Journal of Network and Computer Applications 32 (2009) 684–702

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Pandora: An SMS-oriented m-informational system for educational realms

Lambros Boukas *, Georgios Kambourakis, Stefanos Gritzalis

Laboratory of Information and Communication Systems Security, Department of Information and Communication Systems Engineering, University of the Aegean, Samos GR-83200, Greece

A R T I C L E I N F O

Article history:
Received 20 July 2007
Received in revised form
22 May 2008
Accepted 5 July 2008

Keywords:
SMS
Mobile systems
Prototype implementation
Mobile informative services

A B S T R A C T

e-Informational systems based on the Internet infrastructure and services like e-mail, WWW, etc., are a de-facto option for various educational realms, in order to enhance the quality and diversity of services offered to their educators and students. On the other hand, despite the fact that pure mobile services like short message service (SMS) or multimedia message service (MMS) have managed to highly penetrate the wireless market to a great degree and gain users' wide acceptance, are rarely employed to support or offer informational services in the context of education. In this paper, we describe in detail a fully functional SMS-oriented mobile-informational (m-informational) system named Pandora that was designed and developed from the onset to specifically support a plethora of services obtainable mainly by the students of our university. The analysis and our contribution are two-fold starting from the theoretical background and continuing to the technical part of the Pandora system. We present and discuss several issues, including the different services that Pandora supports, system architecture, Pandora's box, core, Web services, security, etc. We demonstrate that the proposed system is practical to implement, flexible, effective, secure, affordable and above all scalable and potentially extensible.

1. Introduction

During the last decade the majority of educational institutions are moving towards Internet-based solutions to mainly promote distance education activities and generally improve the quality of services offered to both students and educators. The rapidly increasing use of computers in education and in particular the migration of many university courses to web-based delivery has caused a resurgence of interest among educators in non-traditional methods of course design and delivery (Wesson and Van Der Walt, 2005; Pei-Luen et al., 2008; Markett et al., 2006; Leung, 2006; Motiwalla, 2006; Zanev and Clark, 2005; Stone, 2004; Shiratuddin et al., 2003; Stone et al., 2002). At present, several enterprises, institutions and research centres are developing many applications destinated to education. So far, we are witnessing an outstanding growth concerning e-Universities, virtual Universities (http://vu.org/), hybrid models (Young, 2002), education portals, etc.

On the other hand, apart from pure academic tasks, institutions seek to find cost-saver and better, in terms of quality, effectiveness and flexibility, ways to provide informational services to anyone involved and interested in. For example, many students prefer to electronically submit their requests to the registrar and receive, e.g. the corresponding certificates. In this context, popular mobile services like short message service (SMS) and multimedia message service (MMS) can offer a straightforward and efficient way to cover the aforementioned demands to any user while on the move. For instance, while away, a student can retrieve his/her grades by sending a simple SMS to the registrar. Such tensions are verified by the New Media Consortium 2008 Horizon Report stating that "higher education is facing a growing expectation to deliver services, content and media to mobile and personal devices" (The New Media Consortium, 2008).

At present, SMS service is often characterized as a "killer-application". It has been proven very popular among the users of mobile phones, and 2G/3G mobile operators continue to promote it by offering many SMS-oriented services, like news, weather, sports, buddy finder, exchange market, location finder, etc. (Turel et al., 2006; Hsu et al., 2006). So far, several researchers utilize SMS and MMS services to support special instructional needs or achieve timely e-content delivery (Pei-Luen et al., 2008; Markett et al., 2006; Leung, 2006; Motiwalla, 2006; Zanev and Clark, 2005; Stone, 2004; Shiratuddin et al., 2003; Stone et al., 2002). Also, "Texting" can provide a comfort bridge to the transition process for new undergraduates at university (Harley et al., 2007). Moreover, other works highlight the

* Corresponding author. Tel.: +30 2273082010; fax: +30 2273082009.
E-mail addresses: lboukas@aegean.gr (L. Boukas), gkamb@aegean.gr (G. Kambourakis), sgritz@aegean.gr (S. Gritzalis).

1084-8045/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jnca.2008.07.002

significance to students of the privacy enabled by means of SMS messages being exchanged (Faulkner and Fintan, 2005). However, to the best of our knowledge no such system targeting specifically to educational realms and at the same time tailored around students and educators' special informational needs, exists until now. Contributing to this issue we mobilized into making the Pandora system. Pandora follows the well-known client-server model, but the requests towards the server are simple SMS messages. The same applies for the corresponding answers.

It is important to distinguish Pandora from fully fledged mobile platforms like the Android one (http://code.google.com/android/). The latter is a software stack for mobile devices including an operating system, middleware and key applications. On the contrary, Pandora, although highly customizable, is strictly an SMS/MMS-oriented m-informational application targeted to higher education.

In fact, Pandora itself must cooperate with any of the mobile providers available in the country; say Vodafone, Cosmote, Wind in Greece. This means that all SMS's are received or sent out through the network resources of a specific mobile provider. Naturally, Pandora's mobile provider is selected by the institution considering the percent of geographical coverage that the provider has, costs per SMS, quality of services, etc.

Pandora is able to offer an adequate group of various services and is designed to be extensible to cover any future needs that may come along. At a glance, Pandora services can be classified into three major categories, namely administrative, strictly academic and of general interest. The latter contains services like news, sports, finance, phone and e-mail directory, etc., while the former includes services related to various academic activities as certificates provisioning and applications submission. Finally, as the term implies, administrative services support the system itself. On the top of that, Pandora's services are backed up by a corresponding Web interface (see http://pandora.samos.aegean.gr) that enables the users to register and administer their Pandora accounts. In the following, our intension is not only to describe Pandora's services and general architecture, but to analyse and thoroughly discuss its major technical parts. In our opinion this would be a major importance for anyone who is interested in building an analogous system and/or extending its functionality.

Of course, another important issue is who pays for all these services. The answer to this question is the users themselves. This is possible through pre-paid time renewal mobile cards that are widely available and sold everywhere, e.g. in kiosks, mini-markets, etc. Pandora offers an innovative per subscriber billing service. Specifically, each Pandora user must buy and use time renewal pre-paid cards of the mobile subscriber that Pandora cooperates with. Therefore, independently of the mobile provider a specific user has subscribed to, he must buy a pre-paid card of the provider that Pandora system cooperates with. Considering that an SMS costs much lesser than a call duration of 30 s or an e-mail sent through the mobile network, a pre-paid card, say 10 €, can supply the user with a large number of SMS's (with current prices in Greece about 117 SMS's).

The rest of the paper is organized as follows. The next section describes Pandora's services. Section 3 depicts the system's overall architecture, while Section 4 presents and technically analyses major Pandora components. Section 5 exhibits future developments to Pandora and ongoing work. The last section draws a conclusion.

2. Pandora services

As already mentioned Pandora's services are classified into three main categories. That is: administrative, informative (of general interest) and strictly academic. Prior to obtaining Pandora's services candidate users must subscribe to Pandora. This is accomplished through the Pandora's web interface provided at: http://pandora.samos.aegean.gr. The registration page mandates the user to provide an e-mail address in the form: username@department.aegean.gr. This is necessary in order for the system to be sure that the applicant belongs to the academic community. E.g. only the Aegean's university users afford e-mail addresses with aegean.gr as the domain name. After the registration is completed the Pandora system sends a registration link by e-mail to the mailbox provided by the candidate user. The user must hit this link in order to activate his/her account. Otherwise, for the sake of security, the account remains frozen. From now on the system knows to which department's registrar's office to forward each incoming application by identifying the "department" field contained in each user's e-mail address.

In the following we briefly describe each one giving its general syntax. Each Pandora's subscriber must follow this syntax in order to prepare the SMS in his/her mobile device. Following the basic principle of Zipf's law, which is that human beings are subject to the principle of least effort (Zipf, 1949), Pandora's commands are properly adapted to reduce the level of complexity involved when memorizing or entering an SMS command. Simply put, according to the well-known Mooers' law "an information retrieval system will tend not to be used whenever it is more painful and troublesome for a customer to have information than for him not to have it" (Mooers, 1959). Actually, one could say that the SMS philosophy itself, permitting 160 characters at maximum, obeys the aforementioned law(s). It is to be noted that service codes given below have been translated into English. In fact, they are the first three letters of the corresponding Greek words. For example, the service code for SPORTS is SPO. To facilitate the reader we also provide in parentheses the corresponding word in English.

2.1. Administrative services

- *Balance renewal*: This service is responsible to renew the credit balance of each user. This is done by buying a new pre-paid card and sending the 16-digit code, which is written on the top up card, to the Pandora system. The general syntax of the command is:
  - *REN (RENEW) CODE*, where CODE is the 16-digit number found on the pre-paid card.
- *Subscribers service*: This service enables a Pandora user to subscribe or unsubscribe from a specific service, where service is one of the following: WEA (WEATHER), SPO (SPORTS), FIN (FINANCIAL NEWS), FOR (FORMULA1 NEWS), GRE (GREEK NEWS), INT (INTERNATIONAL NEWS), CUL (CULTURE NEWS), SCI (SCIENTIFIC-ENGINEERING NEWS). The general syntax of the command is:
  - *SUB (SUBSCRIBE) SERVICE_CODE {ACTION}*, where ACTION is ON (subscribe) or OFF (unsubscribe). For example: SUB SPO ON.
- *Balance service*: By using this service the Pandora subscriber can inquire Pandora about his current credit balance. The general syntax of the command is:
  - *BAL (BALANCE).*

2.2. Informative services

- *Weather forecast service*: Through this service a Pandora subscriber is able to keep up with the current and future weather conditions in 20 principal Greek towns. For instance, Athens, Thessaloniki, Heraklion, Patras, Samos, Chios, Chania, etc. The general syntax of the command is:
  - *WEA (WEATHER) TOWN {TIME}*, where TIME is TODAY or TOMORROW. For example: WHE ATH TOMORROW.
- *News service*: By using news service Pandora's subscribers can learn about the news for the corresponding categories that have previously subscribed to (see Section 2.1). The general syntax of the command is:
  - *NEW (NEWS) FIELD*, where FIELD is one of the following: WEA (WEATHER), SPO (SPORTS), FIN (FINANCIAL NEWS), FOR (FORMULA1 NEWS), GRE (GREEK NEWS), INT (INTERNATIONAL NEWS), CUL (CULTURE NEWS), SCI (SCIENTIFIC-ENGINEERING NEWS).

2.3. Academic services

- *E-mail service*: By utilizing this service a Pandora user can retrieve the e-mail address for any member of the academic staff. The general syntax of the command is:
  - *MAIL STAFF*, where STAFF the surname of anyone who belongs to the academic staff. For instance: MAIL BOUKAS.
- *Registrar's service*: By using this service students can at any time communicate with the registrar's office and submit various applications to it. The registrar's office, when available, edits the applications and sends via surface mail or even e-mail, if possible, the corresponding certificates. The general syntax of the command is:
  - *APPL (APPLICATION) USERNAME PASSWORD LIST(CERTIFICATE, NUMBER)*, where USERNAME and PASSWORD is the user's Pandora-specific username and password accordingly and LIST contains one or more application forms following the structure: CERTIFICATE, NUMBER. The CERTIFICATE field is also encoded containing one of the following: MIL (certificate for military service), ERA (certificate that the student has been erased from the institution's student record, QAL (certificate that the student is qualified to get a degree), EXA (certificate that the student has been examined in a certain course), CERT (certification of studies so far), DEG (officially signed degree), GRA (degree transcript—analytical grades), BOA (boarding certificate). Note, that this list can be easily updated according to the students' needs and the registrar's office requirements.
- *Register for courses service*: This service enables the students to register for courses at the start of each semester. The general syntax of the command is:
  - *REG (REGISTER) USERNAME PASSWORD LIST(COURSE CODES)*, where USERNAME and PASSWORD is the user's Pandora-specific username and password accordingly and LIST contains one or more course codes. For example: REG 1999099 YOUR_PASSWD 32101021 32110200.
- *Boarding service*: This service informs students about the boarding house's timetable. The general syntax of the command is:
  - *BOA (BOARDING) TYPE {TIME}*, where TYPE is one of the following: S for supper and L for lunch. TIME can be TODAY or TOMORROW. For instance: BOA L TODAY.
- *Phonebook service*: By employing this service a Pandora's subscriber is able to retrieve the phone number of a person that belongs to the academic staff. The general syntax of the command is:
  - *TEL (TELEPHONE) STAFF*, where STAFF is the surname of anyone who belongs to the academic staff. For instance, TEL BOUKAS.

3. Pandora's general architecture

Fig. 1 depicts the overall Pandora architecture including all the involved entities separated by virtual domains. Hereupon we present each entity giving a short description.

- The database server entity accommodates the Pandora's system database. It is responsible to support all the informative and administrative services. The database stores all the information the clients may request and also offers some auxiliary services like: incoming message recording, Web page administration, users' sensitive data secure storage, system logging.
- *The Web server entity*: Accommodates the Pandora's system Web pages and it is responsible to provide the users with the designated Web services (see http://pandora.samos.aegean.gr).
- *Communication network*: This entity binds together (a) the Database server entity with the Web server and (b) the Database server with the Application's server domain. This network may be implemented either by using a local area network (LAN) or remotely via a corresponding Internet connection. A hybrid solution is also possible. For example, the Database server entity may lie in the same LAN with the Application's server entity, while the Web server lies in some remote realm connected to the Internet.
- *Pandora's box entity*: This is the most prominent component of the Pandora system. It is responsible to coordinate all the other subsystems (entities) and to effectively control and dispatch the various requests (applications) arriving from the clients. Specifically, it receives SMSs containing requests for services through the GSM network, it processes them and finally, if appropriate replies, providing the correct information. It often requires a connection to the Internet in order to supply the users with the requested services.
- *RS232 or USB port*: This entity provides the interface to connect the GSM or universal mobile telecommunications system (UMTS) modem to the Pandora's box.
- *GSM/UMTS modem*: Any device that is able to accept subscriber identity module (SIM) cards and supports AT commands can serve as a GSM modem. In our case we employed a Nokia 32 (Private Branch Exchange) PBX connectivity terminal shown in Fig. 2.
- *The Client entity*: Anyone who has registered to the Pandora system. Clients use mobile devices like cell phones or handheld, laptop or desktop computers with a GSM/UMTS interface to send SMSs towards the GSM/UMTS network. They can also browse Pandora's Web site if required and an Internet connection exists.

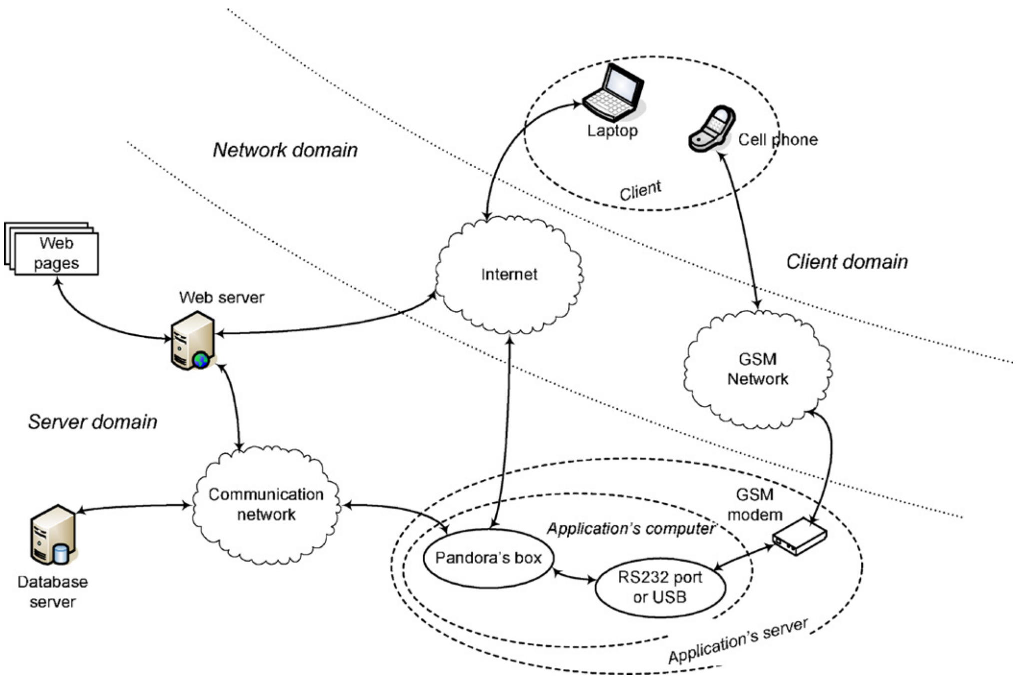


Fig. 1. General overview of Pandora's system architecture.

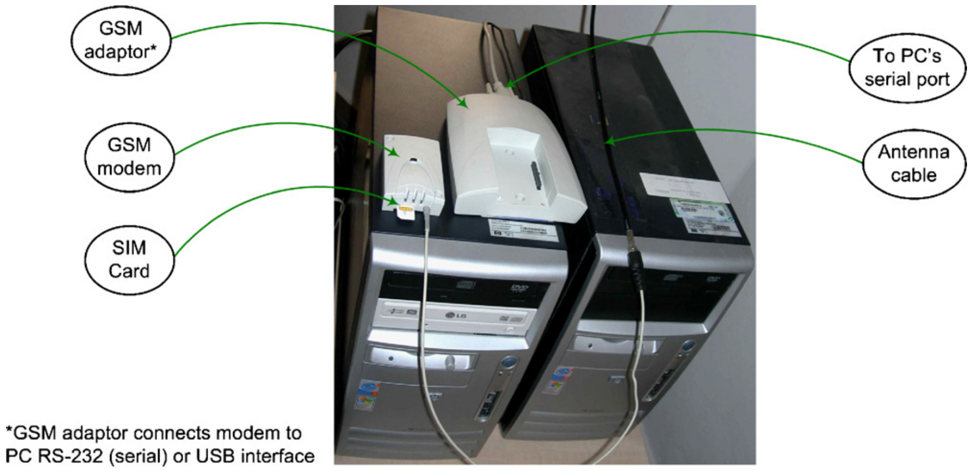

GSM adaptor*
GSM modem
SIM Card
To PC's serial port
Antenna cable
*GSM adaptor connects modem to PC RS-232 (serial) or USB interface

Fig. 2. Pandora's hardware.

- *The GSM network entity*: GSM network is utilized to send and deliver SMS messages to and from the clients. Newest mobile networks like UMTS services can also be used.
- *The Internet entity*: Internet infrastructure is used from the Server domain in order to retrieve the corresponding data depending on the service requested and, if appropriate, answer back by e-mail. Pandora's web pages are accessible via the Internet.

4. Pandora's box

In this section, we describe the overall structure and functionality of the Pandora application, namely the Pandora's box. As already mentioned, Pandora's box lives in the application's computer and is responsible for various distinct functions. Thus, it is possible to split it into different logical parts. All these parts are coordinated by the system's core.

4.1. Technical requirements

Before we examine the actual Pandora application, it is necessary to present all the technical components that we utilized in order to implement it (see Table 1).

Table 1
Technical requirements to implement Pandora's box

No.	Requirement	Description
1	Java 2 Platform Standard Edition (J2SE) version 1.5.0	Used to implement Pandora's box
2	MySQL version 4.1.10	Used to implement database functionality
3	PHP version 5	Used to realize the Pandora's Web pages
4	Apache 2.0.55	The server needed for Pandora's Web site to work
5	JavaMail API version 1.4	Used by Pandora's box to send e-mails
6	Java Communications API version 2.0	Needed to implement communication functionality between GSM modem and Application's computer
7	Nokia 32 PBX connectivity terminal	The GSM modem

4.2. *Pandora's box structure*

The Pandora's box structure is depicted in Fig. 3. The kernel itself will be analysed in the next section. Here, we shall present all the other parts described on the right side of Fig. 3.

4.2.1. *Service manager*

It is absolutely necessary for the Pandora's box to afford a handling mechanism for all the incoming requests. This element is called "the service manager". For example, think of 100 requests arrived at a certain time duration. The system must cope with that by responding to each one of them accordingly. First of all we assume that any incoming SMS containing a request is well-formed (see Section 2). Otherwise, the system will automatically respond back to the client's device indicating a syntax error. In case the received SMS is well-formed the system will proceed to process it. To accomplish this and depending on the case (requested service), the Pandora's box must execute a certain portion of programming code called *Pandora's task*.

Any Pandora's task must have a certain structure. This structure depends on a function set that every Pandora's task must implement. There are three such functions[1] called *Validate*, *Respond* and *Execute*. The first of them is in charge of validating whether the incoming SMS is a well-formed one. In case of error this function will report back to the service manager indicating an error code (e.g. invalid number of arguments, invalid argument). If not, the validate function shall give the green light to the service manager to proceed processing the message.

After that, the Respond function takes action. Its responsibility is to execute all the necessary steps to complete the requested task by the client. The majority of the requested services require data already stored in the database. But this is not always the case. Moreover, some services require immediate SMS-type response back to the client. In that case this function shall return to the service manager the exact content of the response. It is however, implied that the Respond function is different depending on the corresponding Pandora's task.

The above scenario—Validate and then Respond—is the most common one. Nevertheless, there exist some exceptions or cases that one or both of the aforementioned functions are not executed. For example, there are Pandora's tasks that do not require the execution of the Validate function. As a result, the Execute function designates the exact sequence and the exact modus operandi the other two functions will execute. All the arriving SMS requests are en-queued in a first in first out (FIFO)-type structure (queue). The pseudocode responsible to handle each SMS request is presented hereunder:

```
ServiceManagementAlgorithm()
    response = NULL
    WHILE ((SMSRequest = SMSRequestList.next()) ! = NULL)
        serviceIdentifier = GetServiceIdentifier(SMSRequest)
        serviceParameters = GetServiceParameters(SMSRequest)
        SWITCH (serviceIdentifier)
        CASE "A":
            response = ATask.execute(serviceParameters)
            BREAK
        CASE "B":
            response = BTask.execute(serviceParameters)
            BREAK
        …
        DEFAULT:
            BREAK
        END SWITCH
        IF (response ! = NULL) THEN
```

[1] The terms "function" and "method" are used interchangeably throughout the text.

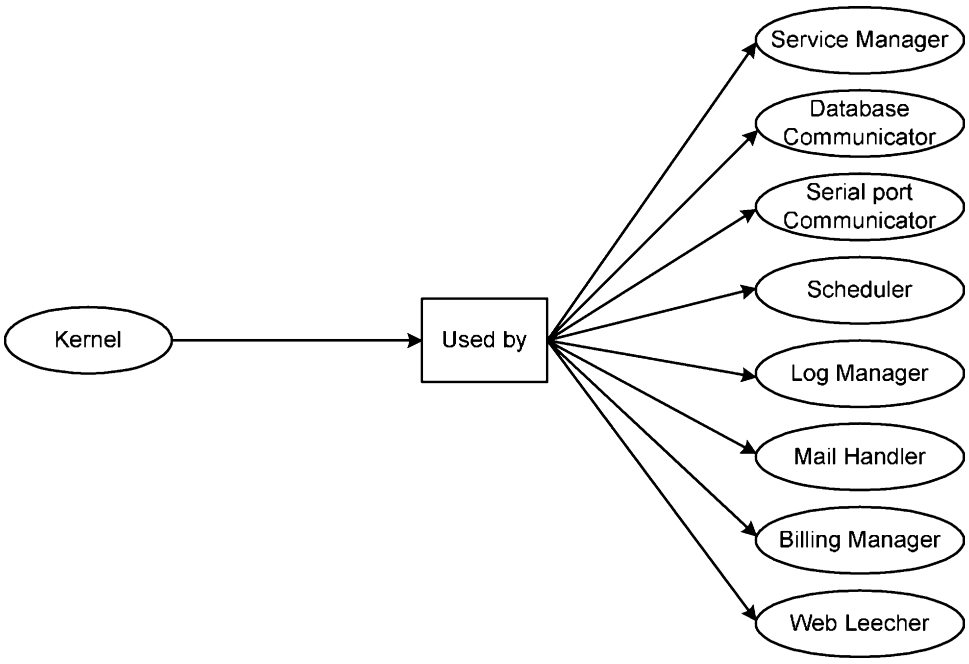


Fig. 3. Pandora's box structure.

```
            CUSTOMER_TEL_NO =
            GetTelephoneNumber(SMSRequest)
            message = CreateSMS(Response)
            SendSMS(CUSTOMER_TEL_NO, message)
        END IF
END WHILE
```

4.2.2. Billing manager

As stated in the Introduction Pandora offers an innovative fully automatic per subscriber billing service as well. This is implemented by the Billing manager. The Pandora system is based on its own SIM card to send SMS responses towards the clients (subscribers). Hence, in order for Pandora to be financially viable the credit balance of its SMS card must be somehow renewed. This must be done by Pandora's clients. In other words, Pandoras' subscribers must compensate for the services they receive.

It is well known that all major mobile operators offer at least one solution to their customers in order to enable them to update their credit balance via time renewal pre-paid cards of various amounts. A sample of such cards is depicted in Fig. 4. This is usually done by sending to a fixed number (say 1212) an SMS containing the 16-digit code printed on the card. Mobile subscribers can also learn about their current balance sending the corresponding SMS to the same number. It is to be noted that the aforementioned two services are free of charge. In this context, the Pandora's SIM card credit balance is updated every time a Pandora user sends to the Pandora system a 16-digit number found in a pre-paid card that he previously purchased. The only requisition is that those cards must be issued under the licence of the mobile operator that the Pandora has subscribed to. However, bear in mind that the Pandora's users can become customers of any mobile provider they want.

Further down we present algorithmically the scenario of renewing the Pandora's credit balance. It is worth noting that in a certain moment the algorithm itself can be in five different states implemented by corresponding functions described in Table 2.

In its general form the algorithm can be implemented by the following pseudocode, while a graphical representation of the billing procedure is illustrated in Fig. 5:

```
AddCreditsAlgorithm()
    state = IDLE
    done = FALSE
    WHILE (!done)
        SWITCH (state)
            CASE "IDLE":
                state = BUSY_CREDITS_LEFT_REQUEST
            CASE "BUSY_CREDITS_LEFT_REQUEST":
                BCLReq()
            CASE "BUSY_CREDITS_LEFT_RESPONSE":
                BCLResp()
```

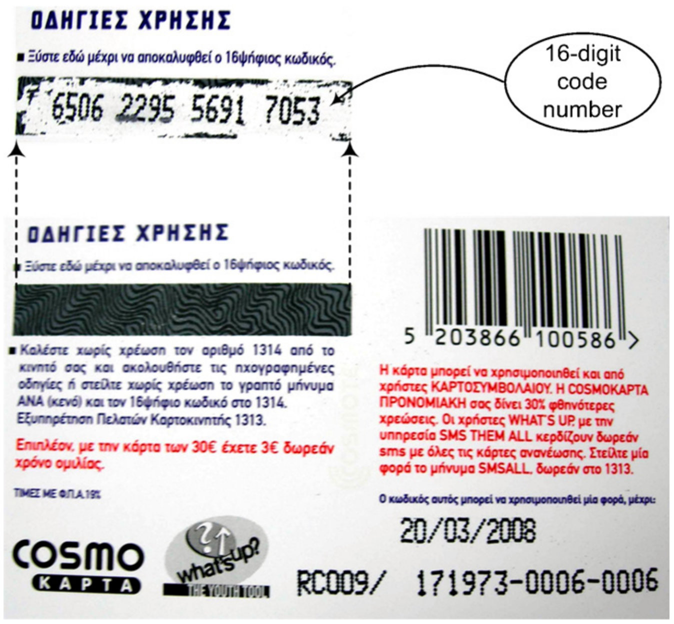

ΟΔΗΓΙΕΣ ΧΡΗΣΗΣ
Ξύστε εδώ μέχρι να αποκαλυφθεί ο 16ψήφιος κωδικός.
6506 2295 5691 7053
16-digit code number
Καλέστε χωρίς χρέωση τον αριθμό 1314 από το κινητό σας και ακολουθήστε τις ηχογραφημένες οδηγίες ή στείλτε χωρίς χρέωση το γραπτό μήνυμα ΑΝΑ (κενό) και τον 16ψήφιο κωδικό στο 1314. Εξυπηρέτηση Πελατών Καρτοκινητής 1313.
Επιπλέον, με την κάρτα των 30€ έχετε 3€ δωρεάν χρόνο ομιλίας.
ΤΙΜΕΣ ΜΕ Φ.Π.Α.19%
5 203866 100586
Η κάρτα μπορεί να χρησιμοποιηθεί και από χρήστες ΚΑΡΤΟΣΥΜΒΟΛΑΙΟΥ. Η COSMOΚΑΡΤΑ ΠΡΟΝΟΜΙΑΚΗ σας δίνει 30% φθηνότερες χρεώσεις. Οι χρήστες WHAT'S UP με την υπηρεσία SMS THEM ALL κερδίζουν δωρεάν sms με όλες τις κάρτες ανανέωσης. Στείλτε μία φορά το μήνυμα SMSALL δωρεάν στο 1313.
Ο κωδικός αυτός μπορεί να χρησιμοποιηθεί μία φορά, μέχρι:
20/03/2008
cosmo ΚΑΡΤΑ
what's up? THE YOUTH TOOL
RC009/ 171973-0006-0006

Fig. 4. A pre-paid time renewal card.

Table 2
Renew credit algorithm's states

No.	State	Description
1	IDLE	Pandora's box is in idle state
2	BUSY_CREDITS_LEFT_REQUEST (BCLReq function)	Pandora's box has sent a request to learn about its credit balance
3	BUSY_CREDITS_LEFT_RESPONSE (BCLResp function)	Pandora's box received an answer about its credit balance state from the mobile provider
4	BUSY_ADD_CREDITS_REQUEST (BACReq function)	Pandora's box has sent a request towards the mobile provider to update its credit balance
5	BUSY_ADD_CREDITS_RESPONSE (BACResp function)	Pandora's box received an answer about its credit balance renew request from the mobile provider

```
                CASE "BUSY_ADD_CREDITS_REQUEST":
                        BACReq()
                CASE "BUSY_ADD_CREDITS_RESPONSE":
                        BACResp()
                DEFAULT:
                        done = TRUE
                        BREAK
                END SWITCH
        END WHILE
```

where the BCLReq(), BCLResp(), BACReq(), BACResp() functions are expressed by the following portions of pseudocode accordingly:

```
BCLReq()
        content = CREDITS_LEFT_FORMAT
        message = CreateSMS(content)
        IF (SendSMS(CENTER_TEL_NO, message) = = TRUE) THEN
                state = BUSY_CREDITS_LEFT_RESPONSE
                sleep(MILLISECS_TO_SLEEP)
        END IF
        BREAK
```

After the execution of BCLReq() the system must check at regular time intervals in milliseconds if a matching SMS response from the mobile provider centre has arrived. Upon arrival the system must process and isolate from the whole SMS the usable content, namely the credit balance amount. The last two steps are implemented by the BCLResp() function:

```
BCLResp()
        SMSList = ReadSMS()
```

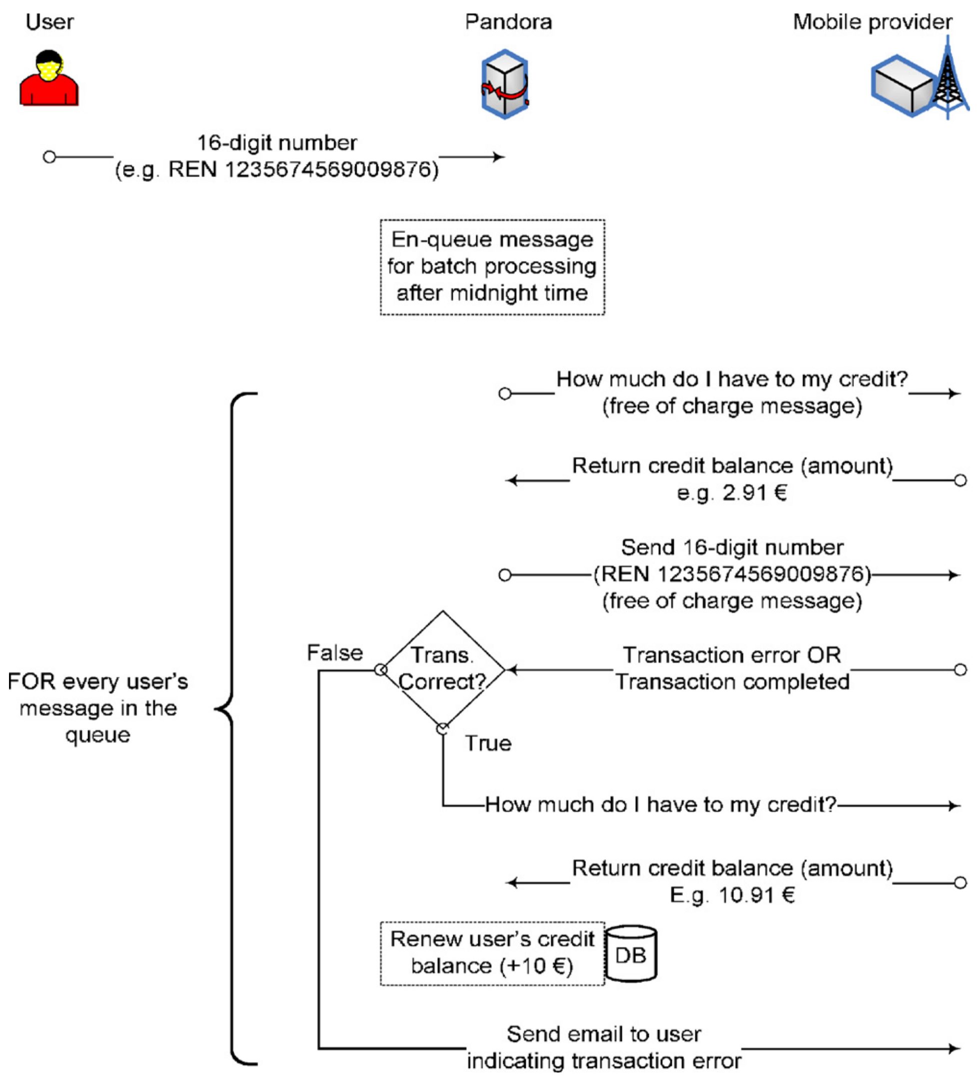

User
Pandora
Mobile provider
16-digit number (e.g. REN 1235674569009876)
En-queue message for batch processing after midnight time
How much do I have to my credit? (free of charge message)
Return credit balance (amount) e.g. 2.91 €
Send 16-digit number (REN 1235674569009876) (free of charge message)
FOR every user's message in the queue
False
Trans. Correct?
Transaction error OR Transaction completed
True
How much do I have to my credit?
Return credit balance (amount) E.g. 10.91 €
Renew user's credit balance (+10 €)
DB
Send email to user indicating transaction error

Fig. 5. Pandora's billing protocol.

```
        WHILE ((message = SMSList.next()) != NULL)
            IF (IsCreditsLeftResponse(message)) THEN
                initCreditsLeft = GetCreditsLeft(message)
                DeleteSMS(message)
                state = BUSY_ADD_CREDITS_REQUEST
                BREAK
            END IF
    END WHILE
    BREAK
```

After that the billing manager sends an update credit balance SMS to the mobile provider centre using the BCLReq() function:

```
BCLReq()
    content = ADD_CREDITS_FORMAT
    message = CreateSMS(content)
    IF (SendSMS(CENTER_TEL_NO, message) == TRUE) THEN
        state = BUSY_ADD_CREDITS_RESPONSE
        sleep(MILLISECS_TO_SLEEP)
    END IF
    BREAK
```

During the last step the billing manager waits for a response SMS from the mobile provider centre and upon arrival it processes the SMS contents. There are two possibilities: (a) Renew credit balance success and (b) Renew credit balance failure. In the former case everything goes as expected and the SMS received contains the new credit balance of the Pandora's SIM. Therefore, the new credit balance for the Pandora's subscriber who sent the initial SMS message is calculated based on the following formula:

$$credits = \frac{(finalCreditsLeft - initCreditsLeft)}{UNIT\_COST}$$

where *finalCreditsLeft* is the new credit balance (after the update process), *initCreditsLeft* represents the previous credit balance (before the update process) and *UNIT_COST* is the per unit cost charged by Pandora for its services. In the latter case an error is generated and sent to the client:

```
BACResp()
    SMSList = ReadSMS()
    WHILE ((message = SMSList.next()) ! = NULL)
        IF (IsAddCreditsResponse(message)) THEN
            done = TRUE
            IF (IsFailedAttempt(message)) THEN
                content = FAILED_ATTEMPT_FORMAT
                msg = CreateSMS(content)
                SendSMS(CUSTOMER_TEL_NO, msg)
            END IF
            IF (IsSuccessfullAttempt(message)) THEN
                finalCreditsLeft = GetCreditsLeft(message)
                credits = (finalCreditsLeft – initCreditsLeft) / UNIT_COST
                AddCreditsToAccount(CUSTOMER_ID,credits)
                content = SUCCESSFULL_ATTEMPT_FORMAT + "" + credits
                msg = CreateSMS(content)
                SendSMS(CUSTOMER_TEL_NO, msg)
             END IF
             BREAK
        END IF
      END WHILE
```

One last issue here is the exact method the billing manager charges for the services that Pandora provides. There are two parameters to consider: (a) mutable charge depending on the requested service and (b) mutable charge depending on the number of SMSs necessary to fulfill each client's request. Our solution is presented algorithmically down under:

```
BillingClassification(SERVICE_NAME, MSGS)
    unitCredits = GetUnitCreditsForService(SERVICE_NAME)
    creditsRemoved = unitCredits * MSGS
    RemoveCreditsFromAccount(CUSTOMER_ID, creditsRemoved)
    RETURN creditsRemoved
```

where *SERVICE_NAME* is the name of the requested by the client service, *MSGS* represents the number of SMSs necessary to carry out each client's request and *creditsRemoved* the credit points removed from subscriber's account.

4.2.3. Log manager

The Log manager is vital for the smooth operation of the whole Pandora system and helps administrators to improve it by providing the necessary feedback. To put it in another way, the Log manager starts when Pandora boots and is responsible for tracking and recording the corresponding log files every major event the system generates. The exact structure of every log file must be clearly defined. In our system the log files' records consist of the following fields:

[Date] [Class Method] Risk_level: Event, where *Date* the exact date and time the event took place, Class and Method the Java method of the Java class that generated the event, *Risk_level* characterizes the risk level of the event taking values from the set L = SEVERE, WARNING, INFO, CONFIG, FINE, FINER, FINEST} and *Event* is the technical description of the event occurred. The subsequent text box depicts two sample records taken from a corresponding log file.

```
[12 Οκτ 2005 1:13:47 μμ] [smsapp.comm.SerialConnector close]
WARNING: java.lang.NullPointerException
[12 Οκτ 2005 2:49:53 μμ] [smsapp.AegeanSMS batteryCharge]
SEVERE: java.lang.NumberFormatException: For input string: ''AT+CBC0''
```

4.2.4. Scheduler

Scheduler is a vital component for Pandora system. The Pandora's dedicated scheduler is responsible to: (a) en-queue incoming new Pandora jobs to its queue, (b) find next job for processing, (c) check whether a job needs to be repeated and calculates the repeat rate for that job and (d) delete finished jobs from its queue.

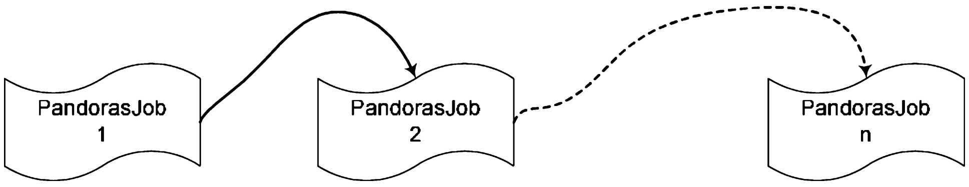


Fig. 6. Pandora's scheduler job vector.

Table 3
Renew credit algorithm's states

No.	Field name	Description
1	Execution steps	This field contains the necessary steps and procedures for the Pandora's box to complete the job
2	Execution date/time	This field includes the date and time, if any, the job must be executed
3	Repeat rate	This field contains the job's repeat rate, if any.
4	Number of iterations	This field holds the number of iterations the job must be executed, if any.

At first, in order to be able to process Pandora jobs Scheduler must have a concrete data structure and infinite length. This is true because it is not possible to know beforehand how many jobs will come in at a certain moment. For this reason we use a job vector which allows dynamical job handling. The Pandora's job vector is presented in Fig. 6. Each Pandora job must comply with the structure described in Table 3. Generally, the Scheduler is able to handle Pandora jobs based on three criteria: Date/time, repeat rate, number of iterations. For example, we can define a job to execute at October 12, 2006, 14:15:40 pm, with a repeat rate: daily, and a number of iterations: 10 times at maximum.

The scheduler vector keeps a number of Pandora jobs which wait for a specific event to happen in order to be executed. The necessary steps sequence performed by the scheduler to choose which job must be executed is as follows:

```
RunPandorasJobs()
    diff = NULL
    minDiff = Max(LongValue)
    now = GetCurrentMilliSeconds()
    pandorasJob = PandorasJobList.first()
    WHILE (pandorasJob ! = NULL)
        IF (pandorasJob.MilliSecondsToExecute() < = now) THEN
            ExecutePandorasJob(pandorasJob)
            IF (UpdatePandorasJob(pandorasJob) = = NULL) THEN
                PandorasJobList.remove(pandorasJob)
            END IF
        END IF
        ELSE
            diff = pandorasJob.MilliSecondsToExecute() - now
            minDiff = Min(diff, minDiff)
            pandorasJob = PandorasJobList.next()
        END IF
    END WHILE
    RETURN minDiff
```

where *PandorasJobList* is the job vector containing all the current Pandora's jobs and *minDiff* represents the minimum time in milliseconds until the next job execution is to happen. More simply put, the Scheduler checks all of Pandora's jobs that exist in its vector. For each one determines whether its execution time is greater from the current moment (time). If true this job must wait. Otherwise, the Scheduler executes the job by calling its pre-specified execution steps. Moreover, if the job needs not to be repeated it is immediately removed from the job's vector. If not, the scheduler determines the next time point the job must be repeated. This is done by the following method:

```
UpdatePandorasJob(P_JOB)
    calendar = GetCalendar()
    calendar.time(P_JOB.MilliSecondsToExecute())
    IF (P_JOB.repeat() = = MONTHLY) THEN
        calendar.add(calendar.MONTH + 1)
        P_JOB.setMilliSecondsToExecute(calendar.time())
    ELSE IF (P_JOB.repeat() = = YEARLY) THEN
        calendar.add(calendar.YEAR + 1)
        P_JOB.setMilliSecondsToExecute(calendar.time())
    ELSE
        P_JOB.setMilliSecondsToExecute(P_JOB.MilliSecondsToExecute()+P_JOB.repeat())
    END IF
```

```
        IF (P_JOB.count() ! =  FOREVER) THEN
            P_JOB.setCount(P_JOB.count() - 1)
        END IF
        IF (P_JOB.count  = =  0) THEN
            RETURN NULL
        ELSE
            RETURN P_JOB
        END IF
```

The UpdatePandorasJob method gets the argument *P_JOB* representing the Pandora's job for which its next execution time point must be determined and returns the same but updated job (*P_JOB*). In other words, the repeated rate time is added to the current time (the time the job was executed), we decrease the total number of iterations by one and check if there is some violation. If true the method returns null, otherwise returns the updated Pandora's job.

4.2.5. *Web leecher*

Web leecher is responsible to enrich the Pandora's system database with fresh information. This information can be found in corresponding Web sites either in clear text form or in extended markup language (XML) files. The XML files that the Pandora's application can elaborate come in the form of well-known rich site summary/really simple syndication (RSS) files (RSS feeds) that contain news headlines. The cardinal functions that Web leecher handles are: (a) collects remotely stored content, (b) splits up the retrieved content, (c) collects useful content and (d) stores collected data in Pandora's database. The Web leecher structure is depicted in Fig. 7.

As shown in Fig. 7 the parent leecher embodies the common parts or functionality of the Site leecher and RSS leecher functions as both need to collect information from Internet sites. Therefore, this common functionality (RetrieveContent) described algorithmically hereunder has been incorporated inside their parent:

```
RetrieveContent(URL)
    data  =  NULL
    content  =  NULL
    connection  =  openConnection(URL)
    WHILE ((data  =  connection.readLine()) ! =  NULL)
        content  =  content + data
    END WHILE
    content  =  removeTagsAndExtraSpaces(content)
    RETURN content
```

where *URL* represents the address of the remote Web site that we want to retrieve and *content* is the actual content of the remote file.

On the other hand, the Site leecher employees two functions. The first of them called *parse* defines the appropriate—different for each page—rules for parsing the Web page, perform pattern matching and thus locate the requested data. The second in the raw function called *execute* is responsible to store the useful content returned in the Pandora's system database.

Finally, the RSS leecher is in charge to process and retrieve useful content from the specified RSS feeds. Its function can be represented by the following pseudocode:

```
RSSLeecher(URL, TAG_NAME)
    ContentList  =  NULL
    content  =  RetriveContent(URL)
```

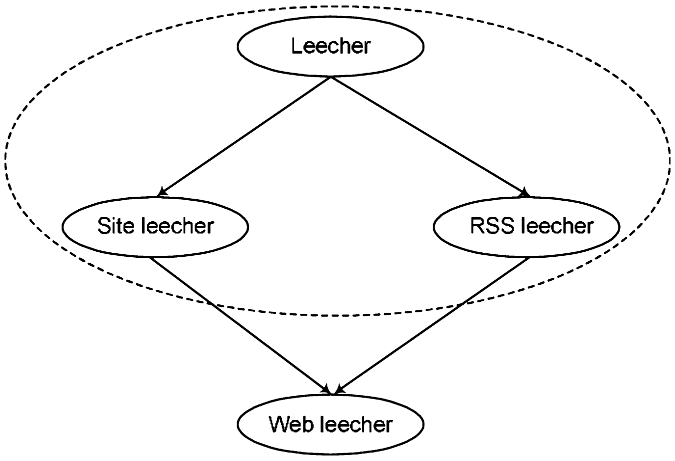


Fig. 7. Web leecher structure.

```
    IF (content != NULL) THEN
        ContentList = XMLParser(content, TAG_NAME)
    END IF
    RETURN ContentList
```

where *URL* represents the address of the remote RSS feed file that we like to retrieve its content, *TAG_NAME* is the actual tag marking inside the RSS feed file that contains the useful content and ContentList a list with the usable information in the RSS feed file. Moreover, the XML parsing procedure is described next:

```
XMLParser(DOC, TAG_NAME)
    WHILE (((startTag = GetNextStartTag(DOC)) != NULL)
    || (endTag = GetNextEndTag(DOC) != NULL))
        IF (startTag.name() == TAG_NAME) THEN
            title = NULL
            i = 0
            length = endTag.position() - startTag.position()
            WHILE (i < length)
                title = title + DOC.charAt(startTag.position() +i)
                i = i + 1
            END WHILE
            RSSList.add(title)
        END IF
    END WHILE
    RETURN RSSList
```

where *DOC* is the name of the file that will be split up in order to retrieve usable content, *TAG_NAME* is the actual tag marking inside the RSS feed file that contains the useful content and *RSSList* a list with the usable information in the RSS feed file.

4.2.6. *Mail handler*

Apart from SMS delivering, Pandora offers the possibility to send e-mail messages to their subscribers. This functionality can serve various Pandora's system needs: (a) instant reporting of errors generated by the system, (b) response to clients' requests depending on the service provided and (c) implementation of special services. In order for the Mail handler to be operational the following parameters must be defined:

- *smtp.host*: smtp server address.
- *smtp.port*: smtp.host port number for incoming mails.
- *smtp.auth*: A parameter that defines whether user authentication is required.

4.2.7. *Serial port (or USB) communicator*

As already reported, Pandora requires a serial or USB modem to receive and send SMSs. This modem connects to the serial or USB port of the host machine (application's computer). This means that Pandora itself must provide a serial or USB port manager in order to receive and transmit commands towards the specific port. This task is assigned to Pandora's port communicator. It is responsible for opening and closing the serial/USB port as well as to transmit and receive data through it. The following pseudocode describes port opening:

```
OpenSerialPort(PORT_NAME, PORT_OWNER, PORT_ID, PORT_PARAMETERS)
   PortList = listAllPorts()
   PortId = PortList.first()
   WHILE (PortId != NULL)
       PortId = PortList.next()
       IF (PortId.type() == PORT_SERIAL) THEN
           IF (PortId.name() == PORT_NAME) THEN
               open(PortId, PORT_OWNER, PORT_ID, PORT_PARAMETERS)
           RETURN TRUE
           END IF
       END IF
   END WHILE
   RETURN FALSE
```

where *PORT_NAME* is the port name, e.g. COM1, COM2, USB1, to which we want to connect, *PORT_OWNER* the alias name we assign to the port, after we create it, *PORT_ID* is the number or identifier of the port to which the application hears. For example, it can be 2214, 4545, etc. Finally, *PORT_PARAMETERS* specify some parameters to initialize communication with that port as baud rate, data bits, stop bits and parity. Next, we present the responsible pseudocode to read data from the port:

```
ReadData(RAW_DATA)
    Data = “”
    WHILE ((c = readChar(RAW_DATA) ! = -1))
        Data = Data + intToChar(c)
    END WHILE
    RETURN Data
```

where *RAW_DATA* the actual data that destined to the application through the port in raw format, and *Data* the meaningful for the application data after processing. Note, that the application reads the input stream character by character until the last one. Data transmission procedure through the port is similar, thus it is not presented here.

4.2.8. Database communicator

Most functions performed by Pandora’s box exploit part of the Database communicator entity. The majority of the services that Pandora offers entail some sort of communication with the system’s database. For instance, read, write, update, append, delete data from/to the database’s tables (see Appendix A). Then, it is necessary to implement a general purpose function described hereupon responsible to define a common background for any database transaction we may need:

```
01  ExecuteQuery(SQL_QUERY)
02      ResultsList = NULL
03      Result = NULL
04      statement = createStatement(SQL_QUERY)
05      IF (statement.hasResults() = = TRUE) THEN
06          WHILE ((Result = statement.getResult()) ! = NULL)
07              ResultsList.add(Result.columnName, Result.object)
08          END WHILE
09      END IF
10      RETURN ResultsList
```

where *SQL_QUERY* represents the query towards the database in SQL format; e.g. *SELECT * FROM ‘table’* and ResultsList is a list with the query results. Note, that ResultsList may be (return) NULL also. The above ExecuteQuery function (see line 5) differentiates queries depending on whether they return something or not. For example, thinking of SQL, UPDATE and INSERT queries do not return something, while SELECT queries always return information, even if null.

4.3. Pandora’s kernel

This section is dedicated to the most important part of the Pandora’s box, the kernel. The kernel is responsible for fundamental Pandora’s operations and takes part in almost every function the system performs. In other words, the kernel offers a common base for all other Pandora’s components in order to work. This common base is implemented through corresponding key functions which can be categorized as: *GSM format*, *SMS*, *System*, and *GSM modem* functions.

The former category encompasses all functions that convert SMS messages to GSM format and vice versa. Briefly, the GSM format mandates the representation of all messages in hexa-decimal form (hexa-decimal octets or decimal semi-octets). The 7-bit alphabet (http://www.dreamfabric.com/sms/default_alphabet.html) used to encode and decode every SMS message has been standardized by the European Telecommunications Standard Institute (ETSI) organization in technical document GSM 03.38 (GSM, 1998a, b).[2] There are two ways of sending and receiving SMS messages: by text mode and by protocol description unit (PDU) mode. The text mode (unavailable on some phones) is just an encoding of the bit stream represented by the PDU mode. Alphabets may differ and there are several encoding alternatives when displaying an SMS message. The most common options are “PCCP437”, “PCDN”, “8859-1”, “IRA” and “GSM”. These are all set by the AT-command AT+CSCS, when you read the message in a computer application. If you read the message on any mobile device, the device will choose a proper encoding. On the contrary, an application capable of reading incoming SMS messages can thus use text mode or PDU mode. If text mode is used, the application is bound to (or limited by) the set of preset encoding options. In some cases, that is just not good enough. If PDU mode is used, any encoding can be implemented. Thus, in order for Pandora’s box to be capable of receiving and sending SMSs it must incorporate functions to convert a character to hexa-decimal form and backwards. According to text mode, an SMS message consists of ASCII characters, while in SMS PDU mode the same message is encoded as a string, but in hex form. For example, the PDU string contains not only the message, but also a lot of meta-information about the sender, his SMS service centre, the time stamp, etc. It is all in the form of hexa-decimal octets or decimal semi-octets. Here, we present the responsible functions in pseudocode form:

```
Char2Hex(ch)
```

[2] The SMS message, as specified by the ETSI organization (documents GSM 03.40 and GSM 03.38), can be up to 160 characters long, where each character is 7 bits according to the 7-bit default alphabet. Eight-bit messages (max 140 characters) are usually not viewable by the phones as text messages; instead they are used for data in e.g. smart messaging (images and ringing tones) and OTA provisioning of WAP settings. 16-bit messages (max 70 characters) are used for Unicode (UCS2) text messages, viewable by most phones. A 16-bit text message of class 0 will on some phones appear as a Flash SMS (aka blinking SMS or alert SMS).

```
          hex = NULL
          WHILE (i < Length(alphabet))
                    IF (alphabet[i] = = ch) THEN
                              IF (i < 16) THEN
                                        hex = "0" + ToHex(i)
                              ELSE
                                        hex = ToHex(i)
                              END IF
                              BREAK
                    END IF
                    i = i + 1
          END WHILE
          RETURN hex
```

where *ch* is the character to be converted to Hex form, and *hex* is the converted character.

```
Hex2Char(hex)
          ch = NULL
          IF (ToInteger(hex) < Length(alphabet)) THEN
                    ch = alphabet[ToInteger(hex)]
          END IF
          RETURN ch
```

where *hex* is the character to be converted to its original form, and *ch* is the converted character. After that the fully fledged function that converts any string into the corresponding hex format is as follows:

```
String2Hex(str)
          hexStr = NULL
          WHILE (i < Length(str))
                    SWITCH (str.charAt(i))
                              CASE "A":
                              CASE "α":
                              CASE "ά":
                              hexStr = hexStr + Char2Hex("A")
                              BREAK
                              CASE "B":
                              CASE "β":
                              hexStr = hexStr + Char2Hex("B")
                              BREAK
                              ...
                              CASE "Y":
                              CASE "υ":
                              CASE "ύ":
                              CASE "ΰ":
                              CASE "ϋ":
```

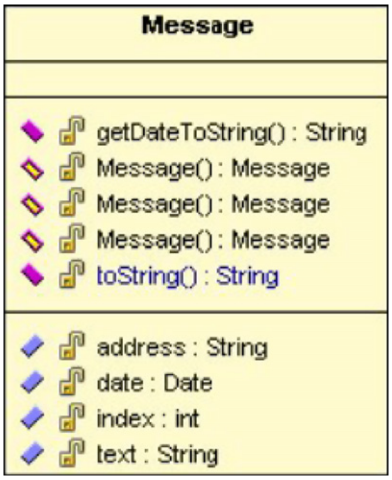


Fig. 8. The Java custom class "Message".

```
20                  hexStr = hexStr + Char2Hex("Y")
21                  BREAK
22                  …
23                  DEFAULT:
24                  hexStr = hexStr + Char2Hex(str.charAt(i))
25                  BREAK
26              END SWITCH
27              i = i + 1
28          END WHILE
29          RETURN hexStr
```

The second group of functions is accountable for handling the messages themselves. For instance, the send and delete message functions are enrolled in this category. Every SMS message that Pandora receives or sends has a concrete structure (from the elements that comprise it). This structure is implemented in our system by the Java class *Message* as depicted in Fig. 8. We note that every message is characterized by four instance variables: (a) address: the phone number of the sender or the recipient of the incoming/outgoing message, (b) the date the message was sent in the form: yyyy/mm/dd/hh/mm/ss, (c) the index of the system's memory in which the message has been temporarily stored and (d) the actual SMS request from or response to the client text. In this context further down we present the pseudocode of the functions SendSMS, ReadSMS and DeleteSMS accordingly:

```
01      Void SendSMS(SMSResponse, SerialConnector)
02          pdu = ToPdu(SMSResponse.getAddress(), SMSResponse.getText())
03          length = (Length(pdu) / 2) - 1
04          SerialConnector.SendData("AT+CMGS = " + length + "\r")
05          WHILE (SerialConnector.HasData())
06              SerialConnector.SkipData()
07          END WHILE
08          SerialConnector.SendData(pdu)
09          SerialConnector.SendData("\n")
```

where *SMSResponse* is the message in the appropriate form, namely an instance of the Message class, to be sent by the system, and *SerialConnector* an instance of the class SerialPortCommunicator responsible to forward the message from the Pandora's box to the GSM modem. Note that first of all the message must be converted into PDU format. After, to be capable to execute the corresponding AT command we must have the length of the message. In lines 5–7 after the input stream becomes clear from return characters, the message in PDU form is sent followed by the character that finishes the procedure:

```
01      ReadSMS(SerialConnector)
02          SerialConnector.SendData("AT+CMGL = 4\r")
03          response = SerialConnector.ReadData()
04          line = ReadLine(response)
05          WHILE ((line ! = NULL) AND (line ! = "OK"))
06              index = GetIndex(line)
07              pdu = ReadLine(response)
08              text = pdu.getText()
09              address = pdu.getAddress()
10              date = pdu.getDate()
11              Message message = new Message(text, index, address, date)
12              SMSList.add(message)
13              line = ReadLine(response)
14          END WHILE
15          RETURN SMSList
```

where *SMSList* is the list that contains the actual SMSs received by the system according to the form that the class Message mandates. The system knows whether there are any SMSs pending in the queue by sending the corresponding AT command (line 2). After that, the GSM modem will send all the pending SMSs in PDU form. At this point Pandora's box processes line by line every incoming message in order to convert it as the class Message designates. On completion, a new instance of a Message object is created and stored for further processing (see lines 11 and 12).

The GSM modem has limited memory space for storing the incoming SMSs. Thus, Pandora's box must implement the DeleteSMS function in order to periodically free memory from the GSM modem. This requires the index in the modem's memory of the SMS to be deleted:

```
01      DeleteSMS(index, SerialConnector)
02          IF (index >0) THEN
03              SerialConnector.SendData("AT+CMGD = " + index + "\r")
04              RETURN TRUE
05          END IF
06          RETURN FALSE
```

The third category of the kernel's functions deal with the system itself. For example, booting the system up or shutting it down. The former is presented in pseudocode below. First of all, the Connect function checks out the serial (or USB) communication with the GSM modem. After that the system informs the modem that the PDU representation of the data is to be used. Finally, the system tests the communication with the database. If everything goes as expected the system comes up and runs successfully:

```
Connect(SerialConnector, DatabaseConnector)
    connected = FALSE
    IF (SerialConnector.Open() = = TRUE) THEN
        SerialConnector.SendData("ATE0\r")
        SerialConnector.SendData("AT+CMGF = 0\r")
        response = SerialConnector.ReadData()
        IF (response = = "OK") THEN
            IF (DatabaseConnector.Open() = = TRUE) THEN
                connected = TRUE
            END IF
        END IF
    END IF
    RETURN connected
```

Last category of functions is responsible to provide information about the modem per se. These functions are very simple and employ the following AT commands towards the modem (ETSI, 1997): (a) signal quality AT+CSQ, (b) remaining battery power (if the modem uses battery) AT+CBC, (c) device's serial number AT+CGSN, (d) manufacturer of the device AT+CGMI and (e) device's name AT+CGMM.

4.4. Pandora's processes

As already mentioned in Section 4.2.4, Pandora's functionality is implemented by corresponding processes called Pandora's Jobs. What this means is that a set of Pandora's jobs, each one with different capabilities, are responsible for the system to work properly. All the distinct Pandora's jobs that exist in the system and described in the following are: *InitJob*, *FinalJob*, *WorkerJob*, *SubscriptJob*, *CreditJob*, *ContentJob*. In a nutshell, the only disparity between two Pandora's jobs is the run(·) method that they implement. In other words, in order to add a new functionality to the system it is sufficient to featly alter the run method of an existing Pandora's job or add a new one.

The InitJob process is the most important one because it is responsible to initialize the system. To accomplish that, InitJob utilizes the corresponding Kernel's system function, namely connect (see Section 4.3). Actually, the connect function is called from inside the body of the InitJob run(·)'s function. Bear in mind that every Pandora's job is inserted in the scheduler, which in turn administers the job vector. Hence, it is not possible to call the connect function from the beginning; we need a corresponding Pandora's Job to do that.

On the downside, the FinalJob process is in charge for shutting the system down. Like InitJob, this process calls the corresponding Kernel's system function, namely disconnect.

The WorkerJob process is extensively employed by the Pandora system. Its task is to serve the SMS incoming requests repeatedly depending on a predetermined frequency. In fact, WorkerJob is an integral part of Pandora's Service Manager (see Section 4.2.1), which it calls over time according to the programmed time rate.

The fourth process named SubscriptionJob serves everything regarding pending subscriptions to Pandora's services. A given user can be subscribed to any Pandora service either via the system's Web page, or by sending a corresponding SMS request. When this happens a record for this user is appended to the Pandora's database designating that the user wants to be a subscriber of the service specified. According to the system's job vector when the SubscriptionJob process comes along the system retrieves from the database all the records containing pending subscription requests to services. After that the Service Management Algorithm takes over (see Section 4.2.1) taking as input the SMS requests for new subscriptions.

The CreditJob process is responsible to update the credit balance for every Pandora's subscriber who has applied so. Since this process is time consuming, we decided it to be executed not in real time but in batch once daily after midnight. Otherwise, as described in Section 4.2.2 the whole system will be bound to suspend outgoing services until every request asking for balance renewal becomes fulfilled. This job exploits the Add Credit Algorithm for every pending subscriber record posing balance renewal in the Pandora's database. This means that (in the interim) the system records such requests in its database and some time after midnight serves them in batch using the CreditJob process.

The role of the last job, namely ContentJob is to automatically refresh the Pandora's news content. Actually, it is a general process depending each time on the informative services that Pandora supports. According to our implementation, ContentJob takes care of news content update process from the Internet through corresponding Web pages and RSS feeds. Thus, this process cooperates with the Web leecher component described in Section 4.2.5.

5. Future development

Although Pandora is self-contained there is always space for improvements, i.e. some advanced features can be added in order to enhance its services and better fulfill its purposes. In this respect we are planning to make Pandora accessible not only visually through SMS/MMS and HTML web browsers but also acoustically through both wired and mobile phones. The acoustic representation of the Pandora information will further contribute to the pervasive character of the system, i.e. by making it accessible from practically everywhere. Moreover, such a feature will be proved valuable to disabled individuals and students of developing countries, e.g. to those who cannot afford a mobile phone. Specifically, the audio nature of the application is expected to enhance the user's (visually impaired

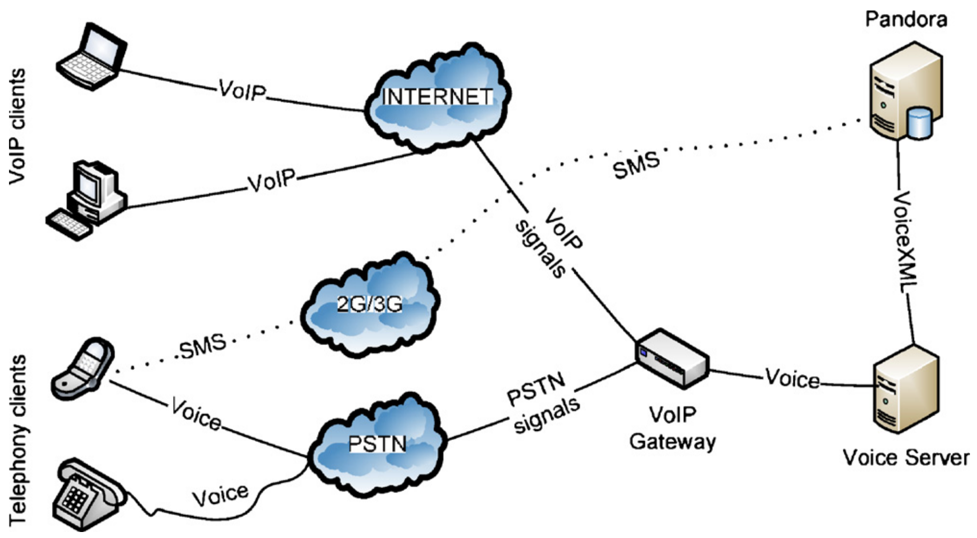


Fig. 9. System architecture for Voice calls.

people) experience by accepting voice commands for navigation throughout stored information and for controlling the application flow. Implementation details of such an architecture depicted in Fig. 9 are given below:

- *Pandora*: In the case of voice session, the Pandora's engine has to: (a) generate a dynamic grammar of the available options (menu), one of which will be the user's choice for presentation, (b) upon fulfilling a request, transform Pandora-text sequences retrieved from the database in VoiceXML (VoiceXML, 2008), and (c) send the generated VoiceXML files to the voice server for further processing.
- *Voice server*: This is the component responsible for the transformation of text documents to audio data. It includes a Voice browser, a Text To Speech (TTS) engine and an Automatic Speech Recognition (ASR) engine. Moreover, a VoIP gateway is an add-on component that plays an important role during the transformation procedure, but is not an actual component of the voice server itself. On the other hand, the Voice browser receives a request from the user and executes multiple tasks which include: (a) receives VoiceXML and grammar files from Pandora, (b) specifies the execution flow according to the instructions in the VoiceXML file, (c) isolates the text to be spoken and forwards it to the TTS engine. Also, it forwards the grammar to the ASR engine, (d) generates the request made by the user and delivers it to Pandora. The TTS engine receives the text from the VoiceXML file meant to be spoken, transforms it into streaming sound and sends it to the VoIP gateway in order to forward it to the end-user. Finally, the ASR engine receives a grammar, which is a set of terms that is able to recognize the client prompt and identifies if the prompt corresponds to any word in the grammar. If true it returns the term textually.
- *VoIP gateway*: This receives calls from the public switched telephone network (PSTN), converts PSTN signals to VoIP signals and forwards them to the Voice server. It is to be noted that the latter entity will accept only VoIP signals. Signals originating from the Internet (from VoIP clients) might be session initiation protocol (SIP) (Rosenberg et al., 2002) signals, or signals of some proprietary VoIP protocol, like Skype. The VoIP gateway is also responsible for the transformation of incoming VoIP signals to the protocol that the Voice server recognizes.

While the aforementioned voice subsystem will certainly boost Pandora's usability, it requires major changes and testing. In the meantime, we are retro-fitting our initial implementation in order to augment its array of services and make them more effective. In a future work we shall present MMS and mail2SMS via MMS functionalities, which far enhances Pandora's operational capabilities and thus maximizes its usability. Another direction is to have our Java implementation transferred to a more robust and extensible platform like the Kannel one. Kannel (http://www.kannel.org/) is an open source wireless application protocol (WAP) gateway but it can also work as an SMS gateway for GSM networks.

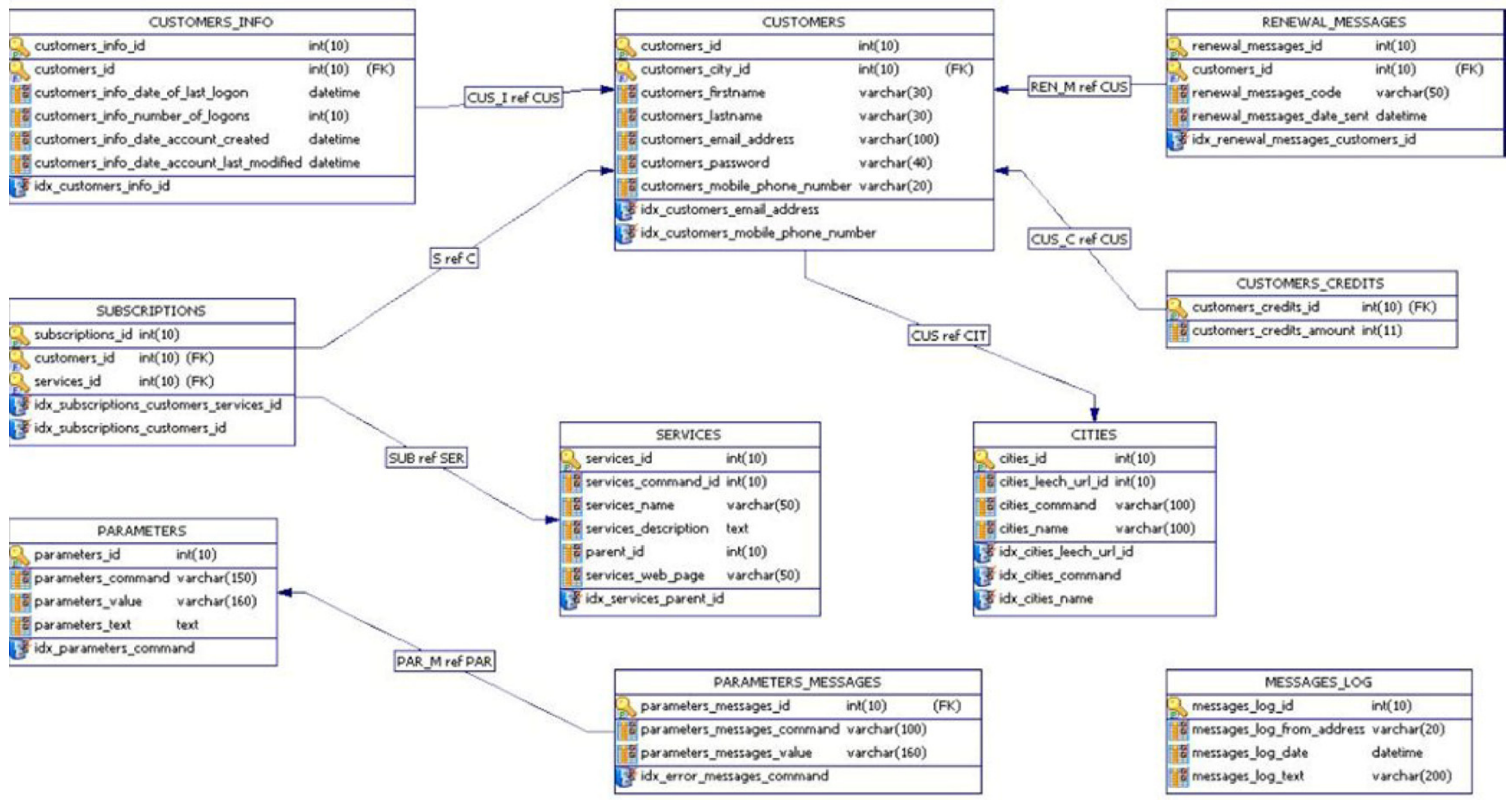
6. Conclusions

Gone are the days where the acquisition of a service dictated for interpersonal contact. Employment of mobile technologies provides a means for students, and sometimes educators, to use a familiar social communication style to acquire certain information or start creating social networks within the university. Actually, it is a fact that mobile phones have penetrated young people lives more than the Internet has, becoming a fashion and in some countries, like the developing ones, has clearly dominated the Internet. In this context, Pandora moves education's e-informational systems towards popular mobile services, namely SMS and MMS, capitalizing on the embedded bonus these services offer, i.e. the immediacy to a learning institution's information services.

This paper provides a detailed description of the Pandora prototype components, discussing their functionality and analyzing their aspects. It can be used as a template for anyone interested in building, expanding and deploying a Pandora-like system. In a more holistic view, the Pandora system must be considered as a flexible framework which can adapt and potentially cover on-demand any informational needs that may arise. This means that the educational realm using Pandora can custom-tailor it to support and further improve distance services offered to the academic community.

Appendix A

Layout of Pandora's database (tables and relationships).



CUSTOMERS_INFO
customers_info_id int(10)
customers_id int(10) (FK)
customers_info_date_of_last_logon datetime
customers_info_number_of_logons int(10)
customers_info_date_account_created datetime
customers_info_date_account_last_modified datetime
idx_customers_info_id
CUS_I ref CUS
CUSTOMERS
customers_id int(10)
customers_city_id int(10) (FK)
customers_firstname varchar(30)
customers_lastname varchar(30)
customers_email_address varchar(100)
customers_password varchar(40)
customers_mobile_phone_number varchar(20)
idx_customers_email_address
idx_customers_mobile_phone_number
REN_M ref CUS
RENEWAL_MESSAGES
renewal_messages_id int(10)
customers_id int(10) (FK)
renewal_messages_code varchar(50)
renewal_messages_date_sent datetime
idx_renewal_messages_customers_id
CUS_C ref CUS
CUSTOMERS_CREDITS
customers_credits_id int(10) (FK)
customers_credits_amount int(11)
S ref C
SUBSCRIPTIONS
subscriptions_id int(10)
customers_id int(10) (FK)
services_id int(10) (FK)
idx_subscriptions_customers_services_id
idx_subscriptions_customers_id
SUB ref SER
CUS ref CIT
SERVICES
services_id int(10)
services_command_id int(10)
services_name varchar(50)
services_description text
parent_id int(10)
services_web_page varchar(50)
idx_services_parent_id
CITIES
cities_id int(10)
cities_leech_url_id int(10)
cities_command varchar(100)
cities_name varchar(100)
idx_cities_leech_url_id
idx_cities_command
idx_cities_name
PARAMETERS
parameters_id int(10)
parameters_command varchar(150)
parameters_value varchar(160)
parameters_text text
idx_parameters_command
PAR_M ref PAR
PARAMETERS_MESSAGES
parameters_messages_id int(10) (FK)
parameters_messages_command varchar(100)
parameters_messages_value varchar(160)
idx_error_messages_command
MESSAGES_LOG
messages_log_id int(10)
messages_log_from_address varchar(20)
messages_log_date datetime
messages_log_text varchar(200)

References

Chin-Lung Hsu, Hsi-Peng Lu, Huei-Hsia Hsu. Adoption of the mobile Internet: an empirical study of multimedia message service (MMS), Omega. Int J Manage Sci, [special issue on Telecommunications Applications] 2007;35(6):715–26.

ETSI (European Telecommunications Standards Institute). Digital cellular telecommunications system (Phase 2), use of data terminal equipment-data circuit terminating equipment (DTE-DCE) interface for short message service (SMS) and cell broadcast service (CBS) (GSM 07.05), 5th ed. 1997.

Faulkner X, Fintan C. When fingers do the talking: a study of text messaging. Interact Comput 2005;17(2):167–85.

GSM. Digital cellular telecommunications system (Phase 2+), alphabets and language-specific information. European Standard (Telecommunications series), GSM 03.38 version 7.0.0 Release 1998, 1998a.

GSM. Digital cellular telecommunications system (Phase 2+), technical realization of the short message service (SMS); point to point (PP). European Standard (Telecommunications series), GSM 03.40 version 6.0.0, 1998b.

Harley D, Winn S, Pemberton S, Wilcox P. Using texting to support students' transition to university. Innov Educ Teaching Int 2007;44(3):229–41.

Leung L. Unwillingness-to-communicate and college students' motives in SMS mobile messaging. Telematics Inform 2007;24(2):115–29.

Markett C, Arnedillo Sanchez I, Weber S, Tangney B. Using short message service to encourage interactivity in the classroom. Comput Educ 2006;46(3):280–93.

Mooers CN. Mooers' Law; or why some retrieval systems are used and others are not, Zator Technical Bulletin 136, Cambridge, MA: Zator Company; 1959 [Editorial of same title, American Documentation, vol. 11(3), p. i; July 1960; and reprinted in The Scientist, vol. 11(2), p. 10, March 17, 1997, 1959].

Motiwalla LF. Mobile learning: a framework and evaluation. Comput Educ 2007;49(3):581–96.

Pei-Luen PR, Gao Q, Wu L-M. Using mobile communication technology in high school education: motivation, pressure, and learning performance. Comput Educ 2008;50(1):1–22.

Rosenberg J, Schulzrinne H, Camarillo G, Johnston A, Peterson J, Spark R, et al. Session initiation protocol, RFC 3261, June 2002.

Shiratuddin N, Hassan S, Landoni M. A usability study for promoting eContent in higher education. Educ Technol Soc 2003;6(4):112–24.

Stone A. Mobile scaffolding: an experiment in using SMS text messaging to support first year university students. In: Proceedings of the IEEE advanced learning technologies '04, 2004. p. 405–9.

Stone A, Briggs J, Smith C. SMS and interactivity—some results from the field, and its implications on effective uses of mobile technologies in education. In: Proceedings of the IEEE workshop on wireless and mobile technologies in education '02, 2002. p.147–51.

The New Media Consortium & EDUCAUSE Learning Initiative. The Horizon Project 2008, p. 5, available at: 〈http://www.nmc.org/pdf/2008-Horizon-Report.pdf〉, 2008.

Turel O, Serenko A, Bontis N. User acceptance of wireless short messaging services: deconstructing perceived value. Inform Manage 2007;44(1):63–73.

VoiceXML. Voice extensible markup language (VoiceXML) version 2.0, available at: 〈http://www.w3.org/TR/voicexml20/〉, 2008.

Wesson JL, Van Der Walt DF. Implementing mobile services: does the platform really make a difference? In: Proceedings of the ACM SAICSIT '05, 2005. p. 208–16.

Young G. Hybrid teaching seeks to end the divide between traditional and online instruction. Chron Higher Educ 2002:A33–4.

Zanev V, Clark R. Wireless course management system. In: Proceedings of the 43rd annual southeast regional conference, vol. 2 ACM-SE 43. ACM Press; 2005. p. 118–23.

Zipf GK. Human behavior and the principle of least effort: An introduction to human ecology. Cambridge, MA: Addison-Wesley; 1949.