Intro to UML: after-course work

22 April 2010

1

You are about to start a new iteration of a project. Currently, the system includes two classes `Customer` and `Employee`. Each of these classes has two `String` attributes `name` and `address`. `Employee`s work in `Department`s, and class `Employee` has an operation `getDepartment` which returns the main `Department` in which an `Employee` works.

1. Draw a UML class diagram to record these facts. Your diagram should include attributes and operations, but need not show multiplicities or navigabilities.
2. In this iteration, it has been decided that names and addresses should be represented not by strings but as objects of new classes, and that there should be a new common abstraction `Person` of `Customer` and `Employee`. Draw a new UML class diagram showing this design.

2

Consider the following outline:

> After the ReformHealthcare party's surprise win in the 2015 UK General Election, a new type of health clinic is to be set up, using treatments whose effectiveness is highly controversial. Because the health outcomes associated with the new clinics will be of wide interest, a new computer system must quickly be developed to track patient treatments and outcomes.
>
> All patient appointments must be entered into the system by the clinic administrators. During the appointment, the doctor or nurse will enter details of the patient's condition and any treatment prescribed. Patients are requested to report on any changes in their health, one week after each appointment. They can do this either by filling in a web form or by telephoning the clinic, in which case an administrator will enter the information.

> The system must be able to produce reports of statistics such as the number of patients treated in a given period, their conditions, and their reported state of health a week afterwards.

Draft a use case diagram for the system. Comment briefly on any important ambiguities or problems you find in the system description.

3

Consider again the following code fragment:

```
public void addNode(Object node) {
    if (!canAddNode(node)) {
        return;
    }
    getNodes().add(node);

    if (Model.getFacade().isAModelElement(node)
            && Model.getFacade().getNamespace(node) == null) {
        Model.getCoreHelper().addOwnedElement(getHomeModel(), node);
    }

    fireNodeAdded(node);
 }
```

1. Draw a sequence diagram to represent just the scenario in which `canAddNode` returns `true`, `isAModelElement` returns `true`, and `getNamespace` returns a non-null namespace.
2. Use fragments to draw a sequence diagram to represent the whole of the behaviour of this fragment.
3. Think about (but you need not write about) the relative merits of using sequence diagrams to represent individual scenarios of interest, compared with using them to represent all possible behaviour.

4 (optional, open-ended)

Reconsider the Java applet you wrote last time. Document its design using several kinds of UML diagram.