Role of IoT in Smart Grid Infrastructure

An Informal/Semi-formal/Formal Specifications
submitted in partial fulfillment of the Stage I requirements for

SOFTWARE ENGINEERING AND MANAGEMENT
SS G562

in

Master of Engineering (Software Systems)

by

Sanchita Badkas (2019H1030520P) (Group 4)

Soumak Roy (2019H1030024P) (Group 4)

Under the supervision of

Dr. Shashank Gupta

Assistant Professor
Department of Computer Science and Information Systems

Birla Institute of Technology and Science

Pilani, Pilani Campus, Rajasthan (India)

October, 2020

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

Department of Computer Science & Information Systems

Outline of the Project: The project involves a software system that is deployed at the power grid as well as in the buildings of the city. The system plays a vital role in managing the power requirements of the city in an efficient manner. In order to conserve the electricity, the system deployed at buildings ensures that electricity isn't wasted unnecessarily by controlling the devices by taking data from various sensors as input. Also, the system sends the users alerts about the upcoming maintenance of their devices in order to ensure that the devices don't consume more electricity than requires. Using these alerts the users can replace/repair the appliances in order to avoid any short circuit incidents. The system is also deployed at the main power grid of the city inorder to capture the data from the systems deployed at user homes and ensure sufficient electricity supply to the city. Also, using this data the system equips the city council with knowledge about the usage patterns of electricity in certain areas of the city. This knowledge along with inputs from the city council regarding the future projects in the city, the system would make certain investment decisions. These investments would be used for strengthening and repairing of the power distribution infrastructure by the council. The electricity usage tends to peak whenever the city hosts any events like a concert, political campaign or festival celebrations. The power grid equipment can get overloaded in these cases. In order to avoid the power failures due to overloading of the equipment, the system decides to deploy backup equipment in order to maintain load balancing so that the equipements don't get overloaded. This deployment of backup depends on the electricity usage patterns of the city. The city council is also planning to move towards clean energy generation by reducing their dependency on fossil fuels. In order to achieve this the city has deployed solar panels and windmills. Unfortunately, these sources are dependent on weather conditions and aren't sufficient to fulfill the power requirements of the whole city. Thus, based on the weather conditions the system needs to inform the thermal power plant of the amount of electricity it needs to generate in order to fulfill the city's requirements. The system is an essential part of the city's plan of moving towards a smart power grid.

Informal Specification 1: The city needs a supply of 2500KWH of energy. It wants to reduce its dependence on fossil fuels and hence is moving towards harnessing the natural resources in order to generate clean energy. However, the electricity generated by these is dependent on weather conditions due to which the supply fluctuates. Hence, the remaining needs of energy needs to be fulfilled using the thermal power plant in the city. The thermal plant needs an estimate of the energy that it needs to generate in order to ensure continuous flow of electricity. The solar panels work only when the weather is sunny and the pollution in air is low. The pollution of the city is measured using sensors and categorized into 'Good' or 'Bad'. When the day is sunny and air quality is good the solar panels are capable of generating 1500KWH of energy. However, it's contribution drops to zero when it isn't sunny. Also, when the pollution index is high the solar panels are capable of generating only 500KWH of electricity. The city has also deployed windmills. These windmills at full capacity i.e when wind speed is between 15-24m/s generates upto 1000KWH of electricity. When the wind speed is 6-15m/s it generates upto 500 KWH of energy. However, if the speed of the wind drops below 6m/s then the windmill isn't able to generate electricity. Also, the windmill is designed to survive winds of 25m/s beyond which the windmill will suffer from damages and won't be able to generate energy. The system needs to output the amount of electricity the thermal power plant needs to generate based on the amount of electricity the solar panels and windmills generate.

Semi-Formal Specification:

<u>Notation used</u>:

- Air quality : G: Good B: Bad
- Wind speed: A : Less than 6m/s, B : Between 6-15m/s, C: Between 15-24m/s, D : Above 25m/s

Is it Sunny?	Air quality	Wind speed	Electricity to be generated by the thermal power plant
Yes	G	Don't care	1000 KWH
Yes	B	A	2000 KWH
Yes	B	B	1500 KWH
Yes	B	C	1000 KWH
Yes	B	D	2000 KWH
No	Don't care	A	2500 KWH
No	Don't care	B	2000 KWH
No	Don't care	C	1500 KWH
No	Don't care	D	2500 KWH

Formal Specification:

<u>Operations:</u>

1. classify_wind(): integer → char
2. compute_thermal(): char, char, char → integer

<u>Description:</u>

1. The **classify_wind()** function takes wind speed as input and classifies it into one of the four categories as follows, A (<6m/s), B(6-15m/s), C(15-24m/s) and D(>24m/s).
2. The **compute_thermal()** function takes whether it is sunny outside (Y/N), air quality(Good-G/Bad-B) and this wind speed category as input and computes the amount of electricity that the thermal power plant will have to generate under these conditions.

<u>Axioms:</u>

- ➢ classify_wind(wind_speed): if wind_speed < 6 → speed_label = A
- ➢ classify_wind(wind_speed): if (wind_speed >= 6 and wind_speed < 15) → speed_label = B
- ➢ classify_wind(wind_speed): if (wind_speed >= 15 and wind_speed <= 24)→ speed_label = C
- ➢ classify_wind(wind_speed): if wind_speed > 24 → speed_label = D
- ➢ compute_thermal(sunny, air quality, speed_label): if (sunny = No and (speed_label = A or speed_label = B)) → 2500

- ➢ compute_thermal(sunny, air quality, speed_label): if ((sunny = No and speed_label = B) or (sunny = Yes and air_quality = B and (speed_label = A or speed_label = D))) → 2000
- ➢ compute_thermal(sunny, air quality, speed_label): if ((sunny = No and speed_label = C) or (sunny = Yes and air_quality = B and speed_label = B)) → 1500
- ➢ compute_thermal(sunny, air quality, speed_label): if (sunny = Yes and (air_quality = G or (air_quality = B and speed_label = B))) → 1000

Informal Specification 2: Considering a huge dependency of the city on electricity, the power grid station needs to ensure it's steady supply. In order to do that, it has kept a set of devices as backup. These same equipment can also be deployed during the peak hours of electricity usage in order to have a proper load balancing so that the equipment is not overloaded as overloading might result in reducing the life of the equipment and increasing the cost of maintenance. Based on the previous experience, it has been observed that the electricity usage surges during the morning hours and goes down as the day passes by. Also, if the city hosts any event like a concert, political campaign or festive celebrations a rise in electricity usage is seen. 4 sets of required equipment is kept as backup and needs to be deployed for load balancing. The day is divided in 3 parts as morning hours/office hours, evening hours and night hours. During normal days when the city isn't hosting any event 2 sets of backup equipment is deployed during the morning and evening hours. No load balancing is required during the night hours if there is no event in the city. However, the requirement of electricity increases during the events and this is largely dependent on the size of the event. The city categorizes it's events into large sized (attendance>5000), medium sized (attendance between 1000-5000) or small sized (attendance<1000). In case of small sized events the power grid can efficiently serve the cities electricity needs and no change would be made in the regular operations of load balancing. In case the city hosts a large or medium scale event during the office hours all the backup equipment are deployed. For a large scaled event during the evening hours, all equipment is deployed. For all the remaining scenarios 2 sets are deployed for ensuring smooth supply of electricity in the city.

Semi-Formal Specification 2:

Notation used:

- Time of the day: F : 7am-3pm, S: 3pm-11pm, T: 11pm-7am
- Scale of event: L: Large, M: Medium, S:Small

Time of day	Event going on?	Scale of event	Deploy # standby equipments
F	TRUE	L	4
F	TRUE	M	4
F	TRUE	S	2
F	FALSE	Don't care	2
S	TRUE	L	4
S	TRUE	M	2
S	TRUE	S	2
S	FALSE	Don't care	2
T	TRUE	L	2
T	TRUE	M	2
T	TRUE	S	0
T	FALSE	Don't care	0

Formal Specification 2:

Operations:

1. get_third(): time → char
2. find_scale(): long → char
3. load_balancer(): char, bool → integer
4. get_attendance(): none → long

Description:

1. The **get_third()** function takes a timestamp as input and computes which third of the day it falls in, first, second or third. If the timestamp falls in first third of the day i.e between 7am-3pm the function would return F, if the time falls between 3pm-11pm the function returns S and if it falls between 11pm-7am the function returns T
2. The **find_scale()** function takes the potential attendance of an event as input and categorises its scale into Small(S), Medium(M) or Large(L). If the people attending the

event is less than 1000, it's categorized as S. For an event with 1000-5000 people the function returns M and for events with people more than 5000 the function would return L

3. The **load_balancer()** function takes two inputs: Third of the day from the get_third() function, Boolean value about whether the city is hosting an event or not and utilizes the find_scale() and get_attendance() functions to determine the scale of the event if the event is happening. It outputs the number of equipment sets that needs to be deployed additionally for load balancing.
4. The **get_attendance()** function returns attendance of an event, taking it as input from the event organisers

Axioms:

- get_attendance(): → attendance
- get_third(current_time): (if current_time > 0700 and current_time < 1500) → third = F
- get_third(current_time): (if current_time > 1500 and current_time < 2300) → third = S
- get-third(current_time): (if current_time < 0700 or current_time > 2300) → third = T
- find_scale(attendance): (if attendance <= 1000) → scale = S
- find_scale(attendance): (if attendance > 1000 and attendance <= 5000) → scale = M
- find_scale(attendance): (if attendance > 5000) → scale = L
- load_balancer(third, event_on): (if event_on = False and third = F or third = S) → 2
- load_balancer(third, event_on): (if event_on = False and third = T) → 0
- load_balancer(third, event_on): (if event_on = True and (((third = F or third = S) and find_scale(get_attendance()) = L) or (third = F and scale = M))) → 4
- load_balancer(third, event_on): (if event_on = True and (((third = S or third = T) and find_scale(get_attendance()) = M) or ((third = F or third = S) and scale = S) or (third = T and scale = L))) → 2

Informal Specification 3: In a city that has different zones like residential, commercial and industrial the city council needs to take a decision about the new investments that need to be made towards each area in order to meet the rising demands of the electricity. The council already has some amount of bare minimum funds allocated to each area. However, the amount of funds allocated needs to be revised based on parameters like the number of upcoming projects in the area, the size of these projects and a predicted spike in usage of the electricity by the people in each area. Based on these parameters the council can decide to decrease the investment, increase the investment or keep it unchanged. The increase and decrease of investments can happen by a major chunk of 20% or minor chunk of 5-20%. Since the electricity usage of residential areas is low compared to that in commercial and industrial areas, the budget allocated towards it won't increase beyond 20%. The number of upcoming projects in an area could be grouped by ranges like less than 5 upcoming projects, 6-10 upcoming projects or more than 10 upcoming projects. Each individual project can be a large scale project, medium scale project or a small scale project. Since, every project size is different an average size of all the projects in a particular area would be considered for making the investment decision. Also, the machine learning algorithm deployed for predicting the spike in electricity usage outputs categorical values of high, medium, low if the spike is going to be more than 70%, within range of 40-70% and less than 40% respectively. If the electricity usage in an area is high and it has more than 10 projects planned in future then irrespective of size of the projects the council would want to increase the investments in that area by more than 20%. However, the cap on investment in residential areas will always remain less than 20%. Also, in the same case if the number of upcoming projects is between 5-10 in residential areas then depending on the average size of the project the investment would either be increased or stay the same. If the average size is small then the investment decision wouldn't change else it would increase. With the same number of upcoming projects in a commercial area the investment would increase by 5-20% and in the industrial area it would increase by more than 20%. In case the number of upcoming projects is within range of 0-5, then irrespective of the area and size of the projects the investment decisions wouldn't be affected. If the predicted spike is in the medium range with more than 10 upcoming projects, then irrespective of size of projects in residential areas the investment wouldn't change. In case it is a commercial area with more large sized projects then the investment would be increased by more than 20%. However if that's not the case, then investment can be increased by a minor chunk. In an industrial area, if the average size of projects is not small then investments would rise by a major chunk. However, if the average size is small then the investments increase within a range of 5-20%. With increase in spike being medium, and projects being in range of 5-10, if the area is residential or commercial then only if the average size of the project is large the investment would rise else they remain unchanged. For an industrial area with the same parameters the investment decision would remain unchanged. However, for any area if the number of upcoming projects then irrespective of their scale if the predicted spike is medium the investments would decrease by a minor chunk. Similarly, if the machine learning algorithm predicts the spike in an area to be low, no increase would be seen in any case for residential and commercial zones in any case. Instead the investments would decrease with the decrease in number of projects. i.e if the projects lie in the range of 5-10 the investments would decrease by a minor chunk and if the number of projects is lower than 5 than investments would be reduced by more than 20%. An exception to this decrease will be in the industrial area where even if the predicted spike is low and more than 10 projects are being planned in the area whose average size is larger than investments would be boosted up by a minor chunch else they would remain unchanged.

Semi-Formal Specification 3:

- Predicted spike: H:High, M:Medium, L: Low
- Number of upcoming projects in an area: A: More than 10, B:6-10 projects, C:5 or less projects
- Type of area: R: Residential, C: Commercial, I: Industrial
- Average size of projects: L: Large scale, M: Medium scale, S: Small scale
- Investment decisions: A: Increase by major chunk (20%), B: Increase by minor chunk (5-20%), C: No change in planned investment, D: Decrease by minor chunk (5-20%), E: Decrease by major chunk (20%)

Predicted Spike	Number of upcoming projects in an area	Type of area	Average size of projects	Investment decisions
H	A	R	Don't care	B
H	A	C	Don't care	A
H	A	I	Don't care	A
H	B	R	L	B
H	B	R	M	B
H	B	R	S	C
H	B	C	Don't care	B
H	B	I	Don't care	A
H	C	Don't care	Don't care	C
M	A	R	Don't care	C
M	A	C	L	A
M	A	C	M	B
M	A	C	S	B
M	A	I	L	A
M	A	I	M	A
M	A	I	S	B
M	B	R	L	B
M	B	R	M	C
M	B	R	S	C
M	B	C	L	B
M	B	C	M	C
M	B	C	S	C
M	B	I	Don't care	C
M	C	Don't care	Don't care	D
L	A	R	Don't care	C
L	A	C	Don't care	C
L	A	I	L	B
L	A	I	M	C
L	A	I	S	C
L	B	Don't care	Don't care	D
L	C	Don't care	Don't care	E

Formal Specification 3:

Operations:

1. get_proj(): none → integer
2. get_spike(): none → char
3. project_num(): integer → char
4. decide_investment(): char, char, char, char → char

Description:

1. The **get_spike()** function would get the predicted spike from the machine learning algorithm
2. The **get_proj()** function takes input from the user about the number of upcoming projects in the city and returns the input given.
3. The **project_num()** function takes the number of projects from the get_proj() function and returns the associated char value with it. For e.g., if the number of projects are less than 5 then the function returns C
4. The **decide_investment()** function takes four character arguments. The first argument is percentage hike which is given by the machine learning model working on past data via get_spike() function. The second argument is the number of projects that are planned in future which would be given via project_num() function. The third argument is area of the city which can be Residential(R), Industrial(I) or Commercial(C). The final argument is the scale of the upcoming projects, which can be classified into Large (L), Medium(M) or Small(S). Based on the different combinations of these argument values, the function outputs future investment decisions, also in terms of categories. The five categories of investment decisions are - E (decrease current investment by 20%), D (decrease current investment by 5-20%), C (no change in current investment), B (increase current investment by 5-20%) and A (increase current investment by 20%).

Axioms:

- get_spike() --> spike
- get_proj() → number_proj
- project_num(number_proj): if number_proj > 10 → num_proj = A
- project_num(number_proj): if number_proj > 5 and number_proj <= 10 → num_proj = B
- project_num(number_proj): if number_proj <= 5 → num_proj = C
- decide_investment(spike, num_proj, area, scale_proj): if (spike = L and num_proj = C) → invest_label = E
- decide_investment(spike, num_proj, area, scale_proj): if ((spike = L and num_proj = B) or (spike = M and num_proj = C)) → invest_label = D
- decide_investment(spike, num_proj, area, scale_proj): if (spike = H and (num_proj = C or (num_proj = B and area = R and scale_proj = S))) or (spike = M and ((num_proj = A and area = R) or (num_proj = B and area = I) or (num_proj = B and ((area = R or area = C) and (scale_proj = M or scale_proj = S))))) or (spike = L and (num_proj = A and (area = R or area = C or (area = I and (scale_proj = M or scale_proj = S))))) → invest_label = C
- decide_investment(spike, num_proj, area, scale_proj): if (spike = H and ((num_proj = A and area = R) or (num_proj = B and area= C) or (num_proj = B and area = R and (scale_proj = L or scale_proj = M)))) or (spike = M and ((num_proj = A and ((area = C and (scale_proj = M or scale_proj = S)) or (area = I and scale_proj = S))) or (num_proj = B and (area = R or area = C) and scale_proj = L))) or (spike = L and num_proj = A and area = I and scale_proj = L) → invest_label = B
- decide_investment(spike, num_proj, area, scale_proj): if (spike = H and ((num_proj = A and (area = C or area = I)) or (num_proj = B and area = I))) or (spike = M and num_proj = A and ((area = C and scale_proj = L) or (area = I and (scale_proj = L or scale_proj = M)))) → invest_label = A

Informal Specification 4: The system needs to alert the user about the next maintenance of their electronic devices. The smart device deployed at user homes keeps track of the usage of all smart appliances in the house using which it can alert the user about the upcoming maintenance of the appliance. The user can receive these alerts on an application present on his/her mobile phone. Also, the smart appliances keep track of the power supply they get from the power grid. In case, the supply is below 220V or above 240V the user gets an electric supply alert. The device categorizes the appliances into three categories based on the amount of usage that's left before the device reaches its maintenance. These categories are red (when the appliance's remaining usage is below 15%), yellow (when the appliance's remaining usage is between 15-40%) and green(when the appliance's remaining usage is above 40%). Also, the power consumed by the device is tracked by the smart device. In case the power consumed by the device is less or more than it's normal consumption and the appliance usage remaining is in yellow range then the user needs to be sent a maintenance alert. Also, irrespective of power consumed by the appliance if the remaining usage is in red band, the alert needs to be sent and if the appliance's remaining usage is in green band then no alert is sent. If the electricity supply and the electricity consumed the device is normal then no alert needs to be sent if the appliance's usage lies in yellow band.

Semi-Formal Specification 4:

<u>Notation:</u>

- Power supply to the device: B: Below 220V, N: Between 220-240V, A: Above 240V
- Power consumed by the device: B: Below normal, N: Normal, A: Above normal
- Percent usage left before next maintenance: R: Less than 15% Y: Between 15-40% G: Above 40%
- Send maintenance alert : Y: Yes, N: No
- Send electric supply alert: Y: Yes, N: No

Power supply to the device	Power consumed by the device	Percent usage left before next maintenance	Send maintenance alert	Send electric supply alert
B	Don't Care	Don't Care	N	Y
N	Don't Care	R	Y	N
N	B	Y	Y	N
N	Don't Care	G	N	N
N	N	Y	N	N
N	A	Y	Y	N
A	B	R	N	Y

Formal Specification 4:

<u>Operations:</u>

1. difference(): date, date → integer
2. usage_left(): date, date → char
3. send_alert(): string → alert
4. alert_module(): char, char, char → send_alert

<u>Description:</u>

1. The **difference()** function takes two dates as input where the first date is greater than the second date i.e first date occurs in future while the second one has occurred in past and computes the difference in days between the two dates.
2. The **usage_left()** function takes the date of last maintenance and date of next maintenance and classifies the usage left before next maintenance into 3 categories - Red(R) if the usage left is less than 15%, Yellow(Y) if the usage left is between 15-40% or Green(G) if the usage left is above 40%. The usage left percentage is calculated by dividing the number of days of utilization remaining of the device by the difference between the next maintenance date and last maintenance date.
3. The **send_alert()** function takes as input which type of alert to send the user - maintenance or electric supply, and returns an alert object containing an appropriate alert message to the user.

4. The **alert_module()** function decides which alert to send the user based on the power supply, power consumption and percentage usage left before next maintenance of each device. The alert message is passed to the send_alert() method which returns the alert object which is sent to the user on his mobile device via the system application

Axioms:

- difference(date1, date2) → (date1 - date2)
- send_alert(electric_supply)
- send_alert(maintenance)
- usage_left(last_maintenance, next_maintenance): if ((difference(next_maintenance, current_date)/difference(next_maintenance, last_maintenance))*100) < 15 → category = R
- usage_left(last_maintenance, next_maintenance): if ((difference(next_maintenance, current_date)/difference(next_maintenance, last_maintenance))*100) > 40 → category = G
- usage_left(last_maintenance, next_maintenance): if ((difference(next_maintenance, current_date)/difference(next_maintenance, last_maintenance))*100) is in range(15,40) → category = Y
- alert_module(supply, usage, category): if supply = B → send_alert(electric_supply)
- alert_module(supply, usage, category): if (supply = N and (category = R or (category = Y and (usage = A or usage = B)))) → send_alert(maintenance)
- alert_module(supply, usage, category): if (supply = A and usage = B and category = R) → send_alert(electric_supply)

Informal Specification 5: A smart city can consist of different types of areas and localities, which might lead to variation in power consumption patterns. The power consumption trends of office buildings are drastically different compared to that of residential buildings or industrial complexes. This requirement summarises the incorporation of a degree of automation(smartness) into the functioning and power of various devices based on a set of deciding factors. Environmental data sensors, motion sensors, time of day along with a few other factors determine the power usage of various devices present in rooms. The various categories of rooms considered are - Residential rooms(bedrooms, drawing rooms etc.), Washrooms, Surveillance rooms, Commercial rooms (offices, shopping malls, restaurants etc.) and Server rooms (IT-based). The devices we want to "smarten", i.e introduce some automation into, include air conditioners, electric light sources (bulbs, tubes etc) and electric heaters. Each class of device can operate in two modes - Auto and Manual. The system needs to decide the capacity at which each of the smart devices operate in order to conserve electricity. In case the device is in manual mode the device operates at full capacity as the user is allowed to manipulate the usage as per his needs. However, the user can forget to switch the device back to auto mode and hence a timer of 30mins is set after which the device resets itself to auto mode. An exception to this is the surveillance room which always operates in manual mode and hence a timer would not be set for it. In auto mode, the device would take decisions about the capacity at which it would operate based on the data it receives from the sensors in the room. If the motion sensor doesn't detect any activity in the room, the device is switched off. However, the AC in the server room must always be switched on irrespective of any sensor's reading. Time of day is provided to the system using which it divides the day into morning (7am-3pm), evening(3pm-11pm) and night (11pm-7am). Based on this input, if the motion sensor picks and activity in the room, the lights in residential area would remain switched off during the morning and night hours. Also, for residential room lights when motion sensors senses an activity, they would operate at full capacity during evening hours. For washrooms, the lights operate at 60% capacity during the morning and at full capacity during the rest of the hours if the motion sensor picks up motion in the room. Lights in server room would operate at 100% capacity if motion is detected else they are switched off. For air conditioner's the operation capacity is also determined by the value sensed by the temperature sensor. The value detected by the temperature is converted into classes A,B or C such that if the temperature is above 25C it lies in class C, if it's between 11-25C it lies in class B and if it's below 11C it lies in class A. For residential rooms and washrooms, when the motion is detected, if the temperature is in class A, the Air conditioner's are turned off. Similarly, if it's in class B,Ac's operate at 60% of it's full capacity and if it's in class C then the AC operates with full capacity. However, for commercial rooms the AC operates at 60% capacity when temperature lies in class A and 100% when it's in class B or C. Irrespective of the temperature and time of the day, if the motion sensor senses activity in a commercial room the lights operate with full capacity. Heater is also a smart device whose operation capacity is decided by the sensor data. If motion is detected by the sensor in residential area, then irrespective of the time of the day, if the temperature is in class A, heater operates with 100% capacity. With same conditions, if temperature is in class B or C, then heaters in residential rooms would be turned off. For heaters in commercial rooms, if the motion sensor picks up motion then irrespective of the time of the day, if the temperature is in class A, then heaters would operate with 100% capacity and would be turned off if the temperature is within class B or C. The user would be able to switch a particular device to manual mode via the app installed on his/her mobile device.

Semi-Formal Specification 5:

<u>Notations:</u>

- Type of room : R: Residential, W: Washroom, S: Surveillance, C: Commercial
- Type of device: A: Air conditioner, L: Lights, H: Heater
- Motion sensor detects action: Y: Yes, N: No
- Time of the day: M: Morning(7am-3pm) E: Evening(3pm-11pm) N: Night(11pm-7am)
- Mode: M: Manual, A: Auto
- Temperature class: A: Below 11C, B: Between 11-25C, C: Above 25C
- Devices operate at capacity: A: 100%, B: 60%, C: Switched off

Type of Room	Type of device	Motion sensor detects action	Time of the day	Mode	Temperature class	Device operates at capacity	Manual-to-Auto Switch Timer

R	A	Y	Don't Care	A	A	C	Don't set
R	A	Y	Don't Care	A	B	B	Don't set
R	A	Y	Don't Care	A	C	A	Don't set
R	L	Y	M	A	Don't Care	C	Don't set
R	L	Y	E	A	Don't Care	A	Don't set
R	L	Y	N	A	Don't Care	C	Don't set
R	H	Y	Don't Care	A	A	A	Don't set
R	H	Y	Don't Care	A	B	C	Don't set
R	H	Y	Don't Care	A	C	C	Don't set
R	Don't Care	N	Don't Care	A	Don't Care	C	Don't set
S	Don't Care	Don't Care	Don't Care	M	Don't Care	A	Don't set
S	Don't Care	Don't Care	Don't Care	A	Don't Care	Not valid condition	
W	A	Y	Don't Care	A	A	C	Don't set
W	A	Y	Don't Care	A	B	B	Don't set
W	A	Y	Don't Care	A	C	A	Don't set
W	Don't Care	N	Don't Care	A	Don't Care	C	Don't set
W	L	Y	M	A	Don't Care	B	Don't set
W	L	Y	E	A	Don't Care	A	Don't set
W	L	Y	N	A	Don't Care	A	Don't set
C	A	Y	Don't Care	A	A	B	Don't set
C	A	Y	Don't Care	A	B	A	Don't set
C	A	Y	Don't Care	A	C	A	Don't set
C	Don't Care	N	Don't Care	A	Don't Care	C	Don't set
C	L	Y	Don't Care	A	Don't Care	A	Don't set
C	H	Y	Don't Care	A	A	A	Don't set
C	H	Y	Don't Care	A	B	C	Don't set
C	H	Y	Don't Care	A	C	C	Don't set
V	A	Don't Care	Don't Care	A	Don't Care	A	Don't set
V	L	Y	Don't Care	A	Don't Care	A	Don't set
V	L	N	Don't Care	A	Don't Care	C	Don't set
R	Don't	Don't Care	Don't Care	M	Don't Care	A	Set

	Care						
W	Don't Care	Don't Care	Don't Care	M	Don't Care	A	Set
C	Don't Care	Don't Care	Don't Care	M	Don't Care	A	Set
V	Don't Care	Don't Care	Don't Care	M	Don't Care	A	Set

Formal Specification 5:

Operations:

1. get_slot(): time → char
2. smart_consumption(): char, char, bool, char, char, char → char, SetTimer/DontSetTimer
3. send_alert(): string → alert

Description:

1. The **get_slot()** function takes a timestamp as input and computes which slot of the day it falls in, morning, afternoon or night. If the timestamp falls in first third of the day i.e between 7am-3pm the function would return M, if the time falls between 3pm-11pm the function returns E and if it falls between 11pm-7am the function returns N
2. The **send_alert()** function is reused from the above specifications
3. The **smart_consumption()** function takes 6 arguments as follows: type of the room, device type, action_sensed by the motion sensor, slot of the day via getSlot() function, mode the device is operating in and the current temperature outside the building and returns the capacity at which the device should operate along with whether a timer should be set for switching to auto mode or not.

Axioms:

- ➢ get_slot(current_time): (if current_time > 0700 and current_time < 1500) → third = M
- ➢ get_slot(current_time): (if current_time > 1500 and current_time < 2300) → third = E
- ➢ get_slot(current_time): (if current_time < 0700 or current_time > 2300) → third = N

- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if room_type = **R** and mode = A and ((device_type = A and action_sensed = Yes and temp = C) or (device_type = L and action_sensed = Yes and third = E) or (device_type = H and action_sensed = Yes and temp = A)) → A, DontSetTimer
- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if room_type = **R** and device_type = A and action_sensed = Yes and mode = A and temp = B → B, DontSetTimer
- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if room_type = **R** and mode = A and ((device_type = A and action_sensed = Yes and temp = A) or (device_type = L and action_sensed = Yes and third = M) or (device_type = L and action_sensed = Yes and third = N) or (device_type = H and action_sensed = Yes and (temp = B or temp = C)) or action_sensed = No) → C, DontSetTimer
- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if room_type = S and mode = M → A, DontSetTimer
- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if room_type = **W** and mode = A and ((device_type = A and action_sensed = Yes and temp = C) or (device_type = L and action_sensed = Yes and (third = E or third = N))) → A, DontSetTimer
- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if room_type = **W** and and mode = A and ((device_type = A and action_sensed = Yes and temp = B) or (device_type = L and action_sensed = Yes and third = M)) → B, DontSetTimer
- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if room_type = **W** and mode = A and ((device_type = A and action_sensed = Yes and temp = A) or action_sensed = No) → C, DontSetTimer
- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if room_type = **C** and mode = A and ((device_type = A and action_sensed = Yes and (temp = B or temp = C)) or (device_type = L and action_sensed = Yes) or (device_type = H and action_sensed = Yes and temp = A)) → A, DontSetTimer

- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if room_type = **C** and device_type = A and action_sensed = Yes and mode = A and temp = A → B, DontSetTimer
- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if room_type = **C** and mode = A and ((device_type = H and action_sensed = Yes and (temp = B or temp = C)) or action_sensed = No) → C, DontSetTimer
- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if room_type = **V** and (device_type = A or (device_type = L and action_sensed = Yes)) → A, DontSetTimer
- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if room_type = **V** and device_type = L and action_sensed = No and mode = A → C, DontSetTimer
- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if (room_type = **R** or room_type = W or room_type = C or room_type = V) and mode = M → A, SetTimer
- ➢ smart_consumption(room_type, device_type, action_sensed, third, mode, temp): if room_type = **S** and mode = A → send_alert(error)