

Software Engineering 2

A.Y. 2019/2020

**SafeStreets**

RASD – Requirement Analysis and Specification Document

Version 1.0

**Authors:**

Daniele Comi 10528029 - 944534  
Anton Ghobryal 10501942 - 945577

**Professor:**

Elisabetta Di Nitto

Contents

[1. Introduction 3](#_Toc22478870)

[1.1 Purpose 3](#_Toc22478871)

[1.2 Scope 3](#_Toc22478872)

[1.3 Definitions, Acronyms, Abbreviations 3](#_Toc22478873)

[1.4 Revision history 3](#_Toc22478874)

[1.5 Reference Documents 3](#_Toc22478875)

[1.6 Document Structure 3](#_Toc22478876)

[2. Overall description 3](#_Toc22478877)

[2.1 Product perspective 3](#_Toc22478878)

[2.2 Product functions 3](#_Toc22478879)

[2.3 User characteristics 3](#_Toc22478880)

[2.4 Assumptions, dependencies and constraints 3](#_Toc22478881)

[3. Specific requirements 4](#_Toc22478882)

[3.1 External Interface Requirements 4](#_Toc22478883)

[3.1.1 User interfaces 4](#_Toc22478884)

[3.1.2 Hardware Interfaces 4](#_Toc22478885)

[3.1.3 Software Interfaces 4](#_Toc22478886)

[3.1.4 Communication Interfaces 4](#_Toc22478887)

[3.2 Functional Requirements 4](#_Toc22478888)

[3.3 Performance Requirements 4](#_Toc22478889)

[3.4 Design Constraints 4](#_Toc22478890)

[3.4.1 Standards compliance 4](#_Toc22478891)

[3.4.2 Hardware limitations 4](#_Toc22478892)

[3.4.3 Any other constraint 4](#_Toc22478893)

[3.5 Software System Attributes 4](#_Toc22478894)

[3.5.1 Availability 4](#_Toc22478895)

[3.5.2 Security 4](#_Toc22478896)

[3.5.3 Maintainability 4](#_Toc22478897)

[3.5.4 Portability 4](#_Toc22478898)

[4. Formal Analysis using Alloy 4](#_Toc22478899)

[5. Effort spent 4](#_Toc22478900)

[6. References 4](#_Toc22478901)

# Introduction

## Purpose

### 1.1.1 General Purpose

SafeStreets is a crowded-sourced application that intends to provide users with the possibility to notify authorities when traffic violations occur, specifically parking violations. The application allows users to send pictures of violations, including their date, time, and position, to authorities. The main purpose of SafeStreets is to reduce the number of accidents that may be caused by certain violations that can be avoided easily. The following examples may illustrate and visualize the type of violations that may be captured and notified to the authorities:

* Double parking
* Expiry of the parking time limit
* No parking area
* Parking in places reserved to people with disabilities
* Parking in the middle of bike lanes
* Parking near bus stops
* Parking on crosswalk
* Parking on residents reserved spots
* Parking ticket missing
* Possible vehicles damage by third parties (e.g. broken glass)

SafeStreets stores the information provided by the users, completing it with suitable metadata. In particular, when it receives a picture, it runs an algorithm to read the licence plate and stores the retrieved information with the violation, including also the type of violation (input by the user) and the name of the street where the violation occurred (which can be retrieved from the geographical position of the violation). In addition, the application allows both end users and authorities to mine the information that has been received, for example by highlighting the streets (or the areas) with the highest frequency of violations, or the vehicles that commit the most violations. Of course, different levels of visibility are offered to different roles, for example the authorities can see the licence plate numbers of the vehicles that commit any violation while the end user cannot see that.

Moreover, there’s another functionality that can be provided by SafeStreets. If the municipality offers a service that allows users to retrieve the information about accidents that occur on the territory of the municipality, SafeStreets can cross that information with its own data to identify potentially unsafe areas, hence suggest possible interventions depending of the type of the most committed violation in that area. The following examples show which intervention could be suggested depending on the preceding examples of violations presented earlier in this paragraph:

* Add a barrier between the bike lane and the part of the road for motorized vehicles
* Install a towaway zone sign
* Increase parking slots
* Increase local police controls

The main purpose of this functionality is that SafeStreets also identifies areas with critical number of accidents and reports suggestions as a possible solution as an automatized method to engage with the problem. Thus, it could help the authorities to highlight where the interventions should be provided, and this functionality should make it easier to point out the areas with critical statistics. So, if the municipality provide the needed information, it helps with the traceability of the main problem, therefore handling it providing also a higher measurement on local security.

### 1.1.2 Goals

Taking the abstraction as an outcome of the “real-world” only, we should be able to define the goals as a part of the requirement engineering of an S2B to satisfy the stakeholders’ requests:

* G1: Every registered user should be able to notify violations
* G2: Every recognized authority should be able to access the application
* G3: Every recognized authority should be able to retrieve any information about any violation that has been pointed out by a registered user
* G4: Every communication from the user must include a violation that has been committed by a recognizable vehicle
* G5: Every registered end user should be able to mine general information about the violations committed in a certain area
* G6: Every recognized authority must be able to verify the notified violations by the registered users
* G7: Every recognized authority must be able to receive suggestions about improving the local security

Reading these goals, we should acknowledge the fact that the system considers two most end users: the normal user and the authorities. They’ll be defined later on.

## 1.2 Scope

## 1.3 Definitions, Acronyms, Abbreviations

## 1.4 Revision history

## 1.5 Reference Documents

## 1.6 Document Structure

# Overall description

## Product perspective

## 2.2 Product functions

## 2.3 User characteristics

## 2.4 Assumptions, dependencies and constraints

# Specific requirements

## 3.1 External Interface Requirements

### 3.1.1 User interfaces

### 3.1.2 Hardware Interfaces

### 3.1.3 Software Interfaces

### 3.1.4 Communication Interfaces

## 3.2 Functional Requirements

## 3.3 Performance Requirements

## 3.4 Design Constraints

### 3.4.1 Standards compliance

### 3.4.2 Hardware limitations

### 3.4.3 Any other constraint

## 3.5 Software System Attributes

### 3.5.1 Availability

### 3.5.2 Security

### 3.5.3 Maintainability

### 3.5.4 Portability

# Formal Analysis using Alloy

# Effort spent

# References