Omáda Programming

Team 15 - Java Card

Version 1.5

Brasoveanu Andrei-Alexandru

Delvin Varghese

Dominic Lee

3/22/2013

Version History

Date	Name	Reason for Changes
08/02/2013	Brasoveanu Andrei Alexandru	First version of document
20/02/2013	Brasoveanu Andrei Alexandru	Added security and implementation details
01/03/2013	Dominic Lee	Updated kannel implementation
15/03/2013	Delvin Varghese	Updated applet architecture
22/03/2013	Delvin Varghese	Final proofread and revision

Table of Contents

i. Manifesto

1. Introduction
1.1 Purpose
1.2 Document Conventions
1.3 Intended Audience and Reading Suggestions
1.4 Project Scope
1.5 References

2. Overall Description
2.1 Product Perspective
2.2 Product Features
2.3 Operating Environment
2.4 Design and Implementation Constraints

3. External Interface Requirements
3.1 User Interfaces
3.2 Hardware Interfaces
3.3 Communications Interfaces

4. Other Non-functional Requirements
4.1 Performance Requirements
4.2 Reliability Requirements
4.3 Security Requirements

Manifesto

Omáda Programming has been based on the agile development methodology. The manifesto for this has been shown below:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

This was implemented in various ways including:

- Programming on the client's premises to maximise efficiency. Co-location has been an important part of Omada programming's approach as the team has made multiple visits to the client and had much face-to-face contact.
- Pair programming: Delvin Varghese and Dom Lee have been working on the applet side of the project and have met up initially to program together.
- Each week implementation of new code has taken place to present to the client a tangible way of seeing work progress.
- The team has been very involved with the client in terms of the project and made sure the team's vision of the completed software is in line with the client's needs.
- Sustainable development has also been achieved in our view as the team has been developing at a constant pace.
- The team has also been self-organising and divided the task into 3 manageable chunks to be divided among the four members:
- The Applet required knowledge in Javacard (Java bytecode) and various SIM industry specifications provided as part of the development kit, including:

  3GPP TS 51.011
  3GPP TS 03.48
  3GPP TS 11.14
  3GPP TS 23.038
- The Server is being developed using JSP,nodeJS,EDS and RDS technologies.

- The Gateway/Kannel implementation has been done according to standards presented in the official documentation .
- However, it is to be noted that the team has set out with the aim of responding to change as we have agreed upon helping other members of the team should they require assistance on any difficulty.

1. Introduction

1.1 Purpose

Developing a 'fortune cookie' (cf. unix 'fortune') application which is to be deployed with JavaCard technology. The client's marketing research has shown that a niche for such an application exists in several countries where most of the population will find this useful for religious or superstition reasons. The client had done extensive market research as part of their previous projects and they shared this with us in the hope that we would find this useful. As part of this, they found that there is a big market in developing countries especially in the client's previous interest areas in Central Africa, where SIM based applications are still popular, for "Good Luck" messages and prosperity advice. This is also a very scalable model which can be deployed in other regions like South Asia or Latin America.

Also the targeted market is present in geographical areas were the most used mobile devices are not smartphones, as so the application requires development on a module available to all devices: the SIM card. In essence the architecture establishes a connection between the handset and a http server, this can be used for future development that would allow bringing social networks to the users without the reach of more modern machines.

1.2 Document Conventions

Abbreviations

WAP	Wireless Application Protocol
HTTP	Hypertext Transfer Protocol
SIM	Subscriber identity module
*nix	Unix based
SMS	Short Message Service
MSISDN	Mobile Subscriber Integrated Services Digital Network-Number
SMSC	short message service center
SMSGW	SMS Gateway
SMS-PP	SMS Point-to-Point
AWS	Amazon Web Services
EBS	Elastic Bean Stalk
RDS	Relational Database

1.3 Intended Audience and Reading Suggestions

The document is intended for readers involved in the project directly or indirectly. The technical value of the document is intended for someone with knowledge of development methodologies in programming and networking.

1.4 Project Scope

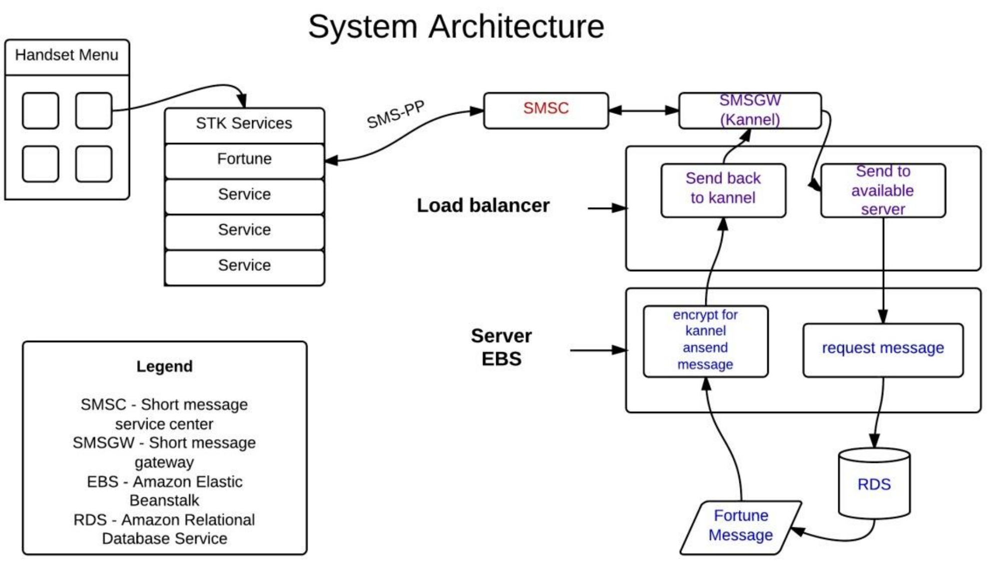
The project aims to give users that do not have access to modern handsets to new technologies. Ultimately the scope is to construct a basic architecture that can be used for future advances in the area.

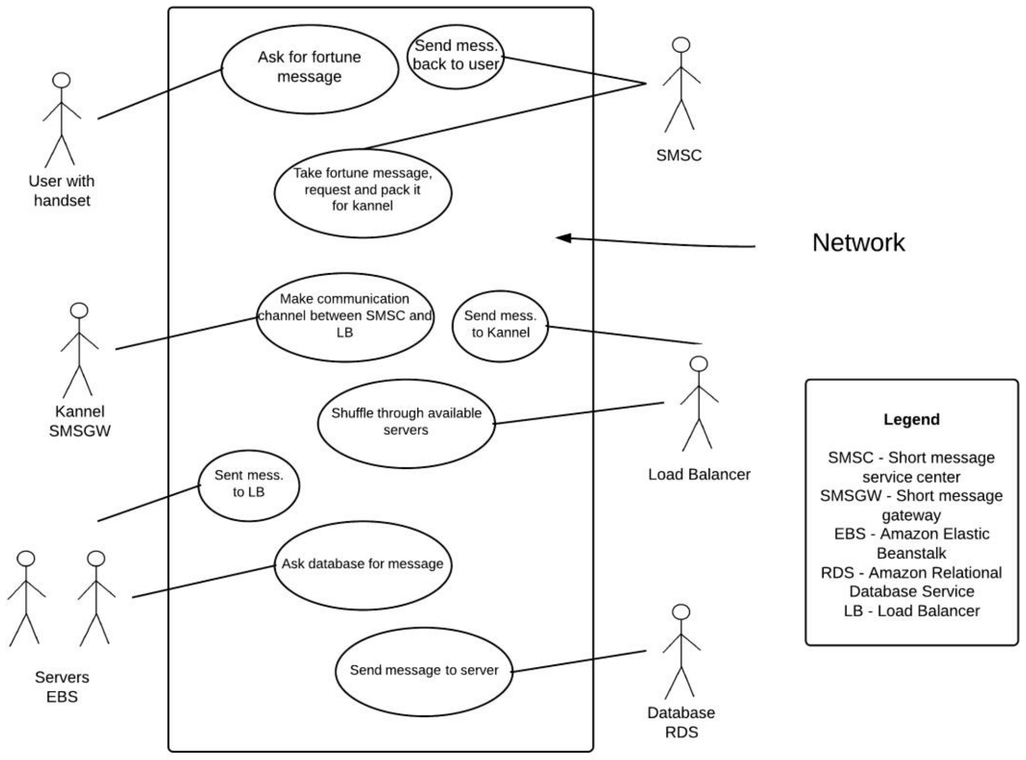
1.5 References

1.http://en.wikipedia.org/wiki/Fortune_(Unix)
2.http://www.kannel.org/
3.http://en.wikipedia.org/wiki/Javacard
4.http://en.wikipedia.org/wiki/Kannel_(telecommunications)
5.http://en.wikipedia.org/wiki/SMS
6.http://en.wikipedia.org/wiki/SMS_gateway
7.http://en.wikipedia.org/wiki/Hypertext*Transfer*Protocol

2. Overall Description

2.1 Product Perspective




Ask for fortune message
Send mess. back to user
SMSC
User with handset
Take fortune message, request and pack it for kannel
Network
Make communication channel between SMSC and LB
Send mess. to Kannel
Kannel SMSGW
Shuffle through available servers
Load Balancer
Sent mess. to LB
Ask database for message
Servers EBS
Send message to server
Database RDS
Legend
SMSC - Short message service center
SMSGW - Short message gateway
EBS - Amazon Elastic Beanstalk
RDS - Amazon Relational Database Service
LB - Load Balancer

2.2 Product Features

The product's main feature is to provide randomly generated fortune cookies: proverbs, advice, inspirational quotes etc. This is done using a menu on the phone handset. The advice is very useful for people who need motivation or just a uplifting message.

The phone can also provide updates/bonus messages from the server.

2.3 Operating Environment

The environment targeted is that of a SIM card. The specific technology that will be used for implementation is JavaCard.

2.4 Design and Implementation Constraints

Kannel implementation will be required.For this part of the project, development will have to be done on a *nix based operating system. Kannel requires the following software environment:

C compiler and development libraries and related tools. The gnome-xml (a.k.a. libxml) library, version 2.2.0 or newer. POSIX threads (pthread.h). GNU Bison 1.28 if you modify the WMLScript compiler. DocBook markup language tools (jade, jadetex, DocBook style-sheets, etc; see README.docbook).(see ref.2)

3. External Interface Requirements

3.1 User Interfaces

The applets user interface is test based and has only one option: requesting a fortune message.

3.2 Hardware Interfaces

Communication between the applet and the server that will return our message will have three stages:

Firstly the handset will send a SMS-PP to an SMSC server, from here the data is sent to Kannel. Kannel transforms the 'WAP received data' into valid HTTP packets.This packet is then sent to a load balancer which delivers it to the first available server in the cluster.

Kannel is a compact and very powerful open source WAP and SMS gateway, used widely across the globe both for serving trillions of short messages (SMS), WAP Push service indications and mobile internet connectivity.(see ref.2) The server and database are deployed on the Amazon Web Services(AWS) cloud, this was our choice since it provides a strong infrastructure. The servers are instances of AWS EBS (Elastic Beanstalk) and the database is a RDS. AWS Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring.

Amazon Relational Database Service (RDS) makes it easy to set up, operate, and scale a relational database in the cloud. You launch a Database (DB) Instance with automated backups and monitoring metrics. This implementation help with the balancing of the network as new instances of EBS are automatically opened when the previous ones reach a 90% load.

3.3 Communications Interfaces

The communication between the application is performed through SMS-PP technology on one side and HTTP on the other. The "middle man" between these are SMSC and SMSGW(Kannel) servers.

4. Other Non-functional Requirements

4.1 Performance Requirements

The projects target is to support several hundreds of concurrent user accessing the system.Scalability is very important so the system has been planned with this in mind.

4.2 Reliability Requirements!!

Unit testing will be done for the applet and servlet after and during development. The servlet will be deployed on a cluster of servers which has to be tested. Live and penetration testing will be implemented.

4.3 Security Requirements

The application will use a MSISDN which in translation is the phone number of the user, this information has to be handled under the Data Protection Act (1998). The server will have to be safe from malicious attack as SQL-Injection, DoS(Denial-of-service). Ciphering SMS from the handset to the server, and vice versa. Redundancy checks for SMS's to make sure they've not been tampered with will be implemented. The load balancer, server, ESME/SMSC will all be the same VPN network so that requests can only be accepted on the VPN bind address.

Appendix A: Glossary

Java Card

Java Card refers to a technology that allows Java-based applications (applets) to be run securely on smart cards and similar small memory footprint devices. Java Card is the tiniest of Java platforms targeted for embedded devices.http://en.wikipedia.org/wiki/Javacard

Kannel

In computing, Kannel is an open source WAP gateway.It provides the essential part of the WAP infrastructure as open source software to everyone so that the market potential for WAP services, both from wireless operators and specialized service providers, will be realized as efficiently as possible.http://en.wikipedia.org/wiki/Kannel_(telecommunications)

SMS

Short Message Service (SMS) is a text messaging service component of phone, web, or mobile communication systems, using standardized communications protocols that allow the exchange of short text messages between fixed line or mobile phone devices.http://en.wikipedia.org/wiki/SMS

SMSC

A short message service center (SMSC) is a network element in the mobile telephone network. Its purpose is to store, forward, convert and deliver SMS messages. The full designation of an SMSC according to 3GPP is Short Message Service - Service Centre (SMS-SC)http://en.wikipedia.org/wiki/Short*message*service_center

SMSGW

SMSGW service allows members to send text messages to mobile phones, using any standard e-mail program or web browser.http://en.wikipedia.org/wiki/SMS_gateway

SMS-PP

SMS point-to-point is the protocol described in the 3GPP TS 23.040 standard fro WAP to HTTP communication.

HTTP

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems.HTTP is the foundation of data communication for the World Wide Web. Hypertext is a multi-linear set of objects, building a network by using logical links (the so-called hyperlinks) between the nodes (e.g. text or words). HTTP is the protocol to exchange or transfer hypertext.http://en.wikipedia.org/wiki/Hypertext*Transfer*Protocol