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 1 (WE1)

January 13th, 2021

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 7) _____

Question 2 (MAX 6) _____

Question 3 (MAX 3) _____

Question 1 Alloy (7 points)
Consider the mechanisms of a publish-subscribe framework. In the framework there are 2 types of components, publishers and subscribers. For simplicity, assume that a component can only be either a publisher, or a subscriber, but not both. Publishers publish events, and subscribers receive them. Each event regards a topic. Each publisher publishes events that correspond to a set of topics; dually, subscribers subscribe to topics in which they are interested, and for which they will receive events. Each publisher keeps a buffer of events to be sent of up to 10 events, and each subscriber keeps a buffer of at most 5 incoming events.

Point A (2 points).
Define Alloy signatures that capture publishers, subscribers, and events they publish/subscribe. Define suitable constraints to make the system consistent.

Point B (1 points).
Define an Alloy signature to describe a configuration of the publish-subscribe framework.

Point C (2 points).
Define a predicate capturing the operation with which a subscriber subscribes to a topic. The operation takes as input a configuration and a subscriber and updates the topics of the subscriber in the configuration.

Point D (2 points).
Define a predicate capturing the operation with which an event is dispatched to a subscriber. The operations takes as input a configuration, an event, and a subscriber, and adds the event to the buffer of the subscriber in the configuration.

Solution

Point A

```
sig Topic{}
sig Event{
  topic : Topic
}

sig Component{}

sig Publisher extends Component{
  pubtopics : set Topic,
  outbuffer : set Event
}{ outbuffer.topic in pubtopics
   #outbuffer <= 10
}


sig Subscriber extends Component{
  subtopics : set Topic,
  inbuffer : set Event
}{ inbuffer.topic in subtopics
   #inbuffer <= 5
}
```

Point B

```
sig Configuration{
  pubs : set Publisher,
  subs : set Subscriber
}
```

Point C

```
pred subscribe[c, c' : Configuration, s, s': Subscriber, t : Topic]{
  // pre-conditions
  s in c.subs
  not (t in s.subtopics)
  // post-conditions
  s'.inbuffer = s.inbuffer
  s'.subtopics = s.subtopics + t
  c'.pubs = c.pubs
  c'.subs = c.subs - s + s'
}
```

Point D

```
pred dispatch[c, c' : Configuration, s, s' : Subscriber, e : Event]{
  // pre-conditions
  e in c.pubs.outbuffer
  s in c.subs
  not (e in s.inbuffer)
  #s.inbuffer <= 4
  e.topic in s.subtopics
  // post-conditions
  s'.inbuffer = s.inbuffer + e
  s'.subtopics = s.subtopics
  c'.pubs = c.pubs
  c'.subs = c.subs - s + s'
}
```

Question 2 Testing (6 points)

Consider the following piece of C code:

```
0 int func(int x){
1   int k, i, y;
2   int res = 0;
3   if (x < 1)
4     return res;
5   if (x % 2 != 0)
6     return res;
7   y = x;
8   i = 0;
9   while (i <= 10 && y % 4 != 0){
10    y = y*y;
11    i = i+1;
```

```
12  }
13  if (i > 5)
14    res = k;
15  else
16    res = res+1;
17  return res;
```

Point A (2 points):
Identify the def-use pairs for program `func` and say whether they highlight anything unusual.

Point B (2 points):
Use symbolic execution to determine the feasibility (and, in case, the path condition) of the following path:

0, 1, 2, 3, 5, 7, 8, 9, 10, 11, 12, 9, 10, 11, 12, 9, 13, 15, 16, 17

Point C (2 points):
For any possibly erroneous situation highlighted by the def-use analysis, use symbolic execution to argue whether or not the problem can indeed manifest itself.

Solution

Point A:
x: <0, 3>, <0, 5>, <0, 7>
y: <7, 9>, <7, 10>, <10, 9>, <10, 10>
k: <?, 14>
i: <8, 9>, <8, 11>, <8, 13>, <11, 9>, <11, 11>, <11, 13>
res: <2, 4>, <2, 6>, <2, 16>, <14, 17>, <16, 17>

There is possibility that k is used at line 14 without having been initialized first.

Point B:
0. x = X
2. res = 0
3. $X \geq 1$
5. X = 2K
7. y = X
8. i = 0
9. $0 \leq 10$ and not exists P s.t. X = 4P (i.e., not exists P s.t. 2K = 4P)
10. y = X*X = 4K*K
11. i = 1
9. $1 \leq 10$ and not exists P s.t. X*X = 4P (i.e., not exists P s.t. 4*K*K = 4P)
The last condition is not feasible, since it is enough to consider K*K = P to violate the condition

The path is not feasible.

Point C:
Given the symbolic execution of Point B it is clear that the loop cannot be executed more than once. Hence, the value of variable i will never be greater than 1, and at line 13 it will never be i > 5, so line 14 will never be executed. Hence, the problem highlighted by def-use analysis will never manifest itself.

Question 3 Project Management (3 points)

Assume that the publish-subscribe framework described in Question 1 is implemented through a server that receives all publish and subscribe events and acts as dispatcher between publishers and subscribers.

Calculate the function points for the part described server. Motivate your choices.

Refer to the following table to associate weights to the function types:

	Weights		
Function types	Simple	Medium	Complex
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiry	3	4	6
Internal Logic Files	7	10	15
External Interface Files	5	7	10

Solution
Internal Logic Files
Topics(topicID, description) → simple, weight 7
Events (eventID, topicID, body) → simple, weight 7
Subscriptions(subscriptionID, topicID, componentEndPoint) → simple, weight 7

There is no External Interface File

External Inputs
Subscribe → it requires only the update of the internal data structure, simple, weight 3
UnSubscribe → it requires only the update of the internal data structure, simple, weight 3
publishEvent→ it triggers the retrieval of all subscriptions associated with the corresponding topic. We can classify this as simple, weight 3

External Outputs
sendEventToRecipient → the received event is sent to all endpoints of the subscribing components, simple, weight 4

There is no External Inquiry

Total number of FPs = 4*7+2*3+4=34