

マルチフェースエージェントを用いたヒューマン コミュニケーションのためのツール

小山 和也†, 布川 博士‡, 白鳥 則郎†

† 東北大学電気通信研究所 / 情報科学研究科

‡ 宮城教育大学理科教育研究施設

概要

人間のコミュニケーションの一部を代行することで協同作業を支援するシステムとして、マルチエージェントモデルに基づくコミュニケーションツールが提案されている。

本稿ではこのようなシステムの1つとして既に提案されているマルチフェースエージェントを拡張し、周囲の状況によって動作を変更することを容易に表現できるエージェントモデル、マルチシチュエーションエージェントを提案する。マルチシチュエーションエージェントは複数のオブジェクトによって構成され、周囲の状況のチェック機構を持つエージェントである。

A Tool for human communication using Multi Situational Agent

Koyama Kazuya†, Nunokawa Hiroshi‡, Shiratori Norio†

† Research Institute of Electrical Communication / Graduate School of Information Science,
Tohoku University

‡ Research Institute for Science Education, Miyagi University of Education

abstract

Communication tools based on Multi-Agent model with agents acting on behalf of the user are being considered as a means of supporting the cooperative works.

In this paper, we suggest Multi-Situational-Agent model, which is an extension of Multi-Face-Agent already suggested as a kind of such system. Multi-Situational-Agent model can easily express changes of agent's movement which depend on environmental situation. Multi-Situational-Agent is an agent which consists of multiple objects and has a mechanism to check environmental situation.

1 はじめに

近年、コンピュータネットワークの発達とその個人利用の増加に伴い、コンピュータネットワークを人間のコミュニケーションのためのメディアとして利用できる可能性が高まってきた。これに伴い、コンピュータを、従来の一人の人間の作業を支援するものでなく、複数の人間による協同作業を支援するために利用しようとする、グループウェアシステムの研究が盛んになって来ている。

このグループウェアシステムは既に様々なものが提案されているが、その中の1つにマルチエージェントモデルに基づくコミュニケーションツールというものがある。これは、「書類」や「秘書」などといった協同作業に関わる要素をエージェントとして表現し、それぞれにコミュニケーションの代行機能を持たせ、ユーザがこれらのエージェントを媒体としたコミュニケーションを行なうことで協同作業に於けるユーザのコミュニケーションに関する負担を軽減するものであり、システムの柔軟性が高く有効なアプローチであることから、既にいくつかのシステムが提案されている[1, 2]。

しかしながら、既存のモデルに基づくシステムでは、エージェントに柔軟な動作を行なわせようとする、その動作表現が複雑になってしまうという問題があった。特に、人間のコミュニケーションというものは、人間関係、相手の状態、周囲の環境などといった、コミュニケーションを取り巻く様々な状況によって変化する。しかし、従来のモデルではこのような状況によるエージェントの動作の変更の表現を考慮していないため、このような状況によって変化するコミュニケーションを支援するエージェントを構築しようとすると、その記述が極めて複雑になってしまった。このような問題を解決する試みとして、相手によって動作を変えることを容易に表現できるエージェントモデル、マルチフェースエージェント(MFA) [3]が既に提案されているが、これは様々な状況の中の人間関係に依存した動作変更しか考慮されていなかった。

そこで本稿では、このMFAの拡張として、様々な状況によって動作を変えることを容易に表現できるエージェントモデル、マルチシチュエーションエージェント(MSA)の提案を行ない、このモデルに基づいた記述言語の実装について述べる。

以下では、第2章でMSAモデルについて述べた後、第3章で記述言語の実装、第4章で記述言語によるアプリケーションの記述例について述べ、最後に第5章で考察と検討を述べる。

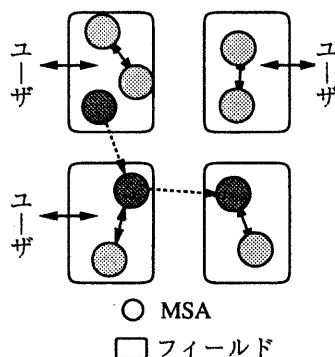


図1: システム概要

2 エージェントモデルの提案

2.1 モデル概要

本稿で提案するシステムは、従来のエージェントを拡張したモデルであるマルチシチュエーションエージェントによって構成されるコミュニケーションツールである。

このシステムは、図1のようにMSAとフィールドから構成される。フィールドは、ユーザー一人毎に一つずつ存在する、エージェントが存在するための環境であり、MSAは、固有の内部状態とメソッドを持ち、コミュニケーションを支援する機能を実現する実体である。フィールドはエージェントの動作に必要な最低限の環境を与えると同時に、システムの局所性[2]を表現する。MSAはメソッドコールを受けると予め与えられている動作表現にしたがって動作し、必要ならフィールド上を移動し、ユーザやそこにあるMSAとインタラクションを行い、最終的に得られた結果を返す。

以上は既存のモデルとほぼ同じであるが、さらにMSAは特徴として、動作中に周囲の状況を常にチェックし、その状況に適應できるように自らの動作を変更することが出来る。具体的には、状況に応じて自らを変更することが容易な構造として特殊なオブジェクトの組合せによるエージェントを構成を行ない、さらに状況を容易に表現できるような状況のチェック機構を持たせる。これにより、周囲の状況によって容易に動作を変更することを可能にする。

以下で、エージェントの詳細な定義を与える。

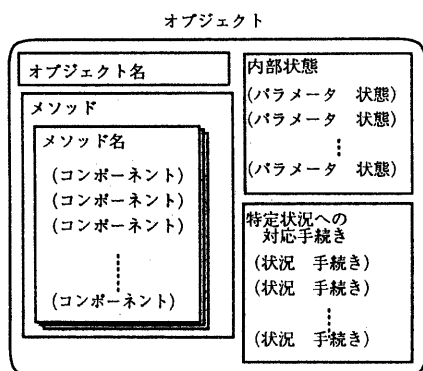


図 2: オブジェクト

2.2 エージェント構造の定義

2.2.1 オブジェクトの定義

ここでは、まず、エージェントを構成する部品となる、オブジェクトを定義する。

オブジェクトはエージェントの動作に必要な機能や知識を表現するモジュールであり、オブジェクト名、メソッド、内部状態、特定状況への対応手続き、の4つの項目によって定義される(図2)。それぞれの項目は以下の通り。

メソッド: オブジェクトが外部に提供するサービス。メソッド名と動作表現によって定義される。動作表現は、コンポーネントという動作の適当なブロックを実行順に並べる形で与える。

内部状態: オブジェクトの状態。パラメータとパラメータの取る状態によって定義される。

特定状況への対応手続き: 特定の状況に対応するようにエージェントを変更するための手続き。それぞれ、ある状況と、その状況になった時に実行されるべき手続きで表現される。(状況の表現については後述)

このようにして定義されたオブジェクトは、フィールドに記録され、MSA に読み込まれる。

2.2.2 MSA の定義

次にオブジェクトを用いて MSA を定義する。

MSA は基本的には、ある1つのオブジェクト(これを代表オブジェクトと呼ぶ)のメソッドを実現するための実体である。一方、あるオブジェクトのメソッドを実現するには、メソッドの内部で他のオブジェクトにメソッドコールを行なうなど、他のオブジェクトが必要となる場合がある。このため、代表オブジェクトの

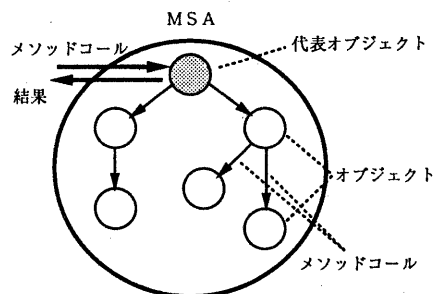


図 3: MSA

メソッドが任意のフィールドで実現可能であるためには、動作に必要なオブジェクトが常に代表オブジェクトとともに存在する必要がある。

そこで本稿では、MSA を、代表オブジェクトと、それが動作する上で必要となる全てのオブジェクトの集合体として定義する(図3)。そして MSA は、代表オブジェクトの持つメソッドをそのまま MSA のメソッドとして外部に提供し、メソッドコールを受けて動作し、結果を返す。このように定義される MSA は、コミュニケーションを支援する実体として、生成、動作、移動、削除などの基本単位となる。

2.3 状況のチェック機構

次に、先に与えた MSA の構造を用いてエージェントの動作を変更する方法を与える。以下では、まずチェックすべき状況について定義した後、状況のチェック機構と、それを用いた動作変更法について述べる。

2.3.1 状況概要

まず、本稿に於ける状況について定義する。

本稿では、状況を「コミュニケーションに影響を与える様々なパラメータの状態に対する条件の組み合わせ」とする。ここでいうパラメータとは、例えば、ユーザのログイン状態、時間、エージェントの有無、スケジューラに予定が登録されているかどうか、相手が上司であるかどうか、などといった事柄である。そして状況は、このパラメータの状態に対する条件の組合せ、すなわち「相手がログインしていて、かつ相手が上司でなく、かつ PM10:00 を過ぎている」といったような内容になる。

このパラメータは、影響を受ける MSA から見るとその存在する場所によって、

- (1) システムの外側のパラメータ
- (2) フィールド内部のパラメータ
- (3) 他の MSA の持つパラメータ
- (4) 自 MSA の持つパラメータ

の4つに分類することが出来る。そしてMSAは、状況によって動作を変更するために、常にこれらのパラメータの状態を調べ、それを予め持っている条件に照らし合わせて、自分が特定の状況に置かれているかどうかを確認しなければならない。

2.3.2 環境オブジェクトと状況オブジェクト

次に状況のチェック機構について述べる。

2.2.1で述べたように、MSAの状況による動作変更は、MSAを構成するオブジェクトの特定状況への対応手続きによって、ある状況になったときにMSA自体を変更する手続きを実行することで行なわれる。

これを実現するためには、予め状況となる条件を何らかの形で表現してMSAに持たせ、MSAの動作時に様々なパラメータの状態を調べ、それを表現されている状況と照らし合わせる必要がある。しかしながら、パラメータは様々な場所に様々な形で存在しているため、そのままでは状況の表現が複雑になってしまうという問題がある。

そこでMSAでは、状況の表現を簡単にするために、状況に関わる全てのパラメータをいったん内部状態としてMSA内部に取り込み、MSAに与える状況をMSAの内部状態に対する条件として表現する。そして、これを実現するために、エージェントを構成するオブジェクトとして、環境オブジェクトと状況オブジェクトという2種類のオブジェクトを導入する。

環境オブジェクト: MSA以外にあるパラメータの状態をMSAの内部状態に取り込むためのオブジェクト。メソッドとしてパラメータの状態を自オブジェクトの内部状態に変換する動作を持つ。

状況オブジェクト: エージェントに含まれるオブジェクトの内部状態に対する条件という形で状況を表現するオブジェクト。メソッドとして同一MSA内のオブジェクトの内部状態を調べて条件が満たされるかどうかを結果として返す動作を持つ。

環境オブジェクトによって全てのパラメータをエージェントの内部状態として取り込むことを可能にし、状況オブジェクトによって1つの状況を内部状態に対する条件と言う形で1つのオブジェクトで表現することが可能にする。

2.3.3 状況チェック機構

次に、環境オブジェクトと状況オブジェクトを用いた実際の状況のチェック機構の動作について述べる。

まずパラメータの取り込みであるが、2.3.1で述べた通り、MSAが調べるべきパラメータの存在場所は大きく4つに分類できる。このうち、(4)の自MSAの持つパラメータについては考慮する必要はなく、また(3)の他のMSAの持つパラメータについては、そのMSA

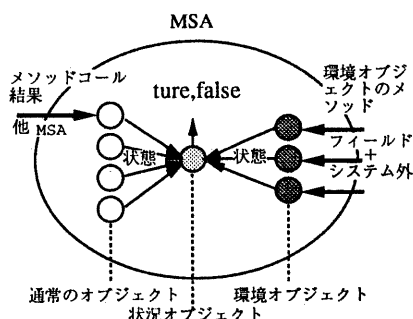


図4: 状況チェック機構

にメソッドコールを行いその結果を受け取る形で状態を調べる事になるが、メソッドコールの結果は通常のオブジェクトの内部状態として取り込まれるため、これも特に考慮する必要はない。ここで問題になるのが、(2)のフィールドや(1)のシステム外部に存在するパラメータであり、これらは何らかの形でエージェントの内部に取り込む必要がある。これを行うのがこの環境オブジェクトであり、MSAは、状況をチェックする前に環境オブジェクトのメソッドを呼び出すことで、(1)や(2)のパラメータの状態をMSA内部に取り込む。

そして、環境オブジェクトによって必要な全てのパラメータがオブジェクトの内部状態の形で表現された後に、状況オブジェクトにメソッドコールを行なうと、状況オブジェクトが同一MSAに含まれる全てのオブジェクトの内部状態を調べ、それが特定の条件を満たしているかどうかによってtrue、もしくはfalseを返す(図4)。

このようにチェック機構を構成する事で、状況に関わるパラメータの調べ方とパラメータの状態に対する条件を別に記述する事が可能になり、状況となるパラメータの取扱を統一することや、調べ方を表現する環境オブジェクトの再利用性を高める事などが可能になる。

2.3.4 MSAの適応動作

最後に、今まで述べた、オブジェクトのメソッドの動作表現、特定状況への対応手続き、状況チェック機構によるMSAの動作の全体について述べる。

MSAの状況によらない通常の動作は、オブジェクトのメソッドの動作表現であるコンポーネント列を順次実行する形で行なわれる。そして、同一オブジェクトに特定状況への対応手続きが与えられている場合、この各コンポーネントの実行前に状況への適応動作が行なわれる。具体的には、

1. MSA に含まれる全ての環境オブジェクトの呼び出し
 2. 特定状況への対応手続きで指定されている状況オブジェクトの呼び出し
 3. 2. で満たされる状況があれば、それに対応する変更手続きの実行
 4. 通常のコンポーネントを1つ実行
- という動作を各コンポーネント毎に行う。

また、特定状況への対応手続きに於ける MSA の変更手続きとしては、

- (1) コンポーネントの置き換え
- (2) パラメータの変更
- (3) オブジェクトの置き換え

などを行なう事が出来る。(1)や(2)では、予めその状況に適した知識を持っている場合のオブジェクトの変更を、(3)では、必要な知識を持たない場合の外部からの知識の取り込みなどを行なうことが出来る。

以上のように MSA を構成することで、エージェントの動作の通常動作と状況への適応動作への分割、状況の容易な表現、エージェントの構成要素のモジュール化などが可能となり、状況によって動作を変更するような複雑なエージェントを容易に構築することが可能になる。

3 記述言語の設計

次に、モデルに基づいた MSA 記述言語について述べる。

本稿では、MSA 記述言語を DeLis [4] を用いて UNIX ワークステーション上に構築した。そして、コミュニケーションツールとなるシステムを、DeLis と、DeLis 上で動く MSA インタプリタと、実際に記述された MSA によって構成し、DeLis と MSA インタプリタをあわせた部分をフィールドとした。

MSA 記述言語による MSA の記述は、部品となるオブジェクトの記述と、それを用いたエージェントの記述の2段階に分けられる。

オブジェクトはモデルをそのまま反映した形で以下のように記述できる。

```
(DefObject オブジェクト名
  (ObjectVariable (パラメータ 初期値) ...)
  (Method
    (メソッド名 コンポーネントの並び)
    ...
    (メソッド名 コンポーネントの並び))
  (Situation
    (状況オブジェクト名 変更ルール)
    ...
    (状況オブジェクト名 変更ルール))
  (ObjectList オブジェクト名の並び))
```

それぞれ、Method がメソッド、ObjectVariable が内部状態、Situation が特定状況への対応手続きを表す。加えて、現状ではオブジェクトの動作に必要な

る他のオブジェクトの名前を ObjectList で与えてやる必要がある。

次に、上記のようにオブジェクトが与えられている状態で、

(DefMSA エージェント名 代表オブジェクト名)

として MSA の名前と代表オブジェクトとなるオブジェクトを指定することで MSA を記述することが出来る。

このような記述は全てフィールドに対して行なう。フィールドに MSA の記述を与えると、フィールド上に新たに MSA が生成され、フィールドから必要なオブジェクトが読み込まれて、エージェントとして動作を開始する。

4 記述例

次に、MSA モデルの評価として、MSA 記述言語を用いたアプリケーションの記述例について述べる。ここでは概略のみを述べる。

4.1 スケジューラ

特徴: 個人の予定を記録し、他者からのスケジュールの空きに関する問い合わせに自動返答を行なうエージェント。問い合わせしてきた相手が友人か仕事関係の人かによって返す答えを変える。また予定の書き込みや予定の詳細の参照が行なえるのはエージェントの持ち主のみ。

構成: 通常のスケジューラとしての機能を通常のメソッドとして記述し、相手による動作の変更を特定状況への対応手続きとして記述する。

利点: 相手による動作の変更をスケジューラの記述とは分割することで、相手による動作変更の記述、変更が容易。

4.2 会議室予約システム

特徴: 会議の日程と参加者のリストを与えると、参加者全員の予定を確認して会議室を予約するエージェント (図5)。参加者への確認の方法が状況によって変化する。具体的には、参加者がログインしていれば直接ウィンドウを開いて確認、参加者がログインしていない場合は、その参加者のスケジューラがあればそれに問い合わせ、スケジューラもなければとりあえず後回しにして別の人の確認に移る。

構成: ログインしているユーザを調べる環境オブジェクトと、フィールド上にあるエージェントを調べる環境オブジェクトを用意し、3つの状況のための状況オブジェクトがそれぞれ環境オブジェクトの内部状態を調べることで状況をチェックする。

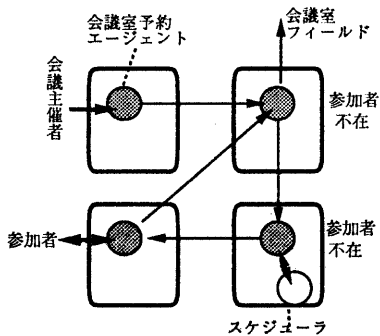


図 5: 会議室予約システム

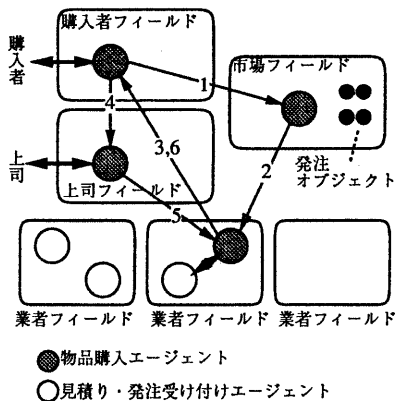


図 6: 物品購入システム

利点: ユーザのログイン状態と存在するエージェントという全く異なる事柄を、全く同じように扱うことが可能になる。

4.3 物品購入システム

特徴: 業者と品物を指定すると、その業者に見積もりを貰い、さらに上司に許可を貰って、最後に品物の発注までを行うエージェント(図6)。見積もりの貰い方や発注の方法といったことは、業者毎に異ってよい。

構成: 業者に依存した知識を記述した発注オブジェクトというオブジェクトを用意し、さらに、市場フィールドというフィールドを用意して、そこに様々な業者の発注オブジェクトを記録する。物品の購入を行うエージェントは、市場フィールドで目的の業者の発注オブジェクトを動的に取り込み、そのオブジェクトにメソッドコールすることで、見積もりや発注を行う。

利点: エージェントが業者に依存した知識を予め持つことなく、様々な業者に対応することが可能である。

5 結論

本稿では、状況によって様々に変化するコミュニケーションを容易に表現可能なコミュニケーションツールの構築を目的として、周囲の状況によって容易に動作を変更できるエージェントモデルであるマルチシチュエーションエージェントを提案し、モデルに基づく記述言語を構築した。本モデルによって、周囲の状況によって動作を変更するような複雑な動作をするエージェントを容易に構築することが可能になった。

一方、現モデルでは、コンポーネントの実行が先頭からの順次実行となっているため作業の複雑な流れを表現することが困難である、メッセージや状況による動作変更などコミュニケーションに関する事柄がプログラムとして表現されるため一般ユーザによる記述が困難になってしまっている、予め与えられている状況にしか対応できない、などといった問題点があり、今後はこれらの問題点を解決していく予定である。

参考文献

- [1] 五十嵐敏明, 布川博士, 野口正一: “自立分散協調計算モデルを用いたコミュニケーションツールの記述”, 情報処理学会研究報告, 92-DPS-58, pp.165-172
- [2] 中内靖, 伊藤嘉邦, 安西祐一郎: “Michel: マルチエージェントモデルに基づく協調作業の新しい枠組み”, コンピュータソフトウェア Vol.9 No.5 Sep.1992
- [3] 遠藤進, 布川博士, 宮崎正俊, 野口正一: “マルチフェースエージェントの試作とその応用”, 情報処理学会第 46 回全国大会
- [4] 三石大, 布川博士, 宮崎正俊, 野口正一: “分散環境のための言語系 DeLis”, 情報処理学会研究報告, プログラミング言語・基礎・実践一, Vol.93, No.11, pp.57-64