A General Agriculture Mobile Service Platform

Hu Haiyan[1,2,*] and Su Xiaolu[1,2]

[1] Key Laboratory of Digital Agricultural Early-warning Technology, Ministry of Agriculture, The People's Republic of China 100081
[2] Agricultural Information Institute of Chinese Academy of Agriculture Science, Beijing 100081, P.R. China
Tel.: +86-10-82106263; Fax: +86-10-82106263
huhaiyan@mail.caas.net.cn

Abstract. Most of today's information services on the web are designed for PC users. There are few services fit to be accessed by mobile devices. In the countryside of China, most of the mobile phone users can not access the Internet. For this reason, We developed General Agriculture Mobile Service Platform. The Platform is designed to make these information services fit to be accessed by mobile users, and to make those mobile phone users can use these services without Internet connection. To achieve that, a descriptive language is designed to describe the services' inputs and outputs, used to passing requests and responses between the platform and the mobile client software. With those descriptions, client software can generate user interface on the client mobile device. Using that interface, user can manipulate service. The communication between client side and the platform can be carried by SMS, MMS as well as TCP, so that the devices which don't have Internet connection can access those services.

Keywords: Mobile service, Mobile platform, SMS protocol, MMS protocol, Mobile protocol.

1 Introduction

Do not have internet connection, one can only be reached by mobile specific communication protocol. Most of the users of cheap mobile devices, as their devices has so limited operability and is not fit for using internet application, will simply chose to have none internet connection. Their mobile device supports only phone call and SMS. To make our service being reachable to these users, a SMS/MMS based communication protocol is developed to handle communication via SMS and MMS, and an independent mobile communication protocol layer is added to the platform. With SMS based communication, the platform can be reached by 100% of mobile users.

The processing capacity, presentability, and operability of mobile devices are limited, complicate data presentation and complicate operation can not be done on mobile device. Standard web based user interface is not fit for mobile users. Between mobile

* Corresponding author.

D. Li, Y. Liu, and Y. Chen (Eds.): CCTA 2010, Part I, IFIP AICT 344, pp. 41–47, 2011.

devices, capacity of display and operate vary so greatly that can not design a standard client user interface to fit all types of devices.

On mobile devices, software installation and running is limited, complicate software functions is not applicable. Client software must adaptable to all these limitation.

2 Platform Design

2.1 Client/Server Architecture

Mobile terminal may not support Internet connection. Client/server is the only applicable architecture to leave room for the special designed mobile communication protocol.

2.2 Mobile Protocol Layer

Seal mobile communication protocol to an independent layer, simplified both the client and the server software design, and made maintenance and upgrading easy.

2.3 Web Service Composition

Server composes available web services into functions which fit for mobile user. The client software can only have a set of very limited functions, so that it can not call complicate services by itself, services must be packaged by server firstly, to make them fit for calling by client software.

2.4 User Interface Auto Generation

Based on the capacity of the current mobile device, client software generates user interfaces for each service according to its service description. Each function offered has a formal description, which includes descriptions to all parameters that function needs. All functions consists the service list, which is the data source of service choice list on the client side. With those descriptions, client software can generate user interface to launch calling to those functions, and to handle the returned data.

2.5 Command Pattern

Client side software launch calling to server side functions by translation user inputs into commands and sending them to server. The server translates each command to the actual calling to launch the corresponding function, and then package the returned data as response to that command. Then client side handles the returned response to generate user interface to display it.

2.6 Information Leaning Based on Personal Knowledge and Demand

Server traces each user's access records, calculates his possible knowledge structure and interest point. With that information, server can lean the information returned by

remove those out of the scope of user's knowledge and those out of user's interest. This function can be switched off manually.

3 Implementations

3.1 SMS and MMS Based Communication Protocol

A single SMS message can carry only 140 byte information, equal to 140 ASCII characters or 70 double byte symbols. This sets the up limit of a single data package. Such a package size is too small to carry information (compared to other communication protocols, such as Ethernet's 1544 byte package). To transmit data over such a tiny package, the first thing must do is to make the package structure as simple as possible to leave room for data; the second thing must do is to make data can be carried by multiple packages, that means data must be dissembled at transmit end and assembled at receive

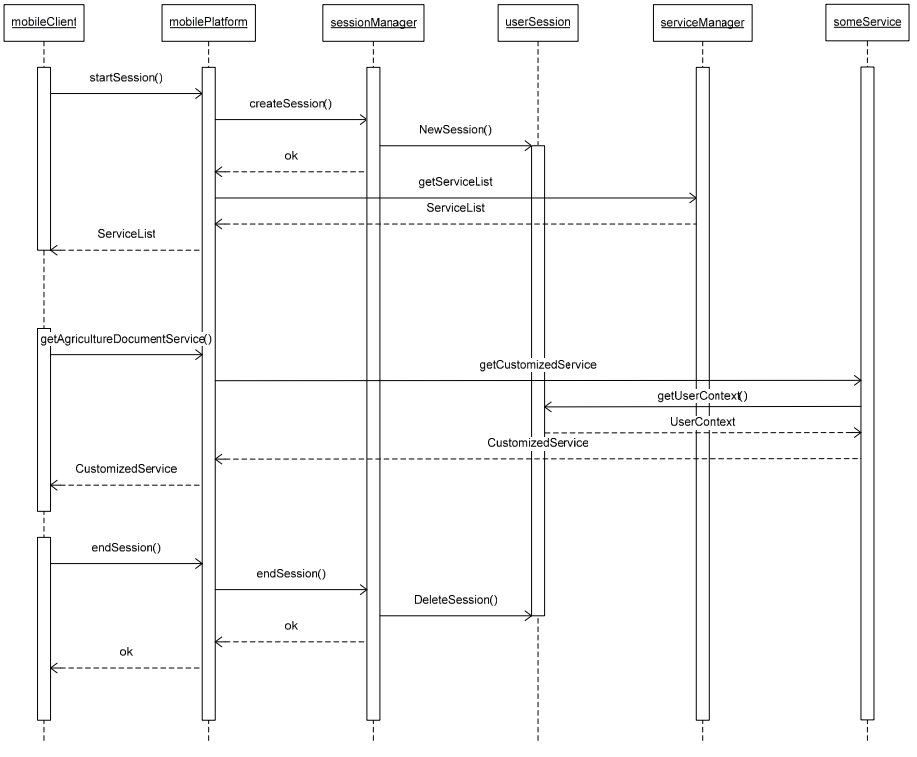

Fig. 1. A typical session sequence

end, some kind of sequence control must be introduced to assure recover data in original sequence [1.2].

There are 3 tasks of the protocol:

- establish session and offers the current endpoint context to server
- submit service request and receive returned data
- end session

Figure 1 shows a typical session, includes session starting and ending, and some service requests/responses. Corresponding to these 3 tasks, 2 types of packages are needed, one is used to establish and end sessions, the other is used to carry service requests and responses. Every package is a SMS message, to keep the session traceable and according the sequence of sending and receiving, each package has a serial number. Because the system also allow user edit and send SMS message manually, and those manually edited message should not be expected to have correct serial number like the automatic generated ones, so if a package do not have serial number, it will be handled as is, without premising in correct sequence. The packages with no serial number can not be used to establish or end session, update contexts, choose service to entry, or to do all other platform specific job, but can be used to carry service requests, whether it is acceptable is leaved to the service it requests.

Serial number should always stay at the beginning 4 character, starts with '#', followed by 3 digits, that means the largest serial number is 999. Statistically, most of user sessions won't use so many packages to make serial number overflow. If overflow really occurs, it still doesn't matter, serial number simply start from 0 again, we simply regards serial 000 is larger than 999, there will be almost zero chance to disturb communication sequence. If a package is sent more than once (often caused by mobile network provider), the duplicated ones will have the same sequence number, if received more than 1 packages which have the same serial number, the package latest received will be kept and the earlier ones will be simply discarded. Unlike most communication protocols, there is no ACK package, the reliability of communication is assured by SMS itself. If ACK is really needed, ACK words can be put directly after the serial number, each ACK word starts with the prefix 'R', followed by 3 digits represents the replied package's serial number, nothing should be put between each pair of neighboring ACK words and between serial number and ACK words. A package with ACK words may look like this (Figure 2):

1	11	21	31	40
#nnnRnnnRn	nnRnnnXXXX	XXXXXXXXXX	XXXXXXXXXX	

Fig. 2. The response package structure

Here 'n' represent digit, as well as 'X' represents data.

The package's length is not fixed. And no method is designed to check its length, assurance of the integrity of package is leave to SMS.

3.2 Voice Service

The mobile communication protocol is not fit for transmit long text, and the mobile device itself is often not fit for read much text. If large text needed to be delivered to user, voice is a reasonable choice. Along with SMS, voice is the only other communication method which all mobile device must supports. Regarding that voice can not be formalized, complicate interaction can not be carried by voice service, the only job fit for voice is deliver text.

The voice service offered by platform is quite simple. Every user by default will own a voice box, at most 9 voice messages can stored in it, each of them has a title and content. If the box is already full, coming of new message will overwrites the oldest one without warning or confirmation. Client software offers a user interface to manage user's voice box, user can see message title list and delete message through this interface, but can not get message contents. To get contents, user must call the number of voice service, and then voice service will read out each message title, starts with a serial number, user push the digit button of the serial number will make voice service read the message with that number to user.

3.3 Client Software Developed with J2ME

Simply say, client software's job is to translate user's operations into calling to services, server finish those calling and return data to client, then client generate an user interface to display the returned data as while as to prepare user's further operation which using some of those data as input. The first, client must be able to communicate with server, so it must have mobile communication protocol layer, client software can detect mobile device's capability of communication automatically, and decide the most suitable communication method, user can also manually set communication method.

Client software must very flexible, and can adapt to all kinds of limits of all different mobile devices. All user interfaces can adjust automatically to fit for the device's capacity. Most of the user interfaces are generated automatically, so that services can be added dynamically, without updating the client software. There are some basic operation interfaces which are not automatically generated, they are client software configuration screen, user profile management screen, voice box management screen, service selection screen. After entering a service, all user interfaces are generated based on the service description and function list of the service. To minimize communication, service description is not strictly formalized, the grammar is very simply, many defaults are automatically applied while no declaration presents. For example, the only allowed data type is number and string, if a piece of data comply with any digital format, it will be regard as digits, otherwise it will be look as string, so that the extra data for data type is not needed[3].

3.4 The 3 Layered Server End Software

Server end is consisted by 3 layers, mobile protocol layer, basic platform service layer, and service composition layer.

At server side there is a counterpart mobile protocol layer handling communications via mobile protocol, but the implementation of this layer is not the same as client side. Unlike the client, server is not naturally support mobile communications; it must connect to some mobile capable devices to extend its mobile capacity. And then, the protocol layer divided into 2 parts, running on the server and its mobile extension devises respectively.

Basic platform service provides basic functions such as user profile management, user activity tracing, personalized sorting and filtering, voice box management, and service management.

The core function of the platform is to manage services, these services are described in OWL-S documents, and ready to be called by the client side. Each service has a name and a description. User can regard each service as an entry point to access a certain kind of resources, by calling that service, user can access the resources the service provides. A simplest service can only provide a resource list and detail data of each resource. In more complicate occasion, category and search are provided to assist user to locate resource mo efficiently. If resources can be created, changed or removed by user, the service should also include CRUD functions. Beside these, a service should not include any unnecessary function. This limit of simplicity on service simplified user interface generation. If a service originally provides more than this, it should be simplified first, before added to platform [4].

The platform provides a general propose filtering and sorting function based on user's personal knowledge structure and current interest point. An OWL document is maintained for each user to records the knowledge the user may have. User's knowledge is inferred from user's activities, so do to user's current interest point. Personalization is implemented by another independent software package, and further discussion on personalization is beyond the scope of this paper. What make sense to the platform is that personalization helps significantly reducing the communicated information, and interactive rounds.

3.5 Web Service Composition with OWL-S

The server does not care about where the OWL-S description comes from, and whether or not it is correct or effective, those jobs are handled by other systems. The server just use these OWL-S descriptions, call web services according to them, if something failed, a standard error message will be shown.

Services is added into platform through service management user interface, this is a web based UI and can not be accessed by common users. Each service has a group of processes, OWL-S describes how to launch these processes. At client side, while entering a service, there is a main menu shown to user, lists all the processes the service had. By select the menu item, client side sends command which at server side launches the corresponding process. At every step of the process triggered, if some parameter is not available from environment and previous outputs, that means user input is needed, at that time, a user interface to input those data is generated. Thanks to the limit of simplicity, all these processes are simple, only sequence flow can appear. So that the

platform need to support only sequence control flow, and can regardless loops or branches the OWL-S processes may have [5,6].

4 Conclusions

By introducing mobile communication protocol, make the platform can cover 100% of mobile users. By introducing web service composition, make it easier to enrich service contents available to mobile users. All these will attract more mobile users and make the mobile technology valuable to agriculture production.

Acknowledgement

The research was supported by the national 863 project, Mobile intelligence service for agricultural scientific & technical information (project code 2007AA10Z236) and special fund of basic commonweal research institute project of information institute of CAAS.

References

[1] Ortiz, E.: The MIDP 2.0 Push Registry (January 2010), http://blog.csdn.net/memhoo/archive/2008/03/02/2139611.aspx

[2] JSR 120 Expert Group: Wireless Messaging API (WMA), JSR 120, JSR 205. SUN corporation

[3] Mahmoud, Q.: Getting Started With the MIDP 2.0 Game API. [EB/OL] (September 2005), http://developers.sun.com/mobility/midp/articles/gameapi

[4] Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly Media Inc., Sebastopol (2006)

[5] Saadati, S., Denker, G.: An OWL-S Editor Tutorial. [EB/OL] (May 2010), http://owlseditor.semwebcentral.org/documents/tutorial.pdf

[6] W3C Member Submission: OWL-S: Semantic Markup for Web Services [EB/OL] (September 2004), http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/