논문 2008-45TC-8-12

이동전화를 이용한 개방형 인터페이스 기반의 요금 지불 서비스 제공 방법

(Method of Fare Payment based on Open API using Mobile Phone)

임 선 환*, 이 재 용**, 김 병 철**

(Sunhwan Lim, Jaeyong Lee, and Byungchul Kim)

요 약

본 논문에서는 요금 지불 서비스를 위한 기능 구조를 설계하였다. 이는 IT 개발자들이 통신망 자원을 이용하여 과금 관련 응용을 손쉽게 만들 수 있도록 하였다. 운용자들이 통합 빌링을 제공할 수 있도록 하는 비즈니스 모델을 지원하기 위해서 지불 인터페이스와 계좌 관리 인터페이스는 필수적이며, 나아가 시장성장에 중요한 역할을 한다. 이 모델을 이용하여 지불 서비스와 계좌 관리 서비스로 구성되는 요금 지불 서비스를 생성하였다. 각각의 서비스는 Parlay X 웹서비스 구조를 기반으로 하였다. 지불 모델링과 계좌 관리 모델링 과정에서, Parlay X 지불 웹서비스에서 Spit Charging 관련 신규 오퍼레이션을 제안하였고, Parlay X 계좌 관리 웹서비스에서 Notification 관련 신규 오퍼레이션을 제안하였다.

Abstract

In this paper, functional architecture for fare payment is designed that enables IT developers to create charge-related applications using telecommunications network elements. To support a business model that enables operators to offer integrated billing, a payment and an account management API is crucial. Using this model, a fare payment service consisting of a payment service and an account management service was created. Each service is based on the architecture of Parlay X web services[5]. For the modeling of payment and of account management, the new operation of "split charging" is required in Parlay X payment web services, as is the new operation "notification".

Keywords: Parlay X, Open API, Payment, Account Management

Ⅰ. Introduction

Telecommunications networks continually evolve in terms of their form of integrated or converged architecture. From the viewpoint of service, integration between the wire and the wireless services is also a current issue. This type of integration would imply that the end user is provided with seamless broadband multimedia services between wire and wireless networks using the same terminal. The current telecommunications service market is saturated, however. The integration between wire and wireless services provides the opportunity for a new of level services with subscribers using the broadband capability of wired service coupled with the mobility of wireless. The integration between wire and wireless services includes a number of concrete examples that have been developed. Open API (Application Programming Interface) can be easily used to implement or provide

* 정회원, 한국전자통신연구원
(ETRI)
** 평생회원, 충남대학교 정보통신공학부
(Chungnam National University)
접수일자: 2008년4월29일, 수정완료일: 2008년8월12일

integration between wire and wireless services.

The Parlay Group defines an API based on CORBA (Parlay/OSA API) and an API based on web services (Parlay X web services) that enables third-party applications to make use of network functionalities[4~5, 7]. Open API is a set of open, standardized interfaces between an application and a telecommunications network. This technology can provide a range of services for the integration of wire and wireless systems independently from network infrastructures, operating systems, or developing languages. The interaction between an application incorporating Parlay X web services and a server implementing Parlay X web services is done with an XML-based message exchange[6]. Parlay X web services follow simple application semantics that allow the developer to focus on access to telecom capabilities using common web-services programming techniques[5].

In this paper, functional architecture for fare payment is designed that enables IT developers to create charge-related applications using telecommunications network elements. The designed fare payment architecture is based on the architecture of Parlay X web services. It was implemented and tested on an IBM RAD (Rational Application Developer). To model the functional architecture, new operations are proposed, and scenario flows are illustrated for them.

This paper is organized as follows: Section 2 briefly describes open API. Section Ⅲ outlines the definition of the payment SCF (Service Capability Feature) and the account management SCF. Section Ⅳ describes the designed fare payment architecture. Section Ⅴ details the implementation of the prototype function and Section Ⅵ is the conclusion.

Ⅱ. Open API

The Parlay Group is a consortium formed to develop open, technology-independent APIs that enable the development of applications capable of operating across converged networks. In this section, the Parlay/OSA (Open Service Access) API and Parlay X API are briefly described. A more detailed description of the Parlay API is available in the literature[1~2, 4, 7].

1. Parlay/OSA API

Parlay/OSA APIs are designed to enable creation of both telephony applications and "telecom-enabled" IT applications[4, 8]. IT developers, who develop and deploy applications outside the traditional telecommunications network space and business model, are viewed as crucial for creating dramatic whole-market growth in next generation applications, services, and networks. An overview of the logical entities involved in Parlay/OSA is illustrated in Fig. 1. The main functions of each element are described below.

- Application: To use network capabilities with APIs.
- Application Server (AS): To contain applications; this is the client side of an API.
- Framework: To relate applications and SCSs And to manage SCFs.
- Service Capability Feature (SCF): Capabilities (i.e. APIs) such as Call Control or User Interaction.
- Service Capability Server (SCS): To contain SCFs; this is the server side of an API.
- Gateway: Physical entity to contain the framework and SCS.

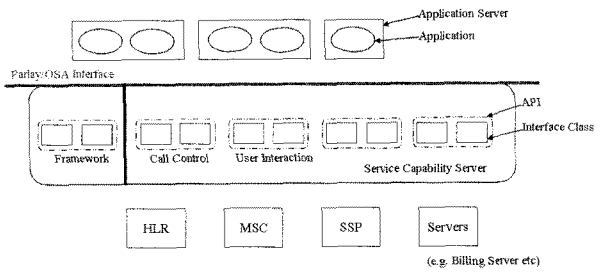

그림 1. Parlay/OSA 논리적 개체 구성도
Fig. 1. Overview of the logical entities involved in Parlay/OSA.

2. Parlay X API

Parlay X Web Services are intended to stimulate the development of next-generation network

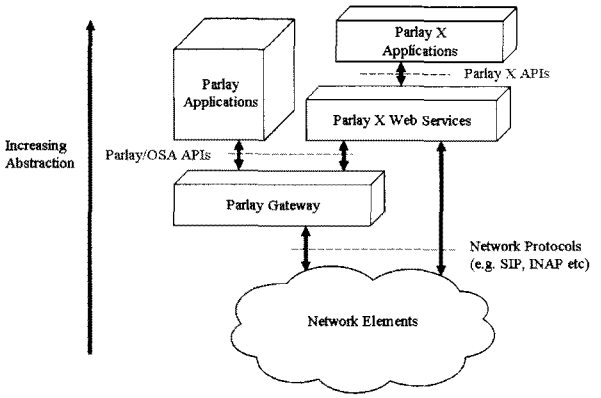

그림 2. Parlay X와 Parlay/OSA API 관계도
Fig. 2. Relationship between Parlay X and Parlay/OSA APIs.

applications by developers in the IT community who are not necessarily experts in telephony or telecommunications[7]. The selection of Web Services should be driven by commercial utility and not necessarily by technical elegance. The goal is to define a set of powerful yet simple, highly abstracted, imaginative, telecommunications capabilities that developers in the IT community can both quickly comprehend and use to generate new, innovative applications.

Each Parlay X Web Service should be abstracted from the set of telecommunications capabilities exposed by the Parlay/OSA APIs, but may also expose related capabilities that are not currently supported in the Parlay/OSA APIs where there are compelling reasons[5]. These tiered levels of abstraction and the Parlay/OSA - Parlay X relationship are illustrated in Fig. 2.

III. Payment and Account Management SCF

1. Payment SCF

Parlay X web services allow the third-party applications to support payments of any content in an open, web-like environment through the payment API. Interfaces for payment functionalities (including sixteen operations) are termed AmountCharging, VolumeCharging, ReserveAmountCharging, and ReserveVolumeCharging[1]. The charge for contents is illustrated in Fig. 3.

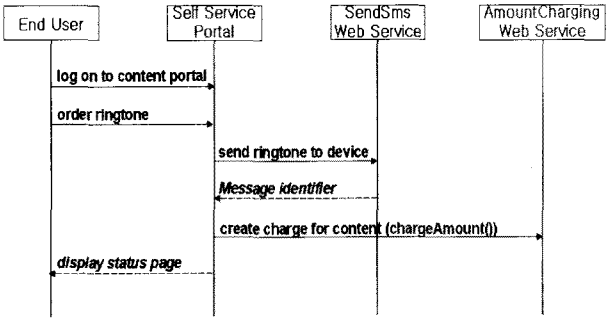

그림 3. 컨텐츠 지불 메시지 흐름도
Fig. 3. Charge for content.

2. Account Management SCF

Parlay X web services allow third-party applications to manage an account through an account management API. The account management interface can display account balances or obtain the transaction history of an account and can recharge an account as well. The interface for account management functionalities (including twelve operations) is termed AccountManagement and so on[2]. A prepaid account recharge is illustrated in Fig. 4.

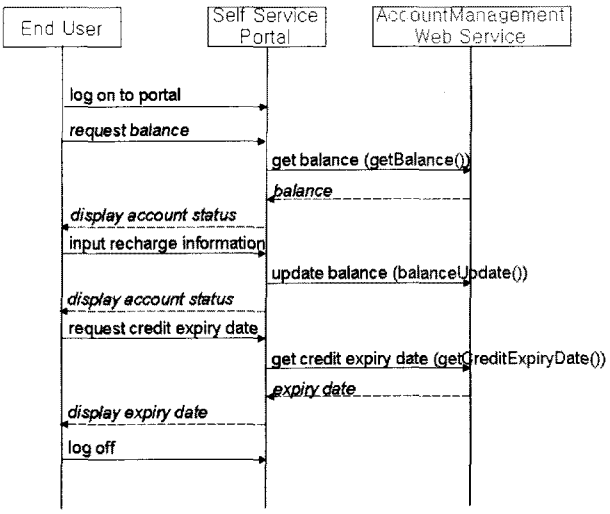

그림 4. 계좌 충전 메시지 흐름도
Fig. 4. Prepaid account recharge.

IV. Designed Fare Payment Architecture

1. Detailed Payment Functional Architecture

Detailed functional blocks of the payment SCF are illustrated in Fig. 5. This architecture consists of the main block, manager block, and protocol adaptor

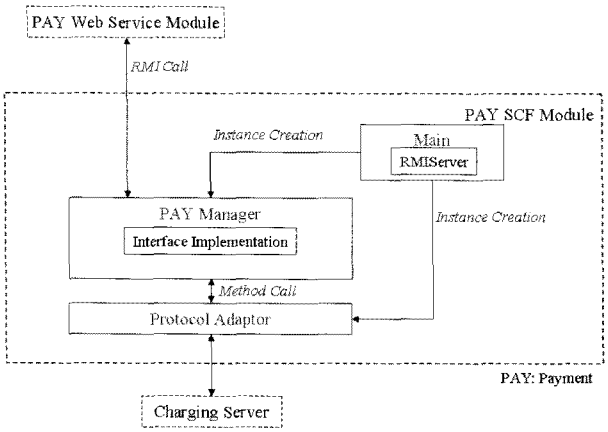

그림 5. Payment 기능 구조도
Fig. 5. Functional architecture of the payment SCF.

block. The main functions of each element are described below.

- The main block is the main thread of the SCF. This block creates the RMI server that processes RMI requests and registers the RMI server with the RMI registry. This block also creates what is known as a manager instance and a protocol adaptor instance. If the created RMI server is invoked from web service module, this server finds the manager instance.
- The manager block implements the payment interface class. This block invokes related operations in order to perform functionalities and then records the transaction history information into a database in a gateway. Related operations (sixteen) include the chargeAmount, refundAmount, getAmount, chargeVolume, refundVolume, reserve Amount, reserveAdditional Amount, reserveVolume, reserveAddtional Volume, chargeReservation, releaseReservation and so on. This block interacts with protocol adaptor using method call.
- The protocol adaptor block receives a request from the manager and forwards the request to a charging server. The results from the charging server are forwarded to the manager.

In parlay x payment web services, the addition of a split charging operation with refund and reservation is proposed in order to enhance the usefulness of the applications. If a user cannot make payments from one reservation (for example if the reservation balance is low) and the sum of multiple reservations of one or several users makes it possible to meet the payment, then this split charging method becomes useful. Split charging creates the ability to make payments from multiple reservations simultaneously. In addition, it can enhance the quality of applications.

```
Interface AmountCharging {
…     //Basic Operations
refundSplitAmount(splitInfo[], charge,
referenceCode);
};

Interface VolumeCharging {
…     //Basic Operations
refundSplitVolume(splitInfo[], volume, billingText,
referenceCode, parameters[]);
};

Interface ReserveAmountCharging {
…     //Basic Operations
reserveSplitAmount(splitInfo[], charge);
reserveAdditionalSplitAmount(splitInfo[], charge)
chargeSplitReservation(splitInfo[], charge,
referenceCode)
};

Interface ReserveVolumeCharging {
…     //Basic Operations
reserveSplitVolume(splitInfo[], volume,
billingText, parameters[]);
reserveAdditionalSplitVolume(splitInfo[], volume,
billingText);
chargeSplitReservation(splitInfo[], volume,
billingText, referenceCode)
};
```

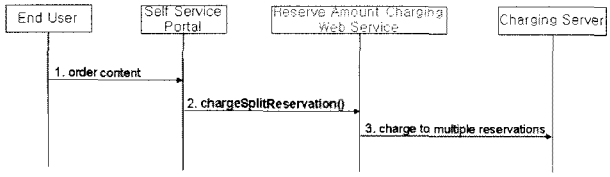

그림 6. 분할 과금 관련 메시지 흐름도
Fig. 6. Flow of charges with split charging.

The flow of a scenario with split charging is illustrated in Fig. 6. Some entities are omitted in this figure to simplify the diagram.

- Assuming that subscribers are interested in receiving the stream of a soccer match and sharing the bill, the subscribers decide to stream the match. The match starts and the provider periodically invokes a chargeSplitReservation operation through the reserve amount charging API. The reserve amount charging web service then charges multiple reservations (chargeSplit Reservation()).

2. Detailed Account Management Functional Architecture

Detailed functional blocks of the account management SCF are illustrated in Fig. 7. This architecture consists of the main block, manager block, and protocol adaptor block. The main functions of each element are described below.

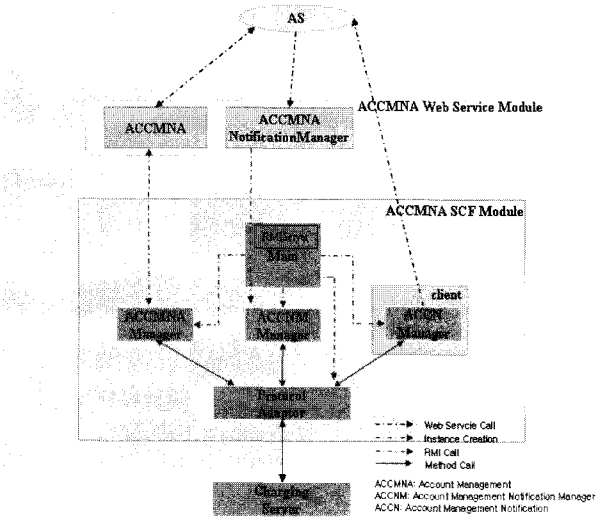

그림 7. Account Management 기능 구조도
Fig. 7. Functional architecture of the account management SCF.

- The main block is the main thread of the SCF. This block creates the RMI server that processes RMI requests and registers the RMI server with the RMI registry. This block also creates what is known as a manager instance and a protocol adaptor instance. If the created RMI server is invoked from web service module, this server finds the manager instance.
- The manager block implements the account management interface class. This block invokes related operations in order to perform functionalities and then records the transaction history information into a database in a gateway. Related operations (twelve) include getBalance, getHistory, getCreditExpiryDate, balanceUpdate, voucherUpdate, getBalanceTypes and so on. This block interacts with protocol adaptor using method call.
- The protocol adaptor block receives a request from the manager and forwards the request to a charging server. The results from the charging server are forwarded to the manager.

In parlay x account management web services, the addition of a notification operation with AccountManagementNotificationManager and AccountNotification is proposed in order to enhance the usefulness of applications. The notification operation is capable of providing information about an account that has been changed by some applications (e.g. Multimedia Service or WAP/WEB pages) to other applications (e.g. by SMS or MMS) after account_date_low notices. In addition, it can enhance quality of applications by allowing subscribers to confirm information about account_date_low notifications in real time.

표 1. 계좌 변경 이벤트 값 목록
Table 1. AccountChangedEvent Enumeration.

Enumeration Value	Description
... //*Basic Values*	... //*Basic Descriptions*
AccountDateLow	Account credit expiry date is below the threshold

Interface AccountManagementNotificationManager
{
… //Basic Operations
changeNotifcation(correlator, criteria);
getNotification(correlator)
};

Interface AccountNotification {
… //Basic Operations
accountDateLow(correlator, balanceExpireDetails);
accountError(correlator, endUserIdentifier, reason);
};

The flow of a scenario with the notification operation is illustrated in Fig. 8. Some entities are omitted in this figure to simplify the diagram.

- In Fig. 8, notifications of account changes are made available to the SMS provider (application A) (startNotification()).
- Assumed here is that a subscriber is interested in receiving a stream of a soccer match (application B). The subscriber decides to stream the match to his MS (Mobile Station). The match starts and the provider periodically charges the account. When a monitored account is charged, a notification is delivered to the SMS provider with the new account information in real time.
- The SMS provider sends the subscriber an SMS message confirming the correctness of the payment.
- When a monitored account date is below a date threshold, a notification is delivered to the SMS provider with the new account information in real time (accountDateLow()).
- The SMS provider sends the subscriber an SMS message confirming an account_date_low.
- Assuming that the prepaid subscriber interacts with a web page to recharge (application C), the provider then recharges the account.
- When a monitored account is recharged, a notification is delivered to the SMS provider with the new account information in real time.
- The SMS provider sends the subscriber an SMS message confirming the correctness of the recharges.
- The SMS provider sends a notification (endNotification()).

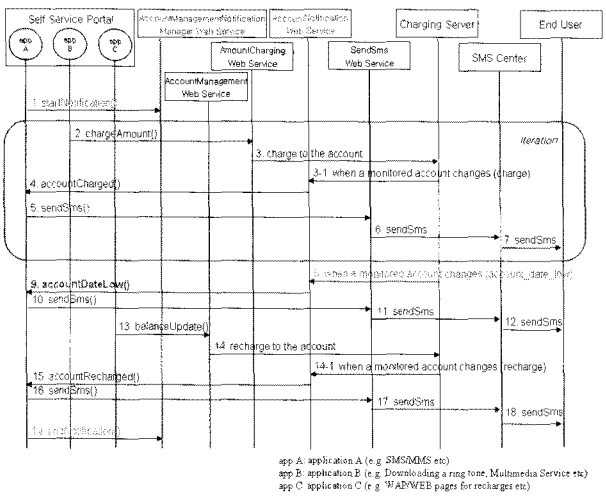

그림 8. 통지 관련 메시지 흐름도
Fig. 8. Flow of accountDateLow with the notification operation.

3. Functional Blocks for Fare Payment

Functional blocks of open service application server and gateway for fare payment using open API are illustrated in Fig. 9 and network architecture is illustrated in Fig. 10. From here, we can know that open service application server provides service user with payment UI (User Interface). And it also stores

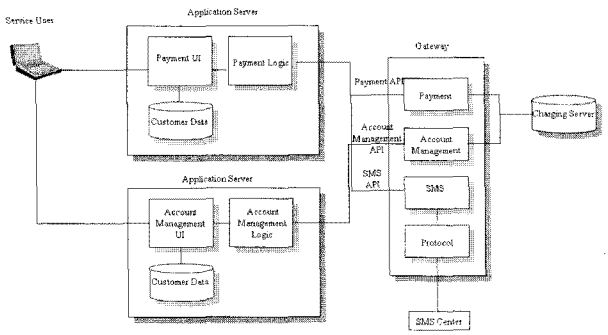

그림 9. 요금 지불 서비스 기능 블록도
Fig. 9. Function blocks for fare payment.

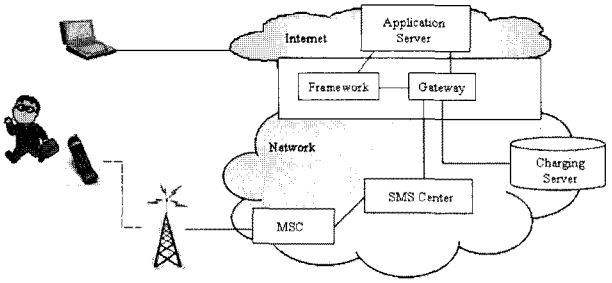

그림 10. 요금 지불 서비스 망 구조도
Fig. 10. Network architecture for fare payment.

the customer data for service user subscription management. If service user requests the fare payment, payment logic in open service application server requests the act for payment of open service gateway using payment API. Open service gateway payment functionality performs the request from payment logic. And then the result is applicable to charging server. The charging server stores payment transaction data and log.

Open service application server provides service user with payment transaction history retrieving functionality. From here, we can know that open service application server provides service user with payment transaction history retrieving UI. And it also stores the customer data for service user subscription management. If service user requests the payment transaction history, account management logic in open service application server requests the act for payment transaction history of open service gateway using account management API. Open service gateway account management functionality performs the request from account management logic. And then the result forwards the payment transaction history in charging server to account management logic in open service application server. Payment transaction history forwarded to account management logic in open service application server is provided with service user through account management UI.

4. Payment Scenario Flow for Fare Payment

Payment scenario flow for fare payment service is illustrated in Fig. 11. If fare payment is requested from service user in init state, it receives service user mobile phone number and also receives

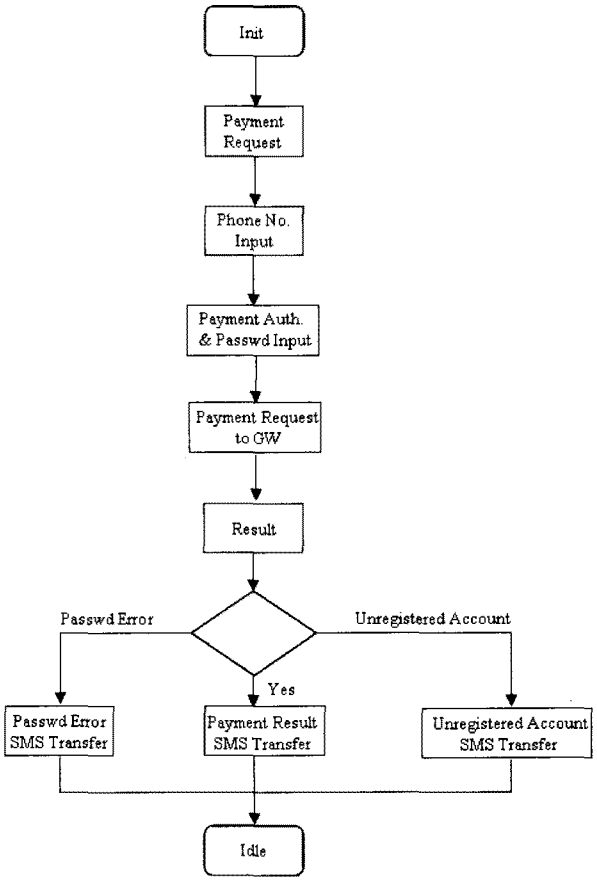

그림 11. 요금 지불 서비스 지불 흐름도
Fig. 11. Flow of payment for fare payment.

password and payment authority. Using input data, it forwards payment request to open service gateway in telecommunication network. If the result is YES, it sends the result using SMS to mobile phone. If service user's password is wrong, it sends password error message using SMS to mobile phone. If the account is unregistered, it sends unregistered account message using SMS to mobile phone.

5. Account Management Scenario Flow for Fare Payment

Account Management scenario flow for fare payment service is illustrated in Fig. 12. If fare payment transaction history is requested from service user in init state, it receives service user account number and also receive password. Using input data, it forwards payment transaction history request to open service gateway in telecommunication network. If the result is YES, it displays the payment transaction history. If service user's password is

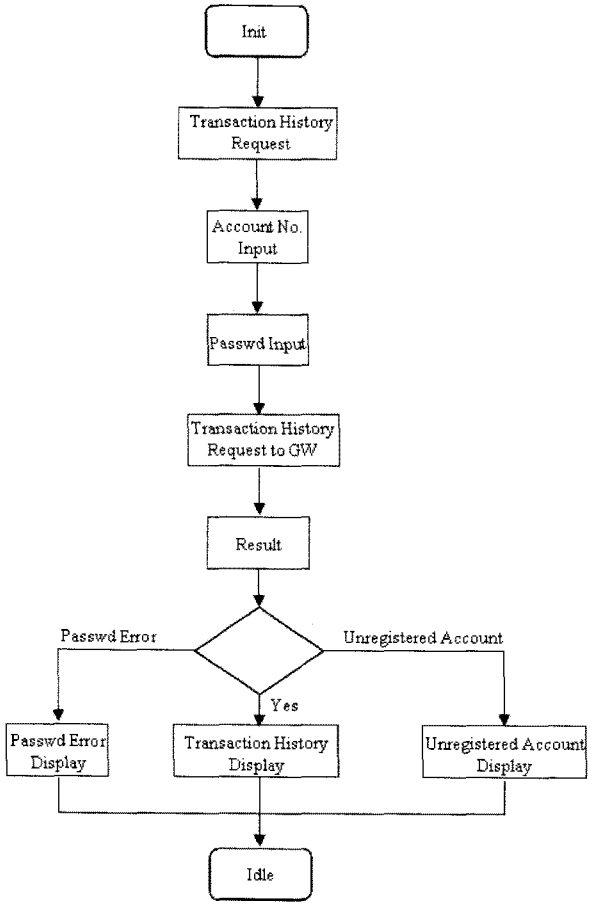

그림 12. 요금 지불 서비스 계좌 관리 흐름도
Fig. 12. Flow of account management for fare payment.

wrong, it displays password error message. If the account is unregistered, it displays unregistered account message.

V. Implementation of the prototype function

1. Environment and Testing

The designed fare payment was implemented and tested on an IBM RAD (Rational Application Developer) using the Java language. The created payment and account management web service modules were loaded on an IBM WAS (WebSphere Application Server, H/W : Sun Server, O/S : Solaris). Payment and account management web services consisted of a web service module and a SCF module. The web service module interacts with the SCF module using RMI, which implies that a security policy between these modules is needed. However, a security policy is in fact unnecessary, as the two modules are parts of the same system. Payment and account management web services interact with an Altibase DB to record transaction history information using JDBC (Java Database Connectivity) and with the charging server for the charging or billing information.

In this test, a fare payment service was used that consisted of a payment service, an account management service. The created services were implemented and tested on Microsoft Visual Studio 2005 using the C# language.

VI. Conclusion

The current telecommunications market is saturated. Regarding new market growth, a range of new intelligent services is on the horizon. Potential subscribers must be introduced to these services, but it is currently not feasible to bring third-party service providers and developers into the vertical architecture of current telecommunications networks. Thus, open, technology-independent APIs that enable the development of applications that operate across converged networks are necessary.

In this paper, functional architecture for fare payment is designed that enables IT developers to create charge-related applications with telecommunications network elements. The designed fare payment architecture is based on the architecture of Parlay X web services. It was implemented and tested on an IBM RAD. For the modeling of the functional architecture, new operations were proposed and scenario flows were illustrated for them. The validation of the designed fare payment model was processed. Specifications regarding the detailed fare payment architecture along with the addition of new operations in Parlay X payment web services and new operations in Parlay X account management web services were obtained.

References

[1] ETSI ES 202 504-6 v0.0.4: Open Service Access

(OSA); Parlay X Web Services; Part 6 Payment (Parlay X 3) (2007-06)
[2] ETSI ES 202 504-7 v0.0.4: Open Service Access (OSA); Parlay X Web Services; Part 7 Account Management (Parlay X 3) (2007-06)
[3] ETSI ES 202 504-4 v0.0.3: Open Service Access (OSA); Parlay X Web Services; Part 7 Short Messaging (Parlay X 3) (2007-06)
[4] ETSI ES 204 915 v0.0.1: Open Service Access (OSA); Application Programming Interface (API) (Parlay 6) (2007-03)
[5] Parlay X Working Group: Parlay X Web Services White Paper v1.0 (2002)
[6] Web Services Working Group: Parlay Web Services WSDL Style Guide (2002)
[7] Parlay X Working Group: Parlay X Web Services Specification v1.0 (2003)
[8] Ard-Jan Moerdijk and Lucas Klostermann, Ericsson Eurolab Netherlands: "Opening the Networks with Parlay/OSA: Standards and Aspects behind the APIs", IEEE Network (2003)
[9] WeiWu, Hua Zou, Fangchun Yang, "Design OSA/Parlay Application Frameworks Using a Pattern Language", Proceedings of ICCT (2003)
[10] Karsten Luttge: "E-Charging API: Outsource Charging to a Payment Service Provider", IEEE Intelligent Network Workshop (2001)
[11] J.W. Hellenthal, F.J.M. Panken, M. Wegdam: "Validation of the Parlay API through Prototyping", IEEE Intelligent Network Workshop (2001)
[12] PayCircle, http://www.paycircle.org
[13] OMA MCC, http://www.openmobilealliance.org

저 자 소 개

임 선 환(정회원)
1997년~현재 한국전자통신연구원 융합기술연구부문 융합서비스플랫폼연구부
<주관심분야 : 이동통신 네트워크, 개인화 서비스>

이 재 용(평생회원)
1988년 서울대학교 전자공학과 학사
1990년 한국과학기술원 전기 및 전자공학과 석사
1995년 한국과학기술원 전기 및 전자공학과 박사
1990년~1995년 디지콤 정보통신 연구소 선임연구원
1995년~현재 충남대학교 정보통신공학부 교수
<주관심분야 : 초고속통신, 인터넷, 네트워크 성능분석>

김 병 철(평생회원)
1988년 서울대학교 전자공학과 학사
1990년 한국과학기술원 전기 및 전자공학과 석사
1996년 한국과학기술원 전기 및 전자공학과 박사
1993년~1999년 삼성전자 CDMA 개발팀
1999년~현재 충남대학교 정보통신공학부 부교수
<주관심 분야 : 이동인터넷, 이동통신 네트워크, 데이터통신>