INSA Buildings Room Management

Project Report

Software Architecture

Louis Chauvet-Smart
Alexandre Cros
Andy Xu

Tutors:
Nawal Guermouche
Ghada Gharbi

Context

This project is about automating different functions for the rooms in INSA's buildings. This application needs to manage the opening and closing of windows and doors, and deal with heating and lights for example. The idea is to collect data from sensors, and act accordingly.

This management system was developed as part of ISS' Software Architecture course at INSA Toulouse, and incorporated some elements of software engineering, as well as some imposed technologies and methodologies.

The Github link for the project's source code is the following :
https://github.com/louis-cs/SOA_Project

Requirements

We want to create a proof of concept for a web application, managing INSA's rooms. The managing will cover:

-The closing of windows
-The closing of doors
-The activation or deactivation of heaters
-The turning on and off of lights
-The triggering of an alarm if someone is detected in the premises at an unauthorised time

These actions will have to take place automatically based on sensors present in the buildings.

Specifications

In order to define the specifications we decided to define the rules that our system will follow to manage the automation of INSA's rooms.

Presence of someone

- If someone is in a room before 10pm
  - ❖ Switch the light on if luminosity is low
  - ❖ Do not change the doors, windows and shutters state
- If someone is detected after 10pm
  - ❖ Turn on the alarm

Luminosity

- If luminosity < 500 lux: turn the lights on
- If luminosity >1500 lux: turn the lights off

Time

- Between 10pm and 7:00am: turn off radiators
- After 10pm and if windows are opened: close windows
- After 10pm and if doors are opened: close doors

Temperature

- If <19°C inside and outside and radiators are off: turn on radiators
- If >19°C inside and outside and windows are opened: close windows
- If <19°C inside and inside < outside and windows are closed: open windows
- If >24°C inside and radiators are on: turn off radiators
- If >24°C inside and outside and windows are opened: close windows
- If >24°C inside and inside > outside and windows are closed: open windows

Of course, we work under the assumption that each room element has a sensor to retrieve the state of all its components.

Architecture

Sensors and actuators simulation

To simulate all the sensors and actuators, we used EclipseOM2M. By requesting a local OM2M platform we created the following architecture:

```
insaRoomsAE
        room1CT
                room1SensorsCT
                        light
                        movement
                        temperatureInside
                        temperatureOutside
                room1ActuatorsCT
                        alarm
                        led1
                        window1
                        door1
                        radiator1
        room2CT
                ...
        room3CT
                ...
```

Since we defined every device as an OM2M container, the "contentInstance" is put directly inside instead of adding a "Descriptor" and a "Data" container for simplicity. Also there is only one unique sensor of its kind for every room while actuators can be multiple per room but the alarm.

Services (see README for more detailed usage)

We decided to implement three services to communicate with the OM2M REST API: one for every device that deals with lighting, one for movements and one for temperatures. All services are very similar to each other as they use the same REST API to get through the http GET method the sensors and actuators values and states. As the devices are simulated, we also implemented the POST method to sensors so that we can test our management system.

The management system is implemented on another web service and it is the one the user interacts with. It communicates with the already created services to interact with the simulated OM2M devices. The user can see a few important information about the devices and the log of actions and can also choose to activate or deactivate the system. The time is also configured inside the management system and can be changed by the user.
A further step of the management system would be so that it is automatically scalable when modifying the environment like the number of rooms or devices, using a searching tool and the parsing of its result (here we implemented the discovery function of OM2M).

This architecture is for us the most logical and easiest to develop using agile methods.

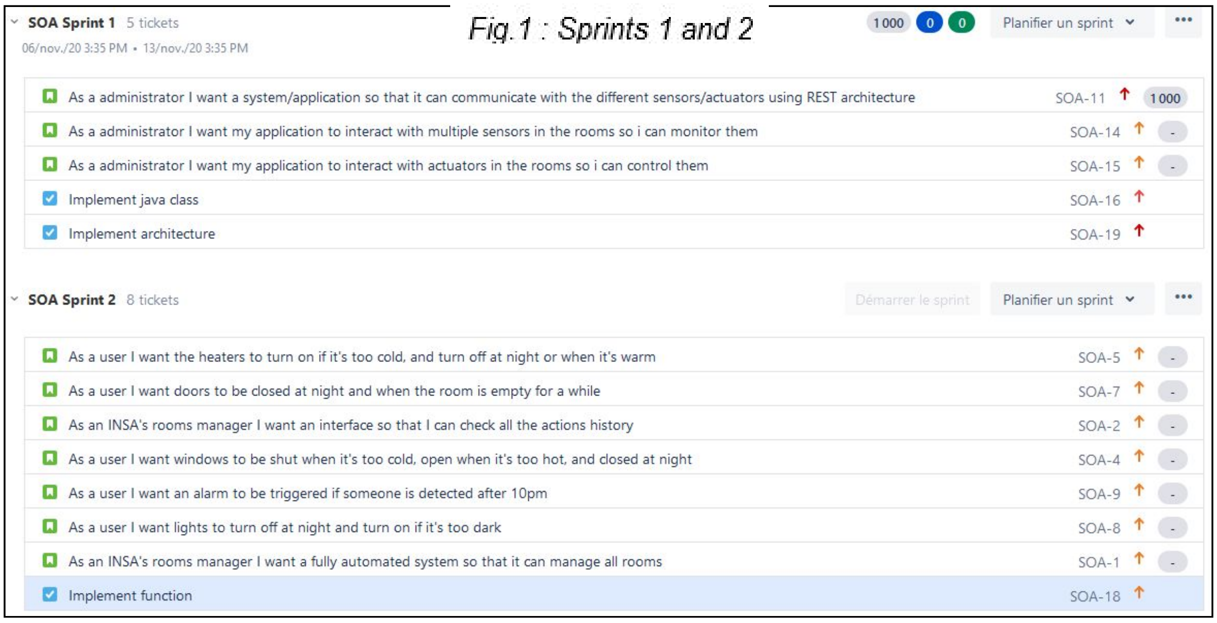
Project Management

For this project, we used the Agile method to manage the team and keep track of the work done and work to do. To help us, we used the Jira platform. The idea behind this method is to list all of the work that needs to be done in the project, and cut it in smaller, manageable tasks. Taking into account the time we have to complete the project, we then estimate the time needed to accomplish each task, and sort them into “Sprints”, which correspond to a list of tasks to accomplish in a specific time window (usually a week or two). The strength of this method is its flexibility, since time is taken at the end of each sprint to see how much progress was made compared to predictions, and to restructure the following sprint taking into account the sprint review.
Because of Covid, we couldn’t exactly keep up the pace we planned before confinement hit, since not all members of the group were in ideal conditions to advance on the project. The sprints were still a very effective way to structure the work to be done, even if the time frame was a little more flexible than it should have been.

Sprints

We initially planned on creating three sprints, but after a tight first sprint, we decided to allocate the rest of the tasks to a longer, second sprint to be sure all members of the group could organise themselves to finish the project in time.

As we can see, the second sprint seems much larger than the first one, but this isn’t actually the case, as most tasks in the second sprint were much smaller than the tasks in the first one. They also had the advantage of mostly being independent from each other, so they could be worked on separately, unlike the conception and architecture tasks from the first sprint.



SOA Sprint 1 5 tickets
06/nov./20 3:35 PM • 13/nov./20 3:35 PM
1 000 0 0 Planifier un sprint

As a administrator I want a system/application so that it can communicate with the different sensors/actuators using REST architecture	SOA-11	1 000
As a administrator I want my application to interact with multiple sensors in the rooms so i can monitor them	SOA-14	-
As a administrator I want my application to interact with actuators in the rooms so i can control them	SOA-15	-
Implement java class	SOA-16	
Implement architecture	SOA-19	

SOA Sprint 2 8 tickets
Démarrer le sprint Planifier un sprint

As a user I want the heaters to turn on if it's too cold, and turn off at night or when it's warm	SOA-5	-
As a user I want doors to be closed at night and when the room is empty for a while	SOA-7	-
As an INSA's rooms manager I want an interface so that I can check all the actions history	SOA-2	-
As a user I want windows to be shut when it's too cold, open when it's too hot, and closed at night	SOA-4	-
As a user I want an alarm to be triggered if someone is detected after 10pm	SOA-9	-
As a user I want lights to turn off at night and turn on if it's too dark	SOA-8	-
As an INSA's rooms manager I want a fully automated system so that it can manage all rooms	SOA-1	-
Implement function	SOA-18	

Fig 1. Sprints 1 and 2

One of the biggest advantages of the agile method for this project was the flexibility, as a lot of decisions were made during the first sprint. For example, we decided to start developing the different services without taking sensors into account, and to implement sensor-simulators later. This was one of multiple ways to structure the development, but thanks to the flexibility of the agile method and sprint reviews, we were able to re-organise tasks to take this new decision into account.

Conclusion

This project was a great illustration of the skills we learned about in the Software Architecture course. Not only were we able to design our own Service-Orientated Architecture, we were actually able to see the advantages of using it. Using the OM2M platform in this application was also a great way to see how it could be used in a more concrete setting, expanding the horizon of applications we would have thought to utilize this technology in.
Even though we had to bend the rules of the agile method to these circumstances, it gave us the guidelines and the flexibility to achieve what we set out to do in the beginning. While we would have liked to achieve more, like a polished user dashboard, we are overall glad with how we were able to keep working under the pandemic's unusual circumstances. We were able to use some of the Agile method's main axioms, those being tasks subdivision, time evaluation and progress review, to finish the project in time. The flexibility it granted us was paramount to staying on the right course and reaching our goal.