# Waste Incinerator Service

## Requirements Analysis

### Structure

analyzing the natural language requirements text we found out the following entities that should be somehow modelled:

- ServiceArea
  - Home
  - BurnIn port
  - BurnOut port
  - WasteIn
  - AshOut
- WIS
- OpRobot
- DDRRobot
- Incinerator
- WasteStorage
  - Scale
  - RP
  - WRP
- AshStorage
  - MonitoringDevice
  - Sonar
  - Led

### Interaction and Behaviour

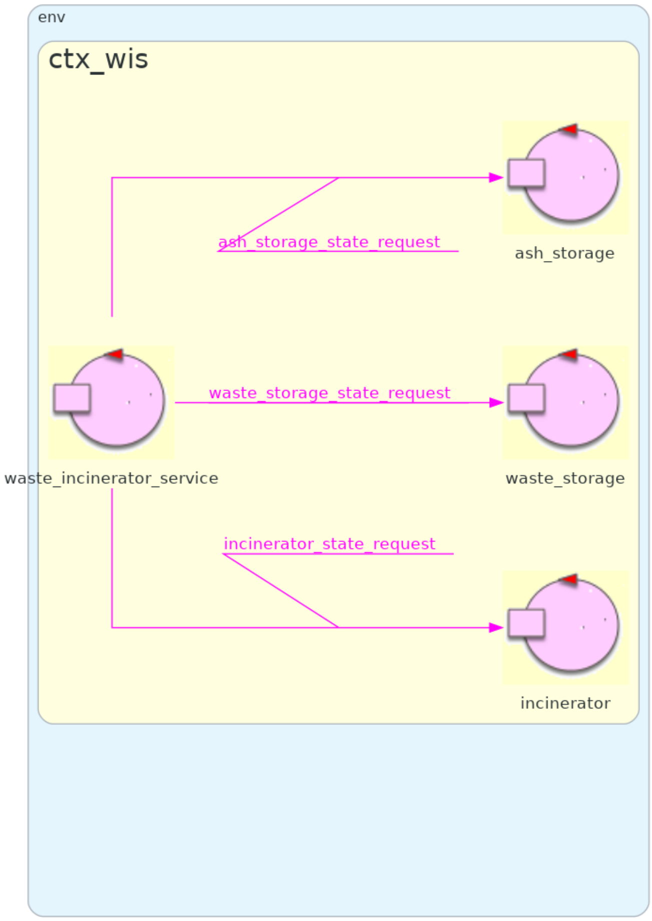
PROF

By requirements we inferred the following informations that need to be modelled:

- activationCommand: (?) -> (Incinerator)
- endOfBurning: (Incinerator)->(OpRobot,WIS)
- BurnInPortInfo: (OpRobot? (fixed) )->(Incinerator)
- BurnOutPortInfo: (?)->(Incinerator)
- scaleInfo: (Scale)->(WIS)
- sonarInfo: (Sonar)->(WIS)

> [!NOTE]
> we merged Interactions and Behaviour sections because at this stage of the project for the majority of this informations we don't know yet if they will be modelled as POJOs' methods or messages between actors

### Model Diagram


env
ctx_wis
ash_storage_state_request
ash_storage
waste_storage_state_request
waste_incinerator_service
waste_storage
incinerator_state_request
incinerator

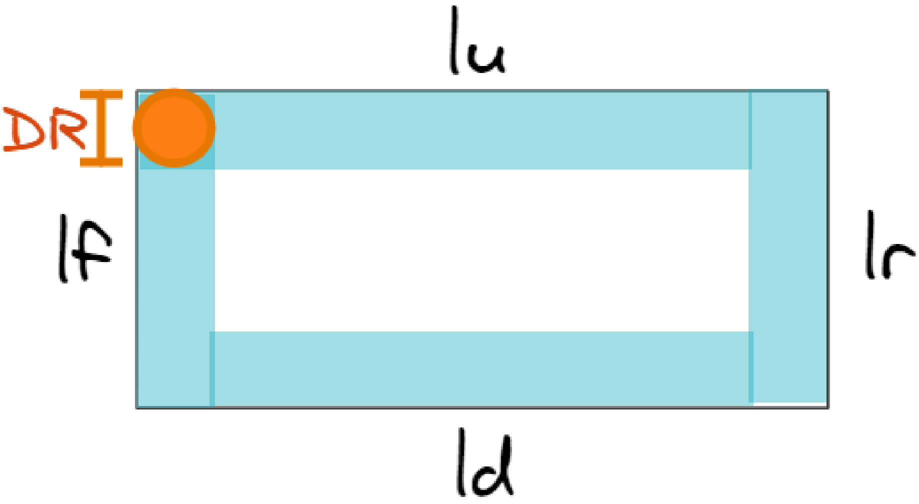
wis_systemArch

PROF

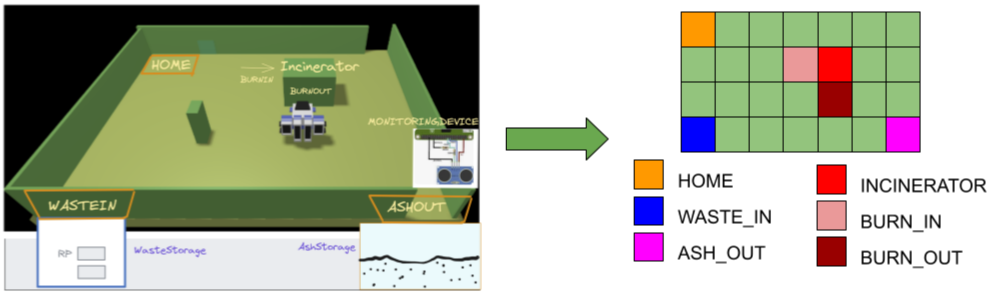
### Service Area Model

the **ServiceArea** is modelled as an Euclidean space delimited by its edges(similar to what has been done in the BoundaryWalk and RobotCleaner projects):

- the **perimeter edge** has length `lf+ld+lr+lu`
- being the ServiceArea rectangular we have `lf=lr && ld==lu`
- we define `DR=2R` being `R` the radius of the DDRRobot circumscribable circle



Given this model we have that **Home**, **BurnIn**, **BurnOut**,**WasteIn**, **AshOut** are all modelled as collections of cells in the serviceArea:



**DDRRobot model**

The **OpRobot**, defined in the requirements as the robot controlled by the WIS, makes use of a DDRRobot (and its control software) given by the customer, we link the detailed definition of DDRRobot and its qak control software.

### Priority

PROF

- core buisness = Incinerator => probably i'll have to analyze the MonitoringDevice before the raspberry and the Scale

### SSGUI

- WasteStorage state
- AshStorage state
- Incinerator state
- OpRobot state

## Problem Analysis

### Entities Models

- ServiceArea -> other serviceAreas models
- WIS -> service (sends/recives messages)

- OpRobot -> service (given as service)
- DDRRobot -> service (sends/recives messages)
- Home -> colections of cells inside the serviceArea
- Incinerator -> actor (sends/receives messages) || pojo
  - BurnIn port
  - BurnOut port
- WasteIn -> collections of cells || coordinates
- WasteStorage -> context?
  - Scale -> actor (if sends/recives messages) || pojo (if Scaleinfo is retrieved using a method)
- RP -> pojo
  - WRP -> pojo attribute || config param of Scale actor, so that Scale sends the number of RPs, not the weights
- AshOut -> collections of cells || coordinates
- AshStorage -> context? (outside service area?)
- MonitoringDevice: -> actor? || context?
  - Sonar -> actor (sends/receives messages) || pojo
  - Led -> actor (receives messages) || pojo (is quite simple)