# Assignment 1, TDA593
# Group 11

Oscar Hansson, Josefin Ulfenborg, Lisa Carlsson
Adam Grandén, Felix Kirchmann, Rasmus Jemth

November 2017

# 1 Social contract

We decided the following responsibilities for the team members

- Attend weekly meetings
- Always keep open discussion to avoid running into any problems
- Don't be late for the scheduled meetings
- Everyone should perform the task they are assigned to do
- Be helpful
- Be friendly with each other! Discuss if a problem arises.
- Take short breaks (few minutes) every half hour
- Take longer breaks (10-15 minutes) every hour

# 2 Goals and objectives

Our goal for this project is getting a 5/VG.

# 3 A brief description of the project context

We are going to do a simulation of robots in a certain environment. This environment will contain obstacles that might move (e.g. other robots) and the robots will be equipped with sensors for this. They will also be given missions (these missions might change at any time), that consist of going between a set of points in the environment, and strategies for performing these missions (which may also change at any time). The robots are also given reward points for being in certain areas, at certain times. How these points are calculated varies based on where the robots are and how the points were previously calculated. There should be interfaces for the non-technical operators, both graphical and interfaces designed for blind people. Finally the robots should be able to turn themselves off in case of technical errors, and the operators should be able stop everything (missions and robots) in case of severe circumstances.

## 3.1 Clarifications and assumptions

- Nested areas

  We can have areas that overlap but that does not make them nested

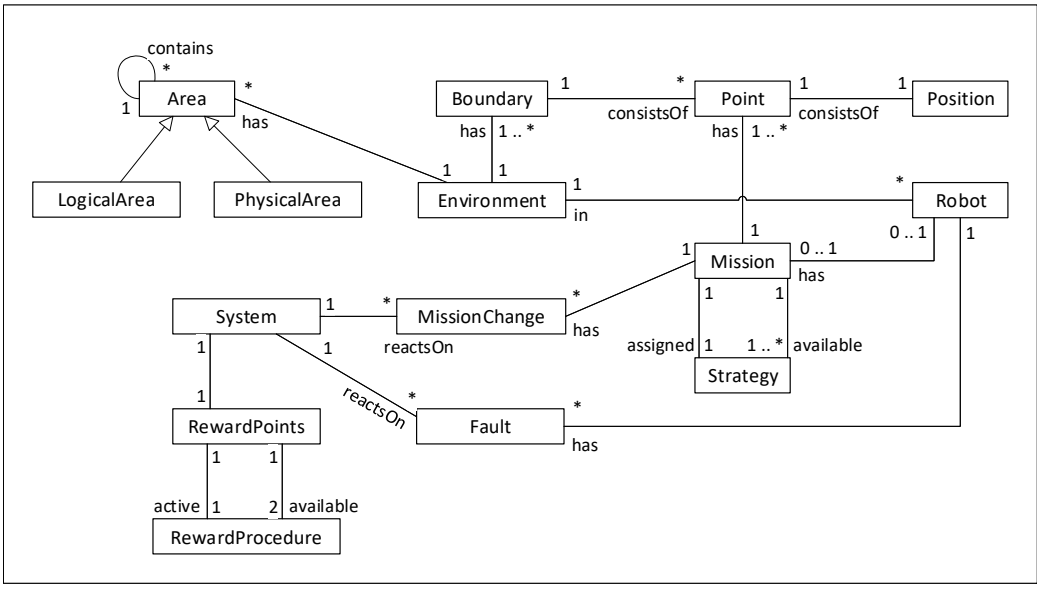
  A nested area is completely nested inside of another area

  Assumption: An area can be nested within any type of area (e.g. a physical area can be a logical one)

- Reward points
  - The points are accumulated
  - The robots should try to get as many points as possible
  - Reward points stored as one global variable
  - Procedures incrementing reward points can be swapped anytime
  - Reward procedure is global, not per-robot
- Assumption: One reward point counter is used for any number of environments
- Assumption: All the mission points have to be visited by the same robot
  - A mission can only assigned to one robot
- User-Provided (Input)
  - Environment, limited by boundaries
    - Assumption: All walls are at a 90° or 0° angle to each other
  - Robots with Positions and Fault status
  - Missions (with changes at runtime)
    - Includes Strategies and target point sequences
- Handled at lower levels in the tech stack (not part of our framework)
  - Obstacles (position + evasion handled by robot firmware / sensors)
  - Movement
- State changes at runtime
  - Missions (including target points)
  - Strategies
  - Reward points + procedures
  - Rover positions
  - Fault status
- Does not change at runtime
  - Environment (boundaries)
  - Logical and physical areas
- Error Handling
  - The system will receive/print output if something goes wrong, but the error handling itself is not done by the system.

# 4 Domain Model

## 4.1 Diagram

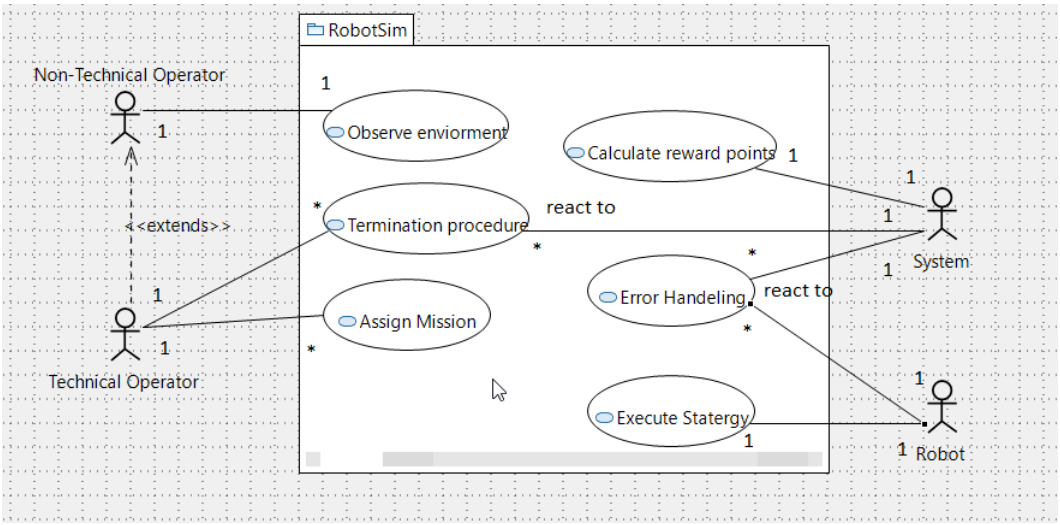



## 4.2 Description

The system can have any number of environments. Each environment has any number of robots, which all have one position, which is a point within the boundry of the environment.
The environment has any number of areas, that can be either logical areas or physical areas. These areas can be nested inside other areas.
Each robot can have 0 or 1 mission, which has one or multiple points, and every mission has multiple strategies avaible to them, but only one strategy which the mission is assigned.
The system calculate reward points based on a reward procedure. There are 2 different procedures, and only 1 of them is active at any time.
The system can react upon faults, of which a robot can have any number, and also mission changes, of which a mission can have any number.

# 5 Use case diagram

## 5.1 Diagram




## 5.2 Description

Since our diagram only covers a subset of all use cases, this section briefly explains why each use case is included.

- Observe Environment: The non-technical operators are supposed to observe changes in the environment through a user interface. The implementation of this user interface will likely be a major part of our future codebase.
- Assign Mission: The technical operator can assign missions to the robots. We have to have a mission for the robot to execute otherwise the system is useless.
- Terminate Procedure: The system reacts to this event, and the operator is the one to turn the system off in case of extreme circumstances. We feel that this part of the system is vital if something fails or malfunctions.
- Calculate Reward Points: The system calculates the reward points for the robots. This is closely related to the execution of the strategies and is the reason this is important to include in our use cases.
- Execute Strategy: The robot executes a certain strategy over the environment. This is related to the calculation of the reward points and the assigning of missions. Without the strategy the robot cannot perform the assigned mission.

- Error Handling: In case of an error with the robots they are to turn themselves off, and the system will react to this event. As with the termination of procedures, the error handling is important if something fails or malfunctions.