Wei Zhenfang, Yan Yi
TSEDM

Homework 1

Modern Methods in Software Engineering

Question 1: What is the difference between an abstract class and an interface?

There are great differences between an abstract class and an interface:

All methods in an interface are abstract, but abstract class can define kinds of different methods with methodbody.

One class can implement multiple interfaces, but can inherit only one abstract father class.

An interface have no inheritance relationship with the class that implements this interface, therefore several irrelevant classes can implement the same interface; however, abstract class belongs to class inheritance framework and is always on top floor of the framework.

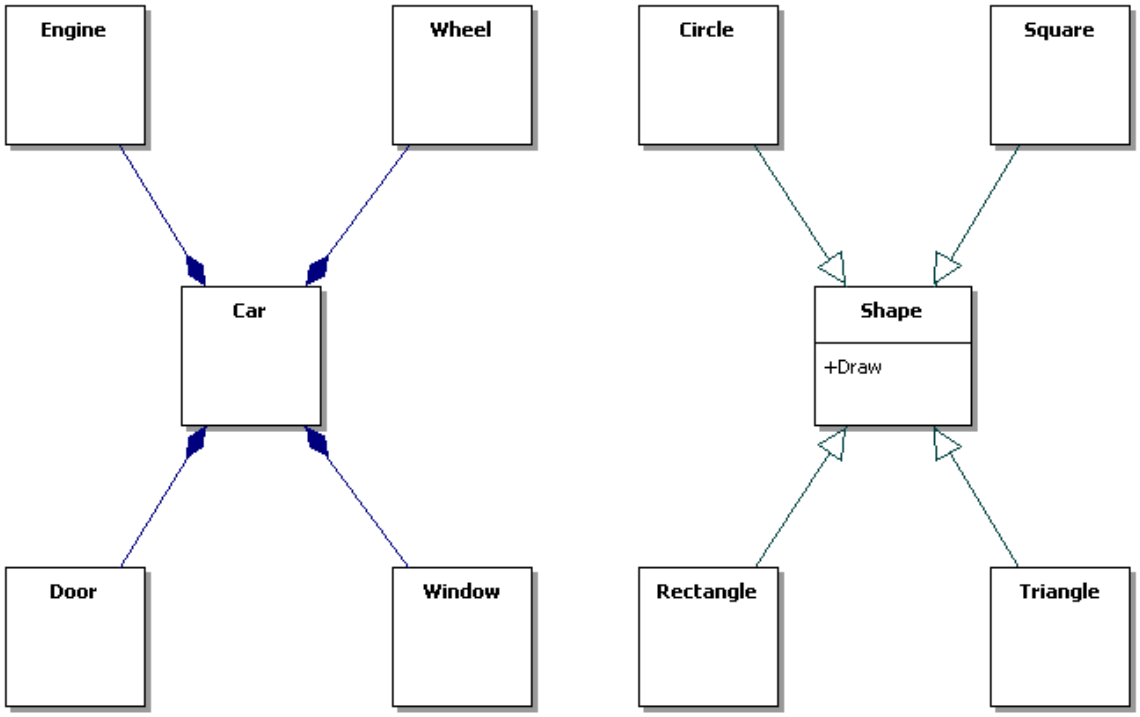
Question 2: i). Give an account of advantages and disadvantages of inheritance and composition. **ii).** Give an example of when inheritance is better and when composition is better

Advantages:

Composition	**Inheritance**
Easier to change the interface of both composite class and the content classes.	Makes the code easier to change if the needed change involves adding a new subclass, dynamic binding and polymorphism can be used.
Allow us to delay the creation of content objects until they are needed	Easy to build and describe Is-A relationship

Disadvantages:

Composition	**Inheritance**
Composite class cannot use dynamic binding and polymorphism	Super class's interface changing will cause many modifications
	Encapsulation is fragile



On the left, we want to describe a car system with engine, wheels, doors, etc. We can use composition rather than inheritance to explain and describe the relationship. On the right, it's better using the inheritance relation to build and implement the system because it shows several obvious "Is-A" relationships and we can easily using the polymorphism here.

Question 3: i). What are some of their advantages and disadvantages over each other? **ii).** Describe development circumstances under which you would prefer to follow waterfall method of development, instead of spiral model?

Advantages:

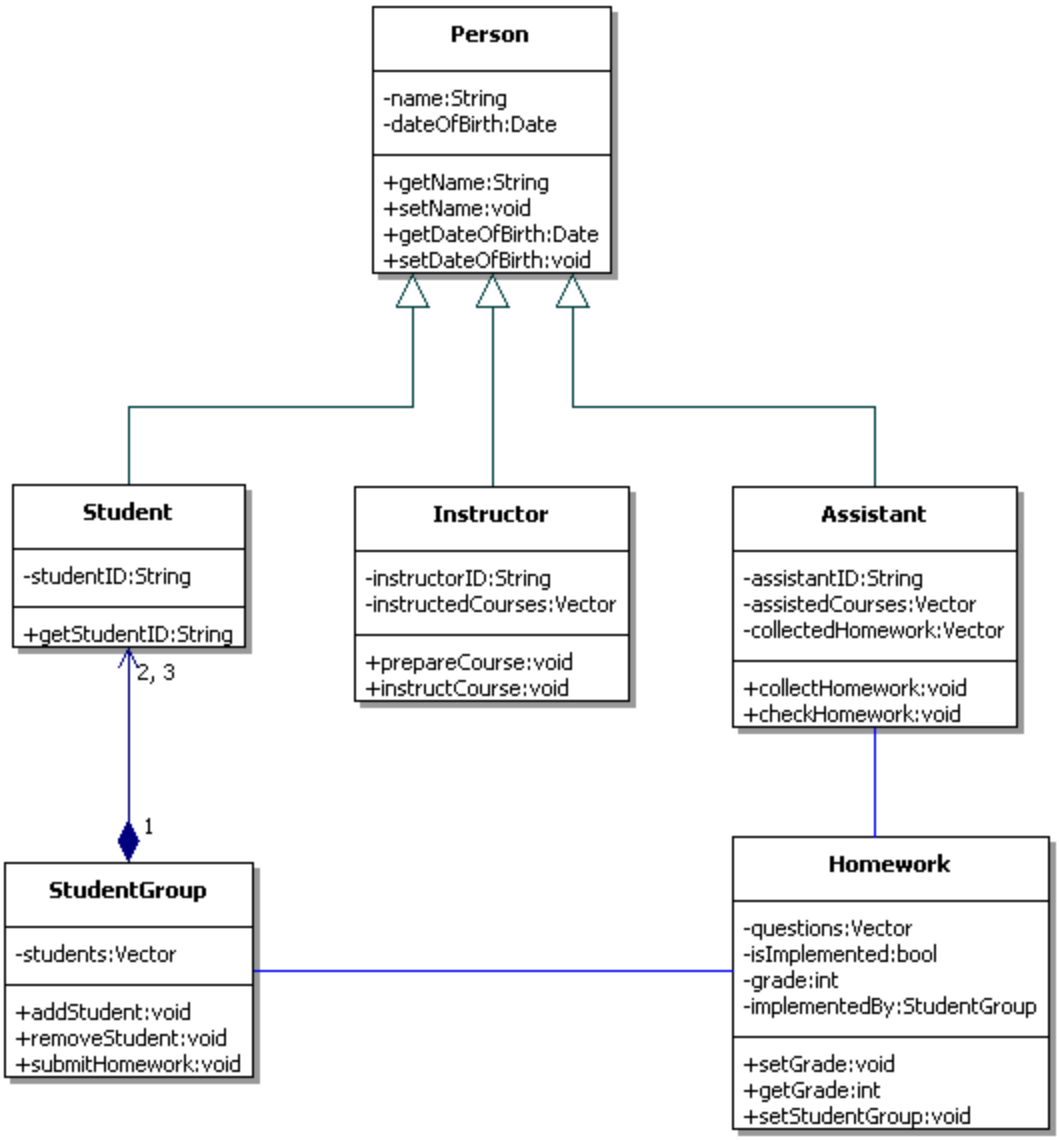
Waterfall Model	**Spiral Model**
the model itself progresses linearly through discrete, easily understandable and explainable "phases" and is thus easy to understand	The important issues can be discovered in earlier stage. As work progresses, estimation becomes more realistic
No need to look back linear system, one activity at a time	Better to deal with nearly evitable changes that software development generally has to undergo
Easy to check progress	Software engineers can start working on project earlier
Provide easily clear milestones in the development process	

Disadvantages:

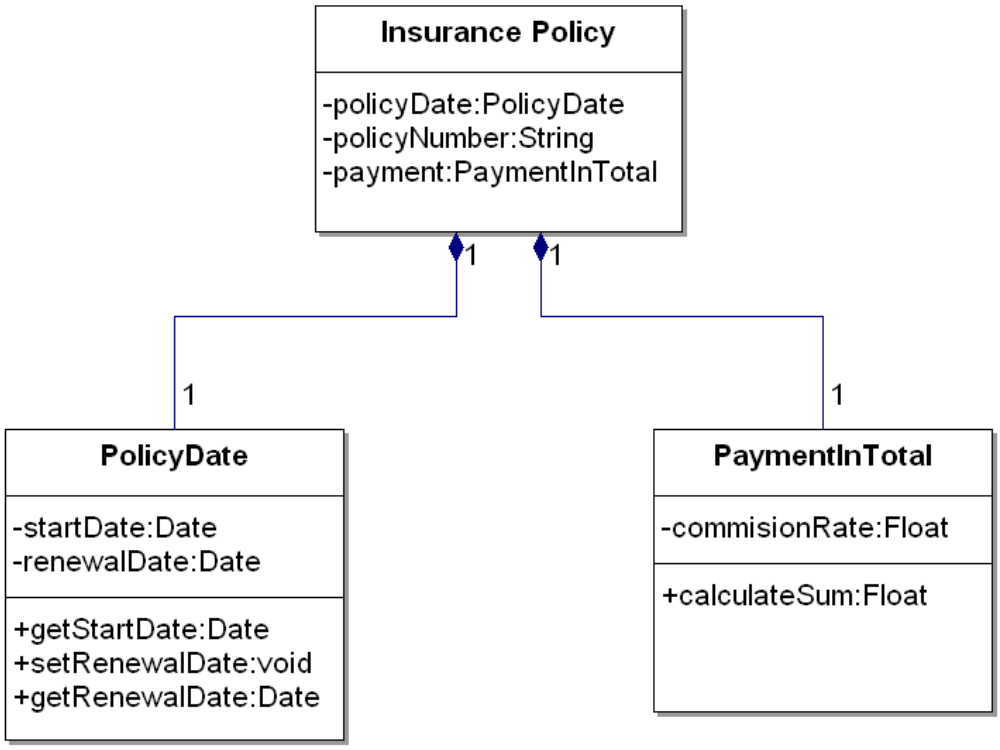
Waterfall Model	Spiral Model
Working with poorly understood requirements	Require highly skilled people in the area of planning, risk analysis, development, customer relation etc.
Poor ability to deal with constantly, evitable changes that clients make after a design is finished	Demand more time and budget as the process needs to be iterated more than once
	Change is not allowed once inside a phase

ii) I would prefer waterfall model of development, instead of spiral model under such development circumstances: 1) short of highly skilled people, have not much budget and time. Because the spiral model is more intended for large, expensive, and complicated projects comparing to waterfall model, the latter would be considered first; 2) developing the software project whose requirements are unchanging or in which designers will predict all the problem areas of the system that could probably occur and make a correct design before starting implementation. In this case, both of the two models are good choices without bothersome ceaseless changes. But we would prefer waterfall model because of its simpleness, clear procedures and linear processing by easily understandable phases and clear milestones.

Question 4: Draw a class diagram for the case of our course. We assume that there are at least the following classes: Person, Student, Instructor and Assistant (you can add as many additional classes as you would like). You should draw associations between them, possible attributes and operations.



Question 5: Prepare a class diagram for the following case:
"For each insurance policy, it will be necessary to know the policy number, the start date of the policy, when it is due for renewal and the commission rate due to the premium for that policy. The start date is when the policy was first take out. The renewal date is when the policy next expires. We also need to know how much money has been paid in total as payments against the policy (in the current year of the policy). The current year of the policy is defined as the renewal date minus one."
Identify any attributes and operations here. Allocate them to appropriate classes. Suggest suitable types for the attributes, operations, and any parameters for the operations. Add all these to your class diagram showing the visibility of both attributes and operations.



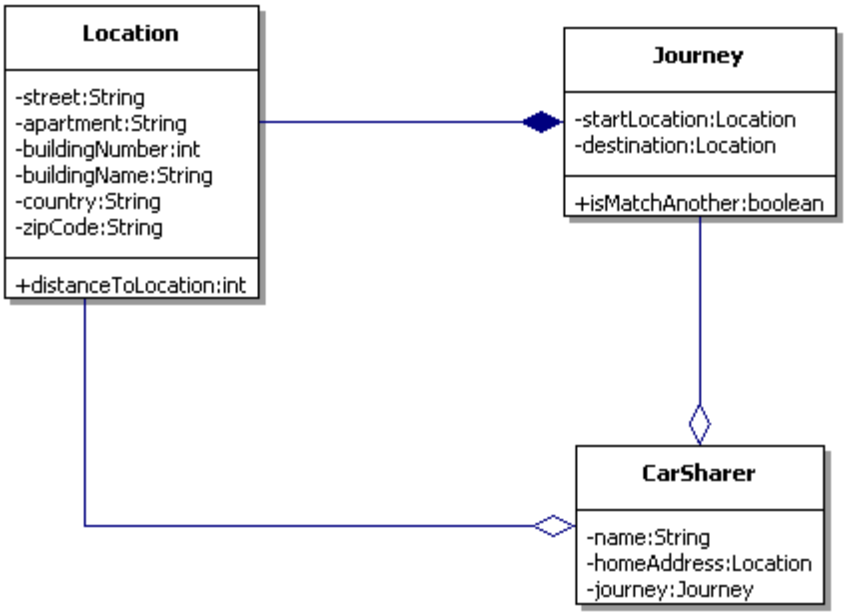
Question 6: Draw a class diagram for a part of a Car Sharing System, based on a personal interview transcript. Identify any aggregation associations.

Andres: *Just remind me, what kind of things do you need to know about the start and the destination of each journey?*

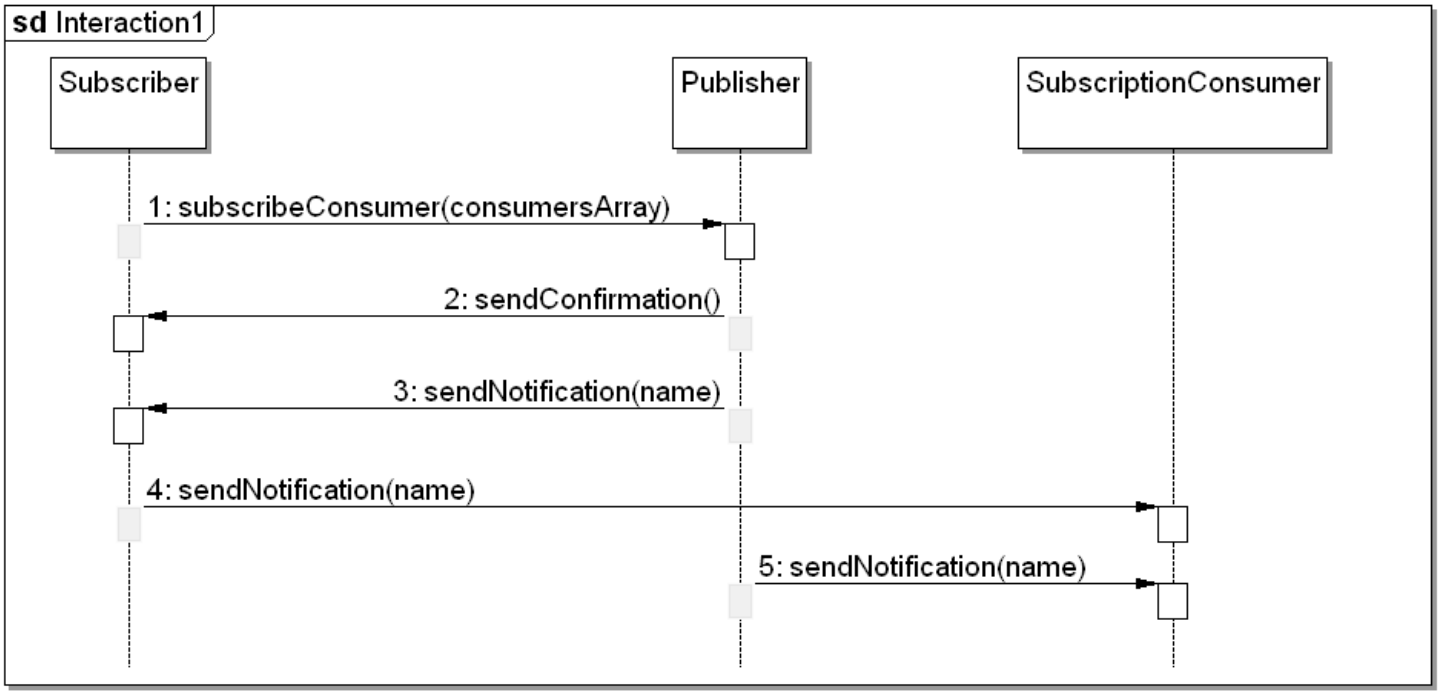
Mihhail: *We'd want to know the building name and number, the apartment number, the street, town or city, county and postal code or zip code. We'd also want to hold similar information for the home adress of the sharer as well.*

Andres: *OK. Didn't you say that the journey start and destination address will be used to match up possible shared journeys?*

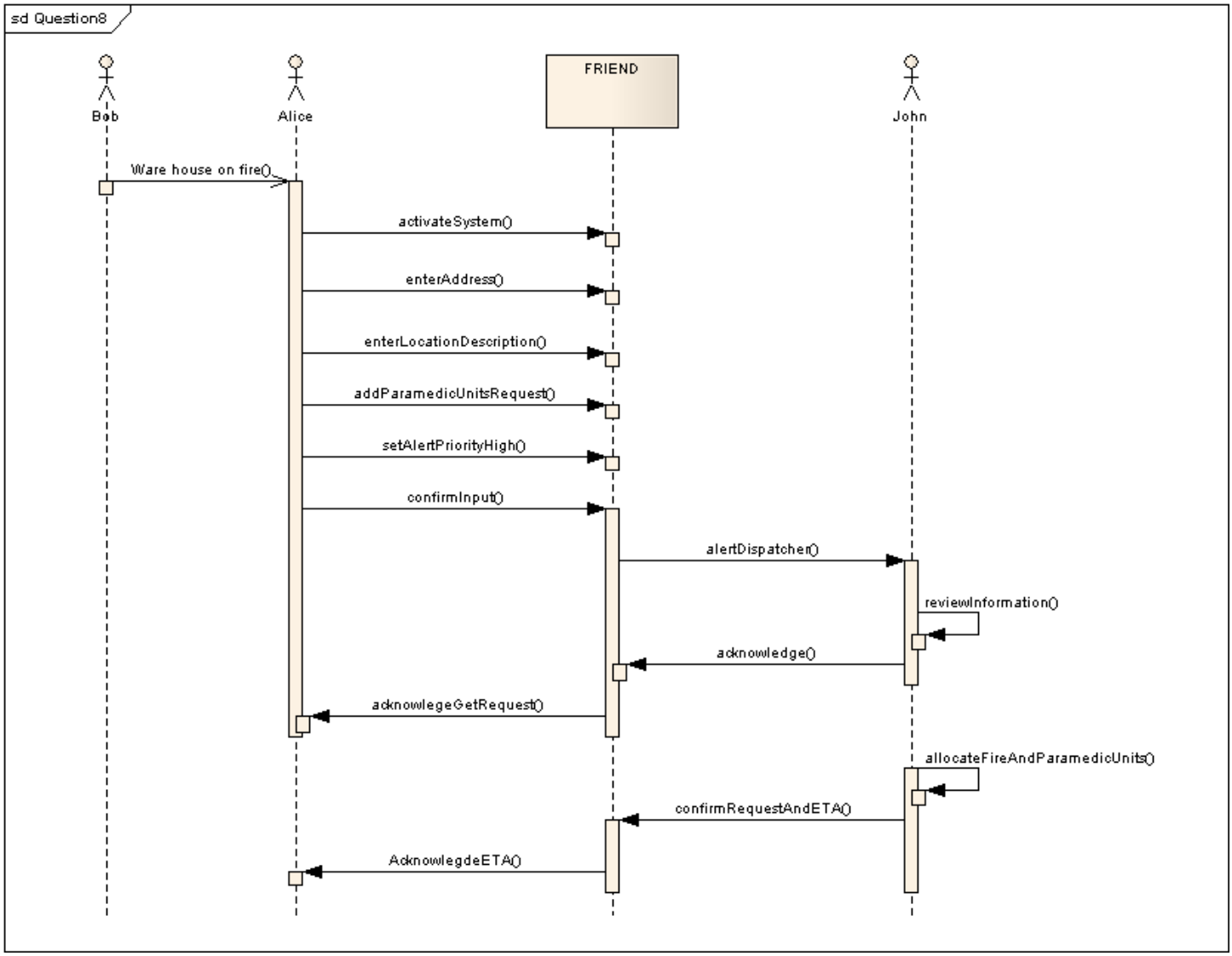
Mihhail: *Yes -interesting point that. I'm not quite sure how you'll do this. We want to be able to establish whether two addresses are close enough to each other to be able to consider them a match for a shared journey. For example, two people may want to get from a start destination on adjacent corners of two different blocks to destination addresses in different floors of the same building. A person looking at the addresses would know that the addresses are similar enough to be a match, but in terms of just text of the addresses, they look completely different.*



Question 7: Draw the Publish/Subscribe protocol as sequence diagrams (publisher, subscriber and subscription consumer).
"Subscriber subscribes consumer at a publisher to receive notification message and publisher sends confirmation to subscriber and notification to consumer (who may be the same or different)."

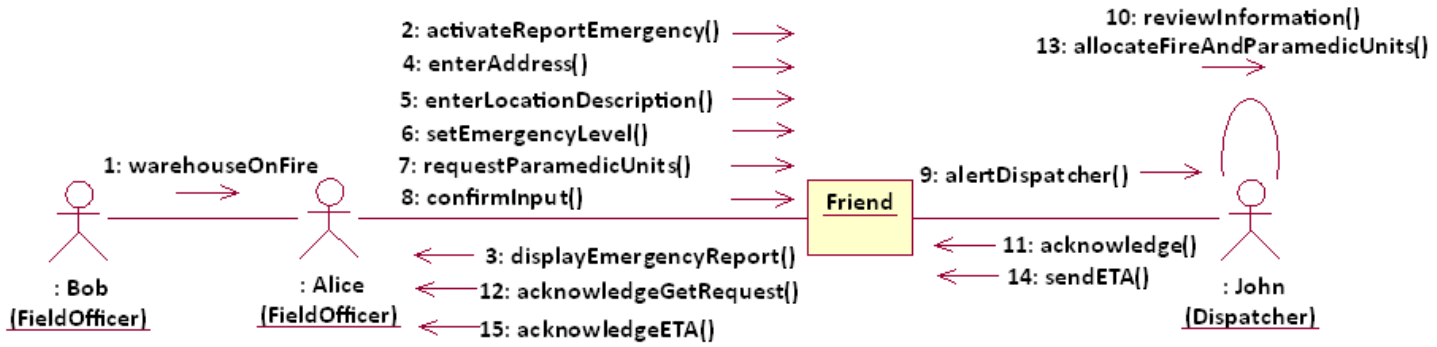


Question 8: Draw a sequence diagram of scenario “WarehouseOnFire” given in figure 2-15 in Book. Include Objects bob, Alice, john, FRIEND and instances of other classes you may need.



Question 9: i). Draw a Collaboration diagram of the similar scenario as of Question 15. **ii).** Analyze which type of Interaction diagram you would prefer. **iii).** If you are given a scenario with numerous concurrent flows of control then which Interaction diagram will you opt for.

i)



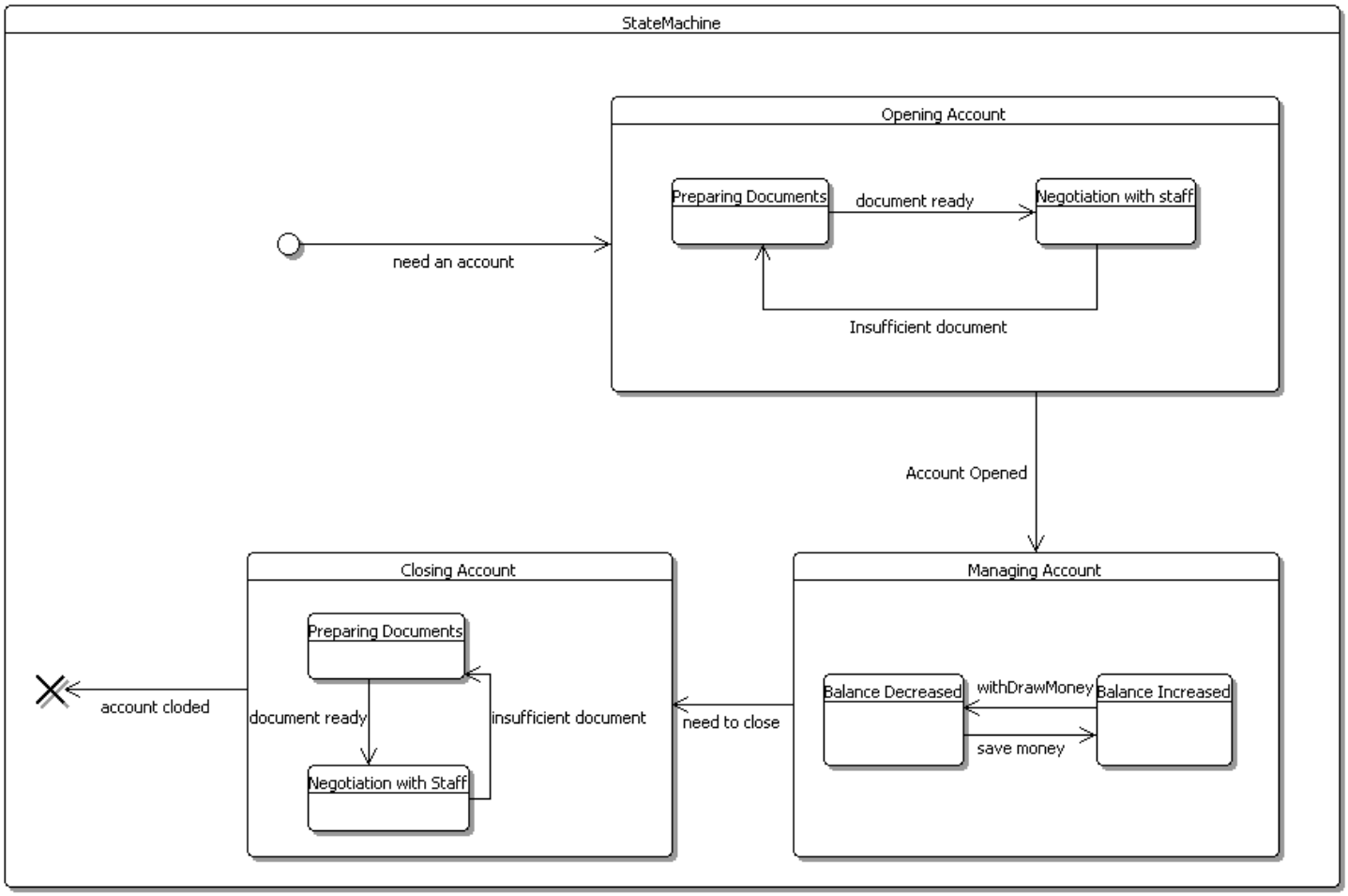
ii) I would prefer Sequence diagrams. Sequence diagrams contain the same information as Collaboration diagrams, but emphasize the order in which the message is called instead of the relationships between the

objects. Besides, Sequence diagrams take up a bit more space, but are much easier to read and follow algorithmically; Collaboration diagram shows the whole process in one dense diagram, but obscures the algorithm to some extent. Like mentioned above, I would prefer Sequence diagrams from a software developer's point of view.

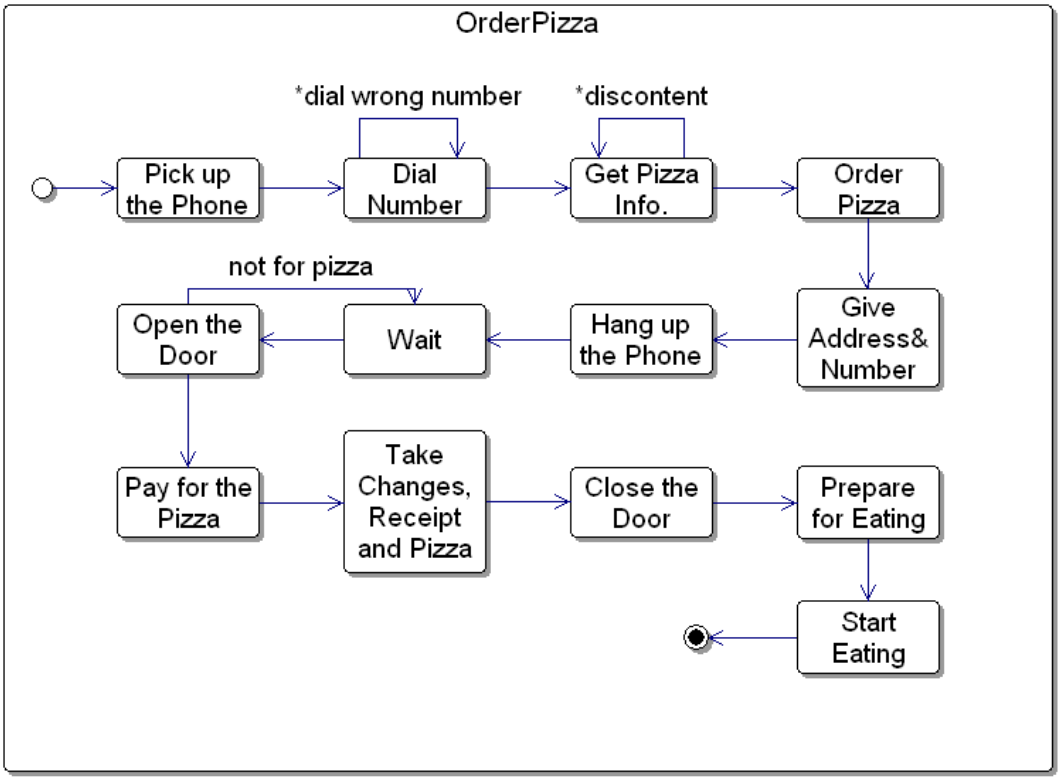
iii) I will opt for Collaboration diagrams. Sequence diagrams emphasize the order in which things happen and only allow us to show simple branching. In contrast, Collaboration diagrams allow us to represent more complex branching like multiple concurrent flows of control.

Question 10: Draw an activity:
"Opening account, deposit/withdraw iterations (with conditions checking) and closing account" as state chart diagram(s).

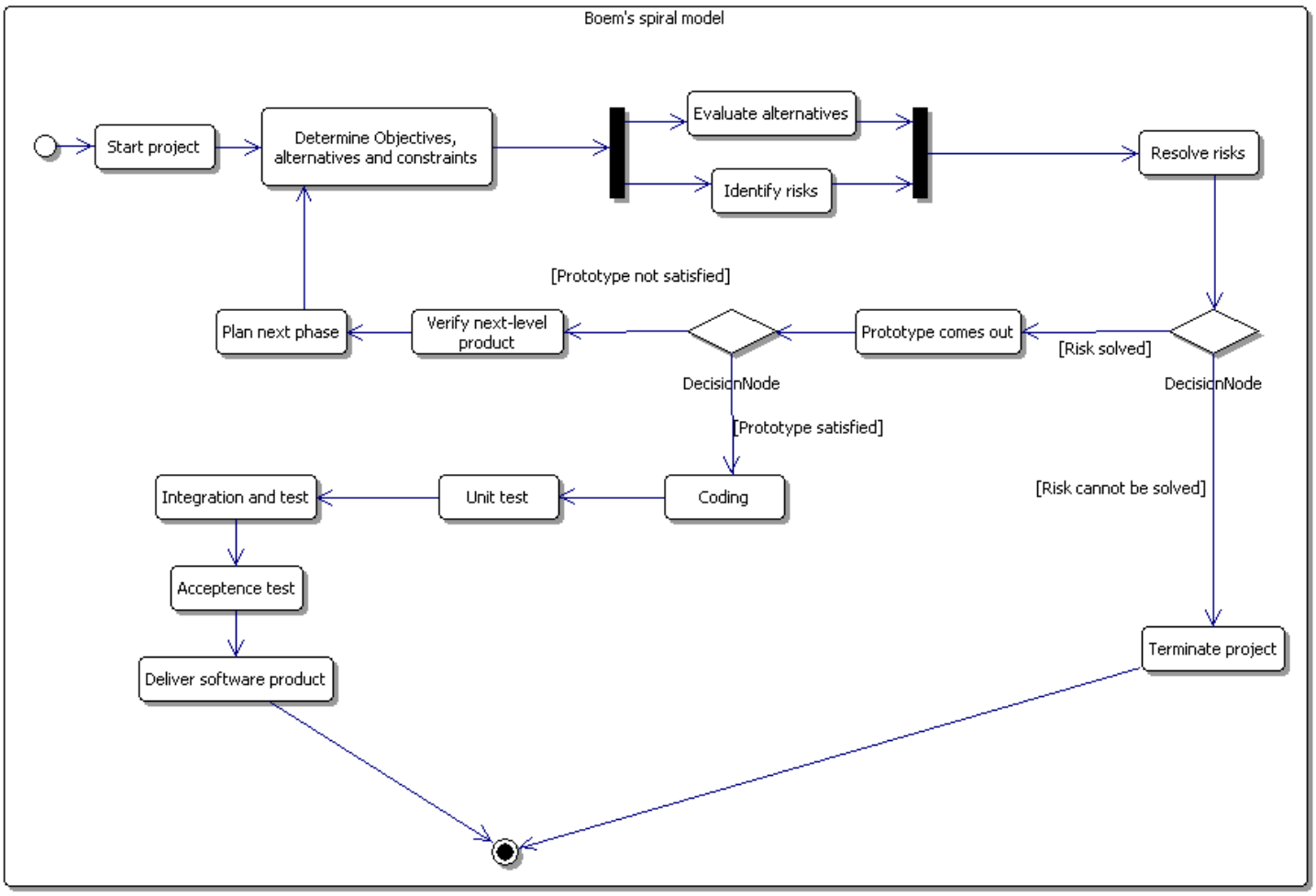
Assume the person need to open an account in a bank and keep the account for days. After he/she leaves the country, he/she has to close the account. During the opening and closing the account, he/she has to prepare for the required documents. During the managing of the account, he/she can withdraw money from the account and save money into the account.



Question 11: Consider the process of ordering a pizza over the phone. Draw an activity diagram representing each step of the process, from the moment you pick up the phone to the point where you start eating the pizza.



Question 12: Draw activity diagram of Boehm's Spiral Model in Figure 15-10. (01 Bonus Point)



Question 13: Give a scenario in which you would prefer to use Entity-Centred models over Activity-Centred Models. (01 Bonus Point)

In a re-engineering or interface project, we need to make a change to the interface of an existing system or replace software of an existing business process. In such cases, the use case and object models are available in the previous project. If the project has a long history, the rationale should be well populated. So what we only need to do is making a different set of activities rather than building a system from scratch. Then we can select an entity-centered model instead of an activity-centered model to make all product-creating works concurrently in order to shorten the development period if a tight deadline is given. Of course, it will be better for using entity-centered model if with experienced developers and good CASE tool support.