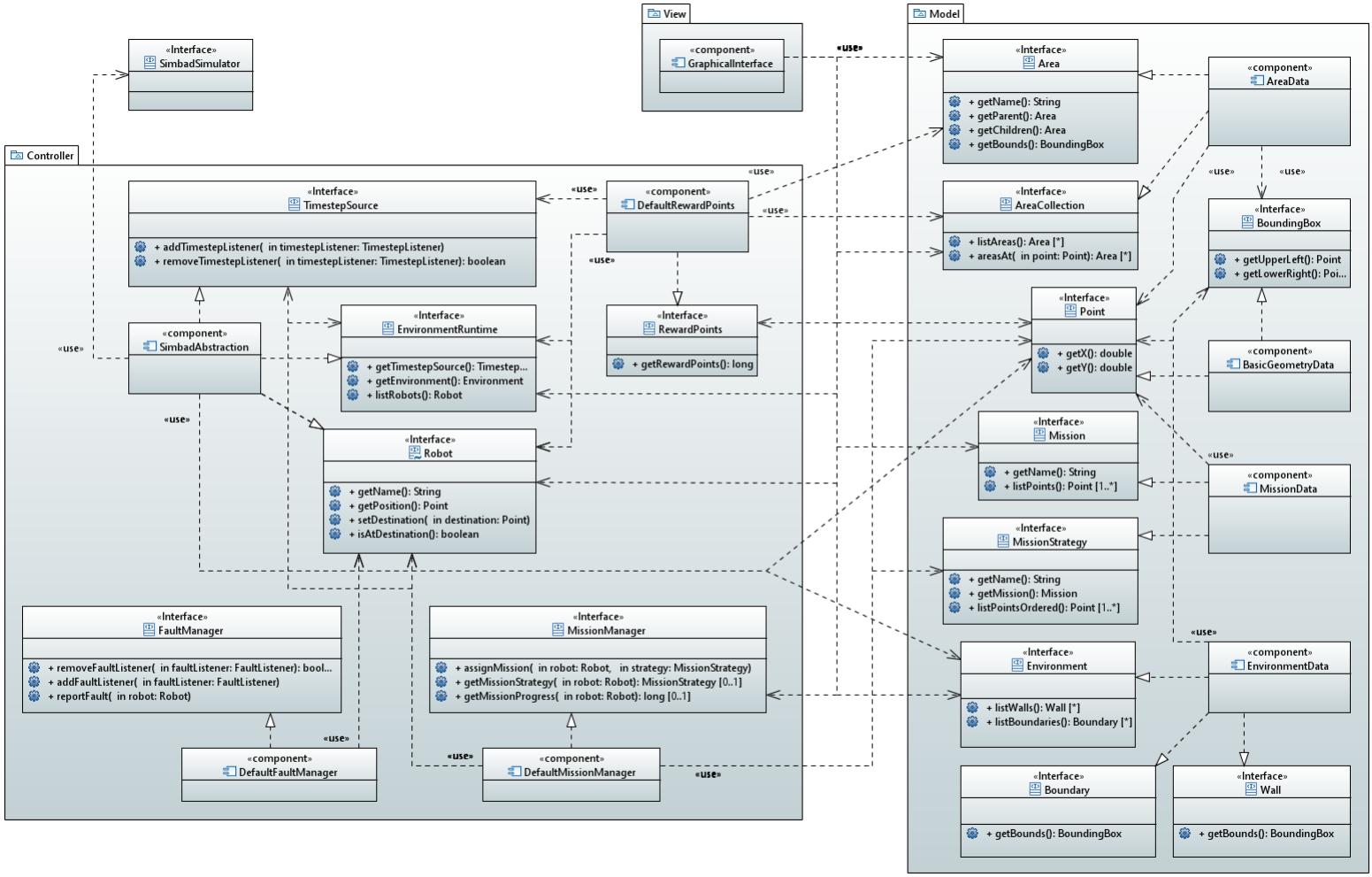
Assignment 2, TDA593
Group 11

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

November 2017

1 Component Diagram



2 Explanatory text

Our architecture attempts to follow the Model-View-Controller-pattern, while using immutable data objects to simplify controller logic. We chose to use the MVC-pattern because it enables us to separate application logic from the implementation of the user interface. Furthermore, it allows us to modularize our dependency on the Simbad robot simulator, which allows it to be swapped out for a different simulation platform or even an interface for physical robots. We did not use Sense-Plan-Act, which is often used in the robotic domain, because for us it seemed to better fit on low-level programming.

In the following explanations, we have marked components with (c) and interfaces with (i).

- **SimbadSimulator (i)**
  This represents the API available through the course's version of the Simbad robot simulator. For reasons of brevity, this is only used to indicate a dependency. It does not contain the available API methods.
- **SimbadAbstraction (c)**
  An abstraction layer on top of Simbad. On instantiation, it is provided with an immutable environment and a list of robots with their positions in that environment. It then uses the Simbad API to simulate the environment and exposes the results through a number of interfaces.
- **Robot (i)**
  Provides information about each robot currently in the simulation, its position and whether it has reached its destination. Also allows setting a destination for the robot.
- **TimestepSource (i)**
  Provides a source that can inform any number of listeners about the speed at which time in the simulation is progressing.
- **EnvironmentRuntime (i)**
  Provides the environment and the list of robots inside it that are currently being simulated.
- **RewardPoints (i)**
  Interface for getting access to the accumulated reward points.
- **DefaultRewardPoints (c)**
  Handles the storage and calculations of the reward points.
- **FaultManager (i)**
  Receives notifications of robot faults and distributes them to any number of fault listeners. Since we do not yet know of an API (e.g. within Simbad) that can be used to detect robot faults, the detection implementation is not yet part of the component diagram.

- **MissionManager (i)**
  Can be used to assign missions (with strategies) to robots and track how far a robot has completed its assigned mission.
- **DefaultMissionManager (c)**
  The implementation of the MissionManager uses a TimestepSource to repeatedly check if a robot has arrived at its current destination point. If this is the case, it assigns the next point from its current mission to the robot.
- **AreaData / BasicGeometryData / MissionData / EnvironmentData (c)**
  These components implement the data structures that are made accessible via their respective provided interfaces.
- **Mission (i)**
  Missions are a list of points for a robot to reach. The order of the points in the list is not relevant, as the order in which a robot will process the points is defined by in MissionStrategy.
- **MissionStrategy (i)**
  A MissionStrategy is an ordering of the points of a specific instance of Mission.
- **Environment (i)**
  Contains the walls and boundaries that define an environment.
- **BoundingBox (i)**
  Defines an arbitrary rectangle by its upper left and lower right corner points.
- **Boundary (i)**
  Specifies a boundary that defines the limits of an environment.
- **Wall (i)**
  Specifies a wall, which is a static obstacle within an environment.
- **Point (i)**
  Defines a single point on the X-Y-plane of any environment.
- **Area (i)**
  Gives access to the name, the children and the parent of an area.
- **AreaCollection (i)**
  Contains a set of areas, and has a utility method to return all areas of the set that are present at a given point.
- **GraphicalInterface (c)**
  Provides the interface displaying environment data, robots, their missions and reward points to non-technical users.

3 Contribution report

We worked a lot together and most of the work together, so we feel everyone contributed in equal parts. Adam and Felix were assigned to transcribe the component diagram to Papyrus, while the others worked more on the explanatory text.