Politecnico di Milano

SAFESTREETS

POLITECNICO
MILANO 1863

Samuele Meta - Stiven Metaj

Supervisor: Matteo Rossi

SOTTOTITOLO OLEEEEEEEE

Department of Computer Science and Engineering

October 25, 2019

Contents

1 Introduction

The following Requirement Analysis and Specification Document (RASD) aims at illustrating a complete overview of the project *SafeStreets*, providing a baseline for its planning and development. It guides the reader in understanding the specifics of the application domain and the relative System in terms of functional requirements, non functional requirements and constraints. It details how, according to these, the System interacts with the external world, showing concrete use case scenarios. A more comprehensive description of the most relevant features will be modelled with the use of the Alloy language. This document is addressed to all the stakeholders - such as users, system and requirement analysts, project managers, software developers and testers - who will evaluate the correctness of the assumptions and of the decisions contained in it.

1.1 Purpose

SafeStreets is a new crowd-sourced mobile application that allows Citizens to notify Authorities about traffic violations, with particular emphasis on parking contraventions. In fact, every day is possible to encounter minor traffic infringements that affect the driving experience, whether it's a double parking or an unduly occupied parking lot reserved for disables. With *SafeStreets*, Citizens can actively participate in road monitoring, partially compensating for the impossible ubiquity of patrols.
To do so, *SafeStreets* requires the User to create an account as Citizen - providing personal credentials such as name, surname, National Insurance Number (NIN) - and to verify it sending a picture of a valid document. From this moment on, the System will allow the Citizen to send photos of traffic violations, enriched by the relative description, position, date and time. At any time, the application allows the Citizen to visualize the history of the reported violations since it's registered. In case of error or incorrect information, the System allows the Citizen to revoke the alert produced.
Futhermore, the System offers the possibility to sign up as an Authority, once filled the registration form and verified the institutional identity through the relative burocracy. *SafeStreets* will allow Authorities to have a complete overview of the incoming signalations and to choose if to accept or reject them.
Since the System stores all the violations and their metadata, *SafeStreets* is able to process them in order to extract meaningful insights and statistics. The System allows both Citizens and Authorities to access them, with a different visibility level according to the role.
Moreover, if the municipality offers a service that allows to retrieve the information about the accidents occured in the covered area, *SafeStreets* will cross the two knowledge bases to

identify potentially unsafe areas and suggest possible interventions.
The above description can be summarized as follows, in a list of goals:

- [**G.1**] The System allows the User to register to the application either as *Citizen* or as *Authority*, providing an identification code and a password.
- [**G.2**] The System allows the User to report a traffic violation by sending a photo of it and the relative date and position.
- [**G.3**] The System allows the User to find out the streets with highest frequency of violations.
- [**G.4**] The System allows the Authority to find out vehicles that commit the most violations.
- [**G.4**] The System allows the Authority to cross the information in order to identify potentially unsafe areas.
- [**G.5**] The System suggests the Authority possible solutions to the problem.
- [**G.6**] The System allows the User to see the reported violations.
- [**G.7**] The System allows the User to retire a violation.
- [**G.8**] The System completes the report with metadata
- [**G.9**] The System suggests the Authority possible solutions to the problem.
- [**G.10**] The System analyzes plate, model quality of photo
- [**G.11**] The System checks if photo is good (security).
- [**G.11**] The System allows the Authority to accept or refuse a violation.

1.2 Scope

Following the original definition of the so call *World and Machine phenomena* (proposed by M. Jackson and P. Zave), we will clarify the scope and the application range of the proposed System; in order to do this we have to define the *entities* involved.

In particular the *Machine* is the System to be developed, in this case a software application, while the *World* corresponds to the part of the real world that is altered by the System.

This takes us to different types of events which we must describe; in fact the *World* and the *Machine* are associated with distinct *phenomena* whose descriptions follow.

1.2.1 World Phenomena

World phenomena are events that occur in the real world and don't impact directly the System. We identify the following ones:

- **A traffic violation occurrence**: any infraction like double parking, parking on sidewalks, no parking sign violation.
- **A car accident**: an accident that involves the *User* or that the *User* sees.
- **A lane for people with disabilities or for bike riders obstructed**: a parking violation that, in particular, obstruct a reserved lane.
- **An increase of violation in a specific area**: many traffic violations can occur in the same in a specific area or in a precise district.

1.2.2 Shared Phenomena

Shared phenomena are world phenomena that are, as the name suggests, shared with the Machine.

These are splitted in two categories: phenomena *controlled by the World and observed by the Machine* and phenomena *controlled by the Machine and observed by the World*. We identify as the first kind the following ones:

- **A *User* registers or logs in to the application**: a *User* must register in order to send violation occurrences or can login to the application if already registered.
- **A registered *User* try to verify the profile**: after a *User* registers to the application he must verify the profile to have the possibility to report traffic violations.
- **A *User* sends photos and information to the *System***: a *User* can send pictures and other information about a violation he/she wants to report.
- **Municipality offers traffic violations information**: municipality can let the *System* retrieve data about occurred traffic violations.

Instead, we identify as the second kind the following ones:

- **The *System* shows a confirmation message**: after a *User* send a violation occurrence the system shows a confirmation message.
- **The *System* asks to confirm the retrieved violation's street**: the application retrieve the street information from the GPS signal of the *User* device and ask hr a confirmation in order to be sure to have the right violation's street information (if not the *User* input it).
- **The *System* shows the list of reported violations by a *User***: a *User* can ask the application to have a list of all his/her reported violations.
- **The *System* shows lists of areas based on stored violations**: a *User* can ask the application to show the areas where there is the higher frequency of violations.

- **The *System* shows lists of vehicles that commit the most violations**: the *Authorities* can ask the application to show the vehicles that commit most violations (i.e. in a specific area).

1.2.3 Machine Phenomena

Finally, machine phenomena are events that take place inside the System, but there is no way to observe them in the real world. We identify the phenomena that follows:

- **The *System* store the reported violations**: all the reported violations are stored in the database of the *System* (including pictures, date, time, type of violation, position and additional information input by the *User*.
- **The *System* compute a veracity control of reported violations**: the *System* uses a Machine Learning algorithm in order to find if the violation pictures are authentic or not (the information is saved to permit to understand veracity of *Users*).
- **The *System* creates lists of areas and vehicles of interest periodically**: a computation is periodically performed in order to have "ready-to-view" lists of areas with higher frequency of violations and lists of vehicles that commit most of the violations.
- **The *System* retrieve information from Municipality**: the API offered by the Municipality is periodically used to check if new data from *Authorities* can be retrieved in order to cross them with information sent by the *Users*.
- **The *System* compute a list of possible interventions in areas of interest**: a Neural Network is used with the data stored in the database in order to execute a Natural Language Process that permits to create sentences about possible interventions based on the reported violations.

1.3 Definitions, Acronyms, Abbreviations

In this section follow definitions, acronyms (including their meaning) and abbreviations used in this document.

1.3.1 Definitions

- *User*: a person/institution which has to sign up either as a Citizen or Authority and who is not able to access any feature of the application
- *Citizen*
- *Authority*
- *Municipality*
- *Traffic violation*
- *Parking violation*
- *Report*

1.3.2 Acronyms

API	Application Programming Interface
DBMS	DataBase Management System
GPS	Global Positioning System
UI	User Interface
URL	Uniform Resource Locator
NIN	National Insurance Number

Table 1.1: *Acronyms*

1.3.3 Abbreviations

- [**Gn**]: n-th Goal
- [**Dn**]: n-th Domain Assumption
- [**Rn**]: n-th Requirement

1.4 Revision History

[TBD]

1.5 Reference Documents

- Specifications document: "SafeStreets. Mandatory project assignment"
- IEEE Standard 830-1993: IEEE Guide to Software Requirements Specifications
- IEEE Standard 830-1998: IEEE Recommended Practice for Software Requirements Specifications
- Alloy documentation: `http://alloy.lcs.mit.edu`
- UML documentation: `https://www.uml-diagrams.org`

1.6 Document Structure

The rest of the document is organized as follows:

- **Overall Description**: in this section a more in-depth description of the application's domain will be provided, highlighting the context of the System and detailing the phenomena previously mentioned. The most relevant functions of the System will be pointed out and their interactions with all the actors will be illustrated with the help

of Class Diagrams. Finally, this section includes all the constraints, dependencies and assumptions that can be used to entail each Goal.

- **Specific Requirements**: in this section the requirements will be fully explained and organized according to their type, associating to them the relative use cases and Sequence Diagrams to clarify the interactions between the User and the System. The main aim is to provide a useful item for both designers and testers.
- **Formal Analysis**: this section includes the formal model generated according to the Alloy language.

2 Overall Description

2.1 Product Perspective

2.1.1 Use Case Templates

2.2 Product Functions

OLE

2.3 User Characteristics

OLE

2.4 Assumptions, Dependencies and Constraints

OLE

3 Specific Requirements

OLE

3.1 External Interface Requirements

OLE

3.1.1 User Interfaces

ole

3.1.2 Hardware Interfaces

ole

3.1.3 Software Interfaces

ole

3.1.4 Communication Interfaces

ole

3.2 Functional Requirements

OLEE

3.2.1 Use Cases

oleole

3.2.2 Sequence and Activity Diagrams

oleole

3.2.3 Mapping on Requirements

oleole

3.3 Performance Requirements

OLE

3.4 Design Constraints

OLE

3.4.1 Standards Compliance

ole

3.4.2 Hardware Limitations

ole

3.4.3 Any Other Constraint

ole

3.5 Software System Attributes

OLE

3.5.1 Reliability

oleole

3.5.2 Availability

oleole

3.5.3 Security

oleole

3.5.4 Maintainability

oleole

3.5.5 Portability

oleole

4 Formal Analysis using Alloy

4.1 CODICE SCHIFOSO

OLE

4.2 SCREEN DEL CODICE SCHIFOSO CHE RUNNO E FUNZIONA TUTTO STRABENE

OLE

5 Effort Spent