Car Gossip
Design Description

Version 1.1

Revision History

Date	Version	Description	Author
2012-11-09	0.1	Initial Draft	David Reypka
2012-11-15	1.0	First Version	David Reypka
2012-11-25	1.1	First revision	Nikola Vranešić
2013-01-17	Final	Final version	David Reypka

Table of Contents

1 Introduction

Purpose of this document

This document is a guideline for the implementation of the Car Gossip project. It is defined after the first drafts of the Project Plan and the Requirements Definition documents.

Revisions to the Design Description will occur after the Alpha and the Beta prototypes to harmonize with changes occurred to requirements and implementation details.

Intended Audience

The Design Description document is primarily used by the project members. It will be used (and modified if needed) throughout the whole development process.

USER	**USE**
Project Manager	To keep track of whether the development process follows the agreed development plan
Other project members	A guideline on how to implement components and define how they should interact
Supervisor and other stakeholders	Overview of the product to be developed

Scope

This document will cover the realization of our Project Plan driven by the Requirements Definition. Details regarding design and planned implementation will be described with the help of explanatory diagrams.

Car Gossip project's main objective is to provide a real-time traffic simulator and the possibility to exchange messages between cars (drivers).

The development of the Car Gossip project will produce a system composed of five main components:

1. Open Data Source: supplies project-oriented modified traffic data;
    a. Parser: parses one open data into a different, more usable format;
    b. MongoDB: stores the parsed data;
2. Desktop Application: uses the open data to simulate and visualize traffic;

a Traffic Simulator: instances ICDs, simulates their communication and visualizes them;
b Internal Car Device: sends, receives and processes messages;

3 Android Application: sends, receives and processes messages and visualizes the results;
4 Web Server: sends, receives, processes and stores messages;
5 Web Application: sends, receives and processes messages and visualizes the results;

Definitions and acronyms

Acronyms and abbreviations

Acronym or abbreviation	Definitions
ETHZ	Federal Institute of Technology Zurich
NS-2	Network simulation tool in the version 2
CSV	Comma-separated values
DB	Database
ODS	Open Data Source
ICS	Internal car sensors
ICD	Internal car device
DSRC	Dedicated short range communication
GPS	Global Positioning System
API	Application programming interface
GUI	Graphical User Interface
App	Application

Definitions

Keyword	Definitions
Swiss Database	ETHZ Open Database is a database used that provides 24 hours of traffic information in Zurich.
Parser	It parses the traffic data from the ETH database to our public database into input data for the Simulator.
Traffic Simulator	Simulates the input data into a traffic simulation and

	simulates the vehicle communication.
Gossip	A message containing information about the cars position and movement.
Alert	A default message sent by drivers to inform others of the traffic status.
MongoDB	A scalable, high-performance, open source NoSQL database.
Morphia	A lightweight type-safe library for mapping Java objects to/from MongoDB.
MapPanel	An EPL (Eclipse Public License) licensed Swing-Component meant for reuse. Java based browsing of openstreetmap tiles (or the like).
BlueCove	A Java library for Bluetooth (JSR-82 implementation).
BluetoothAdapter	An Android Java library class that represents the local device Bluetooth adapter.
Google Maps API	Allows developers to embed Google Maps into both Desktop and Mobile Applications
JBoss Tools	An umbrella project for a set of Eclipse plugins that supports JBoss and related technology.

References

Swiss Database, http://www.lst.inf.ethz.ch/research/ad-hoc/car-traces/
NS-2 movement format files, Kai Nagel, Bryan Raney and Hinnerk Spindler, ETH Zurich
DSRC, http://en.wikipedia.org/wiki/Dedicated_short-range_communications
GPS, http://en.wikipedia.org/wiki/Global_Positioning_System
Java SE 7 (July 28, 2011), Oracle Corporation
Eclipse SDK, Version: 4.2.1
Morphia, Powered by Google Project Hosting
MongoDB, 10gen Inc., Licensed under Creative Commons
MapPanel, Everyone at openstreetmap.org, Ricky Clarkson, Werner de Bruijn
Bluecove, Powered by Apache Maven
BlutoothAdapter, Powered by Google
Google Maps API, Powered by Google
JBoss Tools, Sponsored by RedHat Inc.

2 Software architecture

Conceptual design

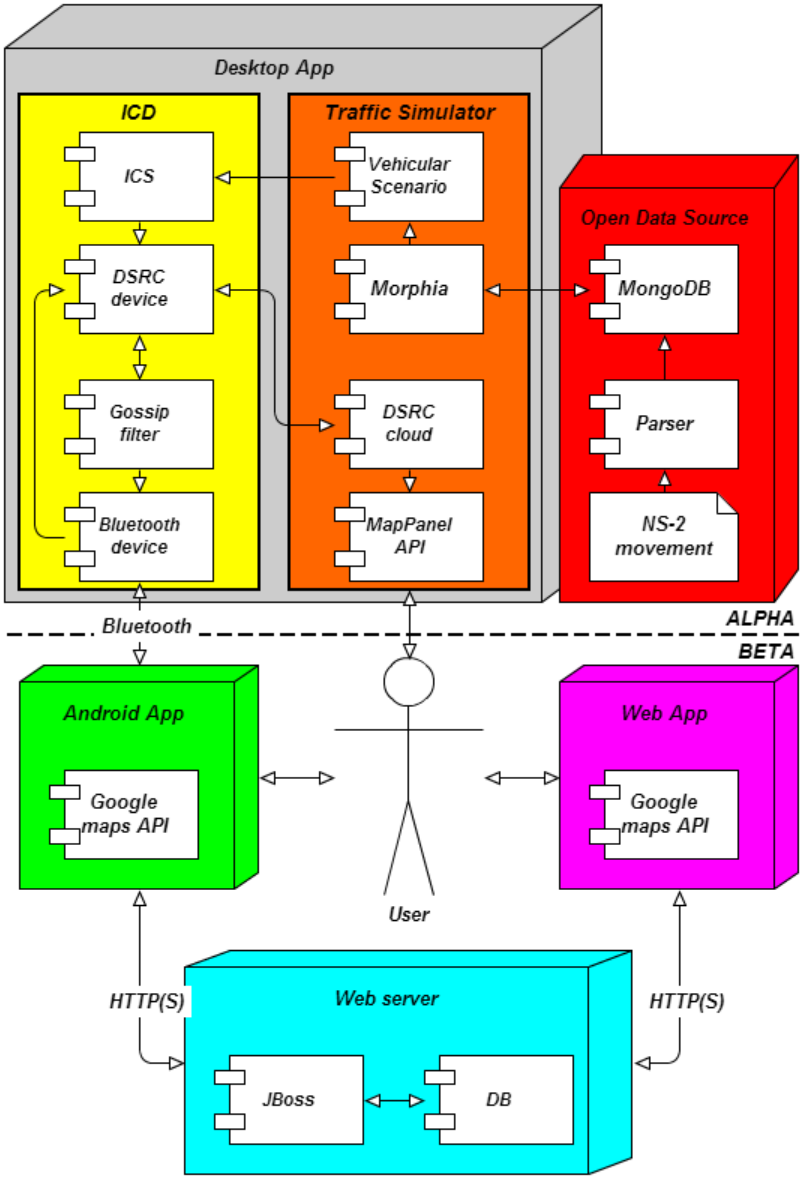

Figure 1: *Deployment diagram*

Open Data Source

Open data is freely available for everyone to use and republish as they wish, without restrictions from copyright, patents or other mechanisms of control. The Car Gossip project uses the ETH DB to create a new Open Data Source with modified data.

Parser component

Using a Parser, car traces in NS-2 movement formatted files from the ETH DB are parsed into a CSV format file which is more suitable for further processing.

MongoDB

The CSV format file is stored within MongoDB which represents the new Open Data Source.

Desktop App

The desktop application logic is divided into two main components: **Traffic simulator** and **ICDs**.

Traffic Simulator

The Traffic Simulator's basic functionality is to simulate cars that move and communicate with each other. The user can customize the simulator through the following parameters:

- time frame: simulation duration
- starting point: GPS location center of the simulation
- radius: distance from starting point within the cars are simulated
- amount of cars: number of cars being simulated

Once the user has set the parameters, the simulator acts as follows:

- it retrieves the parsed traffic data from the ODS based on the user parameters
- creates the cars in the form of ICDs that will then be initialized and scheduled into threads

Since there are no real ICDs to communicate via a wireless DSRC, the Traffic Simulator simulates a DSRC Cloud that collects all broadcasted ICD messages and distributes them to ICDs within the senders range defined by the user.

The visualization of the traffic by means of moving cars and the DSRC by means of gossip spreading are included within the component.

ICD

It is used to simulate real life cars and their internal devices. Its function is to forward

surrounding traffic information as Gossip messages and therefore be able to:

- send and receive messages from other cars' ICD via DSRC Cloud
- send and receive messages from an Android device via Bluetooth

Android App

It is used by the driver. Its functions are:

- visualizing a map with traffic information collected from the ICD or Web Server
- visualizing the Alerts on the map collected from the ICD or Web Server
- enabling the user to send Alerts to other drivers via ICD+DSRC and/or Web Server

Web Server

It is used as a data storage and service backbone for the Android and Web application. Its functions are:

- storing traffic data uploaded from Android devices via the Car Gossip App
- supplying Web services to support the Android and Web App functionality (exp. spam control)

Web App

The Web App is used by users to analyze the traffic data stored in the Web Server. Its functions are:

- visualizing a map with traffic information collected from the Web Server
- visualizing the alert messages on the map collected from the Web Server
- representing analysis results requested by the user

System specification

The used technologies and their platforms can be divided in subgroups.

Open Data Source

- Java, Eclipse: parses ETH traffic data
- MongoDB: stores the parsed data

Desktop App

- Morphia: retrieves the parsed data from MongoDB

- Java, Eclipse: Traffic Simulator and ICD logic
- MapPanel API, Eclipse: for Desktop App GUI
- UML plugins, Eclipse: documentation purposes
- Bluecove, Eclipse: ICD Bluetooth connection

Android App

- Java, Eclipse: Android App logic
- Android SDK, Eclipse: Android App logic and GUI
- Google Maps API, Adroid SDK: Android App GUI
- BluetoothAdapter, Eclipse: Android App Bluetooth connection

Web Server

- Java, JBoss Tools, Eclipse: Web Server logic
- RDBMS: stores data received via an Android App

Web App

- PHP|Phyton|Ruby: Web App logic
- HTML5, CSS3, JS, Google Maps API: Web App GUI

External Components

All eternal components have already been described:

- ETH DB
- UML plugins for Eclipse
- Bluecove
- MapPanel API
- BluetoothAdapter
- Google Maps API

3 External interfaces

An interface is a point of interaction between components, and is applicable at the level of both hardware and software. This allows a component, whether a piece of hardware or a piece of software to function independently while using interfaces to communicate with other components via an input/output system and an associated protocol.

Hardware Interfaces

GPS access

A GPS device is an essential part of the ICS. Since the ICS are simulated, we don't actually have hardware interfaces.

Bluetooth connection

The ICDs and Android devices are meant to communicate via Bluetooth with each other. Each user should be able to connect to the ICD in their car, so a BlueCove API is mandatory for each ICD. Since the ICD in this project is simulated it will use the Bluetooth device of the PC that is running the simulation to connect to an Android device.

Software Interfaces

"A software interface may refer to a range of different types of interface at different "levels". A key principle of design is to prohibit access to all resources by default, allowing access only through well-defined entry points, i.e. interfaces. The idea is to base programming logic on the interfaces of the objects used rather than on internal implementation details. Programming to the interface reduces dependency on implementation specifics and makes code more reusable."

http://en.wikipedia.org/wiki/Interface_(computing)

Used software interfaces are visualized and described in *4.1 Implementation modules / components*.

Communication Interfaces

Web Server connection

Both the Android and the Web App will use an Internet connection, if possible, to communicate with the Web Server in order to store and/or retrieve traffic data.

User Interfaces

Desktop App GUI

The Desktop App's GUI is very simple and it consists of a map and a vertical list positioned on the right side of the window. By pressing a certain location on the map, a window appears to enable the user to enter the Vehicular Scenario parameters for customizing the traffic simulation. Once the parameters are defined, the vertical list on the left side will be filled with car IDs which represent cars on the map.

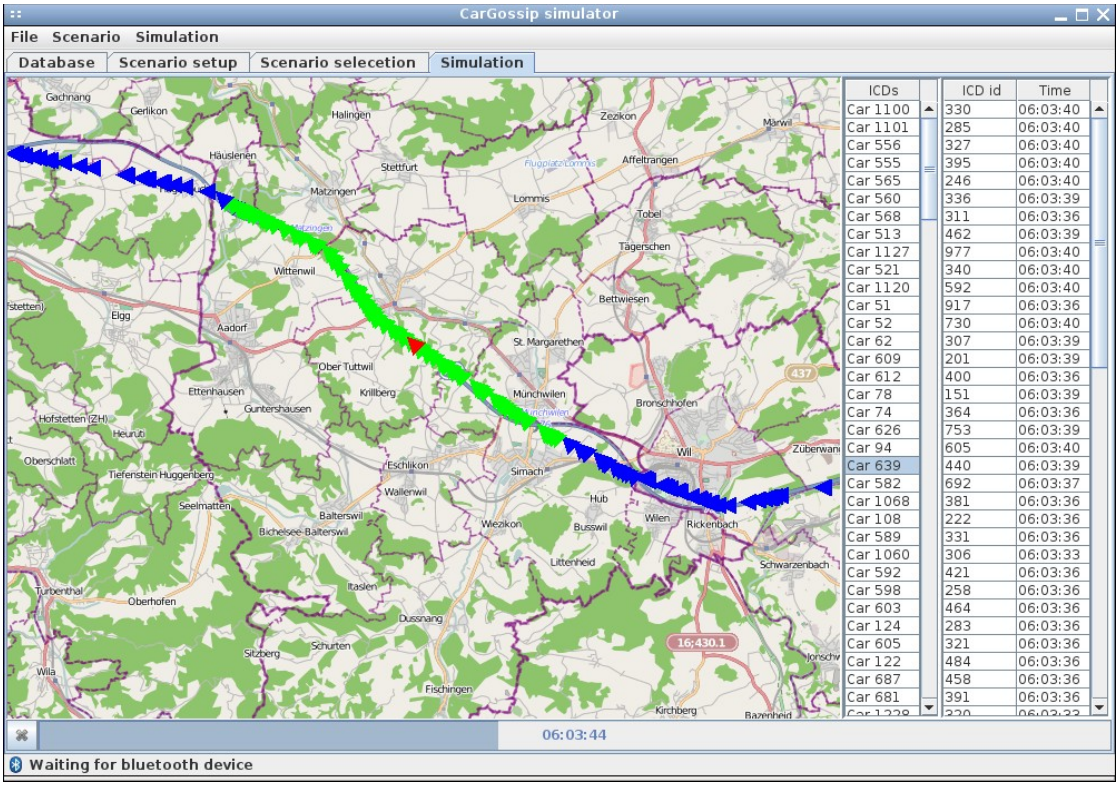

Figure 2: *Desktop App GUI*

By pressing one of the car IDs listed on the right (Figure 2), the simulated car corresponding to the selected car ID is selected on the map. By selecting a car, its ICD is able to connect to an Android device via Bluetooth and his Gossip spreading is viewed on the map. The Gossip spreading will be visualized by coloring cars receiving the Gossips broadcasted from the selected car. The color will be predefined, but the car's coloring transparency will get higher as the timestamp difference increases between the last Gossip received from the selected car and the latest Gossip being broadcasted from the selected car.

Android App GUI

The user is presented with a welcome screen and can bring up a an overlay with options to connect to an ICD.
(Figure 3).

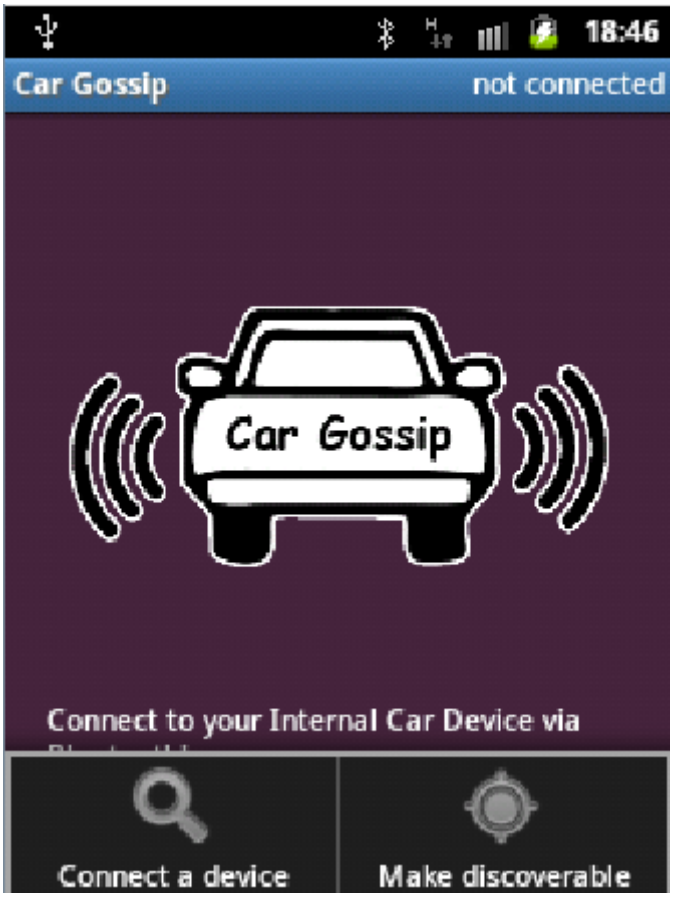

Figure 3: *Android App GUI (Menu button)*

After a selected car connects to the Android App they communicate by exchanging gossips and alerts. Based on that information the Android App creates the surrounding traffic visualization on a map
The map shows the surrounding cars in real-time and the alert messages as alert bubbles. The GUI contains a Menu button. It opens a window with default alert messages as iconic buttons. Pressing one of them, the corresponding Alert would be sent to the Web server for validation and if confirmed it would be sent to the ICD for broadcasting.

Figure 4: *Android App GUI (Alert buttons)*

Web App GUI

The Web App GUI is mostly based on map visualization with the traffic data and alert bubbles.

4 Detailed software design

Implementation modules / components

Desktop app

Desktop app contains four main parts, database handling panel, scenario setup panel, scenario selection panel and simulation panel. Database handling panel handles the database connection and can parse the input files and store them to the database. Scenario setup panel allows the user to create custom scenarios. Scenario selection panel is used to select previously made scenarios and to create a simulation from the selected one. Simulation panel offers simulation visualization and simulation controls.

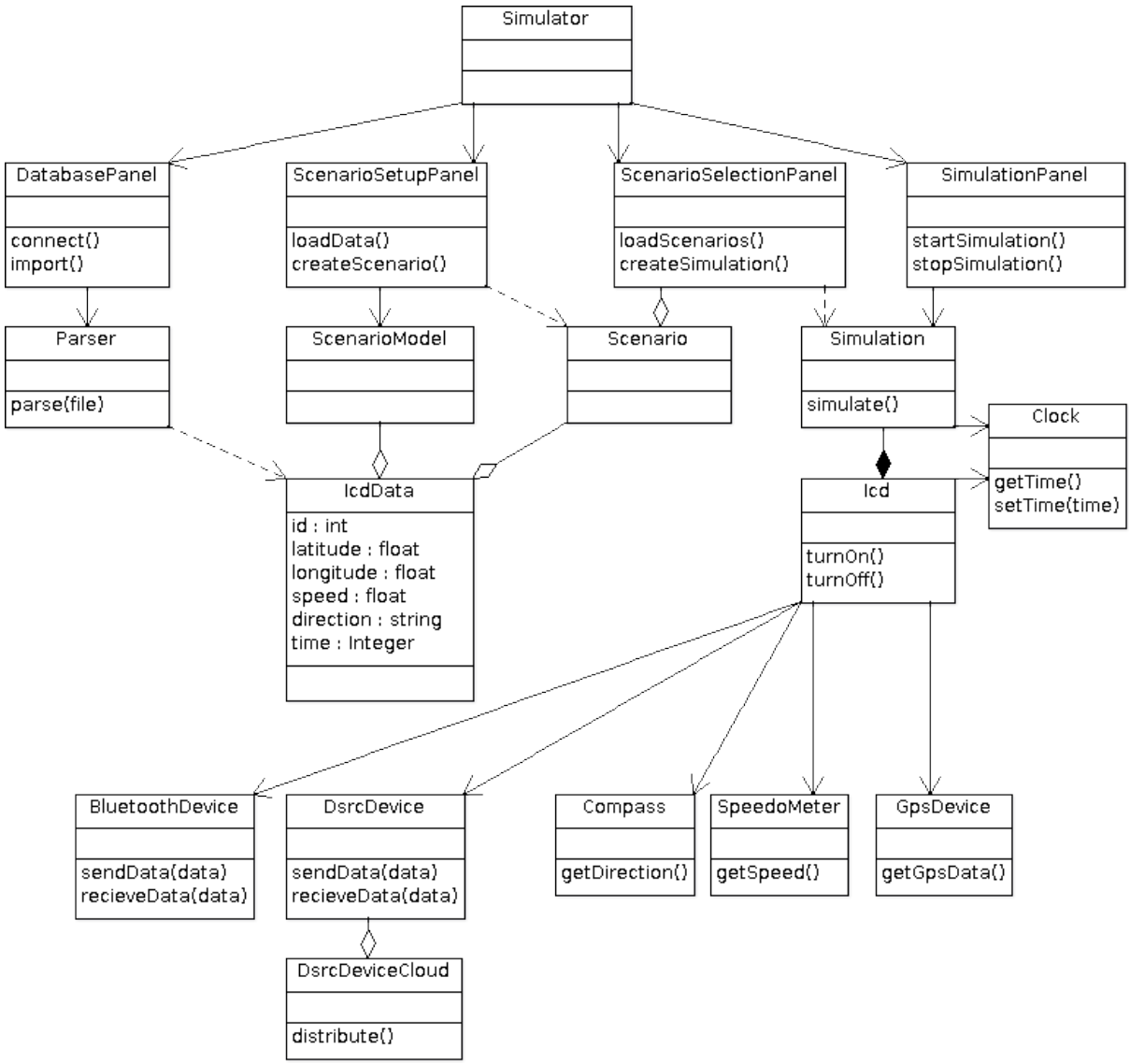

Figure 5: *Desktop app class diagram*

Android App

The Android application can be divided into three logical parts.
Two parts are used as interfaces for the other project components.
The Bluetooth activity provides the user with a welcome screen and, by pushing the Android phones menu button, the option to connect to the Desktop application where the simulation is run on.
The BluetoothCommandService implements thereby the underlying mechanic to connect and send and receive from the other side.
If the connection succeeds, the user is presented with a view of a map adapted from the Google Maps framework.
This view is enhanced by Overlays to display events in the surrounding area and also provide the option to select and send alert messages to the web application through the HTTPRequest class and also via Bluetooth to the responding ICD in the simulator.
Bluetooth messages received by the BluetoothCommandService are directed to the MapWindow, where they can be visualized.

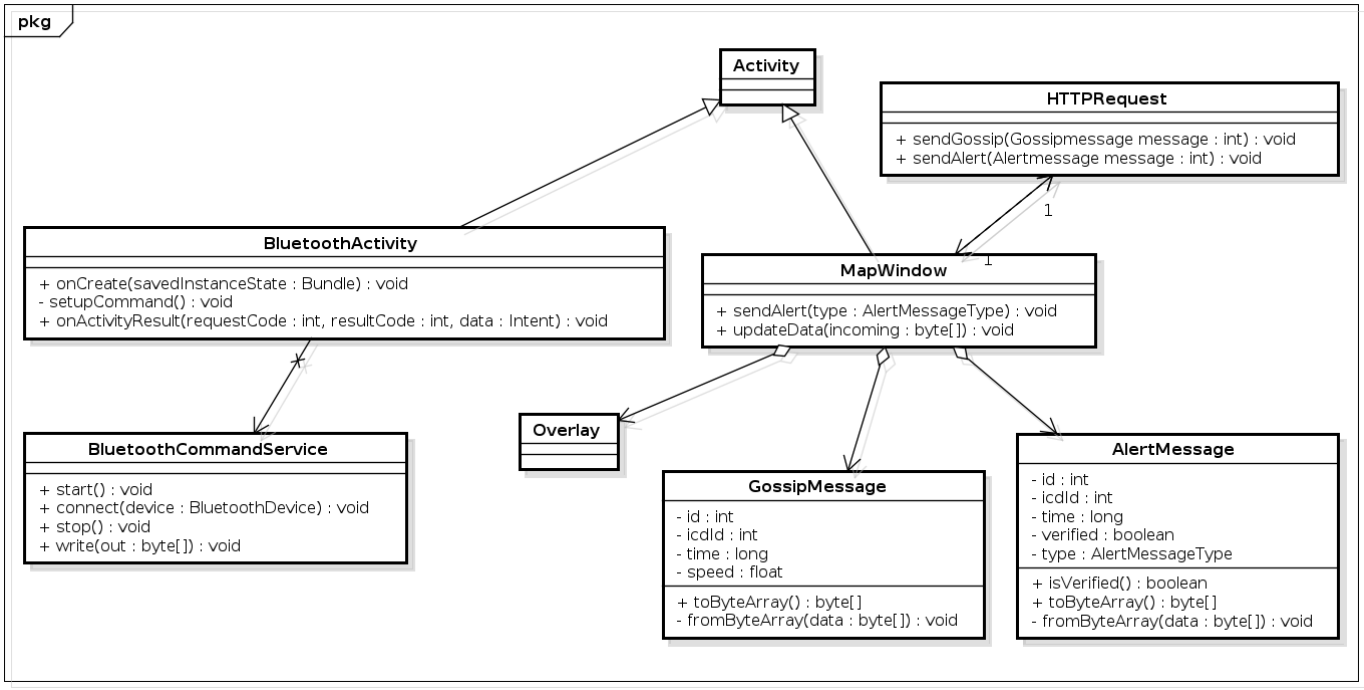

Figure 6: *Android App class diagram*

Data flow / Interactions / Dependencies

Desktop application

Parser reads input files and stores the data in the database. ScenarioCreationPanel requests data from the database, creates scenarios and stores created scenarios in the database. ScenarioSelectionPanel load scenarios from the database and creates the simulation. Created simulation is then passed to the simulation panel.

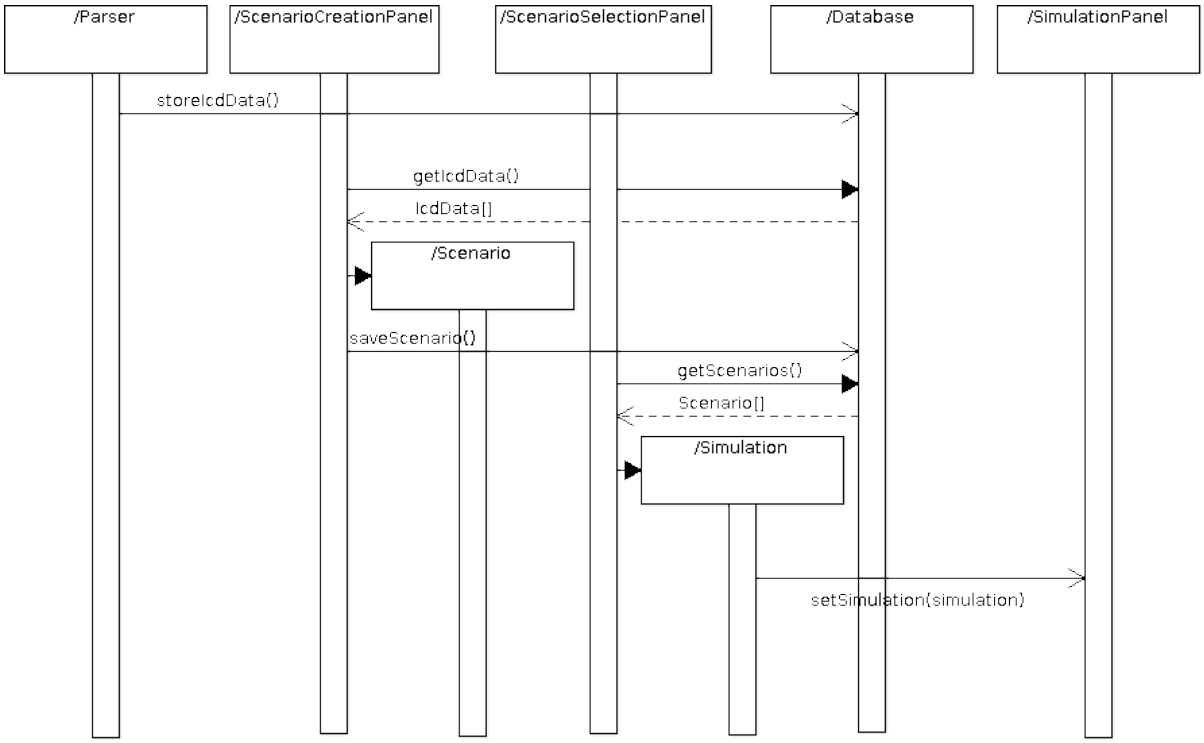

Figure 9: *Database and application objects flow diagram*

The next diagram shows the data flow in the external communication devices between the parts of the desktop app and the android device. Every ICD simulated in the desktop app communicates with the other simulated ICDs via the DsrcDevice which data flow is simulated by the DsrcDeviceCloud. So the chain that every GossipMessage goes through between two ICDs is: ICD1, DsrcDevice1, DsrcDeviceCloud, DsrcDevice2, ICD2. As for the communication with the Android device it is done via the BluetoothDevice.

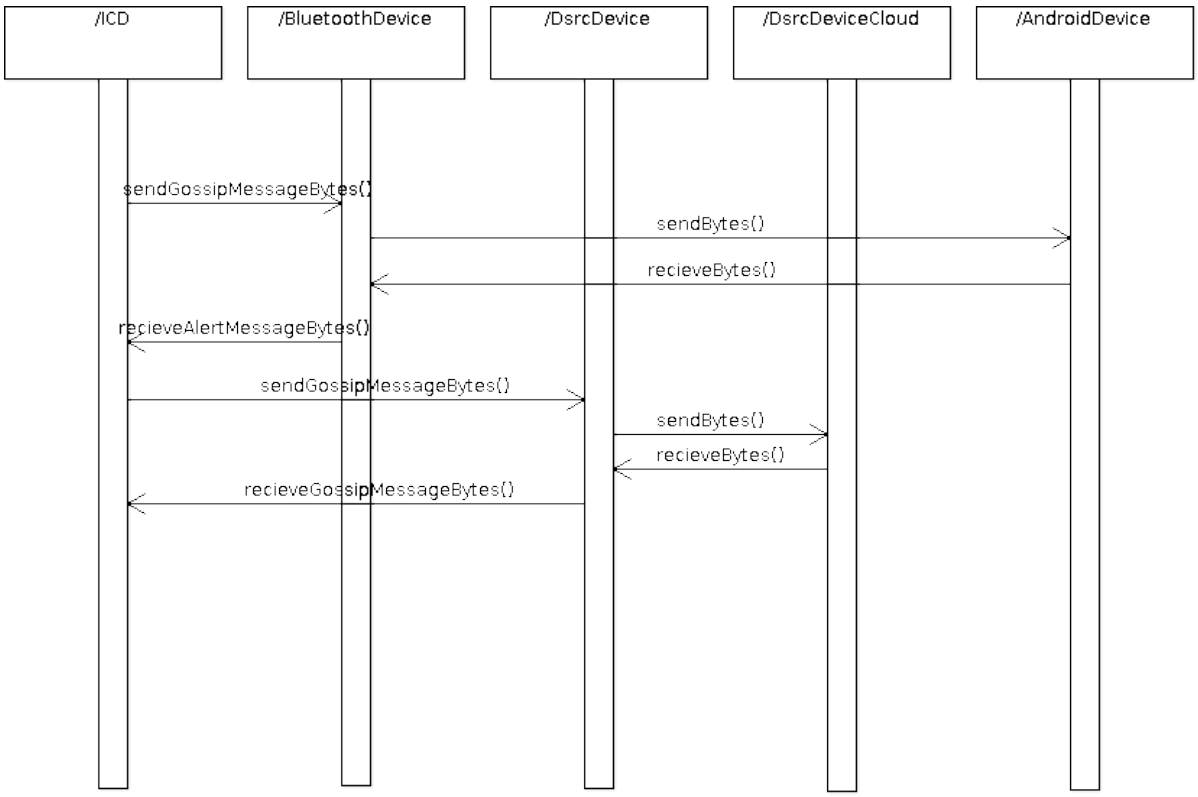

Figure 10: *Simulation and Android external devices data flow diagram*

On the simulation interaction diagram two main types of interactions, ICD data gathering and simulation power controls over the components. ICD gathers data from its sensors: SpeedoMeter, Compass, GpsDevice and Clock. BluetoothAlarm and DsrcAlarm notify the ICD about the new incoming messages and they request the ICD to output its information via the external devices. Simulation controls the Clock time, gathers the IcdData and controls the power state of alarms and the DsrcDeviceCloud.

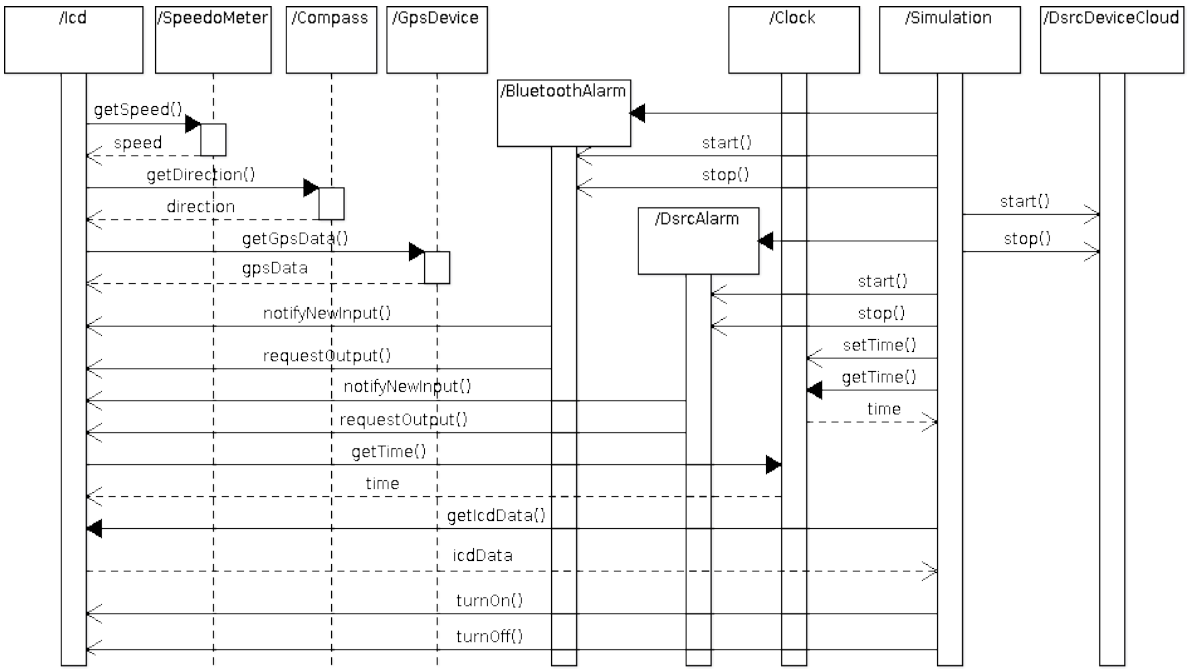

Figure 11: *Simulator interactions diagram*

Database Model

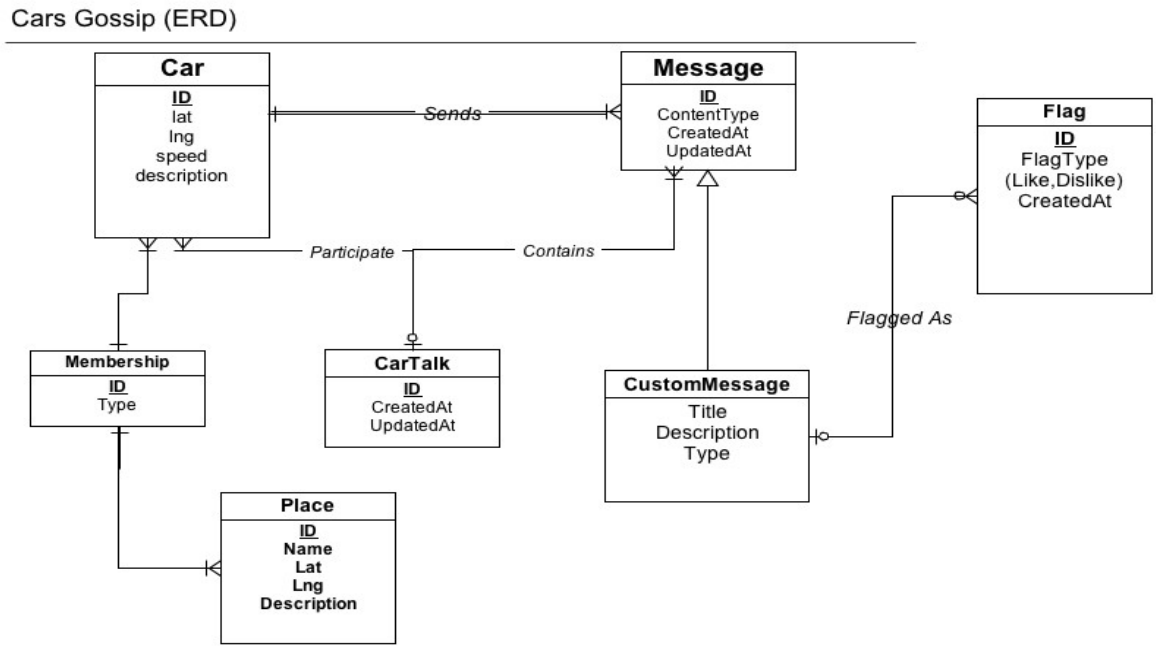

Figure 14: *Web Server DB ER diagram*

Interfaces to External Systems

ICD interface

Send message to other ICDs via DSRC.
Receive message from other ICDs via DSRC.
Send message to Android device via Bluetooth.
Receive message from Android device via Bluetooth.

Android App interface

Send message to ICD via Bluetooth.
Receive message from ICD via Bluetooth.
Send message to Web Server via HTTP(S).
Receive message from Web Server via HTTP(S).

Web Server interface

Send message to Android App via HTTP(S).
Receive message from Android App via HTTP(S).
Send message to Web App via HTTP(S).
Receive message from Web App via HTTP(S).

Web App interface

Send message to Web Server via HTTP(S).
Receive message from Web Server via HTTP(S).

Algorithms

The key algorithm in the Car Gossip system is the Gossip algorithm. Its role is to distribute messages among the ICD devices. It should be robust and it should optimize the message filtering within the ICDs to minimize data flow and maximize data gain.