POLITECNICO
MILANO 1863

COMPUTER SCIENCE AND ENGINEERING
SOFTWARE ENGINEERING II
2025 - 206

RASD

Requirement Analysis and Specification Document

Best Bike Paths

Authors:
Leonardo Guglielmi, Francesco Lo Conte

Version:
1.0
(November 19, 2025)

Contents

1 Introduction **4**
- 1.1 Purpose . . . 4
  - 1.1.1 Goals . . . 4
- 1.2 Scope . . . 4
  - 1.2.1 World phenomena . . . 5
  - 1.2.2 Shared phenomena . . . 5
- 1.3 Definitions, Acronyms, Abbreviations . . . 6
  - 1.3.1 Definitions . . . 6
  - 1.3.2 Acronyms . . . 7
  - 1.3.3 Abbreviations . . . 7
- 1.4 Revision history . . . 7
- 1.5 Reference documents . . . 7
- 1.6 Document structure . . . 7

2 Overall description **9**
- 2.1 Product perspective . . . 9
  - 2.1.1 Scenarios . . . 9
  - 2.1.2 Domain Class Diagram . . . 12
  - 2.1.3 State Diagrams . . . 14
- 2.2 Product functions . . . 18
- 2.3 User characteristics . . . 18
- 2.4 Assumptions, dependecies and constraints . . . 18

3 Specific requirements **18**
- 3.1 External interface requirements . . . 18
  - 3.1.1 User interfaces . . . 18
  - 3.1.2 Hardware interfaces . . . 18
  - 3.1.3 Software interfaces . . . 18
  - 3.1.4 Communication interfaces . . . 18
- 3.2 Functional requirements . . . 18
- 3.3 Performance requirements . . . 18
- 3.4 Design constraints . . . 18
  - 3.4.1 Standard compliance . . . 18
  - 3.4.2 Hardware limitations . . . 18
  - 3.4.3 Any other constraint . . . 18
- 3.5 Software system attributes . . . 18
  - 3.5.1 Reliability . . . 18
  - 3.5.2 Availability . . . 18
  - 3.5.3 Security . . . 18
  - 3.5.4 Maintainability . . . 18
  - 3.5.5 Portability . . . 18

4 Formal anlaysis using Alloy **18**

5 Effort spent **18**

6 References **19**

1 Introduction

1.1 Purpose

The growing interest in cycling, whether as a recreational activity, a means of transportation, or a sport, brings with it a significant challenge: finding routes that are not only efficient, but also safe and well-maintained. Cyclists often lack reliable and up-to-date information on trail conditions, such as the presence of potholes, obstacles, or roads with little traffic. At the same time, many cyclists meticulously log their trips to monitor their performance, collecting valuable data that, however, remains siloed. This creates a gap where vital community knowledge about trail quality is not easily shared or accessible. "Best Bike Paths" (BBP) aims to provide a solution. Commissioned by a cyclists' association, BBP will be a software system designed to create and manage a community-driven inventory of cycling routes. The platform will help bridge this information gap by allowing registered users to track their trips while simultaneously submitting detailed information on the condition of their routes. Other users, registered or not, will then be able to use this collective data to find and display the best possible cycling routes between two points, ranked by a quality score.

1.1.1 Goals

- **G1:** A registered user wants to track their personal cycling activities and related performance statistics.
- **G2:** A registered user wants to contribute to the community inventory by sharing reliable information on the condition of the trails (e.g. quality, obstacles, potholes).
- **G3:** Any user (registered or not) wants to find and view the best cycling route between an origin and a destination, based on up-to-date and relevant data.
- **G4:** The cycling association aims to provide the community with a tool to create, consult, and maintain a reliable and centralized inventory of cycling routes.

1.2 Scope

The project scope covers users interacting with the system, user-generated actions that influence the system, and system-generated actions that impact the outside world.

For this project, the following users interacting with the system have been identified:

- **Registered User**
- **Any User**

A Registered User will be able to use the application to log and store their trips, tracking their cycling activities and related statistics. When available, this data can be enriched with weather information retrieved from external services. Furthermore, this user is the primary contributor to the inventory. They can enter route information in two ways:

1. In **manual mode**, by actively specifying the route status (e.g., optimal, requires maintenance) and the presence of obstacles (e.g., potholes).
2. In **automatic mode**, by allowing the app to acquire data from GPS and mobile device sensors while cycling, in order to automatically detect potential problems such as potholes.

For automatically collected data, the system will ask the user to confirm or correct the information before making it available to the community. Once confirmed or manually entered, this information becomes publishable.

Any user, whether registered or not, can benefit from the collected information. This user can specify a starting point and a destination and ask the system to display available cycling routes on a map. If multiple routes exist, BBP will present them based on a score, calculated based on the route status derived from the data confirmed by users.

1.2.1 World phenomena

- **WP1:** A registered user decides to start a cycling activity.
- **WP2:** Any user needs to find a cycling route between two places.
- **WP3:** A registered user decides to contribute to the BBP inventory.
- **WP4:** A registered user, while pedaling, physically encounters an obstacle or evaluates the condition of a route.

1.2.2 Shared phenomena

World controlled

- **SP_WC1:** The registered user launches the "Register Trip" function on the application.
- **SP_WC2:** The registered user stops the "Register Trip" function on the application.
- **SP_WC3:** The registered user opens the interface for manually entering route information.
- **SP_WC4:** The registered user enters the data (e.g. "optimal" status, "hole" presence) and sends the manual entry form.
- **SP_WC5:** The registered user selects a notification or confirmation request for automatically detected data.

- **SP_WC6:** The registered user presses "Confirm" to validate automatically detected data (e.g. a pothole).
- **SP_WC7:** The registered user presses "Delete" to invalidate an automatically detected data (false positive).
- **SP_WC8:** The registered user modifies an automatically detected piece of data (e.g. corrects the position of the hole on the map) and saves the change.
- **SP_WC9:** Any user enters a source and destination address.
- **SP_WC10:** Any user starts the route search.

Machine controlled

- **SP_MC1:** The system shows the registered user the statistics of the completed trip.
- **SP_MC2:** The system shows the registered user the weather data associated with the trip.
- **SP_MC3:** The system presents the Registered User with a confirmation request for automatically detected data.
- **SP_MC4:** The system shows the user a map with the cycling routes found between the origin and the destination.
- **SP_MC5:** The system displays the details of a route, including its score and confirmed obstacles.
- **SP_MC6:** The system displays an error message (e.g., "Weather service unavailable").

1.3 Definitions, Acronyms, Abbreviations

This section contains the definitions for people that may not know what a specific concept is, acronyms and abbreviations used throughout the document.

1.3.1 Definitions

- **Bike Path:** a route deemed suitable for cycling. This includes paths with a proper bike track or roads where cars are rare and speed limits are compatible with the average speed of a bike.
- **Trip:** a personal record of a user's cycling activity, stored by the system to track performance metrics like distance and speed.
- **Publishable Information:** data about a bike path (e.g., status, obstacles) that a registered user has either entered making it available to the wider community.

- **Path Score:** a metric computed by BBP to rank route options. It is based on the status of the path and its effectiveness in getting the user from their origin to their destination.
- **Obstacle:** any significant element or condition on a cycle path that may represent a danger or hindrance to the cyclist (e.g. pothole).

1.3.2 Acronyms

- **BBP:** Best Bike Paths.
- **GPS:** Global Positioning System.
- **API:** Application Programming Interface.

1.3.3 Abbreviations

- **G*:** Goal.
- **WP*:** World Phenomenon.
- **SP*:** Shared Phenomenon.
- **R*:** Requirement.
- **UC*:** Use Case.
- **D*:** Domain Assumption.

1.4 Revision history

- **Version 1.0 (17/11/2025)**

1.5 Reference documents

This document is based on the following materials:

- The specification of the RASD and DD assignment of the Software Engineering II course a.y. 2025/26.
- Course slides shared on WeBeep.
- Past Requirement Analysis and Specification Documents.

1.6 Document structure

1. **Introduction:** a brief description of the project. It contains the main goals and objectives that the final system wants to achieve.
2. **Overall description:** this section is a high-level representation of the system and of the interactions of the system with the other actors.

3. **Specific requirements:** a detailed list of all the requirements needed for the system to achieve the goals. It contains valuable information for developers.
4. **Formal analysis using Alloy:** a formal description of the model of the system with Alloy.
5. **Effort spent:** the time spent on each section of the document, for each member of the group.
6. **References:** reference to documents or tools used for writing this document..

2 Overall description

2.1 Product perspective

2.1.1 Scenarios

Registering a new user

User "Zoe" has just downloaded the BBP app because she wants to start tracking her trips and contributing to the community. From the welcome screen, she selects "Create Account." She enters her information and accepts the privacy policy. The system verifies the information, creates the new profile in the database, and sends a confirmation. Once the process is complete, Zoe gains "Registered User" status: the app interface changes, enabling the previously disabled "Start Trip" and "Report Problem" buttons.

Activity monitoring and data enrichment

Registered user "Alessandro" is preparing for his weekly training session. He wants to track his performance, including correlation with the weather. He launches the BBP app, logs in, and from the main screen, presses "Start Trip". During the session, Alessandro pedals, focusing solely on the road, while the system silently tracks his GPS route and biometric data (if connected). At the end, Alessandro presses "End Trip". The system saves the route and immediately queries the external weather service. A few moments later, Alessandro views the trip summary: he sees his map, the 45 km traveled, his average speed, and, thanks to the weather integration, notes that the strong headwind he detected has negatively impacted his time. Satisfied with the detailed analysis, he closes the app.

Active contribution to the community (manual entry)

Registered user "Bianca" is riding a popular bike path when she notices that a stretch, previously marked as "Optimal", is now blocked by unreported construction. She decides to alert the community. She safely stops and opens BBP. Selects "Report Problem", and the system geolocates her location. Bianca selects the "Obstacle" category and the "Construction in Progress" subcategory. She adds a text note ("Traffic blocked due to excavation") and submits the report. The system immediately records the information as "Publishable". From that moment, the "Path Score" for that stretch drops dramatically, and future users planning a route in that area will see a warning and will likely be diverted onto alternative routes.

The Auto-Detection cycle

This scenario describes the complete flow of passive data acquisition, from

invisible collection to explicit validation.

- **Detection during the journey**

  Registered user "Carlo" activates "Automatic Mode" to detect potholes before heading to work. He places his smartphone in the handlebar mount and begins pedaling. As he rides along Via Verdi, the device's accelerometer registers a strong vertical impact unrelated to braking. The BBP system analyzes the pattern in real time, classifies it as a "Potential Pothole", and stores the GPS coordinates and timestamp. All this happens in the background, without sending notifications or sounds so as not to distract Carlo from his riding.

- **Post-trip confirmation**

  Arriving at the office, Carlo finishes his trip. The system displays a notification: "Two potential problems detected. Would you like to check them?" Carlo opens the review screen. The system displays a map with the two points. Carlo selects the first one, Via Verdi, and clearly remembers the pothole. He presses "Confirm". The system promotes that data to "Publishable Information".

- **Handling a false positive**

  Carlo examines the second point detected on Corso Italia. He realizes that he deliberately stepped onto a curb to park his bike there. It's not a traffic hazard. He clicks "Ignore". The system discards the raw data and doesn't create any public alerts, preventing the inventory from being contaminated with incorrect data.

Intelligent route planning (General User)

"Diana", a tourist, wants to explore the city by bike but is worried about traffic and poor roads. She accesses the BBP website without logging in and enters "Hotel Plaza" as the origin and "Museo della Scienza" as the destination. The system calculates three possible routes. Diana notices that the shortest route (3 km) has a low "Path Score" and is colored orange on the map, with several "Pothole" icons along the way. An alternative, slightly longer route (3.5 km) has an excellent "Path Score" and is highlighted in green, indicating a bike path in excellent condition. Diana chooses the green route. While navigating, she avoids stress and dangers thanks to information aggregated by the community.

Data updating and maintenance

Registered user "Edoardo" is driving along a road where a large pothole had been reported the previous week (which he also confirmed). He is pleased to note that the municipality has resurfaced the section and the pothole is no

longer there. To keep the inventory updated, Edoardo selects the pothole marker on the map and presses "Mark as Resolved". The system records this new input. If other users confirm the resolution (or if Edoardo has a high confidence level in the system), the obstacle will be removed and the section's "Path Score" will start to increase again.

Historical performance analysis

Registered User "Alessandro", after months of using BBP, wants to analyze his progress. He accesses the "My Trip History" section. The system presents a chronological list of all his saved trips. Alessandro filters by "Last Month" and displays an aggregate graph showing the increase in his average speed and total kilometers traveled. He selects a specific trip from two months ago; the system retrieves all the details from the database, including that day's weather conditions (e.g., "Light rain"), allowing Alessandro to remember why his performance was below average on that date.

2.1.2 Domain Class Diagram

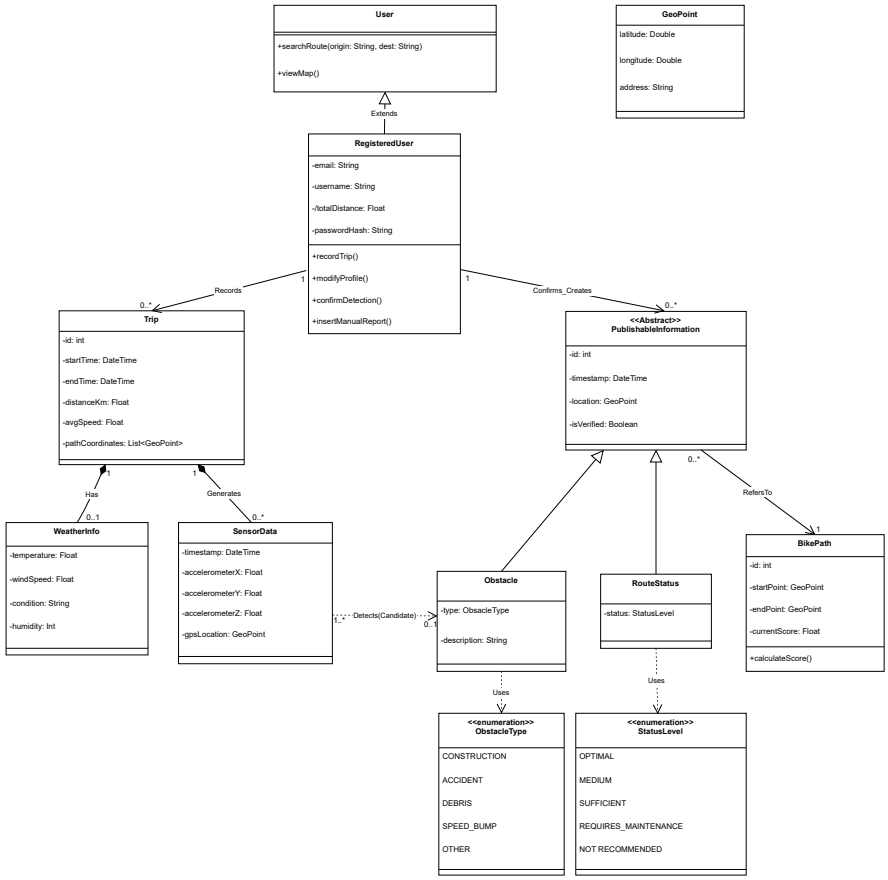

Figure 1: Domain Class Diagram of the BBP system

Figure 1 shows the domain class diagram. The main architectural choices are explained below:

- **User Generalization:** To avoid duplication and facilitate future scalability, the `User` superclass has been introduced. It encapsulates basic functionality accessible to everyone, such as route search and map viewing. The `RegisteredUser` class extends this foundation, adding authentication data and the main writing functionality: `recordTrip()`, `insertManualReport()`, and `confirmDetection()`. This structure allows for easy extension to future roles such as "Administrator" or "Moderator."
- **Information Abstraction and Scoring:** The abstract

`PublishableInformation` class was created to logically group all alerts (whether `RouteStatus` or `Obstacle`). This polymorphic approach greatly simplifies the calculation of the Path Score: the system can iterate over a generic list of confirmed information associated with a trip to calculate its score, without having to use separate logic for each type of alert.

- **Sensor Scalability:** Although the assignment specifically mentions potholes, the model correctly links the raw `SensorData` data to the generic `Obstacle` class via the "Detects (Candidate)" dependency. This design ensures that the system can evolve to detect other types of anomalies in the future without changing the core data model.
- **Trip Composition and Data Lifecycle:** There is a composition relationship between `Trip` and its internal data: `WeatherInfo` and `SensorData`. This indicates that this data is closely tied to the trip lifecycle: if a user decides to delete a trip from their history, the associated weather data and raw sensor data will also be automatically removed, preventing data fragmentation and ensuring database cleanliness.

2.1.3 State Diagrams

User Session Lifecycle

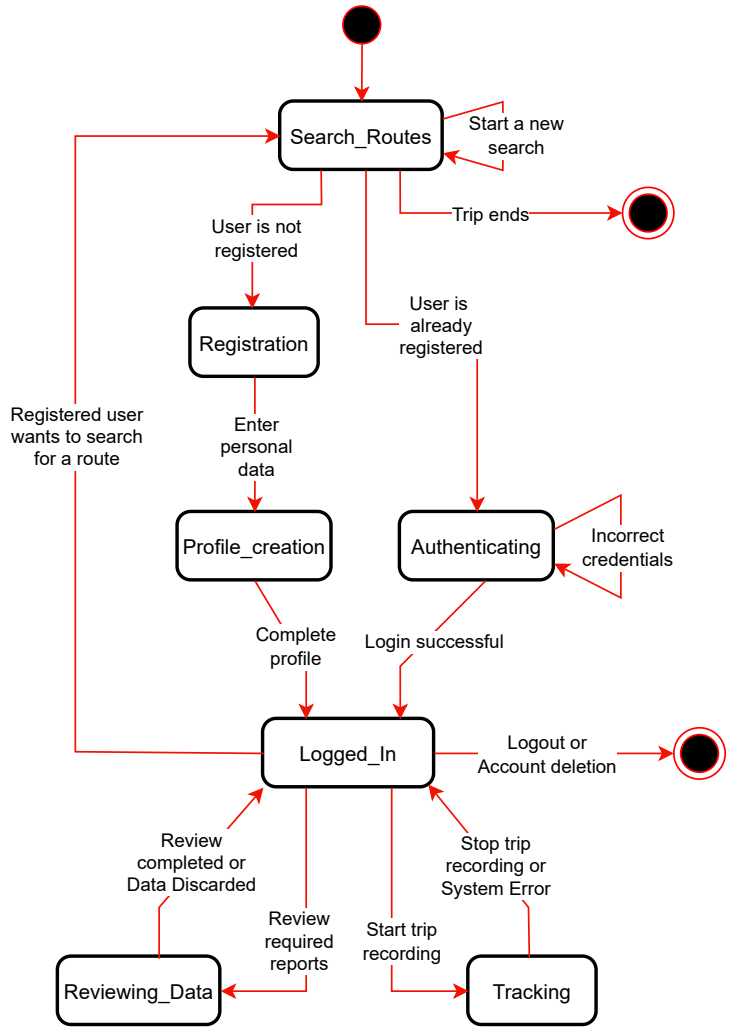

Figure 2: State diagram of a BBP system user's lifecycle

The finite state diagram in Figure 2 models the **user session lifecycle** within the BBP system, defining how the user transitions from the anonymous

browsing state to the fully operational one. The system is designed to ensure that all basic functionality, such as route search and map viewing, is immediately accessible, with a single initial state that converges on `Search_Routes`, the universal entry point. From this anonymous browsing state, the user can choose to authenticate whether they are already logged in or not. Once the `Logged_In` state is reached, the user unlocks the contribution capabilities, which are critical to the system's value. This state serves as a hub, allowing the user to initiate trip tracking by moving to the `Tracking` state (when sensors are active) or to proceed to `Reviewing_Data`. Both contribution states are separated to reflect their high impact on resources (tracking) or data consistency (auditing). The session can end by exiting `Search_Routes` (for both anonymous and registered users) or by `Logout` from the operational state for the registered user.

Trip Lifecycle

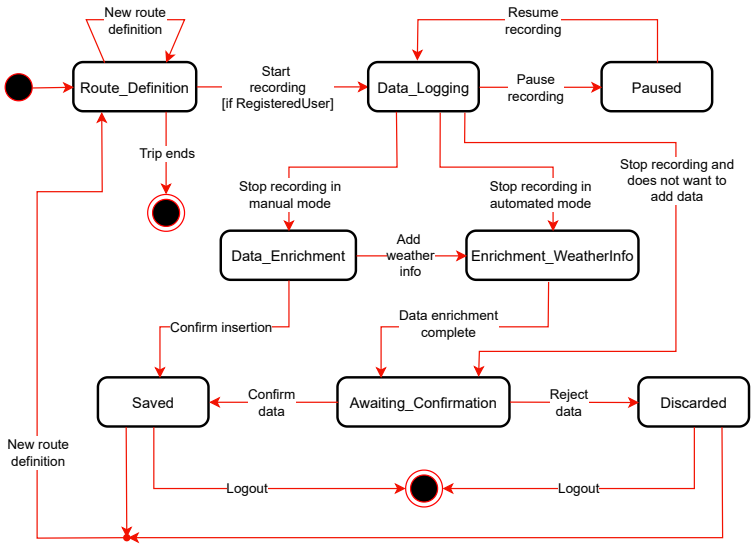

Figure 3: State Diagram of the Lifecycle of a Trip in the BBP System

The diagram in Figure 3 models the complete lifecycle of a `Trip`, from its inception to its final storage or discard. The process begins in the initial `Route_Definition` state, which represents the hub where a new route can be defined or an existing one can be used. The fundamental transition to data acquisition occurs only if the `[if RegisteredUser]` guard condition is satisfied, ensuring that only authenticated users can initiate tracking, based on the system's contribution requirements. Once in the `Data_Logging` state, the system actively logs raw sensor data (GPS, accelerometer) if in automatic

mode. This state offers flexibility, allowing data acquisition to be paused and resumed via transitions. The system manages three distinct transitions when recording is stopped, resulting in separate processing paths:

- **Stop in manual mode**: This transition allows the user to actively add non-sensor data to the route.
- **Stop in automated mode**: Indicates that the route has ended, starting the automatic processing cycle.
- **Stop without data**: If the user does not wish to add any data, they go directly to the confirmation to save or delete the collected data (if collected).

The automated processing cycle begins with `Enrichment_WeatherInfo`, where the system enriches the trip with weather data retrieved from external services. Once enrichment is complete, the flow moves to `Awaiting_Confirmation`. This state is crucial for data quality: here, the user must decide whether to validate the anomalies detected by the sensors (e.g., potholes) or discard them. The cycle closes by returning to the `Route_Definition` state or definitively exiting the system, demonstrating how data only goes from ephemeral to persistent information through a rigorous validation process.

Data Lifecycle

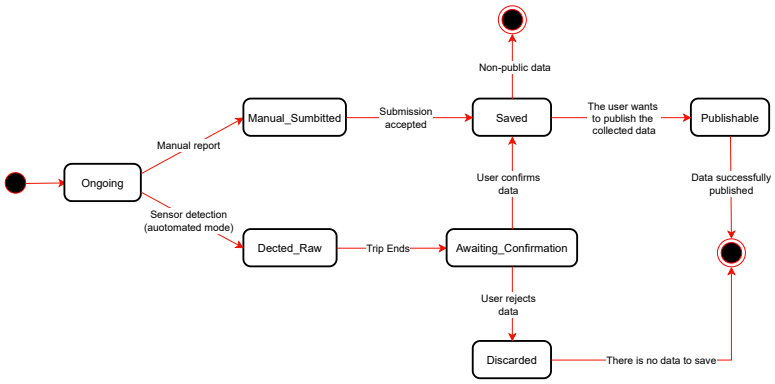

Figure 4: Data lifecycle state diagram in BBP system

The diagram in Figure 4 models the complete data lifecycle, from its origin to its final state. The process rigorously distinguishes data based on its source to direct it to the correct validation path. The flow forks immediately from the initial state:

- **Manual Path:** The user generates a `Manual report` that transitions to the `Manual_Submission` state. The data, being the result of an explicit action, is initially saved and can be published if the user wishes.

- **Automatic Path:** The data passively detected by the sensors transitions to the `Dected_Raw` state. This raw data must pass through the `Awaiting_Confirmation` state at the end of its journey.

The pending confirmation state is the critical checkpoint: the user is responsible for validating the discovery to allow it to move to `Publishable`, or discarding it, moving it to `Discarded`. Only data in the `Publishable` state is integrated and can influence the `Path Score`. The cycle ends with final publication or discard.

Bike Path Status Lifecycle

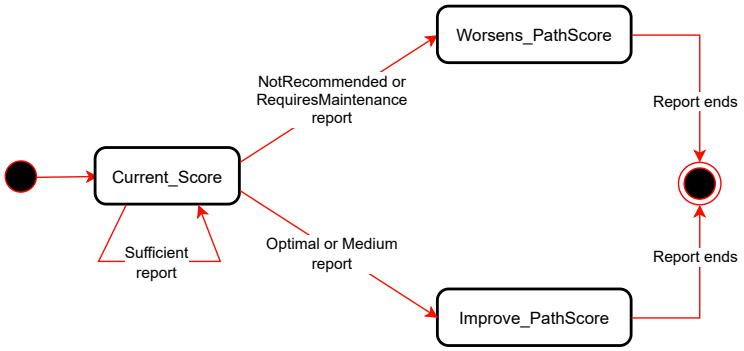

Figure 5: BBP Path State Lifecycle State Diagram

The diagram in Figure 5 models the evolution of a Bike Path's Quality Score in response to user contributions. The entry point is the `Current_Score` state, which represents the value of the path at the time of consultation. This value is dynamic and subject to change based on active reports:

- A hazard or maintenance report (`NotRecommended or Requires Maintenance report`) triggers the `Worsens_PathScore` state. This indicates an immediate degradation in quality.
- Conversely, an `Optimal or Medium report` triggers the `Improve_PathScore` state, indicating an improvement in the path's quality.
- A `Sufficient report` acts as a validation of the current score, maintaining its status and contributing to the data's freshness without drastically altering its perceived quality.

The diagram emphasizes that the score is a dynamic value, constantly recalculated based on the validity and freshness of the active reports in the BBP inventory.

2.2 Product functions

2.3 User characteristics

2.4 Assumptions, dependecies and constraints

3 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 Standard compliance

3.4.2 Hardware limitations

3.4.3 Any other constraint

3.5 Software system attributes

3.5.1 Reliability

3.5.2 Availability

3.5.3 Security

3.5.4 Maintainability

3.5.5 Portability

4 Formal anlaysis using Alloy

5 Effort spent

Guglielmi Leonardo

- 11/11/2025 1h (RASD document structure)

Lo Conte Francesco

- 17/11/2025 4h (Completing Section 1 (Introduction))

6 References