Final project

Autonomous Software Agents - UniTn 2021/2022

Alberto Casagrande - 229362

Introduction

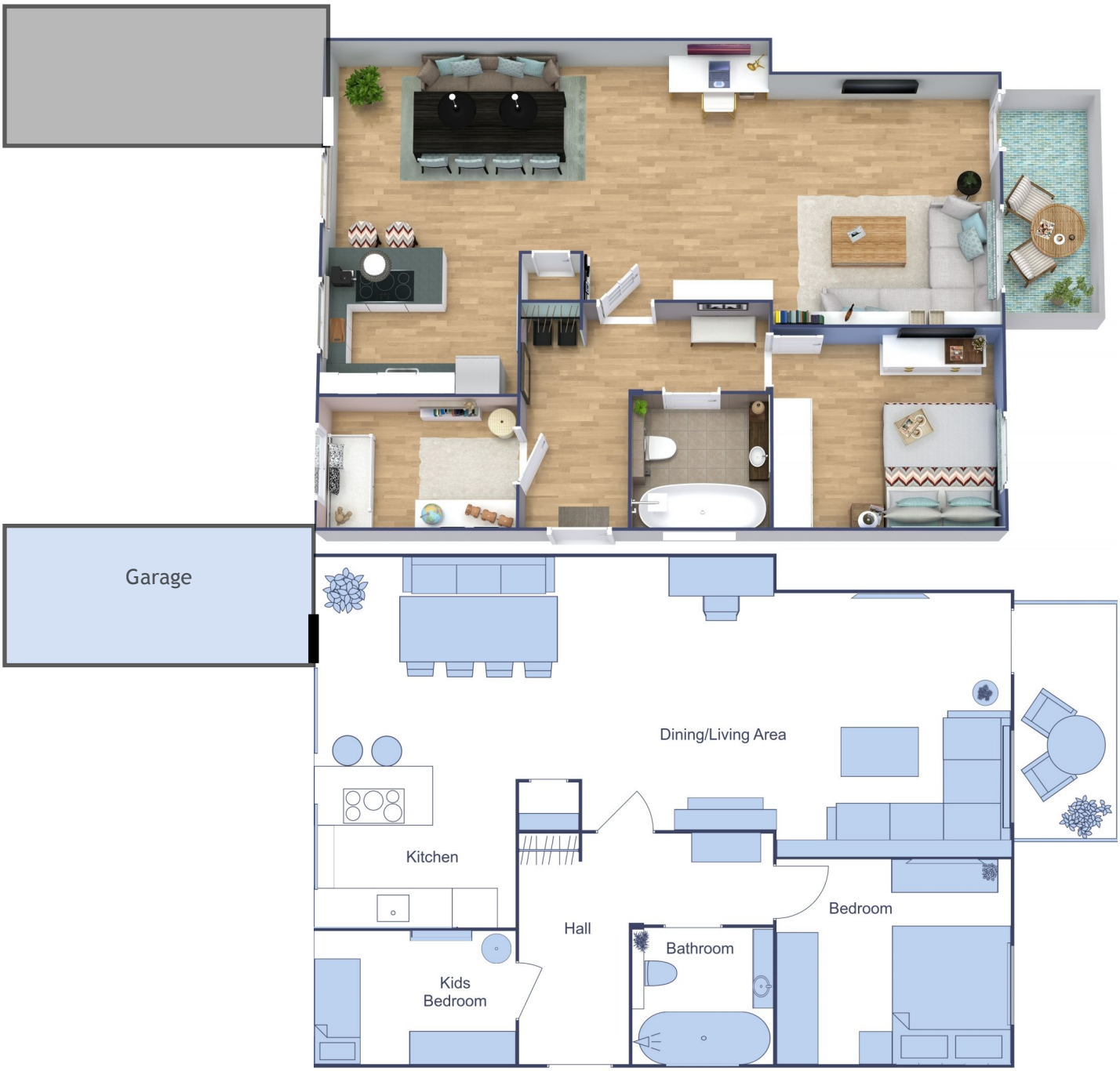
The concept of "**smart house**" refers to an environment designed to help people in their everyday activities. Nowadays, human beings always seek refuge, relaxation, security and prosperity at home, hence influencing the quality of life. The smart house technology aims to increase home automation and security with reduced energy consumption. These smart houses consist of various intelligent sensors and actuators operating on behalf of different agents that may have conflicting objectives. Sensors and actuators are used to supervise the house environment and sometimes the occupant, to communicate with other devices and to support or assist the occupant in their daily activities.

To handle this problem, I thought of a solution based on dividing responsibilities based on the **roles** of various devices/agents. Some of these devices are stupid, i.e. they do not implement any kind of intelligent capability - they are not autonomous and are not able to make decisions, while other devices are controlled by an agent, and are therefore able to make decisions autonomously based on the context and situation they are in. This implies that intelligent devices are aware of the context, which they recognize through different sensors installed in the various rooms. Each agent is therefore responsible for monitoring, controlling and managing a certain function of the house. For example, there is the agent that is responsible for ensuring the **safety** of the residents. An additional agent has been designed to manage and minimize **energy consumption**, while another agent takes care of the **cleaning**.

All of these agents must interact with each other and coordinate, as it is possible that they may have **conflicting** objectives. For this reason, in addition to the device-specific agents, a **house agent** has also been introduced, which deals with more general functions, and interacts with the other agents. The house agent has been designed to have a complete knowledge about the status of the house. Since coordination between agents has been introduced only at the end of the course, it has not been implemented. However, agents can communicate and in particular the house agent can share some of its beliefs about the state of the house to the

other agents. Finally, as for the agent responsible for cleaning the house (vacuum cleaner), it was implemented using a **planning domain**. In this planning domain, the agent must achieve the goal of cleaning all rooms in the house according to a plan constructed based on the preconditions and effects associated with each action.

House description and blueprint



The house considered is a one-story house for residential use, designed for a family of 3 people (parents and child). As we can see from the floor plan, the house is composed of 5 rooms which are connected through the Hall.
At the bottom of the figure there is the entrance to the house, which gives access to the hall. From the hall it is possible to access the two bedrooms and the bathroom, which have independent doors. The single bathroom will be accessed from a hallway outside of either bedroom, allowing anyone to access the bathroom without going through a bedroom.
After entering the house, if you proceed straight, you will find the door that gives access to the living area, which is large enough to allow for a sofa and coffee table, plus a loveseat or chair, to allow all residents to relax together.
The living room is directly connected to the kitchen through the dining room.
From the living area, it is possible to get access to a private outdoor space, such as a balcony or patio. It's a great place to store bikes, scooters, and other sports gear.
In addition, there is also a garage that offers a parking space. From the garage, with an independent entrance we enter the dining / living room.

Rooms

Note: not all the devices listed here in the room descriptions have been implemented due to lack of time.

Garage

In the garage there is an automated swinging blind, a main light, and a wall charger for the electric car. The garage is used by residents to park and recharge the electric car, and to store supplies. It is not heated. From the garage it is possible to enter the house directly thanks to an independent entrance.

Kitchen

The kitchen is accessible only through the dining/living area, and includes a fridge, an oven, an induction hob, a sink and a dishwasher. There is a main light in the middle of the room. In addition, there is a large window to ensure lighting during the day.
The kitchen is mainly used for cooking lunch and dinner, but also for preparing breakfast.

Dining room

The dining room is placed between the kitchen and the living room. The dining room is essentially composed of the dining table, which is illuminated with dedicated lights. Also in this case there is a window.

Living room

The living room is the biggest room of the house and consists of a sofa, a coffee table and a television set on the opposite wall. There is a large window that lights up the room during the day. In addition, there is a main light that illuminates the environment and an additional soft light positioned near the television. Below the television there is an audio system consisting of soundbar, subwoofer and speakers.
In addition, in the living room there is a video intercom connected to an external camera, and also the IoT Gateway (device that allows to link the house things with the Internet).
Finally, in the living room is placed the charging station of the vacuum cleaner, as it is the main area of the house that needs to be cleaned regularly.
The living room is mainly used during the evening hours, after the residents have returned home from work, but can also be used during the day by the child. Usually, people in the living room watch television, which can be controlled either via the remote control, but also via the smart speaker.

Kids bedroom

This room is located near the main entrance. The kids bedroom is relatively small, there is a bed and a small desk as well as a shelf where books are kept. There is also a window, and the illumination of the room is guaranteed by a light positioned in the center of the room itself.
There is also a small lamp near the bed which is mainly used by the child for reading.

Bathroom

The bathroom is the smallest room of the house. It consists of a toilet, a sink, a mirror with a special light, a washing machine and a bathtub. It has obviously a main light and there’s a window above the bathtub. The bathroom is located between the two bedrooms, and is also easily accessible from the living room through the hall.

Bedroom

This bedroom contains a double bed, a wardrobe and another piece of furniture with a built-in mirror. There is a large window right above the bed. Moreover, there is a main light plus additional separated lights (one light for the mirror and a lamp next to the bed). The bedroom entrance is close to the bathroom entrance.

Hall

The hall has the function of welcoming people entering the house. It connects all the various areas of the house and is equipped with a main light. In this area of the house is located the control panel of the security system, together with a siren.

Features common to all rooms

All rooms are equipped with LED lights to ensure efficiency. All rooms except the garage have a smart speaker installed. Additionally, all rooms have at least one light switch, and all windows have roller shutters. As for the windows, they are all equipped with an electric mechanism for automatic opening and closing.
As for the burglar alarm, outdoor detectors are installed on doors and shutters and signal their opening to the control panel in case of intrusion. All doors are equipped with smart locks and of course, motion detectors, thermal cameras and acoustic sensors are installed in all rooms.
Furthermore, all rooms are equipped with a gas/smoke detector, which are connected to the control panel in the hall.
Finally, each room has an independent thermostat to control the electric underfloor heating and the air conditioner (all the rooms apart from the bathroom and the garage have an air conditioner installed).

Devices

Lights

LED lights provide illuminations to the rooms. They can be controlled either through switches or remotely using the specific smartphone application. Light status is either `on`, `off`, or `disconnected`. Actions that can be done on the lights are `switchOnLight()` , `switchOffLight()` , `disconnectLight()`. Prerequisites for `turnOn()` is (`off`). Lights are controlled both by the house agent and by the consumption agent. Each light consumes 10Wh of electricity when switched on.

Soft Light

Soft lights are implemented as a device which extends Light and inherits all its methods and attributes. In this case, the soft light consumes 5Wh of electricity when switched on.

Dishwasher

The dishwasher has a built-in detergent dispensers and is able to detect automatically the size of the load and the amount of grime on dishes in order to adapt the cleaning cycle adjusting water and energy use to operate efficiently for even small loads. Status are `running`, `off` or `paused`. Actions are `turnOn()`, `turnOff()`, `pause()`, `resume()`. Prerequisite for `turnOn()` is (`off`). Prerequisite for `turnOff()` is (`running` OR `paused`). Prerequisite for `pause()` is (`running`). Prerequisite for `resume()` is (`paused`). The dishwasher consumes 1,2 kWh of electricity per washing cycle. Water consumption is around 10/12 liters per load.
This device is implemented as a 'stupid' device, since it is not involved in any kind of intelligent scenario.

Fridge

The fridge has the ability to keep track of supplies and is therefore always able to provide residents with its status, which can be `full`, `half_full` and `empty`. Actions that can be done include `setFull()`, `setHalfFull()`, `setEmpty()`. Status of the fridge is usually updated by the house agent.
Furthermore, the fridge is capable of adapting the internal temperature (and therefore the power required) based on the amount of food present at a given moment. The fridge consumes an average of 100W per hour (of course it depends on the set temperature). Like the dishwasher, in the project the fridge is not involved in any intelligent scenario.

Vacuum cleaner

The vacuum cleaner cleans safely and efficiently all the rooms of the house.
The status of the vacuum cleaner are `on`, `off`. The battery level, from 0 to 100, is mapped into the following three discrete states: `fully_charged`, `half_charged` , `need_recharge`, `charging`. Actions are `turnOn()`, `turnOff()`, `move(to)`, `clean(room)`.
Prerequisites for `turnOn()` action are ((`off` OR `charging`) AND (`fully_charged` OR `half_charged`)). Prerequisite for `turnOff()` is `on`. Typically, the robot vacuum cleaner cleans the house when no one is around. If

the robot vacuum cleaner has the goal of cleaning a certain area of the house but people are present in those rooms at that time, it will delay cleaning until the rooms are empty. Of course, it is also possible to control remotely the vacuum cleaner though a smartphone application.

Television

The smart television is mainly used to watch films via Amazon Prime Video, but can also be used to watch simple tv programs. The TV status is either `on` or `off`. The activity status are `watching_DTT` and `watching_prime`. Actions include `turnOn()`, `turnOff()`, `setChannel(c)`, `openPrimeVideo()`, `selectFilm(f)`. Prerequisites for `setChannel()` are (`on` AND `watching_DTT`) while prerequisites for `selectFilm()` are (`on` AND `watching_prime`). When the television is off, the activity is automatically set to `watching_DTT`.
A specific scenario involving television has been implemented. During the afternoon, Mario and Anna decide to watch a movie in the living room (so they turn on the television, open Prime Video and start a film).
The agent of the house recognizes the presence of people in the living room and receives a message from the television that a film has started. In this case the windows are closed automatically (to avoid outside noise) and the shutters lowered to create a cinema atmosphere. The main light is turned off (if it is on) and the soft light behind the TV is turned on. Television consumes 200W per hour when turned on.

Door lock

Smart door locks allow access via fingerprint authentication. Access to the house is also possible using the smartphone, a password or a simple mechanical key. Status of each door lock includes whether the door is locked (`door_locked`) or not (`door_not_locked`). Actions are `unlockDoor()` and `lockDoor()`.
Prerequisite for `unlockDoor()` is `door_locked`. Prerequisite for `lockDoor()` is `door_not_locked`. The smart door lock is installed on the entrance door of the house and is controlled by the security agent. The door is locked every night from 23 to 7.

Windows

The windows have an automatic opening and closing system, which works in coordination with the heating system. The status of each window can be `opened`, `closed`. Actions available are `open()`, `close()`. Each window itself is not a smart device, but since they can be automatically opened and closed, they are

controlled by the house agent and the security agent, which can decide autonomously when to open or close each window. For example, as it is mentioned in the television device, when a film starts, the window of the living room is closed automatically by the house agent in order to avoid outside noise.

Shutters

The shutters can be raised and lowered via an electric mechanism. Status can be `down`, `half`, `up`. Actions available are: lower the roller shutter completely (`lower()`), raise it completely (`raise()`), and set it halfway up (`setHalfway()`). Prerequisites for `lower()` are (`half` OR `up`). Prerequisites for `raise()` are (`down` OR `half`). Prerequisites for `setHalfway()` are (`down` OR `up`). Shutters of the house are controlled by the house agent and by the security agent. All shutters of the house are lowered at 23 (by the security agent), and raised in the morning when residents get up (by the house agent).

Metrics

Electricity cost

The cost of electricity is 0.30€/kWh during the day and 0.25€/kWh during the night. Consequently, it is cheaper to buy electricity from 7pm to 8am. For this reason, many appliances are used during nighttime hours. Furthermore, all the appliances are programmed to use as little energy as possible (for example, the television in the living room is automatically turned off when no one is watching it for more than 15 minutes, lights are turned off in rooms where there is no-one and the vacuum cleaner cleans only dirty rooms). The average annual consumption of electricity is 4500kWh.

Cleaning time

Vacuum cleaner robot takes 25 minutes to clean each room of the house. Overall, the floor cleaning time can be estimated as 3 hours. However, sometimes the vacuum cleaner robot does not clean all the rooms in a row, but cleans only the rooms that are dirty at that time of the day in order to save time and energy.

People and Agents

People

Residents in the house include Mario and Anna. People can be in one room at a time, or out of home. They usually wake up at around 7.30 and then they have

breakfast together at around 8 in the kitchen. In the scenario considered, the residents remain at home all day. They spend the morning in the dining room working on the computer. At around 12 they move to the kitchen for lunch and from 15.30 in the afternoon they relax in the living room watching a movie on television (they turn on the television and open Amazon Prime Video, selecting a Harry Potter movie). Around 18 Mario goes back to his computer in the dining room and at 19 Mario and Anna go to the kitchen to have dinner. Finally, around 20.15/20.30 they both go to the living room where they stay until 23, when they go to bed.

Agents

- **Security agent:** the security agent is responsible for locking the entrance door, lowering all shutters and closing all windows overnight.
    1. Lock the entrance door at 23;
    2. Unlock the entrance door at 7;
    3. Close all windows at night;
    4. Lower the roller shutters at night.

- **Energy consumption agent:** this agent deals with managing the energy consumption of the house (the consumption of electricity). The goal of this agent is to minimize electricity expenditure and its consumption, trying to use resources (home appliances) as efficiently as possible.
    1. Turn off the lights in a room when nobody is there;
    2. Turn off the television when no one is watching it for more than 15 minutes.

- **House agent**: the house agent assists residents by taking autonomous decisions, while still being responsive to residents' behaviors.
    1. Switch on the lights when people enter in a room;
    2. Create a cinema atmosphere in the living room in case there are people watching a movie (in the simulated scenario this is done every day at 16 when people start watching a movie);
    3. Tell the vacuum cleaner where to clean (basically tells the vacuum cleaner which rooms there is no one in);
    4. Raise the roller shutters when residents get up in the morning (when there is no one left in the bedroom);
    5. Shares its beliefs about empty rooms with the consumption agent in order to make it turn off the lights;
    6. Shares its beliefs about the status of the shutters and the windows with the security agent.

- **Robot vacuum cleaner agent (planning agent)**: the robot is able to autonomously move among all the rooms through doors and clean them daily. It is a goal-based agent, so its goal is to traverse the rooms, clean up all the dirt, and return to the charging station. Therefore I decided to use planning to address the problem of moving among the different rooms of the house and clean the dirt. I have thus defined the planning domain as follows:

```
(define (domain vacuum_agent)
    (:requirements :strips)
    (:predicates
       (room ?source)
       (robot ?vacuum)
       (at_room ?vacuum ?source)
       (adjacent ?source ?destination)
       (on ?vacuum)
       (empty ?room)
       (clean ?room)
       (off ?vacuum)
    )

        (:action Move
            :parameters (?vacuum ?source ?destination)
            :precondition (and
                  (room ?source)
                  (room ?destination)
                  (robot ?vacuum)
                  (at_room ?vacuum ?source)
                  (adjacent ?source ?destination)
                  (on ?vacuum)
             )
             :effect (and
                  (not (at_room ?vacuum ?source))
                  (at_room ?vacuum ?destination)
             )
        )

        (:action Clean
            :parameters (?vacuum ?room)
            :precondition (and
                  (robot ?vacuum)
                  (room ?room)
                  (at_room ?vacuum ?room)
                  (empty ?room)
                  (on ?vacuum)
             )
             :effect (and
                  (clean ?room)
             )
        )

        (:action TurnOn
            :parameters (?vacuum)
            :precondition (and
                  (robot ?vacuum)
                  (off ?vacuum)
             )
             :effect (and
                  (not (off ?vacuum))
                  (on ?vacuum)
             )
        )

        (:action TurnOff
            :parameters (?vacuum)
            :precondition (and
                  (robot ?vacuum)
                  (on ?vacuum)
             )
             :effect (and
                  (not (on ?vacuum))
                  (off ?vacuum)
             )
        )
```

as can be seen, among the predicates (properties of the objects we are interested in), the most relevant are certainly ***adjacent***, which defines which rooms are adjacent, ***empty***, which establishes which rooms there is no one in at that moment, ***at_room*** that models the position of the agent in the house, ***clean***, used to indicate whether rooms are clean or not and **on** / **off** to specify whether the robot vacuum cleaner is on or off. I then defined the four actions that the robot vacuum cleaner can perform: **Move**, **Clean**, **TurnOn** and **TurnOff**. The Move action can be performed by the robot vacuum cleaner to move from one room to another in the house, but the necessary precondition is that these are adjacent. As for the Clean action instead, the only real constraint that the vacuum cleaner has to meet is the fact that it can only start cleaning rooms where no one is around. That's why I have included as a precondition `empty ?room`. Of course, both actions require the robot vacuum cleaner to be on. The effect of applying the clean action is that the room is clean. Regarding the knowledge of the house by the robot vacuum cleaner agent, I assumed that the agent already knows the structure of the house and the various rooms (which rooms are there and how they are connected). Turning to the part about the objective of the vacuum cleaner agent, every morning at 9 the robot vacuum cleaner has the goal of cleaning the sleeping area of the house (bathroom, bedrooms and hall), while at 21.30 in the evening it has the goal of cleaning the area consisting of living room, kitchen and dining room. Before starting to clean the rooms of the house, the vacuum cleaner is informed by some sensors about which rooms are not clean so that it only cleans dirty rooms. So the plan it follows daily to clean the various rooms might change. In addition, since the robot vacuum cleaner has been programmed to clean rooms only when they are empty (no people present), it may sometimes happen that the robot delays the execution of its plan until this condition occurs. For this reason, it could be that in some cases the robot vacuum cleaner interacts with the agent of the house. In particular, as I mentioned before, the house agent shares its beliefs about which rooms are empty with the vacuum cleaner. Once the vacuum cleaner has finished cleaning, it returns by itself to the charging base (included in the agent's goal).

Implementation

Sensors and agent perception

The implemented agents clearly have different goals, as mentioned earlier. For this reason, each agent has its own view of what the environment around it is.
All of the agents' knowledge is encoded in their **belief set**, which is nothing more than a kind of dictionary containing key-value pairs. The value can be either true or false, depending on the current status of that particular device under consideration. These beliefs represent what the agent believes to be true about the environment. Agents are always updated on changes affecting the devices under their control through **sensors**. The sensors have been implemented using the `notifyChange()` function on the property of the device we want to control. Here is a simple example, in which I have implemented the lights sensor:

```
var lightsGoals = []
for (let l of this.lights) {
    let lightGoalPromise = new Promise( async res => {
        while (true) {
            let status = await l.notifyChange('status')
            this.log('sense: light ' + l.name + ' switched ' + status)
            this.agent.beliefs.declare('light_on '+l.name, status=='on')
            this.agent.beliefs.declare('light_off '+l.name, status=='off')
        }
    });

    lightsGoals.push(lightGoalPromise)
}
yield Promise.all(lightsGoals)
```

Every time the attribute `status` of a light changes, it's detected by the `notifyChange()` method and this causes the asynchronous callback to be executed. So, we update the beliefs of the agent using the `declare()` method.
Similar solutions have been implemented to detect changes in the status of roller shutters, windows, people, door locks and also to detect when people are watching a movie in the living room (for the film scenario).
The **house agent** is designed to have full control over what is happening inside the house and therefore is always up-to-date on changes that occur. Its beliefs include the state of:

1. Lights (on or off?) —> `light_on kitchenLight: true/false`
2. Windows (opened or closed?) —> `window_opened kitchenWindow: true/false`
3. Shutters (up, half or down?) —> `shutter_up kitchenShutter: true/false`
4. People (which room each person is in) —> `person_in_room Mario kitchen: true/false`
5. Rooms (empty or not?) —> `empty Kitchen: true/false`
6. Television (watching a movie?) —> `television_mode watching_prime: true/false`

Instead, the **security agent** has a partial (limited) view of the environment. In fact, in its knowledge base, it has only the beliefs concerning the status of the shutters, windows and front door lock (`door_locked entrance: true/false`). All other information about the house and the status of the devices is not available for the security agent.
The same thing concerns the **consumption agent**, who only knows which rooms are free (there is nobody) and which are not. The reason is that it is the only information it needs to achieve its goals.
Finally we have the **vacuum cleaner agent** that apart from the structure of the house (the rooms and how they are connected to each other) is also updated on which rooms are empty at a given time and which rooms are not clean.

Agents acting in a shared environment

All agents have been designed to respond reactively to stimuli coming from the environment. To do this, sensors had to be implemented, as described in the previous section. In addition, to interact with the environment and the various devices, goals and intentions were developed and associated with the various agents. By executing the `*exec` method in the intentions, basically, the states of the various devices inside the house are changed. However, not all agents are capable of making autonomous decisions. For example, the agent in charge of home security does not implement any intelligent behavior; it simply locks the entrance door, closes the windows, and lowers the shutters at 23 each day.
On the contrary, the consumption agent is responsible, for example, for turning off the light in rooms where there is no one. So, if a person leaves a room and there is no one left in that room, the agent is expected to turn off the light completely independently. To do this it was necessary to use, within the appropriate intention (*SwitchOffIntention*), the `notifyAnyChange()` method on the beliefs of the agent itself. As previously stated, the consumption agent's belief set only contains information about which rooms are empty. When one of these changes, all rooms are checked, and in case a room is found where there is no one but the light is on, it is turned off immediately.

```
let status = await this.agent.beliefs.notifyAnyChange()
var belief = this.agent.beliefs
for (const [key, value] of Object.entries(belief)){
   if(key != "genericObservers"){
      if(key.split(" ")[0] == 'empty' && value==true){
         if(this.house.rooms[key.split(" ")[1]].devices[0].status == 'on'){
           this.log('sense: there is no one in the ' + key.split(" ")[1])
           this.log("Let's turn off the", this.house.rooms[key.split(" ")[1]].devices[0].name)
           this.house.rooms[key.split(" ")[1]].devices[0].switchOffLight()
         }
       }
   }
}
```

In this case, key contains for example ‘empty kitchen’ and value contains either true or false. So if the value is true and the light status of that room is ‘on’, the agent turns it off.
And here is an example in which Mario moves from the dining room to the kitchen and Anna moves from the living room to the kitchen:

```
in_room Anna kitchen
in_room Mario kitchen

house_agent>SensePersonIntention#1 sense: person Anna moved to kitchen
house_agent>SensePersonIntention#1 sense: person Mario moved to kitchen
house_agent             Belief changed: person_in_room Anna kitchen
house_agent             Belief changed: not empty kitchen
house_agent             Belief changed: not person_in_room Anna living_room
house_agent             Belief changed: empty living_room
house_agent             Belief changed: person_in_room Mario kitchen
house_agent             Belief changed: not person_in_room Mario dining_room
house_agent             Belief changed: empty dining_room

consumption_agent       Belief changed: not empty kitchen
consumption_agent       Belief changed: empty living_room
consumption_agent       Belief changed: empty dining_room
consumption_agent>SwitchOffIntention#7 sense: there is no one in the dining_room
consumption_agent>SwitchOffIntention#7 Let's turn off the diningLight
consumption_agent>SwitchOffIntention#7 sense: there is no one in the living_room
consumption_agent>SwitchOffIntention#7 Let's turn off the livingLight

Dining room light status is off
Living room light status is off
```

Another autonomous function of the consumption agent consists of turning off the television in the living room when no one is watching it for more than 15 minutes. For this purpose a specific goal and intention have been implemented in which I have applied the `notifyChange()` method to the `mm` parameter of the *Clock*, since it changes by 15 minutes each time. Here is the code:

```
while (true) {
  let status = await Clock.global.notifyChange('mm')
  if(this.agent.beliefs.check('empty living_room')){
     if(this.television.status == 'on'){
         this.log("Nobody is watching TV in living_room. Let's turn it off.")
         this.television.turnOff()
     }
  }
}
```

So every 15 minutes we check if the living room is empty through the `check()` method applied to that specific belief of the agent. If there is no one in the living room and the television is on, the agent turns it off.
As a last example, I would like to talk about the vacuum cleaner agent, which, as mentioned, is the only agent who implements the planning to build on the spot the plan to move between the various rooms and clean them. The agent that controls the vacuum cleaner is directly associated with the device, and it was implemented simply by instantiating an object of the *Agent* class and passing the

associated device as a parameter. The online planner was used for the implementation. As I said before, it is assumed that the agent already knows the structure of the house. In this regard, I have created a simple function that initializes the agent's beliefs as soon as it is instantiated. The function is called `init_vacuum_belief()` and can be found in the main scenario.js file. It simply consists in declaring through the `declare()` function which are the initial beliefs of the agent, such as:

```
vacuumAgent.beliefs.declare('robot vacuum’)
vacuumAgent.beliefs.declare('off vacuum')
vacuumAgent.beliefs.declare('room kitchen')
vacuumAgent.beliefs.declare('room living_room')
vacuumAgent.beliefs.declare('at_room vacuum living_room')
vacuumAgent.beliefs.declare('adjacent living_room kitchen')
```

In this way, the vacuum cleaner knows exactly where it is, what the rooms are, and whether they are directly connected to each other or not.
The four actions (Move, Clean, TurnOn and TurnOff) have been defined by specifying the parameters, preconditions and effects of each action, in a way that reflects exactly what is described in the planning agent section.
The four actions extend a specific class that I created called *myAction*, which, in turn extends **pddlActionIntention**. This makes it possible to use the `checkPrecondition()` and `applyEffect()` methods to check that the preconditions are met and apply the effects of each action to the agent's beliefs.
Finally, it must be considered that the agent operates in a shared environment with the other devices where there is the presence of people moving around. In the defined scenario, as previously mentioned, the robot vacuum cleaner can execute the plan to clean the designated rooms only when there is no one in them. Consequently, some times there is a possibility that no plan is found and the intention fails. For this reason, the **RetryFourTimesIntention** class was developed, which extends **Intention** and is used to retry the execution of the plan when all the preconditions of the actions occur. To do this, I modified the `*exec` method slightly by replacing `notifyAnyChange()` with `notifyChange()` on the specific agent belief that we want to be true to allow the execution of the plan (`empty roomName`). So basically the agent waits for the specific room to become empty before trying again to execute the plan.

Agent interaction and communication

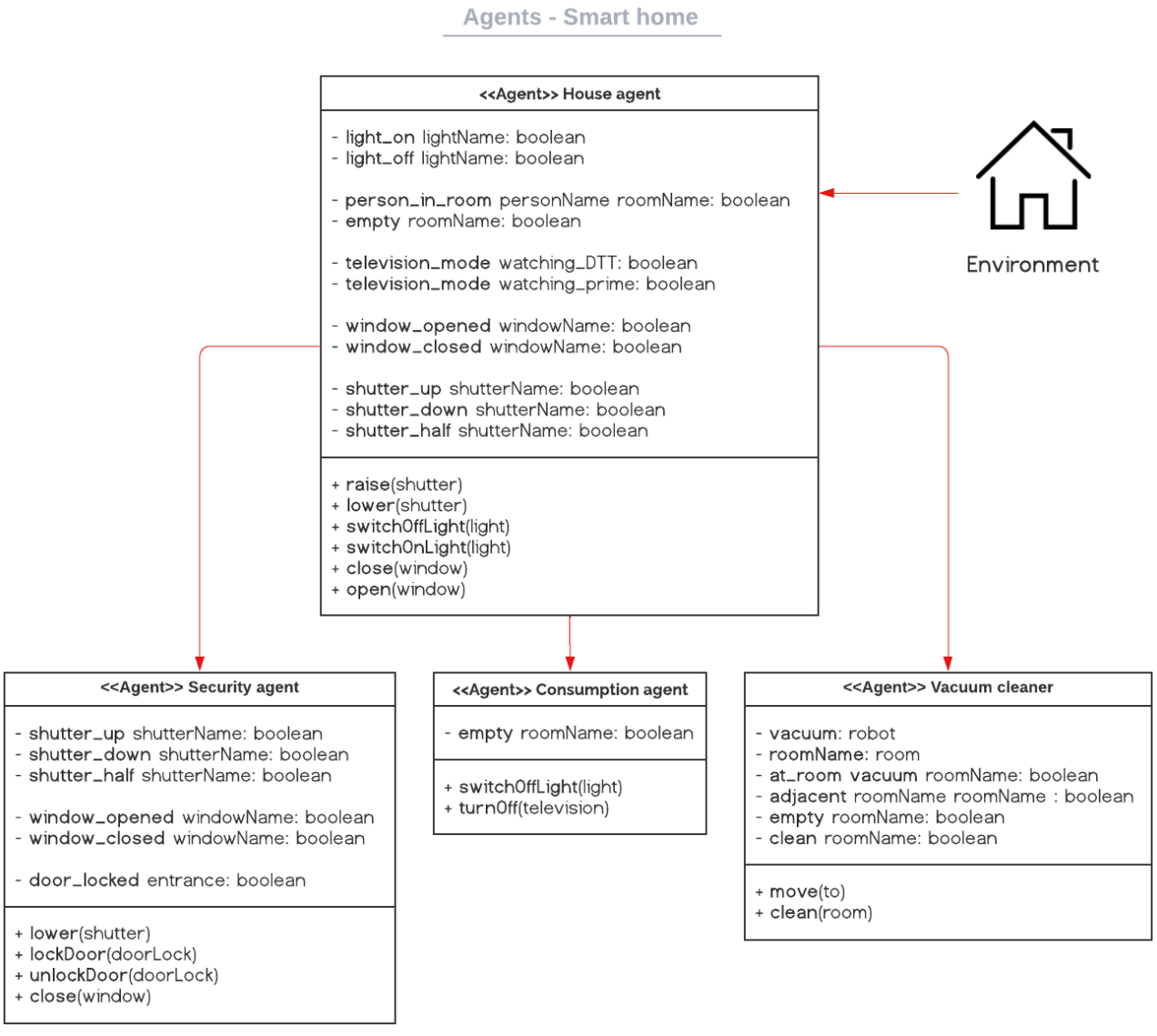
Since coordination between agents has been introduced only at the end of the course, it has not been implemented. However, agents can communicate and in particular the house agent can share some of its beliefs about the state of the house to the other agents. Specifically, the house agent communicates the status

of the shutters and windows to the security agent, while it informs the consumption agent and the vacuum cleaner agent about which rooms are empty.

```
var sensor_empty = (agent) => (value,key,observable) => {
     let predicate = key.split(' ')[0]
     let arg1 = key.split(' ')[1]
     if (predicate=='empty')
          key = 'empty '+arg1;
     else{
          return;
     }
     value?agent.beliefs.declare(key):agent.beliefs.undeclare(key)
}

houseAgent.beliefs.observeAny( sensor_empty(consumptionAgent) )
```

Thanks to this trick, it is possible to make agents communicate, even if it is not a real form of communication, which would require a specific protocol that allows agents to interact with each other and coordinate.



Scenarios

This section describes some scenarios implemented.

Film scenario

In this scenario, through sensors, the house agent detects when someone starts watching a movie on television and intervenes autonomously to create the appropriate atmosphere for watching the movie. Specifically, it lowers the blinds, closes the window to avoid outside noise, turns off the main light, and turns on the soft light near the television.
Once the residents have finished watching the movie, the house agent brings the situation back to normal.

Log:

```
Day 0:  16:00
Television status is on
Television mode is watching_prime
house_agent>SenseFilmIntention#3 sense: television mode is changed to
watching_prime
house_agent                    Belief changed: not television_mode watching_DTT
house_agent                    Belief changed: television_mode watching_prime
house_agent>FilmIntention#4    Film scenario! Let's create a cinema atmosphere!

Living room light status is off
Living room window status is closed
Living room soft light status is on
Living room shutter status is down

house_agent>SenseLightsIntention#0 sense: light livingLight switched off
house_agent>SenseWindowsIntention#5 sense: window livingWindow has been closed
house_agent>SenseLightsIntention#0 sense: light softLightTV switched on
house_agent>SenseShuttersIntention#6 sense: shutter livingShutter lowered down
house_agent                    Belief changed: not light_on livingLight
house_agent                    Belief changed: light_off livingLight
house_agent                    Belief changed: not window_opened livingWindow
house_agent                    Belief changed: window_closed livingWindow
house_agent                    Belief changed: light_on softLightTV
house_agent                    Belief changed: not light_off softLightTV
house_agent                    Belief changed: not shutter_up livingShutter
house_agent                    Belief changed: shutter_down livingShutter

Day 0:  18:00
Television mode is watching_DTT
in_room Mario dining_room
house_agent>SenseFilmIntention#3 sense: television mode is changed to
watching_DTT
house_agent>SensePersonIntention#1 sense: person Mario moved to dining_room
house_agent                    Belief changed: television_mode watching_DTT
house_agent                    Belief changed: not television_mode
watching_prime
house_agent>FilmIntention#4    Residents are no longer watching a movie on TV

Living room light status is on
Living room window status is opened
Living room soft light status is off
Living room shutter status is up
```

Lights scenario

The lights are turned on by the house agent when someone enters a room and are turned off by the consumption agent when there is no one in a room. In this example both Mario and Anna moved from the kitchen to the living room. As a result, the light in the living room is automatically turned on, while the kitchen light is turned off by the consumption agent, as there is no one left in the kitchen.

Log:

```
in_room Anna living_room
in_room Mario living_room

house_agent>SensePersonIntention#1 sense: person Anna moved to living_room
house_agent>SensePersonIntention#1 Let's turn on the livingLight
house_agent>SensePersonIntention#1 sense: person Mario moved to living_room

Living room light status is on

house_agent                    Belief changed: person_in_room Anna living_room
house_agent                    Belief changed: not empty living_room
house_agent                    Belief changed: not person_in_room Anna kitchen
house_agent                    Belief changed: person_in_room Mario living_room
house_agent                    Belief changed: not person_in_room Mario kitchen
house_agent                    Belief changed: empty kitchen

consumption_agent              Belief changed: not empty living_room
consumption_agent              Belief changed: empty kitchen

house_agent>SenseLightsIntention#0 sense: light livingLight switched on
house_agent                    Belief changed: light_on livingLight
house_agent                    Belief changed: not light_off livingLight

consumption_agent>SwitchOffIntention#7 sense: there is no one in the kitchen
consumption_agent>SwitchOffIntention#7 Let's turn off the kitchenLight

kitchen light status is off
```

Vacuum cleaner scenario

Every day at 21.30 the vacuum cleaner agent has the goal of cleaning the kitchen, living room and dining room. At that time, however, Mario and Anna are in the living room and consequently no plan is found (the condition that all the rooms to be cleaned are empty is not satisfied). The agent then waits for the living room to become empty in order to proceed with the cleaning. In this case, at 23 Mario and Anna go to the hall and then to the bedroom and therefore the vacuum cleaner has the green light to clean the various rooms. This behavior could be improved by allowing the vacuum cleaner to clean the other rooms while waiting for the living room to be empty.

Log:

```
Day 0:  21:30
vacuum                          Trying to use intention RetryFourTimesIntention
to achieve goal RetryGoal#28[object Object]
vacuum>RetryFourTimesIntention#27 Intention started
vacuum                          Trying to use intention OnlinePlanning to achieve
goal PddlGoal#27 goal:clean living_room,clean kitchen,clean dining_room,at_room
vacuum living_room
vacuum>OnlinePlanning#28        Intention started

Day 0:  21:45
vacuum>OnlinePlanning#28        No plan found
vacuum>OnlinePlanning#28        [object Object]
vacuum>OnlinePlanning#28        Intention failed: Plan not found
vacuum                          Failed to use intention OnlinePlanning to achieve
goal PddlGoal#27 goal:clean living_room,clean kitchen,clean dining_room,at_room
vacuum living_room: Plan not found
…

Day 0:  22:00
vacuum>RetryFourTimesIntention#27 wait for something to change on beliefset
before retrying for the 2th time goal PddlGoal#27 goal:clean living_room,clean
kitchen,clean dining_room,at_room vacuum living_room

Day 0:  23:00
in_room Anna hall
in_room Mario hall

vacuum                          Belief changed: not empty hall
vacuum                          Belief changed: empty living_room

vacuum>RetryFourTimesIntention#27 living_room is now empty, we can proceed to
clean up
vacuum                          Trying to use intention OnlinePlanning to achieve
goal PddlGoal#27 goal:clean living_room,clean kitchen,clean dining_room,at_room
vacuum living_room
vacuum>OnlinePlanning#29        Intention started

… other stuff

Day 0:  23:30
vacuum_agent>OnlinePlanning#31 Plan found:
vacuum_agent>OnlinePlanning#31 - (turnon vacuum)
vacuum_agent>OnlinePlanning#31 - (clean vacuum living_room)
vacuum_agent>OnlinePlanning#31 - (move vacuum living_room kitchen)
vacuum_agent>OnlinePlanning#31 - (clean vacuum kitchen)
vacuum_agent>OnlinePlanning#31 - (move vacuum kitchen dining_room)
vacuum_agent>OnlinePlanning#31 - (clean vacuum dining_room)
vacuum_agent>OnlinePlanning#31 - (move vacuum dining_room living_room)
vacuum_agent>OnlinePlanning#31 - (turnoff vacuum)
vacuum_agent>OnlinePlanning#31 Starting sequential step (TurnOn vacuum) Effect:
not off vacuum,on vacuum
vacuum_agent>TurnOn#32          Intention started
Turn on vacuum cleaner
vacuum_agent                    Belief changed: not off vacuum
vacuum_agent                    Belief changed: on vacuum
vacuum_agent>TurnOn#32          Intention success
vacuum_agent>OnlinePlanning#31 Starting sequential step (Clean vacuum
living_room) Effect: clean living_room
vacuum_agent>Clean#33           Intention started
Cleaning  living_room

Day 1:  00:00
```

```
vacuum_agent                    Belief changed: clean living_room
vacuum_agent>Clean#33           Intention success
vacuum_agent>OnlinePlanning#31 Starting sequential step (Move vacuum living_room
kitchen) Effect: not at_room vacuum living_room,at_room vacuum kitchen
vacuum_agent>Move#34            Intention started
Move from living_room to kitchen

Day 1:  00:15
vacuum_agent                    Belief changed: not at_room vacuum living_room
vacuum_agent                    Belief changed: at_room vacuum kitchen
vacuum_agent>Move#34            Intention success
vacuum_agent>OnlinePlanning#31 Starting sequential step (Clean vacuum kitchen)
Effect: clean kitchen
vacuum_agent>Clean#35           Intention started
Cleaning  kitchen

Day 1:  00:45
vacuum_agent                    Belief changed: clean kitchen
vacuum_agent>Clean#35           Intention success
vacuum_agent>OnlinePlanning#31 Starting sequential step (Move vacuum kitchen
dining_room) Effect: not at_room vacuum kitchen,at_room vacuum dining_room
vacuum_agent>Move#36            Intention started
Move from kitchen to dining_room
vacuum_agent                    Belief changed: not at_room vacuum kitchen
vacuum_agent                    Belief changed: at_room vacuum dining_room
vacuum_agent>Move#36            Intention success
vacuum_agent>OnlinePlanning#31 Starting sequential step (Clean vacuum
dining_room) Effect: clean dining_room
vacuum_agent>Clean#37           Intention started
Cleaning  dining_room

Day 1:  01:15
vacuum_agent                    Belief changed: clean dining_room
vacuum_agent>Clean#37           Intention success
vacuum_agent>OnlinePlanning#31 Starting sequential step (Move vacuum dining_room
living_room) Effect: not at_room vacuum dining_room,at_room vacuum living_room
vacuum_agent>Move#38            Intention started
Move from dining_room to living_room
vacuum_agent                    Belief changed: not at_room vacuum dining_room
vacuum_agent                    Belief changed: at_room vacuum living_room
vacuum_agent>Move#38            Intention success
vacuum_agent>OnlinePlanning#31 Starting sequential step (TurnOff vacuum) Effect:
not on vacuum,off vacuum
vacuum_agent>TurnOff#39         Intention started
Turn off vacuum cleaner
vacuum_agent                    Belief changed: not on vacuum
vacuum_agent                    Belief changed: off vacuum
vacuum_agent>TurnOff#39         Intention success
vacuum_agent>OnlinePlanning#31 Intention success
vacuum_agent                    Succesfully used intention OnlinePlanning to
achieve goal PddlGoal#29 goal:clean living_room,clean kitchen,clean
dining_room,at_room vacuum living_room,off vacuum
vacuum_agent>RetryFourTimesIntention#29 Intention success
vacuum_agent                    Succesfully used intention
RetryFourTimesIntention to achieve goal RetryGoal#30[object Object]
```

Changes from the previous version

Compared with the previous version of the code, some changes have been made to solve some critical issues. All these changes concern the vacuum cleaner, which has now been implemented by instantiating a new object of the *Agent* class (previously, the vacuum cleaner was implemented by extending the *Agent* class in the device). All code was then adapted to these changes. In particular, to have access to the Vacuum cleaner device methods within the *pddl* actions, it was necessary to slightly modify the *Agent* class so that it accepts the device associated with the agent as a parameter.

```
constructor (name, device) {
        this.name = name
        this.id = nextId++
        this.device = device
        …
}
```

The vacuum cleaner agent is therefore implemented with:

```
var vacuumAgent = new Agent('vacuum_agent', house.rooms.living_room.devices[5])
```

Thus, in the **exec* of the various pddl actions, it is possible to call the device methods via:

```
this.agent.device.move(destination)
```

Finally, in addition to the two previously defined *Move* and *Clean* actions, the *TurnOn* and *TurnOff* actions have also been added, which allow the vacuum cleaner to switch on and off independently. Each action has now a duration, which is handled via a simple *setTimeout(res, duration)* function inside the *checkPreconditionAndApplyEffect()* function. In this way the *Clean* action lasts much longer than the *Move* action for example.

Source code organization

First of all, the architecture at the base of the system is the one present at the link https://github.com/marcorobol/Autonode.js. On top of this architecture, I have implemented my own scenario. All developed code can be found in the folder `/myworld`.

Firstly, in order to handle efficiently the entire house, I decided to implement some classes:

- Person (**`People.js`**): the Person has some attributes, including `in_room`, which tells the room that person is in, at a given time. The only method for this

class is `moveTo(destination)`, which allows us to move people around the house.

- Room (**`Room.js`**): each room is an object that has a `name`, a list of `devices` and a list of names of other rooms that are accessible from that room (`doors_to`).
- Devices (**`Devices.js`**): here I have implemented a class for each device.
- House (**`House.js`**): the house is the main object of the system, and it is composed by a list of `people`, `rooms` and `utilities`. People and rooms are created using the classes defined above, while for the utilities, there is only the `electricity`, which is simply an **Observable**.

- Sensors:
  - **`LightSensor.js`**: the purpose of this file is to update the beliefs of the agent about the status of each light in the house.
  - **`PersonSensor.js`**: the lights sensor concept is replicated for detecting residents' movement within the house through the various rooms. So basically, for each person in the house, we detect when the attribute `in_room` changes through the `notifyChange`() method.
  - **`ShutterSensor.js`**: exactly the same functioning as the lights sensor. When a shutter is raised/lowered, it is detected by the sensor and the agent belief set is updated.
  - **`WindowSensor.js`**: again, this sensor detects window movement and updates the agent's belief set.
  - **`FilmSensor.js`**: here it is checked if there is a movie starting in the living room. To do so, I have applied the `notifyChange`() method to the `activity` attribute of the television. When it changes, the agent's beliefs are updated accordingly.
  - **`DoorLockSensor.js`**: this sensor detects when the status of the door lock of the entrance door changes and updates the belief set of the security agent.

- Intentions:
  - **`CloseWindows.js`**: intention used by the security agent to close all windows at night.
  - **`ManageDoorLock.js`**: intention used by the security agent to lock the entrance door at night.
  - **`LowerShutters.js`**: intention used by the security agent to lower all the shutters at night.

- **`TurnOffTV.js`**: intention used by the consumption agent to turn off the television when no one is watching it for more than 15 minutes.
- **`SwitchOffLights.js`**: intention used by the consumption agent to switch off the lights when there is nobody in a room.
- **`RaiseShutters.js`**: intention used by the house agent to raise the shutters of the house in the morning when there is no one left in the bedroom.
- **`FilmScenario.js`**: intention used by the house agent in order to apply the film scenario in the living room (switch on soft light, close the window, lower the shutter…)
- **`VacuumScenario.js`**: file used to implement the planning scenario for the vacuum cleaner agent.

- Scenario (**`scenario.js`**): in this file all the pieces are put together to create the simulation of the smart house.

Everything related to planning is in the **pddl** folder, while *Agent*, *Beliefs*, *Goal*, and *Intention* are in the **bdi** folder.

Here is the link to the GitHub repository:
https://github.com/albertocasagrande99/SmartHome