Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

Prof. Matteo Camilli, 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 1 (WE1)

February 10th, 2023

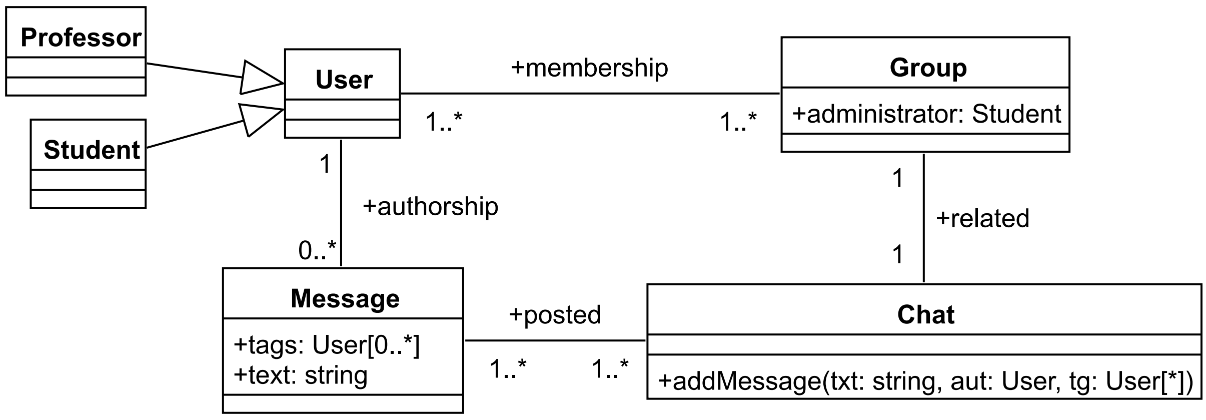
Last name, first name and Id number (Matricola)

Number of paper sheets you are submitting as part of the exam

Notes

A. Remember to write your name and Id number (matricola) on each piece of paper that you hand in.
B. You may use a pencil.
C. Incomprehensible handwriting is equivalent to not providing an answer.
D. The use of any electronic apparatus (computer, cell phone, camera, etc.) is strictly forbidden, except for an ebook reader or a plain calculator.
E. The exam is composed of three exercises. Read carefully all points in the text!
F. Total available time for WE1: 1h and 30 mins

Question 1 – Alloy (8 points)
Consider an application that allows groups of students to interact (with each other, but also with professors) through a chat to discuss courses.
The (simplified) domain model of the application is described by the following UML Class Diagram (notice that the names of the associations have been intentionally chosen to be neutral with respect to the direction of the association):



Alloy_1 (4 points) Define suitable signatures (and any necessary constraints) that formalize the structure of the above domain model.

Alloy_2 (1.5 points) Define a constraint that states that messages can be posted to the chat of a group only by one of its members.

Alloy_3 (2.5 points) Define a predicate that formalizes the behavior of operation *addMessage* defined in the domain model, which must guarantee the properties of point Alloy_2. Note that *addMessage* creates in the chat a new message authored by *aut*, having *txt* as text and tagging the users listed in *tg*.

Solution

Alloy_1

```
sig string{}

abstract sig User {}
{ some g : Group | this in g.members }

sig Student extends User {}
sig Professor extends User {}

sig Group {
  administrator : Student,
  has_chat : Chat,
  members : some User
}

sig Chat {
  msgs : some Message
}{ one g : Group | g.has_chat = this }

sig Message {
  author : User,
  text : string,
  tags : set User
}{ some c : Chat | this in c.msgs }
```

Alloy_2

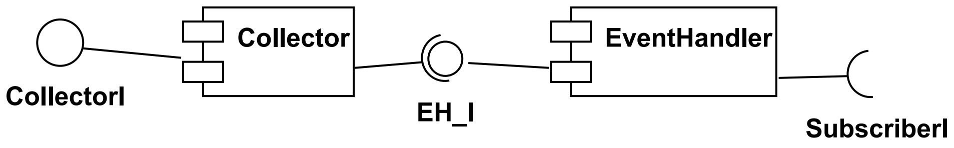
```
fact messageConstraint {
  all m : Message, c : Chat | m in c.msgs implies m.author in c.~has_chat.members
}
```

Alloy_3

```
pred addMessage [txt: string, aut : User, tg : set User, c, c': Chat, g, g': Group]
{
  // precondition
  aut in c.~has_chat.members
  g.has_chat = c

  //postcondition
  g'.has_chat = c' and
  g'.administrator = g.administrator
  g'.members = g.members
  some m : Message | m.author = aut and m.text = txt and m.tags = tg
                     and
                     not (m in c.msgs)
                     and
                     c'.msgs = c.msgs + m
}
```

Question 2 – Architectures (5 points)
Consider the following UML Component Diagram, describing the architecture of the core of a publish-subscribe layer:



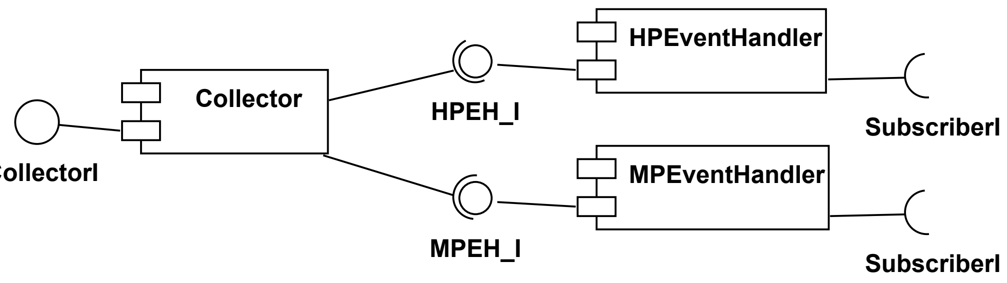
The layer is in charge of receiving published events through the *Collector* and of sending these events to subscribers through the *EventHandler* (which assumes subscribers offer the *SubscriberI* interface to allow for the delivery of events). The *EventHandler* stores internally the table keeping track of event subscribers; we assume that the table does not change over time (or it changes so infrequently that, for our purposes, we can consider it fixed). The *Collector* component is a light-weight worker node that simply takes incoming requests for event publication and dispatches them to the handler.

Arch_1 (2 points)
After the first version of this system is delivered, developers realize that there are 2 kinds of events: high-priority ones (e.g., alarms) and medium-priority ones (e.g., warnings).
Describe how developers could change the architecture above to make sure that the handling of high-priority events is not negatively affected by the presence of medium-priority ones. Provide a new version of the component diagram with your solution and provide a motivation for your choice.

Arch_2 (3 points)
Suppose that we want that the delivery of high-priority messages has a level of availability of 99.99%, and that of medium-priority messages has a level of availability of 99%.
Suppose also that we have at our disposal 2 servers that have 99% availability, 2 servers that have 95% availability, 2 servers that have 90% availability, and 4 servers that have 80% availability.
How would you deploy the components of your component diagram to achieve the desired levels of availability?

Solution

Arch_1
A possible way to solve the issue is to introduce in the architecture separate components to handle the different types of events (high-priority, or HP, ones and medium-priority, or MP, ones).



This architecture fosters scalability, since each component can be replicated to handle increasing loads of requests. The replication requires the creation of a new copy of the *EventHandler*s subscription table, but the assumption about the stability of this table can make this copy less critical. One can indeed imagine that there is a fixed pool of *Collector* components, but the number of handlers for each type of event can increase or decrease based on the current load.

Arch_2
The delivery of each type of messages involves the series of the *Collector* component(s) and the corresponding handler (so, the chain for the delivery of HP events is given by the replicas of the *Collector* component and the replicas of the *HPEventHandler* component). The *Collector* component(s) belongs to each chain, so it must have an availability that allows us to meet all constraints, and in particular those for the delivery of HP events. Hence, it is reasonable to use for it the most available servers, but we also need to leave some highly available servers for the second part of the chain of HP events, i.e., *HPEventHandler*. Hence, if we use (in parallel) 1 server with 99% availability and 2 servers with 95% availability, we reach an availability for *Collector* that is 99.998%. Similarly, if we have in parallel 1 server with 99% availability, 2 with 90% availability, and 1 with 80% availability for *HPEventHandler* we obtain, for this component, 99.998% availability, and the chain of the two components has 99.996 availability, which meets the desired threshold.
Then, concerning the chain for the delivery of MP events, we use the remaining 3 servers with 80% availability to replicate component *MPEventHandler*. The parallel of the 3 servers gives an availability of 99.2%, and the series with the Collector component gives an availability of 99.1%, which is enough for our purposes.

Question 3 – Project Management (3 points)
Consider an implementation activity that comprises the parallel development of 2 parts of a software. The first part concerns the implementation of a mobile application, whereas the second part concerns the implementation of a notification subsystem.

The implementation of the mobile app is planned to take a total of 4 weeks, divided in the following way:

- 2 weeks for the implementation of the user interfaces (budget 2000 euros);
- 1 week for the implementation of the front-end application logic (budget 1500 euros);
- 1 week for testing (budget 1200 euros).

The implementation of the notification component, instead, is planned to take a total of 3 week, divided in the following way:

- 1 week to implement the subscribe/unsubscribe functions (1500 euros);
- 1 week to implement the delivery function (1500 euros);
- 1 week of testing (1200 euros).

After 2.5 weeks, the team developing the mobile app has realized the user interfaces, and 20% of the application logic. The total cost of the activity is 4000 euros.
The team developing the notification component, instead, has developed the subscribe/unsubscribe functions, and 90% of the delivery function, for a total cost of 2000 euros.

Answer the following questions, providing a short motivation to your answers.

PM_1 (1.5 points)
What is the situation of the project in terms of schedule and cost?

PM_2 (1.5 points)
Estimate the budget at completion (EAC) of the mobile app part of the project considering the following two options:

1. Continue to spend at the actual rate.
2. Continue to spend at the original rate.

Solution

PM_1
For the mobile app, we have the following:

- EV (Earned Value) = 2000+0.2*1500 = 2300 euros
- PV (Planned Value) = 2000+0.5*1500 = 2750 euros
- AC (Actual Cost) = 4000 euros

For the notification component:

- EV = 1500+0.9*1500 = 2850 euros
- PV = 1500+1500 + 0.5*1200 = 3600 euros
- AC = 2000 euros

So we have:

- total EV = 2300 + 2850 = 5150 euros
- total AC = 4000+2000 = 6000 euros
- total PV = 2750 + 3600 = 6350

Hence, since total EV < total AC (i.e., 5150 < 6000), the project is, overall, over budget (due to the front-end; indeed, the notification component is in budget, because in this case it holds EV > AC, or 2850 > 2000).
The project is also, overall, behind schedule, since total EV < total PV (both components are indeed behind schedule).

PM_2
We have, for the mobile app:

- BAC (Budget At Completion) = 2000+1500+1200 = 4700 euros
- CPI (Cost Performance Index) = EV/AC = 2300/4000

So, if the assumption is to keep spending at the actual rate, we have:

- EAC (cost Estimate At Completion) = BAC/CPI = 4700 * 4000/2300 = 8174 euros.

If, instead, the assumption is to keep spending at the original rate, then:

- EAC = AC + (BAC – EV) = 4000 + 4700 – 2300 = 6400.