G6046 Software Engineering

Seminar session: Using UML models

Summary

This exercise is designed to ensure that you understand:

- The basic types of useful Unified Modelling Language (UML) models, and
- how to apply these to solve a problem.

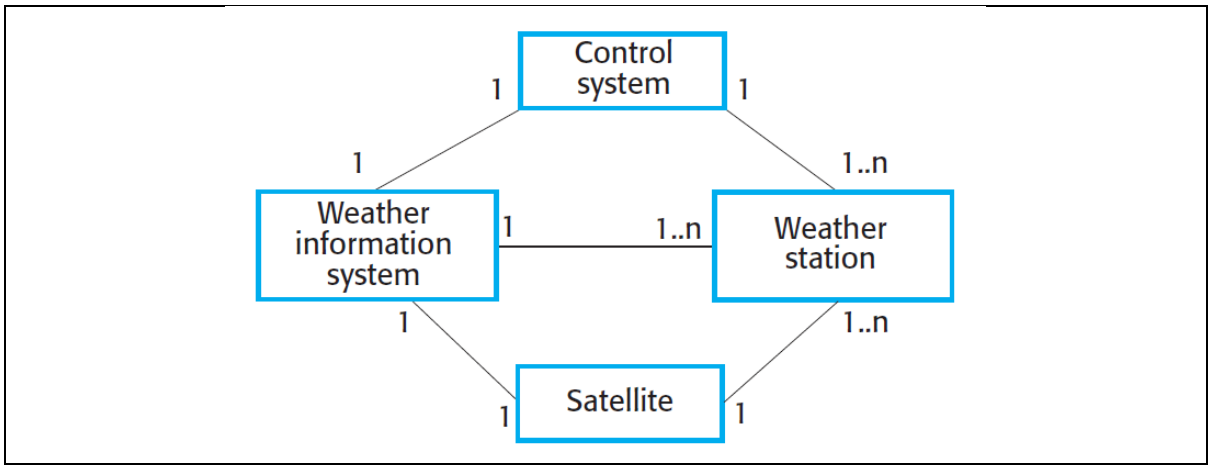
Key concepts

Make sure you understand these basic UML concepts:

- **Use case**: a simple high level model that identifies the key actors (usually people, but can be other systems). Provides a high level of key functional requirements i.e. what does a user expect to be able to achieve using the system?
- **High level context model (domain model**): a high level view of how a system fits into a wider ecosystem. Generally used to identify other systems and the need for interfaces with those systems.
- **High level class diagram:** A simple diagram that identifies key entities and the basic associations (and possibly the cardinality of those relationships, but with no analysis of fields or methods.
- **Class diagram:** as above but with details of fields and attributes.
- **Sequence diagram:** A timeline diagram that shows the sequence or chain or method calls between classes/objects that enables the user to achieve the functionality set out in the use case diagram.
- **State diagram model:** A diagram intended to help a developer understand how a process within the system is intended to operate.

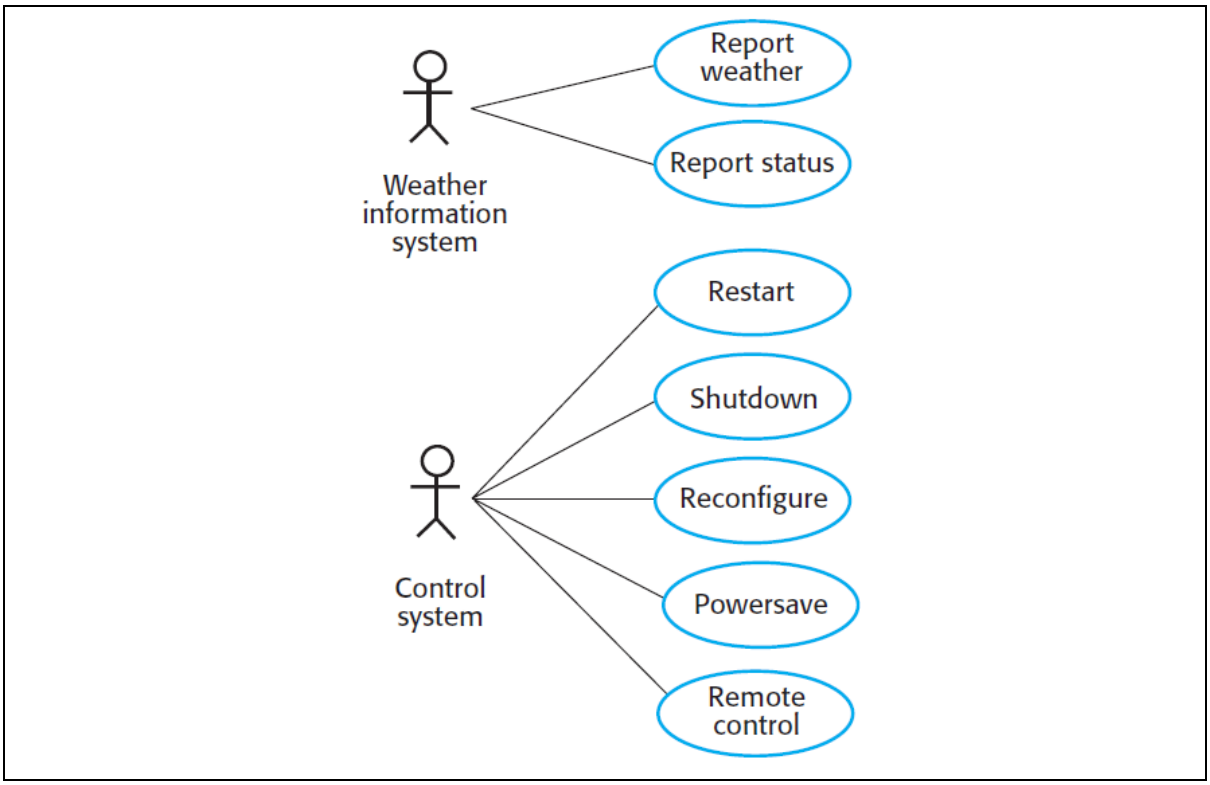
The task

This case study concerns a weather station. This is the context model for the weather station:



We are only concerned with the design. of the weather station. In turn, the weather station forms part of a bigger weather monitoring and information system.

We are then given a use case analysis for the weather station. We can represent that easily using the standard UML use case diagram, but we could use a tabular use case format as an alternative.



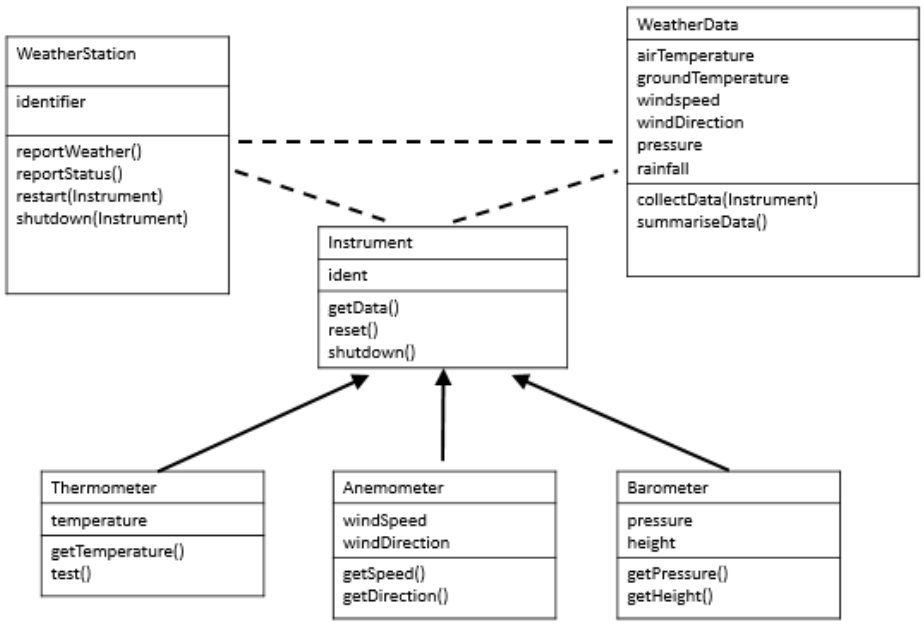
System	Weather station
Use case	Report weather
Actors	Weather information system, Weather station
Description	The weather station sends a summary of the weather data that has been collected from the instruments in the collection period to the weather information system. The data sent are the maximum, minimum, and average ground and air temperatures; the maximum, minimum, and average air pressures; the maximum, minimum, and average wind speeds; the total rainfall; and the wind direction as sampled at five-minute intervals.
Stimulus	The weather information system establishes a satellite communication link with the weather station and requests transmission of the data.
Response	The summarized data is sent to the weather information system.
Comments	Weather stations are usually asked to report once per hour but this frequency may differ from one station to another and may be modified in the future.

Now we are given some information about an OO class analysis for the weather station entity. First, a user requirement is given:

"A weather station is a package of software controlled instruments which collects data, performs some data processing and transmits this data for further processing. The instruments include air and ground thermometers, an anemometer, a wind vane, a barometer and a rain gauge. Data is collected periodically.

When a command is issued to transmit the weather data, the weather station processes and summarises the collected data. The summarised data is transmitted to the mapping computer when a request is received."

From that we produce a proposed basic class diagram:

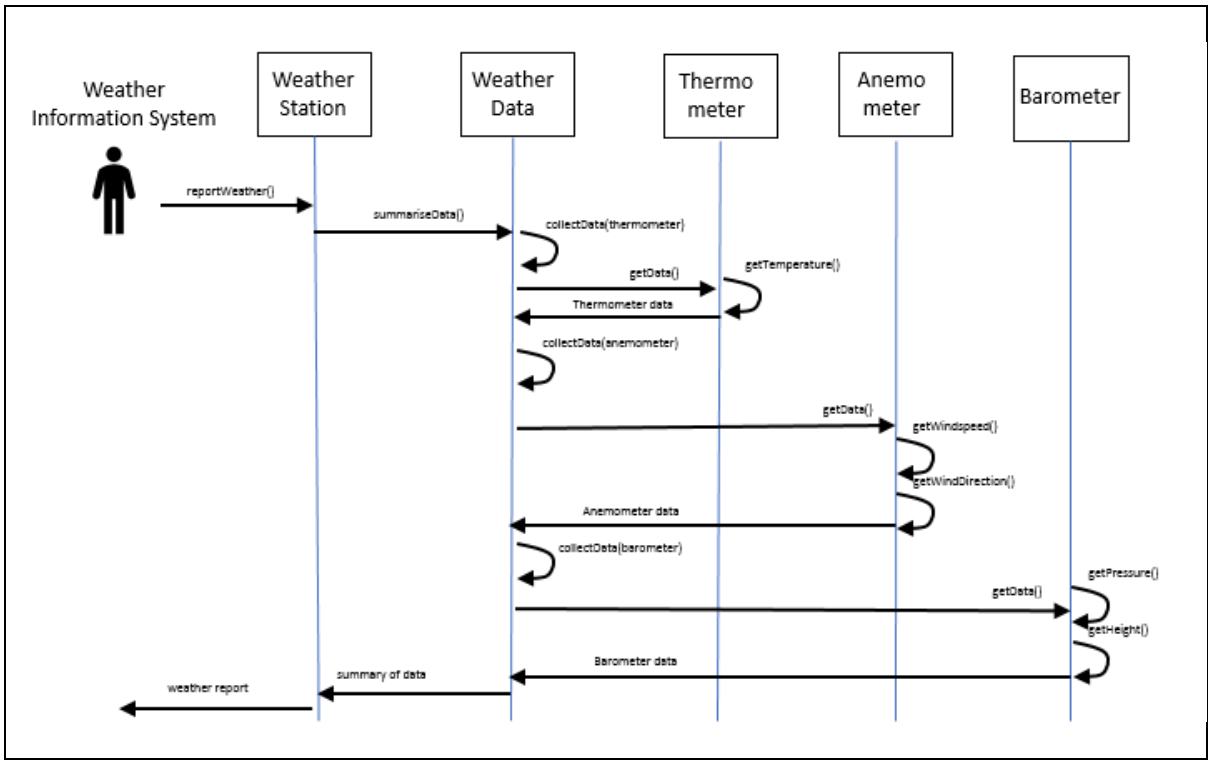


OK, but is the class diagram "fit for purpose"? Does it "work"?

Task

Consider the use case where the Weather Information System needs to be able to "Report Weather". So the Weather Information System is a user of the Weather Station entity. Try to construct a sequence diagram showing the chain of events for this use case.

Here is a possible solution:



Points to note:

- `reportWeather()` is a method of `WeatherStation`.
- `sumamriseData()` `is` a method of `WeatherData`.
- `collectData()` is a method of `WeatherData`. We can show a class calling a method within itself using the curved arrow (a "within class" method call)
- `collectData()` in turn calls the `getData()` method that the three sub classes `Thermometer`, `Anemometer` and `Barometer` have inherited from their parent `Instrument` superclass.
- For each of the Instruments, the appropriate methods within the class are used to produce the data.

So the design does look like it makes sense. It lacks some details, but the structure is sound.

Dr Kingsley Sage
Khs20@sussex.ac.uk