Software Engineering Labs
3IF, INSA Lyon

The objective of this lab series is to apply software engineering principles to develop a high quality software application.

Topic

A government agency for environmental protection needs to monitor the air quality of a large territory. The agency has installed a number of sensors for this purpose, which are spread evenly over the territory. The sensors generate data at regular intervals. The data are gathered by the agency and stored on a central server in a set of files in CSV format.

The sensors are listed in the file `sensors.csv`. A sensor is described in the following format:

```
SensorID;Latitude;Longitude;
```
(Example: `Sensor0;44;-1;`)

The measurements from all the sensors are given in the file `measurements.csv`. Each sensor measurement is represented by a single line in the following format:

```
Timestamp;SensorID;AttributeID;Value;
```
(Example: `2019-01-01 12:00:00;Sensor0;O3;50.25;`)

A sensor generates measurements of four different types, which are described in the file `attributes.csv` as follows:

```
AttributeID,Unit;Description;
```
(Example: `O3;µg/m3;concentration d'ozone;`)

You are required to develop a software application called AirWatcher for the government agency.

The AirWatcher application will allow analysis of the data generated by a sensor to make sure that it is functioning correctly. This will help the government agency to identify and maintain malfunctioning sensors.

Moreover, the application will aggregate the collected information to produce statistics that may help the agency in making decisions about improving the quality of air. For example, the application will allow the calculation of the mean of the quality of air in a circular area specified by the user. The mean of the quality of air may be calculated for a given moment, as well as for a specified period of time.

The application will also enable selecting one sensor and then scoring and ranking all other sensors in terms of similarity to the selected sensor. The similarity will be based on the data generated by the sensors during a specified period of time. The purpose of this functionality is to identify areas with similar air quality.

Another functionality required for the AirWatcher application is to produce the value of air quality at a precise geographical position in the territory at a given moment. Please note that there may or may not be a sensor present at the specified position.

There exist several companies that manufacture and provide “air cleaners” to the government agency. These are industrial scale machines that can purify the air in the surrounding area. The providers install such air cleaners in the territory. The agency and the providers use AirWatcher to observe the impact of the cleaners on air quality, for example, the radius of the cleaned zone, the level of improvement in air quality, etc.

The providers and their air cleaners are listed in the file `providers.csv`. The air cleaners and the start and stop times of their operation are listed in the file `cleaners.csv`. The respective formats of the two files are given below.

`ProviderID;CleanerID;`
(Example: `Provider0;Cleaner0;`)

`CleanerID;Latitude;Longitude;Timestamp(start);Timestamp(stop);`
(Example: `Cleaner0;45.3;1.3;2019-02-01 12:00:00;2019-03-01 00:00:00;`)

Private individuals can also participate in air quality data generation by installing fixed sensors at their homes. The incentive given to the private individuals for contributing data is that they are awarded points for the use of their data. A private individual is awarded one point each time data from their sensor are used in a query on AirWatcher.

The private individual users and their sensors are listed in the file `users.csv` in the format given below. The measurements from the user owned sensors are included in the file `sensors.csv` along with the measurements from all other sensors.

`UserID;SensorID;`
(Example: `User0;Sensor70;`)

There is the possibility that a private individual may act maliciously and corrupt their sensor in order to provide false data. Therefore, the AirWatcher application will allow the government agency to analyze the data provided by a private individual’s sensor and classify its behavior as reliable or unreliable. If a private individual is detected to provide unreliable data, their entire data will be marked as false and will be excluded from all further queries on the application. This will prevent the user from gaining any further points.

The government agency would like the algorithms (analyzing a sensor, calculating mean, comparison of sensors, etc) of AirWatcher to be efficient. Therefore, the performance of the algorithms must be measurable. A simple metric of performance is the duration of execution of an algorithm measured in milliseconds.

Please note that the application does not interact directly with the sensors or the air cleaners. The collection of data is out of scope for this application. The application only analyzes the data contained in the files on the central server. We can assume that the data will not be updated during the execution of the application. Moreover, the application will not modify the data files.

The application will provide a console based user interface to its different users, which will be tailored according to the requirements and privileges of their role. Remote network access to the application is out of scope for the implementation of this application. The console based user interface will be available only on the server itself. The application will access the CSV files locally on the server.

The users of the application will not have direct access to the data set. However, they can access the data through the functionalities provided by the application. There are no particular restrictions on access to the data for any users. The data set is available to you on moodle.

Air quality can be measured by an Air Quality Index (AQI). You can use the ATMO index for this application, which is described here:
https://fr.wikipedia.org/wiki/Indice_de_qualit%C3%A9_de_l%27air

Organization

The project team will comprise of 2 binômes (4 students). You can choose your fellow binôme as you wish, within your group.

Final Deliverables

The deliverables can be written in English or French. Please submit all deliverables in one ZIP file on moodle. The deliverables will be due by midnight (23h59) on the day of your final presentation (soutenance). The documents should be in PDF format.
All documents should be written in sufficient detail. For example, in the requirements specification document, the functional and non-functional requirements of the system should be described comprehensively.

Initialization document
- Introduction
- Planning (team, roles, organization, Gantt chart, etc.)

Requirements specification document *6 points*
- Use case diagram
- Functional and non-functional requirements of the system
- Analysis of security risks (please consult the lectures on “Security by design”)
- Validation tests
- User manual (for the console based user interface)

Design document *6 points*
- Architecture, modular decomposition
- Class diagram
- Sequence diagrams of three major scenarios
- Description and pseudo-code of three major algorithms
- Unit tests

Software application development *6 points*

- Source code + guide to build the executable
- Tests (source code + data + any non-regression procedures)
- Note: Development of only two major functionalities is required.
- Note: Please do not include binaries with the source code.

Presentation and demo *2 points*

- Presentation of the specification and design (please submit the presentation slides)
- Live demo of the implemented functionalities
- Performance evaluation of the executed algorithms
- Source code review

Total: 20 points

Intermediate Deliverables

An intermediate deliverable will be due on moodle by midnight (23h59) three days after each of the first two labs (TPs). For example, for a lab on April 3, the deliverable will be due by the midnight (23h59) of April 6. However, you can continue to revise the deliverables until the submission of the final deliverables. Only the final deliverables will be graded. Intermediate deliverables will allow instructors to follow and discuss your progress.

Lab 1: Initialization document + Requirements specification document (one PDF file)
Lab 2: Design document (one PDF file)
Lab 3: Intermediate source code (one ZIP file)
Lab 4: No deliverable
Lab 5: All final deliverables (one ZIP file)

Final Presentation and Demo

The final presentation and demo of the project will take place during the last lab (TP). Each team will have maximum 20 minutes. You will present the complete specification and the design. This will be followed by a live demonstration of the implemented functionalities as well as a performance evaluation of the executed algorithms. You will also present the source code of the core algorithms.