Case Studies in Software Design

# Smart cities and sensor systems

‘Smart cities’ is a term used to describe the use of technologies and data in order to solve problems and challenges (often related to sustainability) within cities. Currently, many cities are in the process of upgrading and installing infrastructure in order to make themselves smart, using data and technology to improve transport, energy use, health and air quality or to drive economic growth.

In this case study, your team are bidding to AML, an engineering systems company, for the contract to develop their core software for a smart cities infrastructure that could be sold to cities around the world.

* The first stage of bidding is to develop a design for your software that offers both the functionality to deploy and manage a range of sensors across a city, as well as providing high levels of usability for city administrators, residents and visitors.   
  The deadline for this design is Tuesday 27th March 2018.
* In the second round of the contract competition a team of software developers will use your designs to develop, test and evaluate a simulator for your software platform that demonstrates both the usability features and the flexibility of the system to be configured to address different ranges of sensors and environments in different cities around the world.

In the first phase, you are required to demonstrate your understanding of the domain by creating a software functional design (using UML) and user interaction designs for the software.

# About the case study:

As urban populations increase, issues arise around the management of city resources and the sustainability of cities themselves. In order to deal with these issues, cities have moved to the development of innovative smart systems. These systems gather data about the city and allow this data to be used to effectively manage the city infrastructure. This can include everything from roads, to drainage, buildings and energy distribution grids. The data can be used to provide real-time travel information, manage energy supply and control lighting. In many smart cities, this data can also be made available to the public, allowing them to use this data to make decisions about how they use city infrastructure.

Two central idea for the product line that AML are developing are sensor networks and sensing stations.

A sensor network is made up of a number of related sensors placed at various locations around the city. So, for example, a traffic management sensor network would consist of a number of traffic level sensors mounted at traffic lights, junctions and roundabouts around the city. Sensors might count individual vehicles passing a particular point and data might be aggregated locally to identify rates of traffic flow (vehicles per hour) across a location. An environmental sensor network might gather data such as temperature, humidity and air quality (e.g. levels of key pollutants such as Carbon Monoxide, Nitrogen oxides, soot particulates) at different locations. Drainage and flood management sensor networks might capture river levels and rates of flow. As each sensor network will be dealing with different sensors and thus different sensor data, each network will require different data sampling and handling methods. For example, the traffic management network may need to sample aggregate traffic flow data every few minutes, whereas some environmental and drainage management sensors may only need to update once per hour.

A sensing station groups together a set of related sensors and provides communication services to collect data and report it to the network. For example, a traffic sensing station might be set up at a road junction to collect the data from the individual sensors on each road that meets at the junction.

Each sensor within a sensor network should have a location associated with it, so that data points can be mapped to specific locations within the city. It should also be possible to update this location should a sensor need to be moved. Networks should be extensible, allowing new sensors to be installed and then added to a specific sensing station or network. The system should also allow for the addition of new sensor networks and new types of sensors, as new technologies are constantly being developed and smart city infrastructures are evolving.

The software system should be designed to allow smart city managers to create, modify, delete and monitor sensor networks. It should also include the ability to view and analyse the data from each sensor network in a variety of ways, such as live readouts, as well as averaging data over time or looking for peaks or patterns in data. This will allow city managers to maximise the usefulness of the gathered data.

A means of allowing city residents to access data from the system in a read-only manner might also be useful in allowing residents and local companies to develop their own uses of the smart city data.

## Existing Systems/Further Information

A number of cities worldwide have a variety of smart city initiatives currently ongoing. You may find these useful in providing ideas on the kinds of sensors, data and applications that could be involved in a smart city system. Some examples include:

* Barcelona: <http://www.vilaweb.cat/noticia/4175829/20140226/ten-reasons-why-barcelona-is-smart-city.html>
* Singapore: <https://www.wsj.com/articles/singapore-is-taking-the-smart-city-to-a-whole-new-level-1461550026>
* Bristol: <http://www.bristolisopen.com/overview/>
* New York: <http://www1.nyc.gov/assets/forward/documents/NYC-Smart-Equitable-City-Final.pdf>

While some aspects are common to many systems (e.g. environmental monitoring), some are more unusual, such as Barcelona’s smart rubbish bins. Your system should take into account that each city may have different problems, and thus different solutions and allow for as broad a range of sensors and data as possible.

For a more general overview on Smart Cities, the Open University offers a free online course: <http://www.open.edu/openlearn/science-maths-technology/smart-cities/content-section-overview>

## 5 Final Thoughts

The information in this document is typical of what you might expect to get from a client in industry when developing software systems, i.e. incomplete, possibly ambiguous, and certainly vague.

You are also encouraged to talk to the module delivery team during tutorials to clarify any queries you might have as you design (and then implement) a working system.

Remember, your clients will rarely be computer savvy and it is your job as practicing software engineers/computer scientists, to interpret the English language musings into formal designs and ultimately a software package. If the client could do this, they wouldn’t need us.

## Your design documentation should include:

|  |  |
| --- | --- |
| Element | Marks available |
| 1. A class diagram that covers the major classes used in the system, their relationships and the type signatures of their most important methods | 10 |
| 1. Two sequence diagrams showing how objects will collaborate to implement the more complex use cases in the system. You will be required to discuss the diagrams and justify your design decisions. | 10 |
| 1. One sequence diagram showing how readings are collected from at least two different sensors on the same field station, taking readings with different frequencies | 16 |
| 1. Text and annotation to support your diagrams explaining how the major responsibilities of the system have been allocated (e.g. CRC cards, annotations in diagrams, text) | 10 |
| 1. Consistency between division of responsibility, class diagram and sequence diagrams | 4 |
| 1. A set of 3 personas that represent different types of users of the system | 10 |
| 1. A set of 3 scenarios that illustrate issues that might need to be considered in developing a useful and usable interaction design | 10 |
| 1. A set of 3 high level concept designs that illustrate distinct alternative ways that users could interact with the system | 10 |
| 1. A set of 3 detailed storyboards that specify interactions for particular parts of the system so as to ensure a high level of usability. Your storyboards should deal with at least two different types of user | 16 |
| 1. Consistency of the storyboards (item 9) and the UML model (items 1, 2 & 3) | 4 |
| TOTAL | 100% |