LABORATORIO DI INGEGNERIA DEI SISTEMI SOFTWARE

Requirements

Design and build a software system that makes a robot is able to walk along the boundary of a rectangular, empty room.

Requirement analysis

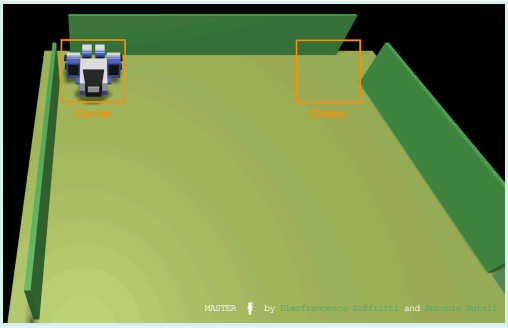
Nouns and verbs

1. Software system: the system will be a set of software products/applications;
2. Robot: the subject that will walk along the boundary of the room. Known robot's characteristics: VirtualRobot2021;
3. To walk: how the robot can move;
4. Boundary: the path that the robot has to follow;
5. Rectangular, empty room: the environment where the robot will be used. Rectangular shape (so also square shape) and no obstacles.

User Story

As a user, I put the robot in one of the four **corners** of the room (the image contains only the two upper corners) and the robot must not face a wall. Then I start the system who will communicate with the robot without cables (wireless connection). The system will say to the robot how to complete its task and I can't do/don't need to do anything else while it's running.

When the system will end, the robot will have done a single and complete walk along the boundary.



First Test Plans

- The robot has to meet 4 times a wall and turn itself before restart to walk;
- If the robot walk clockwise it has to rotate 4 times right, else 4 times left;
- The distance walked and the time spent on opposite sides must be always equals.

Problem analysis

1. The system has to communicate with the robot. The problem suggests that the communication has to be without cables, so using wireless connections;
2. The system isn't affected by the structure of the robot;
3. Communications between the system and the robot must be standardized, simple, light and unambiguous;
4. The system can communicate with the robot using two ways:
   - HTTP POST on port 8090
   - websocket on port 8091
5. The system has to be able to know what the robot has done. So also the robot must communicate with the system;
6. The problem suggests a logical architecture composed by:
   - a main service used to say to the robot what it has to done and to retrieve its feedbacks;
   - a service used by the robot to receive commands and to respond.
7. There is a conceptual abstraction gap because there are two possible ways of communication but both requires a request-response schema.

Test plans

1. ```
   @Test
   public void test4Wall() {
      BoundaryWalk appl = new BoundaryWalk();
      String feedbacks = appl.walkBoundary("counterclockwise", 250);
      int walls = countWallsFromFeedbacks(feedbacks);
      assertTrue(walls == 4);
   }
   ```
2. ```
   @Test
   public void testClockwiseRotation() {
      BoundaryWalk appl = new BoundaryWalk();
      String feedbacks = appl.walkBoundary("clockwise", 250);
      int rightRotation = countRotationFromFeedbacks("right",feedbacks);
      assertTrue(rightRotation == 4);
   }
   ```
3. ```
   @Test
   public void testCounterclockwiseRotation() {
      BoundaryWalk appl = new BoundaryWalk();
      String feedbacks = appl.walkBoundary("counterclockwise", 250);
      int leftRotation = countRotationFromFeedbacks("left",feedbacks);
      assertTrue(leftRotation == 4);
   }
   ```
4. ```
   @Test
   public void testTimeOnOppositeSides() {
      BoundaryWalk appl = new BoundaryWalk();
      String feedbacks = appl.walkBoundary("counterclockwise", 500);
      int t[] = new int[4];
      for (int i=0; i<4; i++)
         t[i] = sideTime(i,feedbacks);
      assertTrue(t[0] == t[2] && t[1] == t[3]);
   }
   ```
5. ```
   @Test
   public void testRotationAfterWallCounterclockwise() {
      BoundaryWalk appl = new BoundaryWalk();
      String feedbacks = appl.walkBoundary("counterclockwise", 500);
      char feedback[] = divideFeedbacks(feedbacks);
      for(int i=0; i<feedback.length; i++) {
         if (feedback[i] == 'w' && i != feedback.length - 1)
            assertTrue(feedback[i+1] == 'l');
      }
   }
   ```

- ```
  private int countWallsFromFeedbacks(String f) {
     int walls = 0;
  ```

```
    for (int i=0; i<f.length(); i++)
        walls = f.charAt(i) == 'w' ? walls + 1 : walls;
    return walls;
}
```

- ```
  private int countRotationFromFeedbacks(String direction, String f) {
      char d = direction == "left" ? 'l' : 'r';
      int rotation = 0;
      for (int i=0; i<f.length(); i++)
          rotation = f.charAt(i) == d ? rotation + 1 : rotation;
      return rotation;
  }
  ```
- ```
  private int sideTime(int side, String f) {
      String sideFeedback = f.split("-")[side];
      System.out.println(sideFeedback);
      int sideTime = 0;
      for (int i=0; i<sideFeedback.length(); i++)
          sideTime = sideFeedback.charAt(i) == 'f' ? sideTime + 1 : sideTime;
      return sideTime;
  }
  ```
- ```
  private char[] divideFeedbacks(String f) {
      char result[] = new char[f.length()];
      int j = 0;
      for (int i=0; i<f.length(); i++) {
          if (f.charAt(i) != '-') {
              result[j] = f.charAt(i);
              j++;
          }
      }
      return result;
  }
  ```

By Daniele Colautti
email: daniele.colautti6@studio.unibo.it


  ```