プログラミング 8－17
（1996. 8. 29）

Javaを用いた異種エージェント間での協調支援エージェントの開発に関する研究

齋田 明生†, 田村 直之‡, 金田 悠紀夫‡

†神戸大学自然科学研究科, ‡神戸大学工学部

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

本稿では，ネットワーク上に散在する種々のエージェント間で，ユーザーからの要求に応じて，メッセージのやり取りを行えるコミュニケーションエージェントのモデルの提案を行う．本モデルでは，ユーザーからの要求をProlog言語を用いて記述することで，ユーザーは多様なエージェントに対し要求を統一的に表現できる．本モデルの応用としては，分散データベース等の分野が考えられる．

現在，本研究のコミュニケーションエージェントのプロトタイプが，Java及びHORBを用いてネットワーク上のワークステーションで実装されている．

The development of Communication-agent among heterogeneous agents on a network by using Java

Akio SAITA†, Naoyuki TAMURA‡, Yukio KANEDA‡

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

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

In this paper we propose a model of Communication-agent which can interact many kinds of message according to user's requirements among heterogeneous agents on a network. Using Prolog in this model, user can express totally his or her requirement to various agents. As an application of this model, we consider a distributed-database.

The prototype of this Communication-agent has been implemented by using Java and HORB.

1 はじめに

現在，WWWを中心としてインターネットが急速に発展して来ている．このインターネット上には，様々なエージェントやオブジェクトが散在し，ユーザーへのサービスを個々に提供している．このような大規模なネットワークは，統一性のある体系のとれたものとは考え難く，そうした中でユーザーの要求を処理するのは非常に困難である．そこで，一つの体系に沿った整合性のとれたネットワーク上で，ユーザーの要求を処理するシステムが広く考えられている[1, 2].

そこで問題となるのは，ユーザーが各サービスの提供される場所（ホスト名）とその内容を把握していないと，そのサービスが受けられないという点である．また，そのサービスを提供する側の主体となるものも，あくまでオブジェクトに過ぎないものから，自律的に処理を行うエージェントまで色々なものが考えられる[3].

そのため，それらエージェントおよびオブジェクトに対して，ユーザーからの複雑な要求を処理する手法として，エージェントのマイグレーション[4]や，RPC (Remote Procedure Call)[5]等の手法が考えられている．しかし，これらのシステムでは，ユーザーの要求を簡単に表現するのが難しい．

そこで本研究では，ネットワーク内で統一的なメッセージ処理を行い，ユーザーがデーターベースサーバーや，C言語のプログラムであるといった様々な異種エージェントに対し，要求を統一的に表現できるようなシステムの実現を目指し，その実現のための，コミュニケーションエージェントのモデルの提案を行う．また，このコミュニケーションエージェントをJava[6]を用いて実現することで，マシンニュートラルという性質をもつエージェントが実現できる．

2 コミュニケーションエージェント

本研究の目指すシステムの実現のためには，ネットワーク上でユーザーとエージェント間，種々のエージェント間でメッセージを送る仲介者としてのコミュニケーションエージェントの実現が必要になる．

本研究では，自律的なエージェントよりもサーバー型の受動的なエージェント，またはオブジェクトをコミュニケーションの主体として考えているので，処理の依頼というものをコミュニケーションエージェントの主な仕事と考える．

コミュニケーションエージェントに必要な機能として次の二つが挙げられる．

- 必要なエージェントの検索
- 処理の依頼

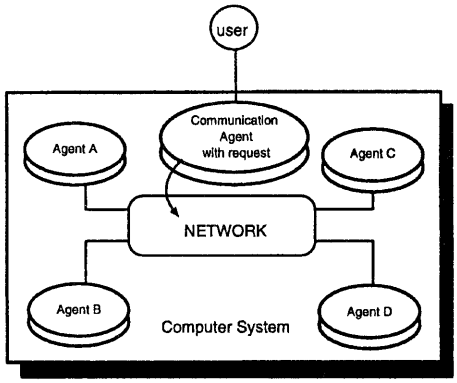

図1: 本研究の目指すシステム

3 必要なエージェントの検索

本章では，処理を依頼するためにまず重要な「必要な相手(エージェント)を見付けてその相手との交信を行う.」ということを実現させる方法について，考察を行う．

必要な相手を見つけるためには，まずその相手の能力を知った上で，妥当な相手に処理を依頼するか，そうでなければ，手あたり次第に処理を依頼し，その結果を待って望ましい結果を期待するかのどちらかになる．

前者では，相手の能力の客観的判断，後者では，結果が妥当かどうかの客観的判断という客観性が必要になる．この客観性は現状ではどうしてもユーザーに依存することになる．

そこで本研究では，前者の考え方から次節で示す能力の表現形式を用い，各エージェントの能力をユーザーが前もって評価しておくことにする．

3.1 エージェントもつ能力の表現形式

ユーザーがエージェントの能力を表現するとなると，その形式を統一しておくことが必要になる．そこで，本研究では次のようなPrologの宣言節の表現形式を採用する．

```
add(X,Y,Sum).
book(Name,Author,Price,Year).
```

この場合，それぞれの行が，エージェントの一つの能力を表している．更にこの表現だけで足りない部分はcommentとして記すことにする．ただし，次の図2のAgent Bのように本と書店のデータベースというような複数の能力を持つエージェントも許される．

この表現形式の利点として次のようなものが挙げられる．

(1) オブジェクトに対しては，表現形式そのものがインスタンス生成メッセージと一致する．
(2) 論理型言語であるPrologは，データベースに対しても親和性がある．
(3) ユーザーからの複雑な要求にも対応できる．

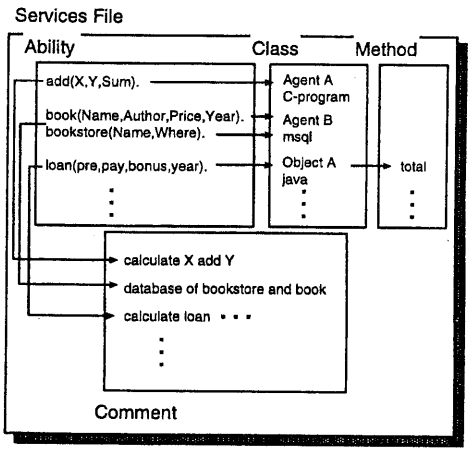

図 2: servicesファイルの中身

本モデルでは，ローカルなエージェントの能力を表すファイルとしてservicesファイルというものを考えており，このファイルに能力を表現したものが記される．また，Prolog節形式の能力表現以外にも以下の三つが記される．

- エージェントの能力をcommentとしてより詳しく記述しておく
- C言語のプログラム，msqlといったエージェントの種類
- オブジェクトに関しては，そのメソッド

この三つを合わせて記すことで，コミュニケーションエージェントが，相手に見合う形式のメッセージを送ることが可能になる．

3.2 検索の手法

前述のservicesファイルを用いて，本研究では，以下のような手法で必要なエージェントを見つけ出すことにする．

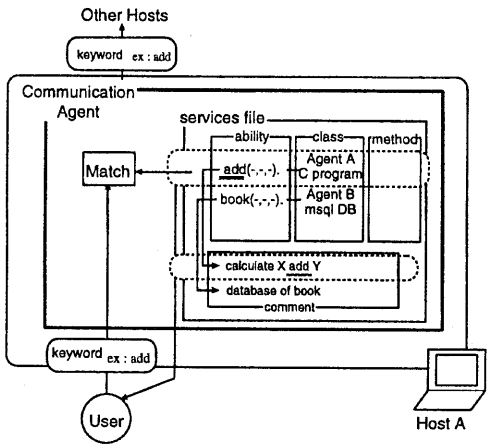

図 3: 検索の手法

まずユーザーからキーワードだけを入力してもらい，それに該当するエージェントを見つけるためにコミュニケーションエージェントがservicesファイルを検索する．具体的には，そのキーワードに該当する関数子を持つ節，または，そのキーワードを含んでいるcommentがファイルにあるかどうかを検索する．また，それと同時に他のホスト上のコミュニケーションエージェントに対して同様の検索を依頼する．

その結果，該当するエージェントが見つかった時，そのエージェントに関するservicesファイルに記された全情報をユーザーに提示する．ユーザーは，その結果から自身の要求をProlog節で記述し，コミュニケーションエージェント

に渡す.

4 処理の依頼

本研究では，静的なサーバー型のエージェントおよびオブジェクトをコミュニケーションの対象と考えている．そのためコミュニケーションの中身としては処理を依頼し，その返答を待つというのが主となる．また，全てのメッセージは，コミュニケーションエージェントを介して行われる．

具体的には，各コミュニケーションエージェント間では，Prologの節形式のメッセージが送られる．そして，ローカルなコミュニケーションエージェントとエージェント間では，そのエージェントに見合う形式のメッセージが送られる．

その形態としては次の三つが考えられる．

(1) 静的なエージェントに対しては，メッセージを送り，その結果を待つ
(2) オブジェクトに対しては，クラス定義として存在していると考えられるため，そのクラスに対し，インスタンスの生成を依頼し，その後，そのインスタンスに対してメソッドの呼び出しという形でメッセージを送り，その返事を受け取る
(3) オブジェクトの再利用[7]という考えから，オブジェクトのクラスをユーザーの利用するマシン上に取って来て，そのクラスに対し(2)と同じ処理を行う

この三つの形態に対して，本研究ではそれぞれ次のようなメッセージを用いる．ここでは，コミュニケーションエージェントをC-Agentと記す．

(1) C-Agent → Agent **ask(Prolog節).**
Agent → C-Agent **ans(Answer).**
Answerには，処理が成功すればユニフィケーションされたProlog節，失敗すればnoがはいる
(2) C-Agent → Object **instask(Prolog節).**
Object → C-Agent
1と同様の返事がなされる
(3) C-Agent → Object **getobj(Prolog節).**
Object → C-Agent **result(Result).**
Resultには，オブジェクトの移動に成功したらyesを，失敗であればnoがはいる

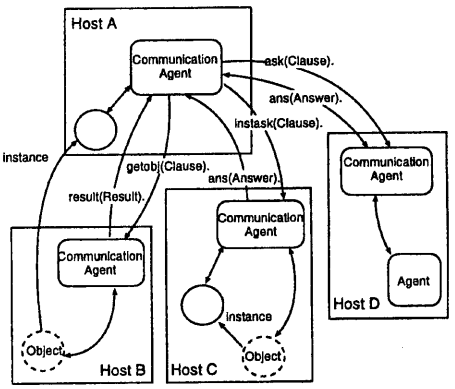

図4: 処理の依頼

具体的な，ローカルなメッセージ交換のためには，Prologの節をデータベース用に変換したり，Cの関数やJavaのオブジェクト用に変換したりする機能が必要になる．

4.1 メッセージ変換

ローカルなメッセージ交換のためには，ユーザーからの要求であるProlog節を，各エージェントに見合う形式に変換する必要がある．その変換形式は，エージェントの種類が分かれば，一意に定まると考えられる．よって，各コミュニケーションエージェントが，本システムで用いるエージェントの種類にあった変換のテンプレートを持っていれば良いことになる．

5 コミュニケーションエージェントのモデルと実装

本研究で考えるコミュニケーションエージェントは次の図5のようになる．このモデルをJavaを用いて実現することで，マシンニュートラルな性質を持つコミュニケーションエージェントの実現ができる．そのため，このコミュニケーションエージェントをインストールする手間が省かれる．また，ユーザーインタフェースという点でもJava言語の持つクラスライブラリを有効に活用できる．

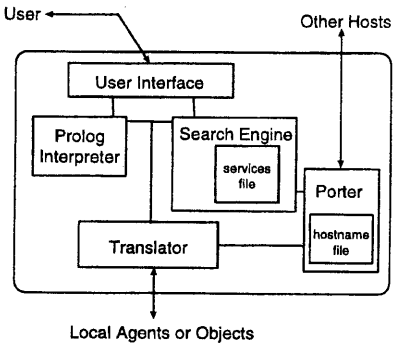

図 5: コミュニケーションエージェントのモデル

それぞれのモジュールの説明を行う.

(1) User Interface
最初のユーザーからのキーワード検索を受け付け, Search Engine に検索を依頼する. 結果をユーザーに提示し, その後のユーザーからの要求を Prolog Interpreter に渡す.

(2) Prolog Interpreter
Prolog で表現されたユーザーからの要求を解釈し, 各モジュールにタスクを与える.

(3) Search Engine
キーワードからローカルな services ファイルを検索し, マッチするエージェントが同一ホスト上に存在するかどうかを調べる. また同時に, Porter を介し他のホストの Search Engine に検索を依頼する. この検索結果は, User Interface に返されユーザーに提示されると同時に, Prolog Interpreter に渡され, その後の処理に使われる.

(4) Translator
Prolog Interpreter から送られて来たユーザーからの要求を示す Prolog の節を, 各エージェントに見合う形式に変換する. その変換方法は, エージェントの種類によって決められている. またローカルなエージェント宛でないメッセージは, 無変換のままその節を Porter に送る.

(5) Porter
このモジュールを介して, 他のホスト上のコミュニケーションエージェントの Porter とメッセージのやり取りを行う. 他のホスト上のエージェントの検索や他のホスト上のエージェントに処理を依頼するのに必要となる.

本モデルを, Java[6]および HORB[8, 9]を用いて, Sun のワークステーションをつないだネットワーク上で実現する.

6 具体例 (車の購入)

本システムの具体的な例として, ユーザーからの次のような要求を考える.

> ユーザーは, 車(Car,Cardealer)を購入しようと考えている. その支払い方として次の方法を考えている.
>
> 現在銀行(A-Bank)に預けているお金で前金を払い 3 年間月々2 万円, ボーナス月 20 万円のローン(loan)を組む.
>
> この条件で, 買うことのできる車を探したい.

本システムでは, まずユーザーは, 括弧内の英単語をキーワードとしてコミュニケーションエージェントに検索を依頼する. その結果から, この要求を次のように表現することができる.

```
?- a-bank(user,Cash),
   loan(Total,Cash,20000,200000,3),
   cardealer(Car_name,Car_data,
             Cost),Cost <= Total.
```

この処理は次の図のようになる.

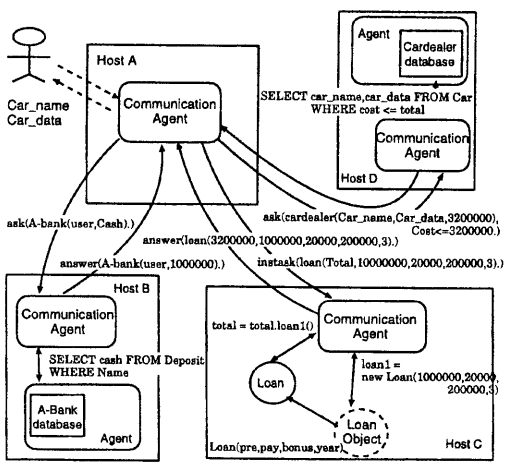

図 6: 処理の流れ

(1) まず銀行のデータベースに対し，ユーザーの預金額を問い合わせる．
(2) ローンを計算するオブジェクトに対して，(1)の結果を元に支払総額の算出を依頼する．
(3) (2)の結果から，カーディーラーのデータベースに対し，問い合わせを行う．

上記のような形で，ユーザーは必要な結果を得ることができる．また，それぞれの節に対し該当するエージェントが複数ある場合，つまり複数のディーラー，複数の銀行が考えられる時，本システムでは，ユーザーにとってより多くの選択肢が得られることになる．

7 今後の課題

(1) モデルの改良
現在のモデルに対し，ユーザーからの要求を完全なPrologで表現すれば，より複雑な要求に対処できる．また，それに伴いエージェント間のメッセージもより高度なものに改良する必要がある．
(2) 対応するエージェントの種類
現在本システムが対応できるエージェントの種類は，Java言語のオブジェクト，C言語プログラム，またデータベースとしてmsqlサーバーおよび，Javaのオブジェクトに対応している．
(3) 現実的な応用
現段階のシステムでは，ユーザーが要求する現実的な要求には対応できない．より現実的な応用を意識したシステムを作成し，それに基づく評価を行いたい．

8 まとめ

本研究では，特定の規約にしたがったネットワーク内で，そこに存在するエージェントやオブジェクトの間で，ユーザーからの要求に沿って処理を依頼するコミュニケーションエージェントのモデルの提案を行った．各エージェントの能力をPrologを用いて表したことで，統一的な表現で，ユーザーからの要求を処理できるモデルが提案できた．しかし，表現の統一により対象となるネットワークに種々の制限を加えなければならなくなったことは，今後考慮すべき点である[10].

また，本モデルでは各エージェントとして，静的なエージェントやオブジェクトを考えているので，本モデルの応用対象としては，分散データベース等が考えられる．ただし，今後は動的なエージェントにまで対象を広げ，より汎用性の高いモデルの考察を行いたい．

参考文献

1) 特集 オブジェクトの先を行くエージェントの時代へ NIKKEI COMPUTER, pp.48-62,1994.5.30.
2) 武田 英明, 飯野 健二, 西田 豊明, 知識コミュニティにおける仲介機能 MACC,93,pp.49-57,1993.
3) Micheal Luck, Mark d'Inverno, *Agency and Autonomy: A Formal Framework* London.
4) Danny B.Lange, *Agent Transfer Protocol ATP/0.1 Draft 3* IBM Research, Tokyo Research Laboratory,June 18,1996.
5) Matthew J.Zelesko and David R. Cheriton, *Specializing Object-Oriented RPC for Functionality and Performance* Proceedings of the 16th ICDCS, pp.175-187,1996.
6) Sun Mycrosystems Inc. *Java Tutorial* Online-Manual,http://Java.sun.com/doc/tutorial.html
7) 田原 康之, 粂野 文洋, 本位田 真一, 開放型ネットワークにおける協調によるフリーソフトの再利用 bit別冊, pp.9-20,1996.
8) 平野 聡, 新しいネットワークプログラミングパラダイム *HORB* つくばソフトウェアシンポジウム '96, pp.87-90,1996.
9) Hirano Satoshi, *The Magic Carpet for Network Computing:HORB Flyer's Guide* OnlineManual,http://ring.etl.go.jp/openlab/horb/doc/guide/guide.htm
10) 村上国男, マルチエージェントシステムとその応用 電子情報通信学会誌 vol.78 No.6,pp.570-577,June,1995.