エージェントフレームワークにおけるリポジトリ機構の設計と実装

打矢隆弘[†] 武田敦志[†]
菅沼拓夫[†] 木下哲男[††]

マルチエージェントシステムを利用者や環境に適応させ，柔軟でかつ高度なサービス生成・調整を実現するために，エージェントの動作結果を効果的に利用する方法を提案する．従来のマルチエージェントシステムでは，エージェントの動作環境でのタスク遂行後に，その動作履歴等を保持・管理したり，それをマルチエージェントシステム全体に効果的に反映させたりすることは困難であった．そこで我々は，リポジトリを利用してエージェントの組織構成・再構成を行うリポジトリ型マルチエージェントフレームワークに着目し，リポジトリ機構にエージェントの動作結果を活用する機構を新たに導入することで，上記の問題を解決する．本論文では本手法に基づくリポジトリ機構の設計について述べ，プロトタイプの実装とその動作実験を通して提案手法の有効性を示す．

Design and Implementation of Repository Architecture on Agent Framework

Takahiro Uchiya,[†] Atushi Takeda,[†] Takuo Suganuma[†]
and Tetsuo Kinoshita[††]

It is important to use agent's behavioral history of a multiagent system effectively as a way to realize advanced service generation/tuning and make the multiagent system adaptable with respect to users and environment. However, it is difficult for a multiagent system that the system holds and manages its behavioral history and reflects them to the system's behavior effectively after the task execution. We focus on a repository-based multiagent framework that carries out the organization and reorganization of agents using the repository. We propose a new mechanism of the repository that utilizes the behavioral history to deal with the above problem. In this paper, we explain the design of the repository mechanism and a prototype system based on the proposed method. Then we demonstrate the experimental results to show the effectiveness of the proposed method.

1. まえがき

インターネットに代表される広域分散環境においては，サービスに対するユーザの要求やサービス利用形態は多種多様である．またネットワークやプラットフォームの資源も有限であり，かつ資源が共有されることによって，その性能的な変動も大きい．このような性質を持った広域分散環境において，利用者要求や環境の変化に柔軟に対応し，安定したサービスを提供するための手段として，マルチエージェントシステムの適用が注目されている．マルチエージェントシステムは，複数のエージェントが協調して組織的活動（組織構成，再構成，交渉等）を行い，動的にサービス機能を構成・調整することで，ユーザに安定したサービスを提供するシステムである．

マルチエージェントシステムを利用者や環境に適応させ，柔軟でかつ高度なサービス構成・調整を行うためには，エージェントの動作結果を効果的に利用する必要がある．たとえば，ビデオ会議サービスを提供するエージェント[4),5)]は，動作中に協調的にサービス品質パラメータを変更させ，動作中の環境に適応する．ここで，あるネットワーク環境やプラットフォーム上で動作した後，その動作結果からサービスの調整・制御にとって有用な情報が抽出できれば，再び同様の条件でビデオ会議サービスの利用要求が発生したときに，システム構成時の組織構成のための手続きや初期パラメータの決定等の処理が効率化でき，要求発生からサービス提供開始までの時間を短縮することができる．

† 東北大学電気通信研究所
Research Institute of Electrical Communication, Tohoku University
†† 東北大学情報シナジーセンター
Information Synergy Center, Tohoku University

しかしながら従来のマルチエージェントシステムでは，エージェントの動作環境におけるタスク遂行後に，その動作履歴等を保持・管理したり，それをマルチエージェントシステム動作全体に効果的に反映させたりすることは困難であった．これは，エージェントがネットワーク環境上に分散して存在するため，たとえ何らかの仕組みによりその永続性が保証されているとしても，個々のエージェントの動作結果を系統的・集約的に再利用するための手段が与えられていなかったことに起因する．

これまで，我々はエージェントリポジトリを利用してエージェントの組織構成・再構成といったマルチエージェントシステムのライフサイクルを支援するリポジトリ型マルチエージェントフレームワークを開発してきた[1),2),6)]．リポジトリ型マルチエージェントフレームワークとは，サービスや機能を実現するコンポーネント群をエージェントのサーバであるリポジトリに格納し，ユーザの要求やネットワーク/プラットフォームの状況に応じて動的にサービスを構成・調整・再構成することにより柔軟なサービス提供・運用を可能とするエージェントフレームワークである．このフレームワークのリポジトリ機構に対してエージェントの動作結果を活用する機構を導入することにより，従来に比べサービス組織構成・調整を効果的に行うマルチエージェントシステムを構築できると考えられる．たとえば，1) 一度利用したマルチエージェントシステムの組織構成情報を利用して次回の組織構成の際に組織構成を効率化する，2) 動作終了後のエージェントの推論履歴を基にエージェント内部の推論サイクルを効率化しサービスの即応性を高めることでサービスを高度化する，3) ユーザ要求やプラットフォームの環境に応じてリポジトリで提供するサービスの対象を制御し，サービスのカスタマイズや適応的なサービスの提供を実現する，等があげられる．

本論文では，上述した機能を実現するために，従来のリポジトリ型マルチエージェントフレームワークにおけるリポジトリ機構に対して，新たに以下の 3 つの機構を導入する．

（M1） 動作結果利用機構

エージェントの過去の動作結果を活用することにより，サービスを実現しているエージェント群の組織構成の調整やエージェント推論速度を調整し，サービス提供の効率を向上させる機構．

（M2） エージェント ライフサイクル管理機構

一度動作環境で動作したエージェントを保持し，その後のエージェントライフサイクルの管理を行う機構．

（M3） リポジトリ管理エージェント

（M1），（M2）を含めたリポジトリの動作全体を管理するエージェント．

上記（M1），（M2），（M3）を組み込んだ新たなリポジトリ機構を Active Agent Repository（AAR）と呼ぶ．本論文では AAR の設計および実装について述べ，動作実験を通して本手法の有効性を示す．

本論文の構成は以下のとおりである．まず 2 章で AAR を提案し，3 章で AAR の設計について述べる．次に，4 章では AAR のプロトタイプの実装について述べ，5 章で評価実験の結果を示す．最後に 6 章でまとめと今後の課題を述べる．

2. Active Agent Repository の提案

2.1 リポジトリ型マルチエージェントフレームワーク

本提案の基盤技術となるリポジトリ型マルチエージェントフレームワーク[1),2)]の例として，ADIPS フレームワークの概要を述べる．**図 1** に ADIPS フレームワークのモデルを示す．

ADIPS フレームワークでは，利用者はユーザエージェントと呼ばれる利用者の要求を獲得するエージェントに対してサービス要求を行う．ユーザエージェントは「リポジトリ」と呼ばれるエージェント（ソフトウェア部品）の保管庫にその要求をサービス要求通知（タスクアナウンス）として送信する．

リポジトリの基本的機能は，エージェントワークプレース上で動作するユーザエージェントからの要求を受信し，それをリポジトリ内で待機しているエージェント群に配信し，当該要求が処理可能と判断したエージェントが，自分自身の複製をワークプレースで動作する新しいエージェントとして生成する，あるいは，当該エージェントが，契約ネットプロトコル[14)]を利用しリポジトリ内の他のエージェント群と自律的に協調して，マルチエージェントシステム組織を構成し，同組織をワークプレース上で動作するマルチエージェントシステムとして生成したりする処理を実行するものである．これらの処理は，リポジトリに格納された各

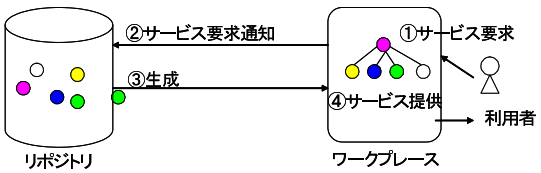

図 1 ADIPS フレームワーク
Fig. 1 ADIPS framework.

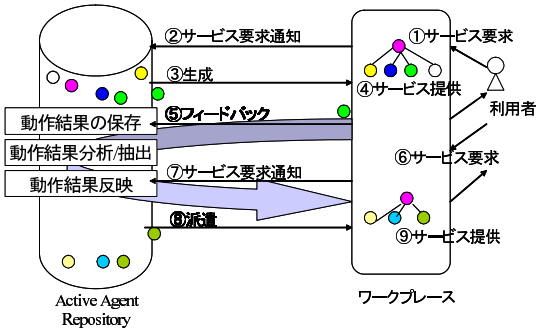

図 2 AAR 型マルチエージェントフレームワーク
Fig. 2 AAR-based multiagent framework.

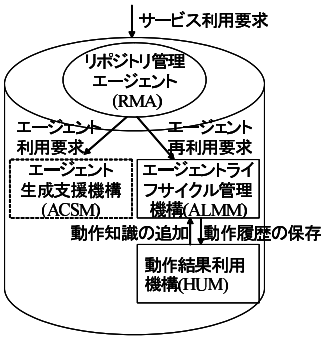

図 3 AAR の機能構成
Fig. 3 Architecture of AAR.

エージェントが持つ知識に基づいて自律的に実行される．なお，エージェントの定義やその知識は，上記フレームワークが提供するエージェントプログラミング言語を用いて記述され，ファイルとして蓄積・管理されており，リポジトリが，これらのファイルを読み込み，リポジトリ内で待機するエージェントを準備している．

ADIPS フレームワークに代表されるリポジトリ型マルチエージェントフレームワークの特徴として，エージェントのサービス機能の管理の集中化があげられる．すなわち，リポジトリ内のエージェントに対する知識の調整/追加/改良を行うことにより，そのリポジトリを用いて構成されるマルチエージェントシステム全体の動作を管理することが可能である．この特徴を活かして，エージェント群の動作後に，それらの動作結果をリポジトリにフィードバックし，これをエージェント動作知識として集中的に管理することにより，必要に応じてこれらの知識に基づき，過去の動作結果をエージェント群全体に反映させることが可能となる．さらに，エージェントの組織構成機能と連携することにより効果的なサービス提供が行える．

2.2 Active Agent Repository の概念

マルチエージェントの効果的なサービス構成・調整を目指して，本論文では，リポジトリ型マルチエージェントフレームワークにおけるリポジトリ機構を拡張した Active Agent Repository（AAR）を提案する．**図 2** に AAR に基づくマルチエージェントフレームワーク（AAR 型マルチエージェントフレームワーク）の動作概要を示す．

AAR 型マルチエージェントフレームワークでは，組織構成を行ってワークプレースにインスタンシエートし，サービス提供を実行したエージェント群が，サービス終了後に AAR に復帰（フィードバック）する．AAR では，復帰後のエージェントの動作結果の情報を保存し，必要に応じてその動作結果を分析/抽出し，それをエージェントの知識に反映させる．これにより過去に提供したサービスと同様の条件で再度サービス要求通知が発生した場合に，AAR 内で行われる複雑な組織構成動作を簡略化し，条件に即した適切なエージェント群を即応的に選択して送出（ディスパッチ）することができる．**図 3** に AAR の機能構成を示す．

AAR 型マルチエージェントフレームワークにおける AAR の基本的な設計指針として，従来のリポジトリ機構が持つ契約ネットプロトコルによるエージェントシステム形成の機能に加えて，エージェントの動作結果を活用する機構をリポジトリに付加することがあげられる．さらに，フィードバック後のエージェントの永続性を保障し，効果的なライフサイクル支援を行う機構，および，リポジトリ内のエージェントを総合的に管理する機構を標準的に装備することとする．

上記の設計指針に基づき，AAR は動作終了後のエージェントシステムをフィードバックし，動作結果をエージェントシステムに反映させて，次回の要求の際により効率的な再生成を行う機能を備えたリポジトリに拡張する．

そこで，AAR はエージェント生成支援機構（ACSM: Agent Creation Support Mechanism），エージェントライフサイクル管理機構（ALMM: Agent Lifecycle Management Mechanism），動作結果利用機構（HUM: History Utilization Mechanism），およびリポジトリ管理エージェント（RMA: Repository Management Agent）から構成される．以下，これらの機能の概要を述べる．詳細は 3 章以降の設計において説明する．

（M0） エージェント生成支援機構（ACSM）
従来のリポジトリ型フレームワークのリポジトリに存在する機構であり，エージェント生成/組織構成/再構成の処理を実行する．

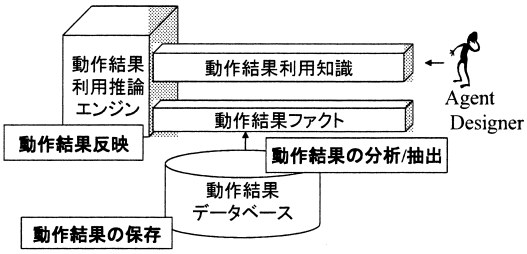

図 4 動作結果利用機構
Fig. 4 History utilization mechanism.

（M1） 動作結果利用機構（HUM）
後述の ALMM と連携してエージェントの過去の動作結果を保存するとともに，それを抽出/加工してエージェントの知識に反映し，エージェント組織構成・調整処理が効果的に行えるように調整する．

（M2） エージェントライフサイクル管理機構（ALMM）
ワークプレースでサービス提供を行った後リポジトリにフィードバックしたエージェントを保持するとともに，それ以降のそのエージェントのライフサイクルを管理する．

（M3） リポジトリ管理エージェント（RMA）
リポジトリ全体の動作を管理するためのエージェント．特にここではサービス利用要求獲得時に ACSM と ALMM に要求メッセージを振り分ける役割を持つ．

3. Active Agent Repository の設計

3.1 動作結果利用機構

動作結果利用機構（HUM）は，エージェントの動作結果を活用するための機構であり AAR において最も重要な機構である．**図 4** に HUM の内部構造を示す．動作結果利用機構は動作結果利用推論エンジンと動作結果データベースから構成される．動作結果利用推論エンジンは，ルール型知識に基づく推論機構であり，エージェントの動作結果データから必要なデータの抽出を行い，それらをリポジトリ内のエージェントに反映させる処理を行う．また動作結果データベースは，フィードバックしたエージェントの動作結果データを格納するデータベースである．

ここで動作結果利用機構の動作を簡単に説明する．エージェントがワークプレースからリポジトリにフィードバックすると，エージェントライフサイクル管理機構から動作結果利用機構へエージェントの動作結果データが送信される．そのデータを動作結果データベースに保存する．このとき，動作結果データベースでは，データベースへのデータの格納と同時に，そのデータが動作結果利用推論エンジンのワーキングメモリ内にファクトとして追加される．次に，推論エンジンが起動され，動作結果利用知識を用いてエージェントの動作結果データから必要なデータを抽出し推論を行い，エージェントの動作結果を各エージェントに反映させる．

3.1.1 動作結果データベース

動作結果利用機構は，エージェントのワークプレースでの活動履歴をデータとして動作結果データベースに蓄積/管理する．動作結果データベースでは各エージェントの動作結果がそれぞれ個別のデータとして表現される．動作結果データ（HD）のモデルを**図 5** に示す．また，**図 6** に動作結果データの例を示す．

AAR 型マルチエージェントフレームワークで扱うエージェントはルール型のエージェントを想定する．これは，エージェントの知識やデータの利用状況を獲得しやすく，また動作結果として利用しやすいことが特徴としてあげられる．動作結果データとして，ユーザ名，ワークプレース名，組織構成情報，ワーキングメモリ履歴，メッセージ履歴，推論履歴等が収集でき，これらはエージェントの動作と密接に関連しているため，非常に利用価値があると考えられる．動作結果データのモデルとしては，上記のような利用価値のあるデータを収集し，有効に次回のサービス構成・調整に反映することを目的として設計した．上記のデータは特定のアプリケーションやユーザに依存することなく，エージェントの動作結果として獲得可能である．

ここで，リポジトリ記述は，リポジトリ内でエージェントを生成するためのエージェントプログラムの記述である．メタデータ記述はその動作結果データのメタデータであり，データ ID，データ生成時刻，データ保存期間，最終アクセス時刻，データベース名等が記述される．エージェント名はリポジトリ内での抽象的な名前やワークプレースでのインスタンス名，ワークプレース名はエージェントが活動した環境，ユーザ名はエージェントを利用したユーザ名が記述される．組織構成情報はワークプレースで活動時の最終的なエージェント組織構成情報が記述され，動作結果データを利用して組織構成の調整をする場合に使用される．ワーキングメモリ状態記述はワークプレースで活動時の最終的なファクトの集合が記述され，エージェントが保持するデータを利用する場合に使用される．ワーキングメモリ履歴はワーキングメモリの発火回数の多いファクトのデータが記述される．メッセージ履歴は

```
    HD := <RD,HDB>
    RD := リポジトリ記述
   HDB := { hd | hd = <hddesc, aname, wpname, uname, OCDesc, WM,
          WMH, MH, IH>}
hddesc := メタデータ記述
 aname := エージェント名
wpname := ワークプレース名
 uname := ユーザ名
OCDesc := 組織構成情報
    WM := ワーキングメモリ状態記述
   WMH := ワーキングメモリ履歴
    MH := メッセージ履歴
    IH := 推論履歴
```

図 5 動作結果データのモデル
Fig. 5 Model of history data.

```
(data
:RD VideoConferenceManager.agt
:hddsec (hd :DCT " 2001/07/18 ": id 1 :ET " 2002/08/18 09:00 "
          : DB aar001.DB :LAT " 2001/07/18 09:00 ")
:RNAME VideoConferenceManager
:WNAME videoconferencemanager1.robin.shiratori.riec.tohoku.ac.jp
:PLACENAME robin.shiratori.riec.tohoku.ac.jp
:UNAME Takahiro
//組織構成情報
:OCDesc (Members :manager "(interface)"
                :member1 Video :member2 Audio :member3
                  WhiteBoard :member4 Ui)
//ワーキングメモリ状態記述
:WM ((create :author "takahiro@shiratori.riec.tohoku.ac.jp"
             :date "2001/05/13")(java :class javaprog.TclController))
//ワーキングメモリ履歴
:MostFiredMemoryFact (agent-state :state initial)
//メッセージ履歴
:MessageReceivedCount 36
:MostMessageSender Vic
:MostMessageSenderSendCount 10
//推論履歴
:MostFiredRuleName inter-1
:SecondFiredRuleName inter-2
:ThirdFiredRuleName inter-3
:BpAccessCount 2
:BpEventReceivedCount 2
)
```

図 6 動作結果データの例
Fig. 6 Example of history data.

エージェント動作時のメッセージ送受信の履歴が記述される．推論履歴は，推論機構におけるルールの発火回数の多いルール名が記述され，このルール名を利用する場合に使用される．

3.1.2 動作結果利用推論エンジン

3.1.1 項で説明した動作結果データを利用して，エージェントに動的に知識を追加する動作結果利用推論エンジンについて説明する．動作結果利用推論エンジンでは，エージェントの動作結果データがファクトとし

```
(historyknowledge huk001
 (knowledge
  (rule rule1
   (data :RNAME video :WM ((savedata :performance ?p1))) = ?agent1
  ~(data :performance < ?p1)
   (data :RNAME audio :WM ((savedata :performance ?p2))) = ?agent2
  ~(data :performance < ?p2)
   (data :RNAME whiteboard :WM ((savedata :performance ?p3))) = ?agent3
  ~(data :performance < ?p3)
   (data :RNAME vcm) = ?manager
    -->
   (makelink best-performance-videogroup ?manager:RNAME (?agent1:RNAME
    ?agent2:RNAME ?agent3:RNAME))
 )
)
```

図 7 動作結果利用知識の例
Fig. 7 Example of utilization knowledge.

てワーキングメモリにセットされる．またエージェント開発者/管理者により，動作結果データの利用方法を記述した動作結果利用知識がルールとして格納される．**図 7** に動作結果利用知識の例を示す．

ここで，過去に利用した複数のエージェントが存在すると仮定する．このルールは，過去に利用したエージェントの中で各機能ごと（ビデオ会議における映像サービスを制御する video エージェント，音声サービスを制御する audio エージェント，共有エディタを制御する whiteboard エージェント）に最も性能の良かったエージェント群を選択するための利用知識が記述されている．

動作結果利用推論エンジンは，ルールの条件部とファクトとのパターンマッチングにより発火するルールを決定し，そのルールのアクション部を実行する一般的なルールベースシステムとして構成されるが，動作結果利用機構の機能を実現するために以下のようなアクションの拡張を行っている．

(a) makelink アクション

（`makelink` パフォーマティブ 組織構成エージェント名 （エージェント名 1 エージェント名 2 ...））

組織構成を調整するためのアクションである．このアクションを実行すると，動的にエージェントの知識に新しい組織構成知識が追加され，次回の利用の際には，より高速に組織構成を行うことが可能になる．

(b) makerule アクション

（`makerule` エージェント名 追加するルール）

単体のエージェントに新しい知識を追加するためのアクションである．このアクションを実行すると，リポジトリ内に存在するエージェントに動的にルール型の知識が追加され，エージェントの振舞いや初期パラメータの変更を行うことができる．

(c) changeruleorder アクション

（`changeruleorder` エージェント名 （発火頻度の高いルール））

ルールの優先度（発火順序）を調整するアクションである．このアクションを実行するとルールの発火順序を制御し，動作時に頻繁に発火するルールを優先的に発火させることができる．その結果エージェントの推論サイクルを効率化することができる．

3.2 エージェント ライフサイクル管理機構

エージェントライフサイクル管理機構（ALMM）は，ワークプレースにインスタンシエートされた後リポジトリにフィードバックしたエージェントのライフサイクルを管理する機構である．本機構は以下の 3 つの機能から構成される．

1) エージェント 送受信機能

ワークプレースから復帰したエージェントを受信し，後述のエージェント生存環境に保持させるための機能である．また，エージェント生存環境に存在するエージェントをディスパッチ（再生成）する際の送信機能を有する．

2) エージェント 生存環境

ワークプレースから復帰したエージェントを保持し，次回のサービス提供までそのエージェントの永続性を保障するために，ワークプレースのよ

うにエージェントが存在し動作可能な環境がリポジトリ内に必要である．エージェント生存環境はリポジトリ内において，フィードバックしたエージェントが再利用されるまでに，動作した結果を保存し，知識を調整するために協調動作するためのエージェント動作基盤を提供する．

3） ライフサイクル管理機能

エージェント生存環境内のエージェントのライフサイクルを管理する．保持期間の経過したエージェントを停止し，またフィードバック後の同一の知識を持ったエージェントが生存環境に存在する場合には古いエージェントを停止する．

3.3 リポジトリ管理エージェント

リポジトリ管理エージェント（RMA）は主に，ワークプレース/リポジトリ間のメッセージの窓口としての役割を持つ．以下に RMA の機能をあげる．

（1） エージェント /サービス利用要求処理機能

ワークプレースから〈request-agent〉または〈request-service〉のパフォーマティブでエージェント要求/サービス要求が送られた場合，RMA は要求に適合したエージェントにサービスを実現する組織の形成を依頼する．このとき，ACSM と ALMM に対する要求メッセージの振り分けを行う．すなわち，以前利用されたエージェントが利用可能な場合は ALMM に対しエージェント再利用メッセージを送り，それ以外の場合は ACSM にエージェント利用メッセージを送る．

（2） エージェント検索要求処理機能

ワークプレースから〈search-agent〉のパフォーマティブでエージェント検索要求が送られた場合，RMA はエージェント名前管理テーブルを利用してエージェントの存在を調査し，true または false を返す．

（3） 利用可能エージェント /サービス名要求処理機能

ワークプレースから〈request-agentnames〉または〈request-servicenames〉のパフォーマティブを用いて利用可能なエージェント名やサービス名に関する問合せ要求が送られた場合，RMA は，現在利用可能なエージェント名あるいはサービス名一覧を返す．

4. プロトタイプシステムの実装

3 章で設計した Active Agent Repository（AAR）のプロトタイプを実装した．AAR の実装環境として，Sun Ultra SPARCStation（OS: Solaris7）上の Java2 を利用した．エージェントフレームワークのベースとして TAF[3] を使用した．TAF のエージェント知識記述はルール型知識記述であり，エージェント組織を系統的に再利用するための動作結果等を獲得しやすいという利点を持つ．ただし，TAF は，エージェントシステム開発の教育支援用環境として開発されていたため，分散環境での利用は想定されていなかった．そこで，まず，JavaRMI を利用してリポジトリとワークプレース間の通信機能を付加し，分散環境での利用を可能とした．次に，3 章において述べてきた機能拡張を付加したリポジトリ AAR を実装した．なお HUM の推論機構については TAF の推論機構を拡張した．また，知識記述については基本的に TAF のルール記述に準拠して行い，必要に応じてアクション部の拡張を施した．実装した総クラス数は 50 クラス，プログラムソースコードの総ライン数は約 10,400 行であった．

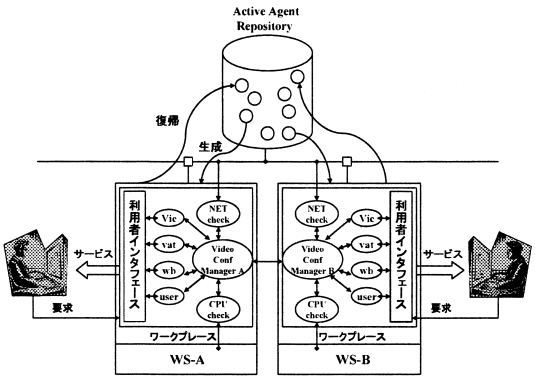

図 8 AAR を用いた FVCS：やわらかいビデオ会議システム
Fig. 8 FVCS: Flexible Videoconference System using AAR.

5. 実験と考察

5.1 評価実験とその目的

評価実験用アプリケーションとして，やわらかいビデオ会議システム（FVCS: Flexible Videoconference System）[4),5)] を用いた．FVCS のエージェント構成を**図 8** に示す．FVCS は，マルチメディア通信機能，ネットワーク・プラットフォーム状況監視機能等をエージェント化し，マルチエージェントシステムとして実現したビデオ会議システムである．本論文では FVCS を用い，マルチエージェントシステムにおけるエージェント組織構成の効率化とエージェントの推論サイクルの効率化によるサービスの高度化の 2 点に着目して以下の評価を行った．これにより本提案の有効性を確認する．FVCS を用いて以下の実験を行った．

（Ex.1）エージェント組織構成の効率化の評価

（Ex.2）推論サイクルの効率化によるサービスの高度

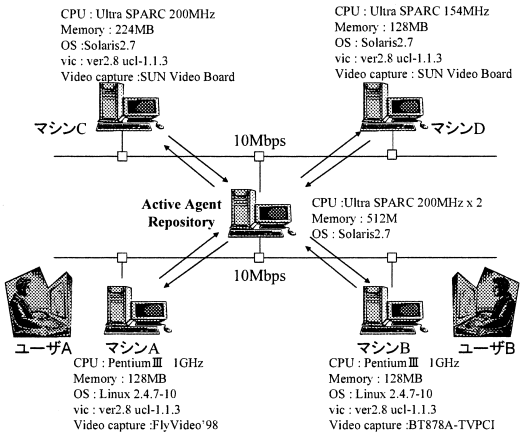

図 9 実験環境
Fig. 9 Experiment environment.

化の評価

（Ex.1）では，活動後のエージェント群の履歴から得られる組織構成情報を利用して，同じユーザから同条件のサービス要求が発生した場合に，その実行時間が短縮されるかどうか（組織構成の効率化）を検証することを目的とする．

（Ex.2）では FVCS の動作結果を利用してエージェントの推論サイクルの効率化を図り，サービスの向上が実現されているかを検証することを目的とする．

5.1.1 実験環境

実験で使用したハードウェア環境を**図 9** に示す．Sun Microsystems 社の Ultra SPARCStation マシン（Solaris，CPU400 MHz，メモリ 512 MB）を Active Agent Repository として動作させ，マシン A（Linux，CPU1 GHz，メモリ 128 MB），B（Linux，CPU1 GHz，メモリ 128 MB），C（Solaris，CPU200 MHz，メモリ 224 MB），D（Solaris，CPU 154 MHz，メモリ 128 MB）をワークプレースおよびビデオ会議用端末として用いた．

5.1.2 実験条件

実験で使用した動作結果利用知識の一例を**図 10** に示す．なお，これらの知識の記述形式は，TAF [3] のルール記述形式に準拠している．この例では「FVCS の最終的な組織構成情報を利用してエージェント組織構成を高速化する」知識，および，「ルール発火回数を利用して FVCS の Vic エージェント，CPUcheck エージェントの推論サイクルを効率化する」知識がそれぞれ記述されている．

5.1.3 動作実験

以下の手順に従って実験を行った．

（1）マシン A 上で稼動するワークプレース A からサービス提供を受けるユーザ A と，マシン B 上で稼動するワークプレース B からサービス提供を受けるユーザ B の間でビデオ会議システム（FVCS）を起動する．始めにユーザはワークプレースを起動し，インタフェースを介してビデオ会議サービスの要求を行う．サービス要求はネットワークを介して Active Agent Repository に通知され，リポジトリで契約ネットプロトコルを用いて FVCS の組織構成が行われ，エージェント群が各ワークプレースに生成してサービスを開始する．

（2）ビデオ会議終了後，ワークプレースで活動したエージェントをリポジトリにフィードバックする．

（3）再びユーザ A，B 間でビデオ会議を行うようサービス要求を行う．

以上の流れを 1 セットとして，5 セットの施行を行った．次に，マシン C，D を用いて，ユーザ A，B が上記の 5 セットの施行を同様に行った．

5.2 実験結果と評価

（Ex.1）エージェント組織構成の効率化の評価

各エージェントの動作結果がデータベースに保存された後，動作結果データが動作結果利用推論エンジンのワーキングメモリに追加された．そのときの動作結果利用知識に基づき，前述したルール speedup-organization 等が発火し，その makelink アクションにより，FVCS の組織構成エージェントである VideoConfManager エージェントに新しい組織構成知識（direct-award による直接落札組織構成）が動的に追加された．ここで再びサービス要求を行うと，RMA が以前利用したサービスであることを認識し，ALMM 内の VideoConfManager エージェントにタスク通知し，VideoConfManager エージェントが直接落札組織構成を実行してサービスが提供された．これらの処理において 1 回目（フィードバック前）の FVCS のサービス提供までの組織構成時間と 2 回目（フィードバック後）の組織構成時間を比較した結果（5 回施行時の平均）を**図 11** に示す．

この図からすべてのマシンにおいて，フィードバック後のエージェント組織構成時間がフィードバック前より短縮されることが分かった．これはフィードバック後のエージェント群が，以前の組織構成情報を利用した組織構成を行っているためであり，動作結果を利用することにより組織構成の処理が効率化されるからである．以上の結果から，従来のフレームワークでは契約ネットプロトコルを用いた組織構成のみをサポートするため，組織構成時間は毎回変化しないのに対し，AAR 型マルチエージェントフレームワークではフィー

```
(historyusingknowledge FVCSKnowledge
 (knowledge
  (rule speedup-organization
   (data :RNAME VideoConfManager
         :USERNAME ?uname
         :PLACENAME ?place
         :OCDesc (Members :manager "(interface)"
                         :member1 ?m1 :member2 ?m2
                         :member3 ?m3 :member4 ?m4
                         :member5 ?m5 :member6 ?m6)) = ?agent1
   (data :RNAME ?m1 :PLACENAME ?place :USERNAME ?uname) = ?agent2
   (data :RNAME ?m2 :PLACENAME ?place :USERNAME ?uname) = ?agent3
   (data :RNAME ?m3 :PLACENAME ?place :USERNAME ?uname) = ?agent4
   (data :RNAME ?m4 :PLACENAME ?place :USERNAME ?uname) = ?agent5
   (data :RNAME ?m5 :PLACENAME ?place :USERNAME ?uname) = ?agent6
   (data :RNAME ?m6 :PLACENAME ?place :USERNAME ?uname) = ?agent7
   -->
   (makelink direct-award ?agent1:RNAME  (?m1 ?m2 ?m3 ?m4 ?m5 ?m6))
  )

  (rule change-rule-order
    (data :RNAME Vic
          :USERNAME ?uname
          :PLACENAME ?place
          :mostFiredRuleName ?m1
          :secondFiredRuleName ?m2) = ?agent1
    (data :RNAME CPUcheck
          :USERNAME ?uname
          :PLACENAME ?place
          :mostFiredRuleName ?m3) = ?agent2
    -->
   (changeruleorder ?agent1:RNAME (?m1 ?m2))
   (changeruleorder ?agent2:RNAME (?m3))
  )
 )
)
```

図 10 実験で使用した動作結果利用知識の例
Fig. 10 Example of experimental utilization knowledge.

	1回目（契約ネットプロトコルを用いた組織構成）	2回目（組織構成情報を活用した組織構成）
マシンA	18529(msec)	14842
マシンB	21458	17374
マシンC	22867	17246
マシンD	24080	20748

図 11 FVCS エージェント群の組織構成時間
Fig. 11 Organization time of the FVCS agent.

ドバック後のエージェント群による組織構成が効率化され，組織構成の処理時間が短縮されることを検証した．このように，マネージャ（組織構成）エージェントがコントラクタ（落札者）エージェントの情報を保持し，これを次回の組織構成において活用することにより，契約ネットプロトコルのみを利用する組織構成に比べ，効率の良い組織構成が行える．

（Ex.2）推論サイクルの効率化によるサービスの高度化の評価

ワークプレースで活動した FVCS エージェント群のうち，CPUcheck エージェントと Vic エージェントのルール発火回数を利用して，エージェントのルール優先順序を調整し，推論サイクルの効率化を図った．評価方法として FVCS のビデオ会議時に使用される動画パラメータのうち，画像の滑らかさを表す fps（Frame

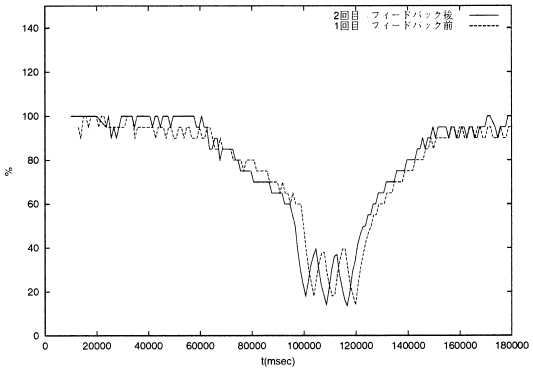

図 12 FVCS における CPU 負荷の変動による fps の制御（マシン A）
Fig. 12 Control of fps against change of CPU load by FVCS with machine A.

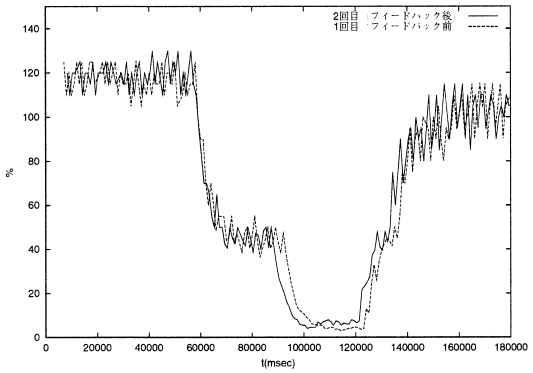

図 14 FVCS における CPU 負荷の変動による fps の制御（マシン C）
Fig. 14 Control of fps against change of CPU load by FVCS with machine C.

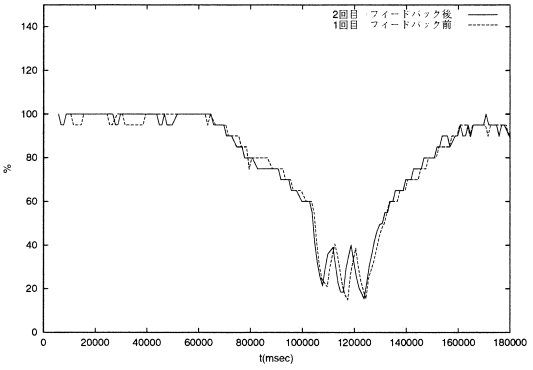

図 13 FVCS における CPU 負荷の変動による fps の制御（マシン B）
Fig. 13 Control of fps against change of CPU load by FVCS with machine B.

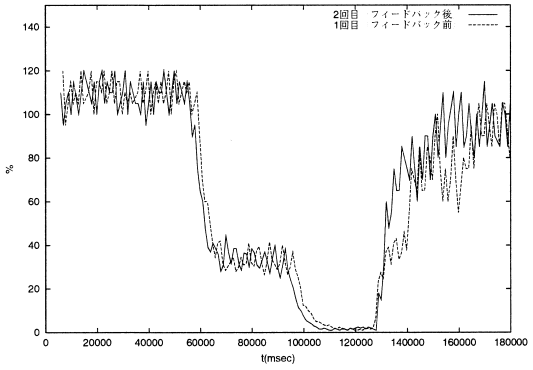

図 15 FVCS における CPU 負荷の変動による fps の制御（マシン D）
Fig. 15 Control of fps against change of CPU load by FVCS with machine D.

per Second）に着目し，各マシンの CPU 負荷変動に対する fps 値の変化を測定した．Vic エージェントは CPU のスレッショルド（しきい値）を User エージェントから獲得し，CPU の負荷がスレッショルドを超えると，提供している動画パラメータを調整することで CPU の負荷を下げるように動作する．そこで FVCS を動作させて安定させた後，1 分間高負荷をかけてスレッショルドを超える負荷を与え，1 分後に負荷を解除するという設定のもとに fps の変動を測定した．以上の設定は FVCS の 1 回目，2 回目の動作ともに同じとした．**図 12**，**図 13**，**図 14**，**図 15** にマシン A，B，C，D におけるそれぞれの測定結果を示す．ここで X 軸は経過時間（msec），Y 軸は利用者要求に対する提供パラメータ fps の QoS（提供中の fps 値/利用者の要求 fps 値．単位は%）を示す．グラフの 60,000 msec で CPU 負荷をかけると Vic エージェントが CPU 負荷を下げるように働きかけ，fps の値を徐々に下げていく．CPU 負荷を解除した状態（120,000 msec）になると，今度は fps を利用者の要求値まで徐々に上げるように働きかける．4 つのグラフのいずれも 2 回目（フィードバック後）の FVCS のエージェント群が 1 回目（フィードバック前）と比較して CPU 負荷に対して fps を即応的に制御していることが分かる．これは即応性の面でフィードバック後のエージェント群のサービス提供がフィードバック前と比較して高度化されていることを意味する．この理由として，フィードバックしたエージェントの推論サイクルにおいて，Vic エージェントは fps を調整するルールが優先的に発火するように変更されたこと，同様に CPUcheck エージェントでは，CPU の変動を監視するルールが優先的に発火するように変更されたことによるものである．

実験結果から，フィードバック後のエージェントの

ルール知識の順序を変更することで，マルチエージェントシステムとしてのサービス提供の質的向上が可能となることを確認した．このように，従来のフレームワークではエージェントのルール発火順序は変化しないため，サービス提供時の処理時間が変化しないのに対し，AAR 型マルチエージェントフレームワークではフィードバック後のエージェントの推論サイクルを効率化し，マルチエージェントシステム全体としての処理効率も向上することが示された．

一般的に，ルール型の知識機構を持つエージェントにおいて，エージェントの推論処理の動作結果を利用して，発火頻度の高いルールが優先的に利用されるようにルールの照合順序を変更することにより，次回の利用時の推論処理が短縮され，システム全体としての即応性も改善される．本論文では，エージェントの推論機構を単純なものに保ちつつ，エージェントの推論処理効率を向上させるために，上記の考え方に基づく手法を採用している．

以上の結果により，本論文で提案した Active Agent Repository を備えた AAR 型マルチエージェントフレームワークは，従来のフレームワークに比べて，効率的なエージェント組織構成と高度なサービス提供が可能となることが検証された．また，本実験を通して，様々なプラットフォーム環境においても，リポジトリで動作結果を活用することにより，マルチエージェントが提供するサービスの高度化が実現できることを示した．

5.3 関連研究との比較

はじめに本研究を既存のエージェントフレームワークと比較する．現在，分散環境で利用可能なマルチエージェントフレームワークとして Aglets [7]，Plangent [8]，Odyssey [9]，Voyager [10]，Kafka [11]，Concordia [12]，AgentSpace [13] 等がある．しかしながら，これらのフレームワークは動作終了後のエージェントの動作結果（動作履歴）を系統的にマルチエージェントシステムに反映させる機構を備えていないため，マルチエージェントシステム全体としてシステムを自動的に調整したり，適応化したりする機能は実現されていない．

これに対して，AAR 型マルチエージェントフレームワークは，分散環境でのエージェントの永続性を保障し，動的なエージェントシステムの生成・協調・フィードバック・再利用を提供してマルチエージェントのライフサイクルを包括的に支援する．さらに，エージェントリポジトリに動作結果を活用する新たな機構を導入することで，エージェントの動作結果を利用して，マルチエージェントシステム全体を系統的に調整・変更し，その効率化・高度化を図ることが可能となる．

次に，既存のエージェントの学習[15)~17)]と，本研究について比較する．本研究ではエージェントの学習に焦点を当てるのではなく，系全体（マルチエージェントシステム全体）に動作結果から得られる情報を反映させて，再びそのシステムが呼び出されたときに，システム全体の動作効率を改善しようとする点に特徴がある．したがって，個々のエージェントは必ずしも学習する必要はなく，むしろフレームワークのリポジトリがシステムの動作結果を積極的に活用することにより，リポジトリ内でのサービス構成の調整や，生成されたサービスの効率化や高度化を目指すものである．このように，本論文で提案したリポジトリ機構と，これに基づくエージェントフレームワークは，マルチエージェントシステム全体としてのサービス提供の効率化や高度化を指向する際の新しいフレームワークを提供する．

6. むすび

本論文では，動作結果を利用したマルチエージェントの効果的なサービス構成・調整を実現するために，リポジトリ型マルチエージェントフレームワークにおけるリポジトリ機構を拡張した Active Agent Repository を提案し，その設計とプロトタイプの実装を行った．そして，検証実験を通して，Active Agent Repository の導入によってマルチエージェントの組織構成の効率化・サービス提供の高度化が可能となることを示した．

リポジトリ機構の抱える課題としては，フィードバック後のエージェント数の増加によりリポジトリシステムの負荷が増大することがあげられる．この課題の解決法として，マルチリポジトリを用いたリポジトリ間での協調による負荷分散を検討中である．また，動作結果データとして，ユーザ名等の個人情報を利用しているため，プライバシ上の問題が存在するが，この解決法としてリポジトリ/ワークプレース間での通信時のアクセス制御および，動作結果データに関するアクセス制御等を検討している．

今後の予定として，動作結果利用機構に動作結果利用テンプレートを導入することにより，エージェントシステム設計者の動作結果利用知識の記述支援を行う予定である．また現状では，エージェント設計者が動作結果利用知識を明示的に記述する方法をとっているが，今後，有用な動作結果利用知識を獲得・蓄積し，これを利用した支援機能をリポジトリに組み込んでゆくことも検討したい．さらに，AAR を改良/機能拡

張することにより，ユーザ情報に応じたサービスのカスタマイズ化やプラットフォーム環境の情報に応じた環境に適応したサービス提供等を実現していく予定である．

参 考 文 献

1) 藤田　茂，菅原研次，木下哲男，白鳥則郎：分散処理システムのエージェント指向アーキテクチャ，情報処理学会論文誌，Vol.37, No.5, pp.840–852 (1996).
2) Fujita, S., Hara, H., Sugawara, K., Kinoshita, T. and Shiratori, N.: Agent-Based Design Model of Adaptive Distributed System, *Applied Intelligence*, Vol.9, No.1, pp.57–70 (1998).
3) 原　英樹，今野　将，菅原研次，木下哲男：ソフトウェアエージェント開発用教育用システム TAF の設計と実装，ソフトウェアエージェントとその応用特集ワークショップ（SAA2000）講演論文集，pp.183–190 (2000).
4) 菅沼拓夫，藤田　茂，菅原研次，木下哲男，白鳥則郎：マルチエージェントに基づくやわらかいビデオ会議システムの設計と実装，情報処理学会論文誌，Vol.38, No.6, pp.1214–1224 (1997).
5) Suganuma, T., Lee, S., Kinoshita, T. and Shiratori, N.: An Agent Architecture for Strategy-centric Adaptive QoS Control in Flexible Videoconference System, *Next Generation Computing*, Vol.19, No.2, pp.173–191 (2001).
6) Uchiya, T., Katoh, T., Suganuma, T., Kinoshita, T. and Shiratori, N.: An Architecture of Agent Repository for Adaptive Multiagent System, ICOIN-16, pp.7A-4.1–7A-4.12 (2002).
7) Lange, B.D. and Ohshima, M.: *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley, ISBN 0201325829, (Aug. 1998).
8) Ohsuga, A., Nagai, Y., Irie, Y., Hattori, M. and Honiden, S.: Plangent: An Approach to Making Mobile Agents Intelligent, *IEEE Internet Computing*, Vol.1., No.4, pp.50–57 (1997).
9) General Magic Inc., Odyssey Web Page (1998).
http://www.genmagic.com/agents/odyssey.html
10) ObjectSpace Inc., ObjectSpace Voyager Technical Overview (1997).
http://www.objectspace.com/Voyager/
11) Kafka: Yet Another Multi-Agent Library for Java.
http://www.fujitsu.co.jp/hypertext/free/kafka/
12) Mitshbisi Electric Information Technology Center. Concordia.
http://www.meitca.com/HSL/Projects/Concordia/
13) 佐藤一郎：AgentSpace：モーバイルエージェントシステム，日本ソフトウェア科学会 Workshop on Muti Agent and Cooperative Computation, (MACC'98) (Dec. 1998).
14) Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver, *IEEE Trans. Comput.*, Vol.29, No.12, pp.1104–1113 (1980).
15) 山村雅幸，小林重信：エージェントの学習，人工知能学会誌，Vol.10, No.5, pp.683–689 (1995).
16) 宮崎和光，荒井幸代，小林重信：Profit Sharing を用いたマルチエージェント強化学習における報酬配分の理論的考察，人工知能学会誌，Vol.14, No.6, pp.1156–1164 (1999).
17) 松浦賢一，嘉数侑昇：非均質エージェント系における組織的行動の形成：2 次元追跡問題における考察，情報処理学会論文誌，Vol.38, No.6, pp.1083–1093 (1997).

(平成 14 年 7 月 9 日受付)
(平成 14 年 11 月 5 日採録)

打矢　隆弘

1976 年生．2001 年東北大学大学院情報科学研究科博士前期課程修了．現在，同大学院同研究科博士後期課程在学中．エージェント指向設計開発方法論，エージェント指向コンピューティングに興味を持つ．電子情報通信学会学生会員．

武田　敦志

1977 年生．2002 年東北大学大学院情報科学研究科博士前期課程修了．現在，同大学院同研究科博士後期課程在学中．エージェント指向コンピューティングに興味を持つ．

菅沼　拓夫（正会員）

1966 年生．1997 年千葉工業大学大学院博士後期課程情報工学専攻修了．現在，東北大学電気通信研究所助手．博士（工学）．やわらかいネットワーク，エージェント指向コンピューティングに興味を持つ．IEEE，電子情報通信学会各会員．

木下 哲男（正会員）

1953 年生．1979 年東北大学大学院修士課程修了．同年，沖電気工業（株）入社．1996 年東北大学電気通信研究所助教授．2001 年同大学情報シナジーセンター教授．知識工学，エージェント技術，知識情報システム等の研究開発に従事．工学博士．1989 年度情報処理学会研究賞，1996 年度同論文賞，2000 年度電子情報通信学会業績賞各受賞．電子情報通信学会，人工知能学会，日本認知科学会，AAAI，IEEE，ACM 各会員．