F-010

ユビキタス環境用帰還型エージェントフレームワークの設計

Design of Circulation Agent Framework for Ubiquitous Environment

伊藤 翔太† 打矢 隆弘† 内匠 逸†
Shota Ito Takahiro Uchiya Ichi Takumi

1. はじめに

近年，インターネット等の広域分散環境が急速に発展しており，ユーザからのサービス要求やサービスの利用形態は多様になっている．この環境下において，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，エージェントシステム(以降AS)の適用が注目されている．

本研究では，数あるエージェントフレームワークの中から DASH フレームワークに着目している．DASH フレームワークは，エージェントフレームワークの特徴として，リポジトリを用いた動的な組織構成を行う．すなわち，サービス要求に基づき動的に構成された AS は，利用者とのインタラクションを通じて個々の利用者に徐々に適合してゆく．しかし，サービス提供終了後にメンバ間の契約は解除され，組織自体は消滅する．利用者が再度同一のサービスを要求した場合に，最適化された前回の AS 組織を継続利用できれば，再組織化・最適合のプロセスが省略できる．

そこで本研究では，ユビキタス環境で前回の AS 組織を継続利用するための仕組みとして，ユビキタス環境用帰還型エージェントフレームワークを提案する．これは，DASH フレームワークに改良を加え，AS の帰還機能を追加することで，以下の実現を目的としている．

- 組織構成の効率化
- 帰還エージェントシステムの再利用

本稿では，DASH フレームワーク上で AS を構成する際の動作の非効率性を問題点として挙げ，解決方法としてエージェントへ帰還機能を追加した先行研究について説明する．また，これを元に作成した単一環境用帰還形フレームワークについて説明する．最終的に，先行フレームワークに対する問題点を指摘し，その解決を目的とした提案フレームワークを説明し，動作検証と評価について述べる．

2. 先行研究

2.1 現行のフレームワークの問題点

DASH フレームワークはエージェントが動的に組織化や再構成を行うことを特徴としている．しかし，それらを行う際に行われるメッセージ交換や，入札待ち時間等によって，最終的に必要な組織を得られるまでに長い時間を要する場合がある．

また，DASH フレームワークの仕様により，一度利用が終了したエージェントは機能が停止または待ち状態となり，ワークプレースの動作終了時にその環境で利用されていたエージェントの情報はすべて破棄される．したがって，ユーザが再度同一のサービスを利用する場合，エージェントは再び同じような組織化・再構成を行う必要がある．

更に，DASH エージェントが一度経験した事象を記憶しそれを元に学習を行う場合は，エージェントの設計者が情報を何らかのファイルとして保存するように設計する必要があり，フレームワークとしてエージェントの活動で得たデータを保存する機構を持たない．

2.2 先行研究

先行研究[1][2]では，リポジトリ型エージェントフレームワークに対する永続性に関する設計を行っている．リポジトリ型エージェントフレームワークを基盤として，ユーザに適応したエージェントシステムに対して，それをリポジトリにフィードバック(帰還)させて保持し，永続性を保証する機構を導入する．これにより以前利用したエージェントシステムを異なる環境で再利用することが可能となる．

また，リポジトリ部に使用後のエージェントシステムを保持/管理する機構(エージェントライフサイクル管理機構)及び，サービス要求とリポジトリ内のエージェント群を適切に組み合わせる機構(ユーザ指向サービス調整機構)を新たに導入している．

2.2.1 エージェントライフサイクル機構

ワークプレースでサービスを終了した AS をリポジトリに帰還させ，その後のライフサイクルを管理する．具体的にはエージェントのフィードバック，待機状態，再生成，リポジトリ内での協調的活動を包括的に支援する．

2.2.2 ユーザ指向サービス調整機構

ユーザが他の環境（ワークプレース）に移動して再度適応した AS を使用したい場合には，ユーザはワークプレースを介してサービス再利用要求を通知する．再利用要求を受け取ったリポジトリは，リポジトリ内の以前利用した AS にタスクを依頼する必要がある．ここでユーザの移動先が移動前の環境と同様の環境の場合，そのエージェントシステムは有効に動作する．しかし，環境が大きく異なる場合は，以前使用したエージェントシステムを再使用すると反って適応度が減少する恐れがある．そこで，この機構では環境の差異を判断し，性能低下に起因するエージェントを置換するタスクを生成する等のサービス調整を行う．

2.3 単一環境用帰還型フレームワーク

上述の先行研究をモデルとして，AS を帰還させる動作を備えたフレームワーク（単一環境用帰還型エージェントフレームワーク）のプロトタイプが開発された（図 1）．このプロトタイプでは，帰還したエージェントシステムを管理する機構は設計されておらず，ワークプレースに生成されたエージェントをリポジトリに帰還させる動作

†名古屋工業大学 大学院 工学研究科 情報工学専攻
Nagoya Institute of Technology

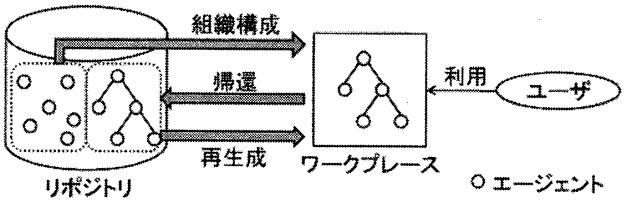

図 1　単一環境用帰還形フレームワーク概略図

及びリポジトリに帰還したエージェントを再度ワークプレースに移動させる動作を，フレームワーク自体に埋め込んだ形となっている．

2.3.1 設計

単一環境用帰還型エージェントフレームワークは，DASH フレームワークの改良と，ルールセットファイル（Feedback ルールセット）の追加がなされている．帰還や再生成を行うエージェントはエージェント設計者が決定し，このルールセットをエージェントにインクルードすることでそれらの動作が可能となる．

以降，プロトタイプの帰還・再生成の各動作について説明し，問題点を指摘する．

2.3.2 帰還動作

ユーザがワークプレース上の UI から AS の帰還を要求した場合，同ワークプレース内のエージェント全体に対して，save-service メッセージが送信される．帰還可能なエージェントは Feedback ルールセットが適用されており，同ルールセット内の帰還ルールが発火することでリポジトリへの移動を開始する．移動したエージェントには saved-service ファクトが保存され，対象エージェントが帰還エージェントであると判断可能な状態を作る．

2.3.3 再生成動作

ユーザがワークプレース上の UI から AS の再生成を要求した場合，ワークプレース側の設定ファイルによって静的に指定されたリポジトリ全体に対して，restart-service メッセージが送信される．再生成可能なエージェントは帰還と同様に Feedback ルールセットが適用されており，同ルールセット内の再生成ルールが発火する．発火する対象のエージェントは，帰還動作で付加された saved-service ファクトをワーキングメモリに保持しているエージェントである．

3. ユビキタス環境への適応手法の提案

3.1 先行プロトタイプの問題点

前節で述べたプロトタイプには，先行研究を実装する理由となった現行のフレームワークの問題点をすべて解決していない．これは，おもにエージェントの帰還機構がユビキタス環境に適応していないということを中心に見て取れる．これらの問題点を，プロトタイプを作成したことによる新たな問題点と合わせ，以下に列挙する．

- リポジトリの負荷の増大
  帰還 AS は，組織を成す前のエージェントと同一のリポジトリ内に保存される為，帰還を重ねる度にデータ量が増加する．長期的にシステムを利用することを考えると，こういったリポジトリの負荷も考慮にいれる必要がある．
- 帰還した AS の効率的な管理が困難
  帰還 AS を整理する機構が存在しないために，対象となり得る全てのエージェントに対して帰還・再生成の命令をブロードキャストする方式を取らざるを得ない．これは通信量の観点から，非効率的である．
- 単一環境でのみ動作
  帰還先のリポジトリは，あらかじめ設定ファイル上で静的に記述している．また，リポジトリから再生成する宛先（ワークプレース）も静的に決定している為，実質単一環境用のフレームワークとなっている．ユビキタスな環境に対応するためには，複数のワークプレースで活動するユーザによって帰還・再生成の環境を設定可能である必要がある．

3.2 解決法の提案

図 2 に提案手法の概略図を示す．3.1 で述べたように，先行プロトタイプにはユビキタス環境で利用するにはいくつかの問題点が指摘できる．以下ではこれらの問題点に対する解決法として，２つの実装に大別して説明する．

(F1)　リポジトリの分割

リポジトリへの負荷の軽減策として，リポジトリを分割する．また，この実装によって帰還 AS の効率的な管理も目標としている．現行のフレームワークや，先行プロトタイプで利用されるリポジトリは単一であるが，これを Public リポジトリと Private リポジトリに分割する．Public リポジトリには，従来と同様に組織構成を行う前のエージェントを保存する．それに対して Private リポジトリには，帰還 AS を構成するエージェントを保存する．保存場所を分割することで，帰還動作を繰り返すことによって起こり得る容量面でのリポジトリの圧迫を防ぐことが可能であると考える．

(F2)　帰還 AS 管理エージェント

帰還 AS 管理エージェントは，エージェントライフサイクル管理機構およびユーザ指向サービス調整機構の役割を果たすことを目標としている．現実装段階ではエージェントライフサイクル管理機構の機能のみを備えている．この実装によって，単一環境でのみ動作が可能であったプロトタイプを，複数環境で動作可能にする．帰還 AS 管理エージェントは帰還可能な AS のリスト及び帰還後の AS のリストを保持し，各エージェントの帰還・再生成の可否を判断する．

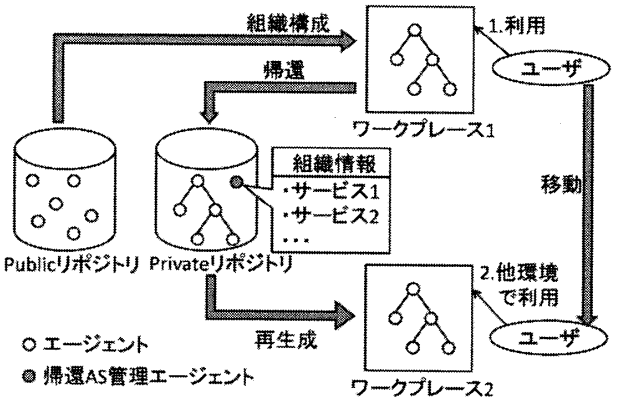

図 2　提案手法の概略図

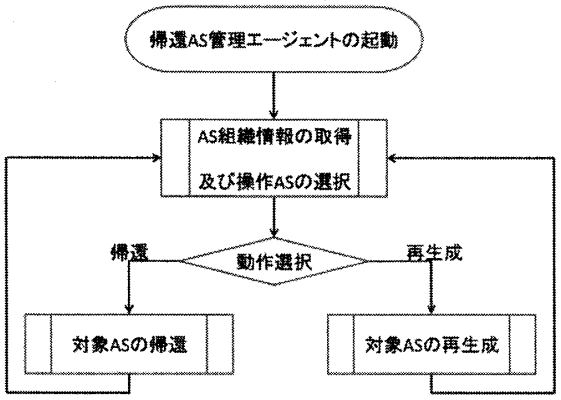

図 3　帰還 AS 管理エージェントのフローチャート

3.3 設計

提案手法におけるリポジトリの分割に関しては，既存のリポジトリシステムを基に設計する．Public リポジトリは，既存のリポジトリを改変することなく利用する．Private リポジトリは，リポジトリの起動時に帰還 AS 管理エージェントを自動で呼び出す仕様とする．

帰還 AS 管理エージェントの動作を図 3 に示す．帰還 AS 管理エージェントが起動すると，ワークプレースのエージェントからのメッセージ待ち状態になる．そのメッセージを AS 組織情報として処理し，ユーザの操作によって帰還もしくは再生成の操作を対象エージェントに行う．操作を行うごとに，再び待ち状態に移行する．以降，フローチャート中の工程を説明する．

3.3.1 帰還 AS 情報の取得

図 3 中の AS 組織情報の取得及び操作 AS の選択の工程について説明する．ワークプレースの起動後，帰還 AS 管理エージェントはワークプレースにインスタンシエートされたエージェントからの組織情報の送信を待つ．その後，ワークプレース側のユーザは任意のタイミングでワークプレース上のインタフェースを介して，帰還 AS 管理エージェントにリクエストを送信する．リクエストを受信した帰還 AS 管理エージェントは，インタフェース用エージェントに移動命令を送信し，ワークプレース側へ移動させる．その後，ワークプレース上でインタフェース用エージェントが帰還可能 AS のリストを表示し，ユーザに選択させる．以上の動作を図 4 に示す．

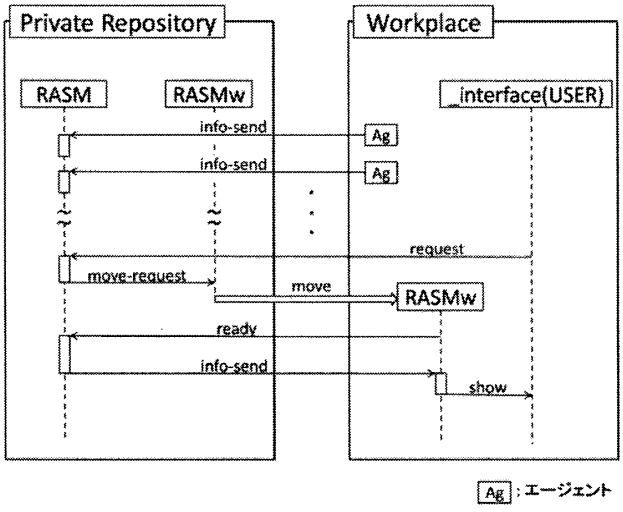

図 4　帰還 AS 情報の取得

3.3.2 帰還操作

帰還操作の流れを図 5 に示す．ユーザが選択した帰還対象の AS に対し，インタフェース用エージェントが帰還命令を送信する．全てのエージェントの帰還が終了すると，インタフェース用エージェントから帰還 AS 管理エージェントへ対象 AS の帰還後の新しいリストを送信し，自身もリポジトリへ帰還する．

3.3.3 再生成操作

再生成操作の流れを図 6 に示す．ユーザが選択した再生成対象の AS に対し，インタフェース用エージェントが再生成命令を送信する．以降帰還操作と同様にリストの処理を行う．

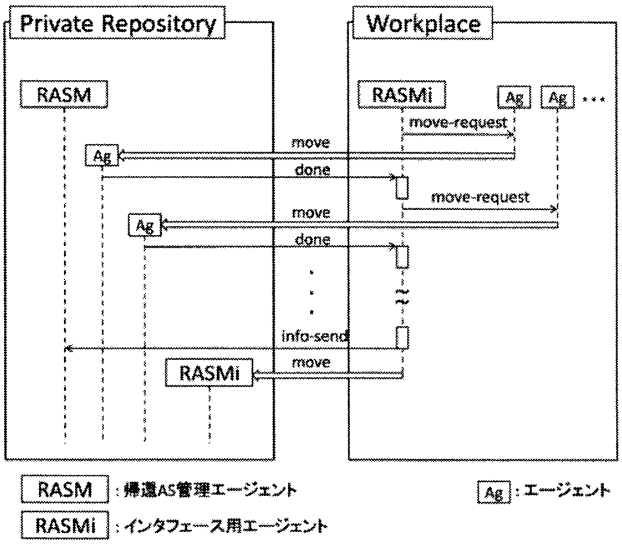

図 5　帰還操作

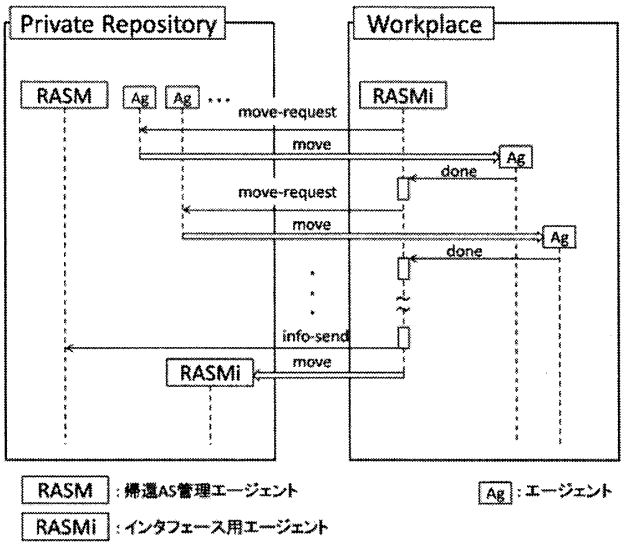

図 6　再生成操作

4. システムの評価

提案プロトタイプの有用性を示すため，以下の実験を行った．

4.1 概要

この実験は組織構成及び帰還・再生成にかかる時間を調査し，帰還・再生成の有用性を示すことを目標としている．時間の調査を行った環境は表 1，表 2 の通りである．

表 1　実験環境(PC1)

OS	Windows Vista Home Basic
プロセッサ	Intel Celeron CPU 440 2.00GHz
メモリ(RAM)	2.00GB
システムの種類	32bit OS
JAVA	Version 1.6.0_14

表 2　実験環境(PC2)

OS	Windows 7 Home Premium
プロセッサ	Intel Core 2 Duo CPU E8500 3.16GHz
メモリ(RAM)	4.00GB
システムの種類	64bit OS
JAVA	Version 1.6.0_14

PC1 環境では，1 台の計算機上でフレームワークの全てを動作させて計測を行う．対して PC2 環境では，スペックの同じ 2 台の計算機を使い，一方にはリポジトリ，もう一方にはワークプレースを動作させ，ネットワークを跨いだシミュレーションを行った．さらに，組織構造の違いによる比較の為，利用するエージェントシステムとして以下の 2 種類を採用した．

- SocialWare
  タスク通知によって組織を構成する．10 種類のエージェントによる大規模な組織モデルを持つ．
- OnsenSeries
  直接落札によって組織を構成する．通常ワークプレースで動作するエージェントは 3 種類だが，組織再構成を行う事で特定のエージェントを入れ替える動作を行う．

尚，計測は各動作に対して 10 回行い，その加算平均を単位 ms まで計測する．

4.2 結果

実験により得られた結果を表 3，表 4 に示す．例としてエージェントシステムを SocialWare，環境を PC1 とすると，組織化に必要な 24558ms に対して帰還と再生成の合計が 10414ms 下回る 14144ms を記録している．同様に，PC2 では 12371ms に対して 8822ms 下回る 3549ms である．OnsenSeries では組織化と再構成の合計を見ると，PC1 では組織構成の 30302ms に対して帰還・再生成が 5732ms，PC2 では 27600ms に対して 7545ms と，大きく下回っている．

表 3　組織構成時間の合計(SocialWare)

		PC1	PC2
動作時間(ms)	組織化	24558	12371
	帰還	6794	1930
	再生成	7350	1619
	帰還＋再生成	14144	3549

表 4　組織構成時間の合計(OnsenSeries)

		PC1	PC2
動作時間(ms)	組織化	15858	13316
	再構成	14444	14284
	組織化＋再構成	30302	27600
	帰還	2784	3699
	再生成	2948	3846
	帰還＋再生成	5732	7545

4.3 考察

以上の結果より，組織構成に必要な時間に対して帰還・再生成に必要な時間は多くの場合で下回っている．また，今回実験を行った環境下では，動作環境や AS によらず，時間の短縮が確認できる．よって，組織を再度利用する場合は再度組織構成を行うより，提案手法によって帰還・再生成を行うほうが効率的であると言える．

この結果は組織構成時のメッセージ量による大幅な遅延や，入札待ち時間等のエージェントの動作停止時間によって必要であった組織構成時間が，AS を形成する各エージェントに直接インスタンシエートの命令を送る提案手法を利用する事で削減できたことが原因であると考えられる．

5. まとめ

本研究では，既存のエージェントフレームワーク DASH の組織構成の非効率性に着目し，解決法としてユビキタス環境用帰還型エージェントフレームワークを提案した．提案フレームワークでは，主にリポジトリの分割と帰還 AS 管理エージェントの開発によって，フレームワーク上の AS に帰還機能を実装する事で，組織構成の省略と AS の永続性を目指した．評価実験では提案フレームワーク上で帰還・再生成機能を利用し，組織構成の必要時間を計測する事で，提案フレームワークの有効性を実証した．

今後の課題を以下に示す．

- 提案フレームワークにおける例外処理の対応
  現段階で提案プロトタイプには，ワークプレース上の AS の動作状況やシステムの緊急停止などの例外処理に対して軟弱性があるため，それに対応する必要がある．
- 帰還 AS 管理エージェントが保持する組織情報の設定
  提案フレームワークでは帰還した AS の組織情報として，エージェント名と最上位マネージャ名のみを保持している．このほかにサービス名や生成日時等，利用者にとって有用な情報を保持する事を検討している．
- 帰還後の AS の知識再利用方法の検討
  AS の帰還・再生成後に，以前動作して得た知識をどのように利用できるか考察する必要がある．また，帰還 AS 管理エージェントが保持する情報に知識再利用に向けた情報を追加する事も検討している．

参考文献

[1]打矢 隆弘, 武田 敦志, 菅沼 拓夫, 木下 哲男, “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”, 情報処理学会論文誌, Vol.44, No.3, pp.799-811(2003).

[2]打矢 隆弘, 菅沼 拓夫, 木下 哲男, 白鳥 則郎, “リポジトリ型エージェントフレームワークを用いたマルチエージェントの永続性の実現手法”, 2003 年電子情報通信学会総合大会, S-6, (2003).