Performance Evaluation of CAMA for Agriculture Domain

JAYEETA CHANDA[1]
SANATNU CHANDA[2]

[1]B.P. Poddar Institute of Management and Technology
Kolkata, India
[2]IBM, Kolkata India
[1]jayeeta.chanda@gmail.com
[2]santanu.chanda@in.ibm.com

Abstract. Mobile devices are becoming more and more popular and accessible day by day. In this paper, we propose a context aware middleware architecture (CAMA) that is suitable for use in the agriculture domain. Farmers retrieve context aware information regarding agricultural domain from agriculture databases with the help of mobile devices that have a GUI that display information in vernacular. Our proposed architecture (CAMA) acts as an interface between farmer and database and correctly maps the retrieved data to its vernacular form and vice versa and also ensures uniform abstraction of the database and location data .The middleware maps the data retrieved in English to its vernacular counterpart using the grammar that is defined in the mapping module. We have defined a formal approach to validate the functionality of the mapping module and also present performance analysis for sample query.

Keywords: Context Aware Computing, Context Aware Middleware.

(Received March 12, 2010 / Accepted June 23, 2010)

1 Introduction

Context-aware ubiquitous computing emphasizes using context of users, devices, etc to provide services that are appropriate to particular person, space and time. Types of contexts are location, light, touch, temperature etc [2] (details are given in Table 8 of Appendix). To allow a user to concentrate on her task, mobile devices should be context aware and be able to automatically adapt to the changing context. Context aware systems are concerned with the acquisition of context (e.g. using sensors), the abstraction and understanding of context (context reasoning), and application behaviors based on the recognized context (e.g. triggering actions based on context)[9] . Modern software design uses methods of encapsulation to separate business logic and graphical user interfaces. The middleware-based approach introduces a layered architecture to context-aware systems with the intention of hiding low-level details. The task involved in the middleware that is designed for context-aware computing can be generalized as sensor data aggregation, context reasoning and context aggregation. In this paper, we propose context-aware middleware architecture (CAMA) for the agriculture domain that acts as bridge between agriculture database and the mobile devices and also abstracts agricultural data and location data uniformly. The CAMA is proposed for retrieval of location based agriculture related information from the database to the mobile devices of the farmer and a part of the architecture is implemented in this paper. The GUI of the mobile device and the answer to the query is in the vernacular of the farmer . The middleware is divided into three modules namely location generator (for location data), mapping module (for vernacular conversion) and SQL generator (for query retrieval)

2 Related Work

Context aware middleware encompasses uniform abstractions and reliable services for common operations.

In [8], Licia Capra et.al. proposes reflective middleware where execution context can be changed dynamically while application is in execution. Paper [10] [5] propose SOCAM (Service Oriented Context Aware Middleware) where [10] aims at building and rapid prototyping of context aware mobile services. SOCAM takes a service-oriented approach to build the middleware which supports tasks including acquiring, discovering, interpreting, accessing various context and also supports interoperability between different context-aware systems. In [5], OSGi-based infrastructure is proposed which manages context-aware service reliably. Papers [6] [7] [1] deal with design and implementation of CAMUS (Context Aware Middleware for Ubiquitous System).

[6] discusses the requirements of a context aware middleware in a distributed environments. [7] gives modular approach to allow different components to work cooperatively and supply context synthesis. In [1], author implemented CAMUS in pervasive home environments. But none of them have dealt with vernacular conversion, which is very important for agriculture domain. The e-Choupal model is a web-based information and procurement tools for Indian farmers. In [3], research objective is to develop hardware and software components that enable a seamless interaction between plants and artifacts in scenarios ranging from domestic plant care to precision agriculture. In this paper, we propose a context aware middleware architecture that provides context aware information to the farmer. The system empowers rural India with different relevant information in their vernacular. It can also provide GUI in local language adapted to the mobile device of the farmer which is not available in echoupal model. In echoupal model, queries can be made in local languages in SMS format. But no mechanism is provided to send the answer back in local languages. Since the local language is restricted to agriculture domain we have to deal with limited vocabulary. In our system, we have provided the mechanism of translation from English to Bengali for the limited vocabulary. SOCAM is taken because it is already developed for a mobile environment but it does not cater to the local language translation. These works will surely give more penetration of mobile services to the rural areas of India.

3 Scope of Work

In this paper we have proposed a context aware middleware architecture(CAMA) which is responsible for
1) Mapping the menu-driven vernacular to its English counterpart and vice versa.
2) Generate location of the user based on the information provided by the physical location sensors.
3) Generate location aware SQL corresponding to the request given by the application/users
Information in English is stored in a suitable database at the backend. Mapping module is implemented to convert the English information to its vernacular counterpart in which a farmer is more comfortable rather than English. Sensor data aggregation and context reasoning is proposed to be done in the location generator module of our architecture. The information varies depending on the location context of the users. For example, the price of rice or the soil information will vary from one location to another. So, when a user moves from one place to another, it is sensed by physical location sensors (GPS, Base-Station) and information will be retrieved accordingly making the architecture context aware. In rest of the paper, the architecture (CAMA) is discussed in section 4, the functionalities of different modules of CAMA is discussed in section 5, implementation of the part of the architecture (mapping module) along with result analysis is done in section 6.

4 CAMA Architecture

The architecture of the context aware middleware for agriculture domain (CAMA) is shown in Fig 1.

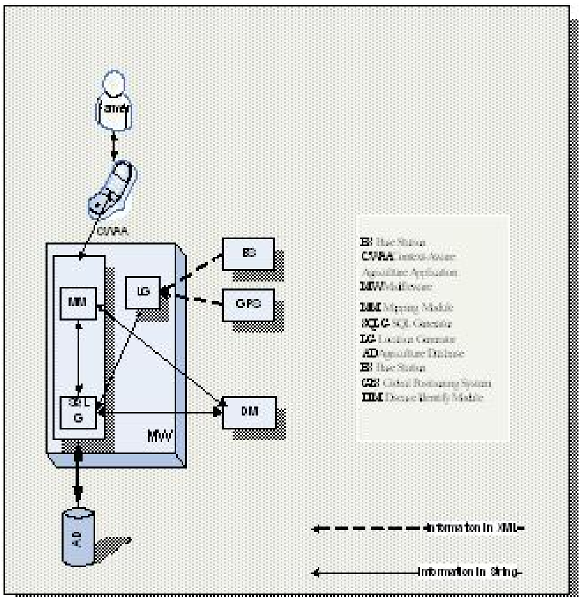


Figure 1: Architecture of the middleware

It consists of the following modules
1) Mapping module: This module is responsible for
a)Converting the menu driven vernacular icon to the corresponding English language

b)Converting English information stored in the database to its vernacular counterpart
2) Location Generator: This module will find out the location of the users based on the information given by the GPS enabled mobile devices or the base station. This will also has the functionality of providing deduced context based on direct context e.g. finding the name of the district from the information given by the GPS and base station.
3) SQL Generator: The output of Mapping module and Location Generator is used in this module to generate SQL.

5 Functionalities Provided by Each Module

5.1 Mapping Module

The GUI of the application on the mobile device is menu driven and in the vernacular of the farmer. This module is responsible for converting the menu driven vernacular icon to the corresponding English language. Mapping module will provide English counterpart of the vernacular icon, which is really a tough job. But, here, since the middleware is restricted to the agriculture domain, the vocabulary is limited. The vernacular buttons are implemented using PNG images of the local languages. If the first menu displays the vernacular words corresponding to
1. Disease
2. Weather
3. Crops
4. Soil
5. Fertilizers

The user wants to retrieve the information regarding the price of the rice for a particular region where he/she is presently locating. He/she will chose Crops and option 3 will be selected. The 2nd sub menu will display the vernacular words corresponding to
1. Rice
2. Wheat
3. Coffee etc

Farmer will choose Rice and option 1 will be chosen.

The next sub-menu is
1. Buying Price
2. Selling Price
3. Availability
4. Weekly Advice

The option 1 will be chosen.
The message from GUI interface of the mobile device will be received by the middleware in SMS format as 3.1.1, which means option 3 of the main menu and option 1 of the 1st submenu and then option 1 of the 2nd submenu have been selected by the user. In the Mapping module, the required English language translation can be done if the English counterpart of the numeric options is stored in a mapping file for menu . The mapping file can be in stored in a tabular fashion. From the tables, Mapping module will retrieve the information Rice and Price and will be sent as string str1 and str2 to the SQL generator. Assignment of str1 and str2 is decided by the Mapping module depending on the information it has regarding the different tables of the Agriculture database. In the Agriculture Database, there will be individual tables for all the options, which is assigned to str1. In each table, the options assigned to str2 are fields of the table. There will be an additional field location for each table of the database. Next, suppose user wants to know weather information of a particular place where he/she is staying, he/she will choose option 2 of the main menu. The middleware is responsible for generating the date and location. The table information will change weekly and are stored as day wise of the week. The middleware will find what is the day of the week and assigned this as str2.

If the information to be retrieved regarding Diseases, a proper diagnosis requires a manual intervention at the database end. It will take help of the Disease Identify Module (DIM). DIM is a GUI, which has all the information regarding disease file of the database. So, whenever disease option is chosen, MM will directly ask for DIMs manual intervention for diagnosis of the diseases. Through GUI of DIM, some more information regarding the diseases will be asked and the processes continue until a valid indication that is stored in the database has been acquired. During this whole process, Location generator will produce location information, which is required for the location dependent information regarding the diseases. Once the inferred diagnosis is done, that indication (str2) and particular crop for which diagnosis is done along with location information is sent to the SQL generator for further information retrieval from database regarding the disease. In this paper, we have not implemented the manual intervention at the database end i.e. the implementation of DIM is not done. The retrieval of disease information is also menu-driven.

5.2 Location Generator

This module will find out the location of the users based on the information provided by the GPS or Base-Station. This will also have the functionality of providing deduced context based on direct context e.g. finding the name of the district from the information given by the GPS and base station. The functionality of this module can be matched with the Context Reasoner and Context Provider module of SOCAM (Service oriented Context Aware Middleware) in [10]. Context Providers in [10] acquire location context data from sensors and convert them into OWL (Web Ontology Language). Context Reasoner has the functionality of providing deduced context based on direct context. OWL [4] is modeled through an object-oriented approach and the structure of a domain is defined in terms of classes and properties. In our model, XML can be used for this purpose. XML supports representation of information that is easily manipulated by machines and readily understood by human. So, the information regarding the location of the farmer is sent in XML format. To provide the functionality of the Context Reasoner, the whole area under consideration can be divided into regions and will be assigned a regionId.The information in the Agriculture database can be stored based on the regions. So, whenever the direct location context is sent by the base station or GPS, the middleware will generate deduced context by assigning it to corresponding regionId. That id is sent as a string to the SQL generator. This module is not implemented in our paper.

5.3 SQL Generator (SQLG)

This module is responsible for generating query. Basically, it is the interface between middleware and the database. Following are the example of how query is generated for different scenarios
CASE I $(NormalQuery)$
The output of the mapping module $(str1, str2)$ and Location generator $(regionId)$ is used in this module to generate SQL. The SQLcan be generated as follow :
SELECT str2 FROM str1 WHERE location = regionId
This is used for normal menu driven operation, which is implemented in this paper
CASE II
$(InteractiveQueryforidentificationofdisease)$
The output of DIM $(str1, str2)$
along with regionId from Location Generator
is sent to SQLG to identify disease.
SELECT Disease, Remedy FROM str1 WHERE location = regionId AND indication = str2
Case II is designed for the implementation of DIM.

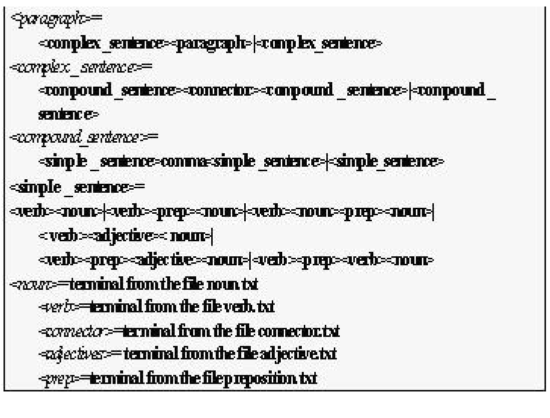
We use JDBC to generate and execute query. Compiling a query can become expansive overhead since query will be executed several times. So, a precompiled SQL created by using PreparedStatement object of JDBC can solve this problem.

6 Implementation of English to Vernacular Conversion in Mapping Module

Conversion of English to vernacular is required when the communication is from database to the user e. g. if the user selects Weekly adviceor Disease, the advice which is stored in the database in English language has to be converted to vernacular of the farmer for GUI display in local language. We sub-divide the mapping module into two modules along with the mapping file. Mapping file can be different for different vernacular implementation.
In our case, the mapping file is maintained for English to Bengali translation.

6.1 Converter

This module converts the English answer to a transformed English string from which word by word translation produces the corresponding vernacular. The grammar has been formulated for this conversion. The challenge of this module is that the construct of English language is different from any other Indian languages e.g. Apply Fertilizers can be converted to Bengali as sar proyog karun. Here, Fertilizer is sar and Apply is proyog karun. The main task is to identify Apply as verb and fertilizers as noun and interchange them.
A CFG is proposed in Fig. 2 for English paragraph.
Rules to be followed



```
<paragraph>=
    <complex_sentence><paragraph>|<complex_sentence>
<complex_sentence>=
    <compound_sentence><connector><compound _sentence>|<compound_
    sentence>
<compound_sentence>=
    <simple _sentence>comma<simple_sentence>|<simple_sentence>
<simple _sentence>=
<verb><noun>|<verb><prep><noun>|<verb><noun><prep><noun>|
    <verb><adjective>< noun>|
    <verb><prep><adjective><noun>|<verb><prep><verb><noun>
<noun>=terminal from the file noun.txt
    <verb>=terminal from the file verb.txt
    <connector>=terminal from the file connector.txt
    <adjectives>= terminal from the file adjective.txt
    <prep>=terminal from the file preposition.txt
```

Figure 2: The Proposed CFG

Rule 1:
Identify noun, verb, adjective, preposition and connectors using CFG of Fig 4 and different tables of Appendix which are related to this domain.
Rule 2:

a) Interchange the positions of noun and verb.
b) If the noun is followed by bracket (quantity), then it will remain as it is with the noun. Quantity will have some numerical value followed by ml, gm,kg,lt etc.
c) If more than one noun are separated by comma and or/and preposition, then nouns order will remain as it is.
Rule 3:
Adjectives are placed in the front of the noun. In case of more than one adjectives one after another, their order will remain as it is when being placed in front of noun.
Rule 4:
Prepositions are placed after the noun.
Rule 5:
Connectors position will remain as it is.
Rule 6:
If two verbs are separated by preposition to, second verb will be placed after noun followed by first verb
Rule 7:
Adverbs will be placed before verb.
Table 2 depicts the conversion from English to Bengali that is made by the Mapping Module Using the rules defined above.

6.2 Translator

The implementation of Translator (T) is rather straightforward. The output of the converter module can easily be translated to to the desired vernacular output
The main part of this sub module is a database that acts as a dictionary, which converts from English to Bengali. This will be done with the help of diffrent translator table. The table depicts how a source CFG is converted to desired vernacular output.

6.3 Performance Analysis

The implementation is divided into three major parts .i) Client ii) Middleware iii) Server
The client is a mobile handset with Java Application support. The handset should support Java Unicode. (This system has been tried successfully for Nokia N95 8GB, Nokia N70, Nokia N73.). The middleware installation is done on a workstation using J2SE 1.6 and GDM 6002D GSM Data Module. Installation of server is also done in a workstation, which supports HTTP protocol. The server is a SQL based database system with support of Sun SDK 1.6. The system was tested with Apache Tomcat 5. 5.12 web server and MySQL 4.1. database system.
The calculation of required time to execute the application is done with the following time parameter
$t_{procmidl}$ = the time calculated at the middleware from the instant of receiving the SMS up to the instant of sending the query to the server
$t_{midlserv}$ = the time calculated at the middleware from the instant of sending the query to the server up to the instant of getting the reply from the server.
$t_{totmidl}$ = the total time of operation in the middleware i.e. the instant of receiving the SMS up to the instant of sending the SMS back to mobile.
$t_{procserv}$ = the time calculated at the server from the instant of receiving the query up to the instant of sending the answer back then
t_{nd} = Network delay between middleware and server = $t_{midlserv}$ - $t_{totmidl}$ - $t_{procserv}$
and $t_{totproc}$ = Total processing time = $t_{procmidl}$ - t_{nd}
The table 1 contains the results at normal conditions i.e. information is available in the database dictionary. All times are calculated in milliseconds.Figure 3 is the graph obtained from the results of table 1.

Table 1: Performance Analysis

Qno	$t_{procmidl}$	$t_{midlserv}$	$t_{totmidl}$	$t_{procserv}$	t_{nd}	$t_{totproc}$
1	109	9109	17609	31	8969	8640
2	109	3079	6469	31	2957	3512
3	110	29797	35672	16	29671	5956
4	109	5828	11781	16	5703	6078
5	110	11859	18219	16	11733	6486
6	109	3031	9093	16	2906	6187

6.4 Performance Analysis

The implementation is divided into three major parts .i) Client ii) Middleware iii) Server
The client is a mobile handset with Java Application support. The handset should support Java Unicode. (This system has been tried successfully for Nokia N95 8GB, Nokia N70, Nokia N73.). The middleware installation is done on a workstation using J2SE 1.6 and GDM 6002D GSM Data Module. Installation of server is also done in a workstation, which supports HTTP protocol. The server is a SQL based database system with support of Sun SDK 1.6. The system was tested with Apache Tomcat 5. 5.12 web server and MySQL 4.1. database system.
The calculation of required time to execute the application is done with the following time parameter
$t_{procmidl}$ = the time calculated at the middleware from the instant of receiving the SMS up to the instant of sending the query to the server
$t_{midlserv}$ = the time calculated at the middleware from the instant of sending the query to the server up to the instant of getting the reply from the server.
$t_{totmidl}$ = the total time of operation in the middleware i.e. the instant of receiving the SMS up to the instant of

sending the SMS back to mobile.
$t_{procserv}$ = the time calculated at the server from the instant of receiving the query up to the instant of sending the answer back then
t_{nd} = Network delay between middleware and server = $t_{midlserv}$ - $t_{totmidl}$ - $t_{procserv}$
and $t_{totproc}$ = Total processing time = $t_{procmidl}$ - t_{nd}
The table 1 contains the results at normal conditions i.e. information is available in the database dictionary.

Table 2: Conversion made By the Mapping Module Using the rules defined

Slno	*English*	*Bengali*
1	Prune affected part	Osustha ongsho kete din
2	Provide sufficient sunlight	Proyojonio surjalok din
3	Apply fertiliser	Sar proyog karun
4	Use good seeds	bhalo bij byabahar karun
5	Apply insecticides(DDT) twice daily	Dine dui bar DDT proyog karun

All times are calculated in milliseconds.Figure 3 is the graph obtained from the results of table 1.

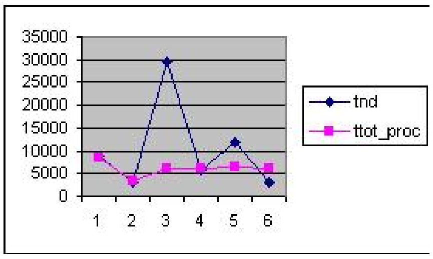


Figure 3: Performance Analysis Graph

7 Conclusion and Future Work

In this paper, we present a context aware middleware architecture (CAMA) for agriculture domain and have defined a formal approach to validate the mapping module. Our experimental results demonstrate reasonable performance for query retrievals .In future, we can implement the Location Generator module. We can also formalize grammar for other domains (paramedical, adult education etc.) that are applicable for rural areas and grammars can be defined for other Indian languages as well.

References

[1] Aekyung Moon, L. H. K., Hyoungsun Kim Kangwoo. Designing camus based context-awareness for pervasive home environments. *Proceedings of the International Conference on Hybrid Information Technology*, 1:666–672, Cheju Island, Korea 9-llth Nov 2006.

[2] Baldauf, M. and Dustdar, S. A survey on context aware systems. *International Journal of Ad Hoc and Ubiquitous Computing*, Vol 2, No. 4:263 – 277, 2007.

[3] Christos Goumopoulos, A. K. and O-Flynn, B. Proactive agriculture: An integrated framework for developing distributed hybrid systems. *Lecture notes in ComputerSeries*, (ISBN 978-3-540-73548-9):214–224, 2007.

[4] Deborah L. McGuinness, F. v. H. Owl web ontology language. w3c recommendation. *www.w3.org/TR/owl-features, World Wide Web Consortium*, 10 February 2004 2004.

[5] Gu T. Pung, D., H.K. Zhang. Toward an osgi-based infrastructure for context-aware applications. *Journals of Pervasive Computing.*, Vol: 3. Issue: 4(7-8):66–74, Oct-Dec 2004.

[6] Kiani, M. S. L. Y.-K. L., S.L. Riaz. Context awareness in large scale ubiquitous environments with service oriented distributed middleware approach. *Proceedings of the Fourth Annual ACIS International Conference on Computer and Information Science*, 60:513–518, Las Vegas, 1 1-14 Dec 2005.

[7] Kiani, M. S. L. Y.-K. L., S.L. Riaz. A distributed middleware solution for context-awareness in ubiquitous system. *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, (7-8):451–454, Hong-Kong, 17-19 Aug 2005.

[8] Licia Capra, W. E. and Mascol, C. Reflective middleware solutions for context-aware applications. *Lecture Notes In Computer Science*, 2192(ISBN: 3-540-42618-3):126 – 133, 2001.

[9] Schmidt, A. Ubiquitous computing - computing in context. *PhD dissertation, Lancaster University*, 2003.

[10] Zhang, T. G. H. K. P. D. Q. A middleware for building context-aware mobile services. *59th IEEE Vehicular Technology Conference*, 5:2656–2660, Milan ltaly, I7-21 May 2004.