Ateneo de Manila University Convergent Messaging Platform

In partial fulfillment of the requirements in

CS 197: Integrative Project

submitted to

Luis F. G. Sarmenta, Ph. D.
William Emmanuel S. Yu
Department of Information Systems and Computer Science

submitted by

Ryan Gilbert Garcia
Frederic Esguerra

Ateneo de Manila University
Loyola Heights, Quezon City

24 February 2004
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>3</td>
</tr>
<tr>
<td>I. Introduction</td>
<td>3</td>
</tr>
<tr>
<td>II. Scope and Limitations</td>
<td>3</td>
</tr>
<tr>
<td>III. Review of Related Literature</td>
<td>4</td>
</tr>
<tr>
<td>IV. Project Description</td>
<td>6</td>
</tr>
<tr>
<td>V. Benefits</td>
<td>6</td>
</tr>
<tr>
<td>VI. System Architecture</td>
<td>7</td>
</tr>
<tr>
<td>A. Use Case Diagram</td>
<td>7</td>
</tr>
<tr>
<td>B. System Overview</td>
<td>8</td>
</tr>
<tr>
<td>C. System Architecture</td>
<td>9</td>
</tr>
<tr>
<td>D. Class Diagram</td>
<td>10</td>
</tr>
<tr>
<td>VII. Methodology</td>
<td>11</td>
</tr>
<tr>
<td>VIII. Results and Conclusion</td>
<td>11</td>
</tr>
<tr>
<td>IX. Writing and Deploying Applications</td>
<td>12</td>
</tr>
<tr>
<td>X. Current Applications in Development</td>
<td>12</td>
</tr>
<tr>
<td>XI. Requirements and Specifications</td>
<td>13</td>
</tr>
<tr>
<td>References</td>
<td>13</td>
</tr>
</tbody>
</table>
Abstract

The Ateneo de Manila University Convergent Messaging Platform basically covers three things:

i) Connectivity modules to interface with various messaging infrastructure and devices (GSM07.05, CIMD, HTTP, etc.)

ii) A framework for developing mobile messaging applications and services

iii) An application container

This application container highlights efficiency and provides an easy to use framework for institutions wanting to take advantage of mobile messaging services. Having a wide range of possibilities for mobile messaging services, it would benefit any institution, from big corporations to small organizations, to use this application container. While the framework lessens development time for programmers to produce mobile messaging applications, the container provides for easy management and more efficient utilization of mobile messaging infrastructures. For any large institution, it would be costly to provide each of its sub-entities with their own connection to cellular network providers; or provide these sub-entities with mobile messaging devices.

The application container’s main goal is efficiency, through reliable handling of, possibly, many applications running simultaneously. An extension application would also be provided to act as a client application that would basically allow for visibility of remote or external resources.

I.] Introduction

Mobile messaging services appear to have an unlimited amount of possible applications. The range would vary from registration systems, to financial information systems. Implementing simple mobile messaging applications, however, require tedious work from developers; primarily due to the fact that developers of such applications are obliged to write lengthy code to interface with an SMSC, a mobile device or perhaps other devices related to mobile messaging. Each of the mentioned devices has distinct protocols, and having to learn to interface with each of them consumes considerable amount of development time. This emphasizes that most mobile messaging application are written from scratch which wastes both time and effort.

In addition, large organizations which have numerous subordinate entities would have to avail of multiple connections with network providers, possibly increasing the costs for the said organization. These organizations end up with many cluttered mobile messaging services to support the numerous business processes that the organization may have.

A framework for writing mobile messaging service applications and a centralized container to manage such applications would thus decrease costs for any organization wanting to utilize mobile messaging technology; the organization shall benefit from decreases in both development costs and deployment costs.

II.] Scope and Limitations

The objectives are to define a framework and to write an application container for mobile messaging. The product shall scale to suit the needs of businesses and organizations of different sizes. The main tasks are to first, design a mobile messaging application framework to simplify the development process of writing mobile messaging applications, then to build a robust application container to allow multiple mobile messaging services at a time. The project will consist of a web-based user interface for the mobile messaging application container, to manage multiple user accounts and mobile messaging applications. The web interface shall facilitate in managing the mobile messaging services allowing addition and deletion of new services, and producing service usage reports.

An extension application will also be developed and this lightweight application shall serve as a remote client to the mobile messaging application container allowing the execution of messaging applications on remote locations possible. This model may typically be employed in order to make visible, other external resources. For instance, a department may already have legacy systems, local databases, machineries, devices or any other information system that they may want to expose via SMS messaging, but are restrained by the fact that these resources are distant from the server running the application container that shall be implemented. Thus the need for the proposed extension
application, on which users may choose deploy their mobile messaging applications into, is emphasized.

The application container may link with varied mobile messaging related infrastructures and devices such as SMS centers, MMS centers, mobile phones or GSM modems depending on the connectivity modules available. Some of the connectivity modules that shall be provided in this project include the CIMD protocol, Nokia AT-command protocol and the HTTP protocol. These varied protocols shall be written in such a way that they are structured as pluggable modules.

III.] Review of Related Literature

There are only a few similar works available at the time of implementation. Most of them had functionality similar to the proposed system. Most of the projects found did have the capability to interface with different protocols; however, the proposed system has the intention to cover functionalities that these works lack. Three key functionalities that are an advantage for the proposed system are:

1. to provide a generalized framework for mobile messaging application development
2. to provide a centralized application container for mobile messaging applications
3. to provide facilities for management of mobile messaging services and mobile messaging infrastructures

Some of the similar projects are illustrated below:

a.] “SMSC-Gateway” is a software package that can be used to send/receive SMS to and from cellular phones using various connections to SMSCs (Short Message Service Centers). Currently, the following connections are implemented

- GSM modem (AT commands).
- ISDN terminal adapter for X.31 connection to SMSC (EMI protocol).
- TCP/IP connection to SMSC (EMI protocol).

SMSC-Gateway implements a client-server architecture allowing multiple applications to utilize the connections available, whether on SMSC or an ME(Mobile Entity / Cellular phone). Inputs fed to SMSC-Gateway are plain-text entries that consist of a list of recipients (at least one), an empty line, the message text and a final line consisting of a dot. Shell scripting may be one of the ways used to implement a messaging service for this product. This application includes a web interface that gives a summary of the usage of the messaging infrastructure that it manages.

![SMSC-Gateway Interface](image)

b.] “SMSC-Sender” is another software application enabling you to dispatch SMS messages to mobile phones through SMS Centers based upon the TAP and UCP protocols. This however does not provide any means for development of services; rather it simply provides a graphical user interface on which users may work on to utilize the messaging infrastructure behind. Commonly, this is used to broadcast SMS messages to multiple recipients taking away the difficulty of having to input these messages on a cellular phone’s keypad.
<table>
<thead>
<tr>
<th>Software</th>
<th>Platform</th>
<th>Language</th>
<th>Client-Server</th>
<th>CIMD</th>
<th>AT Commands</th>
<th>SMS-Service Application Framework</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMSC-Sender</td>
<td>Windows</td>
<td>ASP</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>$79</td>
</tr>
<tr>
<td>SMSC-Gateway</td>
<td>Linux</td>
<td>C</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Free</td>
</tr>
<tr>
<td>Proposed System</td>
<td>Java</td>
<td>Java</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 - Comparison Table

Some of the few key terms important for the project:

A.] Computer Interface to Message Distribution (CIMD)
Cellular phone networks often have multiple messaging centers (SMSC, MMSC) to manage the messaging traffic. SMS messages are not sent directly from sender to recipient; rather, the messages first go through a messaging center before being dispatched to the receiver. There are varying types of messaging centers, some of which act as store and forward devices (i.e. SMS Center) while others immediately attempt to submit the messages (i.e. UUSD Center) as soon as the receiver is located. In order to provide richer content, cellular phone networks often allow external businesses to connect to their equipment. CIMD is one of the more popular protocols used by external applications to communicate with SMS message centers.

B.] AT Commands
AT commands program a variety of modem hardware settings and were adopted by other modem manufacturers who wanted to market their wares with the coveted phrase Hayes-compatible. Sending and receiving of SMS messages may be executed by a computer connected to a mobile device (Mobile Phone / GSM Modern) by transmitting AT commands over some link (IrDA, Bluetooth, Data Cable). Nokia has made available AT command specifications particular to their mobile phones.

C.] Mobile Messaging service application
We would like to define a mobile messaging application as a packaged archive containing class files and deployment descriptors that shall encapsulate all application logic concerning the manipulation of incoming mobile messages and the consequential mobile message replies if any.

D.] Java Management Extensions
JMX is the best answer if one needs good management and monitoring. Most organizations are faced with the challenge of extending their business to the realities of e-commerce without sacrificing their hard-earned systems already firmly in place. JMX offers both flexibility and extensibility. It allows these applications to be exposed as manageable
components rather than having to waste time on duplicating already existent code. JMX offers a new development model on which developers are encouraged to abstract a problem into components; where these components are easily incorporated together under a single MBean server.

E.] Java Messaging Service

The Java Message Service allows applications to send, receive, create and read information as messages. Software components are able to communicate independent of each other under the asynchronous model offered by JMS. JMS offers two kinds of messaging protocol: point-to-point and published-subscribed. A simple example of a point-to-point is the producer-consumer model where a message can be consumed by only one consumer. The publisher-subscriber model is often utilized for broadcasting messages across a number of subscribers. With both models you have potentially many producers and many consumers, yet in the point-to-point model it is assured that only one consumer gets to consume each message.

F.] Enterprise Java Beans

With enterprise beans, developers are able to implement enterprise-scale applications using a managed component model. There are four types of enterprise beans: session beans, entity beans, message-driven beans, and managed enterprise beans. Entity beans are used to represent the domain-model of a particular problem. Entity beans are often used to represent the persistent entities in a business problem. The session façade design pattern suggests that entity beans should be wrapped around by session beans. Session beans are typically the entities that execute the business processes involved in the enterprise system. Message-driven beans are beans added on to the EJB 2.0 specifications as a complement to the JMS specifications. Message-driven beans are assigned as consumers to JMS queues and topics, opening a new way to interface with EJB’s via JMS.

IV.] Project Description

There are two major components in the system. First of which, is the centralized enterprise application which is tasked to manage multiple mobile messaging applications, and the second of which is a scaled-down client applications on which mobile messaging applications may also be deployed. The centralized enterprise application’s task is to manage the mobile messaging applications deployed on it and also the remote client applications that connect to transact with it. This server deals with issues such as transaction management, scheduling, message acknowledgement and the like. The client application, on the other hand, may be utilized by various branches within an organization to host their mobile messaging applications capable of database queries, business logic execution and the like.

V.] Benefits

There are three main advantages that are aimed at in the development of such a framework. First is that the availability of a framework allows for rapid application development. Developers would not need to worry about interfacing with various messaging infrastructure, allowing them to focus more on solving their business problem. Second is that the framework bridges the gap between currently existent information systems and the mobile messaging infrastructure available from cellular network providers. Providing mobile messaging as a new medium to ride on for currently existing systems would be easier to implement. Last, the application container provides a centralized and manageable system. The system does not only show great advantage because of the key benefits mentioned earlier, but it would also benefit any businesses planning to follow the mobile messaging trend in the Philippines. Up until the present, the Philippines has been seen with one of highest volume of SMS usage in the world. This shows that the Philippine market is actively participating in this wireless technology craze. Developing this framework will entice businesses and entrepreneurs to take advantage of the mobile messaging fad in the Philippines.
VI.] System Architecture

The core functionality of the proposed system is graphically represented in the use case diagram:

![Use Case Diagram](image)

The Mobile Messaging Application Container will preferably be the only link to the network service provider/s for an organization. It shall act as the gateway between all the mobile messaging applications within the organization and the mobile messaging infrastructure. It includes role based authentication to manage the end users of the system, and the applications developed by the end users of the system. Once an application developer is given an account for the system, the developer may deploy multiple mobile messaging applications to suit the needs of the department he represents. The application developer shall be responsible for all the content and services the developer may choose to provide, however, these may be overruled by the system administrator in extreme cases. A downloadable container extension will be available for the end users (application developers), on which the developers may choose to deploy their applications instead. Whatever content or services deployed on the remote extension application may also be managed by the system administrator.

The system administrator has the capability to view and manage all mobile messaging applications serviced by the system, and administer all the authorized users of the system.
Future development may include the application container’s interfacing with multiple network providers. This shall allow currently existing mobile messaging applications to service requests from multiple cellular network providers such as Smart, Globe, Sun Cellular, and others. MMS services should also be integrated with the system, granting the same power for SMS messaging offered by the proposed system to MMS messaging technology.

**System Overview**

![System Diagram](image)

Figure 3

J2EE technologies were highly utilized in the implementation of this project. Scalability, speedy development of the project and performance gains are some of the factors that pushed for the employment of J2EE technology. The software is divided into four tiers, each of which, focusing on a specific functionality.

A JBoss service was written to provide an interface for management of the variety of protocols that may be used for both MMS and SMS messaging. The protocols to use for the messaging infrastructures available may be configured during run-time through this service. Since the service was written such that it is compliant to the Java Management Extensions specifications as a JMX Mbean, the component may be ported to other J2EE application container.

All business logic is encapsulated within the Enterprise Java Beans tier. All routing of messages to their respective target messaging applications, and tallying of service usages for report generation purposes are handled by the enterprise beans. As messaging applications are uploaded on to the system, their archives (.jar) are persisted into a database. The messaging application’s class is loaded from the archive and instantiated upon the first time a message requiring the service is received by the system. A reference to the instance of the messaging application is then retained; that all subsequent operations on the messaging application will no longer have the overhead of having to dynamically load the service from the archive persisted in a database.

The remote extension is a Java application that may be invoked as either a console-application or as a swing-GUI based application. The remote extension communicates with the ADMUCMP server using the JMS API. The great thing about the Java Messaging Service specification is that it defines the API on doing asynchronous messaging in J2EE, without defining the communication protocol used behind the API. The communication protocol is left as the responsibility of the J2EE application container provider. Thus, application container providers as JBoss, IBM, and Oracle all come up with a variety of communication protocols that developers may choose from when using the JMS API. JBoss in particular provides a number of communication protocols as RMI, HTTP, and plain sockets to name a few.
A web interface is provided for easy management of the messaging application container. This tier uses the struts framework and it is tightly coupled with the EJB tier using local calls to communicate with the enterprise beans.

![System Architecture](image)

The completion of the project shall allow developers to write applications similar to the example given below. The developers shall archive their application together with some optional deployment descriptors and perhaps any other external libraries or resources, and after which shall upload the archive for automatic deployment. The framework may be utilized to write messaging applications that serve as connectors to existing information systems, regardless of form. A mobile messaging application developer for example, may utilize any functionality that the java programming language offers, such as connecting to another EIS via RMI/IIOP or perhaps through the utilization of web services. The application container in this manner may be seen as a mobile messaging gateway, where the defined protocol of communication is via the implementation of a messaging application. In such cases however, the application developer must be aware that the application being developed will still be subject to external constraints such as firewalls or unavailability of external resources. The basic proposed framework is further illustrated by an example SMS application and a rough sketch of the class diagram that the end user (mobile messaging application developer) should be concerned with.

```java
import net.stereowireless.sducmpp
public class MessagingApplicationExample extends BasicSmsApplication
public SMSMessage onMessageReceived(SMSMessage message)
{ SMSMessage replyMessage = new SMSMessage(message getOriginatorAddress(), message getDestinationAddress());
  replyMessage setMessage("Hello world!");
  return replyMessage;
}
```

![Figure 5](image)
The resulting framework from the project is primarily aimed at shortening the development time for messaging services. Thus, the first step taken was to expose to developers an API that provides only the basic and relevant functionalities of mobile messaging. Any future advancement of the framework will build upon these base specifications. There are a lot of possibilities on enriching the framework further perhaps by providing other powerful functionalities to a messaging application such as timer services or simplified persistence models.

To further hasten the development process for messaging services, an emulator for sms applications was written so as to allow developers to go through black box testing with independence from the messaging application container. This would lessen the burden of having to repetitively deploy sms applications on a production server.

Figure 6

Figure 7
Included as a utility to the project is the development of a CIMD-SMSC emulator, whose main purpose is to enable developers to begin or perhaps resume development without the need for a real connection to a cellular network provider. This shall reduce costs for development, particularly the costs for testing on the infrastructure of network providers for any party interested in mobile messaging using the CIMD protocol. The emulator will serve as a handy tool for future developers in the university.

The emulator is capable of sending and responding to CIMD commands, inclusive of the checksum verification and re-requesting in the case of erroneous packets. The emulator displays a clean summary of the conversation between itself and a client program. A detailed CIMD conversation listing is also available and is illustrated in the succeeding screenshot. Currently, the emulator is capable of simulating message deliveries.

![Emulator Screenshot](image)

**Figure 8**

**VII.] Methodology**

The team used the iterative process in implementing the project, mainly to meet the immediate need of a working system. Designing of the framework and implementation of the container was given top importance. The team focused on the production of the basic functionalities; and enhancements, improvements and implementation of additional features were only addressed on succeeding iterations. The project will be iterated over until the accepted level of quality and full implementation of all features of the software is achieved. Code refactoring and regression testing was done every after the completion of an iteration focused on some functionality.

**VIII.] Results and Conclusion**

A fully functional messaging application container capable of both SMS and MMS messaging was completed in a span of five months. SMS services such as ADMUINFO, an SMS service providing information to students, faculty and staff of the Ateneo, were implemented using the framework.

Future improvements on the framework may include availability of functionality for messaging applications such as simplified persistence models, or timer services. In terms of ease of development, code-generators or IDE’s may be constructed.
IX.] Writing and Deploying Applications

A.] Writing applications for ADMUCMP
Any application that wishes to use ADMUCMP may inherit from
net.ateneo.wireless.admucmp.BASICSMSApplication. The developer should implement the
method onMessageReceived which passes the SMS message as its parameter. It is inside this
method where all the processing and handling of the SMS message is done. The SMS message
contains the originator address, destination address, the content and other attributes related
to SMS messaging. The method should end by returning an SMS message. A simple
application, MessagingApplicationExample, was provided in figure 4 for reference.

B.] Local deployment in ADMUCMP
Archive all the classes and resources needed by the application into a jar file.
Deploying applications would require the developer to login to,
http://apps.wireless.ateneo.net/admucmp/login.jsp. After logging in, the screen will show all
applications currently deployed by the user. The user can either add or remove an application.

C.] Remote deployment in ADMUCMP
Download the ADMUCMPRemote.jar from
http://apps.wireless.ateneo.net/rgarcia/index.html. To run application,

Windows:
  java -classpath ADMUCMPRemote.jar:.
  net.ateneo.wireless.admucmp.remote.ADMUCMPRemote -classname
  MySMSApplication -keyword myapplication

Linux:
  java -classpath ./ADMUCMPRemote.jar:.
  net.ateneo.wireless.admucmp.remote.ADMUCMPRemote -classname
  MySMSApplication -keyword myapplication

X.] Current Applications in Development

ADMUINFO
An SMS application in development is ADMUINFO. This service allows students, parents,
faculty, staff, etc to query any information regarding Ateneo. As of the present time, the ADMUINFO
service focuses on services on student’s classes. Students will be able to check for university
announcements down to their specific departments.
This would greatly lessen the burden of parents and students who have to call the school at
times of storms to check if indeed classes were suspended. In the same way, teachers could easily
post announcements regarding their classes through their mobile devices.

ADMUMAIL
Besides ADMUINFO, another current application in development is ADMUMAIL. This is an SMS
application that caters for both students and teachers inside Ateneo. This application serves people
who use mail servers such as mail.ateneo.edu , mailobservatory.ph, and other mail servers that are
managed by the Campus Network Group(CNG).
XI. Requirements and Specifications

1.) Minimum Software Requirements
   Java 2 Runtime Environment [v.1.4.x]
   J2EE Application Container
   RDBMS

2.) Minimum Hardware Requirements
   Hosting Server
   Either of the following:
      SMSC / MMSC connection
      Mobile device with connectivity hardware (IrDA, data cable, Bluetooth)

3.) Development Tools
   J2SDK 1.4.2
   J2SDKEE 1.4
   JAVA Communications API 2.0
   APACHE ANT 1.5.4
   J2EE Application Container

References:


G. Succi, M. Marchesi. "Extreme Programming Examined". Addison-Wesley, 2001

Sullins, Benjamin C., “JMX in Action”. Manning Publications Co., 2003
