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

June 27th, 2018

Last Name
First Name
Id number (Matricola)

Note

1. The exam is not valid if you do not fill in the above data.
2. Write your answers on these pages. Extra sheets will be ignored. You may use a pencil.
3. Incomprehensible hand-writing is equivalent to not providing an answer.
4. The use of any electronic apparatus (computer, cell phone, camera, etc.) is strictly forbidden.
5. You cannot keep a copy of the exam when you leave the room.
6. The exam is composed of three exercises. Read carefully all points in the text!

Scores of each exercise:

Exercise 1 (MAX 8) _____

Exercise 2 (MAX 3) _____

Exercise 3 (MAX 8) _____

Question 1 Alloy (8 points)

The following text describes, in a simplified way, the procedures that are put in place to manage applications to the PhD positions offered by Politecnico di Milano and the subsequent evaluation of candidates.

There are various PhD curricula (Industrial Chemistry and Chemical Engineering, Information Technology, …).

Candidates apply to exactly one specific curriculum. They can select one or more specific topics within this curriculum (they can also decide not to choose a specific topic).

For each curriculum, an evaluation committee is appointed. At the beginning, the only action the committee can perform is to accept/reject the graduate title of candidates. When this operation is completed, the committee can start assigning a score (which can be A, B, or C) to each candidate whose title has been accepted (the other candidates are excluded).

Consider the following Alloy signatures:

```
abstract sig Boolean {}
one sig True extends Boolean {}
one sig False extends Boolean {}

sig Curriculum {}
sig Topic {}

sig Committee {
   curriculum: Curriculum,
   titleRecognitionCompleted: Boolean   //if this is True it means that the committee has
                                        //completed the titles acceptance/rejection phase of
                                        //all candidates
}
```

Provide an answer to the following points:

A) Define the signature representing a candidate. Besides the elements that you will consider important, based on the text above, this signature should include the following two relations:

```
   masterTRecPending: Boolean,
   masterTAccepted: Boolean,
```

masterTRecPending is True for a candidate if the committee has not evaluated his/her master title, yet, False otherwise. Instead, masterTAccepted is True for a candidate if his/her master title has been evaluated to be acceptable, False if it has been evaluated to be unacceptable or if it is still to be evaluated.

B) Define the function listOfCandidates that, given a certain committee, returns the list of candidates applying to the corresponding curriculum.

C) Define a fact ensuring that, for each committee, titleRecognitionCompleted is True when all candidates of the corresponding curriculum have their title evaluated (either positively or negatively).

D) Define any other fact concerning the score assignment by the committee that you feel is needed to ensure the adherence of the model to the text above.

Solution

```
abstract sig Boolean {}
one sig True extends Boolean {}
one sig False extends Boolean {}
sig Curriculum {}
sig Topic {}

sig Committee {
   curriculum: Curriculum,
   titleRecognitionCompleted: Boolean
}

abstract sig Score {}
one sig A extends Score {}
one sig B extends Score {}
one sig C extends Score {}

sig Candidate {
   curric: Curriculum,
   topics: set Topic,
   masterTRecPending: Boolean,
   masterTAccepted: Boolean,
   score: lone Score
} {masterTAccepted = True => masterTRecPending = False}
// when masterTAccepted = False and masterTRecPending = False implies title
// is rejected by the committee

fun listOfCandidates[comm: Committee]: set Candidate {
   comm.curriculum.(~curric)
}

fact masterTitleRecognitionCompleted {
    all comm: Committee | comm.titleRecognitionCompleted = True <=>
      all c: Candidate | c in listOfCandidates[comm] implies c.masterTRecPending = False
}
```

```
fact scoreDefinition {
    all c: Candidate | c.score != none <=>
       (one comm: Committee |
            c.curric = comm.curriculum and comm. titleRecognitionCompleted = True
            and c. masterTAccepted = True)
}

pred show[]{
  #Committee > 1
}

run show for 5 but 3 Committee
```

Question 2 Function points (3 points)
A web site allows users to post articles and to comment on articles inserted by other users. In particular, a user should be able to:

1) Register to the system
2) Login into the system
3) Post a new article
4) Delete a previously inserted article
5) Browse the list of all articles
6) Browse the list of articles inserted by a given user
7) Read an article and its corresponding comments
8) Write a comment for an article
9) Delete a previously inserted comment

You are asked to identify and compute the Function Points for the case described above, focusing on **Internal Logic Files**, **External Inquiries** and **External Outputs**. Please provide an explanation for each point you consider and for the associated complexity. Refer to the following table to associate weights to the function types:

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

Solution

Internal Logic Files: these are all data handled by the software system we are considering: Users, Articles, Comments. Their data structure is a simple one, so their complexity depends essentially on the number of Users, Articles and Comments in the systems and can be considered from simple to medium. Therefore, the number of Function Points for Internal Logic Files can vary between 21 (3*7) and 30 (3*10).
External Outputs: there is no External Output in the description, so the Function Points count in this case is 0.

External Inquiries:
Browse the list of all articles: its complexity depends on the complexity of Article.
Browse the list of articles inserted by a given user: its complexity depends on the complexity of Article and Users.
Read an article and its corresponding comments: its complexity depends on the complexity of Article and Comments.
Therefore, the number of Function Points for External Inquiries varies between 9 (3*3) and 12 (3*4).

There are no External Interface Files as the system appears to be completely self-contained.
Most of the functionalities listed above can be considered as External Inputs.

Question 3: Design and availability estimation (8 points)
Consider a software system that acquires and elaborates information from a number of different sources by polling them periodically. The two component diagrams below provide an overview of the types of components defined for this system, with two possible types of interaction between some of them.

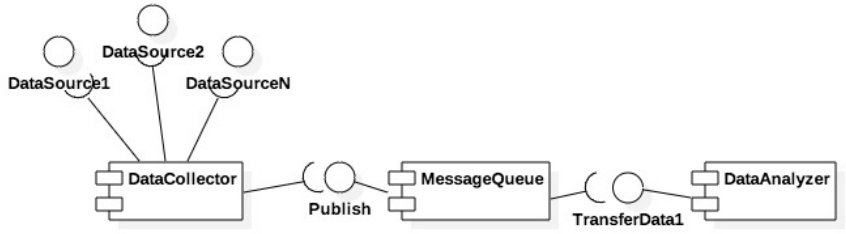


Figure 1. Component Diagram 1.

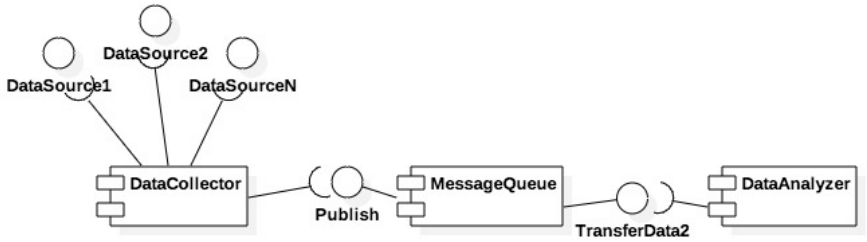


Figure 2. Component Diagram 2.

A) Describe the system in the two versions, focusing on the possible interactions between the various components, based on the information you can derive from the component diagrams.

B) With reference to the component diagram in Figure 1, define a sequence diagram that describes the way data flow through the system. How would this sequence diagram change in the case of the component diagram of Figure 2?
NB: You do not need to redraw the whole diagram, a sketch or a short explanation will be sufficient.

C) Assume that the components of your system offer the following availability:
DataCollector: 99%
MessageQueue: 99.99%
DataAnalyzer: 99.5%
Provide an estimation of the total availability of your system (you can provide a raw estimation of the availability without computing it completely).
Assuming that you wanted to improve this total availability by exploiting replication, which component(s) would you replicate? Please provide an argument for your answer.
How would such replication impact on the way the system works and is designed?

Solution
Component diagram of Figure 1: The DataCollector component is exploiting the interfaces offered by some data sources to acquire data and the interface of the MessageQueue to pass collected data to the other components. Based on the way it interacts with the other elements, we can assume that the DataCollector is continuously or periodically polling the sources and is transferring the received data to the MessageQueue. We do not know if such transfer occurs in batches or not. The MessageQueue exploits the interface offered by the DataAnalyzer to pass the data to this component. Once again, we have no

information on whether this transfer occurs in batches. We could derive such information from an analysis of the interfaces offered by MessageQueue and DataAnalyzer, respectively.
Component diagram of Figure 2: In this case, the MessageQueue does not actively push the data to the DataAnalyzer, but it offers interface TransferData2 so that the DataAnalyzer can pull data as soon as it is ready to process them. Also in this case, both a batch or a per data approach is possible. The rest of the system behaves as already described above.

Sequence diagrams

Diagram compatible with component diagram of Fig 1

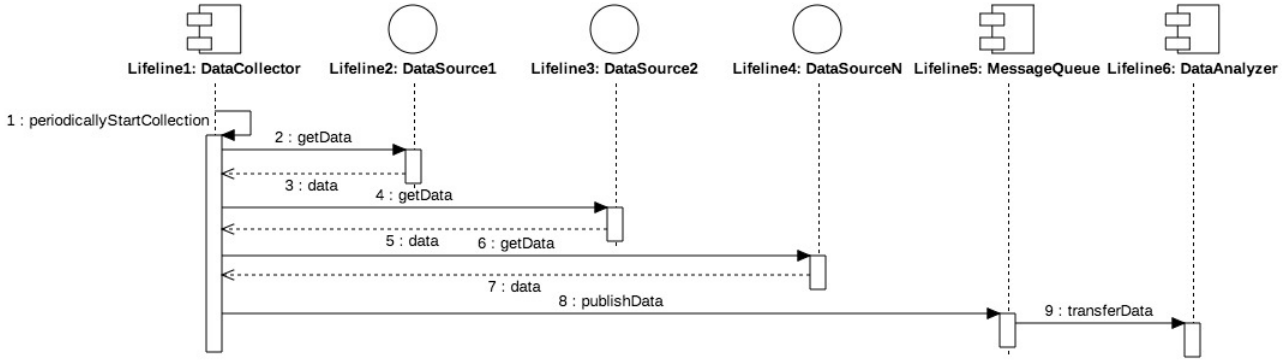
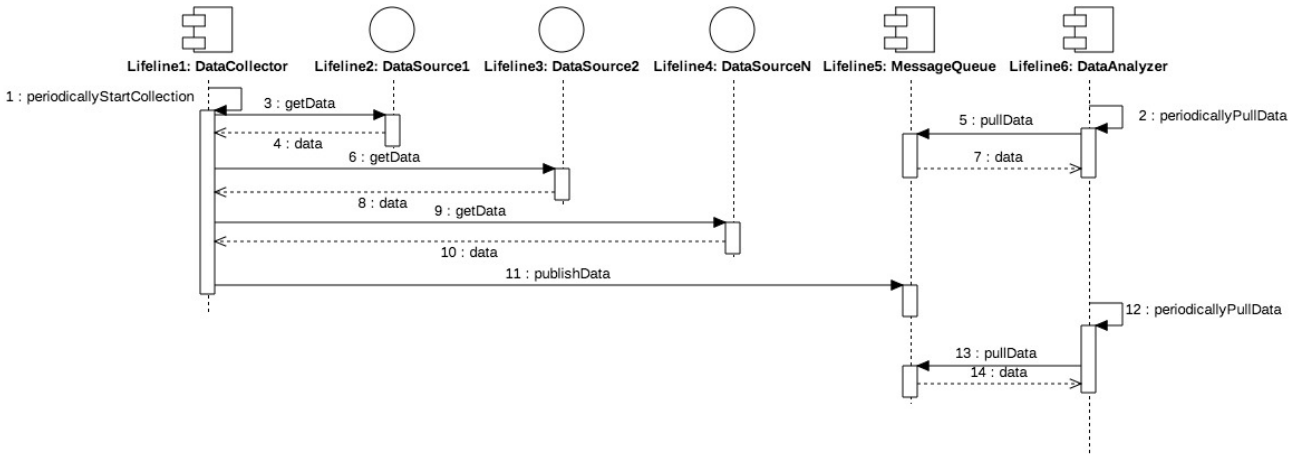


Diagram compatible with component diagram of Fig 2



Availability
Note that, regardless of the way the interaction between the MessageQueue and the DataAnalyzer works, data have to flow through the whole chain of components to be processed. This implies that we can model the system as a series of component.

The total availability of the system is determined by the weakest element, that is, the DataCollector.
$A_{Total} = 0.99*0.9999*0.995 = 0.985$

If we parallelize the data collector adding a new replica, we can achieve the following availability:

$(1-(1-0.99)^2) *0.9999*0.995 = 0.995$

At this point, even if we increase the number of DataCollector replica, we do not achieve an improvement as the weakest component becomes the DataAnalyzer. We can parallelize this component as well to further improve the availability of our system.

Let's consider the impact of the DataCollector parallelization on the rest of the system. If both replicas acquire information from the same sources in order to guarantee that all data are offered to the rest of the system, then the other components will see all data duplicated and will have to be developed considering this situation. For instance, the MessageQueue could discard all duplicates. Another aspect to be considered is that both DataSources and MessageQueue have to implement mutual exclusion mechanisms that ensure the communication between them and the two DataCollector replicas does not raise concurrency issues. Another option could be that only one DataCollector replica at a time is available and the other is activated only when needed (for instance, if the first one does not send feedback within a certain timeout).