Webサービスによる分散XMLデータの共有方式

服部哲[1]・田畑邦晃[1]

情報学部　情報工学科 ({ahattori, tabata}@ic.kanagawa-it.ac.jp)

Distributed XML Data Sharing System Using Web Services

Akira HATTORI[1], Kuniaki TABATA[1]

abstract

In this paper, we propose a system for sharing distributed XML data to organizations based on Web services technologies. In our system, organizations add some tags to a template, which is a common part of the tags among the XML data of them, and inform each other the tags. The system consists of tag accumulating server, data providing servers, and application servers. These servers communicate with each other by SOAP (Simple Object Access Protocol). The tag accumulating server stores and provides the tags which are added to the template. The data providing server searches for XML data which the organization manages. As a result of prototype system, which is to share welfare information, we found that an organization could easily register the tags which the organization added to the template with the accumulating server and distributed XML data could be exchanged properly among organizations.

Key Words: XML, Information sharing, Web services, Template

1．　はじめに

インターネットの技術を利用して組織間で情報共有することは重要であり，今日では，XML (eXtensible Markup Language)を活用する必要性が高まっている[1]. XMLはタグを利用してデータを記述する言語であるが，タグの名前と構造(以下，タグセット)を自由に決め，データの構造と意味を保持してインターネット上でデータ交換できる点に特徴がある[2]. 各組織はそれぞれの視点で集めた情報をXMLデータとして分かりやすく記述し蓄積することができるため，各組織に分散したXMLデータを共有するためのシステムの開発が期待されており，分散データを統合的に検索できるようにする研究[3][4]やXMLデータを交換・利用するためのプラットフォーム開発の研究[5]などが行われている.

一方，ネットワーク上の分散したアプリケーションを

連携させる技術としてWebサービスが注目を集めている[6]. Webサービスは XMLに基づく分散処理技術であり，WSDL (Web Service Description Language)で記述されたサービスをSOAP (Simple Object Access Protocol)を利用して実行することができ，アプリケーション間でやり取りされるデータはすべてXMLで記述される.

そのため，近年，Webサービス技術を利用して分散XMLデータを共有するシステムがいつくか提案されている[7)~10)]. たとえば，XMLデータにWebサービスを実行するための情報を記述する方法や，分散データを連携させるシステムが研究されている. しかし，これらの研究では各組織のXMLデータで利用するタグセットを統一にする，あるいは，各組織が異なるタグセットを利用して同じ情報を記述することを前提としている. 組織はそれぞれが目的を持って活動し情報を収集・活用しているため，情報共有に係わる組織間のタグセットを統一にすることは困難である.

そこで本研究では，XMLのツリー構造について上位のタグを決めておき，下位のタグを各組織が自由に決め，そのタグを相互に閲覧できるようにすることで情報共有するという考えを導入した，Webサービスによる分散データの共有方式を提案する. 本方式では，各組織はそれぞれの視点に基づく情報を記述し共有することができる.

以下，2章で共有方式とそれに基づくシステム構成を提案し，3章で福祉情報を利用した試作システムについて述べる. 4章ではシステムの試作に基づいて考察し，最後にまとめと今後の課題を述べる.

2. 分散XMLデータの共有方式の提案

本章では，情報共有に係わる各組織におけるタグの拡張性について整理し，Webサービスによる分散XMLデータ共有方式の運用方法とシステム構成を示す.

2．1 タグの拡張性に関する検討

XMLの特徴の一つはタグセットを自由に決めることができる点にある. そこで，情報共有に係わる各組織がどの程度自由にタグを決められるかに着目すると，次の三つの方針に分類できる.

方針1：タグセットを集中管理し各組織にタグの追加を認めない

方針2：各組織が全く自由にタグを定義できる

方針3：タグセットの管理を階層化し一部を集中管理する

方針1では，タグを利用した情報収集や交換が容易であるが，各組織は共通のタグで情報を記述する必要がある. 同一組織内であればタグセットを集中管理しやすいが，各組織はその目的のために主体的に情報を収集しているものであるため，方針1は組織間で情報を共有するには不向きである.

方針2では，他の組織から情報を収集するだけでもタグの変換が要求される. タグの変換にはオントロジーも考慮する必要があるが，情報共有を始める前にオントロジーを作成しておくことが前提となる. そのため，事前に共有する情報を決めておく必要があり，方針2では組織独自の視点で集めた情報を交換しにくい.

方針3では，集中管理されるタグを利用した情報収集ができ，かつ独自情報の記述も可能である. 各組織はそれぞれに目的を持って情報収集するが，全く目的が異なる組織同士が情報共有するとは考えにくいので，最小限の共通部分があったほうが情報を共有しやすい. そのため方針3は組織間におけるXMLデータの共有に適しており，本方式では方針3を採用した.

2．2 システムの運用方法

本方式では，共有されるXMLデータについて，上位のタグセットを集中管理する. これをテンプレートとし，その下で各組織は自由にタグを追加することができる.

まず，情報共有に係わる組織は協力してテンプレートを作成する. 各組織はテンプレートを構成するタグの下に組織独自のタグを追加し，追加したタグの一覧(タグセットとそのタグによりどのような情報を記述するか)を他の組織に報告し，自身の情報を検索する機能をWebサービスとして公開する. 各組織は報告されたタグの一覧を見て，どの情報を利用するかを決め，共有情報を処理するアプリケーションを作成する.

2．3　システム構成

提案方式のシステムは，タグ管理サーバ，データサーバ，アプリケーションサーバにより構成される(図 1)．タグ管理サーバとデータサーバの機能は基本的に Web サービスとして提供され，各サーバ間は SOAP を利用してデータをやり取りする．SOAP は XML でエンコードされたデータを主として HTTP (HyperText Transfer Protocol)でやり取りするためのプロトコルである [6]．

・タグ管理サーバ

情報共有に係わる組織が共同で管理し，各組織が他の組織にタグの一覧を報告するためのものである．そのため，XML を利用してテンプレートと報告されたタグ一覧，およびデータサーバ上の検索機能(Web サービス)の呼び出し方法を記述する WSDL 文書のアドレスを管理し，それらの情報を Web サービスにより提供する(紹介機能)．紹介機能はアプリケーションの開発者や利用者にもタグの一覧を提供する．ある組織のタグの一覧，WSDL 文書のアドレスは一つの要素として管理される．そのため，組織の担当者がその組織で追加したタグの一覧を登録する機能を持つ．紹介機能と登録機能はテンプレートを利用する．これらの機能により，各組織は他の組織にテンプレートの下にどのようなタグを追加したかを報告することができる．また，検索機能を提供することが困難な組織のために，代理でデータサーバ上の XML データを検索する機能を提供する．

・データサーバ

何らかの情報を収集している組織によって管理されるサーバであり，テンプレートを構成するタグに基づきデータを検索する機能を Web サービスとして提供する．この Web サービスのインタフェースを各データサーバ間で統一にする．検索結果は，各組織で管理する XML の形式で返される．データサーバを管理する組織によってデータサーバがアプリケーションサーバになることもある．Web サービスによる検索機能の提供が困難な場合は Web サーバ上に XML データを置き，検索機能はタグ管理サーバの代理機能を利用する．

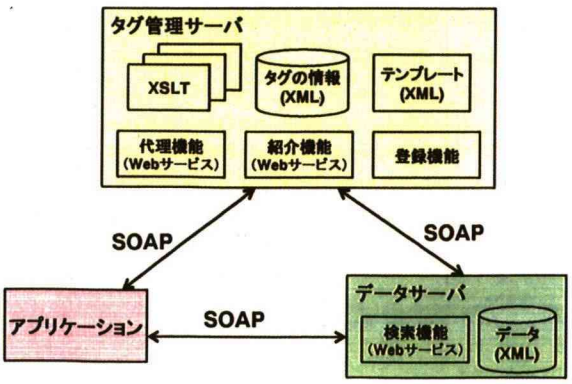

図 1　システム構成

・アプリケーションサーバ

データサーバのデータを利用する組織独自のアプリケーションシステムである．各組織が登録したタグの一覧をタグ管理サーバから受け取り，データサーバの検索機能を呼び出す．データサーバから受け取ったデータをどのように処理するかはアプリケーションに依存する．

3．　福祉情報に基づく試作システム

本章では，福祉情報を例として試作したシステムについて述べる．試作では，Web サービスの実行環境として Apache Axis[11]を利用し，Axis を実行するためのサーブレットエンジンとして Apache Tomcat[12]を利用した．

3．1　福祉情報提供システムとデータサーバ上の Web サービス

近年，入り口付近に段差があるかどうかなど，店舗や公共施設のバリア・バリアフリー情報(以下，福祉情報)を記載した福祉マップが数多く作成され，Web 上で公開されているものも存在する．福祉情報は障害者や高齢者のみでなく多くの人に有用である．現在，様々な組織あるいは個人で情報を収集しているが，一組織や一個人での情報収集には限界がある．そのため，これらの情報を共有可能にすれば，そのような問題の解決につながる．また，様々な組織が独自の視点で福祉情報を集めているが，いくつかの項目は共通している．

これらの理由から，福祉情報を対象にして試作システムを作成した．試作システムのデータサーバ上の Web サービスは，施設名や住所で福祉情報を検索する機能(getFacilityInfo メソッド)と，facility 要素の ID 属性で検

索する機能(getInofByID)を提供する．

3．2　タグ管理サーバの構造

(1) テンプレートとタグの一覧の管理方法

福祉情報を収集しているいくつかのボランティア組織のそれぞれが実際に利用している福祉情報の調査票や，Web 上の福祉情報提供システムを参考にして，試作システムではテンプレートのタグ構造を図 2 のようにした．

一方，各組織でどのようなタグを追加したかを管理するため，テンプレートの facility 要素の子要素のタグから，独自に追加したタグまでのツリー構造をそのまま利用する．たとえば，ある組織が図 3 に示した構造で福祉情報を蓄積しているとすると，その組織が追加したタグの一覧は図 4 のように記述される．図 2 と図 4 から，この組織は toilet 要素の下に二つのタグ(location と type)，equipments 要素の下に一つのタグ(entrance)，および，facility 要素の子要素として comment 要素を内容とするタグ(others)を追加しているということが分かる．

(2) 登録機能

本研究ではタグ管理サーバの登録機能を Web アプリケーションとして開発した．このアプリケーションは，組織が追加したタグを抽出するプログラム，タグの一覧を XML 形式で追加するプログラム，ユーザインタフェースを処理するプログラムから構成される(図 5)．

組織の担当者が検索機能を提供する Web サービスの WSDL 文書のアドレスを入力すると，タグ抽出プログラムは WSDL 文書を解析して getFacilityInfo メソッドを呼び出して検索を実行し，検索結果から facility 要素を一つ取り出し，テンプレートと比較することでその組織で追加されたタグを抽出する．抽出結果は，ユーザインタフェース処理プログラムを介して，そのタグでどのような情報を記述するかを入力するためのテキストボックスとともに表示される．WSDL 文書のアドレスではなく XML データのアドレスが入力された場合は，そのアドレスに基づき XML データから facility 要素を取り出し，同様にタグを抽出しテキストボックスを表示する．組織の担当者がタグで記述される情報の内容を入力す

```
<facility id="1">
  <basicInformation>
    <facilityName>市役所</facilityName>
    <address>○○市△△××丁目</address>
  </basicInformation>
  <equipments>
    <toilet>
      <location>玄関を入り市民課の横を通り抜けた所</location>
      <type>多目的トイレ</type>
    </toilet>
    <parking>あり(3台)</parking>
    <entrance>スロープの幅が狭い</entrance>
  </equipments>
  <others>
      <comment>きれいに清掃されている</comment>
  </others>
  <checker>福祉まちづくりの会</checker>
</facility>
```

図 3　福祉情報の例

```
<dataSource ID="識別番号" wsdl="WSDL文書のアドレス">
  <equipments>
    <toilet>
      <location>施設内のトイレの位置</location>
      <type>トイレの形状</type>
    </toilet>
    <entrance>施設入口</entrance>
  </equipments>
  <others>
    <comment>調査員のコメント</comment>
  </others>
</dataSource>
```

図 4　図 3 に対応するタグの一覧

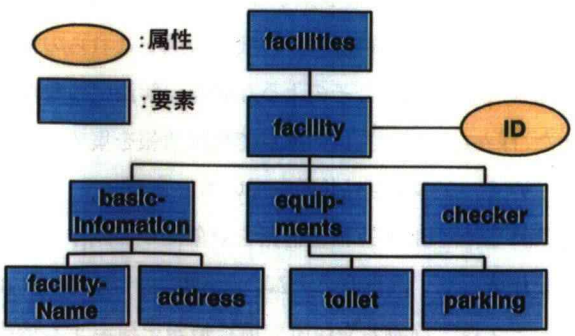

図 2　テンプレートのタグ構造

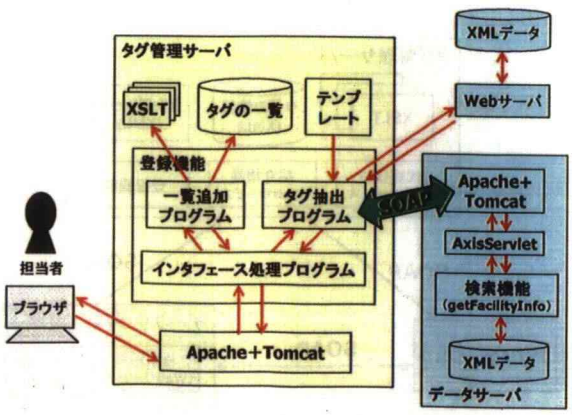

図 5　タグ一覧の登録処理の流れ

ると，一覧追加プログラムは入力内容から図 4 のような XML 形式の要素を生成しタグの一覧に追加する．

(3) 代理の検索機能の実現

他の組織と情報共有する場合，情報の内容によっては他の組織と共有せず，組織内に閉じておきたいこともあると考えられる．Web サービスにより検索機能を提供する組織であれば，そのプログラムの中で共有可能な情報のみを含んだXMLデータを検索結果としてアプリケーションサーバに返すことが可能であるが，タグ管理サーバがデータサーバの代わりに検索機能を提供する場合でも，共有可能な情報のみを含んだ XML データを返すべきである．

そのため，登録機能は追加したタグで記述された情報を共有するかどうかを指定するためのチェックボックスも表示する．そしてタグの一覧の登録時に XML データのアドレスが入力された場合は，チェックされたタグのみを一覧に追加するとともに，代理検索した結果をチェックされたタグのみを含むXMLデータに変換するための XSLT (eXtensible Stylesheet Language Transformations)を生成する．そのため，タグ管理サーバが代理で検索機能を提供する場合でも，組織の担当者は共有可能なタグのみを登録することができ，また，代理機能は生成された XSLT を介すことで共有可能な情報のみを含んだ XML データを返すことができる．

(4) 紹介機能

タグ管理サーバはタグの一覧を提供する機能を持つが，この機能も Web サービスであり，アプリケーションサーバは SOAP により利用することができる．アプリケーションサーバがタグ管理サーバにタグの一覧を要求すると，登録された数だけ図 4 に示した構造のタグ一覧が SOAP レスポンスとして返される(図 6)．この情報を利用することでアプリケーションサーバは，追加されたタグの一覧を利用者に提示したり，各データサーバやタグ管理サーバ上で検索機能を実装する SOAP プログラムのメソッドを呼び出し，福祉情報を検索することができる．一方，紹介機能はタグの一覧を閲覧するための Web アプリケーションとしても提供される．

```
<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <dataSource ID="201"
      wsdl="http://localhost/axis/services/DataSourceAST?wsdl">
   <checker>
    <checkerName>調査員の名前</checkerName>
   </checker>
  </dataSource>

(中略)

  <dataSource ID="206"
      wsdl="http://localhost/axis/services/DataSource?wsdl">
   <equipments>
    <toilet>
     <location>施設内のトイレの位置</location>
     <type>トイレの形状</type>
    </toilet>
    <entrance>施設入口</entrance>
   </equipments>
   <others>
    <comment>調査員のコメント</comment>
   </others>
  </dataSource>
 </soapenv:Body>
</soapenv:Envelope>
```

図 6　紹介機能の SOAP レスポンスの例

3．3　試作システムによる登録・検索

図 7 は，タグ一覧の登録画面において図 3 に示した構造で情報を提供する組織の Web サービスについて記述した WSDL 文書のアドレスを入力した結果であり，location や type などその組織独自のタグが抽出されている．そのため，組織の担当者は，その組織が提供する Web サービスの WSDL 文書のアドレス(または XML データのアドレス)と，テンプレートを構成するタグの下

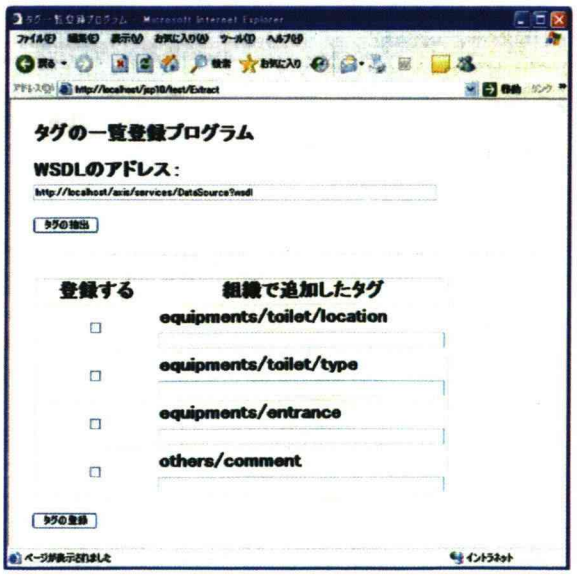

図 7　追加されたタグの抽出結果

にどのようなタグを追加したかを簡単に登録できる.

一方,試作システムでは施設名や住所で福祉情報を検索するアプリケーションを作成した.このアプリケーションは紹介機能からタグの一覧を取得し(図 6),データサーバ上の Web サービスを呼び出す.

図 8 は検索条件として住所に「西尾市寄住町」と入力し検索を実行した結果である.図 8 の結果を得るときのアプリケーションサーバとデータサーバ(図 8 のサーバ 206)における SOAP リクエストとレスポンスをそれぞれ図 9 と図 10 に示す.SOAP リクエストでは,検索条件の施設名と住所をそれぞれ fName 要素と fAddress 要素で,またタグの一覧でデータサーバを識別するための ID 属性の値を ds 要素で指定して,検索機能を提供する Web サービスの getFacilityInfo メソッドを呼び出す.ただし施設名は条件として入力されていないので fName 要素は空要素である.指定されていないのでまた図 11 は検索結果から詳細情報を検索した結果である.図 11 では,検索対象となったデータサーバを管理する組織がテンプレートの下に追加したタグによって記述された情報の部分を点線の枠で示した.このように組織独自のタグで記述された情報も共有可能である.

なお,タグ抽出プログラム,getFacilityInfo メソッド,および getFacilityInfo メソッドを呼び出すプログラムを付録として記載する.

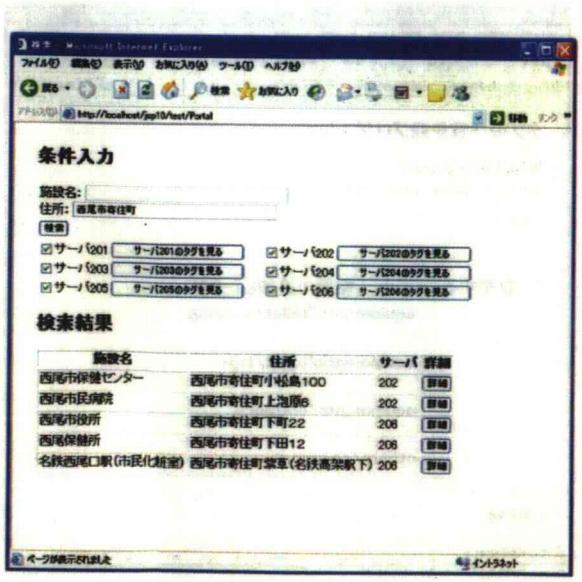

図 8　福祉情報の検索結果

4.　考察

本章では,システムを試作した結果に基づき,提案システムの有用性について検討する.

(1) Web サービスによる検索機能の提供について

提案方式では XML データの検索は,各組織が管理するサーバ上で行われるが,タグ管理サーバが各組織から XML データを集めて検索するという方法も考えられる.しかしこの方法では,他の組織と共有したくない情報もタグ管理サーバに集められたり,それを避けるためには共有する情報と共有しない情報を組織の側で分けて管

```
<?xml version="1.0" encoding="utf-8" ?>
<soapenv:Envelope
    xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <ns1:getFacilityInfo xmlns:ns1="http://DefaultNamespace">
   <fName />
   <fAddress>西尾市寄住町</fAddress>
   <ds>206</ds>
  </ns1:getFacilityInfo>
 </soapenv:Body>
</soapenv:Envelope>
```

図 9　検索機能への SOAP リクエストの例

```
<?xml version="1.0" encoding="UTF-8" ?>
<soapenv:Envelope
    xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soapenv:Body>
  <facility id="1">
   <basicInformation>
    <facilityName>西尾市役所</facilityName>
    <address>西尾市寄住町下町22</address>
   </basicInformation>
   <equipments>
    <toilet>
     <location>玄関を入り市民課の横を通り抜けた所</location>
     <type>多目的トイレ</type>
    </toilet>
    <parking>あり(3台)</parking>
    <entrance>スロープの幅が狭い</entrance>
   </equipments>
   <others>
    <comment>きれいに清掃されている</comment>
   </others>
   <checker>福祉まちづくりの会</checker>
  </facility>
  <facility id="4">
   <basicInformation>
    <facilityName>西尾保健所</facilityName>
    <address>西尾市寄住町下田12</address>
   </basicInformation>

(中略)

  </facility>
 </soapenv:Body>
</soapenv:Envelope>
```

図 10　検索機能からの SOAP レスポンスの例

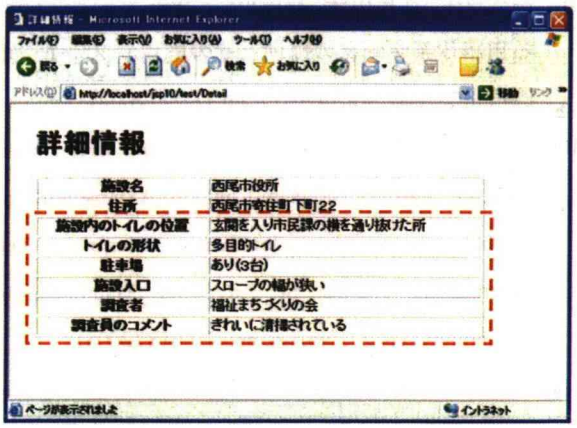

図 11　詳細情報の表示画面

理する必要があり手間がかかるなどの問題がある. タグ管理サーバは代理検索の機能も備えるが，XML データ自体はデータサーバ上に置かれており, データを一箇所に集約する手法ではない.

また，本研究ではデータ共有に係わる各組織が Web サービスにより検索機能を提供するようにしている. そのため, アプリケーションプログラムの開発言語やオペレーティングシステムに依存することなく検索機能を呼び出すことができるため, 様々な組織との情報共有に適している. また, 各組織のサーバ上で検索することで, 検索要求に対して組織内で閉じておきたい情報を検索結果から削除した上で返すことができる.

(2) テンプレートの導入について

テンプレートを導入したことにより，分散 XML データに一貫性を持たせたまま各組織は自由にタグを追加することができ，また，タグ管理サーバを通じて相互に追加したタグを報告することで, そのタグで記述されたデータを容易に交換することができた. 従来研究のようにタグセットを組織間で統一にする場合, そのタグで記述することができないデータを交換することは困難であったが，本システムはその問題を解決できる.

さらに, 各組織の視点で集めた情報をテンプレートの下に集約するといったことも期待できる. しかしそのためには, 各組織の情報が同じものを対象にしていることが分かるようにする必要があり, テンプレートのタグセットだけでなくその内容についても集中管理するなどの工夫が必要である.

(3) タグの一覧の管理方法について

本研究では, 情報共有に係わる組織はどのようなタグを追加したかをタグ管理サーバに登録することで他の組織に報告するというアプローチを採っている. 各組織が DTD (Document Type Definition)を管理するという方法も考えられるが，この方法では，タグが変更された場合，他の組織がそのことを知ることが困難である.

一方, タグ管理サーバではテンプレートの下にどのようなタグが追加されたかを管理する. そのため，タグの一覧が登録された後でも定期的にデータサーバから XML データを取得し，タグ管理サーバ上の情報を更新できる利点がある.

また本研究では, 情報共有に係わるすべての組織からタグの一覧を見ることができる. タグの一覧を登録する際にその情報を見ることができる組織を指定し, 特定の組織とのみ情報を共有するという方法も考えられる. しかし多様な視点からの情報を集めるという点では, すべての組織からタグの一覧を見ることができたほうが有利である.

(4) タグの一覧の登録について

試作システムでは，タグの一覧を登録する際，タグ管理サーバが入力された WSDL 文書や XML データのアドレスを利用してデータサーバから XML データを検索・取得し，組織で追加されたタグを抽出することができた. したがって，組織の担当者は容易にタグの一覧を登録することができる. また, タグを抽出する仕組みは, データサーバ上でタグの変更がないかどうかを確認し, タグの一覧を最新の状態に保つことにも応用できる. しかし試作システムでは文字データと要素を両方含む混在要素を考慮していない. 多様な視点からの情報を共有するためには，これらを含んだ XML データの場合でもタグの一覧を登録できるようにする必要がある.

5.　おわりに

本研究では，Web サービスにより各組織に分散した XML データを共有するための方式を提案した. 本方式は，各組織の XML のツリー構造について，上位のタグセットをテンプレートとして共通化し, それより下に各

組織でどのようなタグを追加したかを相互に閲覧できるように集約的に管理する．福祉情報を対象として，アプリケーションサーバ，タグ管理サーバ，データサーバからなるシステムを試作した結果，追加したタグの登録が容易に行え，Web サービスにより分散した XML データを組織間で適切に交換・共有できることを確認した．

今後は，データの融合や混在要素の処理など，システム試作の結果明らかになった問題の解決が課題である．また，組織間の交渉を通じて共有する情報を決めるというアプローチに基づくシステムも検討する必要がある．

参考文献

1) Anant Jhingran, Nelson Mattos, and Hamid Pirahesh: Information integration: A research agenda, IBM Systems Journal, Vol 41, No. 4, pp.555-562 (2002).
2) Elisa Bertino, Elena Ferrari: XML and Data Integration, IEEE Internet Computing, Vol.5, No.6, pp.75-76 (2001).
3) Bernd Amann, Catriel Beeri, Irini Fundulaki and Michel Scholl: Ontology-Based Integration of XML Web Resources, International Semantic Web Conference 2002, pp.117-131 (2002).
4) Laks V. S. Lakshmanan, Fereidoon Sadri: XML Interoperability, WebDB 2003, pp.19-24 (2003).
5) 大谷泰昭, 和田宇生, 山崎暢也, 橋本明彦, 吉永成利, 長濱勝文, 斉藤隆之, 池田隆志: 知見情報交換プラットフォームによる産業基盤先端ソフトウェア育成・普及実証実験, 次世代デジタル応用基盤技術開発事業, pp.241-248 (2000).
6) Thomas Erl: Service-Oriented Architecture, PRENTICE HALL (2005).
7) Mark Hansen, Stuart E. Madnick, Michael Siegel: Data Integration using Web Services, DiWeb 2002, pp.3-16 (2002).
8) Fujun Zhub, Mark Turnera, Ioannis Kotsiopoulosc, Keith Bennettb, Michelle Russelld, David Budgena, Pearl Breretona, John Keanec, Paul Layzellc and Michael Rigbyd: Dynamic Data Integration using Web Services, Proceedings of the IEEE International Conference on Web Services, pp.262-269 (2004)
9) 高久雅生, 江草由佳, 石塚英弘: Web サービスによる用語体系データの提供とその応用システム, 知識情報学会誌, Vol.14, No.1, pp.11-22 (2004).
10) Hyun Kim, Hyung-Sun Kim, Joo-Haeng Lee, Jin-Mi Jung, Jae Yeol Lee and Nam-Chul Do: A framework for sharing product information across enterprises, The International Journal of Advanced Manufacturing Technology, Vol.27, No.5-6, pp.610-618 (2006).
11) Apache Axis: http://ws.apache.org/axis/, Apache Software Foundation (2006 年 9 月 26 日アクセス).
12) Apache Tomcat: http://tomcat.apache.org/, Apache Software Foundation (2006 年 9 月 26 日アクセス).

付録 1　タグ抽出プログラム(メソッド)

```
public Vector doSearch() throws Exception {
 // getFacilityInfoメソッドを呼び出して検索を実行する部分は,
 // 付録3とほぼ同じなので省略
}

public void doExtract() throws Exception {
 try {
  javax.xml.parsers.DocumentBuilderFactory objDbf =
      javax.xml.parsers.DocumentBuilderFactory.newInstance();
  javax.xml.parsers.DocumentBuilder objDb =
      objDbf.newDocumentBuilder();
  Document objDoc = objDb.parse(this.commonF);
  Node c = objDoc.getElementsByTagName("facility").item(0);
  NodeList ccl = c.getChildNodes();
  for (int i=0; i<ccl.getLength(); i++) {
   if (ccl.item(i).getNodeType() == Node.ELEMENT_NODE) {
    String ccln = ccl.item(i).getNodeName();
    NodeList ecl = this.orgElement.getChildNodes();
    System.out.println(ecl.getLength());
    System.out.println(this.orgElement.getNodeName());
    int j;
    for (j=0; j<ecl.getLength(); j++) {
     if (ecl.item(j).getNodeType() == Node.ELEMENT_NODE) {
      if(ccln.equals(ecl.item(j).getNodeName())) break;
     }
    }
    NodeList cclcl = ccl.item(i).getChildNodes();
    int k;
    for (k=0; k<cclcl.getLength(); k++)
     if (cclcl.item(k).getNodeType() == Node.ELEMENT_NODE)
      break;
    if (k<cclcl.getLength()) subComp(ccl.item(i), ecl.item(j), ccln);
    else {
     NodeList eclcl = ecl.item(j).getChildNodes();
     for (k=0; k<eclcl.getLength(); k++) {
      if (eclcl.item(k).getNodeType() == Node.ELEMENT_NODE) {
       if (hadExtracted(ccln + "/" + eclcl.item(k).getNodeName()
           + ",")==false) {
        this.checkAttr(eclcl.item(k), ccln + "/" +
            eclcl.item(k).getNodeName());
        this.dispElem(eclcl.item(k), ccln + "/" +
            eclcl.item(k).getNodeName());
       }
      }
     }
    }
   }
  }

  NodeList ecl = this.orgElement.getChildNodes();
  for (int i=0; i<ecl.getLength(); i++) {
   if (ecl.item(i).getNodeType() == Node.ELEMENT_NODE) {
    String ecln = ecl.item(i).getNodeName();
    int j;
    for (j=0; j<ccl.getLength(); j++) {
     if (ccl.item(j).getNodeType() == Node.ELEMENT_NODE) {
      if (ecln.equals(ccl.item(j).getNodeName())) break;
     }
    }
    if (j==ccl.getLength()) {
     this.checkAttr(ecl.item(i), ecln);
     NodeList eclcl = ecl.item(i).getChildNodes();
     int k;
     int f=0;
     for (k=0; k<eclcl.getLength(); k++) {
      if (eclcl.item(k).getNodeType() == Node.ELEMENT_NODE) {
       if (this.v.contains(ecln + "/" +
           eclcl.item(k).getNodeName())==false) {
        this.v.removeElementAt(this.v.indexOf(ecln));
        f=this.dispElem(eclcl.item(k), ecln + "/" +
            eclcl.item(k).getNodeName());
       }
      }
      if (f==0) { this.v.addElement(ecln); }
     }
    }
   }
  }
 } catch (Exception err) { err.printStackTrace(); }
}
```

```
private void subComp(Node n1, Node n2, String s1) {
 NodeList n1cl = n1.getChildNodes();
 for (int i=0; i<n1cl.getLength(); i++) {
  if (n1cl.item(i).getNodeType() == Node.ELEMENT_NODE) {
   String n1cln = n1cl.item(i).getNodeName();
   NodeList n2cl = n2.getChildNodes();
   int j;
   for (j=0; j<n2cl.getLength(); j++) {
    if (n2cl.item(j).getNodeType() == Node.ELEMENT_NODE) {
     if(n1cln.equals(n2cl.item(j).getNodeName())) break;
    }
   }
   NodeList n1clcl = n1cl.item(i).getChildNodes();
   int k;
   for (k=0; k<n1clcl.getLength(); k++)
    if (n1clcl.item(k).getNodeType() == Node.ELEMENT_NODE)
     break;
   if (k<n1clcl.getLength())
    subComp(n1cl.item(i), n2cl.item(j), s1 + "/" + n1cln);
   else {
    NodeList n2clcl = n2cl.item(j).getChildNodes();
    for (k=0; k<n2clcl.getLength(); k++) {
     if (n2clcl.item(k).getNodeType() == Node.ELEMENT_NODE) {
      if (this.v.contains(s1 + "/" + n1cln + "/" +
          n2clcl.item(k).getNodeName())==false) {
       this.checkAttr(n2clcl.item(k), s1 + "/" + n1cln + "/" +
           n2clcl.item(k).getNodeName());
       this.dispElem(n2clcl.item(k), s1 + "/" + n1cln + "/" +
           n2clcl.item(k).getNodeName());
      }
     }
    }
   }
  }
 }
 NodeList n2cl = n2.getChildNodes();
 for (int i=0; i<n2cl.getLength(); i++) {
  if (n2cl.item(i).getNodeType() == Node.ELEMENT_NODE) {
   String n2cln = n2cl.item(i).getNodeName();
   int j;
   for (j=0; j<n1cl.getLength(); j++) {
    if (n1cl.item(j).getNodeType() == Node.ELEMENT_NODE) {
     if (n2cln.equals(n1cl.item(j).getNodeName())) break;
    }
   }
   if (j==n1cl.getLength()) {
    this.checkAttr(n2cl.item(i), s1 + "/" + n2cln);
    NodeList n2clcl = n2cl.item(i).getChildNodes();
    int k; int f=0;
    for (k=0; k<n2clcl.getLength(); k++) {
     if (n2clcl.item(k).getNodeType() == Node.ELEMENT_NODE) {
      if (this.v.contains(s1 + "/" + n2cln + "/" +
          n2clcl.item(k).getNodeName())==false)
       f=this.dispElem(n2clcl.item(k), s1 + "/" + n2cln + "/" +
           n2clcl.item(k).getNodeName());
     }
    }
    if (f==0) { this.v.addElement(s1 + "/" + n2cln); }
   }
  }
 }
}

private void checkAttr(Node n, String s) {
 NamedNodeMap nnm = n.getAttributes();
 for (int i=0; i<nnm.getLength(); i++)
  this.v.addElement(s + "/@" + nnm.item(i).getNodeName());
}

private boolean hadExtracted(String s) {
 if (es.indexOf(s)>=0) return true;
 else return false;
}

private int dispElem(Node n, String s) {
 NodeList ncl = n.getChildNodes(); int f=0;
 for (int i=0; i<ncl.getLength(); i++) {
  if (ncl.item(i).getNodeType() == Node.ELEMENT_NODE) {
   if (this.v.contains(s + "/" + ncl.item(i).getNodeName())==false) {
    this.checkAttr(ncl.item(i), s + "/" + ncl.item(i).getNodeName());
    this.dispElem(ncl.item(i), s + "/" + ncl.item(i).getNodeName());
    f=1;
   }
  }
 }
 if (f==0){ System.out.println(s); this.v.addElement(s); }
 return 1;
}
```

付録2 getFacilityInfo メソッド

```
public Element [] getFacilityInfo(Element [] elems) {
  NodeList n1;
  Element[] output;
  String s1, s2;
  Vector v;
  Enumeration enu;

  try {
   if (elems[0].getElementsByTagName(
        "fName").item(0).hasChildNodes()) {
     s1 = elems[0].getElementsByTagName(
        "fName").item(0).getFirstChild().getNodeValue();
   } else { s1 = ""; }
   if (elems[0].getElementsByTagName(
        "fAddress").item(0).hasChildNodes()) {
     s2 = elems[0].getElementsByTagName(
        "fAddress").item(0).getFirstChild().getNodeValue();
   } else { s2 = ""; }

   javax.xml.parsers.DocumentBuilderFactory objDbf =
       javax.xml.parsers.DocumentBuilderFactory.newInstance();
   javax.xml.parsers.DocumentBuilder objDb =
       objDbf.newDocumentBuilder();
   Document objDoc = objDb.parse("C:¥¥Program Files¥¥Apache
Group¥¥Tomcat 4.1¥¥webapps¥¥axis¥¥Mikawa.xml");

   if (s1.equals("") && s2.equals("")) {
    n1 = objDoc.getElementsByTagName("facility");
    output = new Element[n1.getLength()];
    for (int i=0; i<n1.getLength(); i++)
     output[i] = (Element)n1.item(i);
   } else if (s1.equals("")) {
    n1 = objDoc.getElementsByTagName("basicInformation");
    v = new Vector();
    for (int i=0; i<n1.getLength(); i++) {
     Element e = (Element)n1.item(i);
     if ((e.getElementsByTagName( "address"
          ).item(0).getFirstChild().getNodeValue()).indexOf(s2) >= 0)
      v.addElement((Element)e.getParentNode());
    }
    output = new Element[v.size()];
    enu = v.elements();
    int j=0;
    while (enu.hasMoreElements()) {
     output[j] = (Element)enu.nextElement(); j++;
    }
   } else if (s2.equals("")) {
    n1 = objDoc.getElementsByTagName("basicInformation");
    v = new Vector();
    for (int i=0; i<n1.getLength(); i++) {
     Element e = (Element)n1.item(i);
     if((e.getElementsByTagName("facilityName"
          ).item(0).getFirstChild().getNodeValue()).indexOf(s1) >= 0)
      v.addElement((Element)e.getParentNode());
    }
    output = new Element[v.size()];
    enu = v.elements();
    int j=0;
    while (enu.hasMoreElements()) {
     output[j] = (Element)enu.nextElement(); j++;
    }
   } else {
    n1 = objDoc.getElementsByTagName("basicInformation");
    v = new Vector();
    for (int i=0; i<n1.getLength(); i++) {
     Element e = (Element)n1.item(i);
     if (((e.getElementsByTagName("facilityName"
          ).item(0).getFirstChild().getNodeValue()).indexOf(s1) >= 0)
       && ((e.getElementsByTagName("address"
          ).item(0).getFirstChild().getNodeValue()).indexOf(s2) >= 0))
      v.addElement((Element)e.getParentNode());
    }
    output = new Element[v.size()];
    enu = v.elements();
    int j=0;
    while (enu.hasMoreElements()) {
     output[j] = (Element)enu.nextElement(); j++;
    }
   }
  } catch (Exception err) {
   System.err.println(err); output = new Element[1];
  }
  return output;
}
```

付録3 getFacilityInfo メソッドを呼び出すプログラム(メソッド)

```
public Vector searchByWS() throws Exception {
 String tns, tns1, s1, s2;
 try {
  javax.xml.parsers.DocumentBuilderFactory objDbf =
      javax.xml.parsers.DocumentBuilderFactory.newInstance();
  javax.xml.parsers.DocumentBuilder objDb =
      objDbf.newDocumentBuilder();
  Document objDoc = objDb.parse(this.wsdlURL);
  NamedNodeMap nnm =
      objDoc.getDocumentElement().getAttributes();
  tns = nnm.getNamedItem("targetNamespace").getNodeValue();
  tns1 = nnm.getNamedItem("xmlns:tns1").getNodeValue();
  s1 = objDoc.getElementsByTagName(
      "wsdl:service").item(0).getAttributes().getNamedItem(
      "name").getNodeValue();
  s2 = ((Element)(objDoc.getElementsByTagName(
      "wsdl:service").item(0))).getElementsByTagName(
      "wsdl:port").item(0).getAttributes().getNamedItem(
      "name").getNodeValue();
 } catch (Exception err) {
  System.err.println(err);
  return (new Vector());
 }

 Service  service = new Service(new URL(this.wsdlURL),
     new QName(tns,s1));
 Call call = (Call)service.createCall(new QName(tns,s2));

 SOAPBodyElement[] child = new SOAPBodyElement[3];
 SOAPBodyElement[] input = new SOAPBodyElement[1];
 child[0] = new SOAPBodyElement(XMLUtils.StringToElement(
     "", "fName", this.name));
 child[1] = new SOAPBodyElement(XMLUtils.StringToElement(
     "", "fAddress", this.address));
 child[2] = new SOAPBodyElement(XMLUtils.StringToElement(
     "", "ds", this.dsID));
 input[0] = new SOAPBodyElement();
 input[0].setName("getFacilityInfo");
 input[0].setNamespaceURI(tns1);
 input[0].addChild(child[0]);
 input[0].addChild(child[1]);
 input[0].addChild(child[2]);

 Vector elems = (Vector) call.invoke( input );
 Vector es = new Vector();
 SOAPBodyElement elem = null ;
 Element e = null ;
 Enumeration enu = elems.elements();
 while (enu.hasMoreElements()) {
  elem = (SOAPBodyElement) enu.nextElement();
  e = elem.getAsDOM();
  es.addElement(e);
 }
 return( es );
}
```