データベースシステム 116－50
（１９９８．７．９）

アクティブ・メディエーション・システムにおける エージェント通信

中田 優作，木内 光，小西 修
高知大学理学部情報科学科
780 高知県高知市曙町 2-5-1
{yunakata, hikiuchi, konishi}@is.kochi-u.ac.jp

概要

我々は，ネットワーク分散環境（インターネット，CORBA など）における様々な情報源にアクセスし，その情報を処理するために，エージェント技術と能動データベース技術を統合した，アクティブ・メディエーション・システム HI-AMS を開発している．WWW 上のダイナミックに変化し発展する情報源の世界には，従来のスキーマ統合に基づくマルチデータベース技術の適用は難しい．また，最近のキーワードサーチエンジンによる Web 技術では，高度な質問処理は実現できない．本稿では，オープンな環境における情報源とユーザアプリケーションの仲介を行う HI-AMS が，どのように，協調エージェント，能動データベース，メディエータ，そして，ORB などの新しい技術を統合するかを示す．

キーワード： メディエータ，協調情報エージェント，エージェント通信，ECA ルール，能動データベース，義務論理

Agents Communication in the Active Mediation System: HI-AMS

Yusaku NAKATA, Hikaru KIUCHI, Osamu KONISHI
Dept.of Information Science,Faculty of Science,Kochi University
2-5-1 Akebono-cho Kochi 780 Japan
{yunakata, hikiuchi, konishi}@is.kochi-u.ac.jp

Abstract

We describe the architecture, design, and implementation of an active mediation system(HI-AMS) which integrates agent technologies and active database. The role of HI-AMS is the integration of schema or integration of information for heterogeneous information resources including semi-structured data, and madiating between an enduser and information resources.In internetworking and the World Wide Web environments, the multi-database based on conventional schema integration do not cope well with this semi-structured data and dynamic changing environment. On the other hand, recent Web technologies based on keyword search engines are incapable of processing high query. In this paper, we show how HI-AMS integrates new technological developmentssuch as multi-agents, active database, mediator, and object request broker as a mediation system in a dynamic and open environment.

Key words: mediator, cooperative information agents, communication between agents, ECA rules, active database, deontic logic

1 はじめに

近年の急速なインターネットの普及により，ネットワーク分散環境に様々な情報源が存在するという新しい環境が実現された．これら異種分散情報源は半構造データなども含んでおり，スキーマやデータ定義が不明瞭である．このため，利用者が自力で目的の情報を探し得ることは非常に困難なことである．メディエーション・システムとは，半構造データを含む異種情報源を統合し利用者の質問に最適な結果を返す自律的なシステムである．メディエーション・システムには，アプリケーションの情報源からの独立や最適な結果を返すなどの知的処理といった柔軟なアーキテクチャ，オントロジー，スキーマ統合・データ統合，ラッパー，データ解析・評価，ユーザインタフェースといった数多くの機能が必要となる．我々は，この問題を解決する一つの方法として，エージェント技術とアクティブ・データベース技術を統合した，アクティブ・メディエーション・システム HI-AMS を開発している．[2]HI-AMS は，ダイナミックでオープンな環境に対応，スケラビリティ，エージェント，そして能動データベースといった考えに基づいて開発されている．HI-AMS はユーザインタフェースエージェント，アクセスエージェント，データ統合エージェント，データ評価エージェント，そして，ソース（情報源）エージェントとこれらのエージェントの協調促進をはかるメディエータ・エージェントからなる．エージェントが互いに通信しあい，協調しながら目的をはたす．これらエージェント間の通信は，要求とその要求の採否と見ることができ，ここに，許可，禁止，義務の判断を入れることが有効である．我々は，このエージェント群の協調と判断を促進するところのメディエータ・エージェントと能動データベースからなるメディエータ機構を考える．関連研究に，エージェントベースのメディエータの研究として，"InfoSleuth"[1] があるが，次の点で HI-AMS は異なっている．

- InfoSleuth は，情報源を Web 環境としているが，HI-AMS は，さらに既存のスキーマデータや ORB 環境にも対応している．

- エージェント群の協調促進を，エージェントの状態と行動推移を ECA ルールで表現した能動データベース上で実現している．

- データ評価機能を積極的に位置づけ，自己組織化マップによる自動クラスタリング機能を有している．

2 HI-AMS のアーキテクチャ

我々の提唱するアクティブ・メディエーション・システム HI-AMS のアーキテクチャを図 1に示す．

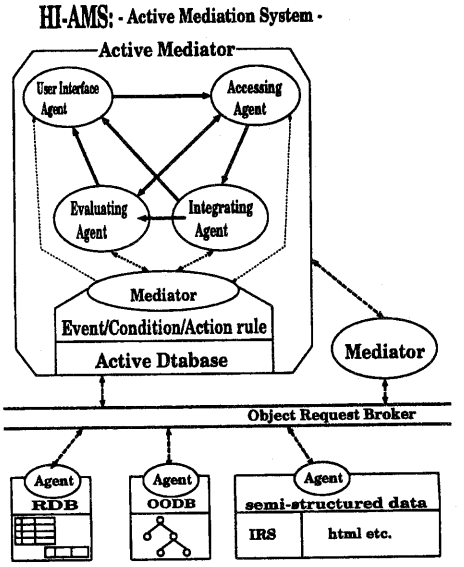

図 1: HI-AMS アーキテクチャ

HI-AMS は，アクティブデータベースをプラットホームとし，ユーザインタフェースエージェント（質問・検索法），アクセスエージェント（アクセス・集約），及び情報統合，評価エージェント（情報解析・学習・探索戦略），そして各エージェントを調停し，協調促進させるためのメディエータから構成される仲介者システムである．各エージェントはユーザからの要求に見合った情報を，様々な分散した情報源にアクセスし，収集・統合・評価するために協調して作業を進めていく．また，分散された各情報源のフロントエンドにはデータベースエージェントが存在し [4]，メディエータはこのデータベースエージェントと通信することにより，ドメインや情報源の情報を収集する．この

情報をもとに，アクセスエージェントはデータベースエージェントに適切な質問をし，情報を収集する．収集された情報は統合エージェントによって統合され，評価エージェントによって評価される．このアーキテクチャの特徴として，メディエータが情報源を選択すること，新しい情報源の開拓と登録ができること，そしてエージェントの追加というスケラビリティが挙げられる．

3 協調促進と義務論理

●エージェント間の協調

アクティブ・メディエーション・システムを構成する各エージェントは，ユーザからの質問に答えるために，互いに協調しながら作業を進める．このとき，各エージェントの調停を行い，協調を促進させる役割を担うメディエータは，各エージェントの状態を管理しなければならない．そのため，各エージェントは処理依頼を受けたり，処理を実行・終了したりと，何らかの状態変化が起こる度に，メディエータに自分の状態を告知する．これにより，各エージェントはメディエータを介して，協調相手の状態を知ることが可能となり，より豊かなエージェント間の協調が行える．具体的にエージェント群の振る舞いの流れを説明する．

(1) 情報源エージェントは，ドメイン制約も含めたドメイン情報をメディエータに通知する．

(2) ユーザエージェントからの質問が，アクセスエージェントに送られる．

(3) アクセスエージェントは，その質問をメディエータに送る．

(4) メディエータは，各ドメイン情報と質問のマッチングをとり，最適の情報源エージェントをアクセスエージェントに知らせる．

(5) アクセスエージェントは，その情報源エージェントに質問を発し，結果を受け取る．

ここに，メディエータは，各エージェントの行動（生成，起動，休止，消滅など）やエージェント間のメッセージ転送を把握して，それらの活動の促進を図る．

●義務論理

義務論理 [3] とは，様相論理の一つであり，「しなければならない（義務）」，「してはいけない（禁止）」，「しても良い（許可）」の３つの状態をモデル化したものである．

義務論理モデルの３つの状態をメディエータの各エージェントの状態を表すものとして用いる．この状態での義務論理モデルでは，エージェントの初期状態は「可能」であり，処理依頼のメッセージを受け取ると「義務」の状態に移行し，処理の主体も移される．処理が終了すると，依頼主に処理結果等のメッセージを返して「可能」状態に戻る．もし，エージェントが処理依頼のメッセージを受け，「義務」状態に移行したが，依頼された処理に失敗した場合や，一貫性制約を破る可能性が存在する場合は，「禁止」状態にして，それ以後の処理依頼の受け付けや行動を行わないようにする．問題解決後，メディエータが「禁止」状態から「可能」状態に戻し，初期状態に戻る．

これらは，エージェントの要求の間の関係を明示する．例えば，ある要求の履行は，他の履行を義務づける．あるいは，他の履行を許可または，禁止する．このような要求間の義務論理関係の表現は，要求に対する理由の階層を表現し，要求間の優先度の基本的な表現を与える．これは，HI-AMS の目的とそれを実現する機能において，ある要求を含むか除外するかに対する理由を表現することになる．

ここに，エージェントの要求間の関係は，メッセージトランザクションとして捉えられ，エージェントからの要求に対して，必ず，その対応エージェントは答えなければならないとする．例えば，エージェントAからエージェントBに要求がある場合，次のように記述される．

```
obligation (has_request(B) --->
has_response(A))
```

ここに，エージェントBの has_request (B) は，必ずAからBへの request という事実があるということが前提になり，次のように記述される．

```
[AtoB(request)] has_request(B)
```

これは，また次のようにも記述できる．

```
not has_request(B) --->
[not AtoB (request)] not has_request(B)
```

また，上記の obligation は，次のようにも表現できる．

```
not has_response(A) --->
forbid (AtoB(request) and not BtoA(response))
```

●メッセージトランザクション

2章で示したHI-AMSのシステム上のエージェント群の振る舞いは、次の2点で制約される。

(1) エージェントは、自己の外へのアクションについては、必ずメディエータに問い合わせねばならない。メディエータは、それに必ず答えなければならない。また、メディエータは問い合わせという事実がないのに、応答することは禁止される。

(2) エージェントは、メディエータから紹介された他のエージェントへのみ、要求をすることが許可される。

各エージェント間, またはエージェントとメディエータのメッセージのやりとりは, 相手のあるメッセージトランザクションと捉えることが出来る. 例えば, ユーザインタフェースエージェントが送った処理依頼メッセージを受け取ったアクセスエージェントは, メディエータに依頼された処理を行うために必要な情報源のドメイン情報などを質問する. これは以下のようなトランザクションとして表現される.

```
transaction get source data (query:q)
 agent:
   uia: UserInterfaceAgent
   aa : AccessAgent
   m  : mediator
 uia can send
   { messages:
       request(result data(q)) to aa
   }
 aa can send
   { messages:
       request(source data(q)) to m
   }
 rules
   check agent state:
   on before source data(q)
   if requested(result data(q))
   do execute
Goal = {source data(q)}
end-transaction
```

4 協調促進とECA表現

各エージェントの協調を促進するメディエータは, アクティブ・データベースのフロントエンドに存在し, 各エージェントの状態をアクティブデータベースに更新することによって管理する. この時, アクティブデータベースには, エージェントの状態を記述する次のようなテーブルが必要である.

```
AGENT
[agent, state, operation, from, to, ope_state]
```

このテーブルはエージェント agent の状態が state で, 処理 operation をエージェント from から依頼された, またはエージェント to へ依頼して, その処理の状態が ope_state であることを表現している. このエージェントの状態を管理するテーブルで, アクティブデータベースのECAルール機構を用いることにより, 義務論理における一貫性制約が表現出来る. 例えば, 次のような一連のエージェント間のやりとりについて考えてみる. アクセスエージェントはメディエータへ,「ユーザインタフェースエージェントから質問を受けた」というメッセージを伝える. メッセージを受け取ったメディエータはエージェントの状態を更新する. このとき, 次のようなルールが定義できる.

E: UPDATE TO *AccessAgent*
  WHERE *current.agent = 'aa'* **AND**
    new.operation = 'getQuery' **AND**
    new.from = 'uia'
C: if *'finish'* != **SELECT** *ope_state*
  FROM *UserIAgent*
  WHERE *agent = 'uia'* **AND**
    from = 'user'
    operation = 'getQuery'
A: UPDATE TO *AccessAgent* **SET** *state* = '禁止'
  WHERE *from = 'uia'*

つまり, アクセスエージェントがメディエータへそのようなメッセージを送ることが出来るのは, それ以前にユーザインタフェースエージェントがユーザから質問を受けているという前提があるという制約を表現している. この制約が破られた場合は, アクセスエージェントの状態を義務論理における「禁止」状態にすることによって, その後の行動を禁止しなければならない. この「『禁止』状態の時に行動を起こしてはいけない」という制約もECAルールで表現することができる. つまり, 次のようなルールが定義できる.

E: *before* **UPDATE TO** *AccessAgent*
  WHERE *current.agent = 'aa'* **AND**
    new.operation = 'sendQuery' **AND**
    new.to = 'mediator'
C: if '禁止' == **SELECT** *state*

FROM *AccessAgent*
WHERE *agent = 'aa'*
A: *abort*

これらのECAルールのトリガとなるのは，各エージェントからの状態通知を受け取って，メディエータがエージェントの状態をデータベースに更新しようとした時である．このように，アクティブ・データベースのECAルール機構を用いることによって，義務論理における制約を表現することができ，より豊かなエージェント間の協調が行える．

5 システム構成

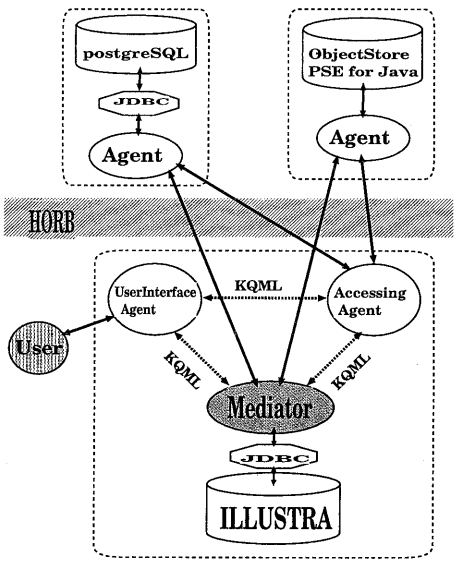

図 2: 実装モデル

今回は対象データを文献データに絞り,HI-AMSのプロトタイプシステムとして実装を行なった．メディエーションシステムにおいてエージェントの協調促進を担うアクティブデータベースとしてInfomix社のILLUASTRAを用い,外部の情報源としては異種分散情報源の統合という観点から,フリーのリレーショナルデータベースとして広く普及しているPostgreSQL 6.2.1と,Javaベースのオブジェクト指向データベースであるObjectStore PSE for Javaの二種類のデータベースを用いた．エージェントは全てJava言語によって記述し,ILLUSTRAにアクセスするメディエータ及びPostgreSQLのデータベースフロントエージェントはJDBCを介すことによってデータベースの情報を獲得することが可能である．また,ObjectStore PSEはJavaベースであるため,Java言語で記述されているエージェントは直接アクセスすることが出来る．エージェント間通信に関しては様々な通信形態を考慮に入れ,今回はメディエーションシステム内における通信にはエージェント間知識操作言語であるKQML(Knowledge Query and Manipulation Language)を用い,外部のエージェントとの通信にはHORBを用いた.KQMLソフトウェアは米国スタンフォード大学機械工学科で開発中のJava Agent Template(JAT) version0.3を利用している．各エージェントの役割は，メディエータがメディエーションシステム内のエージェントの状態を管理することによってエージェント間通信の協調促進を行ない，また，外部のデータベースフロントエージェントにHORBを介してアクセスし，そのデータベースのスキーマ情報や，あるいはデータベースに更新があったかどうかなどを通知してもらう．ユーザーインターフェースエージェントはユーザーとGUIを介してユーザーの要求を受け取り,ユーザーの要求をアクセスエージェントに送ってその回答をユーザーに伝える．アクセスエージェントは外部のデータベースフロントエージェントに問い合わせ,最終的に集められた異種データを統合してユーザーインタフェースエージェントにその結果を返してやる．

以下に，今回のプロトタイプシステムで実際に使われるエージェント間通信言語の例とその内容について説明する．

```
(ask-state :sender UIAgent
        :receiver Mediator
        :language KQML
        :ontology agent
        :content (tell :sender UIAgent
                       :receiver AccessAgent
                       :language KQML
                       :ontology agent
                       :content (author,Mike))
```

このメッセージは,ユーザーインターフェースエージェントがアクセスエージェントにメッセージを送りたい時に,アクセスエージェントがメッセージを受け取れる状態か,すなわち「可能」状態かどうかをメディエータに問い合わせるものである．送りたい内容は最

初の content 以下の部分で, ユーザーから受けた要求, すなわちここでは「著者が"Mike"である文献」があるかどうかをアクセスエージェントに依頼するものである. 仮にこのメッセージを解析したメディエータが以下のメッセージを送ったとする.

```
(tell :sender Mediator
    :receiver UIAgent
    :language KQML
    :ontology agent
    :content (yes))
```

これはアクセスエージェントの状態が「可能」状態であり, アクセスエージェントへのメッセージ送信が可能であることをメディエータがユーザーインターフェースエージェントに伝えるメッセージである. このメッセージを受け取ることによってはじめて, ユーザーインターフェースエージェントはアクセスエージェントにメッセージを送ることが出来る.

```
(ask-infosource :sender AccessAgent
           :receiver Mediator
           :language KQML
           :ontology agent
           :content (author,Mike))
```

このメッセージはアクセスエージェントがユーザーインターフェースエージェントから受け取った内容, つまり著者が Mike であるような文献を収集するためにはどこへアクセスしたらよいかをメディエータに問い合わせするものである. このメッセージに対するメディエータの回答は例えば以下のようなメッセージになる.

```
(tell-infosource :sender Mediator
        :receiver AccessAgent
        :language KQML
        :ontology agent
        :content (apus.is.kochi-u.ac.jp,Agent))
```

回答には, アクセスする場所と問い合わせをするデータベースフロントエージェントの名前が書かれている. アクセスエージェントが収集してきた異種データはメディエータスキーマに統合される. 以下にメディエータスキーマの例を示す.

```
interface Document (extent Document){
  attribute String docNo;
  attribute String title;
  attribute Date date;
  attribute String abstract;
  void updateDocNo(in String no);
  String selDocNo(in String n, out String docNo);
  String selTitle(in String t, out String title);
  String selAuthor(in String a, out String author);
  rule cancelUpdate :
    on after updateDocNo(...), create(...)
    if exists d in Document :
      (d != self) and (d.docNo = self.docNo)
    do abort;
};
```

6 おわりに

本稿では, アクティブ・メディエーション・システム HI-AMS が協調エージェントや能動データベース, メディエータを統合する方法について紹介し, エージェント通信に義務論理を導入し, ECA ルール機構によって制約を表現することによって, エージェントの協調促進を行えることを示した. また, 実際にプロトタイプを開発し, 実験することによって, 異種分散情報源からの情報統合が行われることを確認した. 今後, 情報源のフロントエンドに存在するソースエージェントとメディエーション・システムの間での義務論理の導入を検討する.

参考文献

[1] Bayardo Jr. R.J. and et al., " InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments, " SIGMOD'97, pp.195-206, May 1997.

[2] 小西 修, "異種情報源統合のためのアクティブメディエーション・システム-HI-AMS:High Intelligent - Active Mediation System-",Mem.Fac.Sci.Kochi Univ.(Inform.Sci), 17, March 1997.

[3] H.Weigrand , et.al., "Interoperable Transactions in Business Models - A Structured Approach", Advanced Information Systems Engineering, LNCS 1080, pp. 193-209, 1996.

[4] Pitoura, E. and Bhargava, B., "A Framework for Providing Consistent and Recoverable Agent-Based Access to Heterogeneous Mobile Databases", SIGMOD Records, Vol.24, No.3, pp.44-49, Sept.1995.