Information for the exam and some questions answered that were asked in class

The test consists of one case study description, followed by questions that require you to draw the different models.

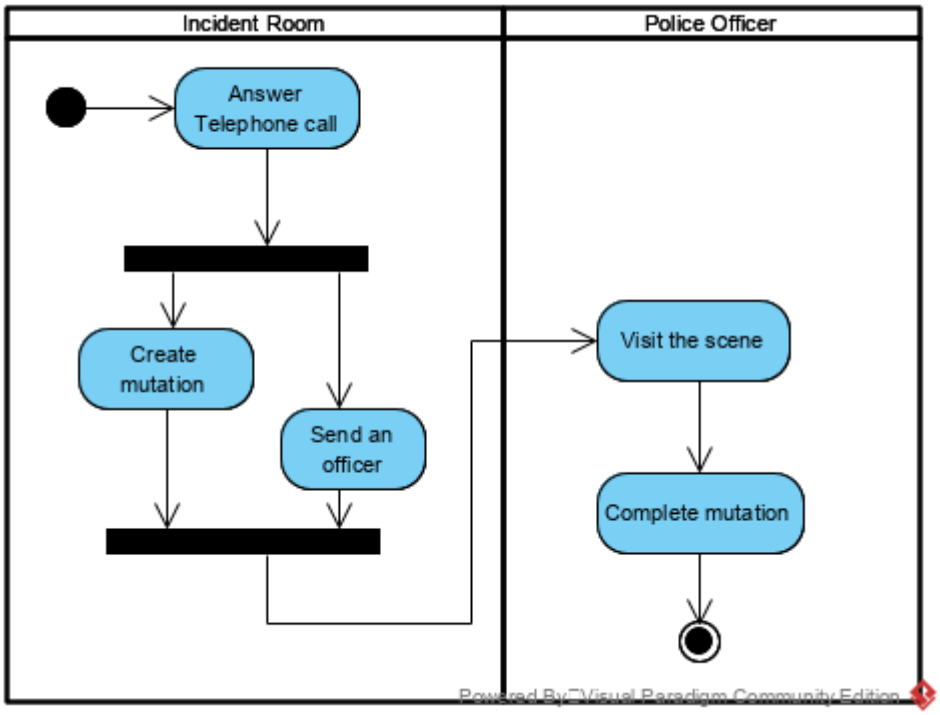
Activity diagrams

Only activities flow, not data flow. There is enough time in the test. I suggest that you first draw a rough sketch of the first version of the activity diagram when you read through the case study. Then redraw the activity diagram, and, maybe, redraw it a third time.

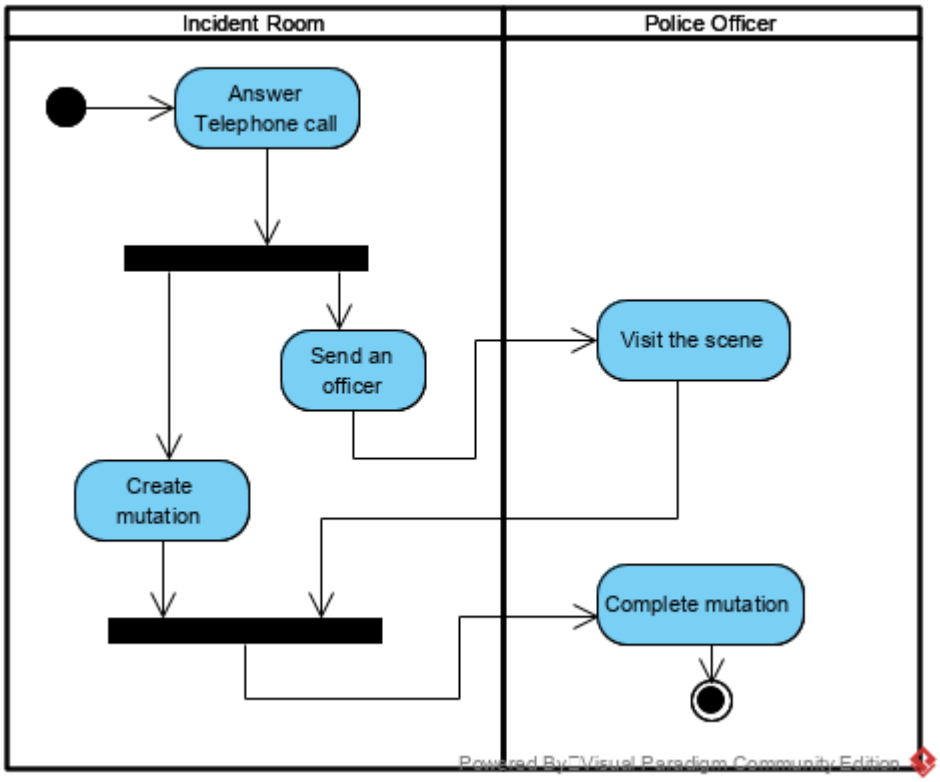
Some notes regarding exercise D-1.2

Depending on the interpretation the diagram will change:

Interpretation 1: The incident room officer does two things – in any order – namely, create a mutation and send an officer (parallel actions). In the activity diagram below it is implied that only after both actions have taken place the police officer will visit the scene.

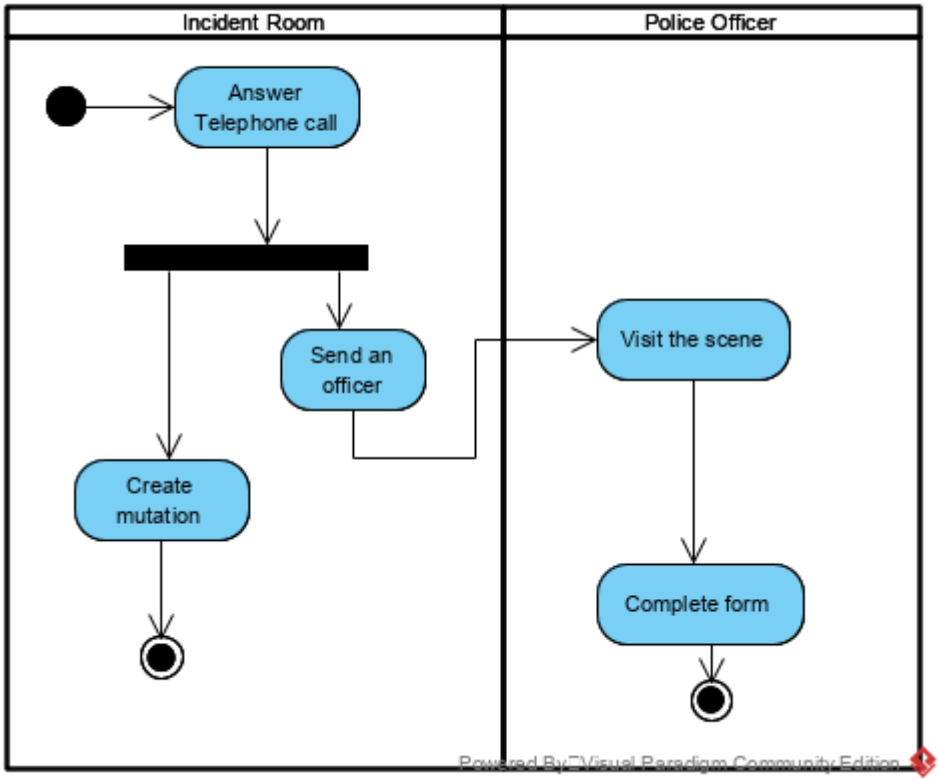


Interpretation 2: The police officer visits the scene regardless of whether the mutation was created. The mutation has to be created, though, before it can be completed.



Option 3:

The police officer does not have to complete the created mutation, but completes another form that is not dependent on the creation of the mutation. The fork does not have to rejoin in this case. Both arrows could point to one end node, or there can be more than one end node. There can always just be one start node.



Use Case Diagram and Descriptions

I suggest you do a rough drawing of the use case diagram and then redraw it. A use case description can most probably be done in one go.

Note: Only shows bubbles for requirements that represent 1 total encounter with the computer at a time. I.e., process order form, book a ticket, etc. Does not model sequence, but high level requirements. The sequence that takes place within a requirement is modelled in the use case description. Each bubble will have its own use case description. An include or extend can be seen as a "subprogram" that will be called from another bubble. It is usually a section of the requirements that will appear in several bubbles of the use case diagram. Instead of recoding it in every bubble's use case description, it will be modelled once in its own use case description and be used by several other use case descriptions. (This is similar to coding a sort, for example, in its own method and calling it from several other methods – avoids duplication of code.) It can also be used to simplify a complex and lengthy use case.

Not all actors from the activity diagram will necessarily appear in the use cases. Only those that directly interact with the system.

One use case can be used by more than one actor. But if one actor can do **all** the requirements of another actor, and more, indicate this by an arrow that goes from the one actor to the other actor. Eg. Figure 1.M where the cashier can also do all the interactions that the admin staff can do.

The system clock can also be an actor if it initiates an interaction with the system, such as end of month accounts, etc.

In a use case description the interaction of an actor with the system is modelled, step by step.

Alternatives can often be drawn in the use case description as loops or as if-else's. Both methods can be used. If you relate this to programming – you will use an if-else if this is a logical flow in your program. Eg. If you have a discount voucher when you order something online, the discount will be calculated and deducted from your total before the final total is printed. This is a normal flow of events. On the other hand, if you try and draw money from an empty bank account, this will result in an error message and can be handled as an exception to the rule. This will then be coded as an alternative/exception at the end of the use case description. The normal flow "if-else" will also reflect similarly in the sequence diagram where the normal flow will be coded as opt and alt. Exceptions are not normally shown in the sequence diagrams, although a method to show an exception has been created in https://johanvergeer.github.io/posts/uml-sequence-diagram-exception-handling, by using a break statement, or by https://www.modelio.org/forum/7-general-help/4354-solved-how-to-model-exceptions-in-sequence-diagrams.html where a try-catch concept is used, but this is not standard UML. You do not need to know this for the test. You can ignore exceptions in your sequence diagrams).

Class diagram

You are requested to draw a class diagram from the case study. A class diagram only models the data and the relationships between the data. (An Activity diagram models the activities, the use case diagram the requirements, and the use case description the interactions of a user with the system). Underline all nouns and decide which ones will be the classes and which ones the attributes.

You do not need to add the methods, only the attributes that are clear from the case study.

If a specific type of object can be of different subtypes, use generalisations. (Eg. A boat can be either a sail boat or a motorboat – they most probably have some different attributes and/or different associations) See also the rules for when generalisations should be used in the slides. "We use a generalization if • subclasses can be distinguished that have different attributes • subclasses can be distinguished that have different associations. See slides in Design 2a from slide 62 for examples.

There was a question about how an association class will be coded in Java. (Only for those that are interested in this – not for the test). This does not have an easy solution. Referring to the employee has a contract with a company example from the slides. With an association class only one object of contract may exist that links a specific employee with a specific company. Nothing in Java prevents you from having two objects that link a specific employee and company. It will need a combination of code in the controllers, classes and the database to ensure that an employee can only have one contract with a specific company. When a new contract is created, the database will have to be searched to see whether such a contract already exists. If one already exists a new one cannot be created.

Sequence diagram

A sequence diagram models a use case description, by adding the classes that will be used and the methods that will be used in the classes. It is, therefore, a more detailed diagram of the flow and interactions with the system and the classes. You do not need to add the database interactions, but can draw the diagrams similar to the ones in the slides assuming that the Customer and the Book objects, for example, already exist.

Note the use of loops, opt(similar to alt, but only has a true part – will only be done if the condition specified is true), ref (similar to a subprogram that will be drawn on its own – often used when the diagram becomes too complex) and alt (similar to an if-else).

See the note under use case descriptions for the exceptions which will be ignored in the sequence diagrams.

State machine diagrams

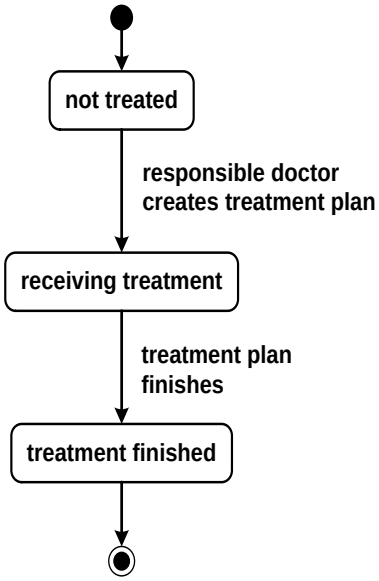
State machine diagrams are used to model the state of some object in the system. It will usually be represented in one (or a combination of a few) attributes of a specific object. For example, The state machine diagram for RedHot's repair process, Figure 3C of exercise D-3.1. The object will be the article and the states will be represented by a "state" attribute that can have as value – no defect, defect reported, cost to be estimated, waiting for permission, etc. or a combination of attributes representing the state, such as defect known – yes or no, defect reported – yes or no, cost estimated – yes or no, etc. The separate states can be combined to give the state of the object at any time in Redhot's repair process.

Remember that an object can never be between states. It always transitions immediately from one state to another. There is always one start symbol, but there does not have to be an end symbol, as the object may stay in one of the states for as long as the object is in the system.

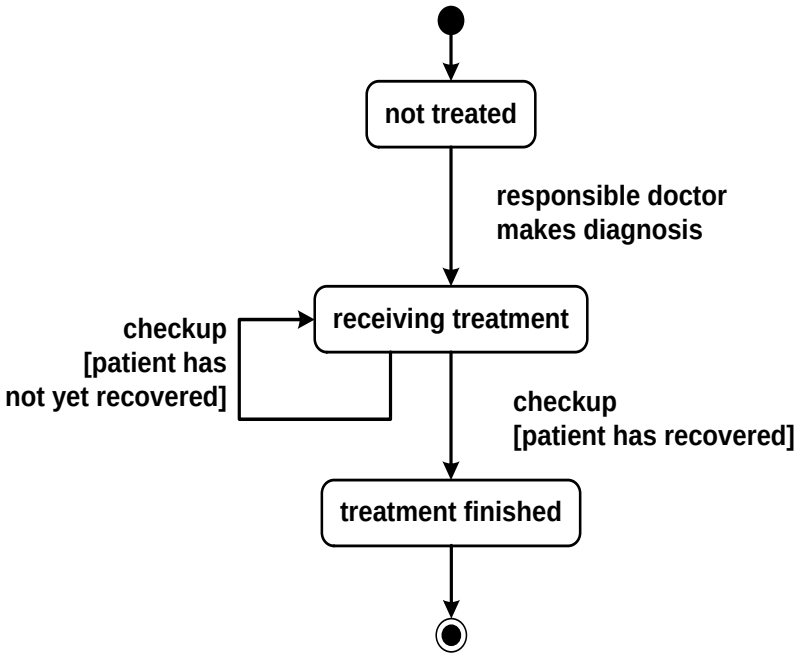
There was a question about when to include loops that returns to the same state.

NB. If there are guards on transitions leaving a state, make sure that there is always a possibility to leave this state.

Example 1: No guards (if- else) have been specified for any of the transitions. A patient will be in any of the states at any time and transition to another state depending on the event specified on the transition.

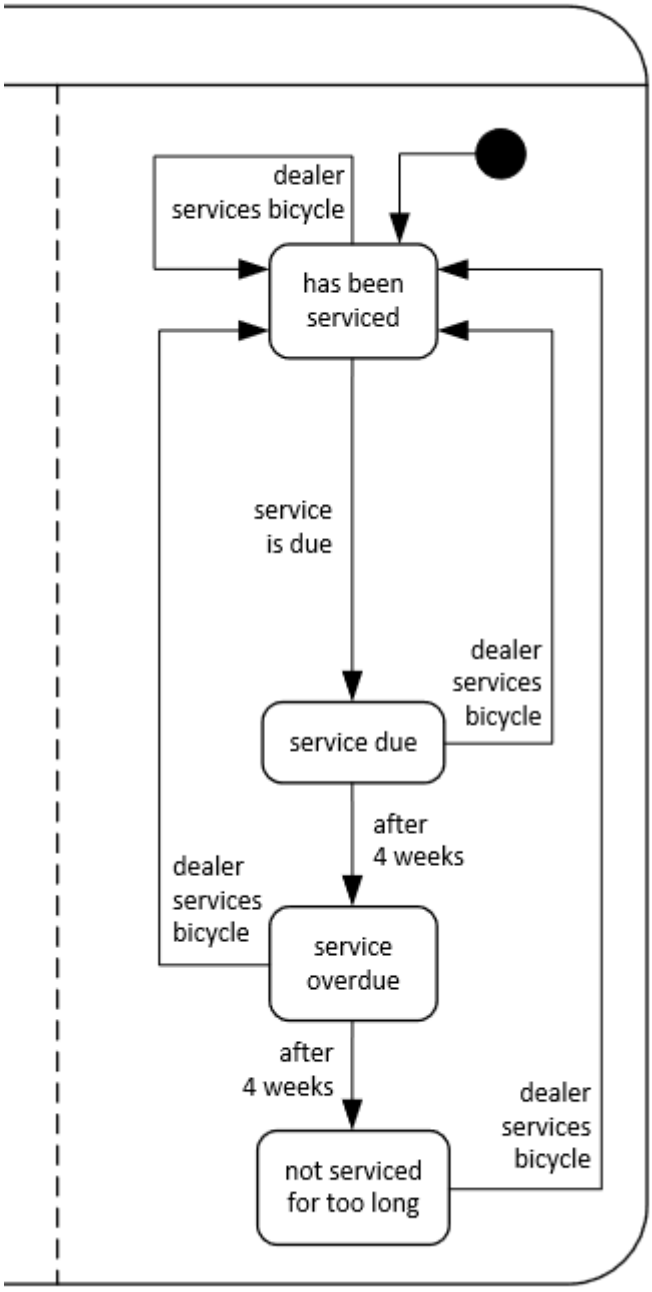


Example 2: This is the same example as the previous one, except that guards have been added due to a "checkup" event that could have different outcomes. **All** outcomes must be covered if you have guards. One of the guards is [patient has recovered] as an outcome of the checkup event, but this does not cover **all** possible outcomes. There is an "else" part that needs to be indicated as well to cover the other option where [patient has not yet recovered]. In this case it will not go to another state, but stay in the same state. That is why there is a loop returning to the state.



Example 3: From 2017 exam solution. In this case there are no guards. The bicycle is leased thus it is in a serviced state when the person leasing the bicycle starts with the lease. Under the lease agreement the bicycle has to be serviced every 6 or 12 months. The person can bring the bicycle in for a service before the 6/12 months has expired, or wait till the service is due. If the bicycle is brought in before the service is due, the bicycle will be serviced and the date of the last service noted, therefore, it is a specific event and the bicycle will go back to a state of being serviced, but with a new service date recorded in the system. This is a specific event so recorded as a loop that

goes from “has been serviced” back to “has been serviced”. The state stays “has been serviced” but other data has changed in the event.



Some useful websites: I am a bit reluctant to recommend web sites as those that I have explored have concepts that we have not included in the course, and you do not have to know for the test. This can be very confusing.

Be careful when you look at these sites as they tend to contain some things we have not covered.

If you want to see different diagrams, google the model names and go to images to see many examples.

Activity diagrams

https://www.smartdraw.com/activity-diagram/
has some extra information that we are not using in this course

https://sparxsystems.com/enterprise_architect_user_guide/14.0/model_domains/activitydiagram.html

https://creately.com/blog/examples/activity-diagram-templates/

https://creately.com/blog/diagrams/activity-diagram-tutorial/#examples – use your discretion as not all examples seem right.

Use case diagram

https://www.youtube.com/watch?v=zid-MVo7M-E

Class diagrams

https://www.youtube.com/watch?v=UI6lqHOVHic

Sequence diagrams

https://www.youtube.com/watch?v=pCK6prSq8aw

State Machine diagrams

Some of the state machine diagrams on the internet are actually activity diagrams.