TECHNICAL UNIVERSITY OF CLUJ-NAPOCA
FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

A Payment Service Provider for Mobile Users

- Diploma Thesis -

ADVISOR
Senior Lecturer Eng. Cosmina IVAN

GRADUATE
Andrei SZABO

TECHNICAL UNIVERSITY OF CLUJ-NAPOCA
FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

FACULTY DEAN
Prof. Dr.Ing. Sergiu NEDEVSCHI

HEAD OF DEPARTMENT
Prof.Dr.Ing. Kalman PUSZTAI

A Payment Service Provider for Mobile Users

Graduate: Andrei SZABO

1. CONTENTS:
   A. Written Sections: Introduction, Mobile Technologies, Mobile Payments, The Specifications and Architecture of the Payment Provider Framework, Framework Implementation, User Guide, Evaluation and Conclusions, Bibliography
   B. Appendices: Entity Relationship Diagram, UML Sequence Diagrams, CD containing the application source code plus paper in electronic format
2. DOCUMENTATION PLACE: Technical University of Cluj-Napoca
3. ADVISORS: Senior lecturer Eng. Cosmina IVAN
4. ISSUE DATE: October 2003
5. DUE DATE: June 21st, 2004

ADVISOR
Senior Lecturer Eng. Cosmina IVAN

GRADUATE
Andrei SZABO

Table of Contents

Table of Figures

1. Introduction

Payment has evolved through time from the exchange of physical tokens, such as coins and notes, to writing checks, to changing payment card details over the phone or over the internet. This evolution has involved a shift from the exchange of tokens of fixed value to the exchange of information between the participating parties in the transaction. In the case of payment cards, the exchange of information is carried out between the consumer and merchant's banks and goes on over networks managed either by regional payment providers or by global card organizations.

The emergence of e-commerce has taken the payment process even further, making it fully digitized and requiring no physical contact between the involved parties. Virtual payments brought enormous benefits to sellers and customers but and also placed extra pressures on payment service providers, including banks, credit card companies and mobile operators; these pressures are mostly related to robust security and interoperability issues. High-speed mobile data networks have opened up a new channel for e-commerce, while the advent of mobile payments added a new layer of complexity by using devices with limited capabilities.

According to [1], a mobile payment is defined as the process of two parties exchanging financial value using a mobile device in return for goods or services. A mobile device for the purposes of this paper defines a wireless communication device, including mobile phones, PDAs, wireless tablets, and mobile computers. The success of mobile payments relies on the same factors that have fuelled the growth of physical world non-cash payments, namely: security, interoperability, privacy, global acceptance and ease of use.

Financial institutions, independent vendors and mobile network operators currently provide mobile payments. Many differences exist between these proprietary implementations. Although there are a number of organizations that have been set up to develop common standards for mobile payments, none of the approaches has reached the level of acceptance of a common standard.

This paper outlines the background research into the field of mobile technologies and the mobile payment domain, as a prerequisite for the development of a solution for mobile payments. A technical overview of existing mobile network technologies and mobile

communication services is followed by a description of the generic features of a mobile payment system, including some mobile payment principles and payment types.

Following the research in the mobile payment domain, a solution for a payment service provider was realized. The distributed nature of Web Services was selected to provide the extensibility and interoperability requirements of this framework. This solution allows integration over existing Internet protocols to current mobile payment infrastructures.

This paper than describes the implementation of the Web Services framework for two different payment approaches, to represent the different payment techniques in the mobile payment domain. The realization of the framework implementation is illustrated through the development of a payment system. This system attempts to establish the complex relationships that exist between the actors in the mobile payment system. It describes the integration of Web Services into a real mobile network system, using a SMS gateway and an application server.

The developed payment system provides a web interface that allows the clients to register with the system and exposes a set of methods as Web Services; these can be called by registered clients for payment initialization, authorization, charging and finally, payment settlement purposes. For testing and demonstration purposes, a simple content provider's web site was also developed; this site sells digital goods online and uses the developed payment system for charging purposes.

2. Mobile Technologies

The concept of cellular communications was conceived in Bell Laboratories during the late 1960s. However, the technology necessary for commercial service did not exist at the time. The first operational cellular networks were installed in the 1980s in Sweden, Denmark, Norway and Finland. These networks were based on an analogue cellular standard known as Nordic Mobile Telephone (NMT). Less than two years later, cellular services were launched in the United States using AMPS (Advanced Mobile Phone System) technology. Today, over 100 member countries of the International Telecommunication Union (ITU) have licensed one or more cellular operators to provide mobile services. [7]

2.1 Mobile Network Technologies

There have been a number of different technologies developed to provide mobile cellular service. These can be classified as first generation (1G), second generation (2G), 2.5G and third-generation (3G) technologies. The following sections describe shortly these technologies based on the generation they were part of.

2.1.1 1G Technologies

1G mobile phones were based on the analogue system. The introduction of cellular systems in the late 1970s was a quantum leap in mobile communication, especially in terms of capacity and mobility. Semiconductor technology and microprocessors made smaller, lighter, and more sophisticated mobile systems a reality. However, these 1G cellular systems still transmitted only analogue voice information. The prominent ones among 1G technologies were advanced mobile phone system (AMPS), Nordic mobile telephone (NMT), and total access communication system (TACS). With the introduction of 1G phones, the mobile market showed annual growth rate of 30 to 50 per cent, rising to nearly 20 million subscribers by 1990.

2.1.2 2G Technologies

2G phones using global system for mobile communications (GSM) were first used in the early 1990s in Europe. GSM provides voice and limited data services, and uses digital modulation for improved audio quality. The development of 2G cellular systems was driven by the need to improve transmission quality, system capacity, and coverage. Further advances in semiconductor technology and microwave devices brought digital transmission to mobile communications. Speech transmission still dominates the airways, but the demand for fax, short message, and data transmission is growing rapidly. Supplementary services such as fraud prevention and encryption of user data have become standard features, comparable to those in fixed networks. 2G cellular systems include GSM, digital AMPS (D-AMPS), code-division multiple access (CDMA), and personal digital communication (PDC).

GSM is an open, non-proprietary system that is constantly evolving. One of its strengths is its international roaming capability. Today GSM offers consumers a same-number phone with seamless access in more than 161 countries. GSM satellite roaming has also extended service access to where terrestrial coverage is not available. The first GSM networks began operation in 1991 and continued to expand beyond Europe to become a world standard. In 2001, the GSM Association represents the interests of more than 600 GSM, satellite and 3rd Generation networks operators, regulator and administrative bodies and key manufacturers/suppliers to the GSM industry in 174 countries. There were more than 646 million GSM subscribers at the end of 2001.

GSM offers users a variety of advanced features and services. These include circuit-switched voice, fax, and data, as well as voicemail and voicemail notification. Additional services include wireless application protocol (WAP), high-speed circuit- switched data (HSCSD), mobile location services (MLS), and cell broadcast. From the outset, GSM has been a system designed with stringent levels of inbuilt security. With constantly enhanced transmission protocols and algorithms added to the flexible and 'future proof' platform, GSM remains a very secure wireless standard.

The work on GSM continues under the aegis of ETSI (European Telecommunications Standards Institute). Specifications are constantly undergoing review and change to keep up

with demand for additional services. Many telecom analysts consider GSM as the preferred platform to migrate to third generation communications systems.

HSCSD (High Speed Circuit Switched Data) [7] enables transmission of data over current circuit-switched GSM links at speeds of up to 57.6kbps. This speed is achieved by concatenating, i.e. adding together, consecutive GSM timeslots, each of which is capable of supporting 14.4kb/s. Up to four GSM timeslots are needed for the transmission of HSCSD. A permanent connection is established between the called and calling parties for the exchange of data.

As it is circuit switched, HSCSD is more suited to applications such as videoconferencing and multimedia than bursty type applications such as e-mail, which is suited to packet switched data.

2.1.3 2.5G Technologies

2.5G networks incorporate 2G technology with GPRS' higher speeds to support data transport. 2.5G is a bridge from the voice-centric 2G networks to the data-centric 3G networks.

GPRS (General Packet Radio Service) is a packet-switched service that allows data communications to be sent and received over the existing Global System for Mobile (GSM) communications network. This allows the user of a computing device with a GPRS modem to connect to the Internet and exchange data with servers and web sites for mobile access to e-mail, corporate databases, and browsing the web. With GPRS, the user will experience data rates significantly faster than a GSM circuit switched data connection. GPRS is an evolutionary step towards 3G technologies, such as EDGE (Enhanced Data GSM Environment) and UMTS (Universal Mobile Telephone Service).

Some advantages of GPRS, relative to GSM’s HSCSD, are listed in the followings: [8, 9]

- Faster Data Transfer Rates
- Always-On Connection
- Broad Application Support
- Security Support

Of course, as most other technologies out there, GPRS has also its downside. Let us see shortly some of the GPRS' drawbacks: [10]

- Limited Cell Capacity
- Speed
- Suboptimal Modulation
- Transit Delay
- No store and forward

2.1.4 3G Technologies

Third generation (3G) is the generic term for the next big step in mobile technology development. The formal standard for 3G is the IMT-2000 (International Mobile Telecommunications 2000). This standard is being pushed by different developer communities such as W-CDMA is backed by Ericsson, Nokia and Japanese handset manufacturers and CDMA2000 is backed by the US vendors Qualcomm and Lucent.

EDGE (Enhanced Data Rates for Global Evolution) is a 3G radio technology standardized by ITU and 3GPP, built on the existing GSM/GPRS network with minor additions relating to the air interface. [11]

EDGE capability can be easily upgraded to existing GSM/GPRS networks using appropriate software. GPRS on average triples the current GPRS data rates and the capacity of each hardware and frequency channel, enabling lower usage cost for existing end-user services, as well as the new advanced services delivery over GSM bands, such as video streaming. While a number of mobile operators are considering implementing EDGE as an interim data technology between GPRS and UMTS, the success of EDGE depends very much on the timely availability of the products and applications.

WCDMA (Wideband Code Division Multiple Access) is by far the most widely adopted 3G air-interface technology in the new IMT-2000 frequency bands. Standardized by 3GPP and ITU, WCDMA has gained broad acceptance within the wireless communication industry. Around 40 operators have announced plans to start commercial operation by end of year 2003. In 2005, there will be close to 100 WCDMA networks in operation. [11]

From the outset, WCDMA was designed to provide cost-efficient capacity for both modern mobile multimedia applications and traditional mobile voice services. One of the key benefits of the technology is efficient, flexible support for radio bearers, in which network capacity can be freely allocated between voice and data within the same carrier. WCDMA also supports both multiple simultaneous services and multimedia services comprising multiple components with different service quality requirements in terms of throughput, transfer delay, and bit error rate.

In WCDMA, user data is spread over a bandwidth of circa 5 MHz. The wide bandwidth supports high user data rates and provides performance benefits due to frequency diversity. The wideband carrier is also cost-effective as more users can be served by means of a single transceiver. HSDPA will increase WCDMA's peak user data rates from the currently achievable 384 kbit/s to up to 10 Mbit/s in 2005-2006 (with 3GPP release 5).

CDMA2000 is the 3rd Generation solution based on IS-95. Unlike some 3G standards, It is an evolution of an existing wireless standard. CDMA2000 supports 3G services as defined by the International Telecommunications Union (ITU) for IMT-2000. 3G networks will deliver wireless services with better performance, greater cost-effectiveness and significantly more content. The goal is access to any service, anywhere, anytime from one terminal - true converged, mobile services.

The CDMA2000 standard is evolving to continually support new services in a standard 1.25 MHz carrier. The first phase of CDMA2000, or CDMA2000 1X will deliver average data rates of 144 kbps. Phase two, labeled CDMA2000 1xEV, will provide for data rates greater than 2Mbps. [12]

Standing for "Universal Mobile Telecommunications System", UMTS represents an evolution in terms of services and data speeds from today's "second generation" mobile networks. As a key member of the "global family" of third generation (3G) mobile technologies identified by the ITU, UMTS is the natural evolutionary choice for operators of GSM networks.

Using fresh radio spectrum to support increased numbers of customers in line with industry forecasts of demand for data services over the next decade and beyond, "UMTS" is synonymous with a choice of WCDMA radio access technology that has already been selected

by approaching 120 licensees worldwide. [13]

UMTS is already a reality. Japan launched the world's first commercial WCDMA network in 2001, and WCDMA networks are now operating commercially in Austria, Italy, Sweden and the UK with more launches anticipated during 2003-2004.

2.1.5 The Future of Mobile Technologies: 4G

Fourth-generation (4G) cellular services, intended to provide mobile data at rates of 100Mbits/sec or more, were originally scheduled for 2010. Some cell phone companies have moved the target up to 2006, while rival wireless systems could bring similar bandwidth to a few fortunate networkers a lot sooner. [14]

The enthusiasm for 4G is not due to accelerated progress; it is because third-generation (3G) services have proven so disappointing. Instead of one standard worldwide, there are three incompatible systems in the United States alone. Voice is carried over the circuit-switched infrastructure inherited from second-generation (2G), not the promised IP. The touted streaming video is just a low-resolution slideshow. Most importantly, the data rates are closer to dial up than DSL.

This is partially due to the technology's immaturity: 3G systems rolled out so far could be considered beta versions, with the real thing still in the future. However, 3G will never live up to its creators' promises. Despite early excitement about data, the main economic incentive for 3G is increased capacity for narrowband voice. Though data rates will increase, there is not enough bandwidth to transfer large e-mail attachments quickly, let alone stream audio or video at broadcast quality as the cell phone vendors first claimed.

2.2 Mobile Communication Services

2.2.1 SMS

Short Messaging System (SMS), also known as text messaging or mobile messaging is a digital cellular network feature. It allows for sending of text and numeric short messages to and from digital mobile phones. The number of characters that can be sent is dependent on the language in use; currently language support is limited to European languages, Chinese and Arabic. This service is widely popular in Europe and Asia while in the US it is practically non-existent.

SMS, as defined by the GSM standard as several unique features [15]:

- A single short message can be up to 160 characters of text in length. Those 160 characters can comprise of words, numbers, or an alphanumeric combination. Non-text based short messages (for example, in binary format) are also supported. These are used for ring tones and logos services for instance.
- The Short Message Service is a store and forward service**,** in other words, short messages are not sent directly from sender to recipient, but always via an SMS Center instead
- The Short Message Service features confirmation of message delivery**.** This means that unlike paging, users do not simply send a short message and trust and hope that it is delivered. Instead, the sender of the short message can receive a return message back notifying them whether the short message has been delivered or not.
- Short messages can be sent and received simultaneously with GSM voice, Data and Fax calls. This is possible because whereas voice, Data and Fax calls take over a dedicated radio channel for the duration of the call, short messages travel over and above the radio channel using the signaling path. As such, users of SMS rarely if ever get a busy or engaged signal as they can do during peak network usage times.
- Ways of sending multiple short messages are available. SMS concatenation (stringing several short messages together) and SMS compression (getting more than 160 characters of information within a single short message) have been defined and incorporated in the GSM SMS standards.

2.2.2 MMS

Multimedia Messaging System (MMS) is one of the most recent developments in mobile messaging. Just as the traditional short messaging (SMS), MMS provides automatic and immediate delivery of personal messages. The difference with MMS comes from the fact that it allows users to enhance their messages by incorporating sound, images and other rich content. [16, 17]

MMS offers more than just a broadening of content. With MMS, it is not only possible to send multimedia messages from one mobile phone to another, but also from phone to an email address and vice-versa. A big advantage of MMS is that the message is a multimedia presentation in a single entry, not a text file with attachments, making it much simpler and user-friendly.

MMS is an open industry standard, and MMS messages can be delivered using existing networks and protocols. MMS is also bearer-independent, which means it is not limited to GSM or WCDMA networks. The speed of MMS transmission, although quick, is still dependent on the message size and on the bearer used. However, since the receiver is not aware of the ongoing transmission before the message has been delivered, the delay is imperceptible, making MMS as convenient to use as SMS.

2.2.3 WAP

Wireless Application Protocol or WAP is an accepted, open industry standard, developed by the WAP Forum, that allows wireless devices to access information and services on the Internet. Designed for use over mobile digital networks, WAP allows you to deliver Internet information to mobile phones.

Basically, a handset sends a request, using WAP protocol, to a special WAP gateway server. Upon interpretation of the address or URL in the request, the content is fetched by the gateway and transferred via WAP to the handset. The web site' information should be described in WML (Wireless Markup Language) format, WML being the mobile equivalent of HTML for web pages.

WAP certainly does not replace the use of PCs to access the Internet. It simply complements them with an “anywhere, anytime” access solution. Current WAP devices have limitations when it comes to speed, graphics and colors; this is changing with the introduction of GPRS.

WAP has been very popular in Asia, except in Japan where I-mode is dominates the market. WAP is an open standard in contrast to I-mode, which is a proprietary standard. Operators switching over to I-Mode will have to make changes to their existing infrastructure, which means high expenses. WAP is being promoted by most of the big mobile phone manufacturing companies of the world and these form 50% of the global production of mobile phones. In addition, there are difficulties with the configuration of a WAP phone for new WAP services. 20 or so different parameters need to be entered to gain access to the WAP service, which may discourage users.

2.3 Mobile Platforms

2.3.1 Mobile Operating Systems

The operating system for mobile terminals is not standardized and currently there is an ongoing battle to become the technology standard of the future. Each of the operating systems is gathering a number of application developers around them, who mostly develop their products for one OS only. There are three major players, who have each developed their own operating system.

Symbian [21] is a consortium of leading mobile handset manufacturers Nokia, Psion, Ericsson, Panasonic, Samsung, Siemens and Sony Ericsson, and was establishes in June 1998. Symbian develops and supplies the advanced, open, standard operating system – Symbian OS – for data-enabled mobile phones. The operating system, which is based on Psion’s earlier software, is especially designed for two types of wireless information devices: smart phones (mobile phones with add-on applications and PC connectivity) and communicators (handheld computers with connectivity to or built-in mobile phones).

As at December 2003 18 phones that use Symbian OS from five manufacturers are shipping worldwide (including the Nokia 6600, Fujitsu F900i, Sony Ericsson P900, Motorola A925)

and a further 26 phones from nine manufacturers are in development. During 2003, 6.7 million phones using Symbian OS shipped worldwide, and more than 1 million phones shipped in December 2003 alone.

Microsoft is a little late on pounding the opportunity provided by the mobile market. It entered into the market with its own operating system called **Microsoft Smartphone OS**. The new Microsoft Smartphone OS is a compelling and powerful addition to the Microsoft family of Mobile products like Pocket PC OS and Windows CE. Like Symbian UIQ, it combines the telephony features with the functionality of a PDA in the shape of a mobile phone. There are two different versions of the Smartphone OS currently in the market. The first is released in 2002 and called Microsoft Smartphone 2002. The Smartphone 2002 platform is based on the Microsoft CE 3.0 operating system and contains many of the core applications provided with the Pocket PC based computing devices, including email, IM and Pocket Internet Explorer. Motorola released its MPX200 cell phone in the U.S. and Europe market based on the Microsoft Smartphone 2002 operating system. The Smartphone 2003 is the latest of the two OS. Orange has already made its SPV 200 based on this platform. In devices with Smartphone 2003 OS, the .NET Compact Framework is in ROM. This Operating System is also built on the completely new architecture; it runs on Microsoft Windows CE.NET 4.2 enhancing APIs and overall programmability.

Licensed by industry leaders—including Aceeca, AlphaSmart, Fossil, Foundertech, Garmin, GSPDA, HuneTec, Kyocera, Lenovo, palmOne, PerComm, Samsung, Sony, Symbol, and Tapwave—the **Palm OS** platform is in over 33 million hardware products worldwide [23]. Its success and popularity has given rise to a huge community of users, enterprises, developers, manufacturers, wireless operators and infrastructure and middleware providers. The company behind Palm OS is PalmSource headquartered in Sunnyvale, California, with development offices in Montpellier, France and sales offices worldwide.

2.3.2 Mobile Programming Platforms

BREW (Binary Runtime Environment for Wireless) is an application execution environment released by Qualcomm in February 2001. Initially being restricted to work on only CDMA networks, now it is capable of running on any network and can be ported to any handset. [23]

Like J2ME, BREW is also vendor neutral development platform. It runs right above the hardware and runs on many different OS such as Palm OS. Regardless of platform, the main advantage of BREW is its low memory footprint (i.e. only 150K).

There are two essential components to the BREW platform. First is the BREW SDK, which is used by developers to create applications on it. Natively, it only supports the C and C++ programming languages, which poses a deeper learning curve for new developers. Because of this pressure, BREW decided to add Java functionality by selecting IBM's Java virtual Machine. The JVM runs in a layer top of the BREW platform and provides standardized J2ME APIs for the application developer.

The second part of the BREW platform is for the end user. It is a piece of software or firmware required by a handset to run BREW applications on it. This component is available at no cost and in some case; it is equipped with BREW enabled cell phones.

A key feature of BREW is the BREW Distribution System (BDS). While in other systems, the developer has to worry about testing, security and billing of the application they make, in BREW, all these are managed by the BDS, which interfaces with the carrier's billing system and permits them to sell them your application.

The **J2ME** (Java 2 Micro Edition) [24] is currently the most popular platform for developing applications for Mobile devices. J2ME is a smaller Version of Java 2 Standard Edition targeted towards consumer end embedded and small devices, which includes Cellular Phones, PDAs, Blueberry devices, Set-top boxes etc. Companies like Nokia, Sony Ericsson, Motorola, Sendo, Palm, and RIM have already based their platforms on J2ME specifications. As with Java, the development of J2ME is also passed through the Java Community Process (JCP), which is a collection of different companies to work on a platform.

The overall J2ME architecture can be viewed as:

- A Java virtual machine targeted to the consumer end user device. For mobile devices, the virtual machine is called KVM i.e. K is for Kilo, to demonstrate the small memory footprint.

- A group of libraries and APIs to use the device's capabilities and other functionalities. These libraries and APIs are grouped separately according to devices' type. They are known as Profiles and Configurations.
- Several tools to accompany development, deployment and device configuration.

The first two constitutes the runtime that is installed on the end user's mobile device to execute J2ME based applications.

The **.NET Compact Framework** is the latest initiative from Microsoft to compete with Java and BREW. The compact framework is a limited version of the complete .NET Framework and provides a runtime, the programming libraries and the development tools to create and execute applications on Smartphones running the .NET CF.

The most significant benefit is that the programming model for .NET Compact Framework devices is identical to that used by the developer using .NET to build applications for desktop PCs and servers. The only problem with .NET CF is the lack of penetration in the market and subsequently fewer number of devices with pre-installed framework. Microsoft is trying to cover up this by partnerships with major handset provides such as Motorola and Orange. The .NET Compact Framework offers a very good level of application portability for developers across Microsoft Windows server, desktop, and mobile device operating systems, while J2ME potentially offers a level of portability across any operating system provided support has been developed for that operating system.

3. Mobile Payment

3.1 Mobile Payment Environment

Mobile telecommunications continue to be phenomenally successful with more than 1 billion people, almost one in six of the world's population, using mobile phones. This historic milestone has been reached only 12 years after the launch of the first GSM networks.[2] The success of NTT DoCoMo's I-mode service in Japan, which currently has around 41 million data subscribers [3], illustrates the appetite for compelling mobile data service. In Europe, meanwhile, the huge uptake of short messaging (SMS) has demonstrated the need for non-voice services in that market. According to [2], there were over 30 billion SMS messages sent in 2001.

A new study by Frost & Sullivan, the international marketing consulting company, believes the foundations of m-commerce are currently being laid and that the market will take-off within the next few years. The report expects that trade worth USD 25 billion will be generated through mobile payments in 2006, which equates to around 15 percent of estimated online e-commerce consumer spending. The new report presents mobile payments as an opportunity to most parties involved from consumers and operators to merchants and system and device vendors. Banks and credit companies must get involved to protect their existing turf from new competition. [5]

As more sophisticated devices are developed, new applications have emerged that take advantage of color screens, longer battery life and greater storage capacity. These new applications include enhanced messaging (EMS) and multimedia messaging (MMS) which enable the downloading of data files, streaming video and images. The American Federal Communication Commission's (FCC) directive, mandating the addition of global positioning (GPS) in mobile phones, will enable location based mobile commerce.

3.2 Mobile Payment Principles

Existing mobile networks (GSM networks) are not designed to support data transport services beyond SMS therefore providing slow connection speeds and a limitation in the choice of

compatible applications. However, the introduction of packet-switched networks (e.g. GPRS), offering connection speeds similar to that of a fixed network, together with the inherent advantage of mobility, provides a setting that is ideal to the proliferation of payment services for the use of content and services.

3.2.1 The Actors

In order to understand the lifecycle of a mobile payment, it us useful to consider the typical case of a credit card transaction. In any credit card transaction, there are at least four parties involved: the user (consumer), the merchant (content provider), the issuer and the acquirer. As the user and the merchant's roles are clear, let us see shortly what the responsibilities of the issuer and the acquirer are. The acquirer is defined as a bank that arranges with the merchants so that the merchant can accept plastic card payments. When the customer makes a purchase at a merchant's location and uses a credit card for payment, the money is sent to the acquirer, which puts it in to the merchant's bank account and sends the details of sale to the card issuer. The card issuer records the amount of money paid and what was paid for on the customer's monthly bill. The acquirer usually charges the merchant a small fee or a small percentage. The issuer, as you probably might have guessed, is a financial institution that issues the identification payment card (e.g. Visa® Card) to the cardholder identified by the primary account. [4]

Figure 1 shows the relationships that exist between the involved parties. Note that the trusted third party and the payment service provider from the figure correspond to the issuer and the acquirer mentioned earlier.

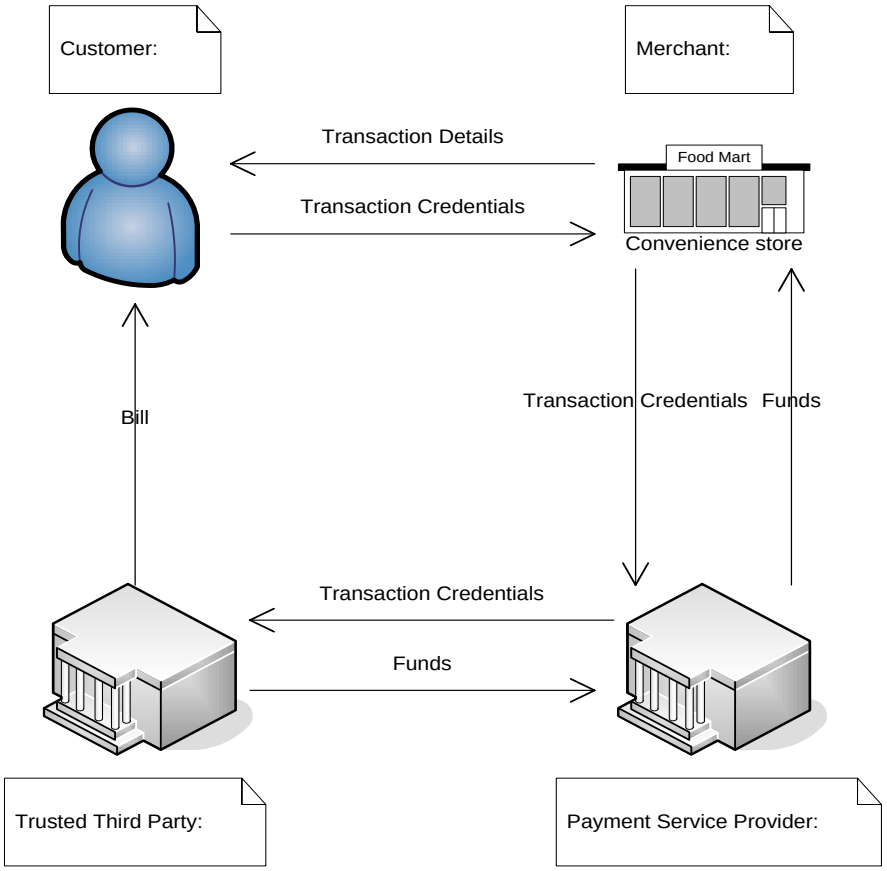


Figure 1: Typical Credit Card Transaction

This was the case of a regular credit card transaction. Let us see now how the above figure looks like in the case of mobile payments; Figure 2 pictures the scenario. In this case, the network is also an integral part of the transaction flow. The transaction dynamics are similar, although the form factor that contains the transaction credentials is different. In addition, in the case of remote payments, the transport of payment details will involve a mobile network operator and use of either a browser-based protocol, such as WAP or HTML, or a messaging system, such as SMS or USSD. Alternatively, the transport of payment details could be via Bluetooth, infrared, RFID or contact-less chip in the case of proximity payments.

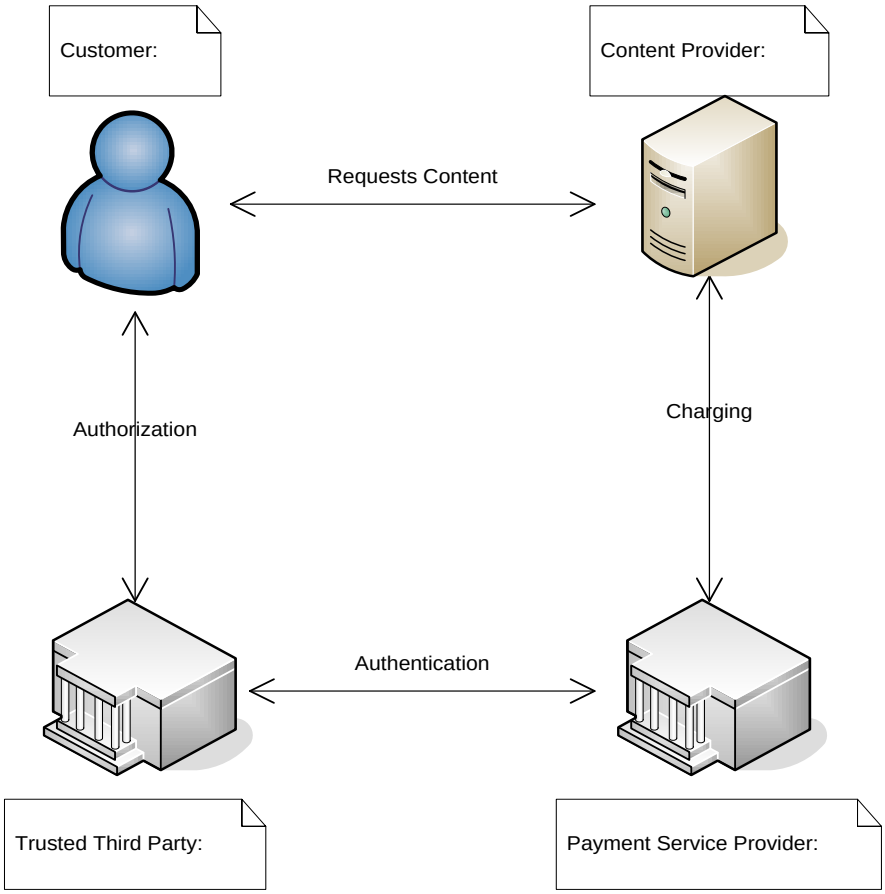


Figure 2: Relationships between the involved actors in a mobile payment. A trusted third party provides for authorization and authentication.

The customer in this case is a person owning a mobile device and willing to use that device for paying for a service or the supply of content. Throughout the rest of this paper, the customer will be referred to as the mobile user. The content can be physical goods or downloadable digital content; the service can also be either physical or digital service. The mobile user's roles include registering with the payment service provider (PSP), initializing the purchase, authenticating and authorizing the payment.

The merchant, or the content provider, is some organization that sells either physical or digital goods to customers. This actor will be further referred to as the content provider, as most of the time, the content that is being the subject of the purchase is some digital good. The role of the content provider can include forwarding authorization requests to customers and delivery of goods.

The payment service provider is the party responsible for the payment process. The customer may chose to register with a payment service provider to avoid retyping payment details on the mobile device every time she makes a purchase. The PSP acts as a middle actor that controls the payment between the consumer and the content provider. A PSP could be a financial institution, a mobile network operator, a credit card company or an independent payment provider.

The trusted third party (TTP) is an organization used to authenticate the involved parties and to authorize the payment settlement. This actor could be a financial institution, a mobile network operator or a credit card company. A financial institution or bank could take upon the roles of both the PSP and the TTP. Hence, for the purpose of this paper, the PSP will be responsible for all the roles of the TTP and there will be no further reference to the TTP. Therefore, we are left with only three involved actors in the mobile payments; the figure that represents more accurately the implemented payment provider is the following one:

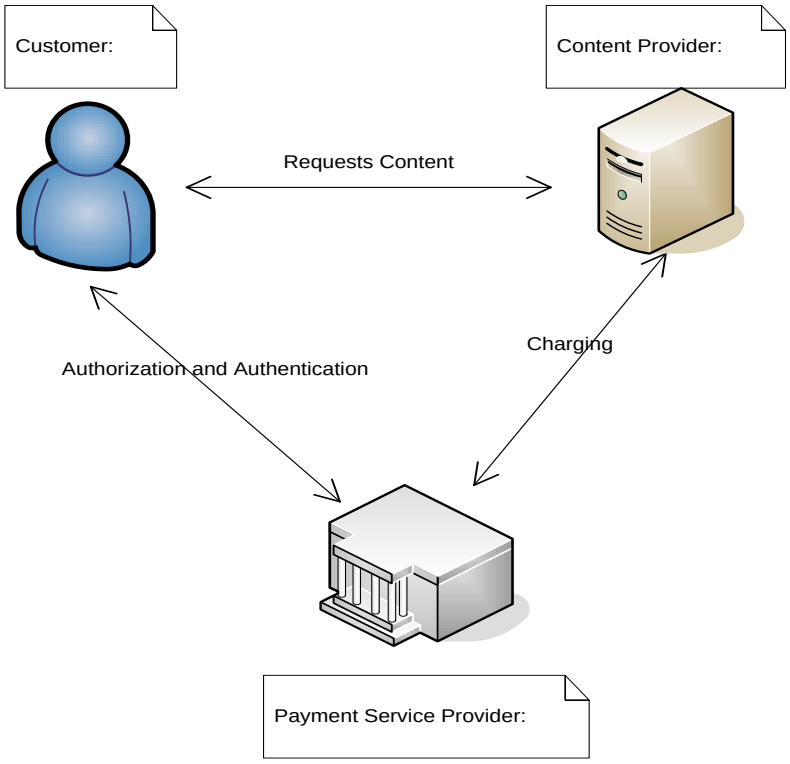


Figure 3: Relationships between the involved actors in a mobile payment. A trusted third party is no longer present.

3.2.2 The Characteristics

Mobile payments may be characterized into several categories depending on different criteria: transaction type, settlement type, content type and content value. An outline of some of the various payment types is presented in the followings.

- Transaction Type
  - *Pay per View* – the mobile user pays for each view or increment of the desired content. For example downloadable music files or video clips.
  - *Pay per Event* – the user pays for an event. This event may be the use of a service for a particular time interval or value.
  - *Pay per Unit* – the mobile user pays for each unit of content provided by the content provider. Units can be based on volume or duration of content, such as per byte or per minute. The amount of units used for each session will be billed to the user. Such examples could be used in downloadable games or streaming video services.

- *Flat Rate* – the mobile user pays a periodic amount to access the content on an unlimited basis during the period; e.g. unlimited access to online newspapers.

- Settlement Type
  - *Pre-paid* – the mobile user pays in advance before obtaining the content with prepaid accounts that are deducted after each session.
  - *Post-paid* – the mobile users receive and use the content before they are charged for it. They are billed after the access to the content was obtained, on their monthly phone bill for example.

- Content Type
  - *Digital goods* – represents downloadable digital content: video or music files for example.
  - *Digital services* – video streaming services for example.
  - *Physical goods and services* – pay parking for example.
  - *Ticketing*

- Content Value
  - *Micro-payments* – represent purchases of usually less than 10 Euros or USD. Examples include ring tones and pay parking.
  - *Macro-payments* – represent purchases of more than 10 Euros or USD. Examples include purchase of plane tickets.

3.2.3 The Scenarios

There are several key user payment scenarios, namely: automated point-of-sale payments (vending machines, parking meters and ticket machines); attended point-of-sale payments (shop counters, taxis); mobile-accessed internet payments (merchant WAP sites); mobile-assisted internet payments (fixed internet sites using phone instead of credit card), and peer-to-peer payments between individuals. Mobile-accessed Internet and peer-to-peer payments make up the bulk of payments, accounting for 39 percent and 34 percent of spending respectively in 2006 according to [5].

Point of Sale

In this payment scenario, the two parties are at the same location and the mobile device is used only as a payment instrument, not to communicate information about the goods or services being transacted. For example, if someone uses a wireless phone to order and pay for a traffic report, the payment is considered virtual because the company providing the traffic report is distant from the person receiving it. On the other hand, if someone uses a wireless phone to buy a soda from a vending machine that is considered a POS mobile payment because the vending machine and the consumer are at the same location.

At present, the POS model of mobile payments has not had much success, because it is essentially a substitute for a credit or debit card; the user does not gain access to any new services as a result of using it. Mobil Speedpass is the most successful type of POS mobile payment today, with more than six million users, but acts more as a loyalty device for Mobil than as a source of new revenue. [6]

The content provider initiates the payment at the point of sale. The user can authorize the payment either directly, by sending an SMS message to the PSP for example, or indirectly, via the content provider, such as using an IrDA, Bluetooth or some other wireless link.

This payment scenario is depicted in the following figure:

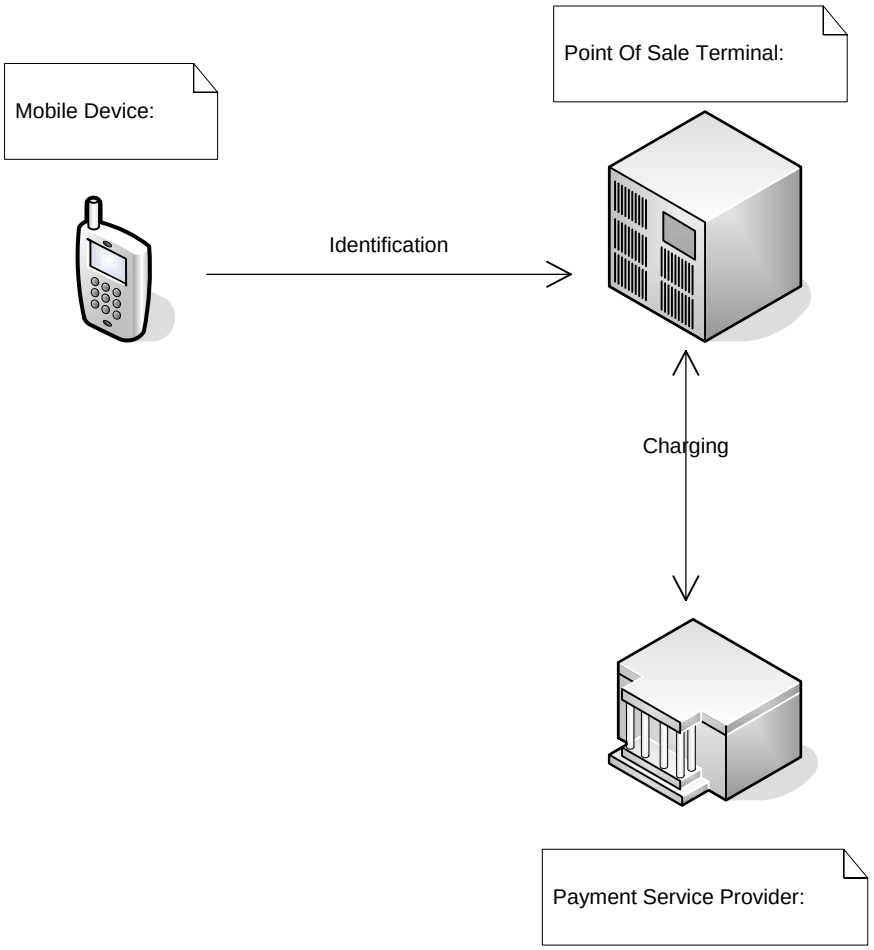


Figure 4: Payment System for Point of Sale

Mobile-Accessed Internet Payments

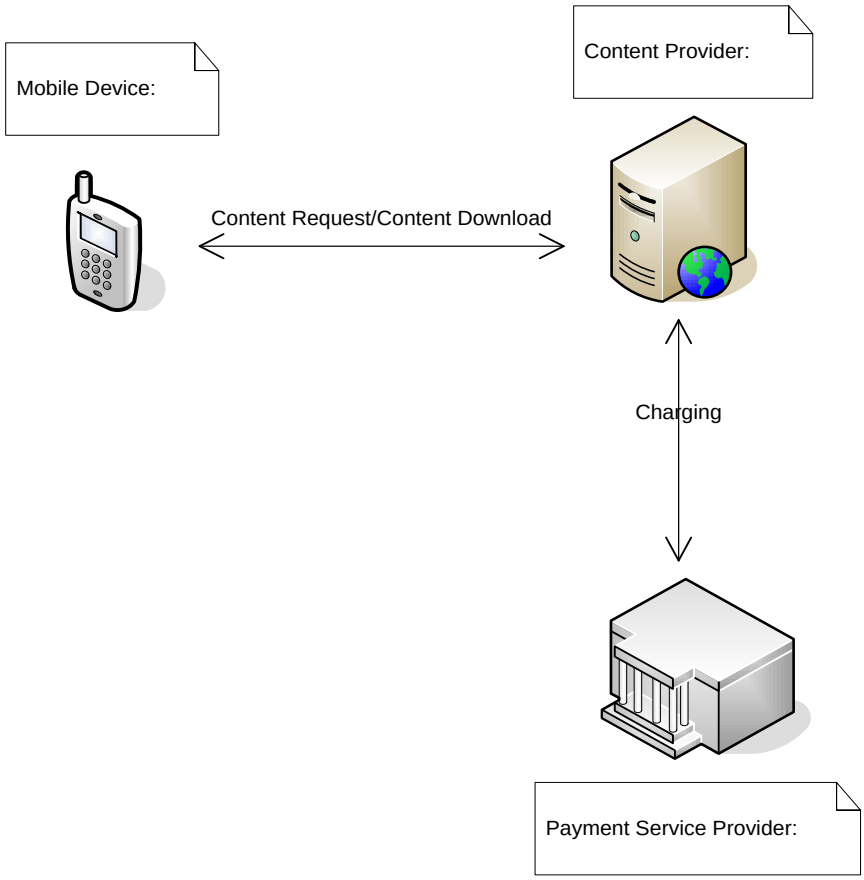


Figure 5: Payment System for Mobile-Accessed Internet Payments

In this payment scenario, the content provider offers digital content to mobile users by the means of a WAP enabled Internet site. Either a metered or an event-pricing model can purchase the content.

Metered content can be a streaming service, such as video, a radio channel or an on-demand gaming service. The payment of the transaction is dependent on a metered quantity of the provided service, such as the duration of the service, the data volume delivered, and a.s.o.

Event content may involve the full download of the content and the consumer pays a predefined price per downloaded item. The transaction is dependent on the successful download of the content, as it is worthless in the case of a failed download.
Once the mobile user makes a request for content to the content provider, the later one initiates a charging session with the PSP. The PSP will than seek for authorization and

authentication from the mobile user and will settle the payment using either a pre-paid or a post-paid method.

Mobile-Assisted Internet Payment

This scenario is similar to the previous one. The difference is that the content is purchased via a PC Internet connection and the mobile device is used only for authenticating the content recipient as the mobile user and authorizing the transaction.

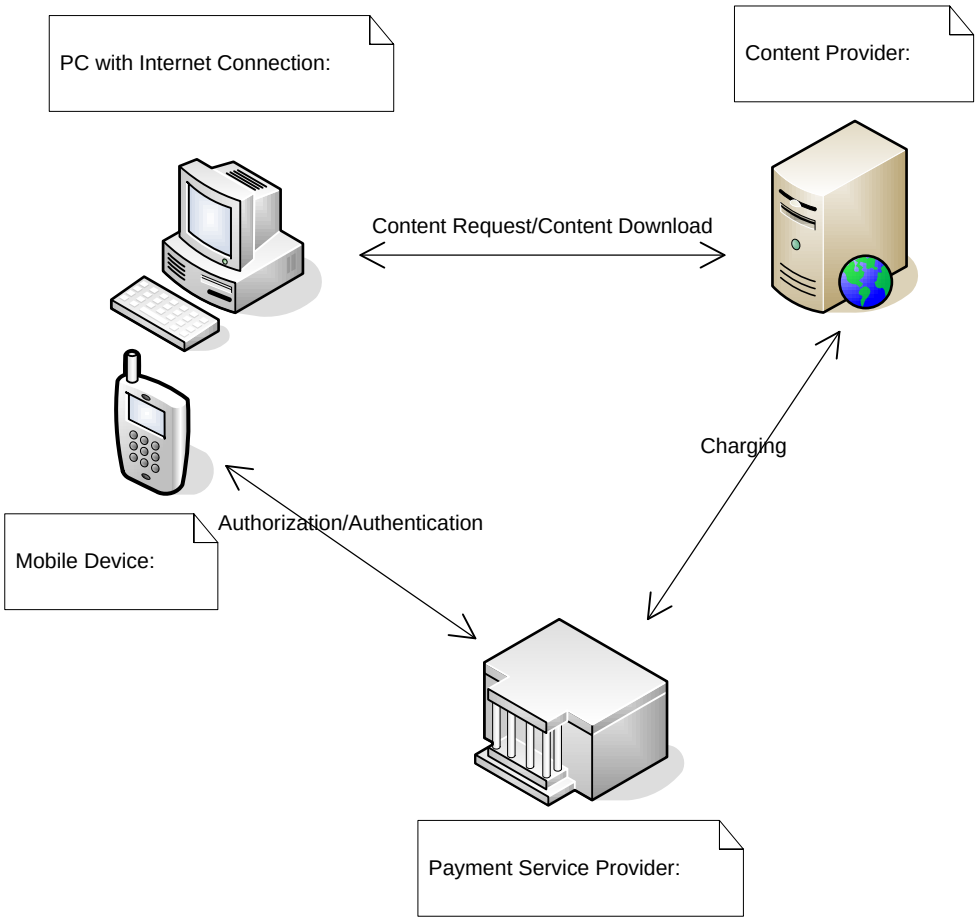


Figure 6: Payment System for Mobile-Assisted Internet Payments

3.2.4 The Lifecycle

Within any mobile payment system, there are some general interactions between the involved parties. These interactions vary from one payment scenario to other, but some of them are present in each scenario. The following generic sequence diagram outlines these generic interactions:

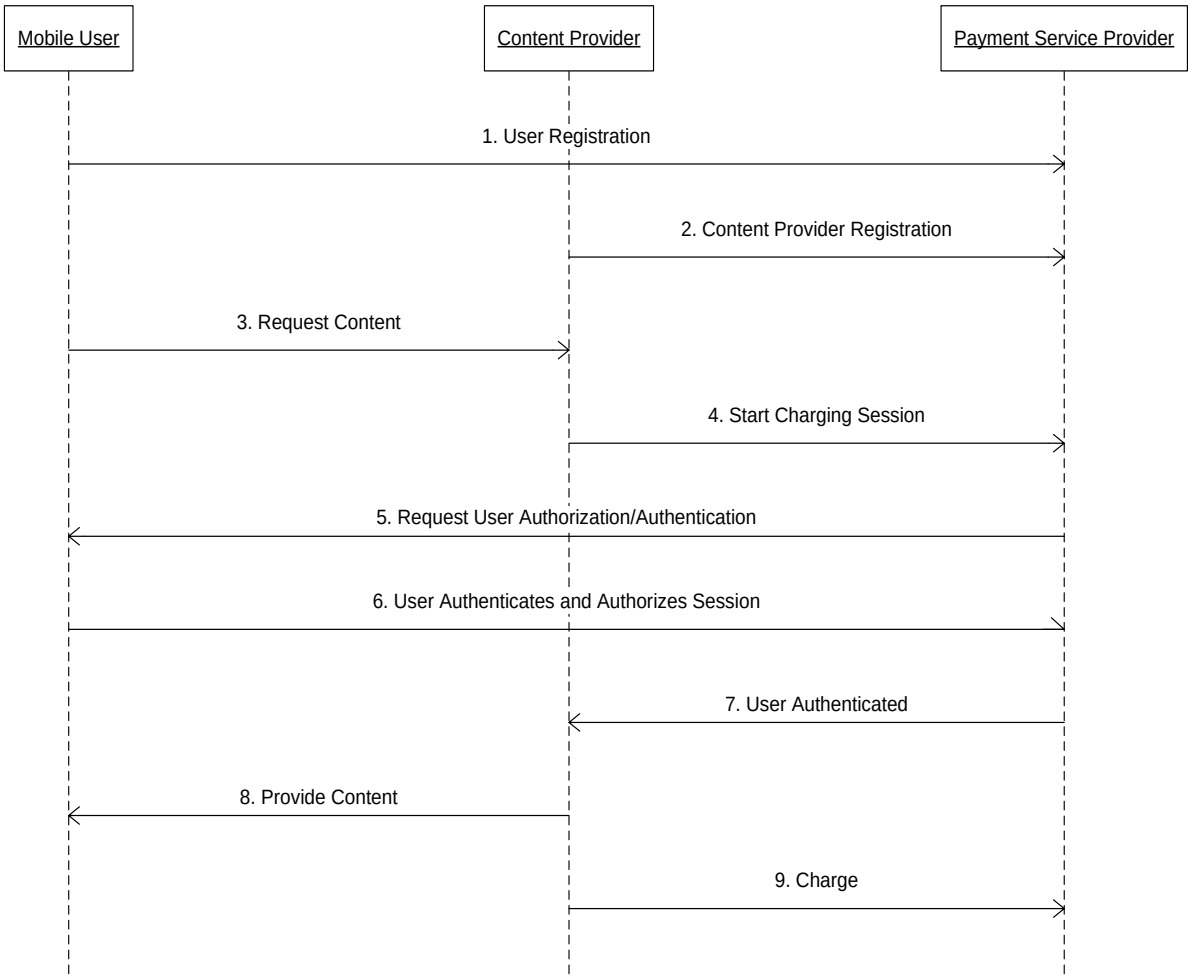


Figure 7: Generic Interactions in a Payment System

- **User Registration** - The mobile user needs to register with the payment service provider before they can avail of services offered by a registered content provider. The user will submit details of how they wish to pay for the services and they may be able to personalize the payment to suit their needs. In addition, the user may be requested to submit a personal identification number (PIN), which will be used during the user authentication process. A user identification number will be returned to the user following the completion of registration. This id will uniquely identify the user during each payment transaction.
- **Content Provider Registration** - The content provider needs to register with the payment service provider, to make their service available to mobile users via the

payment system. Service details, such as payment type for the service and the financial information for receipt of payment after use of the service. Following registration, the content provider will receive a service identification number, for identification in further operations with the payment service provider.

- **Request Content** - Once the mobile user has registered with the payment service provider, they are able to request the use of services provided by a registered content provider.
- **Start Charging Session** - On receiving the content request, the content provider will initiate a charging session request with the payment service provider. Both the user's id and the service's id will be sent to the payment service provider to uniquely create a charging session, which will be represented by a unique session identification number.
- **Request User Authorization/Authentication** - Before a charging session can be started, the mobile user must authorize that they are willing to pay for the content. An authorization request will be sent from the payment service provider to the user, usually in the form of a payment contract. This contract will state the terms and conditions of the payment agreement between the mobile user and the content provider. The user must reply to the payment service provider with confirmation of acceptance of the terms in the contract, before the charging session process can continue.
- **User Authenticates and Authorizes Session** - This is usually done by submitting their PIN from the mobile device being used. Authorization and authentication may be performed within the one request and the authorization confirmation may include the authentication PIN.
- **User Authenticated** - The payment service provider will notify the content provider whether the mobile user has been authenticated successfully. If authorization and authentication requests return a positive result, then the charging session will be initiated. The payment service provider will issue a unique session id to the content provider, signaling the start of a charging session.
- **Provide Content** - The content provider will now provide the requested content or service to the user.
- **Charge** - Depending on the payment scheme related to the requested service, the content provider would request a charge operation to the payment service provider at the end of the service or at different intervals during the service. On receipt of the charge request, the payment service provider will settle the payment transaction

between the two parties and notify each party of the result of the transaction. This will usually be presented to the mobile user as a payment receipt.

3.3 Mobile Payment: Mobile Operator Payment

Mobile payments can be split into three categories: mobile content, out-of-band and proximity [26]. Network operators are well suited to deliver payment services for mobile content because of their expertise in the area of billing. This type of payment is sometimes called "in band" payment because the content and the payment channel are the same. An example could be chargeable WAP over GPRS. Users would be offered either a subscription or a per usage payment model.

An example of in band payment: Anne is on holyday and uses her camera phone to take a picture and send it via MMS to a friend of her. She is charged €1 to her pre pay account. This would involve MMS and 2.5G technologies using a mediation system integrated with real time stored value micro payment system.

3.3.1 Network Operator Payment Systems

There are many ways of charging for data services in a packet switched network: per item billing, data volume and connection speeds.

Managing the charging for all these different possibilities is a complex task. Users might require separate billing rates for streaming video depending what they are watching and expect different quality of service. In the same time, a user might be perfectly happy to pay per MB. However, if the connection fails after downloading 3 MB out of the 4 MB file she needed, the user might not be willing to pay for the useless downloaded 3 MB.

Operators are not interested in providing a stand-alone payment application. Charging and payment are at the heart of their wireless data systems and form part of the infrastructure. The goal is to deliver content to mobile users that is personalized, location and time specific, with low friction payment that is suited to the service.

Operators deploying these systems will take on the role of payment provider, which could be a double-edged sword. On one hand, they want to control the relationship between the user and the content providers; on the other, they are faced with investing heavily in payment systems that can support these complex models. They can choose to outsource transaction processing to an ASP that is en expert in aggregating micro payments, or they could upgrade their systems to support data billing services. Serious operators will most likely turn to the second option.

3.3.2 Mobile Payment Standards

PayCircle [27] is a consortium formed by U.S. and European technology companies including Hewlett-Packard, Lucent, Oracle, Sun Microsystems and Germany's telecom equipment maker Siemens. The consortium aims to provide mobile device users worldwide a standard means of making mobile payments, regardless of the payment systems used by merchants or service providers. PayCircle intends to define open and uniform interfaces based on existing standards, without the need to install any new software.

The Parlay Group [28] is an open multi-vendor consortium formed to develop open technology-independent application programming interfaces (APIs) enabling: technology, Internet and eBusiness companies, independent software vendors (ISVs), software developers, network device vendors and providers, service bureaus, application service providers (ASPs), application suppliers, and large and small enterprises to develop applications and technology solutions that operate across multiple networking platform environments.

The Parlay APIs allow third-party applications to be hosted within a telecom operator’s own network and allow applications running on external application servers to offer their services to the operator’s subscriber base via a secure gateway. In addition to providing secure, manageable access to capabilities in the service provider’s network, the Parlay gateway also links third-party applications to the operator’s business environment via the Parley framework.

In February 2003, Orange, Telefonica Moviles, T-Mobile and Vodafone signed an agreement to form a new **Mobile Payment Services Association** aimed at delivering an open,

commonly branded solution for payments via mobile phones, designed to work across all operator networks. The solution will work across country boundaries and will seek to complement existing industry solutions. In June 2003, the consortium has re-branded itself as **SimPay** [29].

3.3.3 Vendor Billing Solutions

Current vendors of mobile billing solutions for GPRS and 3G services include Amdocs, Cerillion Technologies, Convergys, EHPT, Geneva Technology, Kenan Systems, Portal, Sema, and TelesensKSCL [30]. The services provided include content billing management and CRM (Customer Relationship Management) services.

3.4 Mobile Payment: Out-of-Band Payment

Out of band payment [26] refers to the fact that the payment channel is separate to that used for a shopping. For example, a credit card holder may use their mobile device to authenticate and pay for a service they consume on the fixed line Internet or interactive TV. In order to make the wireless device suitable for authenticating payments, financial institutions are especially interested in wireless PKI, shared secret (or symmetrical key) schemes, or merging with their chip card programs via dual slot or dual chip devices.

An out of band payment example: Anne is looking to by tickets online for a concert she wants to go to. She browses to the ticket vendor site, books the tickets and pays with her Visa card. The payment authentication request is sent to her mobile phone via SMS, she authenticates using a personal PIN, digitally signing the order. A receipt is sent to her phone.

3.4.1 Financial Institutions

Banks are already exploiting the potential of mobile phones to be used as personal secure payment terminals. [26] Different payment schemes exist where a bank deducts some amount of money from a user’s account to pay for services or digital goods. Authentication of the transaction is done through various methods: using a dual slot phone for credit card payments,

PIN authentication via a SIM toolkit application and with the use of a digital signature based on the PKI mechanism.

At the moment, users are required to be in the possession of a mobile device that has been registered before with the payment service provider. Authentication and authorization is done via a PIN; the user selects a PIN upon registering with the payment service provider, and, at the time of the transaction, the user is required to enter the PIN. This is than compared with the one with which the user registered in the first place.

3.4.2 Reverse Charge SMS

Reverse Billing SMS [31] is a process whereby small charges are made to a subscriber's mobile phone in return for a service. Typically, the subscriber sends an SMS message to a short code number (usually 5 digits), often including a keyword in the message and in return receives a service or a piece of information for which a charge is made on that subscriber's mobile phone.

Typical applications include charged voting systems for television and the press, subscription campaigns for a football team's news service allowing fans to receive video goal flashes at 50p a goal, or prize draw and competition type marketing applications.

3.4.3 Vodafone m-pay

Vodafone m-pay bill lets users make small payments for services on the web and on the mobile internet by adding the amount to their monthly mobile phone bill or deducting it from their prepaid airtime credit.

Vodafone m-pay cards lets users register their credit and debit cards for larger online purchases. It is fast and secure: web payments are controlled by user name, password and mobile phone, whilst Vodafone live! and mobile internet transactions ask users to enter a PIN

code on their mobile phone; there is no need to enter credit card details every time one buys something.

3.4.4 MobiPay

MobiPay [32] is a Spanish company, which launched a pilot scheme for mobile payments. Based on a cooperative model between mobile telephone operators and financial institutions, MobiPay is owned by Banco Bilbao Vizcaya Argentaria and Santander Central Hispano, as well as all Spanish mobile network operators (which include Vodafone but not O2). MobiPay operations include pay at a merchant (shoe store, restaurant, etc.), pay on the internet, pay at a vending machine (drink machines, cigarettes, etc.), send money to another person, pay an invoice (electricity invoice, etc.).

The MobiPay security system is based on the following elements:

- personal MobiPay number
- warning and blocking system
- encrypted communications via GSM-secure network
- telephone number associated to MobiPay
- PIN number of the telephone associated to MobiPay

The personal MobiPay number guarantees that the operations are carried out by the customer. This number is only known to them and they are charged with its safekeeping. The system allows up to three erroneous key-in attempts. If exceeded, the system is blocked. All MobiPay operations are authorized by the customer. Authorization is performed when the customer keys in the personal MobiPay number.

This system is very convenient to customers and minimizes the merchants operational process as the merchant no longer needs to ask for proof of the customer's identity or validate the customer's signature. MOBIPAY registration is associated to the customer's mobile telephone number from which the operations are going to be carried out. To make an operation it is therefore necessary to initially introduce the PIN number of the phone.

3.4.5 PayBox

In May 2000 paybox.net AG, provider of a mobile phone based payment system, started operation in Germany. Deutsche Bank owns 50% of the company and is responsible for payment clearing and settlement. The system is independent of particular mobile phone service providers and it does not depend on PKI-structures.

The PayBox approach is straightforward and consists of four steps:

i. the payer communicates his or her mobile phone number to the payee;

ii. the payee forwards this number and price information to paybox.net;

iii. paybox.net checks the identities of payer and payee and checks for any restrictions before a voice computer calls the payer asking if he or she is willing to pay the amount x to merchant y, and the payer is requested to confirm this by typing in his or her PIN;

iv. the payer types in the four digit PIN for PayBox (not his or her SIM-card PIN) and the voice computer thanks for paying. The rest is conventional payment clearing and settlement.

3.5 Mobile Payment: Proximity Payment

A promising payment application for mobile commerce is proximity transactions - using the device to pay at a point of sale, vending machine, ticket machine, tolls, parking, etc. By leveraging parallel technologies, such as Bluetooth and 802.11, mobile devices can be transformed into sophisticated payment devices that can process both micro and macro payments.

A proximity payment example: Anne is at a photo and imaging shop; she transfers her holiday photos from her digital camera to the store computer over a Bluetooth link; the payment request is sent to her phone, also over Bluetooth, where she accepts it, and her credit card information is returned to the store point of sale device. This scenario would involve the use of some kind of wireless technology and a Java applet for example, running on the mobile device and the point of sale terminal.

3.6 Summary

This section has presented the general characteristics of the mobile payment domain. As one can easily notice, there is a great diversity of developed solutions and implementations. However, the underlying generic concepts of a mobile payment are the same.

The next section outlines the specifications and architecture of the developed mobile payment provider. The framework was developed to support the generic interactions involved between the parties; these were presented in section *3.2.4 The Lifecycle*. Following this, a mobile-assisted internet payment scenario was used as a test bed to evaluate and test the framework

4. The Specifications and Architecture of the Payment Provider Framework

This chapter describes the design patterns, the architecture and the specifications of the framework. Because most of the functionality of the framework is exposed throughout web services, it will start with a brief overview of web services technologies. It will be followed by a presentation of the most important design patterns employed during the development of the framework and, finally, the main components of the framework will be presented.

4.1 Web Services

Web services [33] are pieces of business logic available somewhere on the Internet and that are accessible through standard Internet protocols such as HTTP or SMTP. Using a web service could be as simple as logging into a site or as complex as facilitating a multi-organization business negotiation.

A web service has several behavioral characteristics:

- XML-based: by using XML as the data representation layer for all web service protocols and technologies, these can be interoperable at their core level. XML provides networking, operating system and platform independence.
- Coarse grained: building an object-oriented program from scratch requires the creation of several fine-grained methods that are than composed into a coarse-grained service. An individual method is to fine to provide any use at corporate level; web services on the other hand, provide a natural way of defining coarse-grained services.
- Synchronous or asynchronous: synchronicity refers to the binding of a client to the execution of the service while asynchronous operations allow a client to invoke a service and then execute other functions and retrieve the results at a later stage.
- Supports RPC (Remote Procedure Call): web services allow clients to invoke procedures, functions and methods on remote objects using an XML-based protocol.
- Supports document exchange: web services support the exchange of documents, either simple or complex as an entire book, relying on the advantages of XML and its generic way of representing data.

Some of the reasons that led to the choice of using web services to build the framework are:

- Web services are language agnostic, i.e. they are not based on a specific programming language: Java, C, C++, Python, ...
- Web services are not based on a programming data model, i.e. objects vs. non-object environments.
- Web services are relatively simple: they work over existing internet technologies (HTTP, SMTP, etc), can be easily integrated in today's web interface and a small amount of code is enough to expose a web service.
- Support of software industry leaders (Sun Microsystems, IBM, Microsoft, etc.)

Several technologies have been introduced in the last years. However, three of them have emerged as worldwide standards that make up the core of today's web services technology. Well see them described shortly in the following sections.

4.1.1 SOAP

Simple Object Access Protocol (SOAP) provides a standard packaging structure for transporting XML documents over a variety of Internet technologies, including SMTP, HTTP and FTP. It also defines encoding and biding standards for encoding non-XML RPC invocations in XML for transport.

The message structure defined by SOAP consists of three major parts, the envelope, the header and the body. All parts are mandatory in a message, except the header element, which is optional. The envelope is the top-level XML element in a SOAP message. The envelope contains the header and body, and is the unit of communication. The header is used to extend the SOAP message with additional features and functionality, such as security, transactions, and other quality of service attributes associated with the message. The body contains the payload of the message, i.e. the application-defined XML data being exchanged in the SOAP message.

4.1.2 WSDL

Web Service Description Language (WSDL) WSDL is an XML technology that describes the interface of a web service in a standardized way. WSDL standardizes how a web service represents the input and output parameters of an invocation externally, the function's structure, the nature of the invocation (in only, in/out, etc.), and the service's protocol binding. WSDL allows disparate clients to automatically understand how to interact with a web service.

In a Web Service interaction, the WSDL file is produced and published by the service side and the WSDL file is used to obtain the necessary information about a service by the client side. Both parties need copies of the same WSDL file for the interaction to work.

4.1.3 UDDI

Universal Description, Discovery, and Integration (UDDI) provides a worldwide registry of web services for advertisement, discovery, and integration purposes. Business analysts and technologists use UDDI to discover available web services by searching for names, identifiers, categories, or the specifications implemented by the web service. UDDI provides a structure for representing businesses, business relationships, web services, specification metadata, and web service access points.

4.2 Designing with Patterns

This section describes the design patterns implemented in the framework.

4.2.1 Abstract Factory

The Abstract Factory pattern [34] provides an interface for creating families of related or dependent classes without having to specify the actual classes. This design pattern consists of a parent component, the abstract factory, which declares an interface for operations that create abstract product objects. These abstract products declare an interface for a type of product

object. The implementation of this design releases a concrete factory that implements the operations of the abstract factory to create the concrete product objects. The concrete products define a product object to be created by the corresponding concrete factory. This product object implements the abstract product interface. Client objects will only use interfaces declared by abstract factory and abstract product classes.

The Abstract Factory is used to develop frameworks and systems that can be configured with one of multiple families of products.

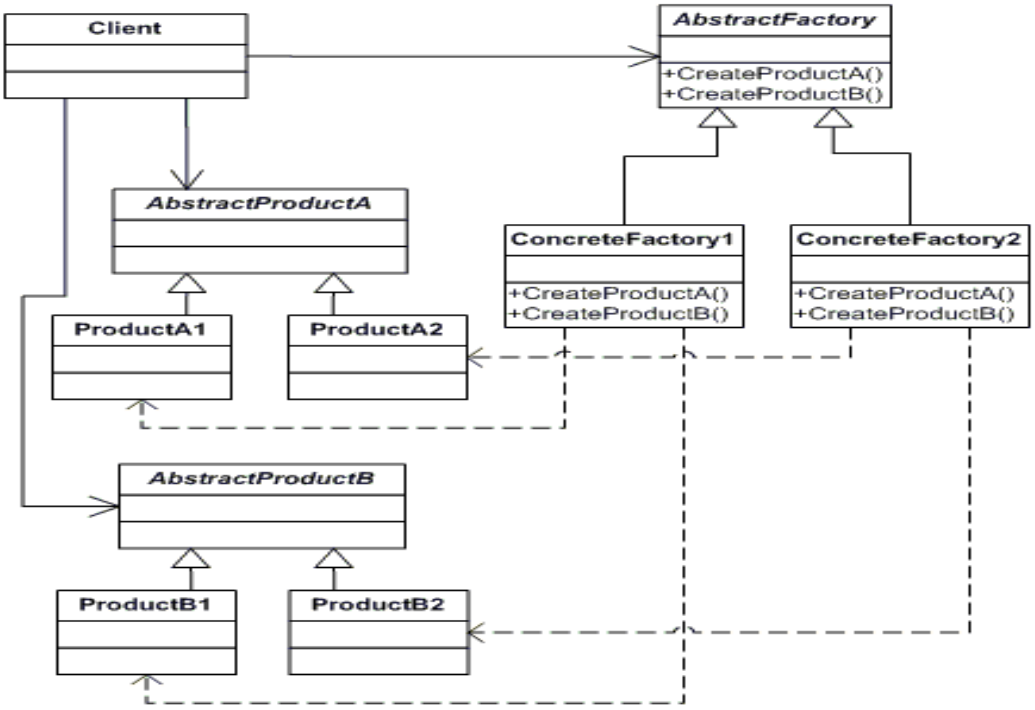


Figure 8: Abstract Factory Design Pattern

4.2.2 Data Access Object (DAO)

The DAO [35] implements the access mechanism required to work with the data source. The data source could be anything from a persistent store like an RDBMS to a XML file. The business component that relies on the DAO uses the simpler interface exposed by the DAO for its clients. The DAO completely hides the data source implementation details from its

clients. Because the interface exposed by the DAO to clients does not change when the underlying data source implementation changes, this pattern allows the DAO to adapt to different storage schemes without affecting its clients or business components. Essentially, the DAO acts as an adapter between the component and the data source.

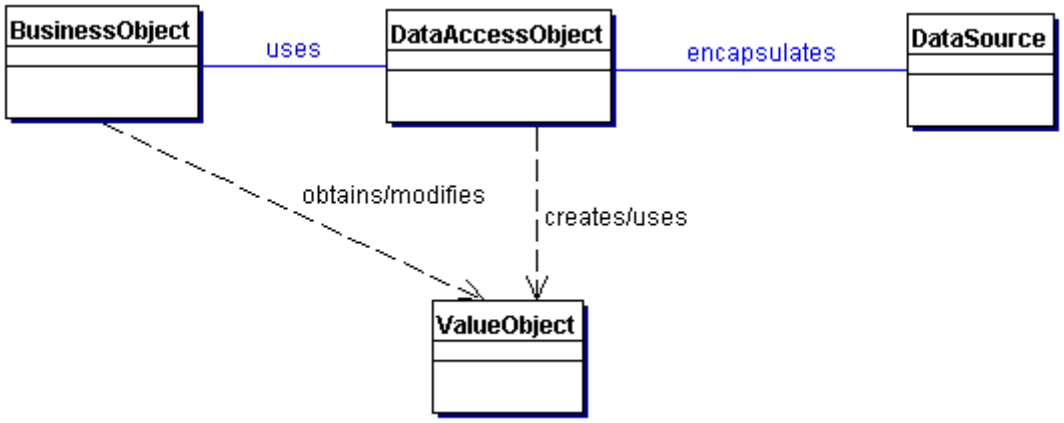


Figure 9: Data Access Object (DAO) Design Pattern

The DAO pattern can be made highly flexible by adopting the Abstract Factory design pattern, as it was the case in our framework. You will see this explained later, in the implementation section of this paper. This approach has several advantages: transparency, easier migration to a different data source, reduces code complexity in business objects, centralizes all data access in a separate layer.

4.2.3 Façade

The Façade design pattern [36] provides a unified interface to a set of interfaces in a subsystem. It does this by defining a higher-level interface that makes the subsystem easier to use.

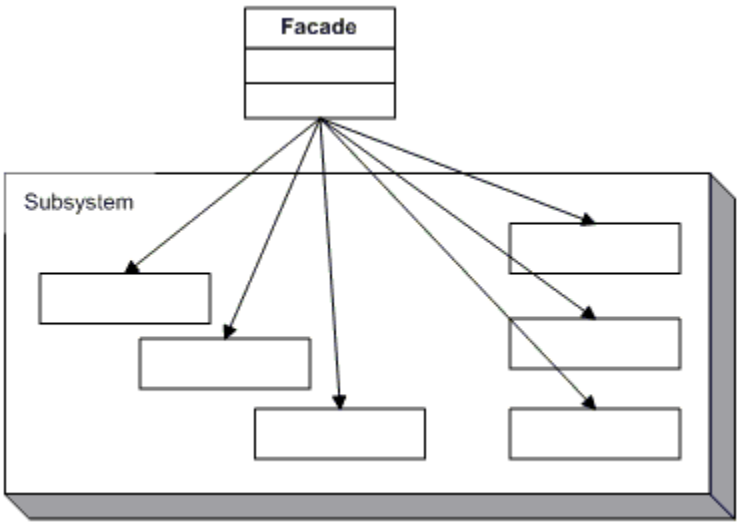


Figure 10: Façade Design Pattern

4.3 Mobile Network Payment System

The payment framework was developed and deployed using a real mobile network (Orange GSM). The overall system consists of three distributed components, the mobile device, the content provider application and the payment service provider framework. The next sections will present each of these.

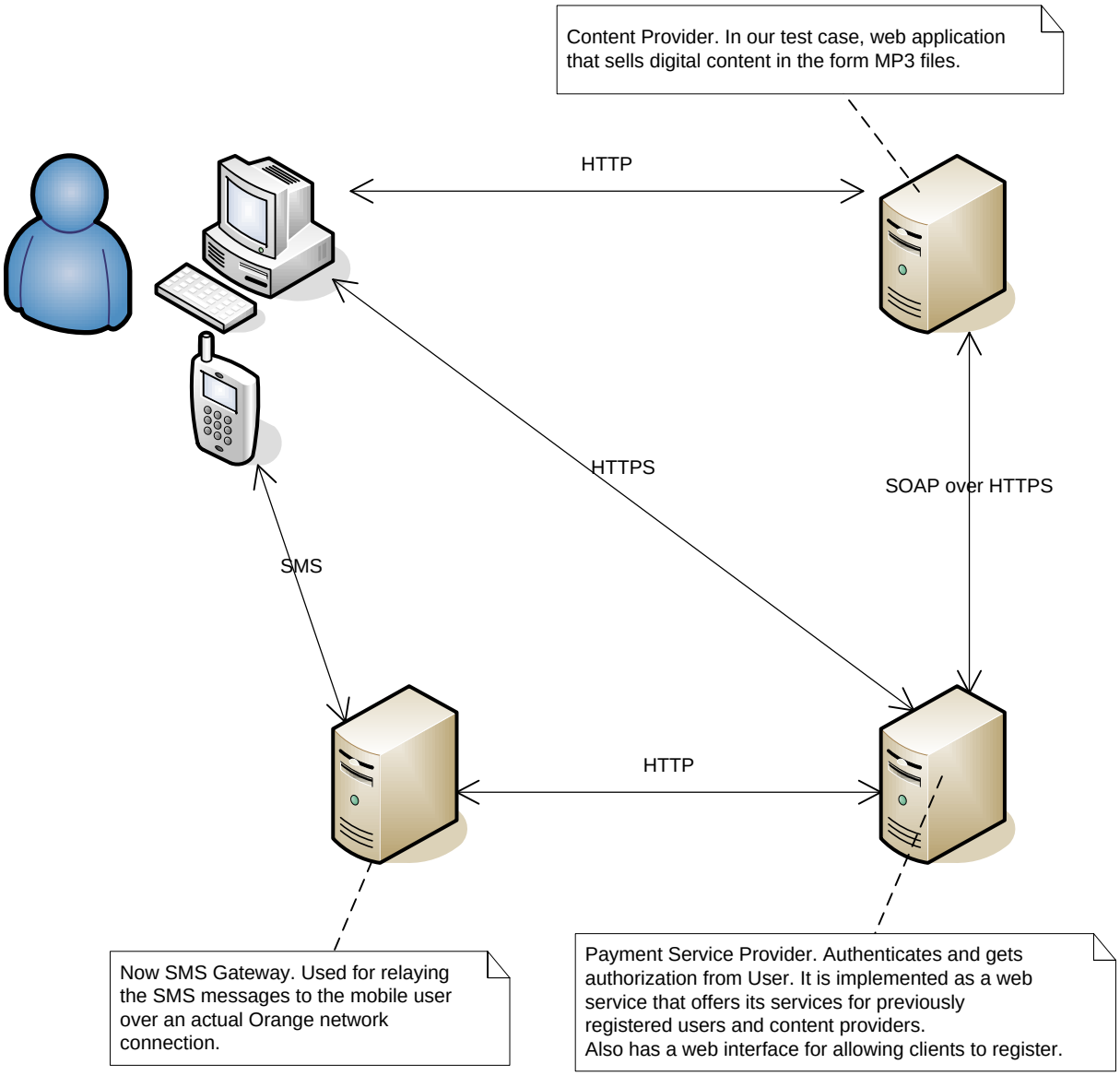


Figure 11: Topology of the Implemented Network Payment System

4.3.1 Mobile Client

The mobile client requests content via HTTP requests from a PC. The requests are sent to the content provider. Before beginning to access the content, the user is requested to register with the payment provider if she has not done so already. Communication with the content provider is done through HTTP while the communication with the payment provider goes on over HTTPS. Upon registering with the payment provider, the user is presented with a list of available payment methods and than required to fill in some personal information and details pertinent to the chosen payment method. Before validating the registration, a code is sent to

the user via SMS; the user is required to enter the code in order to validate the registration. This is done to ensure that users register in fact with mobile devices that they own. Now registration is complete and the mobile user can request to access a service supported by the payment method, which they have just previously registered with.

After the charging session is initiated, the user will receive a SMS message. This message will display the payment contract of the charging session to the user. She will be requested to reply to this SMS with the PIN chosen upon registration. If this PIN is correct, then she should be able to access the requested web content via a URL link.

4.3.2 Content Provider

The content provider deploys the web content on a web server, which is accessed by sending HTTP requests to a Java servlet; in our implementation, Apache Tomcat was used as the servlet container. All requests that are payment related are invoked as web service client operations; these result in SOAP requests being sent to the payment provider. The content provider will have registered each service with each payment implementation of the framework for which the service has support. Any web service calls, resulting from a mobile user request, will propagate through the façade web service to the payment implementation of the mobile user's registered choice.

4.3.3 Payment Service Provider

The payment web service was deployed on an Apache Tomcat servlet container using Apache Axis. It has a registration component that allows mobile clients and content providers to register and a web service component that waits for SOAP requests from the content providers. All the registration details are stored in a database for persistence.

At the beginning of a charging session request from a content provider, a contract is produced by the payment provider from the already stored registration details. This contract must be digitally signed by the mobile user before the charging session is activated. In this implementation, the framework will call a SMS gateway API that makes a HTTP request to

the Now SMS gateway server. The information sent to the server includes the mobile user's number and a text version of the generated contract. At the SMS gateway, the text message is set as the payload of the SMS text message and, using the mobile number, the message is sent to the user's mobile device. On receipt of the SMS message, the user is required to authorize the session. If the user agrees to the terms of the contract, she has to replay to the SMS with her PIN that was set at registration time. When the SMS gateway receives the returning message containing the PIN, then this, along with the mobile number, will be sent back to the payment provider via a HTTP request. The web service will process this information by comparing the user's entered PIN with the original registered PIN. If they are identical, then the user is authenticated. The charging session may now be started.

To account for network delays, especially during the SMS communication process, the payment system was developed to be asynchronous. On completion of the authorization and authentication stages, the framework implementation will only update the database record of that charging session. The content provider will not be informed by the payment service provider when the charging session has been activated. A request will be initiated by the mobile user after they have authorized the payment agreement. This will involve the user accessing a URL link presented to them by the content provider prior to the authorization stage. The URL will have the session id number appended as a parameter, which will be used by the content provider to reference the particular charging session when forwarding the charge request to the payment web service. If the charging session is active, then the charge request will be processed and a payment confirmation object will be returned. Otherwise, the user will be asked to complete the authorization process or start again. When the charging session is terminated, a payment confirmation will be sent to the user's mobile device as a payment receipt.

Depending on the characteristics of the charging session, the final charge request, which may be the only charge request, will be accompanied by a release request from the content provider to the payment web service. Any remaining amounts to be settled will be processed and a final payment confirmation will be returned. The charging session will be removed for the active session in the database.

4.4 The Components of the Framework

The framework consists of two main components: a **registration component** and a **charging component**. These two components implement all the operations supported by the framework. The next figure depicts the overall architecture of the framework, while the next sections describe shortly each of the components and their specifications.

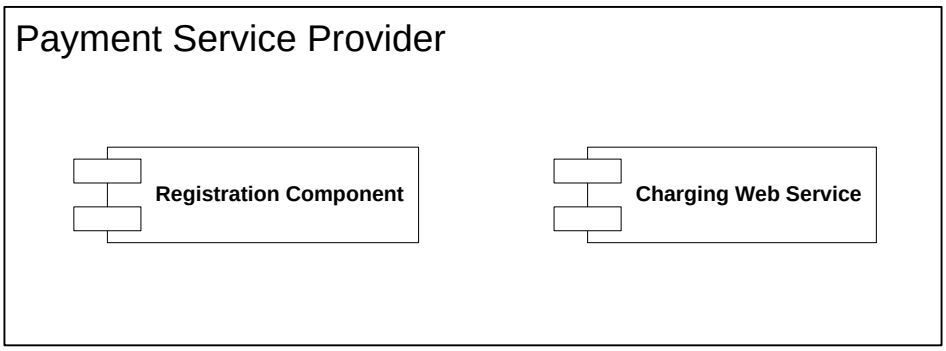


Figure 12: Bird-eye view of the Payment Service Provider Framework

4.4.1 The Registration Component

The registration component is intended to be a web based front end of the framework. Its function is to allow clients to register with the framework. Hence, the component handles the registration of both mobile users and content providers. The registration involves sending client information to the framework, which is extracted and stored in a database for further reference. After registering, the clients will be able to use the services offered by the payment provider.

The information requested from each mobile user at registration time is:

- name: represents the full name of the mobile user.
- phone number: this is the mobile phone number of the user; it has to be in international format and it is used as a means of authenticating the user and authorizing the transaction; this is done by sending SMS messages to the mobile device.
- mobile operator name: this is the name of the mobile operator with whom the user has either e pre-paid account or a valid subscription.

- payment type: this is the payment type that the user wants to use at payment time. Currently only credit card and operator billed payments are implemented. In the case of credit card payments, the payment would be billed directly on the user's credit card. With operator billing, either the user would be billed by the mobile operator, on her phone bill at the end of the month, or the payment would be deducted from a pre-paid account. If the user wants to be able to use several payment types, she will have to register with each payment type separately.
- pin: the mobile user has to enter a personal identification number. This will be used at transaction time to authenticate the user.
- credit card details including credit card number, type and expiration date: these are only requested if the user chooses credit card as its desired payment type.

The information requested from content providers at registration time includes:

- name: name of the company or organization.
- payment type: this is the payment type that the content provider is willing to accept as a means of payment for its goods. If a content provider supports several payment types, it has to register with each one of them.
- service URL: this is the URL of the digital content that the content provider offers.

The architecture of the registration component is pictured in the following figure:

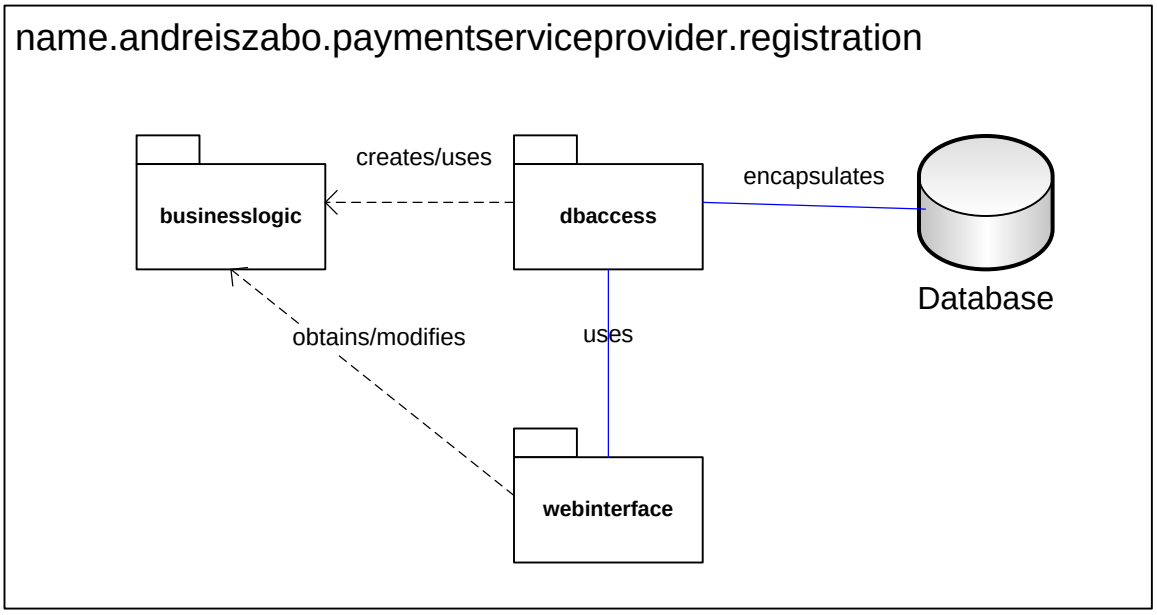


Figure 13: Architecture of the Registration Component

The `webinterface` package contains a number of JSP pages with some servlets. The `businesslogic` package defines all the business objects (Service, Client, etc) while the `dbaccess` package implements the access layer to the background database represented by a MySQL database engine; this is used as the persistence mechanism of the framework. The component is implemented in the `name.andreiszabo.paymentserviceprovider.regsitration` package. For detailed information about the implementation, see section *5.2.1 The Registration Component*.

4.4.2 The Charging Web Service

This component is designed for creating a payment transaction period called a charging session. The charging session can only take place between a mobile user and a content provider that have previously registered with the payment provider. The main function of the payment provider is therefore to transact an agreed amount from the mobile user to the content provider supplying the goods or services. This process should be transparent, atomic and secure to both of the involved parties.

A charging session is defined by the time interval starting from when a payment contract is agreed by all parties until the payment contract is no longer valid. Each charging session has a mobile user, a content provider and one payment provider. There can be however several payment transactions during a charging session; this is dependent on the transaction type used (see section *3.2.2 The Characteristics* for a list of mobile payment's transaction types).

Before a charging session is initiated, several operations have to be performed in the first place. The purpose of these operations is to ensure that all parties agree with the terms of the payment and are aware of the consequences of the transaction. A session is initiated at the request of a content provider as a result of a user's request to use a content service or good. At this point, a contract is created using the content provider and user's details entered during the registration. This contract outlines the payment agreement between the parties and specifies the unit value to be paid for a service during a finite interval of time. The contract is presented to the mobile user who has to sign it to make the charging session active.

To ensure that the mobile user is the same person from whom the payment will be deducted from, the mobile user has to authenticate herself. The contract is sent to the mobile user's device for authorization. To limit the number of messages sent across the network, the user is requested to replay to the message with her private PIN; this is the same PIN the user entered upon registration. When the replay is received and the PIN is checked against the PIN stored in the database and they match, the mobile user is considered authentified and the transaction authorized. Thus, the charging session may become active.

The charging component of the web service is designed using the façade and abstract factory design patterns. The abstract factory creates three abstract products called managers. Each of these manager's role is to handle a portion of the charging web service from the point of view of one of the involved parties. The three managers are the **user manager**, the **charging manager** and the **service manager**. The next paragraphs will describe shortly each of these managers while the next picture provides a static class diagram of the charging web service:

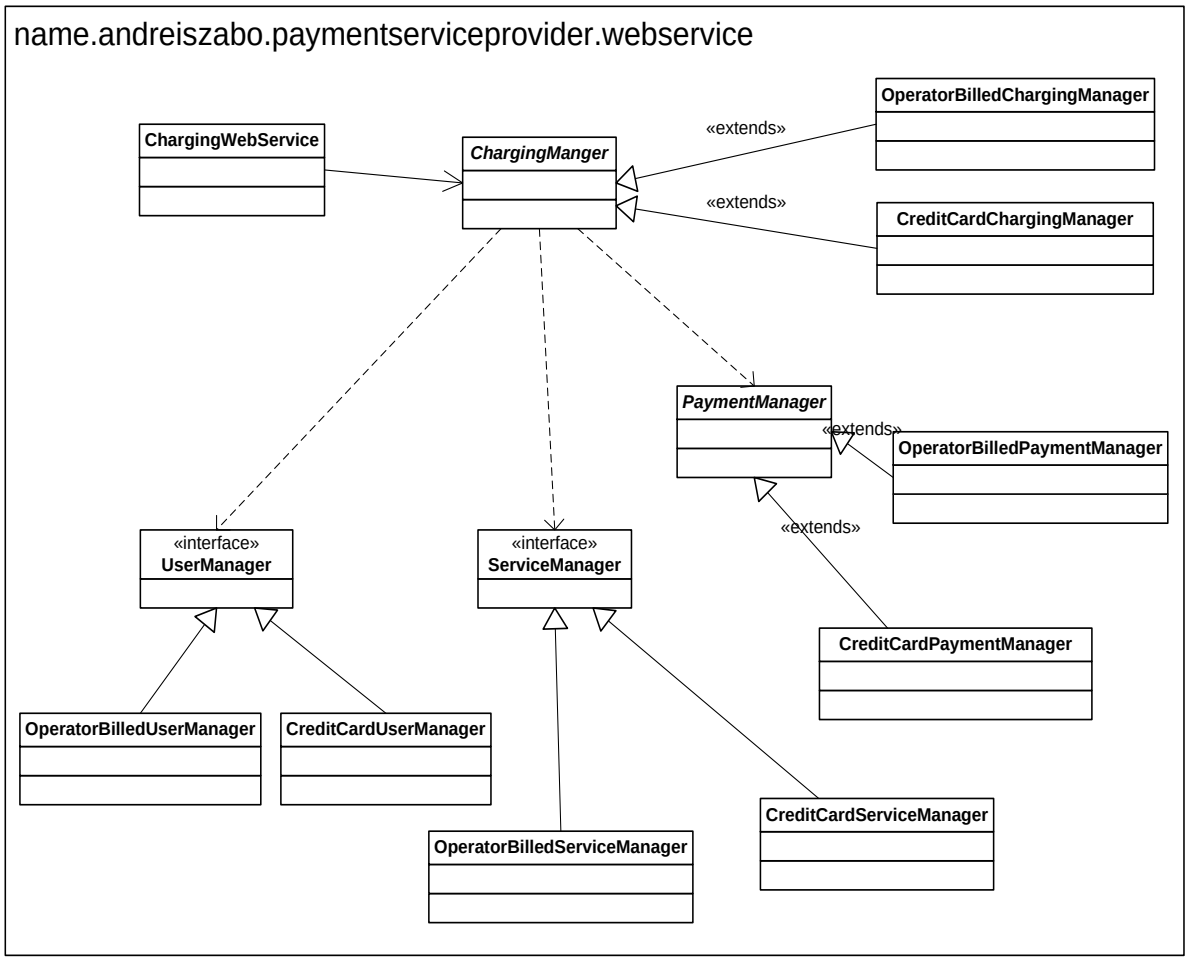


Figure 14: Static class diagram of the Charging Web Service

The **user manager** is in charge of the relationship between the mobile user and the payment service provider. Its main functions are sending the contract to the user, requesting user authentication and sending the payment receipt. The user manager acts as an interface to the communication medium used for the sending the contract and the authorization request and receiving the authentication attempt. In our case, SMS technology is used as the communication medium for the requests and responses.

The main task of the **service manager** is the creation of the charging session contract. Some of the data to be used in the contract is specified by the content provider when the charging session is initiated; this allows for a great flexibility in terms of the transaction type, unit value of the provided service a.s.o. For example, if the content provider would specify information like the unit value of a service at registration time, it would not be able to change that price

without having to re-register or somehow edit the information already stored by the payment provider. Data that likely will not change, like the name of the content provider, is taken from the information entered by the content provider upon registration. Another task of the service manager is the creation of the payment receipt that will be sent to the mobile user when the transaction is complete; the data that is used for the creation of the payment receipt is taken from the charging session details (i.e. total amount charged and payment type).

The **payment manager** interface controls the process of settling payment transactions. This would involve integration with an existing network operator. The payment manager takes the details of payment from the charging session, and, based on these, the transaction is settled and a payment receipt is returned to the mobile user. Once a charging session was settled, it presents no interest anymore; however, it is still kept in the persistence storage, for potential statistical data gathering purposes.

The operations provided by the framework need to be flexible and fit a variety of use cases:

- the content provider sells digital goods in the form of MP3 music files. The price of each downloaded file is €1 or each song can have a different price. At the onset of the charging session, the content provider would call a startChargingSession to initialize a charging session. When the download has finished, the content provider would call a chargeAndRelease operation; this has the effect of settling the charging session.
- the content provider sells digital goods in the form of news video streaming service. The unit in this case could be 1 minute of watching the service and the unit price could be €1 for example. The content provider would call a startChargingSession operation at the beginning. Afterwards, at each 1 minute, the content provider could call a charge operation with the number of units being one and than call chargeAndRelease when the user stops watching the service. On the other hand, the content provider could keep track of the time elapsed since the user started watching the service, and call only chargeAndRelease at the end with the proper number of units. This would have the added benefit of minimizing network traffic.

The example content provider developed for the testing the framework, follows closely the first use case; for implementation details see section *5.2.2 Charging Web Service*. Because it would be unrealistic to gain access to a real mobile network infrastructure for the purposes of this project, the settling of payments is simulated.

5. Framework Implementation Details

This chapter provides some of the implementation details of the framework. It starts with an overview of the used technologies and continues with detailed implementation information relating to the components of the framework.

5.1 Implementation Environment and Tools

This section presents the software environments, tools and technologies used to develop and test the framework.

5.1.1 Programming Language: Java

Java was used as the programming language for the framework; the version used is J2SE 1.4.1. Reasons for which it was chosen include its support for web services through Apache Axis, its support for server side programming and its platform independence.

5.1.2 Apache Tomcat

The Apache Tomcat [37] servlet container is the official reference implementation for the Java Servlet and the JavaServer Pages technologies. Apache Tomcat is developed in an open and participatory environment and released under the terms of the Apache Software License. It was used to install Apache Axis and deploy the web services on a network. It was also used to deploy the registration component of the application and the example content provider web site.

5.1.3 Apache Axis

Apache Axis [xx] is essentially a SOAP engine, a framework for constructing SOAP processors such as clients, servers and gateways. It is written in Java, but a C++

implementation is also under development. Besides being a SOAP engine, Axis also includes a simple stand-alone server, extensive support for WSDL, tools that generate Java classes from WSDL files and vice-versa, a tool for monitoring TCP/IP packets a.s.o.

The best part of Axis is that it has support for the deployment of a web service by exposing Java classes as the service. Hence, all that is required to deploy a web service is to write a regular Java class and tell Axis, through a descriptor file, which methods of the class to expose as a web service; Axis takes care of everything else.

5.1.4 Now SMS Gateway

To enable the communication between the framework and the mobile user and to deploy the payment system in the real mobile domain, a SMS gateway was needed. The one that was used is Now SMS gateway, which is also a MMS/WAP gateway and has a free 60-day trial version available online at [xx]. The most important features of the SMS gateway are that it supports SMS connectivity via one or more GSM modems (or GSM phone connected to an infrared adapter as in our case) and that 2-way SMS for interactive application development is supported. These are the main features that it was chosen for, and also the fact that it is easy to install and configure comparative to one of the most successful open source SMS/WAP gateways, namely Kannel.

5.1.5 Log4j

Inserting logging statements into code is a low-tech method used for debugging. In some cases, it may be the only way though as debuggers are not always available or applicable; this is often the case of distributed applications. Log4j makes it possible to enable logging at runtime without modifying the application binary. The Log4j package is designed in such a way that the logging statements can remain in the shipped code without incurring heavy performance cost. Logging behavior can be controlled by editing a configuration file and never touching the application binary.

One of the features of Log4j is the notion of inheritance among loggers. Using a hierarchy of loggers, it is possible to control which statements are logged at an arbitrarily fine granularity but with great ease. The target of the log output can be a file, a remote Log4j server, a remote Unix Syslog daemon, or even a NT Event logger among many other output targets.

5.2 Framework Components Implementation Details

This section will present implementation details of both the payment framework and the example content provider developed to test the functionality of the framework. The framework was implemented to support two payment methods: operator billed and credit card payments. Both the mobile user and the content provider would have to be registered with the same payment method in order for the charging session to begin. Hence, a content provider would have to register with all the payment methods they wish to support for a service.

The overview starts with the registration component, continues with the charging web service and the content provider and ends by looking at two utility packages; these were developed to provide utility functions that are used by the components of the framework.

5.2.1 Registration Component

This section will present some implementation details regarding to the development of the registration component of the framework. The component was developed in three separate packages, each of them having its own purpose. These are described in the followings.

i. **`name.andreiszabo.paymentserviceprovider.registration.dbaccess`**

This package implements the layer that provides access to the underlying database. It is implemented using the DAO design pattern described in section *4.2.2 Data Access Object (DAO).*

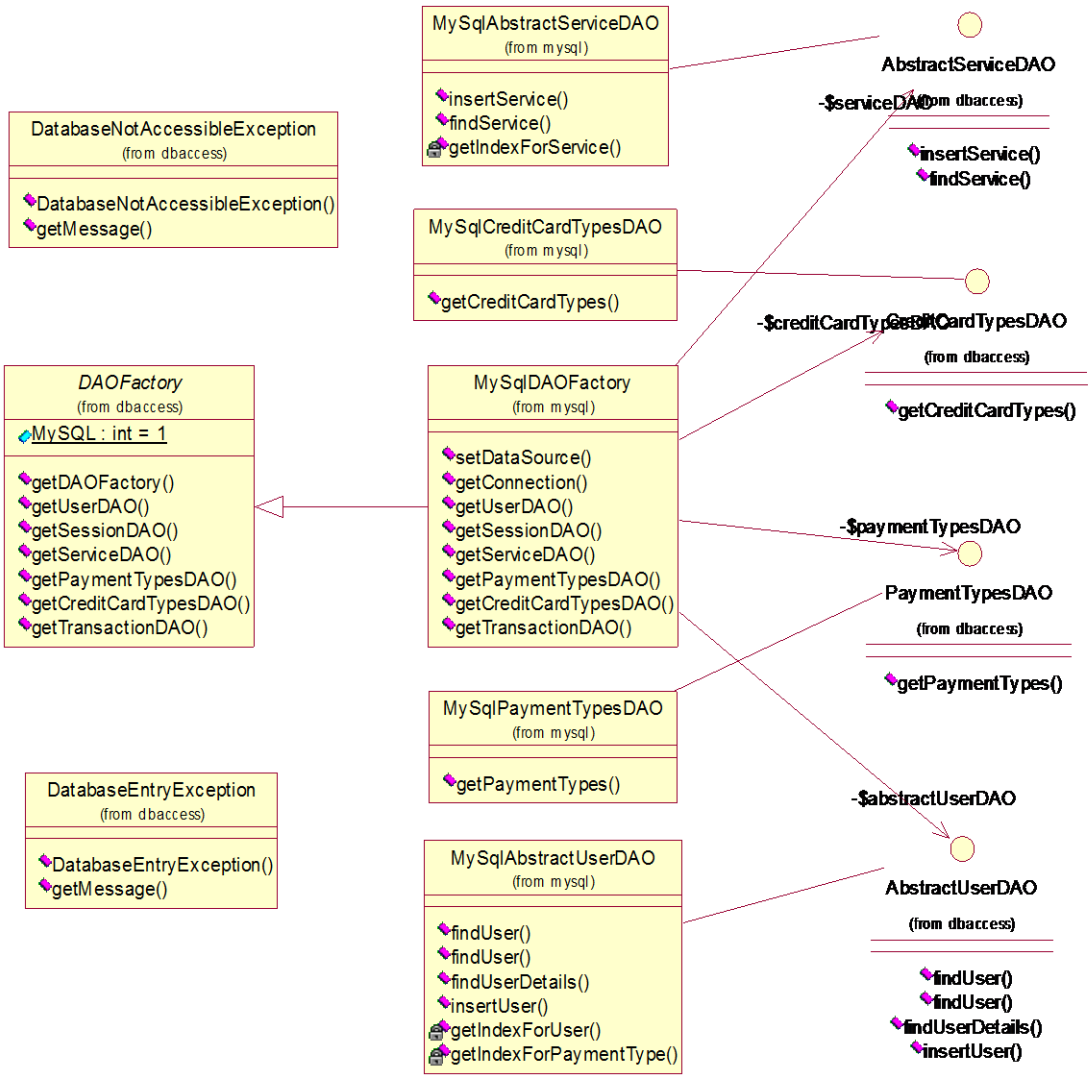


Figure 15: `dbaccess` Package Class Diagram

The `DAOFactory` class produces the DAOs required to work with the different underlying databases. This strategy uses the factory method implementation in the factories produced by the abstract factory. The DAOs produced by the `MySqlDAOFactory` are `AbstractUserDAO`, `PaymentTypesDAO`, `CreditCardTypesDAO` and `AbstractServiceDAO`. This strategy, of using the abstract `DAOFactory` class makes it easy to support different storage mechanisms; different concrete DAO factories that implement storage-specific access only have to inherit and implement the `DAOFactory` abstract class.

The package also defines two custom exceptions, `DatabaseEntryException` and `DatabaseNotAccessibleException`. These are used as wrappers around exceptions

generated at lower levels. For example, if a `SQLException` occurs when trying to access the database, the exception is caught by the database access layer and wrapped around a `DatabaseNotAccessibleException` that is thrown to the upper layer. This makes sense as a client working with the framework does not need to know what is the underlying persistence mechanism used, nor does it need to know details about how to use it. The `DatabaseEntryException` is thrown whenever a client tries to register twice with the framework using the same information, or whenever a lookup operation is performed in the database and there are no matching records.

Transfer objects are used as data carriers. DAO objects use these objects to receive and to return data to clients. In our implementation, the transfer objects are located in the second package of the registration component.

ii. `name.andreiszabo.paymentserviceprovider.registration.businesslogic`

The class diagram of the package is depicted in the following figure:

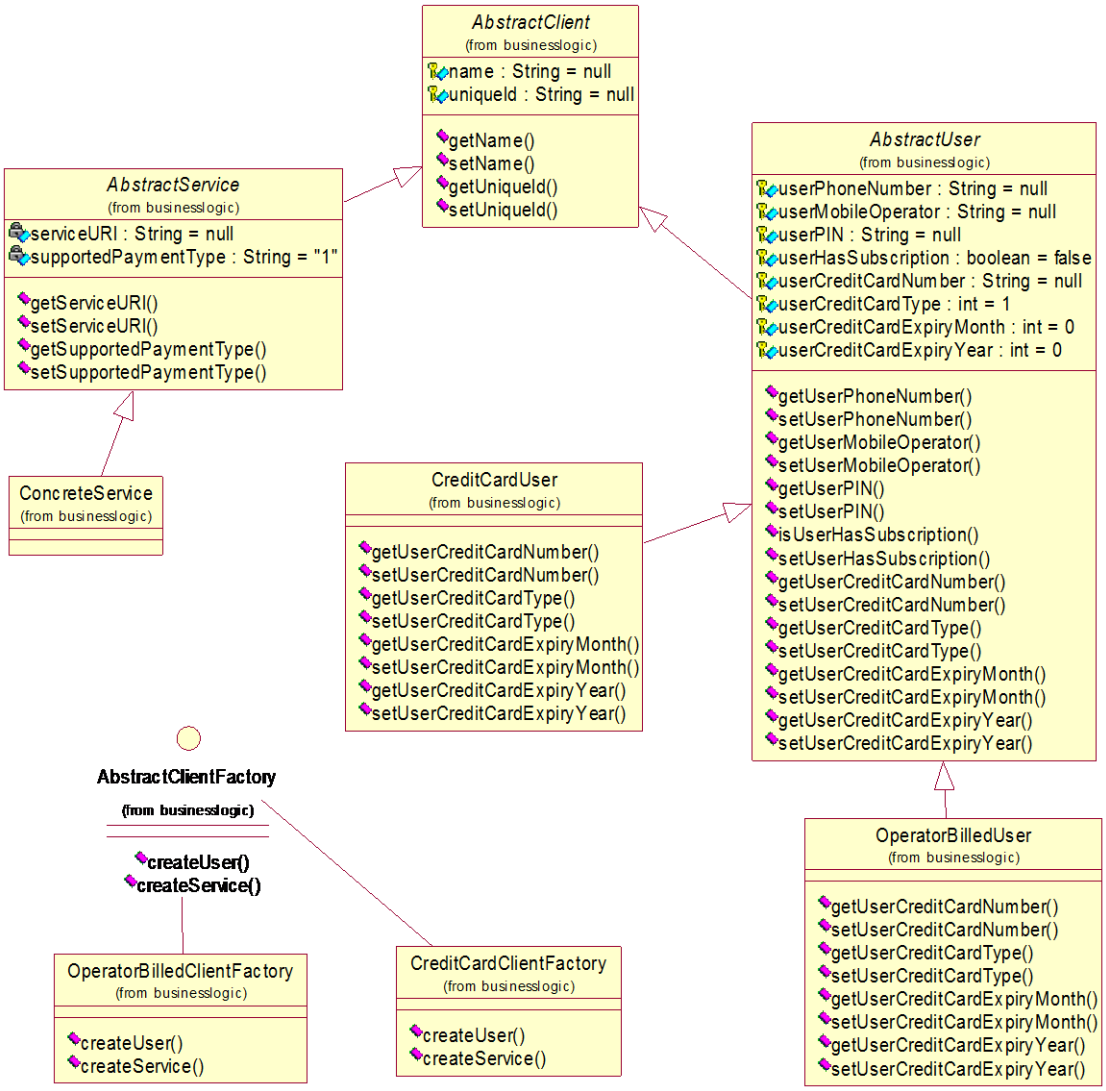


Figure 16: `businesslogic` Package Class Diagram

The clients of the framework can be classified in two categories: mobile users and content providers. Both mobile users and content providers have a name and a unique id; hence the `AbstractClient` abstract class which is inherited by two abstract classes `AbstractService` and `AbstractUser`. Each of these classes add attributes specific to content providers and mobile users respectively. We have one single concrete service, modeled by the `ConcreteService` class that implements the `AbstractService` class and two concrete mobile users: `CreditCardUser` and `OperatorBilledUser` each of them implementing the `AbstractUser` class.

The package also uses the abstract factory design pattern throughout the `AbstractClientFactory` class. This factory is implemented by two different concrete factories `OperatorBilledClientFactory` and `CreditCardClientFactory`. Each of these factories knows how to create a new mobile user and content provider corresponding to their respective payment methods; i.e. the `OperatorBilledClientFactory` knows how to produce an `OperatorBilledUser` and a `ConcreteService`.

iii. `name.andreiszabo.paymentserviceprovider.registration.webinterface`

This package contains the beans and servlets that, together with the JSP files and the HTML files, make up the web interface of the framework. It is made up of two separate packages: the **`beans`** and the **`servlets`** packages.

The **`beans`** package contains a number of Java bean classes. These are used to validate the information that the user inputs in the JSP files. Each JSP file has a bean associated with it and, when the user is finished entering data in the forms of a particular JSP file, the corresponding bean checks the values of the data. If the data is valid, the user is allowed to go on, otherwise the page is redisplayed and the user is asked to enter the correct values for the data. The `beans` package also contains the `PopulateDropDown` class; its role is to populate the drop down boxes from the JSP files with data taken from the database.

The **`servlets`** package contains the logic of the web interface. The most important class, `ControllerServlet`, as its name suggests, controls the logical flow of the registration process. Requests are forwarded from each JSP file to the `ControllerServlet`, and based on where the requests comes from, the servlet will forward it to the corresponding destination, that can either be a JSP page or another servlet. The `InitServlet` takes care of reading the initialization parameters from the deployment descriptor file of the web application and making them available to the rest of the framework's component; the class also initializes the connection pool that will be used to access the database. The other two classes of the package, `AddUserServlet` and `AddServiceServlet` can only be reached when the `ControllerServlet` forwards execution to them. Their role is to add a new user, respectively a new content provider, to the database, by taking the information forwarded to them as parameters in the HTTP request.

5.2.2 Charging Web Service

This section will describe some of the implementation details of the charging component of the payment provider framework. The component is developed in two main packages that will be described in the followings.

The operations that are exposed as web services are defined by the `ChargingWebService` class, which acts as a stateless façade to the underlying system.

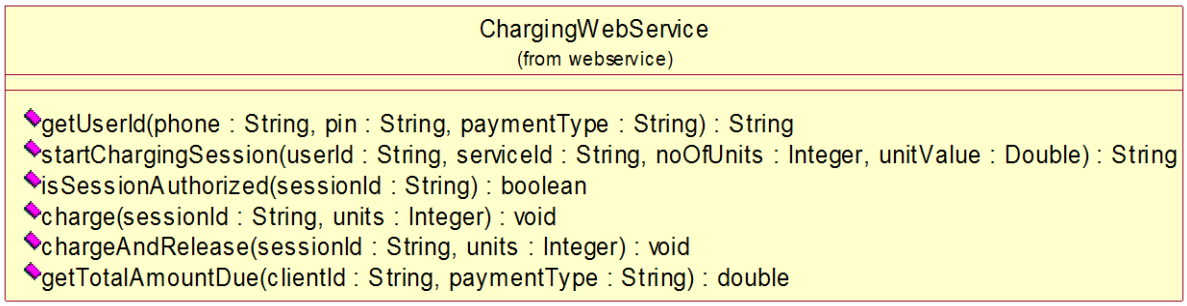


Figure 17: `ChargingWebService` Class Diagram

The operations are described in the followings:

- `String getUserId(String phone, String pin, String paymentType)` – this is called by the content provider to check if the mobile user wishing to gain access to the content provider's services or goods is a registered user with the payment provider. The id of the user is returned if the user is found to be a registered client, null otherwise. As one can see, a mobile user is identified by its phone number, pin number and desired payment type.
- `String startChargingSession(String userId, String serviceId, Integer noOfUnits, Double unitValue)` – this is called by the content provider to start a new charging session. It specifies the parties that take part in the transaction (identified by their ids) and the number of units with their unit values that are charged at the onset of the session. A contract specifying the details of the transaction is sent to the mobile user at this time. A charging session id is returned if everything goes well, null otherwise.

- `boolean isSessionAuthorized(String sessionId)` – after starting the charging session, but before allowing access to the service or good that it is selling, the content provider checks to see if the user has authentified and authorized the session. If yes, access to the restricted area is allowed, otherwise the user is asked to authenticate herself and authorize the session.
- `void charge(String sessionId, Integer units)` – this is used by the content provider to charge an additional number of units to a charging session that is already active. This could be the case of an online video streaming service, where the user is charged some agreed amount for each 1 minute of service for example.
- `void chargeAndRelease(String sessionId, Integer units)` – this is called by the content provider to settle a charging session. It allows also the content provider to charge a number of units to that session and than settle it.

i. `name.andreiszabo.paymentserviceprovider.webservice.dbaccess`

This package provides the access layer to the persistence mechanism. It is also implemented using the DAO design pattern. The class diagram of the package is the following one:

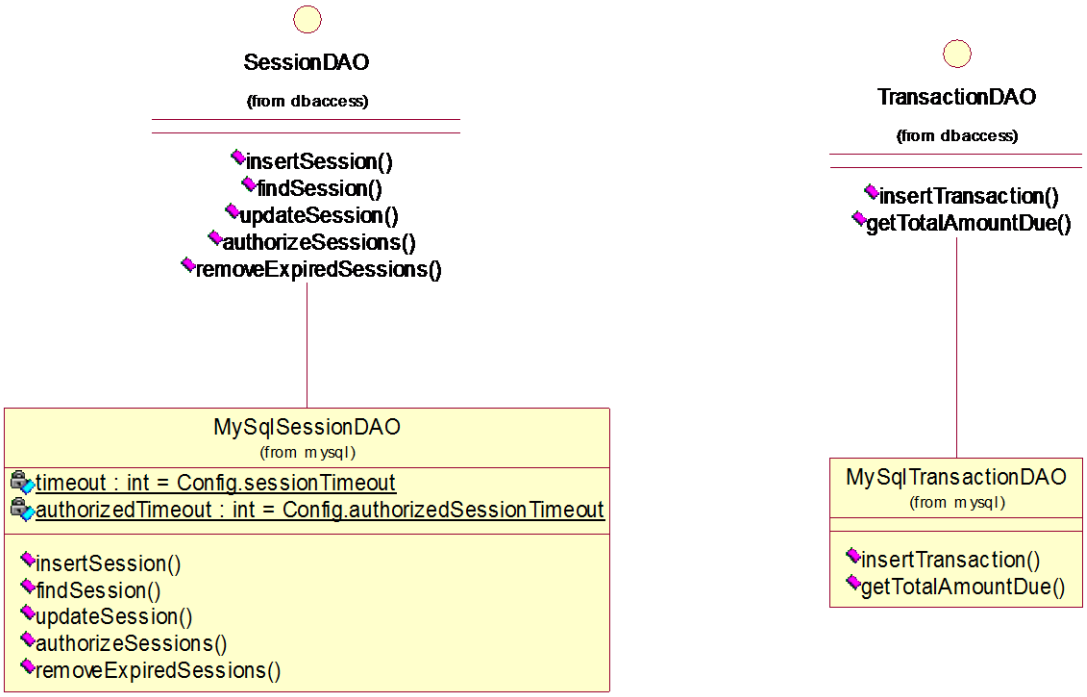


Figure 18: `dbaccess` Package Class Diagram

The `SessionDAO` and `TransactionDAO` interfaces define all the operations that can be performed upon `Session` and `Transaction` objects respectively. The `MySqlSessionDAO` and `MySqlTransactionDAO` represent concrete implementation of the corresponding interfaces for the MySql database. They contain the necessary SQL code required to work with the database and they shield the client from those details.

ii. `name.andreiszabo.paymentserviceprovider.webservice.businesslogic`

The second package of the component defines all the classes that represent the entities of the domain model. The `Session` and `Transaction` classes are used to create the so-called transfer objects. A `Session` object holds all the information about a particular charging session while a `Transaction` object holds all the information of a transaction. Whenever a charging session is finished and it needs to be settled, a transaction is created and the client id together with the payment type and the total amount spent, are written in a separate table of the database; the table is called `accounting`. Because gaining access to a real network operator's infrastructure for the purpose of the project was unrealistic, there was no other way of properly "settling" a charging session. Thus, the above method is used, where the `accounting` table holds entries of finished charging sessions in the form of (userId, totalAmount, paymentType).

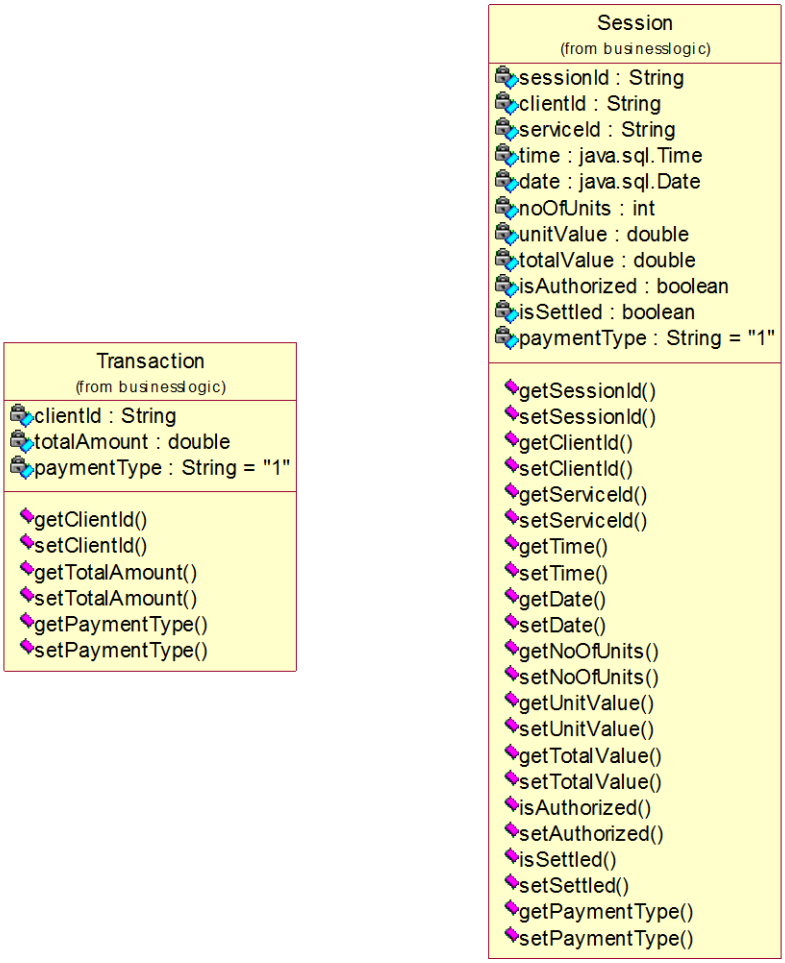


Figure 19: `Session` and `Transaction` Class Diagrams

The `PaymentReceipt` class contains all the relevant information that will be sent to the mobile user at the end of the charging session. This information includes the total amount charged during the session and the payment method. The class implements a `toString()` method that generates the text that will make up the payload of the SMS message sent to the user.

The `Contract` class, similarly, contains all the relevant information that will be sent to the user at the beginning of the charging session. This information includes the content provider's name, the payment method, the number of units and the unit value. The class also implements a `toString()` method that generated the text that will make up the payload of the SMS message sent to the user.

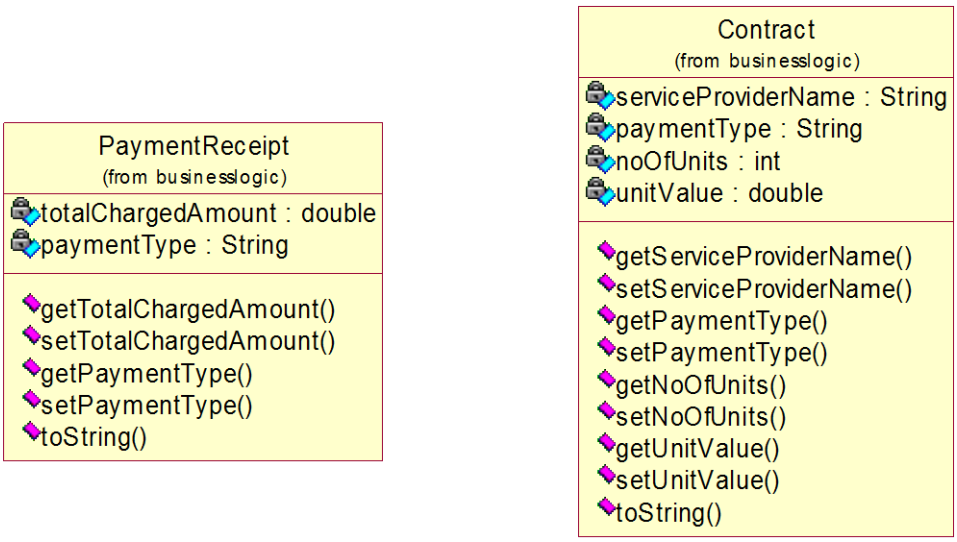


Figure 20: `PaymentReceipt` and `Contract` Class Diagrams

The `UserAuthorization` class implements the interface of the framework with the `name.andreiszabo.smsgatewayaccess` package used for communication with the SMS gateway. It implements two methods for this, one that takes a `userId` and a `Contract` object and sends the text version of the `Contract` to the mobile user, and the other one that takes a `userId` and a `PaymentReceipt` object and sends the text version of the `PaymentReceipt` to the mobile user. The class uses the database access layer of the registration component for retrieving the phone number of the user whose user id is given as parameter to the methods; however, the database access layer is not used directly but throughout the `DAOFacade` class.

The `DAOFacade` performs database related operations that do not depend on the type of user or service being accessed. The purpose of the class is to shield the web service operations from the underlying database implementation and the corresponding DAO objects. Thus, all calls to the database access layer, both of the registration component and that of the charging component, go through this class.

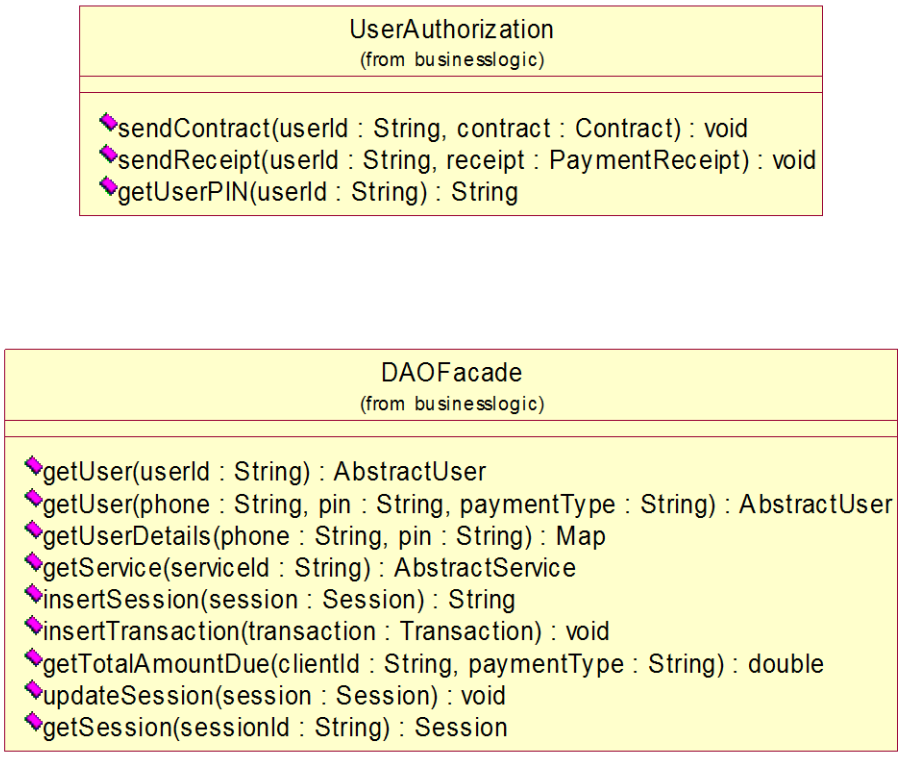


Figure 21: `UserAuthorization` and `DAOFacade` Class Diagrams

The `ChargingManager` interface is an abstract factory for the implementation, which creates three abstract products called managers. Each of these managers handles a proportion of the charging web service, from the perspective of one of the parties involved. The tasks of the `ChargingManager` interface are the following:

- creates a new charging session given a userId, serviceId, number of units and unit value;
- releases a charging session given by its sessionId;
- returns the authorization status of a charging session given by the sessionId;
- charges a session given by its session id with the number of units specified.

The `PaymentManager` controls the process of settling payment transactions. This would involve integration with existing mobile payment infrastructures. It uses the session contract details to determine the attributes of a charging session and following this, the transaction is completed and a payment receipt is returned to the user. The only operation performed by the class is settling a charging session. This involves marking the session as being settled, creating a `Transaction` object from the charging session details and persisting the attributes

of the `Transaction` object in the `accounting` table of the database. The `Transaction` object is written to the database using the `TransactionDAO`.

The `ServiceManager` extracts relevant information from the charging session details and construct both the payment agreement between the parties and the payment receipt. Both of these will be sent to the mobile user in the form of SMS messages. Thus, the `ServiceManager` knows how to create the `Contract` and `PaymentReceipt` objects given a `Session` object.

The `UserManager` represents the interface to the communication medium. Its two main tasks are that of asking the `UserAuthorization` class to send the contract and the payment receipt to the mobile user. It passes on `Contract` and `PaymentReceipt` objects to the `UserAuthorization` class; the class knows how to get the text version of those objects.

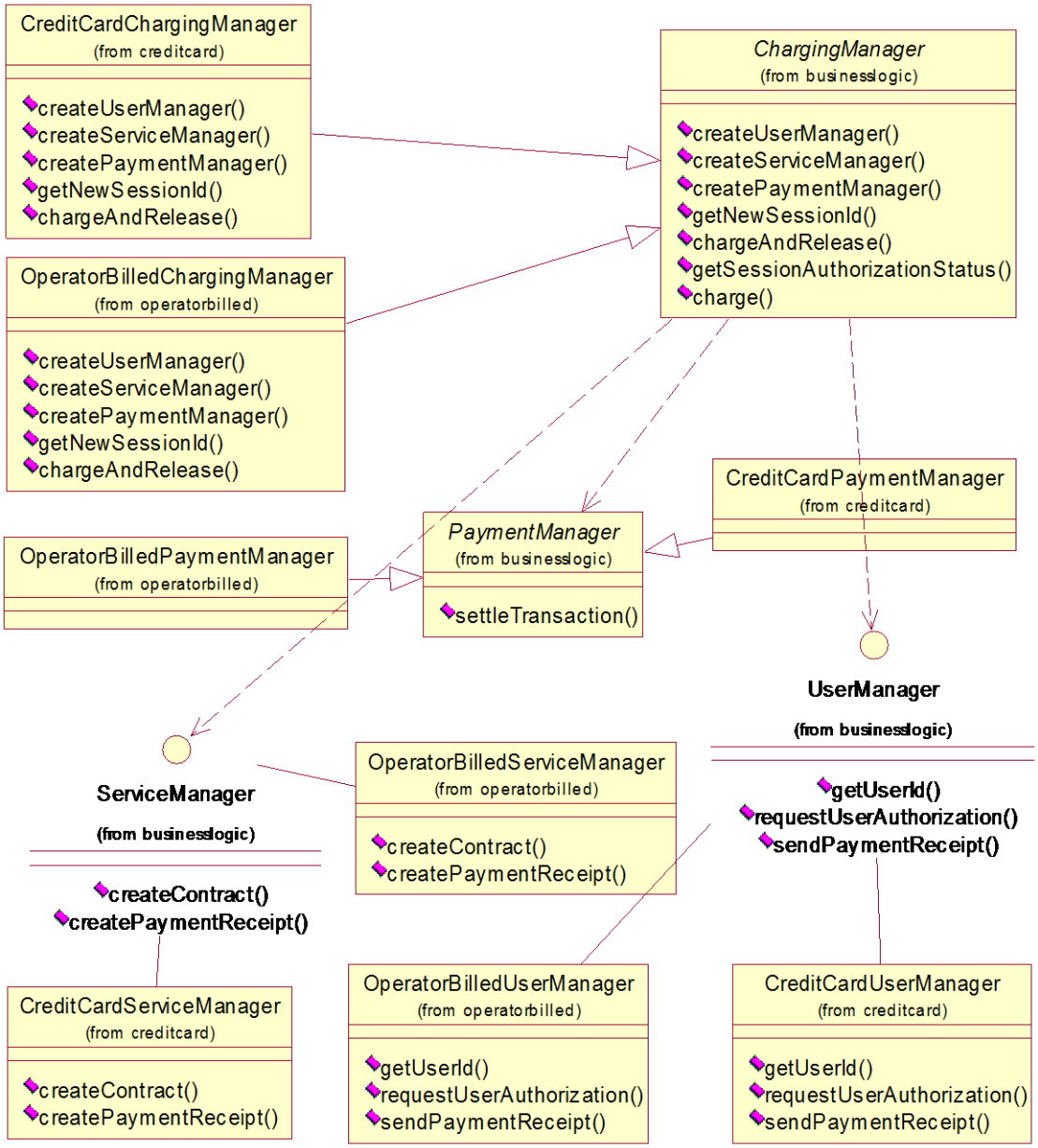


Figure 22: Managers Class Diagrams

5.2.3 Content Provider

The content provider was implemented with the intent of testing some of the services of the payment framework; it was not intended to be a fully-fledged online store or anything similar. The most important requirements for the content provider were to allow access to the restricted area only to users that are registered with the payment provider and to make the

content available in such a way that user's cannot access it without having to pay for it. The intended functionality of the content provider was that of allowing registered users to download an MP3 file and charge the user accordingly for this.

Each content provider is assigned a unique id upon registering with the payment framework. The content provider has to register for each payment method it wants to support so it might end up with several ids. It has to keep track of these ids because when accessing the services of the framework it must provide one of these ids. In our case, these ids are stored in the deployment descriptor of the content provider web application; see section *6.2.3* `web.xml` for more details on this.

The content provider was implemented as a simple web application consisting of two main servlets: a `ControllerServlet` and the `SendMp3File` servlet. The `ControllerServlet` checks to see if the user is registered with the payment provider and, if yes, it starts a charging session and allows access to the restricted area. If not, it asks the user to register first. Checking if the user is registered and starting a charging session are done by calling the corresponding services of the payment provider; see section *4.3.2 The Charging Web Service*.

When the user selects a file to download, the `SendMp3File` servlet checks if the charging session was authorized by the user. If yes, the servlet sends the selected file to the user and terminates the charging session by calling `chargeAndRelease`. Adding support for new MP3 files is easy. The directory from where the servlet takes the files is specified in the deployment descriptor file of the application; see section *6.2.3* `web.xml`. To add a new MP3 file to the list of available songs, one has to modify the JSP file by adding the song name and place the appropriate file in the specified directory. The servlet will be able to find the file; there is no need to modify the servlet code at all.

5.2.4 Utility Code

There are two packages developed that provide common functionality used by the other components of the framework. These are the **`name.andreiszabo.util`** and **`name.andreiszabo.smsgatewayaccess`**. In the followings, we are going to take each

package and look at its functionality.

The **`name.andreiszabo.util`** has only a single class, namely `UniqueIdGenerator`. Its purpose is to generate unique ids; these ids are assigned to each mobile user and content provider that registers with the framework. Charging sessions are also assigned a unique id generated by this class. The algorithm works as follows: when a new unique id is needed, a random number is generated using `SecureRandom`. Next, a `MessageDigest` of the random number is created, the `byte[]` returned by the `MessageDigest` is encoded into some acceptable textual form and the result is checked if it is already being used; if it is not already taken, it is suitable as a unique identifier. The `MessageDigest` class is suitable for generating a "one-way hash" of arbitrary data. That hash values never uniquely identify their source data, since different source data can produce the same hash value. A `MessageDigest` takes any input, and produces a String which: is of fixed length, does not allow the original input to be easily recovered (in fact, this is very hard), does not uniquely identify the input; however, similar input will produce dissimilar message digests.

The **`name.andreiszabo.smsgatewayaccess`** contains all the classes that make up the interface to the communication medium. It contains the `SMS` class; this defines a SMS object as being identified by source and destination phone numbers and a text message. The `SMSSender` class takes care of effectively interfacing with the SMS gateway. Its main role is to take a `SMS` object instance and, using the `HttpClient` package from Jakarta, builds the HTTP requests by which the SMS is handed over to the SMS gateway that will eventually send it. The identification information for the SMS gateway, like its IP address and user credentials, has to be specified in the deployment descriptor file of the web application; see section *6.2.3 `web.xml`* for how to do this.

Another important class of the package is `SMSReceiver`. This is developed as a servlet; its main role is to process incoming SMS messages. The SMS gateway has to be set up to support 2-way SMSs and must be configured so that all incoming SMSs are forwarded via HTTP requests to the `SMSReceiver` servlet. For an overview of how to do this, see section *6.1.2 Installing the Now SMS Gateway*. The servlet takes the source phone number of the SMS message and the PIN sent in the body of the message, and, if the PIN is valid, goes to the database and authorizes all charging sessions of the respective user.

6. Users Guide

This section will provide some brief information about how to setup up the payment service provider and the example content provider. It provides generic information about the steps required to install all the components used by the system and information about how to deal with the various configuration files.

6.1 Deploying the System

6.1.1 Installing and Setting Up the MySQL Database

The system was tested on the free MySQL database engine. The versioned that was used is 3.23.58; it can be downloaded free of charge from [38]. For installation notes, see the MySQL documentation.

The database that is used by the system is available as an SQL script file in the archive file that contains the application. This makes it easy to deploy the database under MySQL; for the exact steps required to do this, check the MySQL documentation. After deploying the database, a user name and a password should be set up and granted with full access right to the database; a record should be kept of the user name and password as these will be used to configure the connection pool.

For accessing the database from the Java programming language, a driver is required. The one that was used in our case is MySQL Connector/J, the official JDBC driver for MySQL. It can be obtained from [39] free of charge. For information on how to use it, check the documentation that comes with it.

6.1.2 Installing the Now SMS Gateway

The SMS Gateway used by the framework for sending and receiving messages is Now SMS Gateway. A trial version of 60 days of it can be downloaded free of charge from [40].

A user name and password should also be assigned for accessing the SMS gateway; this can be done using the graphical interface of the gateway. The user name and password will be required later to configure the payment provider application. The last step should be connecting the mobile phone to the computer and making it visible to the gateway. For our purposes, a Nokia 7110 phone was used, and it was connected to the computer via an infrared USB adapter. For different phone models and other ways of connecting them to the PC, as well as for more information on the setup of the gateway, see the documentation that comes with it.

6.1.3 Deploying the Payment Service Provider

Apache Tomcat was used as the servlet container to perform the test. The payment service provider comes in the file called `psp.war`. Deploying war files in Tomcat is easy; one just has to copy the war file in the `webapps` directory of Tomcat. This is however not enough to get the application running, as there are several configuration files that need to be edited; see the following sections on how to do this.

After deploying the web application, we must somehow export part of our application as web services. We have done this using Apache Axis [41]. For information on how to do this, [42] provides an excellent working example and much more.

6.1.4 Deploying the Example Content Provider

The example content provider was developed as a proof of concept. It is not meant to be neither a production ready example, nor an exhaustive usage example of all the features provided by the payment service provider; it is only test case of how the functionality exposed by the framework as web services could be used by a potential content provider looking for a payment provider.

These being said, the content provider is deployed easily by dropping the `contentprovider.war` file into the `webapps` directory of Tomcat. Further configuration through properties files is however required; see next sections on how to achieve this.

6.1.5 Support for SSL

For security purposes, both the registration of new clients, throughout the web interface of the payment provider, as well as the web service calls, take place using the Secure Sockets Layer (SSL) security protocol. SSL is a protocol designed by Netscape Communications to enable encrypted, authenticated communications across the Internet; it used mostly in communications between web browsers and web servers. URL's that begin with 'https' indicate that an SSL connection will be used. SSL provides three important things: privacy, authentication and message integrity.

The steps required for the exchange of SOAP messages over an HTTPS secure connection with mutual authentication can be summarized as: create self-signed certificates for both the client program and the Tomcat server; store the certificates and the corresponding private keys in respective keystore files; populate the client's truststore with the server's. A detailed example of this process can be found in [43]. For information on how to set up Apache Tomcat to use SSL, see [44].

6.2 Configuration Files

6.2.1 `log4j.properties`

Log4J is used for logging purposes. Both the content provider and the payment service provider use this; hence, there are two `log4j.properties` files located as follows: one of them in the `webapps/PSP/WEB-INF/classes` and the other one in `webapps/ContentProvider/WEB-INF/classes`. These directories are located in the Tomcat home and may have a different name if the web applications were deployed under a different context path name.

The payment service provider has different loggers for different modules of the framework. By default, the logging levels of all the loggers are set to `info` and each logger records its messages in the specified file. Let us take a sample entry from the file:

```
# Database accesses are logged by DBAppender; these are the
#properties for DBAppender.
log4j.logger.name.andreiszabo.paymentserviceprovider.registration.
dbaccess=info, DBAppender
log4j.appender.DBAppender=org.apache.log4j.RollingFileAppender
log4j.appender.DBAppender.File=c:///Program Files/Apache
Group/Tomcat 4.1/webapps/PSP/logs/db.log
log4j.appender.DBAppender.MaxFileSize=100KB
log4j.appender.DBAppender.MaxBackupIndex=1
log4j.appender.DBAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.DBAppender.layout.ConversionPattern=%d{ABSOLUTE} %-
5p %c{2} - %m%n
```

To change the level that determines the messages that are logged just change for example `info` to `debug`. This will cause all messages generated by the `dbaccess` package to be logged to the specified file. For an in depth look at the various configuration options available with log4j properties files, see [45].

6.2.2 `server.xml`

This file is located in the `conf` directory of the Tomcat home. The modifications that are done in this file, concern two purposes: the SSL configuration and the connection pool configuration. The SSL configuration is described in detail in [44], while the connection pool configuration steps are covered in [46].

6.2.3 `web.xml`

`web.xml` files are also known as web application deployment descriptor files. Hence, each web application must have one. The file is located in the `WEB-INF` directory of both the content provider application and the payment service provider.

The content provider's file has three configuration parameters:

- `mp3-dir`: specifies the location from where the mp3 files that are sold by the content provider are taken and sent to the client.
- `operatorBilledId` and `creditCardId`: this is an id assigned to the content provider upon registration with the payment service provider. When making calls to the web services of the payment service provider, the content provider has to send along its id. For each different payment method the content provider register with, it receives a unique id.

The payment service provider also has a number of configuration parameters:

- `serverIP`: represents the IP address of the SMS gateway that is used to send the messages.
- `portNumber`: represents the port number on which the SMS gateway listens for connections.
- `userName`: represents the credentials used to connect to the SMS gateway.
- `userPassword`: represents the credentials used to connect to the SMS gateway.
- `database`: this parameter is used to specify the underlying database engine used for the database. In the current implementation, only MySQL is supported as a persistence mechanism. Hence, the only valid value of this parameter is 1.
- `sessionTimeout`: represents the number of seconds after which an unauthorized session (i.e. a session that was started and not authorized) is considered expired and deleted from the database.
- `authorizedSessionTimeout`: represents the number of seconds after which an authorized, but not settled session is considered expired and deleted from the database.

7. Evaluation and Conclusions

This section outlines some of the criteria that should be used when evaluating the framework. It also presents future work that could be done to improve the system and ends with a paragraph outlining the conclusions of this paper.

7.1 Evaluation and Future Work

7.1.1 Network Failure and Latency

Network latency considerations were not taken into account during the initial design of the framework. In simulations and theory, latency is not an issue, but with the deployment of the framework in a real mobile network, it was found necessary to adopt some design features to offer support for this problem. During the design, the system was modeled to use a synchronous authorization mechanism of the user. Upon sending the SMS message to the user requesting for authorization, the implementation would wait for the reply of the user. However, latency issues involved in sending the reply, caused by both the network and the inherent difficultness of writing messages on a mobile device, have determined the changing of the authorization process from a synchronous one to an asynchronous one. Thus, once the SMS message asking authorization was sent, the process would cease; only when the user would have replied and requested the service from the content provider, the process would be started again and completed this time.

7.1.2 Security

Security is a major concern in the world of mobile communications and that of electronic payments. When the two domains overleap security becomes even more of an issue.

The registration process of the framework encapsulates the sending of sensitive information (like credit card numbers, expiry dates, a.s.o.) over the network into s single transaction. The user only has to enter this sensitive data only once, namely at registration time. Afterwards, payments can be made by only specifying the mobile phone number and the PIN that identify

a user; the PIN number must still however be safeguarded. To reduce the chance of information being stolen while sent through the network, the registration process works using the SSL protocol for security reasons. This way, the information entered by the user is encoded, sent over the wire to the server, where it is decoded. The communication between the content provider and the payment provider, namely the web service calls, also go over SSL connections. Thus, practically all-sensitive information exchanged between the actors goes over SSL connections.

7.1.3 Future Work

Further development of the framework could involve the use of Enterprise Java Beans (EJB) technologies, the inclusion of web services security and integration of other payment methods in the framework.

The use of EJBs would provide additional support for concurrency, scalability and security issues while web services security would allow security at application level and not transport level as SSL provides.

7.2 Conclusion

The goal of this paper was the research of the mobile payments area and the development of a provider of mobile payment services. At the time of the research, there were many enclosed proprietary solutions for the problem of mobile payments, but no open, generally accepted standard. Much effort was invested in the study of these proprietary solutions, but, due to their enclosed nature, this became soon difficult.

Research of the mobile payment services revealed the existence of three main actors: the mobile client, the content provider and the payment service provider. There are a set of common relationships between each actor, which can be expressed as a set of generic interactions between the actors. Combining web technologies and some well-known design patterns, a generic payment provider was implemented based on web services. The framework was designed and implemented to support two different payment methods. To further evaluate

the framework, an example content provider was developed and the framework was deployed using a real mobile network operator.

Among the encountered difficulties, worth mentioning is the absence of an open standard in the field of mobile payments. This has slowed down the design and implementation of the framework and made it harder. Another drawback was latency in real mobile networks and the inability of sending data in the form of cookies or something similar to mobile devices; these were addressed through design decisions.

The proposed framework for mobile payments does not provide a complete solution for all mobile payments, but it merely suggests an open and interoperable solution for a generic mobile payment services provider.

7. Bibliography

[1] Mobile Payment Forum, http://www.mobilepaymentforum.org

[2] GSM Association, http://www.gsmworld.com

[3] NTT DoCoMo Inc, http://www.nttdocomo.com

[4] WR2000 Group LLC, http://123merchantaccount.com/glossary.html

[5] CommerceNet Scandinavia**,** http://www.mobile.commerce.net

[6] Aaron McPherson, *The Evolution of Mobile Payments*, Financial Insights Feb 2004

[7] Paul Budde, *Wireless Technology - Mobile Communications - Satellite, Data, and Fleet*

[8] Mobile Streams Ltd, http://www.mobilegprs.com

[9] Alan Sicher, Randall Heaton, *GPRS Technology Overview*, Dell White Paper Feb. 2002.

[10] Amit Vyas, Peter O'Grady, *A Review of Mobile Commerce Technologies*, Department of Industrial Engineering, University of Iowa, May 2001

[11] Nokia, http://www.nokia.com

[12] Telefonaktiebolaget LM Ericsson, http://www.ericsson.com/cdmasystems

[13] The UMTS Forum, http://www.umts-forum.org

[14] 4Gcouk Limited, http://www.4g.co.uk

[15] Dr. Subrahmanyam Karuturi, *What is SMS?*, http://www.funsms.net

[16] *MMS (Multimedia Messaging Service) Explained*, January 2002, http://www.esato.com

[17] *Multimedia Messaging*, http://www.nokia.com

[18] *WAP Explained*, http://wapwala.net

[19] NTT DoCoMo http://www.nttdocomo.com

[20] Mobile Streams Ltd, http://www.mobileussd.com

[21] Symbian Ltd, http://www.symbian.com

[22] PalmSource, Inc., http://www.palmsource.com

[23] *What's NEW in BREW Version 2.0*, http://www.devx.com/Brew/Article/10187

[24] *Java 2 Platform, Micro Edition (J2ME)*, http://java.sun.com/j2me

[25] *Microsoft .NET Compact Framework*, http://msdn.microsoft.com/mobility/netcf

[26] *Mobile Payments: Preparing for the mCommerce Revolution*, Trintech White Paper, March 2002

[27] PayCircle, http://www.paycircle.org

[28] The Parlay Group, http://www.parlay.org

[29] Simpay Limited, http://www.simpay.com

[30] *Charging Mechanisms for Mobile Services*, NorthStream White Paper, 2002

[31] *Reverse SMS Billing*, http://www.fastsms.co.uk/reverse-billing

[32] Mobipay International, http://www.mobipay.com/en/home.htm

[33] David Chappell, Tyler Jewell, *Java Web Services*, O'Reilly, March 2002

[34] *Abstract Factory*, http://www.dofactory.com/Patterns/PatternAbstract.aspx

[35] *Core J2EE Patterns - Data Access Object*, http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html

[36] *Façade*, http://www.dofactory.com/Patterns/PatternFacade.aspx

[37] The Apache Software Foundation, http://jakarta.apache.org/tomcat/

[38] MySQL AB, Database engine download, http://dev.mysql.com/downloads/mysql/3.23.html

[39] MySQL AB, Driver download, http://dev.mysql.com/downloads/connector/j/3.0.html

[40] Now Wireless Limited, http://www.nowsms.com/downloads/smsmmsgateway.htm

[41] The Apache Software Foundation, http://ws.apache.org/axis

[42] Erik Hatcher and Steve Loughran, *Java Development with Ant***,** August 2002, Chapter 15 – Working with Web Services

[43] Pankaj Kumar, *J2EE Security for Servlets, EJBs and Web Services*, Chapter 11 – Web Service Security

[44] *SSL Configuration HOW-TO*, http://jakarta.apache.org/tomcat/tomcat-4.1-doc/ssl-howto.html

[45] Ceki Gülcü, *Short introduction to log4j*, March 2002, http://logging.apache.org/log4j/docs/manual.html

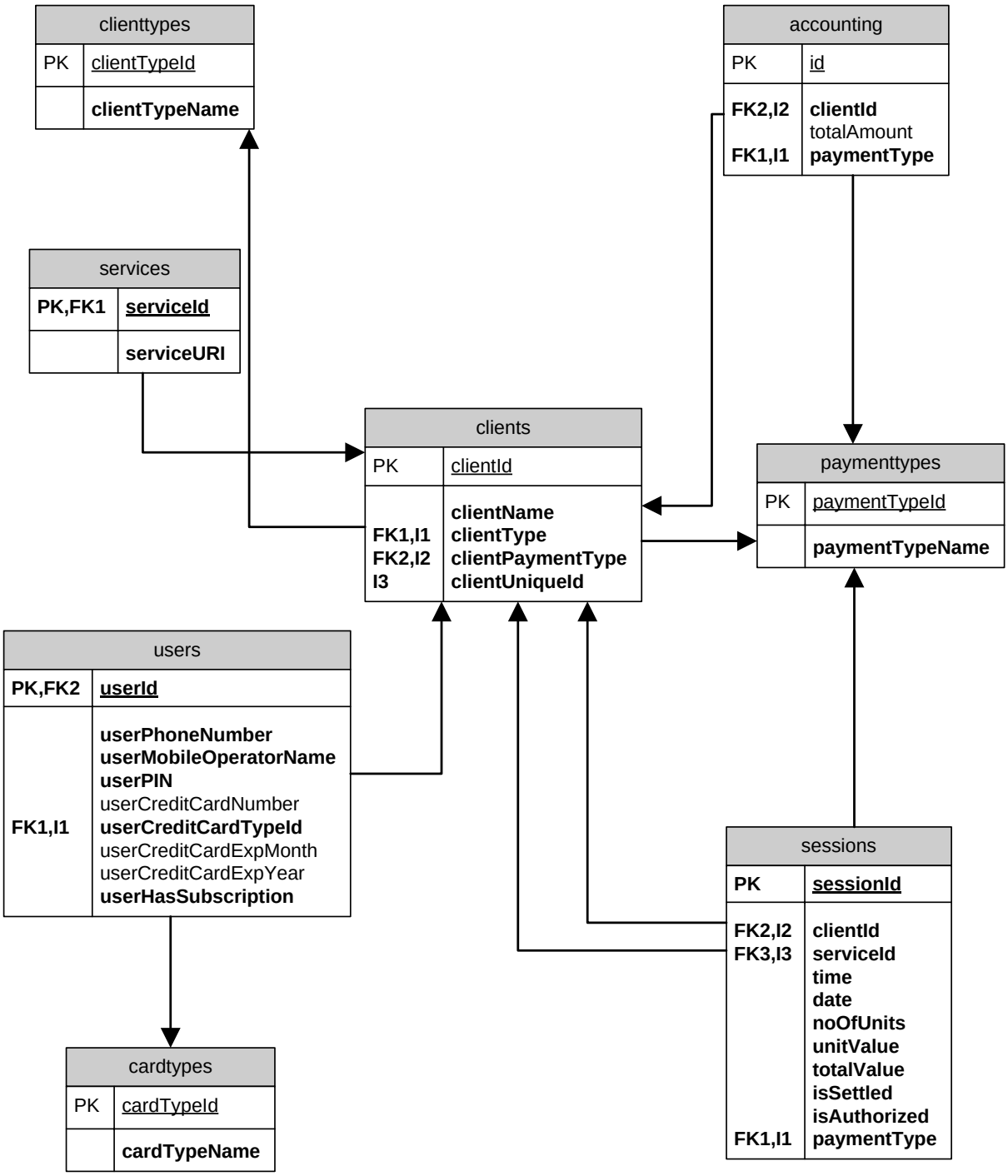
[46] *JNDI Datasource HOW-TO*, http://jakarta.apache.org/tomcat/tomcat-4.1-doc/jndi-datasource-examples-howto.html

[47] Bruce W. Perry, *Java Servlet & JSP Cookbook*, O'Reilly, January 2004

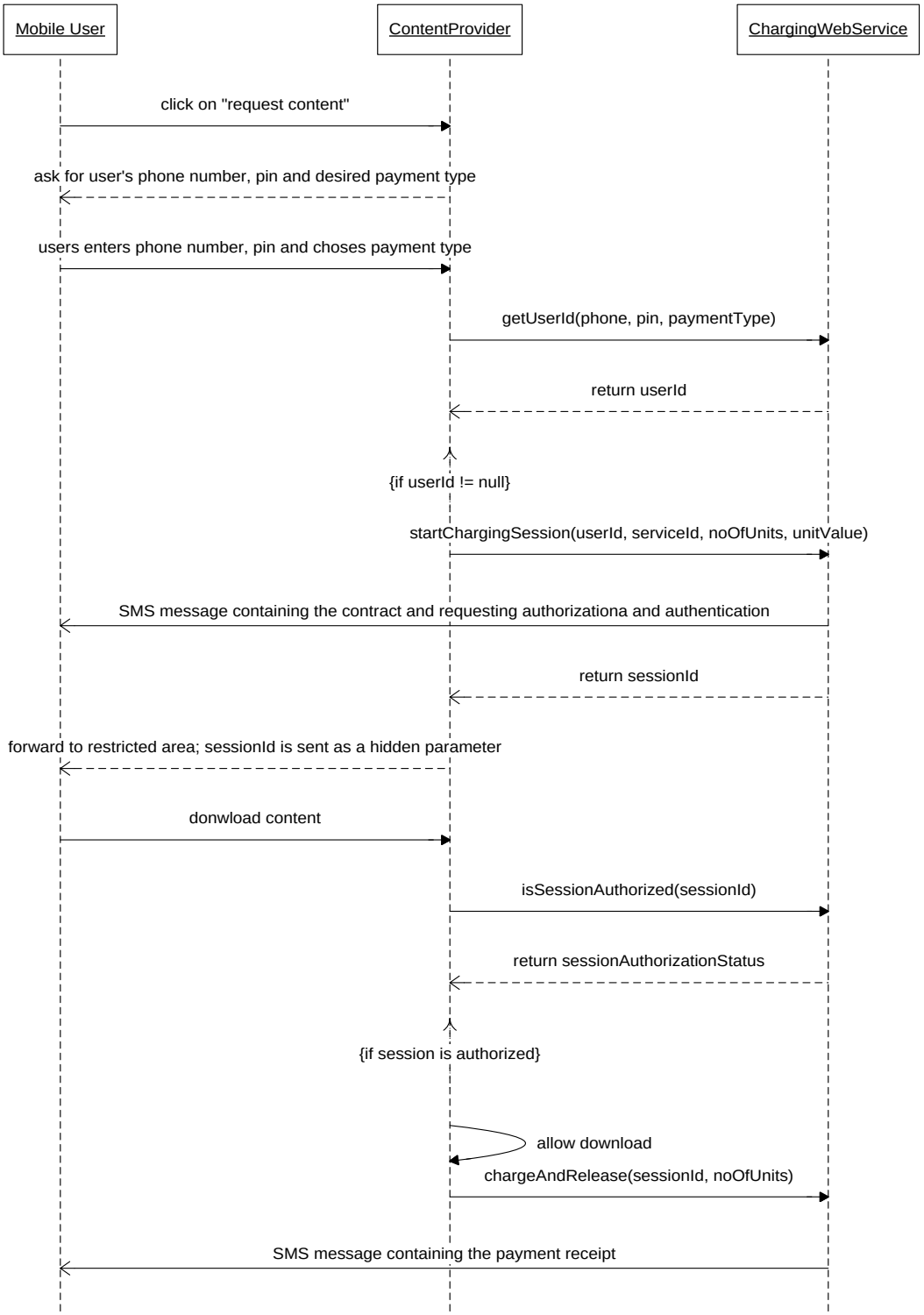
[48] Brett McLaughlin, *Java and XML*, O'Reilly, 2000.

Appendix

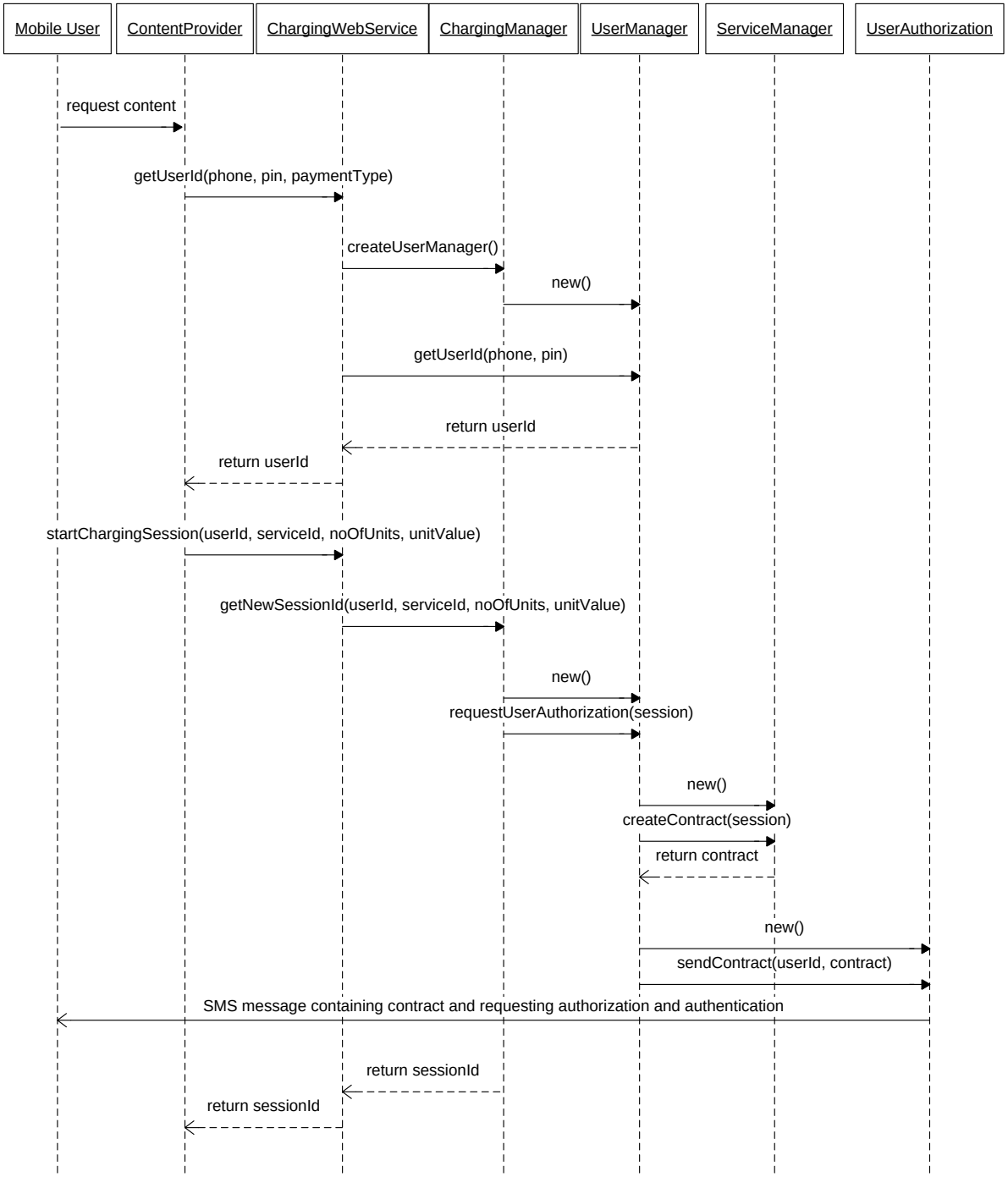
A. Entity Relationship Diagram of the Database



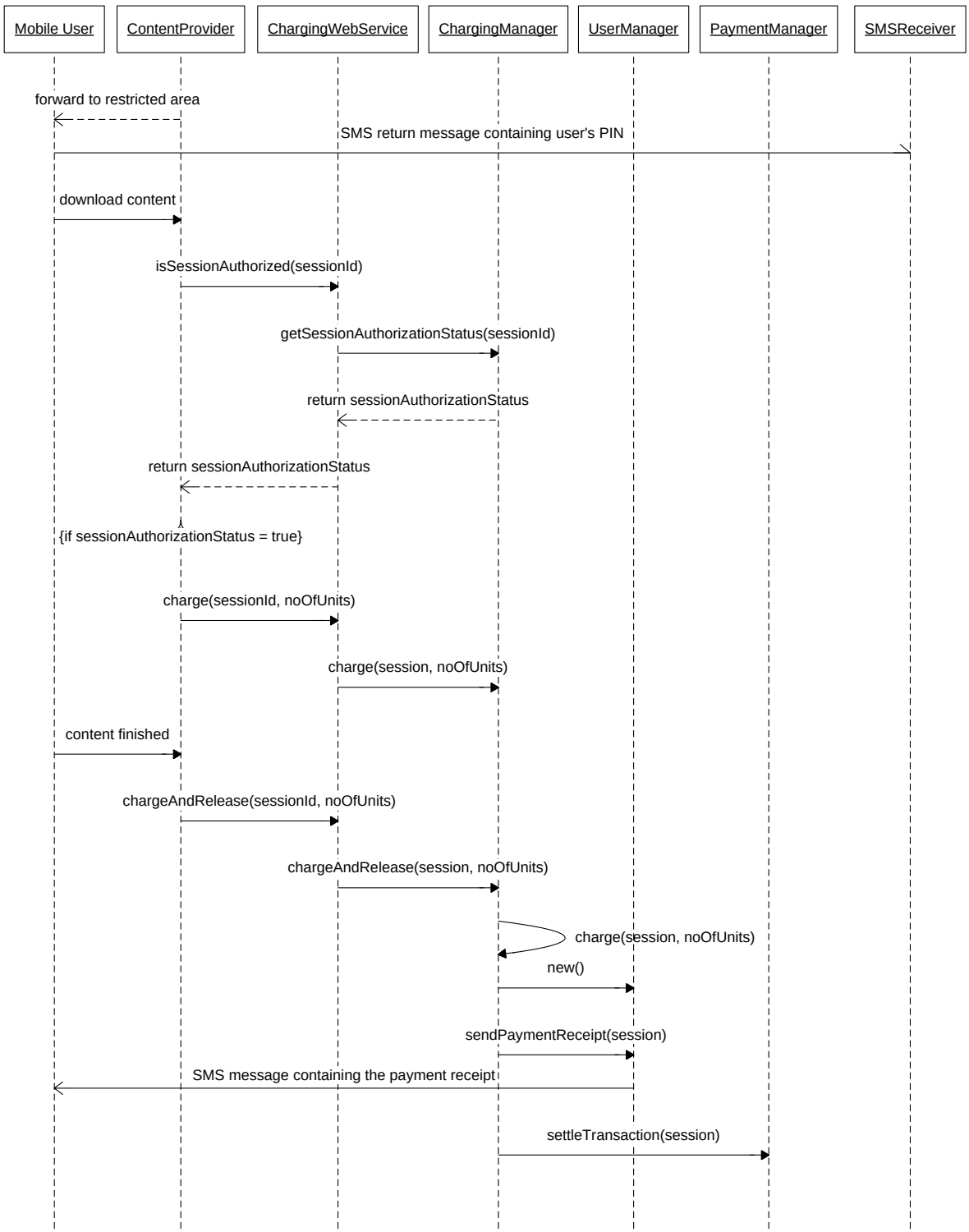
B. Charging Sequence Diagrams



APPENDIX


Mobile User
ContentProvider
ChargingWebService
ChargingManager
UserManager
ServiceManager
UserAuthorization
request content
getUserId(phone, pin, paymentType)
createUserManager()
new()
getUserId(phone, pin)
return userId
return userId
startChargingSession(userId, serviceId, noOfUnits, unitValue)
getNewSessionId(userId, serviceId, noOfUnits, unitValue)
new()
requestUserAuthorization(session)
new()
createContract(session)
return contract
new()
sendContract(userId, contract)
SMS message containing contract and requesting authorization and authentication
return sessionId
return sessionId

APPENDIX


Mobile User
ContentProvider
ChargingWebService
ChargingManager
UserManager
PaymentManager
SMSReceiver
forward to restricted area
SMS return message containing user's PIN
download content
isSessionAuthorized(sessionId)
getSessionAuthorizationStatus(sessionId)
return sessionAuthorizationStatus
return sessionAuthorizationStatus
{if sessionAuthorizationStatus = true}
charge(sessionId, noOfUnits)
charge(session, noOfUnits)
content finished
chargeAndRelease(sessionId, noOfUnits)
chargeAndRelease(session, noOfUnits)
charge(session, noOfUnits)
new()
sendPaymentReceipt(session)
SMS message containing the payment receipt
settleTransaction(session)