# Proposal for an application

Lisa Cattalani and Andrea Pari
Alma Mater Studiorum - University of Bologna, via Venezia 52
47023 Cesena, Italy
*lisa.cattalani@studio.unibo.it, andrea.pari6@studio.unibo.it*

April 21, 2017

## Contents

# 1 Introduction

The current report describes the entire software development process adopted to analyze, design and implement a specific software system. The process has been divided into two phases: the first phase concerns the definition of a model of the software system, while the second phase concerns the implementation of the software system carried out after that the product owner has viewed the model.

# 2 Vision

We want to show how to manage the development of a distributed, heterogeneous and embedded software system by focusing on the analysis and design of the system. The software system at issue will be designed and implemented as an application in the field of *Internet of Things*.

# 3 Goals

The goal is to build a software system able to evolve from an initial prototype (defined as result of a problem analysis phase) to a final and testable product. This goal will be achieved by working in a team and by "mixing" in a proper (pragmatically useful) way an agile (SCRUM) software development with modeling.

# 4 Requirements

A differential drive robot (called from now only robot) must reach an area (B) starting from a given point A. To reach the area B, the robot must cross an area equipped with N ($N \geq 1$) distance sensors (sonars). The signal emitted by each sonar is reflected by a wall put in front of it at a distance of approximately 90 cm. Moreover:

- The section of the wall in front of each sonar is painted with a different illustration.
- The robot is equipped with a distance sensor (sonar) and (optionally) with a Web Cam both positioned in its front. It owns also a Led.
- The robot should move from A to B by travelling along a straight line, at a distance of approximately 40-50 cm from the base-line of the sonars.

Design and build a (prototype of a) software system such that:

- Shows the sonar data on the GUI associated to a console running on a conventional PC.
- Evaluates the expression

  $(s_k + s_k + 1 + ... s_N) / (N - k + 1)$

  where k is the number of the first sensor not reached by the robot and $s_k$ is the value of the distance currently measured by that sensor. If the value of the expression is less than a prefixed value DMIN (e.g. DMIN = 70), play an alarm sound.
- When the robot reaches the area in front of a sonar, it:
  - first stops;
  - then rotates to its left of approximately 90 degrees;
  - starts blinking a led put on the robot;
  - takes a photo of the wall (in a simulated way only, if no WebCam is available) and sends the photo to console by using the MQTT protocol;
  - rotates to its right of approximately 90 degrees to compensate the previous rotation;
  - stops the blinking of the led and continues its movement towards the area B;

- When the robot leaves the area in front to the last sonar, it continues until it arrives at the area B.
- Stops the robot movement as soon as possible:
  - when an obstacle is detected by the sonar in front of the robot;
  - when an alarm sound is played;
  - the user sends to the robot a proper command (e.g. STOP).
- Makes it possible to restart the system (by manually repositioning the robot at point A) without restarting the software.

# 5 Requirements analysis

The first step for a correct requirement analysis is to read carefully the requirements text and to create a glossary of terms. The creation of a glossary is crucial in order to understand the applicative domain of the system and to avoid any ambiguity or misunderstanding during the requirements analysis.

The terms are reported in the following table in the order in which they appear into the requirements text. For each terms is shown the related meaning and eventual synonims that appear into the text.

| Term | Definition | Synonyms |
|---|---|---|
| *Differential drive robot* | A differential drive robot is a two-wheeled drive robot system with independent actuators for each wheel. The name refers to the fact that the motion vector of the robot is the sum of the independent wheel motions. | Robot |
| *Distance sensor* | A distance sensor is a device that is able to locate the presence of an object (also called obstacle) in front of it. The sensor estimates the distance from the object by sending waves toward it and by calculating the response time. | Sonar<br>Sensor |
| *Prototype* | A prototype is an incomplete version of a software system used to demonstrate concepts, to try some project options and to find out problems and possible solutions. A prototype can present all or a few characteristics of the final system. | Model |
| *GUI* | This term refers to the *Graphical User Interface* associated to the console. Through this interface, the user can view the data sent to the console by the distance sensor of the robot. | Interface<br>User interface |
| *Console* | In the current context, the console represents the means by which the user and the robot interact with each other. Through the console, the user sends commands to the robot and receives the photos taken by it. | Command panel |
| *Alarm sound* | The sound emitted by the console when the value calculated through the expression of the distance is under a certain threshold. In particular, the sound is emitted when a sensor on the wall has detected an object in front of itself (then along the path of the robot). | Alarm |
| *MQTT protocol* | MQTT is the communication protocol through which the robot sends the taken photos to the console. | |
| *Obstacle* | This term denotes any type of object detected by the sonar of the robot that prevents the robot itself to reach the area B. | |
| *User* | The user is the one who will use the software system, giving commands to the robot through the console and receiving the photos taken by it. | End user |
| *Command* | A command is an order given from the user to the robot through the console. Two commands are allowed: STOP (through the console) to stop the movement of the robot, RESTART (manually) to reposition the robot at a given point A. | |
| *Restart the system* | The process through which the robot is repositioned in a given point A without restarting the software. | Restart<br>Reboot<br>Reboot the system |

## 5.1 User Stories

A **user story** is an informal description of one or more features of a software system, and it's expressed in natural language in order to be understandable by anyone. A user story is then described from the end user's point of view, and not from the system's point of view.

The goal of the user stories is to describe with sufficient precision the end user's real needs, then what the user wants from the system, without to express technical details about the analysis, the design and the implementation of the software system.

The user stories will be defined using the following format:

*"As a **user**, I want <GOAL> so that <BENEFIT>"*

where *<BENEFIT>* denotes the acceptance criteria that must be respected in the story to ensure a proper implementation of the related features.

The stories collected by analyzing the requirements text are the following.

1. As a **user**, **I want that** the robot moves from a given point A to area B by travelling along a straight line.
2. As a **user**, **I want that** the robot stops when it reaches the area in front of a sonar and rotates towards it **so that** it can take a photo of the wall and send it to the console.
3. As a **user**, **I want that** the robot rotates to its original position after it took a photo **so that** it can continue to move towards the area B.
4. As a **user**, **I want that** an alarm sound is played whether the value calculated through the expression of the distance is under a certain threshold (i.e. a sensor on the wall has detected an obstacle in front of itself).
5. As a **user**, **I want** to use the console in order to send commands to the robot **so that** I can stop its movements.
6. As a **user**, **I want** to view the sonar data on the GUI.
7. As a **user**, **I want** to see the photos taken and sent by the robot to the console.
8. As a **user**, **I want** to restart the system by manually repositioning the robot at a given point A (i.e. without restarting the software).

## 5.2 User Interaction

Per rappresentare l'interazione tra agenti esterni e il sistema utilizzeremo diagrammi uml. L'unico attore è l'utente che interagisce con il sistema attraverso una serie d'azioni/comandi che rappresentano un caso d'uso nel diagramma. Il sistema è composto principalmente dalla console e dal robot.

NB: caso d'uso solo quello che può far direttamente l'utente, quindi mettere il robot, inviargli i comandi e guardare sul radar la posizione del robot. Scenari: Console , il robot,

| | |
|---|---|
| Name | Place Robot |
| Descrizione | Piazzare il robot |
| Attori | Utente |
| Precondizioni | La scelta del punto determinerà la traiettoria che poi avrà il robot. |
| Scenario principale | L'utente metterà il robot in un punto A. A partire da quel punto il robot potrà iniziare a muoversi. |
| Scenari alternativi | — |
| Postcondizioni | Il robot deve riavviarsi ogni volta che viene posizionato. |

| | |
|---|---|
| Name | Send Command |
| Descrizione | Invio di un comando al sistema da parte dell'utente |
| Attori | Utente |
| Precondizioni | La console è attiva e funzionante sul computer e il sistema è pronto a ricevere comandi. |
| Scenario principale | 1. L'utente invia un comando al robot tramite il pannello dei comandi presente sul PC. I comandi inviati sono START per azionare il robot e STOP per fermarlo. 2. Il server invia il comando al robot. 3. Il robot valuta il comando e lo esegue. |
| Scenari alternativi | — |
| Postcondizioni | Il robot si aziona se riceve il comando START, si ferma se riceve il comando STOP. |

| | |
|---|---|
| Name | View Sonar |
| Descrizione | Visualizzare la posizione del robot sul sonar |
| Attori | Utente |
| Precondizioni | La console dovrà funzionare sul pc. |
| Scenario principale | L'utente potrà vedere il robot nella GUI e si potranno seguire i movimenti. |
| Scenari alternativi | — |
| Postcondizioni | Il robot si aziona se riceve il comando START, si ferma se riceve il comando STOP. |

### 5.3 (Domain) Model

### 5.4 Test Planning

—TO DO

## 6 Problem analysis

### 6.1 Logic architecture

—TO DO

### 6.2 Risk analysis

—TO DO

## 7 Work Plan

—TO DO

## 8 Implementation

—TO DO

## 9 Testing

—TO DO

## 10 Deployment

—TO DO