AAR/SCF：AAR型サービス調整フレームワークによるユーザ指向サービスの実現

打矢隆弘[†] 加藤貴司[†]
菅沼拓夫[†] 木下哲男[††]

ブロードバンドネットワークの普及にともない，自宅，学校・会社，出張先等の様々な環境から同一ユーザがネットワークサービスを受けるようになってきた．このようにユーザが複数のコンピュータを利用する場合に，システムがユーザの要求/環境に適合したサービスを動的に提供する技術のニーズが高まっている．このようなニーズに対し，我々はエージェント指向アプローチを用いて動的にサービスを構成し，かつそれぞれの端末の環境に応じてサービスを調整するフレームワーク（AAR/SCF: AAR based Service Coordination Framework）を提案する．AAR/SCFは，Active Agent Repository（AAR）と呼ばれるエージェントリポジトリを用いて，ユーザ要求に従って動的にマルチエージェントシステムを構築し，ユーザの端末上に生成し，サービスを提供する．さらに，サービス終了後に，マルチエージェントシステムをエージェントリポジトリに回収し，次回以降のサービス提供の際にそれらの再利用を図る．AAR/SCFの特徴は，そのマルチエージェントシステムを次回以降に利用される端末の環境に応じて適切に調整し，環境適応的なユーザ指向サービスを提供することである．本論文では，AAR/SCFの概要とその動作について述べ，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．

AAR/SCF: Realizing User-oriented Service Using AAR-based Service Coordination Framework

Takahiro Uchiya,[†] Takashi Katoh,[†] Takuo Suganuma[†] and Tetsuo Kinoshita[††]

With the rapid growth in broadband network, a user has come to receive a particular network service in different environment, such as at home, at school, at office, or at a business trip place. Thus, in case that the user uses various computer platform, it is required to provide the service that adapts to the user requirement and environment dynamically. To address the issue, we propose the AAR (Active Agent Repository) based service coordination framework (AAR/SCF). AAR/SCF constructs the service dynamically by using the agent oriented approach to coordinate the service according to the platform environment. In AAR/SCF, a multiagent system is constructed according to user requirement by using agent repository called AAR, and agents instantiate to the user's platform and provides the service. Moreover, after finishing the service provisioning, they are collected to the repository so as to reuse them in the next service provisioning. The feature of AAR/SCF is to realize the environment-adaptive, and user-oriented service provisioning by adjusting the multiagent system with platform environment. In this paper, we describe the outline of AAR/SCF and show the effectiveness of the proposed method through the experimental results.

1. まえがき

近年のネットワーク技術の発達にともなうブロードバンドの普及により，コンピュータに関する高度な専門知識を持たないエンドユーザが，コンピュータ/ソフトウェアを利用する機会が増加してきた．従来のソフトウェア技術は，ある固定的なアプリケーションをシステム上にインストールし，そのアプリケーションを起動することでユーザにサービスを提供している．そのため，ユーザは自分自身で利用したいアプリケーションの特徴や利用方法を判断しなければならず，ソフトウェア運用に対する高度な知識を必要とされる．加えて，ユーザの要求が変化したり，複数のコンピュータ

† 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

を使用する等のコンピュータ使用環境（プラットフォーム・ネットワーク）が変化したりした場合に，それに応じて適切な調整を行うためには，ユーザのさらなるソフトウェア利用知識やネットワーク知識等が必要となる．しかし，ユーザがそのような知識を学ぶことはかなりの労力を必要とし，難しいものである．そこで，一般ユーザが快適にサービスを受けるために，システムがユーザの要求/環境に適合したサービスを動的に提供する技術のニーズが高まっている．

このようなニーズに対し，エージェント指向アプローチによる動的なサービス構成技術が注目を浴びている．これは，分散環境上のマシンから必要なエージェントを配信し，そのエージェントにタスクを実行させることで動的なサービス提供を行うものである．その一手法として我々は，エージェントリポジトリを用いて動的なエージェント組織構成を行い，マルチエージェントシステムを構築するリポジトリ型エージェントフレームワークを開発してきた[1)]．しかしながら，リポジトリ型エージェントフレームワーク上で動作するシステムは，ユーザが単一の環境を継続的に使用することを前提としているため，ユーザが複数のコンピュータを使用するような場合には，端末間で継続したサービス提供ができず，非効率であった．ところが，現在のユーザのコンピュータ利用形態として，単一のコンピュータでのサービス利用のみにとどまらず，異なる環境にある複数のコンピュータから，ある端末で利用したサービスと同様のサービス（WWW，メール，チャット，ビデオ会議等）を利用する機会が増加している．このような場合に，ある環境ですでに動的に構築されたソフトウェア（マルチエージェントシステム）を再活用することができれば，使用する端末すべてにあらかじめソフトウェアをインストールする必要はなく，またある環境でユーザの嗜好や要求をソフトウェアに反映させておけば，別の環境で再び一からシステムがユーザ要求に合わせて内部の構成を作り直す必要はなく，ユーザの要求を反映したソフトウェアをユーザに継続的にかつ即時に提供することができる．このとき，環境が著しく異なる場合に特定のプラットフォームや特定のネットワークに依存するエージェントを単純に再利用したのでは，サービスを低下させる原因となりうるので，環境に合わせてエージェント組織を組み換えてサービスを適切に調整する必要がある．

そこで，本論文では，ユーザが複数のコンピュータを非同期に使用する際に，ユーザの要求を反映したマルチエージェントにより構成されるサービスを保存し，次回の要求の際に環境に応じてサービスを調整することで，環境を考慮したユーザ指向のサービス提供を実現する．具体的な実現手法としては，我々が開発してきた既存の能動的エージェントリポジトリ（AAR）型エージェントフレームワーク[2)]を拡張し，環境の差異を吸収してサービスを調整するフレームワーク AAR/SCF（AAR 型サービス調整フレームワーク）を提案する．AAR/SCF ではユーザのサービス利用履歴を保持し，これを活用する機構（ユーザ指向サービス調整機構）を新たに備えることにより，ユーザ環境の差異を判断して，その環境に応じた適切なサービス調整を実現する．

本論文で提案する AAR/SCF を用いることで，一度利用したサービス（マルチエージェントシステム）を別環境で要求した場合でも，本フレームワークによりその環境に適合するようにマルチエージェント組織の構成が自律的に調整されるため，ユーザはエージェントの置換等を意識することなく，環境に適したサービスを継続的に受けることができる．すなわちユーザが複数のコンピュータを使用する状況において，プラットフォーム/ネットワーク環境が異なる場合でも，ユーザはプラットフォームやネットワーク環境に関する知識を必要とせずに，適切なサービスを利用できる．

本論文の構成は以下のとおりである．2 章では，既存リポジトリ型エージェントフレームワークの概要と課題を述べる．3 章では，課題を解決する新しいフレームワーク AAR/SCF を提案し，その内部機構であるユーザ指向サービス調整機構の概要と，サービス調整手法について述べる．4 章では，プロトタイプの実装と，既存エージェントアプリケーションを用いた評価実験について述べ，その結果を示す．最後に 5 章でまとめと今後の課題を述べる．

2. 既存 AAR 型エージェントフレームワークの概要

2.1 既存 AAR 型エージェントフレームワーク

本章では，本提案の基盤技術として用いる ADIPS フレームワーク[1)]，およびそれを発展させた AAR 型エージェントフレームワーク[2)]の概要を示す．

ADIPS フレームワーク（ADIPS: Agent-based Distributed Information Processing System）は種々の計算機プロセス（コンポーネント）を自律的なエージェントとして実装し，分散環境で自律的，協調的に作業を実行するソフトウェアモジュールの集団を構成することでサービスを提供することを目的として開発されてきた．このフレームワークは，ソフトウェアを複数のエージェント（マルチエージェント）で構成し，

ユーザ要求／動作時の環境の変化に合わせて，分散環境上でエージェントを構成・再構成して動的にサービスを提供するものである．このフレームワーク上では，サービスは複数のエージェントの組合せで構築されることが大きな特徴であり，たとえばビデオ会議システムにおいては，Video エージェント，Audio エージェント，WhiteBoard エージェント，CPU 監視エージェント，ネットワーク監視エージェント等の組合せによって 1 つのビデオ会議サービスを提供する．ADIPS フレームワークでは，リポジトリと呼ばれるエージェントの保持機構に様々なエージェントが格納され，リポジトリ機構の働きにより，ユーザの要求や環境条件を満たすエージェント／エージェントシステムが動的に選択・合成され，同組織をエージェントの実行環境であるワークプレース上に生成してサービスを提供する．エージェントの定義やその知識は，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．具体的なエージェントの選択・合成（組織構成）手法としては，タスクを分割/依頼するエージェント（マネージャ）と，タスクを実行するエージェント（コントラクタ）が，契約ネットプロトコル[3]を拡張した組織構成プロトコルに基づいて動的にマルチエージェント組織を構成する．

ところが，従来の ADIPS フレームワークではエージェントがその動作環境での動作終了後には破棄されてしまい，その動作結果等を新たなマルチエージェントシステムの生成に反映させることはできなかった．そのため，マルチメディア通信サービスにおける QoS 調整等において，過去のエージェントの動作結果を活用したサービスの高度化や，ユーザの要求に合わせて構成/再構成されたマルチエージェントシステムの効果的な再利用を図ることが難しくなっていた．

これに対し ADIPS フレームワークの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導入することにより，過去の動作結果を有効に活用することで，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が構築できる Active Agent Repository（以下 AAR）を提案し，これに基づくエージェントフレームワークを開発してきた（**図 1**）[2]．AAR は ADIPS フレームワークにエージェントの永続的な保持機構を提供するものであり，ユーザへのサービス提供を実行し終えたマルチエージェントを復帰（フィードバック）させて回収し，保持する．

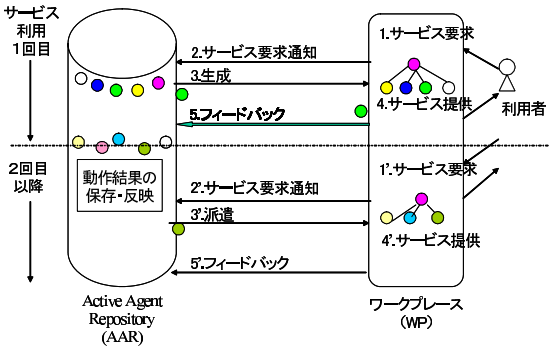

図 1 既存 AAR 型エージェントフレームワーク
Fig. 1 AAR-based multiagent framework.

このとき，利用されたエージェントの動作結果に関する情報も保存し，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，各エージェントは以前の動作結果が活用でき，組織構成を高速化したり，自身の推論処理動作を効率化したりして，サービス制御の応答速度を上げ，以降に動作環境に派遣された際にサービス品質を向上できる．

このように，我々は，ADIPS フレームワークの動的なマルチエージェントシステムの構築という特徴を生かしつつ，動作したマルチエージェントを AAR にフィードバックし，その動作結果を活用して，次回のユーザからのサービス要求の際に積極的にマルチエージェントを再利用することでシステム全体としてのサービスの効率化と高度化を可能とし，その結果，同一環境においてユーザが同種のサービスを要求した場合には，サービスの構成時間の短縮やサービスの高度化（ビデオ会議システムにおける動画制御速度の向上等）が実現できることを示した．

2.2 AAR 型エージェントフレームワークの課題

AAR 型エージェントフレームワークにより，ユーザの要求に合わせて構成/再構成されたマルチエージェントを，同一環境で有効に再利用することが可能となった．

ところが，最近では，単一の端末でのサービス利用にとどまらず，異なる環境にある複数の端末から同様のサービスを利用する機会が増加している．具体例として，学校あるいは会社でビデオ会議を行ったユーザが，帰宅後に再度ビデオ会議を行いたい場合等がある．そこでは，ビデオ会議を支援するマルチエージェントシステムがユーザの要求（動画重視，音声重視，文字チャット重視等）や環境条件に基づいて新たに構成され，ユーザは必要なサービスを利用することになる．このときそのユーザが，以前，同様の環境で使用した

マルチエージェントを呼び出して利用することができれば，ユーザは自分に合致したサービスの継続的な使用が可能となる．しかし，前回使用時と今回の使用時で環境が大幅に異なる場合には，前回のマルチエージェントシステムの構成が環境条件と適合せず，これがサービス品質の低下につながってしまうことがある．たとえば，ユーザが高性能な CPU を持つ端末でのビデオ会議を終了し，その後に低性能の CPU を持つ端末においてビデオ会議を行う場合を考える．高性能な CPU を持つ端末の場合であれば，頻繁に動画の制御/監視を行うエージェントを用いてきめ細かな制御を行うことで，安定した滑らかな映像を提供することができる．しかし，同じエージェントを低性能な CPU を持つ環境で再利用してしまうと，その頻繁な制御/監視自体が環境のリソースを消費してしまい，適切な制御ができなくなる．

本論文では，このような「ユーザが複数の端末を使用する場合に，特定の利用者環境を対象として構成されたマルチエージェントシステムを効果的に再利用できない」という問題を解決するための新しいフレームワークを提案する．以降，本論文では，ユーザの要求に合うように構成/再構成されたサービスを「ユーザ指向サービス」と呼び，ユーザの使用する端末が物理的に変化しても，サービスを構成するマルチエージェントを必要に応じて調整し，ユーザ指向サービスの環境適応的な提供を実現する．すなわち，複数の端末を使用するユーザがサービスを再利用する際に，リポジトリ型エージェントフレームワークが，以前の利用時との環境の差異を判断/吸収し，当該環境に適合したマルチエージェント組織を構築する新しい仕組みを提案する．

3. ユーザ指向サービス実現のための新フレームワーク AAR/SCF の設計

3.1 ユーザ指向サービス実現のための環境情報の考察

本節では，ユーザ指向サービスの提供に影響する環境情報の定義とその選択指針について述べる．ユーザ指向サービス（ネットワークサービス）は，各端末のプラットフォームのリソース，各端末のネットワークのリソースを使用して提供されるものであり，これらのリソースのうち，リソース自体の変動がサービスの品質に直接影響するものを，サービス提供に影響を与えるコンピューティングリソースであると定義する．以上の定義を基に，本フレームワークでは以下のリソースの情報を基本的かつ有用な情報として選択し，これらを最大限活用して，ユーザ指向サービスの実現を目指す．以下，選択したリソースの環境情報とその選択理由を述べる．

オペレーティングシステム（OS）の種類： OS 特有の機能の有効活用により，インタフェースの高速化等に活用可能である．

CPU 情報： クロック周波数等の情報の利用により，過負荷回避／ CPU の最大限利用に活用可能である．

メモリ情報： メモリ不足回避等による安定したサービス提供に活用可能である．

解像度情報： 端末画面の大きさに応じた適応的インタフェースの構成等に活用可能である．

ネットワーク接続方式： 有線(ISDN, ADSL, LAN, etc.)・無線等の特定のネットワークに適したサービス提供に活用可能である．

ネットワーク実効スループット： ネットワーク帯域に応じたマルチメディアサービスの品質調整に活用可能である．

以上の理由により，これらの環境情報がユーザ指向サービス実現に有用であると考えられる．なお，上述したような静的なリソース情報に加えて，動的なリソース情報，たとえば端末のプラットフォーム／ネットワークリソースの変動具合等を環境情報として扱うことは今後の課題としたい．

3.2 AAR/SCF の設計方針

前節で述べた環境情報を活用して，環境適応的なユーザ指向サービスを実現するために，本論文では，以下の方針でエージェントフレームワークを設計する．

（D1）ユーザ特性の恒常的反映 基本的な機能を持つ複数のエージェントをエージェントリポジトリに準備して，ユーザの要求によりそれらを組み合わせてエージェントシステムを構成／再構成し，ユーザの特性をサービスに反映させる．ユーザの特性を反映したエージェントシステムは，次回以降のサービスまでエージェントリポジトリに回収・保持され，その特性をつねに保持する．

（D2）環境適合性の向上 環境に特化した複数のエージェントをエージェントリポジトリに準備し，環境の差異をエージェントの置換により吸収する．具体的には，前回と環境が異なる場合には，サービス実現に必要なエージェントが自律的に環境依存性を判断し，必要に応じてエージェントを組み換え，マルチエージェントシステムを再構築して，環境に適合する．

（D1）の設計方針のもと，既存 AAR 型エージェント

フレームワークが，上記の設計方針と同様の方針で開発されているため，これを拡張する形でフレームワークを実装するものとする．

（D2）の設計方針のもと，環境情報を活用して環境の差異を判断・調整する機構，すなわちユーザ指向サービス調整機構（USCM: User-oriented Service Coordination Mechanism）を提案し，この機構をリポジトリに組み込んで使用するものとする．また，本論文では，各エージェントが自律的に環境を判断するためのエージェント知識記述方式を定義する．

3.3 AAR/SCF の概要

前節で述べた USCM の導入にあたり，1 つの AAR に複数のユーザのサービス利用履歴やマルチエージェントを保持しておくことは，プライバシ保護やアクセス負荷軽減という観点から好ましくない．そこで AAR を共通リポジトリとプライベート AAR に分離し，プライベート AAR 内に各ユーザのプライベートなサービス利用履歴を保存/活用する USCM を導入する．また各プライベート AAR で端末上のワークプレースで使用されたマルチエージェントを回収・保持するものとする．

これを AAR 型サービス調整フレームワーク（AAR/SCF: AAR-based Service Coordination Framework）と呼ぶ（**図 2**）．

共通リポジトリはインターネット上のサーバに配置され，複数のエージェント開発者が新たに作成した新規エージェントを保持する．共通リポジトリには種々のサービス実現において共通的に利用されるエージェントや，環境に特化したエージェントが多数開発／蓄積されているものとする．

一方，プライベート AAR は，AAR を個人用にしたものであり，各ユーザに 1 つずつ準備される．プライベート AAR は，インターネット上で永続性を持つサーバ（LAN 上のゲートウェイや自宅のホームサーバ等）に設置され，ユーザの特性を持つマルチエージェントを次回の利用まで保持する．また，プライベート AAR 内の USCM において，後述するサービス利用履歴を保持し，ユーザからのサービス要求発生時に，ユーザの端末の環境情報とサービス利用履歴を活用してサービスの調整を行う．

ワークプレースは，端末でのエージェントの実行環境であり，ユーザが使用する端末上に配置される．ワークプレースには，常駐型の環境情報取得エージェントが存在し，端末上の環境を自動的に取得して，ユーザのサービス要求時に USCM に環境情報を通知する．

AAR/SCF では，ユーザが初めてサービスを要求したときは，共通リポジトリ内のエージェントにより組織を形成してサービスを提供し，使用後はプライベート AAR にそれらを保持する．そして，再度サービスが要求されたときは，まず，USCM が，プライベート AAR に保存されている以前の環境情報と，現在のユーザ環境との差異を抽出する．そして，差異が検出された場合には，プライベート AAR 内のサービス実現に必要なエージェントシステムにタスクを通知し，そのエージェントらが自律的に環境依存性を判断し，エージェントシステムを再構築して，環境に適応する．以上の一連の処理は，利用者からのサービス要求に基づいて自動的に実行されるため，利用者はサービス調整等を意識せずに，その時点での利用環境に適合したサービスを受けることができる．

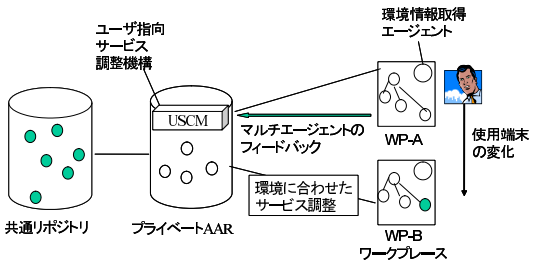

図 2 AAR/SCF の概要
Fig. 2 Outline of AAR/SCF.

3.4 環境情報を扱うためのエージェント知識記述

各エージェントが環境に応じて適切に動作するために，本フレームワークでは，エージェントに環境依存性を判断する知識を与え，エージェントが環境を自律的に認識・判断する．すなわち，本論文では，エージェントの環境依存性を「エージェントが自身のタスクを実行するにあたり，特定の条件を満たす動作環境でしか動作しない場合には，エージェントは環境に依存し，それ以外の場合は環境に依存しない」と定義する．エージェントが環境に依存する場合は，開発者は，サービス実行に必要なコンピューティングリソースの知識を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ルール型知識として記述する．設定すべきコンピューティングリソース知識で扱う環境情報は，3.1 節で述べた環境情報となるが，開発者は，必要なリソース情報のみを記述すればよい．なお，エージェントが環境に依存しない場合には，従来どおり，開発者はこうした知識を記述する必要はない．

例として，OS が Linux で，CPU 周波数が 1.0 GHz 前後で正常に動作する CPU 監視エージェントに与える知識を**図 3** に示す．エージェントが必要なコンピューティングリソース情報を (initial_facts) 内に記述する

```
(initial_facts
 //サービス実行に適したコンピューティングリソース
 (mytarget :env (info :os "linux"
                      :cpuspeclow 800
                      :cpuspechigh 1200)
           :func CPUMonitor)
)
(knowledge
 //タスク実現可能性の判断
 (rule task-check
  (mytarget :env ?env)
  (task-check :id ?id
      :task (task :name "CPUMonitor" :env ?presentenv))
  (== ?env:os ?presentenv:os)
  (<= ?env:cpuspeclow ?presentenv:cpuspec)
  (>= ?env:cpusepchigh ?presentenv:cpuspec)
  ~(bid :id ?id)
  -->
  (make (bid :id ?id :content (task :name "CPUMonitor")))
 )
```

図 3 コンピューティングリソースに対するエージェント知識の例
Fig. 3 Example of Agent Knowledge against Computing Resource.

ことで，エージェントは自身の環境動作知識を持ち，(knowledge) 内に組織構成知識を記述することで，タスクが通知された際に，タスク名，環境情報と自身の環境動作知識を比較して，タスクに入札するかどうかを自律的に判断する．

3.5 プライベート AAR におけるユーザ指向サービス調整機構 USCM

本節では，ユーザ指向サービス調整機構 USCM の設計の詳細を説明する．

USCM は（F1）サービス利用履歴保持機能と（F2）サービス調整機能の 2 つの機能を持つ．

（F1）サービス利用履歴保持機能は，あるサービス要求が発生してそれに応じてマルチエージェントが形成された際に，その情報をサービス利用履歴として保持する．ここで，サービス利用履歴情報は次のような考え方に基づき設計した．サービス利用履歴情報は，（a）サービス要求，（b）端末の環境情報，（c）エージェントシステムの組織情報，の 3 つの要素からなる記述の集合である（**図 4**）．（a）サービス要求はサービス名が保持される．（b）端末の環境情報は 3.1 節で述べたものであり，サービスに影響のあるリソース情報をすべて含む．（c）エージェントシステムの組織情報は，以前の動作環境での組織構成情報を持つマネージャ名である．以上の設計により，新たにサービス要求と環境情報が与えられた際に，過去のサービス要求と環境情報を参照することにより，過去のエージェントシステムの組織情報を検索し，以前のエージェントシステムを特定することができる．

（F2）サービス調整機能は，端末の環境情報とサービス利用履歴を活用して，環境の判断とそれに応じた調整を行う．ユーザのサービス要求発生時に，USCM は以下のように動作する．

```
サービス利用履歴＝{サービス名, 環境情報, マネージャ}
                 ＝{FAMES,
                   (:CPU 1000MHz :Memory 512MB :解像度 1280＊1024
                   :OS WindowsXP …),
                   FAMESMA@20030308112250.w1.robin
                  }
```

図 4 サービス利用履歴の例
Fig. 4 Example of history of service use.

（1）環境情報の取得：サービス要求発生元の環境にいる環境情報取得エージェントに対して，環境情報を要求して，端末の環境情報を受け取る．

（2）以前のサービス利用の有無の判断：サービス要求とサービス利用履歴のサービス名を比較して，そのサービスが以前利用されたことがあるかどうか判断する．新たなサービス要求と判断した場合は，USCM は共通リポジトリにサービス要求を通知し新規のサービスを構成する．以前利用したサービスと同様のサービスと判断した場合には，（3）により環境の差を判断する．

（3）以前利用時との環境の差異の判断：サービス利用履歴の環境情報と（1）で受け取った現在の端末の環境情報を比較して，環境が同等かどうか判断する．環境情報には OS 情報，CPU 情報，メモリ情報，解像度情報，ネットワーク接続方式，ネットワーク実効スループットが含まれており，それぞれの項目ごとに以前利用時の環境情報と比較し，すべて同等と判断した場合に環境を同等と判断し，1 つでも異なる項目がある場合は，環境が異なると判断するという簡単な方式を用いている．たとえば，USCM での環境判断のための知識として，「CPU 速度は現在の環境の CPU 速度 x (MHz) を基準として前後 $\mathrm{a} \times x$ (MHz) 以内を，メモリ容量は現在の環境のメモリ容量 y (MB) を基準として前後 $\mathrm{b} \times y$ (MB) 以内を同等とする（a, b は定数）」を与え，これに基づいて環境を判断する．

現在の環境情報の全項目が過去のある環境情報と同等であった場合は，USCM は以前使用したマルチエージェントが再利用可能と判断し，プライベート AAR に保持していたマルチエージェントをそのまま再利用するための処理を行う．具体的には，サービス利用履歴のマッチした履歴に記述されているマネージャに対し，環境情報を含まないタスク通知を送信する．これにより，以前のマルチエージェントが直接落札による組織構成を実

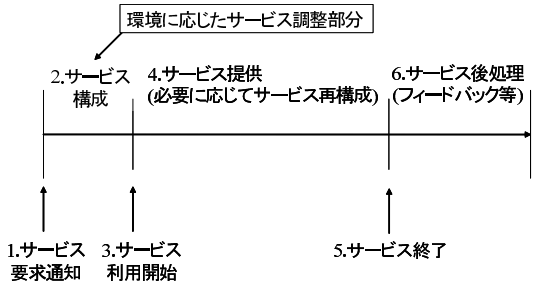

図 5 サービス構築の流れ
Fig. 5 Flow of service construction.

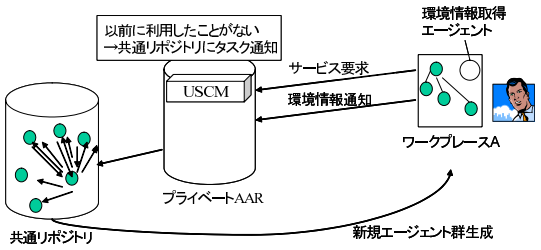

図 6 新規のサービス構築
Fig. 6 Construction of new service.

行し，それらをそのまま再活用できる．

現在の環境が過去に利用した環境と異なると判断した場合には，USCM はユーザが以前に利用したマルチエージェントはそのままでは再利用できないと判断し，必要なエージェントを部分的に再構成するための処理を行う．具体的には，サービス利用履歴に記述されているマネージャに対し，環境情報を含む直接落札メッセージを送信する．それを受け取ったマネージャが，以前の組織における各タスク実行エージェントにタスク通知を送信すると，各タスク実行エージェントは自身の動作知識に基づいて環境情報を含んだタスク内容を判断し，実行不可能な場合に落札拒否メッセージを発行する．USCM は 1 つ以上の落札拒否メッセージを受け取った場合，すなわちプライベート AAR 内だけでは組織が形成できなかった場合には，拒否されたタスクを実現するエージェントを探すために共通リポジトリに対してタスク通知を行い，必要なエージェントを見つけて置換する．

この USCM を導入することにより，ユーザのサービス要求時に環境に依存しないエージェントらは再利用してユーザの要求を反映しつつ，CPU 監視エージェント等の環境に依存するものは，それぞれの環境に適したエージェントに置換して利用することで，ユーザが複数の端末を使用しても，環境に応じて自律的にサービスを調整して，ユーザ指向サービスを継続的に提供することができる．

3.6 サービス構築手法

本節では，AAR/SCF におけるサービスの構成手法について述べる．**図 5** は AAR/SCF におけるある端末での基本的なサービス構築の流れである．この流れにおいて，“2. サービス構成” が本フレームワークのサービス調整の本質部分であり，ここに焦点を当てて議論を進める．3.6.1 項では，新規のサービス構築手法について述べる．3.6.2 項では，以前と同等な環境で同様のサービスを要求した場合のサービス提供手法について述べる．3.6.3 項では，本論文の主要部である，以前と異なる環境で同様のサービスを要求した場合のサービス調整手法について述べる．

3.6.1 新規のサービス構築

ユーザが以前に使用したことがないサービスを要求した場合の，サービス構築について述べる．

ある環境でユーザがサービスをワークプレースに要求すると，システムは以下のように動作する（**図 6**）．

（1. サービス要求通知） サービス要求がプライベート AAR に通知される．

（2-1. サービス構成—環境情報の取得） プライベート AAR 内の USCM がワークプレースの環境情報取得エージェントから環境情報を受け取る．

（2-2. サービス構成—過去のサービス利用有無の判断） プライベート AAR 内の USCM で，受け取った環境情報をサービス利用履歴と照合して，そのサービスを利用したことがあるか判断する．ここではないと判断された場合を考える．

（2-3. サービス構成—新規サービス構築要求）USCM はプライベート AAR 内のエージェントだけではサービスが提供できないと判断し，共通リポジトリに新規のサービスを構築するためのサービス要求を通知する．

（2-4. サービス構成—新規エージェント群生成） 共通リポジトリ内でエージェントが組織を構成し，ユーザの端末上のワークプレースに生成され，サービス提供を開始する．

3.6.2 前回使用時と同等な環境でのサービス提供手法

前回使用時と同等な環境でサービスが再要求された場合に，システムは以下のように動作する（**図 7**）．

（1. サービス要求通知） 3.6.1 項と同様に行われる．

（2-1. サービス構成—環境情報の取得） 3.6.1 項と同様に行われる．

（2-2. サービス構成—過去のサービス利用有無の判断） 3.6.1 項と同様に行われる．ここではありと判断

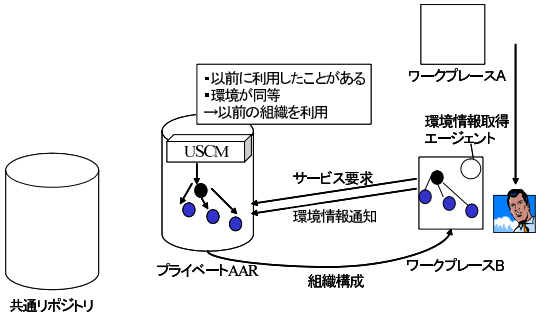

図 7 前回使用時と同等な環境でのサービス提供手法
Fig. 7 Service provisioning in equivalent environment of previous case.

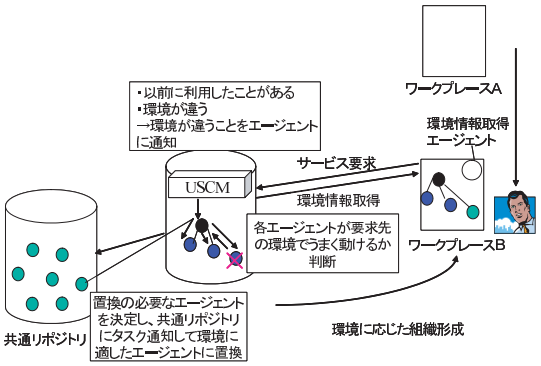

図 8 前回使用時と異なる環境でのサービス調整手法
Fig. 8 Service coordination in different environment to previous case.

される．

（2-3. サービス構成—環境情報の照合） USCM で環境情報の照合を行い，USCM が環境が同等であると判断し，サービス利用履歴に記述されていたマネージャ名を参照し，このマネージャに対し環境情報を含まない直接落札メッセージを通知する．

（2-4. サービス構成—組織構成） 直接落札メッセージを受け取ったマネージャは自身で獲得した以前の組織構成知識を用いて，直接落札による組織構成を実行し，以前のマルチエージェント組織を構成する．

以上のプロセスにより，システムが以前の環境情報と同等と判断した場合は，ユーザの要求を反映済みのマルチエージェントをそのまま別の環境で再利用して，ユーザ指向サービスを提供できる．このことにより，ユーザの要求を反映するための再構成プロセスが不要となり，サービス提供開始までの時間が短縮される．

3.6.3 前回使用時と異なる環境でのサービス調整手法

前回使用時と異なる環境でサービスが再要求された場合に，システムは以下のように動作する（**図 8**）．

（1. サービス要求通知） 3.6.1 項と同様に行われる．

（2-1. サービス構成—環境情報の取得） 3.6.1 項と同様に行われる．

（2-2. サービス構成—過去のサービス利用有無の判断） 3.6.1 項と同様に行われる．ここではありと判断される．

（2-3. サービス構成—環境情報の照合） USCM で環境情報の照合を行い，その結果，以前利用した環境と異なると判断すると，USCM はプライベート AAR 内の以前利用したマネージャに対し，現在の環境で動作可能か各エージェントに判断してもらうために，環境情報を内包したタスク通知を送信する．

（2-4. サービス構成—環境を考慮した組織構成） マネージャは自身で獲得した以前の組織構成知識により以前動作したタスク実行エージェントを特定し，それらにタスク通知を送信する．タスク実行エージェントは，環境依存性を判断し，タスク実行可否の判断を行う．タスク実行が不適と判断したエージェントは拒否メッセージをマネージャに送信する．

（2-5. サービス構成—部分的再構成） マネージャはすべてのタスク入札が集まらなかった場合，拒否されたタスク要求を USCM に通知する．それを受け取った USCM は共通リポジトリに対し，そのタスク要求を通知する．共通リポジトリ内ではそのタスクを実行可能なエージェントがそのタスクを受理し，プライベート AAR 内のマネージャに入札メッセージを送信する．マネージャはすべてのタスクに対する入札を受け付けると，落札メッセージを入札した各エージェントに送信する．

（2-6. 環境に応じた組織形成） プライベート AAR，共通リポジトリにより環境情報を基に部分的に再構築されたマルチエージェントが組織を構成し，ユーザにサービスを提供する．

以上のプロセスにより，システムが以前の環境と異なると判断した場合は，ユーザの要求を反映済みのマルチエージェントのうち，環境に依存しないエージェントは再利用し，環境に依存するエージェントを部分的再構成により置換することで，環境に応じたマルチエージェントを構成する．これにより，環境に応じた適切なユーザ指向サービスが実現される．

4. プロトタイプシステムの実装と実験

4.1 プロトタイプシステムの実装

ユーザ指向サービス調整機構 USCM の内部設計を行い，それを基にプライベート AAR に USCM を実装した．ユーザの端末上のワークプレースには自動的に端末の環境情報を取得する環境情報取得エージェントを設計・実装した．共通リポジトリは既存フレームワーク[6)]のリポジトリをそのまま利用した．このように，共通リポジトリ，プライベート AAR，ワークプレースを用意し AAR/SCF のプロトタイプを試作した．実装言語は Java を用いた．なお，実験での簡素化のため，環境情報としては CPU 情報を対象とした．

4.2 実験内容

評価アプリケーションとして非同期メッセージングシステム FAMES[4)]およびやわらかいビデオ会議システム FVCS[5)]を用いて，3 章で提案したフレームワークの動作検証を行った．FAMES，FVCS はともにマルチエージェントで構成されるユーザ指向のエージェントアプリケーションである．実験での前提条件として FAMES，FVCS のエージェントは，自身が環境に依存する場合は，適切に動作する環境の範囲を知識として持ち，組織構成時にタスクアナウンスの対象環境が自身の環境と該当した場合に入札して，生成された環境で正常に動作するものであるとする．また，プライベート AAR 内の USCM の環境判断知識は，「CPU 速度が現在の環境の CPU 速度 x (MHz) を基準として前後 $0.2x$ (MHz) 以内を同等とする」（DK）と定義した．

実験内容は以下の 2 つである．

（1）前回使用時と同等な環境でのサービス提供手法の動作検証

（2）前回使用時と異なる環境でのサービス調整手法の動作検証

上記 2 つの検証項目について，FAMES，FVCS を用いた動作検証を行った．本論文では，FAMES は環境に依存するエージェントを組織に含まないことから，異種環境でも以前と同様の組織が構成される反面，異種環境のサービス調整の作用が顕著に現れないことが確認されたため，（1）同等環境での動作検証に特に焦点を当て実験結果を示す．また，FVCS は環境に依存するエージェントが組織に含まれることから，異種環境におけるサービス調整の効果が顕著に確認できたため，（2）異種環境での動作検証に特に焦点を当て，実験結果を示す．

評価項目としては，特定のアプリケーションに依存しないフレームワーク特性の評価に焦点を絞り，サービス構成処理時間，および，エージェント間通信回数を採り上げる．これは，アプリケーションに特化したネットワークサービスの性能がエージェント開発者による各エージェントの実装に依存して決まり，これに応じて性能自体も変化することによる．こうした特定アプリケーションに依存した機能を含む場合の取扱いは今後の課題である．

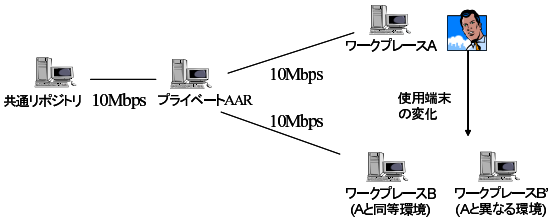

図 9 実験環境
Fig. 9 Experiment environment.

表 1 FAMES の実験端末
Table 1 Experiment platform of FAMES.

	ワークプレースA	ワークプレースB	ワークプレースB'
CPU	PentiumIII 1GHz	PentiumIII 1.13GHz	Pentium 4 2GHz
Memory	256MB	384MB	1024MB
OS	Windows XP	Windows XP	Windows XP

表 2 FVCS の実験端末
Table 2 Experiment platform of FVCS.

	ワークプレースA	ワークプレースB	ワークプレースB'
CPU	PentiumIII 1GHz	PentiumIII 1GHz	Ultra SPARC 154MHz
Memory	256MB	256MB	128MB
OS	Linux 2.4.7-10	Linux 2.4.7-10	Solaris2.7
Vic	ver2.8 ucl-1.1.3	ver2.8 ucl-1.1.3	ver2.8 ucl-1.1.3
Video Capture	FlyVideo'98	BT878A-TVPCI	Sun Video Board

また，フレームワークの比較対象として，2.1 節で説明した既存リポジトリ型エージェントフレームワークの代表例である DASH[6)]を従来フレームワークとして用いた．

4.3 実験環境

実験環境および各端末の性能を**図 9**，**表 1**，**表 2** に示す．ユーザがある端末 A でサービスを利用・終了した後，別端末 B，B' に移動し，そこから再び同様のサービスを受ける場合を想定する．CPU 情報を比較した場合，端末 B は端末 A と同等の環境であり，端末 B' は端末 A と異なる環境である．

4.4 アプリケーション FAMES を用いた AAR/SCF の評価

FAMES は従来の分散型電子メールシステムに，適

応型サービス構成機能，知的メッセージング処理機能の 2 機能を付加し，(1) 送信後の取消しや回覧等の高度な配送機能の実現が困難，(2) 受信者の不在等の状況により配送が非効率化する可能性が高いといった問題点を解決するためのマルチエージェントシステムである．FAMES は，ユーザの作業状態や要求に応じて，メールの回覧機能，削除機能等が動的に付加されるようにシステムを再構成し，ユーザにサービスを提供する．

ワークプレース A において，ユーザは FAMES を使用時に回覧機能，削除機能等の標準の FAMES エージェント組織に組み込まれていない機能を要求し，そのユーザ要求に基づいて FAMES のマルチエージェントは再構成動作を行い，その要求を満たしたサービス提供を行った．その後，別端末からユーザがサービス「FAMES」を再要求したとき，システムは次のように動作した．

4.4.1 実験 1-1：現在の環境が前回使用時と同等な環境の場合

(1. サービス要求通知) ワークプレース B はプライベート AAR にサービス要求「FAMES」を通知した．

(2-1. サービス構成—環境情報の取得) プライベート AAR 内の USCM はワークプレース B にいる環境情報取得エージェントに環境情報を要求し，端末 B の環境情報を取得した．

(2-2. サービス構成—過去のサービス利用有無の判断) USCM はサービス要求「FAMES」がサービス利用履歴に保存されている端末 A で使用したサービス名「FAMES」と一致するために，以前に利用したことがあるサービスだと判断した．

(2-3. サービス構成—環境情報の照合) USCM は，環境判断知識 (DK) を利用して，サービス利用履歴とその環境情報を比較した．ワークプレース A の環境情報は，CPU 速度 1,000 MHz であり，ワークプレース B の環境情報は CPU 速度 1,130 MHz であるので，1,130 (MHz) − 1,000 (MHz) < 0.2 × 1,000 (MHz) となり，USCM はワークプレース B をワークプレース A と同等な環境であると判断し，プライベート AAR の FAMES マネージャに環境情報を含まないタスク通知を行った．

(2-4. サービス構成—組織構成) FAMES マネージャは以前の組織構成の知識に基づいて，プライベート AAR 内のタスク実行エージェントに環境情報を含まないタスク通知を行い，その結果，前回と同様の組織がワークプレース B 上に生成されサービス提供を開始した．

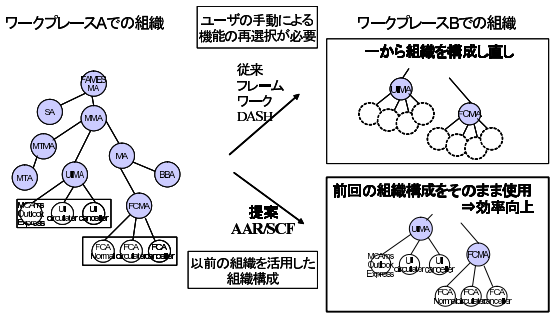

図 10 FAMES の実験結果（同等環境）
Fig. 10 Result of experiment of FAMES.

これにより，ユーザはワークプレース B で改めてユーザの要求を反映したユーザ指向サービスの一部である，回覧機能，削除機能等を再要求しなくても，即座にその機能を使用できることを確認できた（**図 10**）．

同実験を 10 回試行したところ，従来フレームワーク DASH では，初期のサービス構成処理時間が約 15 秒であった．それに加え，2 回の組織再構成処理が必要であった．1 回の組織再構成処理には，ユーザの手動での要求入力と再構成処理時間約 15 秒が必要であり．この際にエージェント間で約 110 回のメッセージ送受信が行われた．

一方，本 AAR/SCF では，あらかじめユーザの特性を反映したサービスが構成されるため，上記の組織再構成処理は不要となり，組織再構成時間すべてと，エージェント間メッセージ通信すべてを削減できた．

以上の結果より，AAR/SCF は，前回使用時と同等の環境の場合は，前回と同様の組織を構成することにより，ユーザ要求の反映処理にともなう組織再構成処理時間およびエージェント間通信回数を削減できることを確認した．

4.4.2 実験 1-2：現在の環境が前回使用時と異なる環境の場合

USCM はワークプレース B' が以前の環境と異なると判断し，環境情報を含むタスク通知を前回の FAMES エージェントに通知した．FAMES は環境（この場合は特定の CPU 速度）に依存するエージェントがいないことから，すべてのタスク実行エージェントが入札を返し，結果的に同等環境と同様のエージェント組織が構成された．つまり，異種環境の場合も，組織再構成処理時間およびエージェント間通信回数が削減されることが確認された．

4.5 アプリケーション FVCS を用いた AAR/SCF の評価

FVCS は，動画機能を提供する Video エージェント，

音声機能を提供する Audio エージェント，文字チャット機能を提供する WhiteBoard エージェント，CPU 監視エージェント，ネットワーク監視エージェント等のエージェントから構成され，サービス提供時のユーザ要求の変化やプラットフォーム・ネットワーク性能の変化に応じて組織を再構成し，安定したビデオ会議サービスを提供するものである．

端末 A において，標準のビデオ会議サービスが構成された後，ユーザは文字チャット重視のビデオ会議を要求し，それに従って FVCS のマルチエージェントは再構成動作を行い文字チャット機能が充実したサービスを提供した．その後，別端末からユーザがサービス「FVCS」を再要求したとき，システムは次のように動作した．

4.5.1 実験 2-1：現在の環境が前回使用時と同等な環境の場合

USCM はワークプレース B が以前の環境と同等と判断し，環境情報を含まないタスク通知を前回の FVCS エージェントに通知した．その結果ワークプレース A と同様の組織がワークプレース B 上に生成された．この結果から，FAMES 同様に，ユーザ要求の反映処理が不必要になることから組織再構成処理時間およびエージェント間通信回数を削減できることが確認された．

4.5.2 実験 2-2：現在の環境が前回使用時と異なる環境の場合

（1. サービス要求通知） ワークプレース B’ はプライベート AAR にサービス要求「FVCS」を通知した．

（2-1. サービス構成—環境情報の取得） プライベート AAR 内の USCM はワークプレース B’ にいる環境情報取得エージェントに環境情報を要求し，端末 B’ の環境情報を取得した．

（2-2.サービス構成—過去のサービス利用有無の判断） USCM はサービス要求「FVCS」がサービス利用履歴に保存されている端末 A で使用したサービス名「FVCS」と一致するために，以前に利用したことがあるサービスだと判断した．

（2-3. サービス構成—環境情報の照合） USCM は，環境判断知識（DK）を利用して，サービス利用履歴とその環境情報を比較した．ワークプレース A の環境情報は，CPU 速度 1,000 MHz であり，ワークプレース B’ の環境情報は CPU 速度 154 MHz であるので，$1{,}000\,(\mathrm{MHz}) - 154\,(\mathrm{MHz}) > 0.2 \times 1{,}000\,(\mathrm{MHz})$ となり，USCM はワークプレース B’ をワークプレース A と異なる環境であると判断した．

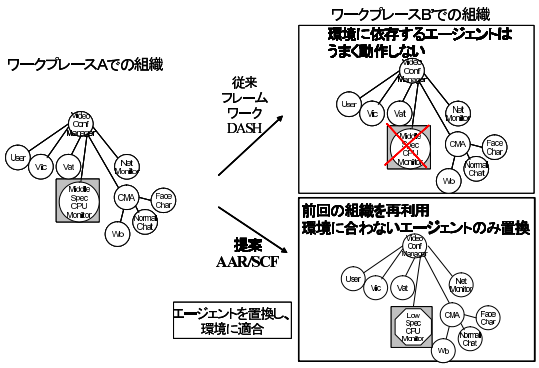

図 11 FVCS の実験結果（異種環境）
Fig. 11 Result of experiment of FVCS.

（2-4. サービス構成—環境を考慮した組織構成） USCM はプライベート AAR 内のマネージャ（VideoConfManager）に環境情報を含んだ直接落札メッセージによるタスク通知を行った．VideoConfManager は以前の組織構成の知識に基づいて，プライベート AAR 内のコントラクタに環境情報を含むタスク通知を行った．各コントラクタは自身が持つ適切動作可能な環境範囲知識に従って，タスク通知に反応した．MiddleSpecCPUMonitor エージェントを除くコントラクタは自身がその環境で動作可能と判断したが，MiddleSpecCPUMonitor エージェントは自身の適切動作可能な環境範囲が 500〜1,500 MHz であったため，CPU 速度が 154 MHz であるワークプレース B’ で適切な動作は不可能と判断し，VideoConfManager エージェントに落札拒否メッセージを送信した．

（2-5. サービス構成—部分的再構成） VideoConfManager エージェントは落札拒否を受け取るとそのタスクを USCM に通知した．USCM はそのタスクを受け取り共通リポジトリに対してタスク通知を行った．そのタスクに対して，LowSpecCPUMonitor エージェントが入札を行い，MiddleSpecCPUMonitor エージェントが LowSpecCPUMonitor エージェントに置換された組織が形成された．形成された組織の各エージェントがワークプレース B’ に生成され，サービス提供を開始した．

これによりワークプレース A で構築された文字チャット重視というユーザの特性を保ちながら，端末の環境に合わせて必要なエージェントを置換して，サービスを調整できることが確認された（**図 11**）．

同実験を 10 回試行したところ，従来フレームワー

クDASHでは，初期のサービス構成処理が約20秒であった．それに加え，1回の組織再構成処理が必要であった．1回の組織再構成処理には，ユーザの手動での要求入力と再構成処理時間約20秒が必要であり，この際にエージェント間で約40回のメッセージ送受信が行われた．また，従来フレームワークDASHでは，環境情報を扱えないために，標準のCPU監視エージェントがワークプレースB'でも使用されたが，CPU性能の低い端末で頻繁にCPUの監視を行ったために，その挙動がシステムを不安定にし，ビデオ会議サービスの品質を低下させることが観測された．

一方，本AAR/SCFでは，あらかじめユーザの特性を反映したサービスが構成されるため，初期のサービス構成処理に約25秒かかったが，上記の組織再構成処理は不要となり，組織再構成時間すべてと，エージェント間メッセージ通信すべてを削減できた．また，CPU監視エージェントが低性能用のエージェントに置換されたために，CPU監視回数が減りきめ細かい動画制御がされなくなったものの，ビデオ会議システムを著しく不安定になることが回避され，安定したビデオ会議サービスが提供された．

以上の結果より，AAR/SCFは，前回使用時と異種の環境の場合は，ユーザ要求の反映処理にともなう組織再構成処理時間およびエージェント間通信回数を削減し，なおかつ，環境の差異を判断・調整することで，ネットワークサービスの品質を調整可能であることを確認した．

4.6 関連研究との比較

本フレームワークは他の研究では見られない，ユーザの端末上の環境情報を利用してエージェントシステムの組織構成を変え，サービスを調整するという点に特徴がある．

ユーザの要求に応じてサービスを構成してユーザに提供するシステムとして，DANSE[7]がある．DANSEとは，ユーザの要求とコンテクストに応じてユーザのサービス環境を動的に構成する適応型ネットワーキングサービス環境である．DANSEは，ハードウェアからソフトウェアまでのあらゆるサービス構成要素を統一的にネットワークリソース（NR）として扱い，サービス実行時にユーザ要求やコンテキストに応じて必要となるNRを探索し，これらを組み合わせてサービス環境に割り当て，サービスを提供する．しかし，DANSEでは実現したサービスをサービス終了後に保持し，再利用するための枠組みが不足しているために，次回，同様な条件でユーザ要求が発生したとしても，再度同じ手順で，ネットワークリソースの組合せを決定するために，ネットワークを探索する必要が生じてコストがかかってしまい，ユーザ指向のサービスの継続的な提供が困難である．

これに対し本フレームワークでは，サービスを提供したマルチエージェントをプライベートAARにフィードバックして，サービスを再利用可能にし，以前と同様の端末からサービスを再要求した際にも，前回のマルチエージェント組織を活用して即時にユーザ指向のサービスを提供することができる．

ソフトウェアリソースを動的にユーザに割り当てるものとしてWWWとJavaアプレットを用いて，ユーザの端末上にソフトウェアを必要に応じてダウンロードして実行するものがある[8]．しかし，このシステムではその対象となるソフトウェアが，Javaアプレットで実行可能なものに限られてしまい，ソフトウェアの汎用性が不足している．

本フレームワークでは，既存の計算機プロセス資源をエージェント化しているために，サービスを実現するJava言語以外の種々のプロセスから成るソフトウェアを，エージェントの集団（マルチエージェントシステム）として構成し，利用することが可能である．

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Plangent[9]，Odyssey[10]，Voyager[11]，Concordia[12]，AgentSpace[13]等がある．これらは主に知的モバイルエージェントフレームワークであり，マルチエージェントではなくシングルエージェントを分散環境で移動させて，サービスを提供する仕組みである．シングルエージェントを学習させながらつねに使用することでユーザ指向のサービスを継続的に提供することが可能であるが，端末の環境に基づいてサービス調整をするためには，あらかじめシングルエージェントの知識に，使用されるすべての端末の情報が必要である．また，ユーザが移動して次に使用する端末が不明である状態では，エージェントを移動させることはできない．

一方，本フレームワークは，マルチエージェントの組織構成と端末上への生成の支援を行うため，各エージェントの知識には自分が適切に動作する環境に対する知識を持たせるだけでよく，環境の差異は，リポジトリ内で行われるエージェントの組織構成の変化で柔軟に対応する．環境情報はシステムにより自動的に取得・利用されるため，エージェント開発者の負担は少ない．さらに，ユーザが次に使用する端末が不明である状態でも，エージェントシステムはネットワーク上のプライベートAARに待機されているため，次回どの端末からもサービス要求が可能である．

5. むすび

本論文ではユーザ指向のサービスを提供するAAR/SCFの提案を行った．AAR/SCFはマルチエージェントシステムをプライベートAARにフィードバックして再利用可能にし，ユーザ指向サービスの継続的な提供を実現する．さらにユーザの使用する端末の環境情報に基づいてサービス調整を自動的に行う点に特徴がある．本論文では，プロトタイプの実装と動作実験を通じて，AAR/SCFの有効性を示した．

今後の予定として，複数のエージェントアプリケーションを用いた，定量的な性能評価を行う．また，本論文ではUSCMの環境判断に，簡単な比較による処理知識を採用したが，今後，各環境情報の性質と特性を考慮し，適切な環境判断知識を順次追加修正していきたい．さらに，プライベートAARのセキュリティの向上についても考察していきたい．応用課題として，現在はユーザごとにプライベートAARを準備することを想定しているが，ユーザのグループ化によりグループ内で使用したマルチエージェントの蓄積と共有によるエージェントシステムの発展・進化についても検討予定である．

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) 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).
3) 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).
4) 北形 元, 加藤貴司, 菅沼拓夫, 今野 将, 木下哲男：FAMES：エージェントに基づく柔軟な非同期メッセージングシステムの設計と実装, 情報処理学会論文誌, Vol.43, No.2, pp.487–498 (2002).
5) 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).
6) Sugawara, K., Hara, H., Kinoshita, T. and Uchiya, T.: Flexible Distributed Agent System programmed by a Rule-based Language, *Proc. 6th IASTED International Conference of Artificial Intelligence and Soft Computing*, pp.7–12 (2002).
7) 板生知子, 松尾真人：適応型ネットワーキングサービス環境DANSE, 電子情報通信学会論文誌B, Vol.J82-B, No.5, pp.730–739 (1999).
8) Richardson, T., Bennet, F., Wood, K.R. and Hopper, A.: Global Teleporting with Java: Toward Ubiquitous Personalized Computing, *Computer* (1997).
9) 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).
10) General Magic Inc.: Odyssey Web Page (1998). http://www.genmagic.com/agents/odyssey.html
11) ObjectSpace Inc.: ObjectSpace Voyager Technical Overview (1997). http://www.objectspace.com/Voyager/
12) Mitsubishi 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 Multi Agent and Cooperative Computation (MACC'98) (Dec. 1998).

(平成15年 5 月16日受付)
(平成15年12月 2 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．PRIMA 2002 Student Poster Award，2002年度電子情報通信学会学術奨励賞等受賞．電子情報通信学会学生会員．

加藤 貴司

1971年生．2001年東北大学大学院情報科学研究科博士後期課程修了．現在，東北大学電気通信研究所助手．博士（情報科学）．マルチエージェントシステムにおけるエージェントの協調に興味を持つ．人工知能学会，電子情報通信学会各会員．

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．1997 年東北大学電気通信研究所助手．2003 年同助教授．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，エージェント型ネットワークミドルウェア等の研究開発に従事．The 8th JWCC Best Presentation Award，情報処理学会第 54 回全国大会大会奨励賞等受賞．博士（工学）．IEEE，電子情報通信学会各会員．

木下　哲男（正会員）

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．知識情報処理技術の研究開発に従事．1996 年東北大学電気通信研究所助教授，2001 年同大情報シナジーセンター教授．知識工学，エージェント工学，知識型設計支援システム，エージェント応用システム等の研究開発に従事．情報処理学会 1989 年度研究賞，1996 年度論文賞，電子情報通信学会 2000 年度業績賞等受賞．工学博士．電子情報通信学会，人工知能学会，日本認知科学会，IEEE，ACM，AAAI 各会員．