**TAU-Robot Exercise 1 - Design Document**

**Yaron Libman and Ilya Aizin**

Last update: March 30th 2016

Scope of this document: Explanation of components used in implementation and design rational.

* For further explanation see class and sequence diagrams.

**Introduction**

We decided to divide the project into few logical classes. Each class is responsible for a specific part of the simulation.

**House**

*Uses:* Direction

* Represents a single House, contains the house information and map.
* Responsible for creating the house object, has simple constructor that creates the house and the map by a hard coded house (for exercise 1).
* Implements a copy constructor.
* Responsible for moving the vacuum inside the house by direction.
* Responsible to updating dirt level, battery level and vacuum location by request.
* Checks validity of the house.

**OneSimulation**

*Inner class of Simulator*

*Used by:* Simulator

*Uses:* House, Abstract Algorithm, Sensor

* Represents a single simulation of a single abstract algorithm running on a single house.
* Responsible for updating house state when abstract algorithm makes a move.
* Checks validity of Abstract algorithms moves and updates simulation status accordingly (i.e. Abstract algorithms made an illegal move, out of battery etc.)
* Calculates single simulation score using winner information provided by Simulator

**Simulator**

*Uses:* House, OneSimulation, Naïve Algorithm, Abstract Algorithm, Sensor, Abstract Sensor

* Loads Abstract Algorithm implementations, houses and configuration information.
* For each house, runs all algorithm simulations in parallel.
* Coordinates multiple simulations by executing them step by step in round robin fashion.
* Keeps track of the winner and simulation finishing conditions (Max\_Steps).
* When simulation finishes initiates score calculation by OneSimulation by passing winner and global simulation information.
* Prints calculated scores.

**Naïve Algorithm**

*Simple implementation of Abstract Algorithm*

*Used by:* Simulator, OneSimulation

*Uses*: Sensor only

* Chooses next step randomly between staying and moving to any non-wall cell.

Note that the algorithm has no direct access to the house, it can only see what’s going on through the sensor.

**Sensor**

*Simple implementation of Abstract Sensor*

*Used by*: OneSimulation

*Uses*: SensorInformation, House, Direction

* Represents a single Robot Sensor.
* Responsible for updating the algorithm about its current environment, returns a Sensor Information object.

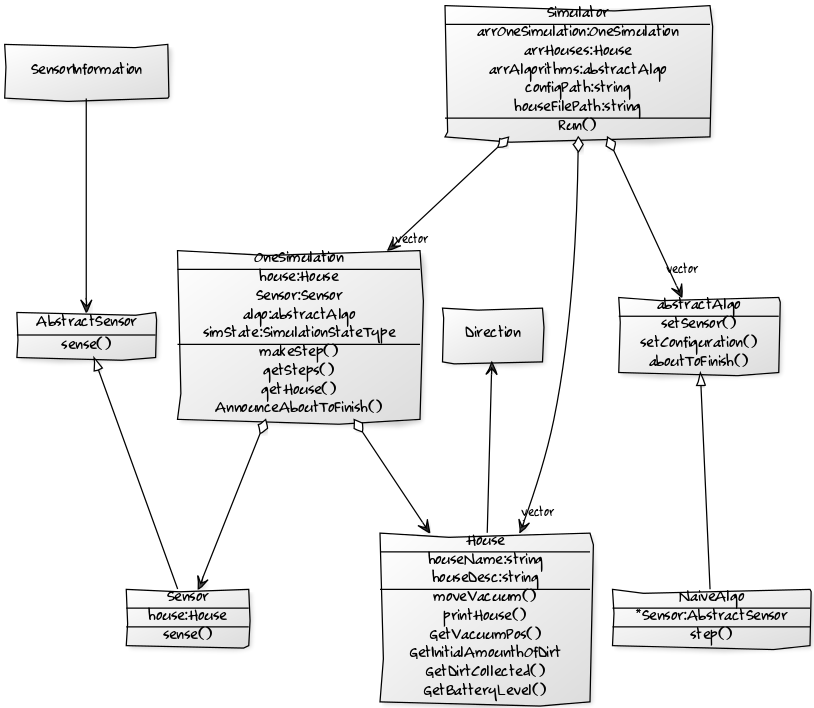
**Error Handling**

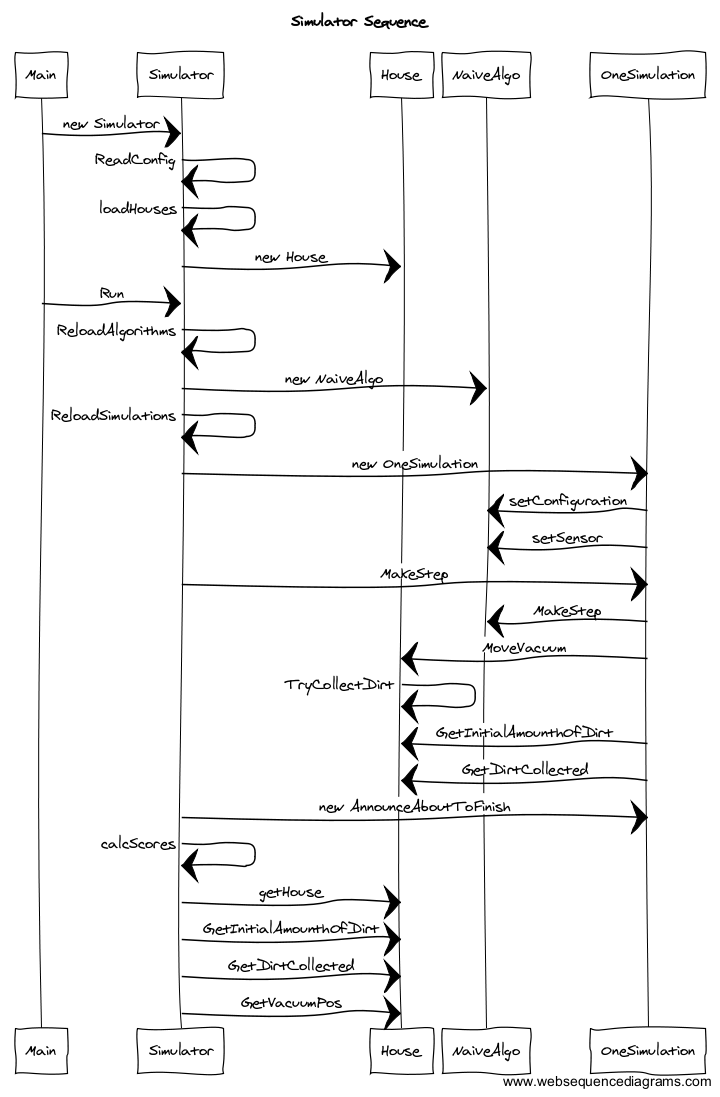
In exercise 1 we handle following potential erroneous situations:

* User passes unrecognized or incorrect arguments. In that case “usage” information is printed to cout.
* Configuration file “config.ini” is not present or cannot be opened.

Each error is printed to the user, and the program will exit after the message is printed.

**Class Diagram**

****

**Sequence Diagram**