推薦論文

柔軟なインタフェース適応のための Web サービスグループ管理システム

吉川 貴[†] 太田 賢[†]
中川智尋[†] 倉掛正治[†]

Web サービスに代表される分散コンピューティング環境において，ユーザの要求に合わせて自由にサービスの切替えや相互接続を行い，ユーザビリティの向上を図ることは困難であった．その原因の 1 つとして，入出力インタフェースの相違があげられる．たとえば，通貨の単位や国名の省略形といったインタフェース要素の相違を吸収するためには，なんらかの変換処理を行う必要がある．この課題を解決するために，本研究では同じような機能を持つサービス群をサービスグループとしてまとめ，グループ内のインタフェース変換・適応処理を行うサービスグループ管理システムを提案する．本手法の有効性を検証するためにプロトタイプの構築を行い，要求される機能を実現できることを確認し，サービス切替え時間を測定した．

Service Group for Flexible Interface Adaptation among Web Services

Takashi Yoshikawa,[†] Ken Ohta,[†] Tomohiro Nakagawa[†] and Shoji Kurakake[†]

In distibuted computing environment, it is difficult to improve usability by switching and composing services flexibly with user's requirements. It is considered that interface diversity is a potential source of problem. This paper describes about Service Group Management System (SGM), which finds interface conversion paths to adapt interface elements and it allows switching/composing services. Experiments on a prototype can switch to a service alternative with interface conversion, and we also measured switching time.

1. はじめに

昨今，セルラ通信や無線 LAN，短距離無線通信といった無線通信技術の発展にともない，ユーザの移動を束縛しない自由なモバイルコンピューティング環境が実現されている一方，Web サービスや CDN などの広域分散サービスが注目を集め，その開発が進んでいる．

モバイルコンピューティング環境でもこれらの分散サービスの利用をこれまで以上に広げるには，デスクトップ環境に比べて変化しやすいユーザのコンテキストや環境の変化に動的に適応していく必要がある．

図 1 にシナリオを示す．ユーザは携帯端末を用いて近傍のレストラン検索サービスやイベント情報サービスを使用し，レストランの位置をローカル地図サービス上に表示している．さらに，ユーザは別の場所に移動した後にも同じサービスの組合せを利用したいと考え，機能的に等価なレストラン検索サービスとイベント情報サービス，ローカル地図サービス，クーポンサービスを動的に検索し，組み合せて利用する（図 1 右側）．

上述のシナリオのように，移動した先で機能の似通ったサービスを検索し，組合せてそれまでと同等の組合せを実現することはこれまで困難だった．

この理由として，現状ではサービスどうしの関連性が考慮されていないため，機能の等価なサービスや，サービスの組合せを検索すること自体ができないとい

† 株式会社 NTT ドコモマルチメディア研究所
Multimedia Laboratories, NTT DoCoMo, Inc.

本論文の内容は 2003 年 6 月のマルチメディア，分散，協調とモバイル（DICOMO2003）シンポジウムにて報告され，DICOMO2003 プログラム委員会委員により情報処理学会論文誌への掲載が推薦された論文である．

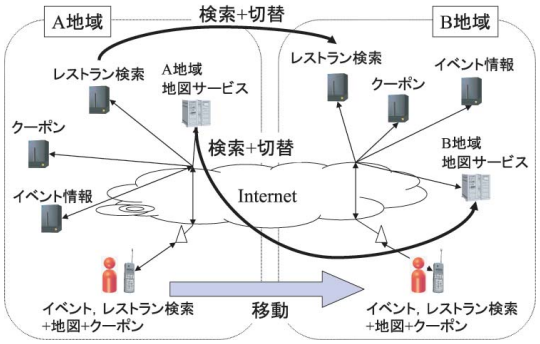

図 1 シナリオ：ユーザのコンテキスト変化によるサービス切替え
Fig. 1 Scenario: Service switching from user context.

う問題がある．そこで，サービスどうしの機能面での関連性を保持し，サービスどうしが機能的に等価であるかどうかを判断可能にする枠組みが必要である．

また，サービスの利用先を別のサービスに自動的に切り替える場合，入出力インタフェースの相違により相互接続性を保証することが困難であるという課題もある．特に，他のサービス提供者の提供するサービスに切り替えることを考慮すると，入出力インタフェースが完全に一致することは少なく，シナリオのように単純に処理を切り替えることができない．よって，サービスを動的に接続したり切り替えたりして利用するためには，インタフェースの相違を吸収して柔軟なサービス接続を可能にする技術が必要である．

そこで本研究では，機能的に等価なサービスの集合体をサービスグループと定義し，サービスグループの動的な生成，管理を行うサービスグループ管理システムを提案する．本研究では，サービスどうしの機能面での関連性の記述については，既存のオントロジ記述言語を用いることとし，もう 1 つの課題である入出力インタフェースの相違を吸収する，インタフェース変換・適応の枠組みを提案する．これにより，複数のサービス間での切替え・相互接続を可能とするフレームワークを構築する．

以下，2 章で研究課題や前提，関連した研究について述べる．3 章でサービスグループ管理システムの内容に触れ，特に変換パス生成方法について説明する．4 章でシステムの設計，5 章でプロトタイプの実装内容や動作について述べ，6 章で現状の問題点や今後の課題をまとめる．

2. 背　　景

2.1 課　　題

前述のとおり，サービスどうしの関連性を記述し，機能的に等価なサービスを検索するという課題があるが，これについては既存のオントロジ関連技術[1)~3)] を流用して解決するものとし，本研究では対象としない．

一方，サービスの利用先を他のサービスに切り替えることでユーザの状況変化に適応しやすくなるが，入出力インタフェースの相違のために，相互接続性を保証することが難しいという課題がある．現状の Web サービスでの入出力インタフェースの相違には以下のようなものが見受けられる．

- **入出力要素の型や数，順番**：入出力要素の型や順番，要素数などに違いがある場合，そのままでは切り替えることができない．
- **入出力要素の内容**：型や順番が同じでも，要素の内容が違う場合がある．たとえば，同じ「name」という文字列要素でも，フルネームを入れるべきか，苗字だけでよいかという違いが生じる．数字の単位や，文字列の省略形などについても同様の問題がある．
- **クラス要素**：Web サービスでは，入出力要素に複数の変数をまとめたクラスを指定することがある．この場合は要素を分割し，個別に型や内容の検証を行う必要がある．
- **不足要素**：また，2 つのサービス間で入出力要素の有無に違いがある場合がある．たとえば，あるサービスでは名前を文字列で入力する必要があり，もう一方では必要ない場合などである．この場合，該当する要素の必要ないサービスから必要なサービスに切り替えるためには不足分を補う必要がある．

これらの相違を吸収するためには入出力要素の変換モジュールを用意し，要素の変換を自動的に行う必要があり，処理をクライアント端末から完全に隠蔽するのが理想的である．また，すべてのサービスごとに変換モジュールを作りこむことは非現実的であるため複数の変換モジュールを組み合わせ，インタフェースを変換するサービスのパスを生成する機能が必要になる．

よって本研究ではサービスグループ管理システムの機能の 1 つとしてインタフェース変換パス生成機能を実現し，上述の課題の解決を目指す．

2.2 関 連 研 究

サービス合成，特に分散サービスの相互接続性を課題とした関連研究が積極的に行われている．

DAML プロジェクト[1)] は，DAML-S や OWL-S といった分散サービスのオントロジ記述言語を開発することでセマンティックなサービス発見や組合せを模索している．また，サービスの入出力インタフェースについてもオントロジ記述を行い，サービスどうしの接

続性の検証などを試みている[2]．DAML-S を用いることでインタフェースマッチングが可能になるが，相違が存在する場合にそれを吸収する手法が用意されていないため，サービス切替えや合成を行うことができないという問題がある．また，サービスの入出力要素についてのオントロジ記述は，サービス提供者に対して大きな負担になる．

また，Ninja プロジェクト[4]では分散コンポーネントの協調時に適切なプロトコル変換パスを生成し，サービスを提供する Ninja Paths の研究が行われている．インタフェース変換モジュールを動的に発見し接続するため，インタフェースの相違を吸収できる利点があるが，すべての分散コンポーネントが NinjaPaths の提供する基底クラスを継承して実装される必要があり，制約が大きいという問題がある．

Ja-Net プロジェクト[5),6)]では，インタフェースの相違を吸収するためのインタフェース記述言語を提案している．ConditionSet と呼ばれる要求条件をブロードキャストし，インタフェースマッチングを行うが，ConditionSet の条件に合わない場合は適応できない．また，入出力要素の変換などは考慮されていないため，前述の課題のように要素の内容に違いがある場合は切替えが不可能である．

MINDSWAP プロジェクト[7]では，Web サービスの入出力インタフェース要素について DAML-S を用いてオントロジ記述を行い，サービス間のインタフェースマッチングを実現している．入出力インタフェースが意味的に近いサービスから順に表示され，切替えや相互接続が可能なサービスを検索できる．また，WSDL[8]への Grounding を行い，サービスの自動実行まで行うが，本研究の課題であるインタフェースの相違の吸収や変換はサポートしていない．

表 1 に示すようにインタフェース変換（入出力要素変換）の課題を解決しているのは Ninja Paths だけだが，前述のように実装の際の負担が大きい．提案する手法ではインタフェース変換の課題を解決し，他の分散サービスアーキテクチャよりも柔軟性の高いサービス切替えや接続を可能にすることを目的とする．また同時に，サービス機能や入出力要素についてのオントロジ記述の利用や，サービス提供者に特別な負荷をかけないなど，高い実現可能性を目指す．

表 1 他の研究との比較
Table 1 Comparison among related researches.

	Ja-Net	Mindswap	Ninja
実現可能性	○	△	×
動的なフロー構築	○	○	△
入出力マッチング	○	○	○
入出力のオントロジ	×	○	×
入出力要素変換	×	×	○
切替え・接続の確実性	○	○	△

2.3 前提条件

サービスにはメタデータが付加され，サービスの機能についてのオントロジ記述や，各入出力要素についてのオントロジ記述がなされるとする．オントロジの記述により，たとえばサービス A の入力要素とサービス B の入力要素が意味的に同じか，サブセットなのか，表現上は等価であるかどうか，といった判断をシステムが下すことが可能になる．

また，プリミティブな要素変換モジュールが Web サービスとして実装され，サービスディレクトリに登録されているとする．たとえば現状でもすでに単位変換や通貨変換のモジュールなどが存在する[9]．これらの要素変換モジュールについても上記のような入出力要素についてのオントロジが記述されているとする．

サービスのメタデータ記述にはオントロジ記述言語である DAML-S を用いる[2]．また，サービス提供者はメタデータ内に，自身のサービスの機能と，入出力要素についてのオントロジ記述を各要素ごとに行うものとする．具体的には公開されているオントロジツリーへのリンクを DAML-S を用いて記述することになる．

3. サービスグループ

3.1 アプローチ

本研究では似た機能を持つサービスの集合をサービスグループ（以下，SG）と定義し，SG 単位でのメタサービス検索や，SG 内でのインタフェース適応を実現する．SG の主な機能には，サービスの登録・更新，サービスの利用，サービスの切替えなどがあるが，本研究で主に取り扱うインタフェースの相違に関する課題は 3.3 節のインタフェース適応機能で解決する．

以下，3.2 節ではシステム全体のフローを述べる．3.3 節ではサービス切替え機能とそのために必要なインタフェース適応処理について述べ，3.4 節では生成されたインタフェース変換パスを連続実行するプロキシモジュールについて説明する．

3.2 全体のフロー

まず，サービス提供者は自身のサービスを公開する際に，既存の SG に参加するためにサービスのメタデータを記述し，SG 管理システム（SGM: Service Group Management System）に登録する（**図 2**）．サービス提供者はメタデータに少なくとも 1 つ以上の機能的に等価な既存のサービスの URI を記述し，サービス

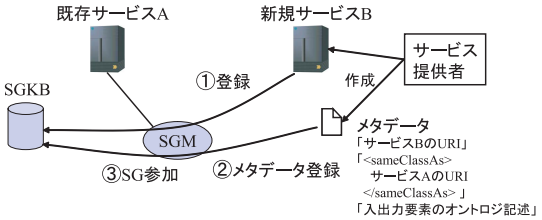

図 2 サービスグループへの新規参加
Fig. 2 Adding new service to Service Group.

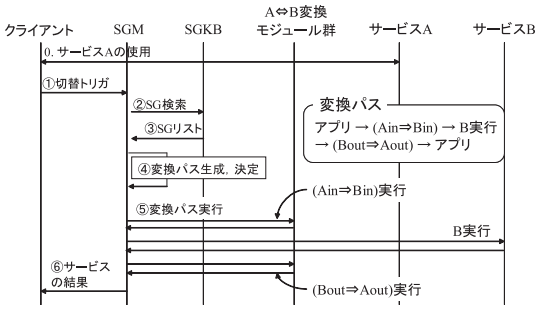

図 3 インタフェース変換とサービス切替えの手順
（サービス A からサービス B への切替えの例）
Fig. 3 Interface conversion and switching service (ex. switch from service A to B).

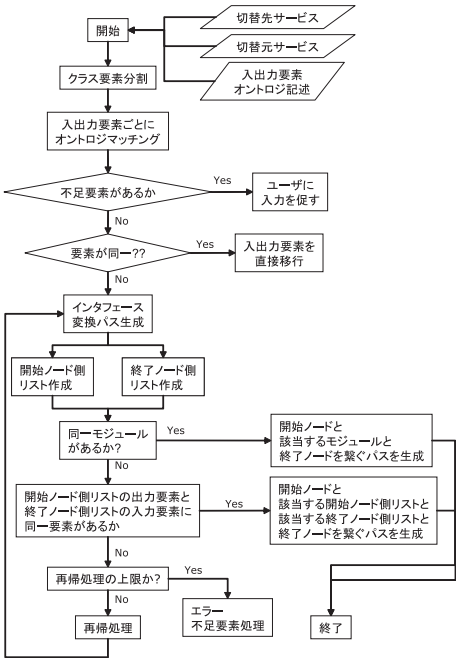

図 4 インタフェース変換フロー
Fig. 4 Flowchart of interface conversion.

グループ知識ベース（SGKB）に登録する．SGM は SGKB を参照し，サービスどうしの関連性を把握して SG を構築する．

クライアント側から切替えのトリガが発生した場合，まず SGM が SGKB にアクセスし，等価サービスのリストである SG リストを取得する（**図 3**）．次に，SGM は SG リストの中から切替え候補のサービス一覧を生成し，それぞれのメタデータの入出力要素記述を参照して，必要に応じて単純な要素変換モジュールをサービスディレクトリから検索し，インタフェース変換パスを生成する．SGM は切替え候補サービスごとに生成した変換パスを評価し，最も早く切替えが可能な変換パスと切替え先サービスを選択する．最後に，SGM は選択した変換パスに沿って入出力インタフェース要素を変換し，サービス切替えを実行する．

この手法では SGM に接続しているクライアントアプリケーションや Web サービスクライアントからはサービス切替えが隠蔽されるため，クライアント側の変更は必要ないというメリットがある．また，切替え元，切替え先サービスもともに既存のものが使用可能である．

サービス切替えだけでなく，複数のサービスを接続して使用したいという場合にも SGM を介してインタフェース変換パスを生成することで，相互接続性を高めることができる．この場合，クライアントは SGM に対して使用するサービスのフローを通知する．サービスフローにはどのサービスの出力要素をどのサービスの入力要素とするのか，といった内容が記述される．SGM はサービスフローに従ってインタフェース変換パスを生成した後に，サービスの連続実行を行って最終的な結果をクライアントに返す．

3.3 インタフェース適応

サービス切替えの際にインタフェースの相違が相互接続性を妨げているため，インタフェース適応の手法が必要であることは 2.1 節で述べたとおりである．このうち，最初の 2 つの「要素の型や順番，内容」の課題については既存の変換モジュールを検索してインタフェース変換パスを生成することで解決する．3 つ目の「クラス要素」の課題については，インタフェース要素のオントロジ記述から要素を分割・再構築することで解決を図る．最後の「不足要素」の課題は，ユーザに入力を促すことで解決を目指す．**図 4** にインタフェース適応のフローを示す．

まず，インタフェース変換パス生成の手順を説明する．図 4 の下半分に処理のフローを示す．また，**図 5** に具体例を示す．図 5 はレストラン検索サービス A から同様のサービス B へ処理を切り替える例で，サービス A は日本円を入力可能なサービス，サービス B はポルトガルの通貨であるエスクードを入力可能なサー

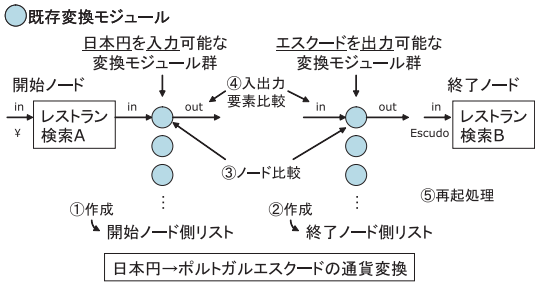

図 5 インタフェース変換パス生成（通貨変換の例）
Fig. 5 Creating interface conversion path (currency exchange).

ビスである．以下では，切替え元サービス A を開始ノード，切替え先サービス B を終了ノードと定義し，開始ノードの入力要素である日本円を終了ノードの入力要素であるエスクードに変換するパスを生成する手順を示す．また各番号は図 5 内の番号に相当する．

(1) 開始ノードの入力要素を入力可能な変換モジュールを検索する（開始ノード側リスト）．
(2) 終了ノードの入力要素を出力可能な変換モジュールのリストを作成する（終了ノード側リスト）．
(3) 両リストを比較し，同一ノードがあった場合はそれを変換モジュールとして変換パスを生成し，終了．
(4) ない場合は開始側ノードリストの出力要素と終了側ノードリストの入力要素を比較する．同一要素があれば，該当する変換モジュール 2 つをつないで変換パスを生成し，終了．
(5) ない場合は開始側ノードリストのノードを開始ノードに，終了側ノードリストのノードを終了ノードに設定し，再帰的に処理を行う．

以上の手順を逆方向についても行い，終了ノードの出力要素を開始ノードの出力要素に変換する変換パスを生成する．実用上は再帰処理の回数をあらかじめ決めておき，その範囲内でパスが見つからない場合はエラーとする処理が必要と考えられる．

変換パスが複数見つかった場合，最適な変換パスの選択にはインターネットで用いられる既存のルーチングアルゴリズムが適用できる．通常のルーチングではホップ数や帯域幅を考慮して最適経路を探索するが，本方式では変換モジュールの実行時間を基にルートに重み付けを行い，Dijkstra のアルゴリズム[10] などに代表される最適経路探索アルゴリズムを用いる．これによって，実行時に最も短い時間でインタフェース変換が完了する変換パスを探索することができる．

次にインタフェース要素の分割・再構築について説明する．SGM は SGKB に問い合わせ，サービスの

```
<rdf:Description about="s:GourmetNavi.getShop">
  <drd:Input rdf:resource="mml:InDataClass">
    <drd:Type>class</drd:type>
    <drd:Unit>
      <drd:Name rdf:resource="mml:genre" />
      <drd:Unittype rdf:resource="xsd:string" />
    </drd:Unit>
    <drd:Unit>
      <drd:Name rdf:resource="mml:jpy" />
      <drd:Unittype rdf:resource="xsd:int" />
    </drd:Unit>
  </drd:Input>
  <drd:Output rdf:resource="mml:OutDataClass">
    …
  </drd:Output>
</rdf:Description>
```

図 6 入出力インタフェース要素のオントロジ記述
Fig. 6 Ontology description of I/O elements.

URI からインタフェース要素のオントロジ記述を取得する．**図 6** にオントロジ記述の具体例を示す．図 6 はレストラン検索サービスの例で，入力要素として mml:InDataClass が記述され，その構成要素として，店のジャンル（mml:genre）や価格帯（mml:jpy）を入力するように記述されている．このようなオントロジ記述を参照することでインタフェース要素がプリミティブなデータ要素か，クラス要素かの情報，クラスであれば内部の構成についての情報を得ることができる．これを基に要素の分割・再構築を行い，要素ごとにインタフェース変換パスを生成してインタフェース適応を行う．

次に不足要素の課題について説明する．切替え先サービスで必要な要素が切替え元サービスで不要だった場合，補完しなければならないが，これを自動的に補うことは難しいため，ユーザからの入力を促す処理を行う．インタフェース要素のオントロジ記述を参照し，マッチングすることで不足する要素が発見された場合，SGM はクライアントに不足要素の情報を返す．クライアント側では情報を基にユーザからの入力を促すダイアログボックスなどを表示し，不足要素を補完する．

また，SGM は以下にあげる機能を持つ．

- 一度探索したパスを保存し，2 回目以降の探索処理を省略する．
- 各モジュールの実行時間を監視し，著しく変化した場合には，最適パスを探索しなおす．

インタフェース適応機能により，様々なインタフェースの相違の課題を解決し，サービス提供者に特別な負担を強いることなくサービスの切替え処理が可能になる．ユーザは自身のコンテキスト変化に応じて自由にサービスの切替えを行えるため，ユーザビリティの向上につながる．サービス提供者にとっては自身の顧

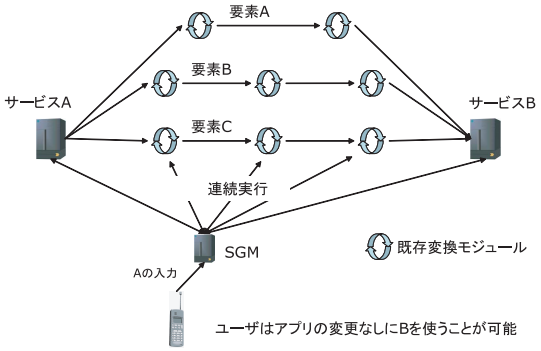

図 7 入出力インタフェース要素の変換
（図には入力要素のみ図示．出力要素も同様に変換する）
Fig. 7 Converting I/O interface (Figure shows only input. Output must be also converted).

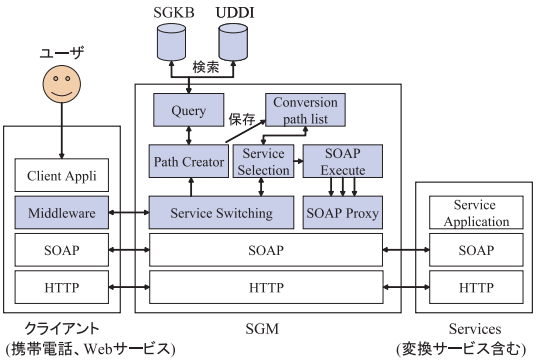

図 8 リッチクライアントモデルのモジュール構成
Fig. 8 Software architecture of rich client model.

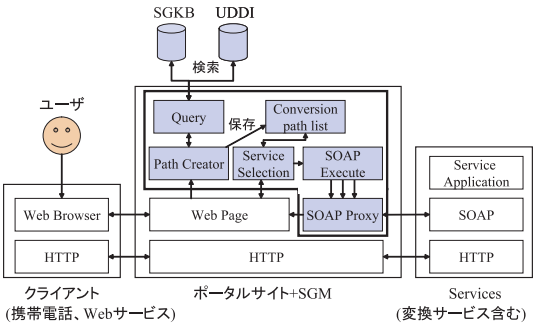

図 9 シンクライアントモデルのモジュール構成
Fig. 9 Software architecture of thin client model.

客が他のサービスを利用する可能性が出るというデメリットがある一方で，他のサービスからユーザが流入してくるメリットが生じる．

3.4 サービス切替え処理

インタフェース変換パスを生成した後，SGM は入出力要素ごとに変換モジュールを実行し，サービス切替えを行う．手順を**図 7** に示す．

切替え時は，要素ごとに入力と出力の双方で変換パスを経由し，変換モジュールの実行はすべて SGM が行う．これにより，サービス切替えが実行されても，クライアントからは切替えが隠蔽され，あたかも既存サービス A を使用しているように扱えるため，アプリケーションの変更は必要ない．

4. システム設計

図 8，**図 9** にモジュール構成を示す．携帯端末の能力やアプリケーションは多種多様であり，クライアント端末上で高機能アプリケーションを動かすリッチクライアントモデル（図 8）と，ブラウザのみが稼動するシンクライアントモデル（図 9）を検討する必要がある．以下，図 8 を用いて機能を説明する．

クライアント上では，Web サービスのクライアントとなるアプリケーションが端末ミドルウェアの上で動作する．ミドルウェアは SGM 上のサービス切替えモジュールと協調し，切替えのトリガや SOAP メッセージを SGM に送信する機能を提供する．ライブラリ形式で提供することで，たとえばサービス切替えボタンをアプリケーションが実装できるような形態を検討中である．

SGM にはサービス切替えモジュール，変換パス生成モジュール，クエリモジュール，サービス選択モジュールと SOAP 実行モジュールが実装される．サービス切替えモジュールは端末ミドルウェアからの切替えトリガを受け，変換パス生成モジュールを起動する．変換パス生成モジュールはクエリモジュールを介して知識ベースから SG リストを取得し，切替え先サービスとそれを実行するために必要な変換パスのリストを生成した後に変換パスリストに保存し，サービス選択モジュールに送信する．サービス選択モジュールは切替えモジュールから指定されるポリシ☆に従って切替え先サービスと変換パスを選択し，SOAP 実行モジュールに伝える．SOAP 実行モジュールは SOAP プロキシコードを生成し，選択された変換パスに従って順番に変換モジュールと切替え先サービスを実行する．サービスの実行結果はサービス切替えモジュールから端末ミドルウェアに送信され，クライアントアプリケーションは処理を完了する．

アプリケーションの機能を提供する Web サービスには変更を加えない（図 8 右側）．また，変換モジュールも 2.3 節の前提条件で述べたように Web サービス

☆ たとえば切替え元サービスとのオントロジツリー上の距離などがある．

表 2 各サービス，変換モジュール一覧
Table 2 List of services and convertors.

サービス	メソッド名	Out 要素	型	オントロジ	In 要素	型	オントロジ
A	getShop	クラス	OutData	mml:OutDataClass	クラス	InData	mml:InDataClass
B	getShopName	店名	String	mml:ShopName	ジャンル	String	mml:genre
					エスクード	String	mml:pte
	getShopAddr	住所	String	mml:ShopAddr	ジャンル	String	mml:genre
					エスクード	String	mml:pte
	getShopTel	電話番号	String	mml:ShopAddr	ジャンル	String	mml:genre
					エスクード	String	mml:pte
C	getShopName	店名	String	mml:ShopName	ジャンル	String	mml:genre
	getShopAddr	住所	String	mml:ShopAddr	ジャンル	String	mml:genre
ConvA	jpy2usd	US ドル	int	mml:usd	日本円	int	mml:jpy
	usd2jpy	日本円	int	mml:jpy	US ドル	int	mml:usd
ConvB	pte2usd	US ドル	int	mml:usd	エスクード	String	mml:pte
	usd2pte	エスクード	String	mml:pte	US ドル	int	mml:usd

注 1. mml:http://mml.yrp.nttdocomo.co.jp/# 注 2. エスクードはポルトガルの通貨である．

として実装され，SOAP でアクセス可能とする．

図 9 のシンクライアントモデルでは，SGM の機能はすべてサーバサイドで実現され，端末側に機能を追加する必要がないため，現行の携帯機でも実現可能である．

5. プロトタイプシステムの構築と評価実験

5.1 評価項目

本論文で提案する Web サービスグループ管理システム，およびインタフェース変換・適応処理をリッチクライアントモデルで実装し，手法の妥当性を検証するために評価実験を行った．まず，2.1 節であげた課題について機能を確認するため，入出力インタフェースの異なるサービス間のサービス切替えを行い，クラス要素の分割や型変換，内容変換，不足要素が生じた際の処理が正しく行われることを確認した．これは，前章で述べた設計手法を用いて提案手法を正しく実現できるかどうかを検証するためである．また，サービス切替え時に切替え処理に要する時間を SGM 上で測定した．サービスの切替えに要する時間は，サービスの応答時間に直接影響するため，短いことが望ましい．これを測定することでシステムのオーバヘッドを測定するとともに今後の設計の指針として用いる．

5.2 プロトタイプ構成

提案手法をリッチクライアントモデルで実装しサービス切替えの実験を行った．**図 10** に実験環境を示す．実験用にレストラン検索の Web サービスを 3 つ，変換モジュールを 2 つ実装した．リストを**表 2** に示す．サービス A は入出力に複数の要素をまとめたクラス要素を用いるサービスである．サービス A の用いる入出力のクラス要素を**図 11** に示す．サービス B は A と同数の入出力要素を持つが，プリミティブな型のみ

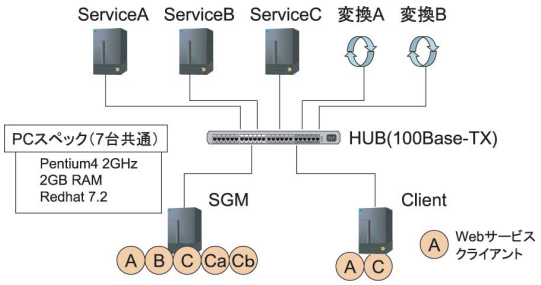

図 10 プロトタイプシステムの構成
Fig. 10 Prototype system.

```
public class InData{
   public String genre;  //ジャンル
   public int jpy;       //値段（日本円）
}
public class OutData{
   public String shopname; // 店名
   public String address;  // 住所
   public String tel;      // 電話番号
}
```

図 11 サービス A の入出力クラス
Fig. 11 I/O class members of service A.

を扱うため，出力要素の数のメソッドが用意されている．また，扱う通貨が A と違うため，通貨変換を必要とする．サービス C はプリミティブな型のみを扱い，入出力要素が A，B に比べて少ない．変換モジュールは日本円と US ドルの通貨変換（ConvA）と US ドルとポルトガルエスクードの通貨変換（ConvB）を用意した．また，SGM 上にはすべての Web サービスを扱えるクライアントモジュールを用意し，クライアント端末上には切替え元のサービスを扱うクライアントモジュールのみを用意した．さらに，インタフェースマッチングのために非常に簡単なオントロジツリーを用意した．

すべての実装は Java で行い，Pentium4 2 GHz,

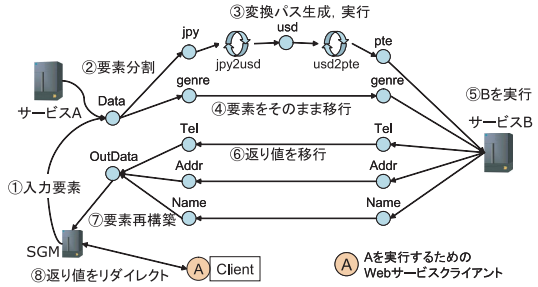

図 12 サービス A から B への切替えフロー
Fig. 12 Switching flow from service A to B.

2 GB RAM，Redhat Linux 7.2 のデスクトップ PC7 台を 100 Base-TX の HUB に接続した環境で実験を行った．

5.3 サービス切替え実験

2.1 節であげた課題について機能を確認するため，まずサービス A からサービス B への切替えを行い，クラス要素の分割や型変換，内容変換を行う．次にサービス C から B への切替えを行い，不足要素が生じた際の挙動を確かめる．最後に，SGM 上で測定した切替え時間について述べる．

まずサービス A から B への切替え（**図 12** 参照）では SGM はサービス A のメタデータから入力要素の構造を取得し，プリミティブな要素に分割する (2)．SGM は各要素をサービス B の入力要素とオントロジ上で比較する．ここで価格を表す mml:jpy はサービス B の mml:pte とオントロジツリー上では同一ツリー内に存在するが，クラスが一致しないため，変換パス生成が行われて「mml:jpy → jpy2usd → mml:usd → usd2pte → mml:pte」というパスが生成される (3)．一方の要素であるジャンルはオントロジ上でクラスが一致するためそのまま移行される (4)．SGM は変換パスを実行し，サービス B の入力要素に変換を行った後，サービス B の各メソッドを実行し，結果を得る (5)．SGM は B のそれぞれの出力要素を A の出力要素と比較し，一致したので出力要素をそのまま移行し (6)，サービス A の出力要素である OutData クラスに再構築してクライアントアプリケーションに返す (8)．

次にサービス C から B への切替えでは入力要素の中に不足分が検出される．サービス C を扱うクライアントは mml:genre を入力することはできるが，もう 1 つの mml:pte は入力できない．そこで SGM はクライアント端末にフィードバックをかけ，クライアント端末上のアプリケーションが画面に入力を促す．ユーザが入力要素を入力すると SGM はサービス C の 2 つの出力要素を基に実行メソッドを決定し，出力要素のマッチングを行ってから切替えを実行する．サービス B と C の間には型やオントロジの違いを持たせなかったため，特に変換パスが生成されることなく処理は終了した．

表 3 実行時間測定結果
Table 3 Result of execution time.

切替えフロー	メソッド	実行時間（ms）
A → B	SOAP スタブ準備	1,056.6
	InData 要素分解	0（計測不能）
	ConvA.jpy2usd	489.4
	ConvB.usd2pte	38.6
	ServiceB.getShopTel	118.6
	ServiceB.getShopAddr	19.0
	ServiceB.getShopName	27.0
	OutData 再構築	0（計測不能）
	合計	1,749.2
C → B	SOAP スタブ準備	1,073.2
	ServiceB.getShopName	473.8
	ServiceB.getShopAddr	32.2
	合計	1,579.2

以上の実験により，インタフェース適応を行うことでサービス切替えを自動的に行い，かつクライアントから隠蔽して実行できることを確認した．また，既存の Web サービスやクライアントアプリケーションを変更する必要がないことや，プリミティブな変換サービスのみを流用して処理が行えることを確認した．

次にサービス切替えに要する実行時間を測定した（**表 3**）．今回はネットワークの遅延をできる限り除き，切替え先サービスと変換モジュールの実行時間のみを取得するため，すべてのノードを LAN 内に設置して実験を行った．表 3 に示したように全体の切替え時間に対して SOAP のスタブ生成など，Web サービスの実行準備がオーバヘッドとなった．実行準備が整った後はサービスによってばらつきはあるものの，0.5 秒以内に処理が終了し，全体では 2 秒以下で切替えが終了した．今回の実験では変換パスが短いので，短時間で切替え処理が行われたが，実際にはパスが長くなることが考えられるため，切替え時間の短い変換パスを発見することは重要と考えられる．

6. おわりに

本論文では，モバイルコンピューティング環境における分散サービスのユーザビリティを向上させるため，同じような機能を持つサービス群をサービスグループとしてまとめ，グループ内のインタフェース変換・適応処理を行うサービスグループ管理システムを提案した．提案手法はサービスの入出力要素を要素ごとに変換することで，インタフェースの相違を吸収し，柔軟

なサービス切替えや相互接続を可能にする．また，既存の要素変換モジュールを用いてインタフェース変換パスを生成するため，サービス提供者に負担をかけずに機能を実現できる．

今後，本手法の有効性を検証するために大規模なプロトタイプシステムを実装し，性能評価実験を行う．特に，変換パスを生成するために要する時間や，生成された変換パスのホップ数とサービス実行時の処理のオーバヘッドの関係などについて実験を行い，手法の有効性と問題点を検証する予定である．

現在の主な問題点としては以下の 2 点がある．

- **インタフェース変換の不確実性**：すべての場合に備えて変換モジュールが用意されることは期待できないため，インタフェース変換が必ずしも成功するとは限らないという問題点がある．また，変換モジュールが正しく変換を行うかどうかを検証する必要もある．
- **変換パス生成に要する時間**：変換パス生成をフラッディングで行うため，要する時間が全サービス数の 2 乗のオーダで増加しており，スケーラビリティに課題を残している．現在のシステムでは一度探索したパスを保存し，2 回目以降からは時間をかけずに変換パスを提供できるが，サービスの停止や新規サービスの参加など環境は動的に変化していくため，スケーラビリティの課題は無視できない．

今後，以上の課題を解決し，より柔軟で自由度の高いサービス利用環境を構築していきたい．

参 考 文 献

1) DARPA: DAML+OIL web page. http://www.daml.org/
2) Ankolekar, A., et al.: DAML-S: Web Service Description for the Semantic Web, *The 1st International Semantic Web Conference* (*ISWC 02*) (2002).
3) W3C: Resource Description Framework (RDF). http://www.w3.org/RDF/
4) Chandrasekaran, S., Madden, S. and Ionescu, M.: Ninja Paths: An Architecture for Composing Services over Wide Area Networks, CS262 class project writeup, UC Berkeley (2000).
5) 須田達也，板生知子，中村哲也，松尾真人：サービス創発のための適応型ネットワークアーキテクチャ，信学論（B），Vol.J84-B, No.3, pp.310–320 (2001).
6) Fujii, K. and Suda, T.: Loose Interface Definition: An Extended Interface Definition for Dynamic Service Composition, *1st Annual Symposium on Autonomous Intelligent Networks and Systems* (2002).
7) Sirin, E., Hendler, J. and Parsia, B.: Semi-automatic Composition of Web Services using Semantic Descriptions, *International Workshop on Web Services: Modeling, Architecture and Infrastructure* (2003).
8) W3C: Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/wsdl
9) XMethods, Inc.: XMethods. http://www.xmethods.net/
10) Dijkstra, E.W.: A note on two problems in connexion with graphs, *Numerische Mathematik*, No.1, pp.269–271 (1959).

(平成 15 年 12 月 26 日受付)
(平成 16 年 10 月 4 日採録)

推 薦 文

本論文は，モバイルコンピューティング環境下で機能的に等価なサービスや相互接続可能なサービス，また相互に信頼関係にあるようなサービスの集合体をサービスグループとして定義し，その動的な生成，管理を行う機構を提案している．サービスはオントロジ記述言語で関連づけられ，入出力インタフェースの不一致を解決する機能を持つ．分散サービスにおける相互接続や切替えを可能とするフレームワークを明らかにしており，優秀な論文として推薦する．

（DICOMO2003 プログラム委員会委員　水野忠則）

吉川　貴（正会員）

平成 11 年慶應義塾大学総合政策学部卒業．平成 13 年同大学大学院修士課程修了．同年（株）NTT ドコモ入社．以来，モバイルコンピューティングに関する研究に従事．現在，同社マルチメディア研究所勤務．

太田　賢（正会員）

平成 6 年静岡大学工学部情報知識工学科卒業．平成 8 年同大学大学院修士課程修了．平成 10 年同博士課程修了．平成 11 年 NTT 移動通信網（株）入社．現在，(株) NTT ドコモマルチメディア研究所勤務．平成 9 年度日本学術振興会特別研究会特別研究員．モバイルコンピューティングに関する研究に従事．

中川 智尋（正会員）

平成 10 年東京大学工学部電子情報工学科卒業．平成 12 年同大学大学院修士課程修了．同年（株）NTT ドコモ入社．以来，アドホックネットワーク，モバイルサービス環境の研究に従事．現在，同社マルチメディア研究所勤務．

倉掛 正治（正会員）

昭和 58 年東京大学工学部計数工学科卒業．昭和 60 年同大学大学院修士課程修了．同年日本電信電話（株）入社．平成 2 年より 1 年間，米国南カリフォルニア大学に客員研究員として滞在．平成 12 年より 2 年間，米国 DoCoMo USA Labs に勤務．文字認識，コンピュータビジョン，ユビキタスコンピューティング，モバイルサービス環境の研究に従事．現在，（株）NTT ドコモマルチメディア研究所主幹研究員．IEEE，ACM 各会員．