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 – WE2 exam simulation

In this document, referring to a specific example, we include some representative questions that could be selected for the exam WE2 test. The number and specific formulation of questions will depend on the complexity of the case that you will be asked to analyse at the exam.

TSPmonitor case study

Consider the following situation. A software house has been tasked to create *TSPmonitor*, a crowd-based application that allows users to report on the status of Public Transport in the city. In particular, the application should allow citizens to provide short reports on the status of the bus, tram, and metro lines they take. The reports created through the application include the level of crowdedness of the line, the comfort of the trip, and the punctuality of the service. Reports created through the application are stored for statistics computation and data mining purposes. In addition, if the user allows it, reports are automatically transformed into tweets that make use of predefined hashtags.
The application can also notify users who request it about changes in the detected status of lines (e.g., when a line becomes crowded), and it can suggest alternative paths when problems strike their preferred lines. Also, thanks to its mining capabilities, the application can provide forecasts on the status of lines in the next 3 days.

Part 1 Requirements (8 points)

Q1: Given the description given above, with reference to the Jackson-Zave distinction between the world and the machine, identify the relevant world and machine phenomena, including the shared ones, providing a short description if necessary. For shared phenomena specify whether they are controlled by the world or the machine.
Q2: Referring to the phenomena identified above, define in natural language one specific goal for *TSPMonitor*, together with the associated domain assumptions and requirements. Explain why the identified requirements and assumptions are relevant to the fulfillment of the goal.
Q3: Define a use case diagram including the relevant actors and use cases.
Q4: Specify the most relevant use case among those identified in the diagram.

Solution

Q1.
World (non-shared) phenomena:
Passengers board vehicles
Passengers leave vehicles
Vehicles travel around city
Number of people actually on the vehicle is close to maximum capacity
Actual time of arrival of vehicle
Problem strikes line

World-controlled shared phenomena
User sends report with crowdedness of vehicle
User sends report with comfort of vehicle
User sends report with information about punctuality of vehicle
User allows system to create tweets from reports
User asks for forecast of status of line
User informs the system about his/her preferred lines

Machine-controlled shared phenomena
System creates tweet from received report
System notifies user of change of status of preferred line
System suggests alternative path to user when status of line is critical
System provides forecast of status of line

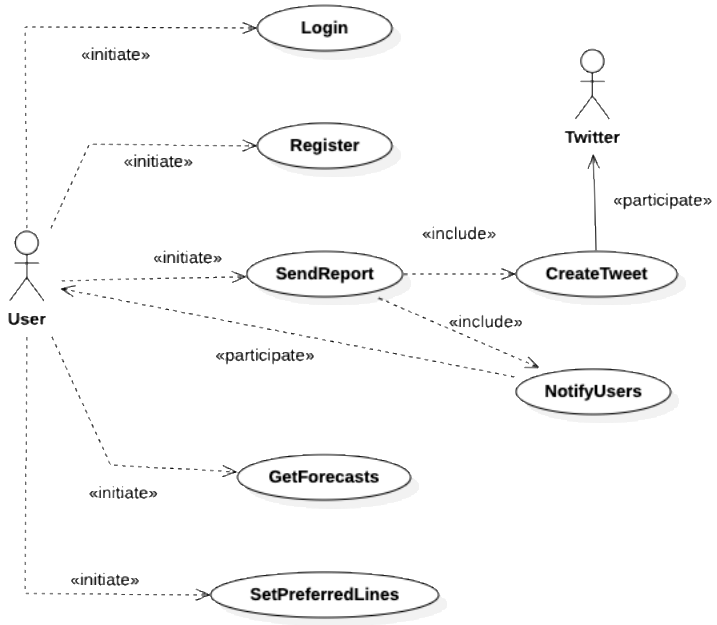
Q2.
Goal:
G: The user is informed about the current status of preferred line

Domain Assumptions:
DA1: Report sent by user provides accurate picture of current status of line
DA2: On each vehicle there is always an active *TSPmonitor* user
DA3: Active *TSPmonitor* users send reports on the status of vehicles when they board a vehicle
DA4: Information provided by users about preferred lines is accurate
DA5: Notification updates sent are received by users within few seconds

Requirements:
R1: The system shall allow users to send report with information about status of line
R2: The system shall allow users to indicate their preferred lines
R3: Upon receiving a new report on the line, the system updates the recorded status of the line, if the new status is different from the old one
R4: When it detects a change in the status of a line, the system shall send a notification to all users who have indicated the line as a preferred one

From DA2 and DA3, we have that whenever the status of the vehicle changes (the vehicle becomes crowded, or it arrives late at a stop), there is a user who sends a report to the system, which is possible thanks to R1. From DA1 and R3, we have that the system keeps an up-to-date picture of the status of a line. From R4 and DA5 we have that whenever the status of a line changes, a notification is sent to users and soon received by these last ones; in addition, from R2 and DA4 we have that the system knows which users are interested in the status of the line, and from DA5 we know that the notification will be received by the user shortly after the status of the line changes. Hence, we can conclude that the user is up-to-date with respect to their preferred lines.

Q3



Q4:

Name: SendReport
Actors: User, Twitter
Entry Condition: User notices situation aboard a vehicle of a line
Flow of Events:

1. User opens form to create report
2. User fill out details of the report:
    a. Line number
    b. Crowdedness level
    c. Comfort level of vehicle (e.g., heating off)
    d. Perceived punctuality
3. System completes the information of report with additional data (timestamp, GPS position)
4. System stores the report
5. If the User allowed it, System creates Tweet out of report
6. System computes status of line
7. If the status of the line changed with respect to previous situation, System notifies interested Users.

Exit Condition: Report is stored in System, Users are notified of changes in status

Part 2: Design (8 points)

Q5: Assuming you need to implement the system analyzed above, identify the most relevant components and interfaces describing them through UML component or class diagrams. Provide a brief description of each component.

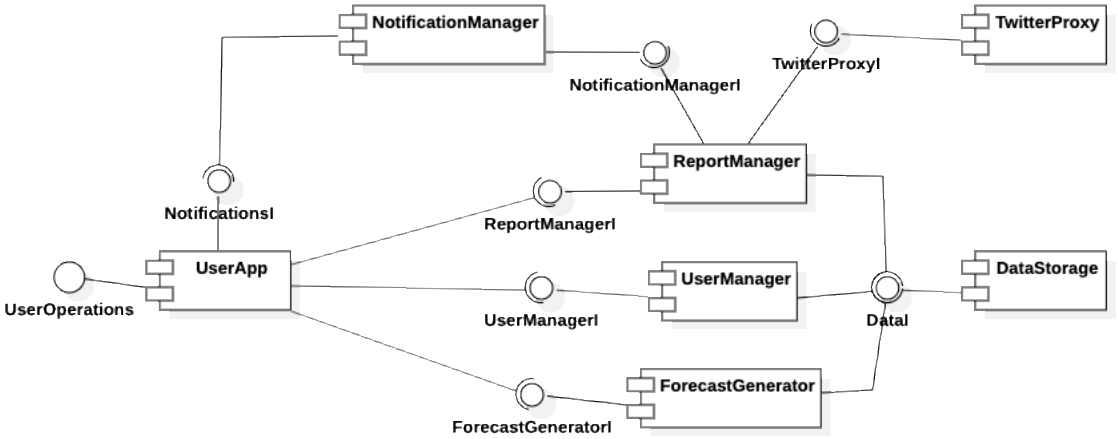
Q6: Define a runtime-level sequence diagram describing the interaction among the components for what concerns the use case described in Q4. Provide a brief description.

Q7: Describe through a deployment diagram a possible deployment of the system. Provide a justification for your choice.

Solution

Q5:

A possible architecture is the following:



Component *UserApp* is the front-end to the system used by Users. It provides functions to login/register, send reports, get forecasts, set the preferred lines. It also provides an interface that allows the system to notify users of critical situations and changes in the status of lines.

Component *UserManager* handles the registration and login of users, and also the preferences of users in terms of preferred lines; hence, it provides functions that allow users to register, log in, and set their preferred lines. The component stores the user information in the *DataStorage*.

Component *ReportManager* handles the sending of reports by Users. It provides a function to receive reports from users. It uses the *NotificationManager* and the *TwitterProxy* to, respectively, notify users of critical situations/changes in the status of lines, and to publish tweets. It stores the reports in the *DataStorage*.

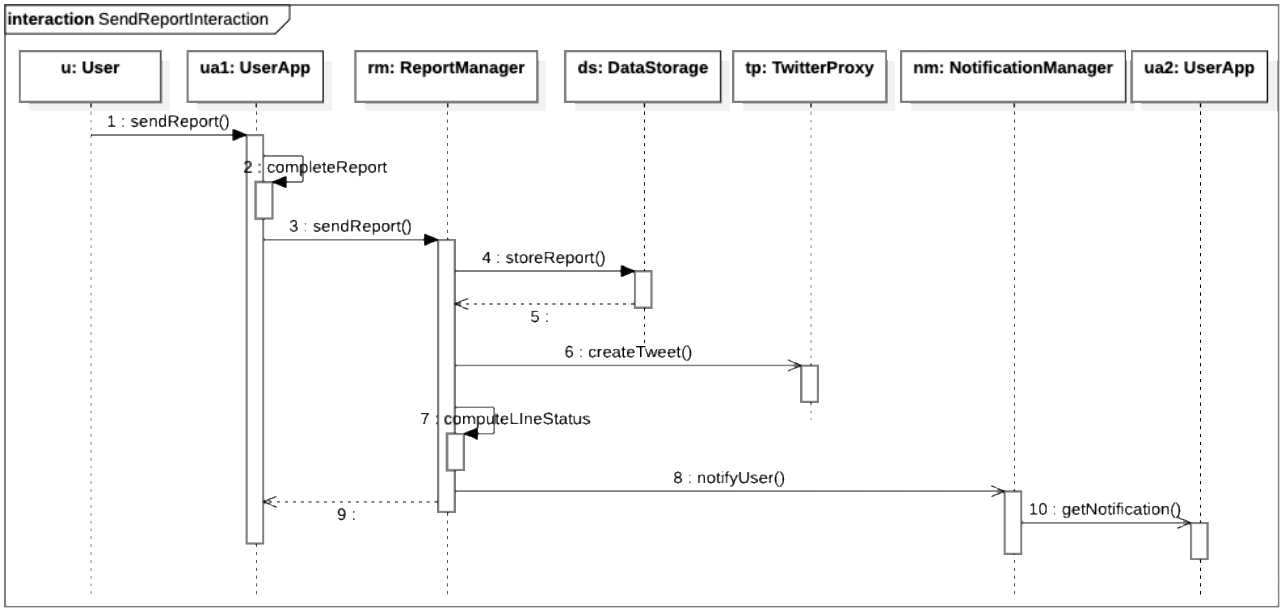
Component *NotificationManager* provides an interface that allows the *ReportManager* to send notifications, which are then relayed to the *UserApp*.

Component *TwitterProxy* allows the system to interact with the Twitter system, and it provides an interface that allows *ReportManager* to publish tweets.

Component *ForecastGenerator* provides users with forecasts about lines. It retrieves the reports from the *DataStorage*, and uses them to compute the forecasts.

Q6:

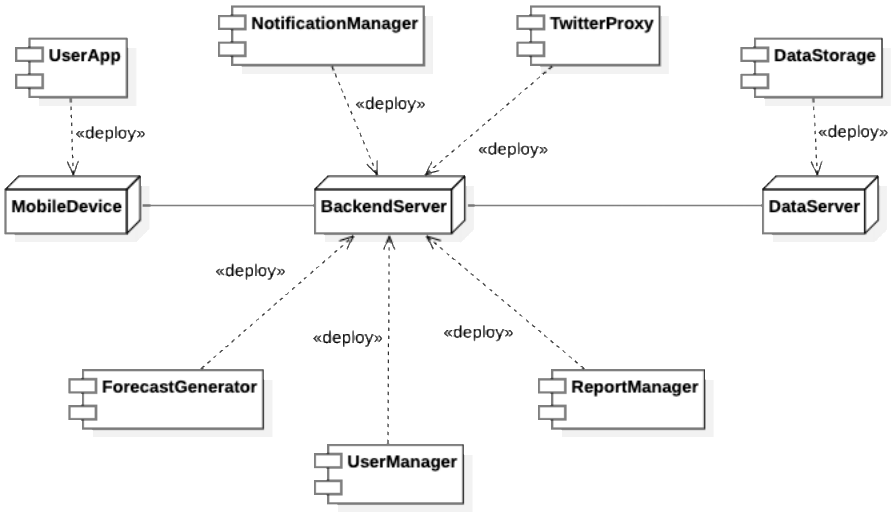
A possible runtime-level Sequence Diagram for the interaction concerning the sending of the report is the following:



The Sequence Diagram considers the case in which both the tweet and the notifications to users have to be sent. Notice that the recipient of the notification is not necessarily the same user who sent the report.

Q6:

A possible deployment for the system is the following:



The *UserApp* is simply deployed on user mobile devices. On the backend side, we use 2 servers, one for the business logic (encapsulated in components *UserManager*, *ReportManager*, *ForecastGenerator*, *TwitterProxy* and *NotificationManager*) and one for the data. In this way, the data server can be replicated independent of the rest of the system, to increase the reliability of data.