協調作業モデル記述言語の設計

樋地正浩*1　布川博士*2　白鳥則郎*2

*1 日立東北ソフトウェア　*2 東北大学電気通信研究所

概要

協調作業を支援するシステムの構築には，協調作業の分析によるモデル化とそれに基づく記述言語が必要である．本稿では，社会学的分析に基づき，自律的オブジェクトを持つ協調型計算モデルを用いてモデル化した協調作業モデルを記述するために設計した言語系について述べる．また，あわせてこの言語系を用いて協調作業のためのコミュニケーションが記述できることを示す．本稿で述べた記述言語は，メンバの自律的な行動を記述する関数，基本的コミュニケーションを記述する関数を持ち，これらの組み合せにより協調作業モデルを記述する．

和文キーワード

協調作業，ＣＳＣＷ，グループウェア，コミュニケーション，協調型計算モデル，自律

A Design Language Describing a Cooperative Work Model

Masahiro HIJI *1　Hiroshi NUNOKAWA *2　Norio SHIRATORI *2

*1 Hitachi Tohoku Software Co.,Ltd
*2 Research Institute of Electrical Communication , Tohoku University

Abstract

It is necessary to develop computer support for cooperative work based on modeling cooperative work and a description language .

In this paper , We propose a cooperative model based on the analysis of cooperative work in terms of groups , members and human communication from social science . This model is based on a cooperative computational model with autonomous objects . In this paper we explain the design and specifications of a language for a description of this model . Next we show some examples of communications in cooperative work using this language .

This language has some functions for describing autonomous object movement and communication among autonomous objects .

英文キーワード

Cooperative Work , CSCW , Groupware , Human Communication , Cooperative Computation Model , Autonomous

1．はじめに

コンピュータを単に個人の事務作業の代替のためのツールとして利用するのではなく，人々の協調作業を支援するシステムとして利用するための新たな研究分野（ＣＳＣＷなど）が注目され，現在盛んに研究が行なわれている．協調作業支援システムを構築する上では，システム構築手法という単なる技術的側面のみでなく，支援対象である人々の協調作業の分析という協調作業自身に関する分析や議論も必要である．

これらをふまえ我々は，協調作業の分析およびその計算モデル化という立場から，協調作業を行なう人々の間のコミュニケーションに焦点をあてた分析を行なっている[1]．協調作業の計算モデル化の目的は，（１）計算モデルを用いた人間の協調作業および協調作業におけるコミュニケーションの一般化，（２）協調作業支援のための共通プラットフォームの構築，および（３）計算モデルに基づくプログラミング言語の作成とこの言語による種々の協調作業支援システムの構築，の３点である．

我々は以上の考えに基づき，協調型計算モデル[3]に自律性を導入した自律的オブジェクトによる協調作業モデルを提案した[1]．我々の提案した計算モデルの特長は，協調作業を（１）集団の形態とその構成メンバ，および（２）集団間やメンバ間のコミュニケーション，というの２つの側面から分析し，それぞれを計算モデル内で扱えるようにしたことにある．

本稿では，[1]の計算モデルに基づくプログラミング言語の設計，実装および協調作業の記述例について述べる．このプログラミング言語は，[1]の計算モデルに基づくプログラミング言語の一例であり，[1]の計算モデルの実現の可能性の実証である．この言語はUNIX ワークステーションで構成されるネットワーク上に実装されており，実際に複数のワークステーションを用いて協調作業の記述が行なえるようになっている．本稿ではこの言語自身について，その機能を述べると共に，いくつかの協調作業の記述を試みる．

本稿の構成は以下の通りである．はじめに２章において[1]にそって，協調作業モデルに必要な要素について議論し分析を試みる．次にこれらの議論を踏まえ，自律的オブジェクトを持つ協調型計算モデルにより協調作業のモデル化を行なう．３章では２章の協調作業モデルを記述するために設計した言語系について説明する．４章では，３章で設計した言語系を用いて，協調作業のプラットフォームとなるコミュニケーションの記述例について示す．５章は本稿のまとめである．

2．自律的オブジェクトを用いた協調作業モデル

2.1 協調作業のモデル化に必要な要素[1]

ここでは，協調作業自身の分析に基づき，協調作業を行なう人々（メンバ）間で行なわれるコミュニケーション，グループとメンバの関係という視点から，協調作業のモデル化に必要な要素を[1]に基づき考察する．

協調作業は，グループの目標を達成するために，状況に応じてコミュニケーション対象を選択し，選択した対象に応じた適切なコミュニケーション行為により目的を達成していく作業と考えることができる．このようにコミュニケーションは協調作業を行なう上で必要不可欠なものである．さらに，協調作業におけるコミュニケーションは，何か特定の対象や特定のコミュニケーション行為が行われるのではなく，状況や必要に応じてさまざまな対象やコミュニケーション行為が選択され，コミュニケーションが行われる．

協調作業を行う人々（メンバ）の集団であるグループは動的な側面を持つ．すなわち，グループはメンバや他のグループにより必要に応じ作られたり，目的を達成すると解散したりする．また，グループのメンバは，グループの目的を達成するためにグループに参加したり，自分の役割が終われば退出するなど，自律的にグループ間を移動する．また[2]で述べられているように，あるメンバ

は複数のグループに所属し，各々のグループの作業を同時並行的に行なっていると考えることができる．

すなわち協調作業モデルには，コミュニケーション，グループとメンバの関係について以下の点を表現できる能力が求められる．

（1）協調作業の場であるグループとその動的な側面の表現

（2）グループを構成するメンバの動的な側面の表現

（3）メンバ間で行われるさまざまなコミュニケーション形態の表現

（4）目的に応じたコミュニケーション対象と形態の選択

（5）コミュニケーションによるメンバの変化の表現

協調作業をモデル化するための計算モデルには，これらの記述能力を満たすことが求められる．

2.2 自律的オブジェクトを用いた協調作業モデル

我々の提案している協調作業モデルは，[3]の協調スコープを持つ協調型計算モデルに自律性を導入した計算モデルを用いている．本稿ではこの計算モデルを，自律的オブジェクトを持つ協調型計算モデルと呼ぶ．自律的オブジェクトを持つ協調型計算モデルは，2.1節で述べた（1）～（5）の要素を満たす計算モデルであり，この計算モデルにおける自律的オブジェクトは，[3]の協調型計算モデルの計算主体の中で，以下のような自律性を持つものである．

・計算主体の協調必要性が生じた時に動的に交信対象を選択

・交信対象や交信対象の状況を判断し，状況に応じた交信形態を選択

・どのフィールドに属するかを計算主体自身が決定できる

・交信により協調の仕方や処理の仕方などの計算主体自身の行動を変更できる

モデル化の対象	協調作業モデルの表現
メンバ	自律的オブジェクト
グループ	フィールド
作業中のグループ	協調スコープ

表2.1 協調作業のモデル化

自律的オブジェクトを持つ協調型計算モデルを用いて協調作業をモデル化すると表2.1のように表現できる．このモデルでは，メンバのグループへの参加条件は，フィールドの境界条件として記述する．同一のフィールドに属する自律的オブジェクトは協調する可能性がある．また，同一の協調スコープを開いている自律的オブジェクト同士は，即座に交信可能であり，ある作業を同時に行なっていると言える．表2.1のようにグループとメンバを規定することにより，メンバ間のコミュニケーションは，自律的オブジェクトを持つ協調型計算モデルの交信形態を用いて，表2.2のように表現できる．

3．協調作業モデルの記述

本章では，2.2節で述べた表現能力を持つ自律的オブジェクトについて，その基本構成とその記述言語について述べる．

交信形態	メンバ間のコミュニケーション行為
通知型契約交信	・他のグループの誰かとの意見，質問，議論 ・グループのメンバとの意見，質問，議論
通知型非契約交信	・他のグループへの連絡，説明 ・グループのメンバへの連絡，説明
協調型契約交信	・グループ内の同時に作業をしているメンバやその中の特定のメンバへの質問，意見，議論
協調型非契約交信	・グループ内の同時に作業をしているメンバやその中の特定のメンバへの説明，連絡
メッセンジャ[5]	・同時に作業していない特定メンバへの質問等 ・条件を指定した不特定の人への説明，連絡

表2.2 交信形態とコミュニケーション行為

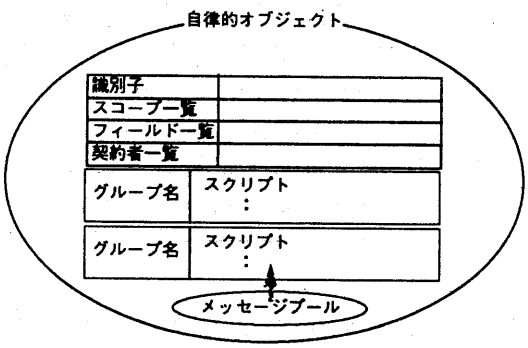

図3.1 自律的オブジェクトの構造

3.1 自律的オブジェクトの基本構成

自律的オブジェクトは，図3.1に示す構造を持つ．

識別子とは個々の自律的オブジェクトを特定するものであり，自律的オブジェクトを他の自律的オブジェクトと区別するために用いられる．

スコープ一覧とは自律的オブジェクトの持つスコープ変数の名称の一覧であり，動的に協調する対象を選択するために用いられる．

フィールド一覧とは自律的オブジェクトの所属するフィールドの名称の一覧であり，自律的オブジェクトの所属している全てのフィールドを管理するために用いられる．

契約者一覧とは契約型交信を行なう場合の契約相手の識別子の一覧であり，契約相手を保持するために用いられる．自律的オブジェクトは，契約型交信により申し込まれた契約を受諾した自律的オブジェクトの識別子の一覧を保持する．結んでいた契約関係を解除した場合，その自律的オブジェクトの識別子は契約者一覧から削除される．

スクリプトとは自律的オブジェクトの処理内容を記述したものであり，メッセージに対応したスクリプトにより実際の処理を行う．スクリプトは，自律的オブジェクトの所属するフィールドごとに記述される．

メッセージプールとは他の自律的オブジェクトからのメッセージを格納するものであり，他の自律的オブジェクトからきたメッセージを一時的にプールしておくために用いられる．自律的オブジェクトはメッセージプールから1つづつメッセージを読み込み，メッセージに対応したスクリプトにより処理を行う．

3.2 グループと自律的オブジェクトの記述

協調作業モデルの集団の形態と構成メンバの関係は，（1）フィールドに関する記述（2）フィールドと自律的オブジェクトの関係の記述，（3）自律的オブジェクトに関する記述，の3つの関数群により記述される．これらの関数群とその機能を表3.1に示す．これらの関数群により，（1）メンバによる動的なグループの設立や解散，（2）メンバのグループへの参加や退出，（3）協調作業を行なうグループの指定，という2.1節で述べたグループとメンバの関係を記述することができる．

3.3 自律的オブジェクトのコミュニケーションの記述

自律的オブジェクトは，表2.2に示したコミュニケーション形態に応じた複数の交信形態を持つ．自律的オブジェクトは，必要に応じて適切な交信形態を選択し，コミュニケーション対象にメッセージを送ることによりコミュニケーションできる．交信形態を通じてやり取りされるメッセージは以下のリスト構造を取る．

メッセージ：（交信形態，送信先グループ名，送信先オブジェクト名，送信元グループ名，送信元オブジェクト名，テキスト）

自律的オブジェクトは，伝えるべき内容を上記のメッセージの形式にし，交信形態を用いて送信先グループに対応するフィールドに送る．フィールドは，メッセージを受け取ると，交信形態と送信先オブジェクト名の値に応じてグループ内の自律的オブジェクトへメッセージを伝える．自律的オブジェクトは，フィールドを介してメッセージ

項　目	関　数	機　能
フィールドの生成	(DefineGroup <GroupName> (MakeScope <ScopeName>) (InCondition <Condition>))	・<GroupName> の識別子を持つフィールドの生成 ・フィールドの作業時に使用する<ScopeName>，フィールドへの参加条件<Condition>を持つ
フィールドの破棄	(DeleteGroup <GroupName>)	・<GroupName>の識別子を持つフィールドの破棄
フィールドへの参加	(InGroup <GroupName> <ObjectName>)	・<GroupName>の識別子を持つフィールドへの参加
フィールドからの退出	(OutGroup <GroupName> <ObjectName>)	・<GroupName>の識別子を持つフィールドからの退出
自律的オブジェクトの生成	(DefineAutonomousObject <ObjectName> <ObjectMovement>)	・<ObjectName>の識別子を持つ自律的オブジェクトの生成 ・<ObjectMovement>は自律的オブジェクトの行動の仕方
作業フィールドの指定	(Open <ScopeName> <ObjectName>)	・<ScopeName>の識別子を持つフィールドの作業を行う
作業フィールドの変更	(Change <ScopeName> <ShutScopeName> <ObjectName>)	・<ShutScopeName>から<ScopeName>へ作業フィールドを変更
作業フィールドの終了	(Shut <ShutScopeName> <ObjectName>)	・<ShutScopeName>の識別子を持つフィールドの作業を終了

表3.1 フィールドと自律的オブジェクトの記述関数群

を受け取るとそのメッセージを伝えた交信形態に応じてメッセージを受け取り，受け取ったメッセージの処理を行なう．メッセージの処理中にグループから送られた新たなメッセージはメッセージプールに格納され，処理中のメッセージの処理が終了した後に，メッセージプールから読み出され，処理される．

協調作業モデルのコミュニケーションは，（1）フィールドに属する対象とのコミュニケーションの記述，（2）同時に作業をしている対象とのコミュニケーションの記述，（3）不特定の相手とのコミュニケーションの記述，の3つの関数群により記述される．これらの関数群とその機能を表3.2に示す．

3.4 言語の実装

3.2節，3.3節で述べた協調作業モデルを記述する言語系は，DeLis (Decentralized Lisp Interpreters) [4]と呼ぶ，分散環境を記述するための言語系の上に作成されている．DeLis は Lisp をベースに通信機能とGUI記述機能を付加した言語であり，プログラムはリスト形式の形で記述され，インタプリタにより実行される．また取り扱うプログラムやデータもリスト形式の形で記述され，プログラムをデータとして取り扱うことも，データをプログラムとして取り扱うこともできる．DeLisインタプリタは分散システム上に複数存在し，各DeLisインタプリタは通信機能を通じてデータの授受を行なう．

交信形態	関　　数	機　　能
通知型非契約交信	(InformSend <toGroup> <fromGroup> <fromObject> <Message>)	・<toGroup>の識別子を持つフィールドに<Message>の内容を伝える
通知型契約交信	(Contract <toGroup> <fromGroup> <fromObject> <contractCondition>)	・<toGroup> の識別子を持つフィールドに<contractCondition>の条件で契約を申し込む
	(ContractReply <toGroup> <toObject> <fromGroup> <fromObject>)	・<toGroup>の<toObject>の識別子を持つ自律的オブジェクトへ契約の受諾を伝える
	(Send <toGroup> <toObject> <fromGroup> <fromObject> <Message>)	・<toGroup>の<toObject>の識別子を持つ自律的オブジェクトに<Message>の内容を伝える
	(ContractEnd <toGroup> <toObject>)	・<toGroup>の<toObject>の識別子を持つ自律的オブジェクトに契約の解除を伝える
協調型非契約交信	(WhoInTheScope <scopeName>)	・<scopeName>を開いている自律的オブジェクトの識別子の一覧を得る
	(Send <toGroup> <toGroup> <fromObject> <fromObject> <Message>)	・<toGroup>の<toObject>の識別子を持つ自律的オブジェクトに<Message>の内容を伝える
協調型契約交信	(WhoInTheScope <scopeName>)	・<scopeName>を開いている自律的オブジェクトの識別子の一覧を得る
	(Contract <toGroup> <fromGroup> <fromObject> <contractCondition>)	・<toGroup> の識別子を持つフィールドに<contractCondition>の条件で契約を申し込む
	(ContractReply <toGroup> <toObject> <fromGroup> <fromObject>)	・<toGroup>の<toObject>の識別子を持つ自律的オブジェクトへ契約の受諾を伝える
	(Send <toGroup> <toObject> <fromGroup> <fromObject> <Message>)	・<toGroup>の<toObject>の識別子を持つ自律的オブジェクトに<Message>の内容を伝える
	(ContractEnd <toGroup> <toObject>)	・<toGroup>の<toObject>の識別子を持つ自律的オブジェクトに契約の解除を伝える
メッセンジャ[5]	(DefineMessenger <MessengerName> <Information> <Script> <ScriptForm>)	・<Information>の情報と<Script>の機能の<MessengerName>の識別子を持つメッセンジャを生成
	(SendMessenger <toObject>)	・メッセンジャを<toObject>の識別子を持つ自律的オブジェクト に送り出す
	(ReceiveMessenger)	・メッセンジャを受け取る

表3.2 コミュニケーションの記述関数群

設計した協調作業モデルを記述する言語系をDeLisを用いて作成することにより，分散システムを意識することなく，協調作業モデルを素直にインプリメントすることができた．DeLisはインタプリタ実行であるため，フィールドや自律的オブジェクトの動的な生成，破棄といった動的な側面を記述しやすい．また，データをプログラムとして取り扱うことができるため，コミュニケーション対象に送るメッセージのテキストにリスト形式で処理の内容や協調の仕方を記述し，受け取ったリスト形式を実行することにより，動的に処理の仕方や協調の仕方を変更することもできる．さらにこの言語系は，DeLisの動作する計算機であればそのまま動作するため，容易に異機種から構成されるネットワーク上の協調作業支援のためのプラットフォームとすることができる．

4．コミュニケーションの記述例

以下では，3章で述べた協調作業モデルとその記述関数を用いた協調作業におけるコミュニケーションの記述を示す．

4.1 グループのメンバ全員への説明

主に説明を中心とした会議などでは，ある特定のメンバからの内容の説明が中心になる．これは特定のメンバからグループのメンバへの一方向のコミュニケーションと見なすことができる．このような形態は，開催案内の連絡などグループで作業を行なう場合，さまざまなところで用いられる．

これは，図4.1に示すように，会議をしているグループをフィールドで，グループに所属するメンバを自律的オブジェクトで表現することにより，通知型非契約交信を用いて記述することができる．発言者となる自律的オブジェクトは，会議フィールドに対して通知型非契約交信を用いて説明する内容を順次送信する．会議フィールドは，フィールドの持つ自律的オブジェクト一覧を参照し，そこに保持されている識別子を持つ自律的オブジェクト（グループの参加メンバ全員）に契約型非契約交信により送られた説明内容を順次伝える．伝えられた説明内容は，個々の自律的オブジェクトで処理される．説明の途中で新たにグループに参加した場合，フィールドの自律的オブジェクト一覧に，新たに参加した自律的オブジェクトの識別子が加えられる．その結果，参加した時点から，他の自律的オブジェクトからの通知非契約交信によるメッセージを受け取ることができる．すなわち，フィールドに属するほかの自律的オブジェクトとのコミュニケーションができるようになる．

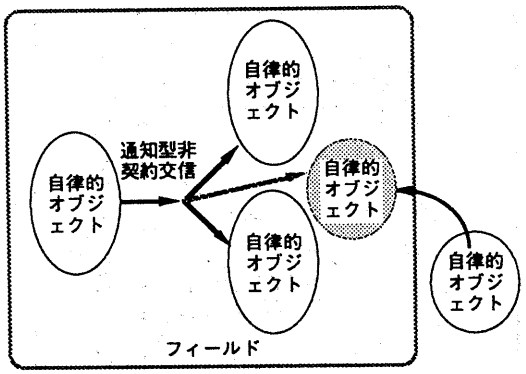

図4.1 グループのメンバ全員への説明

このプログラムは，表3.1，表3.2の記述関数を用いて図4.2のように記述することができる．

4.2 グループの特定メンバとの質問，意見のやり取り

ゼミなどのように議論を中心とした協調作業では，グループの個々のメンバが必要に応じて議論するメンバを変更しながら，個々の作業を進めて行くことになる．これは，図4.3に示すように議

```
(DefineAutonomousObject 'UserA              ;自律的オブジェクト
    '(UserObject  message name channel) ) ; UserAの生成

(InGroup 'Group1 'UserA )                   ; UserA の Group1
                                            ; フィールドへの参加

(defun ObjectMovement ( message name channel)
 (case (nth 0 message)                      ;自律的オブジェクトの動作
   ('In                                     ;フィールドへの参加
     (progn (setq toGroup (GetGroupName ))  ;参加フィールドの指定
            (InGroup toGroup (nth 2 message))
            (setq toMsg (ChangeMessage message 0 'null))
            (list toMsg t)))
   ('Out                                    ;フィールドからの退出
     (progn (setq toGroup (GetGroupName )) ;退出フィールドの指定
            (OutGroup toGroup (nth 2 message))
            (setq toMsg (ChangeMessage message 0 'null))
            (list toMsg t)))
   ('InformSend                             ;通知型非契約交信
     (progn (setq fromGroup  (nth 1 message))
            (setq fromObject (nth 2 message))
            (setq toGroup (GetGroupName )) ; 送信先フィールドの指定
            (setq message (GetMessage))    ; 送信メッセージの指定
            (list (InformSend toGroup null
                    fromGroup fromObject  message) 'null)))
   ('InformContractSend
```

図4.2 記述関数によるプログラム例（一部）

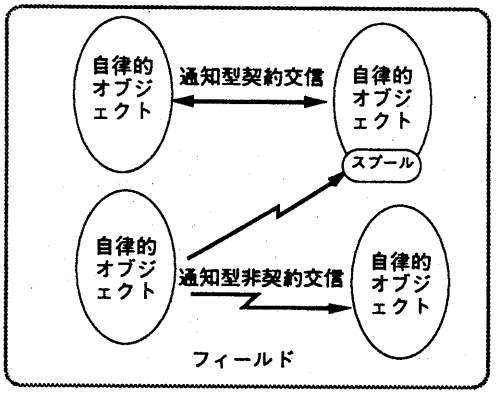

図4.3 特定のメンバとの質問，意見のやり取り

論をしているグループをフィールドで，グループに所属するメンバを自律的オブジェクトで表現することにより，主に通知型契約交信を用いて記述できる．議論したい内容を持つ自律的オブジェクトは，通知型契約交信の Contract 関数を用い，フィールドに対して議論するテーマを <contractCondition> に記述し，フィールドに属する自律的オブジェクトに契約の申し込みを行なう．フィールドは，フィールドの自律的オブジェクトからテーマに関して契約の申込がなされたことをフィールドに属する全ての自律的オブジェクトに伝える．契約申込を受け取った各自律的オブジェクトは，その内容について議論に参加するか否かを判断し，参加する場合には ContractReply 関数を用いて契約を申し込んだ自律的オブジェクトとの間で契約関係を結ぶ．契約関係を結んだ自律的オブジェクト間では，Send 関数を用いて議論を行なう．議論を終了する場合は，ContractEnd関数を用いて，契約の解除を伝える．議論に参加していない自律的オブジェクトは，いつでも他の議論を行なうことができる．また，一つのフィールド内で複数の内容について議論することも可能である．通知型契約交信により議論をしている自律的オブジェクトに対する通知型非契約交信によるメッセージは，契約関係が解消されるまで契約関係にある自律的オブジェクトのメッセージプールに保持され，契約関係の解消時点で，読み込まれ，処理される．

5．終わりに

本稿では，[1]で提案した協調作業モデルを記述するために設計した記述言語について述べた．本稿の記述言語は，自律的オブジェクト自身の行動に関する関数とコミュニケーションに関する関数からなる．これらの関数により，自律的オブジェクトを用いた協調作業のためのコミュニケーションを容易に記述することができる．この記述言語をDeLis 上に実装することにより，グループやメンバの持つ動的側面を容易に記述することができただけではなく，メンバの処理や協調の仕方も柔軟に記述することができる．また，DeLis の動作する各種の計算機上で利用できるため，協調作業支援のための共通プラットフォームとすることができる．

本稿で述べた記述言語は，基本的なコミュニケーションを記述するものであり，実際の協調作業で行なわれるコミュニケーションを記述するためには，これらの基本的関数を組み合せ，コミュニケーションプロトコルを記述する必要がある．今後，このコミュニケーションプロトコルを記述するための枠組について検討する必要がある．

参考文献

[1]樋地，布川，白鳥：自律的オブジェクトによる協同作業のモデル化，情報処理学会グループウェア研究会報告 93-GW-2-2 pp.9-16 (1993)

[2]塚田，岡田，松下：作業の多重性に着目した協調作業支援，情報処理学会グループウェア研究会資料　92-GW-3-4 pp.25-32.(1992)

[3]武宮，矢野，布川，野口：協調スコープを持つ協調型計算モデル，情報処理学会プログラミング研究会資料　91-PRG-3 pp.37-46.(1991)

[4]三石，布川，宮崎，野口：分散環境のための言語系 DeLis，情報処理学会プログラミング-言語・基礎・実践-研究会資料　93-PRG-10-8 pp.57-64 (1993)

[5]五十嵐，布川，野口：自律分散協調型計算モデルを用いたコミュニケーションツールの記述，情報処理学会 マルチメディア通信と分散処理研究会報告 92-DPS-58-21 pp.165-172. (1992)