

Assignment 3, TDA593
Group 11

Oscar Hansson
Adam Grandén

Josefin Ulfenborg
Felix Kirchmann

Lisa Carlsson
Rasmus Jemth

November 2017

2 Description of classes, operations and relations

2.1 MissionManager

The MissionManager can assign missions to robots with a strategy that is given by the Strategy class. These missions can be removed by the MissionManager and we can also get the current progress of a given mission. It also supplies methods for adding and removing MissionProgressListeners.

2.2 DefaultMissionManager

Implements MissionManager. It handles the movement of the robot, and also contains a list of robots, a list of missions, a list of strategies, and a map of robot and strategy pairs.

2.3 MissionProgressListener

Supplies the method onMissionProgress, to be run when a robot reaches a point in it's mission.

2.4 Mission

Has methods for getting the name of the mission, and getting the points.

2.5 MissionData

Implements Mission. MissionData represents an ordering of points which the assigned robot has to follow. It also contains a string name, to be able to identify it.

2.6 MissionStrategy

Has methods for getting the name, the mission which the strategy is applied to, and the ordered points.

2.7 MissionStrategyData

Implements MissionStrategy. Represent an ordering of points from a given mission which is the route the robot should take. A strategy is also connected to a mission.

2.8 Point

Has methods getX() and getZ(), to be able to get the x and z position of a point.

2.9 PointData

Implementations point. Have float values for x and z position.

2.10 Robot

Contains methods to be able to access the data of a robot. The name, the position point and the destination point. Associated with Point due to its position and destination point. The MissionManager is responsible for the movement of the robots which is decided by their mission and strategy.

2.11 SimbadRobot

Implementation of the Robot interface, contains the data necessary to be able to model a robot in the Simbad simulator.

2.12 DefaultFaultManager

The class will reports all faults from a specific robot to the system.

2.13 Fault

The Fault Interface will implements a listener on the SimbadRobot where the Robot will get a listener. The Simbad Robot will then report errors from the Fault interface.

2.14 AreaCollection

Returns lists of areas, either all areas. Can also check which area/areas a certain point is in.

2.15 AreaCollectionData

Implements the interface AreaCollection. Contains a list containing every area in the environment. The AreaCollection is used by the WebInterfaceDependencyProvider to display the areas and the RewardPoints class to calculate reward points.

2.16 AbstractArea

Represent an area in the environment, represented with a bounding box. Has * children, which is areas inside itself, and * parents which are areas the current area is inside. Contains a name which identifies the area. It is either a logical area or a physical area. The areas are contained in the AreaCollection.

2.17 Physical Area

Extends from the AbstractArea class. A physical area is an area which is bounded by walls.

2.18 Logical Area

Extends AbstractArea class. A logical area is an area which is not bounded by walls.

2.19 RewardPoints

Has a method which returns the number of reward points the system has.

2.20 DefaultRewardPoints

Represents the counter of reward points gained from accomplishing. Uses the EnvironmentRuntime to get access to the robot and the TimestepSource so that it can update the points every 20 seconds. The reward point value for each area is stored in a hash map or of that kind where some areas gives 1 point and some 2 points.

2.21 Procedure

Interface which supplies a method for calculating rewardPoints.

2.22 ProcedureA

Implements the procedure interface. Used when computing reward points in physical areas.

2.23 ProcedureB

Implements the procedure interface. Used when computing reward points for the logical areas.

2.24 Environment

Interface to get the objects in the Environment. Has one method for listing the locationControllers, and one for listing the staticObstacles.

2.25 EnvironmentData

Container of all the objects which make up an Environment.

2.26 StaticObstacle

Interface which supplies methods for getting the color, line, and type of an obstacle.

2.27 StaticObstacleData

Implements StaticObstacle. Has a color, type, and an OrthogonalLine. The type can either be WALL, used to make room and other obstacles, or BOUNDARY, used to encapsulate the environment.

2.28 OrthogonalLine

Interface for representing a line. Has methods for getting x and z coordinates.

2.29 VerticalLine

Extends OrthogonalLine. Has a method for getting an x endpoint.

2.30 VerticalLineData

Implements VerticalLine.

2.31 HorizontalLine

Extends OrthogonalLine. Has a method for getting an z endpoint.

2.32 HorizontalLineData

Implements HorizontalLine.

2.33 LocationController

Supplies methods for getting the location and radius of a LocationController.

2.34 LocationControllerData

Implements LocationController. Is used to grant exclusive access to certain areas.

2.35 BoundingBox

It has two methods, one for getting the upperLeft point and one for the lower-Right point.

2.36 BoundingBoxData

Each area has a BoundingBox that bounds the area. The class consists of two variables, upperLeft and lowerRight, which both are points that together decides the covered bounding area.

2.37 WebInterfaceDependencyProvider

An immutable container object that has a reference to every service whose data the web interface should display. It currently references EnvironmentRuntime, RewardPoints, AreaCollection and MissionManager.

2.38 WebInterfaceServer

Handles the server which handles the interface. Have the ability(method) to start and stop the server.

2.39 EnvironmentRuntime

Returns the environment, a list containing all robots and the timestep source of the system.

2.40 Endpoint

An endpoint of the web application provides methods that are exposed to web clients over HTTP.

2.41 StatusPageEndpoint

Provides a simple status page overview of the robotic system, including missions and reward points. This overview is returned as a single HTML page.

2.42 TimestepSource

The SimbadInstance implements TimestepInstance which adds a TimestepListener to the EnvironmentRuntime. Where the TimestepListener sets the speed of the EnvironmentRuntime.

3 Contribution report

Everyone worked with designing the class diagram, and then everyone made a class diagram in papyrus which was related to their use case. Everyone worked with deciding with how the mission were to be implemented. The class implementation were then delegated to the group members.

- Oscar
 - Creating a class diagram for the "Execute strategy" use-case.
 - Wrote descriptions to the classes, operations and relations.
 - Implemented the class(es):
 - AbstractArea
 - PhysicalArea
 - LogicalArea
 - AreaCollectionData
- Josefin
 - Creating a class diagram for the "Create area" use-case.
 - Worked with code generation (fixed the templates of all classes and interfaces).
- Lisa
 - Creating a class diagram for the "Calculate reward" use-case.
 - Combined all the use-case diagrams and made a class diagram for the entire project in papyrus.
 - Wrote descriptions to the classes, operations and relations.
 - Worked with code generation.
- Adam
 - Creating a class diagram for the "Error handling" use-case.
 - Wrote descriptions to the classes, operations and relations.
 - Restructure the papyrus diagram
 - Implemented the class(es):
 - LoactionListener
- Felix
 - Creating a class diagram for the "Observe environment" use-case.
 - Developing a roadmap to provide a clear overview of the subtasks necessary to complete the assignment
 - Reviewing the class diagram after merging; changing and adding classes to ensure coherency between class and component diagrams (done in OneNote, implemented in Papyrus by Lisa)
 - Implemented the class(es):
 - MissionData
 - MissionStrategyData
 - DefaultMissionManager

- Rasmus

Creating a class diagram for the "Assign mission" use-case.

Wrote descriptions to the classes, operations and relations.

Implemented the class(es):

Environment

EnvironmentData

StaticObstacle

StaticObstacleData