Waste Incinerator Service

Requirements Analysis

Structure

Analyzing the natural language requirements text, we identified the following entities that should be modeled:

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

Interaction and Behavior

From the requirements, we inferred the following information that needs to be modeled:

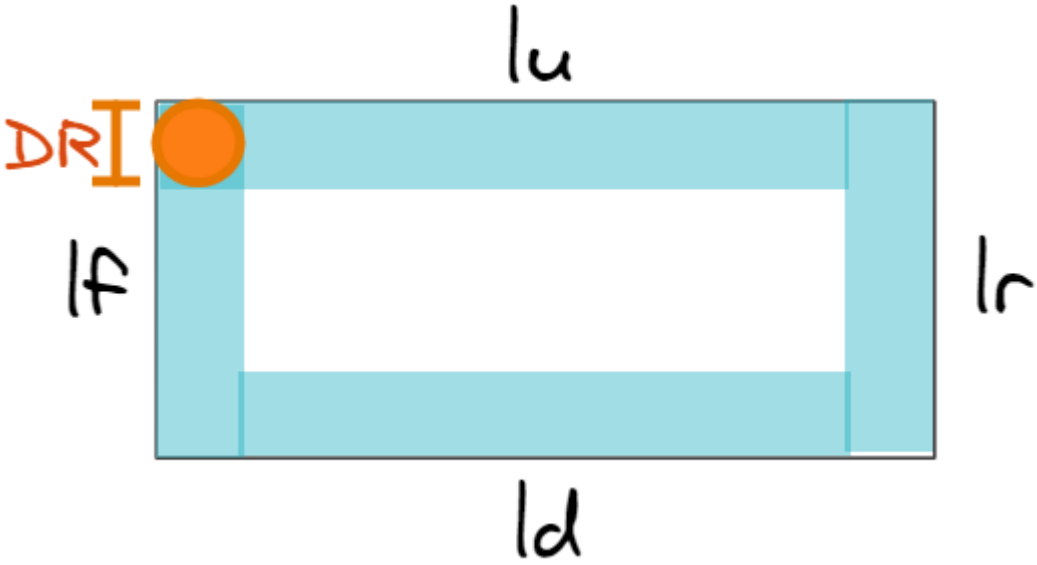
Information	**Source**	**Destination**
activationCommand	unspecified	Incinerator
isBurning	Incinerator	unspecfied
ashLevel	Sonar	unspecified
loadRP	WasteStorage	OpRobot
burnRp	OpRobot	Incinerator
loadAsh	unspecified (OpRobot || Incinerator || WIS)	unspecified (OpRobot || Incinerator || WIS)
unloadAsh	OpRobot	AshStorage

> [!NOTE]
> We merged the Interactions and Behavior sections because at this stage of the project, for the majority of this information, we don't know yet if it will be modeled as POJOs' methods or messages between actors.

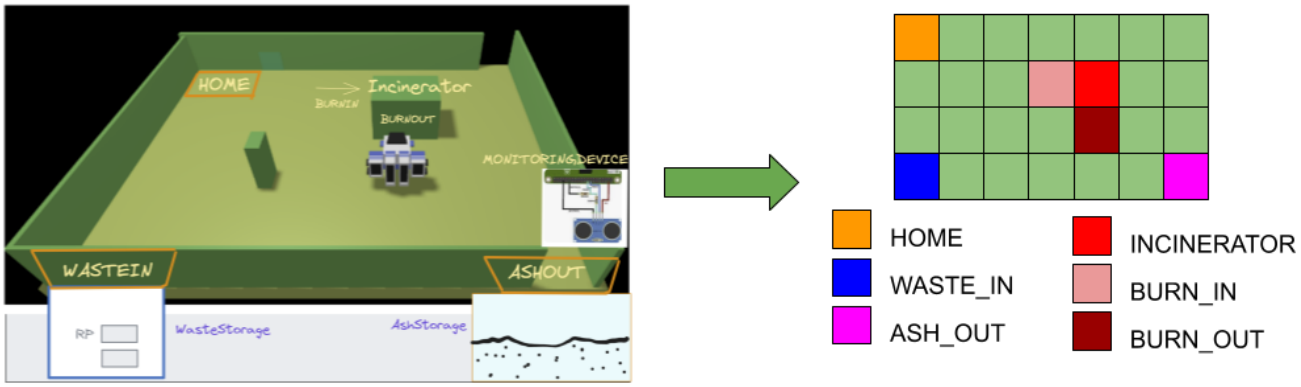
Service Area Model

The **ServiceArea** is modeled 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` and `ld = lu`.
- We define `DR = 2R`, where `R` is the radius of the DDRRobot's circumscribable circle.

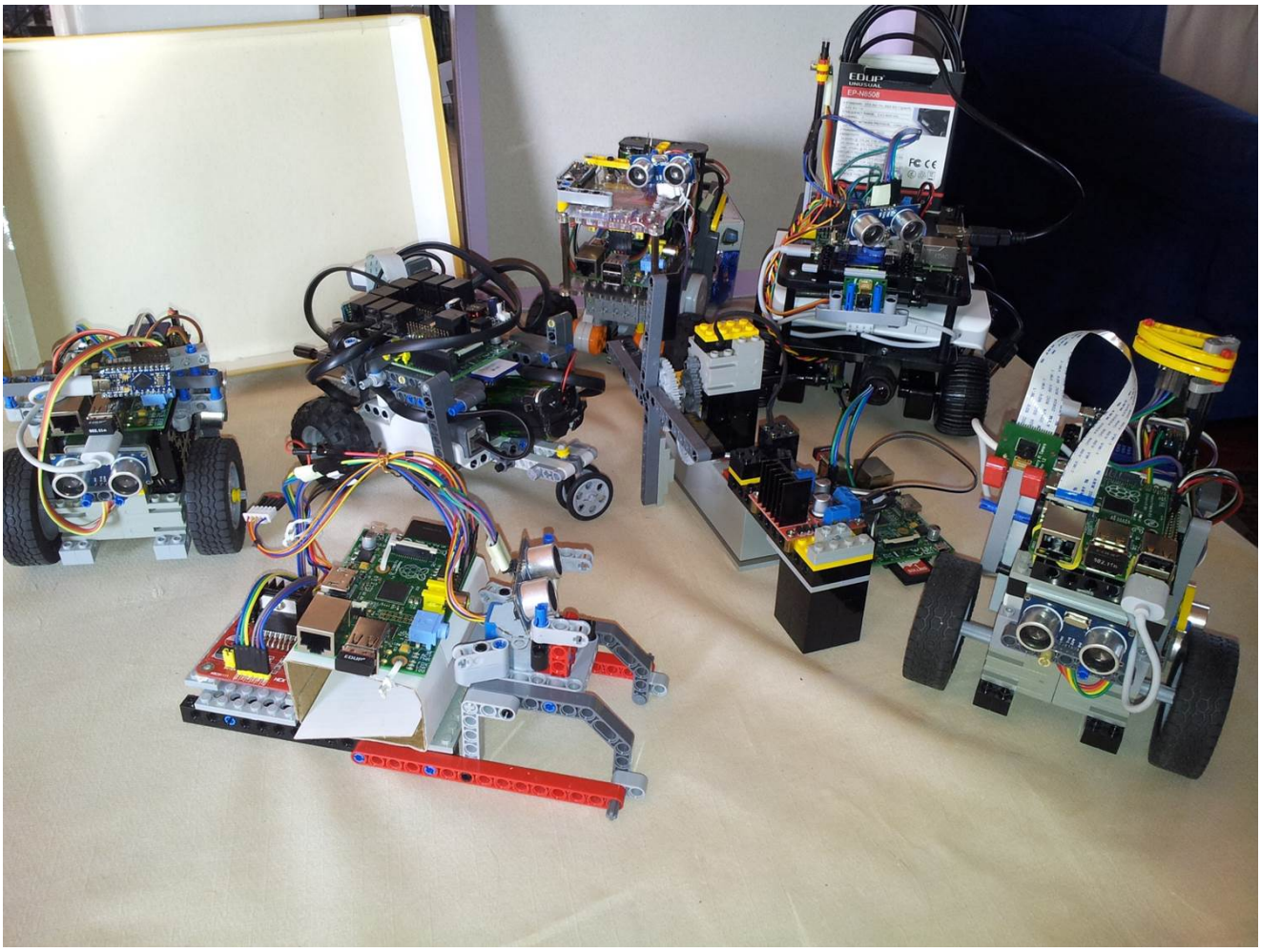


Given this model, we have that **Home**, **BurnIn**, **BurnOut**, **WasteIn**, and **AshOut** are all modeled as cells in the ServiceArea:



DDRRobot Model

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



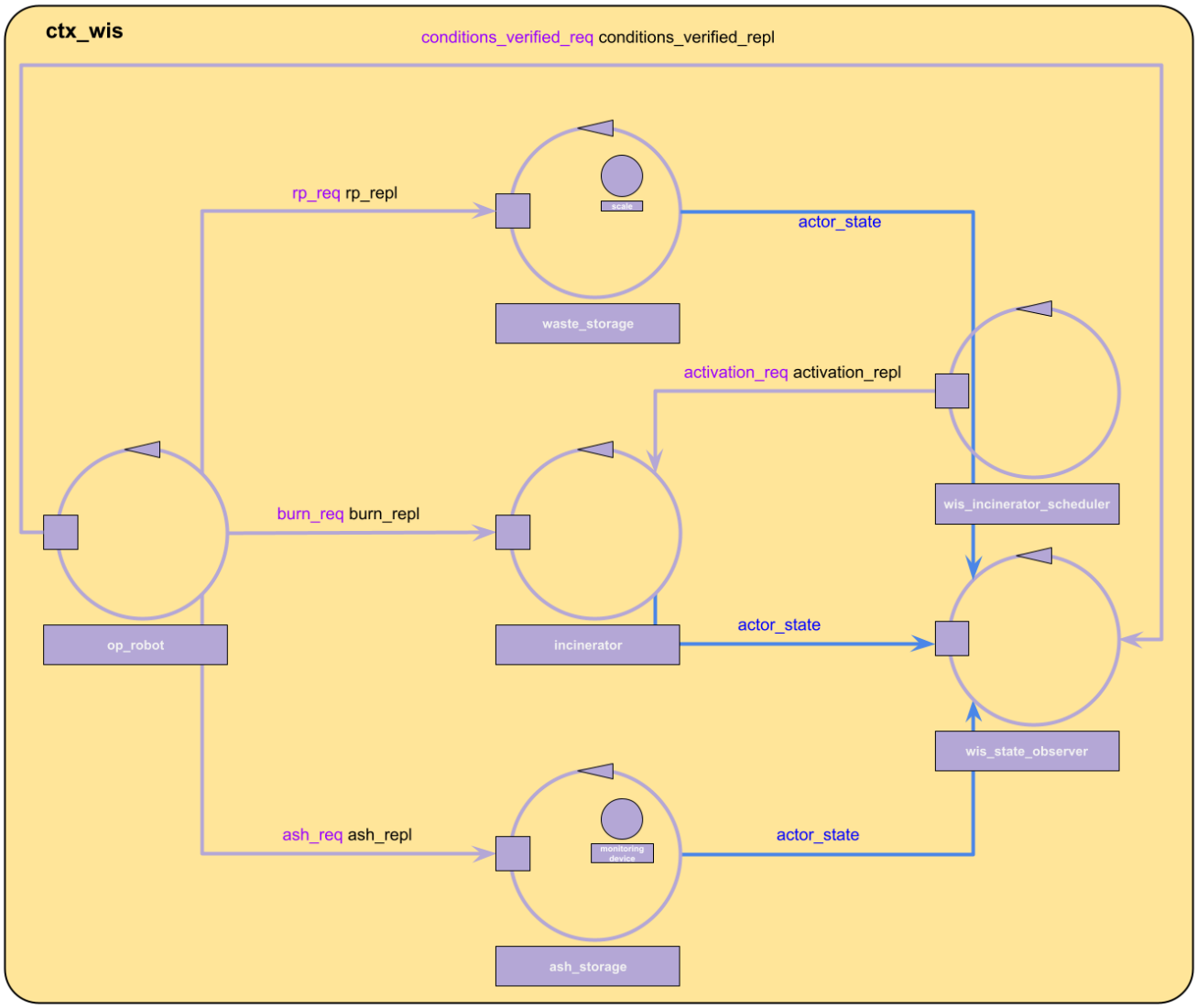
WIS, WasteStorage, Incinerator, and AshStorage Models

WasteStorage, **Incinerator**, and **AshStorage** need to exchange messages according to the requirements, so they are modeled as actors.
Scale can be modeled either as an actor or a POJO inside WasteStorage. For now, we will represent it as a POJO and **defer the discussion to a later moment**.
The same applies to the **MonitoringDevice**, **Sonar**, and **Led**, which will be modeled as POJOs inside AshStorage for now.

In the first prototype of the model, the OpRobot will contain the majority of the business logic for the realization of a 'WIS cycle' (move to waste storage, load an RP, move to incinerator, burn the RP, move to ash storage, unload ash).
However, there are **no specifications** about where to inject the business logic (injecting all business logic into WIS could also be an option), so we **defer this discussion to the Problem Analysis**.

WIS will be divided into two sub-components: a **WISStateObserver** that will act as an observer of WasteStorage, AshStorage, and Incinerator, checking that the constraints are satisfied, and a **WISIncineratorScheduler** that will activate the incinerator.
This organization of WIS is not evident from the requirements; hence, it may change in future sprints.

The following diagram illustrates the structure based on requirements:


ctx_wis
conditions_verified_req conditions_verified_repl
rp_req rp_repl
scale
actor_state
waste_storage
activation_req activation_repl
wis_incinerator_scheduler
burn_req burn_repl
op_robot
incinerator
actor_state
wis_state_observer
ash_req ash_repl
monitoring device
actor_state
ash_storage