Integration of Web Information Sources by Mobile Users

Wisut Sae-Tung Tadashi OHMORI Mamoru HOSHI
Graduate School of Information System, University of Electro-Communications

1 Introduction

In today's computer internetworks, many information sources provide their contents via WWWs by embedding their client applications into Web pages. For example, ordinary HTML document servers, CGI-form texts with backend database servers, and java-based clickable maps with backend data services can be found everywhere in the Internet. Let us call such an information source *a Web Information Source* (WIS). In general, these WISs are maintained by multiple autonomous organizations. Let us call such an organization a *cell*. In this environment, mobile users move across cells and download client applications from WISs to query information. This environment is shown in Figure 1.

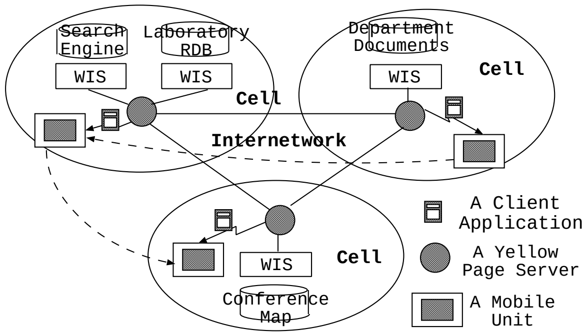

Figure 1: Web Information Sources in a mobile environment

Under this situation, this paper proposes a new information-integration system for mobile users; our goal is to realize an environment where mobile users move across different cells, find/download some interesting WISs from there, and can *integrate* those WISs as a new (more convenient) WIS on users' mobile computers in a disconnected mode. Previous works[1,2] have discussed integration of WISs on a centralized site of developers, but do not consider appropriate style of integration for mobile users and how to resolve semantic conflict in such a situation.

For our goal, we firstly propose *navigational integration* of WISs; this is a new style of integration for mobile users. Next, we propose a domain hierarchy; this is a new semantic conflict resolution technique. Using these ideas, we propose a system architecture where necessary common data models of WISs are downloaded (from each cell) into users' mobile computers. Then we describe, in this system, how the users apply navigational integration of WISs in a network-disconnected mode on their computers.

2 Navigational Integration

The *navigational integration* of WISs is defined as a process of retrieving original Web pages from several sites of WISs of interest and generating new Web pages (as a new WIS); furthermore, each of these new pages must include *derived links*, which represent relationship between data of these different pages.

Let us consider for example in Figure 2-a. The first (WIS1) is an ordinary WWW document server. It consists of a Web top-page and some HTML-pages (under a given DTD) describing departments; further, the top-page includes URL links to these HTML-pages. The second (WIS2) is a CGI-based page through which a user can input condition parameters to retrieve a dynamic result page. Then, the user wants to generate a new WIS (WIS3) from the original WISs by creating *derived links* between these WISs. The derived links pass the data (i.e., laboratory name) from the output page of the WIS1 to WIS2 as the input conditions to get the result page of WIS2 (see Figure 2-b). By clicking these derived links, the user can navigate between contents of the web pages of two WISs. However, *semantic conflicts* between two data such as different format representations, part-of relationships or is-a relationships may occur here. Thus, to integrate multiple WISs together, such semantic conflicts must be resolved.

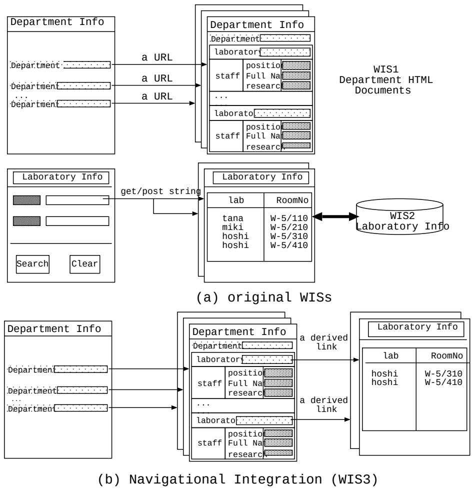

Figure 2: Navigational integration

2.1 definition

To describe the above concept formally, let us consider a navigational integration from a WIS S to another WIS R. (S is called a sender. R is called a receiver). Then, a derived link from S to R represents that an *element* in a Web page of S has a relationship with an element (in a Web page) of R. Let x and y be such an element of S and that of R, respectively. Then a navigational integration from S to R by creating derived links from x to y is described as follows:

$$S \Rightarrow R \quad \stackrel{def}{=} \quad f(S.x) = R.y,$$

where f is a function to convert semantics from $S.x$ to $R.y$. (the symbol, $\Rightarrow$, means creating derived links.)

Currently, we restrict navigational integration in such a way that an element x of a WIS is given by a dot-notation (e.g., $S.x1.x2$); (this dot-notation is that used in many object-oriented SQL-like languages.); furthermore, we assume that an *element* in navigational integration must take an ATOMIC value. By these restrictions, we can avoid ambiguity which would occur when considering integration from a list to another list.

Figure 3 describes four cases of navigational integration (from S to R). Case 1 and 3 are used when the receiver R is an ordinary document WIS. Case 2 and 4 are used when R provides an access method using an entity y (e.g., a CGI-form text). As shown in this figure, we use a mapping table F in case of a one-to-many relationship from $S.x$ to $R.y$.

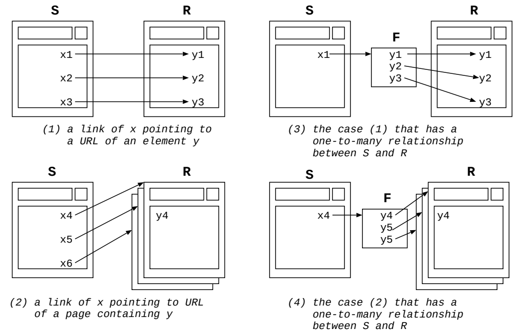

Figure 3: A general form of navigational integration

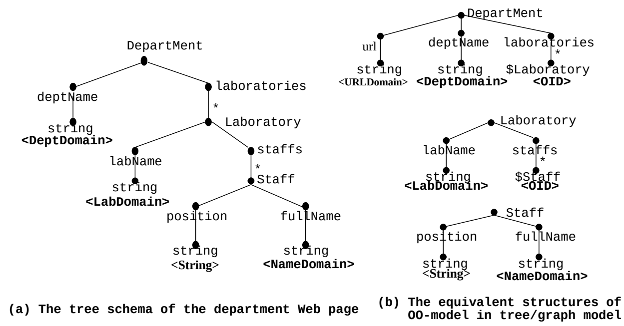

Figure 4: Schema and structure

2.2 common data model

The common data model we use is a variant of the object-oriented (OO) model. As a unique property, we add a concept called *domain hierarchy*, which will be described later in Section 3.

In our policy, a web page is regarded as a nested structure of objects. Further, in order to model a function-embedded WIS such as a CGI-page, we give methods to this structure. Such a structure of objects with methods are called *interface definitions* in our system. Then, we wrap the server function of a WIS by a set of interface definitions. (Nested data occurring in a web page are represented as relationships between interface definitions via the multi-valued attribute.)

Let us show how to wrap the first WIS of Figure 2-a. This WIS is a structured document. Its DTD is described in Figure 4-a. Then, Figure 4-b describes an equivalent schema in our data model.

Figure 5 shows the interface definitions wrapping the first WIS. In order to explain interface definitions, let us use the interface definition of *Department* (in Figure 5-a). This interface def-

```
database ISInformation
address  http://HOST1/cgi-bin/execute.pl
interface  Department
body
    url             URLDomain,
    deptName        DeptDomain,
    laboratories    setof Laboratory,
method
public:
  static {DepartMent} getByURL(URLDomain $url);
private:
  void printDatail();
  OID  new(string $deptName,OID $lab)
implement
 sub new {
     my $package  = shift;
     my $deptName = shift;
     my $labOID = shift;
     my $this;

     $this = DepartMent->newEmptyObject();
     $this->{deptName} = $depName;
     push(@{$this->{laboratories}},$labOID);
     bless $this;
     return $this;
 }
  ...
  endInterface
```

(a) The "Department" Interface Definition

```
database ISInformation
address  http://HOST1/cgi-bin/execute.pl
interface  Laboratory
body
   labName      LabDomain,
   staffs       setof Staff,
method
public:
   ...

endInterface
```

(b) The "Laboratory" Interface Definition

```
database ISInformation
address  http://HOST1/cgi-bin/execute.pl
interface  Staffs
body
   position    PostDomain,
   fullName    NameDomain,
method
...
...
endInterface
```

(b) The "Staff" Interface Definition

Figure 5: Interface definitions for describing a Department document

inition has three attributes: *orig_url*, *deptName* and *laboratories*. The second attribute is a single-value of string and its domain is *DeptDomain*. The third attribute is a multi-value keeping OIDs of instances of a Laboratory interface. Using such multi-valued attribute, we can represent a nested structure of documents. (The first attribute *orig_url* is the URL of the original document page.)

As shown above, the data-types in our system are only atomic data and OID. *Domain* must be assigned for all atomic data. Here, a *domain* in our approach refers to a pool of values that have an exact data-format and meaning.

In the method part, there are two types of methods; a **public** and **private** types. Only *public* methods are invoked explicitly. In this example, *getbyUrl()* method is the way to retrieve an original document having a given URL.

Some of public methods can have a keyword *static* in their method signatures. Roughly speaking, a static (and public) method indicates a class method to retrieve objects from the (wrapped) WIS by invoking an native method. (The other methods are used for filtering such generated data-objects.)

3 Domain Hierarchy

In order to perform navigational integration described above, we must resolve the semantic conflict among different WISs. To achieve this ability, we propose a *domain hierarchy* (DH), which is maintained among all cells in a given network. Here, a *domain* in our approach refers to a pool of values that have an exact data-format and meaning. Figure 6 shows an example of the domain hierarchy, which is maintained by two cells. The domain hierarchy consists of two components: (i) multiple local domain hierarchies (LDHs); each LDH is maintained individually in each cell, and (ii) a common domain hierarchy (CDH); this is shared by all cells. (When multiple cells share the same CDH, these cells are called a *region*.)

In each cell, there is a *wrapper*[1,2,3], which manages all WISs that belong to the cell. This wrapper wraps and describes schema and access methods of a given WIS into a unified form called *an interface definition*[3]. Using this wrapping, attributes and arguments of access methods are mapped to the domains in the local domain hierarchy of the cell. Moreover, domains of each LDH are linked to the domains in the CDH for providing semantic convertibility of data in different cells.

Construction: Domains are organized into a hi-

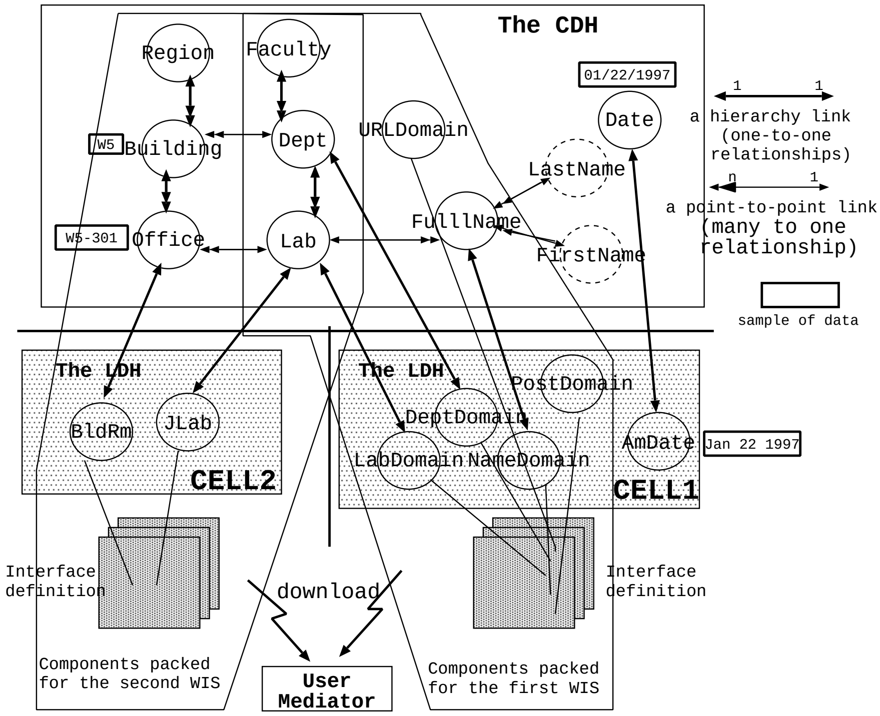

Figure 6: Domain hierarchy and downloading mechanism

erarchical structure, where a type of hierarchal relationship is a one-to-one relationship (data-format conversion), or an is-a relationship, or a part-of relationship. Therefore, for any given domain D, its parent domain is equal to, is more general than, or contains D. The *hierarchy links* (depicted by bold links in Figure 6) represent such hierarchical relationships.

As a result, for any given domain D, D has a hierarchical relationship with its parent. However, D may have other relationships with some domains D'_i $(i = 1, ..., n)$. These relationships are described as *point-to-point links* (depicted by fine links in Figure 6) between D and D'_i. Therefore, D can have one hierarchy link to its parent and several point-to-point links to D'_i. (In implementation, each link is described by conversion functions for converting semantics between the two associated domains.) We refer to these links (with their conversion functions) as the *domain rule* of D.

Download: While a mobile user moves across networks and finds multiple WISs, he downloads software components describing services of WISs from their cells into his mobile computer. A *software component* of a WIS consists of (i) interface definitions of the WIS, (ii) domain rules of the domains used in these interface definitions, (let us refer to these domains as X), and (iii) the domain rules of the ancestor domains of X. Therefore, as a result of downloading, a sub-part of the DH is maintained in the user's mobile computer. The download processing is shown in Figure 6.

In our system, a *mediator*[3] is installed on a user's computer. Then, by using this sub-part of the DH, the mediator on the user's computer resolves the semantic conflict between two data of different cells by finding a conversion path between domains of these two data. For example, in Figure 6, the conversion path from the domain *NameDomain* to the domain *JLab* is NameDomain→FullName→Lab→JLab. The conversion functions along the conversion path are used to convert the semantics between them. Note that this step is performed on the mobile computer in a disconnected mode.

Legality: Conversion paths are not always legal. If a conversion path causes a many-to-many relationship between a start domain and a destination domain, unexpected data will emerge. Such a conversion path is regarded illegal. The mediator must consider this restriction when finding a conversion path.

Reachability: In our scheme, a conversion path between two domains is assured to exist if there is: (i) a domain that is an ancestor of both domains; or (ii) only one point-to-point link between ancestors of both domains. [1]

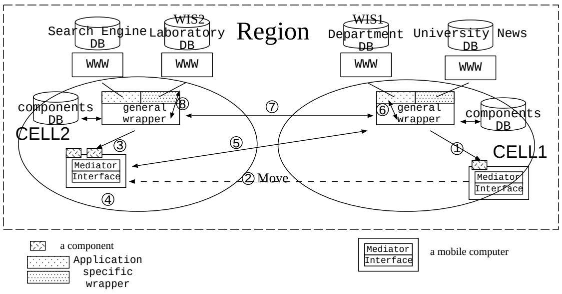

Figure 7: System architecture and integration processing

Integration Process: The overall behavior of the system is shown in Figure 7. A mobile user moves across two cells from CELL1 to CELL2 and downloads software components of WIS1 (in CELL1) and WIS2 (in CELL2) as shown in step 1-3. Next, by the indication of the user, the mediator on the user's computer uses the sub-part DH in the mobile computer to resolve semantic conflict between WIS1 and WIS2, and it generates a SQL-like script for navigational data-access from WIS1 to WIS2 (step 4). The mediator also automatically embeds this script into the copy of the top page of WIS1. The user can use it as a new WIS (WIS3). We should note that the copy of the top page of WIS1 has been already downloaded with the software components. Thus, the mediator can do this embedding step on the mobile computer in the disconnected mode. As a result, the user can navigate from the contents of WIS1 to those of WIS2 by clicking derived links in WIS3 (step 5-8).

4 Example

Here we are ready to demonstrate how navigational integration is implemented. Assume that a mobile user finds the two WISs (WIS1 and WIS2 in Fig.2-a) and wants to do navigation-mode integration of them (WIS3 in Fig.2-b). To do so, we provide a SQL-like query language.

[1] Generally speaking, if a domain X has an ancestor who knows a point-to-point link to Z and if the same condition holds for another domain Y, then X and Y are reachable to each other.

The following script is a command to do navigational integration from a given (single) department page of the first WIS (whose url = `URLx`) to the second WIS:

```
select *
from   D in Department, L in LabBuilding
source Department of ISInformation
       on http://HOST1/cgi-bin/execute.pl,
       LabBuilding of uecdbWeb
       on http://HOST2/cgi-bin/execute.pl
where  Department->getbyURL(URLx),
       L in LabBuilding->getByLaboratory(
       L:Lab2JLab(D:LabD2Lab(
        D->{laboratories}->{labName})))
```

For example, in the above SQL-like statement, *getByLaboratory* is the method defined in the interface definition of the second WIS; this method is the major function of this WIS. It argument requires data whose domain is *JLab*. Because the domain of $D \rightarrow \{laboratory\} \rightarrow \{labName\}$ is *LabDomain*, the LabD2Lab and Lab2JLab functions are inserted to convert data from the domain *LabDomain* to the domain *Lab* and from the domain *Lab* to the domain *JLab*, respectively.

As a result, the above SQL-like script can execute navigation-mode integration (of the Data-Access parts) with resolving semantic conflicts.

The mediator (located on the user's computer) generates the above script by the interaction with the user. After that, this script is automatically embedded in the front page of WIS1. As a result, the front page becomes that shown in Fig.8.

Figure 8 is the HTML source of the front page embedded with the query command. When the user activates such an embedded link, the query command is sent to be executed at WIS1 and returns an intermediate result to the user. Figure 9 shows this result page. As shown here, the next step of the execution is nothing but a derived link, and this is implemented as a cgi link to activate WIS2.

Reference

[1] D. Florescu, A. Levy and A. Mendelzon: "Database Techniques for the World-Wide Web: Survey". *SIGMOD Record*, 27(3): pp. 59-74, September 1998.
[2] S. Chawathe et al. "The TSIMMIS project: Integration of heterogeneous information sources." In *Tech. Rep of 100th DBS of Information Processing Society of Japan*, pp. 7-18, Tokyo, Japan, 1994.
[3] W. Sae-Tung, T. Ohmori, and M. Hoshi. "An information integration architecture for mobile users in WWW environment". *Trans. Information Processing Society of Japan*, 39(4):pp. 888-900, April 1998.

```
<!DOCTYPE HTML PUBLIC "-//IBM//DTD HPB HTML//EN">
<html>
<head>
<title>IS HP Top Page (ENGLISH)</title>
</head>
<body background="../images/bgd/is-bgd1.gif" bgcolor="#FFFFFF">
<div align="right">
...
...
...
<table width="640" cellspacing="0" cellpadding="0">
  <tr>
    <td valign="top" width="480">
    ...
    ...
    <a href="http://HOST1/cgi-bin/execute1.pl?PPSQL=
       select *
       from   D in Department, L in LabBuilding
       source Department of ISInformation
           on http://HOST1/cgi-bin/execute.pl,
              LabBuilding of uecdbWeb
           on http://HOST2/cgi-bin/execute.pl
       where  DepartMent->getByURL('S.html')
       and    L in LabBuilding->getByLaboratory(
                L:Lab2JLab(D:LabD2Lab(
                  D->{laboratories}->{labName})))">
       Information Systems Science(S) </a>/<br> <a
       href="http://HOST1/cgi-bin/execute1.pl?PPSQL=
       select *
       from   D in Department, L in LabBuilding
       source Department of ISInformation
           on http://HOST1/cgi-bin/execute.pl,
              LabBuilding of uecdbWeb
           on http://HOST2/cgi-bin/execute.pl
       where  DepartMent->getByURL('N.html')
       and    L in LabBuilding->getByLaboratory(
                L:Lab2JLab(D:LabD2Lab(
                  D->{laboratories}->{labName})))">
       Information Network Science(N)</a> /
    ...
    ...
  </tr>
</table>
</body>
</html>
```

a link embedded with a query command

Figure 8: A front page embedded with query commands

```
<!DOCTYPE HTML PUBLIC "-//IBM//DTD HPB HTML//EN">
<html>
<head>
<title>Information Systems Science</title>
</head>
<body bgcolor="#FFFFFF">
<!-- begin Department Object-->
<h2><A Name="908454679-0-deptName">Dep. of Information Systems Science
</A>(S Major)</h2> <div align="center"><center>
.....
.....
.....
<!--begin Laboratoy Object-->
<h3><A Href="http://HOST1/cgi-bin/execute.pl?PPCode=PPCode1035&Step=2&
Method=getDocument&Arg1=Information System Design">
<A Name="908454679-1-labName">Information Systems Design Laboratory</A></A>
</h3> <div align="center"><center>
<table width="90%" cellspacing="0" cellpadding="0">
  <tr>
    <td valign="top">
    <img src="http://www.is.uec.ac.jp/images/photo-zu/s-jikken2.JPG"
    alt="s-jikken2.JPG" align="right" border="0" hspace="8" vspace="8"
    WIDTH="280" HEIGHT="183">
    <!-- begin Staff Object-->
    <A Name="908454679-2-position">Professor</A>
    <A Name="908454679-2-fullName">Toshinori Watanabe</A>'s research
    focuses on adaptive intelligent information systems with
    inference, cognition and evolution mechanisms, modeling and
    verification of information systems from the standpoint of
    concurrency, and the mathematical fundamentals there of.<br>
    <br>
    <!--end Staff Object-->
    <!--begin Staff Object-->
    <A Name="908454679-3-position">Associate Professor</A>
    <A Name="908454679-3-fullName">Shunichi Tano</A>'s life work is
    on the realization of intelligent systems. He is now focusing on
    the computer of the next generation which is interactive and can
    recognize multi-modal input and output such as voice, handwritten
    input, gestures.<br>
    <br>
    <!--end Staff Object-->
....
....
</table>
```

a absolute tag converted by the wrapper

a data tag embedded by the specific wrapper

a object tag embedded by the specific wrapper

a execution tag embedded by the general wrapper in the proxy server

Figure 9: A new source embedded with tags