# Software Engineering

Books or notes are **not** allowed.
Write only on these sheets. **Concise** and **readable** answers please.

Surname, name, matricola _____

*Bus information*
Consider a metropolitan area with many bus lines and bus stops. A display at each bus stop shows the estimated waiting time for all the bus lines stopping there. Besides, a user can obtain the same information sending an sms to a certain number, with the ID of the bus stop.

In the following you should analyze and model the application that provides waiting time information for buses.

The solution proposed models not only the software part but the whole system. Another option would be to model only the software part. However the software part has at least two modules (a central program and a program in each bus stop). So analyzing the system is probably clearer.

1 (15 points) – a. Define the context diagram (including relevant interfaces)

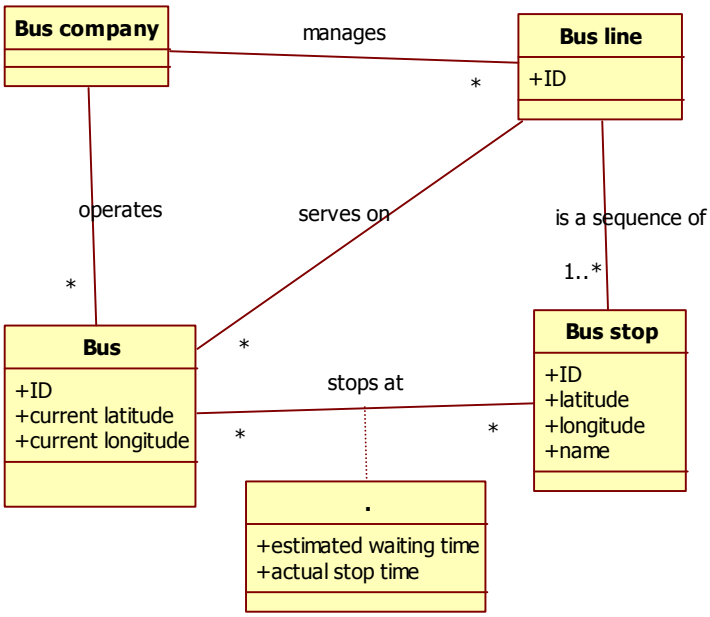

Interfaces: screenshots on display to show bus info to the Customer.
Functions sendSms, receiveSms on interface with sms gateway, sms received with ID of bus stop, sms sent with bus line number + estimated arrival time

List the requirements in tabular form

| ID | Type (Functional Non Functional) | Description |
|---|---|---|
| 1 | F | Compute estimated arrival time for a bus at a bus stop (for all buses stopping at that bus stop, for all bus stops) |
| 1.1 | F | Display information from F1 at bus stop |
| 1.2 | F | Provide information from F1 by sms to users requesting it by sms |
| | F | 1.3 Obtain geographical position of buses, 1.4 obtain or estimate speed of buses, 1.5 store distance between bus stops, 1.6 compute estimated time from bus stop to bus stop, 1.7 compute next bus stop given current bus stop and bus line |
| 2 | | Send receive info between bus stop and central system |
| 3 | NF Precision | The estimated arrival time (see R1) should have precision within x% of actual |
| 4 | NF Privacy | Cell numbers of people requesting information by sms should NOT be stored by the system |
| 5 | NF usability | Display at bus stop should allow users with normal sight to read information at least from a distance of z meters |
| 6 | NF performance | Estimated arrival times should be refreshed at least every w seconds (w is linked to x of R2, if precision is one minute it is useless to refresh every second) |
| 7 | NF performance | User should receive information via sms at most within k sec from sending sms |

Define the  key concepts and entities and their relationships (UML class diagram) for the application

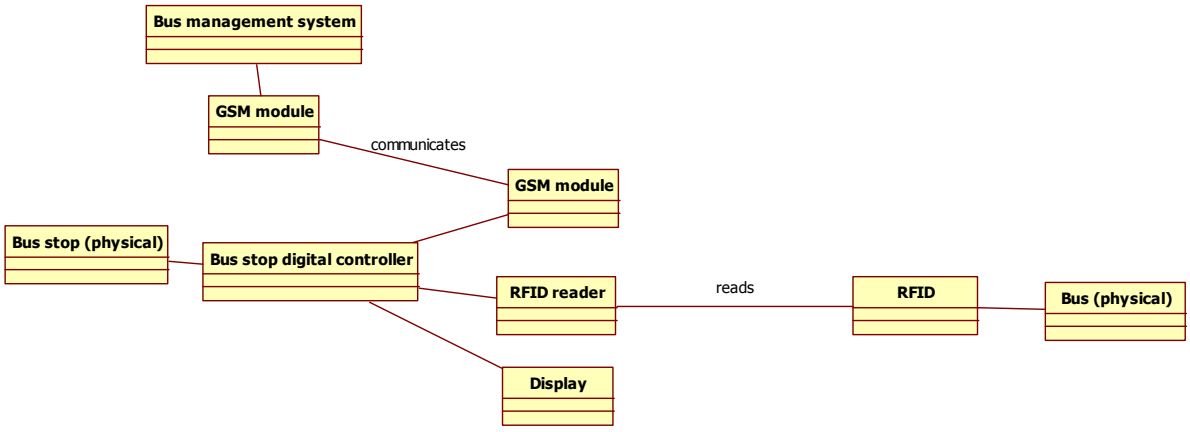
A bus line is a sequence of bus stops (ex line 13 stops in bus stops 10, 3 and 24). A bus serves on a certain bus line and at each time has a certain position. The bus line defines the path a bus must follow and is therefore essential to estimate arrival times
The simplest algorithm that can be used is
Speed = distance / time and therefore estimated time = = remaining distance to next bus stop /average bus speed between bus stop x and bus stop y)



Define the system design (key hardware and software components of the application, UML class diagram) for the application.  Describe the key design choices.

There are many design options. One option is to put a GPS and a GSM module on each bus, so that each bus communicates its position to the central management system frequently (ex each minute). The system knows the position of each bus stop. Therefore, the system can estimate the speed of each bus on each trip busstop to busstop, and send the estimated arrival time to each bus stop. So the bus stops must be connected to the central system (wifi or wimax or GSM).

Another option is to put a transponder (ex RFID) on each bus and each bus stop, and connect the bus stops to the central system. The bus stop recognizes a bus when it passes (or stops) in front of it, and communicates this info to the central system, that can compute the estimates time for the bus to reach the next bus stop.
The GPS solution is more expensive (more hardware, more communication) but also more precise, and has other advantages (the GSM module can be used to ask for maintenance or help or to reroute the buses).
We describe here the non GPS solution. Software modules will run on the bus stop digital controller and on the central bus management system.



Define one scenario describing the update of waiting times on the display of a bus stop subsequent to a bus stopping at the previous bus stop.
Precondition: Bus B serves line L1, with stops S1, S2, S3. B is between S0 and S1
Postcondition: B is at S1

| Step | Description | Req ID |
|---|---|---|
| 1 | B arrives at S1 | |
| 2 | S1 sends info to central system, B is at S1 at time T1 | 1.3 |
| 3 | Central system retrieves line of bus B, L1, and computes stop next to S1 on L1 = S2 | 1.7 |
| 4 | Central system computes estimated arrival time for B at S2 = T2 | 1.4<br>1.5<br>1.6 |
| 5 | Central system sends info to S2 : B serving L1 will arrive at T2 | 2 |
| 6 | Display at S2 updates waiting time for L1 | 1.1 |

2 (8 points) -Define black box tests for the following function, using equivalence classes and boundary conditions. The computes if a child has priority for a transplant

boolean hasPriority(int birthDay, int insertionDay)

Input: birthday, birthday of the child, expressed as number of days since 1 1 2000
insertionDay, day when the transplant was asked for, also expressed as number of days since 1 1 2000

Output: TRUE if the child is aged less than 2 years (by definition 730 days)
AND
The transplant was asked for when the child was less than one year old (365days)
FALSE otherwise
Remarks: the tests should be performed today (13 7 2012 or day 4577 from 1 1 2000)
Childs born before 1 1 2000 have no right to transplant

Ex. hasPriority(4383, 4384) → T (or born 1 1 2012 and transplant asked the day after)
hasPriority(3653, 3654) → F (or born 1 1 2010 and transplant asked the day after)

SOLUTION
Each row in the table represents an equivalence class. Total number of equivalence classes = 3x3x3x2 = 54 For this reason some combinations are pruned at the first invalid range encountered. For the same reason boundary test cases are defined around one equivalence class only (see $T_b$ test cases).

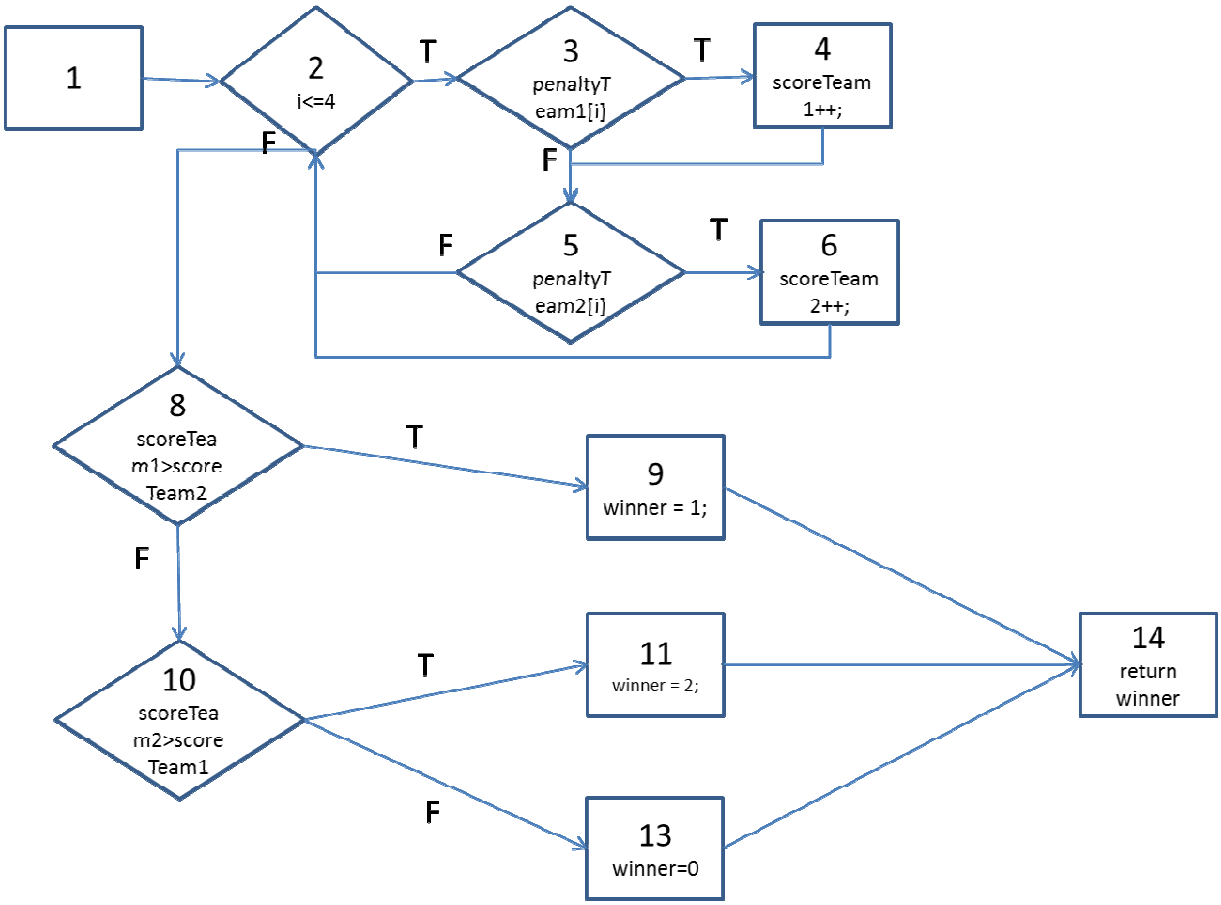
| *birthDay* | *insertionDay* | *Birthday vs insertion Day* | *Child Age* | *Valid Invalid* | *Test case* |
|---|---|---|---|---|---|
| [minint, 0[<br>[0, 4577]<br>[4578, maxint] | [minint, 0[<br>[0, 4576]<br>[4577, maxint] | insertionDay < birthDay<br><br>birthDay <= insertionDay <= birthDay+365<br><br>insertionDay > birthDay+365 | Age <= 2 years<br><br>Age > 2 years | | |
| [minint, 0[ | - | - | - | I | T(-3,1000) → err |
| [0, 4577] | [minint, 0[ | - | - | I | T(1000,-3) → err |
| | [0, 4577] | insertionDay < birthDay | - | I | T(1000,9999) → err |
| | | birthDay <= insertionDay <= | Age <= 2 years | V | T(4384,4390) → TRUE |

| | | | | | |
|---|---|---|---|---|---|
| | | birthDay+365 | | | Boundaries:<br>(birthDay=insertion Day)<br>Tb(4384,4384)→ TRUE<br><br>(insertionDay= birthday+1)<br>Tb(4384,4385)→ TRUE<br><br>(insertionDay= birthday+1)<br>Not tested because already tested in eq. class<br>insertionDay < birthDay<br><br>(insertionDay= birthday+365)<br>Tb(4077,4442) → TRUE<br><br>(insertionDay= birthday+366)<br>Tb(4077,4443) → FALSE<br><br>(insertionDay= birthday+364)<br>Tb(4077,4441) → TRUE |
| | | | Age > 2 years | V | T(3653,3654) → FALSE |
| | | insertionDay > birthDay+365 | Age <= 2 years | V | T(4077,4474) → FALSE |
| | | | Age > 2 years | V | T(3653,4474) → FALSE |
| | [4577, maxint] | - | - | I | T(3653,5000)→ Err |
| [4578, maxint] | - | - | - | I | T(5000,3650)→ Err |

3 (7 points) – For the following function define the control flow graph, and define test cases to obtain the highest possible node coverage, edge coverage, multiple condition coverage, loop coverage, path coverage.
For the test cases, **write only the input value**.

```
public int penalties(boolean[] penaltyTeam1, boolean[] penaltyTeam2){

1                   int scoreTeam1= 0, scoreTeam2 = 0 , winner=0;
2                   for (int i=0;i <= 4; i++){
3                             if (penaltyTeam1[i])
4                                   scoreTeam1++;
5                             if (penaltyTeam2[i])
6                                   scoreTeam2++;
7                   }
8                   if (scoreTeam1 > scoreTeam2)
9                         winner = 1;
10                  else if (scoreTeam2 > scoreTeam1)
11                        winner = 2;
12                  else
13                        winner = 0;
14                  return winner;
15}
```



| Coverage type | Feasibility (Y/N) | Coverage obtained (%) | Test cases |
|---|---|---|---|
| Node | Y | 100 % | `T1([T,T,T,T,T], [T,F,T,T,T])`<br>`T2([T,F,T,T,T], [T,T,T,T,T])`<br>`T3([T,T,T,T,T], [T,T,T,T,T])` |
| Edge | Y | 100% | `T1, T2, T3` |
| Multiple condition | N | - | `-` |
| Loop | N | 2/3 | `T1` |
| Path | N* | * | * |

*The possible paths in loop are the total number of possible combinations of penaltyTeam1 and penaltyTeam2 is $2^{10}$ = 1024 combinations.
Then there are 3 more paths on node 8 to 14 (8-9-14, 10-11-14, 10-13-14 ).
So, the total number of paths is 1024*3= 3072.
For practical reasons we do not write tests to cover all these possible paths.

4 (1 points) – Give the definition of calendar time in the context of project management, and its unit of measure

Days, weeks, months in calendar, either absolute (july 13) or relative (day1, day2 ..)

5 (1 points) – Describe the CCB (Configuration Control Board) approach for change management

See slides

6 (1 points) – what is a Configuration Item?

See slides

7 (bonus question) – In question 2 test cases may depend on the day they are executed. How would you approach this problem from the point of view of regression tests?

In practice every day the test cases should be 'shifted' by one day
For instance on 13 7 2012 we have this test T(4384,4390) → T
On 14 7 2012 we need T(4385,4391) → T

Doing this by hand is error prone and time consuming.
So the tester could write a script to automatically produce inputs to test cases in function of when the test cases were written (ex 13 july ) and when they are run (today > 13 july)

T( produceInput(4384, 13 july 2012), produceInput(4390, 13 july 2012)) → T

with

produceInput(4384, 13 july 2012) → 4384 +( today - 13 july 2012 )