M.E.N.A.C.E.

Motorized Embedded Network-connected smArt Car Entity

DIT524 - PROJECT:
SYSTEMS DEVELOPMENT

Date 2017-02-24

Isak Magnusson
Kosara Golemshinska;
Laiz H. B. de Figueroa;
Melinda Ivók;
Nina Uljanić;
Rema Salman;
Syeda Elham Shahed.

REQUIREMENT SPECIFICATION

Introduction

The purpose of this document is to give an exhaustive description of the requirements for the M.E.N.A.C.E. SmartCar project.

The system itself consists of two major components: the embedded software for the SmartCar and the Android mobile application that will serve as a remote controller for the car using Bluetooth connection. With this pairing of the two devices, a user would be able to direct the car which will autonomously blink its lights when completing a turn, as well as avoid any obstacles in its path. The latter will be resolved by the car stopping and prompting the user for further instructions. An additional functionality would be a fully autonomous mode in which the car follows a pointed light on the ground while simultaneously avoiding the obstacles by circumventing them.

Domain

The stakeholders involved in this project could be considered as following:

- Ivica Crnkovic, Emil Alégroth, Gul Calikli, and Michal Palka as product owners;
- Andres Ricardo De Biase and Monica-Alexandra Murgescu, as advisors and product owners assistants;
- Dimitris Platis and Ioannis Gkisas, as technical advisors;
- 7 students, as developers.
- Users can be considered anyone that want to see how the product behaves and/or wants to play with the car along the process or in the end.

The system to be produced has two domains identified, being the control (Figure 1) and the car domain (Figure 2). They will have different interaction with the user in each scenario as shown in the figures cited above.

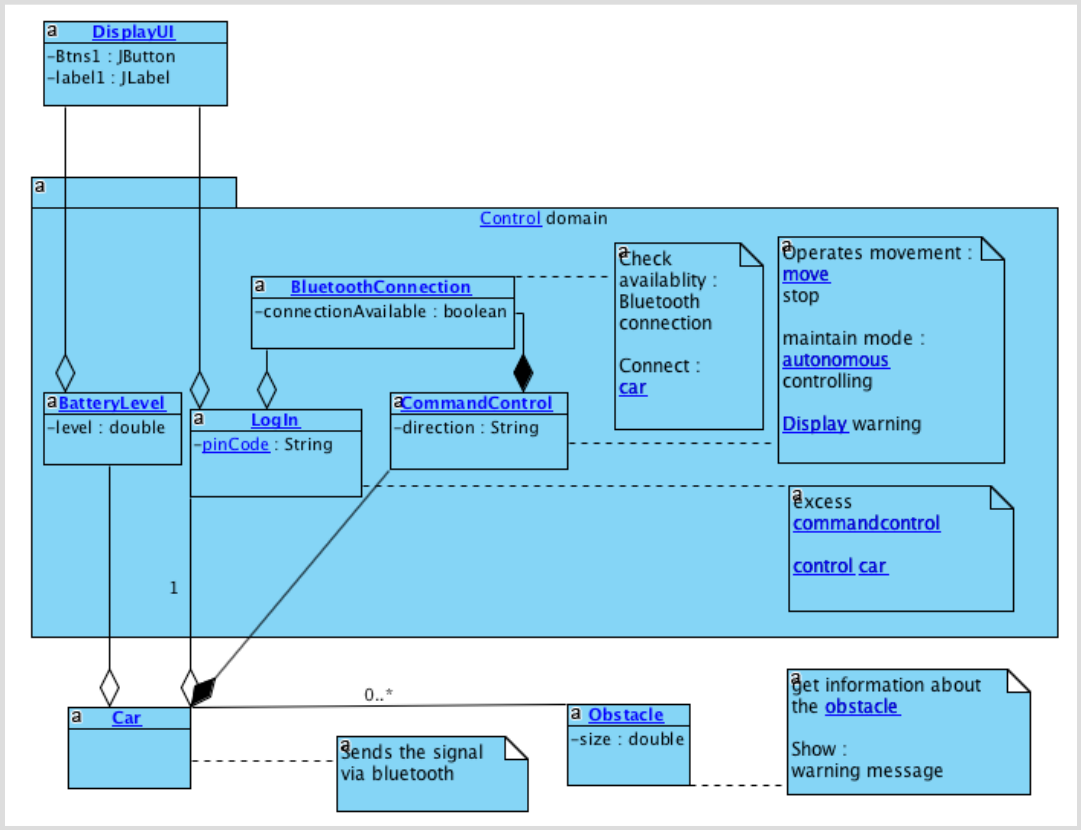

Figure 1. The control domain and its interaction in the system (Author: Syeda; exported from Visual Paradigm).

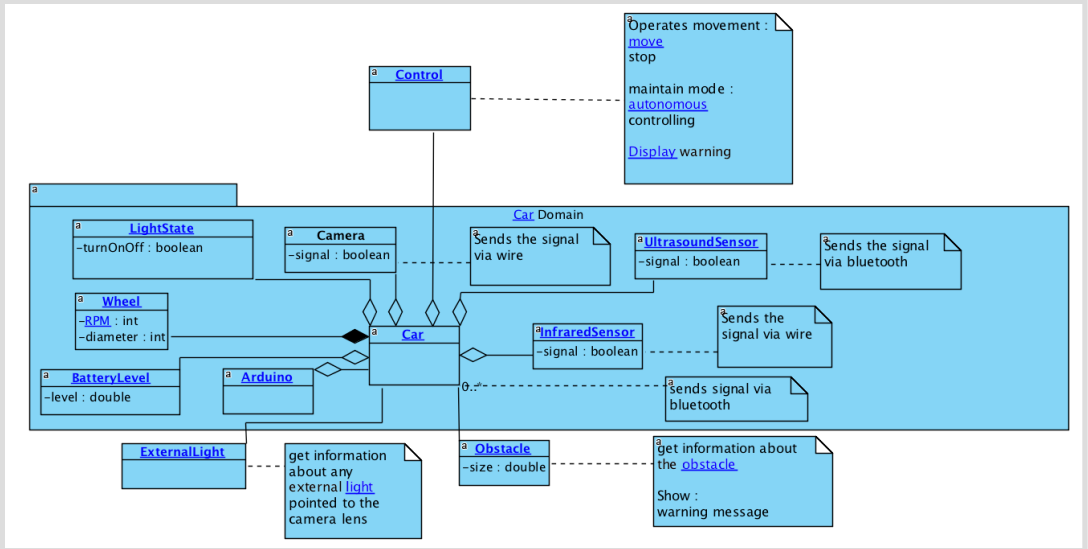

Figure 2. The car domain and its interaction in the system (Author: Syeda; exported from Visual Paradigm).

Requirements

Use Cases:

The team has identified seven cases with different scenarios and flows: Log in, Move Car, Avoid Obstacles, Blink Lights, Connect Car, Reset Car, Follow Light. The cases are described in the Appendix I at the final of this document with the sequence diagrams for each use case.

User Stories:

Trello (available on: www.trello.com) was used as a tool to manage the user stories created by the team, the figures 3-7 were extracted from this tool. The tasks listed in each user story were estimated using planITpoker tool (available on: http://www.planitpoker.com), the results generated is shown on Table 1.

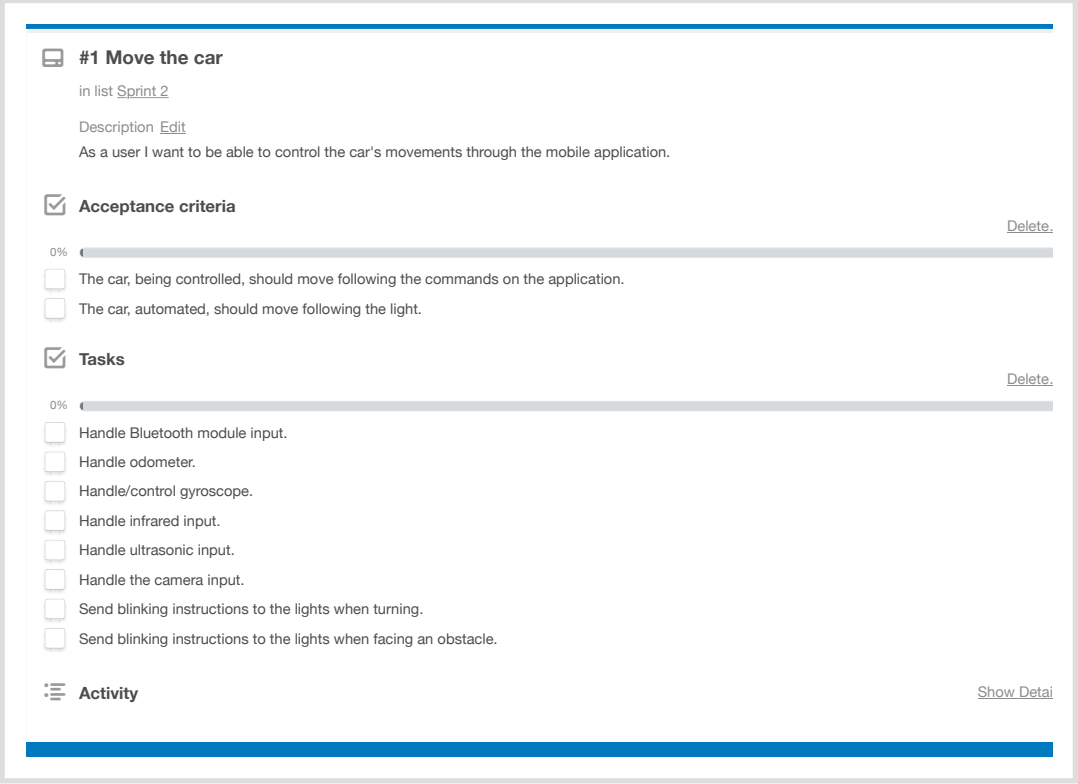

Figure 3. Move the car user story with its acceptance criteria and tasks (Exported from Trello).

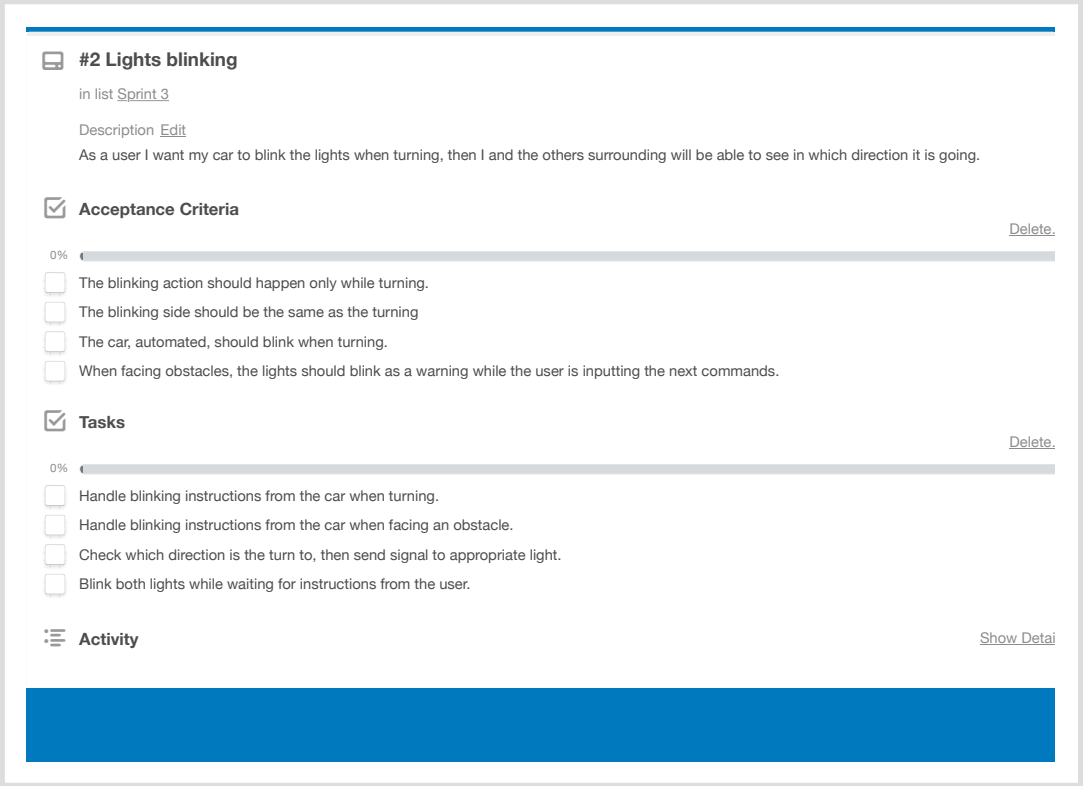

Figure 4. Lights blinking user story with its acceptance criteria and tasks (Exported from Trello).

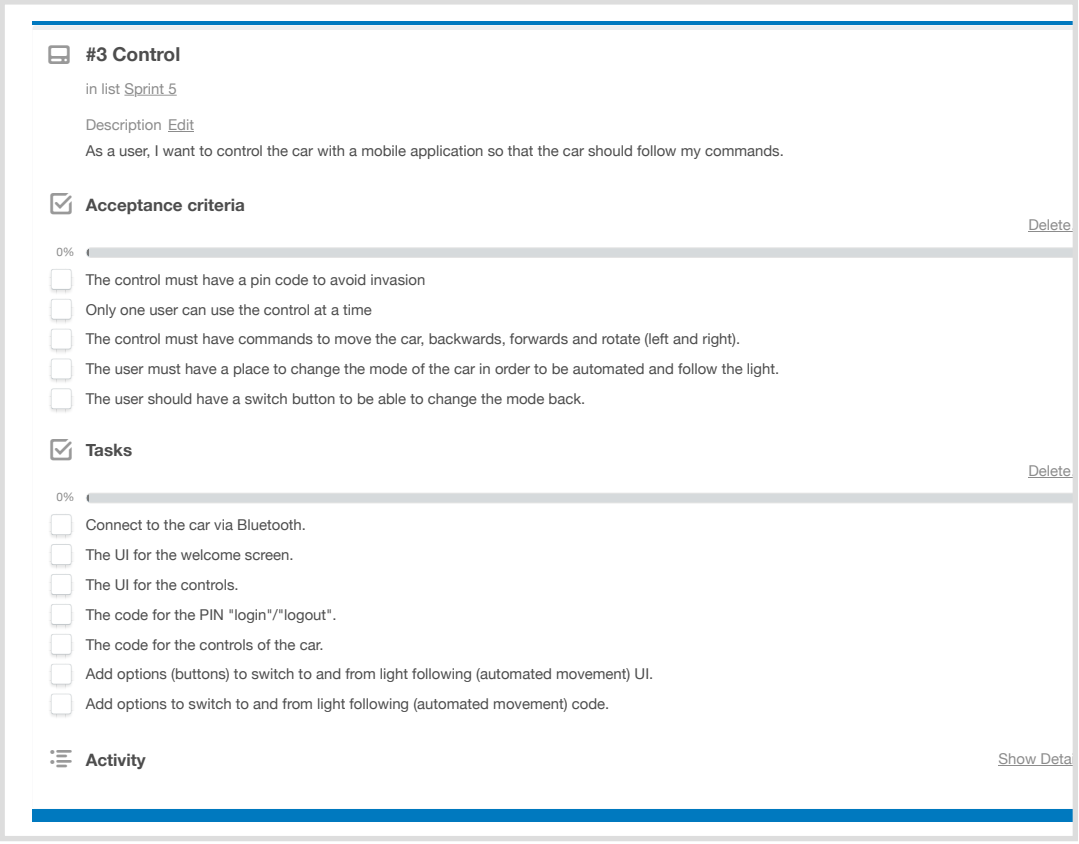

Figure 5. Control user story with its acceptance criteria and tasks (Exported from Trello).

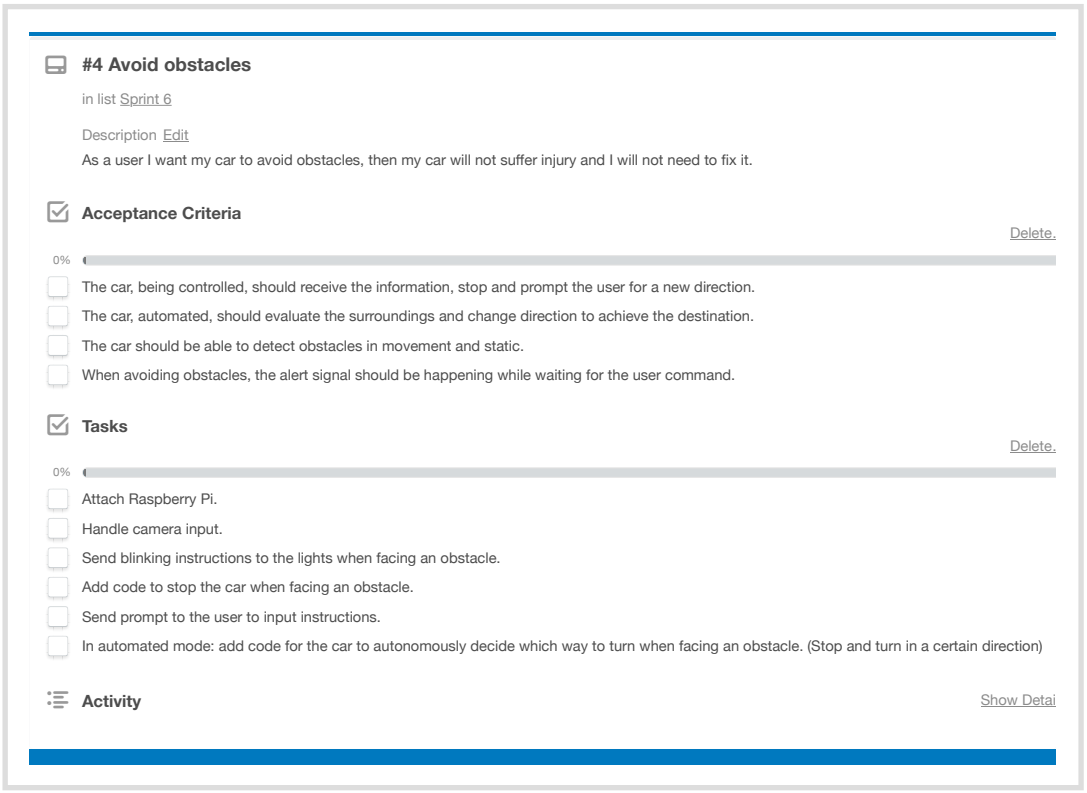

Figure 6. Avoid Obstacle user story with its acceptance criteria and tasks (Exported from Trello).

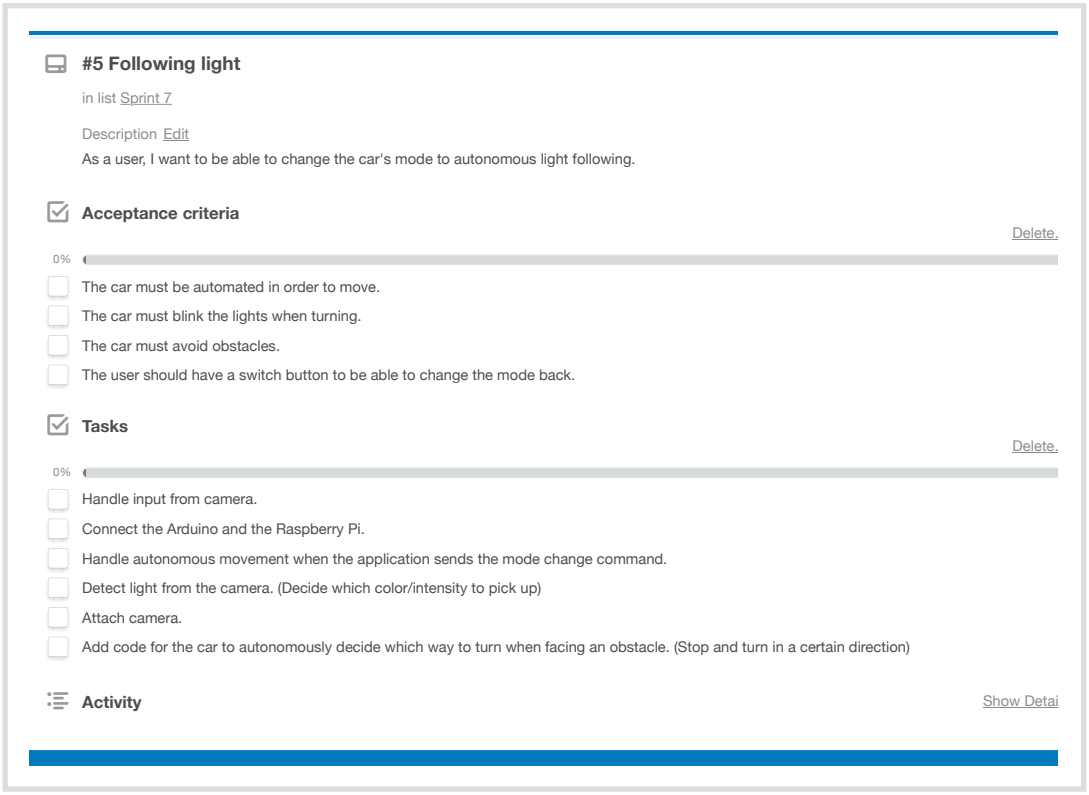

Figure 7. Following Light user story with its acceptance criteria and tasks (Exported from Trello).

M.E.N.A.C.E.

Motorized Embedded Network-connected smArt Car Entity

Table 1. User stories' effort and hour estimation for the team (adapted by Laiz).

Story Title	Effort Estimation	Hour Estimation
#1 Move the car: Handle Bluetooth module input.	40	24
#1 Move the car: Handle infrared input.	20	16
#1 Move the car: Handle odometer.	20	16
#1 Move the car: Handle the camera input.	40	24
#1 Move the car: Handle ultrasonic input.	40	24
#1 Move the car: Handle/control gyroscope.	13	12
#1 Move the car: Send blinking instructions to the lights when facing an obstacle.	8	8
#1 Move the car: Send blinking instructions to the lights when turning.	20	16
#2 Lights blinking: Blink both lights while waiting for instructions from the user.	3	6
#2 Lights blinking: Check which direction is the turn to, then send signal to appropriate light.	20	16
#2 Lights blinking: Handle blinking instructions from the car when facing an obstacle.	13	12
#2 Lights blinking: Handle blinking instructions from the car when turning.	13	12
#3 Control: Add options (buttons) to switch to and from light following (automated movement) UI.	13	12
#3 Control: Add options to switch to and from light following (automated movement) code.	40	24
#3 Control: Connect to the car via Bluetooth.	8	8
#3 Control: The code for the controls of the car.	100	5
#3 Control: The code for the PIN "login"/"logout".	40	24
#3 Control: The UI for the controls.	20	16
#3 Control: The UI for the welcome screen.	20	16
#4 Avoid obstacles: Add code to stop the car when facing an obstacle.	40	24
#4 Avoid obstacles: Attach Raspberry Pi.	40	24
#4 Avoid obstacles: Handle camera input.	100	40
#4 Avoid obstacles: In automated mode: add code for the car to autonomously decide which way to turn when facing an obstacle. (Stop and turn in a certain direction)	100	40
#4 Avoid obstacles: Send blinking instructions to the lights when facing an obstacle.	13	12
#4 Avoid obstacles: Send prompt to the user to input instructions.	5	6

Story Title	Effort Estimation	Hour Estimation
#5 Following light: Add code for the car to autonomously decide which way to turn when facing an obstacle. (Stop and turn in a certain direction)	100	40
#5 Following light: Attach camera.	½	4
#5 Following light: Connect the Arduino and the Raspberry Pi.	40	24
#5 Following light: Detect light from the camera. (Decide which color/intensity to pick up)	100	40
#5 Following light: Handle autonomous movement when the application sends the mode change command.	40	24
#5 Following light: Handle input from camera.	100	40

Quality:

The external quality requirements aimed by the team to the system are:

* **Usability**

Can be divided as the following:

- The system's adopting and expectations: Any good technical system has always a good interaction design, which focuses on the user's goals. Furthermore, to create a usable system first step should be taken is understanding the goals in the context of the user's environment, task or work flow, and letting these needs introduced in the design.
- Evaluation: Usability relies on the user-feedback throughout the evaluation phases. During the usability evaluation (prototype) of a system, real people and not just the developers or designers use the system for testing it and feedbacks are taken for improve the system.
- User-centered design: Where the user is satisfied with the interface that all of the goals, mental models, tasks and requirements are met. The combination of analysis, design and evaluation all approached starting from the user's point of view creates usable systems.

✱ Reliability

A system, product or component performs specified functions under specified conditions for a specified period of time. Can be divided into:

- Maturity: a system, product or component meets needs for reliability under normal operation.
- Availability: a product or system is operational and accessible when required for use. (availability of a solution in terms of % out of the total execution time)
- Fault Tolerance: a system, product or component operates as intended despite the presence of hardware or software faults.
- Recoverability: degree to which, in the event of an interruption or a failure, a product or system can recover the data directly affected and re-establish the desired state of the system.

The internal quality requirements aimed by the team to the system are:

✱ Code Quality

The code will be overall structured and well documented to make it easier to read, understand, and to detect errors. The functionality of the code will be described in different places in the code itself, for instance, each class will consist of a header and preferably description of the class' functionality and the methods to be used will also be provided with a description.

The Code Quality is one of the most important internal qualities used in the system. However, by maintaining it as explained previously, will insure the re-usability, readability, testability, and understandability characteristics of the code.

✱ Compatibility

A product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment. Can be divided into:

- Co-existence: A product can perform its required functions efficiently while sharing a common environment and resources with other products, without detrimental impact on any other product.

- Interoperability: Two or more systems, products or components can exchange information and use the information that has been exchanged.

Focusing on:

- Hardware Compatibility: the product can be used with applicable configurations of hardware components. Since our car has two main hardwares (Arduino and the smartphone).
- Operating System Compatibility: the product can run on intended operating system versions, and follows typical behavior. We have two systems (Car and Control) that will be operating in different environments (C++ and Java, respectively).
- Sustainability: effects on the environment, e.g. energy efficiency, switch-offs, power-saving modes, telecommuting. The car system must save battery.
- Standards Conformance: the product conforms to applicable standards, regulations, laws or ethics. Since the project will perform on the university environment, the standards that it should follow is the ones provided by the course and by the team.

Prioritization

The group created the product backlog, shown on the user stories above and made the estimation for each task, on Table 1 above. Since this project is split in 7 sprints, the first estimation and prioritization can have some changes along the process, as allowed by the Agile methodology applied in the development process. At the beginning of each sprint a meeting for planning should be organized to see if the sprint backlogs are updated or if any change is required. The group has decided to develop the mobile application at the same time as the car's features development. The higher and crucial tasks are going to be done in the earlier sprints following this order, for the car:

- Move Car;
- Blink Lights for the Move Car;
- Avoid Obstacles;
- Blink Lights for the Avoid Obstacles;
- Follow Light;

- Avoid Obstacles for the Follow Light.

And following this order, for the mobile application:

- Control with buttons to move the car;
- Control with possibility to answer prompts;
- Control with pin code for protection;
- Control with good interface.

The group velocity for Sprint 1 to 3 were estimate as 0.6, since the group has 3 others courses with exams and deadlines. For Sprint 4 and 5 the velocity is expected to be around 0.8 and for the last two sprints the group expects to be in its full velocity 1.0, when no other courses would be influencing the deliveries. The Sprints backlogs will be decided along the sprint planning meeting.

Appendix I

M.E.N.A.C.E (Group 9)

Use Case Specification: Log in

Version <1.1>

Laiz Heckmann Barbalho de Figueroa

Revision History

Date	Version	Description	Author
2017-02-02	1.1	Fixed the whole document to introduce pin code instead of a user account	Syeda Elham Shahed

Table of Contents

Use Case Specification: Log in

1. Log in

1.1. Brief Description

This use case, in order to access the control of the car, allows a user to login on the system. The system will prompt the user to do the login when the system is initialized. The user will select to do the login, being asked by the system for a pin code. The system will check if the input done by the user matches with the predefined pin code. After that, the system will allow the user to access the control.

2. Flow of Events

2.1. Basic Flow

The user will open the application. The system will prompt the user to do the login. The user will choose to do the login. The system will ask the user for the pin code. The user will input the pin code and submit it. The system will verify it and give the user access to the car's control.

2.2. Alternative Flows

2.2.1. First Alternative Flow

When the system prompts the user for the information if the user input a wrong pin code, a message will be shown to the user asking for a valid pin code in order to continue the process.

2.2.2. Second Alternative Flow

When the system attempt to validate the pin code inputted by the user and there is no connection with the internet, the system will give a message to the user asking for he/she to verify the connection before continuing.

3. Special Requirements

The performance of the system will be done with a high speed data validation. The interface will facilitate the use of it and allow more interactivity by the user.

A bluetooth function and internet connection need to be available in order to the system to perform.

3.1. System with high speed response

3.2. Friendly interface

3.3. Interactive interface

3.4. Bluetooth and Wi-Fi functions availability

4. Preconditions

4.1. Application open

4.2. Internet connection available

5. Post Conditions

5.1. Access to the car's command controller

5.2. Bluetooth connection

6. Scenarios

6.1. Successful internet connection

Basic flow
First Alternative Flow

6.2. Successful first attempt to log in

Basic flow

6.3. Failed first attempt to log in

First Alternative Flow

6.4. Failed internet connection

Second Alternative Flow

Use Case: LogIn

The System Sequence Diagrams (SSD) for both domains, control (Fig. 1) and car (Fig. 2).

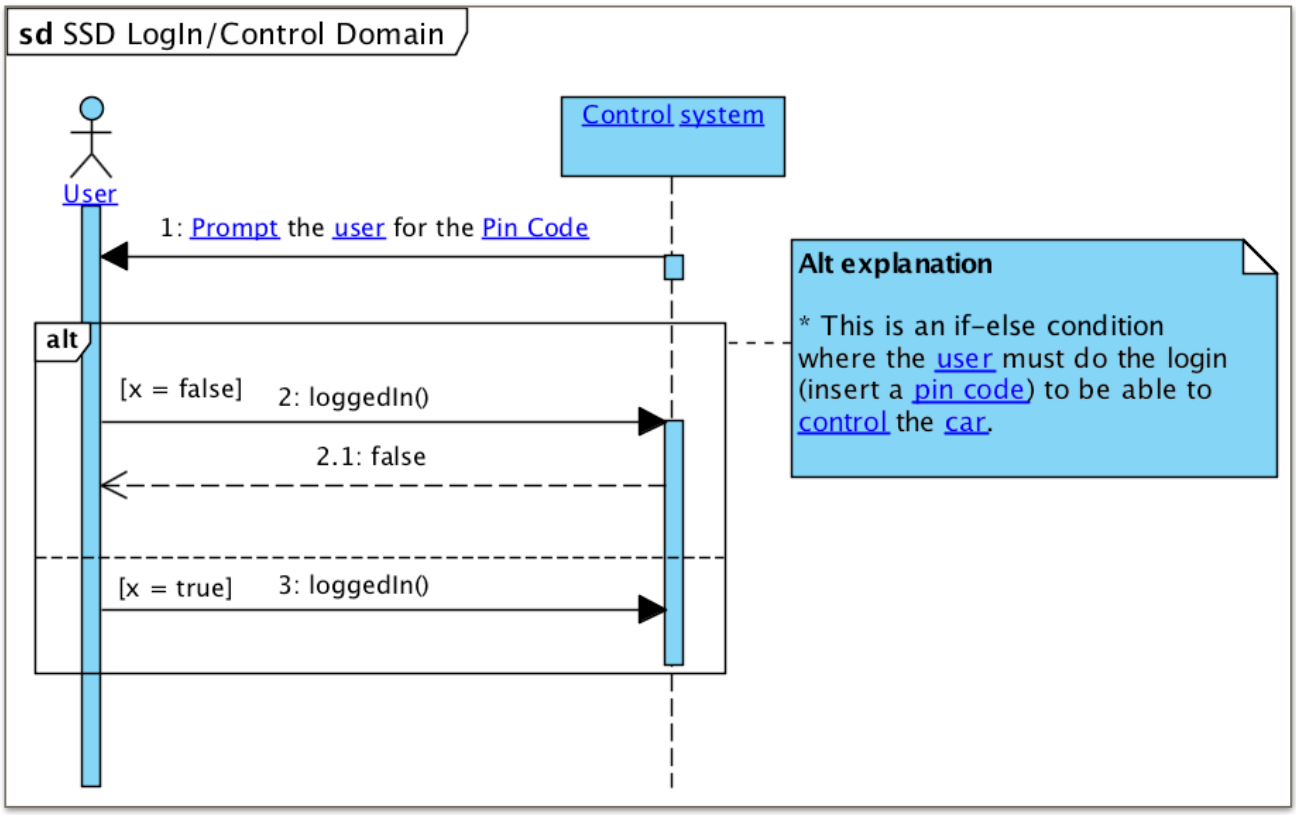

Figure 1. LogIn SSD in the Control domain view (Author: Syeda).

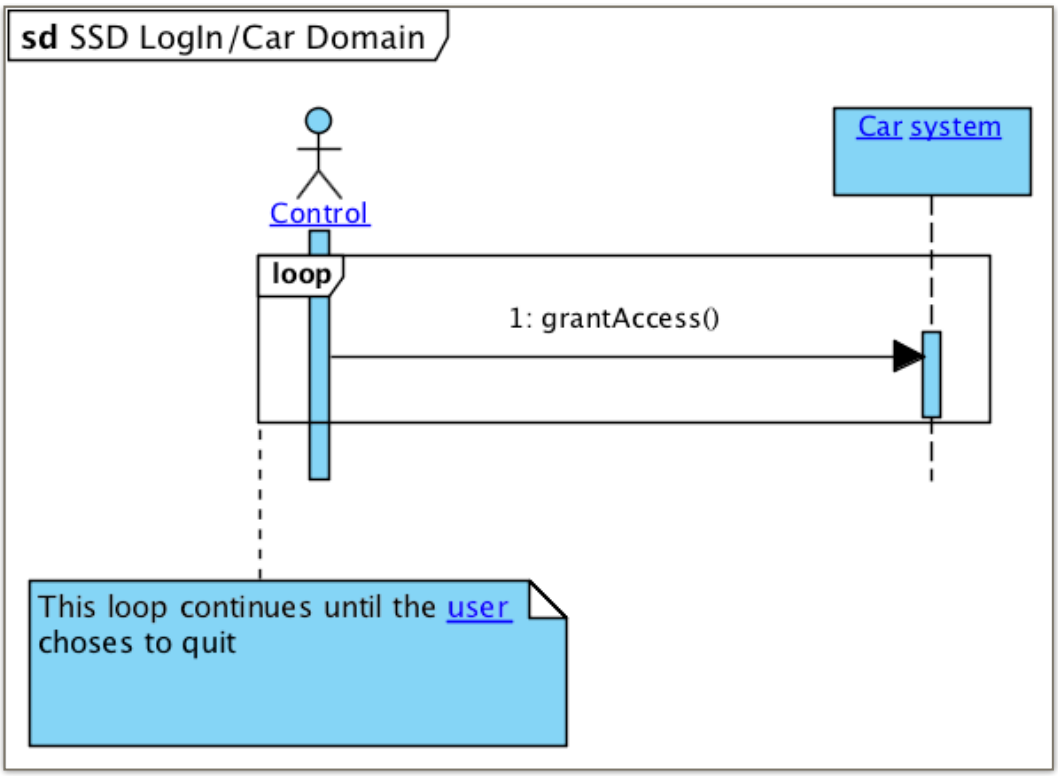

Figure 2. LogIn SSD in the Car domain view (Author: Syeda).

M.E.N.A.C.E (Group 9)

Use Case Specification: Connect Car

Version <2>

Nina Uljanic

Revision History

Date	Version	Description	Author
02/02/2017	V.1.1	Add the first alternative flow; Add some Special Requirements; Add most of the Conditions; Add Extension point; Add the scenarios.	Laiz H. B. de Figueroa
21/02/2017	V. 2	Added the steps to the basic flow. Added all subflows.	Kosara Golemshinska

Table of Contents

Use Case Specification: Connect Car

1. Create Account

1.1. Brief Description

In order for the user to be able to control the car, the connection between the car and the application must be established.

2. Flow of Events

2.1. Basic Flow

2.1.1. Application launch

The use case begins when the user launches the application in order to access the car controls.

2.1.2. Log in

The user logs into the application.

2.1.3. Car connection

After the user is logged in, the application is connected to the car and the latter can now receive directional commands.

2.1.4. Commands transmission

The user sends directional commands to the car through the application.

2.1.5. Log out

As soon as the user decides to terminate the session, they can log out of the application.

2.1.6. Connection termination

After the log out, the connection to the car is terminated. The use case ends.

2.2. Alternative Flows

2.1.1. First Alternative Flow

If the connection fails while attempting to connect to the car, the application will prompt the user for verify the wireless connection to the car in order to proceed with the session.

2.1.2. Second Alternative Flow

If the connection fails while the application is attempting to reach the car, the application will prompt the user for verify the wireless connection in order to successfully connect to the car.

2.1.2.1. First alternative sub-flow

If the problem with the connectivity continues after the checks, the application will prompt the user to **reset** the car in order to successfully connect to the car

2.3. Subflows

2.3.1. Application launch subflow

2.3.1.1. The user launches the application.

2.3.1.2. The welcome screen is displayed with the options to log in or create account.

2.3.2. Log in subflow

2.3.2.1. The user selects the log in option.

2.3.2.2. The user inputs their username and password.

2.3.2.3. The user's input is verified against the existing user accounts.

2.3.3. Car connection subflow

2.3.3.1. If the user's input is correct, the application attempts to connect to thecar.

2.3.3.2. The user is notified if the connection is successful.

2.3.4. Commands transmission subflow

2.3.4.1. The application displays the available controls.

2.3.4.2. The user inputs a directional command.

2.3.4.3. The user's input is sent to the car for execution.

2.3.5. Log out subflow

2.3.5.1. The user selects the log out option from the context menu of the application.

2.3.5.2. The application prompts the user for confirmation of the log out.

2.3.5.3. The log out is completed.

2.3.6. Connection termination subflow

2.3.6.1. The connection to the car is terminated.

2.3.6.2. The application notifies the user the connection is no longer available.

3. Special Requirements

The interface must be responsive and allow for easier interactivity.

A bluetooth function and internet connection must be available in order for the system to run properly.

3.1. Friendly interface

3.2. Interactive interface

3.3. Bluetooth and Wi-Fi functions availability

4. Preconditions

4.1. Application open

4.2. User Logged in

4.3. Internet connection

4.4. Bluetooth connection

4.5. Battery charged

5. Post Conditions

5.1. Control the car

6. Extension Points

6.1. **Inclusion: Log in Account**

To be able to establish the connection with the car, the user must be logged in.

7. Scenarios

7.1. **Successful Connection**

Basic Flow

7.2. **Failed Connection**

First Alternative Flow
Second Alternative Flow

7.3. **Car reset necessity**

First alternative subflow

Use Case: Connect Car

The System Sequence Diagrams (SSD) for both domains, control (Fig. 1) and car (Fig. 2).

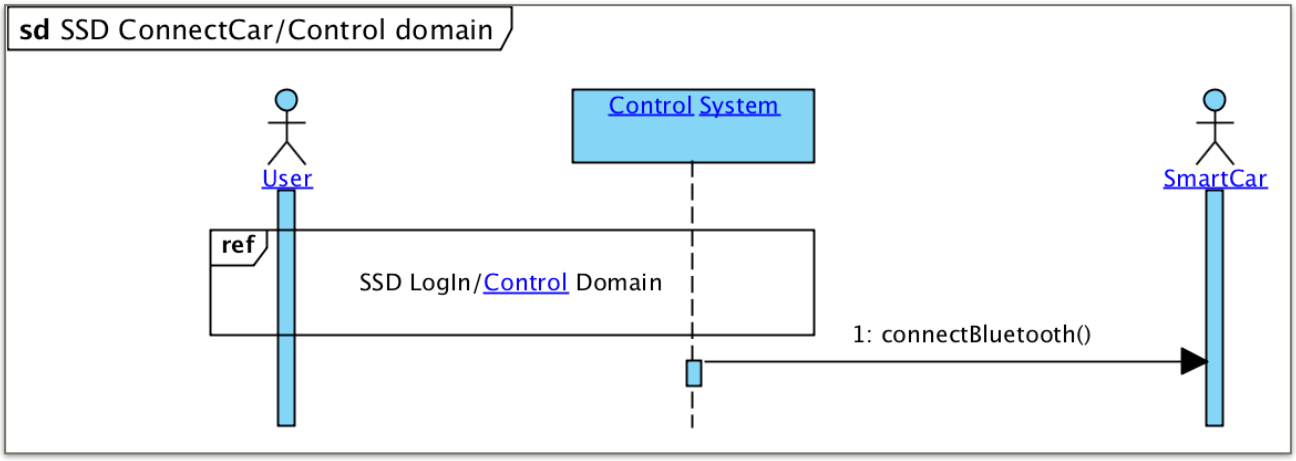

Figure 1. Connect Car SSD in the Control domain view (Author: Kosara).

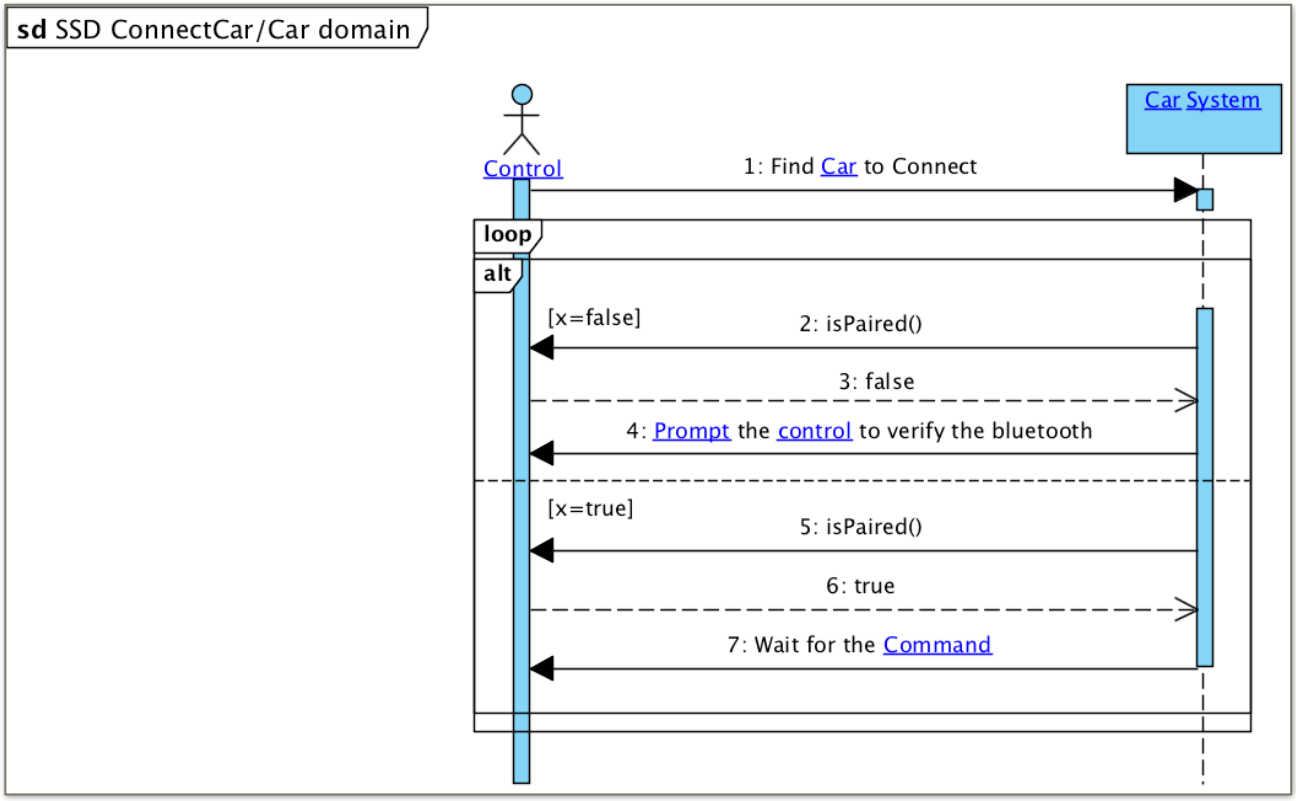

Figure 2. Connect Car SSD in the Car domain view (Author: Kosara).

M.E.N.A.C.E (Group 9)

Use Case Specification: Move Car

Version <2.0>

Melinda Ivók

Revision History

Date	Version	Description	Author
19/02/2017	2.0	Added stop car on the Brief Description; Changed Log in with password and username to pin code in all the document.	Laiz Figueroa

Table of Contents

Use Case Specification: Move Car

1. Move Car

1.1. Brief Description

This use case allows the car user to move the car in any direction and to stop the car. The controller, available as a mobile application, provides buttons to choose from going forwards, backwards, turning left or right and to stop the car. It is connected to the car through Bluetooth.

2. Flow of Events

2.1. Basic Flow

PLACE CAR
This use case starts with the car user placing the car on a big, flat surface so it can move freely, and somewhere safe so the car does not fall down.

START APPLICATION
The user starts the application on their mobile phone.

LOG IN
The user logs in by providing the pin code.

WAIT FOR CONNECTION
The control connects to the car through the Bluetooth. The user waits for it to be established.

PRESS BUTTONS
The control displays the buttons which the user selects from. These buttons are: move forwards, move backwards, turn left, turn right, and stop. The user presses continuously to move the car.

2.2. Alternative Flows

2.2.1. First Alternative Flow

CANNOT PLACE CAR
At BF PLACE CAR, the user does not find a suitable surface to place the car on. The user case ends immediately to not risk the safety of the car.

2.2.2. Second Alternative Flow

APPLICATION FREEZES
When the application freezes at any point, the user restarts it.

2.2.3. Third Alternative Flow

CANNOT LOG IN
AT BF LOG IN, the pin code given is not accepted. The user retries.

2.2.4. Fourth Alternative Flow

CANNOT ESTABLISH CONNECTION
AT BF WAIT FOR CONNECTION, when the control can not connect to the car, the application is restarted.

2.2.5. Fifth Alternative Flow

MOVE TOWARDS OBSTACLE
AT BF PRESS BUTTONS, the car determines that there is an obstacle in the way, therefore the car stops and waits for further instructions.

2.2.6. Sixth Alternative Flow

CANNOT DECIDE DIRECTION
AT BF PRESS BUTTONS, when the user presses buttons having the opposite directions at the same time, the car stops and waits for an unambiguous command.

3. Special Requirements

The mobile application's user-friendly interface will facilitate the use of it and allow more interactivity by the user to control the car.

Bluetooth and internet connections need to be available in order to the system to perform.

3.1. User-friendly interface

3.2. Interactive interface

3.3. Bluetooth and internet functions' availability

4. Preconditions

4.1. Application

4.2. User Logged in

4.3. Internet connection

4.4. Bluetooth connection

4.5. Battery charged

5. Post Conditions

5.1. Car and control connected

5.2. Access to the car's command control

5.3. Car is moved from its original placement

6. Extension Points

6.1. Inclusion: Log in

To access the control and move the car the user must complete the log-in inserting the pin code.

7. Scenarios

7.1. Successful connection between car and control

Basic flow

7.2. Failed attempt to find a suitable environment

First Alternative Flow

7.3. Failure in connection

Second Alternative Flow
Third Alternative Flow
Fourth Alternative Flow

7.4. Problems in the commands given

Fifth Alternative Flow
Sixth Alternative Flow

Use Case: Move Car

The System Sequence Diagrams (SSD) for both domains, control (Fig. 1) and car (Fig. 2).

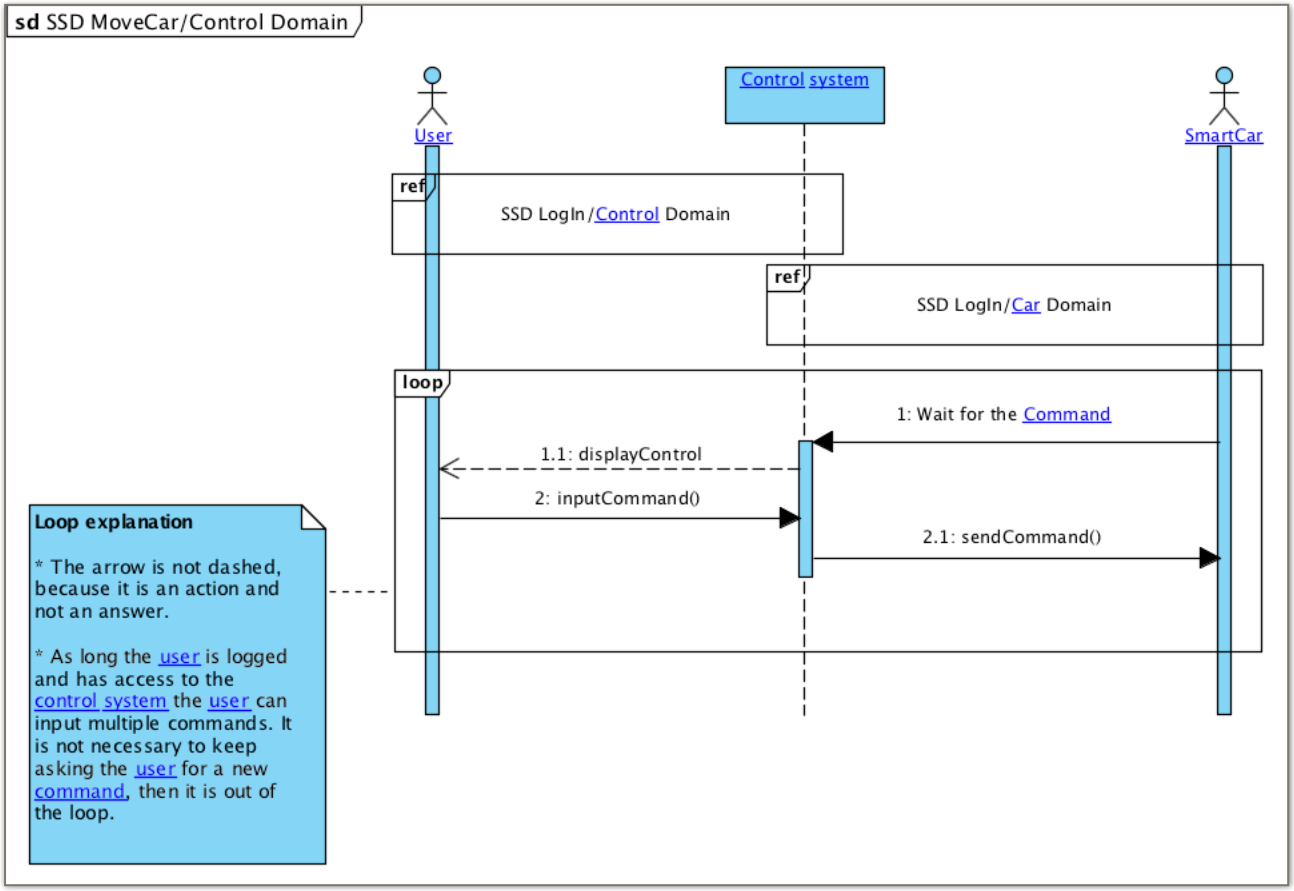

Figure 1. Move Car SSD in the Control domain view (Author: Laiz).

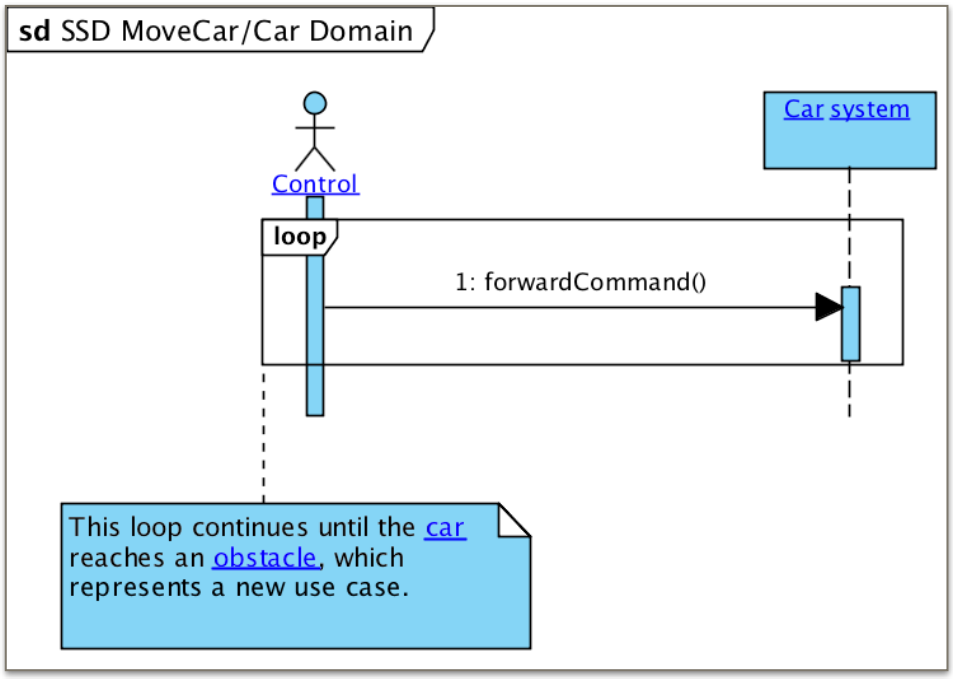

Figure 2. Move Car SSD in the Car domain view (Author: Laiz).

M.E.N.A.C.E (Group 9)

Use Case Specification: Avoid obstacle

Version <2.0>

Syeda Elham Shahed

Revision History

Date	Version	Description	Author
2017-02-02	1.1	Add Preconditions	Rema Salman
2017-02-20	2.0	Updated preconditions. Updated flow of events.	Nina Uljanic

Table of Contents

Use Case Specification: Avoid obstacle

1. Avoid obstacle

1.1. Brief Description

This use case allows a car to avoid any obstacle ahead of it. The car registers the obstacle and sends a notification to the application, which displays the warning to the controller. The controller can determine subsequent actions accordingly.

2. Flow of Events

2.1. Basic Flow

2.1.1 First Basic Flow

This use case starts when controller logs in to the mobile application. The application shows available features. The controller chooses the 'Move' operation. The remote control retrieves all available options for 'Move' operation. The controller chooses the 'Go front' option from the remote control. The car moves to the front direction.

2.1.2 Second Basic Flow

The car stops if there is an obstacle in front of it. The car reports to the application/system about any obstacle in front of it (car). The controller is notified. It (car) doesn't move until the controller chooses new operation.

2.1.3 Third Basic Flow

The controller chooses a new option. The car moves to the specified direction and performs according to the newly chosen operation. The use case ends.

2.2. Alternative Flows

2.2.1. First Alternative Flow

At 2.1.1, the controller sees an obstacle and chooses new action for the car to execute, before the car senses the obstacle; the car changes the direction of movement or stops. The use case resumes at 2.1.3.

2.2.2. Second Alternative Flow

At BF 2.1.1, the controller chooses either go left, go right or go back. The car moves according to the chosen option. The use case resumes at 2.1.2.

2.2.3. Third Alternative Flow

At 2.1.2, the car faces obstacle for each of the chosen options by the controller. The controller is required to manually solve the problem. The use case ends.

2.2.4. Fourth Alternative Flow

At 2.1.3, the car faces new obstacle or the same obstacle again. The use case resumes at 2.1.2.

2.3 Sub Flows

2.2.5. First Sub Flow

During the whole use case, if at any time the connection between the car and the application is lost, the system will prompt the controller to connect to Bluetooth. The use case resumes from 2.1.1 or 2.1.

2.2.6. Second Sub Flow

The controller chooses to reset the car any time in the use case. The use case resumes at 2.1.2.

2.2.7. Third Sub Flow

If the user gives the same direction four times consecutively and the car faces the same obstacle for that, the car will notify the system with an error message.

3. Special Requirements

3.1. Friendly interface

At any time the Car can notify about obstacle and the notification is visible in the application.

3.2. Bluetooth functions availability

A Bluetooth function and internet connection needs to be available for the system to work.

3.3. Avoiding obstacle: Size

Moves needed by the car to avoid obstacle, depends on the size and state (moving or stationary) of the obstacle.

4. Preconditions

4.1. Application Open

4.2. Mobile connected to the car via Bluetooth

4.3. Moving Car

5. Post Conditions

5.1. Car moves to other direction

6. Extension Points

6.1. Inclusion: Move Car

To confront an obstacle and avoid it, the car needs to be moving.

7. Scenarios

7.1. Successful first attempt to avoid obstacle

Basic flow

7.2. Avoid obstacle multiple times

Basic flow
Third Alternative Flow
Fourth Alternative Flow

7.3. Failed first attempt to avoid obstacle

First Alternative Flow
Fourth Alternative Flow

7.4. Failed Bluetooth connection

First Sub Flow
Second Sub Flow
Third Sub Flow

Use Case: Avoid Obstacles

The System Sequence Diagrams (SSD) for both domains, control (Fig. 1) and car (Fig. 2 and Fig. 3).

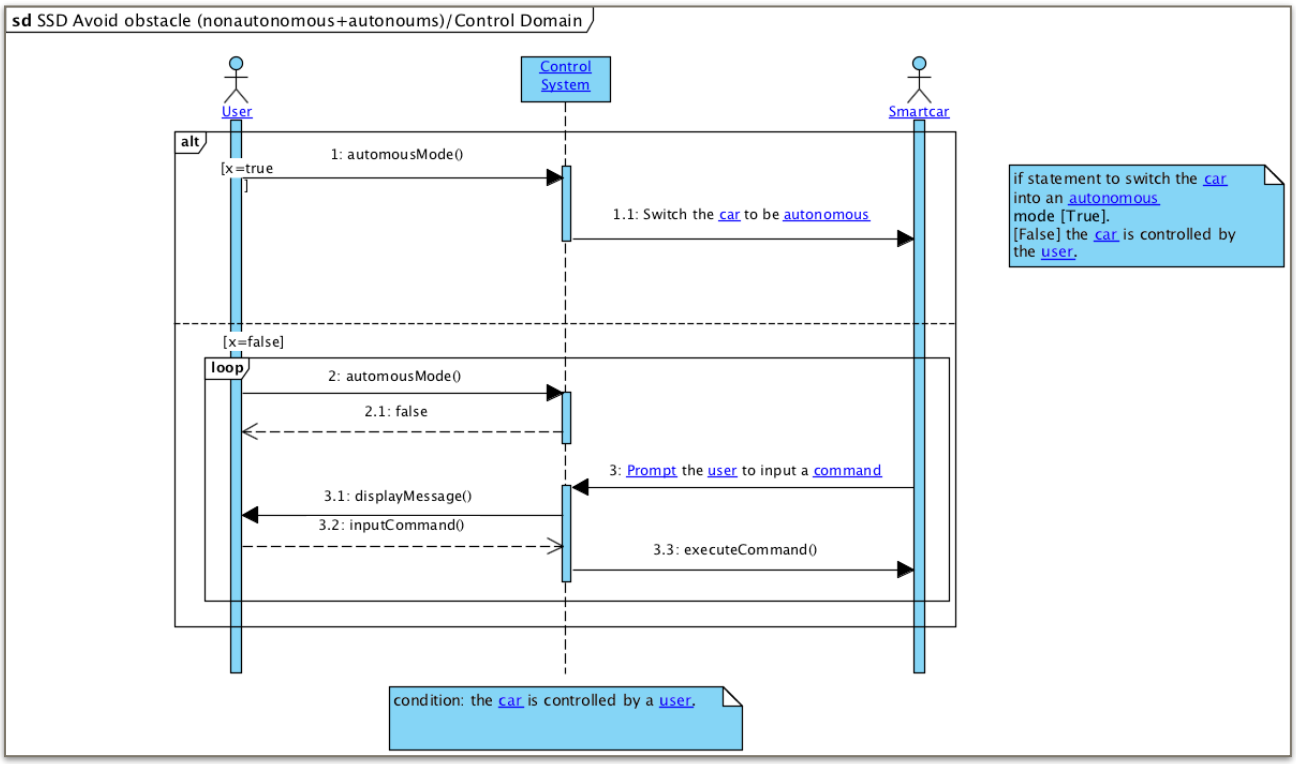

Figure 1. Avoid Obstacles SSD in the Control domain view (Author: Nina and Rema).

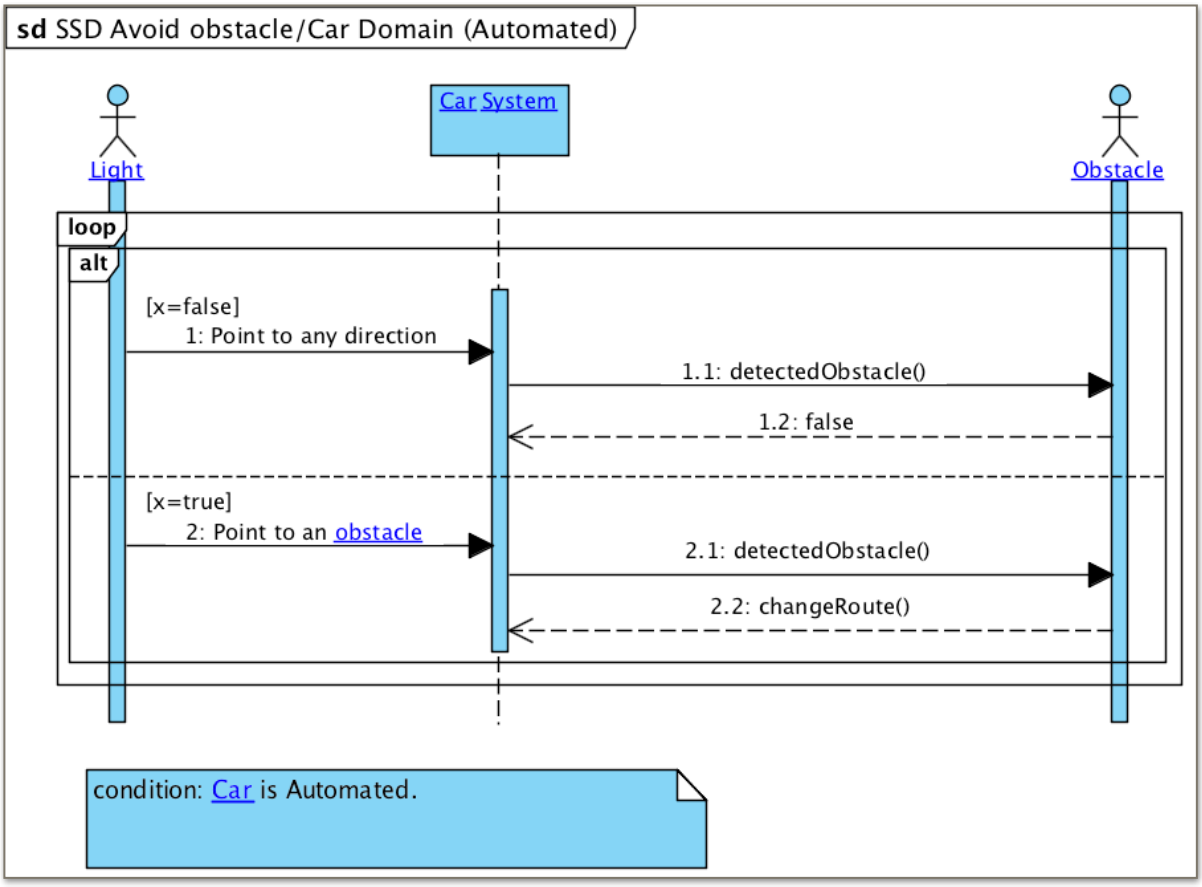

Figure 2. Avoid Obstacles SSD in the Car domain view when the car is autonomous (Author: Nina, Rema and Laiz).

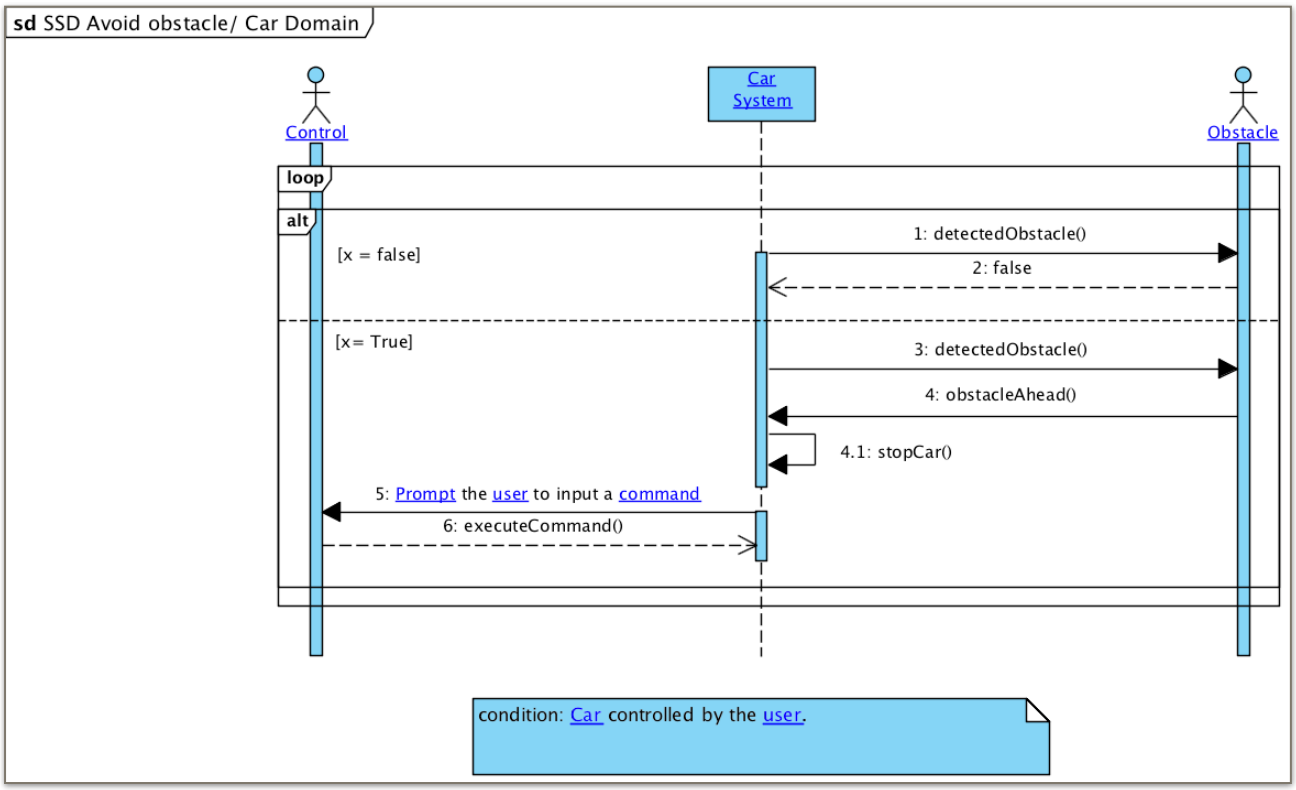

Figure 3. Avoid Obstacles SSD in the Car domain view when the car is being controlled (Author: Nina and Rema).

M.E.N.A.C.E. (Group 9)

M.E.N.A.C.E (Group 9) Use Case Specification: Follow a light

Version <2.0>

Revision History

Date	Version	Description	Author
23 Feb 2017	2.0	Added preconditions and postconditions	Isak Magnusson

Table of contents

1. Brief description

The car should be able to autonomously follow a pointed light on the floor in a dark environment when prompted by the user. Unlike the standard state of the car whereby it is linked to the mobile app and it receives all of its instructions from the user currently logged in, when following lights, the car should switch to an autonomous mode in which it relies on its proximity sensors to avoid obstacles and on its light sensors to detect if there is light present in the vicinity of the sensors and move towards that light. When the light source disappears or the user wishes to quit the session, the car should stop immediately and prompt the user for further instructions, i.e., if the user plans to continue using the light source to move the car, or if they prefer to resume the directing of the car manually. One final feature included would be to measure the distance travelled and the respective direction during the light following and send that information to the mobile app.

2. Basic flow

2.1. Change mode

The use case starts when the user selects “follow a light” as a mode option from the app menu. The user is then prompted to wait while the car shifts from taking directions from the app to autonomous light following. While in this mode, the car remains connected to the mobile app but the controller functions are disabled for the user.

2.2. Light detection

The car then engages its light sensor and checks for changes in the light levels in the area directly in front of it. If such are present, the car will move forward unprompted by the user. It then checks again after a small interval of time if the changes in light levels persist, and if so, moves forward again.

2.3. Route recording

Simultaneously with activating its light sensor, the car also starts recording the distance it travels and the direction in which it travels and continuously communicates this information to the mobile app.

2.4. Obstacle avoidance

While in “follow a light” mode, the car continually checks for obstacles on every move. If the proximity sensors detect a barrier under a certain threshold, the car stops and turns

in the direction of the light.

2.5. Exit mode

When the user selects the “stop” option from the app menu, the car stops moving and the system prompts the user to wait while the car reverts to the mode for taking directions. The car resumes its pre-light following connection to the controller.

3. Alternate flows

3.1. Low battery

If the car has low battery at any step in the basic flow, the user will be notified by an on-screen prompt and they won’t be able to proceed to the light following until the batteries are changed.

3.2. Lost connection

If at any step during the basic flow, the car loses connection with the app, the car will stop and the app will prompt the user to refresh their network connection and check that the car is still switched on.

3.3. No light detected

If the light sensor doesn’t detect any light at all during a certain timeframe, the user will be prompted to either make sure the light is close enough to the sensor, or quit the “follow a light” mode.

3.4. Light pointed at obstacle

If during LIGHT DETECTION, the light is pointed directly at an obstacle, the car will stop and prompt the user via the app to change the direction of the light. After a short interval, the car’s light sensor will check again if the light points away from the obstacle and if it does, it the move in that direction. If not, the car will prompt the user again using the app.

3.5. Light disappears

If during LIGHT DETECTION, the light source disappears, the car will stop and, after a short interval of time, prompt the user via the app to select one of two options: either to instruct the car to stand-by and continue engaging its light sensor to detect light, or to quit the “follow a light” mode and resume standard direction taking.

4. Subflows

4.1. Change mode

4.1.1. The subflow starts when the user selects the “follow a light” option from the menu.

4.1.2. The app prompts the user to wait while the car is switching its mode.

4.1.3. The car remains connected to the mobile app while switching.

4.1.4. The user controls are then disabled for the user so as to avoid conflicts between the car’s autonomous light following and the user’s instructions from the app.

4.2. Light detection

4.2.1. The car’s light sensor is switched on.

4.2.2. The car’s light sensor checks for changes in the light levels directly in front of it.

4.2.3. If there are any, the car moves towards the light source.

4.2.4. If there are none, the car will prompt the user to check if the light is close enough to the sensor, or else exit the light following mode.

4.2.5. The car continues to check for light sources at regular short intervals and to move towards them.

4.3. Route recording

4.3.1. The car’s “follow a light” mode is engaged.

4.3.2. The car’s odometer starts recording the distance travelled.

4.3.3. Parallel to the previous step, the car’s gyroscope starts keeping track of the direction traveled.

4.4.4. The car sends that route information to the app on regular intervals.

4.4. Obstacle avoidance

4.4.1. While in the light following mode, the car checks at regular intervals for any barriers around it using the data provided by its proximity sensors.

4.4.2. If the distance to such barriers is under a certain threshold, the car stops.

4.4.3. The car then checks again where the light source is located.

4.4.4. If the light is pointed at the barrier, the car prompts the user via the app to change the position of the light.

4.5. Exit mode

4.5.1. The user selects the “stop” option from the app menu.

4.5.2. The car stops moving.

4.5.3. The app prompts the user to wait.

4.5.4. The light sensor is switched off.

4.5.5. The user controls are enabled again.

5. Preconditions

5.1. A light on the floor

5.2. A dark environment

6. Postconditions

6.1. The regular controllers are activated

6.2. The car stops following the light

Use Case: Follow Light

The System Sequence Diagrams (SSD) for both domains, control (Fig. 1) and car (Fig. 2).

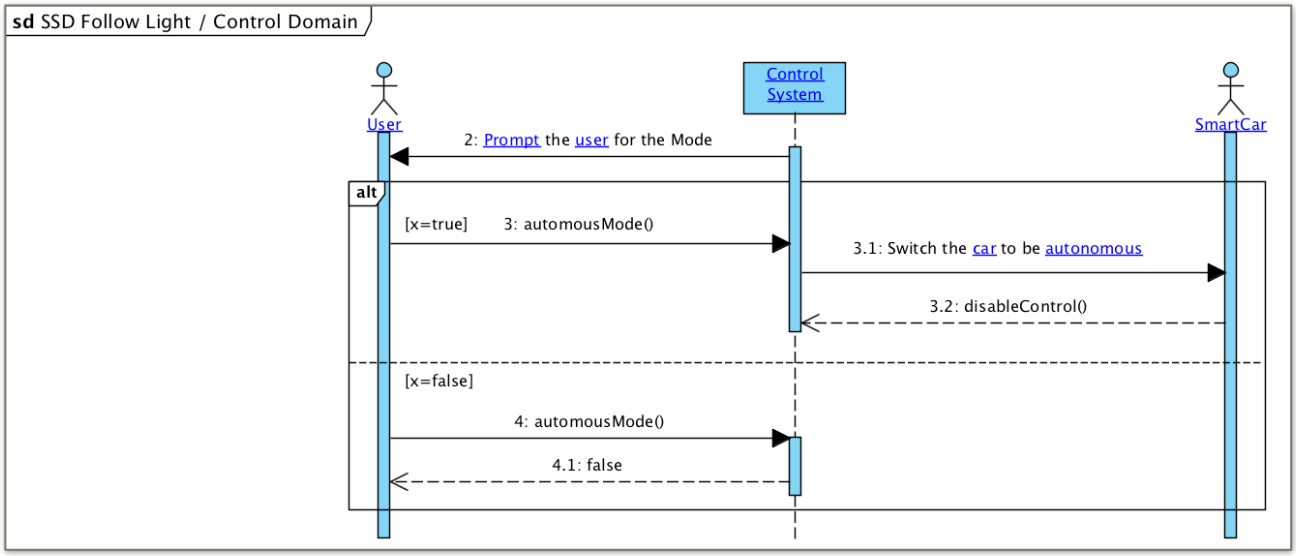

Figure 1. Follow Light SSD in the Control domain view (Author: Isak and Laiz).

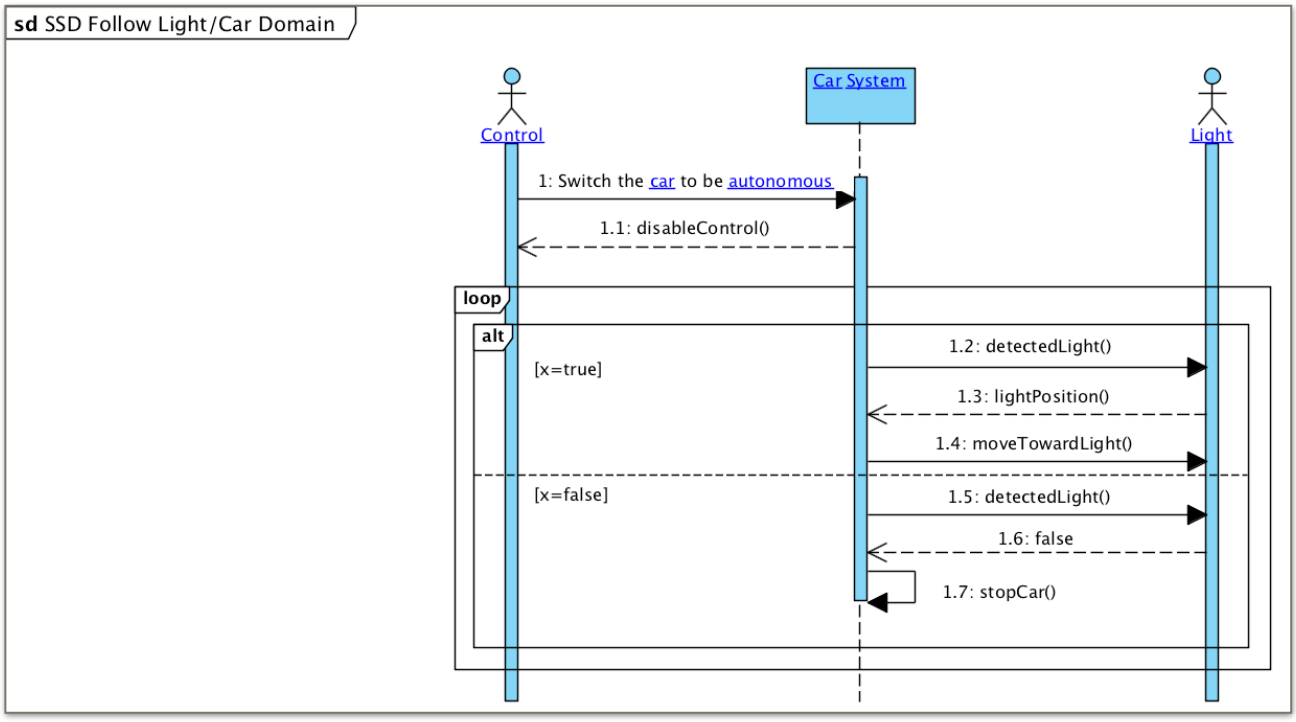

Figure 2. Follow Light SSD in the Car domain view (Author: Isak and Laiz).

M.E.N.A.C.E. (Group 9)

M.E.N.A.C.E (Group 9) Use Case Specification: Blink lights

Version <2.0>

Kosara Golemshinska

REVISION HISTORY

Date	***Version***	***Description***	***Author***
20 Feb 2107	*< 2.0 >*	- *Editing the number of the lights that, going to be provided as a hardware (**Brief description**)* - *Adding the other situation of blinking during, avoid obstacle use case.(**Brief description +Basic flow)*** *- Adding the points* ** 3. Special Requirements* **4. Preconditions* ** 5. Post Conditions*	*Rema Salman*

Table of contents:

1. Brief description

The car is equipped with 2 light diodes at its frontside corners. The car should automatically blink those diodes in 2 main situations.

- First one, while making a turn.This only applies to the mode in which the car is taking directions from the app.

- The second situation, when the car faces an obstacle. As explained in "Use case:Avoid Obstacle" the car will stop. Meanwhile, the diodes will blink until the user gets notified to change directions from the app.

The blinking should start as soon as the instruction for the turn is received and continue for several seconds so as to be noticed by an observer. Finally, which diodes blink depends on the direction in which the car is moving.

2. Flow of Events:

2.1 Basic flow

2.1.1. Change of route detected

The basic flow starts when the car engages in a sideways motion. If the sideways movement controls are used in the app, the basic flow can begin. The car then checks which direction the controls from the app have sent as an instruction and proceeds to the next step of the flow.

2.1.2. Blinking

Using the information from the previous step, the car then switches 2 of its diodes on and off for several seconds. Which diodes are engaged depends on the direction in which the car is moving, as well the car stopping situation of avoid obstacle use case, while the user gets the notification and gives new directions through the app.

2.2 Alternate flows

2.2.1. Consecutive turns

If during movement, the user directs the car to complete 2 or more consecutive turns and they are at intervals shorter than what a blinking signal would take, the blinking is queued and continues until the user stops directing the car to continue turning.

2.2.2. Going in circles

As a special case of the previous alternate flow, directing the car to move in a circular route would cause the blinking to continue for as long as the car receives instructions to move sideways

2.2.3. Obstacle avoidance

When detecting a barrier, the blinking is executed while the car stops and waits for further instructions. Typically, the car will then turn and start moving in that new direction. This also involves engaging the light diodes to alert the user that the car will turn.

2.3 Subflows

2.3.1 Change of route detected

- *The sideways motion controls are used in the app.*
- *The app checks which one is used in particular and sends that information to the car.*

2.3.2 Blinking

- The car uses the information from the app to decide which pair of light diodes to engage.
- *The light diodes are switched on and off at short regular intervals.*
- *The blinking stops after several seconds.*

3. Special Requirements

3.2. Bluetooth function availability

A Bluetooth function and connection needs to be available for the system to work.

3.3. Avoiding obstacle: Size

The blinking case to be done, after the car avoids an obstacle depends on the size and state (moving or stationary) of the obstacle.

4. Preconditions

- *Moving Car or moving car before facing the obstacle, in case of obstacle avoidance.*
- *Mobile connected to the car via Bluetooth*

5. Post Conditions

- The blinking for couple of seconds, to prompt the user.

Use Case: Blink Light

The System Sequence Diagrams (SSD) for both domains, control (Fig 1, Fig 2 and Fig 3) and car (Fig 4, Fig 5, Fig 6 and Fig 7), in both scenarios, automated and non-automated, in both use cases, Avoid obstacles and Move car.

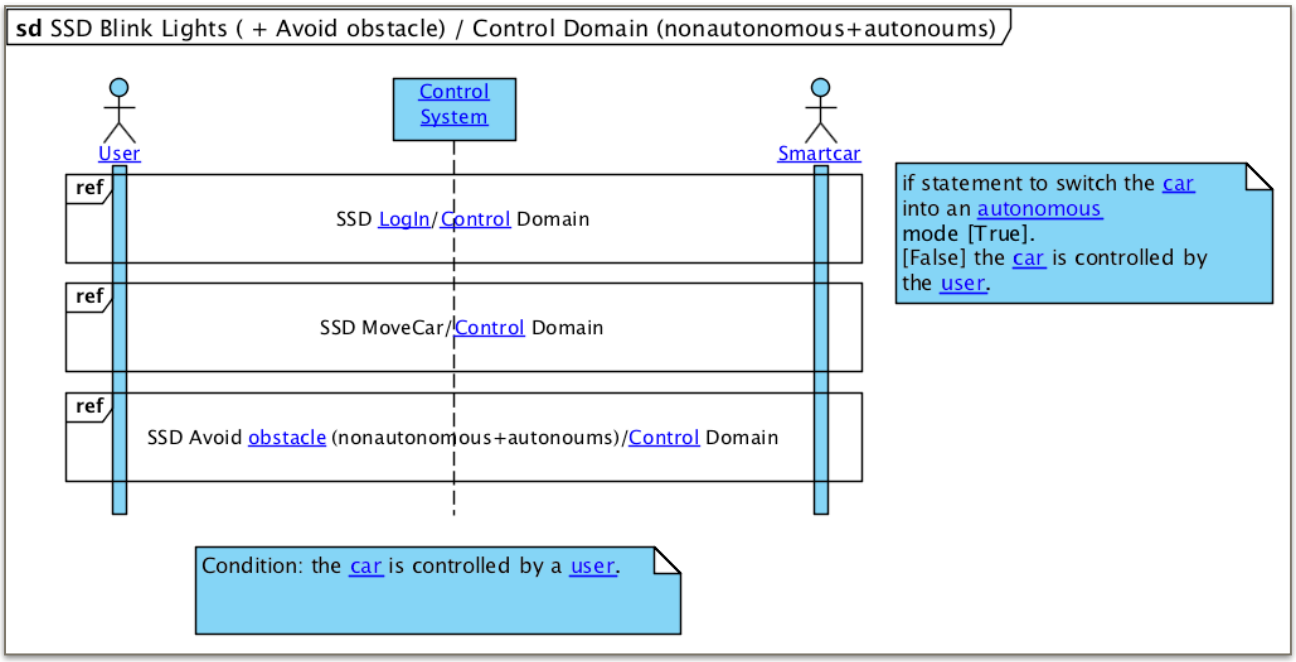

Figure 1. Blink Light SSD + Avoid Obstacle in the Control domain view for automated and non-automated scenarios (Author: Rema and Laiz).

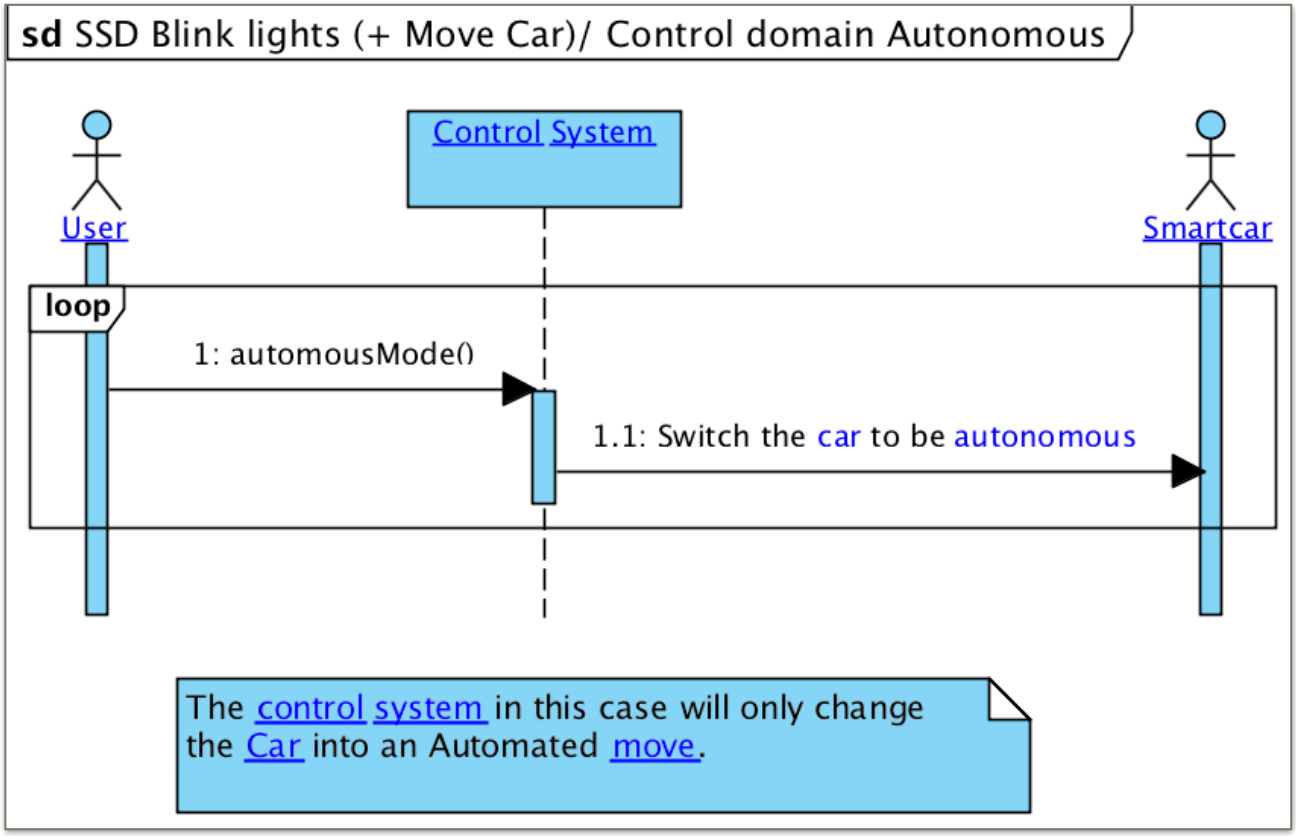

Figure 2. Blink Light SSD + Move Car in the Control domain view when the car is autonomous (Author: Rema).

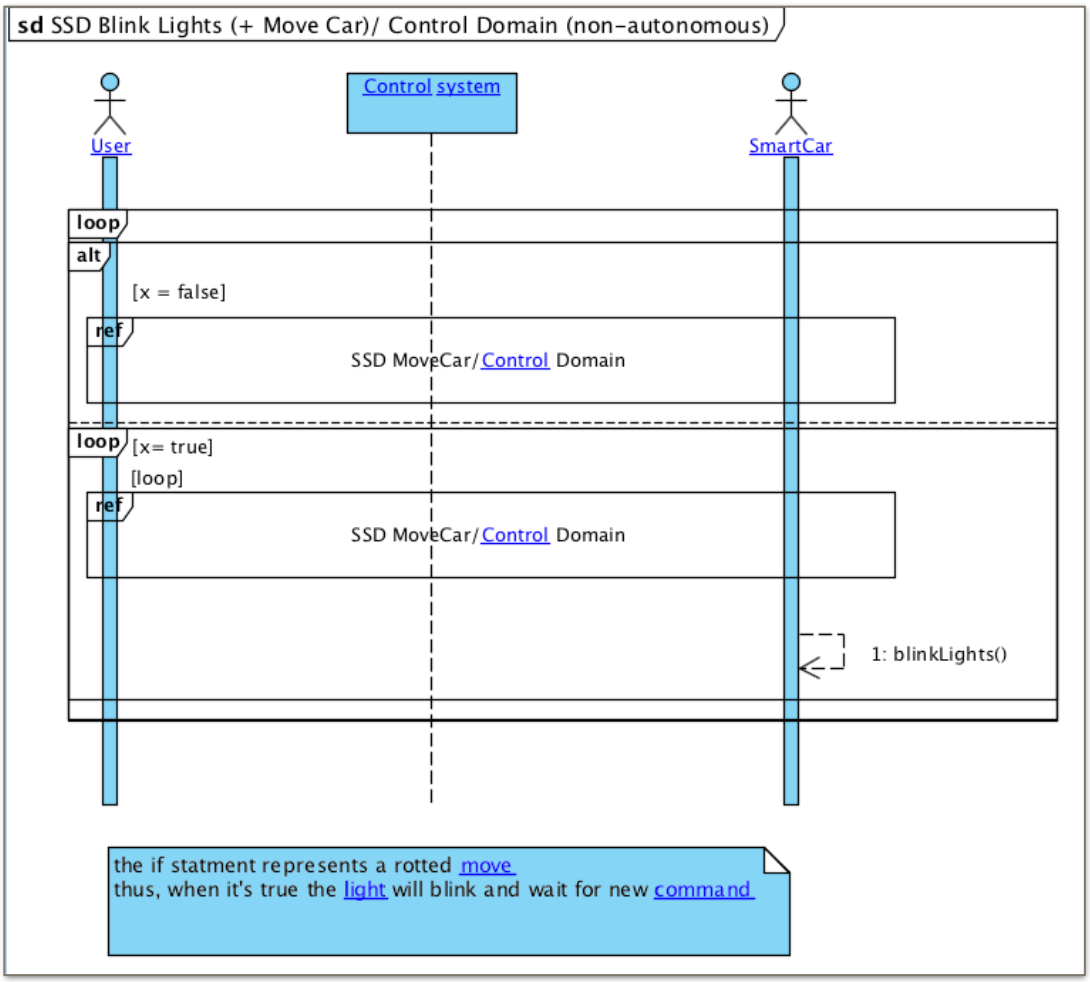

Figure 3. Blink Light SSD + Move Car in the Control domain view when the car is being controlled (Author: Rema).

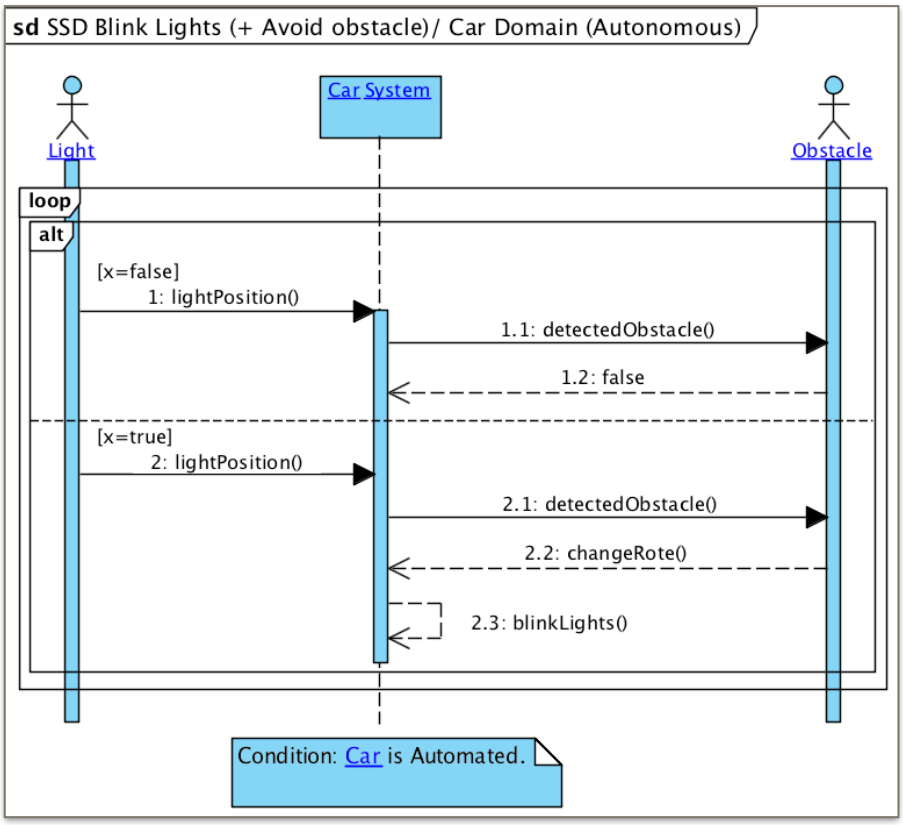

Figure 4. Blink Light SSD + Avoid Obstacles in the Car domain view when the car is autonomous (Author: Rema and Laiz).

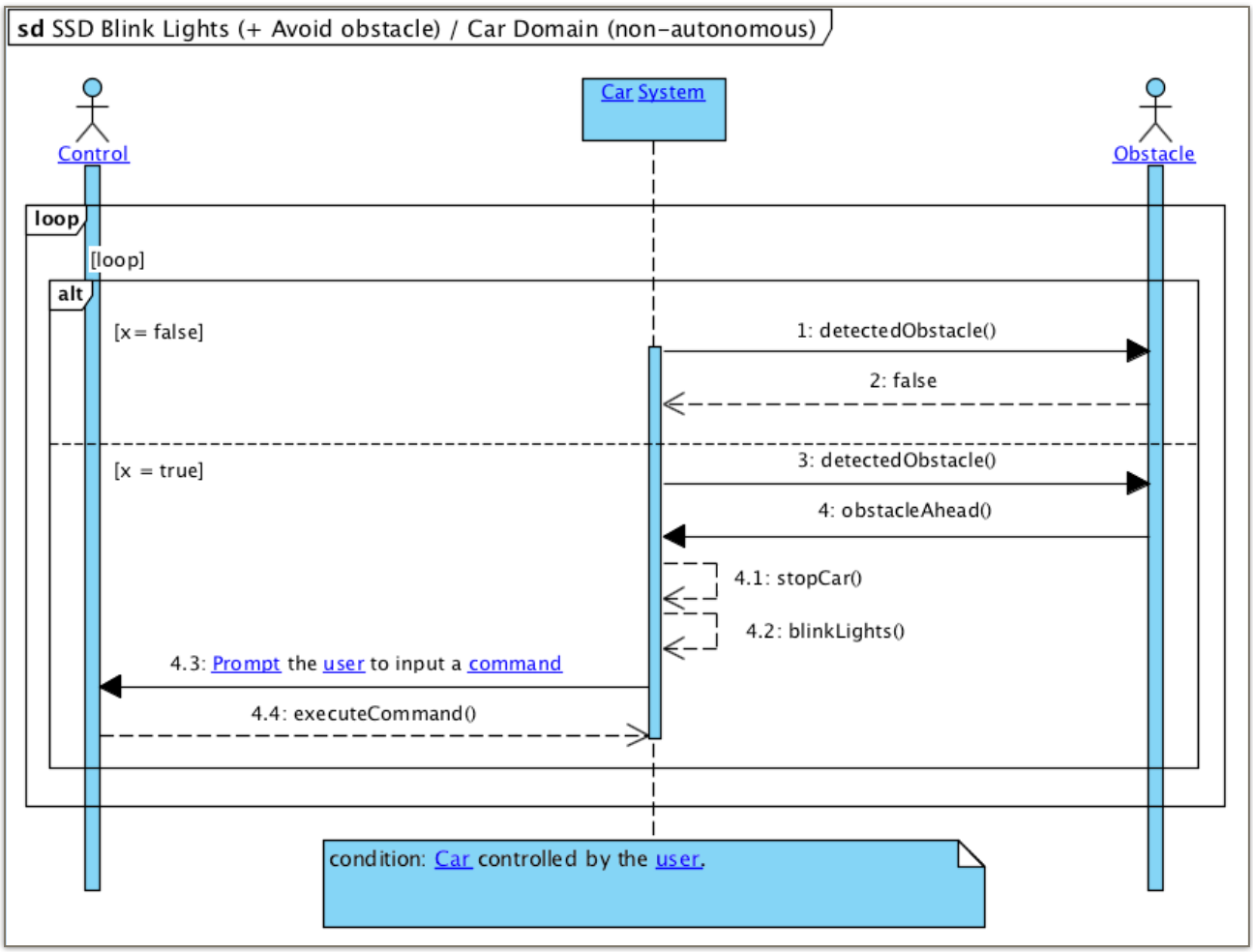

Figure 5. Blink Light SSD + Avoid Obstacles in the Car domain view when the car is being controlled (Author: Rema and Laiz).

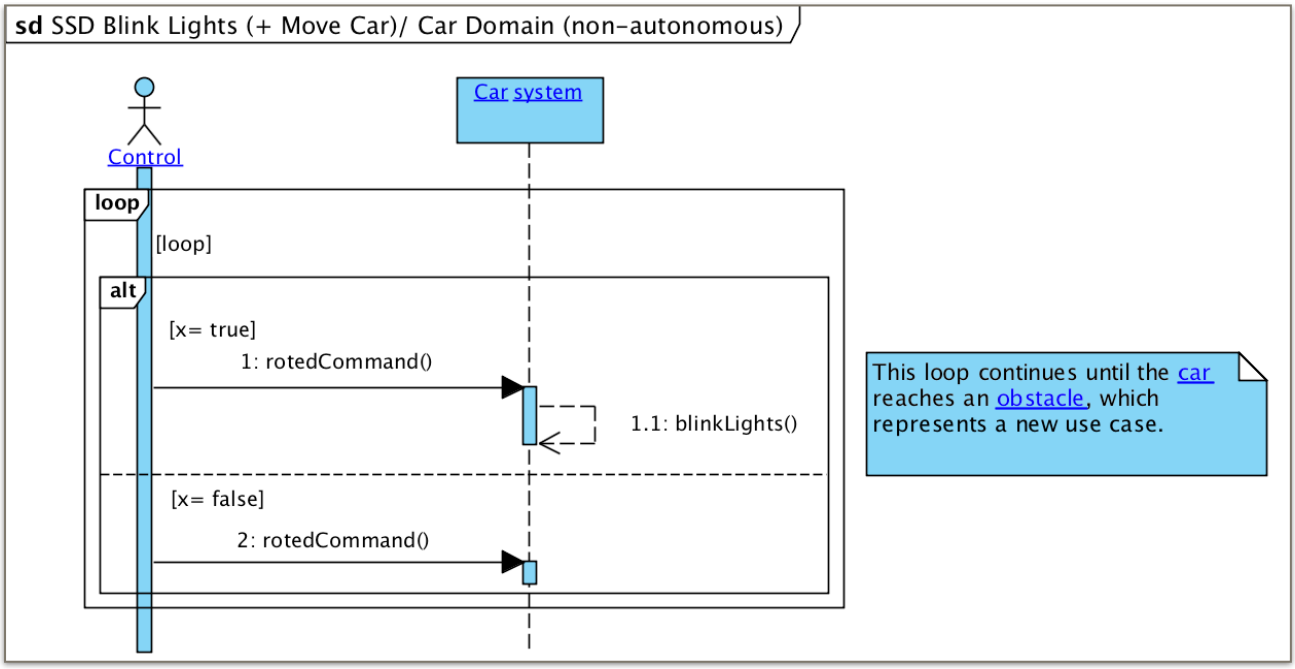

Figure 6. Blink Light SSD + Move Car in the Car domain view when the car is being controlled (Author: Rema).

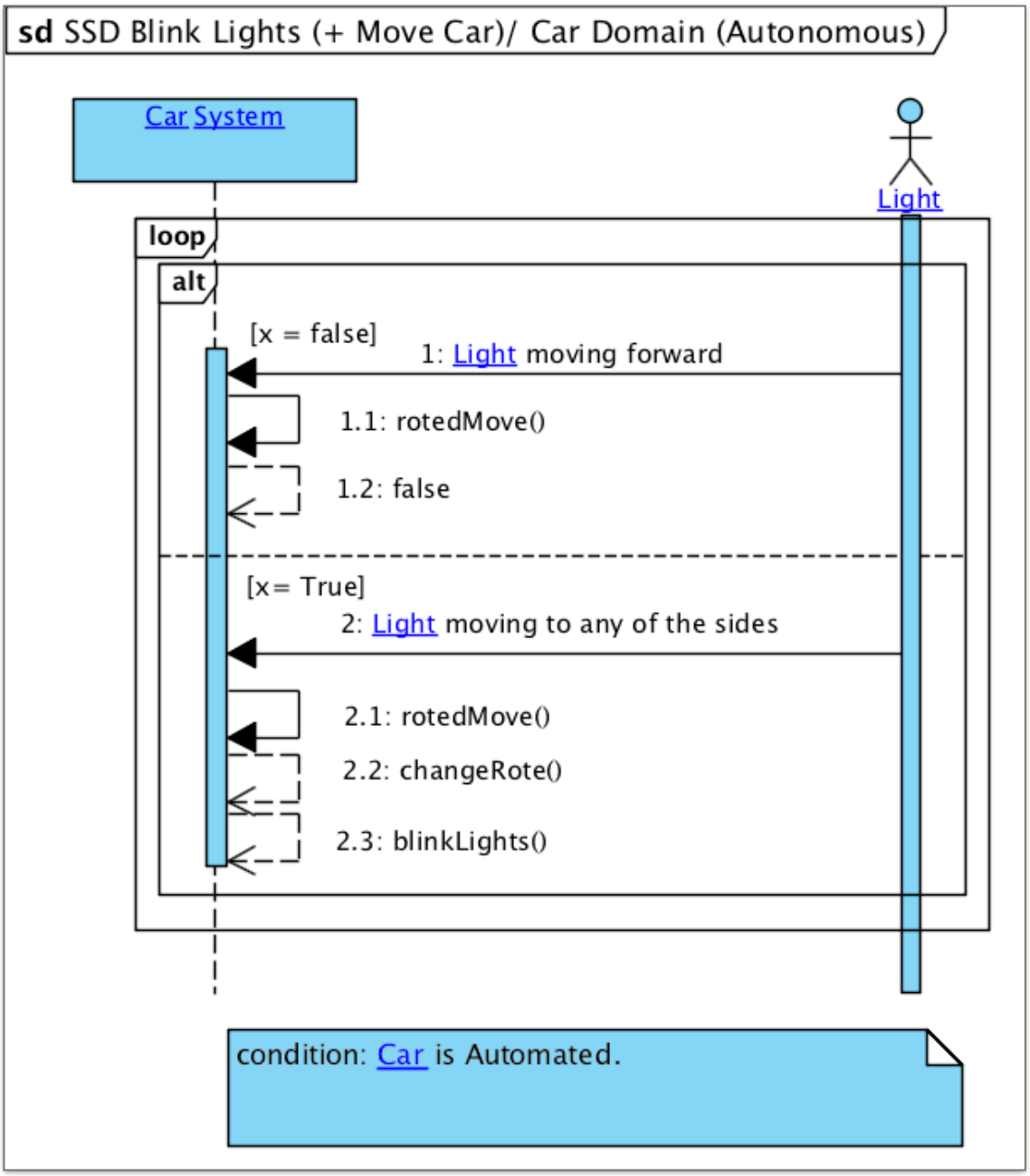

Figure 7. Blink Light SSD + Move Car in the Car domain view when the car is autonomous (Author: Rema).

M.E.N.A.C.E (Group 9)

Use Case Specification: Reset Car

Version <2.0>

Nina Uljanic

Revision History

Date	Version	Description	Author
02/02/2017	V.1.1	Add the first alternative flow; Add some Special Requirements; Add most of the Conditions; Add Extension point; Add the scenarios.	Laiz H. B. de Figueroa
22/02/2017	V.2.0	Changed the logging in from using username and password to pin code in the entire document.	Melinda Ivók

Table of Contents

Use Case Specification: Reset Car

1. Reset Car

1.1. Brief Description

In case of system issues, the car will need to be reset by the user.

2. Flow of Events

2.1. Basic Flow

The use case begins when the car is malfunctioning. The system will prompt the use to reset the car in order to solve the problem. The user approaches the car to flip the reset switch off and after a short pause back on. The reset is initiated within the car's system and a new connection is established between the car and the application. The user will be able to continue controlling the car afterwards.

2.2. Alternative Flows

2.2.1. First Alternative Flow

In case of hardware problems, such as the switch cable being disconnected, the user will need first to fix the hardware issue(s). Basic flow of this use case continues.

2.2.2. Second Alternative Flow

If the car has been reset but the connection between the car and the phone still does not work, the user must reset the phone application through the OS as well, before attempting to reconnect to the car. The user case continues.

3. Special Requirements

The interface will be responsive and allow more interactivity by the user.

A Bluetooth function and internet connection must be available in order to the system to perform.

Tools needed for fixing complex hardware issues must be acquired.

3.1. Friendly interface

3.2. Interactive interface

3.3. Bluetooth and Wi-Fi functions availability

3.4. Tools to fix the hardware

4. Preconditions

4.1. Application open

4.2. User logged in with pin code

4.3. Internet connection

4.4. Battery charged

4.5. Car Malfunctioning

5. Post Conditions

5.1. Car functioning

5.2. Control the car

5.3. Bluetooth connection

6. Extension Points

6.1. Inclusion: Connect Car

To be able to know if the car's system has a problem, the connection with the car must either already exist or the connecting process must have been attempted. Afterwards guide the user through troubleshooting steps.

7. Scenarios

7.1. Successful Solution to Malfunctioning

Basic Flow

7.2. Failed First Attempt to Solve Car's System Malfunctioning

First Alternative Flow
Second Alternative Flow

Use Case: Reset Car

The System Sequence Diagrams (SSD) for both domains, control (Fig. 1) and car (Fig. 2).

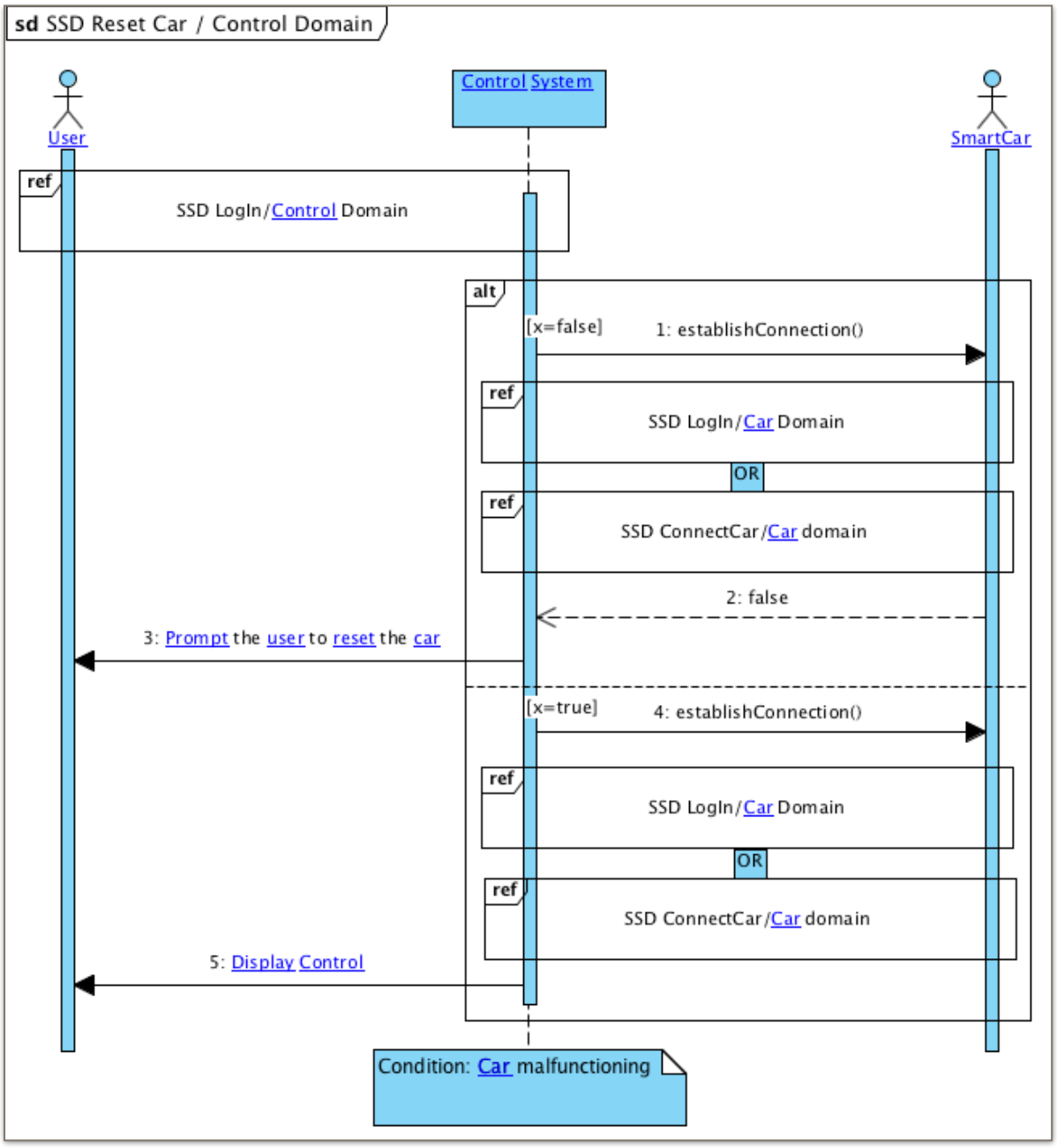

Figure 1. Reset Car SSD in the Control domain view (Author: Melinda and Laiz).

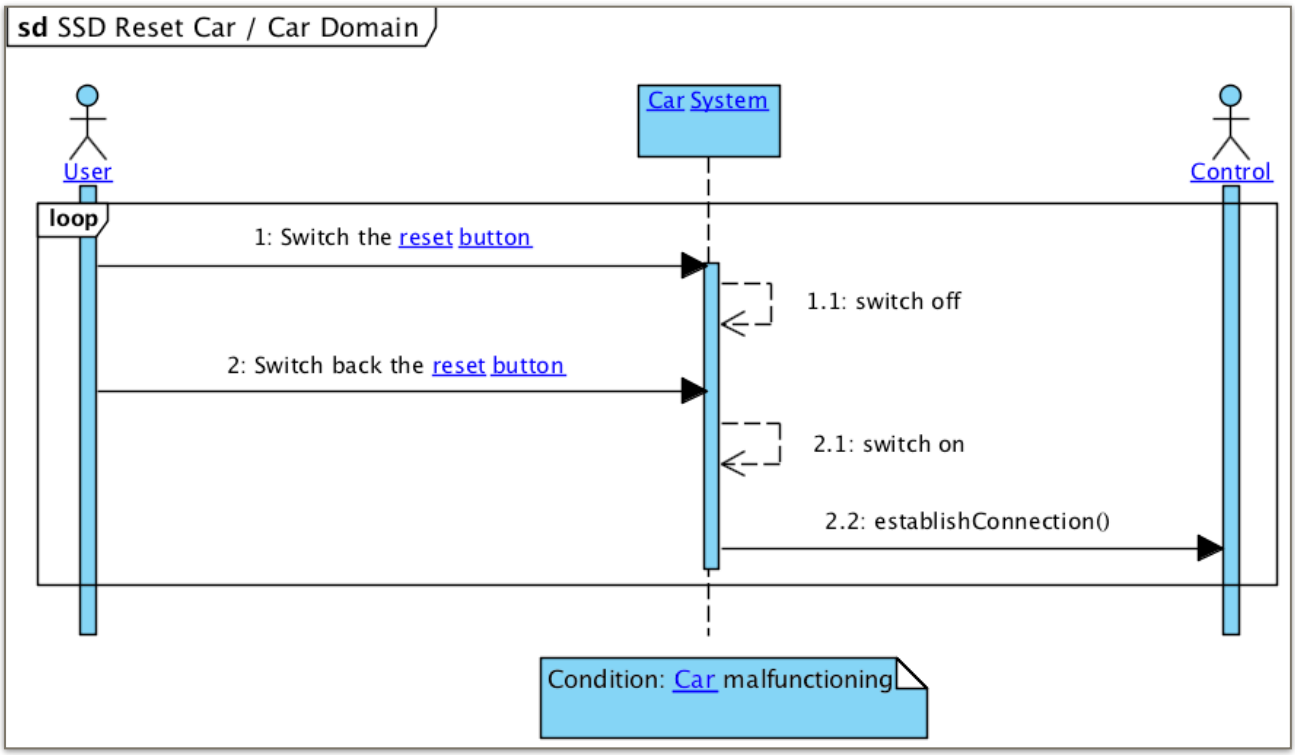

Figure 2. Reset Car SSD in the Car domain view (Author: Melinda and Laiz).