知能と複雑系　125－7
(2001. 7. 24)

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　菅沼拓夫　木下哲男　白鳥則郎

東北大学電気通信研究所/情報科学研究科　〒980-8577 仙台市青葉区片平 2-1-1
Tel: 022-217-5454. E-mail: {takahiro,suganuma,kino,norio}@shiratori.riec.tohoku.ac.jp

あらまし　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・調整を行うための手段として，エージェントの動作結果を効果的に利用することが挙げられる．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，それをシステム動作全体に効果的に反映させ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し，上記の問題の解決を目指す手法を提案する．本稿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．

キーワード　エージェント　マルチエージェントシステム　エージェントフレームワーク

Design and Implementation of Repository Architecture on Agent Framework

Takahiro Uchiya, Takuo Suganuma, Tetsuo Kinoshita, Norio Shiratori

Research Institute of Electrical Communication, Tohoku University Sendai 980-8577 Japan.
Tel: +81-22-217-5454. Email:{takahiro,suganuma,kino,norio}@shiratori.riec.tohoku.ac.jp

Abstract　It is important to use agent's behavioral history of a multiagent system effectively as a way to realize advanced service organization /adjustment 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 in the agent workplace. We focus on a repository-based multiagent system that carries out the organization and reorganization of agents using the repository, and we propose a new method and a new function of repository that utilizes behavioral history to deal with the above problem. In this paper, we explain 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.

keywords　Agent, Multi Agent System, Agent Framework

1. まえがき

インターネットをはじめとする広域分散環境においては，一般にユーザの要求や環境は多種多様であり，またネットワークやプラットフォームの資源も有限でかつその機能的な変動も大きい．こうした複雑な利用者要求/環境の変化に対応するための新しいシステムとしてマルチエージェントシステムの枠組みがある[1,2]．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）を行うことで動的にサービス機能を構成・調整し，ユーザにサービスを提供するシステムと捉えることができる．

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・調整を行う上で，エージェントの動作結果を効果的に利用することが挙げられる．例えば，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，当該環境に適応するが，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，その動作結果を活用することができれば，再び同様の環境でビデオ会議サービスが要求されたとき，システムを構成するマルチエージェントの組織化や初期パラメータの決定などの処理を簡略化することが可能となる．

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でのタスク遂行後に，その動作履歴等を保持・管理し，それをシステム動作全体に効果的に反映させ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たとしても，個々のエージェントの動作結果を系統的に利用/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している．

これまで，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を開発してきた．リポジトリ型マルチエージェントフレームワークとは，サービス機能を実現するコンポーネント群をサーバ（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を可能とするエージェントフレームワークである[1,2]．そこで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，環境に適応してサービス組織構成を効果的に行うマルチエージェントシステムを構築することができる．

本稿で提案する手法では，マルチエージェントの組織構成の効率化を図るために，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する．具体的には，リポジトリ機構に対して，以下の3つの機構を導入する．

(M1)エージェントライフサイクル管理機構

一度インスタンシエートした動作環境にて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．

(M2)動作結果利用機構

(M1)と連携して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を調整し，サービス提供の効率を向上させる機構．

(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)を組みこんだ新たなリポジトリ機構を Active Agent Repository(AAR)と呼ぶ．AAR は，活動後のエージェントの動作結果をリポジトリ内に保存し，そこから得られる知識を以降の組織構成時に利用する．これによりエージェントは，過去の動作結果の経験を活かして，より効果的に組織構成を行うことが可能となる．

本稿では 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．すなわち，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し，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 リポジトリ型フレームワーク

本提案の基盤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の事例である 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．

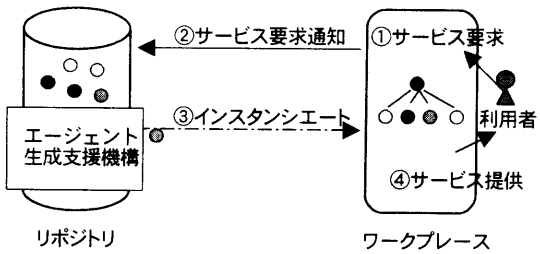

図1 ADIPSフレームワーク

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タスクとして送信する．

リポジトリ内では，与えられたタスクに対しエージェント相互の協調動作により，利用者の要求や環境に適したエージェント群が決定される．これらのエージェントはワークプレースと呼ばれるエージェント動作環境に生成(インスタンシエート)される．こうした処理を実行する仕組みがリポジトリに組み込まれているエージェント生成支援機構であり，これにより契約ネットプロトコルに基づく組織構成などを行うことができる．

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挙げられる．すなわち，リポジトリ内のエージェントに対する調整・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結果を，それらの動作後にリポジトリにフィードバックし，これを集中的に管理することにより，必要に応じてこれらの情報を加工して，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できる．更に，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可能性がある．

2.2 Active Agent Repository の概念

マルチエージェントの組織構成機能の高度化を目指して，本稿では，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．

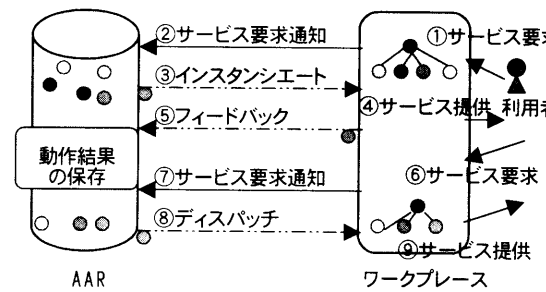

図2 AAR 型マルチエージェントフレームワーク

本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抽出・加工し，それをエージェントの知識として追加する．これにより過去に提供したサービスと同様の条件にて再度サービス要求通知が行われた場合に，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素早く選択して送出(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 はエージェント生成支援機構(ACSM)，エージェントライフサイ

クル管理機構(ALMM)，動作結果利用機構(HUM)，およびリポジトリ管理エージェント(RMA)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．

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が担当していたエージェント組織構成・再構成とインスタンシエーションを行う．

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てリポジトリに復帰したエージェントを保持すると共に，それ以降のエージェントのライフサイクルを管理する．

動作結果利用機構(HUM): 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共に，それを抽出・加工してエージェントの知識として追加し，エージェント組織構成の処理が高速化されるようにエージェントの調整を行う．

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分ける役割を持つ．

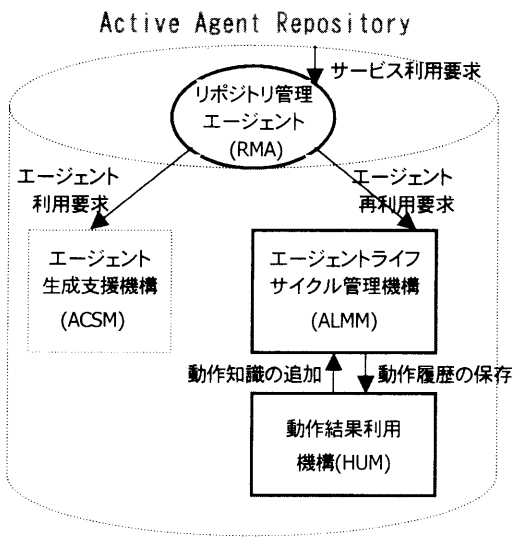

図 3 AAR の機能構成

3. Active Agent Repository の設計

2 章で提案した AAR の各機構とエージェントの設計について述べる．

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．HUM 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．

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してくると，エージェントライフサイクル管理機構(ALMM)から HUM 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，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エージェントの組織構成に関する情報，すなわち，エージェントを再度呼び出して利用しようとする際に使用する組織構成情報を抽出する．本推論エンジンでは，ルール型知識に基づく推論機構を採用している．

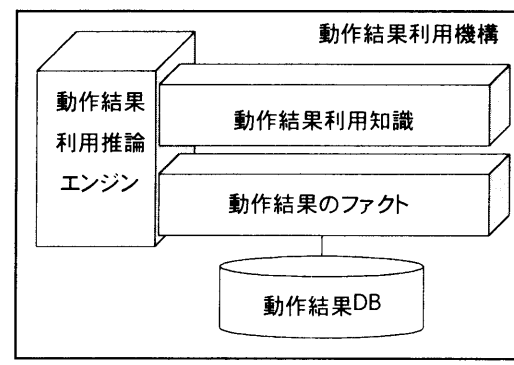

図 4 動作結果利用機構

3.1.1 動作結果データベース

動作結果利用機構(HUM)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．

```
HD = 〈 RD, HDB 〉
RD = リポジトリ記述
HDB = { hd | hd =〈hddesc, aname, wpname,
      uname, OCDesc, WM, WMH, MH 〉}
hddesc = データメタ記述
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
```

図5 動作結果データ

3.1.2 動作結果利用推論エンジン

3.1.1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6に動作結果利用知識の例を示す．ここでは複数のエージェントのうち機能としての性能が良かったものを再利用する際の知識の一部が記述されている．このルールは，data という識別名を持つ3つの条件で示される性能を満たすエージェント群を選択するものである．

```
(historyknowledge huk001
(knowledge
(rule rule1
(data :AGENTNAME video :wm.performance max )
= ?data1
(data :AGENTNAME audio :wm.performance max)
= ?data2
(data :AGENTNAME vcm :USERNAME ?u
      :PLACENAME ?p ) = ?ma1
-->
  (makelink huk001 ?ma1 ?data1 ?data2))))
```

図6 履歴利用知識の設計

動作結果推論エンジンは，ルールとファクトをパターンマッチングしてアクション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，HUM機能を実現するために以下の2種類のアクションを備えている．

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 ･･･)

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，動的にこの組織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，エージェントの振る舞いや初期パラメータの変更を行うことができる．

3.2 エージェントライフサイクル管理機構

ワークプレースにインスタンシエートされ，その後，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がエージェントライフサイクル管理機構(ALMM)により管理される．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するためには，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，復帰したエージェントが再利用されるまでの期間，動作した結果を保存したり，知識を調整するために協調動作するための基盤環境を提供する．

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機能である．具体的には，利用されなくなったエージェントを整理し，また同一の知識を持ったエージェントを統合する機能等を有する．

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)の主要な役割は，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機能することである．以下にRMAの機能を挙げる．

(1)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．

(2)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い合わ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．

(f3)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．

RMAは，ワークプレースから送られたサービス要求とエージェント組織構成を連結する手段となる3種類のテーブルを保持する．すなわちエージェント名前管理テーブル(AMT)，サービス使用状況管理テーブル(SMT)およびパフォーマティブ管理テーブル(PMT)である．図7に各々の概要を示す．また，図8にRMAの動作アルゴリズムを示す．RMAは要求メッセージのパフォーマティブにより要求を判断し，かつSMTを利用して以前利用したサービスかどうかを判断することができる．

```
AMT = { a_desc | a_desc = < aname, aid> }
 aname = エージェント名
 aid = エージェント ID

SMT = { s_desc | s_desc = < sname, uname,
          wpname, oinfo > }
 sname = サービス名
 uname = ユーザ名
 wpname = ワークプレース名
 oinfo = 組織構成情報

PMT = { a_perf }
 a_perf = 使用可能なパフォーマティブ
```

図7 RMAが保持するテーブル

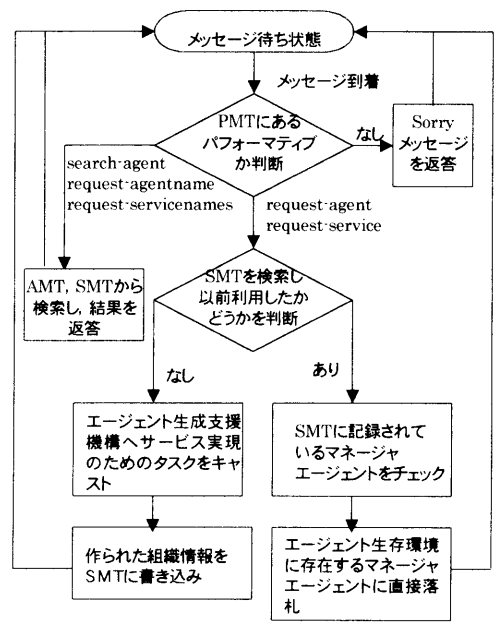

図8 RMAの動作アルゴリズム

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 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るため，分散環境での利用は想定されていない．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 章の設計に基づき，リポジトリに機能拡張を施し AAR を実装した．HUM の推論機構については TAF の推論機構をそのまま利用した．また，知識記述についても基本的には 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 クラスである．

5. 実験と考察

5.1 評価実験

評価実験用の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 Video Conference System)[4,5]を選択し，(Ex.1)動作検証と(Ex.2)エージェント組織構成の効率化の評価を行った．(Ex.1)では，以前利用したエージェントを動作終了後リポジトリにフィードバックし，再度利用要求が発生した時にリポジトリからディスパッチして再使用できるかどうかを確認することにより，設計と実装の妥当性を検証した．(Ex.2)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，次回の同じユーザからの同種のサービス要求に対して，その実行時間が短縮されるかどうかを検証した．

5.1.1 実験環境

FVCS のハードウェア環境としては，100BaseT の HUB で接続された Sun Microsystems 社の Ultra SPARCStation 3 台を用いた．これらの上に，Java 言語で実装された本フレームワーク（AAR 型マルチエージェントフレームワーク）を実装した．SPARCStation の 1 台は Active Agent Repository として，残りの 2 台はワークプレース{A, B}として動作させた．

5.1.2 実験条件

Active Agent Repository で使用した動作結果利用知識の例を図 9 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「あるワークプレースにおける最終的な FVCS の組織構成を動作結果データとして利用して，次回の同じユーザの要求の際の組織構成時に契約ネットプロトコルを使用せずに direct-award で組織構成することでエージェント組織化の速度を向上する」という知識の一部が示されている．

```
(historyusingknowledge FVCSKnowledge
(knowledge
 (rule speedup-organization
 (data :AGENTNAME VideoConferenceManager
     :PLACENAME ?place :USERNAME ?user) = ?data1
(data :AGENTNAME Video
 :PLACENAME ?place :USERNAME ?user) = ?data2
 .......
 ->
(makelink direct-award ?data1 ?data2 ?data3 ··)))
```

図 9 実験で使用した FVCS の履歴利用知識

5.1.3 動作実験

以下の手順に従って実験を行った．

(1) ワークプレース A，B 間でビデオ会議システム（FVCS）を起動する．始めに利用者はワークプレースを起動し，インタフェースに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リポジトリに通知され，リポジトリで FVCS の組織構成が行われ，エージェント群がワークプレースにインスタンシエートしてサービスを開始する．

(2) ビデオ会儀終了後，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) 再びワークプレース A，B 間でビデオ会議を行うようサービス要求を行う．

5.2 実験結果と評価

（1）動作検証

FVCS の終了と共にエージェントが活動を停止し，

FVCSのエージェント群がリポジトリにフィードバックされた．エージェント内部のワーキングメモリを含む実行結果は動作結果データベースに保存された．次に，再びFVCSを起動すると，動作結果利用知識により，以前活動したエージェント群に対して組織構成の要求がdirect-awardで通知され，各々のエージェントは以前のワーキングメモリの状態を伴って，再びワークプレースに生成され，ビデオ会議を行うことができた．以上により設計と実装の妥当性を確認した．

（2）エージェント組織構成の効率化の評価

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，そして，動作結果利用知識を利用して，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マネージャ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がサービス使用状況管理テーブルを用いて以前利用したサービスであることを認識し，ALMM内のマネージャエージェントにタスク通知し，マネージャエージェントが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回目のFVCSのサービス提供までの組織構成時間と2回目の組織構成時間を比較した結果，前者は平均17.8秒，後者は平均13.6秒であった．即ち，2回目の組織構成において，過去の動作結果を利用した組織構成により高速化が図られていることを確認した．

以上の結果により，本稿で提案した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，ユーザ要求/環境に適合した効果的なエージェント組織構成が可能となることを検証した．

6. むすび

本稿では，マルチエージェントの組織構成の効率化を実現するために，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が可能となることを示した．

今後の課題として，

- 動作結果利用テンプレートの導入
- エージェントの動作結果利用知識をエージェント自身が自律的に獲得/運用していく仕組み

などの検討が残されている．

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] Shigeru Fujita，Hideki Hara，Kenji Sugawara，Tetsuo Kinoshita，Norio Shiratori，"Agent-based Design Model of Adaptive Distributed Systems", The International Journal of Artificial Intelligence，Neural Networks，and Complex Problem-Solving Technologies，vol.9，No.1，pp.57-70，July/August (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] Takuo Suganuma，SungDoke Lee，Tetsuo Kinoshita and Norio Shiratori，"An Agent Architecture for Strategy-centric Adaptive QoS Control in Flexible Videoconference System", Next Generation Computing，Vol.19，No.2，pp.173-191，2001.