Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

Prof. Elisabetta Di Nitto and Matteo Rossi

20133 Milano (Italia)
Piazza Leonardo da Vinci, 32
Tel. (39) 02-2399.3400
Fax (39) 02-2399.3411

Software Engineering II

September 4th, 2020

Last Name

First Name

Id number (Matricola)

Notes

This exam is handled online. Rules

1. Only use a computer, NOT a tablet, NOR a smartphone
2. Activate the feed of your webcam
3. Share the screen of your computer
4. Keep the microphone on
5. No dual screens
6. No virtual machines
7. When you upload a file through the form, make sure to include your "person id" (the 8-digit number that starts with "10") in the name of the file, AT THE BEGINNING OF THE NAME (so the name of the file should be, say, "10143828_etc.", if your person id is "10143828").
8. The exam is open book, so you can check the course materials (notes, slides, books, past exams, etc.), which can be in paper form, or in electronic form. If the materials are in electronic form, you MUST use the same computer on which you are taking the exam to display them.
9. You cannot interact with other people during the exam.
10. The exam is composed of three exercises. Read carefully all points in the text!
11. **Total available time: 1h and 30 mins**

Scores of each question:

Question 1 (MAX 6) _____

Question 2 (MAX 7) _____

Question 3 (MAX 3) _____

Question 1 Alloy (6 points)

We want to develop an application that allows people to fill out crosswords. Each crossword is a schema of *n* rows by *m* columns, where each cell of the schema is identified by its ⟨row, column⟩ coordinates. A cell of the schema can be a "black box", or it can hold a letter. Some cells are associated with a number, which corresponds to a definition of a word. Definitions are categorized as "across" (i.e., the word must be written horizontally in the schema), or "down" (i.e., the word must be written vertically).

The schema is initially empty, that is, cells that can hold a letter are empty; users' goal is to correctly guess which letter goes into each cell. Users can fill an empty cell with a new letter, and delete or change an existing letter. In addition, a user can ask the application to check whether the letters already inserted in the schema are correct, and to reveal the letter that should go into a given cell.

Example of empty crossword:

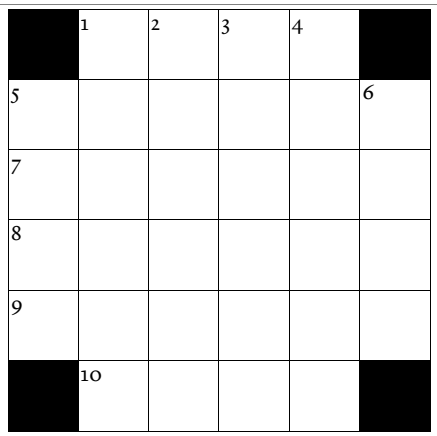
Tuesday, September 1, 2020
By Caleb Madison
© Atlantic

Across

1 Measurement for a bridge or a pair of wings
5 Topic in geometry
7 Michelangelo, for one
8 Download illegally
9 Keeps up everyone else at the sleepover, say
10 "Only Time" singer born Eithne Pádraigín Ní Bhraonáin

Down

1 Site dedicated to an idol
2 Lorenzo de' Medici, to Michelangelo
3 What a beekeeper keeps
4 Fuze competitor
5 Drains, as strength
6 Jeanne d'Arc et al.: abbr.



Consider the following Alloy definitions (which exploit the definitions related to the built-in *Int* signature):

```
sig Schema {
   cells: Int -> Int -> Cell,
   nrows: Int,
   ncols: Int
}{ all i : Int | i in (cells.Cell).Int <=> (1 <= i and i <= nrows)
   all j : Int | j in Int.(cells.Cell) <=> (1 <= j and j <= ncols)
}

abstract sig CellContent{}
sig Letter extends CellContent{}
one sig BlackBox extends CellContent{}

sig Definition{}
```

Note that relationship *cells* of signature *Schema* associates a *Cell* with each pair of coordinates ⟨i, j⟩, where *i* and *j* belong, respectively, to intervals [1..*nrows*] and [1..*ncols*] (the constraints included in the signature ensure that indexes *i* and *j* are valid if, and only if, they belong to the aforementioned intervals.

Point A (1 point). Define signature *Cell*.

Point B (3 points). Define the concept of *Status*, which represents the status of the crossword. More precisely, the status is defined by two schemas: one, called `expected`, in which all cells that are not black boxes are filled, which represents the expected result, and one, called `current`, that represents the cells as filled out by the user at that point in time (in which some cells might not have been filled out, yet). Define any constraint that is necessary to establish the correspondence between the two schemas.

Point C (2 points). Define a predicate modelling the operation *fillCell* that takes as input the coordinates i, j of a cell in the current schema of a crossword status and a letter *ltr*, and updates the status by assigning *ltr* to cell $\langle i, j \rangle$.

Solution

Point A

```
sig Cell {
   content : lone CellContent,
   num : lone Int,
   acrossDef : lone Definition,
   downDef : lone Definition
}{ content in BlackBox -> #num = 0 and #acrossDef = 0 and #downDef = 0
   #num = 1 <=> #acrossDef = 1 or #downDef = 1
}
```

Point B

```
sig Status {
   expected : Schema,
   current : Schema
}{ expected.nrows = current.nrows
   expected.ncols = current.ncols
   all i, j: Int |
       1 <= i and i <= nrows and 1 <= j and j <= ncols
       implies
       #expected.cells[i][j].contents > 0
       and
       expected.cells[i][j].contents = BlackBox iff current.cells[i][j].content = BlackBox
       and
       expected.cells[i][j].num = current.cells[i][j].num
       and
       expected.cells[i][j].acrossDef = current.cells[i][j].acrossDef
       and
       expected.cells[i][j].downDef = current.cells[i][j].downDef
}
```

Point C

```
pred fillCell[i: Int, j: Int, ltr: Letter, s: Status, s' : Status] {
  // pre-conditions
  1 <= i and i <= s.expected.nrows
  1 <= j and j <= s.expected.ncols
  not s.expected.cells[i][j] = BlackBox

  // post-conditions
```

```
  s'.expected = s.expected
  all x,y: Int |  1 <= x and x <= s.expected.nrows and
                  1 <= y and y <= s.expected.ncols and
                  not (i = x or j = y)
                  implies
                  s'.current.cells[x][y].content = s.current.cells[x][y].content
  s'.current.cells[i][j].content = ltr
}
```

Question 2 Design (7 points)
Consider the following problem: in several courses, students are looking for other peers to create study groups or project development teams. Students have specific characteristics (e.g., the study course they are taking, the classes they are attending, a set of time slots they can dedicate to collaborate with others as part of a study group/project) and issue requests concerning study groups or projects related to specific courses or topics (e.g., a student may want to collaborate with others to take the Software Engineering II research project on quantum computing). We aim to develop a software system that supports students in defining their profile and in formulating their requests. The system then performs a matching between issued requests and sends to all interested users those requests that constitute a best match.

Point A (1 point): Identify the appropriate architectural style for this application, motivating your answer.

Point B (2 points): Define the architecture of the system through a suitable UML diagram where the main system components are identified, together with their provided and required interfaces (you do not need to list in the diagram the operations defined by each interface).

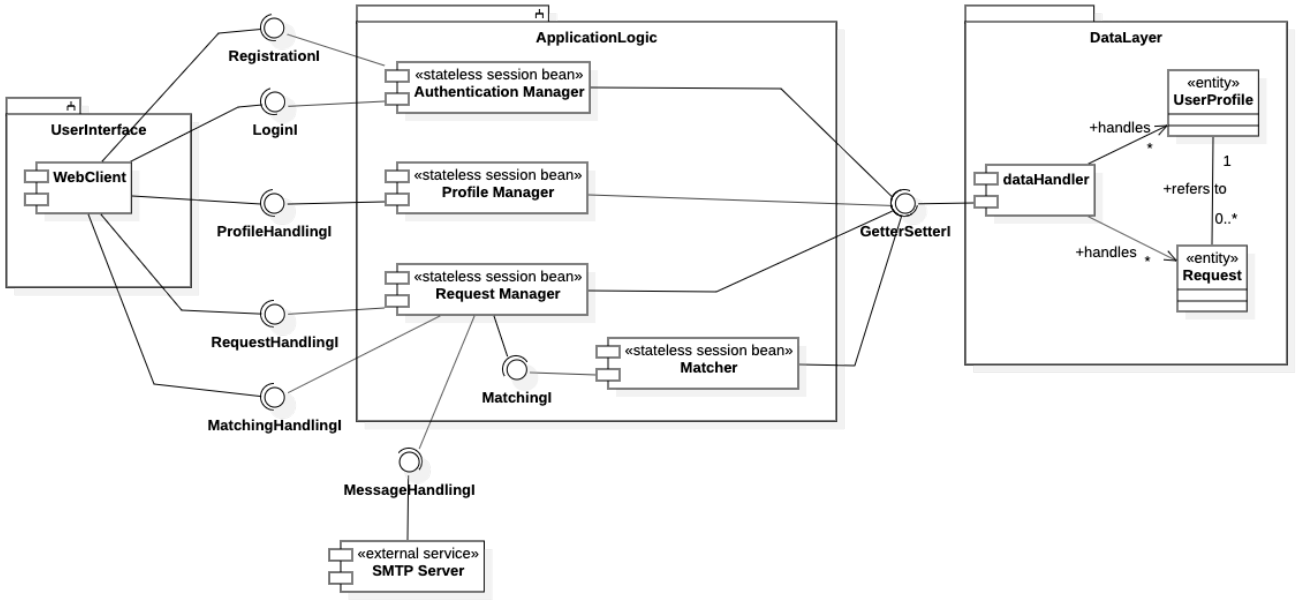
Point C (2 points): Provide a short description for each interface (briefly describe the operations that each interface defines, without detailing their parameters).

Point D (2 points): Assuming to use JEE as implementation platform, associate each component in the architecture with the corresponding JEE component type(s).

Solution

Point A: The system can be implemented as a three-tier client-server web application. This allows multiple users to access the application and insert both profiles and requests. Possible matching peers can be notified to the users either when they connect to the system or asynchronously, for example via email (in which case the system will have to be integrated with an external SMTP server), or through a publish-subscribe mechanism. To ensure that profiles and requests are univocally associated with their creators, the system must implement an authentication system.

Point B:



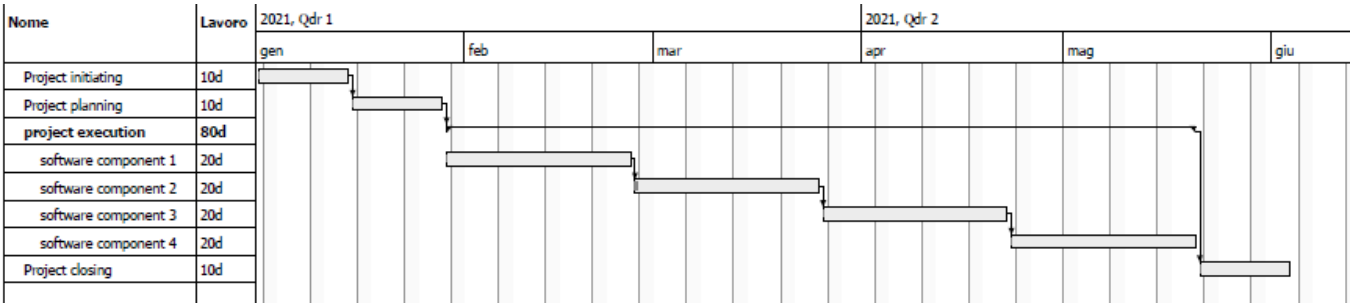
Point C:
The *RegistrationI* interface provides the registration operation.
The *LoginI* interface provides login/logout operations.
The *ProfileHandlingI* interface allows the user to insert information about his/her courses, free slots, and any personal information considered useful for a possible matching.
The *RequestHandlingI* defines the operations to insert a new request, and to modify, visualize, and delete existing requests (i.e., it offers the CRUD operations on requests).
The *MatchingHandlingI* interface offers the operations to browse through the matching requests (obtain the whole list and obtain the most relevant one).
The *MatchingI* interface provides the operation that starts the analysis of a request against all others to find possible matches.
The *GetterSetterI* is a placeholder for the operations that allow clients to get and set the attributes of the entities handled.

Point D:
This point is answered directly in the component diagram

Question 3 Project Management (3 points)
A project has been setup to build an application composed by four software components. The budget assigned to the project is 14000 euros. The project costs are the following:

- 2000 euros for the initiating phase.
- 1000 euros for the planning phase.
- 9000 euros for the executing phase: 2000 euros for the first component, 3000 for the second, 2000 for the third, 2000 for the last.
- 2000 euros for the project closing.

The schedule is the following:



A planned review is provided after 60 days from the project start; the project manager calculates the following parameters according to the Earned Value Analysis:

- Earned Value is 7500 euros.
- Actual Cost is 6000 euros.

As PM of the project please answer the following questions:

1. Inform the stakeholders about the situation of the project in terms of schedule and cost.
2. Estimate the budget at completion (EAC) of the project considering the following two options:
    a. Continue to spend at the actual rate.
    b. Continue to spend at the original rate.

Provide a short motivation to your answers.

Solution

We can compute the following quantities from the project definition:

- PV = 8000 euros
- BAC = 14000 euros

We can then conclude the following:

1. The project is running in budget because the value produced is more than the cost (EV > AC); but it is running behind schedule because the value produced is less than the value planned (EV < PV).
2. The Cost Performance Index of the running project is CPI = EV/AC = 7500 / 6000 = 1.25
    a. using the actual rate EAC = BAC/CPI = 14000/1.25 = 11.200 euros
    b. continuing at the original rate EAC = AC + (BAC – EV) = 6000 + (14000-7500) = 12500 euros