Application-level Solution for Data integration in Ubiquitous Computing Environment

Linhua Zhou, Yongping Zhang, Bo Guan
School of Electronic and Information
Ningbo University of Technology
Ningbo, China 315016
zhoulinhua@zju.edu.cn, {ypz, Guanbo}@nbut.cn

Journal of Digital Information Management

ABSTRACT: *The Ubiquitous Computing envisions a world of communicating mobile devices and sensors. Advances in wireless communications raise users' expectations about the accessibility of information in Ubiquitous Computing environments. Getting meaningful information from disparate sensor nodes and/or mobile devices spread in Ubiquitous Computing environment is an urgent task. On the one hand, the heterogeneity of the sensor nodes makes share and finding data at the mobile devices and sink nodes much harder. On the other hand, nodes in Ubiquitous Computing environment are highly dynamic which means query planning in Ubiquitous Computing environment should be able to dynamically re-make their plan when the network conditions change due to intermittent connectivity.*

To address these two issues, this paper proposes a new service-oriented framework(called UCDS, standing for Ubiquitous Computing Data Services) at application-level, which supports for semantic integration and querying of heterogeneous data sets distributed over the Ubiquitous Computing environment. We propose a RDF-View-based approach for mobile data mediation, and describe the service-composition-based query planning algorithm implemented in UCDS. We explore an ontology based approach to, mapping mobile data to data services (DS for short), publishing DS with the shared domain ontology. We also explore dynamically evolving service interface enabled by richly service description capability using the semantic web languages, and answering queries through DS composition. Experimental results show that the approach adopted by UCDS provides a better performance compared with the conventional approach.

Categories and Subject Descriptors
C.1.4 [Parallel Architectures]: Mobile processors; **H.2.4** [Systems]: Query Processing

General Terms
Ubiquitous Computing, Service Computing, Semantic web

Keywords: Ubiquitous Computing Data Services, Data integration, Query processing, Service composition

Received: 11 June 2011, Revised 28 August 2011, Accepted 5 September 2011

1. Introduction

Recent technological trends in electronics have resulted in a change in people's lifestyle, whereby mobile devices such as mobile phones, PDA's, GPS devices, etc. and sensor devices such as temperature, humidity, location, etc. have become an integral part of everyday life. This trend, together with the advancement in wireless communication technology, has raised users' expectations about the accessibility of services in Ubiquitous Computing environments [16] [19] [22]. This has raised challenges for service representation, service discovery and query answer in a dynamic environment, where services encapsulates mobile data and/or sensor data(call them Ubiquitous data). A Ubiquitous Computing environment, like the one that Ambient Intelligence [7] promotes, is a technologically augmented multi-sensor environment capable of capturing the context and activities of the users in order to respond to their needs. One main objective of such an environment is to provide ambient data services to users based on their needs in different context. However, how to encapsulate these mobile data into data services and how to provide these services to the user given the concrete application remain challenging issues.

The Semantic Web [1] aims to provide a common semantic framework allowing data to be shared and reused across application and community boundaries. It is based on the RDF(Resource Description Framework) framework[1] and OWL(Web Ontology Language) language[2], which is proposed and standardized by W3C organization to represent web information in a flexible, but meaningful way so that data can be exchanged and integrated without loss of semantics. However, Most of existing data in a ubiquitous computing environment is locally stored in mobile devices or sink nodes. Therefore, for semantic web to be really useful and successful in a ubiquitous computing environment, great efforts are required to offer methods to support integration and share of heterogeneous Ubiquitous data sets using ontology model[11]. A plurality of solutions have thus far been prescribed to ease the burdensome of the issue [17] [9] [13]. Chief among them is the semantic web service [6] [15] technologies which

[1] Resource Description Framework (RDF):http://www.w3.org/RDF/
[2] OWL Web Ontology Language Overview:http://www.w3.org/TR/owl-features/

have been adopted in improving the availability of and interoperability between heterogeneous networked resources.

The converge of the Ubiquitous Computing and the Semantic Web service arise three main issues in this respect which are, 1) how to encapsulate these data resources produced from mobile devices? and 2) based on the encapsulation, how to retrieve clients request? and 3) how to adapt clients query in dynamically environments? In this paper, we address the above three issues from a application-level perspective. For the first issue, We propose a data service to encapsulate one or more possibly distributed and heterogeneous Ubiquitous data sets produced by mobile devices or sensor devices. To deal with heterogeneity of data schemas, we suggest to describe Ubiquitous data sets as views over a shared mediated schema [21]. Specifically, the local schema of each Ubiquitous data source is mapped to concepts of a shared OWL ontology, and its terms are used to define Views [4] [12] which describe the Ubiquitous data sets in the mediated schema. We use the DS Profile to describe content and capabilities of the DS which encapsulate the Ubiquitous data sources. For the second issue, we propose a DS Composition-based query processing approach. By a query-rewriting based approach to composing data services that take advantage of ontology language and efficiency of query-rewriting-based planning, we can effectively achieve clients query request. As for the third issue, we adopt a dynamic evolvable approach to producing and publishing data service that make the data service more adaptable to client quest. The approach we proposed distinguishes itself by its capability of dynamically evolving service interface enabled by describing service capability using the semantic web languages.

The paper is organized as follows. In section 2, we comment on some related work. Section 3 talks about the system architecture and technical features. Section 4 describes the rules for mapping between data sources and DS, and the modeling of DS based on these mapping rules. Section 5 presents the key query plan algorithm and the dynamic generation of evolvable WSDL/SOAP interface for query answering in UCDS. Section 6 show initial results of the evaluation of the system by comparing with the DS that adopt traditional approach. Section 7 gives the summary and our future directions.

2. Related Work

Although research on integration and query processing in Ubiquitous Computing environments at the application-level is still in the very early days, integration and query of heterogeneous Ubiquitous data sets are important issues for Ubiquitous computing applications. Our work presented in this paper is related to existing research on representation, modeling and query of Ubiquitous data sets. We therefore comment on some of these in the following.

1. Distributed Database Community. The concept adopting ontology-based mediated schema to encapsulate and query heterogeneous data was addressed by the web community to solve the integration of heterogeneous data sources [10] [18] [4]. However, the very nature of the Ubiquitous Computing environment opens the door to new challenges that limit the applicability of current technologies in this area. For example, resource limitations on the target devices have called for novel solutions to enable efficient ontology-based semantic querying of Ubiquitous data sets.

2. Sensor Network Community. A number of efforts have been made to define and implement a High Level Language for managing data in WSN applications [23] [14] [12]. [23] [14] is a in-network query processing system. [23] shows how to optimize a large number of queries across multiple base stations attached to the same sensor network, exploiting similarities among queries allocated to the same base station during in-network processing. But, both [23] and [14] focus on energy efficient and scale well with the size of the sensor nodes. They do not adopt the semantic web technologies, because it will effect the performance of system and may not necessary for in-network query processing in WSN applications. In large scale WSN applications, however, communication overheads between different small WSN is considerable, so adopting semantic web technology can effectively management data of WSN. Varying from traditional integration of WSN data, this paper address the data query at application-level. [12] is similar to our work in the solution of the semantic knowledge, which is adopted to reduce network traffic overheads, improve scalability and extensibility of wireless networks. But it does not detail the mapping process between the mediated view and the data of WSN, and does not employ web service to integrate data sets of WSN.

3. Ubiquitous computing Community. There are a number of the most recent Ubiquitous computing systems that use ontology models, include [2] [5] [20], and [8]. [2] is the key requirement for modeling context in the smart meeting application. It defines typical concepts and relations for describing physical locations, time, people, software agents, mobile devices, and meeting events. By exploiting the features of physical spaces, [5] uses peoples location and movement as a source of task-level context and as a guide to provide appropriate information or services. [8] proposes an ontology-based context model, in which a hierarchical approach is adopted for designing context ontologies. These ontologies include a common upper ontology for the general concepts in Ubiquitous computing (such as people, location, and activity) and domain-specific ontologies that apply to different sub-domains (like smart cars). All of them try to build specific domain ontologies for Ubiquitous computing. But few of them Consider the semantic modeling and semantic querying of data sources in Ubiquitous Computing environment.

3. System Architecture and Technical Features

3.1 System Architecture

The system architecture is shown in Figure 1, there are

four key components in the core of UCDS.

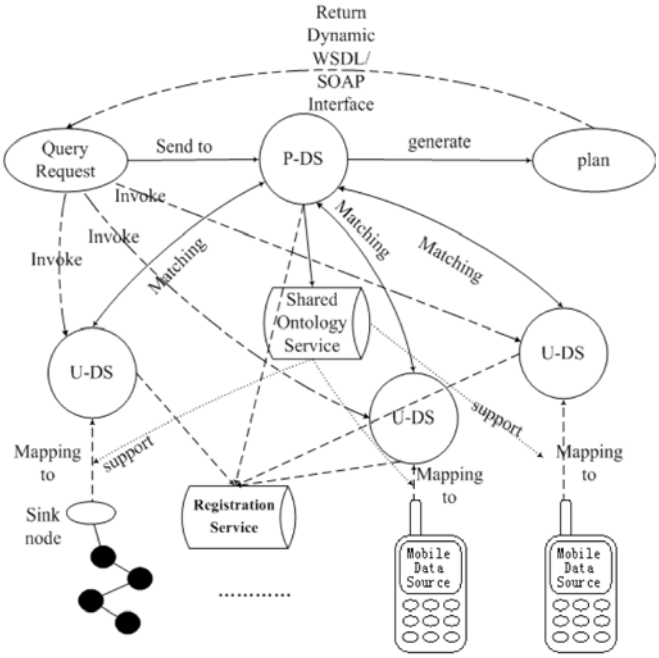

Figure 1. System Architecture and Core Component

1. Planning Data Service (P-DS) is used to process semantic queries and generate query Plan. Firstly, it gets DS information from semantic registration service. Then it find the appropriate DS according to the algorithm we proposed. Afterward, it generate query plan and return a DS which has dynamic WSDL/SOAP interface to user. Finally, the user invoke the DS and get the semantically-enriched data information.

2. Ubiquitous Data Service (U-DS) encapsulates Ubiquitous data sets and register to Registration Service. Its WSDL/SOAP interface is generated by the P-DS before invoked by user. It is described by the extended OWL-S specifications. Detail information can be found in Section 4.

3. Shared Ontology Service is used to expose shared ontologies that are defined using web ontology languages. Typically, the ontology is specified by a domain expert who is also in charge of the publishing, revision, extension of the ontology.

4. Semantic Registration Service maintains the semantic mapping information. Typically, data providers define the mappings from Ubiquitous data to domain ontology, and submit the mapping information and DS registration information to this service.

3.2 Technical Features

The following three features that distinguish this application from other similar semantic data integration tools, which will be introduced in detail in Section 5.

1. Shared Ontology-Based Mapping Rules. We propose a language based on OWL-S [15] for mediating between Ubiquitous data sets and DS that allows mapping simple forms of domain structure and Ubiquitous data schemas. We also show that this language can map between RDF data and structure Ubiquitous data.

2. DS-composition-based query answering. A DS composition based approach to query answering that take advantage of ontology language and efficiency of services composition-based planning. A service composition based query rewriting algorithm is implemented to translate semantic queries into calling a group of DSs which conduct the query request. This algorithm extends and combines relational database and web service techniques for translating queries using views, with consideration of the features of web ontology languages. Otherwise, this algorithm can also be enriched by additional inference capabilities on predicates such as subClassOf and subPropertyOf.

3. Evolvable WSDL/SOAP Interface. A evolvable approach to publishing data service that make the data service more adaptable to client quest. It is automatically generated at runtime according to property definitions of ontology, and

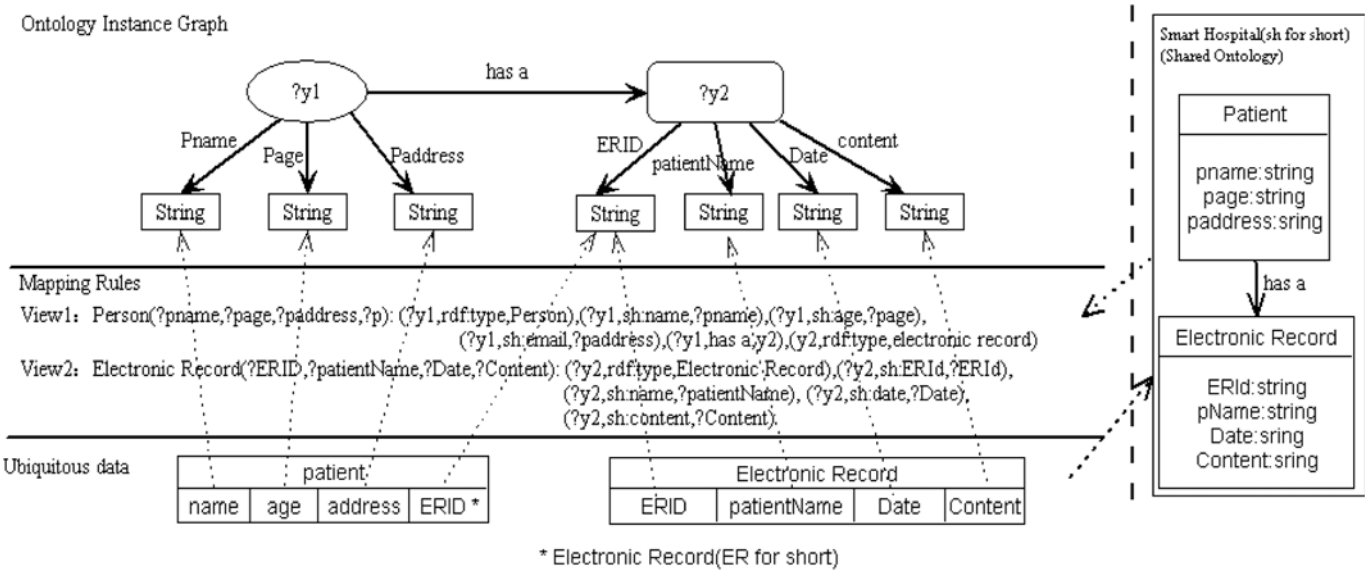

Figure 2. Mapping Rules and Semantic Graph Generated by Mapping Tool

will finally generate a DS with WSDL/SOAP interface.

4. Semantic Mediation

In UCDS, Data Service is employed to encapsulate and expose Ubiquitous data produced in Ubiquitous Computing environment. Therefore, there are several issues to be resolved. Section 4.1 introducs Mapping between mobile Data Sources and RDF-view, and Section 4.2 elaborates on the modeling of DS.

4.1 Mapping Ubiquitous Data Sets

Although it is obvious that the introduced framework can be adapted to any other data source (such as sensor data), we only consider mobile data in this paper as it is the most popular data source used in Ubiquitous Computing environment. In our system, mobile data are mediated and related by a shared ontology, and each structural mobile data is mapped into one or more classes. For example, the mapping scenario in Fig. 2 illustrates the mapping process between two structural mobile data sources (patient and Electronic Record), and a shared ontology(Smart Hospital).

Mappings are typically defined as views in conventional data integration systems in the form of GAV (global-as-view), LAV (local-as-view) [10]. Considering the case in this paper, GAV is to define each class or property as a view over data sources, and LAV is to define each data source as a view (or query) over the mediated schema. The experiences from conventional data integration systems tell us that LAV provides greater extensibility than GAV: the addition of new data sources is less likely to require a change to the mediated schema [10],[12]. Therefore, the LAV approach is employed in our system, that is, each mobile data source is defined as a view over the ontologies. We call such kind of views as Semantic View.

The mid part of Figure 2 show cases how to represent the mappings as semantic views in a Datalog-like syntax. Like in conventional data integration, a typical semantic view consists of two parts. The left part is called the view head, and is a relational predicate. The right part is called the view body, and is a set of RDF triples. There are two kinds of variables in the view definitions. Those variables such as ?pname, ?page, ?paddress, ?ERId, ?content are called distinguished variables, which will be assigned by an data or instance values from the data source. Those variables such as ?y1, ?y2 are called existential variables.

In general, the view body can be viewed as a query over the ontology, and it defines the semantics of the relational predicate from the perspective of the ontology. The meaning of semantic view would be more clear if we construct a Target Instance based on the semantic mapping specified by these views. For example, given a structural tuple as below, applying the View1 and View2 in Figure 2 on this tuple will yield a set of RDF triples.

Structural Tuple:
zju: emp(“Lihua Zhu”,”32",”zheda Road 38#, hangzhou, zhe jiang, china”, “ZhouLinhua@zju.edu.cn”, ”ZJU Hospital”,”10521038", “2008-11-11”),”catch a cold, drugs are Relenza”;
Yielded RDF triples by Applying View1, View2:

```
_:bn1 rdf:type sh:Patient;
      sh:pname “Lihua Zhu”;
      sh:page “32”;
      sh:paddress “zheda Road 38#, hangzhou, zhe jiang,
      china”;
      sh:has a _:bn2.
_:bn2 rdf:type sh:ER;
      sh:ERId “10521038”;
      sh:Date “2008-11-11”;
      sh:Content “catch a cold, drugs are Relenza”.
```

The mapping rules can be generated between Ubiquitous data sets and the shared ontology through Dartmapping tool [3] by a domain expert or individuals who want to publish their data. More detailed Foudermental aspects about mapping tool could be found in another paper [4].

4.2 Modeling of DS

After mapping the Ubiquitous data to RDF-View, The remaining work is how to expose the DS(it encapsulate Ubiquitous data sets) to the application-level users who want to use these data. Therefore, in this section we introduce the modeling of DS. We define a DS as having three major components: the service input I, the service output O, and the content description CD(I, O, CD). CD offer the base for semantic description of the content and capability of DS. Content Description of DS. The data content refers to the underlying data model and schema used in the Ubiquitous data that can be in relational form or structural form. Although it is obvious that the introduced framework can be adapted to any other legacy proprietary format, we only consider structural data in this paper as it is the most popular data model used as yet.

The data sets on the Ubiquitous Computing environment normally has its own schema definition and thus is heterogenous. As mentioned before, exposing them onto the Ubiquitous Computing environment requires mapping these format to a shared semantic web ontology, making the data semantics understandable to unexpected data consumers.

There are many semantic web service proposals and some of them such as the OWLS[3] and the WSMO[4] has well-defined model, it is natural to develop our model upon currently available approaches. We choose OWL-S as our development foundation.

To be in accord with the OWL-S specification, we encode the mapping information in process:inCondition property of process:Result, since in OWL-S specification, the inCondition property specifies the condition under which the result occurs. As a matter of fact, the mapping information can be considered as a kind of constraint or condition that must be satisfied for user’s query requests.

[3] OWL-S: http://www.w3.org/Submission/OWL-S/
[4] WSMO: http://www.w3.org/Submission/WSMO/

Example 1. (Content Description Example in OWL-S)

```
<process:AtomicProcess rdf:ID="DS1">
  <process:hasInput> ... </process:hasInput>
  <process:hasOutput> ... </process:hasOutput>
    <process:hasResult>
      <process:Result>
        <process:hasResultVar>
            <process:ResultVar rdf:ID="?pname" />
        </process:hasResultVar>
        <process:hasResultVar>
            <process:ResultVar rdf:ID="?y1">
            <process:parameterTyperdf:datatype="&xsd;#string">
            &DS1;#?y1
            </process:parameterType>
        </process:ResultVar>
      </process:Result>
      .....
        <process:inCondition>
          <expr:SPARQL-Condition>
            <expr:expressionBody>
            </expr:expressionBody>
          <expr:SPARQL-Condition>
        </process:inCondition>
       ...
        <process:Result>
      </process:hasResult>
</process:AtomicProcess>
```

Although some rule language such as SWRL and RIF can be used, but by present time, no standard rule language has been recommended. Taking a simple example, we describe a DS for db1 as below. The process:hasResultVar part defines the variables that are used in the mapping description.

After modeling the DS, these DS must be registered to the registration service, so the P-DS can find the appropriate DS from the registration service according to the user's query request, and generate query plan and evolvable WSDL/SOAP interface.

5. Query Planning

Given a set of DS described with OWL-S based on mapping rules, our goal is to be able to answer queries issued by any user, making use of all relevant (mapped) DS. This section describes UCDS's functionality of P DS, which performs the following task: generating query Plan and generating evolvable WSDL/SOAP Interface.

5.1 Query Planning Algorithm

The rich description of DS provides the foundation for implementing more selective algorithm for query planning. Our algorithm mainly draws upon the query rewriting approach introduced in traditional database community [10]. The chief goal of the algorithm is to find out an executable query plan that combines a sequence of data service based on the content description and input/output requirement of the DS to achieve the query request. In general, the algorithm has two steps. In the first, we find out candidate DSs which can meet the user's query request, and in the second we try to order services to ensure that they are executable.

The first step of the algorithm is described in details in Algorithm 1. It takes a semantic query(for exampleSPARQL) and a set of service description as input. Firstly, it splits the triples in the query body into a set of tripe groups based on the subject variables used in them.

Definition 1 (Triple Group) A triple group is a set of triples that have the same subject. It represents a fragment of the query body that can later be matched and combined together, thus making the query rewriting to be easier to implement.

In the algorithm, the triple group is called a query chunk. Next, for each query chunk, we find out all candidate DSs and form a candidate set. The criterion used for the determination is based on whether or not the query chunk is contained or subsumed by the content description of a DS.

Algorithm 1 Candidate DS selecting Algorithm
Require: a set of service descriptions S = {ds1, ds2, ..., dsn}, a semantic query statement Q.
1: Split Q into k chunks; C = {q1, q2, ..., qk} based on the variables used in Q;
2: Let SC be a empty ordered set.
3: for all qi 2 C do
4: Set sci = ;
5: for all dsj 2 S do
6: if qi _ dsj then
7: sci = sci [dsi
8: end if
9: if dsj is the last one in S then
10: Return "invalid query"
11: end if
12: end for
13: Add sci to SC.
14: end for
15: Output candidate set list SC

The result of Algorithm 1 is a list of candidate sets, each of which corresponds to a query chunk. A candidate plan is generated by selecting one candidate DS from each candidate DS sets, namely, apply a cartesian product operation over these candidate sets.

In the second step, we consider a candidate plan and try to order the candidate DSs in such a way that the plan will be executable, i.e., will adhere to the input/output requirements of DSs. Algorithm 2 describes a process that given a candidate plan, finds a ordering on them, if such an ordering exists. It proceeds by maintaining a list of available parameters, and at every point adds to the ordering any new DSs whose input requirements are satisfied. Finally, the algorithm outputs a ordered executable DS sequence.

Algorithm 2 Create executable plan algorithm
Require: a SPARQL query statement Q = {q1, q2, ..., qk} and a

corresponding candidate
service set DS = {ds1, ds2, ..., dsk}, .
1: Let B be the set of variables bound by values in the query, which can be viewed as the input requirement of the query.
2: Let bi be the set of variables bound by values either from the original query or from output of some data service at step i.
3: Let O be the set of variables occurring in the selection part of the query, which can be viewed as the overall output requirement of the query.
4: Let (Ini,Oui) be the input/output requirement of a data service dsi.
5: Let S be an empty set.
6: for all i=1,...,n do
7: Choose a candidate service dsj 2 DS that was not chosen earlier and the parameters in Inj is in b(i -1).
8: if There is no such a data service then
9: Return "plan not executable".
10: else
11: bi = bi"1*"Ouj
12: end if
13: Add dsj to S
14: end for
15: Output a service sequence S.

5.2 Generating Evolvable WSDL/SOAP Interface

As mentioned before,since the type of client request and the number of possible combinations increases exponentially against the schema complexity, flexibility and evolvability of DS interface are both valuable and necessary. The UCDS system implements the feature. The basic idea is to utilize the ontology used in both the content description and query request to dynamically generate a new WSDL/SOAP interface. As a matter of fact, an ontological query can be equally translated into a WSDL specification, as the selection part can be viewed as the output and the variables bound by values in the condition part can be viewed as the input, while the type information is self-described in the ontological classes and properties. To illustrate the approach proposed by this paper, consider the following example:

Suppose there is a semantic query request, how does UCDS P-DS service dynamically generate WSDL/SOAP interface. For example, user issue a semantic query (for example SPARQL) to find the address of a patient named "Lihua Zhu".

Q1: SELECT ?pname ?paddress where
?y1 rdf:type sh:Pateint.
?y1 sh:name "Lihua Zhu".
?y1 sh:ERId ?ERId.

Example 2 (Evolvable WSDL/SOAP Iterface Generated by P-DS)

```
.....
<message name="inputParameter">
    <part name="pname" type="xs:string"/>
</message>
<message name="outputParameter">
    <part name="paddress" type="xs:string"/>
</message>
.....
<portType name="DynamicQuery">
    <operation name="query">
    <input message="inputParameter"/>
        <output message="outputParameter"/>
    </operation>
</portType>
.....
```

According to Algorithm 3, UCDS can dynamically generate an WSDL/SOAP interface. In the generated WSDL/SOAP interface, the other content is the same as what the WSDL/SOAP interface should be, except the contents that relate to the operator. The content of the WSDL/SOAP interface generated by the Algotithm 3, against the Q1, list in Example 2. Where system has been automatically assigned the value("Lihua Zhu") of pname, and if need, system can also show the result to user.

More complex case may occur when several data sources needs to be combined to fulfill the query request. The P-DS service has one more thing to do, such as determining what kinds of DS can be combined together to fulfill the query request, other than merely selecting out candidate DSs. Apply Candidate DS selecting Algorithm, system find the candidate DS's set, and then apply Create executable plan algorithm find an executable DS sequence. Afterward system can dynamically generate the WSDL/SOAP interface and return the results to user.

Algorithm 3 Generate WSDL/SOAP Interface
Require: a DS template T, input set created by P-DS PI, output set created by P-DS PO, modification set created by P-DS PM(such as invoking address info).
1: copy the content of DS template to w
2: generating the content of the inputParameter part according to PI
3: generating the content of the outputParameter part according to PO
4: generating the content of the query part using the content of inputParameter and the content of outputParameter
5: modify the content of binding and servicepart according to PM
6: return w

6. Evaluation

One goal of our experiment is to validate that our system can scale up to deal with query answering of large-scale Ubiquitous data sets encapsulated through DS, the other goal our experiment is to validate our system's performance by comparing with the system adopting traditional approach (fixed WSDL/SOAP interface). Since there are no standard Ubiquitous service sets for evaluation of Ubiquitous service composition, we randomly generate sets of DS to evaluate our approach. All experiments are performed on three PCs with 1024MB RAM, JRE 1.5.0.

we use one pc to accommodate DSs which encapsulate Ubiquitous data sets, one pc to accommodate shared ontology service and semantic registration service, the remaining pc to run P-DS.

6.1 Scalability Analysis

In this experiment, we compute handling time against the increasing number of the DSs which encapsulate Ubiquitous data sets. The handling time include the overhead that P-DS service has expended(the Query Planning time and WSDL interface generating time) and the communications overhead between P-DS and DSs. The experiments are carried on the ten groups of test services randomly generated by machine. Figure 3 shows that the handling time changes with the increase of Number of DS in the registry. From the experiment results, we can see that the handling time of our system, even for total 1000 services, is less than 9 seconds, which is an acceptable performance.

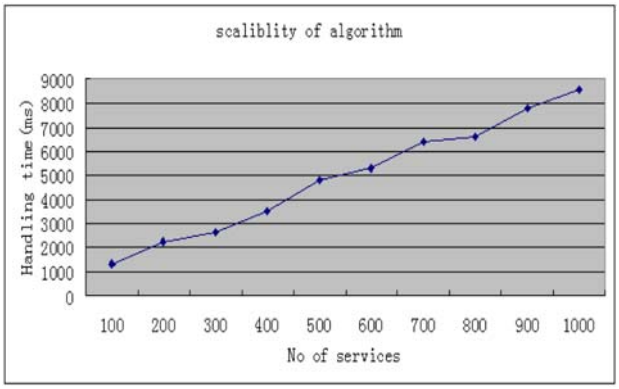

Figure 3. Scalability Analysis

6.2 Performance Analysis

In this experiment, we compare query handling time between fixed WSDL interface mechanism(traditional approach) and dynamic generating WSDL interface mechanism, and number of satisfaction of query request between fixed WSDL interface mechanism and dynamic generating WSDL interface mechanism, respectively. The goal of this experiment is to validate the performance of UCDS system.

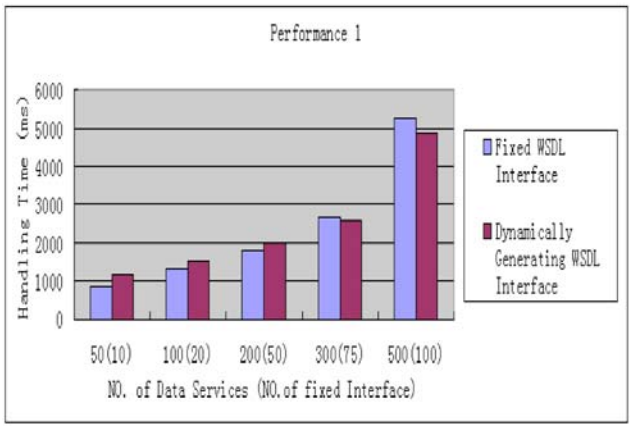

Figure 4. Comparing of Handling Time

First, we see about how the handling time changes with increasing the number of data services against fixed WSDL interface mechanism and dynamic WSDL interface mechanism, respectively.

Figure 4 illustrates how the response time changes with the increase of the number of data services. The value in bracket mean the number of fixed WSDL interface, while the value outside bracket denote the number of data service. From the experiment results, we can see that: the handling time of fixed WSDL interface mechanism is less than that of dynamic WSDL interface mechanism, when the number of data service less than 300 and fixed WSDL interface less than 75. That means our approach is much more fit for occasion which require more interaction between user and system.

It may be a little surprised that the handling time of fixed WSDL interface mechanism is higher at a large number of fixed WSDL interface and is lower at a relatively small number of fixed WSDL interface. According to our analysis, the difference come from the different handling overhead between two mechanism. Fixed WSDL interface mechanism spend more it's handling time at searching WSDL interface, while dynamically generating WSDL interface mechanism spend more it's handling time at query planning and generating WSDL interface. So when the number of data service and the number of fixed WSDL interface is large enough, the performance of the namically generating WSDL interface mechanism is more excellent than that of fixed SDL interface mechanism.

Second, beside concerning the handling time, user also care the number of satisfaction of query request issued by him. So we will see about how the number of satisfaction of query request changes with the increase of services in the registry. We use the five groups of services as the test data. The number of services ranges from 50 to 500.

Figure 5 illustrates how the number of satisfaction of query request changes with the increase of services in the registry. The value in bracket is the total number of query request issued by user, while the value outside bracket denote the number of data service. In our experiment, to simulate the real query request (often have higher number of satisfaction of query request), half of query request is generated from the fixed WSDL interface, and the other half of query request is generated randomly. From the

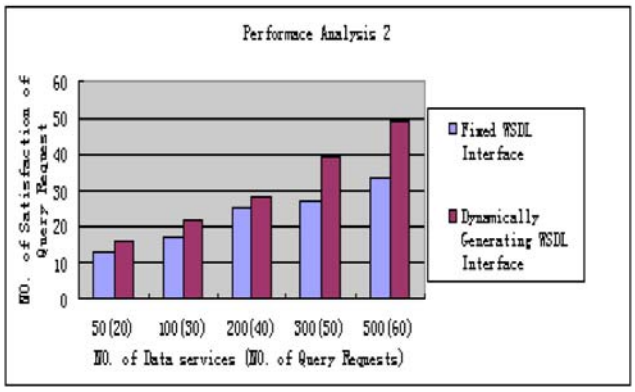

Figure 5. Comparing of Number of Satisfaction of Query Request

experiment results, we can see that: the number of satisfaction of query request of dynamic WSDL interface mechanism is higher than that of fixed WSDL interface mechanism.

From the two experiments, we can conclude that it is a promising way to improve the efficiency of query answering based on data service composition by using the dynamic WSDL interface. The performance of the dynamic WSDL interface is especially excellent comparing with the fixed WSDL interface mechanism in Ubiquitous Computing environment.

7. Conclusion

In this paper, we have discussed some aspects of the relationship between the semantic knowledge with respect to data integration strategies and answering queries in the Ubiquitous Computing environment. We have presented an ontology-based approach to encapsulate ubiquitous data sets into DSs and achieve user's semantic query request by the DS composition approach. In summary, the main contribution is four fold.

1. We propose a shared ontology approach for mediating between ubiquitous data sets and DS that allows mapping simple forms of domain structure and ubiquitous data structure.

2. We propose a data service model that extends the description capability of OWL-S to address the requirement for exposing rich data semantics of Ubiquitous Computing applications.

3. To solve the dynamic characteristics of ubiquitous computing environment, we propose a evolvable approach to publishing data service that make the data service more adaptable to client quest request.

4. We propose a service composition-based approach to answer user's query request.

The purpose of the project from which this research stems is to build a complete mediation mechanism for Ubiquitous Computing applications. Our plans for future work include integrating seamlessly the process of answering queries with the shared semantic knowledge about ubiquitous computing applications as a basis for an effective and efficient integration strategy. Our long-term goal is to develop a well-founded system to support data query and search in the Ubiquitous Computing environment.

8. Acknowledgment

This work is supported in part by Zhejiang Mobile Network Application Key Lab program (NO. MNATKL2011003), China NSF program (NO.60972163), Zhejiang NSF program (No. Y1100598) and Ningbo NSF program (No. 2011A610175).

References

[1] Berners-Lee, T., Hendler, J. A., Lassila, O. (2001). The semantic web, *Scientific American*, 284 (5) 34–43.

[2] Chen, H., Finin, T., Joshi, A. (2003). An ontology for context-aware pervasive computing environments, *Knowl. Eng. Rev.*, 18 (3) 197–207.

[3] Chen, H., Wu, Z., Mao, Y., Zheng, G.(2005). Dartgrid: a semantic infrastructure for building database grid applications. Concurrency and Computation: Practice and Experience, 18.1811-1828.

[4] Chen, H., Wu, Z., Wang, H., Mao, Y. (2006). Rdf/rdfs-based relational database integration. icde, 0:94.

[5] Dearle, A., Kirby, G., Morrison, R., Mccarthy, A., Mullen, K., Yang, Y., Connor, R., Welen, P., Wilson. (2003). Architectural support for global smart spaces. *In:* In Lecture Notes in Computer Science 2574, p. 153–164. Springer.

[6] Chintan Patel, Karthik Gomadam, et al (2010). TrialX: Using semantic technologies to match patients to relevant clinical trials based on their Personal Health Records. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*. 8 (4) ACM

[7] Nikolaos Georgantas, Valerie Issarny, et al. (2010). Middleware Architecture for Ambient Intelligence in the Networked Home. HANDBOOK OF AMBIENT INTELLIGENCE AND SMART ENVIRONMENTS, p. 1139-1169.

[8] Gu, T., Wang, X. H., Pung, H. K. and Zhang, D. Q. (2004). An ontology-based context model in intelligent environments. *In:* Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, p. 270–275.

[9] Gurgen, L., Roncancio, C., Labb´e, C., Bottaro, A., Olive, V. (2008). Sstreamware: a service oriented middleware for heterogeneous sensor data management. *In:* ICPS '08: Proceedings of the 5th international conference on Pervasive services, p. 121–130, New York, NY, USA, ACM.

[10] Halevy, A. (2001). Answering queries using views: A survey, *Journal of Very Large Database* 10 (4) 53–67.

[11] He, Y., Tully, A. (2008). Query processing for mobile wireless sensor networks: State-of-theart and research challenges. Wireless Pervasive Computing, ISWPC 2008. 3rd International Symposium on, p. 518–523, May.

[12] Ibrahim, I. K., Kronsteiner, R., Kotsis, G. (2005). A semantic solution for data integration in mixed sensor networks, *Computer Communications* 28 (13) 1564–1574.

[13] Kim, H., Hwang, J., Suh, B., Nah, Y., Mok, H.-S. (2008). Semi-automatic ontology construction for visual media web service, *In:* ICUIMC '08: Proceedings of the 2nd international conference on Ubiquitous information management and communication, p. 69–73, New York, NY, USA, ACM.

[14] Malhotra, B., Nascimento, M. A., Nikolaidis, I. (2008). Better tree - better fruits: using dominating set trees for max queries, *In:* DMSN '08: Proceedings of the 5th workshop on Data management for sensor networks, p. 1–7, New York, NY, USA, ACM.

[15] Martin, D., Burstein, M., Mcdermott, D., Mcilraith, S., Paolucci, M., Sycara, K., Mcguinness, D. L., Sirin, E., Srinivasan, N. (2007). Bringing semantics to web services with owl-s. World Wide Web, 10 (3) 243–277.

[16] Jian Zhu, Mohammad Oliya, Hung Keng Pung, Wai Choong Wong, (2010). SOLE: context-aware sharing of living experience in mobile environments, *In:* Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia (MoMM '10). p. 358-361.

[17] Nordin, N. A., Shin, W. H., Ghauth, K. I. B. Tamrin, M. I. B. M. (2007). Using servicebased content adaptation platform to enhance mobile user experience. *In:* Mobility '07: Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology, p. 552–557, New York, NY, USA, ACM.

[18] Rachel Pottinger, A. H. (2001). MiniCon: A scalable algorithm for answering queries using views. *VLDB Journal: Very Large Data Bases*, 10 (2–3)182–198.

[19] Cagalaban, Giovanni., Kim, Seoksoo (2010). Towards a Service-Oriented Architecture for Interactive Ubiquitous Entertainment Systems. ICEC 2010, *Lecture Notes in Computer Science* (*LNCS*), V. 6243, p. 409-421.

[20] Gilman, Ekaterina., Su, Xiang., Riekki, Jukka(2011). Towards Context Modelling and Reasoning in a Ubiquitous Campus. *In:* Proceeding of the 2011 conference on Information Modelling and Knowledge Bases. p. 278-287.

[21] Zhou, L., Chen, H., Tong, Y., Ma, J., Wu, Z. (2008). Ontology-based scientific data service composition: A query rewriting-based approach. *In:* 2008 AAAI Spring Symposium, p. 116–121.

[22] Fariba Sadri. (2011). Ambient intelligence: A survey, *Journal of ACM Computing Surveys* 43 (4) 36.

[23] Xiang, S., Lim, H.-B., Tan, K.-L.,Zhou, Y. (2007). Similarity-aware query allocation in sensor networks with multiple base stations, *In:* DMSN '07: Proceedings of the 4th workshop on Data management for sensor networks, p, 1–6, New York, NY, USA, ACM.