プログラミング 8－16
（1996. 8. 29）

ネットワーク環境における マルチエージェントシステム記述用言語

浦田 泰裕†, 田村 直之‡, 金田 悠紀夫‡

†神戸大学大学院 自然科学研究科, ‡神戸大学 工学部

Email:{urata,tamura,kaneda}@timpani.seg.kobe-u.ac.jp

本稿では，開放的なネットワーク上の複数のユーザによるグループウェアとしての利用を目的としたマルチエージェントシステム記述用言語を提案する．ユーザはネットワーク上のホストを陽に指定することによって対象とするホスト上にエージェントを生成したり，対象とするホスト上のエージェントと通信することが可能となる．本研究で提案した言語をTCP/IPプロトコルを用いてネットワーク上のワークステーションで実装する．また，会合スケジューリングの記述について検討する．

Multi-Agent Programming Language on a Workstation Network

Yasuhiro URATA†, Naoyuki TAMURA‡, Yukio KANEDA‡

†The Graduate School of Science and Technology, Kobe University,
‡Faculty of Engineering, Kobe University

Email:{urata,tamura,kaneda}@timpani.seg.kobe-u.ac.jp

In this paper we propose a programming language for developing multi-agent systems to be used as a group-ware by multi-users on an open network. By appointing a host on a network clearly, users can create some agents in the host, and commmunicate with those agents. The environment proposed in this paper is built on a workstation network by using TCP/IP protocol. Morever, in this research we consider programming a meeting scheduling.

1 はじめに

近年のソフトウェアの進歩の流れの中に，ソフトウェアのモジュール化，再利用を目的としたオブジェクト指向がある．その次に来るものとして，人工知能を中心とした分野で，ソフトウェア間の相互作用によってより高度な目的を達成するマルチエージェント（エージェント指向）システムが期待されている[1, 2, 3].

現在，様々なマルチエージェントシステム記述用言語が提案されており[4, 5, 6]，最近ではエージェントの協調プロトコル自体を記述するもの[7]，タプルの相互作用によってエージェントに値を伝搬する規則を記述するもの[8]がある．

一方，導出原理の発見により生み出されたProlog は推論データベース，エキスパートシステム等知識ベースシステムの実装に広く使われてきた．その Prolog を基としたマルチエージェントシステム記述用言語としては筆者らが提案する言語[9, 10]の他に IC-Prolog[11],April[12]等がある．

また，インターネットの普及により，世界中のコンピュータが接続され通信が可能となっている．そのネットワークを活かし，地理的に離れたものどうしがコミュニケーションをとる電子会合支援[13]等のグループウェアが開発されている．

本論文では，開放的なネットワーク上の複数のユーザから利用できるマルチエージェントシステムを記述する言語を提案する．さらにTCP/IPプロトコルを用いた実装について述べ，一例として会合スケジューリングの記述について検討する．

2 マルチエージェントシステム記述用言語

本章では従来提案してきたマルチエージェントシステム記述用言語[9, 10]について述べる．本言語は Prolog を基に拡張しているので Prolog の特徴であるバックトラック，ユニフィケーション，プログラムとデータとの区別がないことを利用できる．また，エージェントをグループ化し，共有の知識ベースを付与する機能を持った“フィールド”と呼ばれる機構を持つ．さらにエージェントには，固有の名前を持ちシステム内に常駐するスタティックエージェント，動的に生成されるダイナミックエージェントがある．本言語では，エージェントの生成，制御，フィールドの生成，制御，エージェント間の通信機能，ユーザインタフェースを提供している．

2.1 エージェントの記述と生成

エージェントはクラスを宣言しそのインスタンスとして生成する．agent_class がクラス名である．エージェントが生成された時に main/1 の手続きが実行される．

```
def_agent agent_class
   main(Self):- /* 初期手続き */
      /* その他の手続き */
enddef.
```

start_up エージェントの組み込み述語 s_create/2 によってスタティックエージェントが生成され，固有の名前を持つ．agent_name は文字列である．エージェントの組み込み述語 create/2 によってダイナミックエージェントが生成され，システム内で固有の ID を持つ．ダイナミックエージェントは destroy/1 で消去される．

```
s_create(agent_class, agent_name)
create(agent_class, Agent_ID)
destroy(Agent_ID)
```

2.2 フィールドの記述と生成

フィールドもエージェントと同様にクラスを宣言しそのインスタンスとして生成する．

```
def_field field_class
   /*共有知識（Prologプログラム）*/
enddef.
```

スタティックフィールド，ダイナミックフィールド はエージェントの場合と同様に以下の組み込み述語で生成，消去される．

```
s_create_field(field_class, field_name)
create_field(field_class, Field_ID)
destroy_field(Field_ID)
```

2.3 フィールドに対する操作

enter_field/1,exit_field/1 でエージェントのフィールドへの出入りを行ない，share/3 で共有知識を操作する．

```
enter_field(Field)
exit_field(Field)
share(Field, AssertList, RetractList)
```

2.4 通信

send/2 は送信を非同期的に行なう．To がエージェントであれば 1 対 1 通信，フィールドであればフィールドに所属するエージェントすべてに送信される．Message は Prolog の節で表現されている．receive/2 は From,Message ともにユニファイ可能ならば成功するが，ユニファイ可能なメッセージが届いていなければ失敗する．wait/2 は From,Message ともにユニファイ可能なメッセージが届くまで待つ．

```
send(To, Message)
receive(From, Message)
wait(From, Message)
```

2.5 ユーザインタフェース

各エージェントは自身を生成したユーザとのインタフェースを持つ．u_write/1 でユーザに対して出力し，u_read/1 でユーザからの入力を受け付ける．

```
u_write(Message)
u_read(Message)
```

この 2 つの組み込み述語によって，エージェントは実行途中に，ユーザに対して動作の確認，助言を受けることができ，プログラムの柔軟性が増すと考えられる．

3 グループウェアとしての利用環境

従来の本言語[9, 10]は一人のユーザが各エージェントの行動を把握した上でエージェント，フィールドを生成するものである（図 1）．

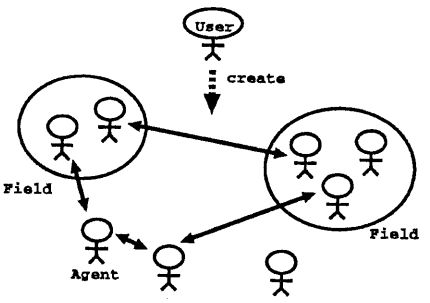

図 1: 従来の利用環境（全エージェント，フィールドを一人のユーザが管理）

よって，複数のユーザのエージェントが協調する場合には各ユーザがエージェントを記述したプログラムを一人の管理者がまとめて実行しなければならない．また，エージェントを集中管理しているので閉鎖的なネットワーク内でのみ通信が可能である．

本研究ではグループウェアとしての利用が可能なマルチエージェントシステム記述用言語を提案する．そのためには次のことが必要である．

- 複数のユーザからの利用
- 開放的なネットワーク上での実現

複数のユーザが任意にエージェント，フィールドを生成すると名前の衝突が起こる可能性がある．名前の衝突を避けるためにはまとめて管理すればよいが開放的なネットワーク環境では一元的に管理することは不可能である．そこで，ホスト名は衝突することはないので，名前の管理をネットワーク上のホストごとに行なうと管理が容易である．

ユーザはネットワーク上のホストの存在を意識し，どのホスト上のエージェント（フィールド）かを指定することによって対象とするホスト上のエージェント（フィールド）との通信が可能となり，ユーザはホスト名さえわかればネットワーク上の対象とするホストで協調するエージェントを生成できる（図 2）．

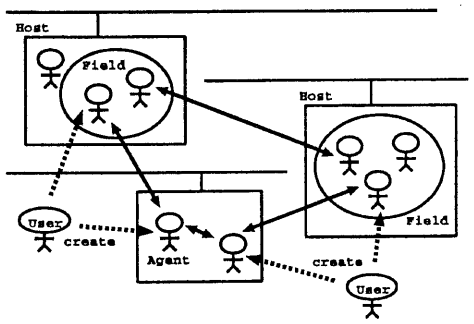

図 2: 本研究で実現する利用環境

具体的には，各ユーザはネットワーク上でスタティックエージェントを介して通信する．

```
send(agent_name@host_name, Message)
```

スタティックエージェントの名前の後ろにそのエージェントの存在するホスト名を指定することによって他のホストのスタティックエージェントと通信する．

そのため，言語としてはいくつかの制約を与えなければならない．

- ホスト名を陽に指定しなければならない．
- ホスト外のダイナミックエージェント（フィールド）と通信できない．
- 別のホストのフィールドに入ることはできない．

しかし，複数のユーザが開放的なネットワーク上でマルチエージェントシステムを構築できるので会合スケジューリングなどを目的とするグループウェアとしての利用が可能になる．

4 実装

従来，プロセス間通信を実現する手段としてPVM(Parallel Virtual Machines)[14]を用いてきた．PVM にはネットワーク上の複数のマシンを仮想的に 1 つのマシンとして利用できるために，通信が抽象化されている，デバッグが容易であるなどの利点がある．しかし，あらかじめ宣言されたホスト間の通信のみ可能なので，閉鎖的環境でしか利用できない．本論文では，TCP/IP プロトコル[15]を用いてプロセス間通信を実現する．TCP/IP プロトコルでは，ホスト名，ポート番号を指定することで通信を行なうので，開放的なネットワーク環境においての通信が可能になる．

本研究で提案する言語の利用環境をエージェントサーバ，エージェントクライアントによって構築する．エージェントサーバはエージェントクライアントからの要求を受けそのホスト上でエージェント（フィールド）を生成する．つまり，エージェント（フィールド）を生成するためにはエージェントサーバが起動されていなければならない．ユーザは任意のホストからエージェントクライアントを起動して指定したホスト上でエージェント（フィールド）を生成する．

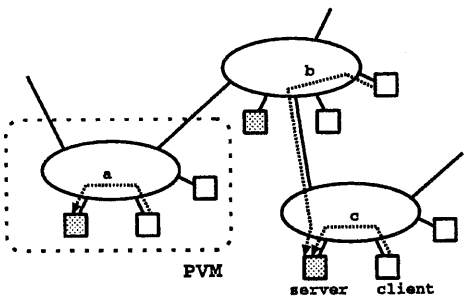

図 3: エージェントサーバ，クライアント

TCP/IP プロトコルを用いることにより，サーバのホスト名さえ知っていれば，そのサーバで行なわれる協調に参加することができる（図 3.b）．また，似たような仕事を行なうサーバが複数ある時は，近くのサーバを選んでプログラミングすると通信コストを削減することができる（図 3.a,c）．

4.1 エージェントサーバ

エージェントサーバはホスト名さえ指定すれば接続できる．つまり，接続方法は WWW サーバと同じである．その役割としては次のものが挙げられる．

- サーバ内のスタティックエージェント（フィールド）の名前を管理する．
- ホスト内のフィールドの共有知識を管理する．

- サーバ内外のエージェントからのメッセージを仲介する.

4.2 エージェントクライアント

エージェントクライアントの役割としては次のものが挙げられる.

- エージェントの定義ファイルをエージェントサーバに送信する.
- エージェントからユーザに対する入出力を受け付ける.

各ホスト上で以下のようにしてホスト名（またはIPアドレス）と定義ファイルを指定してエージェントサーバへの接続を行なう（図4）.

```
% agent_cl timpani.seg.kobe-u.ac.jp agent_2
```

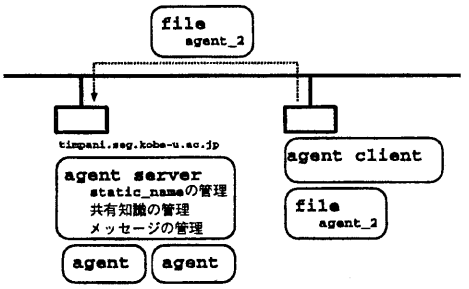

図4: エージェントサーバ，クライアントの実装

5 会合スケジューリング

本章では会合を行ないたい場合に全員の都合のよい日，時間をエージェントに決めさせる会合スケジューリングの記述について検討する.

以下の手順で会合スケジューリングを行なう.

(1) 会合の主催者は参加候補者にメールで会合の開催を知らせる.

(2) 参加者は各自のスケジュールを記述したエージェントをサーバで生成する.

(3) 各エージェントが協調して，全員に都合のよい日，時間を決定する. 都合のよい日，時間がなければエージェントは各ユーザに妥協案を求める.

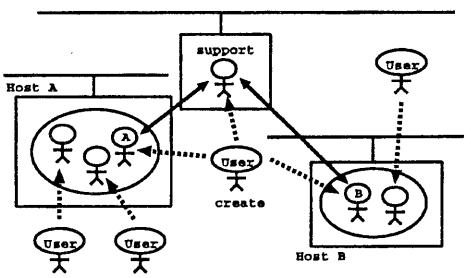

図5: 会合スケジューリング

各ユーザは参加する会合のスケジューリングを行なうホスト上で，それぞれエージェントを生成する. エージェントAはホストA上で共有知識を用いてスケジューリングを行なう. このユーザはホストB上のエージェントBで別の会合のスケジューリングも行なうので，ユーザのスケジュールを管理するサポートエージェントと通信を行ない，スケジュールの重複を防ぐ.

```
% (1)共有知識の検索,(2)他のエージェントと交渉,
% (3)自分のエージェント,ユーザと交渉
negotiate :- date(Days),search(Days),zero(M,D,X),
             not(X is -1),decide(M,D).
negotiate :- please_bads(7,22).
negotiate :- please_us(7,21).

% 他のエージェントに確認し,サポートエージェントに
% 連絡する
decide(M,D) :- send(kenkyu1, ok(M,D)),
               get_agents(kenkyu1,Agents),
               count(Agents,A),waits(A),
               send(support@timpani.seg.kobe-u.ac.jp,
                    ok(M,D)).

% 自分の都合の良い日に都合の悪い人が一人だけなら交渉
please_bads(Month,Date) :- bad(Month,Date,List),
                           count(List,X),
                           X is 1,ask(List,X).

ask([Head|List],1) :- send(Head,please(7,22)),
                      wait(Head,Msg),call(Msg).

% 自分だけが都合の悪い日があれば,
% サポートエージェント,ユーザと交渉する
please_us(Month,Date) :- bad(Month,Date,List),
                         count(List,X),X is 1,
                         compromise(Month,Date).

compromise(M,D) :- send(support@timpani.seg.
                        kobe-u.ac.jp,please(M,D)),
                   wait(support@timpani.seg.
                        kobe-u.ac.jp,Msg),
                   call(Msg),...,negotiate.
compromise(M,D) :- u_write(please(M,D),u_read(Ans),
                   call(Ans),...,negotiate.
```

図6: スケジューリングエージェントの記述例

図6に本言語によるスケジューリングエージェントの記述例の一部を示す．図6では次のような方針で協調するエージェントを記述している．negotiate/0 の第一候補では，フィールド内の共有知識を検索し都合の悪い人がいなくて最も都合の良い人が多い日を選び，他のエージェントに確認しサポートエージェントに連絡する．negotiate/0 の第二候補では，自分の都合の良い日に都合の悪い人が一人だけならそのエージェントに交渉する．negotiate/0 の第三候補では，自分だけが都合の悪い日があれば，サポートエージェント，ユーザの順に妥協案を求める．

6 まとめ

2章では本マルチエージェントシステム記述用言語におけるエージェント，フィールドの宣言方法，組み込み述語について述べた．3章では本研究で提案するグループウェアとして利用できる言語について述べた．4章ではTCP/IPプロトコルを用いて実現したエージェントサーバ，エージェントクライアントによって構築されるオープンな利用環境について述べた．5章では本言語を用いた一例として会合スケジューリングについて検討し，本研究の特徴である複数のユーザによる開放的なネットワーク上でマルチエージェントシステムを構築する有効性を示した．

今後の課題としては次のものが挙げられる．

- ユーザがどのホストにどのようなエージェントが存在するのかという情報を獲得する方法．
- 逆に，エージェントの存在するホストを意識することなくネットワーク上で通信を行なう方法．
- 他のホストのフィールドに入りたい場合のエージェントのマイグレーション．
- エージェント，フィールドはクラスを宣言して，インスタンスとして生成しているが，クラスの宣言に具体的な名前を含んでいる．具体的な名前を生成時に決定する方法．
- 目的に適した言語でプログラムを記述，他言語で記述されたプログラムを再利用するための他言語インタフェース．

参考文献

1) 所真理雄：オブジェクトの社会（上）（下）．bit,Vol.26,No.10,11(1994).
2) 木下哲男，菅原研次：エージェント指向コンピューティング～エージェントの基礎と応用～．株式会社ソフト・リサーチ・センター(1995).
3) 西田豊明：ソフトウェアエージェントとその周辺．電子情報通信学会誌，Vol.78，No.11,pp.1252–1259(1995).
4) Mark C.Torrance: The AGENT0 Manual(1991).
5) Michael Kolb: A Cooperation Language. Proceedings First International Conference on Multi-Agent Systems ICMAS'95,pp.233–238(1995).
6) 来間啓伸，大須賀昭彦，本位田真一：協調アーキテクチャに基づくソフトウェア・モジュールの仕様記述モデル．情報処理学会論文誌，Vol.37,No.6,pp.1171–1186(1996).
7) 桑原和宏，篠原拓嗣，大里延康，石田亨：協調プロトコル記述言語 AgenTalk の実現．電子情報通信学会 信学技報 AI95-18,pp.33–40(1995).
8) 渡辺慎哉，赤間清，宮本衛市：Generic な相互作用を有する並行計算モデル-GIM．コンピュータソフトウェア，Vol.12,No.6,pp.67–76(1995).
9) 川村尚生，斎藤善徳，金田悠紀夫：エージェント間の共有知識を実現した Prolog に基づく協調処理言語，情報処理学会研究報告 92-PRG-8,pp.59–65(1992).
10) 浦田泰裕，齋田明生，田村直之，金田悠紀夫，川村尚生：分散環境下におけるマルチエージェントシステム記述用言語．情報処理学会研究報告 95-PRO-2,pp.153–159(1995).
11) Damian Chu: I.C.PrologII:a Language for Implementing Multi-Agent Systems(1993).
12) 高田 裕志，Francis G.McCabe，和田 裕二：マルチエージェント指向プログラミング言語 April．情報処理学会全国大会第５１回,(1995).
13) 八槙 博史，石田亨：エージェントネットワーク Socia による電子会合支援．情報処理学会全国大会第５０回 d,(1995).
14) Geist, A., Beguelin, A., Dongarra, J., Jiang, Weicheng., Manchek, R., Sunderam, V.: PVM3 USER'S GUIDE AND REFERENCE MANUAL. (1993).（邦訳：「PVM3 ユーザーズガイド & リファレンスマニュアル」，村田英明）．
15) Douglas E. Comer, David L. Stevens: Internetworking With TCP/IP Vol III: Client-Server Programming And Applications. Prentice-Hall Inc.,(1993)（邦訳：「TCP/IP によるネットワーク構築 Vol.III ～クライアント-サーバプログラミングとアプリケーション」，村井純，楠本博之，共立出版株式会社）．