卒業論文 2002年度 （平成14年度）

Web Service連携支援システムの設計と実装

慶應義塾大学

鈴木貴晶

nir@sfc.wide.ad.jp

指導教員

村井 純

徳田 英幸

楠本 博之

中村 修

南 政樹

平成 15 年 2 月 22 日

卒業論文要旨 2002年度 (平成14年度)

Web Service連携支援システムの設計と実装

本研究では、Web Serviceを連携して利用する事を支援するシステムの設計と実装を行った。

インターネットを利用したプログラムの再利用技術として、また、新たなプログラムアーキテクチャの構築基盤として、Web Serviceが最近注目されている。Web Serviceの利点のひとつに、「複数のWeb Serviceをつなぎ、新たなWeb Serviceとして利用できる」事がある。この様なWeb Serviceの連携に関連する既存システムには、UDDIやWSDLがある。しかし、入出力データについての情報不足、利用方法についての情報未整理、Web Serviceを動的に利用し、連携を支援するためのWeb Serviceクライアントの不在という問題を抱えているため、連携を図る事が難しい。

本研究では、Web Serviceへの入出力データについての情報や利用方法を追加情報データベースという形で持つ事により、既存のシステムと組み合わせる事でWeb Serviceの連携を支援するWeb Serviceクライアントを持つシステムを設計、実装した。実装したシステムが、実際にWeb Serviceを連携可能か、評価を行った結果、既存システムとの優位性を実証するに至った。

キーワード
1. Web Service, 2. UDDI, 3. WSDL,

慶應義塾大学 環境情報学部

鈴木 貴晶

Abstract of Bachelor's Thesis

Academic Year 2002

Design and Implementation of
a system supporting web service interconnection

The aim of this study is to design and implement a system that supports the utilization of interconnected web services.

These days web services are becoming more and more popular as application components accessible through the Internet and as the infrastructure for new program architectures .One of the merits of web services is that we can connect several web services with each other and use them as a new web service. Among such systems for the interconnection of web services are UDDI and WSDL. There are three major problems in such interconnecting systems: the lack of information about input and output data;the lack of informational arrangement for users to utilize web services;and lack of web service clients that support the interconnection by the dynamic utilization of web services.

In this thesis, I designed and implemented the system with the web service client that supports web service interconnection by storing the information about those input and output data and about their utilization as an additional information database, and combining this database with the existing systems. I evaluated the ability of the implemented system to interconnect web services and demonstrated its significant advantage over the existing systems.

Keywords :
1.Web Service, 2.UDDI, 3.WSDL,

Keio University , Faculty of Environmental Information

Takaaki Suzuki

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景

現在、インターネット上には、Web application による利用者に情報を提供するサービスが多く存在している。そのサービスの内容は、マピオン (1) などの地図サービス、乗換案内 (2) などの電車情報サービス、日本高速道路公団 (3) の渋滞情報サービスなど、多種多様である。これらのサービスは、家庭へのインターネットの普及や携帯電話の普及により、生活の様々な場面で利用される様になった。例えば、旅行に行くという場合、利用者は、電車情報サービスや飛行機情報サービスで、交通手段を探る事がある。更に、現地で泊まるホテルの場所や旅行先での食事などの情報を調べる為に、地図サービスやグルメ情報サービスなどを利用したりする事もある。

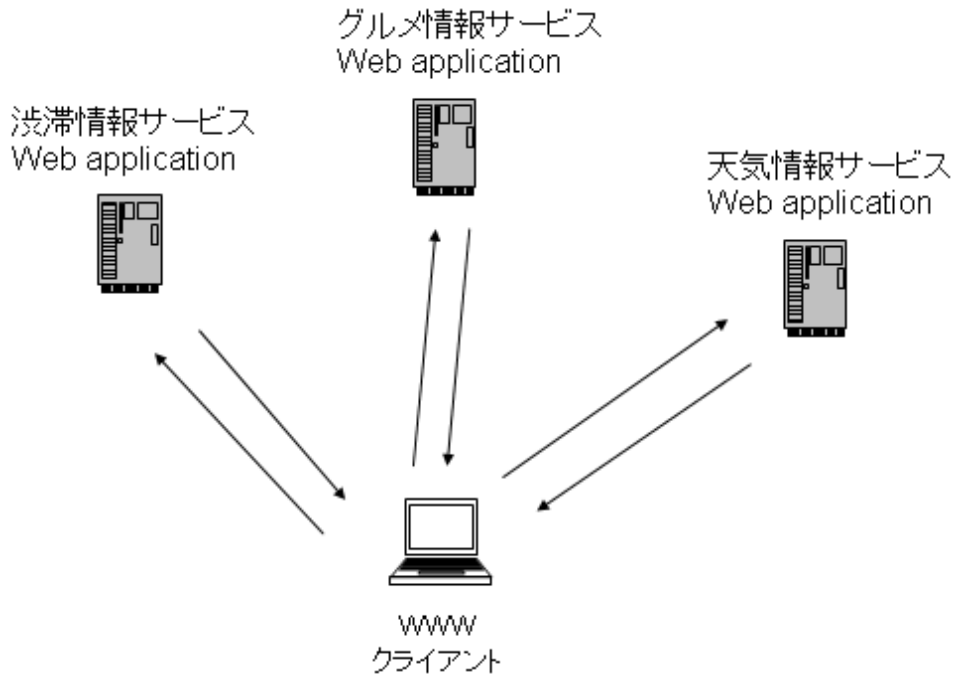

図 1.1: Web application の世界

この様な Web application は、複数の機能によって一つの Web application が構成されている。本を販売する Web application を例にあげると、認証機能・在庫検索機能・

書籍情報提供機能・宅配機能などといった機能によって構成されている。しかし、機能自体はあらかじめ application の提供者によって決められた形でしか利用出来ない。先の本の販売を例にあげれば、宅配機能として用意されているものは運送会社 A だけであり、より細かい配送日指定が可能な運送会社 B のサービスを利用したいというエンドユーザーの要求を満たす事は出来ない。この要求を満たす為には、機能一つ一つが個別に提供され、その機能を任意で組み合わせる事が可能でなければならない。これを可能にするものとして Web Service があげられる。Web Service は、http などのプロトコルを使って呼び出す事が出来るリモートメソッド群であり、複数の Web Service を自由に組み合わせる事で高機能であり付加価値の高い Web applicatio を構築する事が出来る。

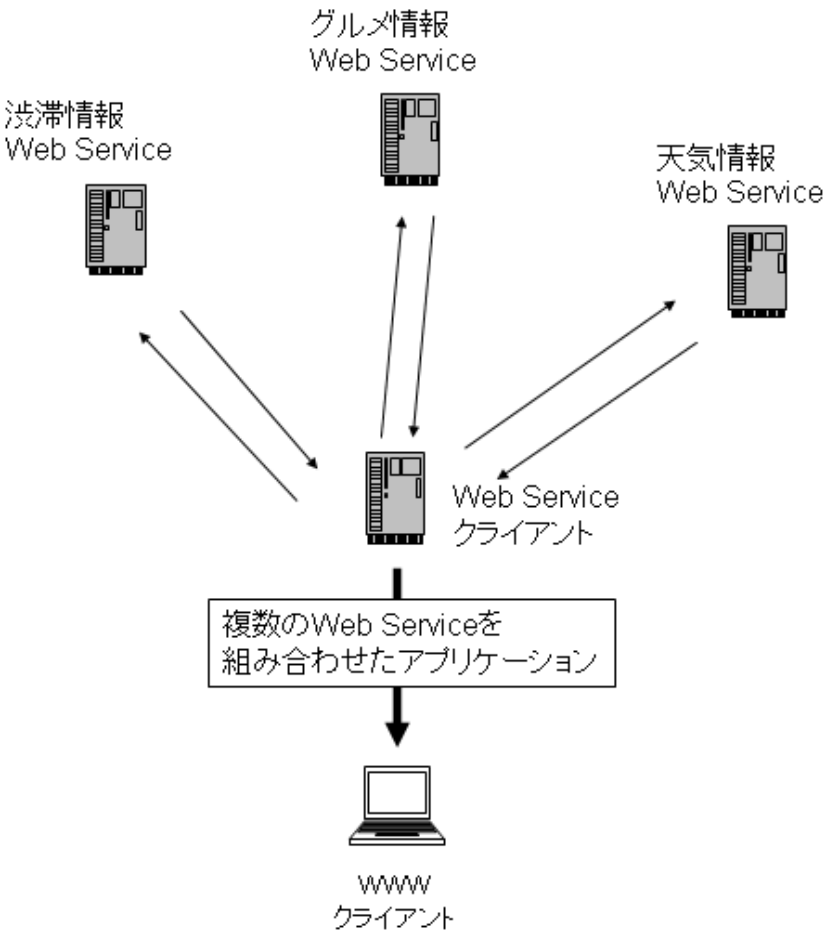

図 1.2: Web Service の世界

1.2 問題意識と本研究の目的

前節で述べた様に、複数の Web Service を自由に組み合わせる事が可能であれば、エンドユーザーは自分の要求を反映した Web application を利用する事が出来る様になる。しかし、現在の Web Service の利用形態は、Web application 提供者が、その組み合わせをあらかじめ決定して Web application を作り上げ、エンドユーザーに提供するという形態である。この為、利用の自由度が奪われてしまっており、エンドユーザーが複数の WebServeice を自由に組み合わせる事は困難である。

そこで、本研究では

- 動的に Web Service を発見し利用する仕組み
- 組み合わせの自由度を高める為の Web Service 連携の流れを組み立てる仕組み

の二つを構築する事で、この問題を解決し、自由な Web Service の連携を可能にする事を目的とする。

1.3 本論文の構成

本論文は全 6 章で構成される。

第 2 章では、既存システムの問題点と本研究のアプローチついて述べる。第 3 章では、本研究で実装する Web Service 連携支援システムの設計について、第 4 章ではその実装について述べる。第 5 章では、本研究の実装が問題点を解決したか評価を行う。第 6 章では、本論文での結論と今後の研究課題について論じる。

第2章　既存システムの問題点と本研究のアプローチ

本章では、まず Web Service とその関連技術について述べる。そして、前章二節で述べた二つの仕組みを構築する為の本研究のアプローチとその際の既存システムの問題点と解決手段について述べる。

2.1　Web Service と関連技術

2.1.1　Web Service とは

Web Service の一般的な定義は、インターネット標準の http などの各種 Web プロトコルを利用してアクセス可能なアプリケーション・コンポーネントである。外部とのインターフェースには XML を用い、Web Service によってインターネット上に様々なコンポーネントが存在し利用出来る事が可能になる。

個々の Web Service は、株価情報の提供や郵便番号の検索といった具体的な機能を持っており、図 2.1の様に利用出来る。インターネット上にあるシステムは、Web Service にアクセスする手段を持っていれば、これらの機能をシステムに組み込む事が可能になる。

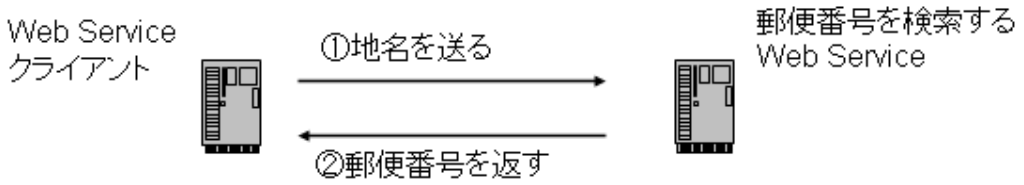

図 2.1: 例　郵便番号 Web Service の利用

Web Service の大きな利点の１つとして Web Service を組み合わせていく事で、より大きなサービスを構築出来る事が挙げられる。この例として、旅行予約システムを示す。

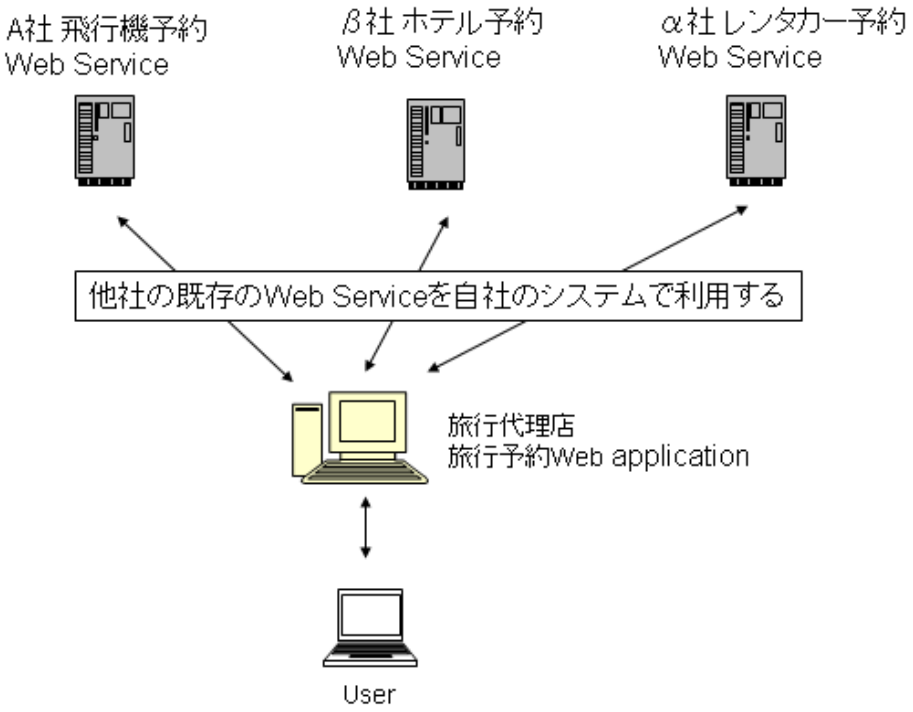

図 2.2: 例　旅行予約システム

図 2.2では、飛行機予約 Web Service、ホテル予約 Web Service、レンタカー予約 Web Service を組み合わせる事によって、旅行予約のサービスという大きなサービスを構築する事が可能になる。更にこの旅行予約システムを発展させたものとして、新たな Web Service を動的に発見し、サービスに組み込む例を示す。

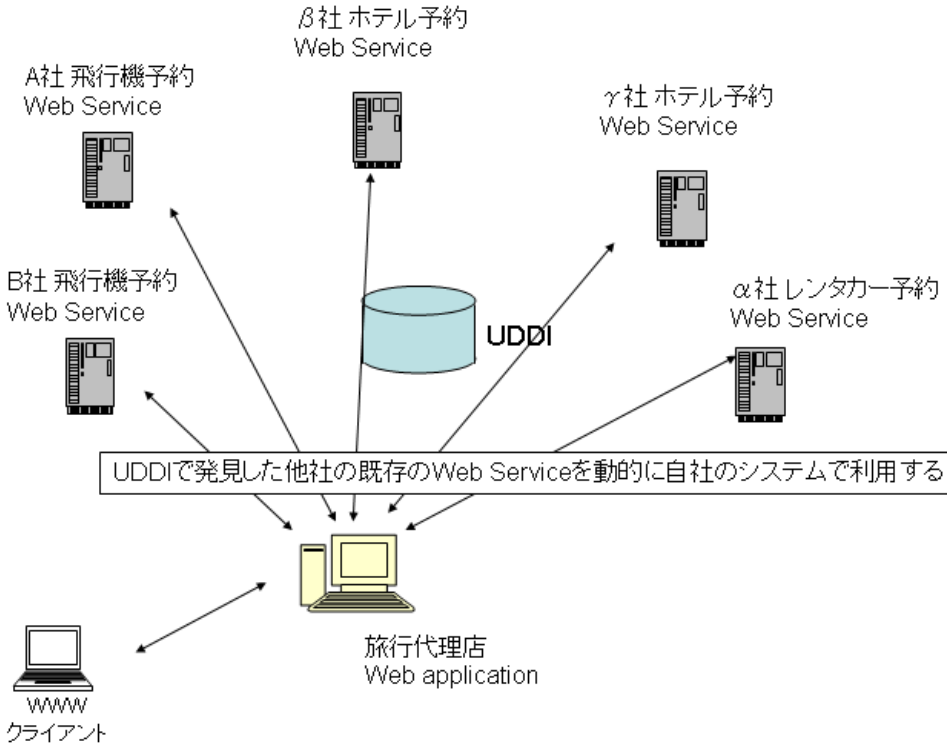

図 2.3: 例 動的発見を用いた旅行予約システム

この様な Web Service の連携を実現する為に、標準技術の策定が進められている。現在、Web Service に関連する標準技術には、Web Service との通信を実現する SOAP(4)、Web Service の検索を実現する UDDI(5)、Web Service のインターフェース記述を実現する WSDL(6) がある。

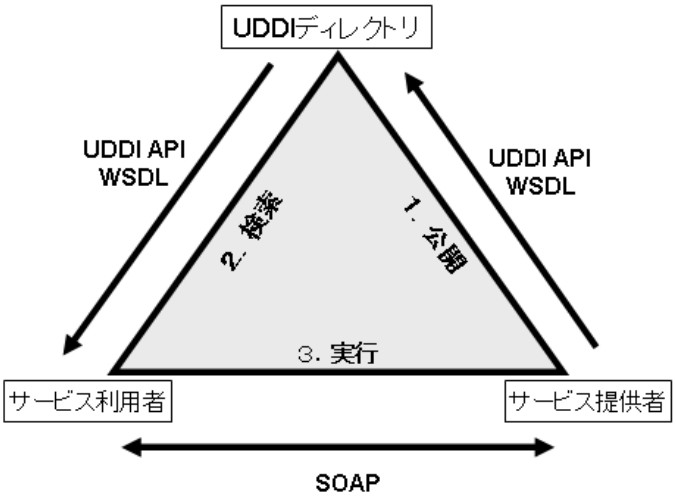

図 2.4: ３つの標準技術

図 2.4は、この３つの標準技術を用いた Web Service の利用を表した図である。
次節以降、それぞれの標準技術ついて述べる。

2.1.2 SOAP

SOAP は、XML によるメッセージ交換を実現する為のプロトコルである。

Web Service にアクセスし、Web Service の持つ機能を利用する為には、Web Service に必要なデータを渡さなければならない。その為には、Web Service に渡すメッセージの構造やデータのフォーマットや、データを送る為の通信プロトコルを決めなければならない。例えば、株価情報を提供するサービスを利用する場合ならば、クライアントは、銘柄の情報を含んだメッセージを Web Service に送る必要がある。そして、Web Service は銘柄に応じた株価情報を含んだメッセージをクライアントに返さなくてはならない。

SOAP では Web Service がやり取りするメッセージの XML ベースのフォーマットの定義や、XML によって int 型などのデータ型を表現する場合の規則について規定している。XML によってメッセージやデータ型を表現出来るという事は、特定のプログラミング言語や環境に依存しないメッセージやデータの表現が実現出来るという事である。よって、SOAP の仕様に基づいてメッセージ交換を行う事で、Web Service はプログラミング言語や環境に依存しない通信を実現する事が出来る様になる。

更に、SOAP では RPC(Remote Procedure Call) を実現する為のデータエンコーディング仕様を定義している。この仕様に従ったメッセージを Web Service に送れば、Web Service の持つ手続き、すなわちメソッドを呼び出す事が出来る。その具体的な方法を図 2.5に示す。

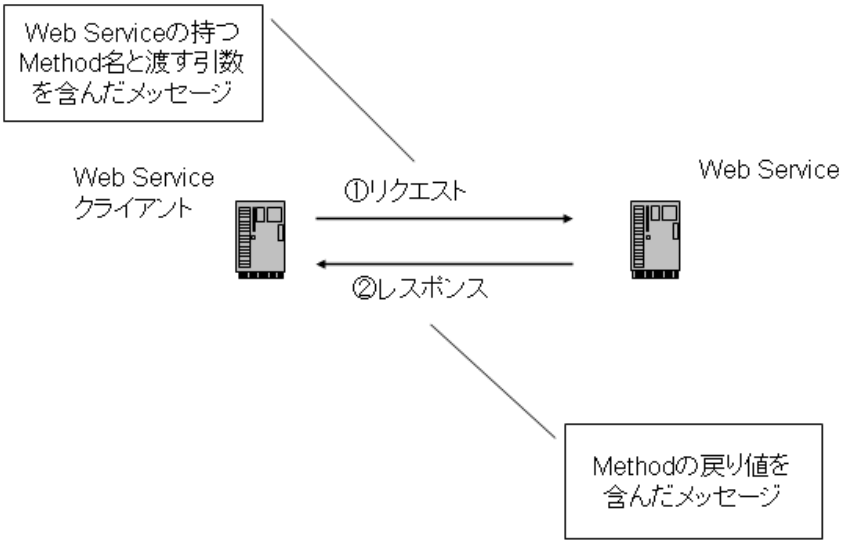

図 2.5: SOAP による RPC の実現

また、SOAP では、下位プロトコルとして任意のプロトコルを利用する事が出来る。その為、HTTP を利用して XML で記述されたデータを送信する事が出来る。これにより、ファイアウォール越しに Web Service 同士を連携させる事も出来る。

2.1.3 UDDI

UDDI （Universal Description, Discovery & Integration）は、Web Service に関する情報を登録したり、あるいはそれらを参照する仕組みに関する仕様、すなわち Web Service における検索システムの仕様である。

Yahoo(7) や google(8) などの検索ページによって、我々は様々な Web ページを発見し、それらの内容を閲覧する事が出来る。これと同様に Web Service を検索する仕組みがあれば、アプリケーションや Web Service 自体に様々な Web Service を発見させ、利用させる事が可能になる。その具体例を図 2.6に示す。

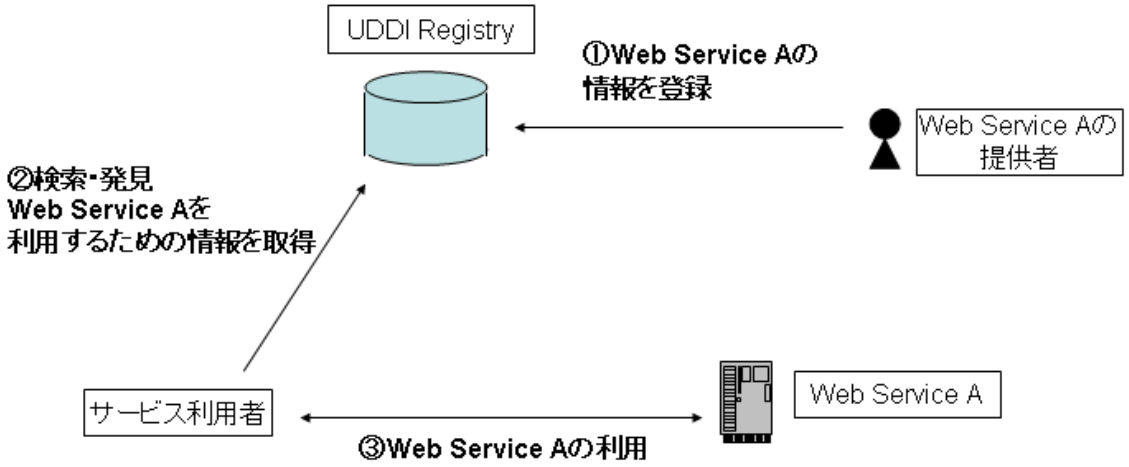

図 2.6: UDDI による Web Service の発見

また、この様なな仕組みがあれば、Web Service を利用するシステムは、どの機能をどの Web Service に実行させるかをあらかじめ決めずに済む。ある機能を実行する時に、その機能を実現する Web Service を動的に発見し利用すれば良い。

UDDI に登録出来る情報には、以下のものがある。

- Web Service を公開している企業・団体・個人に関する情報
- Web Service に関する情報
- Web Service の技術情報
- Web Service の仕様またはカテゴリに関する情報
- ２つの企業・団体・個人の関連に対する情報

こうした情報は、XML で表現された構造化されたデータで表されている。また、これらの情報を検索する方法として、以下の３つの方法が存在する。

- White pages：企業や製品等の名前で検索
- Yellow pages： Web Service の属するカテゴリなどから検索
- Green pages： Web Service の詳細な技術情報から検索

これらのデータを登録および検索する為の機構は、Web Service の形で提供されている。よって、これを利用する為には、先に説明した SOAP を利用し、XML ベースのメッセージを送信する。

2.1.4 WSDL

Web ページの場合、URL を発見する事が出来れば、実際にそのページを閲覧する事が可能となる。しかし、UDDI を利用して発見した Web Service が未知のものである場合、Web Service に渡すメッセージの構造や、メッセージを送る為の通信プロトコルといった情報がなければ、Web Service に実際にアクセスする事は出来ない。また、そうした情報が人間のみならずプログラムが読んで理解出来る様な形になっていなければ、プログラムが動的に Web Service を発見し利用する事は実現出来ない。

そこで、これらの情報を XML によって記述する方法が求められる様になった。WSDL（Web Service Description Language）は、そうした記述方法を提供する XML ベースの言語である。WSDL では、Web Service のインターフェースの定義や、Web Service のエンドポイントの URL や Web Service にアクセスする際に用いるプロトコルの定義を行う事が出来る。これらの定義によって、「A というメッセージを B というプロトコルで、C という URL に送れば、Web Service を呼び出す事が出来る」という事を定義する事が可能になる。

具体的にこうした定義を実現するには、以下の要素を利用する。

- タイプ: XML Schema などを利用したデータ型定義の集合
- メッセージ: Web Service がやり取りするメッセージの定義
- 操作: Web Service が持つ機能およびその入出力の手順に関する定義
- ポートタイプ: 操作の集合
- バインディング: 特定のポートタイプの具体的なプロトコルおよびデータ形式の仕様
- ポート: バインディングとアドレスの組み合わせによるエンドポイントの定義
- サービス: 具体的なアクセスポイントの定義

具体的には、Web Service が受け取る基本的なデータ型はメッセージで表す。Web Service がユーザ定義の型を利用する場合はタイプでそれを定義する。また、Web Service の機能は操作という概念で表し、操作の集合をポートタイプで定義する。更に、バインディングによって、各ポートタイプの持つプロトコルを定義する。これによりポートタイプの持つ個々の操作を利用する為のプロトコルを外部に示す事が出来る。また、バインディングと URL などの具体的なネットワークアドレスとを対応付けた概念がポートである。そしてサービスは、具体的なポートを持ったサービスのアクセスポイントとして定義される。

また、操作は入出力の流れに応じて４種類に分けられる。

- 一方向: メッセージを受信するのみの操作

- 要求/ 応答: メッセージを受信して、対応するメッセージを送信する操作
- 送信請求/ 応答: メッセージを送信して、対応するメッセージを受信する操作
- 通知: メッセージを送信するのみの操作

WSDL では、これまで説明した概念を用いた Web Service の定義方法を提供している。しかし、「この操作は入力された銘柄の株価を返す操作である」という様な、操作の持つ意味について明示的に定義する事は出来ない。また、複数の操作の適切な呼出し手順について記述する事も出来ない。

2.2 本研究のアプローチ

2.2.1 Web Service の繋ぎ目

複数の Web サービスを繋ぎ、利用する為には、単に Web Service の検索に留まるのではなく、多数の Web Service の存在から、その連携を検索する仕組みが重要となる。そこで、本研究では Web Service へ渡す情報と返ってくる情報に注目した。Web Service はインターネット上に散らばる部品と表現される事が多いが、渡す情報と返ってくる情報は、部品で言えば他の部品との繋ぎ目といえる。部品と部品とが接続される場合は、この繋ぎ目の形状が合致している必要がある。つまり、Web Service もその繋ぎ目である渡す情報と返ってくる情報によって組み合わせ可能かを判断出来る。図 2.7に、このモデルを示す。

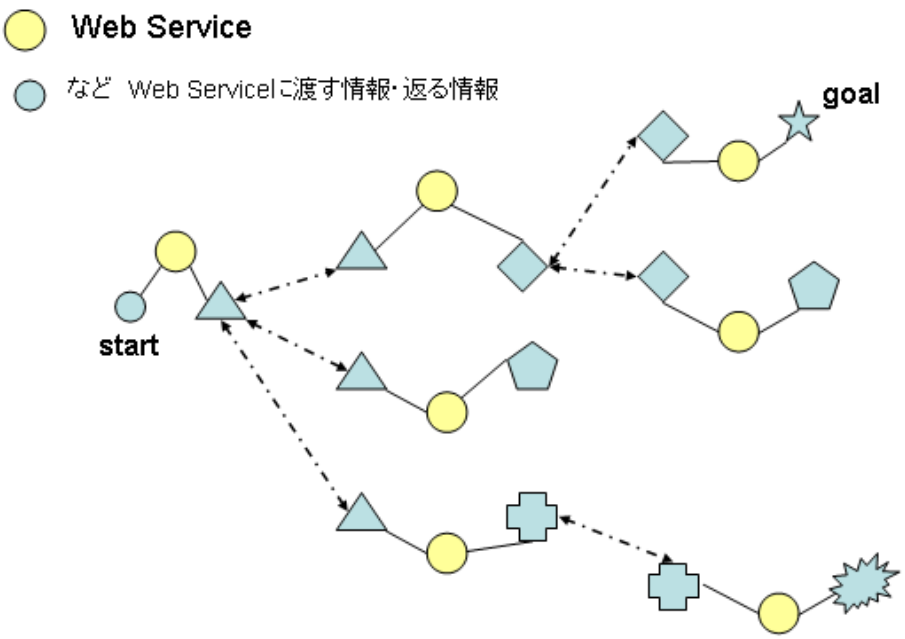

図 2.7: Web Service の繋ぎ目モデル

図では、Web Service を挟んで、左が Web Service へ渡す情報、右が Web Service から返ってくる情報とし、図上の start が今持っている情報、goal が欲しい情報であったとする。図の様に、繋ぎ目の形が一致する Web Service をつなげていく事が出来れば、start から goal へ、あるいは goal から start へと辿っていく道をを探していく事が可能である。本研究では、Web Service へ渡す情報を Web Service への入力、Web Service から返ってくる情報を Web Service の出力と呼ぶ。

具体例を挙げる。User が、”252-0807”という郵便番号の情報を持っていて、これを利用する事が可能な Web Service を検索する。検索結果として、郵便番号の住所を返す Web Service と郵便番号の地域の天気を返す Web Service が渡され、User は住所を返す Web Service を選択し利用し、結果である”神奈川県藤沢市下土棚”という住所を取得する。User は、更にこの住所を利用する事が可能な Web Service を検索する。検索結果として、住所の地域の市外局番を返す Web Service が渡され、User はこれを利用し、結果として”0466”という市外局番を取得する。世の中に、郵便番号を与えると市外局番を返す Web Service が提供されていなかったとすれば、２つの異なる Web Service を組み合わせる事で、新たに郵便番号を与えると市外局番を返すという Web Service を作り上げる事が出来た事になる。

2.2.2 既存技術によるシステムが抱える問題点

先に述べた連携のモデルに現在の既存技術を適用すると、図 2.8の様な問題点が出てくる。

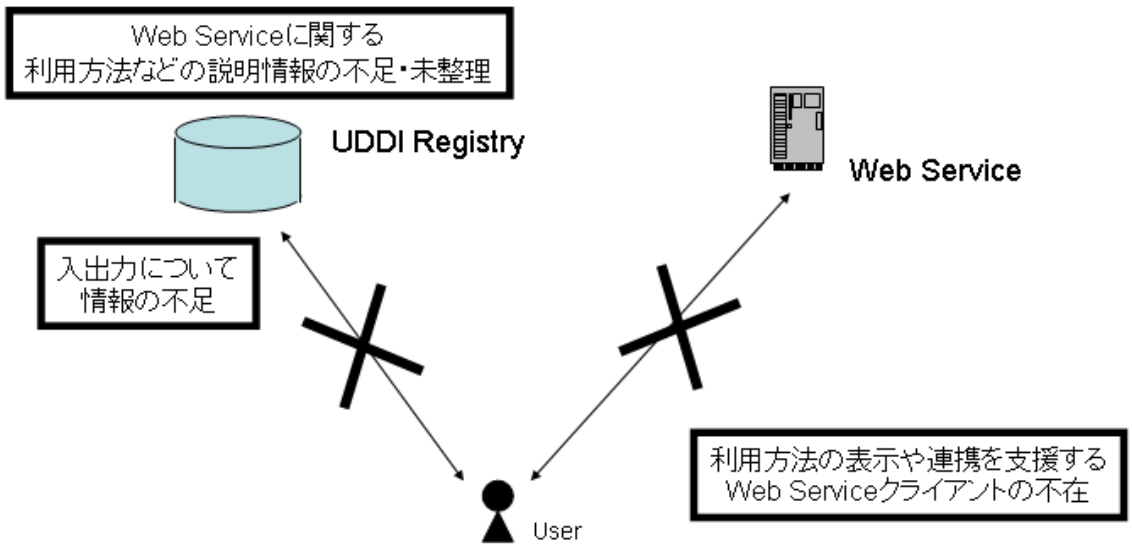

図 2.8: 標準技術の問題点

- 入出力についての情報の不足

- Web Service についての説明情報の不足・未整理
- 連携を支援する Web Service クライアントの不在

以下、それぞれの問題点と本研究の解決手段について述べる。

2.2.3 入出力についての情報の不足

現在の標準技術によって Web Service が UDDI に持つ情報のうち、入出力についての情報として、WSDL 文書内の入出力の型がある。シンタックスである型を元にし、出力値の型を入力値の型として持つ Web Service を探し出す事は図 2.9の様に可能である。

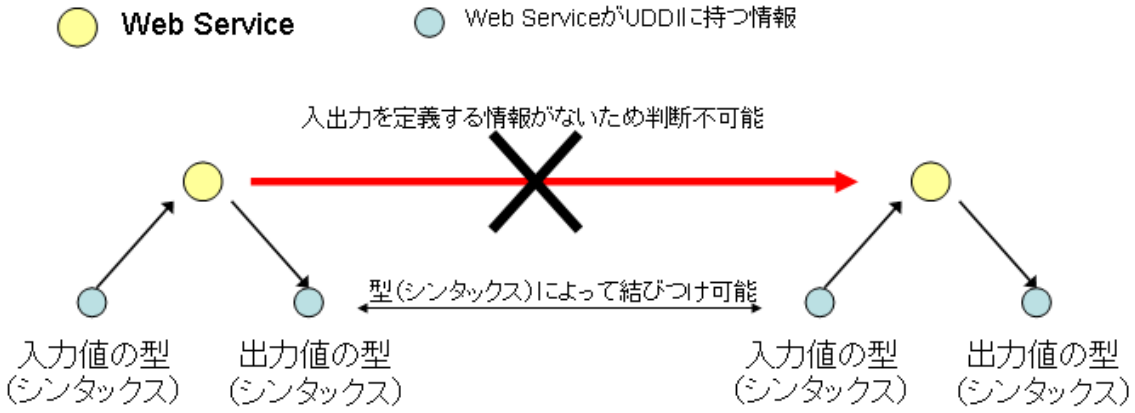

図 2.9: 標準技術のみによる Web Service の結びつけ

しかし、型による結びつけは、膨大な量の Web Service を結びつけの候補として挙げてしまう。例えば、「出力値の型が string 型である」という情報によって、入力値の型が string 型である Web Service を探し出したとすれば、膨大な量の Web Service がその条件に当てはまってしまう。

更に、型という情報のみで Web Service を結びつけた場合、出力値の意味と入力値の意味が同じであるとは限らない。先の string 型を例にとる。string 型の出力値が「東京都文京区小石川」であり、「住所」という意味をもつものであったとする。これを string 型という型の情報のみで連携する Web Service を探し出せば、入力値がもつ意味が「人名」「商品名」「電話番号」といった全く違うものも候補としてあがってしまう。

従って、現在の標準技術では連携可能な Web Service を絞り込む事は不可能である。また、現在の UDDI には入出力についての情報を元にした検索機能は備わっていない。

そこで、本研究では、入出力値のメタデータを追加情報として付加する事で、この問題の解決を図る。追加情報を付加する事により、図 2.10の様に、メタデータをもとに Web Service を結びつける事が可能になる。この結びつけによる Web Service の検索を行えば、連携可能な Web Service を絞り込む事が可能になる。

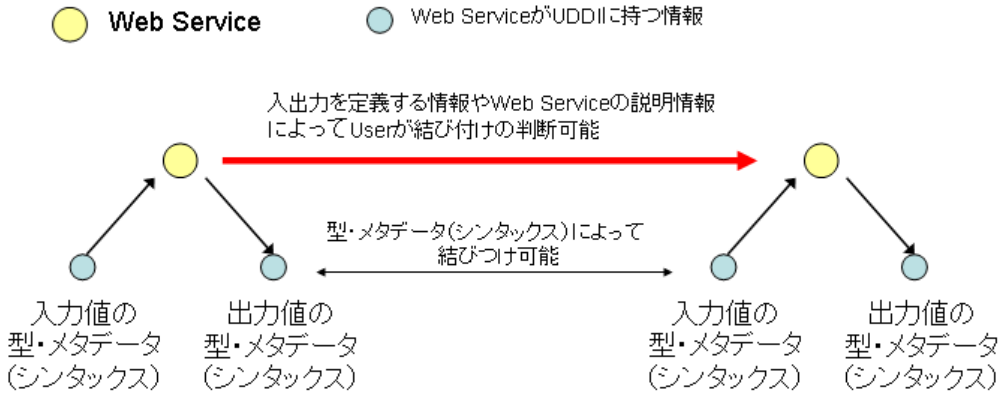

図 2.10: 追加情報を用いた Web Service の結びつけ

また、出力値と連携する Web Service の入力値が同じメタデータを持っているので、連携する Web Service が元の Web Service の出力値にとって有用である可能性も高くなる。

連携する可能性が示唆された Web Service がどの様な Service であるか、それが出力値にとって有用であるかを理解し、どの Service を利用するかを判断するのは User である。これは、Web 上の検索エンジンの利用と共通している。User は検索エンジンを利用する際、複数の keyword を入力する事で、自分が得たいと思う情報が掲載されている Web ページを絞り込んでいく。そして、検索結果からどの Web ページに自分が得たい情報が載っているかを判断するのは User である。

2.2.4 Web Service についての説明情報の不足・未整理

User が Web Service を利用したり、Web Service を利用したシステムを開発したりする際、Web Service がどの様なサービスを提供するか、どの様な処理を行うか、どの様に利用すればいいかといった情報が必要になる。

これらの情報は、現状では UDDI に図 2.11の様に分けられる事なく登録されている。また、説明の仕方も様々であり、メソッドの詳細については、Service Provider が持つ Web ページに書かれている場合や、そもそも記述されていない場合がある。この形式では、説明情報をあらかじめ自分で収集しておく必要がある。

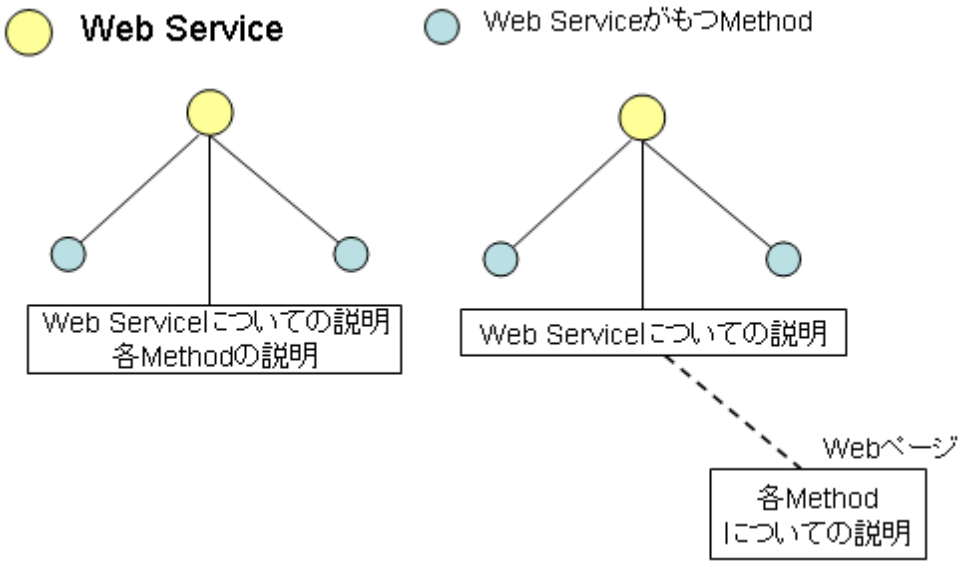

図 2.11: 現在の UDDI での登録形態

従って、この様な状態では、メソッドごとにその処理の内容や利用方法を取得する事は不可能である。

そこで、本研究では、Method ごとの説明を追加情報として付加し、図 2.12の様な形式で情報を持たせる事で、この問題の解決を図る。

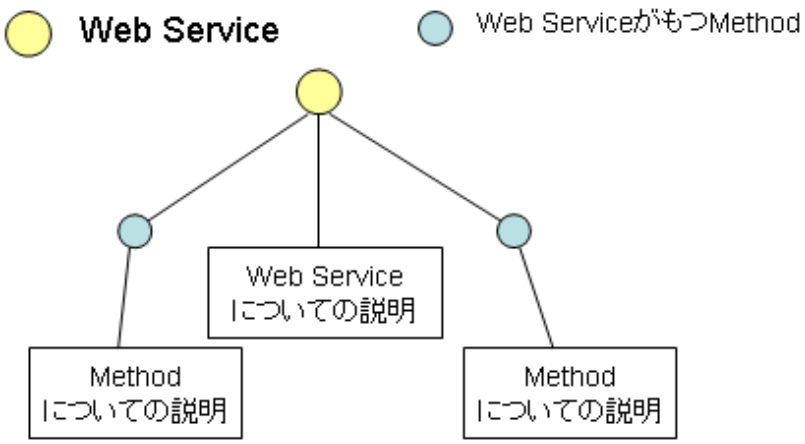

図 2.12: 説明情報の整理

これにより、User はあらかじめ Method の利用方法を集めておく必要がなくなると同時に、Method の利用や連携 Web Service の選択の際に必要な情報を適宜手に入れる事が可能になる。

2.2.5 連携を支援する Web Service クライアントの不在

先の述べた２つの問題により、現在の標準技術では、Web Service の動的な利用を可能にする為の情報や、連携を判断する為の情報が不足している。この為、単純に WSDL 文書を取得し解析する事のみによって Web Service クライアントを作成した場合、User に利用する為の情報として表示可能なものは、Method 名・引数名・引数の型のみである。

入出力についての情報が不足する為、連携可能な Web Service を示唆する事は不可能である事は先に述べた。現状の標準技術だけで Web Service を UDDI の利用により発見し、有用な連携を実現する流れは、図 2.13の様になる。

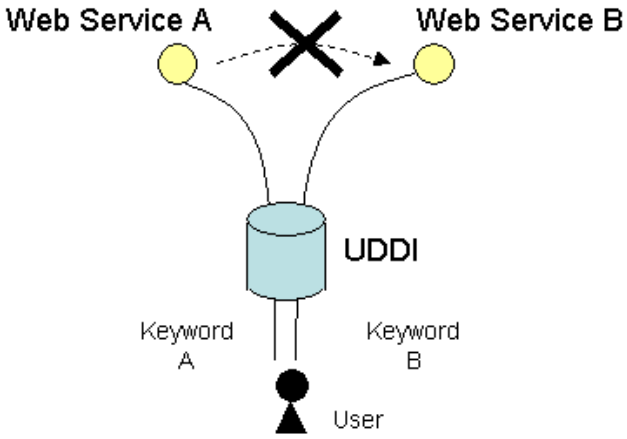

図 2.13: 現在の標準技術だけによる Web Service の連携

まず、User はあらかじめ連携させたい Web Service の処理内容を決めておき、それぞれ別々の keyword によって必要とする用途にあった Web Service を発見する。そして、それぞれの Web Service の詳細な情報を手に入れる事によって、連携可能かどうかを判断する。Web Service を UDDI の利用により発見し連携させる為にはこの方法しかなく、利用する Web Service の情報を元に、連携可能な Web Service を探す事は不可能である。

従って、現在の標準技術のみによる Web Service クライアントでは、Web Service の動的な利用や Web Service の連携を支援する事は不可能である。

そこで、本研究では、先に述べた追加情報を利用する事によって、Web Service の動的な利用や Web Service の連携の支援を可能にする Web Service クライアントを作成する。本研究が作成する Web Service クライアントを利用した Web Service 利用の流れは図 2.14の様になる。

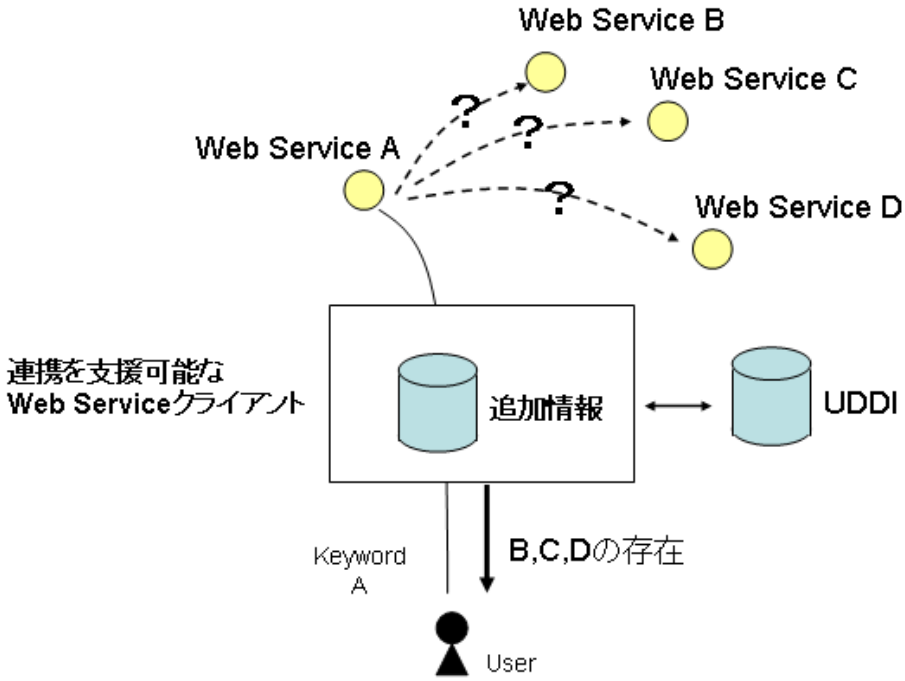

図 2.14: 本研究の Web Service クライアントを用いた Web Service の連携

User は、最初に利用する Web Service を入出力についての keyword によって発見し、利用する。そこで追加情報データベースを利用する事で、１つの keyword によって次々と連携の可能性がある Web Service の存在を知る事が可能になる。

第3章　設計

前章では、既存システムの問題点への本研究のアプローチについて述べた。本章では、それに基づいて構築するシステムの設計について述べる。本システムでは、前章で述べたつなぎ目モデルのうち、start から goal へと辿るケースについてのみ設計を行った。本研究では、start から goal へと辿るケースをフォワードトレース、goal から start へと辿るケースをバックトレースと定義する。

なお、本システム名を WS-Connect と称する。従って、以下本システムを WS-Connect と記す場合がある。

3.1　システムの構成

まず、本システムを含んだ全体の構成について述べる。Web Service の検索と利用を行う際に必要な構成である Web Service Provider・User・UDDI に、WS-Connect を加えた全体の構成を図 3.1に示す。

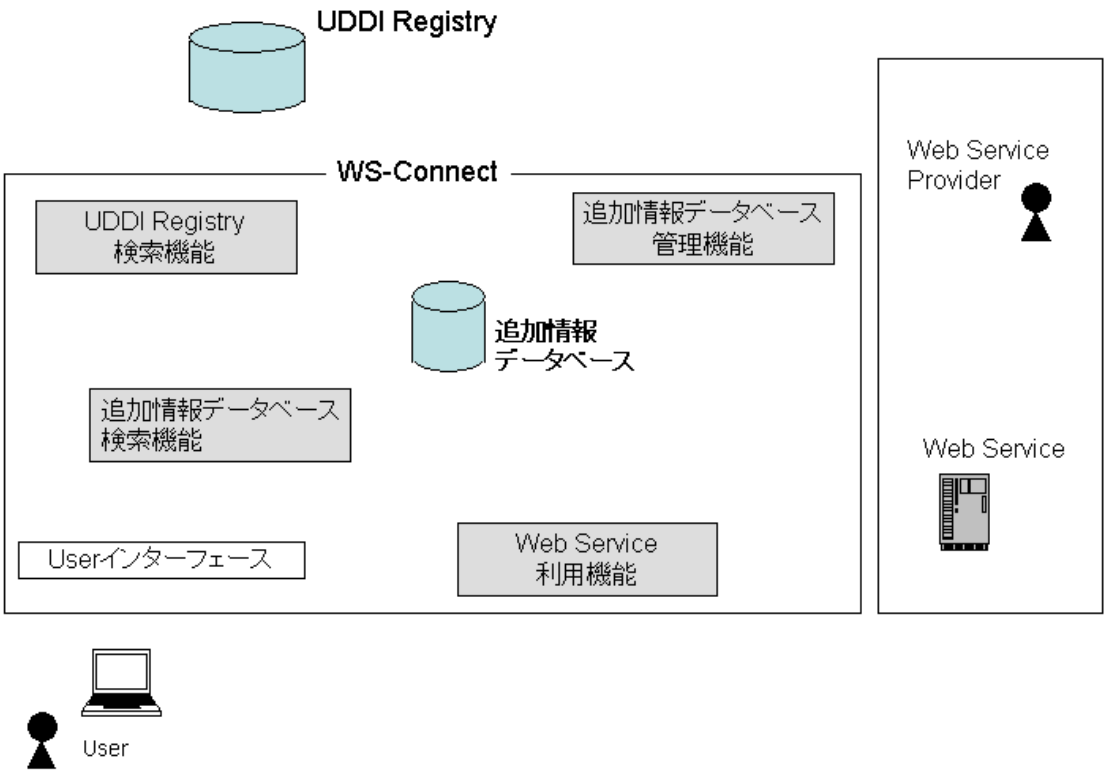

図 3.1: 全体の構成図

WS-Connect は、

- User インターフェース
- 追加情報データベース
- 追加情報データベース管理機能
- 追加情報データベース検索機能
- Web Service 利用機能
- UDDI Registry 検索機能

によって構成される。この WS-Connector が追加情報 DB の情報を活用し、対 Web Service Provider、対 User、対 UDDI それぞれに適宜やりとりをする事で Web Service の連携を支援する環境を実現する。

また、Web Service Provider は、UDDI Registry 及び追加情報データベースにあらかじめ提供する Web Service についての情報を登録しておく必要がある。

3.2 本システムの前提条件

本システムを利用する為に Web Service Provider と User に必要となる事について述べる。

- **Web Service Provider** Web Service Provider は、まず提供する Web Service を UDDI Registry に登録する必要がある。そして登録を行うと発行される Business Key 及び Service Key を元に、本システムを利用し、本システム内の追加情報データベースに提供する Web Service の追加情報を登録する。この２つを行う事により、User によって連携の可能性がある Web Service として探し出される準備が整う。

- **User** User は、UDDI Registry によって検索を行うのではなく、本システムを介して行う必要がある。また、本システムを利用して得た検索結果から、各 Web Service の説明を読んだ上で、最終的にどれを利用するか判断し、実際に Web Service を利用する際には、利用する Web Service が必要としている値を入力しなければならない。

3.3 各機能とその役割

3.3.1 追加情報データベース

追加情報データベースは、User が異なる Web Service をつなげていく上で必要になるが、UDDI に登録されていない情報を登録する。この情報は、Web Service Provider によって登録される。

追加情報データベースに登録される情報は、以下の通りである。

- Web Service Provider が提供者としての情報を UDDI Registry に登録した際発行される Business Key（Business_Key）と追加情報データベースに登録する際 Provider を認証する為のパスワード（Password）
- Web Service を UDDI Registry に登録した際発行される Service Key(Service_Key)
- 提供する Web Service の名前 (Service_Name)
- 提供する Web Service についての説明 (Service_Man)
- 提供する Web Service が持つ全てのメソッド名 (Method_Name)
- 提供する Web Service が持つ各メソッドの説明 (Method_Man)
- 各メソッドの入力、出力についてのメタデータ (Var_Meta,In_Out)

図に追加情報データベースのテーブル関係図 3.2を示す。

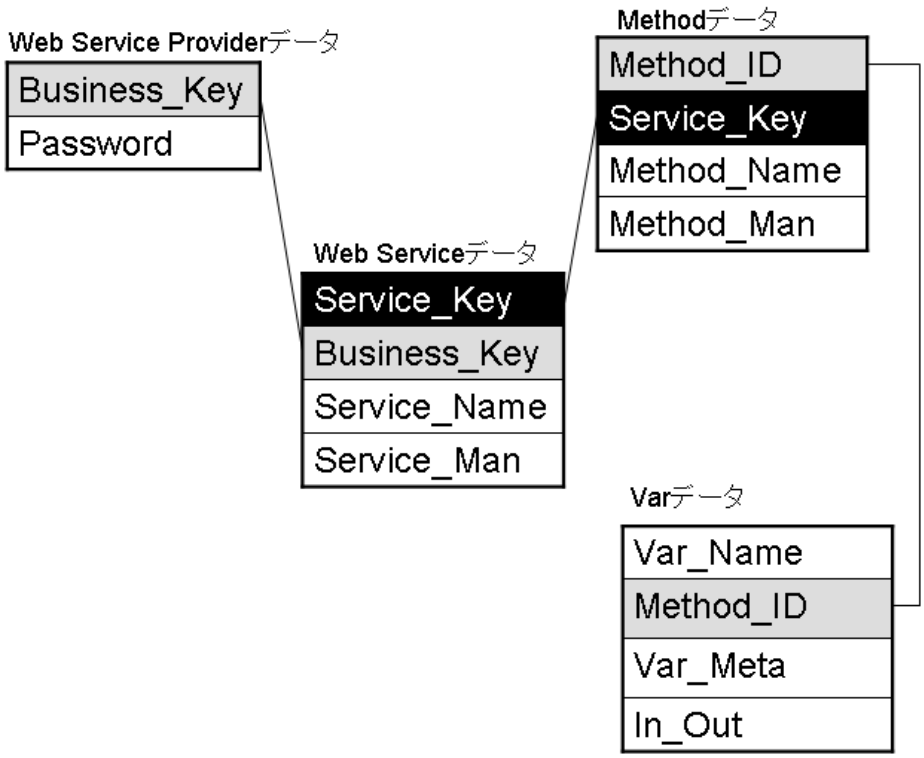

図 3.2: 追加情報データベースのテーブル関係図

この中で、Web Service 名は UDDI にも登録されている情報であるが、本システムを用いて User が利用したい Web Service を選ぶ最初の検索の際、UDDI Registry に問い合わせをするオーバーヘッドをなくす為に、追加情報データベースにも Web Service 名を登録する。

3.3.2 追加情報データベース管理機能

追加情報データベース管理機能は、以下の 2 つの機能を備える。

- Web Service Provider の認証（図 3.3）
- 追加情報データベースの追加情報の登録・削除・変更（図 3.4）

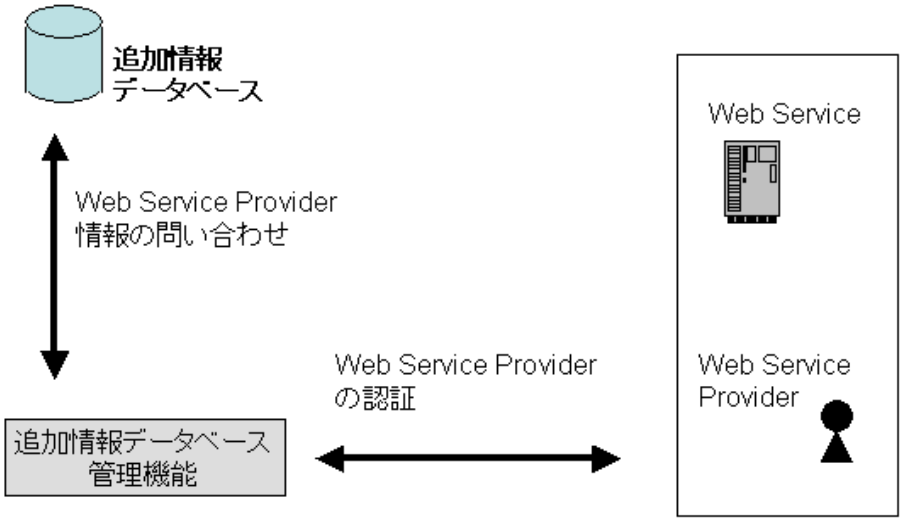

図 3.3: 認証機能

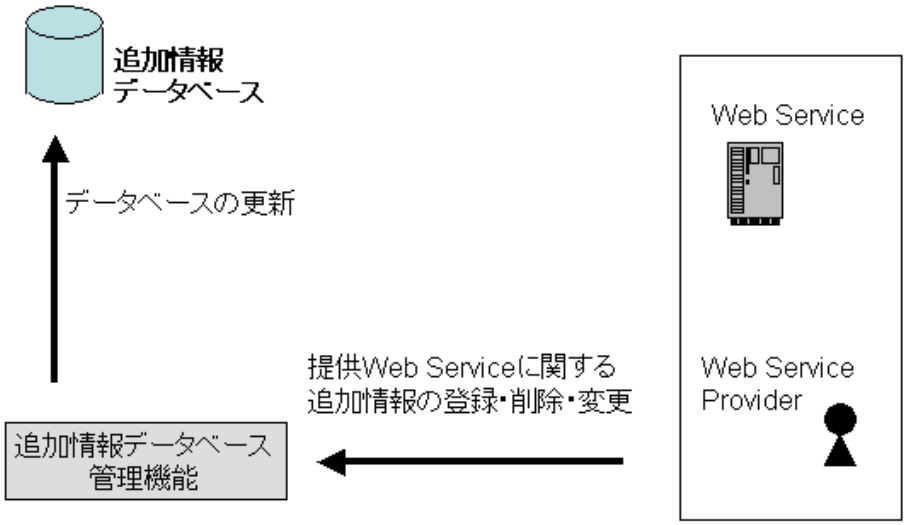

図 3.4: データベースの登録・削除・変更

本機能によって、Web Service Provider は提供する Web Service の追加情報を管理する事が可能になる。

3.3.3 追加情報データベース検索機能

追加情報データベース検索機能は、検索情報データベースに問い合わせを行い、結果を取得する機能を持つ。本機能は、以下の場面に利用される。

- User が最初に本システムに対して与える keyword を入力値のメタデータとしてもつ Web Service の検索と検索結果の User への通知（図 3.5）
- User が実際に利用する Web Service を Web Service 利用機能に通知した際、Web Service 利用機能から検索要求を受けて行う、利用する Web Service の追加情報の検索と検索結果の Web Service 利用機能への通知（図 3.6）
- Web Service 利用後、Web Service 利用機能が利用結果を解析した情報を本機能に渡した際に行う、利用した Web Service の出力値についてのメタデータの検索とその結果を入力値についてのメタデータとしてもつ Web Service の検索及び Web Service 利用結果の User への通知（図 3.7）

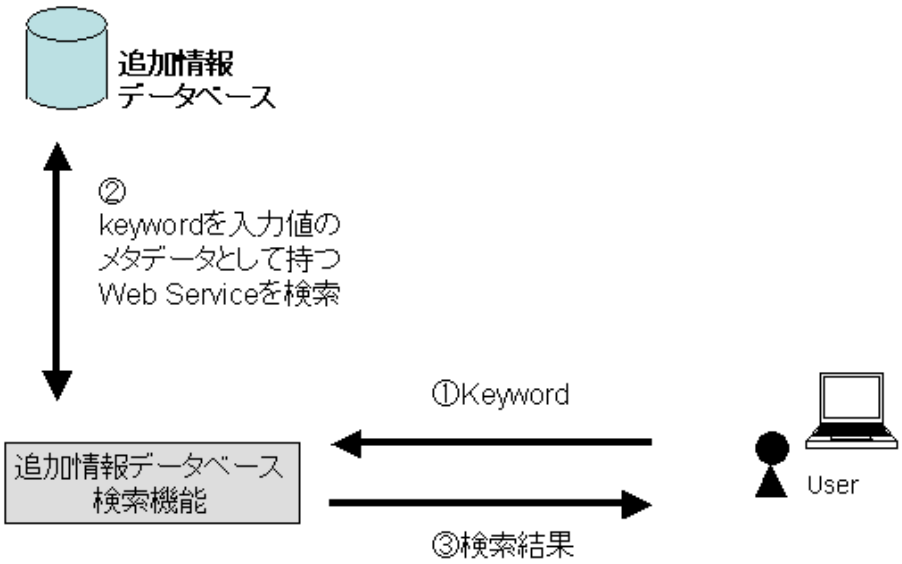

図 3.5: 追加情報データベース検索機能 1

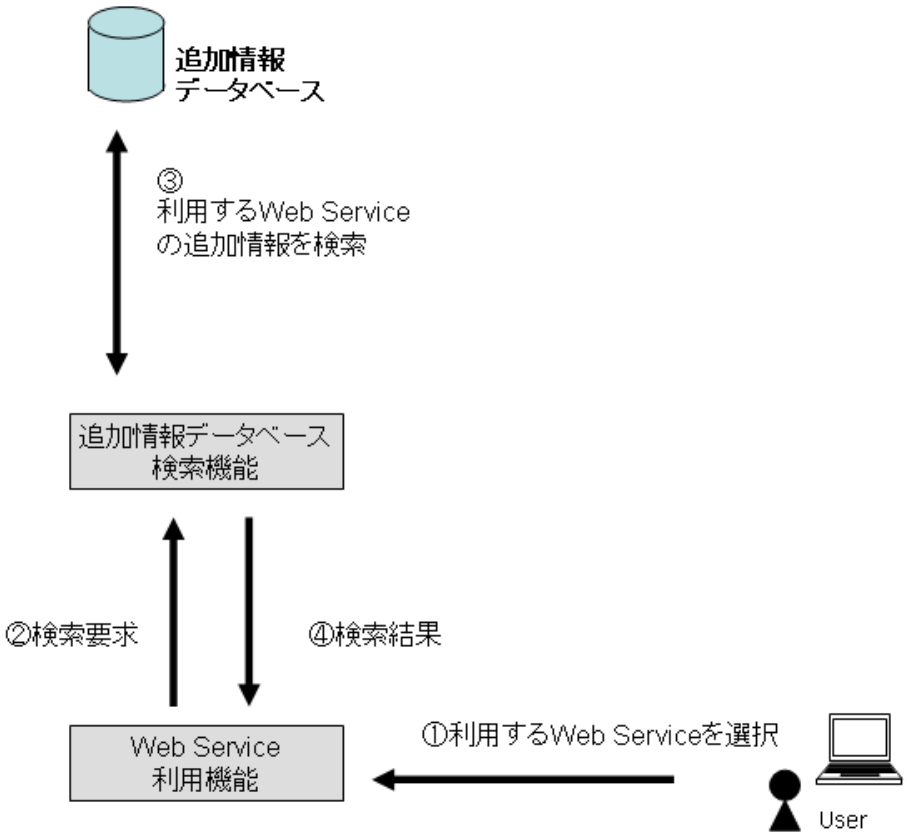

図 3.6: 追加情報データベース検索機能 2

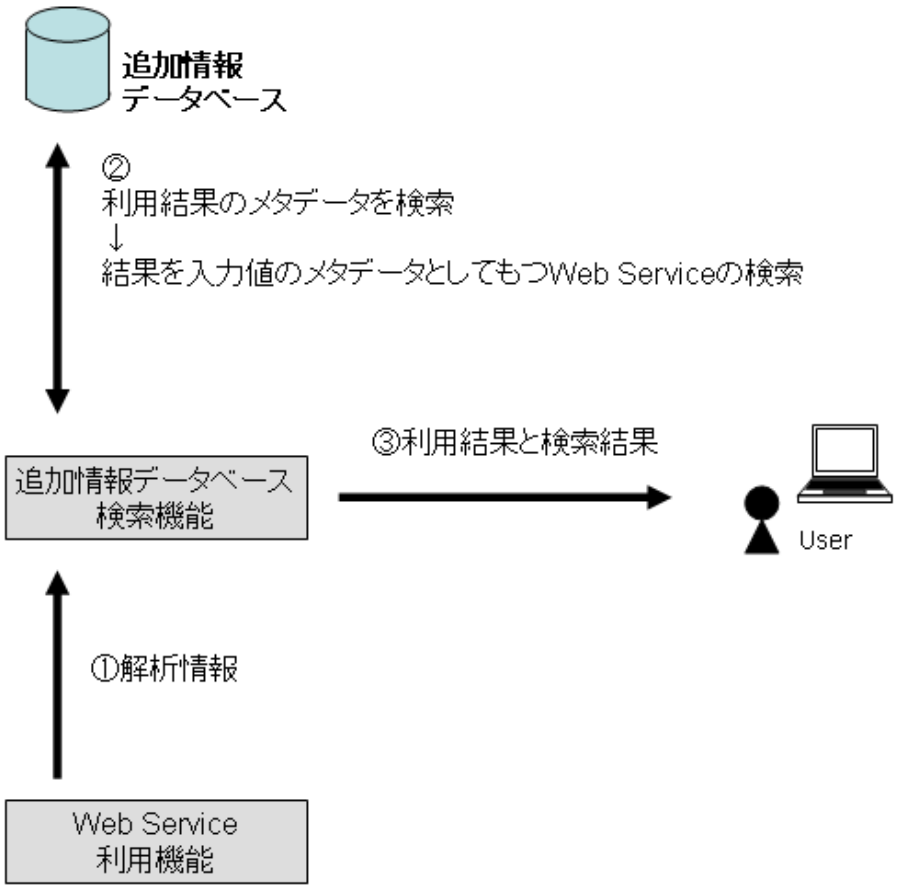

図 3.7: 追加情報データベース検索機能 3

3.3.4 Web Service 利用機能

Web Service 利用機能の動作の流れを図 3.8に示す。

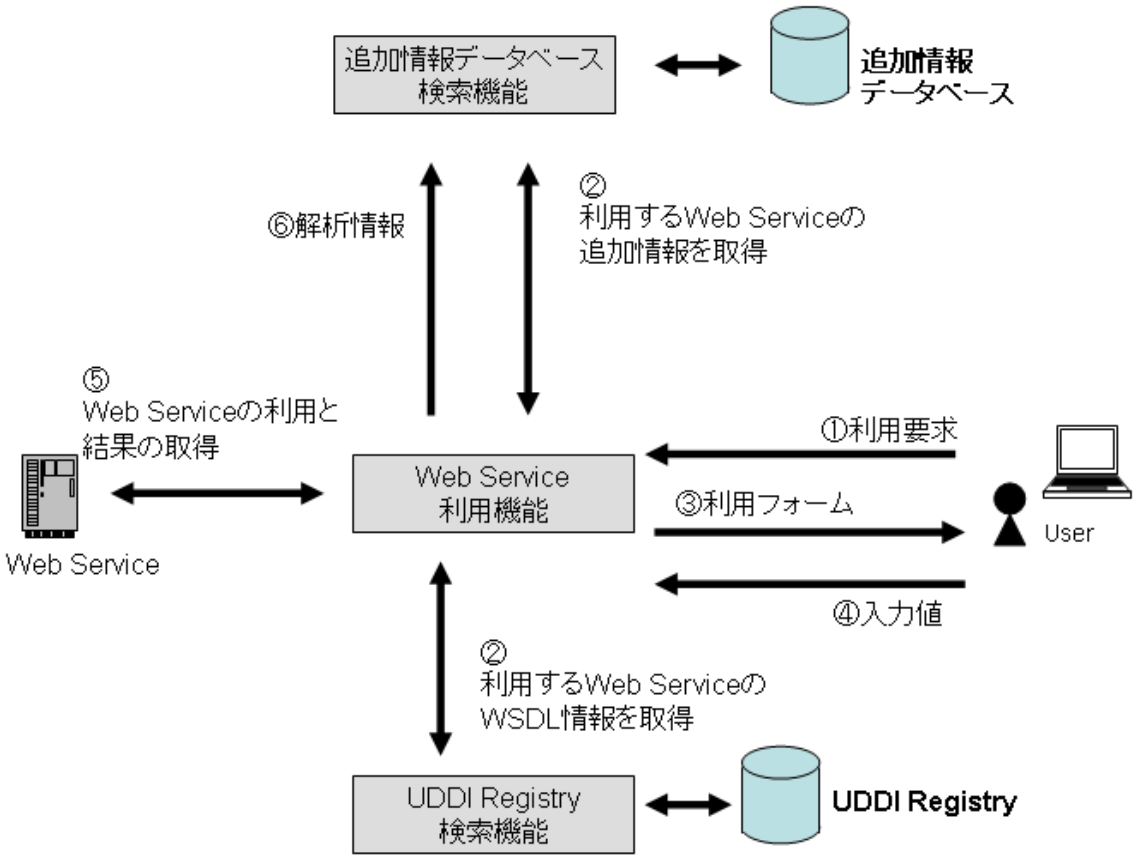

図 3.8: Web Service 利用機能

- User から Web Service の利用要求を受けると、追加情報データベース検索機能へ利用要求のあった Web Service の追加上の検索要求を通知する。また、UDDI Registry 検索機能へ利用要求のあった Web Service の WSDL 情報の検索要求を通知する。
- それぞれの機能から受け取った検索結果を処理し、利用フォームを作成して User へ提供する。
- User からフォームに入力された値を元に、利用要求のあった Web Service を利用し結果を取得する。
- 結果を解析し、解析情報を追加情報データベース検索機能へ渡す。

3.3.5 UDDI Registry 検索機能

UDDI Registry 検索機能の動作の流れを図 3.9に示す。

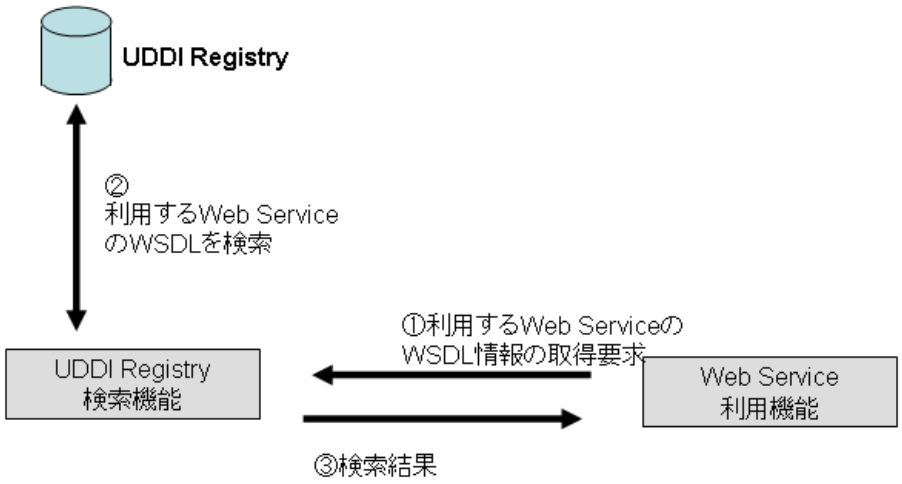

図 3.9: UDDI Registry 検索機能

UDDI Registry 検索機能は、Web Service 利用機能から検索要求が来ると、User から利用要求があった Web Service の UDDI Registry に登録されている WSDL 情報を検索・取得し、結果を Web Service 利用機能へ渡す。検索は、User から利用要求のあった Web Service の追加情報データベースに登録された Service Key を元に、UDDI Registry に問い合わせを行い、該当 Web Service の WSDL 情報がおかれた URL を取得する。

3.4 全体の流れ

WS-Connect を通した全体の流れは、図 3.10の通りである。

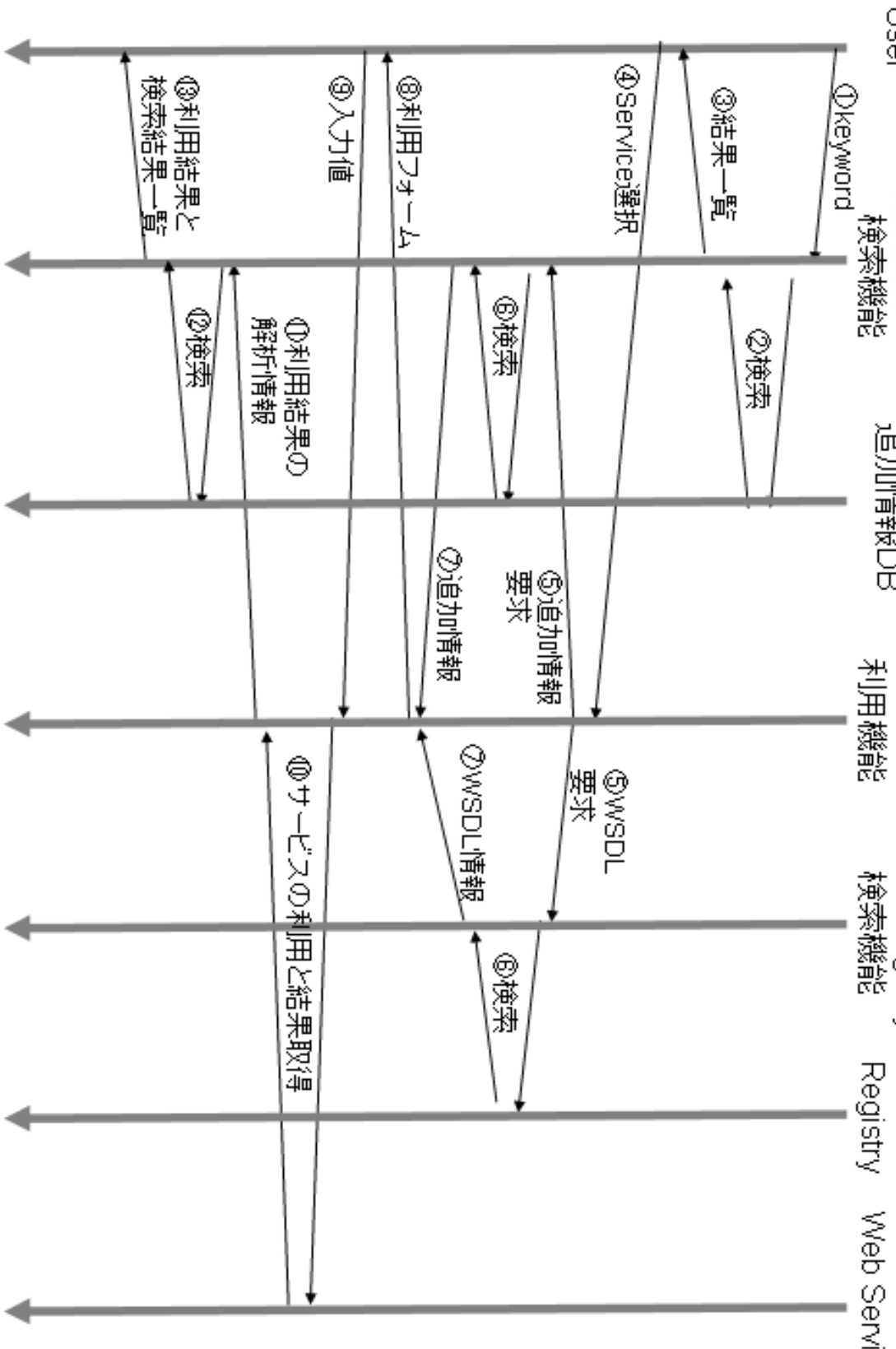

図 3.10: 全体のシーケンス図

1. User は、keyword を追加情報データベース検索機能へ与える。
2. 追加情報データベース検索機能は、与えられた keyword を入力値についてのメタデータとして持つ Web Service を追加情報データベースから検索し、結果を取得する。
3. 追加情報データベース検索機能は、検索結果を処理して該当 Web Service の一覧を User へ渡す。
4. User は、一覧から利用したい Web Service を選択し、Web Service 利用機能へ利用要求を通知する。
5. Web Service 利用機能は、User から利用要求のあった Web Service についての追加情報と WSDL 情報の検索要求を追加情報データベース検索機能と UDDI Registry 検索機能にそれぞれ通知する。
6. 追加情報データベース検索機能は、User から利用要求のあった Web Service についての追加情報を追加情報データベースから検索し、結果を取得する。UDDI Registry 検索機能は、User から利用要求のあった Web Service の WSDL 情報を UDDI Registry から検索し、結果を取得する。
7. 追加情報データベース検索機能、UDDI Registry 検索機能は、それぞれの結果を Web Service 利用機能へ渡す。
8. Web Service 利用機能は、受け取った結果を解析し利用フォームを作成して User へ渡す。
9. User は、利用フォームに値を入力し、Web Service 利用機能へ渡す。
10. Web Service 利用機能は、User から渡された値をもとに Web Service を利用し、結果を取得する。
11. Web Service 利用機能は、利用結果を解析し、解析結果を追加情報データベース利用機能へ渡す。
12. 追加情報データベース利用機能は、利用した Web Service の出力値についてのメタデータの検索とその結果を入力値についてのメタデータとしてもつ Web Service の検索を行い、結果を取得する。
13. 追加情報データベース利用機能は、追加情報データベースの検索結果と Web Service の利用結果を User へ渡す。

→4 に戻り、繰り返す。

第4章　実装

本章では、前章で述べた設計に基づいて構築したWS-Connectの実装について述べる。

本システムは、ユーザーインターフェースとしてWWWを採用した。専用のソフトウェアを用いる事なく、広く普及しているWebブラウザを使う事で、Userは、より簡易にWebサービスを利用する事が出来る。

WS-Connectは以下の環境で構築した。

OS	FreeBSD 4.7-RELEASE
WWW Server	apache-1.3.27_1
開発言語	PHP 4.3.0 + PEAR SOAP-0.7.1
RDBMS	postgresql-7.2.3

表 4.1: 本システム　ソフトウェア環境

OS	Windows 2000 Professional
Webブラウザ	Internet Explorer 6

表 4.2: User側ソフトウェア環境

本システムは、サーバーサイドスクリプトとして実装した。スクリプトは、PHPで記述した。また、PHPライブラリのコード用のコードレポジトリであるPEARのSOAPパッケージを使用した。

4.1　追加情報データベース管理機能

4.1.1　Provider登録

Web Service Providerが追加情報データベースを利用する際、はじめにProvider登録を行う。これは、実際にWeb Serviceに関する追加情報を管理する際、そのWeb ServiceのProviderかどうかを認証する為に必要となる。登録画面を図4.1に示す。

図 4.1: Web Service Provider 登録画面

Web Service Provider は、UDDI Registry にもつ固有の business Key を ID として入力し、パスワードを設定する。

4.1.2 追加情報の追加・変更・削除

Web Service Provider は認証を終えると、図 4.2の様な管理ページを用いて、追加情報データベース上の Web Service の追加情報を管理する事が出来る。

図 4.2: Web Service 管理画面

Web Service の新規登録画面を、図 4.3に示す。

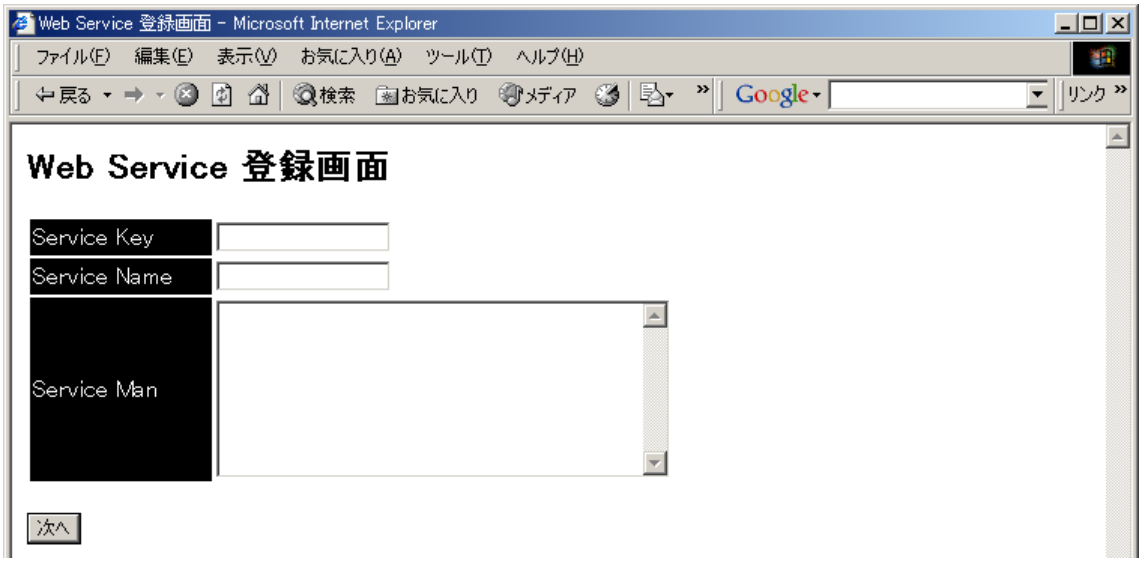

図 4.3: Web Service 登録画面

まず、Web Service Provider は、

- Service Key
- Web Service の名前
- Web Service の説明

を入力し、Method 登録画面に進む。Method 登録画面を図 4.4に示す。

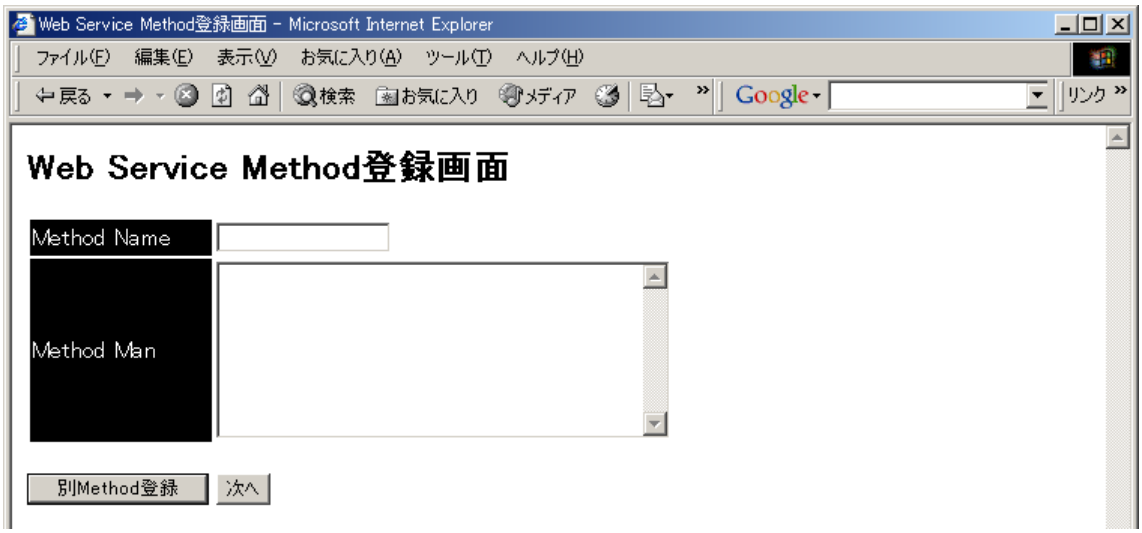

図 4.4: Method 登録画面

Method 登録画面では、

- Service が持つ method 名
- method の説明

をその Web Service が持つ method の分だけ登録した後、入出力情報登録画面（var 登録画面）へと進む。入出力登録画面を図 4.5に示す

図 4.5: 入出力登録画面

入出力登録画面では、

- どの Method の入出力情報を登録するか（Method の名前）
- 変数名とメタデータ（複数の場合は、カンマ切り）
- 登録する情報は、入力の情報か、出力の情報か。

をその Web Service が持つ入出力の分だけ登録する。
以上の操作によって、必要な追加情報が追加情報データベースに登録される。また、登録した追加情報の変更・削除も管理画面によって行う事が出来る。

4.2 追加情報データベース検索機能

本機能は、追加情報データベース内を検索し、必要な情報を取得するが、呼び出されるタイミングによって、動きが異なる。

4.2.1 User が keyword を入力した場合

本システムを Web Service の User が利用する際、最初に呼び出すページは、図 4.6である。

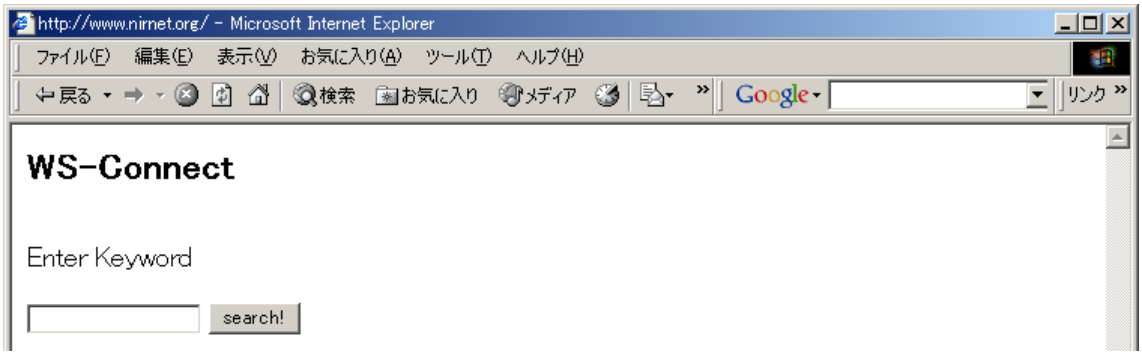

図 4.6: 最初の画面

User は、利用したい Web Service を検索する為に、keyword を入力する。 この時、本機能は、keyword を Web Service の入力値についてのメタデータとして持つ Web Service を追加情報データベースから検索し、図 4.7の様な検索結果ページを User に表示する。

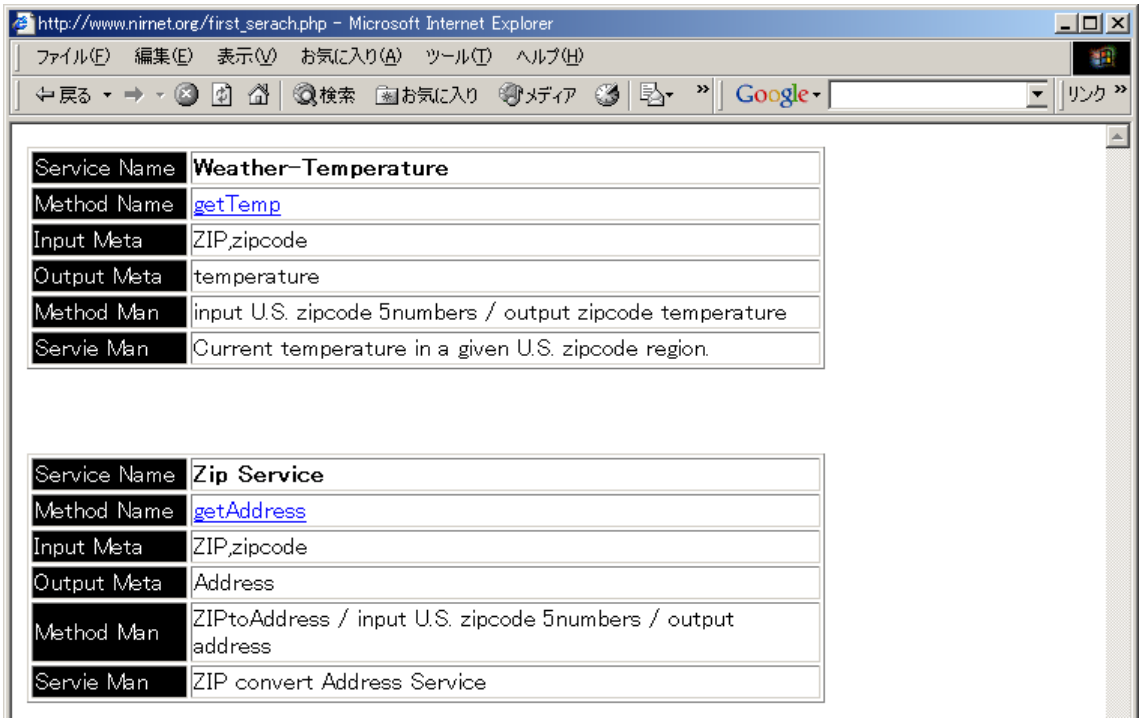

図 4.7: 検索結果画面

User が利用したい Web Service の Method をクリックする事で、利用フォーム画面へと移る。

4.2.2 Web Service 利用機能がフォームを作成する場合

User が検索結果より利用したい Web Service を選択した際、本機能は、Web Service 利用機能がフォームを作成する際に表示する情報を、service Key を元に追加情報データベースから検索し、Web Service 利用機能へと渡す。

4.2.3 Web Service 利用機能が結果を表示する場合

Web Service 利用機能が Web Service を利用した結果を User に表示する際、本機能は、出力値についてのメタデータの検索を行い、Web Service 利用機能へ渡すとともに、その結果を入力値についてのメタデータとしてもつ Web Service を追加情報データベースから検索し、図 4.8の様な検索結果ページを User に表示する。

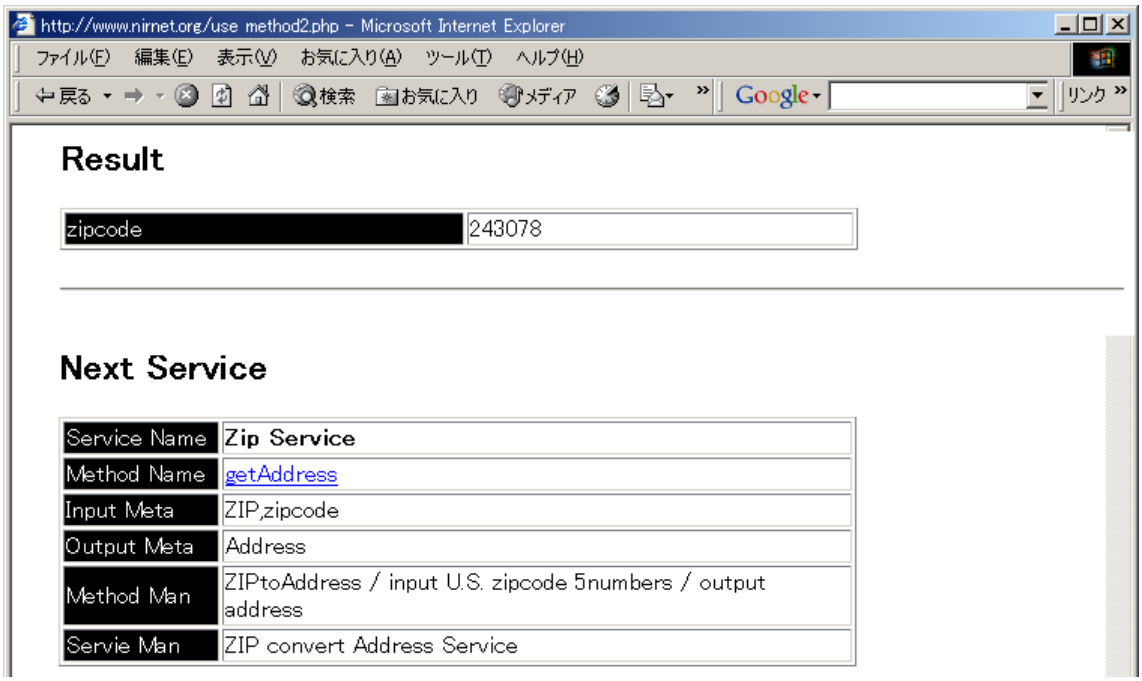

図 4.8: 利用結果画面

4.3 Web Service利用機能

4.3.1 Web Service 利用フォームの作成

本機能は、UDDI 検索機能から受け取った WSDL 情報と追加情報データベース検索機能から受け取った該当 Web Service の詳細情報をもとに、利用フォームを作成し、User に表示する。WSDL 情報の message 要素と portType 要素から、必要とされる入力値の数と名前と型を取得し、それと追加情報を照らし合わせて各入力値についてのメタデータとメソッドの利用法をあわせて表示するフォーム（図 4.9）を作成する。

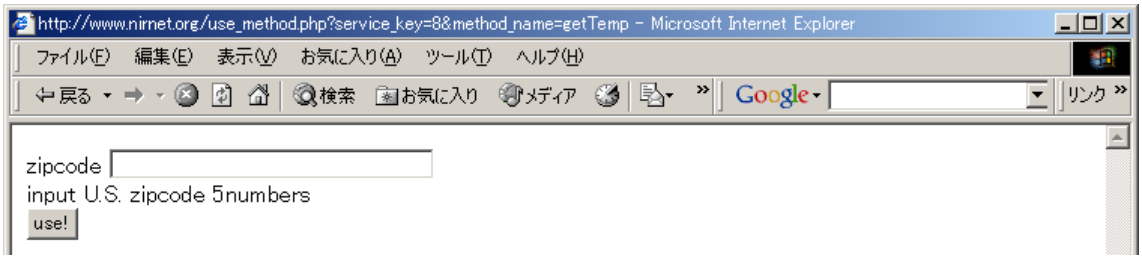

図 4.9: 利用フォーム画面

4.3.2 Web service の利用と結果の出力

本機能は、WSDL 情報からスタブモジュールを作成し、User から受け取った入力値をもとに SOAP メッセージを作成し、該当 Web Service を利用する。受け取った結果を解析し、追加情報データベース検索機能へと渡す。

4.4 UDDI 検索機能

本機能は、UDDI Registry の UDDI API の照会 API である get_serviceDetail を使用し、service Key をもとに、該当 Web Service の WSDL 情報を得る働きをする。

第5章　評価

5.1　評価方法

本章では、前章で実装したWS-Connectについての評価を行う。

本実装が３章冒頭で定義したフォワードトレースを行えるかを、実際に本実装を用いてWeb Serviceの発見・利用・連携を行う事で評価する。

また、現在の標準技術のみによる既存システムによるWeb Serviceの利用と本実装を用いた利用を、いくつかの項目別に比較し、本実装の優位性を評価する。

5.2　実験環境

本実装では、UDDI Registryに登録されているWeb Serviceであれば、追加情報データベースにWeb Serviceについての情報を登録する事で本システムによって利用する事が可能になる。しかし、日経IT Proの記事 (9) において2002年春の時点でUDDI Registryに登録されていて実際に利用する事が可能なWeb Serviceは５０件程度であったと書かれる様に、UDDI Registryに登録されている多くはWSDLを公開するURLやWeb Serviceの公開元のURLが無効になっている場合や、そもそもWSDLインタフェースを持っていない場合が多い。従って、現段階では、UDDI Registryに登録されているWeb Serviceを連携していく事は難しい。

そこで、評価の実験環境を用意する為３つのWeb Serviceを作成し、UDDI Registryに登録した。また、それぞれのWeb Serviceの追加情報を追加情報データベースに登録した。

作成した３つのWeb Serviceの詳細は以下の通りである。

メソッド	Address2ZIP
入力された住所から、その場所の郵便番号を求める	
入力	adr(String)
	メタデータ　address
出力	zip(String)
	メタデータ　zipcode,ZIP
例）	”神奈川県藤沢市下土棚”→ Address2ZIP →”252-0807”

表 5.1: 郵便番号 Web Service

メソッド	getTELNO
入力された住所から、その場所の市外局番を求める	
入力	target_address(String)
	メタデータ　address
出力	telno(String)
	メタデータ　TEL,telephone number
例）	”神奈川県藤沢市下土棚”→ getTELNO →”0466”

表 5.2: 市外局番 Web Service

メソッド	zipcode_to_Temp
入力された郵便番号から、その地域の現在の気温を求める	
入力	zipcode(String)
	メタデータ　zipcode,ZIP
出力	temp(String)
	メタデータ　Temperature
例）	”252-0807”→ zipcode_to_Temp →”8”

表 5.3: 気温情報 Web Service

5.3　実験内容

[実験１]　最初のキーワードを address で始める。

・ 予想される連携の図 5.1

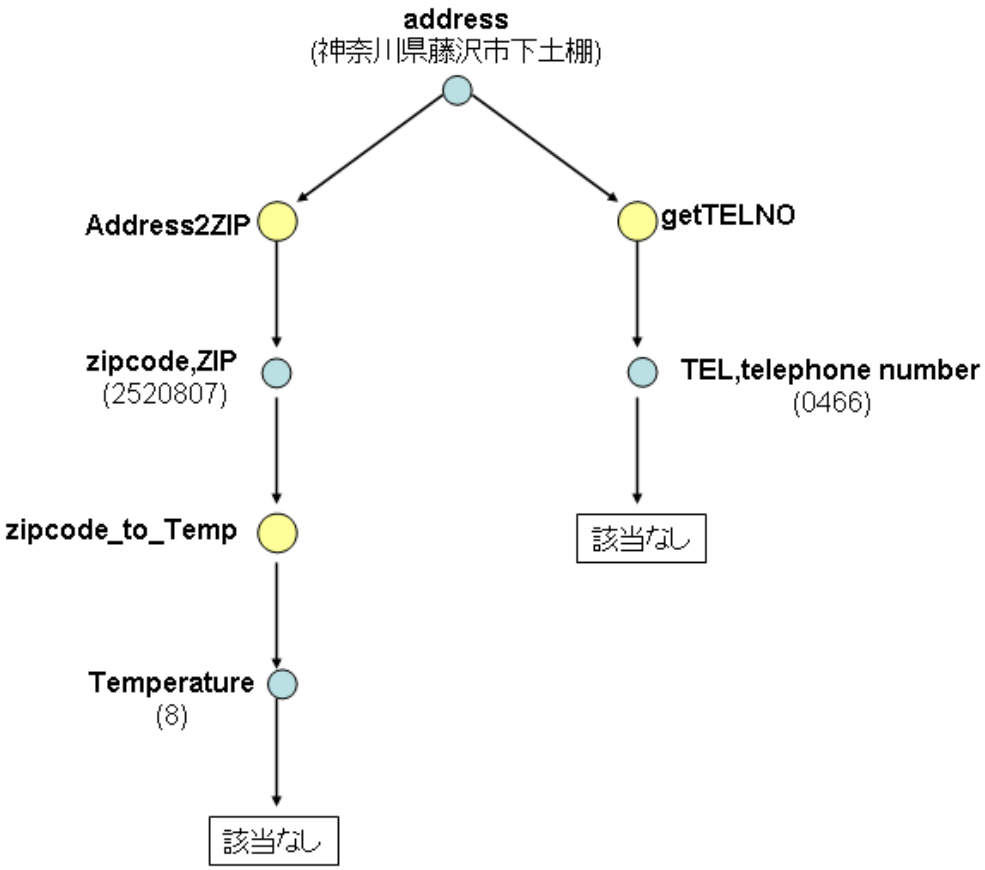

図 5.1: 実験 1 の連携図

・実際の結果
→ Address2TEL と Address2ZIP が表示
→ Address2TEL を利用　　→ Address2ZIP を利用
→結果取得　該当 Web Service なし　→結果取得　ZIP2Temnp が表示
→ ZIP2Temp を利用
→結果取得　該当 Web Service なし

[実験 2]　最初のキーワードを zipcode で始める。
・ 予想される連携の図 5.2

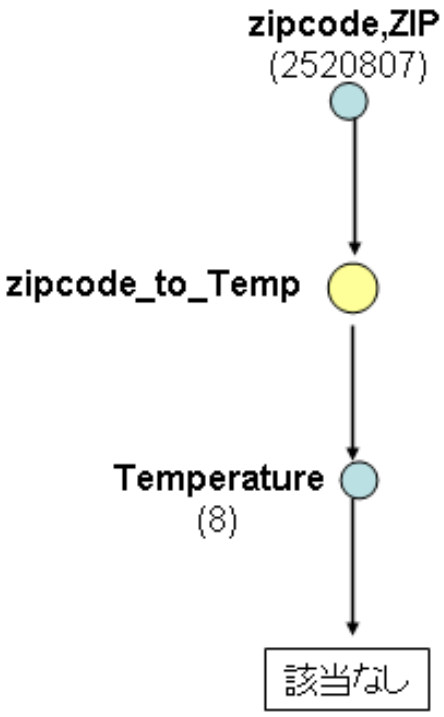

図 5.2: 実験 2 の連携図

・実際の結果
→ ZIP2Temp が表示
→結果取得　該当 Web Service なし
[実験 3]　最初のキーワードを TEL で始める。
・ 予想される連携の図 5.3

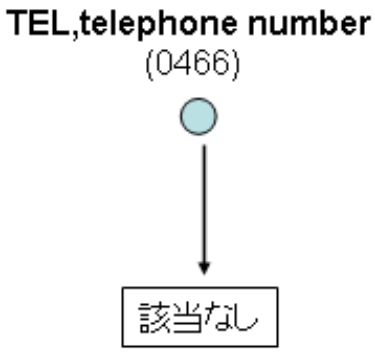

図 5.3: 実験 3 の連携図

・実際の結果

→該当 Web Service なし

5.4 実験による評価

実験により、

- 型があっていてもメタデータがあわないものは候補として表示されない
- メタデータがあうものは、すべて候補として表示出来る

という事が確認出来た。
従って、本実装において正確なフォワードトレースが可能である事を実証出来た。

5.5 比較による定性評価

現在の標準技術のみによる既存システムによる Web Service の利用と本実装を用いた利用を、項目別に比較し、本実装の優位性を評価する。

ここで取り上げる既存システムは、標準技術のみによって構築され、UDDI Registry へ問い合わせを行う事で Web Service を発見し、その場で WSDL 情報を解析する事で、変数名と型の情報を User に提供する事で Web Service の利用を行うシステムを仮定する。

更に、２章で述べた既存システムの問題点として、情報の不足や情報の未整理を挙げたが、この問題を解決する本研究とは違ったアプローチとして、UDDI や WSDL 自体を拡張・改良する事があげられる。このアプローチによって、情報の不足・未整理を解決し、本実装の様な Web Service の連携を支援するシステムを作成した場合を、３つ目の比較対照とする。

評価は、User、Web Service Provider、UDDI Registry の３つの視点からそれぞれの評価項目で比較を行う。以下、比較項目について述べる。

- User の視点から見た比較

  Web Service の利用　発見した Web Service を利用し結果を取得出来るか。

  利用結果から次の Web Service の利用　入出力についての情報で Web Service を結びつけ、連携可能性のある Web Service の一覧を得る事で、Web Service を連携して利用出来るか。

  利用時に表示される情報の多さ　Web Service を利用する際に User に Method の利用方法や処理内容などを提供出来るか。

- System Provider の視点から見た比較

  登録の簡易性　UDDI Registry のみへの登録でよいか、それ以外も必要か。

登録の情報量　登録しなければいけない情報の総量。

登録情報の用意の簡易性　登録しなければいけない情報の用意が簡単に行えるか。

- UDDI Registry から見た比較

追加情報の有無の混在　追加情報を持つものと持たないものが UDDI Registry に混在していても大丈夫か。

追加情報への対応作業量　UDDI Registry が自身のシステムに関して変更を加えなければならない作業の多いか。

	既存システム	本実装	別アプローチ
Web Service の利用	○	○	○
利用結果から次の Web Service の検索利用	×	○	○
利用時に表示される情報の多さ	×	○	○
登録処理の簡易性	○	×	○
登録の情報量	○	×	×
登録情報の用意の簡易性	○	△	×
追加情報の有無の混在	−	○	×
追加情報への対応作業量	−	○	×

表 5.4: 定性評価結果

それぞれの評価についての理由について述べる。

- User の視点から見た比較

Web Service の利用　現在の標準技術にて Web Service を利用する事は可能。

利用結果から次の Web Service の利用　入出力についてのメタデータがなければ不可能なので、本実装及び別アプローチでなければ利用出来ない。

利用時に表示される情報の多さ　追加情報を付加する事で、既存システムよりも処理内容や利用方法など多くの情報を提供する事が出来る。

- System Provider の視点から見た比較

登録の簡易性　本実装は、Web Service についての情報を UDDI Registry と追加情報データベースの２つに分けてもつので、UDDI Registry １つにまとめる他のものより、簡易性に劣る。

登録の情報量　追加情報を付加する分、同じ Web Service でも登録する情報量は増える。

登録情報の用意の簡易性　現在の Web Service 開発ツールの大半は、自動的に WSDL 情報を作成する機能を持つ。WSDL を拡張しメタデータもあわせて持つ為には、これら開発ツール自体にその様な機能をもたせるか、一度自動的に作成された WSDL を読み込んで後からメタデータを付加するツールを作成する必要がある。一方、本実装でもメタデータを用意する必要はあるが、メタデータは WSDL とは切り離している為、自動作成された WSDL 情報はそのまま登録する事が出来る。既存システムでは、メタデータは必要とされない。

- UDDI Registry から見た比較

追加情報の有無の混在　UDDI Registry を拡張した場合、旧登録形態と拡張後の登録形態がまざっていると連携を支援する際の情報表示が正しく行えず、登録されている全てのサービスを拡張後の登録形態に変えなければならない。本実装では、追加情報を別データベースとして用意するので、UDDI Registry に混在する事が可能である。

追加情報への対応作業量　上記作業に加え UDDI Registry のシステム自体を拡張しなければならない。また新しい情報管理インターフェースを作成する必要がある。本実装では、UDDI Registry のシステムには変更を加える必要なく、追加情報を利用する事が出来る。

第6章　結論

6.1　結論

本研究は、既存システムを活用し Web Service の連携を支援するシステムの設計と実装を行った。

Web Service の連携に関連する既存システムには、UDDI や WSDL がある。しかし、入出力についての情報不足、Web Service の利用方法についての情報未整理、Web Service を動的に利用し、連携を支援する Web Service クライアントの不在という問題を抱えている。

この問題に対して、本研究では、追加情報データベースを作成し、既存のシステムとあわせて活用する事で Web Service の連携を支援する Web Service クライアントを実装した。

本研究によって、Web Service を動的に利用し、Web Service を連携させていく事が可能になった。また、評価によって本研究の優位性を明らかに出来た。

6.2　今後の課題

本研究における今後の課題は以下の事項が挙げられる.

6.2.1　バックトレース

本研究では、２章で定義した２つの要件のうちのフォワードトレースの要件を満たすシステムの設計と実装を行った。もう１つの要件であるバックトレースを満たす為には、本システムの追加情報データベース検索機能を拡張し実装する必要がある。

6.2.2　ワークフロー記述言語の搭載

本実装によって、Web Service を連携して利用する事が可能になったが、実際に利用した順を記録する機能は実装されていない。Web Service の連携を記述する手法としてワークフロー記述言語を挙げる事が出来る。ワークフローとは、

- 複数の処理を呼び出す手順、すなわち手続き規則を持っている
- 流れ作業の要領で処理間でデータの受け渡しを行う

- 具体的な業務目的を実現する

というものであり、これを記述する言語がワークフロー記述言語である。Web Service に対応したワークフロー記述言語として、IBM 社が開発した WSFL(10) や Microsoft 社が開発した XLANG(11) などがある。本実装に、この様なワークフロー記述言語を用いて、実際に利用した Web Service の連携を記録する機能を実装する事により、1 度利用した連携を再度利用する事や、有用な連携を他のユーザーに伝えるといったさらなる活用が見込める。

6.2.3 Semantic Web の技術の応用

本システムは、追加情報データベースに記録した入出力についてのメタデータもとに、連携の可能性がある Web Service の検索を行っている。しかし、メタデータのセマンティクスを解釈するといった機能は実装されていない。このメタデータに、メタデータの階層構造やオントロジーといった Semantic Web(12) の概念を適用し、メタデータを RDF(13) や OWL(14) などの記述言語を用いて記述する事により、Web Service の検索においても、より高度な検索・絞り込みを行う事が可能になる。これにより、連携の可能性がある Web Service の候補が増えた際、より有用な Web Service を絞り込んでいく事が出来る。

謝辞

本研究を進めるにあたり、御指導を頂きました、慶應義塾大学環境情報学部教授の村井純博士、徳田英幸博士、同学部助教授の楠本博之博士、中村修博士、同大学環境情報学部専任講師の南政樹氏に感謝致します。

テーマの定まらない時期から絶えずご指導とご助言を頂きました慶應義塾大学大学院政策・メディア研究科博士課程の石田剛朗氏、同大学大学院政策・メディア研究科修士課程の本波友行氏、同大学総合政策学部久松慎一氏に感謝致します。

また、論文執筆時にご助言を頂きました須子 善彦氏、英文の作成をお手伝い頂いた源中由記氏、絵を提供してくれた橋本和樹氏、に感謝致します。

参考文献

[1] 株式会社サイバーマップ・ジャパン **マピオン** http://www.mapion.co.jp/

[2] ジョルダン株式会社 **乗換案内** http://www.jorudan.co.jp/

[3] **日本道路公団** http://www.jhnet.go.jp/

[4] W3C XML Protocol Working Group **SOAP** http://www.w3.org/2000/xp/Group/

[5] uddi.org **UDDI** http://www.uddi.org/

[6] W3C Web Services Description Working Group **WSDL** http://www.w3.org/2002/ws/desc/

[7] ヤフー株式会社 **Yahoo! JAPAN** http://www.yahoo.co.jp/

[8] Google 社 **Google** http://www.google.co.jp/

[9] 日経 BP 社 **IT Pro 2003/1/16 インターネット上で“ 使える ”Web サービス，探せます** http://itpro.nikkeibp.co.jp/free/NC/NEWS/20030116/3/

[10] IBM Corporation **WSFL** http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf

[11] Microsoft Corporation **XLANG** http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm

[12] Semantic Web Activity **Semantic Web** http://www.w3.org/2001/sw/

[13] W3C Semantic Web Activity **RDF** http://www.w3.org/RDF/

[14] W3C Semantic Web Activity **OWL** http://www.w3.org/TR/owl-ref/