移動計算機向け情報検索機構の実験報告

Wisut Sae-Tung　大森 匡　星 守
電気通信大学 大学院 情報システム学研究科

移動計算機の普及に伴いユーザはこれを持ち歩き、ネットワーク上で様々な情報を提供するデータベース、いわゆる情報資源から情報検索する機会が非常に増えている。しかし、これらの情報資源はそれぞれが自律的に設計 / 開発されており、統一的な検索インタフェスを提供していない。このような環境では、ユーザが移動計算機を用いて、同時に複数の情報資源を利用するのは難しい。そこで本稿では、永続オブジェクト操作機能を入れたスクリプト言語 Persistent Perl を提案し、それを用いて移動計算機向け情報検索機構を実現する。このシステムのプロトタイプでは Java 応用プログラムと関係データベースと WWW サーチエンジンとの 3 つの異種情報資源を対象した情報検索の実験を行なう。実験の結果、我々のアプローチが実用的なシステムとして可能性があることが分かった。

An Information Retrieval Architecture for Mobile Computers: An Experimental Report

Wisut Sae-Tung　Ohmori Tadashi　Hoshi Mamoru

The University Of Electro-Communications
Graduate School Of Information Systems

Because of the outstanding hardware technology, a size of computers becomes smaller so that users can take their mobile computers and move from one place to another place within networks. While moving, they may want to access information from the information resources that provide different kind of multi-media information. Unfortunately, the information resources connecting to the network do not provide an integrated interface. This causes the difficulties in manipulating information in heterogeneous information resources in integrated manner. To solve this problem, we have designed and developed the Persistent Perl and use it as a communication language between information resources and a mobile computer. The information-retrieval operation is performed as the remote programming style. In this paper, we describe an experiment and show how to use our approach to create an effective information-retrieval application for mobile users in the heterogeneous environment. The heterogeneous information resources used in this experiment are a Java application, a RDB and a WWW search engine.

Keywords　Heterogeneous Information Retrieval, Mobile Computing, Remote Programming.

1 Introduction

Because of recent advance in hardware technology, users can carry their mobile computers with them whenever they move from one place to another place and would like to access information from information resources on the networks through wireless or wired network communication channel. However, the public networks such as the Internet consist of many information servers that are capable of providing many services and employ a wide variety of hardware platforms, operating systems, databases and applications. Consequently, they often provide different data models and access mechanisms. In such an environment, it causes mobile users to obey their access mechanisms for manipulating information. The mobile users can not access these information resources in an integrated manner. Therefore, they must load each information into their local computer by different access manner and combine this information by themselves.

Figure 1 shows the general heterogeneous information resources environment. We assume that each information resource is spontaneous and a mobile user moves from one place to another to find information from the information resources. For example, a mobile user finds the image database that provides information in a pictorial form in one area and finds some related information on a RDB in the other area. He finds out that the output from the image database can be used as the input for the RDB. In this situation, because two information resources do not provide any integrated interface for accessing their information, the mobile user must repeatedly access the information from the image database through a graphical interface and uses that result to access the RDB through a SQL command.

To solve this difficulty, in our previous research, we proposed a new information-retrieval architecture for mobile users based on a persistent script language to generate an integrated view among unknown information resources and access their information in remote programming style. In this paper, we describe an experiment and show how to use our approach to create an effective information-retrieval application for mobile users in the heterogeneous environment.

The paper is organized as follows. After introducing the concept of our previous research in Section 2, we describe our current experiment in Section 3. Finally, we conclude this paper.

2 System Overview

To help mobile users to integrate information in the heterogeneous environment, there are necessary functions as follows:

[Function 1] Information resources should be able to execute users' requirement sent from mobile users and integrate information from other information resources.

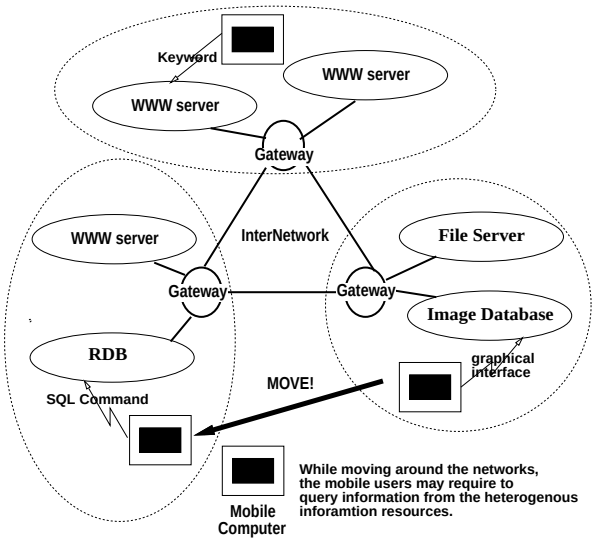

Figure 1: A General Heterogeneous Environment

[Function 2] Information resources should inform mobile users about their services.

[Function 3] Based on the information provided by information resources, mobile users should be able to describe their requirement and send them to be executed at the information resources.

In our research, we have proposed a script language extended with persistence ability, **Persistent Perl**, to implement the above functions. We use it as the communication language between the mobile computer and the information resources and integrate information from several information resources in a remote programming style.

Figure 2 shows our proposal processing. To integrate information from several information resources, we have a system and a set of tools as follow:

1. To accomplish the function 1, we provide a system called **wrapper**. It executes the request sent from the mobile user and integrates information from several information resources in object-relational style.

2. To accomplish the function 2, specification of services provided by the information resource must be defined in a Persistent Perl script. We propose an **interface definition**, which is a class definition in object-relational style to describe data schema, methods and additional information of the information resources.

3. To accomplish the function 3, we provide a tool called **mediator** on mobile computers to help the mobile users to describe their request in an OO-SQL command. The mediator then translates the OO-SQL command to a Persistent Perl script program and sends it to be executed at the information resources.

With the tools shown above, the processing shown in Figure 2 is performed in the following step:

[Step 1] While moving in the network, a mobile user

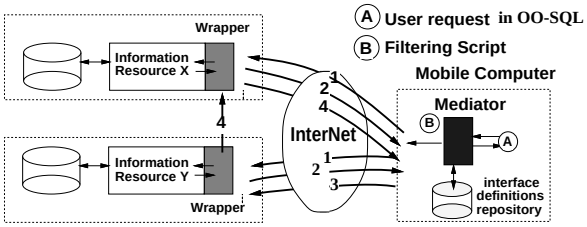

Figure 2: An Information-Retrieval Processing

may meet unknown information resources. Because he does not know data schema and how to access information in the information resources, he submits requests for interface definitions to each of information resources.

[Step 2] Corresponding to the mobile user's request, the wrapper on each information resource sends the **interface definition** that describes services to the mobile user by e-mail or HTTP protocol.

[Step 3] The mobile user registers the interface definitions sent from the information resources into repository in his mobile computer. Then interacting with the mediator on his mobile computer, the mobile user generates a script program and send it to be executed at the information resources.

[Step 4] The wrapper function in each information resource converts the commands in filtering script into the native commands, executes, gathers result. After gathering the result, the wrapper then sends the script to the next information resource or sends the result of the execution back to the mobile user.

In the next section, we will describe a practical experiment using the above processing steps.

3 An Experiment

The following is the experiment that can profit from our approach we discussed above. This experiment is that a mobile user accessing information resources in our university.

3.1 A global view architecture

Figure 3 shows the architecture of a global view mobile-heterogeneous environment in our university, the university of Electro-communications. It consists of some fixed hosts equipped with a wireless interface and communicate with mobile users in a coverage area called cell. A mobile user moving around may receive information from these fixed hosts involving their services. Certainly, the e-mail-address or URL address that the mobile user can request interface definitions can also be propagated through this channel.

Suppose that a mobile user enters into our university and wants to visit laboratories during open house. He reaches to the main gate of the university and receives

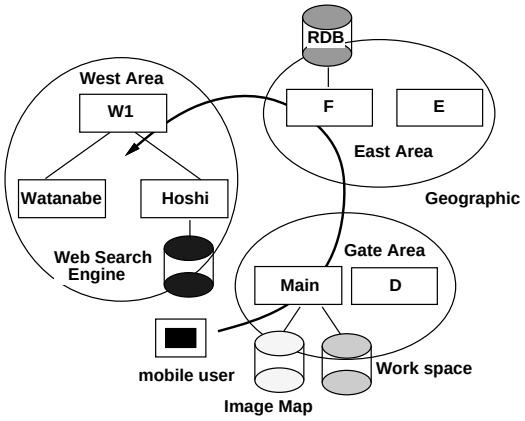

Figure 3: A Global View Architecture

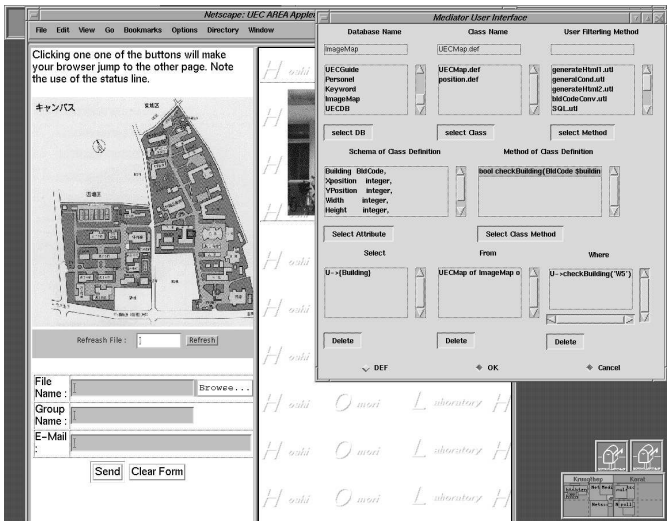

Figure 4: An Interface of System

the URL-address of a home page that contains information for each area in the university. This home page is shown in Figure 4. This home page allows the mobile user to select each section of map and it will show the detail of information of each section. Furthermore, the information resource that provides this home page allows the mobile user to send script to be executed and add the result of execution to each section of map. Therefore, the mobile user can display and accumulate the information that he found to this page while moving in the university.

3.2 Interface Definition

While moving across cells in the university, the mobile user may meet some information that is useful for him. Suppose that the mobile user finds out three information resources:

```
database UECDB
address CCC@DDD.is.uec.ac.jp
class LabBuilding inherit PGPersistent
body
  Building          Building_Code,
  Floor             integer,
  Lab               string,
method
public:
bool getByBuilding(Building_Code
                   $building);
bool getByFloor(integer $floor);
private
OID  new(Building_Code $building,
         integer $floor,string $lab);
void printDatail();
static setof OID getBuilding
                 (string $buildingCode);
setof OID  scanClass();
implement
... .... ...
... .... ...
interface
# Building_Code Rule
Rule:
Building_Code($X) :- chkCode($X).
chkCode($X) :- chkBldCode($X,$Y),
               output($Y).
chkCode($X) :- BuildingCode($X),
              output("'".$X."'").
BuildingCode('ComEd').
BuildingCode('W-1').
BuildingCode('W-2').
External:
  chkBldCode(in,out) code is chkBldCode.pl
Default:
    W-5
Comment:
  Please Enter Building Code as follows:
ComEd, W-X(X is number from 1 to 9), Main
endclass
```

Figure 5: An Interface Definition

1.The first is the server providing the home page described above at the main gate area. Its interface application is written by the Java language. It provides the information about its image map data and some methods for adding the information of execution into each section of the map.

2. The second is a Postgres relational database server that keeps information about laboratories on each building of the university. It is in the east side area of the university.

3. The third server is an application that provides a set of URLs of home pages in this university associated with an assigned keyword. The back-end server of this application is the Alta Vista search engine. It is in the west side area of the university.

The mobile user loads three interface definitions from the information resources and registers them into his mobile computer. One example of the interface definition shown in Figure 5 is the interface definition of a LabBuilding class in the Postgres database.

This interface definition specifies an interface for class named *LabBuilding*. It consists of three attributes: Building, Floor and Lab that are defined in **body** part. It defines methods in **method** part. The implementation of each method is written in **implementation** part. The **interface** part defines the interface rules that are used to solve the schema mismatch problem between domain of the different information resources. We also have a plan to extend its ability to provide a location -dependent access control function in this part. The detail of the interface rule is described in[2].

```
select M->{Building},L->{Floor},
       L->{Lab},La->{URLList}'
from   M in UECMap, L in LabBuilding,
       La in URLList,
source UECMap of ImageMap on
       AAA@BBB.is.uec.ac.jp,
       LabBuilding of UECDB on
       CCC@DDD.is.uec.ac.jp,
       URLList of WWWDB on
       XXX@YYY.is.uec.ac.jp,
where  M->searchByBuilding('W3')
and    L->getByBuilding(
       &bldCodeConv(M->{Building}))
and    URLList->getOIDList(L->{Lab})
and    &right('add',U->{Building},
       'Building',U->{Building},
       'Floor',L->{Floor},'Lab',L->{Lab},
       'URLLINK',UR->{URLList});
```

Figure 6: An OO-SQL Command

3.3 Mediator and OO-SQL

Now, it is the step that the mobile user must generate a script program for executing at these remote resources. However, writing script program is a very difficult task for most users. To alleviate this problem, we provide the tool called *Mediator* to help mobile users to generate script program. To allow mobile users to specify their requirements in declarative style, we propose an Object-Oriented SQL (OO-SQL) for our system.

The mobile user interacts with the mediator through the user interface and describes his requirement in the OO-SQL command. The user interface is shown in Figure 4. After the mobile user finish his OO-SQL command, the mediator will generate a script program from the OO-SQL. The example of the OO-SQL command that queries information from the three information resources described earlier is shown in Figure 6.

This query executes as the following step: First, it finds the building name from the map Image database. Second, it passes the building name to find the information of laboratories in the Postgres database. Third, it uses the laboratory names from the Postgres database as input keywords to find the list of URLs in WWW search engine. Last, Using *right* subroutine, it changes the result into HTML file and save it into the work space. The mobile user can link that result into the clickable map interface.

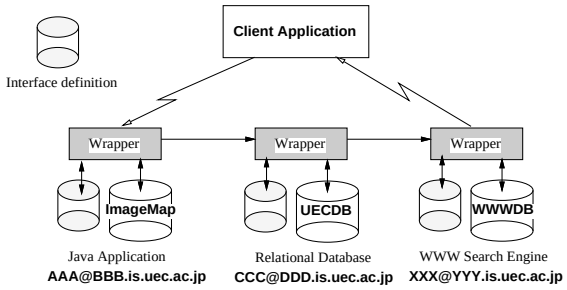

Figure 7: A Processing in the Heterogenous Environment

The mediator on the mobile computer translates this command into a script program. This script program includes the source of *bldCodeConv* subroutine provided by the mobile user that uses to solve a schema mismatch between different information resources [2].

3.4 Wrapper

After generating the script program, the mobile user send it from browser to the information resource. The script will be executed at the remote information resource by the system called *wrapper*. Typically, the wrapper converts an OO-SQL command to its native application command, executes and integrates with the result from the previous information resources. The execution for integration in this experiment is shown in Figure 7. The result page of execution is shown in Figure 8.

In this experiment, we have implemented wrappers for a Java-based image database and a Postgres relational database. On the contrary, the information resource that provides unstructured information, such as a WWW server, is difficult to be wrapped by object-relational model. To alleviate this problem, rather than implementing the wrapper for the WWW server directly, we have decided to implement a wrapper on a WWW search engine.

4 Related Works and Discussion

Integration of heterogeneous information resources has become an important research direction. There are mamy researches that provide solutions to solve this problem. The TSIMMIS project provides pre-defined integrated views through the module called *mediator*. Users can access information from many heterogeneous information resources via SQL-like query command in the client/server style. However, this system must have someone to declare integrated views for users.

The goal of our works is to provide tools for mobile users to integrate heterogenous information reosurces. Our approach tend to extend mobile users' ability so

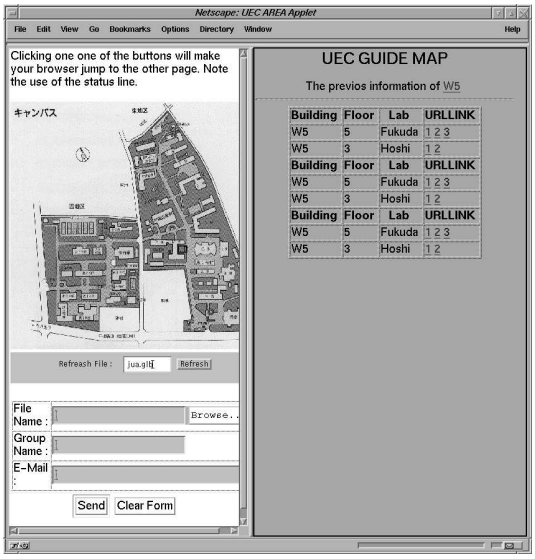

Figure 8: An Execution Result Page

that they can generate the integrated views over heterogenous information resources. The interface defintions of information resources are components for informing their services while the mediator on mobile computer is used as inter-component communicator for mobile user to generate integrated view. The operation of view generation is performed on mobile computer place. Therefore, it is suitable to disconnected state of mobile computing environment.

5 Project Status

In this paper, we have shown the information-retrieval architecture through the experiment. This experiment consists of three information resources currently running on a set of Sun UNIX workstations and a notebook computer running on Linux OS.

This experiment has shown how to use our approach to integrate the information from several heterogeneous information resources. We have also shown how to represent the result in extensible and clickable map. Currently, we plan to extend the ability of this extensible and clickable map as a user interface for mobile users shown in Figure 9.

References

[1] S. Chawathe et al., The tsimmis project: Integration of heterogenous information sources. In *Proceedings of the 100th IPSJ Tech. Rep. DBS*, pages 7–18, Tokyo, Japan, October 1994.

[2] S.T. Wisut, et al., An information-retrieval architecture for mobile computers based on a persistent script language. *IPSJ Tech. Rep. DBS*, Vol.96(109-45), 1996.

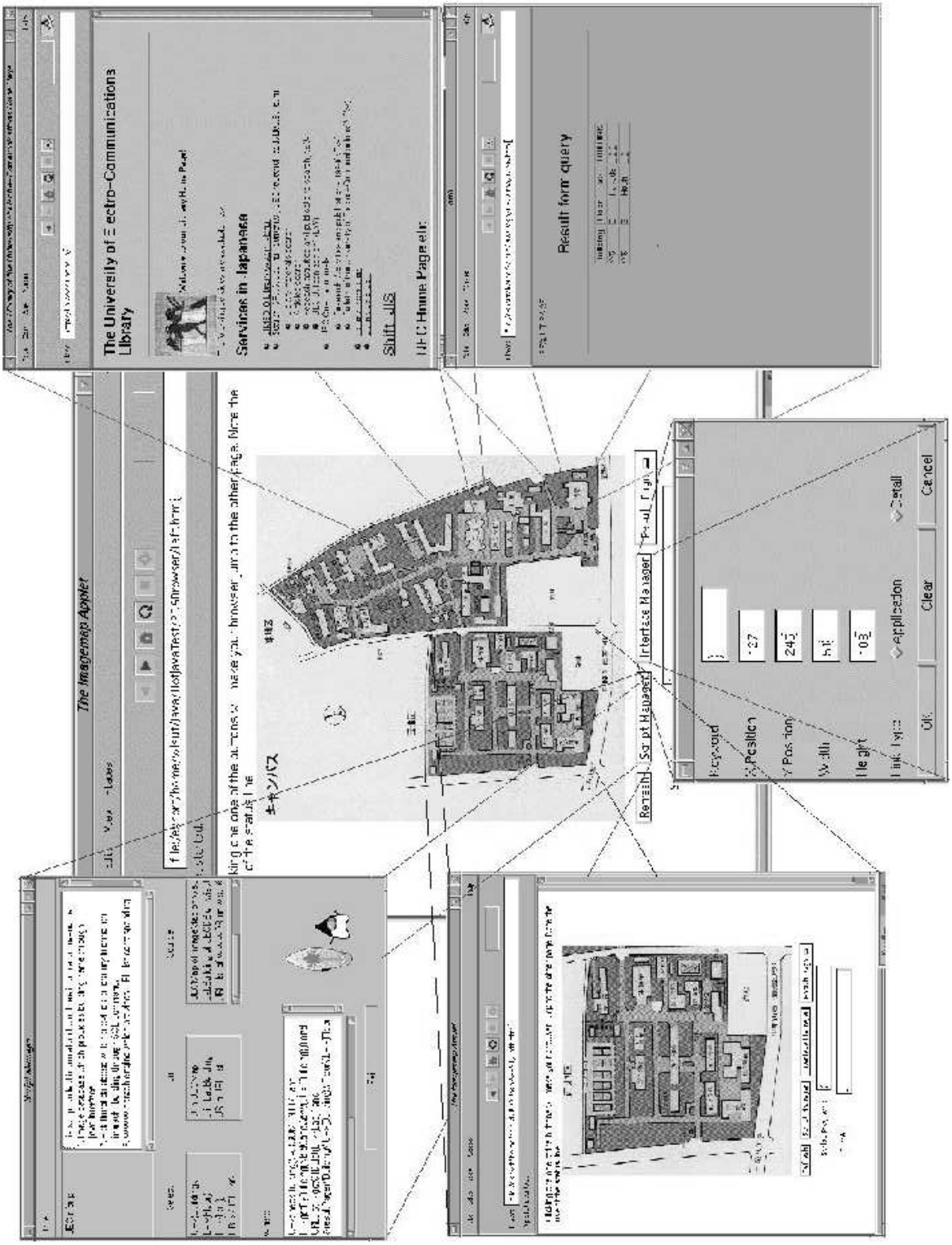

Figure 9: A User Interface