Enhanced Generic Information Services Using Mobile Messaging

Muhammad Saleem, Ali Zahir, Yasir Ismail, and Bilal Saeed

Hanyang University, Erica Campus, Ansan, South Korea
engr_saleemwazir@yahoo.com, alizahir@hanyang.ac.kr,
marwat_telecomm@yahoo.com, cse.bilal@yahoo.com
http://www.hanyang.ac.kr

Abstract. This paper proposes a new, efficient method of building a scalable generic application which can be used to provide various types of information services using mobile messaging. For specific information, mobile users send an SMS (Short Message Service) to the mobile gateway in a proper format which is then forwarded to the generic application. The generic application accordingly creates an automatic query and generates a prompt reply.

With this new architecture, the final query generating algorithm becomes fast, efficient and processing overhead is reduced. Consequently, user gets the information promptly without any long delay.

Keywords: SMS, SMS Gateway, Generic Information, Multiple Databases, Dynamic Query Generation.

1 Introduction

Globally, there are around 2.4 billion people that own cell phones and 80% of them carry their phone all the time. A text message to their cell phone is the perfect and cheaper way to provide useful information and transaction services such as mobile commerce and banking [8, 10], sales reporting [3] , billing information updates etc.

SMS technology already have been used in public transport services [2], mobile-quiz [4], SMS Blogging [9] and payments [5] but still there is room for where this technology can make a big impact.

It is not a trivial job to develop a stand alone system for every new SMS based information or transaction service. Rather, it could be a good idea to have a single system which can provide more than one services and also can take the extensibility and re-usability into consideration.

Authors [1] successfully developed a generic application which can be used to provide more than single service. The system administrator is able to register new messaging services in a very easy way without any programming or design changes. The generic application dynamically communicates with databases and extracts information based on the contents of the SMS. In order to get specific information, a user will send an SMS to mobile Gateway which will forward it to

R.-S. Chang et al. (Eds.): GPC 2010, LNCS 6104, pp. 510–521, 2010.

the desktop application for necessary query execution. After collecting required information from a specific database, a prompt reply will be forwarded to the user in reverse order.

In this paper authors redefined the architecture and algorithm of [1] that lacks in terms of performance and processing overhead. Also the query generating algorithm generates some redundant information for every new user's SMS which slow down the processing and delays the information retrieval. In this new model, authors included the concept of skeleton query, due to which the redundancy in processing is removed and the final query generating algorithm becomes fast, efficient and short. In section V, authors compared new model and algorithm with previous work and proved that the new model is much better than old one in many aspects.

This paper is further organized as follows; section 2 shows an overview of the system, section 3 describes proposed design model for implementing this generic information system, section 4 describes an algorithm for generating a dynamic query, section 5 includes comparison of this model with the model discussed in [1], section 6 contains practical examples of information's systems for which this model can be used, and section 7 concludes future work.

2 Abstract View of the System

This section shows the abstract communication model of the system. The communication scenario of the system is same as described in [1]. The whole communication protocol of the system is divided into five steps as shown in Fig. 1. For commutation, each of the SMS Gateway and server contains and application developed in VB.Net.

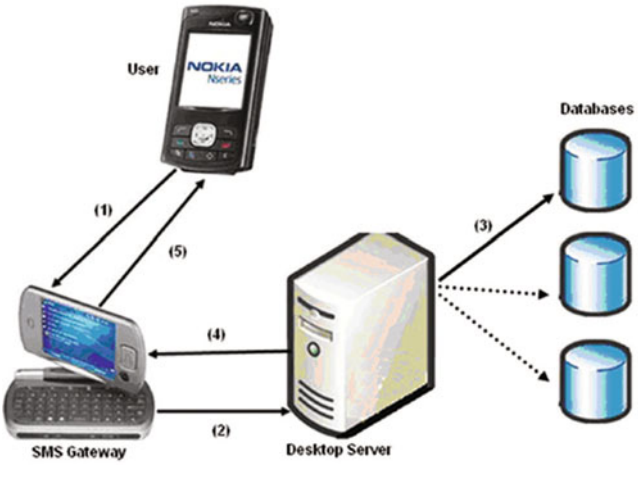

Fig. 1. SMS Messaging Service

1. For specific information, the user types an SMS in a proper format and sends to the SMS Gateway. The format of the SMS is predefined and conveyed to all users through some media like TV or internet.
2. After receiving the user SMS, the SMS Gateway application forwards it to the server application along with the user phone number.
3. Server application splits the SMS into sub-parts and read the semantic of it. After proper processing, the sever application generates a dynamic query and makes a dynamic link with the concerned database.
4. The result of the query execution is sent back to the SMS gateway along with the user number.
5. The SMS gateway then forwards the information to the concerned user as an SMS.

3 Proposed Model

Fig. 2 shows the detailed design model of the required system. This model works almost same as proposed in [1]. In this model authors focus on two main issues given as under;

- How to make the system scalable such that system administrator can register new messaging services without any programming changes.
- How the system is able to understand the semantic of the user SMS and extracts the required results.

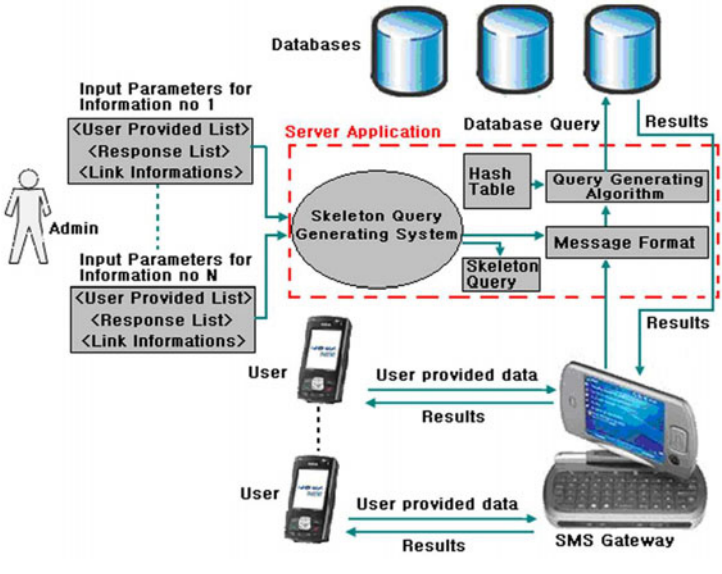

Fig. 2. Proposed Model

Server application is the intelligent entity in the whole system. Both of the above mentioned issues are solved by it. Before users can get any information, the system administrator must register new messaging services. For this Skeleton Query Generating System (SQGS) is used. The SQGS takes three database records as input and generates skeleton query, message format as output. The three input parameters are; *user-provided list, response list* and *link information.* These input parameters are provided by the system administrator using entity relationship diagram (ERD) of the concerned database. After generating the required outputs, the SQGS stored both of them into a specific table called *Hash Table.* Hash Table contains history of each messaging service provided by the system.

Typically *user-provided list* contains those tables attributes which user has to provide in sending SMS. The *response list* contains records which will be sent back to the user in reply. Both of the *user provided list* and *response list* only contain table attributes of a specific database. The third input parameter is the *joining conditions* between database tables. This parameter may be optional if all the records have to be extracted only from a single table. If the *user-provided list* or *response list* contains table attributes, selected from more than one table, then the administrator must provide this third input parameter.

For specific information, each of the users must follow the messaging format generated by the SQGS. The general format of the message is as under;

```
@ <Service No>  <user-provided list>
```

Each of the available messaging services providing by the generic information system are uniquely identified by *Service No.* Typically *Service No* starts from 1 and increases by 1 for every new service added by the administrator.

Skeleton query is the template for the final query to be executed against a specific database for required information retrieval. Skeleton query contains some holes which must be provided by the user through sending SMS. For each of the hole, the user must provide a specific data in SMS. If the third input parameter is empty then the general format of the skeleton query is;

```
Select <response list> from <tables_used> where <user-provided
attribute 1> = gap#1 and <user-provided attribute 2> = gap#2 and
so on.
```

If the *link information* parameter is not empty then the general format of the skeleton query is;

```
Select <response list>  from  <tables_used> where <user-provided
attribute 1> = gap#1 and <user-provided attribute 2> = gap#2 ...
and <link information>
```

Where <*tables_used*> are the names of database tables from where *user-provided list* or *response list* is selected. The SQGS generates the skeleton query by the concatenation of the input parameters with specific SQL key words in a proper order. *Hash table* is used to store the history of all the messaging services proved

by the generic information system. Table 1 shows the entries made by SQGS in *Hash table*.

For each of the new messaging service, the SQGS stores Service No, database name (which stores all information of the new messaging service), message format and skeleton query in the Hash Table.

In table 1, for service no 1 the *<response list>* is *Subject.Sub_name, Grade. Marks, Grade.Gpa, <tables_used>* are *Student , Grade , Subject* and *<user-provided attribute 1>* is *Student.St_id, <user-provided attribute 2>* is *Grade. Exam_name*. As more then one tables are used so the *<link information>* is *Student.St_id = Grade.St_id* and *subject.Sub_code = Grade.Sub_code*. In service no 2 and 3 only 1 table is used so there is no need of *link information*.

The records in *Hash table* are used by query generating algorithm for making the final query and a dynamic link with a specific database. For each of the new messaging service, the SQGS insert a single record into the *Hash Table*.

After adding new messaging service, the system administrator must convey the message format to all users. The system administrator can do this job by using website or some TV advertisement. Also user can send a query SMS to the server application, asking for the format of the SMS. The server application then replies with an SMS containing the exact format.

For each of the user SMS, the server application first split the SMS into subparts and then apply query generating algorithm to generate the final query. Query generating algorithm fills the gap in the skeleton query with proper user provided inputs. Hence skeleton query is once made and then used each time when new SMS message comes into the server application.

Table 1. Hash Table

Service No	Database Name	Message Format	Skeleton Query
1	Exam	@ <1> <St_id> <Exam_name>	Select Subject.Sub_name,Grade.Marks,Grade. Gpa from Student, Grade, Subject where Student.St_id=gap#1 and Grade.Exam_name = gap#2 and Student.St_id = Grade.St_id and subject.Sub_code = Grade.Sub_code
2	Exam	@ <2> <Sub_code>	Select Subject.Sub_name,Subject.crd_hrs from Subject where Subject.Sub_code = gap#1
3	Trans_ Services	@ <3> <Route_No>	Select Bus.Bus_No, Bus.Start_Stop,Bus.Dest_ Stop from Bus where Bus.Route_No= gap#1

4 Query Generating Algorithm

For specific information, the user SMS is forwarded to the server application. The server application applies query generating algorithm on the user SMS to generate the final query for required information extraction. This algorithm is divided into four steps given as under;

Input: User SMS in the general format(*@ <Service No> <user-provided list* >), *Hash Table* which contains the history of all messaging services provided by the system.
Output: An *SQL SELECT* query which executes against a specific database to extract required information, sent back to the user in reply.

1. Make sub-parts of the received SMS by splitting the SMS using spaces. Store the subparts of SMS in an array named "SubFields" such that SubFields(0) contains "@", SubFields(1) contains "Service No", and so on.
2. Execute a query on "Hash Table" for the Service No("SubFields(1)") sent by user in SMS, to extract skeleton query and database name. The query is like;
   Select sekeleton query, Db name from Hash Table where Service_No = SubFields(1)

3. Store skeleton query in a variable "Skquery" and database name in a variable "Db_Name"

4. for i= 0 to UBound(SubFields())
   Skquery = Replace(Skquery,"gap#" & i+1, "'" & SubFields(i+2)& "'")
   end for

At the end of the algorithm the variable *Skquery* contains the final query to be executed against the database stored in variable *"Db_Name"*.

Consider the skeleton query used for service no 1 in table 1, if the user SMS is **@ 1 2008553025 fall2009** then the final query after applying Query generating algorithm is as under;

```
Select Subject.Sub_name,Grade.Marks,Grade.Gpa from Student,Grade,
Subject where  Student.St_id = '2008553025' and Grade.Exam_name =
'fall2009' and Student.St_id = Grade.St_id and subject.Sub_code =
 Grade.Sub_code.
```

The above algorithm can also be implemented by constructing tree of the skeleton query and filling the gaps during traversing of the tree. The general tree of the skeleton query is shown in Fig. 3. The triangle represents sub trees for each node.

This method works well if the skeleton query may contains some syntax errors and we need to parse it for any syntax error. But in the given model, there is no possibility of any syntax error so authors argue that the proposed algorithm is short, fast and best fit for the required problem solution.

5 Comparative Analysis

In this section authors compare the new model with the model described in [1] and analyze the performance, processing overhead and accuracy of the model.

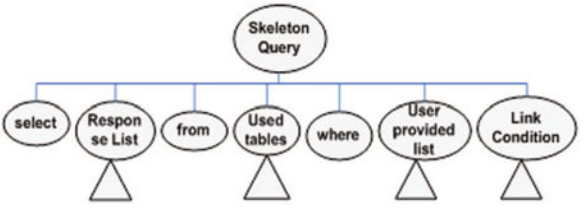

Fig. 3. General Tree of the Skeleton Query

In this new model, authors add a new concept of skeleton query due to which the query generating algorithm becomes very fast, efficient and short. Consequently the processing becomes very fast and users get information without any significant delay.

The main drawback in authors previous work [1] was that for each new SMS, the Query generating algorithm have to construct a new skeleton query and also fills the gap with proper user inputs. In this new model, SQGS only once construct the skeleton query and Query generating algorithm only fills the gap in it with proper data provided by the user in the sending SMS.

In our previous model, *user-provided list, response list, link information's* (if available), *message format, database name, Service No* and *tables used* are saved into the *Hash Table*. But with this new model, for each new messaging service, only *Service No, database name, message format* and *skeleton query* are saved. The columns entries into the *Hash table* are reduced from seven to four. Consequently it becomes easier to process the *Hash table* during query generating algorithm.

Other notable applications based on SMS technology are hospital search and appointments [11], Regional information services [12], Public Transport Service [2], Sales Reporting [3], Mobile-Quiz [4], and Payments [5]. Each of these applications provides specific information and did not take the extensibility and re-usability into consideration. Thus if system administrators want to add some new functionalities to an existing system, they have either to make changes to the existing system or rebuild the entire system from scratch [1].

Also for most of the SMS based applications like [11], [12], all the users are required to deploy an application to their mobiles. This application provides an easy interface for inputting required user data and can be used for security of the SMS. With this interface, it becomes easy for the users to provide required input data without following a strict SMS format. But it is not easy for every user to deploy a new application for each of the different services. Also these applications are device dependent and can only be dumped into specific high specification mobiles. Consequently this requirement limits the total number of users because only those users who have the required application in his/her mobile can get a specific service. But in our approach, we deployed a single application to the Gateway mobile only and all other users can simply get all the information by using a simple SMS interface without deploying any new application.

The only minor limitation to our approach is that all the users have to follow a fix SMS format. If the user SMS is not in the proper format then no information

can be retrieved. But if the user SMS is not in the proper format, then a prompt reply is forwarded to the user providing the correct SMS format. So for the second time, the user can easily provide the correct SMS format.

As our system only provides information and cannot do any transaction ("insertion, deletion and Updation") to the databases so security of the SMS is not the key issue for only information services.

6 Case Studies

The new proposed system can be used for all the practical examples discussed in [1-3, 6-7, 11, and 12]. In this section authors show some practical examples for which how the system works.

6.1 Exam Results

As discussed in [1], suppose the administrator wants to add a new messaging service "Viewing Exam Results through SMS". In which a student provides *student ID* and *exam name/semester* and gets grades in response to his/her SMS. The administrator can use a database named "Exam" which contains the results of all the semesters. The ERD (Entity Relationship Diagram) of the database is shown in Fig. 4.

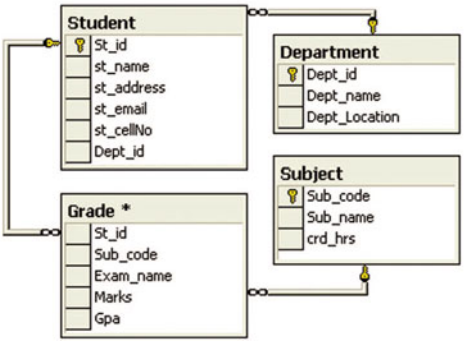

Fig. 4. ERD of the Exam Database

Fig. 5 show the screen shots of the Skeleton Query Generating System. The administrator selects three lists of attributes: one contains fields/attributes to be provided by the student in SMS and the other contains attributes to be provided in response to the user SMS/Query and the third input parameter is *link information* among tables, which is shown in Fig. 5.

As three tables are used during the selection for *user-provided list* and *response list* so all the three tables must be linked by some *join conditions*. Primary key of the tables are usually used for *join* between tables. For table *Student* and *Grade*, the *join condition* is *Student.St_id = Grade.St_id*. While for the *Subject* and *Grade* the *join condition* is *Subject.Sub_code = Grade.Sub_code*.

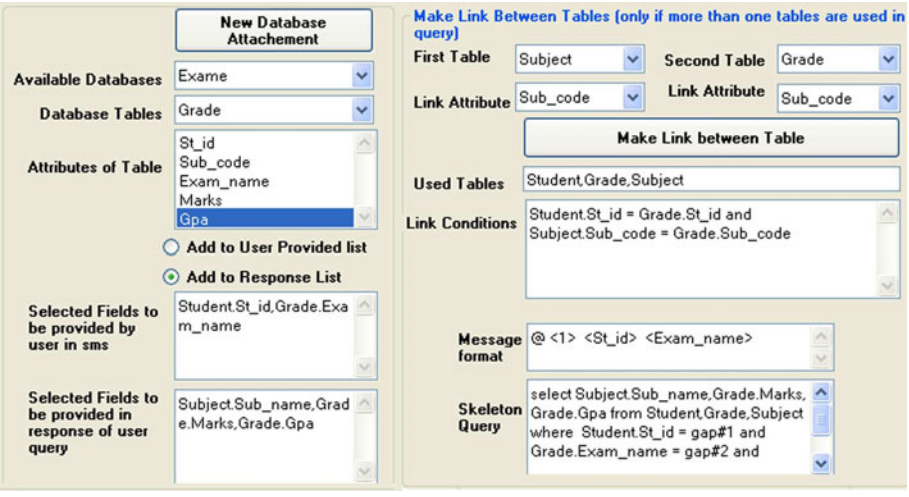

Fig. 5. Screenshot of SQGS for adding new information services

After the administrator provides all these three input parameters, the Skeleton Query Generating System produces the following message format;

```
@ <1> <St_id> <Exam_name>
```

This is the message format used by the student to view his/her results. Where <1> is the unique identifier for that SMS messaging service. The Skeleton Query Generating System produces the following skeleton query;

```
Select Subject.Sub_name, Grade.Marks, Grade.Gpa  from Student,
Grade,Subject where Student.St_id = gap#1 and Grade.Exam_name=
gap#2  and Student.St_id= Grade.St_id and subject.Sub_code =
Grade.Sub_code.
```

The entries saved by the SQGS in hash table are shown in table 1.

After sending SMS **"@ 1 200821 fall2008"** by the student having student id **200821** and exam name **fall2008**, the query generating algorithm produces the following SQL Query;

```
Select Subject.Sub_name, Grade.Marks, Grade.Gpa  from Student,
Grade, Subject where Student.St_id = '200821' and Grade.Exam_name
 = 'fall2008' and Student.St_id = Grade.St_id and Subject.Sub_code
 = Grade.Sub_code.
```

This final query is exactly same as produced by the query generating algorithm described in [1]. After executing the above query on database **Exam** the message sent from student, and result from server application to SMS gateway application is shown in Fig. 6. The SMS gateway application then forwards the data received form the server to the student as an SMS.

Fig. 6. Screenshot of Gateway Application

The administrator can provide other messaging service using the same database like "viewing details of a subject", in which the user send *subject code* and in response the *subject name* and *total credit hours* are returned to him. The *user-provided list* contains *Subject.Sub_code* and *response list* includes *Subject.Sub_name* and *Subject.crd_hrs*. As the entire attributes belongs to a single table *Subject*, so there is no need of *link information*.

After providing these two input parameters, the format of the SMS to be provided by the user is: **@ <2> <Sub_code>**.The structure of the skeleton query is;

```
Select Subject.Sub_name, Subject.crd_hrs from Subject where
Subject.Sub_code = gap#1
```

After sending SMS **" @ 2 CSE545"** having subject code **CSE545**,the final query generated by the query generating algorithm is;

```
Select Subject.Sub_name, Subject.crd_hrs  from Subject where
Subject.Sub_code = 'CSE545'
```

In order to know the format of the SMS to view results, a student may sent an SMS **@ <query> <Service No>** to the SMS gateway, in a reply the correct message format is provided to the student.

6.2 Weather Information

Let the administrator wants to provide weather information to the users, in which the user send the *city code* and the system replies with the complete weather

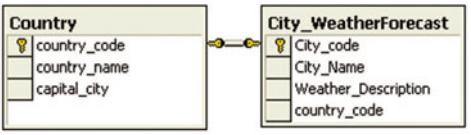

Fig. 7. Weather Information ERD

forecast of the city along with the country name. Fig. 7 shows the partial ERD of the system. After analyzing the ERD of the system, the system administrator selects table attributes *City_code* in *user-provided list*, and *country_name, City_Name* and *Weather_Description* in *response list*. As all these attributes are selected from two table so the third input parameter of the *join* condition between tables is like; *Country.country_code = City_WeatherForcast.country_code* After the administrator provides all these three input parameters, the Skeleton Query Generating System produces the following message format;

```
@  <3> <City_code>
```

The Skeleton Query Generating System produces the following skeleton query;

```
Select Country.country_name, City_WeatherForcast.City_Name,
City_WeatherForcast.Weather_Description  from  Country,
City_WeatherForcast where City_WeatherForcast.city_code = gape#1
```

After sending **SMS "@ 3 24789"** by the user having city code **24789**, the query generating algorithm produces the following SQL Query;

```
Select Country.country_name, City_WeatherForcast.City_Name,
City_WeatherForcast .Weather_Description from Country,
City_WeatherForcast  where City_WeatherForcast. city_code =
'24789'
```

7 Conclusions

In this paper, authors proposed a new architecture for a multi-purpose information system which can be easily used to provide different information in many firms. With this new architecture, the performance of the system becomes increased and processing overhead is reduced. The algorithm becomes very simple, fast and efficient. It is a cheaper way of providing useful information to users in those areas where there is no internet facility.

In future, authors will extend this system to an extend-able system which not only provides information but can also does transactions based on user SMS. The transactions could be like utility bills payments so that customer doesn't have to wait in long queues out side of the banks or other firms. With this new system, the user can make changes in the database records. But security is the main issue to such a system. The message must be sent in a secure way from user to the system.

Acknowledgments. This research work is sponsored by 'Higher Education Commission (HEC), Govt. Of Pakistan' under the scholarship program titled: MS Level Training in Korean Universities/Industry.

References

1. Saleem, M., Doh, K.G.: Generic Information System Using SMS Gateway. In: 4th international conference on computer science and convergence information technologies (ICCIT), South Korea, pp. 861–866 (2009)
2. Ching, L.T., Garg, H.K.: Designing SMS applications for public transport service system in Singapore. In: The 8th International Conference on Communication Systems, Singapore, vol. 2, pp. 706–710 (2002)
3. Wan, K.: An SMS-based sales reporting system for a fashion-clothes franchising company. In: Engineering Management Conference (IEMC), Managing Technologically Driven Organizations: The Human Side of Innovation and Change, New York, pp. 330–334 (2003)
4. Shahreza, M.S.: M-Quiz by SMS. In: Sixth international conference on advance Technologies, Tehran, pp. 726–729 (2006)
5. Aziz, Q.: Payments through Mobile Phone. In: 6th International Conference on emerging Technologies (ICET), Peshawar (2006)
6. Collins, C., Grude, A., Scholl, M., Thompson, R.: txt bus: wait time information on demand. In: Conference on Human Factors in Computing Systems, USA, pp. 2049–2054. ACM, New York (2007)
7. Aschoff, F.R., NovakT, J.: The mobile forum: real-time information exchange in mobile sms communities. In: Conference on Human Factors in Computing Systems (CHI), Italy, pp. 3489–3494. ACM, New York (2008)
8. Wang, H., Huang, X., Dodda, G.R.: Ticket-based mobile commerce system and its implementation. In: 2nd International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems, Spain, pp. 119–122. ACM, New York (2006)
9. Prasad, A., Blagsvedt, S., Toyama, K.: SMSBlogging: Blog-on-the-street Public Art Project. In: 15th international conference on Multimedia, Germany, pp. 501–504 (2007)
10. Jamil, M.S., Mousumi1, F.A.: Short Messaging Service (SMS) Based m-Banking System in context of bangladesh. In: 11th International Conference on Computer and Information Technology (ICCIT 2008), Bangladesh, pp. 599–604 (2008)
11. Edwards, T., Sankaranarayanan, S.: Intelligent Agent based Hospital Search and Appointment system. In: 2nd international conference on Interaction Sciences: Information Technology, Culture and Human, South Korea, pp. 561–567 (2009)
12. Stenentt, D., Sankaranarayanan, S.: Personal Mobile Information System. In: 2nd international conference on Interaction Sciences: Information Technology, Culture and Human, South Korea, pp. 561–567 (2009)