Module Software Systems (201500111)
Design Test, 10 December 2015, 8:45–11:45

- Different questions will be graded by different persons. Therefore we ask you **to use a <u>separate sheet</u> for each question** (<u>not</u> the back side of another question).
- The text of this test is in English, but you may answer in Dutch if you prefer (e.g. names of diagram elements or any additional explication you want to give).
- You are allowed to consult the manual and a printed version of the slides. It is <u>not</u> allowed to consult other materials – including your own notes.
- Diagrams can be drawn with pen or pencil as you like.
- When you are ready, please hand in only the answers to the questions. You can take the test with you.

Questions 1–4 relate to the following case study.

University admission for international students

For students from the Netherlands, there are standard procedures to enroll in a university program. If you come from another country it is more complicated – an *admission board* has to decide whether an applicants fulfil the prerequisites, based on the evidence that the applicants can provide of their previous education. If you want to do bachelor's program, you must have completed appropriate secondary education; if you want to do a master's program, you must have an appropriate bachelor education. Study programs differ a lot across the world, therefore the application process involves human expertise and judgement. For countries from which the University gets a lot of applicants, decisions about admission can be simple, based on rules and experience. But the process cannot be fully automized.

The number of foreign master students has increased over the last few years. A new system is needed to support the handling of these applications. You are asked to make an initial design based on the following information

Any person who wants to apply for a study program at the university can file an application through the website. The first step is starting an application for admission. This will provide you with login details that you can use to edit the application. A lot of information and documents have to be collected and uploaded; applicants usually do multiple edits. As a final editing step, when everything is complete, you can submit the application, changing its state from 'draft' to 'submitted'.

Submitted applications are inspected by the *admission office*, a centralized body for the University as a whole. An admission officer checks the application for completeness. When the admission officer indicates that the application is complet, a notification is sent automatically to the *admission coordinator* of the study program that the applicant wants to follow.

Occasionally it happens that a submitted application is not complete. In that case the admission officer will notify the applicant that something is missing, and change the status back to 'draft', so that the applicant can edit and resubmit the application.

Some applications are never completed. In order to clear up, applications with 'draft' status are removed when applicants have not edited them for three weeks.

The admission coordinator is the staff member who does most of the work in the admission process, but eventually the decison will be taken by the *admission board*. For each study program there is an admission board responsible for that program. (However, an admission board can be responsible for several, related programs). An admission board consists of 3–5 university staff members, typically educational directors and teachers. An admission coordinator is present at meetings of the admission board, but not a member of this board. Usually (but not necessarily) the admission coordinator coordinates admissions for all programs for which the admission board is responsible.

When the admission coordinator gets an application, s/he studies the application and, in most cases, proposes a decision to the admission board.
It may happen, however, that the information is not specific enough. For example, if an applicant has a bachelor degree in an IT-related subject, a list of course titles (and grades) does not reveal whether these courses cover the prerequisite knowledge that is expected for the master program Computer Science.
In such cases there are various lines of action for the admission coordinator. One option is to request the candidate to extend the application with further documents. In that case the admission coordinator sends a message to the applicant. The status of the application is changed to ‘draft’, so that the applicant can resubmit it. The extended application will then come back via the admission office in the usual way. A second option is to ask the applicant by e-mail for clarification of some details (bypassing the system and the admission office). Applicants always respond quickly, as they do want to get a positive decision.
(*There are other ways of action, e.g., contacting someone from the university from which the applicant got a bachelor degree, which we need not include in the model*.)

When requests for further information (if any) have been satisfied, the admission coordinator prepares the application for a decision, by flagging it ‘ready for decision’ and possibly adding comments for the admission board. The comments may include a proposed decision, with arguments – or if the case not clear, the coordinator’s arguments pro and con, leaving it to the admission board to decide.

A decision implies that admission to the study program is granted or refused.
Once the admission board has taken a decision, the admission coordinator will finalize the application file in the system. Apart from storing the decision, this also involves adding a text that will be included in the letter to the applicant to announce and motivate the decision.

The admission office notifies the applicant by sending this letter.
(*Applicants who believe to have been refused for the wrong reasons can contest this by sending a letter to the board of appeal, who will take a final decision. We ignore that here to keep the design limited in size*.)

Admission is granted for one particular year. For example: if you are admitted to the master program BIT for 2015/16, you can start 1 Sep 2015 or 1 Feb 2016. The rules for the 2015/16 program will remain the same for all students who started that year. (I.e., if the program is changed for 2016/17, this does not affect students who enrolled before 1 Sep 2016.)
Due to various problems it could happen that an admission get delayed, and the same person successfully applies for admission for the next year. This is administered as two separate applications.

In addition to what was already mentioned above, the following data need to be stored in the system.

- For each applicant the following information is stored: name, e-mail address, birth date, full address, country, passport_no.
- For an application to a bachelor program: name of (secondary) school, country of the school, diploma type of the school, additional information (if any).
  For an application to a master program: the same information, plus: name of the bachelor degree, university at which the decree was obtained, abstract of the bachelor thesis.
  For each application, in addition, it is stored at which date the application was submitted, at which date a decision was taken, the decision (whether admission was granted or not).
- For each application a number of separate documents have to be uploaded: copies of diplomas, passport, etc. Each such document has a document name and a document content (PDF). Each document belongs to one specific application (if the same document would be needed for two applications, it has to be uploaded twice).
- Study programs are identified by a (three-letter) abbreviation and also have full name. For each program is it known whether it is a bachelor’s or a master’s program.

For each academic year in which the program is offered, one or two start dates are given (usually this is 1 September. In some cases it is also possible to start 1 February. It could happen, though, that a program offers an additional start in February in some years, but not all years.)

- Information about membership of the admission boards is also stored. Historical information is required here: it is unlikely, but not impossible than an issue about an admission decision will arise sometime in the future. Therefore it should be known who was a member of which admission board at any moment in the past. To this end start date and (if applicable) end date of admission board membership are stored. (These need not be aligned with start dates of academic years / programs.)
- Similarly, it is known who is / was admission coordinator for which program for which period.
- For each staff member the following information is stored: name, e-mail address, birth date, employee number, telephone number, office number.

Question 1 (Activity Diagram) [2.25 points]

Make an activity diagram for the admission process.

Question 2 (Use Cases) [2 points]

2a (Actor list) [0.5 points]

Make an actor list for the admission support system with a brief description of each actor.

2b (Use case diagram) [1.5 points]

Make a use case diagram for the admission support system.

Hint: please be aware that use cases describe direct interactions with the system, not activities outside the system.

Additional information: With use case diagrams it is not always clear how much level of detail to include. For example, when the admission coordinator prepares an application for decision, one could argue that the admission coordinator should search for the right the application before s/he can do anything with it. So the use case “Prepare decision” could include another use case “Find application.”

To keep the use case diagram limited in size, You do not have to include such additional details here, as suggested in Figure 1. You only need to include use cases when they are explicitly mentioned in the text.

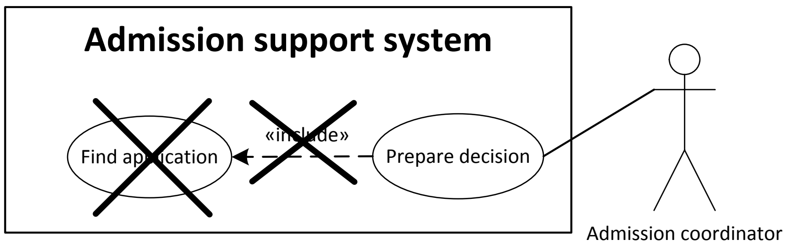

Figure 1: Use case “Prepare decision” –
don’t include further details (unless explicitly mentioned in the text)

Question 3 (Class Diagram) [3 points]

Make a class diagram for the proposed system. If different classes share several attributes, please model this by means of a generalization.

Question 4 (Sequence Diagram) [1.25 points]

Make a sequence diagram for editing an application for admission to the University by the applicant.

Additional information: In order to edit an application you have to open it first. We can distinguish two cases: starting a new application or opening a previously created application. This is shown in Figure 2. (*In either case starting/opening the application will need some interaction with the web server to authenticate the user, which we disregard for convenience*).

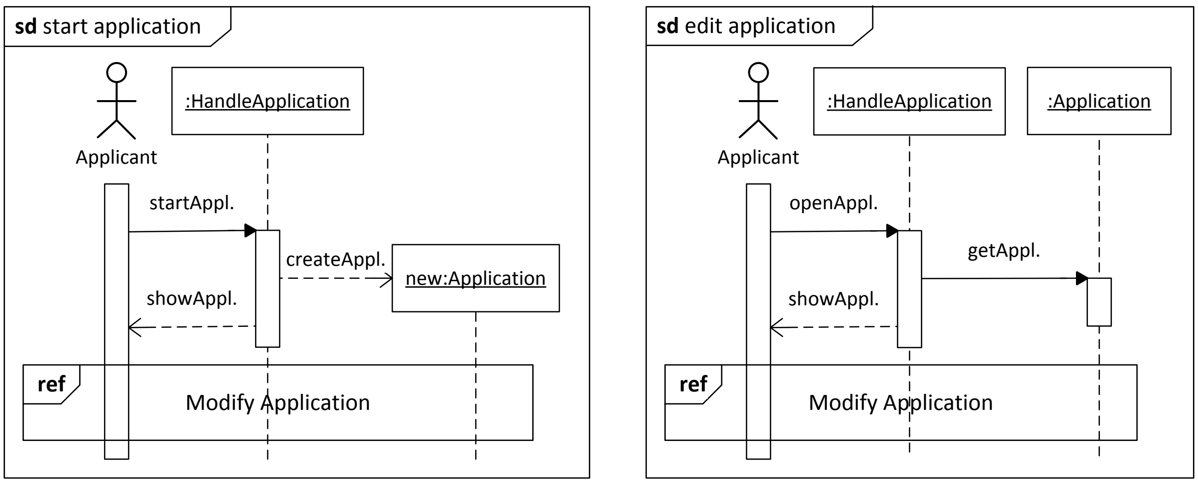

Figure 2: Creating/editing an application for admission

You are asked to give a sequence diagram for “Modify Application”.
Modifying the application could involve the following steps:

- Editing one or more information fields in the application,
- Adding a PDF document as attachment to the application,
- Submitting the application (i.e. modifying its status).

During an editing session, the first two steps can be done any number of times, in any order.
One can also stop and come back at another time (re-invoking “edit application”)
However, when the application has been submitted, no further editing is possible.

Only if, at a later stage, an admission officer or admission coordinator asks for more information, the status will be set back to “draft”, and the applicant can re-invole “edit application” again. In that case the same rules apply.

Question 5 (Software Metrics) [1.5 points]

5a (Coupling) [0.5 points]

As you have learned, the afferent coupling of a class is the number of classes that depend on it; the efferent coupling is the number of classes it depends on.

1. Suppose you start with two very similar classes C1 and C2. You introduce an abstract class A to collect the similarities, and make C1 and C2 inherit from A. How does this change the affarent and efferent coupling of C1 and C2?

2. What is the effect on the afferent and efferent coupling of a class of using the observer pattern?

5b (Cyclometric complexity) [1 point]

Consider the following two Java methods, which implement slight variations on the linear search algorithm you have already seen in Module 1.1 (in Python):

```java
public int find1(List<String> dict, String word) {
    int result = -1;
    int i = 0;
    while (i < dict.size() && result < 0) {
        if (dict.get(i).equals(word)) {
            result = i;
        }
        i++;
    }
    return result;
}

public int find2(List<String> dict, String word) {
    int result = -1;
    int i = 0;
    while (i < dict.size()) {
        if (dict.get(i).equals(word)) {
            result = i;
            break;
        }
        i++;
    }
    return result;
}
```

1. Compute the cyclomatic complexity of these two methods, taking into account that the &&-operator in the **while**-condition in `find1` only evaluates its right-hand side operand if the left-hand side operand yields **true**. Explain your computation; do not just give your answer.

2. Explain the difference you see in the cyclomatic complexity of `find1` versus `find2`, or explain why their complexity is the the same if there is no difference.