受信者へのデータ処理手続きの転送をするための命令であるが、引数のブロック内のみが受信者へ転送される（図5参照）。ただしここで、Schedule クラスのクラスメソッドである notify: date: time: は、チームの間で「スケジュールマネージャーのノーティフィケーション機能が指定された日時に起動される」という共通の認識が存在していることが大前提である。

手続きの変更が終了した後、手続きの実行を行うが、上のフィールドから順に（図中では DATE:、TIME:、PLACE: ... の順に）評価される（ただし doIt:メソッドのブロック内は評価されない）（図4参照）。図の斜体文字部分はユーザがエディタを用いて入力したものであるが、それ以外の箇所はシステムが自動的に生成したものである。そして実行結果を見てそれが正しければメイルを送信する。

TO:	AgentSet input.
SUBJECT:	String input.
DATE:	Date input.
TIME:	Time input.
PLACE:	Place input.
PARTICIPANTS:	AgentSet input.
TOPIC:	String input.

図 2: デフォルトテンプレート

DATE:	*Date today addDays: 3.*
TIME:	Time input.
PLACE:	Place input.
PARTICIPANTS:	AgentSet input.
TOPIC:	String input.
TO:	*PARTICIPANTS.*
SUBJECT:	*TOPIC.*
NOTIFY:	*PARTICIPANTS doIt:*
	[Schedule
	notify: SUBJECT
	date: DATE
	time: (TIME subtractHours: 1)].

図 3: 訂正後のテンプレート

DATE:	*March 15, 1991*
TIME:	*13:00:00*
PLACE:	*Matsushita lab*
PARTICIPANTS:	*Matsuura, Tsukada, Hiraiwa*
TOPIC:	*TCW meeting on new mail*
TO:	%(Matsuura Tsukada Hiraiwa)
SUBJECT:	'TCW meeting on new mail'
NOTIFY:	%(Matsuura Tsukada Hiraiwa) doIt:
	[Schedule
	notify: SUBJECT
	date: DATE
	time: (TIME subtractHours: 1)].

図 4: データ入力後のテンプレート

To: Matsuura, Tsukada, Hiraiwa
Subject: TCW meeting on new mail
FROM:(Agent)%Ichimura
CLASS:(MailClass)MeetingAnouncement
DATE:(Date)March 15, 1991
TIME:(Time)13:00:00
PLACE:(Place)Matsushita lab
PARTICIPANTS:(AgentSet)%(Matsuura Tsukada Hiraiwa)
TOPIC:(String)TCW meeting on new mail
TO:(AgentSet)%(Matsuura Tsukada Hiraiwa)
SUBJECT:(String)'TCW meeting on new mail'
NOTIFY:(Proc)[Schedule notify: SUBJECT date: DATE time: (TIME subtractHours: 1)].

図 5: 電子メイルで転送されるデータ

FROM:	%Ichimura
SUBJECT:	'TCW meeting on new mail'
DATE:	March 15, 1991
TIME:	13:00:00
PLACE:	Matsushita lab
PARTICIPANTS:	%(Matsuura Tsukada Hiraiwa)
TOPIC:	'TCW meeting on new mail'
NOTIFY:	[Schedule notify: 'TCW meeting on new mail' date: March 15, 1991 time: (13:00:00 subtractHours: 1)].

図 6: 受信者の受け取るテンプレート

メイルで転送されるデータは図 5のようなプレーンなテキストであり、既存の電子メイルを利用できる。

ミーティングの参加者の手にするメイルは図 6である。受信者が依頼された処理を了解すればブロック内の手続きの評価を行い、これにより各自のスケジュールマネージャーにノーティフィケーションの指示が伝わる。スケジュールマネージャーの振る舞いは各自が独自に規定するものであり、チームでの共通の認識を守れる範囲で notify: date: time:メソッドの具体的処理を変更できる。これは、オブジェクト間のメッセージを規定しているプロトコルはそのままにして、データ構造やそれへの手続きを変えることができるというオブジェクト指向プログラミング手法と類似した考え方である。

4.2 情報処理手続きの転送の応用例

ここでは受信者への情報処理手続きの転送の応用例を挙げ、上述した PilotMail システムの使用例を参照しながら説明する。

```
NOTIFY: | %Okada doIt:
        | [(AgentSet input) doIt:
        | [Schedule
        |    notify: SUBJECT
        |    date: DATE
        |    time: (TIME subtractHours: 1)].].
```

図 7: doIt:メソッドのネスト

```
PLACE: | %Okada doIt:
       |    [Place input.
       |    selfmail reply.].
```

図 8: PLACE:フィールドの問い合わせ

1. 図 3の NOTIFY:フィールドのプロシジャを図 7のように記述したとする。これは doIt: メソッドをネストにした形である。この例では、Ichimura が送信したメイルはまず Okada に届けられ、その後、Okada がプロシジャの実行に賛同すれば Okada が指定したメンバーにメイルは送られて、そのメンバーのスケジュールにノーティフィケーションのための予約がなされる。たとえば、Okada が推薦する人をこのミーティングに加えたいというような時、Okada の手を極力煩わせずに、Ichimura は Okada の推薦する人のスケージュールに予定を入れることができる。Okada の推薦した人がたとえ Ichimura と面識が無かったとしても、メイルは Okada から送られてきたものであるので、Okada からの依頼としてそのミーティングに参加するということになる。

2. 図 2の PLACE:フィールドのプロシジャを図 8のように記述したとする（図中の selfmail はこのメイル自身を指している変数、また、reply は送信者へメイルを返送するメソッド）。これはグループミーティングを招集しようとするユーザ Ichimura が、ミーティングの場所をどこにすればよいかを Okada に問い合わせようとする例である。このように、構造化メイルの中の MeetingAnouncement 構造を選択することで、未決定部分を他のメンバーに問い合わせることができる。すなわち、Lens システムのような質問専用の構造を用いることなく、質問のためのメイルを記述することが可能である。これにより、仕事に適合した構造で質問のためのメ

イルも整理できる。

5 プロシジャ記述言語

PilotMail システムのプロシジャ記述言語は、オブジェクト指向プログラミング言語のSmalltalk-80 が基盤となっている [11][12] 。これは、オブジェクト指向プログラミング言語の持つ「ソフトウェアを手早く、簡便に、しかも分かりやすく記述できる」という特長が、メイルのプロシジャ記述にも大きく貢献するという理由からであり、くわえて、オブジェクト指向プログラミング言語の目指す「計算機による問題解決や情報処理を試みる対象領域を、オブジェクトというできるだけ自然な形で計算機内にモデル化する」という理念が、人と人のインタラクションを計算機内にモデル化するのに適していたからである。すなわち、数多くのメイルを送受信しなければならないユーザにとって、プロシジャ記述のための労力や所用時間は極力小さくなければならず、負担であってはならない。また、計算機に精通していない人にとっても、問題解決手続きや情報処理手続きが容易に記述できるものである必要があり、それらを自然な形で書けることが望ましいのである。

オブジェクト指向言語を採用することで以上のようなメリットが得られるが、特にSmalltalk-80の言語仕様およびソフトウェア開発環境を導入することで、Smalltalk-80のシステムブラウザの多くの特長を利用できるようになる。具体的には、ユーザが今必要とする機能がどういう位置づけで、システムのどこにあるのかをシステムブラウザで見つけだすことができたり、Smalltalk-80システムがあらかじめ用意している実例（example メソッドなどで起動される）を参照できること、あるいは、システムへの説明要求（コマンド explain により作動）をだせるというようなことである。すなわちこれらは全て、ユーザのプロシジャ記述のための労力や所用時間を削減するのに大きく役立ち、メイル記述時の負担を軽減する。

さらに、Smalltalk-80の言語仕様は本来、ユーザがプログラムを組むと、そのプログラム自体がインタラクティブなマニュアルとしての役割を持つという性質を備えており、プロシジャ作成自体がそのドキュメンテーション作成でもあるという特長がある。

以上のような理由から、Smalltalk-80 がPilotMail システムのプロシジャ記述言語の基盤となった。このSmalltalk-80 の言語仕様の一部を変更することによりPilotMail プロシジャ記述言語は記述できる。以下にPilotMail プロシジャ記述言語に特有の仕様を示す。

- Agent クラスがObject クラスのサブクラスとして存在し、そのインスタンスはユーザを特定するID あるいはグループを特定するID を保持するものである。また、Agent クラスのインスタンスはリテラルとして記述することができる。Agent 定数は

  %Ichimura

  のようにユーザ識別子に%を前置したもので、PilotMail システムがAgent クラスのインスタンスとして認識し、Smalltalk-80 の解釈できる式に変換する。

- AgentSet クラスがSet コレクションのサブクラスとして存在し、そのインスタンスはAgent クラスのインスタンスの集合を保持する。また、AgentSet クラスのインスタンスはリテラルとして記述することができる。AgentSet 定数は

  %(Matsuura Hiraiwa Matsushita)

  のように要素を%(と) でくくって書き、AgentSet クラスのインスタンスになるが、要素として書けるのはAgent 定数だけである。AgentSet 定数はPilotMail システムによってAgentSet クラスのインスタンスとして認識され、Smalltalk-80 の解釈できる式に変換される。

- PilotMail の使用例中に登場した、AgentSet クラスあるいはAgent クラスのインスタンスメソッドであるdoIt:はその引数にブロック式をとるキーワード・メッセージである。そのブロック式はメイルの送信者が記述するが、そのブロックにvalue メッセージを与えるのはメイルの受信者（Agent クラスのインスタンスによって特定されるユーザ）であり、再びメイルの送信者へ実行の流れが帰

ることはなく、その意味でPilotMailプロシジャ記述言語に特有の仕様である。AgentSetクラスあるいはAgentクラスのdoIt:メソッドはPilotMailシステムによってSmalltalk-80の解釈できる式に変換されるが、具体的な処理は電子メイルとのインタフェース操作が主になる。

6 おわりに

本稿では、送信者の意図を正確に受信者に伝えるために、メイルの構造化と受信者への情報処理手続き転送に基づいたPilotMailシステムを提案した。また、誤解の発生のメカニズムに対する我々の考察を呈示し、これに基づいて、誤解を極力予防するための手法を述べ、PilotMailシステムでどのように対処しているのかを述べた。さらに、チームの協同作業の進め方をオブジェクト指向知識表現の計算モデルに対応づけることにより、情報処理を行う際のチームのメンバー間のインタラクションを自然な形で計算機内にモデル化することができた。この理念は、PilotMailシステムのプロシジャ記述言語の設計の根底となっている。

現在、このメイルシステムを拡張して、掲示板機能、協同執筆支援への応用を検討しており、我々の開発したチームウェアデータベース[13]と融合させることにより更に利用価値の高いシステムへと発展させたいと考えている。

参考文献

[1] 石井裕, 大久保雅且, " コンピュータを用いた人間の共同作業支援技術について ", マルチメディア情報と分散協調シンポジューム論文集, 情報処理学会, 1989.

[2] 市村, 松浦, 岡田, 松下, " 分散協調型作業支援システムーチームウェア ", 1990年代の分散処理シンポジウム論文集, 情報処理学会, 1990.

[3] 市村, 塚田, 伊藤, 松下, " PilotMail:グループ会話支援メイルシステム ", 情報処理学会第42回全国大会, 1990.

[4] S. Ichimura, N. Matsuura, K. Okada, Y. Matsushita, " A Database System Suitable for Team Cooperative Work ", Proc. Future Databese'90 Far-East Workshop on Future Database System, April 1990.

[5] N. Matsuura, S. Ichimura, S. Hiraiwa, K. Okada, Y. Matsushita, " A Teamware Workbench for Multimedia Information Management ", Proc. International Computer Symposium '90, December 1990.

[6] Terry Winograd, Fernando Flores, " Understanding Computers and Cognition ", Addison-Wesley Publishing Company Inc., 1986.

[7] Terry Winograd, " Where the action is ", BYTE, December 1988.

[8] Kum-Yew Lai and Thomas W.Malone, " Object Lens: A Spreadsheet for Cooperative Work ", Proc. CSCW'88, 1988.

[9] Thomas W.Malone, et al, " Semistructured messages are suprisingly useful for computer-supported coordination ", ACM Transactions on Office Information Systems, 1987.

[10] Stephen Pollock, " A Rule-Based Message Filtering System ", ACM Transactions on Office Information Systems, July 1988.

[11] A. Goldberg, " SmallTalk-80: The Language and Its Implementation ", Addison-Wesley, reading, 1983.

[12] 竹内 郁雄, 及川 一成 他, " 特集:Smalltalk-80 ", Computer Today, 1984.

[13] S. Ichimura, N. Matsuura, S. Hiraiwa, K. Okada, Y. Matsushita, " A Teamware Workbench for Information Management and Associative Retrieval in the Distributed Environment ", IEEE CS Proc. 1st International Workshop on Interoperability in Multidatabase Systems, April 1991 (to be appeared).