Software Engineering

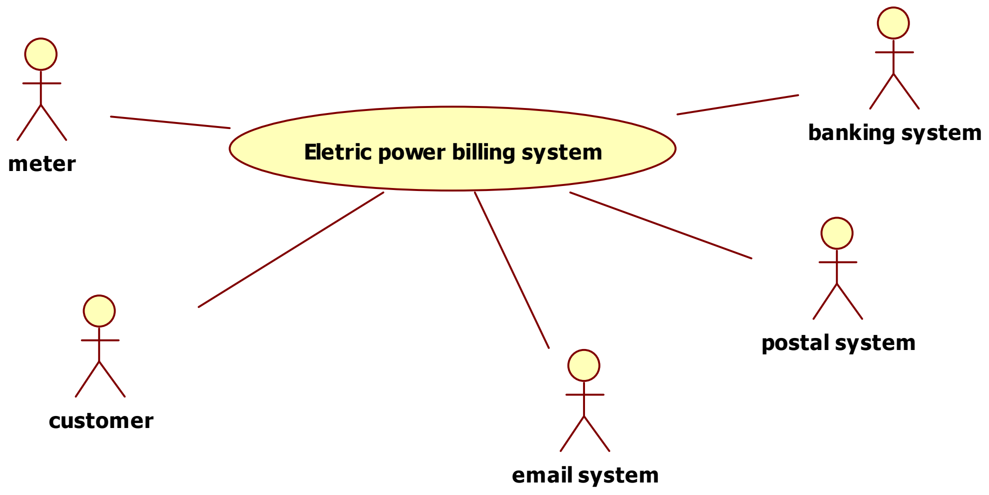
Books or notes are **not** allowed.
Write only on these sheets. **Concise** and **readable** answers please.

Surname, name, matricola ____

Electric Power Billing
Each house connected to the electric network has a meter. The meter computes and stores how much electrical energy (kW hour) is used by the house. The meter sends regularly (say every 24 hours) the consumption over the electric wires (the electric wires of the power network can in fact be used as data transmission lines). A remote central server receives the data, stores it and computes a bill (say every two months), sends the bill to the customer. The bill can be sent in two different ways: electronically (email or web site accessible to the customer via username and password) or on paper.

In the following you should analyze and model the application that collects power usage from each meter, computes the bill and sends it to the customer.

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

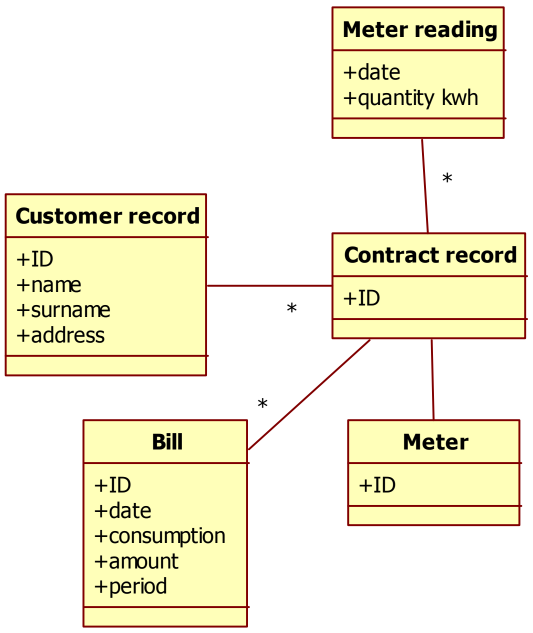


Interfaces: to customer: gui, web page
to postal system: web service offered by post company
to banking system: web service
to email system: web service, POP, IMAP, ..
to meter: electrical wires

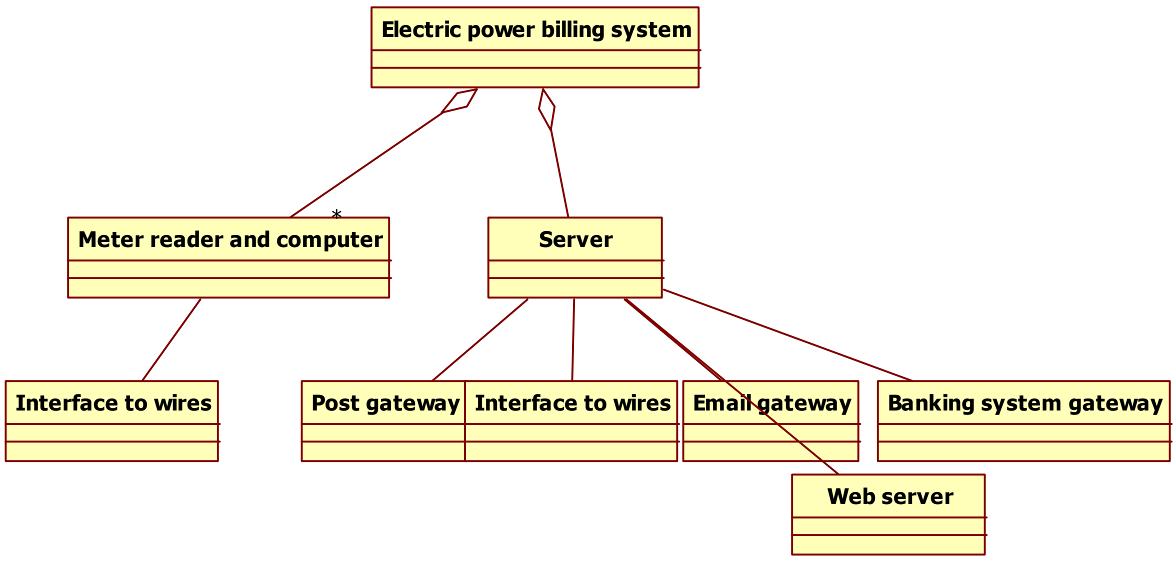
List the requirements in tabular form

ID	Type (Functional Non Functional)	Description
1	F	Read consumption from meter
2	F	Store consumption and date (meter reading)
3	F	Send meter reading to server
4	F	Request and receive meter reading from meter (on server)
5	F	Store meter reading for specific meter (on server)
6	F	Retrieve meter reading for specific meter (on server)
7	F	Compute bill (on server)
8	F	Send bill via postal system (on server)
9	F	Send bill via email (on server)
10	NF	Privacy, data of each customer should not be visible to others
11	NF	Domain Currency is euro
12	NF	Domain Electric consumption in KWh

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



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



Define one scenario describing the computation of a bill for a customer.
Precondition: meter M, x months from last bill passed, all meter readings for meter M for last x months are available

Postcondition: bill for meter M computed and sent

Step	Description	Req ID
1	retrieve all meter readings for meter M for last x months	6
2	Compute consumption in the period	7
3	Apply fee and taxes	7
4	Produce bill	7
5	Send bill, according to consumer preference	8,9

2 (8 points) -Define black box tests for the following function, using equivalence classes and boundary conditions. The function computes if a researcher can participate to a selection to become professor, in function of certain conditions

```
boolean can_participate(int teaching_hours, int different_courses, int NA, int EA)
```

Input: teaching hours: number of hours taught
different_courses: number of different courses where the person has taught
NA: number of articles published
EA: experience acquired, as number of years in doing research or teaching

Output: TRUE if teaching hours > 1500
AND
Different courses >3
AND
NA_normalized >40
(with NA_normalized = NA if EA <= 10, NA_normalized = NA*10/EA if EA >10)
FALSE otherwise

Ex:

can_participate(1600, 4, 60, 11) = TRUE
can_participate(1000, 4, 60, 11) = FALSE
can_participate(1600, 2, 60, 11) = FALSE
can_participate(1700, 7, 30, 5) = FALSE
can_participate(1600, 3, 30, 8) = FALSE

SOLUTION
Each row in the table represents an equivalence class. Total number of equivalence classes = 3x3x3x2x2 = 108. For this reason some combinations are pruned at the first invalid range encountered.

For the same reason boundary test cases are defined only for two combinations of equivalence classes (T4,T5,T6 and T8,T9,T10)

Finally, Na normalized does not have the range [minInt, 0] because it is a derived parameter from EA and NA.

Teaching hours	**Diff courses**	**EA**	**NA**	**NA norm**	**Valid/ Invalid**	**Test cases**
[minInt,0 [[0, 1500] [1501, maxInt]	[minInt, 0] [1,3] [4,maxInt]	[minInt, 0] [1,10] [11,maxInt]	[minInt, 0] [1,maxInt]	[1,40] [41, maxInt]		

[minInt,0 [	-	-	-	-	I	T1→ err
[0, 1500]	[minInt, 0]	-	-	-	I	T2→ err
	[1,3]	[minInt, 0]	-	-	I	T3 → err Boundary: 0 T4 → err T5 →F T6 → F
		[1,10]	[minInt, 0]	-	I	T7 → err Boundary: 0 T8 → err T9 → err T10 → F
			[1,maxInt]	[1,40]	V	T11 → F
				[41, maxInt]	V	T12 → F
		[11,maxInt]	[1,maxInt]	[1,40]	V	T13 → F
				[41, maxInt]	V	T14 → F
	[4,maxInt]	[1,10]	[1,maxInt]	[1,40]	V	T15 → F
				[41, maxInt]	V	T16 → F
		[11,maxInt]		[1,40]	V	T17 → F
				[41, maxInt]	V	T18 → F
[1501, maxInt]	[minInt, 0]	-	-	-	I	T19 → err
	[1,3]	[minInt, 0]	-	-	I	T20 → err
		[1,10]	[minInt, 0]	-	I	T21 → err
			[1,maxInt]	[1,40]	V	T22 → F
				[41, maxInt]	V	T23 → F
		[11,maxInt]	[1,maxInt]	[1,40]	V	T24 → F
				[41, maxInt]	V	T25 → F
	[4,maxInt]	[1,10]	[1,maxInt]	[1,40]	V	T26 → F
				[41, maxInt]	V	T27 → T
		[11,maxInt]	[1,maxInt]	[1,40]	V	T28 → F
				[41, maxInt]	V	T29 → T

Test cases:

boolean can_participate(**int** teaching_hours, **int** different_courses, **int** NA, **int** EA)

T1: can_participate(**-3**, **2**, **30**, **6**)
T2: can_participate(**1000**,**-3**, **30**, **6**)
T3: can_participate(**1000**,**2 6**, **-3**)
T4: can_participate(**1000**,**2**, **6**, **-1**)
T5: can_participate(**1000**,**2**, **6**, **0**)
T6: can_participate(**1000**,**2**, **6**, **1**)
T7: can_participate(**1000**,**2**, **-6**,**1**)
T8: can_participate(**1000**,**2**, **-1**, **1**)
T9: can_participate(**1000**,**2**, **0**, **1**)
T10: can_participate(**1000**,**2**, **1**, **1**)

T11: can_participate(**1000**,**2**, **20**, **1**)
T12: can_participate(**1000**,**2**, **41**, **1**)
T13: can_participate(**1000**,**2**, **41**, **11**)
T14: can_participate(**1000**,**2**, **55**, **11**)

T15: can_participate(**1000**,**4**, **20**, **6**)
T16: can_participate(**1000**,**4**, **45**, **6**)
T17: can_participate(**1000**,**4,45,11**)
T18: can_participate(**1000**,**4,55,11**)

T19: can_participate(**1600**,**-1,55,11**)

T20: can_participate(**1600**,**2, 55, -2**)
T21: can_participate(**1600**,**2, -3, 5**)

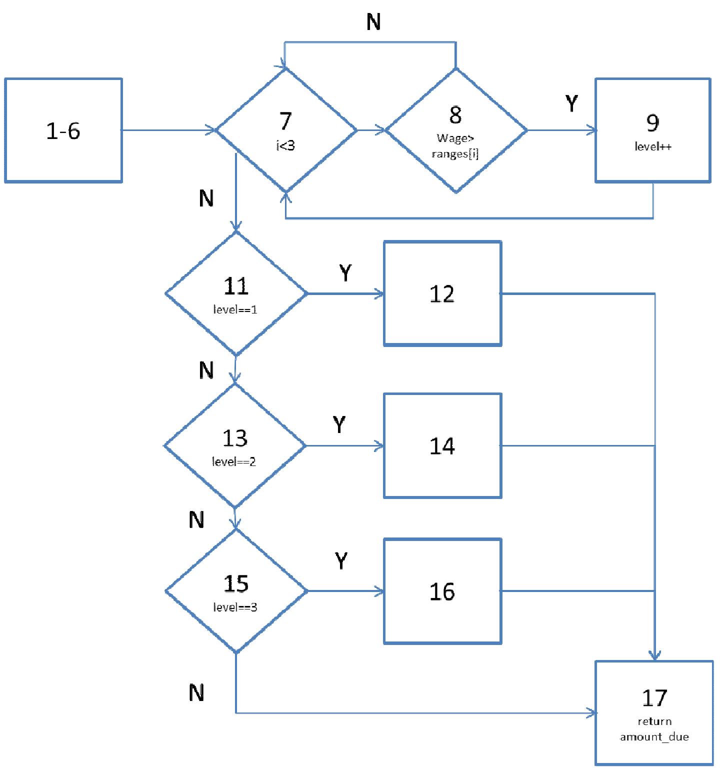
T22: can_participate(**1600**, **2, 30, 5**)
T23: can_participate(**1600**, **2, 41, 5**)
T24: can_participate(**1600**, **2, 41, 11**)
T25: can_participate(**1600**, **2, 55,11**)

T26: can_participate(**1600**, **5, 35, 5**)
T27: can_participate(**1600**, **5, 45, 5**)

T28: can_participate(**1600**, **5, 35, 12**)
T29: can_participate(**1600**, **5, 55, 11**)

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**.

```
1       private int compute_tax(int wage) {
2
3               int ranges[] = {5000,15000,30000};
4               int amount_due = 0;
5               int level=0;
6
7               for(int i=0;i<3;i++){
8                       if(wage>ranges[i])
9                               level++;
10              }
11              if(level==1)
12                      amount_due = 500;
13              else if(level == 2)
14                      amount_due = 1500;
15              else if(level == 3)
16                      amount_due = 3000;
17              return amount_due;
18      }
```



Coverage type	Feasibility (Y/N)	Coverage obtained	Test cases

		(%)	
Node	Y	100 %	T1, T2, T3
Edge	Y	100 %	T0, T1, T2, T3
Multiple condition	N	-	-
Loop	Y	33% (1/3)	T0
Path	Y	100% (4/4)	T0, T1, T2, T3

The following test cases cover all edges of lines 11-16 (the nested ifs):
T0: `compute_tax(1000)`
T1: `compute_tax(6000)`
T2: `compute_tax(16000)`
T3: `compute_tax(32000)`

Theoretically the paths are 32: 2^3 (if inside the for) x 4 (if-else chain) = 8x4=32
However, only 4 are feasible, i.e. the ones corresponding to the four test cases above.
That is because the paths depend on the value of parameter ‘wage’:
$wage<5000$
$5000< wage \leq 10000$
$10000 < wage \leq 15000$
$15000 < wage \leq 30000$
$wage > 30000$

So with the same tests it is also possible to cover all paths (4).

The loop coverage is possible only at 33% because the loop is always executed 3 times.
Finally, there are no multiple conditions, so it is infeasible.

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

Effort is the time taken by the staff to complete a task.
It is measured in person hours (std ieee 1045),
and also person day, person month, person year depend on national and corporation parameters

5 (1 points) – What are the possible types of defects in a requirement document?

Omission/ incompleteness
Incorrect Fact
Inconsistency/contradiction
Ambiguity
Extraneous Information
Overspecification (design)
Redundancy

6 (1 points) – What is a Configuration?

A configuration is a set of Configuration Items (CI), each in a specific version, and it can be seen as a snapshot of the software at certain time. Some CIs may appear in different configurations, and also configuration has a version number