

Geofencing in eHealth

Mobile Services Project

by
Fatimah Zahra (7515)

Faculty of Computer Science
Free University of Bozen-Bolzano
Italy, 2009

Contents

Contents	2
Figures	2
1. Introduction.....	3
2. System Functions.....	3
3. User Interface	5
4. Architectural Design	11
4.1 Patient’s Implementation.....	11
4.2 Caregiver’s Implementation	13
5. Limitations and Future Works	16

Figures

Figure 1 Geofencing in eHealth	3
Figure 2 Use cases	5
Figure 3 Main screen of patients' application	6
Figure 4 Boundary list screen	7
Figure 5 Add, edit and delete boundary screen	7
Figure 6 Main screen of caregivers' application.....	8
Figure 7 Patients list screen.....	9
Figure 8 Add, edit and delete patient screen	10
Figure 9 View location screen	10
Figure 10 Deployment diagram.....	11
Figure 11 Patients' class diagram	13
Figure 12 Caregivers' class diagram.....	15

1. Introduction

Currently, many elder people with dementia live independently. These people do not feel entirely confident going out on their own. They might just take a walk and forget where they are. This situation can cause difficulties for them and their caregivers. With the development of GPS technology in mobile phones and geofencing, these people might have more freedom and safer to go out.

Geofence is a virtual boundary on a geographic area. The boundary is defined by set of coordinates. So the term of geofencing here refers to a use of location-based service for the process of setting up a virtual perimeter that someone or something might cross. When the boundary is crossed, a user can be notified.



Figure 1 Geofencing in eHealth¹

Thus, to help people with dementia stay independent, the author decided to build a mobile application that uses GPS and the principle of geofencing to keep track these people.

2. System Functions

This application has two types of user, which are:

- Patients
- Caregivers (doctors, emergency services, nursing personnel, health care providers)

The following table is the detail description of functionalities for each type of users:

Patients	Caregivers
<ul style="list-style-type: none"> • Setting for the boundary. • Send alerts to the caregivers if patients are out of the boundary and vibrate the patient's mobile 	<ul style="list-style-type: none"> • Manage a list of patients. • Receive alerts when the patients are out of the boundary.

¹ <http://blogs.forum.nokia.com/blog/arto-holopainens-forum-nokia-blog/2009/03/21/geofencing-in-ehealth>

device. <ul style="list-style-type: none"> • Update their location periodically to the caregivers. 	<ul style="list-style-type: none"> • Receive the last updated position of patients. • View the location of patients
----------------------------------------------------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------------

To have a better/clearer understanding of what this application can do and how, the author has performed several analyses which are screen and interaction, usage, and environment analysis.

1. Screen and interaction analysis

- The emulator for the caregivers is the DefaultColorPhone, while for the patients is the DefaultGrayPhone.
- The mobile phones for patients need to have built-in GPS and support WMA (Wireless Messaging API). Mobile phones for caregivers should also support WMA.
- For the caregivers, they need mobile phones with big screen size and high resolution.
- For patients side:
 - Patients need to give their consent for being tracked. After they have agreed, the initial setting for the boundary will be done. After that, the patients only need to turn on the application. No other inputs required.
 - The boundary will be inputted at the initial setting of the patients' mobile devices. The input will be the pairs of points. From this information, the application will create rectangle boundaries.
 - Patients' application will send alert to the caregivers if they are out of the boundary.
 - Patients' application will update their location periodically to the caregivers.
- For caregivers side:
 - Only patient's name, phone number, and status are being displayed in the patients' list.
 - The status of the patients will be displayed with color sign. For this application, green circle for still inside the boundary, and red circle for outside the boundary.
 - Caregivers can view the location of the patients from the last update.

2. Usage analysis

For this usage analysis, use cases have been created to define functionalities for each user's mobile phones. The following is the diagram:

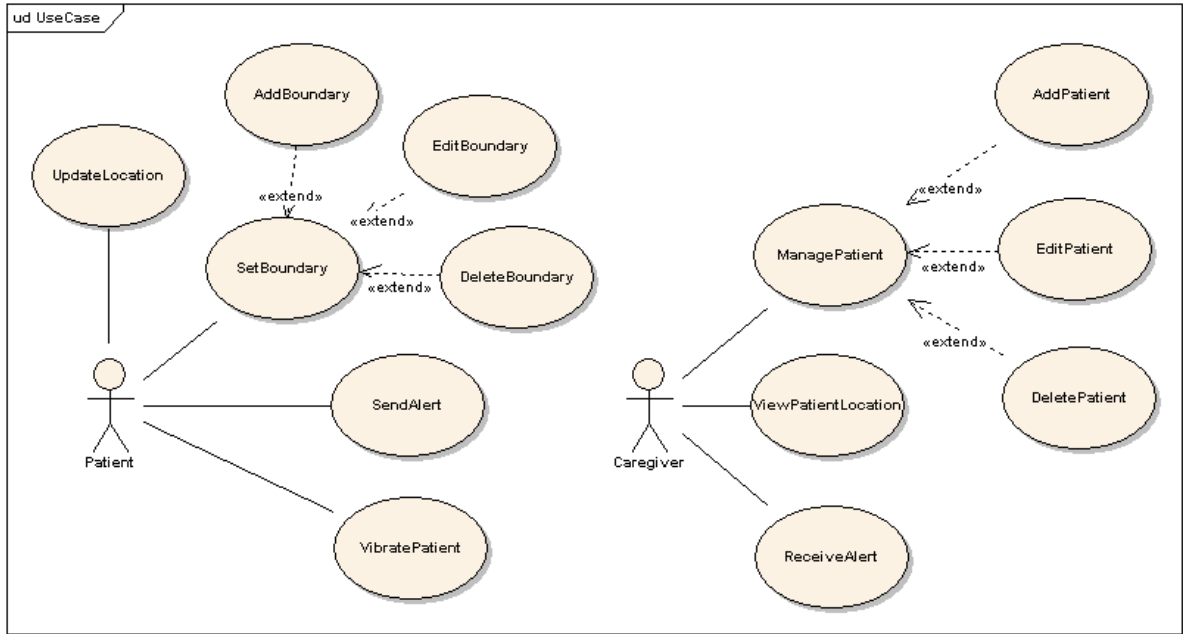


Figure 2 Use cases

3. Environment analysis

The other device/network that this application interacted is only with the GPS satellite.

3. User Interface

The following are the detail flows of interaction between users and the application.

1. Patients' application

- Launch the application

After user clicks 'Launch', main screen of the application will show. If no boundaries set yet, the status will show 'NOT ACTIVE'. This means that the function for checking the user's position against the boundaries is not being performed. After adding new boundary, this function will start checking.



Figure 3 Main screen of patients' application

- Set the boundary
 - After the user click 'Set Boundary', the application will show the boundary list, with pair of points in it. From the 'Menu', users can choose to 'Add Boundary', 'Edit Boundary' or 'Delete Boundary'.



Figure 4 Boundary list screen

- If user chooses 'Add Boundary', the application will show an input form to fill in with coordinates (latitude, longitude). While for 'Edit Boundary', the form will be displayed with data points selected. Also, if user wants to delete the boundary, a confirmation message will be displayed before the real deletion started.



Figure 5 Add, edit and delete boundary screen

2. Caregivers' application

- Launch the application

After user clicks 'Launch', main screen of the application will show. User will need to click 'Enter' to access the functionalities.

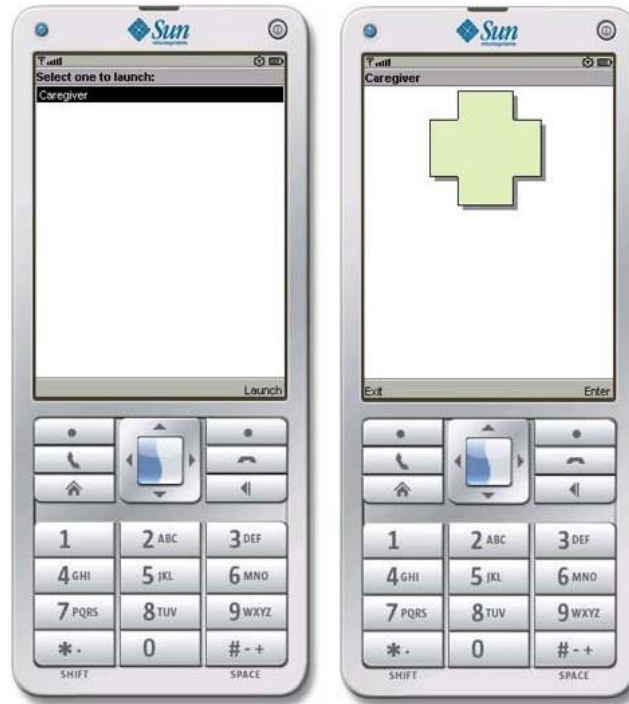


Figure 6 Main screen of caregivers' application

- Manage a list of patients
 - After the user clicks 'Enter', a list of patients is displayed. The patients' data that are being shown consist of name, phone number and status of the patients. The status here means whether a patient is inside the boundary or not. This situation is represented by the circle where green circle means inside the boundary and red circle means outside the boundary. From the 'Menu', user can choose to 'Add Patient', 'Edit Patient', 'Delete Patient' or 'View Location'.



Figure 7 Patients list screen

- If user chooses 'Add Patient', the application will show an input form to fill in with name and phone number of the patient. While for 'Edit Patient', the form will be displayed with data patient selected. Also, if user wants to delete the patient, a confirmation message will be displayed before the real deletion started.



Figure 8 Add, edit and delete patient screen

- View location

If user chooses 'View Location' on a specific patient, the application will show the position of the patient on the map. The 'Details' is for displaying the last modified time of patient's location.



Figure 9 View location screen

4. Architectural Design

For the architectural design, deployment and class diagrams will be used.

From figure 10, it shows that the deployment of this application is very simple and easy. It only needs two execution environments which are the users' mobile phones. From the patients' phone, it will send messages to the caregivers' phone. Also, only patients' phone connects to the GPS satellite.

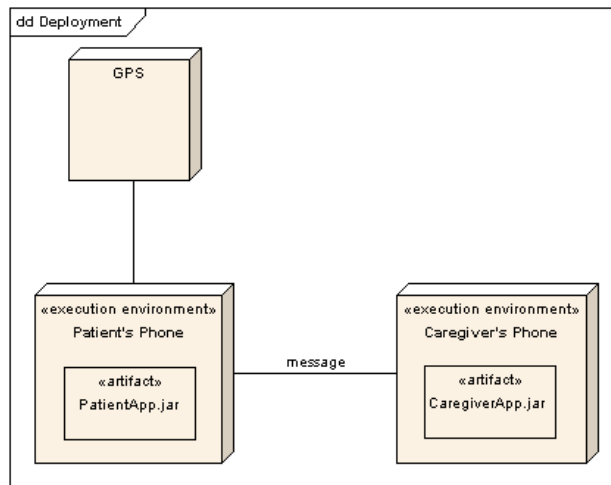


Figure 10 Deployment diagram

For the application's implementation, it will be described using class diagrams. Following are the explanations of the implementation for each user's types.

4.1 Patient's Implementation

The structure of the code for patients' implementation is described as follows:

Class Name	Role	Main Methods	
		Name	Role
PatientMidlet extends MIDlet implements CommandListener	Main midlet	createLocationProvider::void	Create LocationProvider instance for PatientGPSTracker and PatientGPSUpdater class
		setTickerStatus::void	Set ticker string to 'ACTIVE' or 'NOT ACTIVE'
		startApp::void	Display the main form and starts 2 threads for PatientGPSTracker and PatientGPSUpdater
BoundaryList extends List implements CommandListener	Display a list of boundaries	getSelectedBoundary::Boundary	Get the selected boundary
		populateList::void	Add boundaries to the list
BoundaryForm extends Form implements	Create form for add and edit boundary	addBoundary::void	Add the new boundary, connect to BoundaryDAO
		isInputValid::boolean	Checking the input, return false if there is any empty field

Class Name	Role	Main Methods	
		Name	Role
CommandListener		setBoundary::void	Set the boundary that is going to be updated
		updateBoundary::void	Update the boundary selected, connect to BoundaryDAO
BoundaryDAO	Connect to the boundary database and performed CRUD on it	addBoundary::void	Add new boundary
		deleteBoundary::void	Delete a boundary
		getAllBoundary::Vector	Retrieve all boundaries
		getBoundaryId::int	Get the boundary id (record id)
		updateBoundary::void	Update a boundary
Boundary	POJO for represents a boundary	createBoundary::Boundary	Create a boundary from a String with format "lat1 long1 lat2 long2"
		getDBString::String	Return boundary in the database string representation "lat1 long1 lat2 long2"
		getString::String	Return boundary in string with format "(lat1,long1),(lat2,long2)"
		isEqual::boolean	Check whether this boundary equals with the boundary provided in the parameter
		getX::typeXX setXX::void	Getter and setter methods for all the private fields
PatientGPSTracker implements Runnable, LocationListener	Thread for tracking patient periodically	checkBoundary::boolean	Check the current position of the patient against the boundaries set
		sendAlert::void	Send messages to caregiver if the patient out of the boundary and return inside the boundary.
PatientGPSUpdater implements Runnable, LocationListener	Thread for updating patient's location to caregivers periodically	locationUpdated::void	Send messages to caregivers about the current position of the patient periodically
SmsSender implements Runnable	Thread to send messages to caregivers	sendMessage::void	Start a new thread to send a message to caregiver
PatientUtil	General utility class	loadImage::Image	Read an image file and return the image

The relationship between classes is described in the following class diagram:

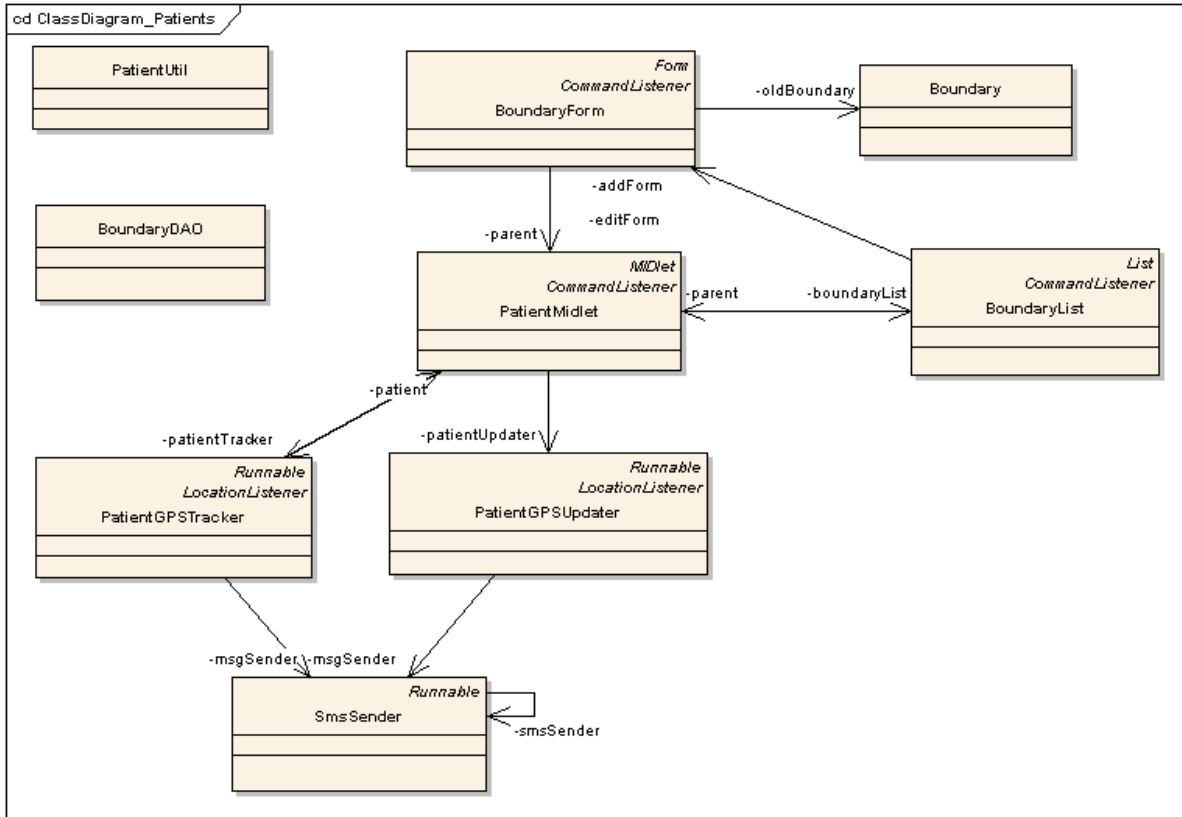


Figure 11 Patients' class diagram

4.2 Caregiver's Implementation

The structure of the code for caregivers' implementation is described as follows:

Class Name	Role	Main Methods	
		Name	Role
CaregiverMidlet extends MIDlet implements CommandListener, Runnable	Main midlet and receive messages from patients	processMessage::void	Filtering incoming messages. If it is an alert message, it will display a warning alert. For update messages, it will update the patient's location in the database. While for alert messages, it will update both patient's location and status in the database.
PatientsList extends List implements CommandListener	Display a list of patients	getSelectedPatient::Poundary	Get the selected patient
		populateList::void	Add patients to the list
PatientForm extends Form implements CommandListener	Create form for add and edit patient	addPatient::void	Add the new patient, connect to PatientDAO
		isInputValid::boolean	Checking the input, return false if there is any empty field
		setPatient::void	Set the patient that is going to be updated
		updatePatient::void	Update the patient selected, connect to PatientDAO
PatientDAO	Connect to	addPatient::void	Add new patient

Class Name	Role	Main Methods	
		Name	Role
implements RecordComparator	the patient database and performed CRUD on it	attachPatientData::Patient	Attach all data of a patient into the Patient instance, based on the phone number
		compare:int	For sorting the patients based on its name in ascending order
		deletePatient::void	Delete a patient
		getAllPatients::Vector	Retrieve all patients
		updateLocation::void	Update the location of a patient
		updatePatient::void	Update the patient's data
		updateStatus::void	Update the patient's status
Patient	POJO for represents a patient	createPatient::Patient	Create a patient from a String with format "phoneNo name status latitude longitude lastModified"
		getXX::typeXX setXX::void	Getter and setter methods for all the private fields
PatientMap extends Canvas implements CommandListener	Display the patient's location using the map	convertCoordinatesToXY::int[]	Convert latitude and longitude coordinates into XY coordinates
		getPatient::Patient	Get the patient for this map
		paint::void	Paint the map and position of the patient
		setPatient::void	Set the patient for this map
DetailsForm extends Form implements CommandListener	Show information of patient that is on the map	DetailsForm → constructor	Display a form with last modified information of patient's location
CaregiverUtil	General utility class	createScaledImage::Image	Scaled an image to fit in the screen size
		loadImage::Image	Read an image file and return the image

The relationship between classes is described in the following class diagram:

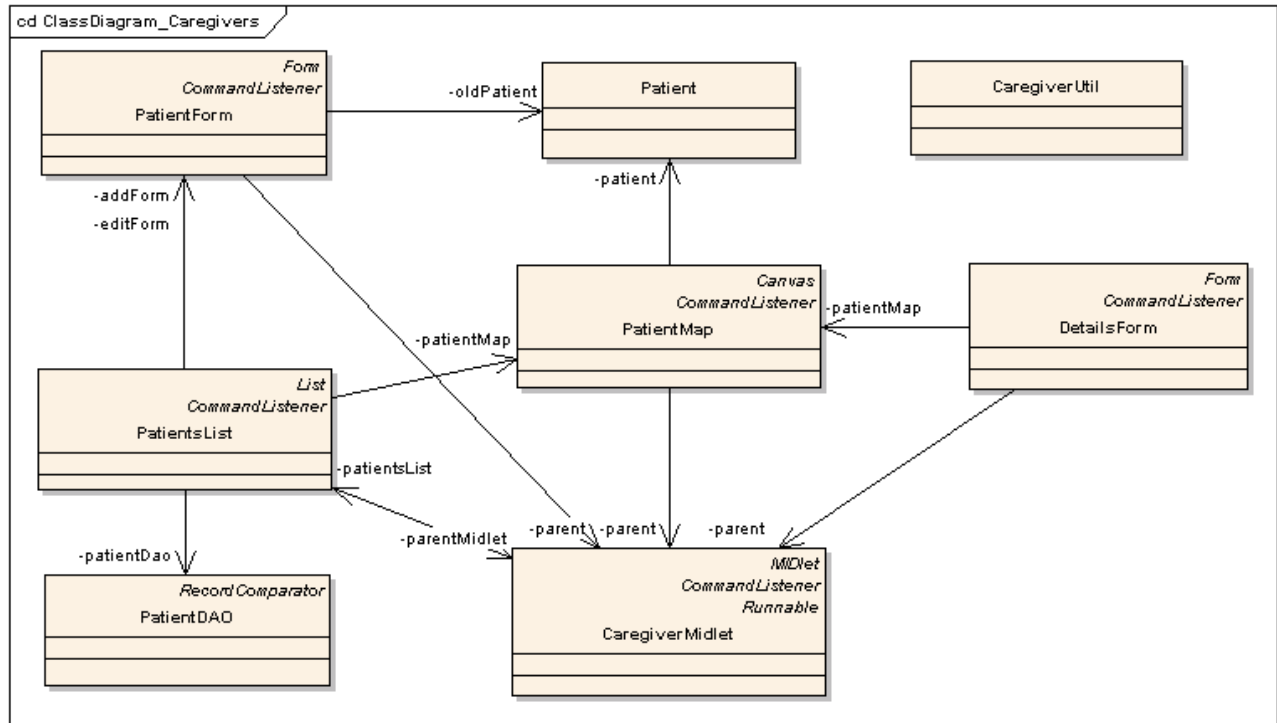


Figure 12 Caregivers' class diagram

In this implementation, there are several code reuses. Below is the detail explanation for each reuse:

- Method name: createScaledImage
It is located at CaregiverUtil class and used for fitting an image to the screen size. This code was taken from Forum Nokia Wiki².
- Method name: secondAlg
It is located at PatientMap class and used for converting latitude and longitude coordinates into XY coordinates. This code was taken from CityGuide project's example of Sun WTK.

Problem encountered while implementation was the mapping of latitude and longitude coordinates into XY coordinates image map. This issue was solved by reusing the code from the CityGuide project (example of Sun WTK) and adjusted the code by changing a little the calculation for the Y-point.

Other problem encountered was the database access. If two applications (patient and caregiver) ran together, one of them could not read the database, and it depends on who ran first. This problem was solved by using different device emulator, so the location of the database will also differ. For caregiver, it used DefaultColorPhone, while for patient, it used DefaultGrayPhone.

² How to Fit an Image to the Screen Size - http://wiki.forum.nokia.com/index.php/How_to_Fit_an_Image_to_the_Screen_Size

5. Limitations and Future Works

This application has several limitations. First, the boundaries are transformed into rectangles. Second, the map used is already fixed, which is only for around Duomo of Bolzano. Third, the caregivers can't track the current location of patients. Fourth, the detail information displayed for patients' location is only the last modified time. Thus, for the future works, the application should be able to transform the boundaries into flexible forms and track the patients' current position. In addition, it will be better if the map displayed is using the GoogleMap API, so that for the location of each patient, it will display a different map.