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 2 – Written Exam 2 (WE2)

June 24th, 2021

Notes

1. Remember to write your name and Id number (matricola) on the pieces of paper that you hand in.
2. You may use a pencil.
3. Incomprehensible handwriting is equivalent to not providing an answer.
4. The use of any electronic apparatus (computer, cell phone, camera, etc.) is strictly forbidden.
5. The exam is composed of 2 parts, one focusing on requirements, and one focusing on design. Read carefully all points in the text!
6. **Total available time: 1h and 30 mins**

System Description: KitchenDesigner

We want to build an application, *KitchenDesigner*, that allows users to define the layout of kitchens and to insert in such layouts the furniture and appliances (refrigerators, stoves, dishwashers, etc.) that go in them.
For simplicity, we consider only rooms that have rectangular shape. Users can define the physical features of the room (length, width, height). Moreover, they can add pieces of furniture and appliances, and move them around. The position of each piece of furniture/appliance is given by the 3D coordinates of the lower left corner of the bottom side of the item, and by its orientation (which is the angle with respect to the x axis, and which can only be a multiple of 90 degrees).
Users register with the application to be able to store and retrieve their designs. After a user finalizes his/her kitchen design, he/she can ask to have the kitchen delivered to a desired address; in this case, the kitchen is sent to production, and the user is given a probable date of delivery (producing a kitchen can take a few days, or even weeks). For simplicity, we do not consider payment. When the kitchen is ready to be delivered, the user is notified of the confirmed date of delivery.
The system keeps track of the designs created by users, to identify the most common combinations of pieces of furniture and appliances. Hence, upon request by a user, given a draft layout for the kitchen, the system returns a list of possible pieces of furniture and appliances that might be added to that kitchen.

Part 1 Requirements (6 points)

Q1 (1 point)
With reference to the Jackson-Zave distinction between the world and the machine, identify the relevant world and machine phenomena for *KitchenDesigner*, including the shared ones, providing a short description if necessary. For shared phenomena specify whether they are controlled by the world or the machine. Focus in particular on phenomena that are relevant to describe the requirements of the system.

Q2 (2 points)
Referring to the phenomena identified above, define in natural language one specific goal for *KitchenDesigner*, together with the associated domain assumptions and requirements. Explain why the identified requirements and assumptions are relevant to the fulfillment of the goal.

Q3 (3 points)
Describe the relevant elements of the domain of the *KitchenDesigner* system. You can use a UML Class Diagram for this purpose.

Solution

Q1

World phenomena:
User decides to buy a kitchen
Kitchen is produced
Kitchen is delivered

Shared phenomena, world-controlled
User creates a new project for a kitchen
User defines the dimensions of the kitchen
User adds piece of furniture/appliance to kitchen
User moves piece of furniture/appliance around the kitchen

User finalizes project
User asks kitchen to be delivered
User asks for suggestions about the kitchen
User registers to system
User logs in
The factory defines the kitchen delivery date

Shared phenomena machine-controlled
System sends the kitchen to the factory for production
System notifies user of delivery date forecast
System notifies user of actual delivery date

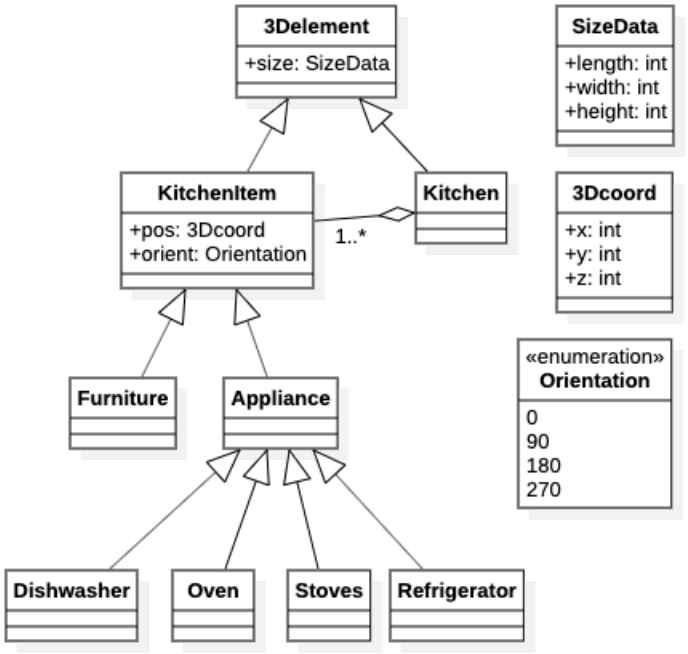
Q2
Goal
G1: Users have their desired kitchens delivered to them

Requirements
R1: The system must allow users to define the dimensions of the kitchen
R2: The system must allow users to add items (pieces of furniture and appliances)
R3: The system must allow users to remove items from a kitchen
R4: The system must allow users to move items around a kitchen
R5: The system must allow users to finalize a kitchen design
R6: The system must allow users to order a finalized kitchen design
R7: The system must notify the factory about the order
R8: The system must inform the user about the forecasted and the actual delivery date
Additional requirements related to the system storing the kitchen designs and mining them as basis to provide suggestions to users, and also requirements related to the user registering and being able to ask for suggestions, are not listed here, because they are not necessary to meet the specific stated goal, though they are relevant for the application.

Domain assumptions
DA1: The information input by the user in the application (size of the room, pieces of furniture, appliances, etc.) is accurate
DA2: After order is confirmed, the kitchen is prepared and delivered

DA1 and R1-R4 guarantee that the system knows the layout of the kitchen that the user wants. Then, R5-R8 and DA2 guarantee that the kitchen is delivered to the user.

Q3
The main elements of the domain are the kitchens and the items that are in them. Hence, the following is a possible Class Diagram describing the domain of the application.



Part 2 Design (8 points)

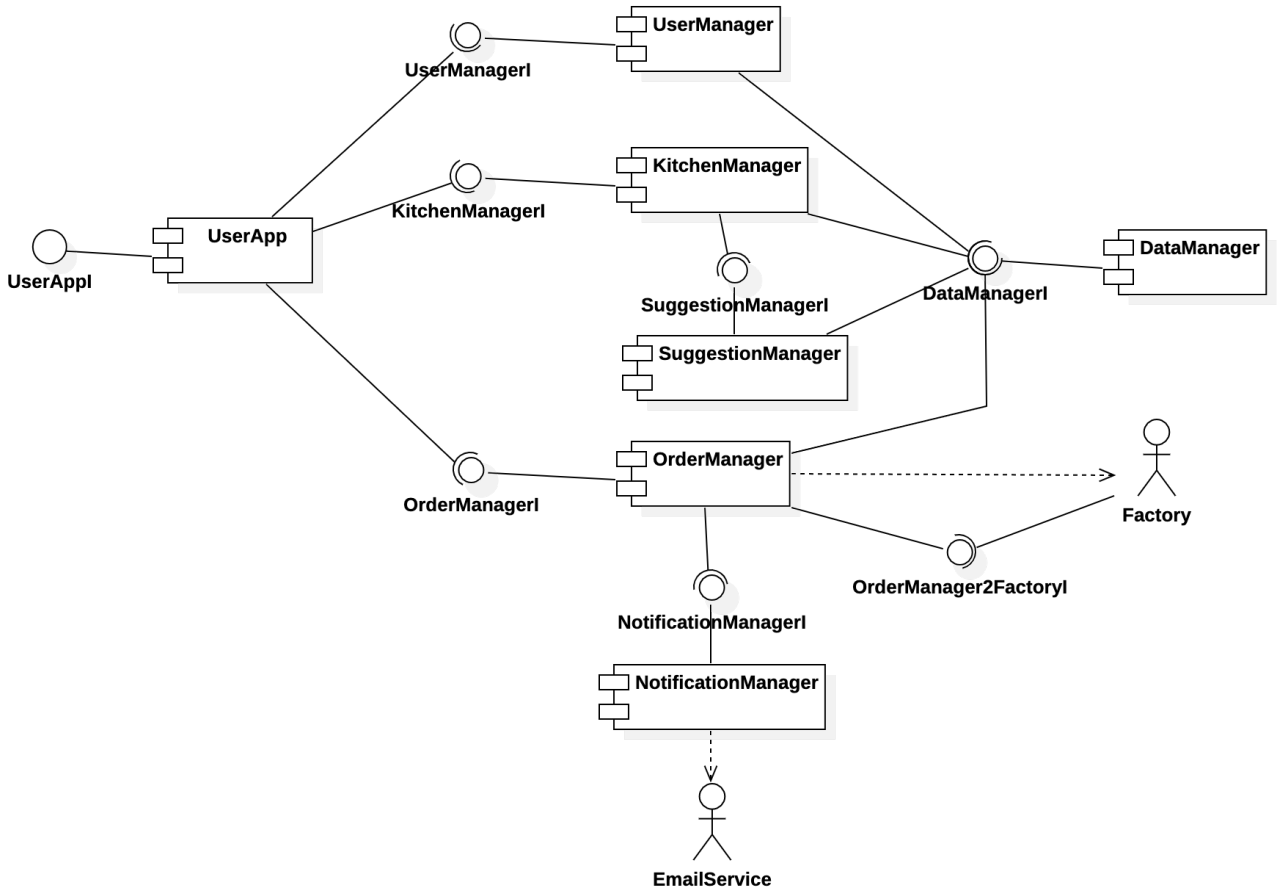
Q4 (4 points)
Assuming you need to implement system *KitchenDesigner* analyzed above, identify the most relevant components and interfaces describing them through UML Component or Class Diagrams.
Provide a brief description of each component.

Q5 (3 points)
For each interface identified in Q4, list the operations that it provides.
For each operation, you do not need to precisely specify its parameters; however, you should give each operation a meaningful enough name to understand what it does; you can also briefly describe what information operations use/produce.

Q6 (1 point)
Define a runtime-level Sequence Diagram describing the interaction that occurs among the *KitchenDesigner* components when the user asks for a list of suggested elements to be added to the kitchen.
If useful (optional), provide a brief description of the defined Sequence Diagram.

Solution

Q4-Q5



UserApp
This is the front-end for users. It allows them to interact with the system by offering the following functions through interface *UserAppI*:

- Register (input: user data)
- Login (input: userid and password)
- Create a new kitchen project (input: name)
- Set the dimensions of the kitchen (input: dimensions)
- Add an item to kitchen (input: item to be added)
- Move item in kitchen (input: item to be moved, new position/orientation)
- Remove item from kitchen (input: item to be removed)
- Ask for a suggestion (returns suggested elements)
- Finalize kitchen
- Place order

The module can keep track of the current kitchen being designed, so the user operates on the "open project", and the information does not need to be included in the calls each time.

UserManager
This component offer, through interface *UserManagerI*, the basic functions for handling users:

- Register (input: user info)
- Login (input: user id and password)

Kitchen Manager
This component provides interface *KitchenManagerI*, which handles functions related to the management of kitchens (excluding placing the order, which is handled by another component):

- Create a new kitchen project (input: name)
- Set the dimensions of the kitchen (input: kitchen id, dimensions)
- Add an item to kitchen (input: kitchen id, item to be added)

- Move item in kitchen (input: kitchen id, item to be moved, new position/orientation)
- Remove item from kitchen (input: kitchen id, item to be removed)
- Ask for a suggestion (input: kitchen id, returns suggested elements)
- Finalize kitchen (input: kitchen id)

These operations are similar to those offered by the *UserApp*, but they also include the id of the kitchen which should be modified.

SuggestionManager

This component provides, through interface *SuggestionsManagerI*, functions related to the retrieval of suggestions:

- Get suggestion (input: kitched id)

The idea is that the component periodically retrieves kitchen designs from the *DataManager*, mines them, and identifies which combinations of items are most common. Hence, it only provides a single function, for getting the outcome of this mining. Hence, the computation is mostly offline, the "get suggestion" function compares what is present in the kitchen with what is most common, and suggests additional elements.

OrderManager

This component provides, through interfaces *OrderManagerI* and *OrderManager2FactoryI* functions related to the management of orders. These are used by 2 different clients. Interface *OrderManagerI* is used by *UserApp*; it provides the following function

- Place order (input: kitchen id)

which is used to start the process to produce a kitchen. To handle the order the *OrderManager* needs to notify the factory of the new order. The handling of the order, and in particular its creation, is outside of the scope of the *KitchenDesigner* application; this is represented in the diagram by the fact that there is an interaction with the factory. When the order is complete, the *OrderManager* is informed of this through interface *OrderManager2FactoryI* (to be used by the external actor "factory") which provides the following operation:

- Kitchen completed (input: kitchen id)

Which simply informs the *OrderManager* that the kitchen is indeed ready (hence the user can be notified of the date of its actual delivery).

NotificationManager

This component handles the notification to users, in particular updates on when the kitchen will be delivered. It provides the following function through interface *NotificationManagerI*:

- Notify user (input: message to be sent, recipient)

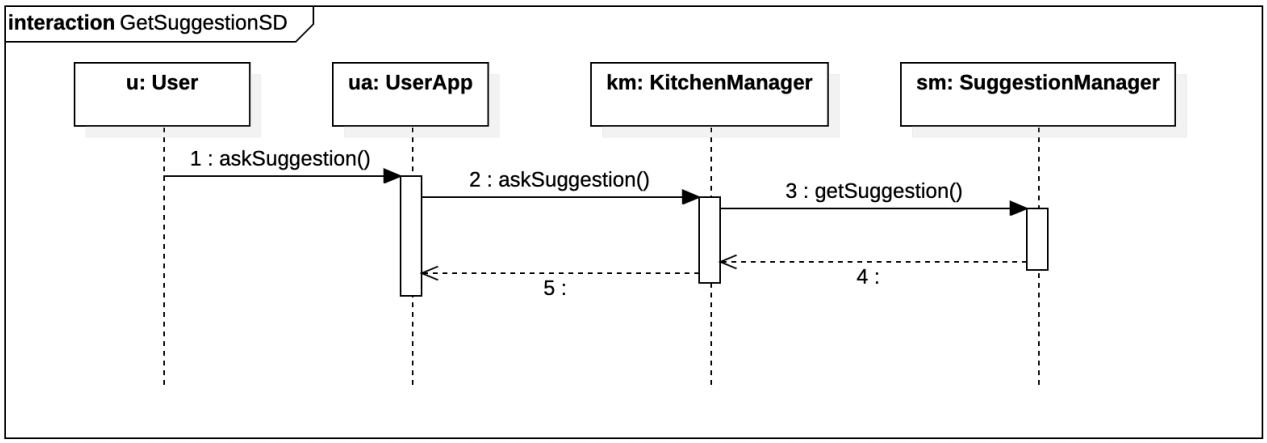
The notification is sent as an email, and for this reason it goes through an email server, which is an external component, outside of the system.

Data Manager

This component handles the data of the system, which consists essentially of Users and Kitchens. It provides interface *DataManagerI*, which includes all necessary functions to handle CRUD operations on data.

The only backend component that does not need to interact with *DataManager* is *NotificationManager*, because it relies on information provided by *OrderManager*.

Q6



As explained above, the mining of the designs is done asynchronously, not when a suggestion is requested, but offline, so it is not represented in this interaction.