

Masters in Software Engineering

Internship – 2014/2015

Interim Report

Study and implementation of advanced mechanisms for improving safety driving in the use of messaging applications

João Paulo Coimbra de Sousa

jpcsousa@student.dei.uc.pt

joao.p.sousa@wit-software.com

Wit Software Supervisor:

Eng. Filipe Cardeal Patrão Freitas Dos Santos

DEI Supervisor:

Prof. Dr. Marco Paulo Amorim Vieira

Date: September 02nd, 2015



FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Masters in Software Engineering

Internship – 2014/2015

Interim Report

Study and implementation of advanced mechanisms for improving safety driving in the use of messaging applications

João Paulo Coimbra de Sousa

jpcsousa@student.dei.uc.pt

joao.p.sousa@wit-software.com

Jury:

Prof. Dr. Joel Perdiz Arrais

Jury:

Prof. Dr. Joao Pedro Morais de Matos Moniz Ramos

Date: September 02nd, 2015



FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

This page was intentionally left in blank.

Abstract

It is a known that communication between people is something intrinsic and essential to every human being. Throughout history the way people communicate has evolved and today about 90% of the people use a mobile phone. These numbers are impressive and reflect a growth of this rather sharp technology. [2]

The problem is that along with this fantastic data there appear frightening statistics, illustrating that people don't always know when to use not the phone. In a practical case, the number of accidents due the improper use of the mobile phones is high - studies show that 35% of drivers use the phone while driving.[3]

The solution I propose provides every driver with a way to communicate safely without placing his safety or of those around him at risk. I will create a set of features that will be integrated in the WIT's product - RCS+. It will detect if the user is driving and if so, provide to the driver one of two safer methods of communicate. The driver, mainly, only has to give a voice command to his/her smathphone.

É um fato que a comunicação entre pessoas é algo essencial e intrínscico em cada ser humano. Ao longo da história a forma de comunicar foi evoluindo, e hoje em dia cerca de 90% das pessoas têm um telemóvel. Estes números são bastante agradáveis e denotam um crescimento da tecnologia de uma forma bastante acentuada. [2]

O problema é que juntamente com estes dados fantásticos surgem estatísticas assustadoras, retratando que nem sempre as pessoas sabem quando devem (e não devem) usar o telemóvel. Num caso prático, o número de acidentes rodoviários devido à utilização indevida do telemóvel é bastanet elevado, em estudos feitos retrata que 35% dos condutores o utilizam. [3]

A solução que pretendo oferecer faz com que cada condutor tenha uma forma de comunicar segura, sem que coloque a segurança dos que o rodeiam em risco. Vao ser criadas uma serie de funcionalidades que serão integradas no produto Android da WIT – RCS+. A app irá detetar, se o utilizador permitir, se o utilizador do smartphone está a conduzir e providenciará um, de dois, métodos seguros de comunicar. O condutor irá interagir com a app, maioritariamente, por comando de voz.

Key-Words

Android , Mobility Detection, Rich Communication Services, Safety Driving, Voice Mechanisms, Android Auto

Acknowledgments

First of all, I would like to express my gratitude to WIT Software for providing this amazing opportunity of working on a big company and providing an amazing atmosphere during the internship.

I would like to express my gratitude to both my supervisors, Filipe Santos and Marco Vieira for their support, patience and time spent along the entire internship. I also want to thank the whole Android team of the RCS product for their tremendous help.

Next, a big thank to my beloved friends who helped on the road so far.

Finally and foremost, the present report represents the end of a five-year chapter as a student of Informatics Engineering at the University of Coimbra. I want to give a special thank to all my family and specifically to the person who I owe everything that I am today: **my father**. He is the most important person in my life, and without his continuous effort and sacrifice this would not be possible. A special and eternal thank.

Table Of Contents

Chapter 1 - Introduction.....	14
1.1. Goal	14
1.2. Context.....	14
1.3. The Company.....	15
1.4. Motivation.....	16
1.5. Document Structure.....	16
Chapter 2 - State of the Art.....	17
2.1. Competitors Analysis – Direct Competitors.....	17
2.3. Comparative Analysis.....	21
2.2. Competitors Analysis – Indirect Competitors	22
2.2.1.Applications that provides safety driving mechanisms	22
2.2.2. Indirect Competitors/ RCS+ Competitors	24
2.3. Frameworks	27
2.4. Opportunities	31
Chapter 3 – Approach.....	32
3.1 Goal	32
3.2 Agile Principles.....	32
3.3.1 Why choose Agile ? What is Scrum ?	33
3.3.2. Scrum Roles	34
3.3.3. How it works	35
3.3.4 Planning.....	37
3.4 Risks Management.....	38
Chapter 4 – Requirements.....	41
4.1. Goal	41
4.2. General Definitions.....	41
4.3. System and Users.....	42
4.4. Functional Requirements.....	43
4.5. Non-Functional Requirements.....	53
Chapter 5 – Architecture	56
5.1 RCS+ Product Overview	57
5.1.1. COMLib Layer	58
5.1.2. UI/UX Layer.....	58

5.2 Mobility Detection.....	60
5.2.1. Mobility Detection Module.....	61
5.2.2. Algorithm Instructions.....	62
5.3 Android Auto	67
5.3.1. Open Android Auto	67
5.3.2. Android Auto Features	68
5.3.3. Architecture	71
5.4 Safety Driving Interface.....	72
5.4.1 Mobility Detection Integration.....	73
5.4.2 Smart Dialer Tab.....	74
5.4.3 Safety Driving Features.....	75
5.4.3 Safety-Driving Interface Architecture	81
5.5 Challenges and Difficulties.....	82
Chapter 6 – Quality Assurance	83
6.1 Functional Testing.....	83
6.2 Networking and OS Testing.....	84
6.3 Usability Testing	84
Chapter 7 – Overview of the Project.....	89
7.1. Overview of the First Semester.....	89
7.2 Overview of the Second Semester.....	94
7.3 Changes in the Project.....	97
Chapter 8 – Conclusion and Future Thoughts.....	98
8.1 Overview.....	98
8.2 Future Work	99
8.3 Final Remarks.....	100
References.....	102
Appendix I – Methodology Overview.....	1
Appendix II – Architectonical Context.....	2
Appendix III – Activity Recognition Algorithm.....	3
Appendix IV – Accelerometer Method.....	4
Appendix V –Bluetooth and Handset Devices	5
Appendix VII – Mobility Detection Algorithm	5
Appendix VII – Android Auto Architecture.....	7
Appendix VIII – Safety-Driving Interface Architecture.....	8

Appendix IX – First Semester Planning.....	9
Appendix X – Second Semester Planning	10
Appendix XI – Second Semester Real Work.....	11

Index of Tables

Table 1 - Competitors Comparison	21
Table 2 - Framework Analysis	30
Table 3 - User Stories Table	37
Table 4 - Risks Priority Table.....	38
Table 5 - Priorities Description	41
Table 6 - Requirements Types	42
Table 7 - Number of Functional Requirements	53
Table 8 - Number of non-functional requirements	55
Table 8 - Testing Results Table	83
Table 9 - Second Testing Results Table	84
Table 10-First Testing Version Results	85
Table 11 - Second Testing Version	86

Table of Figures

Figure 1 - Market Share RCS.....	15
Figure 2 - Project Complexity[1]	33
Figure 3 - Definition of Done Table.....	35
Figure 4 - Scrum Overview[1].....	36
Figure 5 - RCS Suite Architecture	57
Figure 6 - RCS stack.....	57
Figure 7 - UI/UX layer	59
Figure 8 - Interaction RCS app with Mobility Detection Module.....	60
Figure 9 - Mobility Detection Module.....	61
Figure 10 - Activity Recognition Sequence Model	62
Figure 11 - Accelerometer Listener Sequence Model.....	63
Figure 12 - Bluetooth and Handset Devices Sequence Model	64
Figure 13 - Algorithm Sequence Model.....	65
Figure 14 - Give permission so that Android Auto use the RCS+ notifications	67
Figure 15 - Main Screen of the Android Auto Simulator	68
Figure 16 - Message Received in Android Auto.....	69
Figure 17 - Reply to a Message via Android Auto.....	70
Figure 18 - Android Auto Architecture.....	71
Figure 19 - RCS+ Settings	72
Figure 20 - Start the safety driving interface.....	73
Figure 21 - Add contacts to the smart dialer tab.....	74
Figure 22 - Remove Contacts from smart dialer tab	74
Figure 23 - Safety Driving Interface for Portrait Orientation.....	75
Figure 24 - Sub-menu for portrait orientation.....	75
Figure 25 - Listen unread messages	76
Figure 26 - See detailed notification.....	76
Figure 27 - Detail Notification in Landscape Orientation.....	77
Figure 28 - Notification Center - Landscape Orientation	77

Figure 29 - Voice Recording Message	77
Figure 30 - Reply Via Text Message.....	78
Figure 31 - Recording Method.....	78
Figure 32 - Reply call.....	79
Figure 33 - Dismiss Notification	79
Figure 34 - Day/Night Filter.....	80
Figure 35- Safety-Driving Interface Architecture	81
Figure 36 - 1st Semester Overview	90
Figure 37 - 2nd Semester work plan	94
Figure 38- 2nd Semester work phases	94

Glossary

GSMA	The GSM Association (GSMA) is a group of mobile operators and companies related to the mobile market, created in 1995, to define standards for mobile communication systems.
MNO	A Mobile Network Operator is both a provider and manager of the technologies and infrastructures required to supply mobile communication services for their users, such as messaging or telephony.
OTT Application	Over-the-Top refers to applications that deliver multimedia content over the Internet.
IMS	The IP Multimedia Subsystem 1(IMS) is an architectural framework for delivering IP multimedia services.
Mockup	Regarding this project, a Mockup is both a graphical illustration of a feature's use cases and a proposal for the User Interface/User Experience of that feature in the current product.
Product Owner	In Scrum, the Product Owner is the person responsible for defining the User Stories that compose the Product Backlog and assign them a priority in order to conduct the Scrum Team's work. The Product Owner can change the priority of those User Stories in order to maximize the value of the work by the delivery date.
RCS	Rich Communication Services is a global initiative by GSM Association (GSMA) to compete the Over the Top applications that are penetrating the mobile communications market at a fast pace.
Scrum	Scrum is an iterative and incremental agile development framework. This process is task priority-oriented, meaning that it seeks to fulfill tasks with higher priority first.
Scrum Master	Scrum Master is the responsible to help both Product Owner and Team Members. The Scrum Master helps the Product Owner to manage the Product Backlog, and helps the Scrum Team to mark all the User Stories in a Sprint as done, by removing impediments and keeping them focused and productive.
User Story	A User Story defines a possible user action with the product in order to state a project requirement in the everyday language. A User Story can comprise multiple Tasks that define meticulously what must be created by the development Team Members in order to mark the story as done.

Acronyms

API	Application Programming Interface
App	Application
FR_XX	Functional Requirement XX
GSM	Global System for Mobile Communication; originally Group Special Mobile
GSMA	GSM Association
GPS	Global Positioning System
IP	Internet Protocol
MMS	Multimedia Messaging Service
MNO	Mobile Networking Operator
NFR_XX	Non-Functional Requirement XX
OS	Operative System
OTT	Over-the-Top
Ph.D.	D octor of P hilosophy
PoC	Proof of Concept
QA	Q uality A ssurance
RCS	Rich Communications Services
SMS	Short Message Service
TTS	T ext- T o- S peech
UI	User Interface
UX	User Experience
VoIP	Voice over IP
WMC	WIT Mobile Communicator

This page was intentionally left in blank

Chapter 1 - Introduction

The internship is part of the part of the Internship/Thesis of the MSc in Informatics Engineering of the Faculty of Science and Technology of the University of Coimbra. This document has the purpose of presenting the work done during the the first semester. The internship is being done at WIT Software's facilities in Coimbra, from October 2014 to June 2015. The work is supervised by Dr. Marco Vieira (DEI) and Eng. Filipe Santos (WIT Software).

1.1. Goal

It is unquestionable the danger that the mobile phone usage causes while the driving situation. Today about 90% of the people use a mobile phone. [2] However with that fantastic data there appear frightening statistics, illustrating that 35% of the drivers use the phone while driving. [3]

The United States Department of Transportation notes that cell phones are involved in 1.6 million auto crashes each year that cause a half million injuries and take 6,000 lives. According to FocusDriven®, up to 80 percent of all crashes involve some form of driver distraction.[4]

This is a crucial problem and affects all of us, for that reason already exists application in the market that try to help the drivers implementing simple mechanisms such as: suppressing audible alerts of calls and messages or send pre-defined messages.

Confronted with this problem WIT Software wants to build a project that makes every user safer, providing advanced mechanisms while driving. This way the driver can still communicate with their friends while driving without placing his life at risk. This system will be like a co-pilot to the driver, will detect if the user is driving, and then help him to communicate with their friend and family with a simple and intuitive design.

This new system will be integrated in WIT's product – RCS+. Will have two main purposes, firstly detect if the user is driving and help the user sending messages and making calls.

1.2. Context

Nowadays the world of telecommunications is becoming more and more competitive, which forces the operators to offer new and innovative services with lower prices than their competitors.

The appearances of new ways of communicate with the creation of the smartphones and these markets. For that reason the operators have to create solutions that are better than the others.

The OTT applications are on rise and companies that are not related with mobile operators are developing them, and put them in a market that anyone can have access.

In 2012, the number of SMS sent globally reached 8600 billions [5], while the number of messages exchanged between OTT applications reached 5846 billions [[5]. This difference

has been declining with time, being expected in 2016 the number of messages between OTTs to double the number of SMS sent.

Therefore, to overcome this trend, GSM Association (GSMA) created an initiative called Rich Communications Services (RCS). This platform enables the delivery of communication experiences beyond voice and SMS, providing consumers with instant messaging or chat, live video and file sharing – across devices, on any network. [6]

For operators, deploying RCS is a key way to ensure their service retains relevance for consumers, keeping them connected and offering them competitive solutions with alternative applications. RCS delivers the messaging service step-change the consumer is seeking, while enabling the continuation of operator-centric messaging services. The scope and variety offered by RCS is a route to developing more targeted, engaging customer propositions – for example via chat-based games, mobile learning, smart ads and promotions – all helping to drive ‘stickiness’ and revenue. [6]

The Rich Communication Services has already a big market share and the projections are to grow in the near future, as showed in Figure 1.

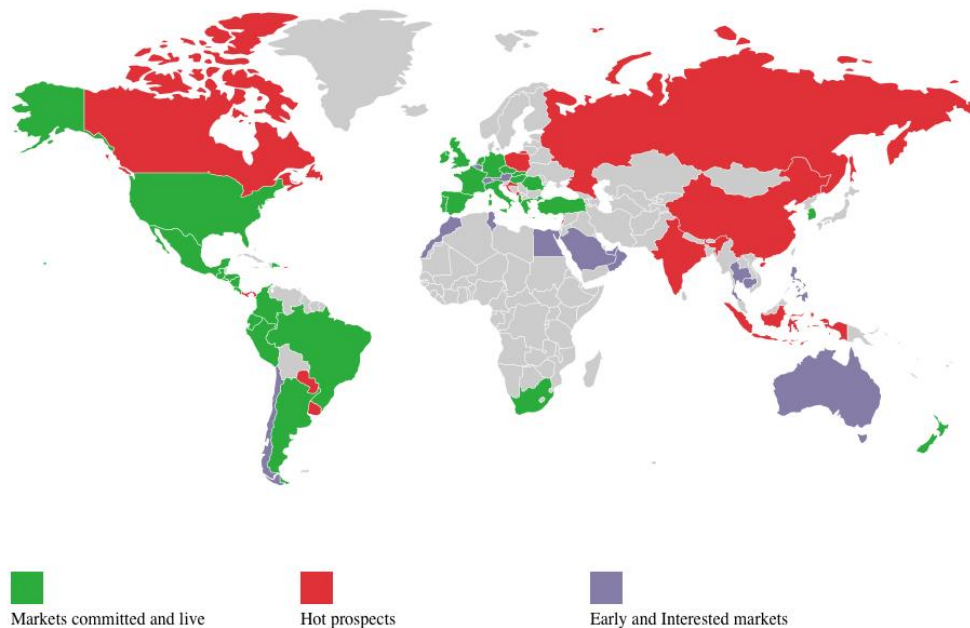


Figure 1 - Market Share RCS

The context of my internship is to develop a mechanism of safety driving that will be added to an existing RCS app. With these improvements in this type of application, it will make them more interesting and be possible to compete with OTT applications.

1.3. The Company

WIT Software is a company, which started as a spin-off of the University of Coimbra in 2001. Its headquarters were at the beginning in *Pedro Nunes Institute* until 2004 when the capacity maxed out, and the team had to move to another place. From then until December 2013, the company grew substantially and had to find a new home again, this time in the building *Centro de Empresas de Taveiro, Coimbra*. Today the company has offices in *Coimbra*,

Porto, Lisboa, Leiria, Düsseldorf (Germany), Reading (United Kingdom) and Silicon Valley (California, USA), and has more than 200 employees.

1.4. Motivation

The problems referred previously are existent and real, but are still underexplored. WIT Software saw an opportunity to overcome their competitors and allow the RCS+ to be more than just a simple messaging app. This application will help the users in their quotidian, make him and those around safer. Some applications already try to implement these mechanisms, but none of them could make it in the way that we want.

1.5. Document Structure

The document will be divided in five core chapters:

2. **State of the Art** – This chapter contains the analyzes made in order to understand what is the project goal analyzing competitors, direct and indirect, such as opportunities that may appear. Were analyzed too some technologies that can be used to help the project elaboration.
3. **Approach** – This chapter explains development methodology used and the project risks.
4. **Requirements** – This chapter explains the requirements defined for the project development.
5. **Architecture** – This chapter contains the non confidential architecture of the internship product. The confidential part will be annexed in the appendix B.
6. **Quality Assurance** – This chapter is focused on verifying the product quality, testing all the purpose features.
7. **Overview of the Project** – This chapter describes the work performed during the internship
8. **Conclusion and Future Thoughts** – This chapter resumes the internship and talk about some interesting features that can be added to the project in the future.

To create a report easier to read some information was removed and added in the followed appendixes.

1. **Appendix A – State of the Art** – This appendix contains is a more extended and detailed version of the chapter 2.
2. **Appendix B – Architecture** – This appendix provides a detailed architecture of the developed features. This document has confidential information, for that reason most precise information is present in this document.
3. **Appendix C – Approach** – This document provides more detail about the software development methodology used and the planning of the project.
4. **Appendix D – Requirements** – This appendix contains all the project requirements.
5. **Appendix E – Quality Assurance** - This appendix explain with more detail the software evaluation.
6. **Appendix F – White Paper** – This document has a paper that was made during the internship to be delivered to the Portuguese Government. This paper is confidential, so it will only be refereced in an specific appendix.

For more precision, please consult those appendixes. Was annexed to in appendix all the important figures in the report.

Chapter 2 - State of the Art

The state of the art aims to provide an overview of the current market where the internship fits. To increase understanding the knowledge about the project this chapter will focus five areas:

1. Direct Competitors
2. Indirect Competitors
3. Future Opportunities
4. Frameworks Available

The first two are done in order to create the best product in the market. To create a product that meet the final user real needs, a plan had to be created. Thus were analyzed what already exists, and detect their strengths and weaknesses.

It is possible, by analyzing competing applications, see the most negative points and focus our strengths on those points.

The third is very important, in order to understand the opportunities that exists to improve the product, and to use something that no one ever use.

Finally, is important to analyze the available frameworks, which will make possible create the safer mobile app.

2.1. Competitors Analysis – Direct Competitors

The direct competitors are the ones that provides to our final user a similar application to the one that is going to be developed.

Is important to learn from the people around us, and better with our competitors. In order to do that will be analyzed those products and find possible approaches and features that can be integrated in the product that will be developed. The best features in those applications will be taken into account to our final product.

To every competitor will be made a short company/product summary, and then will be analyzed the product specification, the disadvantages or things that can be improved and the relevant features for our product.

To help the reader will be displayed only four competitor, to read more please look to the Appendix A – State of the Art.

2.1.1. AT&T DriveMode



AT&T Inc. is an American multinational telecommunications corporation, headquartered at Whitacre Tower in downtown Dallas, Texas.[7]

AT&T create an app that helps drivers avoid distractions from incoming calls and text message alerts. When this app is on, incoming alerts are silenced, text messages can't be sent and incoming calls go directly to voicemail. The main disadvantage of this app is that AT&T DriveMode is only available to AT&T customers. The app is free to those users.

The **main specifications** developed by the app is:

- Auto On/Off – The app start automatically when the vehicle has reached approximately the 40km/h, and turned off when the vehicle stops.
- Auto Reply – If a message is received and the app is on, an auto reply message will be sent to the user.
- Allow List – List of five numbers that is permitted to receive calls while the user is driving.
- Music & Navigation - Only permit music and GPS navigation.

Disadvantages/Things to Improve

- The algorithm to detect if an user is driving isn't the best.
- Restriction of the app by location.

Relevant Features

- Auto On/Off
- Auto Reply
- Allow List
- Music & Navigation

2.1.2. Drive Mode PRO



Drive Mode PRO is an app created by GlassCUBE Inc. The main goal of the app is to announce who is calling and give to every user the decision to answer or reject the call. If the user accepts the call is putted in loudspeaker. [8]

The app has the following specifications:

- Voice recognition when a call is received.
 - Voice detection is operated offline.
 - Give the user the full power by the phone.
- Compatible with almost any voice
 - Compatible with handsets and Bluetooth kits.

- Very low battery usage.
- Enable Talking Caller ID - If the user wants that all calls are rejected automatically and the user doesn't have to worry about the phone.

Disadvantages/Things to Improve

- Manual activation of the "drive mode".
- Few features.

Relevant Features

- Compatibility with Bluetooth devices.
- Voice recognition offline.
- Talking Caller ID.

2.1.3. Safely Go



The Safely Go app was created by The Safely® Family Essentials. The main target is to make every family safe and connected every day. The company receive many awards, for instance, in 2011 was listed in the 100 "most brilliant" companies. The main goal of this app is to prevent distractions of the users while they are driving. So after analysis they have determinate that the best think to do was to decline every calls and SMS while the user is driving. [9]

So the main features are:

- Allow only calls and texts only from 3 contacts, named by "VIP Contacts".
- Reply every other calls and text messages with an SMS that tells that you are driving.
- Enables calls through your Bluetooth or other hands-free device.
- Gives you access only to your top 3 "Driving Apps" (like maps, navigation, or music)

Disadvantages/Things to Improve

- Don't detect if the user is driving. The app needs to be initialized by the user.

Relevant Features

- Every features are very useful. Maybe 3 "VIP Contacts" is limited.

2.1.4 text-STAR



Text-STAR is an app created by Cinqpoint, LLC with the goal of mitigate distracted driving and because of that helps save lives by providing crucial insight and tools to improve road safety. The app allows us too to blocked calls and text messages on a determinate schedule. [10]

The main features are:

- Detect if the user is driving approximately more of 16km/h.
- Schedule auto-reply text in advance if you thing that

you'll be busy or driving.

Disadvantages/Things to Improve

- Detection from GPS data.

Relevant Features

- Block phone calls and text messages.
- Reply with a default message.

2.3. Comparative Analysis

After an analysis of competing applications I was able to detect aspects that did not exist on these applications and some that are interesting to be added to a final solution. Were only considered direct competitors since they are the ones that have the same purpose of what we want to build. So, the follow features are required to have an app that come to the needs of the users:

- Detection of Transportation Mode (Manual/Automatic)
- Block apps that interfere with the driving
- VIP List – List of the users that can receive the calls
- Connection to Bluetooth/headsets
- Read text messages/e-mail and allow the reply
- Text-to-Speech/ Speech-to-Text
- Allow to ring a call that has been received more than three times
- Start a voice call.

The symbols presents in the Table 1 represents the comparison of the features with the applications showed above. The symbols have the following meaning:

- ✓ - feature is provided;
- ✗ - feature is not provided;





				
Detection of Transportation Mode (Automatic)	✓	✓	✓	✓
Detection of Transportation Mode (Manual)	✓	✓	✓	✗
Block apps that interfere with the driving	✓	✓	✓	✓
VIP List	✓	✓	✓	✓
Connection to Bluetooth/headsets	✓	✓	✓	✗
Reply messages/e-mail	✓	✓	✓	✗
Text-to-Speech/ Speech-to-Text	✗	✗	✗	✗
Allow to ring a call that has been received more than three times	✓	✓	✗	✗
Start voice call	✗	✗	✗	✗

Table 1 - Competitors Comparison

2.2. Competitors Analysis – Indirect Competitors

This section will show two types of competitors, ones that are possible competitors of the RCS+ and others that solve the problem of reading and reply messages vocally.

I will start explaining those that solve the problem of sending/reading messages while driving, and next explain the competitors that don't have these functionalities.

2.2.1. Applications that provides safety driving mechanisms

The following applications aren't RCS app indirect competitors. This presents systems that allow to the drivers send and reply to messages by command voice. And comes the question why this is in the section of competitors of the product? The answer is quite simple. These systems provide features that our system needs to have. This analysis helps me understand the features that are required to achieve success. Thus will be presented the features supported by this mechanisms.

2.2.1.1. Google Now



Google Now is an intelligent personal assistant developed by Google. It is available within the Google Search mobile application for Android, iOS and in Google Chrome web browser.

Google Now uses a natural language user interface to answer questions, make recommendations, and perform actions by delegating requests to a set of web services. Along with answering user-initiated queries, Google Now proactively delivers information to the user that it predicts they will want,

based on their search habits. It was first included in Android 4.1 ("Jelly Bean"), which was launched on July 9, 2012, and was first supported on the Galaxy Nexus smartphone. The service was made available for iOS on April 29, 2013 in an update to the Google Search app, and later for Google Chrome on March 24, 2014. Popular Science named Google Now the "Innovation of the Year" for 2012.

Relevant Features

- Can operate in both hands-on and hands-free modes by saying "Google" aloud when the Search app is running (Language must be set to English-US for this to work), or by launching from the home screen. [11]
- Performs web, image, place, and contact searches by voice. [11]
- Can dial, email, or send SMS messages to contacts by voice. [11]
- Can navigate and obtain driving directions, and open other applications on your device just by listening to your voice commands. [11]
- Can post to social networks like Twitter, Facebook, and Google+ for you. [11]
- Is cross-platform, and also available on iOS as part of the Google Search app. [11]

2.2.1.2. Samsung S Voice



S Voice is an intelligent personal assistant and knowledge navigator, which is only available as a built-in application for the Samsung devices.

The application uses a natural language user interface to answer questions, make recommendations, and perform actions by delegating requests to a set of Web services.[12, 13] It is based on the Vlingo personal assistant.[14]

The Galaxy S5 and later Samsung Android devices' S Voice runs on Nuance[15] instead of Vlingo.

Relevant Features

- Making appointments
- Opening applications
- Setting alarms
- Updating social networks such as Facebook and Twitter
- Navigation GPS
- Car mode detection
- When in car mode, notify of:
 - Incoming callers
 - Message senders
 - Play music via Bluetooth

Conclusion

I can conclude that these mechanisms are really necessary for the quotidian of every driver. If that was not true, companies like Google and Samsung does not develop such a tool. However these tools are not specific and sometimes are complicated to use. Thus WIT Software want to add a feature to a product already elaborated and make that application easy to use to every user.

This previous applications inspire me in the elaboration of the requirements of the project in order to understand which mechanisms are really important.

2.2.2. Indirect Competitors/ RCS+ Competitors

The following applications are RCS+ indirect competitors. This presents the OTT applications. And comes the question why this are competitors of WIT Software? The answer is quite easy, although they aren't RCS-based applications, their success "steals" clients of the MNO's. The companies that developed applications to the operators, like WIT Software, consider this application has indirect competitor and have the main goal developed more and better features in order to attract more users, and bring them back to mobile operators.

The following applications only will be referred the relevant features, since they don't have a safe mode implemented.

2.2.2.1. Facebook Messenger



The Facebook Messenger is an instant messaging service and software [16] which provides text and voice communication. Is integrated with Facebook's web-based chat. This application allows the Facebook's users [17] to communicate with his friends, not only by messaging but by VoIP too.

In November of 2014, Facebook Messenger has reached 500 million users [18].

This application is available in multi-platform and in

different

OS, namely Android, iOS and Windows Phone 8.

There is some controversy surrounding the permissions required to govern its functionality.

Some of the permissions include such things as being able to send SMS messages, which may have a cost associated.[19]

Relevant Features:

- Communicate with your friends without open the Facebook.
- Text your phone contacts, even if you're not Facebook friends;
- Record voice messages;
- Know when people have seen your messages;
- See who's available on Messenger and who's active on Facebook;
- Make free calls, even to friends in other countries;

2.2.2.2. WhatsApp



WhatsApp Messenger is an instant messaging app for smartphones that operates under a subscription business model. The proprietary, cross-platform app enables users of select feature phones to use the Internet to transmit communication. In addition to text messaging, WhatsApp can be used to send images, video, and audio media messages. Locations can also be shared through the use of integrated mapping features. The application is distributed to different systems: Android, iOS, Symbian and Windows Phone. On February 2014 Facebook bought WhatsApp for \$19 billion[20].

Relevant Features:

- Allows communication without any costs, is only needed access to the Internet.
- Group chats.
- Send photos, videos, and audio messages;

2.2.2.3. Skype



Skype is a telecommunication application software specialize in providing video chat and voice calls from computers, tablets and mobile devices via internet to other devices or telephones/smartphones.

Skype allows users to communicate by voice using a microphone, video by using a webcam, and instant messaging over the Internet.

In 2014 the application has a market share of 40% of international calls.[21]

Relevant Features

- Send instant messages
- Exchange files and images
- Send Video Messages
- Create Conference calls

2.2.2.4. Viber



Viber is an instant messaging and Voice over IP (VoIP) app for smartphones developed by Viber Media. In addition to instant messaging, users can exchange images, video and audio media messages. This software is available for Mac OS, Android, iOS, Symbia, Bada, Windows Phone, Microsoft Windows. Viber works on both 3G/4G and Wi-Fi networks. It first requires installation on a phone in order to work on a desktop operating system environment.[22] Viber has over 100 million monthly active users from its 280 million global registered users.[23]

Relevant Features

- Works in different environments.
- High definition call quality
- Exchanging of images, video and audio media.
- Address book import
- Text and photo messaging

Conclusion

It's easy to understand that these applications are a threat for all RCS-based application, for some reason this type of application rule the communication market. They provide features that couldn't be found before the existence of RCS-based applications. The texting, chatting and calling are done via these OTT's once most of them are cost free to the end user. By supporting the entire major and, in some cases, minor mobile operating systems; these services have a very high rate of adoption. For that reason the RCS-based applications must have features that those applications doesn't have.

In terms of the proposed features for the internship this applications have flaws. They don't have any security functionalities. In this cases WIT Software wants to be pioneer and elaborate a systems that has security issues implemented. With these functionalities, we expect to have a bigger affluence to the product, because the driver will feel safer with our application then with the others.

2.3. Frameworks

The application will be interacted vocally, in order to support that capability is necessary to have a framework that will help the application read text and convert recording clips to text.

In order to decide which is the best solution, were analysed three frameworks, and at the end was chose the one that were better, according with the analysis.

2.3.1 Acapela TTS



Acapela TTS is a framework design for Android OS and has the purpose of offer a High Quality speech engine specifically adapted to meet the client needs. This

framework provides to Android developers a tool that can easily add a vocal dimension, in many languages, to all their applications.

This framework has the following **relevant features**: [24]

- Very natural and pleasant voices
- Multiple voice formats and qualities
- Multiple Android Devices Compatibility
- Android audio friendly
- Fast and Easy integration
- Multilingual

Disadvantages:

- Android's external framework
- Not free
- Poor documentation

2.4.2 Dragon Mobile Nuance



Dragon Mobile is an API created by Nuance. The company creates Nuance Developers to provide mobile application developers of consumer apps the opportunity to integrate their

innovative apps with Nuance's renowned and proven Dragon Mobile speech platform used by millions around the world today. [25]

This API is very complete and has the following **relevant features**:

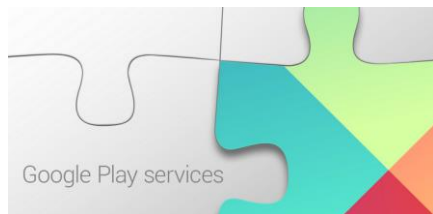
- Advanced Speech Recognition
- Natural Text-to-Speech Voices
- Customization control
- Simple Integration
- Language Extension

- API Documentation

Disadvantages

- Android's external framework
- Not free

2.4.3 Google Play Services



Google, creator of Android's OS had provide API package that provides services to the developers. That framework is named Google Play Services. Initially, when introduced in 2012, it only provided simple access to the Google+ APIs and OAuth 2.0. Although, since then expanded to cover a large variety of Google's services, allowing applications to easily communicate with the services through common means, being internally referred to as simply GMS. [26].

The Google Play Services has the following tools included:

Google Play Game Services

Google Play Game Services can be used by application developers to allow a more competitive and social experience through the use of leaderboards, both public and between friends, achievements and multiplayer sessions.[27] Saved Games API is available to quickly sync game saves on Google's cloud infrastructure.[28]

Location APIs

The Location APIs abstract away specifics about the location technologies, providing Geofencing APIs for scheduling specific actions upon the user entering or leaving specific geographic boundaries, Fused Location Provider for acquiring location information with as reduced power usage as possible and activity recognition for allowing applications to adapt to the current action of the user (e.g. cycling, walking, etc.). This particular APIs were very used during the implementation of the mobility detection. [29]

Maps

Google Maps Android API allows applications to include Google Maps or Street View without the need to open a separate application, allowing full control over the camera and providing means of adding custom markers and overlays over the map.[30]

Auto

Google Auto API provides the possibility of integrate Android's apps with the car providing a simple and intuitive interface which allows to the driver to be focused on the road. This is very important in order to understand the possibility of expansion of our app. This API will be explained in the next section with more detail.[31]

Other

Google Play Services provides other APIs such as the Google Fit API, Google account authentication methods, Google Wallet, Google Adds Google Cast and Google Analytics APIs.[26] Google Play Services is used by almost all Google apps and have system-level powers to provide multiple internal features.

Relevant Features

- Easily Integrated
- Specifically designed for Android applications
- Very Optimized
- Free to use

2.4.4 Google Now



Google Now is an intelligent personal assistant developed by Google. It is available within Google Search mobile application for Android iOS, as well as the Google Chrome web browser.

Google Now uses a natural language user interface to answer questions, make recommendations, and perform actions by delegating requests to a set of web services. This web services will interact with the google play services previously announced.

Google Now is the mainly competitor against assistants such as Apple's Siri.

This service as the following **relevant features**:

- Fully control of the smartphone by voice command
- Now cards[32]
- Add Action to emails[33]
- Answers in search
- Free to use

However have some **disadvantages**:

- Too much control to the user while driving.

Conclusion

In order to choose the best framework to use were analyzed these four and extracted features that were essential for the project elaboration, such as:

1. Text-to-Speech mechanism
2. Speech Recognition
3. Customization Control
4. Easily Integrated
5. Multiple Languages
6. API Documentation
7. Price

After the features extractions were made a comparison table of the four frameworks with the following two simple parameters:

✓ – Positive

✗ – Not positive



Text-to-Speech mechanism	✓	✓	✓	✓
Speech Recognition	✓	✓	✓	✓
Customization Control	✓	✓	✓	✓
Easily Integrated	✗	✗	✓	✓
Multiple Languages	✓	✓	✓	✓
API Documentation	✗	✓	✓	✓
Price	✗	✗	✓	✓

Table 2 - Framework Analysis

After a detailed analysis came to the conclusions that the acapela and Nuance frameworks had to be discarded. They had some minor integration problems and have a major problem, which is that are not free.

With the comparison to the final two – Google Play Services and Google Now was chose the Google Play Services for two main reasons. First, because the location mechanisms; Secondly, and more important, because of the fully controlled environment that is provided. In contrast Google Now provides full control to the driver. Which is not good since the application must ensure driver safety.

2.4. Opportunities

2.4.1. Android Auto

Android Auto is a telematics standards developed by Google, that allow mobile devices run their android operating systems in automobiles trough on dashboard.

The core functionality of this plug-in is to extend this system to the dashboard of the cars that supports this application. For the users use this module need to have the lollipop installed on their mobile phone and have a car that provides a dashboard compatible. The smartphone will be connected to the car by USB and then every notification that the user receives will be sent to the dashboard of the car.

With Android Auto connected there are some functionalities that are very useful as: GPS mapping/navigation, Music control, Telephony, SMS composition and playback, Web search.

This extension is evolving very fast and is already compatible with many apps including Google Maps, Google Play Music, MLB at Bat, Pandora Radio, Spotify, Songza, Stitcher, iHeart Radio, Joyride and TuneIn.

With Android Auto, a driver's mobile device will have access to several of the automobile's sensors and inputs, like: GPS and high-quality GPS antennas, steering-wheel controls, sound system, directional speakers, directional microphones, wheel speed, compass, mobile antennas. On the next future will be possible to retrieve car data.

Actually there are several manufacturers that offered Android Auto in their cars, that is: Abarth, Acura, Alfa Romeo, Audi, Bentley, Chevrolet, Chrysler, Dodge, Fiat, Ford, Honda, Hyundai, Infiniti, Jeep, Kia, Maserati, Mazda, Mitsubishi, Nissan, Opel, RAM, Renault, SEAT, Škoda, Subaru, Suzuki, Toyota, Volkswagen, Volvo.

Considering, it's clear that Android Auto will be the future, and the possibility of having this on the RCS App will be a major progress compared to competitors.



Chapter 3 – Approach

3.1 Goal

The project is going to be integrated in an existing product made by WIT Software. To achieve that, an objective planning had to be made. The planning was done according to a methodology named agile and based on Scrum, which is a framework for iterative and incremental software development. Even not following Scrum to the letter, the overall concepts of this framework were applied.

The document will be divided in five sections:

1. Brief about the agile principles;
2. It will explain the Scrum method and their roles;
3. it will explain the methodology and how it works;
4. it will explain how the planning of the work is done;
5. And last, it will explain the risks inherent in the project, and the method used to overcome those.

3.2 Agile Principles

An Agile methodology has a set of principles that distinguish it from others:

- **Individuals and Interactions** over processes and tools.
- **Working Software** over comprehensive documentation.
- **Customer collaboration** over contract negotiation
- **Responding to change** over following plan.

With the principles stated above, Scrum can solve several risks that used to occur in software development. The main risks and their associated mitigation are the following:

- Risk of **not pleasing the customer**
 - Mitigated through frequent delivery and feedback
- Risk of **poor predictability (estimation and planning)**
 - Mitigated through constant prioritization and estimation
- Risk of **not resolving issues promptly**
 - Mitigated by daily progress meetings
- Risk of **not being able to ship**
 - Mitigated by monthly releases
- Risk of **over-commitment and interruption**
 - Mitigated by “locking and loading” for the Sprint duration

3.3 Scrum

This section will explain the methodology used for the product development – Scrum.

3.3.1 Why choose Agile ? What is Scrum ?

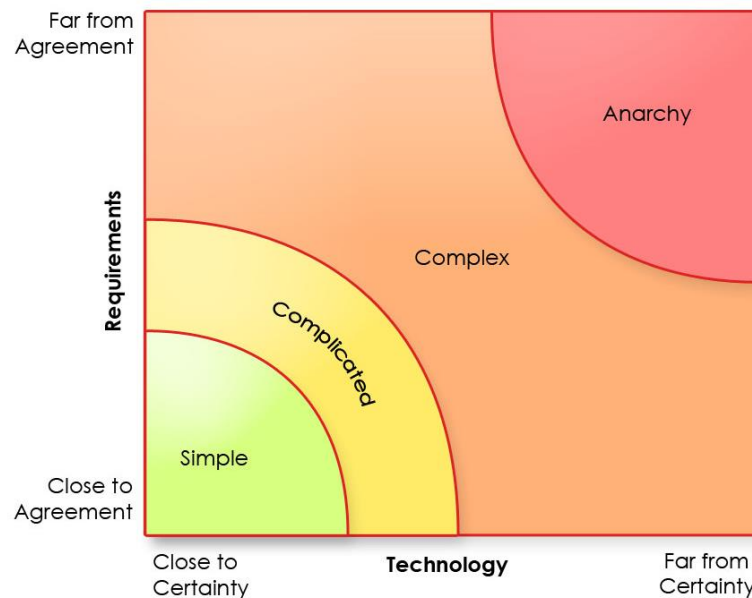


Figure 2 - Project Complexity[1]

According with the **error: reference source not found.**, there are many types of projects in terms of complexity:

- **Simple:** projects that are easily known, namely those which are known 100% and also sure of what has to be done and which technologies to use .
- **Complicated:** projects that have a medium level of complexity. The knowledge of the requirements and the technologies to be used are good.
- **Anarchy:** project that will hardly be made. The requirements are never understood and the technologies that are needed aren't known.

Last, but not least, there is another type of complexity, which is the area of complex types. This kind of project occurs when there are too many possible components involved. In other words there are many possible requirements, and because of that there are many solutions available. For these reasons sometimes, systems are referred to, as, systems not fully predictable.

For these reasons, for this type of project, a waterfall method is not recommended. Planning with all requirements and technologies cannot be well defined, therefore, an Agile methodology is recommended. This methodology allows certain flexibility along the development of the product.

Scrum is the best approach for this kind of projects. “Scrum is a framework for developing and sustaining complex products. A framework within which people can address complex

adaptive problems, while productively and creatively delivering products of the highest possible value”[34].

Scrum allows the necessary agility for projects like the project that is made for the internship, since there is already a project elaborated, new features/requirements are being added. These alterations require discipline and are very complex.

3.3.2. Scrum Roles

In Scrum there are a set of roles, which represents those who are committed to the project namely: the Project Owner, the Scrum Master and the team.

Project Owner

The project owner is the person responsible to analyze the business, customer advocacy and product guidance. His responsibilities are:

- Represents and Manages Stakeholder interests
- Owns the Product Backlog (requirements list)
- Establishes, nurtures and communicates the product vision
- Monitors the project against its ROI goals and investment vision
- Makes decisions about when to create an official release

He has to ensure that his team brings value to the project; he creates a set of requirements that are needed, but ranked in priority and prioritized according to the client’s needs. Every member of the team can write a user story and adds it to the Product Backlog. The Product Backlog represents a comprehensive list of the tasks that need to be done.

The Product Owner of my project is Mr. Rui Gil; he too is a Project Manager, which is good, since he is the one with widest knowledge of the scope of the work and the means to achieve the solution.

Scrum Master

The Scrum Master is the person responsible to maintain a healthy team. He is responsible to lead the team to success and helps it to deal with possible problems The Scrum Master of my project is also my Supervisor, Mr. Filipe Santos.

Team

The team purpose is to execute the vision and the Product Backlog. In my internship I am the only member of the team. My responsibilities are to manage and to fully commit to the work. It is also my responsibility to develop the highest-priority features on the Product Backlog.

3.3.3. How it works

The methodology that is being used, follows an iterating development process. Each work plan has a variable duration of time depending of the features chosen from the Product Backlog by the Product Owner and Scrum Master.

The Product Backlog is composed by a set of user stories; they have different priority and complexity, that way the implementation of the user stories can be made iteratively. The Product Backlog represents the work that has to be done during the internship. Each iteration is called a sprint and has a duration of between 2 and 4 weeks, however, every day there are meetings with the Scrum Master at which I have to report on following questions:

- What did you work on since the last meeting?
- What will you work on today?
- What impediments/blocking issues do you have?

At the end of each sprint the results are presented and their development proceedings analyzed, the plan can then be adjusted if necessary. In order to close a work plan, not only the features composing the work plan need to be concluded. There is the need to explain what the term *done* means. The definition of *done* agreed with the Scrum Master is the following:

Level	Description
Story	UI Conformity according to guidelines Code follows WIT Standards Unit & Integration tests pass Builds with no errors Story implemented Code commented
Work Plan	All bugs fixed.
Release	Summary of changes/additions Approval by Product Owner ^[OBJ] Code committed on SVN Acceptance tests pass

Figure 3 - Definition of Done Table

In the Figure 4 will be described how the different components interact.

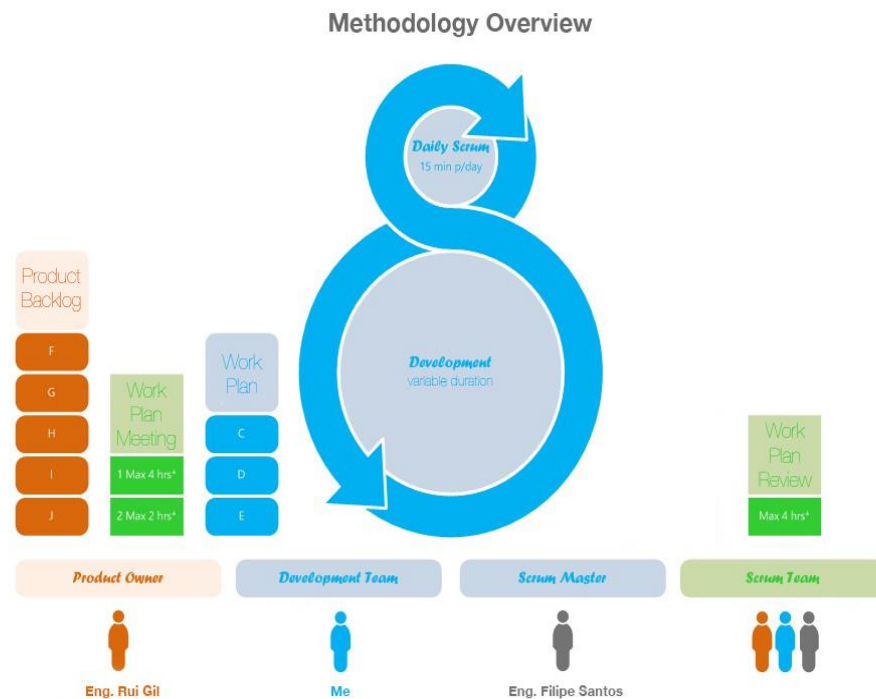


Figure 4 - Scrum Overview[1]

Figure 4 shows the various phases of development of the project.

The Product Owner, in my case Eng. Rui Gil, is responsible along with Eng. Filipe Santos (Scrum Master) to organize my backlog according to the stories priority.

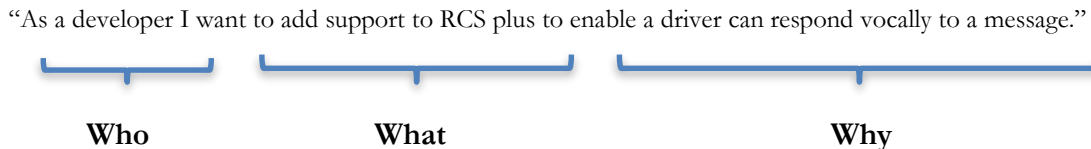
Theoretically the Product Backlog has to be made by the Product Owner, but in this case was made by me and approved by the Product Owner and the Scrum Master. This prioritization results in a subset of user stories, which are contained on the Product Backlog. After this process a cycle of implementation is initiated with a time well defined, between 2 to 4 weeks. I have a daily meeting with the Scrum Master in order to understand how the work is being done and if there's any issue to correct.

When the cycle ends, there's a meeting with the whole team to analyze and review the work done so far. The team will determine what went wrong, and will take corrective action so as to achieve better results, changing the next work plan according to these conclusions. When this period ends, the whole cycle resumes, at which time a new work plan is defined for the next iteration.

3.3.4 Planning

The planning of the project is interconnected with the Scrum approach. As referred to in the previous section a Product Backlog has to be made before the development. For that reason a good understanding of the requirements is essential, and that way the project goals are defined. After that, the Product Backlog will be made according to the requirements. This artefact lists the ideas for the final solution that is expected from the team .

The Product Backlog objective is to achieve the ‘who’, ‘what’ and ‘why’ of the requirements in a simple way. The stories were described as follows:



Each user story belongs to a sub-category and has an associated deadline.

Category	Sub-Category	User Story	Deadline
Implementation	RCS App	<Above>	March
...

Table 3 - User Stories Table

In defining these three user story components, the definition remains very simple and easy to understand. The deadlines are set based on the prioritization done by the product owner and also on the difficulty of each user story.

The Product Backlog, as said before, contains all the features needed to be created in the final solution. However, since this artifact is defined for the duration of the whole project, there’s a need to create work plans at the end of each iteration; this results in the extraction of some user stories from the Product Backlog. The Product Owner and the Scrum Master define the duration of these work plans, and it is based on these tasks that the development is focused, normally from 2 to 4 weeks.

The user stories that integrate work plans are chosen in the course the project. As such, when the planned tasks are completed, a new work plan has to be evolved. Since the company establishes the project it’s possible that small changes have to be made depending on what represents the highest priority at the time.

In conclusion, instead of following detailed planning for the entire project, running the risk of having to reschedule everything in case a higher priority, we chose to mitigate the problem with this approach.

3.4 Risks Management

On the elaboration of a long project, problems may arise and if the team responsible is not prepared this may jeopardize the elaboration of the project and in my particular case lead to the failure of my internship.

For this reason those risks have to be analyzed and a way found to identify and mitigate those same risks. Since the start of project I have taken care to identify all the risks that may arise, for this reason the risks had to be catalogued in a way to be easily identified, and divided by damage probability. When a risk is detected a mitigation technique is applied in order to eliminate the probability of occurrence.

The impact level of the risks to the project were determined as follows:

Classification	Description
1	Very low impact. Can be overcome simply
2	Low Impact. Would require a minor change in the project.
3	Medium Impact. Would require a major effort to make the changes in the project, but possible to make.
4	High Impact. Changes will have to be made to prevent failure of the project.
5	If this happens, the project fails.

Table 4 - Risks Priority Table

In order to identify every risk, they will be catalogued as: id, type, impact, description and mitigation techniques.

Id: 1

Type: Project Management

Impact: 4

Description: If a long sprint doesn't produce the intended result.

Mitigation Technique:

- Create sprints with reduced time.
- Daily meeting with the Scrum Master.

Conclusion: If the sprint is too long and something went wrong, the recovery time is reduced. To fight that risk our team declares that the best way to prevent this problem is to define smaller sprints supplementing these with daily meetings (10/15 minutes).

Id: 2

Type: Project Management

Impact: 3

Description: Bad estimation of the sprint duration.

Mitigation Technique:

- Don't be too optimistic.
- Resort to the experience of the Scrum Master and Product Owner.
- Do not try to discover technologies that are not defined.
- Daily meeting with the Scrum Master.

Conclusion: The estimation of the sprint duration will be made at a meeting with the Product Owner and the Scrum Master; for that reason the team (me) has a important role. On the other hand I am the least experienced, because of that I need to listen to those more experienced than I and seek their advice relative to estimation of sprints. Sometimes I try to understand new technologies that can be made for a specific sprint, but if that is not recommended I must not do it. Meetings with the Scrum Master will assist me to stay with the plan.

Id: 3

Type: Technologies

Impact: 3

Description: Not knowing technologies to use in the elaboration of the project.

Mitigation Technique:

- Analyze and study the technologies before starting.
- Seek help from supervisors.

Conclusion: During the elaboration of the project I encountered technologies that I was not familiar with: for this reason I established a time for study and prototyping.

Id: 4

Type: Project Management

Impact: 2

Description: Alteration of Requirements.

Mitigation Technique:

- Frequently deliver mockups and prototypes of what is being developed to clients

Conclusion: Based on client needs and daily evolution of market requirements the project elaboration may change. For this reason constant contact with the customer is advantage for the project success.

Id: 5

Type: Technologies

Impact: 4

Description: Android API changes.

Mitigation Technique:

- Awareness of the evolution of technology is crucial, this means to listen to news relative to the API's used on the project.

Conclusion: Google usually updates their API's, for that reason I need to be aware of those changes, so that the final product does not present errors.

Id: 6

Type: Technologies

Impact: 4

Description: Android API offline.

Mitigation Technique:

- Prevent the crashes in the times that possible API's are offline

Conclusion: This risk can be prevented is something that I have no control, what must be is to prevent the program from those failures to not cause crashes of the application.

Chapter 4 – Requirements

4.1. Goal

This chapter will focus on the requirements of the project. The main purpose is describing the goals that will be achieved in the internship. This section will allow me to have a guideline of the functionalities and define goals that need to be reached in the final solution.

This section suffer some changes since the start of the project, after the study of the market and comparison with another applications. The project suffer some changes during the process, and that was possible to resolve because the work methodology used – for more detail information consult the Appendix C or the chapter 3.

The purpose of the product is to help every driver that uses RCS+. Statistics tell that 25% of the user's text while drive, and that fact increases the chance of accident on 400%.

For that reason WIT Software want to integrate to the messaging service a mechanism that will help every driver becoming safer.

In this chapter will be presented the list of features that the system will have.

This will be divided in two main groups: functional and non-functional requirements. Each one will have an id, title, priority, type and description. For a better understanding, will be explained in the next section document's structure.

4.2. General Definitions

4.2.1 Priorities

The requirements have different priorities according to their relevance to the project's success. One way of distinguishes those requirements a scale of priorities was created and is presented in the following table.

Type	Description
Must	Requirements that is essential to the success of the project. These are the core functionalities.
Should	Requirements that have a medium priority. They are important to the final solution, but if they aren't done don't imply a failure to the project.
Nice	Requirements of low priority. They have value to the product, but only will be implemented if there's enough time. These are extra requirements.

Table 5 - Priorities Description

4.2.2. Requirements Types

The requirements can be classified also according to their type, making possible to a more detailed definition of the context to which they belong. Below there's a table containing the types that I think that is more important.

Type	Description
Functionality	Requirements that describe the core functionalities of the system.
Usability	Requirements that implies the usage of the user with the application
Performance	System performance.
Confidentiality	Confidentiality of the information retrieved by the user.
Supportability	Scalability, compatibility, configuration, tests, maintainability

Table 6 - Requirements Types

4.3. System and Users

The system, which is the final solution, will be divided in three parts. First will be the mobility detection; followed by the steps that allows to the user that he/she interact with the application: Android Auto and the Safety-Driving Interface functionalities.

The Mobility Detection will be done using multiple tools, namely using the sensors of the phone and geo location.

The Android Auto will provide to the driver a unique experience that will extend the messaging app context to the radio dashboard. This way the driver will have the possibility to receive messages and read, reply.

The Safety-Driving Interface is going to have an UI simple as possible, and will provide functionalities of receive and send messages, complementing with the functionality of making calls. The application will be configured so that won't be intrusive to the user.

To a better understanding how the project was performed please go to chapter 5 or to get more detail to Appendix B – Architecture, that contains the architecture of the application.

4.4. Functional Requirements

These requirements describe what the final product must provide. In other words, are the core features of the application. These features will be divided in three groups, as named in the previous section: the detection of the mobility, Android Auto and the system.

4.4.1 Mobility Detection

FR_01 – Detect mobility using Google API: Activity Recognition	
Priority: Must	Type: Functionality
Description:	<p>The application will have to detect if the user is driving using the Google API – Activity Recognition</p> <p>The application will retrieve information about the user behavior; and will cataloged that information as a one of the following state: driving, walking, running, still, etc.</p> <p>The system will learn about the user’s routines and accurate it’s precision.</p>
FR_02 – Detect mobility using Geofencing methods	
Priority: Nice	Type: Functionality
Description:	<p>The application will have to detect if the user is standing in a place using the Google API: Geofencing.</p> <p>The application will detect if a user is in a place, and watch out when he/she leaves. This method will operate along with the FR_02. This method will allow that the activity recognition mechanism is not always collecting data.</p>
FR_03 – Detect mobility using the previous requirements (FR_01 e FR_02) combined.	
Priority: Must	Type: Functionality
Description:	<p>The application will detect the user’s mobility with the previous two requirements. This will allow that smartphone’s battery can be saved.</p>
FR_04 – Detect mobility via Accelerometer Sensor	
Priority: Must	Type: Functionality
Description:	<p>The application will check if the user is using the phone or not. The way to do that is using the gyroscope present in the phone, and the values in the axis is moving.</p> <p>This method will allow the application to distinguish if the user is driving or not. More precisely, the user can be on a car, but is using the phone, so the system assumes that the user is a passenger.</p>
FR_05 – Detect mobility via handset connections	
Priority: Must	Type: Functionality
Description:	<p>The application will have to detect if the user is driving checking for handset devices.</p>

	<p>The propose of the application is to make the driver safe while is driving, but if he has an handset device, he doesn't have to use the mobile phone to answers calls. On the other hand, the messages need to be read to the user. This allows to the user that the messages can be read on the handset for a better understanding.</p> <p>These handset devices can be Bluetooth connections or handset devices.</p>
--	---

FR_06 – Notify the user that he is driving	
Priority: Must	Type: Functionality
Description:	When the application detects that the user is driving will create a notification that will open the Safety Driving Interface.

FR_07 – Set permission to the RCS+ about the mobility detection methods	
Priority: Must	Type: Functionality
Description:	<p>The user of the application has full control about his/her privacy, so he/she can control if the application will use all the sensors available or not.</p> <p>This will distinguish if use the FR_03 or if will use only the Activity Recognition method (FR_01).</p> <p>The user giving the permission in the RCS+, the mobility detection service will use only the allowed tools.</p>

FR_08 – Turn off automatic mobility detection	
Priority: Must	Type: Functionality
Description:	<p>As the FR_07, the user cannot give permission to the system detect his/her mobility.</p> <p>This FR do not invalidate the use of the Safety Driving Interface. However, the start of that feature will have to be done manually.</p>

4.4.2. Android Auto

FR_09 – Set Permission to Enable/Disable Android Auto	
Priority: Must	Type: Functionality
Description:	<p>The user has control of the RCS+, specifically of the Android Auto extension. The user can give permission to, or not to, provide the Android Auto feature.</p> <p>If the user gives permission, when the smartphone is connected to the car the application will be extended to car’s dashboard. Otherwise will not.</p>

FR_10 – Start Android Auto	
Priority: Must	Type: Functionality
Description:	When the user connects the smartphone to the car - via USB cable, and has given permission for the Android Auto feature to be enable, then the Android Auto will be extended to the car dashboard.

FR_11 – Read Message in Android Auto	
Priority: Must	Type: Functionality
Description:	The Android Auto supports the possibility of read messages at the car using the car columns. This FR will read any message stored at the Android Auto.

FR_12 – Dismiss Notification in Android Auto	
Priority: Must	Type: Functionality
Description:	<p>In Android Auto each message is stored as a notification item. The driver can dismiss that notification when he/she wants sliding the item from right-to-left or left-to-right.</p> <p>When the driver dismisses the notification, it is no longer displayed.</p>

FR_13 – Receive Messages in Android Auto	
Priority: Must	Type: Functionality
Description:	<p>The Android Auto has the capability of displaying messages at the car’s dashboard.</p> <p>The RCS+ receives messages of three types.</p> <ol style="list-style-type: none"> 1. SMS 2. Instant Message from Single Person 3. Instant Message from Group Chat <p>These three types of messages will be registered and shown at the Android Auto dashboard.</p>

FR_14 – Reply to Message in Android Auto	
Priority: Must	Type: Functionality
Description:	<p>The list of notifications displayed at the Android Auto dashboard can be replied. The message will be sent via text, for that reason is needed to convert the voice message said by the driver in text. After the conversion the message will be sent using the received message type (SMS or RCS message).</p> <p>To perform that operation the user can click to reply and proceed the following steps:</p> <ol style="list-style-type: none"> 1. Say the message that want to reply 2. Confirm that the conversion was well performed 3. Send vocally the message

FR_15– Use Google Now in Android Auto	
Priority: Nice	Type: Functionality
Description:	<p>The Android Auto supports the capability of use the Google Now to control the smartphone.</p> <p>This requirement is not important to the purpose of our project, but is important to give the users some control about their smartphone</p>

4.4.3. Safety Driving Interface

FR_16 – Start Safety Driving Interface from RCS+	
Priority: Must	Type: Functionality
Description:	The user can start the Safety Driving Interface manually on the RCS+ settings section.

FR_17 – Start Safety Driving Interface Automatically	
Priority: Must	Type: Functionality
Description:	The user can trust that the application will detect his/her behavior and create the notification that will initiate the Safety-Driving Mode.

FR_18 – Close Safety Driving Interface on the RCS+	
Priority: Must	Type: Functionality
Description:	The user can turn off the Safety-Driving mode on the RCS+.

FR_19 – Close Safety Driving Interface on Android’s notification center	
Priority: Must	Type: Functionality
Description:	The user can turn off the Safety-Driving mode from the notification – only devices that support Android 5.0 will support this feature.

FR_20 – Set Permission to the Safety-Driving Interface read the Messages	
Priority: Must	Type: Functionality

Description:	In order to provide full control of the application to the user, he/she can determine if the Safety Driving Interface can or cannot read the messages loud.
---------------------	---

FR_21 – Set preference to the call response	
Priority: Must	Type: Functionality
Description:	As the FR_20 the user can predefine the call response type <ol style="list-style-type: none"> 1. Native Call 2. RCS+ Call

FR_22 – Set preference to the message response type	
Priority: Must	Type: Functionality
Description:	As the previous the user can set the message response type: <ol style="list-style-type: none"> 1. Message 2. Voice 3. Always ask on the Safety-Driving Interface

FR_23 – Set preference about the incoming messages	
Priority: Must	Type: Functionality
Description:	The user can give permission to the system deal with the incoming messages

FR_24 – Set preference about the voice recording initiation	
Priority: Must	Type: Functionality
Description:	The user can give permission about the start of the voice recording, the possibilities are: <ol style="list-style-type: none"> 1. Start automatically 2. Start manually

FR_25 – Create a notification item to add on the notification center	
Priority: Must	Type: Functionality
Description:	When the driver receives a message, the message is converted on a notification and added to the notification center. Each received message is grouped according with the sender. If the sender already has sent a message then the message will be added to that notification.

FR_26 – Add message received to a existing notification	
Priority: Must	Type: Functionality
Description:	As the previous FR, when a message is received it is grouped according with the sender. If the sender has already sent a message then it will be added to the existing notification.

FR_27 – Create a small notification	
Priority: Must	Type: Functionality

Description:	<p>When a message is received, and the user enables FR23, a small notification will be presented to the driver, highlighting who is the sender and reading the message.</p> <p>At the end of the reading, the small notification will be destroyed.</p>
---------------------	---

FR_28 – Creation a queue to deal with the pending small notifications	
Priority: Must	Type: Functionality
Description:	<p>When multiple messages are received simultaneously, the system will add those to a queue. The messages will be read in order.</p> <p>After the system shows the notification, it will be erased from the queue and proceed to the next.</p>

FR_28 – Stop small notification	
Priority: Must	Type: Functionality
Description:	If the driver wants to skip the reading of a notification, then the user can touch outside the small notification box.

FR_29 – TTS system	
Priority: Must	Type: Functionality
Description:	The system has a Text-To-Speech system, which will allow the system to communicate with the driver.

FR_30 – Delete Notification from notification center	
Priority: Must	Type: Functionality
Description:	The driver can delete a notification from notification center swiping the item from left to right.

FR_31 – See details about a notification	
Priority: Must	Type: Functionality
Description:	In the notification center will only be possible to analyze the not read messages. To listen all messages received while driving, the driver can swipe the notification right to left.

FR_32 – Reply to message by Native Call	
Priority: Must	Type: Functionality
Description:	<p>The driver can reply to a message with Android's Native Call. The call can be done in four situations:</p> <ol style="list-style-type: none"> 1. Notification Center – Landscape Orientation 2. Notification Center – Portrait Orientation 3. Detailed Notification – Landscape Orientation 4. Detailed Notification – Portrait Orientation <p>This requirement has the setting value of the FR_21.</p>

FR_33 – Reply to message by RCS+ Call	
Priority: Must	Type: Functionality
Description:	<p>The driver can reply to a message with RCS+ Call. The call can be done in four situations:</p> <ol style="list-style-type: none"> 1. Notification Center – Landscape Orientation 2. Notification Center – Portrait Orientation 3. Detailed Notification – Landscape Orientation 4. Detailed Notification – Portrait Orientation <p>This requirement has the setting value of the FR_21.</p>

FR_34 – Reply to message by Android’s Native Call automatically when RCS+ type is not enabled.	
Priority: Must	Type: Functionality
Description:	<p>The driver can reply to a message with Android’s Native Call automatically when RCS+ type is not enabled. The call can be done in four situations:</p> <ol style="list-style-type: none"> 1. Notification Center – Landscape Orientation 2. Notification Center – Portrait Orientation 3. Detailed Notification – Landscape Orientation 4. Detailed Notification – Portrait Orientation <p>This requirement has the setting value of the FR_21.</p>

FR_35 – Reply to message by text message	
Priority: Must	Type: Functionality
Description:	<p>The driver can reply to a message via text message. The message can be sent via RCS+ or SMS, according the system capabilities.</p> <p>To send a text message the driver has to proceed to the following processes:</p> <ol style="list-style-type: none"> 1. Speak the message that is going to be converted. 2. The system will convert the message and ask the driver if the message conversion is correct 3. The driver confirms, or goes back to 1. 4. Message is sent via SMS or RCS <p>This feature as to be supported for the following situations:</p> <ol style="list-style-type: none"> 1. Notification Center – Landscape Orientation 2. Notification Center – Portrait Orientation 3. Detailed Notification – Landscape Orientation 4. Detailed Notification – Portrait Orientation <p>This requirement has the setting value of the FR_22.</p>

FR_36 – Reply to message by voice message	
Priority: Must	Type: Functionality
Description:	<p>The driver can reply to a message via voice message. The message can be sent via RCS+ or MMS, according the system capabilities. The recording process initiation behavior is different according with the FR_24.</p> <p>To send a text message the driver has to proceed to the following processes:</p> <ol style="list-style-type: none"> 1. Click in the screen to start (or not, if the setting FR_24 is defined as automatically) 2. Speak the message 3. Click in the screen to stop the recording process 4. Send Message <p>This feature as to be supported for the following situations:</p> <ol style="list-style-type: none"> 1. Notification Center – Landscape Orientation 2. Notification Center – Portrait Orientation 3. Detailed Notification – Landscape Orientation 4. Detailed Notification – Portrait Orientation <p>This requirement has the setting value of the FR_22.</p>

FR_37– Reply to message without any predefined response type.	
Priority: Must	Type: Functionality
Description:	<p>The driver can reply to a message via text or voice message. The message can be sent via RCS+ or SMS/MMS, according the system capabilities.</p> <p>If the driver does not define a response type, when the user click in the button of reply; a dialog will appear asking which response type wants the driver. The response can be of two types: FR_32 e FR_33.</p> <p>This feature as to be supported for the following situations:</p> <ol style="list-style-type: none"> 5. Notification Center – Landscape Orientation 6. Notification Center – Portrait Orientation 7. Detailed Notification – Landscape Orientation 8. Detailed Notification – Portrait Orientation <p>This requirement has the setting value of the FR_22.</p>

FR_38 – Cancel Voice Recording	
Priority: Must	Type: Functionality
Description:	<p>The driver when is recording the voice message (FR_34/FR_35) can cancel the process. This process can be made clicking at the background of the voice-recording layout.</p> <p>This feature as to be supported for the following situations:</p> <ol style="list-style-type: none"> 1. Notification Center – Landscape Orientation 2. Notification Center – Portrait Orientation

	<ol style="list-style-type: none"> 3. Detailed Notification – Landscape Orientation 4. Detailed Notification – Portrait Orientation <p>This requirement has the setting value of the FR_22.</p>
--	---

FR_39 – Sub-menu for the notification item in Portrait Orientation	
Priority: Must	Type: Functionality
Description:	<p>The user can reply by call or message (text or voice). However when the driver has a phone in portrait mode the layout has to be modified in order to facilitate the UI/UX; to do that has to be developed a sub-menu associated to each notification item with the reply options.</p> <p>However, the sub-menu options has two specifications:</p> <ol style="list-style-type: none"> 1. Only one sub-menu option can be active at a time. 2. The sub-menu can be deactivated when the driver clicks outside the layout. <p>This feature as to be supported for the following situations:</p> <ol style="list-style-type: none"> 1. Notification Center – Portrait Orientation 2. Detailed Notification – Portrait Orientation

FR_40 – Listen unread messages from the Notification Center	
Priority: Must	Type: Functionality
Description:	<p>The driver can ear the unread messages. However that option is only active when he/she is in the Notification Center. To perform this operation, the driver has to click the item.</p> <p>This feature as to be supported for the following situations:</p> <ol style="list-style-type: none"> 1. Notification Center – Landscape Orientation 2. Notification Center – Portrait Orientation

FR_41 – Listen all messages received by one user in Detailed Notification	
Priority: Must	Type: Functionality
Description:	<p>The driver can ear all the messages received by a user when is in drive mode (read and unread messages). However that option is only active when he/she is in the Detail Notification (slide item in the notification center right to left). To perform this operation, the driver has to click the item.</p> <p>This feature as to be supported for the following situations:</p> <ol style="list-style-type: none"> 1. Detailed Notification – Landscape Orientation 2. Detailed Notification – Portrait Orientation

FR_42 – See all messages received by one user in Detailed Notification	
Priority: Must	Type: Functionality
Description:	<p>The driver can see all the messages received by a user when is in drive mode (read and unread messages). However that option is only active</p>

	<p>when he/she is in the Detail Notification (slide item in the notification center right to left). The user can scroll to see all the messages.</p> <p>This feature as to be supported for the following situations:</p> <ol style="list-style-type: none"> 1. Detailed Notification – Landscape Orientation 2. Detailed Notification – Portrait Orientation
--	---

FR_43 – Set the night/day filter permission	
Priority: Should	Type: Functionality
Description:	<p>To help the driver while is driving the user can define a day/night filter, that will change the colors/brightness/contrast.</p> <p>This filter can be defined as automatic, and then the system will determine which filter is correctly applied.</p>

FR_44 – Apply the night/day filter	
Priority: Should	Type: Functionality
Description:	<p>To help the driver the system has a feature that changes the environment of the application to help the driver while he/she is driving.</p> <p>According with the setting predefined in FR_40 the system will apply a day or night filter.</p> <p>This feature as to be supported for the following situations:</p> <ol style="list-style-type: none"> 1. Notification Center – Landscape Orientation 2. Notification Center – Portrait Orientation 3. Detailed Notification – Landscape Orientation 4. Detailed Notification – Portrait Orientation

FR_45 – Set permission to enable drive score	
Priority: Nice	Type: Functionality
Description:	<p>The application will have a method of drive score, which will rate the user's driving behavior.</p> <p>The application will detect if the user is driving and using the phone at the same time, without using the save drive mode.</p>

FR_46 – Calculate the Driver Score	
Priority: Nice	Type: Functionality
Description:	<p>The application will calculate the drivers DriveScore according with his/her behavior.</p> <p>The score will increase if the user uses the RCS+ Safety Driving Interface to communicate, and decrease if not.</p>

FR_47 – Set permission to application use the Facebook account in drive score	
---	--

Priority: Nice	Type: Functionality
Description:	The user's Facebook account can be associated to a drive score.

FR_48 – Share the DriveScore with Facebook friends.	
Priority: Nice	Type: Functionality
Description:	The user's score will be shared in Facebook to his/her friends.

FR_49 – Creation of a friends DriveScore ranking friends.	
Priority: Nice	Type: Functionality
Description:	The driver score will be ranked and order by his/her Facebook her friends.

Conclusion

These are the requirements that I propose to accomplish, divided by categories are:

Type of Requirements	# Number
Must	40
Should	2
Nice	7

Table 7 - Number of Functional Requirements

4.5. Non-Functional Requirements

These non-functional requirements describe what the software must guaranteed. These requirements are related to the functional and have the core purpose of maintain the system running when confronted with some problems. They can be defined as different type, such as: security, performance, confidentiality and supportability.

4.5.1 Mobility Detection

NFR_01 – Mobility Detection via Activity Recognition	
Priority: Must	Type: Performance
Description:	The application has to determinate the user's activity with the precision of 90% correctness.

NFR_02 – Mobility Detection via Geofencing	
Priority: Must	Type: Performance
Description:	The application has to determinate the user's activity with the precision of 75% correctness.

NFR_03 – The information of the location of the user mustn't be available to anyone.	
Priority: Must	Type: Security
Description:	The application will not store information about the user. All information will be stored in a database on the smartphone.

NFR_04 – Battery consumption should be low	
Priority: Must	Type: Performance
Description:	The battery used by the mobility detection methods should not increase the RCS+ usage in 2%.

NFR_05 – Code must comply the standards.	
Priority: Must	Type: Supportability
Description:	The code must be in accordance with the WIT standards.

NFR_06 – Architecture must comply the wit standards.	
Priority: Must	Type: Supportability
Description:	The architecture must be in accordance with the WIT standards.

NFR_07 – Don't be intrusive	
Priority: Must	Type: Confidentiality
Description:	The application mustn't be intrusive, can't collect information that isn't authorized.

4.5.2 Android Auto

NFR_08 – Code must comply the standards.	
Priority: Must	Type: Supportability
Description:	The code must be in accordance with the WIT standards.

NFR_09 – The speech to text must be done correctly.	
Priority: Must	Type: Performance
Description:	The conversion of the speech to text must be correct.

NFR_10 – The text to speech must be done correctly.	
Priority: Must	Type: Performance
Description:	The conversion of the text to speech must be correct.

NFR_11 – New message detection must be correct	
Priority: Must	Type: Performance
Description:	The application must detect correctly if a new message was received.

NFR_12 – Lose messages on hold	
Priority: Must	Type: Confidentiality
Description:	The application mustn't lose information about the messages that are on hold. That way the application can tell to the user all the information that he needs to know.

NFR_13 – The interface must respond immediately.	
Priority: Must	Type: Performance
Description:	The interface must respond to the user immediately without crashes. The users won't use an application that doesn't have a quick response time. The maximal time of response must be 3 seconds.

4.5.3 Safety-Driving Interface

NFR_10 – The interface must respect the standards.	
Priority: Must	Type: Supportability
Description:	The entire interface that will be created must be in accordance with what is developed already (RCS App).

NFR_11 – The interface must respond immediately.	
Priority: Must	Type: Performance
Description:	The interface must respond to the user immediately without crashes. The users won't use an application that doesn't have a quick response time. The maximal time of response must be 3 seconds.

Conclusion

These are the non-functional requirements that I propose to accomplish. Divided in categories are:

Type of Requirements	# Number
Must	15
Should	0
Nice	0

Table 8 - Number of non-functional requirements

It is easy to understand that the functional requirements are all “must requirements”. Thus, can be guaranteed the software quality. In *Appendix E – Quality Assurance* is explained how the system was evaluated and how success to all tests.

Chapter 5 – Architecture

This section contains all the architectonic decisions about the work performed during the internship.

The features that will be elaborated are inserted in an existing product. To provide to the reader a better understanding about the context of the application, the first section will give an overview about the WIT's RCS application (RCS+).

The RCS+ is developed for different mobile operative systems. The version that I am developing is for Android, that way the chapter in *Appendix B – Architecture, section 2* will provide an overview about the Android's OS.

The second section presents a detailed architecture of the Mobility Detection algorithm, as well as the API's used.

The third will contain a detailed architecture about Android Auto feature.

Finally, the section four will provide a detailed architecture about the Safety-Driving interface, as well the layout evolution.

The RCS+ has confidential data that cannot be present in this report, that way some information will only be showed in the *Appendix B - Architecture*

5.1 RCS+ Product Overview

The goal of my internship is to develop a feature that provides safety driving and then integrate on a WIT's product - RCS+. For that reason is important that the reader understand this complex application, before start to understand how the features will be implemented/integrated in RCS+.

The Figure 5 represents the IMS network elaborate to a Mobile Network Operator. Shortly, this is an architectural framework for delivering IP multimedia services.

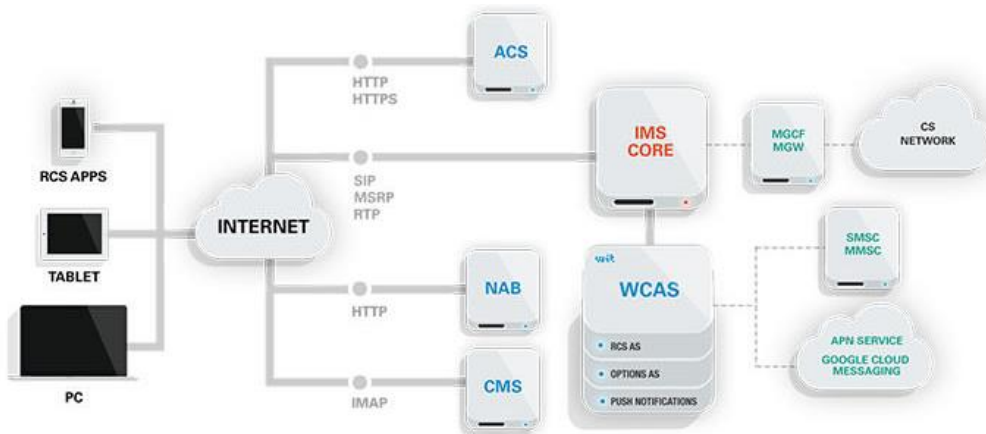


Figure 5 - RCS Suite Architecture

WIT's RCS application (RCS+) lives upon the operation system of the mobile device. As showed the Figure 6 it is divided in two layers: COMLib and the UI/UX layer.

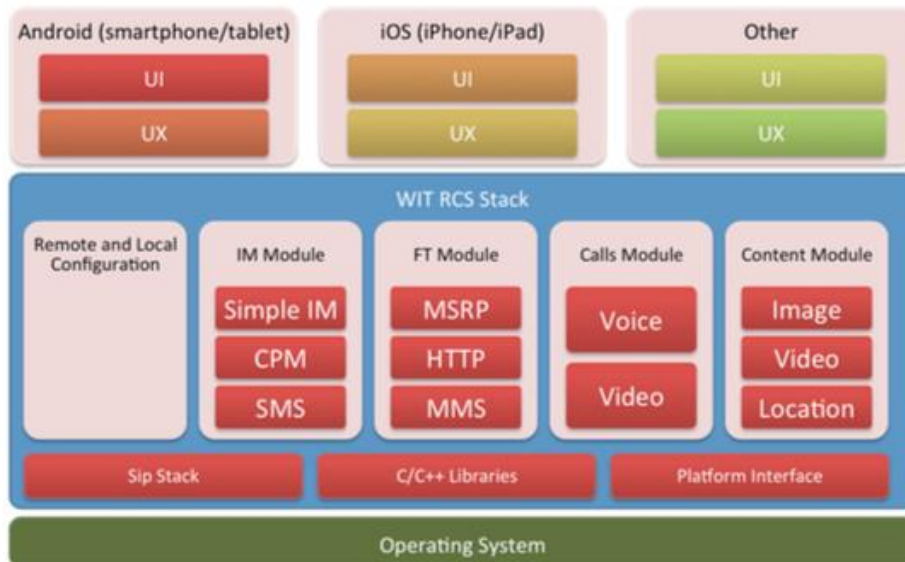


Figure 6 - RCS stack

5.1.1. COMLib Layer

The application uses a communications library called COMLib whose main purpose is to provide all communications functionalities, such as messaging, voice and video calls, and content share. Additionally, it includes many public libraries that are required to process media or to provide secure communications.

In short, COMLib provides all the core services of the application in a Service-Oriented Architecture (SOA). This architectural choice is justified by two main reasons:

In first place, it eases the way to enable and disable library functionalities by configuring the services that may run or not depending on simple configuration files. Furthermore, it allows the application to keep its UI always responsive by providing services that schedule jobs to run on independent threads. This application layer is implemented in C++, meaning that one single version of the code can run on totally different platforms. For example, is possible to run on Android by using a Java Native Interface (JNI) layer that exposes the native code to the UI layer.

Secondly, this RCS application requires a strong interaction with tools that vary from platform to platform, such as the access to the Internet connection state, location sensors, or even radio signal strength. In order to keep the UI side cleaner, the access to platform utilities is executed at the very same level of the communications, and then a specific implementation is applied according the platform.

To enable the communication between the UI layer and the communication stack, COMLib has a sub-layer – named COMLib API – that provides a set of APIs that enable the access to different modules of the application, as well as to register callbacks necessary to receive information after the completion of internal jobs.

5.1.2. UI/UX Layer

This layer stands on top of the communications library, representing all the views and components that are visible to the user, the controllers of those views, and a set of manager that execute the tasks that may involve interaction with the COMLib.

To be more precise, will be only explained the components that are used to implement the proposed features. The Figure 7 will illustrate those modules.

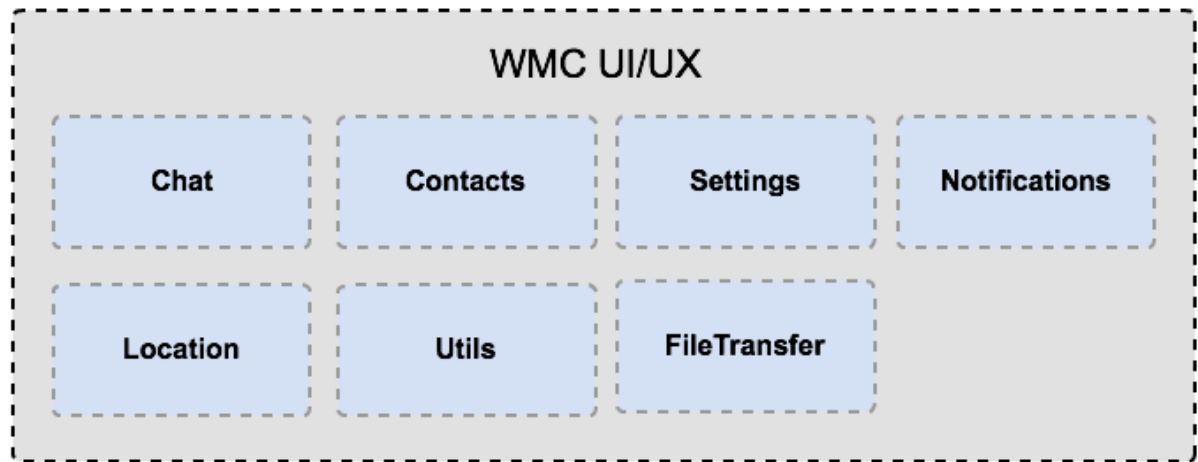


Figure 7 - UI/UX layer

These seven components were used to develop the new features. Each block is composed with sub-modules as: managers, models, controllers and views.

So, why are the packages divided in four groups instead of the three corresponding to the MVC? The answer is simple: managers are controllers, but they are considered a different type for two reasons.

First of all, they are the only objects that interact directly with COMLib in order to retrieve information; secondly, they are instantiated only once upon the launch of the application – they are singletons – in order to be accessed by “normal” controllers via their shared instance.

This way it avoids the assignment of a specific instance to each class that needs to access that manager, which would be very inefficient due to the size of this project.

The **Chat** module contains all the views regarding the chat windows and chat list. These views are commanded by controllers, which will contact with the manager in situations like sending a message or a file transfer, and even deleting a chat entry. All these operations are directed to the manager that will interact with the COMLib in order to start the respective communication process.

The **Contacts** module is responsible for all the management of the contacts, which come from both native and network addresses. The models retain the contact information, controllers command the views to display the necessary information from the models, and the managers are responsible for any operation regarding contacts.

The **Settings** module contains the information about each setting of the application. This module will store all the predefined values of each feature. This module will be important on the control of the application behavior, choosing how the system will behave and what need to be displayed in the UI. This setting module is important to create all settings responsible to the configuration of the Safety-Driving Interface.

The **Notification** module manages all the notifications generated by the application. This module is very important to the elaboration of the internship features. For instance, the Android Auto depends of this module, since works as a notification extender.

The **Location** module is responsible for all the tasks regarding location services. This module will be responsible to retrieve the coordinates relative to a Geofencing.

The **Utils** contains a set of public libraries and frameworks that are required to work on the UI/UX layer. All settings have to be defined in the *Utils* module.

Finally, the **FileTransfer** module is responsible to handle with the file transfer via chat. In a practical case, this module provides capability of sending voice messages.

5.2 Mobility Detection

In this section will be explained all the architectonical decisions performed to develop the mobility detection module and how will interact with the RCS+ .

Before starting the mobility detection analysis, was made a detailed analysis about the Android's OS. To make

Next I will explain how the mechanism of mobility detection will interact with the RCS apps.

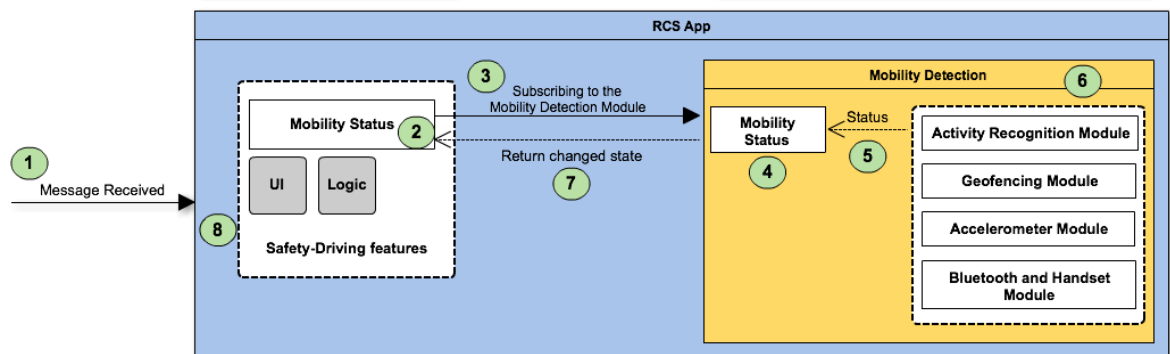


Figure 8 - Interaction RCS app with Mobility Detection Module

With Figure 8 is easy to understand that the changes will be made inside of the RCS+. When the RCS app starts it is created and initiated the service responsible for the mobility detection. That service will run in background and will retrieve and analyze the user's behavior. After that the situation above is the one that will occur. I will explain according with the numbering of the figure above.

- 1 - When a new message is received the system will analyze if the user is driving and according with that, the result will proceed from the normal way or in safe mode state.
- 2 - The RCS app will check if the mobility status is already bounded to the mobility detection service. If is not, then will be bounded in 3. If the mobility status changed to the “driving” state, then a notification is triggered
- 3 - Is made a connection to mobility detection module
- 4 - When the mobility detection service is created is initiated is made a connection to the mechanism that will detect the user’s state: driving or not-driving. This state has a Boolean variable, and when this variable changes will send that information to the RCS app – transition 7.
- 5 - User state return by 6.
- 6 - This will detect the user mobility and returned to 4 by callback.
- 7 - When the user state changes will be by 4 to 2 by callback.
- 8 - This module is responsible for all the safety mechanisms. If the Safety-Driving Interface is open/active then, the system will deal with the driver, helping while is driving.

5.2.1. Mobility Detection Module

In this section will be explained the detection of mobility, only the features 4, 5 and 6 of the **Error! Reference source not found.**

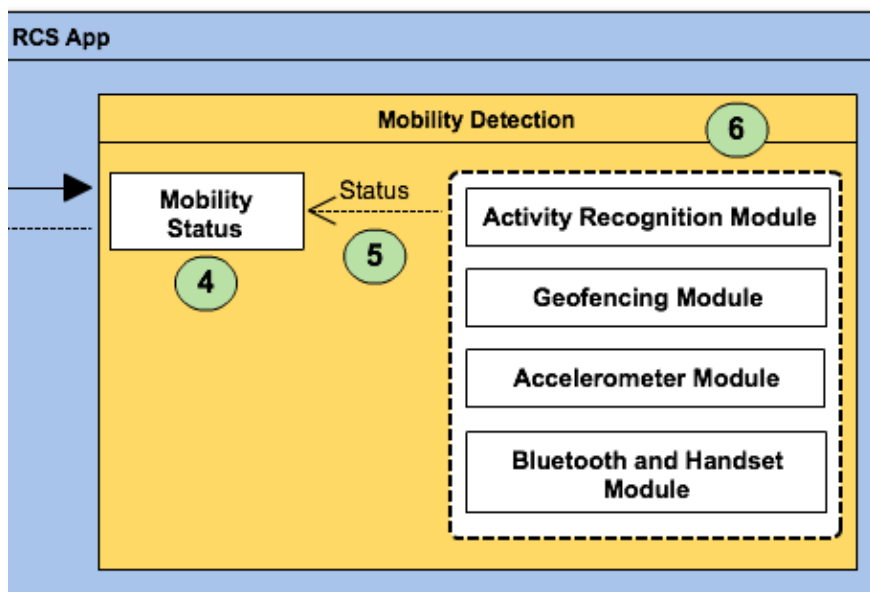


Figure 9 - Mobility Detection Module

Now will be described each module with more detail using sequence diagrams. Then will be explained the data structure of the system. Those diagrams help the reader understand the procedures of each module that composes the Mobility Detection service.

5.2.2. Algorithm Instructions

In this section will be present the algorithms used for mobility detection. Firstly will be explained the activity recognition, accelerometer and Bluetooth and handset methods, and then will be presented the full algorithm.

5.2.2.1. Activity Recognition

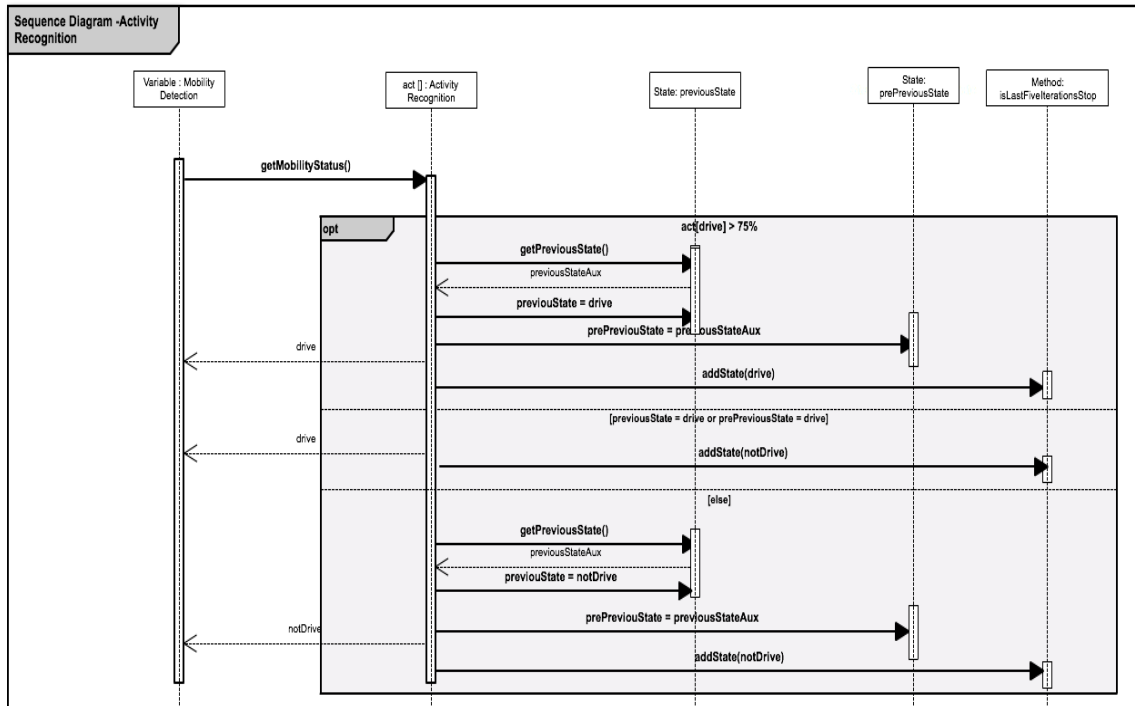


Figure 10 - Activity Recognition Sequence Model

The algorithm will use the Activity Recognition API from Android, it returns the probability for each possible state – driving, walk, bicycle, still, unknown. That method has a timer, and every 30 seconds will retrieve new data and operate the instructions in order to determinate the user’s mobility state.

The purpose of the application is detect if the user is driving, for that reason the only state relevant to return is “driving/not driving”. If the user is driving, then the state will be “drive” and will have more than 75% of probability of success.

If that happens then the state returned will be driving and would be need to update the previous states.

If the probability is lower than 75% then will be needed to verify if the user is indeed not driving or is occasionally stop (p.e. on traffic). If the previous states of the user were driving, then the state returned will be driving, and will be storage it as possible driving. This allows the system to prevent traffic errors.

If neither of the conditions is respected then the system will assume that the user is not driving.

5.2.2.2. Accelerometer Listener

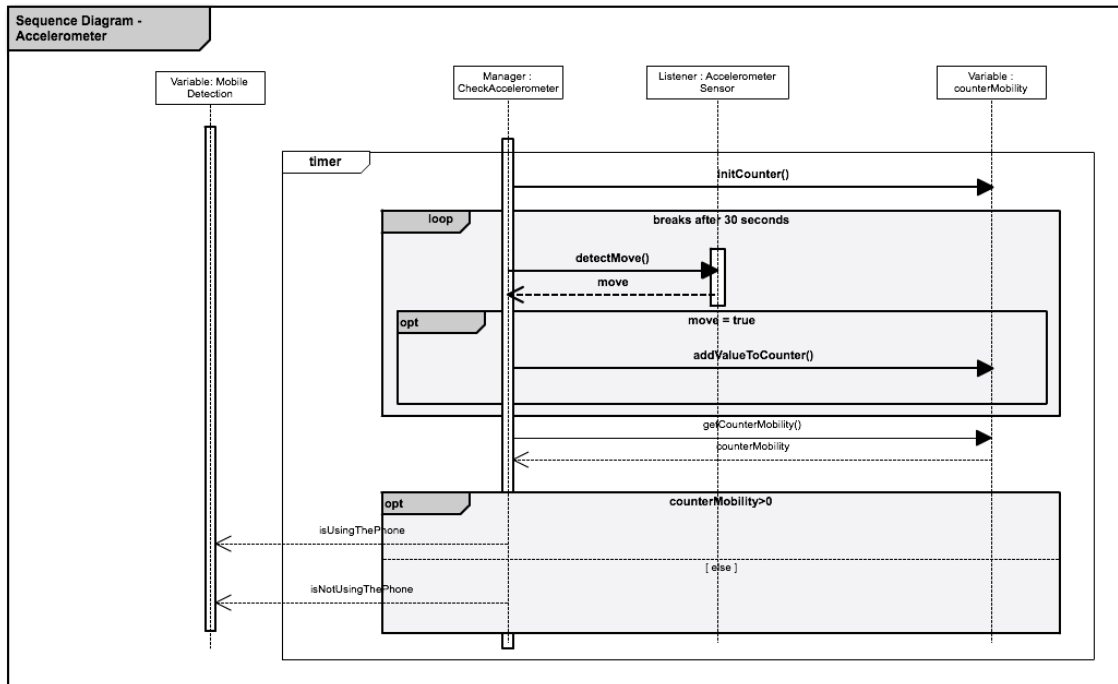


Figure 11 - Accelerometer Listener Sequence Model

To detect if a user is using the mobile phone is used the gyroscope sensor. That way will be possible to distinguish if the user is driving or in passenger seat.

This method is made by using the Android's accelerometer listener. This method is always running and re-starts every 30 seconds. If in those 30 seconds the phone was moved, then will be incremented the counter of movements. If in the end of the 30 seconds that counter is higher than 0, then the user is using the phone. The counter variable is initialized at 0 every 30 seconds.

5.2.2.3. Bluetooth and Handset

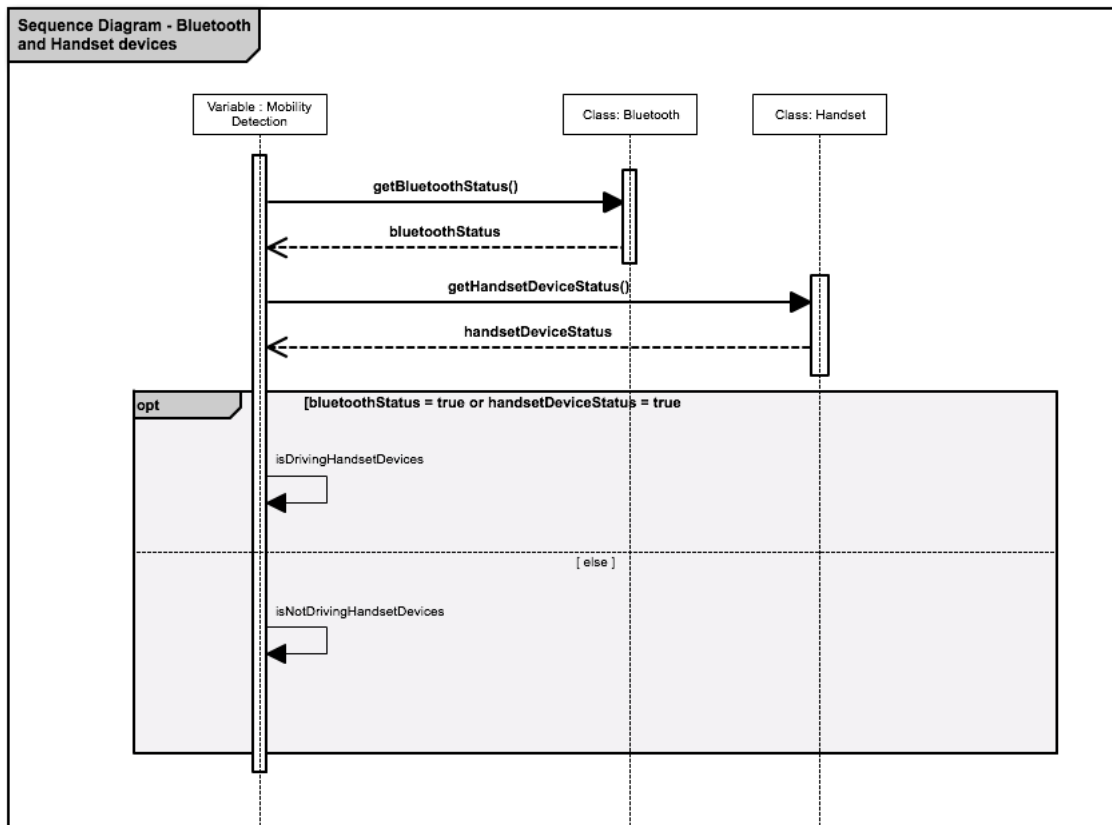


Figure 12 - Bluetooth and Handset Devices Sequence Model

The detection of the Bluetooth and Handset devices are made using API's given by the android. If any of those methods are active then the state will be positive. This method allow to the client listen the messages by those devices.

5.2.2.4. General Mobility Algorithm

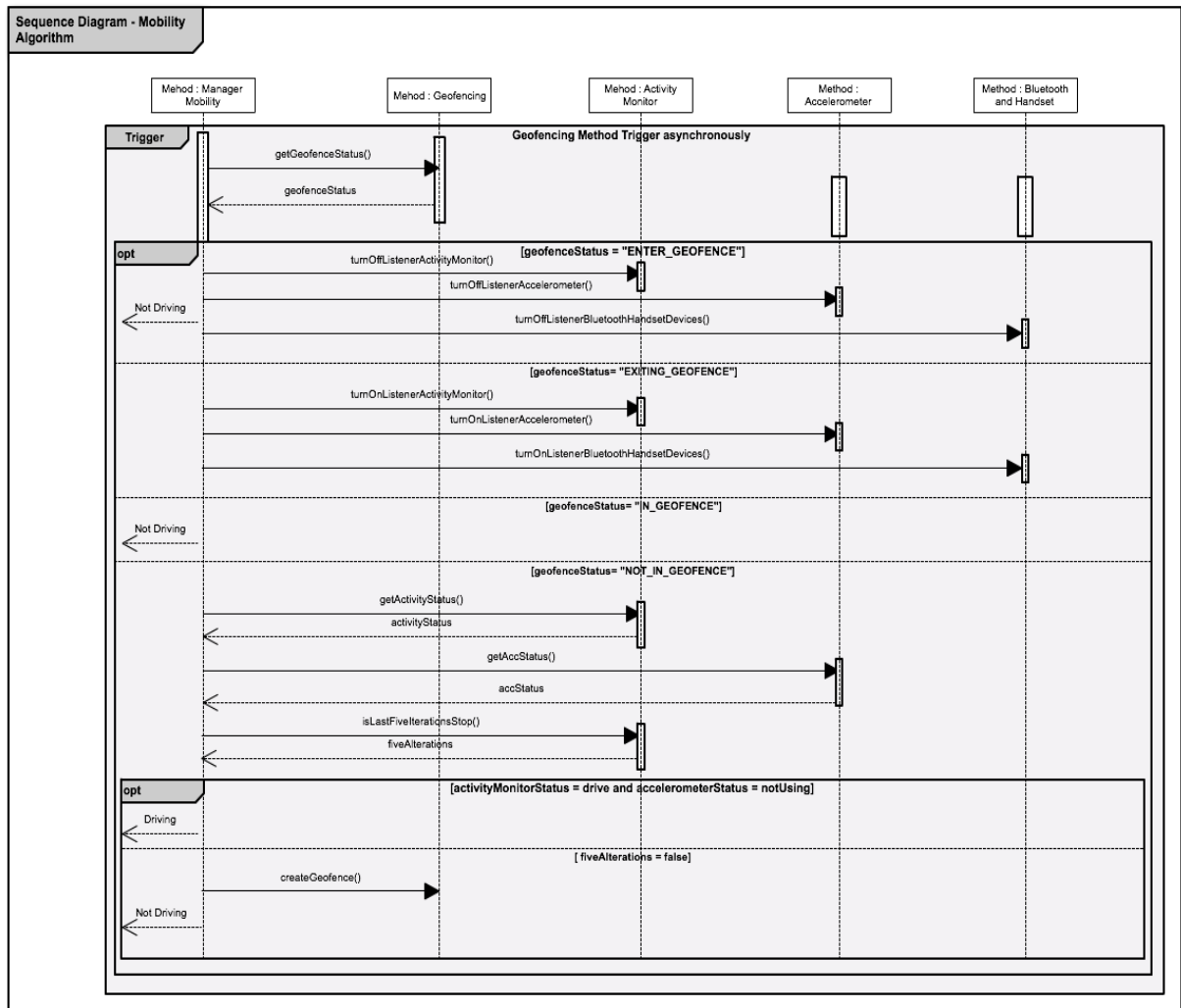


Figure 13 - Algorithm Sequence Model

This sequence model describes the mobility detection algorithm. The manager mobility will be the module responsible for sending the information for the RCS app. Every 30 seconds will be checked if the user is driving or not. Firstly will be retrieved the user's geofence state. That returns four states: "ENTER_GEOFENCE", "EXITING_GEOFENCE", "IN_GEOFENCE", "NOT_IN_GEOFENCE".

According with that information the systems will act.

If there a instruction of "ENTER_GEOFENCE" and "EXITING_GEOFENCE" then the user is entering or exiting a building. These instructions will be good to save battery: when enter a place, the system turns of all the listeners. When the user exits a place then the listeners are re-initiated.

The other instructions are different, if the user is "IN_GEOFENCE", then the user is in a place and then can return the state "not driving".

For least if the user is "NOT_IN_GEOFENCE", this method will allow the system detects if the user is driving or not since he/she is possible out of a place. The system will

retrieve the information given in the previous diagrams. If the user has a state of driving and the not using the phone (accelerometer) then the user is driving, otherwise, if the user is not driving in the last 5 iterations, then will be assumed that he is on a place and created a geofence with that data.

The geofence API detects automatically the entering, exiting of a geofence. The only thing that needs to be done is the creation of the geofence. The last step creates a geofence, and the geofence method automatically detects that the user is in one.

Was performed an analysis of the frameworks to use and the data structure. To allow to the reader a light reading of this document, that information will be sent to the *Appendix B- Architecture*.

5.3 Android Auto

The aim of Android Auto is to extend the RCS+ messaging functionalities to an automobile dashboard's unit. In order to use the system, users must be running Lollipop on their mobile device and must own a vehicle supporting Android Auto. The driver's Android device connects to the vehicle via USB cable. Rather than running its own operating system, the head unit will serve as an external display for the Android device, which runs all of the software, by presenting a car-specific user interface built into Lollipop.

This feature provides the capability of extending some RCS+ features to the car's dashboard. This chapter will be divided in two sections. Each section will have the layout of the feature and the explanation of how it is done in the application source code. The architectonic approach will be presented using the Android Auto simulator.

5.3.1. Open Android Auto

To start the Android Auto the driver has to connect the smartphone to the car radio via USB cable. The user has to give to the RCS+ to share his notification with the Android Auto. To do that the user has to go to Settings/RCS+ settings/ Safety-Driving Settings/ and select the "Permission to use Android Auto". Then the RCS+ will send the notifications to the car dashboard.

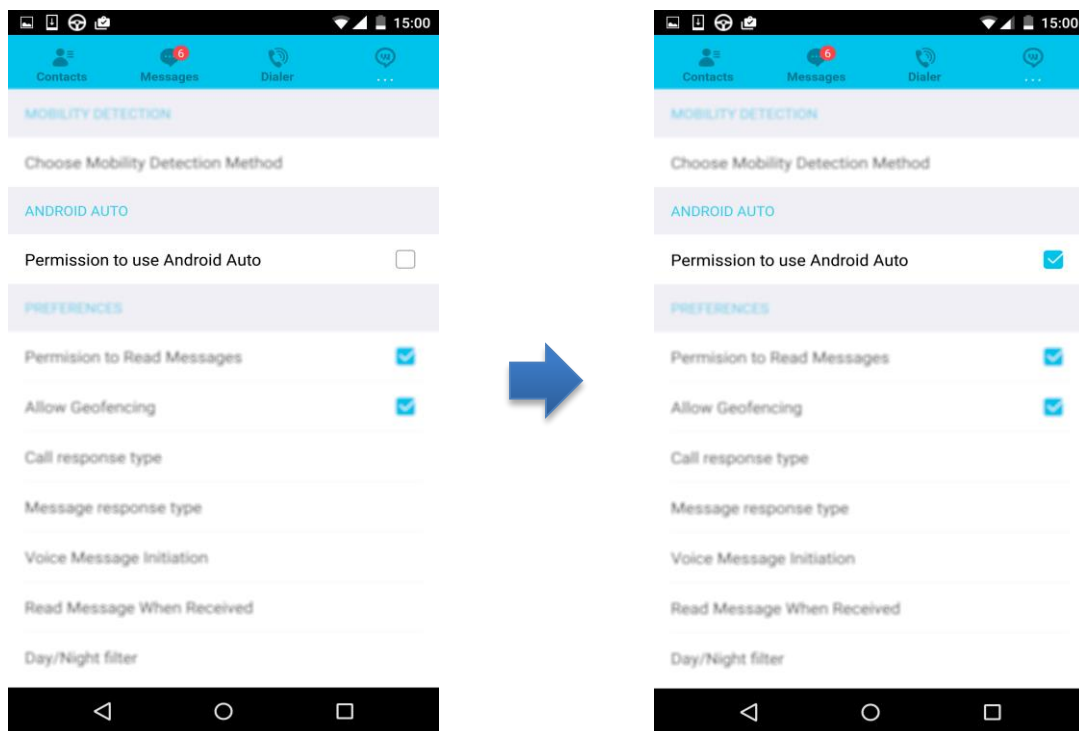


Figure 14 - Give permission so that Android Auto use the RCS+ notifications

Since that was not possible to test the Android Auto on a real vehicle, the tests were made in the Android Simulator. The user opens the simulator and the main screen is opened.

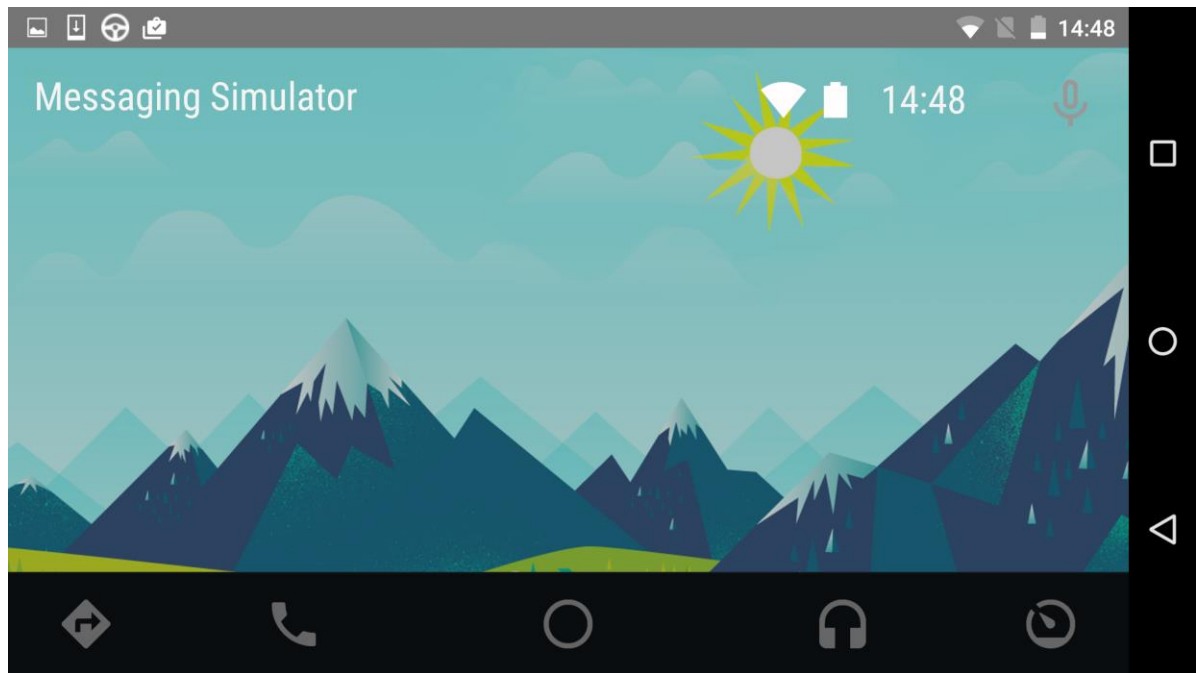


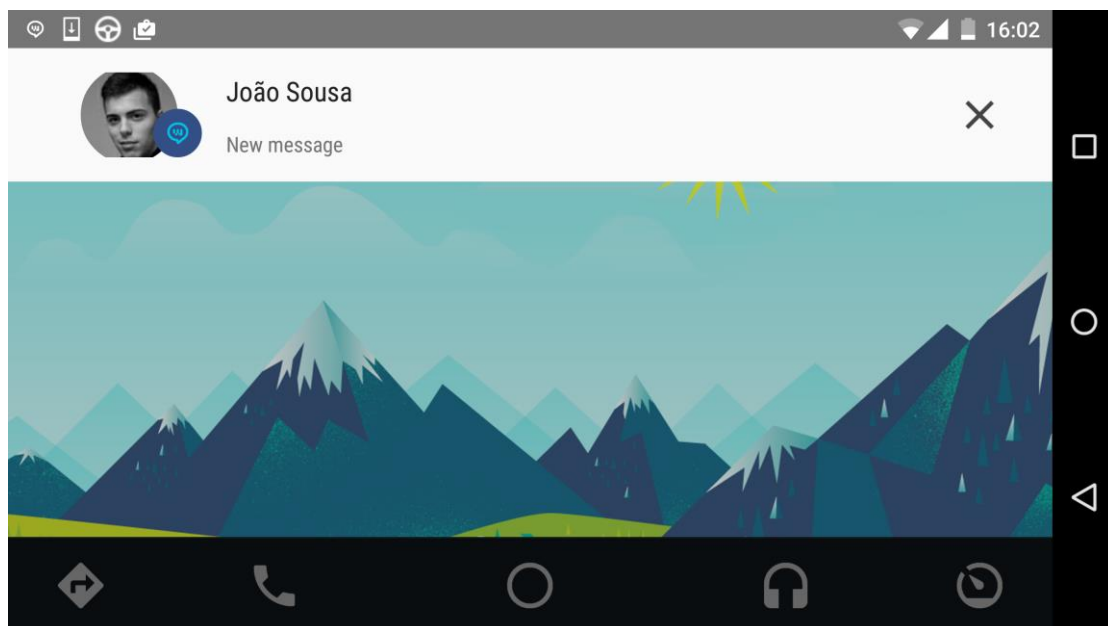
Figure 15 - Main Screen of the Android Auto Simulator

The two core functionalities that the Android Auto provides is the possibility of read and reply to a message, the next two subsections will explain those features.

5.3.2. Android Auto Features

5.3.2.1 Read Message

When a message is received in the RCS+ app, and if the “Permission to use Android Auto” is enabled then, a message is received. The notification will be extended to the car display.



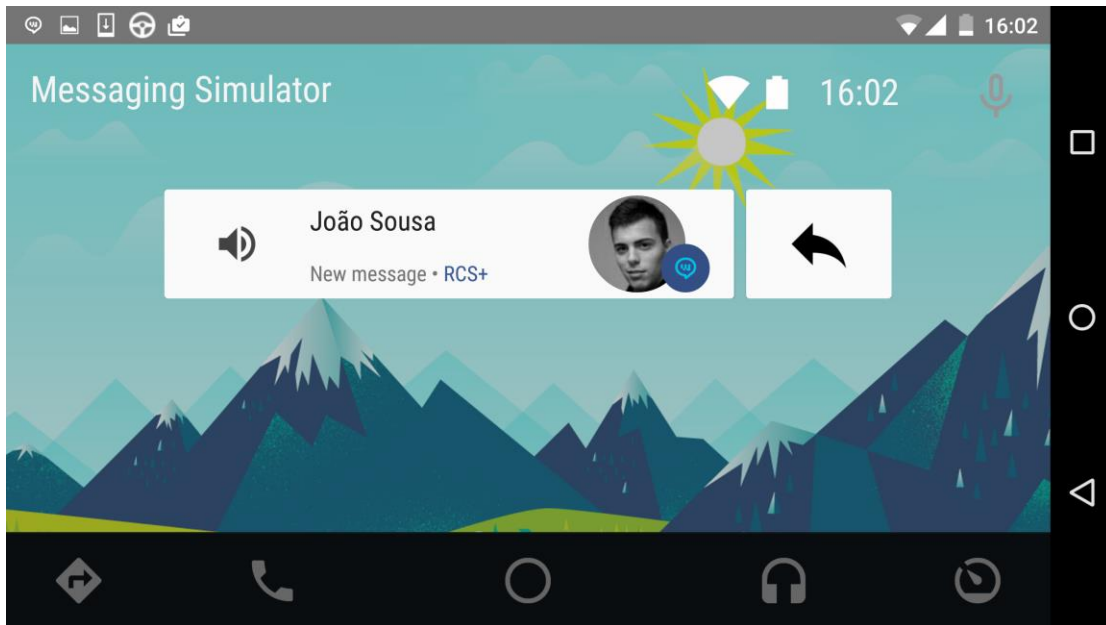


Figure 16 - Message Received in Android Auto

5.3.2.2. Reply Message

When a driver has in the car dashboard a message, and wants to reply he can do it following the next steps.

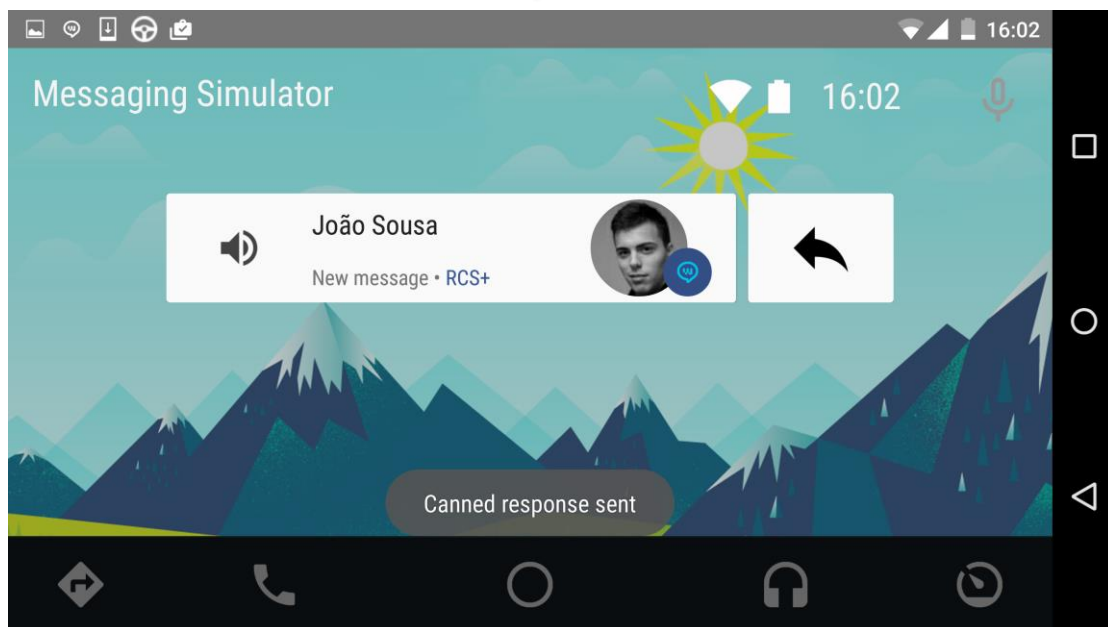
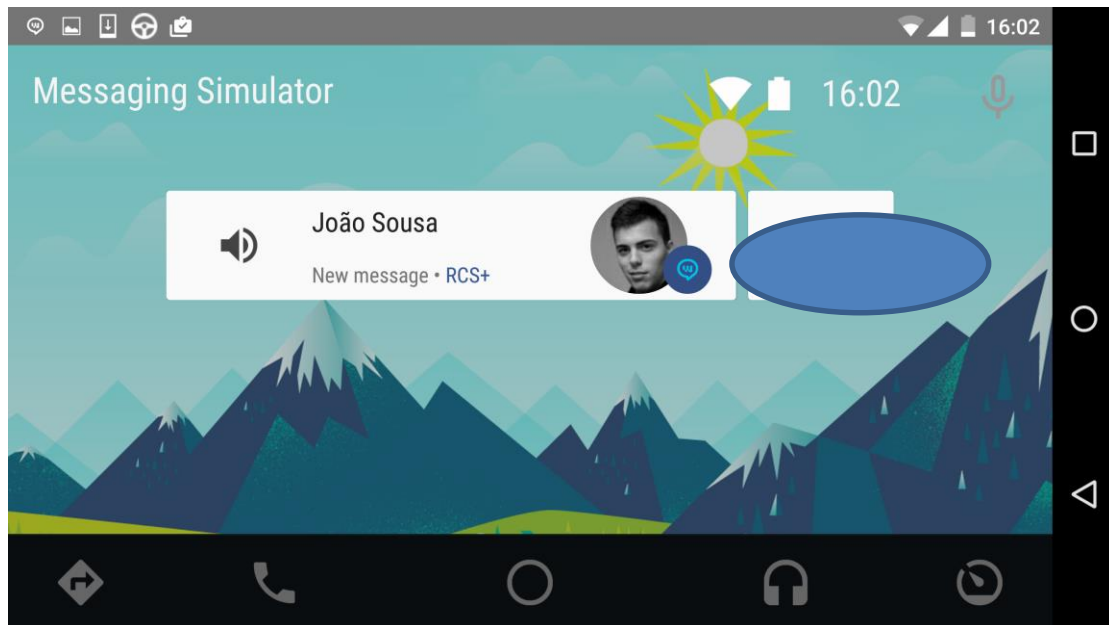


Figure 17 - Reply to a Message via Android Auto

5.3.3. Android Auto Architecture

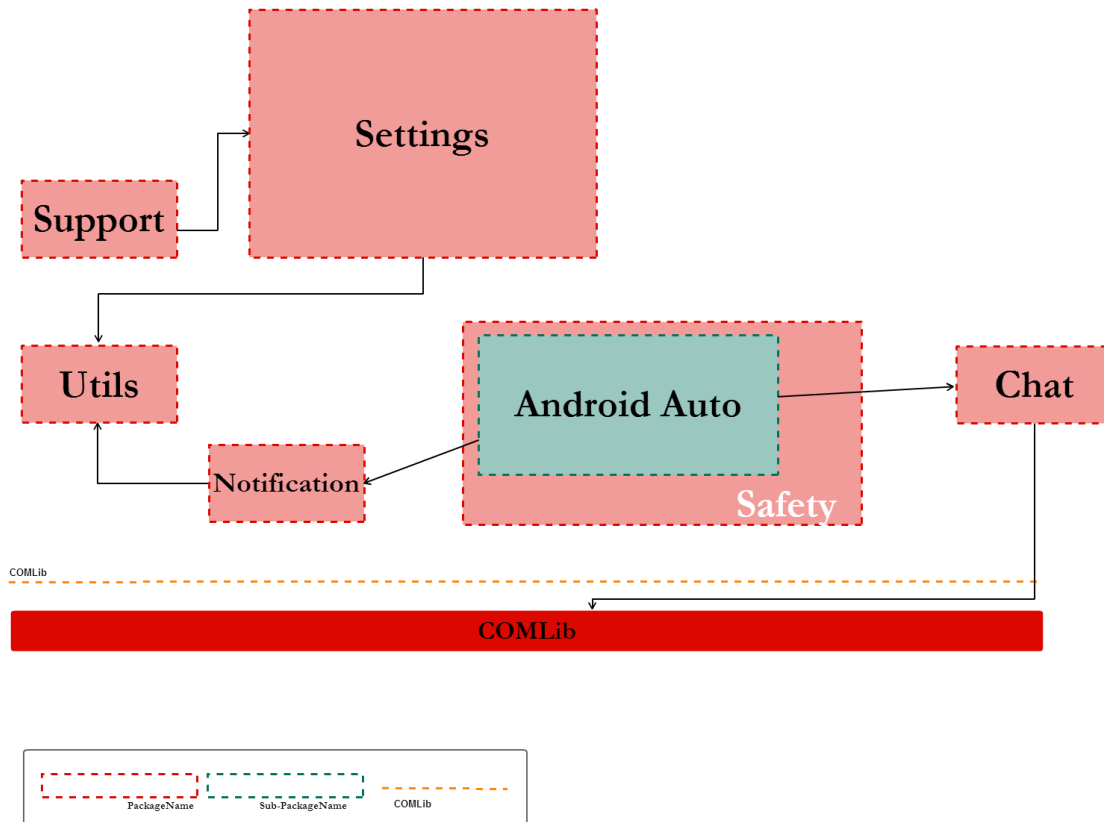


Figure 18 - Android Auto Architecture

Figure 18 illustrates the chosen architecture to develop the Android Auto features. This is a shorter version of its architecture, to see in more detail I recommend you to read the *Appendix B – Architecture*.

The app behavior will depend of the user configuration, for that reason the UI of the settings were designed in the support module, and implemented in the Settings. The information retrieved by the user will be stored in the Utils package. The Safety module, more precisely the Android Auto module will configure the app notifications to follow the received messages to the car dashboard.

In Android Auto package is referenced two broadcast receiver, one to listen to read messages, and the other to deal with the interaction of the user with the car and with the messages replied. When the app receives de instructions that has to send a message, the information the right information is sent to the Chat module, which will send the message using the COMLib.

I strongly recommend to the reader to consult the Appendix B - Architecture. The information in it is more precise and detailed.

5.4 Safety Driving Interface

The Android Auto provides capabilities to extend the messaging application to the car dashboard, however is only available to users that have the Lollipop OS and a car that provides those functionalities.

Since, the number of users that have these tools is much reduced was necessary to develop a number of features that would provide the same capabilities that Android Auto. Thus, was added to the RCS+ those features.

This section will demonstrate all the architectural decisions made in order to develop a final product that is easily and manageable for all types of drivers.

To reach a layout that fulfill the user need was made a set of changes. Those changes are demonstrated in the *Appendix B- Architecture*. I choose to not add in this report to become it easier to read. In this report is only described the final version of the layout.

The Safety Driving Interface are a corresponding set of features that were added to the RCS+. This module have the following features:

- Mobility Detection Integration
- Smart Dialer Tab for VIP Contacts
- Safety Driving Features
 - Display Received Messages
 - Listen to Unread Messages
 - See Detailed Notifications
 - Listen to All Messages
 - Reply by voice message
 - Reply by text message
 - Reply by call (RCS or Native)
 - Dismiss Notifications
- Day/Night Filter

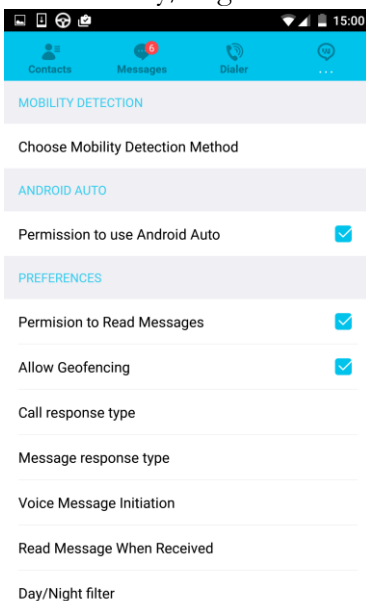


Figure 19 - RCS+ Settings

The system was made to not be intrusive to each user, so all features can be configured in the app settings – as can be seen in Figure 19.

The settings tab is divided in nine configuration menus. The Choose Mobility Detection method has the option of activate the algorithm and detect automatically, and the manual options (on/off)

To set permission to use the Android Auto feature is provided a check to confirm (or not) the usage of Android Auto.

There two similar permissions to set permission to read messages and to allow geofencing detection.

The next four are relatively to the answer to messages, such voice or text.

The call type can be configured as Android's native call or RCS call.

The message response type can be defined as voice, text or ask when the driver tries to reply.

The voice message initiation helps the driver recording voice messages, with the option on, the voice recording process starts automatically.

The read messages when received setting is responsible to create a short notification, and read a message when it is received.

The last settings changes the background colors according with the day time. If it is day, the brightness will be at maximum, if is at night the colors will be darker. This filter can be defined as automatic or manually defined by the user.

After understanding the system settings is important to advance to proper features.

5.4.1 Mobility Detection Integration

After realizing how the mobility detection were developed in important to understand how this module is integrated with the RCS+. All the safer features are triggered in the app settings. The mobility detection method will trigger a notification if was detected a *driving* state. The Safety Driving Interface will be triggered manually if the user wants.

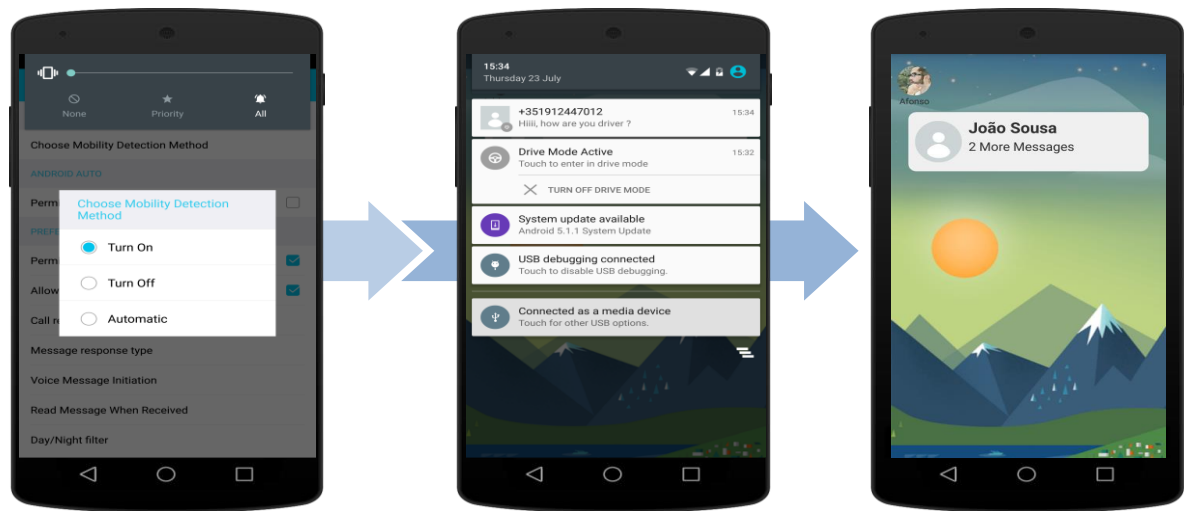


Figure 20 - Start the safety driving interface

The system can be initiated manually if the option selected is “Turn On” or automatically that were analyze the user’s behavior and if detects a driving state will trigger the notification “Drive Mode Active” to initiate the Safety Driving Interface.

To end the safety-driving interface the user has two options to do that: via the app settings – clicking “Turn Off” or in the notification.

5.4.2 Smart Dialer Tab

This feature has the purpose of create a tab in the Safety Driving interface that will provide to each driver to start a call. The user can manage the tab, adding and removing contacts from the contact list, to then start calling those contacts.

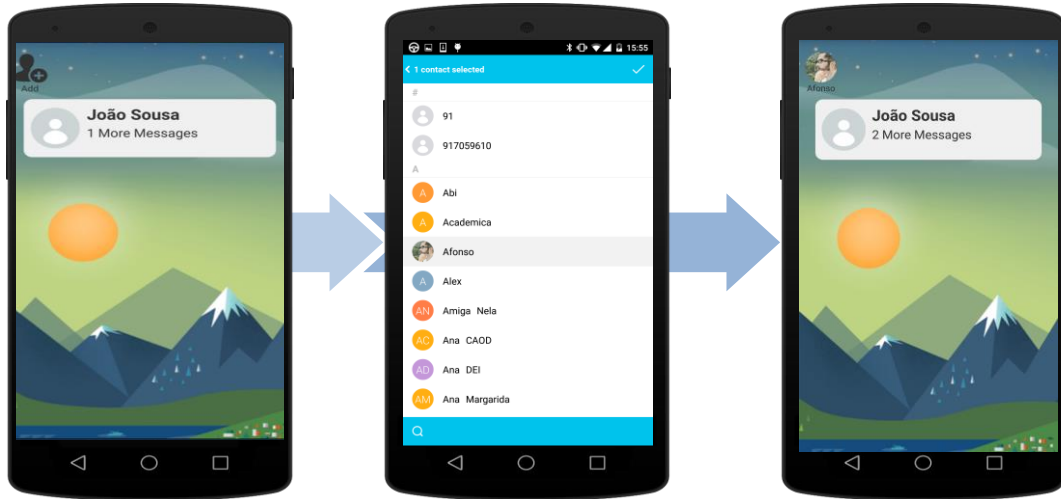


Figure 21 - Add contacts to the smart dialer tab

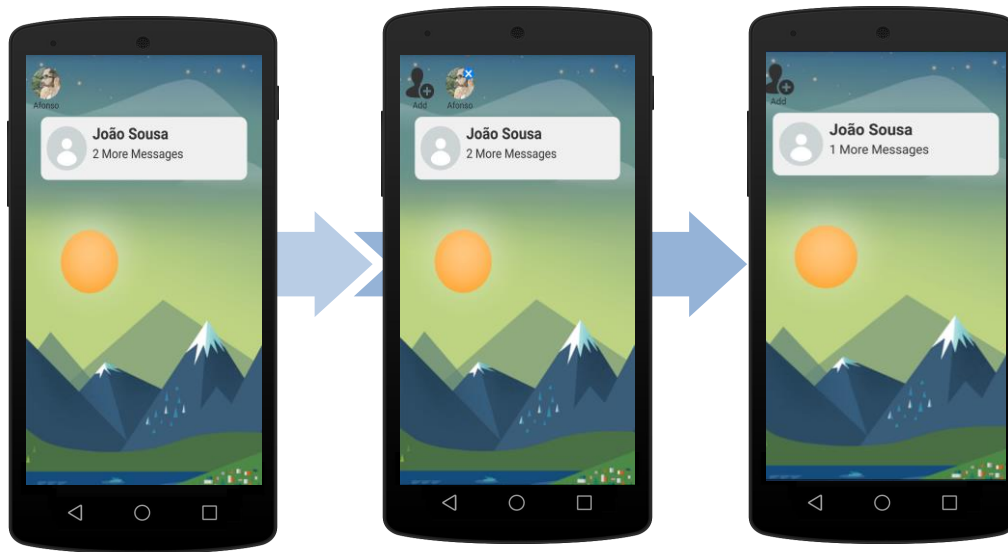


Figure 22 - Remove Contacts from smart dialer tab

5.4.3 Safety Driving Features

This set of features contains the core functionalities of the Safety-Driving Interface, which includes display received messages, listen to unread messages, see detailed notifications, listen to all messages, reply by voice message, reply by text message, reply by call (RCS or Native) and dismiss notifications.

To implement this features were design two layouts – for portrait and landscape orientation. First will be explained the features according with the portrait orientation and then for the landscape.

5.4.3.1. Portrait Structure

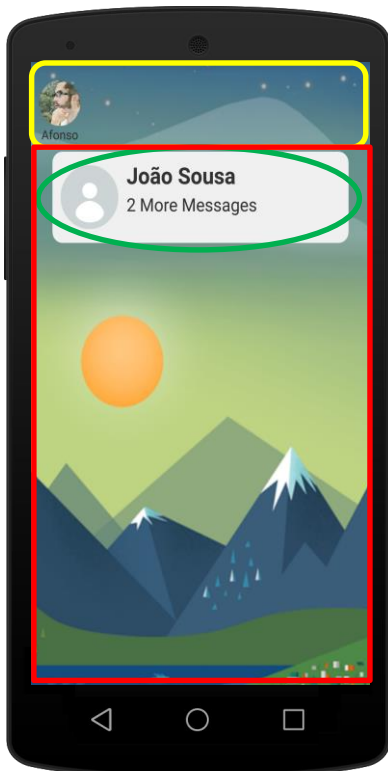


Figure 23 - Safety Driving Interface for Portrait Orientation

Yellow, is the smart dialer tab.

Red, is the notification center that will contain all the received messages grouped by sender.

Green is the notification corresponding to the messages received by a user.

If selects the notification item, then is showed a sub-menu to reply via message or call - Figure 24.



Figure 24 - Sub-menu for portrait orientation

5.4.3.1.1 Read unread messages

To read a message the driver only needs to click in the notification item. The message will be read in the loudspeaker.

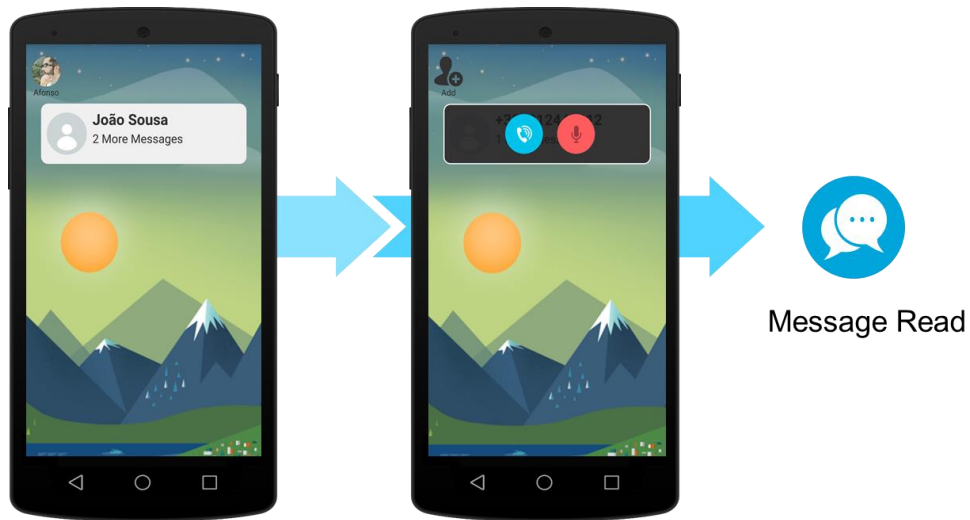


Figure 25 - Listen unread messages

5.4.3.1.2 See detailed notification

If the user wants to see/listen all the received messages, he/she can do that sliding the notification item from right to left. Will be displayed a big notification with all the required information.



Figure 26 - See detailed notification

In order to read all the received notification, the driver has to go to the detailed notification section and click in the notification.

5.4.3.2 Safety-Driving Interface for Landscape Orientation

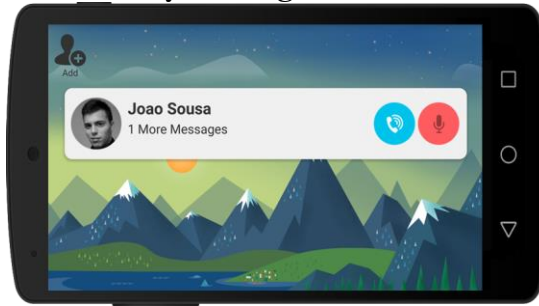


Figure 27 - Detail Notification in Landscape Orientation

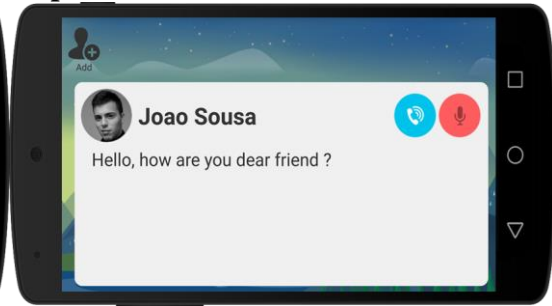



Figure 28 - Notification Center - Landscape Orientation

This layout behavior is the same as the previous – portrait orientation.

5.3.3.3. Reply via Voice Message

The driver can reply to a message clicking in the reply button - . The behavior in the landscape and portrait orientation is the same, so it is only be displayed the portrait.

The user can choose in the settings if he/she wants that the voice recording starts automatically.

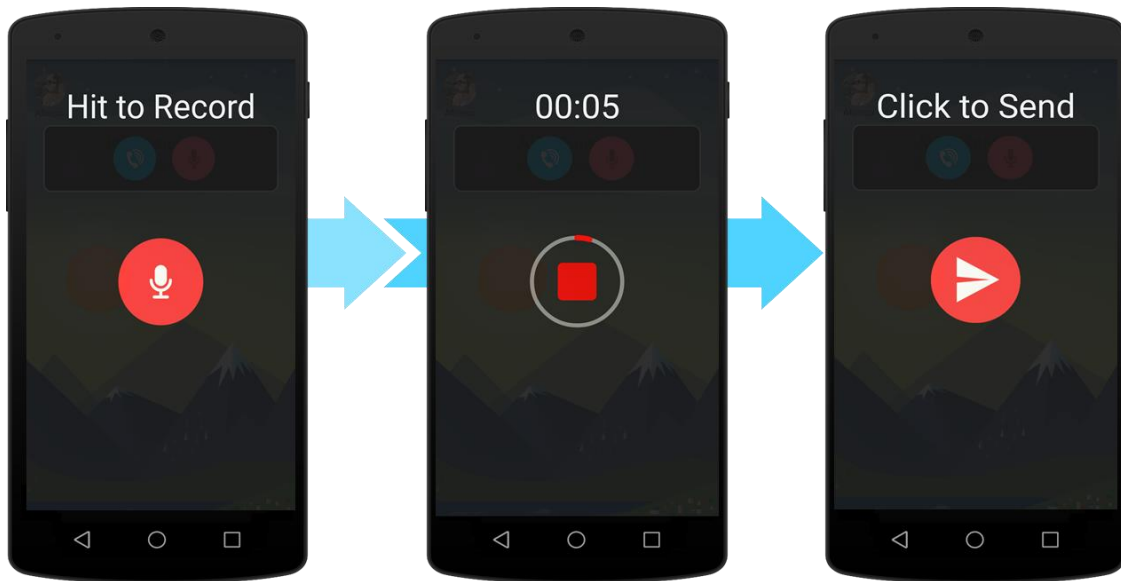



Figure 29 - Voice Recording Message

5.3.3.4. Reply via Text Message

The driver can reply to a message via text message clicking in the reply button - . The user has to set in the settings tab the instruction of that all message will be replied by text message.

The behavior in the landscape and portrait orientation is the same, so it is only be displayed the portrait.

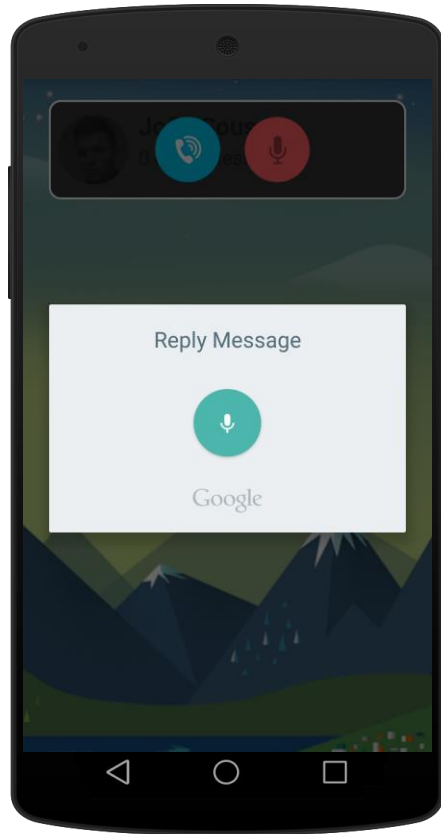


Figure 30 - Reply Via Text Message

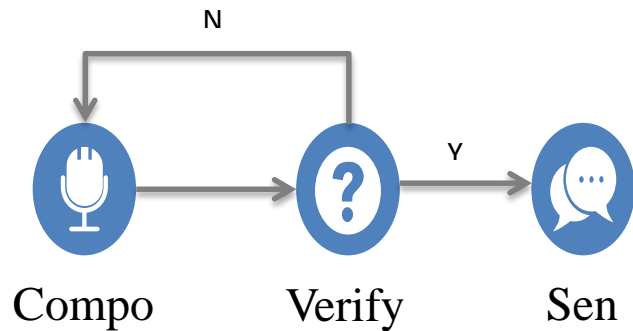



Figure 31 - Recording Method

The Figure 30 will trigger and start to record voice instructions. After the voice recording is started the system will detect when the user ends speaking, and then will ask if the message converted is the one that he/she wants to send. If it is send it; if not, record again and repeat the process. At the end the message will be sent according with the received message type. If is native will be sent as native, if it is a RCS will try to send via RCS and only if cannot connect will send via native – as can be easily understood in the Figure 31.

5.3.3.5. Reply by call

The driver can reply by call clicking the button - . The behavior in the landscape and portrait orientation is the same, so it is only be displayed the portrait.

The user can choose in the settings if he/she wants that the call is performed via RCS+ or Native Call.

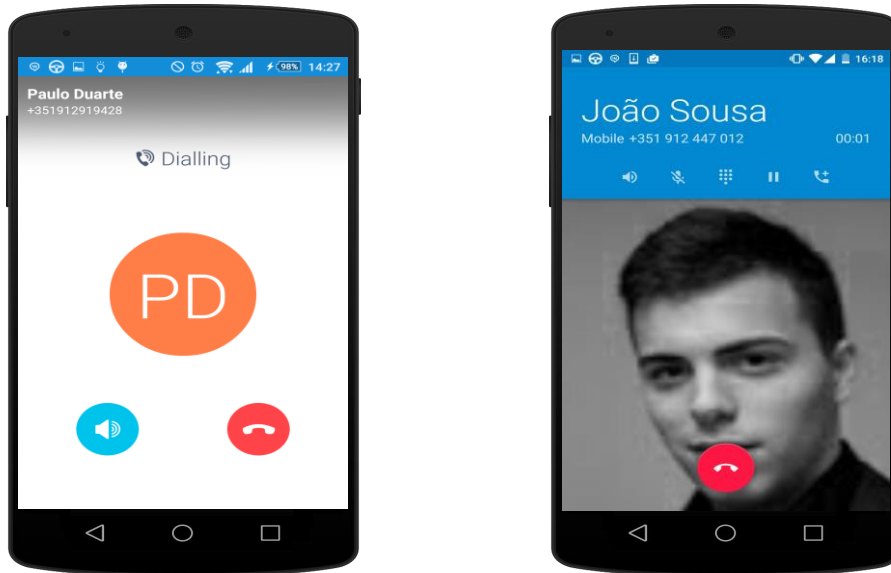


Figure 32 - Reply call

5.3.3.5. Dismiss Notifications

The driver can remove notifications from the notification center sliding the notification item from left-to-right, as the following figure shows.



Figure 33 - Dismiss Notification

5.3.4. Day/Night Filter

The Safety-Driving Interface provides an UI that adapts to the time of the day. If it is day, the background image colors will not be changed; if not, the colors will be darker.

This feature helps the user in the night, not forcing too much the eyes of the driver.

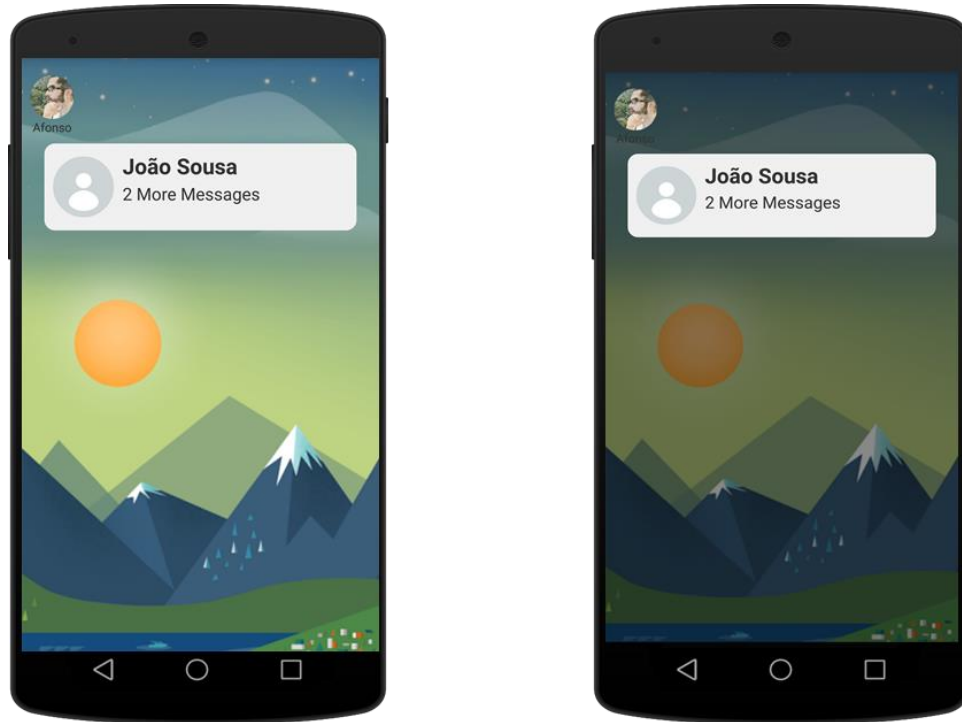


Figure 34 - Day/Night Filter

5.4.3 Safety-Driving Interface Architecture

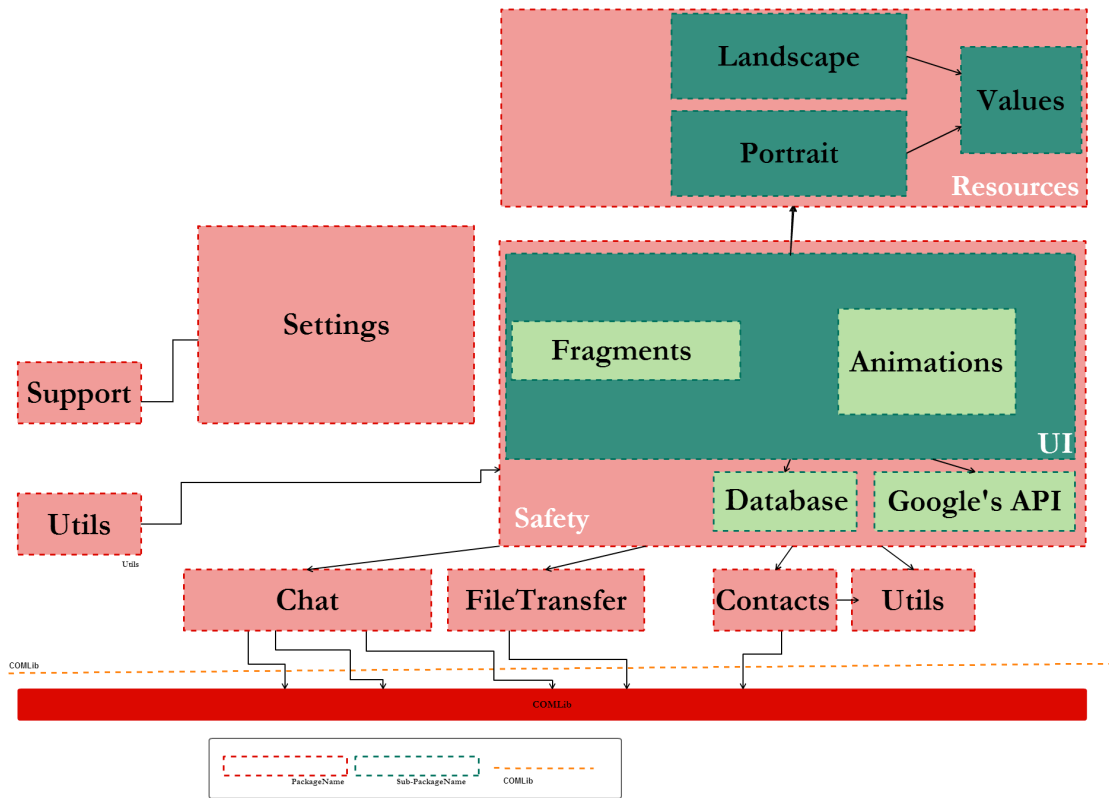


Figure 35- Safety-Driving Interface Architecture

The figure 35 shows on a resumed way the architecture that supports the Safety-Driving features. As the previous, the app is completely configurable by the user in the settings app. The settings structure is design in support module, is implemented in Settings and stored in Utils. The interface will retrieve information to the utils and change its behavior according with the settings.

The Safety as a logic part, and storage and the Google API's that will interact with the UI. The UI has fragments corresponding to each page, all fragments has a different resources according with the screen orientation. The fragments will have animations of swiping and removing in a unique component.

When the information is requested by the driver, the system will ask to the Chat, FileTransfer and Contact. The Chat component will be responsible to send text messages to groupchats or single with many types: SMS or Instant message; the FileTransfer will be responsible for recording the voice messages; finally the contacts will be responsible to start voice calls or to retrieve information about the contact list. Each of these components will interact directly with the respective COMLib API.

In this report, the analysis for confidential reasons, to a better understanding please consult the *Appendix B – Architecture*. If the image is not clear in the end of the document there are an image appendix with this diagram.

5.5 Challenges and Difficulties

The elaboration of this project made me grow and face different challenges. The project is divided in three parts: mobility detection algorithm, Android Auto, and the Safety Driving Interface.

The mobility detection algorithm was an interesting task to made, because it allow me to work with different Android technologies. My inexperience in Android development, allow me to learn almost everything about Android's background tasks and tools. It forced me to make architectonic decisions in order to develop a solution with a good performance. The integration with the RCS+ was not so difficult since were created an independent service to the mobility detection.

The Android Auto, allow me to have my first real interaction with the RCS+. In this task I had two main challenges: understand how the Android Auto works and how to integrate it with an existing product.

The second stage was the most difficult. The Android Auto works from the received notifications, for that reason was important to understand how the RCS+ notifications module works in order to display the received notifications in the car. The second stage was the collection of the information about the message, to do that was needed use the chat module. When the user replies to some message is needed to retrieve the message content from the Android Auto, the contact info from the contact module and the message type to send the message from chat module.

As can be concluded, the usage of this feature implies the usage of different modules of the RCS+. Those modules are complex and have to be changed carefully to not create bugs in the app.

Finally, the last stage implies, as said before, the usage of different modules of the RCS+. This last module was the most complex because it uses seven different modules. I am truly appreciated to the RCS+ Android's team for helping me in the most difficult times.

Chapter 6 – Quality Assurance

The Quality Assurance process is a fundamental on every software project. It is crucial that the final product is delivered without bugs, and with the best UI/UX possible in order to make them enjoy, and consequently use it.

To perform the evaluation several tests were design to validate the implemented features. All types of testing that took place in this internship are described in the following subsections. To see in more detail all the functional, non-functional and usability testing is highly recommended the reading of Appendix D – Evaluation.

This section will be divided in three testing subsections: functional, networking and OS testing, usability.

6.1 Functional Testing

Functional testing is a quality assurance (QA) process and a type of black box testing. These tests are based on the project specifications and analyze each component, testing it. These types of tests are made by analyzing the output, according with the input given. The internal program structure is rarely considered. Shortly, this tests represents *what* the system does, and check if everything is doing according with the expected.

As demonstrated in the appendix C – Requirements, the system is divided in three parts: mobility detection, android auto, and a safety-driving interface. According with that division the functional tests are also divided in those parts.

Were design and performed a total of one hundred and nine tests were performed to the RCS+ app. Nine of those tests were applied to the “Mobility Detection” mechanisms; Sixteen to the Android Auto extension; and eighty-four for the Safety-Driving Interface. The number of tests is sharper in the Safety-Driving Interface because of the importance and number of features that it has.

The number of tests were bigger than these, since the testing process has been continuous during the project implementation. Thus, is clear that the final version have to pass till all bugs are corrected.

	First Run		Final Run	
	Failed	Passed	Failed	Passed
Mobility Detection	1	7	0	8
Android Auto	2	17	0	19
Safety-Driving Interface	5	79	0	84

Table 9 - Testing Results Table

As can be seen, the testing session allows to discover some bugs/problems with features, which where iteratively solved until the final run, where none remained.

The complete set of functional are described in *Appendix E – Software Quality*.

6.2 Networking and OS Testing

Almost all implemented features rely on an internet connection in order to operate correctly. This way, it is imperative that these features can be responsive to those types of errors.

Although, is important to enhance that this application is going to run on mobile OS. This way, is important to guarantee that the system will behave the certain way when confronted with OS problems.

These types of tests are few than the previous, so were design seven test cases. Three of those tests were applied to the “Mobility Detection” mechanisms; one to the Android Auto extension; and three for the Safety-Driving Interface.

	First Run		Final Run	
	Failed	Passed	Failed	Passed
Mobility Detection	1	2	0	3
Android Auto	0	1	0	1
Safety-Driving Interface	0	3	0	3

Table 10 - Second Testing Results Table

As the results show, the only feature that had a problem was in the Mobility Detection module. That issue occur was related to the F2.1.1. For more information about this testing section and the testing results, please consult the *Appendix E- Quality Assurance*.

6.3 Usability Testing

Usability testing is a technique used in user-centered interaction design to evaluate a product by testing it on users. This can be seen as an irreplaceable usability practice, since it gives direct input on how real users use the system.

Usability testing focuses on measuring a human-made product's capacity to meet its intended purpose. In mobile applications this type of testing is really important, since the user has to understand how the system works, and how to use it instinctively. Usability testing measures the usability, or ease of use, of a specific object or set of objects. Whereas general human-computer interaction studies attempt to formulate universal principles.

This section will explain the evolution of the application according with user experience and feedback.

After the development phase, the application were evaluated and tested using a population of six individuals. With this public was possible to gather information, opinion about the features implemented in order to improve the application usability, if needed. On a second phase, was made an analysis about the usability testing according with the ten fundamental usability heuristics, named as 10 heuristics of Nielsen. This last stage helped me to

understand what need to be improved/changed. In the end were made an analysis of what were change changed.

These tests were made according with the final version of the product. Before, and during, the project development the design was discussed and was created according with the WIT's superior's feedback.

To perform the testing session was followed an approach a web usability consultant called Jakob Nielsen. He holds a Ph.D. in human-computer interaction from the Technical University of Denmark in Copenhagen. He said that "a usability test with 5 user will typically uncover 80% of the site-level usability problems". Knowing that, were invited 6 users that will test the system independently. [35]

To provide the invited testers a correct context of the project in order to develop a good testing process, was made a presentation, explaining the core purpose of the application. The application is not going to be used only by experts in technology, knowing that were invited different types of users: three users were extremely familiar with the RCS+, one already has used the app, and the other two do not know the app. The last two were users that had some difficulty with mobile devices. This two were really important to check if the system was really simple and complete.

Were defined a time to conclude each task. The time allocated vary according with the types of the task. Normally, each one lasts at most 20 seconds, although the sending messages tasks last at most 2 minutes. This approach was followed in order to guarantee that the application is easily understood, and since it was made to be used in a car, that requirement is absolutely crucial.

The testing process was managed by me, and recorded by a friend. Were performed 17 tasks as can be seen in the Table 10. In this report will only be discussed the results, to a better understanding of how and which tests were performed, please consult the *Appendix E -Quality Assurance*.

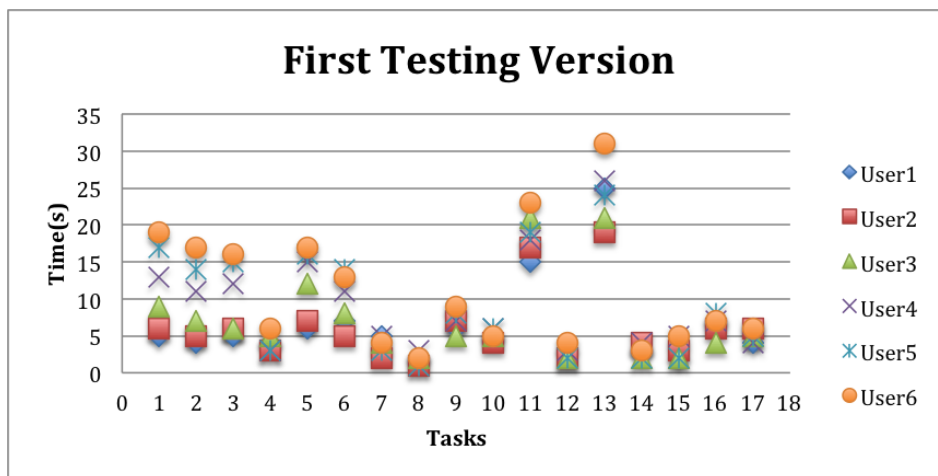


Table 11-First Testing Version Results

The results were quite encouraging and the time spent in each task was the expected. Sometimes take longer than expected; however that occurs because of the user experience with this types of applications, and not with this particular application.

The elaboration of these tasks was important to understand how the users interact with a possible final version of the application. The feedback of the testers was quite pleasant, however some opinions were a very good suggestion to improve user's usability.

After the tests were concluded, was made a team meeting in order to understand which changes must be applied to app. After that, the changes were well accepted by Eng. Filipe Santos and me. At least, the system will be tested again – with different people, to analyze the project evolution. In Table 11 is presented the second phase, and last version of the usability testing.

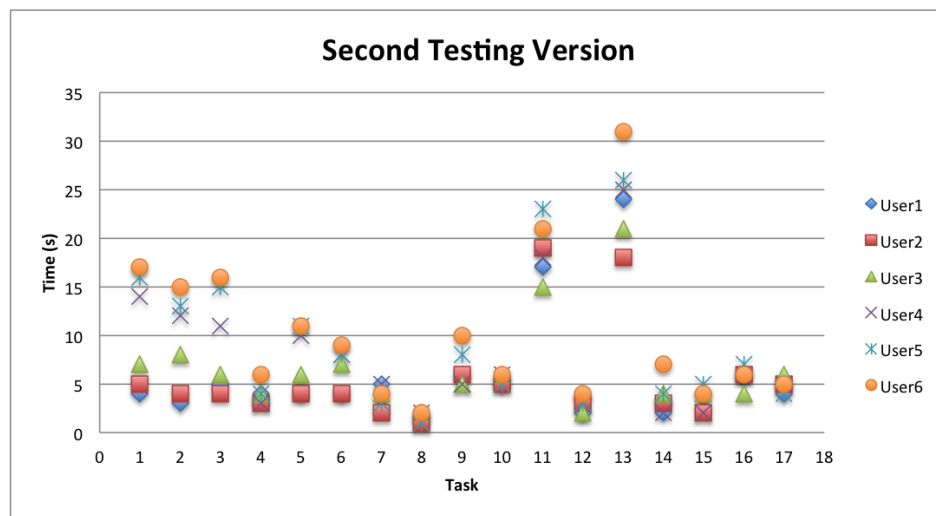


Table 12 - Second Testing Version

The final results show that the system behaves in a very similar way with two different target-groups; That proves that the system was well developed and focused on helping drivers without distracting him/her. The suggestion made in task 5, was the one that showed a performance increase, making the user experience with the RCS+ better, and the changes in the application were minimal.

At the end, the application were analyzed according with Nielsen's 'heuristics'. He said: "The 10 most general principles for interaction design. They are called 'heuristics' because they are more in the nature of rules of thumb than specific usability guidelines." [36]

This section will relate our project with the 10 Heuristics of Nielsen, evaluating our application with the Nielsen principles. The possible values will be a range between 0 and 10, being 10 the value that expresses no problem using the application.

I. Visibility of system status

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

Rating: 8/10

Review: The system has a mechanism that will always saying to the user which steps he/she is doing. This way the user does not need to be looking to the application while it's driving.

II. Match between system and the real world

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

Rating: 9/10

Review: The system provides to each driver a simple and intuitive interface. Uses also a language that is familiar to the driver.

III. User control and freedom

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Supports undo and redo.

Rating: 9/10

Review: The system provides always a way back at each interaction. This way if the user clicks by mistake on a button, he/she can always redo to the main screen.

IV. Consistency and standards

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Rating: 10/10

Review: The main goal of the application is to provide safety while the user is driving, thus the system has to provide a layout that is simple, and provide unique meaning to the user.

V. Error prevention

Even better than good error messages is a careful design, which prevents a problem from occurring in the first place. Either eliminates error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

Rating: 8/10

Review: The system has a layout specially focused in the final user. Each icon only produces a unique functionality. If the driver selects a wrong icon, the system will perform that operation, however the user can cancel anytime that action.

VI. Recognition rather than recall

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Rating: 8/10

Review: The main purpose of the application is to provide a simple interface, and that is being developed in order to provide an easy interaction with the driver. That way, the user's interactions with the app are independent, and the user almost never needs to know the previous state.

VII. Flexibility and efficiency of use

Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

Rating: 8/10

Review: The application was made in order to provide to all types of users, experienced and inexperienced, an easy understanding of the application. The expert users will interact with the application quickly, however the system is made to be easily understood to all types of users.

VIII. Aesthetic and minimalist design

Dialogues should not contain information that is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

Rating: 8/10

Review: In the application, almost all settings are predefined in the RCS+ settings; dialogs do not appear in the application, only in specific situations. The system has a simple layout that provides an easily understanding of how the application works

IX. Help users recognize, diagnose, and recover from errors

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

Rating: 8/10

Review: The application provides to the user a voice mechanism to tell the user what are possible errors, and helps the user to understand the situation

X. Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

Rating: 5/10

Review: Since the system was made to be used when some user is driving. The interface has to be simple, and provide few, as possible, information. That way, documentation and helping guides is not visible. However, all interactions have the support of a voice mechanism that will help the driver to understand what he/she is doing.

Chapter 7 – Overview of the Project

This chapter represent the work produced during in the internship. The internship were divided in two phases – first and second semester. For that reason this section will be divided in two phases too.

In each phase will be represented what was done, how the project flows and the changes that were made according with the planning. The changes were easy to handle due the agile work methodology used.

7.1. Overview of the First Semester

This section will be divided in three parts.

First, will be resumes the planning for the first semester, that has the principal goal of understand the problem and plan a solution.

During first semester I had the objective of create an algorithm that will detect the user's mobility – this is the second part.

Finally, the last part will show the design plan for the second semester – in section 7.2 will be discussed what have change and the project evolution in the second semester.

7.1.1. Goal

The first semester was very good for the analyses of the competitors and the technologies that already existed. This analysis was made with a study of the State of the Art. This stage was of great importance because it allowed me to understand in which models the project would fit.

Thereafter in order to understand what had to be done, was made a study of existing technologies and which technologies where used by the WIT-Software. Thus, after a meeting with my Scrum Master, we decided that must elaborated a mobility detection prototype with many methods testing it thereafter, to reach, at the end of the semester, a conclusion of the best solution. In parallel with prototyping a requirements list was elaborated and the Product Backlog was generated.

At the end of the first semester was well design the needs for the elaboration of the final solution. I have the features detailed and ready to be implemented – as will be explained in the next section.

This first semester was quite rewarding. Although, having worked in this area before, I had not lived the intensity of demand as experienced here. These challenges empower me and I am motivated to achieve my goals.

It's enriching to see how a real project evolves and to adapt my workflow to the company's work methodology.

Good integration with the WIT's team of RCS Android allows me to learn about technologies that I had not used before. I am very pleased with them and extremely motivated to face the work coming in the next stage of the internship.

In the Figure 36 is illustrated more precisely what was made in the first stage of the internship.

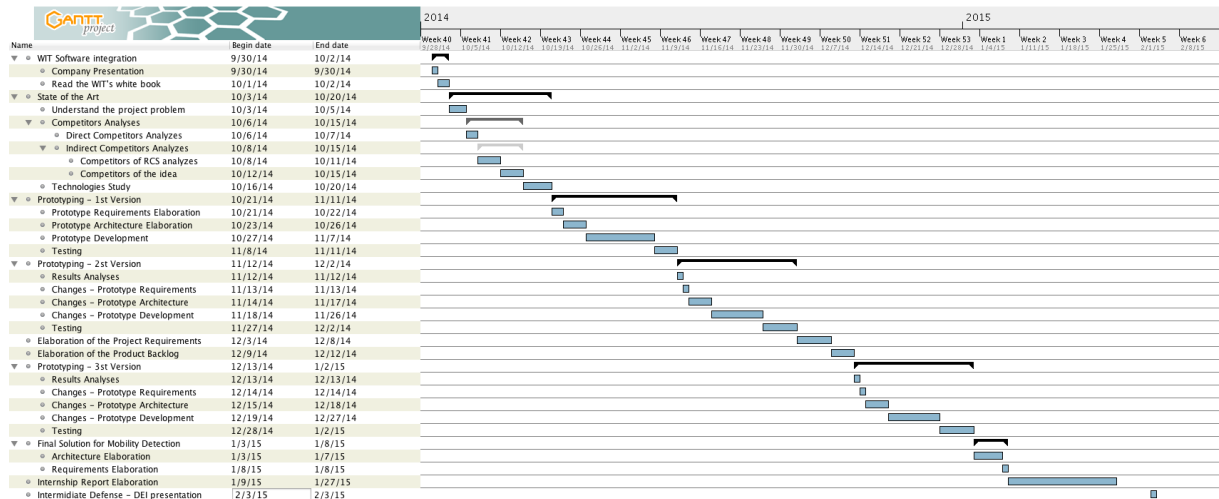


Figure 36 - 1st Semester Overview

This Gantt was made to show the work that was executed in the first stage of the internship. In order to explain this on the best way possible the tasks done were divided in subtasks.

Task 1 – WIT Software Integration

In the first day all trainees were integrated with a tour for the company. After that, was given a presentation about the story and the goals of WIT Software. Then was given to us a white book in order to study and understand the company's work methods and what should be respected in the installations.

Task 2 – State of the Art

2.1 – Understand the project problem

In a beginning of the work I had a meeting with my supervisor, Eng. Filipe Santos, in order to get a better understanding of what were the purpose of the project, and what WIT's expects in the end of the internship.

2.2 – Competitors Analyzes

After knowing what the purpose of the project was, I have to analyze the competitors that already exists in the market. This analyzes was divided in two phases: first understand the strengths/weakness of the applications that already exists in the market and will concur with the WIT's product.

Then was analyzed other two types of applications, the ones that support mechanisms of safety driving but are not specific applications and those that are competitors of the RCS app but does not contain safe mechanisms.

2.3 – Technologies Study

After understand what features I had to implement, was necessary to study the technologies that the application will be used. In my case, I do not have experience in Android development and was need a time to adapt to the frameworks.

2.4 Prototype Versions

In the first semester the goal was determine a final solution for the mobility detection, for that reason were elaborated three versions of prototyping. The differences each one are explained in section 3.

2.5 Elaboration of Project Requirements and Product Backlog

When final solutions were defined was created the requirements of the solution and the Product Backlog. The explanation of what is the Product Backlog can be analyzed in appendix D – Approach.

2.6 Final Solution for Mobility Detection

After all the tests, the architecture of a mobility detection module solid was made. This will be developed in the second stage.

7.1.2. Detection of Mobility

At the beginning of the semester I had a meeting with the Mr. Filipe Santos in order to establish what I was supposed to do on the detection mode. After the meeting it was agreed that I was going to make a prototype that would include several detection algorithms, and after testing various algorithms will be chosen one that can be combined by following.

The following algorithms have been made:

- **GPS data** – GPS data was retrieved, and with that information, paths were created. With the paths information and the temporal information of the user, distance and velocity were calculated. Once these were established, if the velocity was above a certain limit the system would assume that the user is driving.
- **GSM information** – One advantage of GSM detection compared with the GPS is that this method saves more battery. Otherwise it has the disadvantage that location is less precise.
- **Wi-Fi information** – When the phone is connected to a Wi-Fi network, the location can be retrieved using this method, thus being more precise than GSM, and saving more battery than GPS detection. Otherwise, is limited because a user isn't always connected to a network.
- **Google Maps API** – This API gives the same information that GPS data provides, but this method will require access to the 3G data. This method is more precise than preceding method.
- **Accelerometer Sensor** – One aspect that it is not easy to distinguish is whether the user is driving or just in the car. That way the accelerometer sensor will detect if the user is using the phone. If the phone is being used then the system will assume that he is a passenger.
- **Road detection** – Was made a method that distinguished if the user was driving or on a train. In most cases the trains don't circulate in streets, for this reason I implemented a method that detects whether the user is on a road or not.
- **Bluetooth devices detection** – The essence of the application is to help drivers, if connected to a Bluetooth device, to listen messages received while driving.

- **Handset devices detection** – As in the previous method if the user is using another handset device, he will be capable of reading messages received. These two methods also check if the user has the phone in car mode.

After this prototype was developed testing was carried out by some of the company's employees. The objective of this phase was to detect which methods were relevant. After days of tests I came to the conclusion that these methods were very good for mobility detection, but they were a major drain on battery autonomy. The usage of GPS data was so incredibly high, that it exhausted the battery.

After obtaining these results I had a meeting with Mr Filipe Santos and discussed this issue. The detection was perfect, but inefficient.

For this reason a new method was added to the algorithm, that was called Geofencing[37]. If a user stays more than 4 minutes in the same place, that place will be called a geofence, after that the Android API will notify when the user enters/exit one such area.

This method allows the system to turn off access to GPS data when the user is in a geofence. After these changes the system was much more stable.

When this change was made, the prototype was distributed in the team of Android RCS and some friends of mine. We added one more method, called activity recognition[38] to this method. This method was meant to catalog the activity of the user, and once the data was received by the people testing it, an analysis was conducted to see which methods were useful.

After a week of tests, the amount of data received was vast and a pattern emerges, the GPS triangulation works just fine, but still spends a considerable amount of battery. At this point the data collected by the activity recognition was analyzed and found to be extremely accurate. Thus was made a prototype which did not have the GPS tracker. Performance was similar to GPS. For this reason I chose to delete that algorithm of detection, and decided on:

- **Activity Recognition** – This method will return the state of the user at the moment and will have a probability assigned. The possible states of the user are: in a vehicle, on a bicycle, on foot or standing still.
- **Geofencing** - This method will detect if a user is entering a area, and will stay there for some time. With this feature the system doesn't need to have the activity monitor always running.
- **Accelerometer Sensor** - This method is really important in distinguishing between users driving or simply passengers in the car.
- **Bluetooth and Handset devices detection** – The goal of the project is for users to listen to messages while driving. If they have one of these devices, then the message can be read from the device.

From the list above it can be seen that Google Maps API was left out, this happened as a result of data collected not justifying the amount battery usage.

The goal of the first semester was to find and test the best way of detecting mobility of users. With the data gathered I concluded that this system is comprehensive.

7.1.3. Future Planning

At the end of the first semester was decided what should be done in the next phase. Was made a not detailed planning of what should be done. It was divided in the following phases:

- Creation of the final solution for mobility detection.
- Integration with RCS+.
- Text-to-Speech mechanism.
- Sending voice-message.
- Elaboration of the DriveScore.
- Changes in the UI of the RCS+.
- Notify the user of the messages that are on hold.

It is important to notice that I will be ready any new features that may arise. As stated in Scrum, there must be flexibility in a software project and I will adapt to that.

Final solution for mobility detection

As explained at section 6.2, after all the tests were concluded the prototype has been finished. In this second phase of the internship a final solution has to be arrived at and will be integrated in the RCS app.

Integration with RCS App

After the solution is ready to join with RCS app, that step is going to be taken. Once this is in place the RCS app will be capable of detecting the mobility.

Text-to-Speech mechanism

After the integration elaborated another feature needs to be added to the RCS app. When a message is received that message must be read to the driver. This will be done using several steps to ensure it does not interfere with the privacy of the user.

Sending voice-message

If the driver wishes to answer a message, he/she can do it sending a short voice message.

Elaboration of the DriveScore

The app will include a DriveScore mechanism, which consists of giving points in accordance with the driver's behavior. If driver uses the mechanisms of safe driving and not the phone, then he/she will receive points. If the driver chooses to use the phone while driving he/she will obtain a bad score.

Changes in the UI of the RCS app

In order to add new features, as listed above, the application interface must be changed.

Notify the user of the messages that are on hold.

When the driver stops the vehicle, all the messages received while he was in safe mode he/she will be notified.

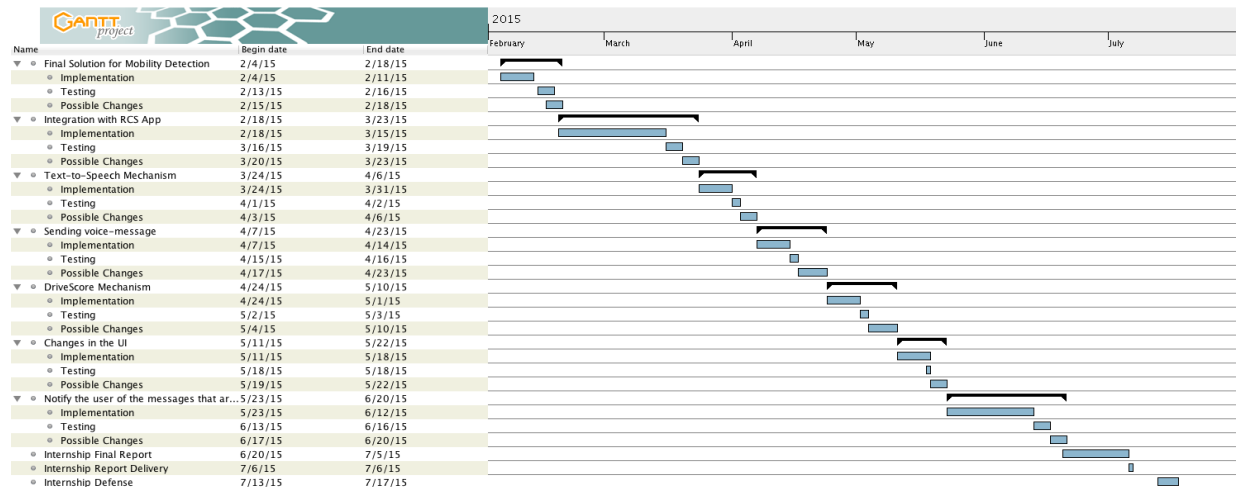


Figure 37 - 2nd Semester work plan

In Figure 37 is illustrated the work plan for the second semester, however since I am following an agile methodology the organization may, and will, change.

The work plan is divided in seven parts. Each part has three sub-tasks:

- 1 – The feature will be implemented
- 2 – The feature will be tested in order to find if exists any bug or issues
- 3 – If occurs any bug the feature will be fix it.

The next section will discuss the work performed in the second semester, and the changes that occur in the original plan.

7.2 Overview of the Second Semester

The second semester had the purpose of create the features that will provide a safer driving to each RCS+ users.

In this chapter will be explained the work performed during the second phase of the internship. It occur some deviations of the original plan. Those changes will be discussed in the next section 7.3.

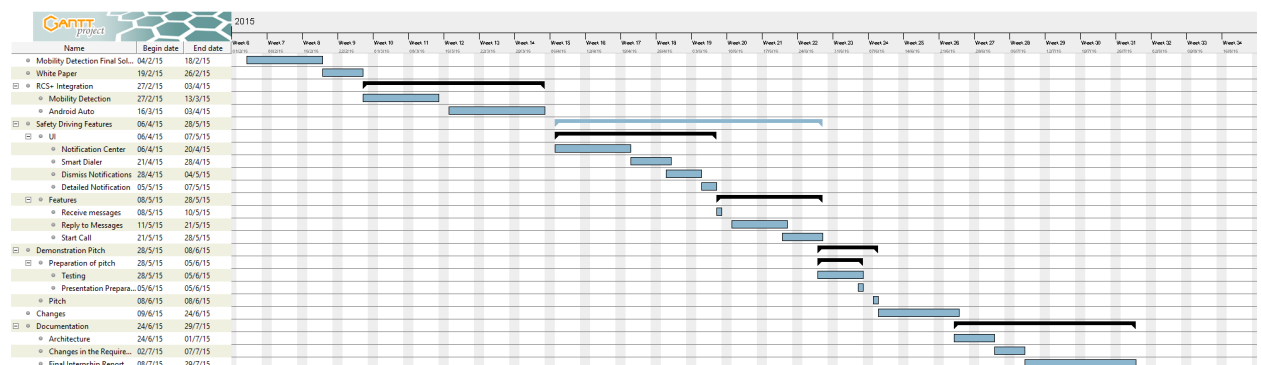


Figure 38- 2nd Semester work phases

The Figure 38 is not completely understandable, to read with more precision please consult the document appendixes.

Next, will be discussed what was performed in each task.

Task #1 – [04/02 , 18/02]

The first task, which lasted two weeks, was the creation of the final solution for mobility detection. This task starts in February 4th and ends in February 15th.

This task was completed with very success. The results of the first semester helped me to develop a final solution with high accuracy and efficiency.

Task#2 – [19/02, 26/02]

After task#1 was completed, it was requested to be done a scientific paper to be delivered in the Portuguese Government in order to explain what the purpose of the project was.

The paper is divided in three parts:

1. A brief background to the existing problem – driving and sending messages at the same time.
2. Explanation of the existing applications/competitors in the market. Were explained the AT&T DriveMode and the Android Auto.
3. Final Solution
 - a. Detailed explanation of the mobility detection algorithm.
 - b. Summary of the safety features that must be implemented.

After this paper was delivered, I and Eng. Filipe Santos had a meeting to discuss the next task. Since I am following an Agile work methodology, at the end of each sprint is planned the next.

The following sprint has the purpose of integrate the mobility detection in the RCS+.

Task#3 – [27/02, 13/03]

The task#3 has the purpose of integrate the mobility detection with the RCS+. This process is explained with more detail in the *Appendix B – Architecture*.

After this sprint were made another meeting, was discussed the December release of the Android Auto, and the opportunity to integrate this feature with the RCS+.

Task#4 – [14/03, 03/04]

Was discussed in the meeting that the Android Auto feature was an interesting feature to be added to the RCS+. The expansion of the app to the car radio was approved by the Scrum Master and the Project Owner.

This task was the main deviation comparing with the first planning, and some decisions has to be made. In the next subsection all will be explained.

Task#5 – [04/04, 21/05]

After developing the Android Auto solution was time to think about the Safety-Driving Interface. Android Auto has an inconvenient that is only available to Android's devices 5.0.

In April, stats says that only 5.4% of the devices has Android's 5.0.[39] For that reason was important that were given to all RCS+ the same type of system. As noticed previously, the Android Auto is limited by Google – in order to provide a controlled environment. So, since the UI/UX of the Android Auto was very good – and tested by Google, so the team decide that was interesting that the created features had some similarity with Android Auto.

This task is divided in two sub-task, which are divided on other few tasks. The division of this big task was made in order to control the evolution of the project, and not reach a state that the system was uncontrolled and/or out of time.

#Task5.1 [06/04, 07/05]

The layout had to be easy to understand and use. For that reason were created different layouts according with the smartphone orientation (landscape and portrait).

This task create the layout for the respective features:

- Notification Center – [06/04, 20/04]
- Detailed Notification – [05/05, 07/05]
- Dismiss Notifications – [28/04, 4/05]
- Smart Dialer – [21/4, 28/4]

#Task5.2 [08/05, 21/05]

After the layout was created to both screens orientation, were developed the features that will provide the UI the right information and way of communicate.

So this task has the purpose of:

Receive Messages [08/5, 10/5] - Filter the received messages, and displayed in the UI. Allows the user to read messages as well.

Reply to Messages [11/5, 21/5] – The system allows the driver to reply via voice message or text message. The message will be sent using two types of messages (IM or SMS).

Start Call [21/5, 28/5] – The Safety Driving Interface will allow the driver to reply a message by voice call.

Task#6 – [28/05, 08/06]

This task was made on an advanced phase of the project. WIT's summoned a meeting with all the interns, in order to evaluate the created system and to give suggestions about thinks that should be improved.

This meeting was organized by WIT's CTO, Eng. Pedro Pereira. To prepare the pitch was dedicated a week to test all the developed functionalities and to fix some bugs. Was created a 10 minutes presentation, and were more 10 minutes to answer to some questions.

The meeting went well, and some aspects of the application was discussed. Namely two thinks that were changed in the product:

1. Was of common opinion that the app should not open automatically after the mobility detector detects a *driving* state. The app should show a pop-up that will allow the user to enter in a safer mode.
2. The app only works in one way – that is the purpose. However the suggestion of creates a Smart Dialer Tab was really amazing. That way the driver can have a VIP contact list in the app, and call for them without having to go to the contact list.

Task#7 – [09/06, 24/06]

Were spent two week to make the purposed changes, and to validate the complete system.

Task#8 – [24/06, 29/07]

The last task was important to create all the project documentation.

7.3 Changes in the Project

The initial plan to the second semester was to develop the tasks discussed in the section 7.1.3. Initially the Android Auto was a future opportunity because was not stable at the time that the planning was made.

After the mobility detection changes and its integration in the RCS+, was analyzed how would be the next task. In the team meeting was discussed that the Android Auto was already stable and released. It was a great opportunity to innovate and to beat our competitors. That way, together we choose that the best approach was to eliminate the DriveScore feature and replace it with the Android Auto.

Now that the project is terminated, we can conclude that this was the best decision. After the Android Auto was developed, was time to start developing the safety features in the RCS+.

The Android Auto has restricted functionality in order to guarantee drivers safety. In the meeting done at the end of the task#4, was planned the best way of integrate the requirements. The Android Auto UI/UX was tested and developed by Google, so we decide that it was interesting to take advantage of that and build a system that has a similar UI/UX.

This system will help the RCS+ users that do not have an Android 5.0. This solution has not the limitations of the Android Auto, so can be developed so many more features.

At the end of the project development the feedback was very good. Our final solution will integrate three different systems: Android's native service to create an optimized mobility detection algorithm; an extension of the RCS+ to the car's dashboard; finally, create a new user interface in the RCS+.

Chapter 8 – Conclusion and Future Thoughts

This final chapter marks the end of this report and the end of the internship at WIT Software.

This document will be divided in three parts.

The first will present an overview of all the work done during the internship.

The second, explain the work that can still be done in the future, always with the purpose of improve the RCS+ and provide a safer environment to each driver.

Finally, will be present some personal thoughts about the internship.

8.1 Overview

The entire internship has the purpose of providing a safe driving situation for each driver that uses RCS+. People need to communicate, is something that it's completely essential. However, while driving its difficult, and dangerous, to perform such activity.

That way WIT-Software wanted to create a tool that provide that safer environment to each driver. That tools will be added to the company's product - RCS+. That way, can be notice an improvement in road safety and in application MNO.

The internship's result is focus in three points that were design to well serve the driver.

1. Mobility Detection method, which will alert the driver to initiate the safety driving mode.
2. Android Auto that will be integrated with the smartphone and the car.
3. Safety-Driving Interface that will serve the users that do not have neither an Android 5.0 nor a car with Android radio.

Mobility Detection

This initial idea of this method is that it will track the user's behavior, and behave according with his/her state. If he/she is driving the system will help him/her. Although the main idea is still present in the final product, were some changes that were performed.

This method will only alert the user for his driving status. If he/she is driving, the system will detect that he/she is driving and create an Android's notification, which will inform the driver of the possibility of access the Safety-Driving Interface.

This method is created to remind the driver that he/she has a "friend" ready to help him/her.

Android Auto

This feature is, maybe, the most important feature developed during the internship. This feature was not present in the first requirement list, but has been referred in the State of the Art section.

Android Auto emerges as an opportunity to put together the RCS+ and the vehicle. The best way to help the drivers is to be integrated on an environment that he/she familiar with. This integration allows the driver to send and receive messages in the car radio and reply it by voice command.

In my opinion, was a well elaborated strategic move to include the Android Auto feature in the RCS+ - even that DriveScore functionality has lost value. There was only time to implement one feature, and we (I, and Eng. Filipe Santos and Eng. Rui Gil) chose the Android Auto.

Safety-Driving Interface

The final module was the implementation of a Safety-Driving Interface. One problem that Android Auto has is that it is restricted to Android 5.0 devices, and user that has a car with Android OS. The principal purpose of the application is to help the drivers, initially was predicted that the system will be done completely by voice command. However, with the integration of the Android Auto, was thought that perhaps was interesting if the Safety-Driving module was a complement to the users that do not have Android Auto support.

After a meeting with the team was emerged the idea of joining the initial ideas with the Android Auto features. That way can be added to the existing features, features as:

- Start calls
- Smart dialer for VIP contacts
- Reply to messages by message or voice
- Listen unread messages
- Listen all messages received by some user
- See all messages received, grouped by user

This features will really enrich the RCS+ application and the driver that needs to communicate.

This Safety-Driving Interface has an interface quite inspiring in the Android Auto in order to provide an UX similar to all RCS+ users. That way if the user that usually uses the Safety-Driving Interface, start to use the Android Auto, he/she will not find it difficult to fit in the new tool.

I am delighted with the final product, and after the stages of testing and discussion I think that the product was well design and elaborated.

8.2 Future Work

I am very happy with the final solution achieved, however there are always margin for progression. In the beginning of the internship was proposed to perform some requirements that lost priority because the new, and more urgent, requirement that have appear.

There are some suggestions that I want to tell about new features that may be interesting in the evolution of the product. First a user evaluation; Second add a social component to the app; And third, an expansion on the Android Auto feature.

DriveScore

An elaboration of a DriveScore mechanism to evaluate the driver's behavior. If the driver receives/send messages or make calls with the safety mechanisms will have his/her score improved. Otherwise, will be decreased.

Facebook Integration

What if the driver's DriveScore was compared with the facebook friends. With that, maybe the driver's will think better about using the smartphone while driving. The facebook recently changes his policy restricting access to its API. However that will not affect this feature since all the communication will be done in the RCS+ and with only RCS+ users.

Android Auto

The Android Auto support is recent, and for that reason the number of functionalities that support it is reduced. In the future the support will be bigger, and will be possible to improve the Android Auto experience.

This new mechanisms helps the drivers, but have a potential value that will increase the value of the RCS+ app.

8.3 Final Remarks

To conclude this report, is time to announce some personal thoughts and feeling about this internship experience.

This internship was a whole new experience, I've already have been working in some companies, but none with the challenge and demand that WIT-Software. The company give me the privilege be one more in the RCS team and give the opportunity of suggest and contribute to the RCS+ for Android evolution. Many of the features were thought by me and then discussed in team. That reveals that the company listen to everybody and provides all tools to its collaborators to grow, including the interns.

The initial plan for the internship has undergone some changes during the time, as discussed in the report. That situation allow me to grow as a potential Engineer, and to work on different fronts. The internship has three different components.

The first has a component of algorithm, and routines analysis.

The second has a strong component of integration of two different systems – Android devices with car dashboard via Android Auto.

Finally, the third – and the hardest part, has a component that allow me to create a new set of features to be added to the RCS+ app. This features will provide a new UI and UX, different of the one's that the RCS+ users are used to.

In the beginning of the internship the CTO – Eng. Pedro Pereira said to me that if my internship product save one single life, every hour spent in the elaboration of this project had been worth. Now that the product is finished I am exciting in providing to all RCS+ users a safer way of communicate.

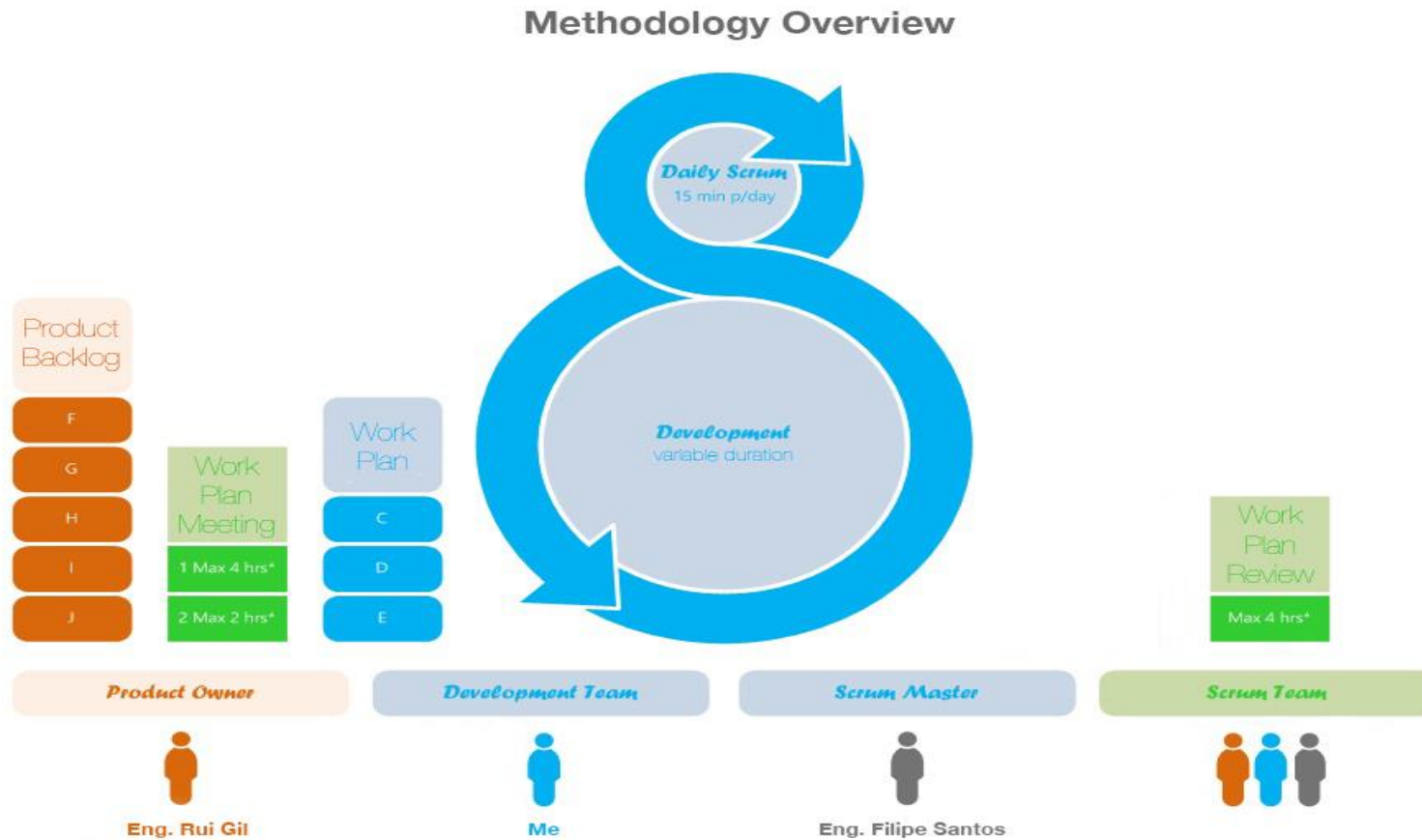
In the end, I am very happy for choosing this project and this company to perform this internship. Every difficulty was worthwhile in the end. I feel, now that I am a better person, and a better professional. I would like to thanks to all Android's RCS+ team member and to Eng. Filipe Santos for all the support.

References

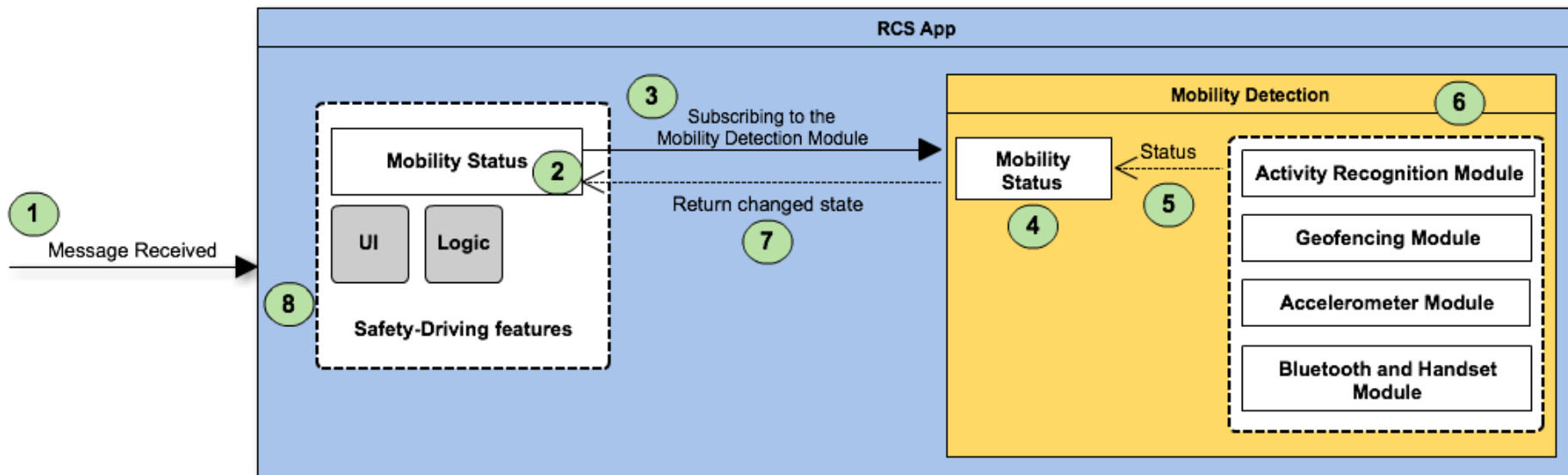
1. Cheng, C. *Why Scrum ? Why Agile Development*. 2013 [cited 2015; Available from: <http://calvinx.com/2014/05/22/why-scrum-why-agile-development/>].
2. Project, P.I. *Mobile Technology Fact Sheet*. 2014; Available from: <http://www.pewinternet.org/fact-sheets/mobile-technology-fact-sheet/>.
3. Gannes, L. *Survey: 35 Percent of Smartphone Owners Use Them While Driving*. 2013; Available from: <http://allthingsd.com/20130114/survey-35-percent-of-smartphone-owners-use-them-while-driving/>.
4. Centers, P.-T. *Facts About Texting & Driving*. 2011; Available from: <http://www.donttextdrive.com/statistics/>.
5. Mobithinking. *Global mobile statistics 2012 Part C: Mobile marketing, advertising and messaging*. 2012; Available from: <http://mobiforge.com/research-analysis/global-mobile-statistics-2012-part-c-mobile-marketing-advertising-and-messaging>.
6. GSMA. *Rich Communications*. Available from: <http://www.gsma.com/network2020/rcs/>.
7. Godinez, V.a.D.M. "AT&T moving headquarters to Dallas from San Antonio." *The Dallas Morning News*. 2008; Available from: http://www.dallasnews.com/sharedcontent/dws/bus/stories/DN-att_28bus.ART.State.Edition2.4d5475b.html.
8. Inc, G. *Drive Mode Description*. 2013; Available from: <https://play.google.com/store/apps/details?id=com.glasscube.drivemodepro>.
9. Play, G. *Safely Go*. 2013; Available from: <https://play.google.com/store/apps/details?id=com.safely.go.driver.safety.stop.texting.driving>.
10. Play, G. *text-STAR*. 2013; Available from: <https://play.google.com/store/apps/details?id=com.cinqpoint.textstar>.
11. Henry, A. *The Best Virtual Assistant for Android*. 2013; Available from: <http://lifehacker.com/5883560/the-best-virtual-assistant-for-android>.
12. Wollman, D. *Samsung announces SmartStay and S Voice features for the Galaxy S III*. 2012 [cited 2014; Available from: <http://www.engadget.com/2012/05/03/samsung-s-voice-smartstay-galaxy-siii/>].
13. Lunden, I. *Samsung: The Death Of The Spec Gives Way To The Birth Of The Human Touch*. 2012; Available from: <http://techcrunch.com/2012/05/03/samsung-the-death-of-the-spec-gives-way-to-the-birth-of-the-human-touch/>.
14. Bryan. *Samsung's New S-Voice Is Just A Skinned Version Of Vlingo Labs Beta on Google Play*. 2012; Available from: <http://www.gizmodfusion.com/2012/05/samsungs-new-s-voice-is-just-a-skinned-version-of-vlingo-labs-beta-on-google-play/>.
15. wikipedia. *Nuance Communications*. Available from: http://en.wikipedia.org/wiki/Nuance_Communications.
16. Tsukayama, H. *Facebook Messenger app change allows free calls via WiFi*. 2013.
17. Zuckerberg, M. *Facebook*. Available from: <http://www.facebook.com>.
18. BBC. *Facebook new Messenger service reaches 500 million users*. 2014 [cited 2014 2014-11-02]; Available from: <http://www.bbc.com/news/technology-29999776>.
19. Fiorella, S. *The Insidiousness of Facebook Messenger's Android Mobile App Permissions*. 2013 2014-10-30]; Available from: http://www.huffingtonpost.com/sam-fiorella/the-insidiousness-of-face_b_4365645.html.
20. TimesOfIndia. *Facebook buys WhatsApp: CEO Mark Zuckerberg explains why*. 2014 [cited 2014; Available from: <http://timesofindia.indiatimes.com/tech/tech-news/Facebook-buys-WhatsApp-CEO-Mark-Zuckerberg-explains-why/articleshow/30714548.cms>].

21. Anorak. *Skype Is Now 40% Of The Entire International Telephone Market*. 2014; Available from: <http://www.anorak.co.uk/383464/money/skype-is-now-40-of-the-entire-international-telephone-market.html/>.
22. App, V.U.D. *Viber Unveils Desktop App*. 2013 [cited 2014 2014-11-10]; Available from: <http://blogs.wsj.com/digits/2013/05/07/viber-unveils-desktop-app/>.
23. Milward, S. *After Rakuten Acquisition, Viber reveals it has 100 million active users*. 2014; Available from: <http://www.techinasia.com/rakuten-acquisition-reveals-viber-has-100-million-active-users/>.
24. Acapella. *Acapella Explanation*. 2014 24-08-2015; Available from: <http://www.acapela-group.com/acapela-for-android/>.
25. Nuance. *Nuance Developers*. 2015; Available from: <http://dragonmobile.nuancemobiledeveloper.com/public/index.php?task=memberServices>.
26. Google. *Google APIs for Android*. 2015 [cited 2015 24-08-2015]; Available from: <https://developers.google.com/android/reference/packages>.
27. Google, *Google Play Services Game*. 2014.
28. Google, *Google Developers - Addign Saved Games to your Android Game*. 2015.
29. Google, *Location API - Making your App Location-Aware*. 2014.
30. Google, *Google Maps API - Adding Maps*. 2014.
31. Google, *Android Auto*. 2014.
32. Google, *Google Now - Cards* 2014.
33. Google. *Google Now - Emails*. 2014; Available from: <https://developers.google.com/schemas/gmail/actions?hl=pt-PT>.
34. Guide, S. *Scrum Guide*. 2013; Available from: <http://scrumguides.org>.
35. Group, N.N. *Jakob Nielsen*. 1995 25-08-2015]; Available from: <http://www.nngroup.com/people/jakob-nielsen/>.
36. Group, N.N. *10 Usaability Heuristics for User Interface Design*. 1995; Available from: <http://www.nngroup.com/articles/ten-usability-heuristics/>.
37. Google. *Geofencing*. 2014; Available from: <https://developers.google.com/features/geofencing>.
38. Google. *Activity Recognition*. 2014; Available from: <https://developer.android.com/reference/com/google/android/gms/location/ActivityRecognitionApi.html>.
39. Trends, D. *ANDROID LOLLIPOP IS ONLY ON 5.4 PERCENT OF DEVICES, AND KITKAT IS STILL THE MOST POPULAR*. 2015 28-08-2015]; Available from: <http://www.digitaltrends.com/mobile/android-lollipop-on-5-4-percent-of-devices/>.

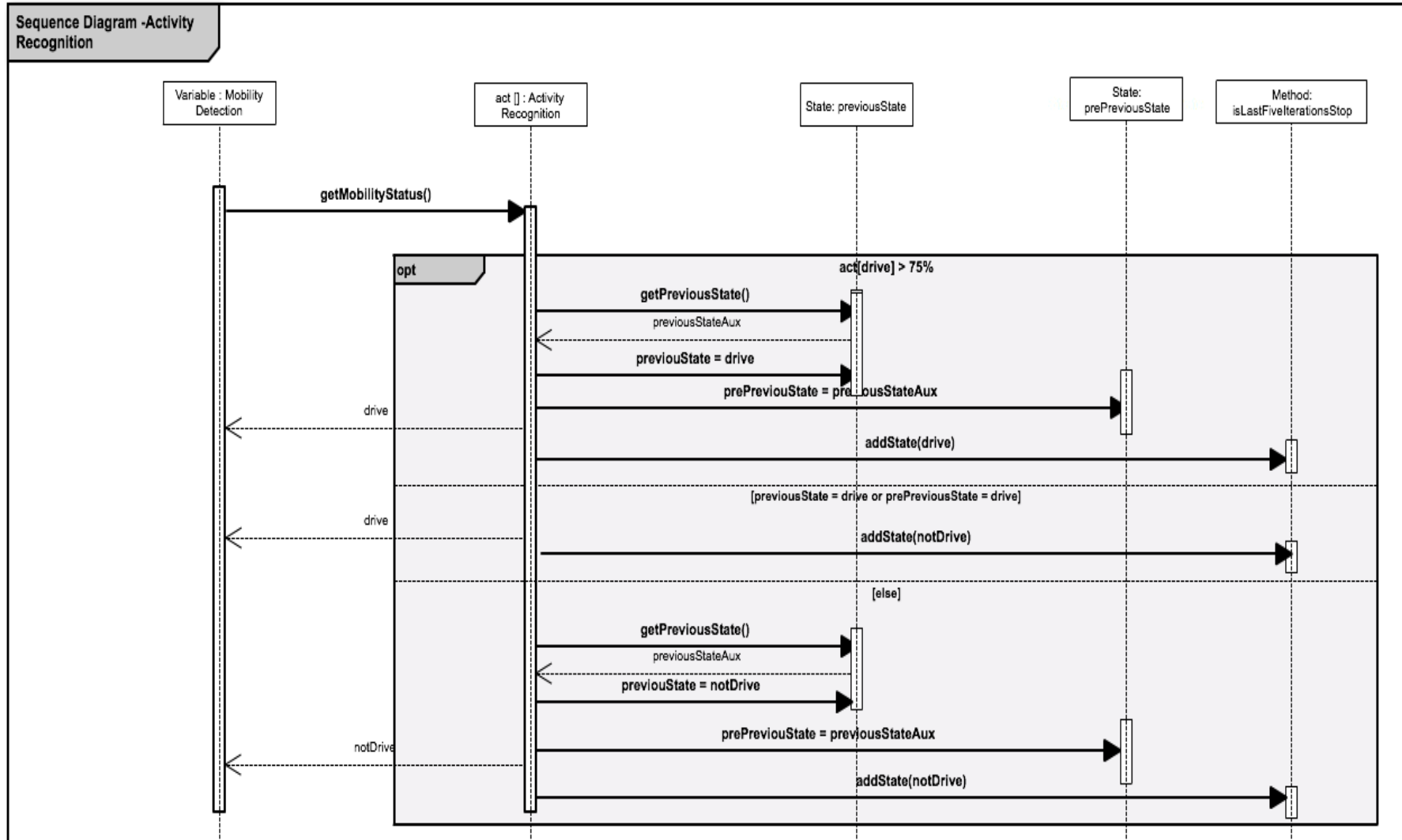
Appendix I – Methodology Overview



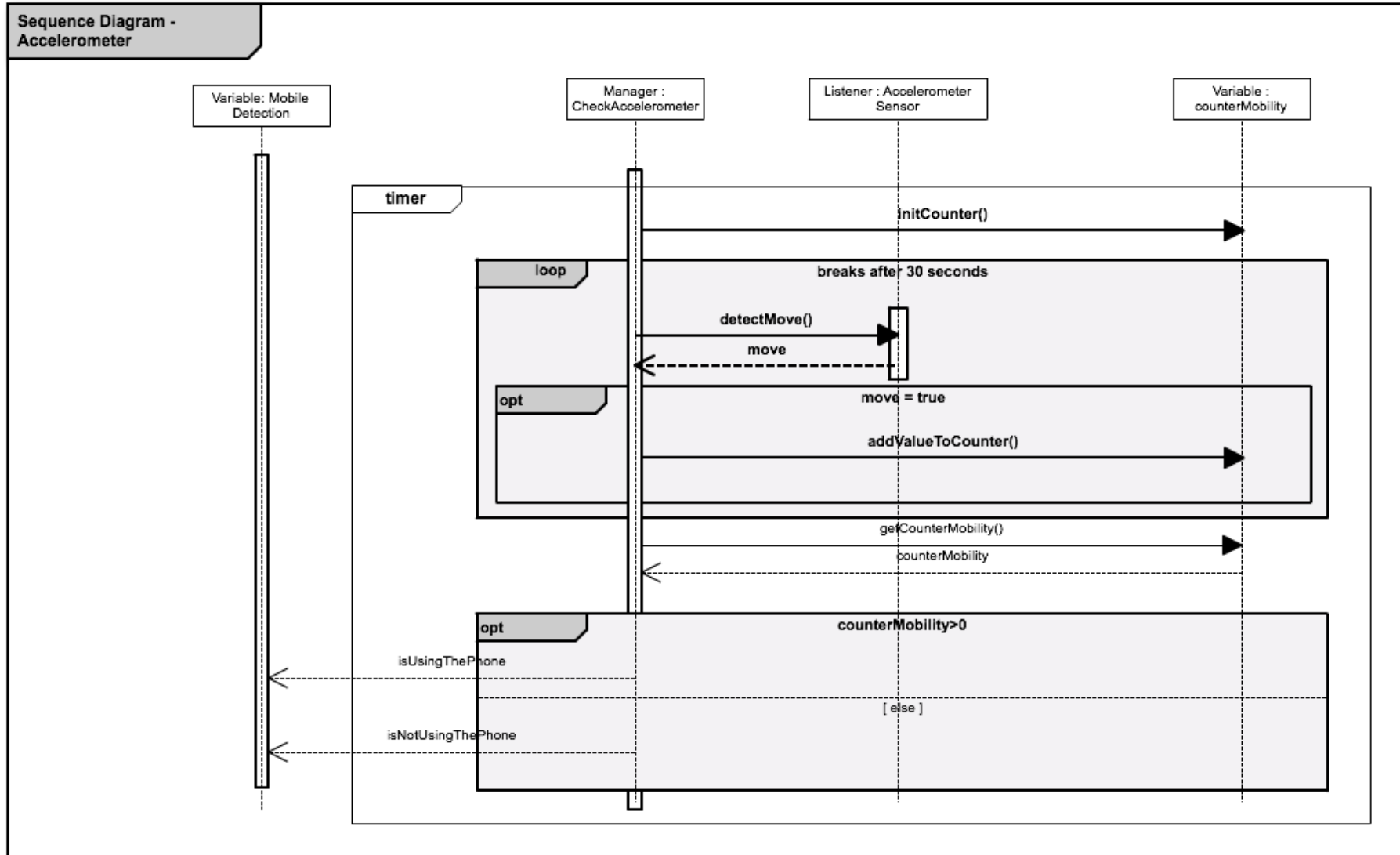
Appendix II – Architectonical Context



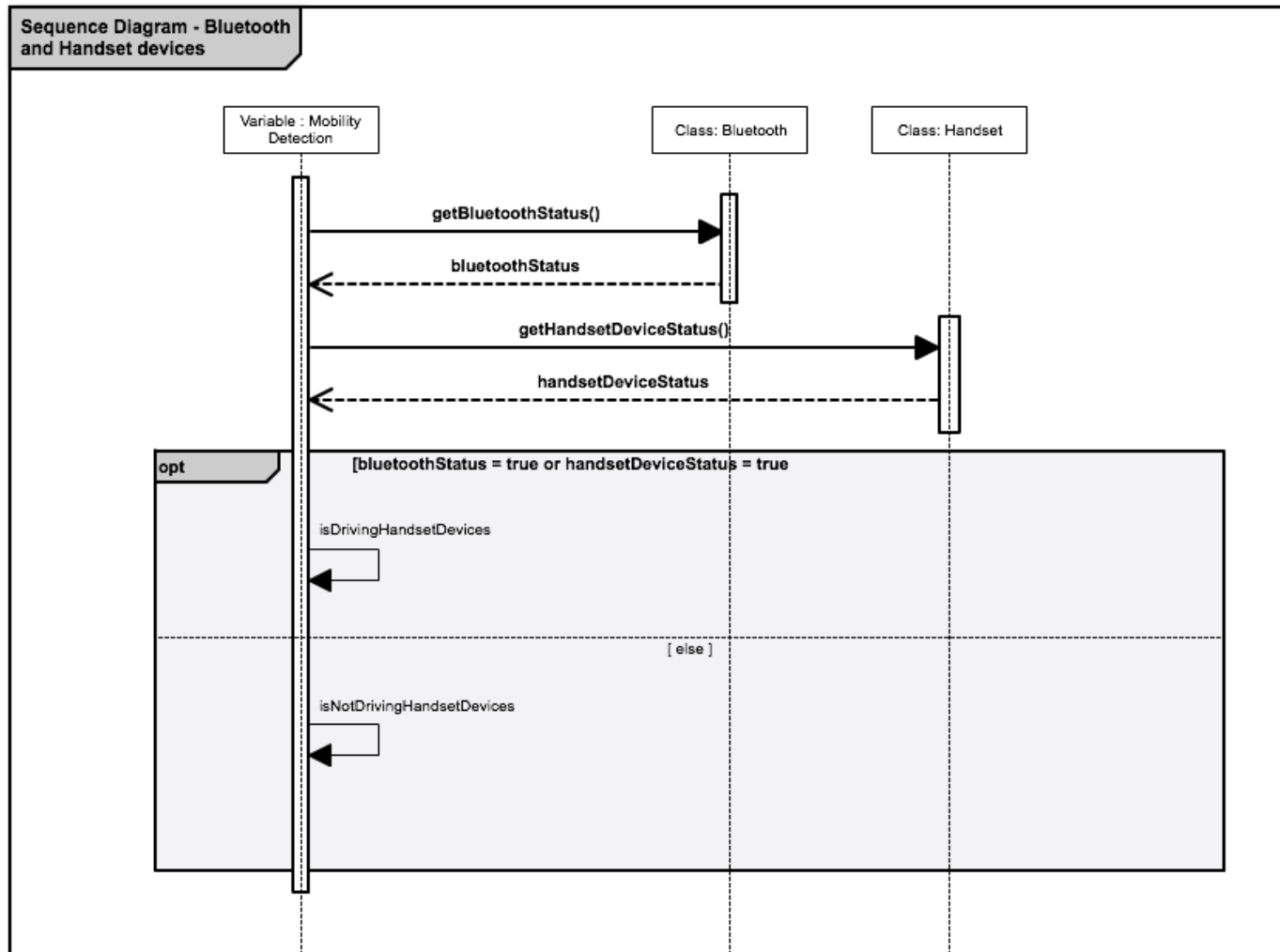
Appendix III – Activity Recognition Algorithm



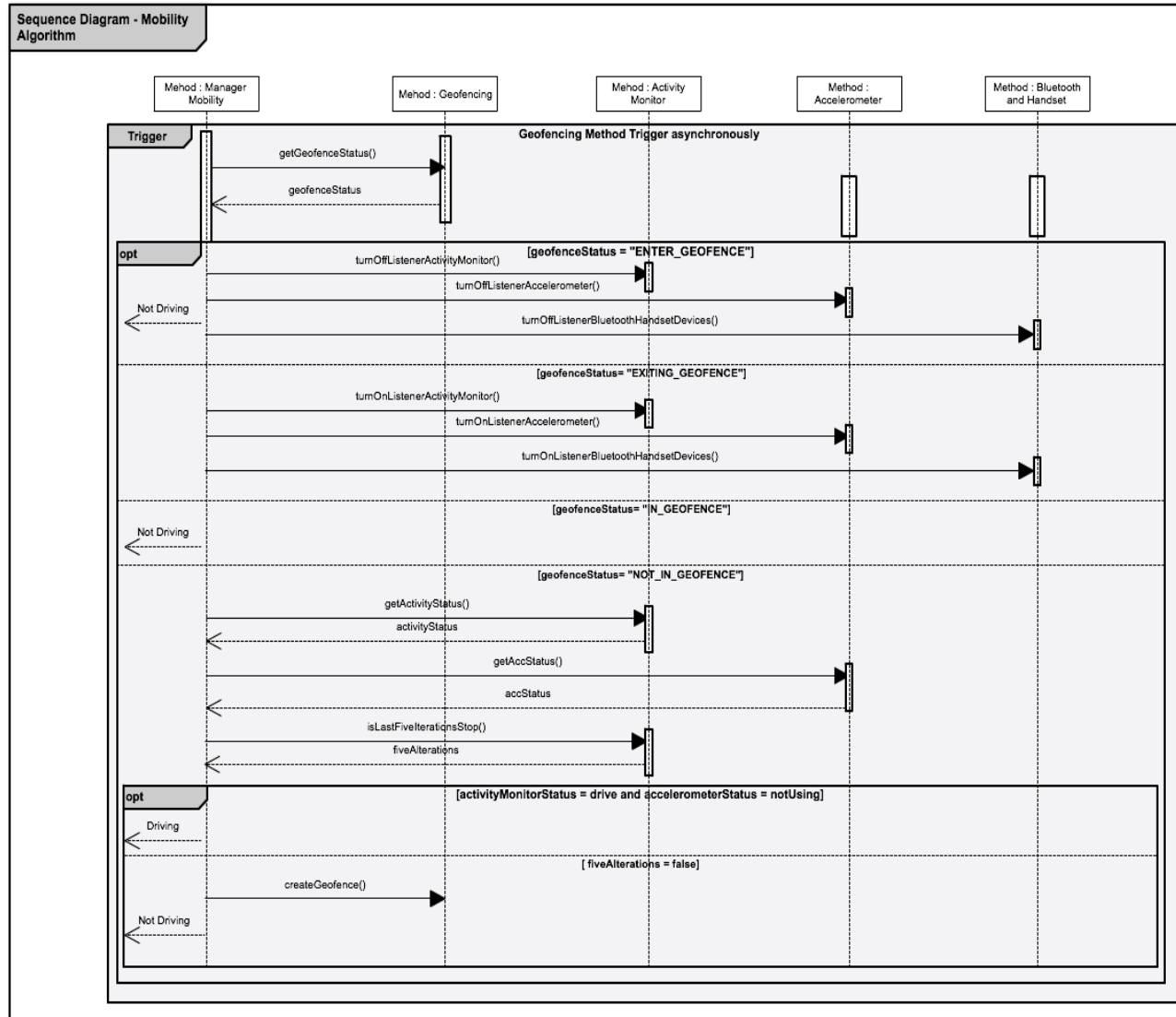
Appendix IV – Accelerometer Method



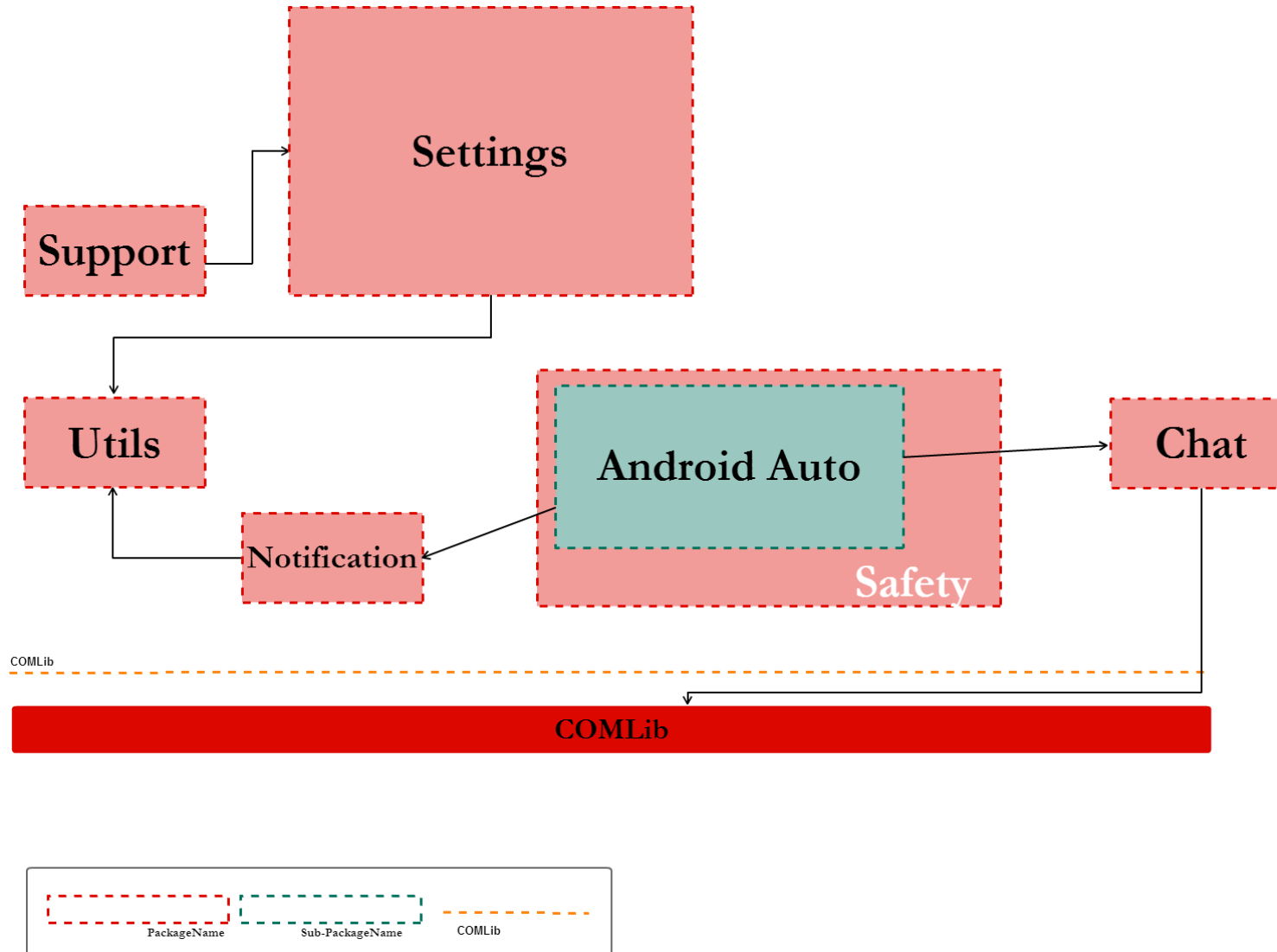
Appendix V –Bluetooth and Handset Devices



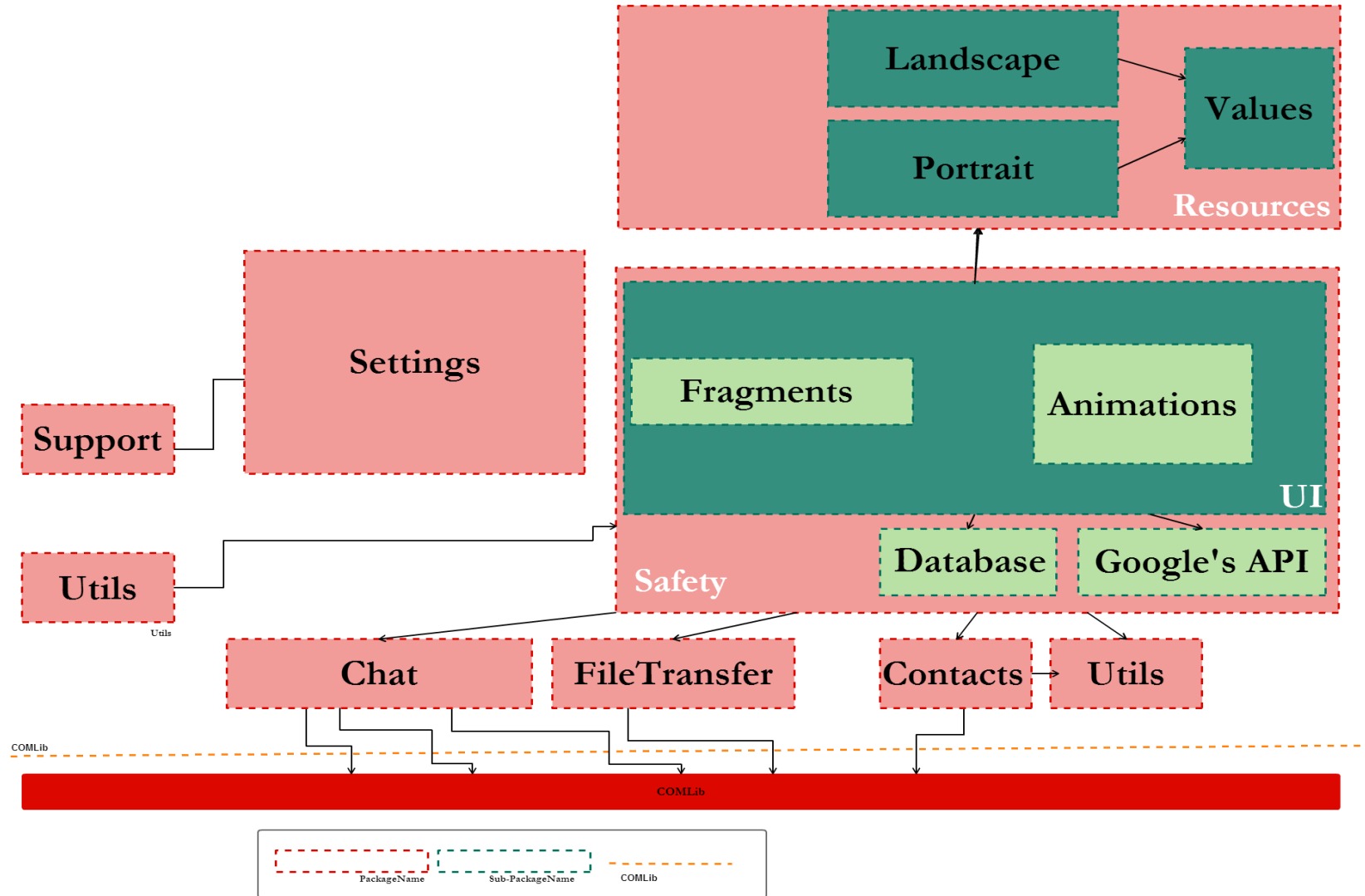
Appendix VII – Mobility Detection Algorithm



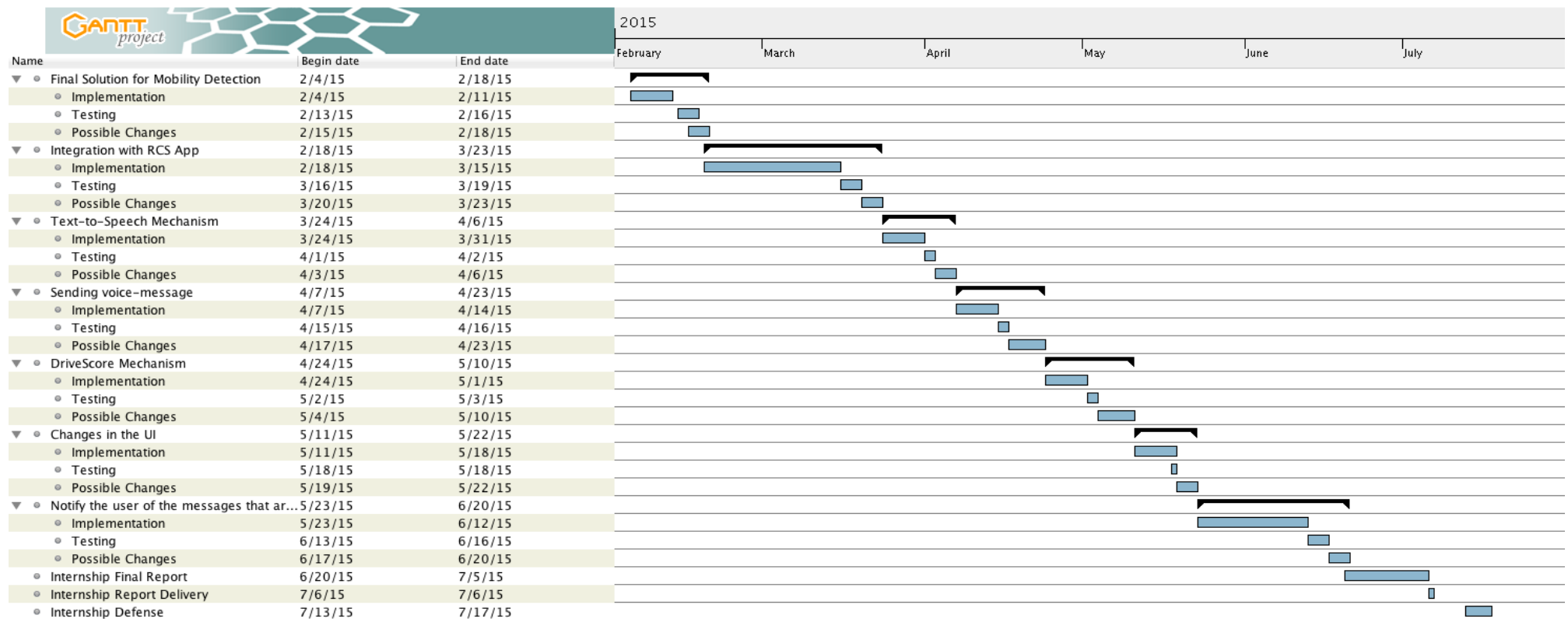
Appendix VII – Android Auto Architecture



Appendix VIII – Safety-Driving Interface Architecture



Appendix X – Second Semester Planning



Appendix XI – Second Semester Real Work

