F-021

エージェントシステムと非エージェントシステムとの再利用可能な接続機構の開発
Development of Reusable Interconnection Mechanism of Agent System and Non-agent System

前村 貴秀* 打矢 隆弘† 今野 将* 木下 哲男‡
Takahide Maemura Takahiro Uchiya Susumu Konno Tetsuo Kinoshita

1. まえがき

分散環境において，非エージェントシステム(Non Agent System : NAS)とエージェントシステム(Agent System : AS)は，それぞれ独立して利用者にサービスを提供してきた．しかし近年では，多様化したサービス要求に対応するために，これら 2 種類のシステムが併存し，相互に連携して，それぞれの特徴を活かした新しいサービスを提供することが望まれる．

論文[1], [2]により，NAS 同士，AS 同士，AS から NAS を利用する方法が提案され，NAS から AS を利用する方法は，論文[3]～[6]により NAS と AS 間の中継処理を行う接続機構を用いる方法が提案されている．しかし現状では，この接続機構の開発において，

(P1)接続機構と AS 間における通信処理の実現が困難
(P2)接続機構の内部処理手続きの再利用が困難

の 2 つの問題点が指摘されている．

(P1)は AS 間で用いられている通信方式が，AS が動作しているエージェントプラットフォーム（以降，AP）毎に異なるために発生する問題点である．(P2)は接続機構の設計段階から再利用を考慮した設計・開発が行われていないために発生する問題点である．これら(P1)，(P2)のため，接続機構を実現する際には，接続するシステムに関する知識を習得した上で，システムの組み合わせ毎に開発を行う必要があり，開発者の負担を増加させる要因となっている．

そこで，本論文では，NAS から AS を利用するための接続機構（以下，NAS-AS 接続機構）を系統的に開発する手法を提案する．具体的には，AP に依存するメッセージ通信機能と ACL 変換機能を再利用可能なモジュールとして提供し，それらを用いて接続機構を設計する手順を提案する．本提案手法を適用することにより，NAS と AS 間のメッセージ通信を実現するための作業負担が軽減され，コード再利用の観点からみた利便性が向上し，接続機構の開発における作業負担の軽減が期待できる．

以下，2 章では本論文で提案する接続機構のインタフェースと AP に依存する機能のモジュール化，及び，それを前提とした設計手順を提案する．3 章で，提案手法を適用した接続機構の試作実験と評価を述べ，4 章で開発済み接続機構の再利用実験と評価を行い，提案手法の有効性を示す．

2. NAS-AS 接続機構の開発手法の提案

本章では，本提案手法における接続機構の開発方針を述べ，これに基づく開発手法を提案する．

2.1 NAS-AS 接続機構の開発方針

問題点(P1)を解決するために“接続機構を開発する際に用いる AP 依存な通信処理手続きを事前に再利用可能なモジュールとして提供する手法”を提案する．また，問題点(P2)を解決するために“接続機構の内部処理手続きの分析を行い，NAS，AS，NAP(非エージェントプラットフォーム)，AP に依存しない処理を再利用可能な手続きとしてモジュール化・インタフェース化する手法”を提案し開発者の負担軽減を図る．

*東北大学大学院情報科学研究科
†名古屋工業大学大学院工学研究科
‡東北大学情報シナジー機構

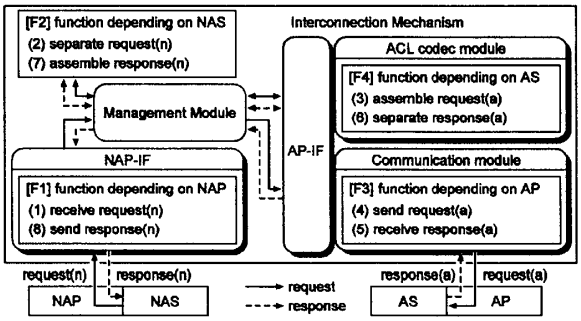

図 1 提案手法に基づく接続機構の構成

2.2 提案手法における設計手順

まず，提案手法に基づく NAS-AS 接続機構の内部構成を図 1 に示す．NAS-AS 接続機構の設計は，F1～F4 の 4 種類の機能モジュールと管理モジュール（Management Module : MM）の 5 つのモジュールの設計に帰着される．以下，提案手法に基づく設計手順を説明する．なお，F1 と F2 については，次の理由により，提案手法による支援の対象としない．すなわち，F1 の NAP と接続機構間の通信機能(NAP-IF)については，現在，既に様々な NAP との通信機能を実現するためのプラットフォームやライブラリが提供されており，開発者は，これらを利用することで NAP との通信機能を容易に実現可能なためである．また，F2 の NAS との通信に伴う要求（応答）の解析・組立機能については，NAS との間でやり取りされる要求（応答）の形式は，NAS によって異なる．本論で対象としている開発者は，NAS の要求（応答）を解析し，組み立てる機能を開発するのに必要な知識を有していることを前提としているため，特段の支援は必要としないものとする．以下，F3，F4，MM の概要と設計手順を示す．

F3 : Communication module : 提案手法では，AP と接続機構間の通信機能(Communication module)を再利用可能なモジュールとして実現し，AP 毎に異なる手続きを統一的に利用可能にするためのインタフェースと共に，開発者に提供する．これにより，開発者はこのインタフェースを通して各 AP との通信機能を呼び出す処理を MM 内に記述するだけで，AS とのメッセージ通信を実現できる．

F4 : ACL codec module : 提案手法では，AS との通信に伴う要求（応答）の解析・組立機能(ACL Codec module)を再利用可能なモジュールとして実現し，当該モジュールを利用可能にするためのインタフェースと共に，開発者に提供する．

以下に F3，F4 の設計手順を示す．

[Step1]要求に含める情報の決定 : AS に対して送信する要求（ACL メッセージ）に含める情報を決定する．例えば，NAS からの要求に含まれる情報や AS に関する情報（パフォーマティブや宛先となるエージェント名）などがあげられる．

[Step2]情報を取得するための手続きの設計 : Step1 で決定した情報を取得するための手続きを設計する．

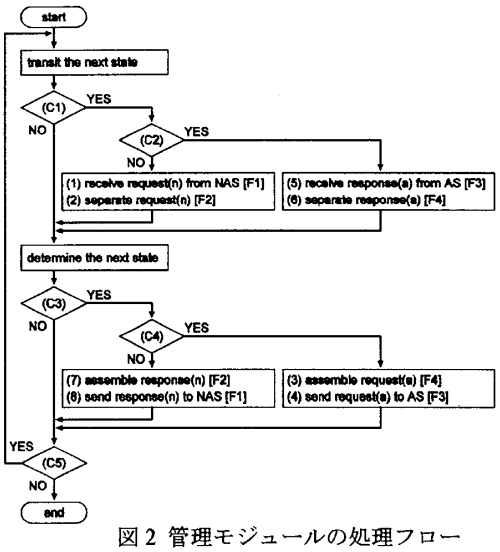

図 2 管理モジュールの処理フロー

[Step3]MM に要求を渡す手続きの設計：ACL メッセージを MM に送信する手続きを設計する．

[Step4]送信する要求を組み立てる手続きの設計：Step1～3 までの設計と AS で採用されているメッセージ形式に基づき，伝達する要求を組み立てる手続きを設計する．

[Step5]MM から応答を受け取る手続きの設計：MM から渡される応答メッセージを受け取る手続きを設計する．

[Step6]応答から抽出する情報の決定：応答（ACL メッセージ）から抽出する情報を決定する．例えば，要求が AS 側で処理された結果として生成された情報や MM 中の条件分岐で必要な情報などがあげられる．

[Step7]抽出した情報を保持する手続きの設計：Step6 で抽出される情報を保持する手続きを設計する．

[Step8]応答から情報を抽出する手続きの設計：Step5～7 までの設計に基づき，Step6 で決定した情報を応答（ACL メッセージ）から抽出するための手続きを設計する．

以下に，管理モジュール(Management Module)の設計手順を示す．

[Step1]NAS からの要求を受信して NAS に応答を送信するまでの処理（図 2）を設計する．なお，図中の各分岐条件は，以下の通りである．

（C1） AS（NAS）からの要求（応答）受信の有無
（C2） AS からの応答受信の有無
（C3） AS（NAS）への要求（応答）送信の有無
（C4） AS への要求送信の有無
（C5） 処理継続の判断

[Step2]NAS や AS との要求（応答）の送受信を通して得られた情報を保持するためのデータ構造を設計する．

2.3 NAS-AS 接続インタフェース

提案手法に基づく接続機構の内部構成（図 1）をもとに，NAS-AS 接続のためのインタフェース機能(1)～(5)の設計について説明する．

（1）**NAP と通信するためのインタフェース**(NAP-IF)：NAP と接続機構の通信機能(F1)を提供するインタフェースで，既存の NAS 用プラットフォームやライブラリで実現される．

（2）**管理モジュール**(Management module)：提案手法に基づいて，開発者が実現するモジュールである．

（3）**通信モジュール**(Communication module)：AP と接続機構との通信機能（F3）を提供するモジュールである．これは，AS とのメッセージ通信を実現するために，AP 間で用いられているネットワークプロトコルを用いて，AP と接続機構間で ACL メッセージの送受信を可能にする機能を提供する．本機能は，以下の 3 種類の機能を利用して実現される．

- **非同期通信機能**：AP の基盤となるエージェントモデルによって要請される非同期型のエージェント通信機能を提供．
- **セッション管理機能**：接続機構と AS 間において，複数の要求／応答メッセージの送受信からなるトランザクション処理を行う場合に必要な機能．具体的には，各メッセージに対して，エージェント名に相当する一意な識別名を付与して，個々のトランザクション処理を識別・管理する．
- **AS 内の未知エージェントからメッセージを受信可能にする機能**：AS 内の全エージェントにメッセージをブロードキャストし，任意のエージェントから応答を受信する場合など，メッセージ送信時には，その存在を認識していなかったエージェントからも応答メッセージを受信可能にする．

（4）ACL **変換モジュール**(ACL codec module)：NAS が採用している情報通信形式を AS が採用している ACL メッセージ形式に変換する機能と ACL メッセージ内のパラメータ値を取得する機能を提供する．具体的には，各 AP に共通するパラメータ（performative, from, to, content 等）を設定／抽出する機能を，AP 毎に再利用可能なモジュールとして実現する．

（5）AP **依存モジュール用インタフェース**(AP-IF)：接続対象となる AP の実装から，AP に依存する処理を AP-IF を通して呼び出し可能な形態で獲得し，再利用可能なモジュールとして実現する．一般的には，AP が異なるとモジュールの利用手順も異なる．しかし，各 AP 依存モジュールは，インタフェースで定義された機能を実現できるように実装されるため，開発者は，AP-IF を通して，各 AP 依存モジュールを利用できる．これにより，接続先を異なる AP に変更する場合でも，同様な手続きでメッセージを送受信するための処理が実現可能となる．さらに，AP と接続機構間の通信に関連する手続きが同一になることで，AP に依存する手続きとそれ以外に依存する手続きの特定が容易になる．次に，AP-IF を介した NAS と AS 間の処理の流れを説明する（図 1）．

[Step1] NAS からの要求（request(n)）を NAP-IF が受信する．次に，要求（request(n)）が MM に渡される（図 1（1））．

[Step2] MM は，要求(request(n))を受け取り，要求メッセージ(request(a))に必要な情報の抽出と解析を行う(図 1（2）)．

[Step3] MM は，ACL 変換モジュールを呼び出し，ACL 形式の要求メッセージ（request(a)）を組み立てる（図 1（3））．

[Step4] MM は，通信モジュールを呼び出し，組み立てた要求メッセージ(request(a))を AS に向けて送信する(図 1（4）)．

[Step5] AS は，要求メッセージ（request(a)）を処理し，接続機構に対し応答メッセージ（response(a)）を送信する．

[Step6] AS からの応答メッセージ（response(a)）を通信モジュールが受信すると，MM は，AP-IF を通して，応答メッセージ（response(a)）を受け取る（図 1（5））．

[Step7] MM は，AP-IF を通して ACL 変換モジュールを呼び出し，NAS への応答（response(n)）に必要な情報を応答メッセージ（response(a)）から抽出する（図 1（6））．

[Step8] 応答メッセージ（response(a)）から抽出した情報を用いて，MM は，NAS で用いている形式に応じた応答（response(n)）に変換する（図 1（7））．

[Step9] MM は，NAP-IF を通して，NAS に応答（response(n)）を送信する（図 1（8））．

3. 提案手法に基づく NAS-AS 接続機構の試作実験

提案手法を適用した接続機構の開発における作業負担の削減効果を評価するため，接続機構の試作実験を行った．具体的には，NAS と AS の異なる組み合わせに対して，それぞれ接続機構を試作し，内部処理の開発コストの減少（記述行数の減少）を定量的に評価した．

3.1 試作実験の設定

実験環境は次の通りである．ハードウェア環境は，ネットワークで接続された PC（OS は Windows XP）によって構成した．NAP については，同期通信を行う Web Server と非同期通信を行う Mail Server の 2 種類を用いた．このように異なる通信形態をとる NAP を用いることにより，NAP に対する内部処理の記述の違いを評価する．AP については，リポジトリ型エージェントフレームワークである ADIPS/DASH フレームワーク[7]（以降，DASH）と FIPA 準拠のプラットフォームである SAGE[8] の 2 つを用いた．異なる 2 種類の AP と接続するための接続機構を試作することにより AP に対する内部処理の記述の違いを評価する．また，各 AS は，要求メッセージに対して自身が保持している情報を，応答メッセージとして返すように実装されている．

［再利用可能なモジュールとインタフェースの試作］

次に，各 AP に対応する再利用可能なモジュールとそれらを利用するためのインタフェースを試作した．具体的には，F3 : Communication module，F4 : ACL codec module，AP-IF（AP 依存モジュール用インタフェース）の 3 つである．なお，モジュールとインタフェースの実装言語には，Java を用いた．試作したモジュールとインタフェースを構成するクラスの総数は 16，総コード量は 3,245 行，コンパイル後のサイズは，63,037bytes であった．

［提案手法に基づく接続機構の試作］

提案手法に基づいて接続機構を開発する場合，開発者が実装する必要があるのは，図 1 の Management Module と F2 のみである．本実験では，NAP（Web Server と Mail Server）と AP（DASH と SAGE）の組み合せのそれぞれに対応する 4 種類の提案手法を適用した接続機構を試作した．具体的には，NAP-IF（F1）として，既存のウェブサーバやメールサーバを利用した．F2 と MM は，NAP と AP の組み合わせに応じて試作した．F3 と F4 は，それぞれ Communication module と ACL codec module を利用した．

［比較対象となるラッパエージェントの試作］

本実験では，提案手法に基づく接続機構の比較対象として，Wrapper Agent と呼ばれるエージェントを用いた接続機構の試作も併せて行った．これは，接続先と同種の AP 上で動作するエージェントを用いて，NAS と AS 間の接続処理を行う方式(Wrapper Agent 方式：以降，WA 方式)である．WA 方式を用いた接続機構の内部構成を図 3 に示す．WA 方式では，F3 を接続先と同じ AP を用いて実現することにより，開発者が，この機能を開発する手間を軽減することができる．本論文では，AP に依存するメッセージ通信処理を実現する手間が軽減されるという点で，提案手法と WA 方式の開発方針が類似していることを考慮して，これを比較対象として採用した．

WA 方式を用いて接続機構を開発する場合，開発者は，MM，F2，F4 を Wrapper Agent として実現する必要がある．本実験では，NAP と AP の組み合せに対応する 4 種類の WA 方式を用いた接続機構を試作した．具体的には，NAP-IF(F1) として，提案手法を適用して開発する場合と同様のソフトウェアを利用した．機能 F2，F4，MM については，Wrapper Agent として，NAP と AP の組み合わせに応じて試作した．機能 F3 は，接続対象と同じ AP を利用して実現した．

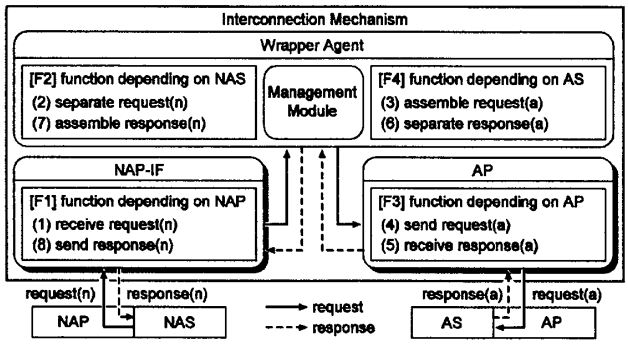

図 3 WA 方式を用いて開発した接続機構の構成

なお，本実験では，NAS と AS 間の接続に関わる記述だけを比較するために，Java の import 文，コメント，空行，閉じ括弧だけの行は除外し，記述量の比較を行う．

3.2 実験の結果と評価

実験結果を表 1 に示す．表中には，接続機構の試作を通して記述した手続きの記述量に加えてコンパイル後のプログラムサイズも併記した．表中の Web と Mail は，それぞれ NAP として用いた Web Server と Mail Server を表し，PM と WA はそれぞれ，提案手法（PM）と WA 方式（WA）を表している．ここで，削減率とは，提案手法を用いることで削減できた割合を表し，式（1）に示すように提案手法と WA 方式間の記述量から算出する．なお，プログラムサイズについても同様の算出法で比較する．

$$\text{削減率} = \left(1 - \frac{\text{提案手法の記述量}}{\text{WA方式の記述量}}\right) \times 100\,[\%] \quad (1)$$

表 1 より，WA 方式よりも提案手法を適用して開発した場合の方が，記述量を 2 割弱から 5 割削減でき，プログラムサイズについては，1 割から 3 割削減されることを確認した．

また，提案手法を用いた場合(表 1(a)～(d))と WA 方式を用いた場合(表 1(e)～(h))を比較すると，WA 方式の方が NAP と AP の組合せによって記述量のばらつきが大きく，削減率にもばらつきがみられた．これは，WA 方式の場合，エージェントの振舞いの一部としてメッセージ通信処理を記述する必要があるため，個々の AP で用いられている知識記述方法の違い（ルール型の知識記述と手続き型の知識記述）が記述量のばらつきにつながった．これに対して，提案手法の場合，AP 毎に用いられている知識記述方法には依存せずに，同様な記述方法でメッセージ通信処理が実現できるため，記述量のばらつきを抑えることができた．

4. 開発済み接続機構の再利用実験

提案手法を適用した接続機構の再利用性を評価するために，接続機構の再利用実験を行い，接続機構の変更作業の負担軽減（コード再利用からみた再利用率の向上）を評価した．

4.1 再利用実験の設定

接続機構によって接続された系全体は，NAS，NAP，接続機構，AS，及び，AP によって構成されている．そこで，本実験では，系を構成するシステムの一部を変更し，再利用できた記述量を計測した．具体的には，以下の 3 種類の変更を対象として実験を行った．

表 1 記述量とコンパイル後のプログラムサイズの

システム構成		行数			プログラムサイズ		
NAP	AP	行		削減率	bytes		削減率
		PM	WA	(1-PM/WA)	PM	WA	(1-PM/WA)
Web	DASH	(a) 65	(e) 91	28.6%	4,649	5,489	15.3%
	SAGE	(b) 66	(f) 80	17.5%	5,169	6,199	16.6%
Mail	DASH	(c) 60	(g) 126	52.4%	4,813	7,026	31.5%
	SAGE	(d) 70	(h) 145	51.7%	5,555	8,656	35.8%

表 2 提案手法と WA 方式間の接続機構の再利用率の比較

再利用率		システム構成：<NAP, protocol, AP>
提案手法	WA方式	
変更前／変更後	変更前／変更後	＜変更前＞→＜変更後＞
(1a) 23% / 25%	(1b) 23% / 17%	<Web, Simple, DASH> → <Mail, Simple, DASH>
(1c) 35% / 33%	(1d) 39% / 21%	<Web, Simple, SAGE> → <Mail, Simple, SAGE>
(2a) 72% / 39%	(2b) 69% / 62%	<Web, Simple, DASH> → <Web, CNP, DASH>
(2c) 68% / 14%	(2d) 53% / 17%	<Web, Simple, SAGE> → <Web, CNP, SAGE>
(3a) 71% / 70%	(3b) 42% / 48%	<Web, Simple, DASH> → <Web, Simple, SAGE>
(3c) 87% / 74%	(3d) 64% / 56%	<Mail, Simple, DASH> → <Mail, Simple, SAGE>

[case 1] NAP を Web Server から Mail Server に変更する．

[case 2] 本実験で用いた AS は，応答メッセージを返すエージェントを決定する手段として，協調プロトコルを利用する．そこで AS の変更については，提供するサービスは同じだが，AS が処理可能な協調プロトコルが異なる AS への変更を行った．具体的には，要求メッセージに対して直に応答メッセージを返す単純なプロトコル（Simple）から契約ネットプロトコル（Contract Net Protocol：以降，CNP）に変更する．

[case 3] DASH 上で動作する AS から SAGE 上で動作する AS に変更する．

4.2 実験の結果と評価

実験結果を表 2 に示す．表中の変更前と変更後は，提案手法を用いた場合と WA 方式を用いた場合における，それぞれの再利用率を表し，protocol は接続機構と AS との間で用いられている協調プロトコルを表している．ここで，再利用率とは，変更前，または，変更後の記述量に対して，再利用することができた記述量の割合を表し，式（2）に示すように変更前（後）の記述量と再利用できた記述量から算出する．

$$\text{再利用率} = \frac{\text{再利用できた記述量}}{\text{変更前（後）の記述量}} \times 100\,[\%] \qquad (2)$$

表 2 に示したように，変更前の記述量に対する再利用率は，2割から 8割，変更後の記述量に対しては，1割から 7割であった．また，変更前と変更後の割合を比較すると，変更前の方の再利用率が高いという傾向がみられた．これは，変更前の記述量が変更後の記述量よりも少ないことによる．

一方，提案手法と WA 方式を比較すると，AP を変更した場合（表 2（3a～3d））が最も効果があり，平均して 2 割改善されている．すなわち，接続機構の変更時における作業では，再利用する接続機構が WA 方式の場合には，AP 毎に知識記述方法が異なるため，Wrapper Agent 内の多くの処理を再度開発する必要があった．これに対して，提案手法では，MM から呼び出すモジュールを変更することにより，DASH と SAGE における知識記述方法の違いを吸収することができた．これは，個々の AP に依存するメッセージ通信機能が AP-IF を介して利用できるため，AP 毎に用いられている知識記述方法に依らず，同様の手続きでメッセージ通信に関わる処理が実現できるためである．

また，NAP を変更した場合（表 2（1a～1d）），他の変更と比べて再利用率が低くなった．これは，NAS の変更，つまり同期通信から非同期通信への変更に伴って，処理の流れを大幅に変更する必要が発生し，再利用できる記述も減少したことによる．このため，方式によらず低い再利用率となった．

更に，AS を変更した場合（表 2（2a～2d））の削減率は，DASH を用いた場合（表 2（2a）と（2b））の差よりも SAGE を用いた場合（表 2（2c）と（2d））の差の方が大きくなった．これは，DASH でエージェントを開発するために用いられるルール型知識記述が，SAGE で用いられている手続き型の知識記述よりも，協調プロトコルを扱うのに適していたことによる．これは，DASH による WA 方式（表 2（2b））の再利用率が高いことからも明かである．

以上，試作実験を通して，提案手法による接続機構の設計・開発では，開発者の作業負担が軽減され，既存機能の有効活用も図れることを示した．

5. むすび

本論文では，NAS と AS 間の相互接続において，AS が提供するサービスを NAS が利用するための接続機構の開発に焦点をあて，系統的な開発手法を提案した．提案手法の特徴は，AP に依存する機能のモジュール化，及び，異なる AP であっても同様の手順で AS と通信可能となるインタフェースを提供している点にある．

提案手法に基づく接続機構の試作と評価実験を通して，接続処理の記述量が最大 52%削減されると共に，接続機構の再利用性も最大で 29%向上することを検証した．これにより，開発者の負担軽減が期待される．提案手法に基づく NAS-AS 接続機構では，接続元のシステムにとって，接続機構を介した先に接続されているシステムが AS なのか NAS なのかを意識せずに接続先の機能を利用することができる．提案手法は，相互接続される 2 つのシステム間での情報表現形式や通信方式の差異を隠蔽し，両者の有用な機能を相互に活用できる知的な接続機構の実現に向けた第一歩と考えている．

参考文献

[1] X. T. Nguyen et al.: "Demonstration of WS2JADE", Proc. of the 4th int. joint conf. on Autonomous agents and multiagent systems, pp. 135-136 (2005).

[2] 高橋ら："エージェント・プラットフォームを相互運用するためのメッセージ通信プロトコル", 電学論, 123-C, 8, pp. 1503-1511 (2003).

[3] "Agentcities task force. integrating web services into agentcities recommendation" (2003).

[4] D. Greenwood et al.: "An automatic, bi-directional service integration gateway", Proc. of WSABE (2004).

[5] D. Greenwood et al.: "Semantic enhancement of a web service integration gateway", AAMAS 2005 workshop on SOCABE (2005).

[6] M. Omair Shafiq et al.: "Agentweb gateway - a middleware for dynamic integration of multi agent system and web services framework", Proc. of WETICE'05, pp. 267-268 (2005).

[7] 藤田ら："分散処理システムのエージェント指向アーキテクチャ", 情処学論, Vol.37, No.5, pp. 840-852 (1996).

[8] "Sage - scalable fault tolerant agent grooming environment". http://sage.niit.edu.pk/.