データベースシステム 109－12
（1996. 7. 24）

分散データ統合のためのエージェント型仲介者モデル

松浦 春選　稲森 真二郎　小西 修
高知大理学部情報科学科
780 高知県高知市曙町 2-5-1
Tel:0888-44-8342
{ matsuura,sinamori,konishi} @is.kochi-u.ac.jp

概要

WWW(World Wide Web) によって実現された、分散された情報源という新しい環境は、現在の分散データベース技術での多くの概念の再考を促している。その中でも、分散された情報源を変換し、統合するという問題は重要である。我々は、この問題を解決する一つの方法として、エージェント型仲介者モデルを提唱した。このモデルでは、ユーザ代行エージェントとデータベースエージェントがデータベースの ECA 機構のもとで協調する。これはエージェントの信頼性を保証するものとなる。本論文では、このエージェント型アプローチのフレームワークについて特にエージェントの挙動の状態に注目した協調方式について述べる。

キーワード
メディエータ、エージェント、ECA ルール、アクティブデータベース

Agents based Mediator Model for Integrating Disaparate Information

Haruyori MATSUURA　Sinjirou INAMORI　Osamu KONISHI
Dept.of Information Science,Faculty of Science,Kochi University
2-5-1 Akebono-cho Kochi 780 Japan
Tel:0888-44-8342
{ matsuura,sinamori,konishi} @is.kochi-u.ac.jp

abstract

The new enviroment with distribution of information facilitated by the World Wide Web requires rethinking many of the concepts in current distributed database technology. The problem of converting and integrating disparate data souces is through the use of mediators. In our proposed model, a mediator consists of user agents and database agents which coordinate on ECA rules of active database. In this paper, we show a framework for agent coordination and control through ECA rules.

key words
Mediator, Agents, ECA rules, Active database

1 はじめに

我々は、分散された情報源の変換・統合を目的とする、複数のエージェントの連携によるエージェント型仲介者モデル (図 1) について考察を行ってきた [7] [8]。このモデルでは、システムは、高い拡張性を持ち、かつ信頼性が保証される特徴を持つ。今回、これらのモデルに従い、最も単純なモデルの実装を行いルールやエージェントについての検証を行ったのでこれについて報告する。

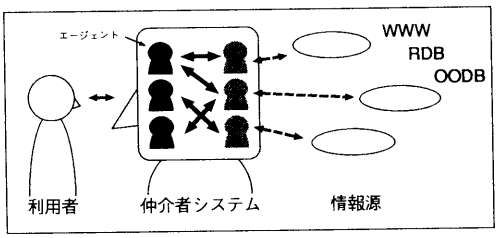

図 1: エージェント型仲介者モデル

2 章では、このモデルとエージェントの定義、要となる ECA 機構について述べ、3 章ではモデルに従ったシステムの実装に必要な構造やステータスについて、4 章では、ルールを用いたエージェント同期の実現、5 章では検証と以後の課題について述べる。

2 基本モデル

本章では、代行検索機構を実現するために考案したエージェントの連携の概念を述べ、これ以降用いるいくつかのキーワードについて定義を行う。

我々はこのモデルをワークフローモデル [1] を参考に構築した。あらかじめ決めておいた流れに従って、システムは常に管理下のエージェントの状態を監視し、適切なエージェントを逐次もしくは同時に起動していく。これにより、エージェントの衝突を回避でき、スムーズな連携を可能にする。

そこで、この連携を実現するための機構として、ドメイン・ワーク・オペレーションの各機構の定義を行い、企業における、作業の流れと比較して説明する。

企業は、ある作業に特化した部署の集合体であり、目標をもつパッケージである。ドメインは、企業と同じく、ある目標を達成するために、エージェントが集合し、作業を行う環境である。後述する、ワーク・オペレーションも、これに属する。今回、構築を目的とするのは、仲介者システムドメインである。

企業には、部や課などの、ある作業に特化した部署が存在し、それぞれが連携をして目標の達成を目指す。ワークは、ある作業に特化したエージェント集合を管理する部署である。ワークは具体的な作業内容（＝オペレーション）と、それを実行するためのエージェントのリストを持ち、作業の流れに従って、適格なエージェントの起動や、他のエージェントの中断を決定する。

また、作業の流れは、ルールで記述される。ルールには、ワークと、実行されるオペレーションを決定する、フロー・コントロール・ルール（FC ルール）と、ワークが、オペレーションの実行に適格なエージェントの選定をおこなう、エージェント・バインディング・ルール（AB ルール）に分けられる。ワーク・オペレーション・エージェントを効率的にルールでコントロールするため、それぞれの状態を常に反映する、ステータスを考える。図 2のように、この変化からルールは発動し、各機構は作業を行う。

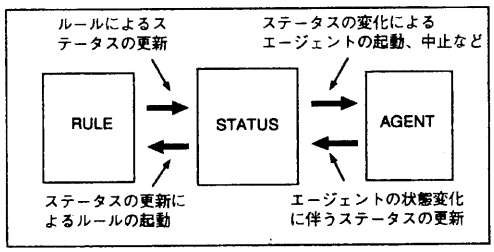

図 2: ルールとステータス、エージェント

簡単な例を挙げる。

例：

- **STEP1**： OperationO1 が "start" の状態に変化
- **STEP2**： イベント O1. 状態 = "start" の発生　ルールが発動
- **STEP3**： ルールによって AgentA1 の状態が "start" に更新 AgentA1 が起動される

これらを用いて連携に必要な流れの制御を行う。これは本研究の最も重要な部分であり、次章で詳しく述べる。

また、このモデルでは KQML [6] のようなコミュニケーション言語を必要とせず、各エージェントの結果のみ授受される。

これらの機構を DBMS を利用して実装を行うことで、連携に必要な流れの制御を DBMS の ECA 機構 [2] [3] を用いて実現できるだけでなく、分散環境でも十分な信頼性を保証でき、さらに優れた拡張性をも有することができる。

3　ステータスと ECA ルール

本章では、エージェント連携が DBMS の ECA 機構を用いてどのように実現されるかについて述べる。ECA 機構は、ドメイン、ワーク、オペレーション、エージェントそれぞれを監視し管理しなければならない。そこで、DB 上にそれらの現状を示すステータスを用意し、これを利用する。以下にこのステータスの内容を示す。これは以前のもの [8] に改良を加えたものである。

```
 -- Domain 変数 --
Domain     :String;     /* Domain 名 */
Domain_done:String;/* Domain の現状 */
Doamin_result :Object; /* 実行結果 */
 -- Work 変数 --
Work       :String;     /* Work 名 */
Work_done   :String; /* Work の現状 */
Work_result :Object;  /* 実行結果 */
 -- Operation 変数 --
Operation  :String; /* Operation 名 */
Operation_done :String;
            /* Operation の現状 */
Operation_result :Object;
                  /* 実行結果 */
Eligible_agents_of_operation
          :LIST(Agent);
           /* 適格エージェント */
 -- Agent 変数 --
Agent    :String;    /* Agent 名 */
Agent_result :OBJECT; /* Agent 結果 */
Agent_done :String; /* Agent の現状 */
```

これらは、以下のようなテーブルとして、RDB 上に実装される。

```
DOMAIN_STATUS[ Domain ,Domain_done ]
WORK_STATUS[ Domain , Work ,
    Work_done ]
OPERATION_STATUS[ Domain , Work ,
    Operation , Operation_result ,
    Eligible_agents_of_operation]
AGENT_STATUS[ Domain , Work ,
    Operation,Agent , Agent_done ]
```

これらは ECA ルールと、DB 外で起動されるワークとエージェントによってのみ更新される。例えば、Work の状態は図 3のようにあらわされる。次にこれらのステータスを用い

WORK_STATUS

Domain	Work	Work_done
D1	W1	start
D1	W2	finished
⋮	⋮	⋮

図 3: WORK_STATUS の例

て ECA ルールを作成する。

ECA コンディションは DB 上のステータスと DB 外で起動されているワークとエージェントの状態の一貫性を保ち、ルールによるエージェントの連携を実現するために最も重要な部分である。もし、複数の ECA コンディションが指定されるときは、それらは論理的結合によって結合される。

ECA アクションではステータスの更新が実行される。ステータスの更新はデータベースの質問処理として実施される。一つ以上の ECA アクションが指定される場合には、全ての ECA アクションが実行される。

前述の通り、ルールにはFCルールとABルールがある。

3.1 フロー・コントロール・ルール

FCルールは、ドメイン内のワークとオペレーションがどのような順番で実行されていくかを記述する。そのためイベントにはステータスの更新と作業の終了が、考えられる。

・ECAイベント：

```
ECA_EVENT = {start,finish,abort,update}
```

startとfinish,abortはドメイン、ワーク、オペレーションの起動と終了、updateはステータスの更新をシグナルとして受ける。

・ECAコンディション：

```
ECA_CONDITION = {
  Domain_started(domain:DOMAIN);
  Domain_finished(...);
  Work_started(domain:DOMAIN,
               work:WORK);
  Work_finished(...);
  Work_aborted(...);
  Operation_started(domani:DOMAIN,
            work:WORK,
            operation:OPERATION);
  Operation_finished(...);
  Operation_aborted(...);}
```

これらは、ドメイン、ワーク、オペレーションのそれぞれの現在の状況を確認するために用いられる。

・ECAアクション：

```
ECA_ACTION = {
  add_Work_started(domain:DOMAIN,
        work:WORK);
  add_Work_finished(...);
  add_Work_aborted(...);
  add_Operation_started(
         domain:DOAMIN,work:WORK,
         operation:OPERATION);
  add_Operation_finished(...);
  add_Operation_aborted(...);}
```

ワーク、オペレーションの起動、終了、中止を決定する。

FCルール例：

```
FC rule FCr1 =
{ E : finish
  C : Work_finished(DOMAIN1,WORK1);
  A : add_Work_finished(DOAMIN1,WORK1);
      AND add_Work_started(DOMAIN1,WORK2);}
```

このルールでは、WORK1の終了を受けて、WORK2の起動を行う。

3.2 エージェント・バインディング・ルール

ABルールはエージェント同期を実現するために用いる。これにより、FCルールで大まかな流れが決定していても、状況に応じた変化を生むことができる。オペレーションが実行状態になると、「最適エージェント」と「競争エージェント」のどちらかでエージェントが選定され実行される。「最適エージェント」はオペレーションに最も適したエージェントが一つ選ばれ実行される。オペレーションの結果は、そのエージェントの結果である。「競争エージェント」はオペレーションに適したエージェントが全て選ばれて実行される。オペレーションの結果は、最も早く終了したエージェントの結果が用いられる。

・ECAイベント： イベントにはオペレーションの決定とエージェントの状態の変化がある。

```
ECA_EVENT = {start,abort,finish,update}
```

startとfinish,abortはオペレーションとエージェントの起動と終了、中止、updateはステータスの更新をシグナルとして受ける。

・ECAコンディション：

```
ECA_CONDITION = {
/* operation / agent relationship */
operation_started_by_agent(
      domain: DOMAIN
      work: WORK,
      operation:OPERATION,
      agent:AGENT);
operation_finished_by_agent(...);
operation_aborted_by_agent(...);
operation_reset_by_agent(...);
```

オペレーションの実行状態とそれにかかわるエージェントについてチェックする。

```
/* operation */
operation_started(
      work: WORK,
      operation:OPERATION);
operation_finished(...);
operation_aborted(...);
operation_reset(...);
```

オペレーションの現在の状態をチェック。

```
/* agent information */
eligible_agent(
      work:WORK,
      operation:OPERATION,
      agent:AGENT);
is_executing_agent(
      task:TASK,
      operation:OPERATION,
      agent:AGENT);}
```

エージェントの選定方法とエージェントの現在の状態をチェック

・ECA アクション：

```
ECA_ACTION = {
/* agent management */
add_agent_executing(
      domain:DOMAIN
      work: WORK,
      operation:OPERATION,
      agent:AGENT);
delete_agent_executing(
      domain:DOMAIN
      work: WORK,
      operation:OPERATION,
      agent:AGENT);
add_agent_forbidden(
      domain:DOMAIN
      work: WORK,
      operation:OPERATION,
      agent:AGENT);
add_agent_finished(...);
add_agent_abort(...);
add_agent_reset(...);
forbid_others_to_start(
      work: WORK,
      operation:OPERATION,
      agent:list(AGENT));
```

エージェントの起動、停止や起動の禁止、あるエージェント以外の起動の禁止などを行う

```
/* result handling */
operation_result(
      work: WORK,
      operation: OPERATION,
      work_result: OBJECT);
set_operation( operation_done:
                OPERATION_DONE );
set_work_finished_if_operations(
     operation_done:OPERATION_DONE
                              );}
```

4 エージェント同期

ここではドメイン:D1, ワーク:W1, オペレーション:O1, エージェント:A1,A2 の最も単純なモデルで、エージェントが競争エージェントのルールによって実行される時どのようにルールが定義されるのかについて述べる。

```
/*   FCrule FCr1   */
{ E : start
  C : Domain_started(D1);
  A : add_Work_start(D1,W1);}
```

ドメイン D1 の起動によるワーク W1 の起動。

```
/*   ABrule ABr1a   */
{ E : start
  C : eligible_agent(D1,W1,O1,A1);
      AND NOT operation_finished(D1,W1,O1);
  A : add_agent_executing(D1,W1,O1,A1);}
/*   ABrule ABr1b   */
{ E : start
  C : eligible_agent(D1,W1,O1,A2);
      AND NOT operation_finished(D1,W1,O1);
  A : add_agent_executing(D1,W1,O1,A2);}
```

ルール ABr1a では、まず eligible_agent() によってエージェント A1 がオペレーション O1 の実行に適しているか判断される。オペレーション O1 が終了していなければ、A1 は実行される。同時に A2 は実行が禁止されていないのでルール ABr1b によって実行される。

```
/*   ABrule ABr2a   */
{ E : finish
  C : is_executing_agent(D1,W1,O1,A2);
      AND NOT operation_finished(D1,W1,O1);
  A : delete_agent_executing(D1,W1,O1,A1);
      add_agent_finished(D1,W1,O1,A2);
      take_result(D1,W1,O1,result);
      set_operation(FINISHED);
      set_work_finished_if_operation(
            FINISHED,ABORTED);
      update_worklists(D1,T2,O2,(A3,A4));}
```

エージェントA2が最も早く終了したと仮定すると、A2の終了によって、ルールABr2aが実行される。オペレーションO1はまだ終了していないので、アクションが実行される。エージェントは終了したので、delete_agent_executing() add_agent_finished() が実行され、結果の受け渡し、ステータスの更新なども行われる。

```
/*   ABrule ABr2b   */
{ E : finish
  C : is_executing_agent(D1,W1,O1,A1);
     AND operation_finished(D1,W1,O1);
  A : delete_agent_executing(D1,W1,O1,A1);
      set_work_finished_if_operation(
            FINISHED,ABORTED);       }
```

エージェントA1はA2より遅れたため、A1が終了したときには、すでにオペレーションO1は終了している。そこで、作業から外され、終了となる。

5 おわりに

我々は、分散された情報源の変換・統合を目的とする、複数のエージェントの連携によるエージェント型仲介者モデルについて考察を行ってきた。このモデルでは、システムは、高い拡張性を持ち、かつ信頼性が保証される特徴を持つ。今回、これらのモデルに従い、最も単純なモデルのルールやエージェントについての検証を行った。今後の課題は、このモデルに従って、プロトタイプを作成することである。

参考文献

[1] Christoph Bußler, Stefan Jablonski "Implementing Agent Coordination for Workflow Management System Using Active Database Systems",RIDE 94'pp.53-59

[2] Gehani, N.and Jagadish,H.V "Ode as an Active DataBase : Constraints and Triggers" Proc. the 17th VLDB Conference,pp.327-336

[3] Gehani,N.H. et al. "Event Specification in an Active Object-Oriented Database" Proc.the 1992 ACM-SIGMOD Conference, pp.81-90 (1992).

[4] Michael R. Genesereth, and Steven P. Ketchpcl,"Software agents."Commun. ACM, vol .37, no.7, pp.48-53.1994.

[5] Patties Maes,"Agents that reduce work and information overload." Commun. ACM. vol37. no.7. pp.31-40. 1994.

[6] T. Finin, J. Weber, G. Wiederhold, M. Genesereth, R. Fritzson, D. McGuire, P. Pelavin, S. Shapiro, and C. Beck, "Specification of the KQML agent-communication language." Technical Report EIT TR 92-04, Enterprise Integration Technologies,1992(Updated July 1993).

[7] 小西 修, 松浦 春選, "分散協調処理による異種問題解決系の統合利用" 情報処理学会研究報告,pp.311-318,95-DBS-104,1995.

[8] 松浦 春選, 小西 修, "システムとその利用者の協調のための発展型仲介者モデル" 電子情報通信学会技術研究報告,pp.37-42,DE96-7,1996.