Section 3.1:

1. In order to coordinate by messages agents needs to have a common protocol and language so that they can understand each other's and interact easily. For instance, one of the ways to communicate in JaCaMo is through .send(Receiver, Performative, Content). Does the agents in our system have a common language?

2. There are many differences when we talk about "Interaction-Centric Coordination" and "Agent-Centric Coordination". The agent centric approach is more focused on the idea that the agent should be able to achieve his task only using what he believes on (most of the time). While as the Interaction-Centric approach focuses on the idea that agents can communicate with each other's and share more information.
   None of the two approaches is better than the other (it depends on the use case we have). But the main **Pros** of using the "Interaction-Centric coordination" over the Agent-centric is: in this approach the communication between the agents makes everyone of them know more about the system state and can lead to more synchronization between the agents. However, this approach also have some **Cons.** For instance, the management of this interaction takes extra work and it could get complicated as the system grows up. Also, we need to develop a common language for all the agents to use in their communications.

3. The general protocols and ontologies that should be used by the agents to coordinate by messages is missing. For example, we don't know if the agents need to agree on something first (System prerequisites) before the system starts and if they can coordination properly, or if they can understand the languages of each others…

Section 3.2:

Q3-4-5-6:

- After performing Q3, we can see that delay(plumbing,10)[source(agenttest)]) was added as a new belief to the Belief base of giacomo agent.

The "tell" performative translates the message sent as a belief in the reciever agent (giacomo in this case) belief base.

- After performing Q4, a new goal updateTolerance(10) was sent to giacomo.
  The "achieve" performative translates the sent message as a goal for giacomo.

Deeper justification:

When agenttest executes .send(giacomo, achieve, updateTolerance(10)). We can see the following result in the console:

```
[giacomo] No fail event was generated for +!updateTolerance(10)[source(agenttest)]
```

[giacomo] Found a goal for which there is no applicable plan: +!updateTolerance(10)[source(agenttest)]

The two lines indicates that the plan (in clientQ1a.asl):

```
+!updateTolerance(V) :
   delay_limit(L) & L > V
          <-
          .print("update tolerance");
          -+delay_limit(L-V).
```

was executed (and failed at L > V because "10 > 10" is false) and in our code we don't have a fail event for this plan. (if we try a value smaller than 10, and the plan will be executed successfully and the agent's tolerance will be updated).

And as we know, this plan only gets executed if a goal !updateTolerance(10) triggers it.

NOTE: After this part I updated the code so that +!upadeTolerance plan doesn't fail again.

- After performing Q5, we can see that penalty(plumbing,50)[source(giacomo)]. was added to agenttest belief base. As we can indicate, this belief was been copied from giacomo who already had it at the first place.

"askOne" performative was used by agenttest to ask a specific agent (giacomo) for a certain value. And in our case giacomo had this value in his belief base so he automatically sent it back to agenttest. But if we look into the documentation at http://jason.sourceforge.net/doc/api/jason/stdlib/send.html we can also have more information: if the receiver agent (giacomo) didn't have this information in his belief base directly, then an event like +?penalty(X,Y) would have been created in giacomo's side and the result of this query will be sent to agenttest.

- Conclusion:

"tell" can be used to add a new **belief** to the receiver agent belief base.

"achieve" can be used to add a new **goal** for the receiver agent.

"askOne" can be used to ask some other agent for a certain value if he is known to have it directly in his belief base, or if he could be able to compute it and send it.

Q8 -Q9: (Enhanced Approach – My Solution)

I enhanced the used approach in the problem:

Instead of letting the company ask the client, the client respond to the company and then the company send the delay(D) to the agent as a belief (tell) and then the agent updates his delay_limit…

I used the following approach:

First the company asks the client (using askOne), the client responds and then if the delay time is compatible, the company will send an “achieve” message to the client, so that the value of delay_limit in his belief base changes directly for him (after achieving the sent goal to updateTolerance).

Enhancement: reduced a step, changed only companyA code, same result obtained.

Section 4.1:

1.a) In such system the agents can communicate properly using a known protocol. Thus, if we take into consideration the autonomy of the agents. We can say that our Client agents in the system are **Data-directed,** they can receive data and select them (they have observation autonomy). The client agents can receive information from the company and decide if they accept the delay of not (they can decide). While as for the company agents they might have less autonomy, as they now can only receive what they should do from the clients (their work depends on the client accepting or rejecting the delay). So we might say that the company agents are **Goal-directed,** they receive goals from the clients and try to achieve them. (Pros: the new protocol made the clients more autonomous. Cons: it made the companies less autonomous and more dependent on the clients).

1.b) Considering the openness of the system as a new factor, now the clients could have more companies to work with, and similarly the companies could have more clients to work with.
Pros: This could be good for both sides as the clients can have more companies that could finish their work faster now. And similarly for the companies they might be able to increase their profit through working with several clients.
Cons: On the other side when the companies have many clients now they will need more time to finish their work and thus will ask for more delays from the clients. And this could be bad for the clients if they can’t take this delay. Also, it could be bad for the companies if they started to lose the trust of their clients.