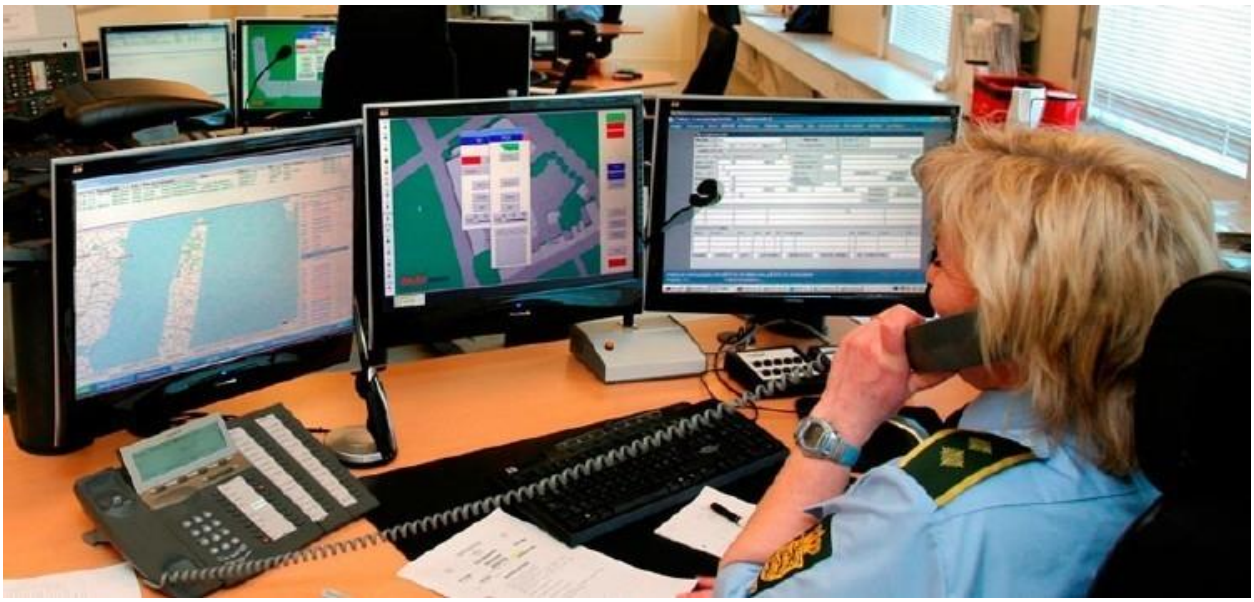
Media Processing: lab task 2



Goal

The goal of this lab task, which will take 3 sessions, is to convert the following problem description to efficient, extensible, maintainable… modern C++ code. This will be done in different steps, first you need to generate a UML class diagram of your proposed solution. You will get feedback on this proposal and then you can start implementing this in C++. Be very careful concerning the global structure (use of interfaces, design patterns), division of responsibilities among the different classes (low coupling – high cohesion), use of const-correctness (be lazy, let the compiler work) and the use of appropriate call and return mechanisms (avoid unnecessary copies). Use also smart pointers (std::shared_ptr or std::unique_ptr) in your application. Finally, each member of the team should implement one of the 3 proposed scenarios using the classes generated by the team as a shareable library. Check these scenarios to understand which subclasses you need with their specific functionality. Finally each member submits his final project to Git. Don't forget to submit your first version of your UML on Toledo this friday 11:59pm.

Story

You need to develop software for an emergency center which a security company wants to sell to as many customers as possible. So flexibility in set-up is very important. Different types of sensors can be installed at the customer's location, which are all connected to the emergency center. Some available types are smoke sensors (fire alarm), motion sensors (burglary alarm) and toxic gas sensors (intoxication alarm). Up to you to declare the necessary attributes for each type of sensor to be able to fulfill all requirements from your scenario. Of course if the security company wants to release new types of sensors, their programmers should be able to update the software with minimal interference with other classes. From the center it should be possible to activate/deactivate the sensors and to test them. When a test signal is sent to a sensor it should react like it would have been triggered. To simplify the management, it should be possible to address 1 single sensor or a whole group of sensors. ***In your test scenarios you will get a specific organization of the emergency center, h*****owever you need an**

implementation which can handle all kind of groupings.

Every single sensor needs to be fully configurable in its actions (both number and type). Again there are a number of possible emergency services it should activate: the fire brigade, the police, evacuation of the building… and this also need to be easily extensible. As an example: motion sensor A in the parking lot should turn on the lights, motion sensor B at the entrance needs to warn the police, while the CO sensor C in the lab should start the evacuation of the building and call the fire brigade. In other words: the action(s) triggered by a sensor do not depend on the type of sensor, but each individual sensor should be configurable.

Every sensor should be uniquely addressable, store the name of the vendor and know where it is located. Be sure that the sensor class itself knows how to generate these unique id's. It's not the task of the user of these classes. By default every new sensor is deactivated.

Some detailed requirements

- By the end of the lab session in week 3 you need to hand in the UML class diagram as an image (e.g. PNG).Use Visual Paradigm, umlet, app.diagrams.net or comparable. Be as detailed as possible: not only class relations, but also multiplicity of relations, complete type of attributes, complete signature of methods…
- From week 4 on, start developing the necessary classes. Be sure your classes are completely defined from your class diagram, so you can split the work. Use your git repository to share your code.
- All actions are simulated by simple messages to standard out, but they should be seen as separate methods so that in realistic situations all these actions can be implemented.
- Implement the << operator for a sensor to get all its information.
- Implement the unary operator ++ and -- to activate and deactivate a sensor? Should you use the prefix or postfix version?
- Add code to give an overview of your sensors in some ordered state (e.g. ordered by ID, ordered by vendor (and for the same vendor by ID),…).
- Use Valgrind Memory Analyzer to check for memory leaks (although the chance is small since you are using smart pointers). Valgrind is a linux program, install Virtual Box with Ubuntu if you want to test this.

Tips

- To reach the requirements, you need to use some design patterns. Take at least a look to Factory, Strategy, Composite, Observer before you decide which one to use.

Scenario 1

Implement a situation as in the Group T building: a building with several modules each having different rooms. Create the following rooms:

- lab Chemistry (5.01) where you put a smoke sensor from Sensor Solution and a gas sensor for No_x from GasSense. The actions for the smoke sensor is activating the building alarm and send a message to KULeuven central dispatch. The gas sensor should activate the room alarm and send an email to lore.hennebel@kuleuven.be.

- Lab Electronics (10.01) where you put a smoke sensor from Sensor Solution and a motion sensor from GotYou. The actions for the smoke sensor are the same as the one in 5.01. The motion sensor should only be active between 10pm and 7am, if activated it should send an sms to the phone of Gert Vanloock.

- Lab Electronics (10.02) where you put a motion sensor from GotYou with same actions as in 10.01

- Alma kitchen (1.03) where you put a smoke sensor from KitchenSafe, a gas sensor for CO from GasSense and a motion detector from BigBrother Is Watching You. The smoke sensor should activate the automatic extinction system and send a message to Alarm dispatch, the CO sensor should activate a local alarm and the motion sensors (activated if Alma is closed) should warn the police and KULeuven security.

- First activate and test the smoke sensor in lab Chemistry.

- Activate and test all sensors in lab chemistry

- Test all sensors in the Group T building (same result as previous since all other sensors are still deactivated)

- Activate all sensors using the ++ operator

- Test the whole building

- Test module 10

- Give an overview of all sensors ordered by vendor

Scenario 2

A security firm controls all houses in a residential area. Implement this situation. A residential area contains multiple houses, each having different rooms (or extensions like barns/sheds). Create the following rooms:

- John is a carpenter and has some machinery in his shed. A smoke sensor of the brand Smokey is definitely needed here. Also add a motion sensor of the brand IcanCU. The actions for the smoke sensor is activating the house alarm and send a message to Fire department of the

neighborhood. The motion sensor should activate the light and send an turn on the air conditioner.

- Vicky is John's wife and she likes to cook, every kitchen of course needs a smoke sensor from the brand Smokey and a motion sensor from IcanCU. The actions for the smoke sensor are the same as the one in the shed. The motion sensor should only be active between 10pm and 7am, if activated it should send an email to vicky at vicky.cooke@gmail.com.

- Dave has an old fireplace in his living room. These need a carbon monoxide sensor from BreatheLabs and a smoke sense from BurningInc. The smoke sensor activates the sprinkler system and sends a message to the fire department of the neighborhood. The CO sensor sets off the alarm of the house.

- Then there is also Kaitlin… She is the evil scientist who likes to do experiments at home. A gas sensor from TN2S is installed in her house. If this sensor goes off the whole neighborhood has to be evacuated due to danger of explosion. All alarms in all houses should be activated. Kaitlin also has a motion sensor of the brand IcanCU in her lab that sends an email if motion is detected between 4:40 and 9:15, since this is the time that the mad scientist sleeps.

- First activate and test the smoke sensor in shed.

- Activate and test all sensors in Vicky and John's house.

- Test all sensors in the neighborhood (same result as previous since all other sensors are still deactivated)

- Activate all sensors using the ++ operator

- Test the whole neighborhood

- Test the mad scientist's house

- Give an overview of all sensors ordered by id

Scenario 3

The Zorglax are an incredibly advanced alien race that has mastered interstellar travel, but their C++ programming skills are lacking. Thus, they would like to install your software on their mothership to monitor their galaxy, Twix. Feedlebrop The Procurer has sent you the following list of requirements:

- We need a motion sensor to monitor the entire Twix galaxy, to warn us of any intrusions. This sensor is permanently active. If triggered it should warn our peacekeeping force, which you call police.

- Planet X AE A-12 has a lot of volcanic activity, so we need a gas sensor to measure the amount of sulphur in the atmosphere. If this exceeds 1 billion tons, an alarm should sound and an electronic message should be sent to our scientists on the surface.

- On the surface of X AE A-12 there are two supervolcanoes that will kill all life on the planet if they erupt, Redstone and Orangestone. Each volcano needs a special smoke sensor, which will activate an array of magma-cooling sprinklers if triggered.

- We are currently using the planet LV-426 for an experiment in creating organic life. Our lifeforms require oxygen to breathe, so if the oxygen level in the atmosphere drops below 1.2 billion tons an alarm should be activated. We also don't want any interference with our samples, a permanently active motion sensor should be added to check if the atmosphere is breached. When triggered, this motion sensor should alert our peacekeeping force and our scientists.

- On the planet are two sites with egg samples, Moon Unit and Dweezil. Each site is tracked by a smoke sensor, which should alert our fire fighters when triggered. In addition, we will install a motion sensor at each site to alert us if the eggs are hatching. We only expect this to happen at night, so the motion sensors should only be active from 20:00 to 8:00. To avoid them being triggered by other movements, we want their activation range to be restricted to 10 meters.

- Due to the advanced nature of our society, all our companies have merged into one evil super company through a series of hostile takeovers. This company will manufacture all sensors. It is called "Disney" (any resemblance to earth companies is purely coincidental).

- To test our setup we require the following:

    - Activate and test all atmospheric sensors, but leave all other sensors inactive ○ Activate all sensors on X AE A-12, and test all of them (including the atmospheric sensor again)
    - Deactivate all sensors on LV-426 and test. This should not do anything.
    - Reactivate LV-426. First test the whole planet, then Moon Unit, then Dweezil. ○ Print an overview of all sensors, alphabetically ordered by location

  Failure to comply will result in the destruction of your planet and the eradication of your entire species. Kind Regards, Feedlebrop The Procurer.