Chapter 4 -- Exercises

A. Investigate the Web site for Rational Software (www.rational.com) and its repository of information about UML. Write a paragraph news brief on the current state of UML (e.g., the current version and when it will be released; future improvements; etc.).

Rational software is based on the theory and structure of UML, however UML is simply a tool to document system design. Rational goes another step and allows the user to design software with additional business designs and components, as well as facilitates communication and organization of the project. These additional functions create an effective tool for enterprise level application.

The Unified Modeling Language™ (UML™) is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. It simplifies the complex process of software design, making a "blueprint" for construction. For database professionals UML can be used for database design. Using the UML for database design allows the business and application teams who are already using the UML for their designs to share a common language and to communicate with the database team.

The Unified Modeling Language (UML) is a widely used standard in the software industry for modeling software. It helps practitioners visualize, communicate, and implement their designs. UML 2.0 is the first major revision to the standard since its inception in 1997. The new specification is designed to support a number of model-driven development paradigms, including Model-Driven Architecture (MDA) as defined by the Object Management Group (OMG). The UML is an integral component of IBM's ongoing commitment to supporting open standards in the software industry.

Latest news according the Rational website, although very technical for someone new to UML.

On November 18, 2003, IBM initiated a new open source project within Eclipse.org, focusing on implementing a metamodel and a set of APIs for UML 2.0. This project, which is an Eclipse Tools sub-project, is an EMF-based implementation of the UML 2.0 metamodel for the Eclipse platform.

The objectives of this project are to provide a useable implementation of the metamodel to support the development of modeling tools, a common XMI schema to facilitate interchange of semantic models, test cases as a means of validating the specification, and validation rules as a means of defining and enforcing levels of compliance.

B. Investigate the Object Management Group. Write a brief memo describing what it is, its purpose, and its influence on UML and the object approach to systems development. (*Hint:* A good resource is www.omg.org.)

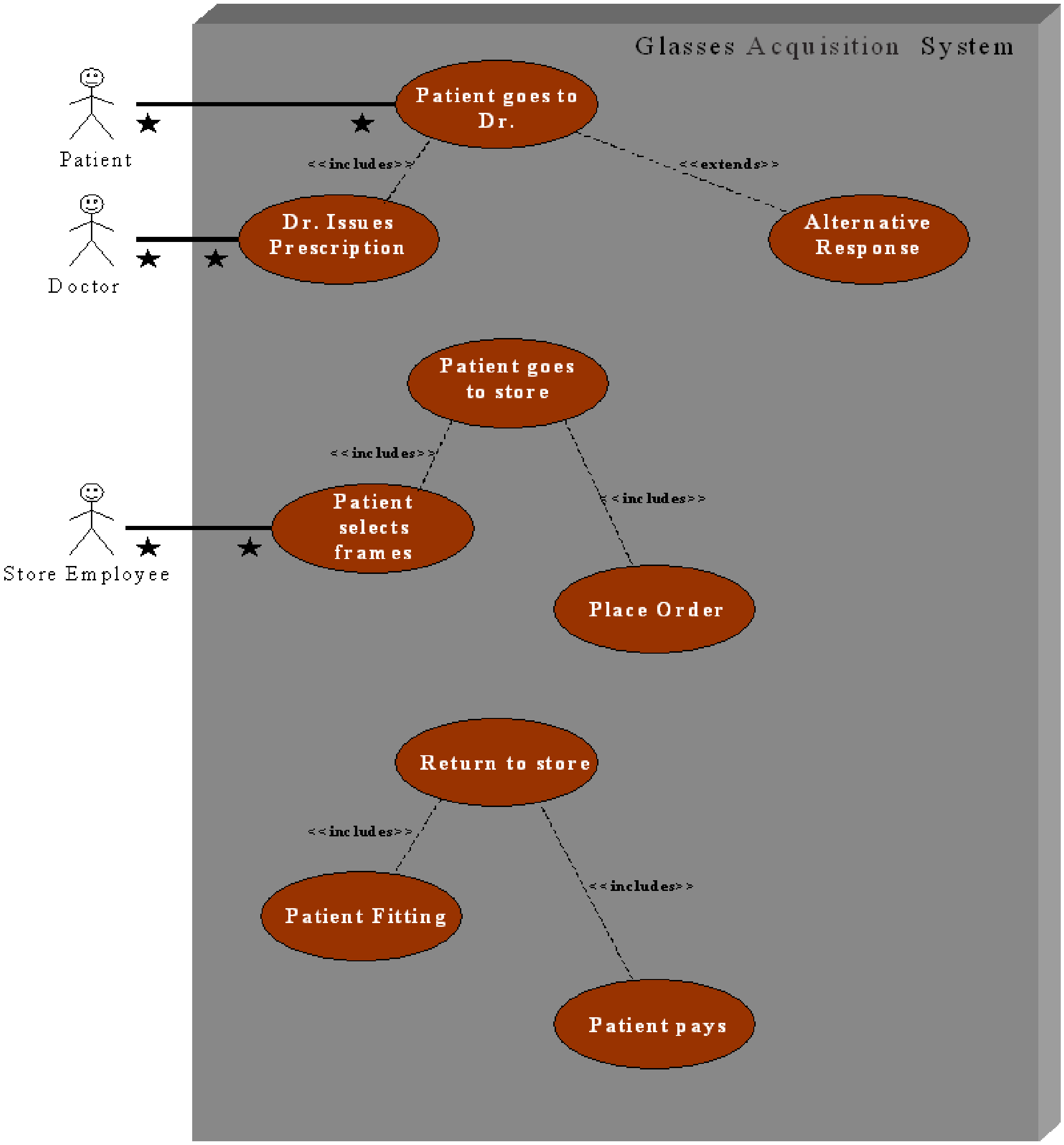
OMG’s MDA (Model Driven Architecture) aims to build industry standards for design architecture that will utilize existing architecture, and allow effective future expansion. MDA is based on the concept of modeling with UML. At the heart of MDA is an approach to design based on treating it as a product, rather than a process. Non-MDA approaches treat design as a process in which developers apply their expertise to handcraft elegant solutions from often ill-defined problems. The result is running code. The design is not captured explicitly, but instead smeared across the code as a set of results of a skilled thinking process.

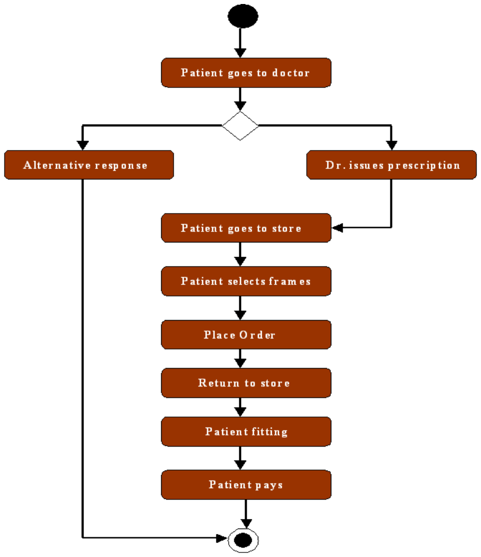
In an MDA approach, promulgated by OMG and based on the work of Shlaer and Mellor, and others, the design is itself a product that is expressed as a set of formal mapping rules. Examples of such rules are "Turn an application class into a class in the implementation with a container for its instances" and "Turn each transition on a statechart diagram into a tuple in a linked list defining the transition and the procedure to call to execute actions". These rules *are* the design, and once they are executed against an application model that captures a problem, the result is (more or less) running code.

In MDA the application model is generalized as a *platform-independent* model1, or PIM, usually visualized using standard UML, and the framework for formal transformation rules is currently being standardized as the QVT (Query, Views, and Transforms) specification. The result of the transformation can either be another, *platform-specific*, model (PSM), or components of the target application, such as database schemas, deployment scripts, test scripts, or indeed executable code. In the latter case, the PIM must be behaviorally complete, defining business logic in terms of abstract actions.

With its rich palette and middleware independence, UML forms the foundation of OMG's Model Driven Architecture (MDA). In fact, a UML model can be either platform-independent or platform-specific, as we choose, and the MDA development process uses both of these forms: Every MDA standard or application is based, normatively, on a *Platform-Independent Model* (PIM), which represents its business functionality and behavior very precisely but does not include technical aspects. From the PIM, MDA-enabled development tools follow OMG-standardized *mappings* to produce one or more *Platform-Specific Models* (PSMs), also in UML, one for each target platform that the developer chooses. (This conversion step is highly automated, but not magic: Before the tool produces a PSM, the developer must annotate the base PIM to produce a more specific but still platform-independent PIM that includes details of desired semantics, and guides choices that the tool will have to make. Because of the similarities among middleware platforms of a given genre - component-based, or messaging-based, for example - this guidance can be included in a PIM without rendering it platform-specific. Still, developers will have to fine-tune the produced PSMs to some extent, more in early days of MDA but less and less as tools and algorithms advance.)

C. Draw a use-case diagram and a set of activity diagrams for the process of buying glasses from the viewpoint of the patient. The first step is to see an eye doctor who will give you a prescription. Once you have a prescription, you go to an optical dispensary, where you select your frames and place the order for your glasses. Once the glasses have been made, you return to the store for a fitting and pay for the glasses.


Glasses Acquisition System
Patient
Doctor
Store Employee
★
Patient goes to Dr.
<<includes>>
<<extends>>
Dr. Issues Prescription
Alternative Response
Patient goes to store
Patient selects frames
Place Order
Return to store
Patient Fitting
Patient pays



D. Create a set of detail use-case descriptions for the process of buying glasses in Exercise C.

1. The system boundary statement should include all steps involved in determining the need through final payment for eyeglasses. Excluded would be any diagnosis or prescription from the doctor for other eye related problems; excluded would be all issues pertaining to driving, flying, walking, or taking a taxi to the store; excluded would be any calls to see if the glasses are ready or visits prior to the glasses being ready; purchase of other equipment such as an eyeglass case; and issues of checks bouncing or credit cards being overcharged. Individual students may see differently what exactly is included and what is excluded. The key, though, is identifying some items that are not to be included in the system. It will be interesting to see how many in the class believe that the going to the doctor, going to the store, and returning to the store are inside versus outside the boundary of the system. This would represent a good in class discussion opportunity focusing on why you might include or exclude these.

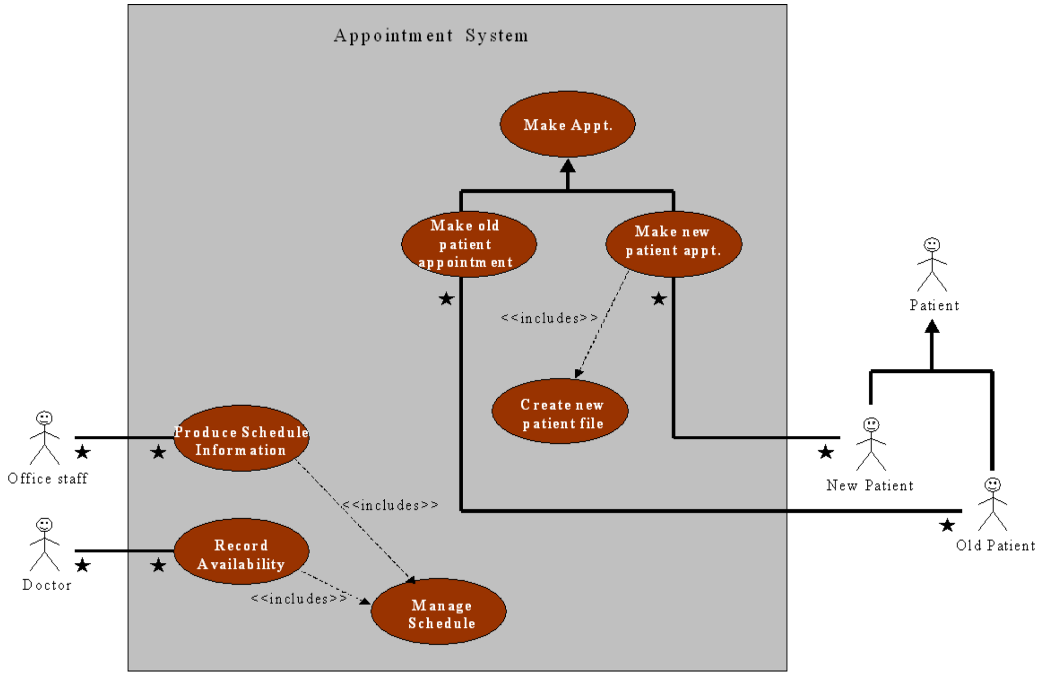
2. The primary actors include the purchaser, the doctor, and the eyeglass store employees. The instructor will have to judge any others that the student might identify (e.g. the subcontractor who receives the prescription and grinds the lenses) and consider whether these actors are inside or outside of the boundary of the system.

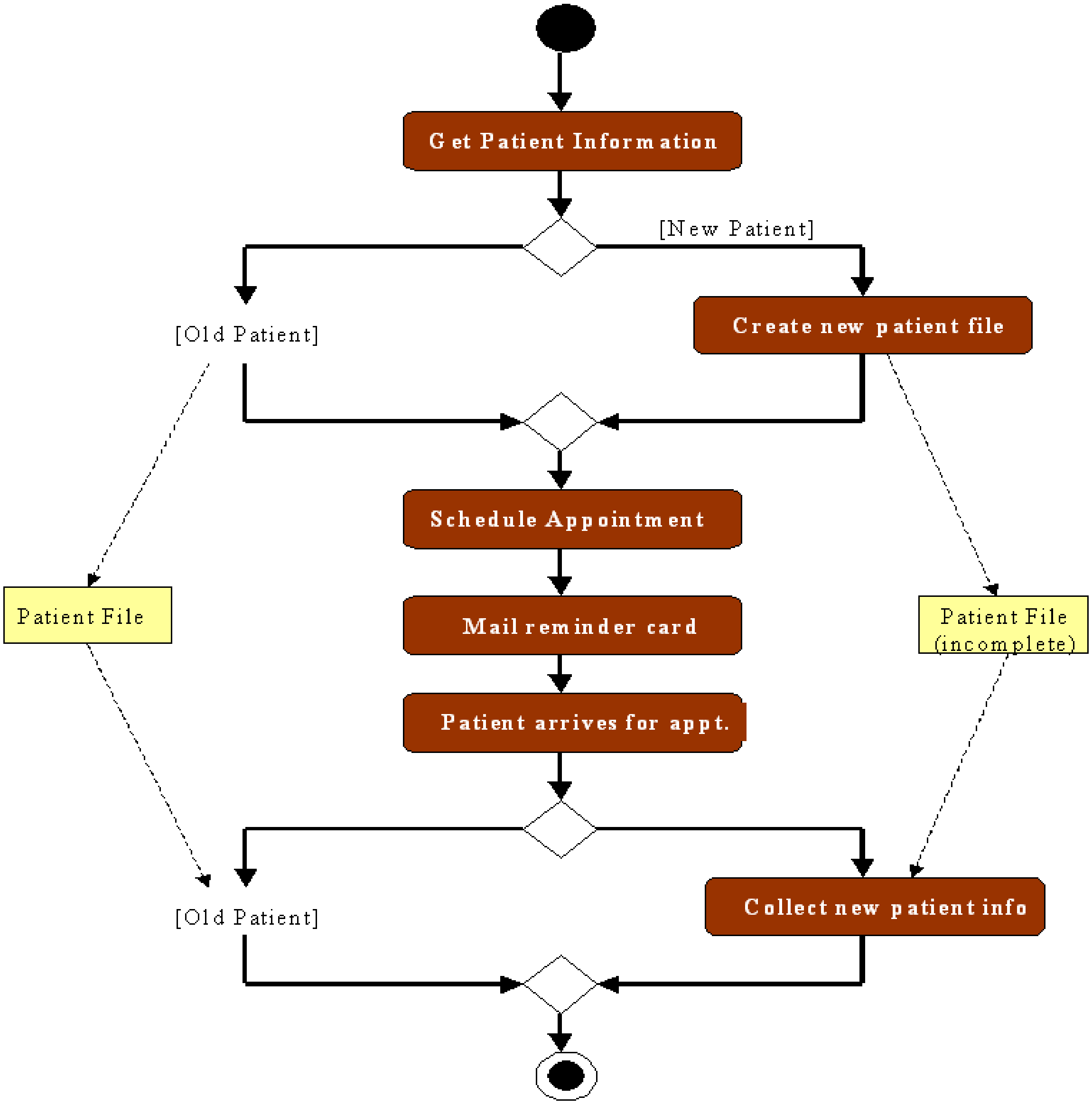
3. Each of the main steps should generate one or more goals. The prescription should be accurate; the frames should be compatible with the lenses, should be affordable and within the appropriate style range as viewed by the customer; the fitting should assure that all elements of the glasses are appropriate; the payment should conclude the transaction such that the store receives appropriate funds and the customer receives the glasses and a receipt.

4. The overview of the major cases will include the use case name, ID number, type, primary actor (the customer in this situation as defined in the problem), and a brief description. There are five logical actions in the case (and three transportation activities):

- Go to eye doctor
    - Receive prescription
- Go to glasses store
    - Select frames
    - Place order
- Return to glasses store
    - Receive fitting
    - Pay for glasses

E. Draw a use case diagram and a set of activity diagrams for the following dentist office system, but do not bother to identify the flow of events within each use case. Whenever new patients are seen for the first time, they complete a patient information form that asks their name, address, phone number and brief medical history, which are stored in the patient information file. When a patient calls to schedule a new appointment or change an existing appointment, the receptionist checks the appointment file for an available time. Once a good time is found for the patient, the appointment is scheduled. If the patient is a new patient, an incomplete entry is made in the patient file; the full information will be collected when they arrive for their appointment. Because appointments are often made so far in advance, the receptionist usually mails a reminder postcard to each patient two weeks before their appointment.





F. Create a set of detail use-case descriptions for the dentist office system in Exercise E.

1. It may be challenging to delineate the boundary. Note that managing the dental office staff or advertising for new customers should clearly be outside this system.

2. The problem identifies this as a dental office system. However, the actions are better described as a scheduling or medical records system. Actors should include the prospective patient, the patient (a subset of prospective patients), office staff, and, perhaps, the dentist.

3. Each step should trigger one or more goals. The new appointment must not conflict with earlier scheduled appointments (other metrics will vary by

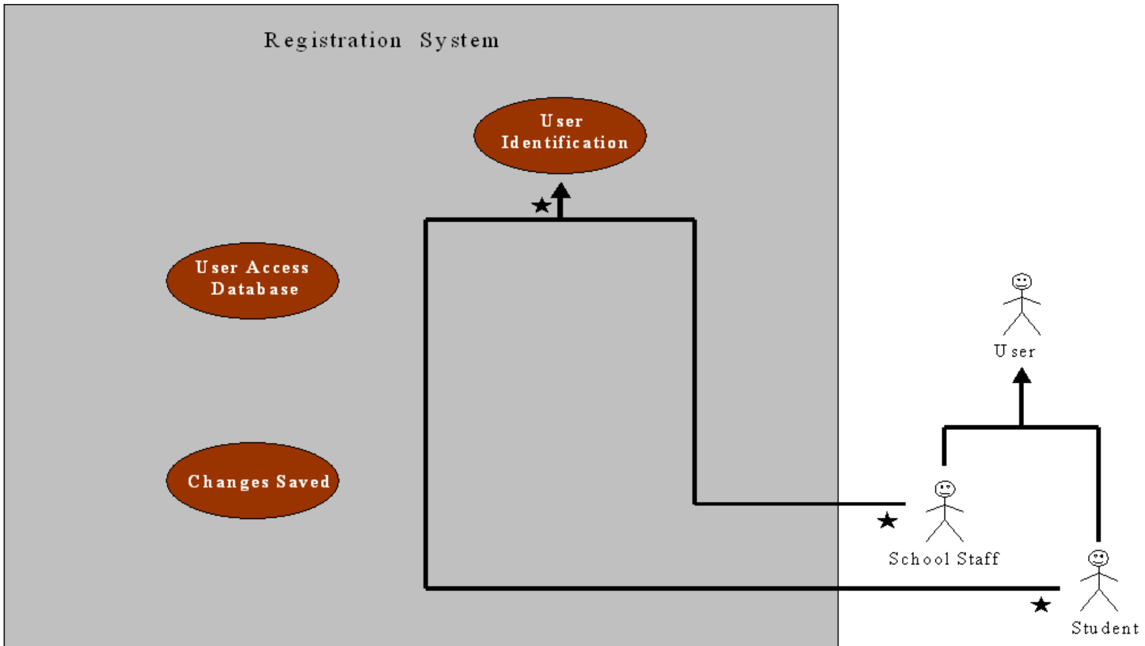
office), collecting information should result in a complete and accurate record, the mailed appointment should occur within the parameters listed and should result in decrease in missed appointments.

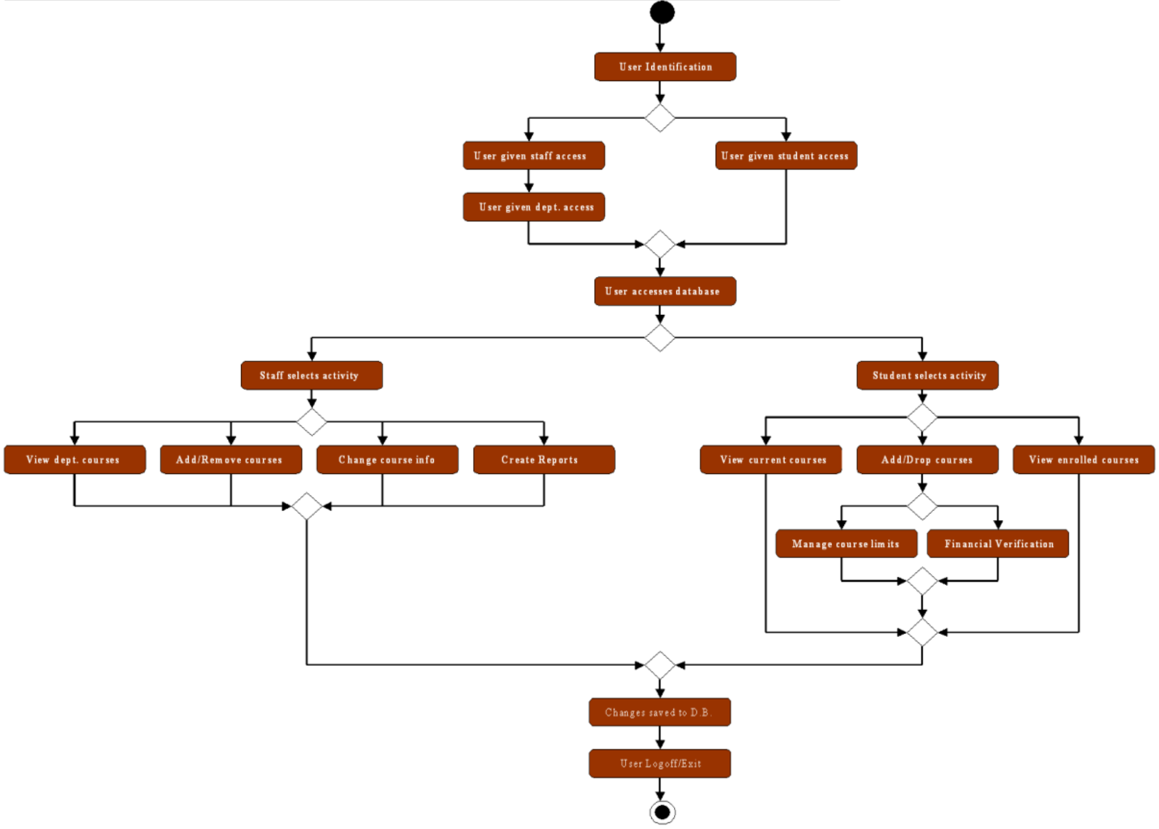
4. Students should identify the following high level tasks:

- See new patients
    - Schedule new appointment
    - Collect full patient information
    - Mail appointment reminder

Note that the rest of the contingencies and conditions should be viewed as expansions of these basic activities. Also note that the steps are not presented in typical logical sequence in the problem – gathering patient information is presented first, but will be conducted after the new appointment is scheduled.

G. Draw a use-case diagram and a set of activity diagrams for an online university registration system. The system should enable the staff of each academic department to examine the courses offered by their department, add and remove courses, and change the information about them (e.g., the maximum number of students permitted). It should permit students to examine currently available courses, add and drop courses to and from their schedules, and examine the courses for which they are enrolled. Department staff should be able to print a variety of reports about the courses and the students enrolled in them. The system should ensure that no student takes too many courses and that students who have any unpaid fees are not permitted to register (assume that fees data are maintained by the university's financial office, which the registration system accesses but does not change).


Registration System
User Identification
User Access Database
Changes Saved
User Logoff/Exit
User
School Staff
Student
★
★
★


User Identification
User given staff access
User given student access
User given dept. access
User accesses database
Staff selects activity
Student selects activity
View dept. courses
Add/Remove courses
Change course info
Create Reports
View current courses
Add/Drop courses
View enrolled courses
Manage course limits
Financial Verification
Changes saved to D.B.
User Logoff/Exit

H. Create a set of detail use-case descriptions for the online university registration system in Exercise G.

Note also that the functionality available to department staff is listed in two separate sections – items 1-3 then later items 4-5.

Department Staff

1. Examine departmental courses

2. Add and remove courses

3. Change information about courses

   Assuming a database with a table for courses, the first three activities should involve selecting courses by department of the information requestor. The first function would relate to a "list" command in SQL or a related language. How much information to display (numbers only, numbers and titles, numbers, titles, and full descriptions) may be pre-specified or left for run-time menu options for the user. A blank, possibly pre-formatted screen should be available to accept new course information; matching by course name or number should bring a current course for deleting (removing) or for revising and re-storing.

4. Print reports about courses

5. Print reports about students enrolled in courses

   Assuming a database with tables for courses, students, and enrollments, these reports should be designed to appropriately merge tables and allow for user specification of additional selection criteria for students/courses or for full output of all details.

   Students may answer this question in terms of step-by-step what the department user must do in order to achieve these tasks without reference to the database functionality embedded in the application.

   Instructors may want to select this exercise if the preponderance of students has already taken a database course. Alternatively, this will be a challenging exercise for students who have not taken a database course, but can be a good precursor to membership in such a course.

Students

1. Examine currently offered courses

   This should draw information from an "offered courses" table. Exceptions should be noted in terms of things like courses offered but not available to a given student (e.g. only for those in a different major thus needing petitioning for entry;

those that are already fully enrolled; those that are for different student levels such as for seniors only; etc.)

2. Add and drop courses from their schedule

- ensure that no student takes too many courses
- ensure that students may not register if they have unpaid fees

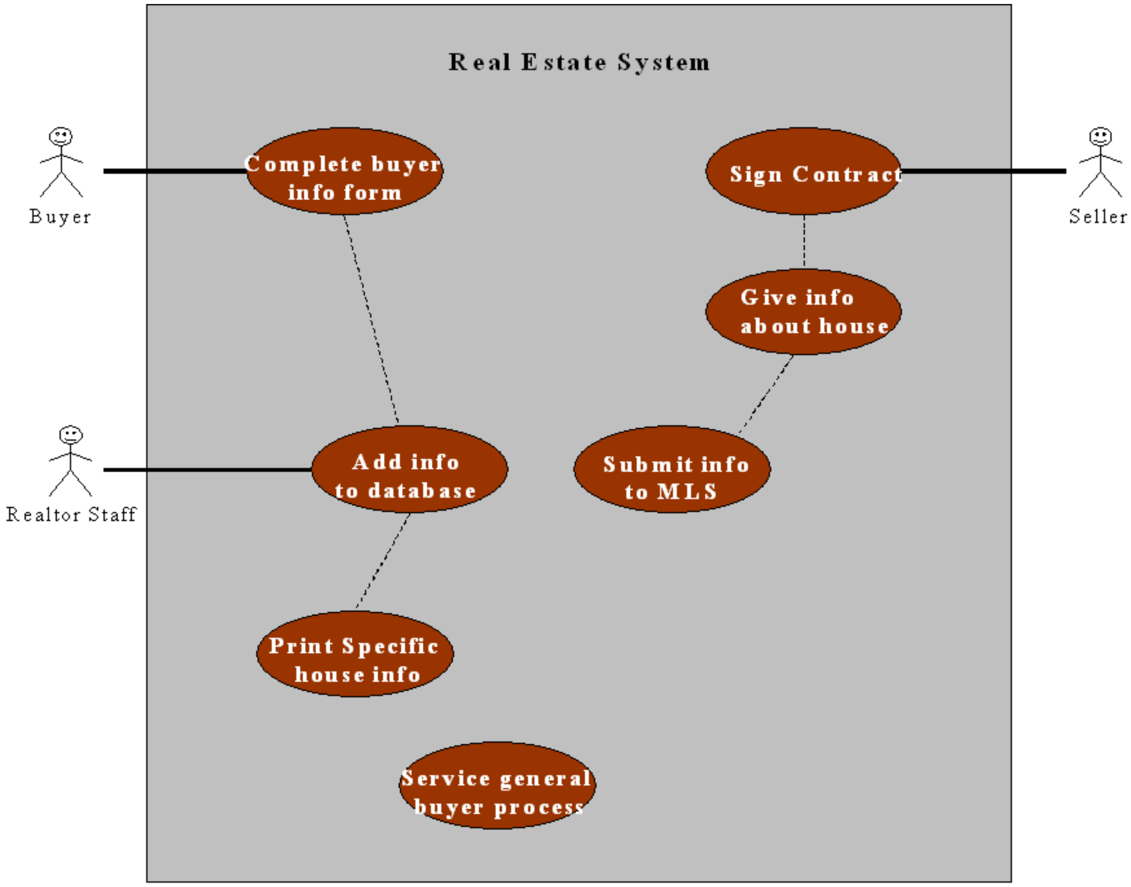
This should enable students to add or delete records from the "enrollment" table. Data regarding their identifier, the identifier of the course, and any other "enrollment" data should be added to the record. It should also add one student to the "courses offered" table's enrollment counter.

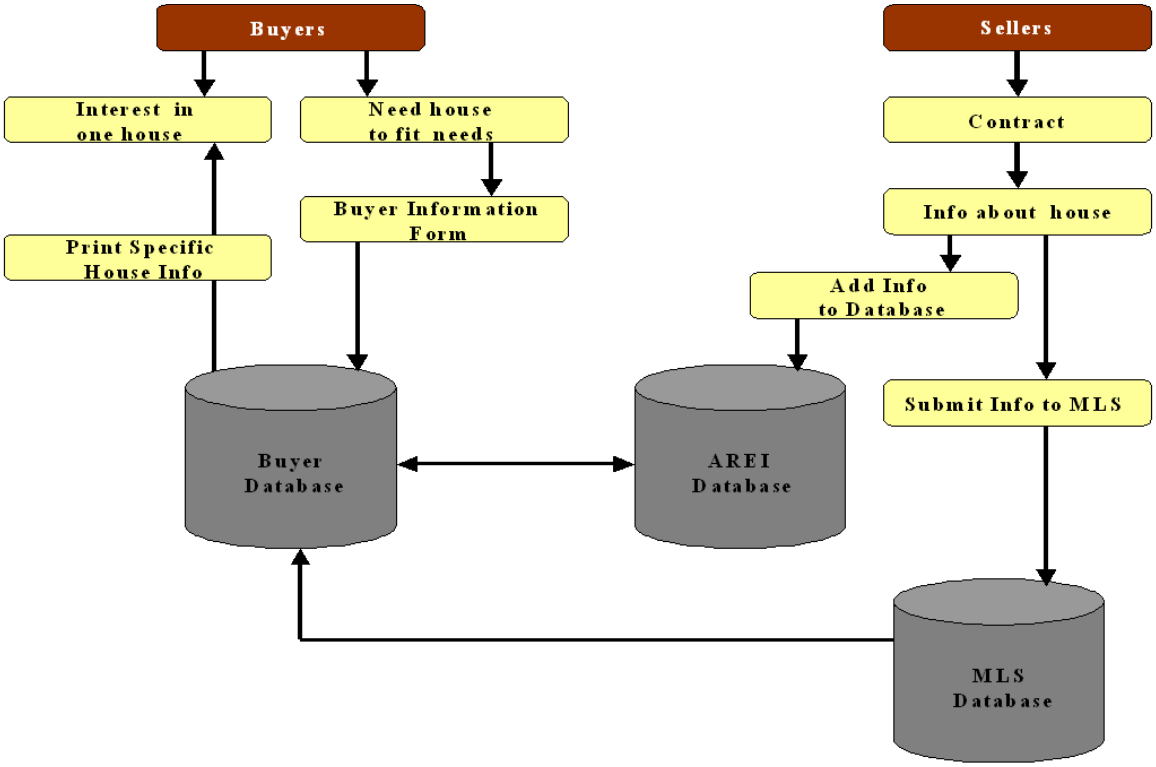
This task will also require utilization of application business logic for integrity constraints. Too many course restrictions will require determining the proper upper limit of courses or units the student is allowed to take (as this will vary based on different business rules – GPA, probation, major, etc.) then comparing this with the prior number committed to and the new total if the requested course is added. The unpaid fees restriction will require matching the student id with the identifier in the financial office database and returning an OK flag or a set of values pertaining to the unpaid fee indication.

3. Examine courses in which they are enrolled

This task will involve providing a listing of the course information from the course table that matches those listed for the student in the "enrollment" table.

I. Draw a use-case diagram and a set of activity diagrams for the following system. A Real Estate Inc. (AREI) sells houses. People who want to sell their houses sign a contract with AREI and provide information on their house. This information is kept in a database by AREI and a subset of this information is sent to the citywide multiple listing service used by all real estate agents. AREI works with two types of potential buyers. Some buyers have an interest in one specific house. In this case, AREI prints information from its database, which the real estate agent uses to help show the house to the buyer (a process beyond the scope of the system to be modeled).Other buyers seek AREI's advice in finding a house that meets their needs. In this case, the buyer completes a buyer information form that is entered into a buyer database, and AREI real estate agents use its information to search AREI's database and the multiple listing service for houses that meet their needs. The results of these searches are printed and used to help the real estate agent show houses to the buyer.


Real Estate System
Buyer
Realtor Staff
Seller
Complete buyer info form
Sign Contract
Give info about house
Add info to database
Submit info to MLS
Print Specific house info
Service general buyer process



J. Create a set of detail use-case descriptions for the real estate system in Exercise J.

Major use cases are listed by actor:

Sellers

1. Sign contract

   Possible expansion steps include: present contract, discuss key items, negotiate any specifics such as percentage commission, finalize contract contents, seller signs contract, AREI employee signs contract, perhaps a witness also signs, dates are included, the contract is filed (how would this differ if on-line?)

2. Provide information regarding house

   Students may list some of the specific details regarding the house. Will the information be written on paper then entered later by data entry specialists, or entered directly onto a computer screen?

AREI

1. Add information to database

This may be a separate step, if data is gathered manually, or it could be an example of a use-case activity with multiple actors.

2. Extract subset and send to multiple listing service.

   The subset is assumed to be constant among houses, therefore the selection logic will be included in the extraction program. Is the data printed and sent on or is it sent by email or other electronic means?

3. Print specific house information (for showing to buyer)

   Use house identifier to locate all data in database pertaining to a particular house, print the data, and share with buyer/buyer agent.

4. Service general buyer process -- note this sequence is complex enough to possibly warrant a separate lower level use case.

   4A. Adds buyer information form to database

   Same issues as with adding seller information

   4b. Search AREI database for buyer desire/specific house match

   Discuss preferences with client, formulate selection values, enter selection values, and extract data subset. The assumption is that this will be displayed on-screen prior to printing in step 4d.

   Students may make the other assumption, that printing results directly following the extraction. There would be advantages to reviewing the displayed results to delete those not appropriate and, potentially, to redefine the search values before printing the hard copy.

   4c. Search multiple listing database for buyer desire/specific house match

   Same issues as 4b with the additional complexities of looping through a set of databases if these have not been unified and the various protocols and methods this might theoretically entail.

   4d. Print AREI database search results

   print the results, retrieve from printer, discuss with client

   4e. Print multiple listing database search results.

   print the results, retrieve from printer, discuss with client

Buyer

1. Completes buyer information form issues similar to seller step 2.

K. Perform a verification and validation walkthrough of the functional models of the real estate system described in exercises I and J.

In order to verify and validate the functional model, a set of rules have been defined to ensure the consistency among the activity diagram (described in I), the use-case description (described in J), and the use-case diagram (defined in I).

First, when comparing an activity diagram to a use-case description, there should be at least one event recorded in the normal flow of events, subflows, or alternate/exceptional flows of the use-case description for each activity or action that is included on an activity diagram, and each event should be associated with an activity or action.

Second, all objects portrayed as an object node in an activity diagram must be mentioned in an event in the normal flow of events, subflows, or alternate/exceptional flows of the use-case description.

Third, sequential order of the events in a use-case description should occur in the same sequential order of the activities contained in an activity diagram.

Fourth, when comparing a use-case description to a use-case diagram, there must be one and only one use-case description for each use case, and vice versa.

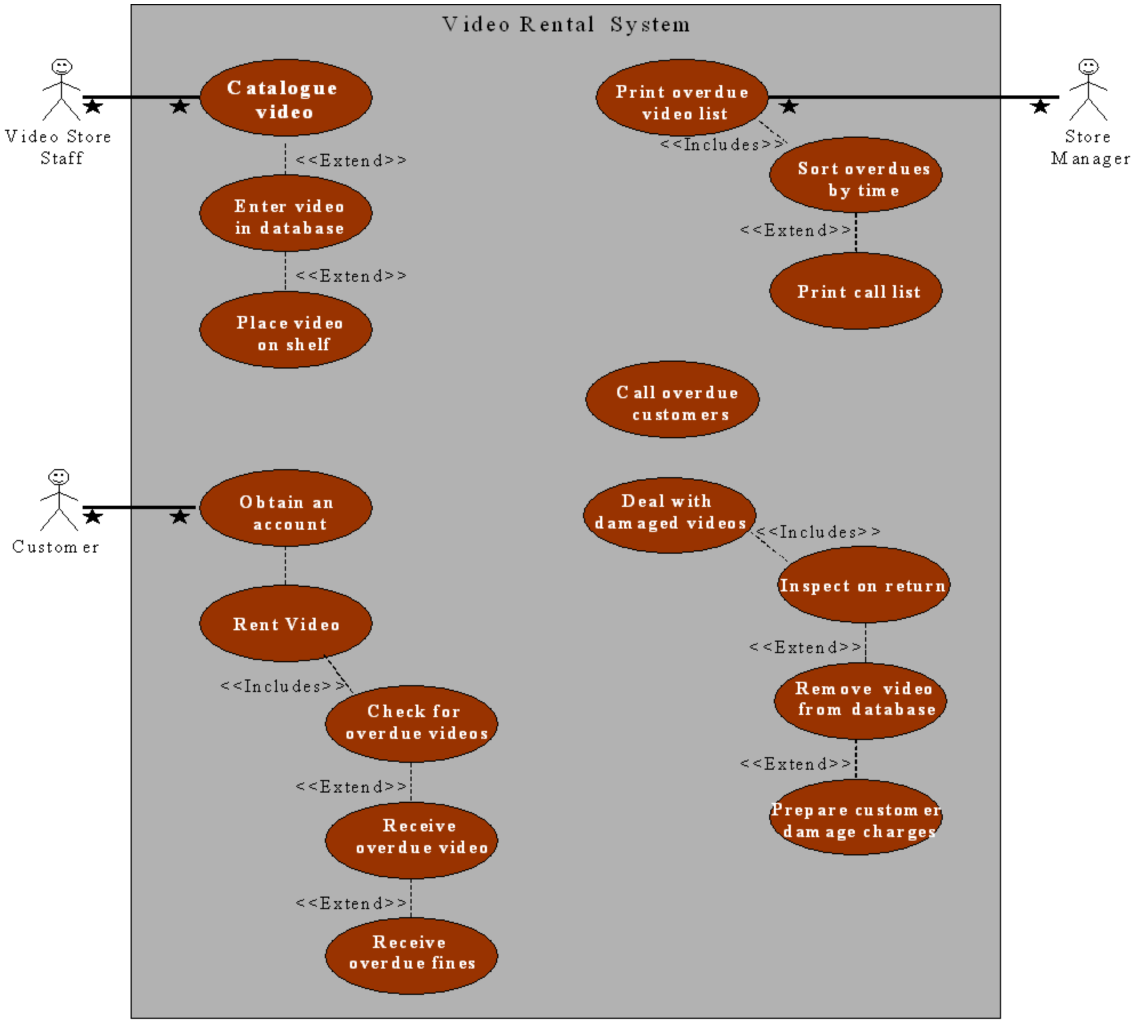
Fifth, all actors listed in a use case description must be portrayed on the use-case diagram. Furthermore, each one must have an association link that connects it to the use case and must be listed with the association relationships in the use-case description. In some organizations, we should also include the stakeholders listed in the use-case description as actors in the use-case diagram.

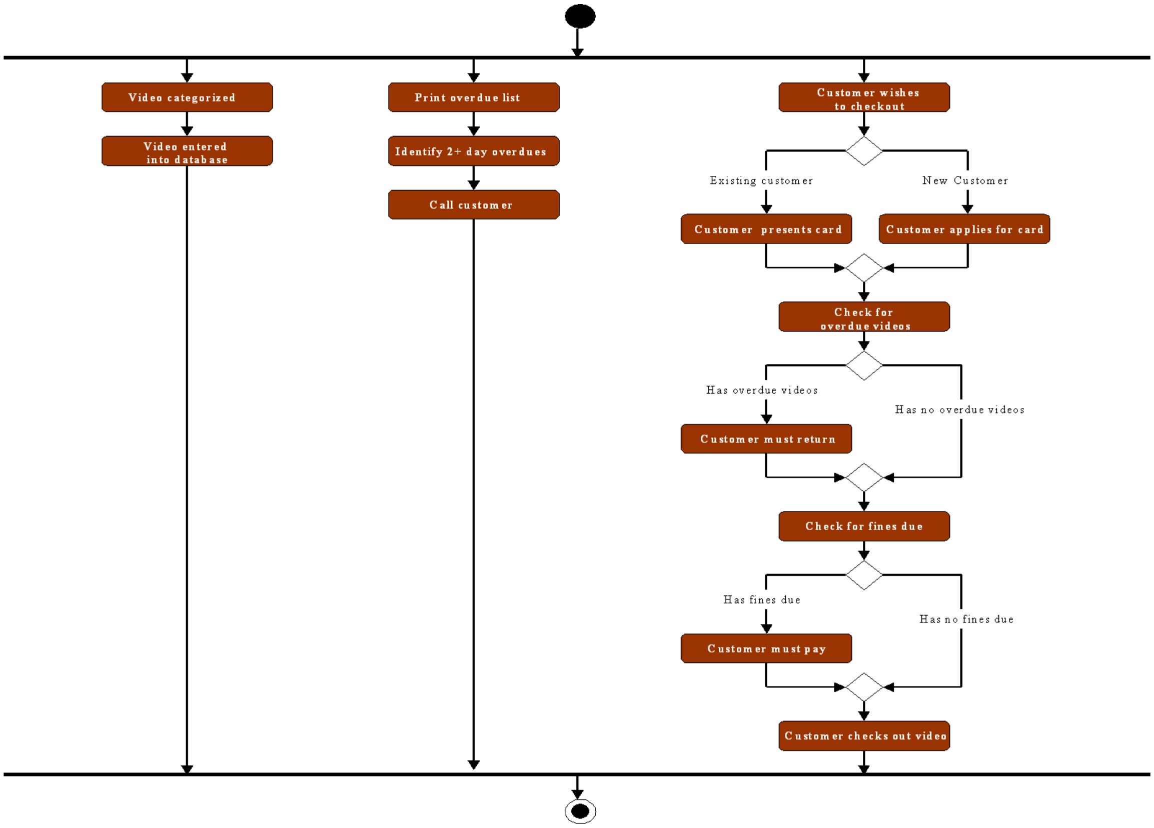
Sixth, all other relationships listed in a use-case description (include, extend, and generalization) must be portrayed on a use-case diagram.

Finally, there are many diagram-specific requirements that must be enforced. For example, in an activity diagram a decision node can be connected to activity or action nodes only with a control flow, and for every decision node there should be a matching merge node. Every type of node and flow has different restrictions.

L. Draw a use-case diagram and create a set of activity diagrams for the following system. A Video Store (AVS) runs a series of fairly standard video stores. Before a video can be put on the shelf, it must be cataloged and entered into the video database. Every customer must have a valid AVS customer card in order to rent a

video. Customers rent videos for three days at a time. Every time a customer rents a video, the system must ensure that they do not have any overdue videos. If so, the overdue videos must be returned and an overdue fee paid before customer can rent more videos. Likewise, if the customer has returned overdue videos, but has not paid the overdue fee, the fee must be paid before new videos can be rented. Every morning, the store manager prints a report that lists overdue videos. If a video is two or more days overdue, the manager calls the customer to remind them to return the video. If a video is returned in damaged condition, the manager removes it from the video database and may sometimes charge the customer.





M. Create a set of detail use-case descriptions for the video system in Exercise L.

Major use cases are listed by actor:

Video store staff

- Catalogue video
- Enter video into database
- Put video on shelf

Customer

- Obtain an account
- Rent a video
- Check for overdue videos
    - Receive overdue video
    - Receive overdue video fine

Store manager

- Print overdue video list (daily)
    - Sort overdue video records/items by length of time overdue
    - Print list

- Call overdue video clients (two or more days late)
- Deal with damaged video
    - Inspect videos upon return
    - Remove damaged videos from database
    - Prepare customer damage charges (rules not specified)

N. Perform a verification and validation walkthrough of the functional models of the video store system described in exercises L and M.

In order to verify and validate the functional model, a set of rules have been defined to ensure the consistency among the activity diagram (described in L), the use-case description (described in M), and the use-case diagram (defined in L).

First, when comparing an activity diagram to a use-case description, there should be at least one event recorded in the normal flow of events, subflows, or alternate/exceptional flows of the use-case description for each activity or action that is included on an activity diagram, and each event should be associated with an activity or action.

Second, all objects portrayed as an object node in an activity diagram must be mentioned in an event in the normal flow of events, subflows, or alternate/exceptional flows of the use-case description.

Third, sequential order of the events in a use-case description should occur in the same sequential order of the activities contained in an activity diagram.

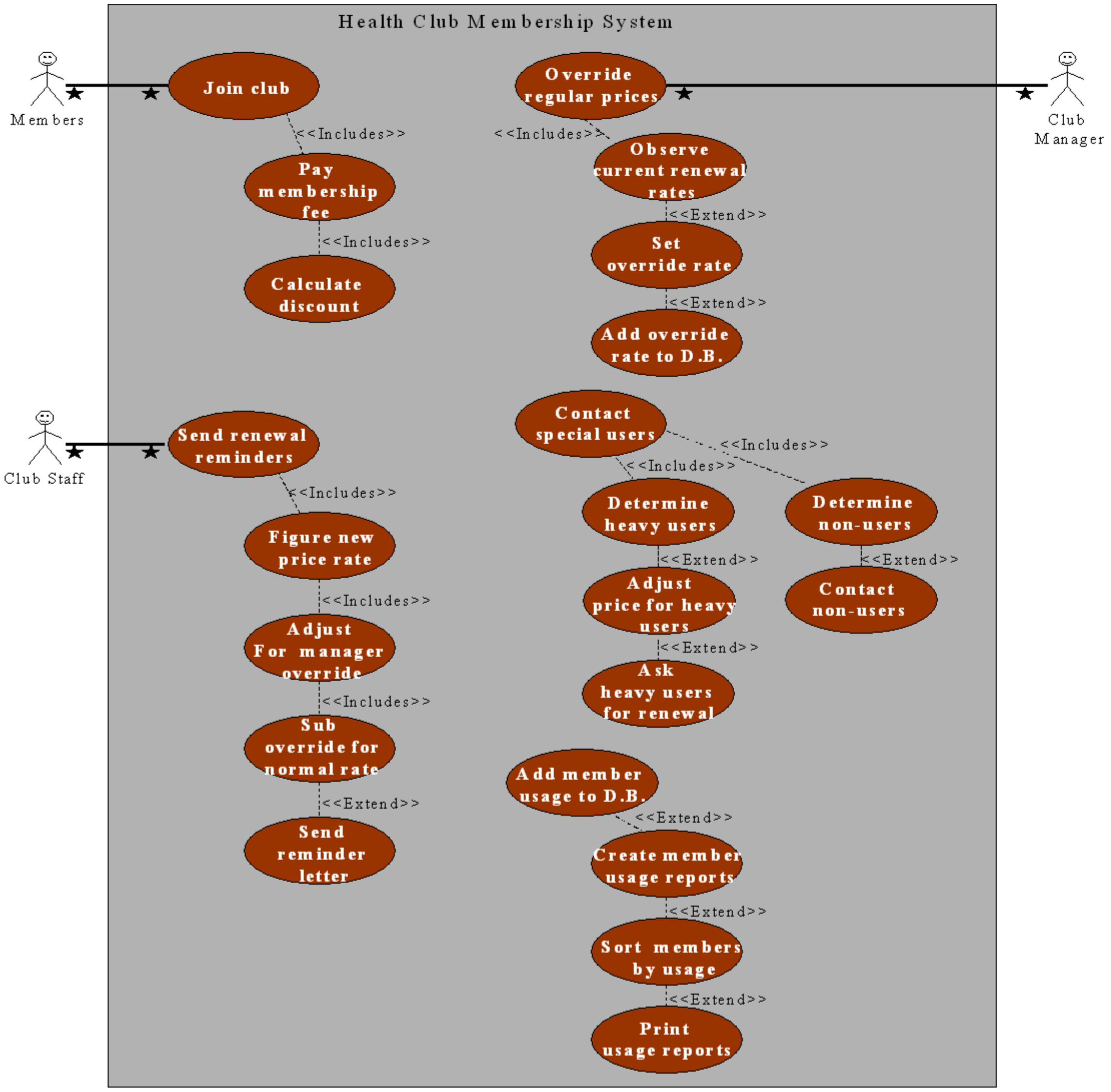
Fourth, when comparing a use-case description to a use-case diagram, there must be one and only one use-case description for each use case, and vice versa.

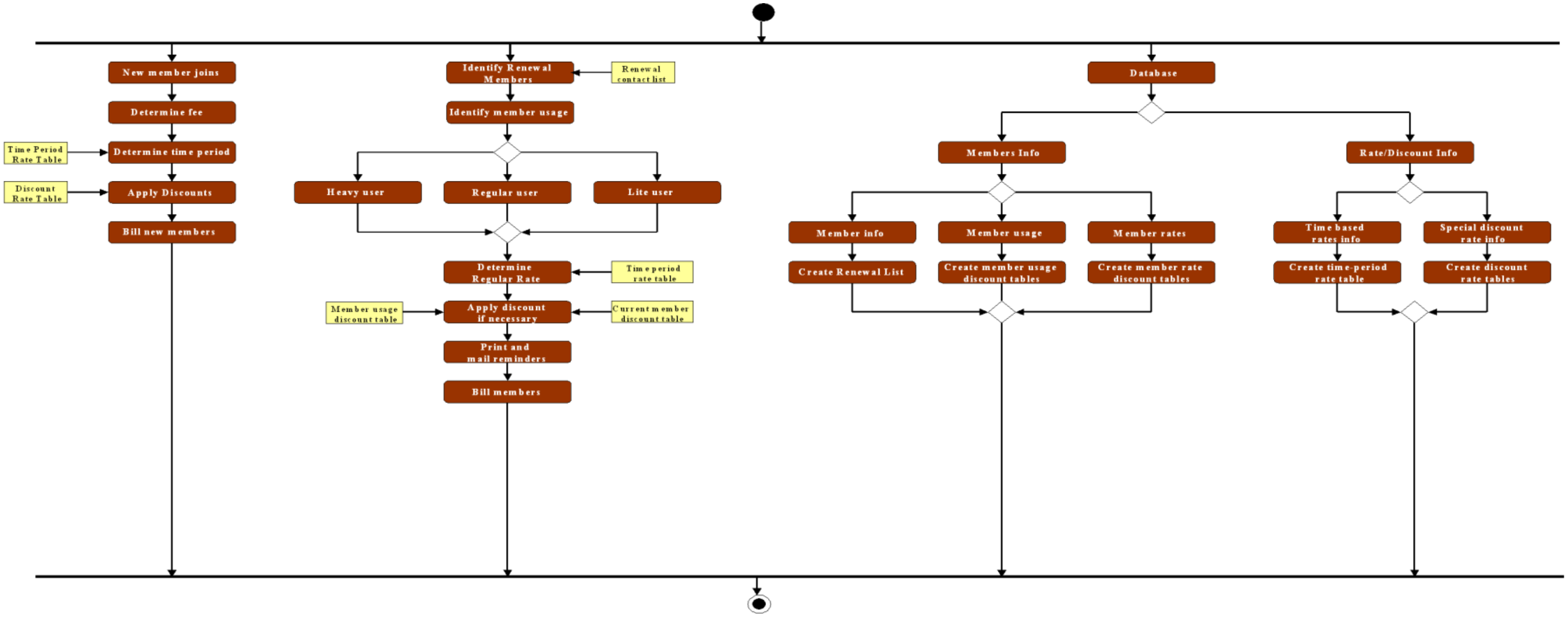
Fifth, all actors listed in a use case description must be portrayed on the use-case diagram. Furthermore, each one must have an association link that connects it to the use case and must be listed with the association relationships in the use-case description. In some organizations, we should also include the stakeholders listed in the use-case description as actors in the use-case diagram.

Sixth, all other relationships listed in a use-case description (include, extend, and generalization) must be portrayed on a use-case diagram.

Finally, there are many diagram-specific requirements that must be enforced. For example, in an activity diagram a decision node can be connected to activity or action nodes only with a control flow, and for every decision node there should be a matching merge node. Every type of node and flow has different restrictions.

O. Draw a use-case diagram and a set of activity diagrams for a health club membership system. When members join the health club, they pay a fee for a certain length of time. Most memberships are for one year, but memberships as short as two months are available. Throughout the year, the health club offers a variety of discounts on their regular membership prices (e.g., two memberships for the price of one for Valentine's Day). It is common for members to pay different amounts for the same length of membership. The club wants to mail out reminder letters to members asking them to renew their memberships one month before their memberships expire. Some members have become angry when asked to renew at a much higher rate then their original membership contract, so the club wants to track the price paid so that the manager can override the regular prices with special prices when members are asked to renew. The system must track these new prices so that renewals can be processed accurately. One of the problems in the health club industry is the high turnover rate of members. While some members remain active for many years, about half of the members do not renew their memberships. This is a major problem, because the health club spends a lot in advertising to attract each new member. The manager wants the system to track each time a member comes into the club. The system will then identify the heavy users, and generate a report so the manager can ask them to renew their memberships early, perhaps offering them a reduced rate for early renewal. Likewise, the system should identify members who have not visited the club in more than a month, so the manager can call them and attempt to re-interest them in the club.


Health Club Membership System
Members
Club Manager
Club Staff
Join club
<<Includes>>
Pay membership fee
<<Includes>>
Calculate discount
Override regular prices
<<Includes>>
Observe current renewal rates
<<Extend>>
Set override rate
<<Extend>>
Add override rate to D.B.
Send renewal reminders
<<Includes>>
Figure new price rate
<<Includes>>
Adjust For manager override
<<Includes>>
Sub override for normal rate
<<Extend>>
Send reminder letter
Contact special users
<<Includes>>
<<Includes>>
Determine heavy users
<<Extend>>
Adjust price for heavy users
<<Extend>>
Ask heavy users for renewal
Determine non-users
<<Extend>>
Contact non-users
Add member usage to D.B.
<<Extend>>
Create member usage reports
<<Extend>>
Sort members by usage
<<Extend>>
Print usage reports


New member joins
Determine fee
Time Period Rate Table
Determine time period
Discount Rate Table
Apply Discounts
Bill new members
Identify Renewal Members
Renewal contact list
Identify member usage
Heavy user
Regular user
Lite user
Determine Regular Rate
Time period rate table
Member usage discount table
Apply discount if necessary
Current member discount table
Print and mail reminders
Bill members
Database
Members Info
Rate/Discount Info
Member info
Member usage
Member rates
Create Renewal List
Create member usage discount tables
Create member rate discount tables
Time based rates info
Special discount rate info
Create time-period rate table
Create discount rate tables

P. Create a set of detail use-case descriptions for the system in Exercise O.

Major use cases are listed by actor:

Members

- Join club
  - Pay fee that varies by membership length (should have look-up table or equivalent logic for durations and price for each)
- Calculate discount (should have look-up table or equivalent logic for discount source and amount)

Club staff

- Send reminder letters (one month before expiration)
  - Calculate new rate
  - Adjust for Manager overrides (where needed)
  - Substitute override for renewal rate in letter (where needed)
  - Send reminder letter

Manager

- Override regular prices with special prices
  - Observe current and renewal rates
  - Set override rate (where judged to be warranted)
  - Add override rate into database

- Ask heavy users for early renewal
  - Determine heavy users from report
  - Adjust renewal price for heavy users
  - Ask heavy users for renewal
  - Determine non-users from report (zero uses in at least a month)
  - Call non-users and attempt to persuade to increase use

Club staff/member (not clear in case who enters this data) – maybe one or both; may happen through automated source data entry, e.g. use of a smart card.

- Add record to member-usage database (each time member uses club)

Club staff/manager (not clear in case who handles this task) – could be one or the other or both.

- Prepare heavy user report
  - Sort members by amount of usage
  - Print heavy user report

The second level activities provide the beginnings of detailed use cases where activities in the case can be grouped into a higher level use case. For the other higher level activities, students should provide more details based on their understanding of this sort of business. Providing more detailed steps will vary with the student's understanding of business operations and the amount of detail they express.

This mini-case includes some activities that are not clearly attributed to any particular actor. The students will probably sort this out in one or another manner. The instructor should decide if only one of the directions to go is correct (and be prepared to explain this to the students) or check to see which answers students come up with and ask them for explanations. Clearly this is a case where in practice the analyst will return to the users to ask for their preferences.

Q. Perform a verification and validation walkthrough of the functional models of the health club membership system described in exercises O and P.

In order to verify and validate the functional model, a set of rules have been defined to ensure the consistency among the activity diagram (described in O), the use-case description (described in P), and the use-case diagram (defined in O).

First, when comparing an activity diagram to a use-case description, there should be at least one event recorded in the normal flow of events, subflows, or alternate/exceptional flows of the use-case description for each activity or action that is included on an activity diagram, and each event should be associated with an activity or action.

Second, all objects portrayed as an object node in an activity diagram must be mentioned in an event in the normal flow of events, subflows, or alternate/exceptional flows of the use-case description.

Third, sequential order of the events in a use-case description should occur in the same sequential order of the activities contained in an activity diagram.

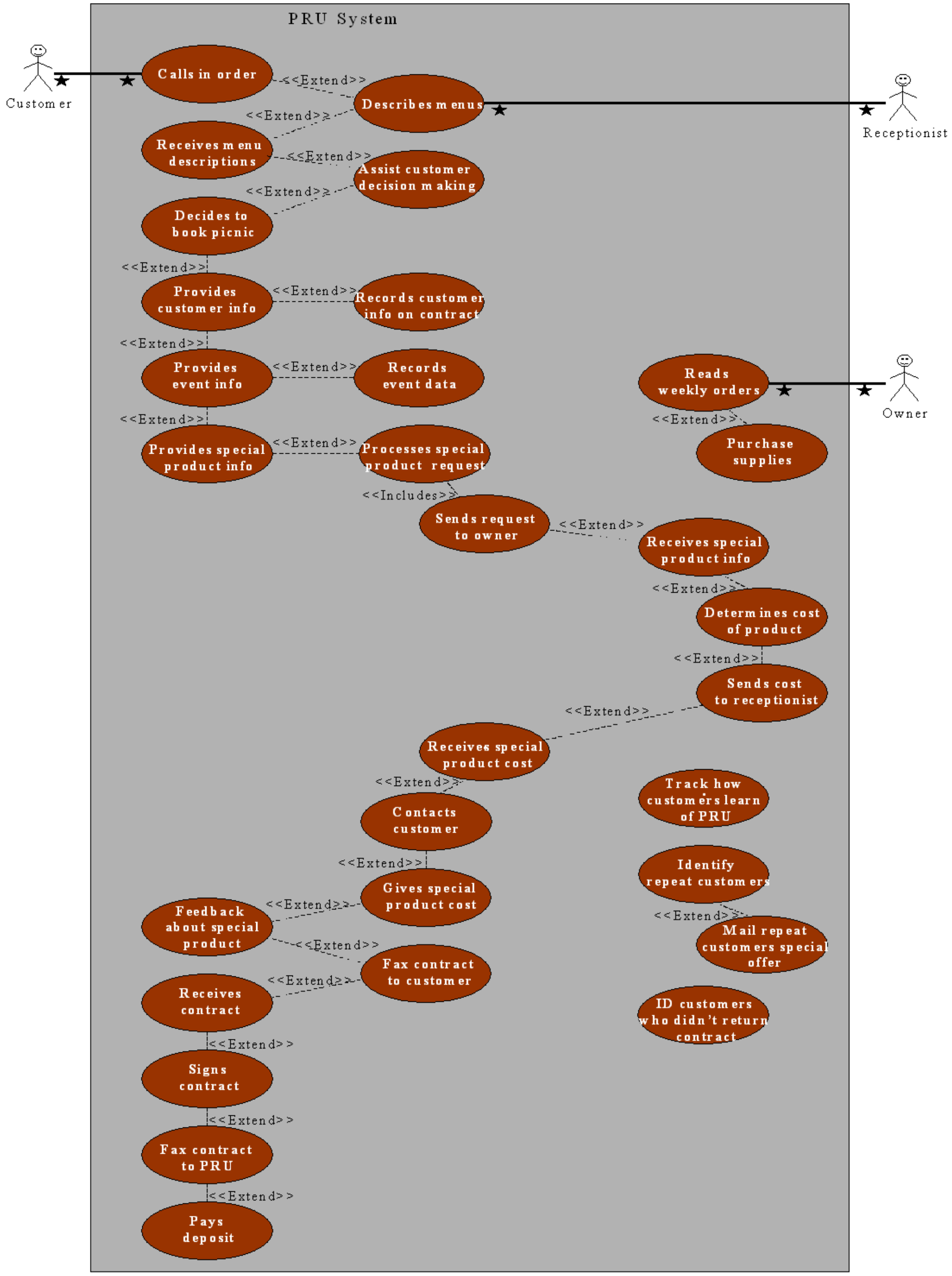
Fourth, when comparing a use-case description to a use-case diagram, there must be one and only one use-case description for each use case, and vice versa.

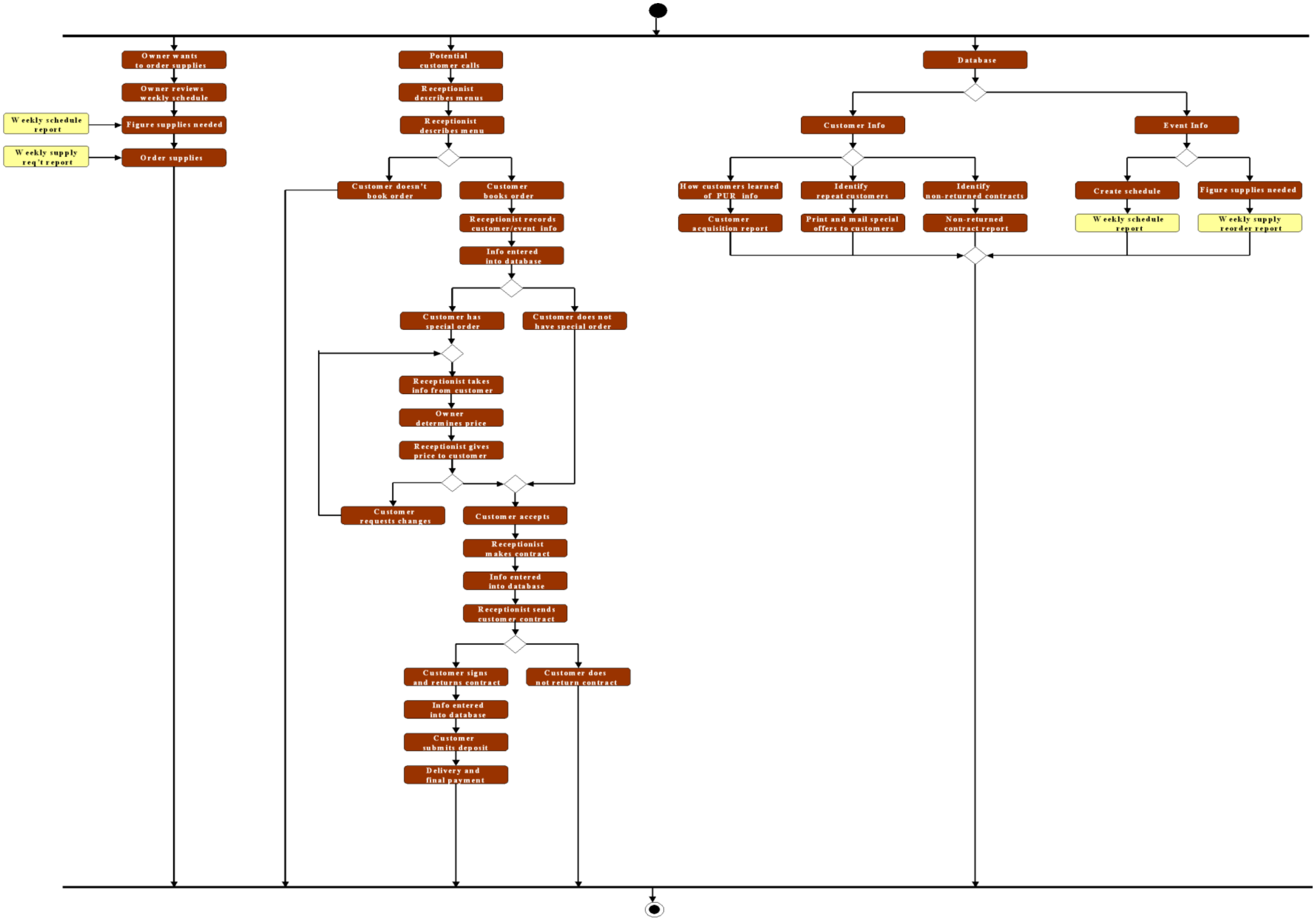
Fifth, all actors listed in a use case description must be portrayed on the use-case diagram. Furthermore, each one must have an association link that connects it to the use case and must be listed with the association relationships in the use-case description. In some organizations, we should also include the stakeholders listed in the use-case description as actors in the use-case diagram.

Sixth, all other relationships listed in a use-case description (include, extend, and generalization) must be portrayed on a use-case diagram.

Finally, there are many diagram-specific requirements that must be enforced. For example, in an activity diagram a decision node can be connected to activity or action nodes only with a control flow, and for every decision node there should be a matching merge node. Every type of node and flow has different restrictions.

R. Draw a use-case diagram and set of activity diagrams for the following system. Picnics R Us (PRU) is a small catering firm with five employees. During a typical summer weekend, PRU caters fifteen picnics with twenty to fifty people each. The business has grown rapidly over the past year and the owner wants to install a new computer system for managing the ordering and buying process. PUR has a set of ten standard menus. When potential customers call, the receptionist describes the menus to them. If the customer decides to book a picnic, the receptionist records the customer information (e.g., name, address, phone number, etc.) and the information about the picnic (e.g., place, date, time, which one of the standard menus, total price) on a contract. The customer is then faxed a copy of the contract and must sign and return it along with a deposit (often a credit card or by check) before the picnic is officially booked. The remaining money is collected when the picnic is delivered. Sometimes, the customer wants something special (e.g., birthday cake). In this case, the receptionist takes the information and gives it to the owner who determines the cost; the receptionist then calls the customer back with the price information. Sometimes the customer accepts the price, other times, the customer requests some changes that have to go back to the owner for a new cost estimate. Each week, the owner looks through the picnics scheduled for that weekend and orders the supplies (e.g., plates) and food (e.g., bread, chicken) needed to make them. The owner would like to use the system for marketing as well. It should be able to track how customers learned about PUR, and identify repeat customers, so that PUR can mail special offers to them. The owner also wants to track the picnics on which PUR sent a contract, but the customer never signed the contract and actually booked a picnic.


PRU System
Customer
Receptionist
Owner
Calls in order
<<Extend>>
Describes menus
<<Extend>>
Receives menu descriptions
<<Extend>>
Assist customer decision making
<<Extend>>
Decides to book picnic
<<Extend>>
Provides customer info
<<Extend>>
Records customer info on contract
<<Extend>>
Provides event info
<<Extend>>
Records event data
<<Extend>>
Provides special product info
<<Extend>>
Processes special product request
<<Includes>>
Sends request to owner
<<Extend>>
Receives special product info
<<Extend>>
Determines cost of product
<<Extend>>
Sends cost to receptionist
<<Extend>>
Receives special product cost
<<Extend>>
Contacts customer
<<Extend>>
Gives special product cost
<<Extend>>
Feedback about special product
<<Extend>>
Fax contract to customer
<<Extend>>
Receives contract
<<Extend>>
Signs contract
<<Extend>>
Fax contract to PRU
<<Extend>>
Pays deposit
Reads weekly orders
<<Extend>>
Purchase supplies
Track how customers learn of PRU
Identify repeat customers
<<Extend>>
Mail repeat customers special offer
ID customers who didn't return contract


Owner wants to order supplies
Owner reviews weekly schedule
Weekly schedule report
Figure supplies needed
Weekly supply req't report
Order supplies
Potential customer calls
Receptionist describes menus
Receptionist describes menu
Customer doesn't book order
Customer books order
Receptionist records customer/event info
Info entered into database
Customer has special order
Customer does not have special order
Receptionist takes info from customer
Owner determines price
Receptionist gives price to customer
Customer requests changes
Customer accepts
Receptionist makes contract
Info entered into database
Receptionist sends customer contract
Customer signs and returns contract
Customer does not return contract
Info entered into database
Customer submits deposit
Delivery and final payment
Database
Customer Info
Event Info
How customers learned of PUR info
Identify repeat customers
Identify non-returned contracts
Customer acquisition report
Print and mail special offers to customers
Non-returned contract report
Create schedule
Figure supplies needed
Weekly schedule report
Weekly supply reorder report

S. Create a set of use-case descriptions for the system in Exercise R.

Major use cases are listed by actor:

PRU

- Take orders
- Buy supplies
- Cater picnic (size: 20-50)
- Collect payment (minus deposit)
- Perform marketing tasks

Receiving Order Use Cases

- Customer
  - Calls in order
  - Receives menu description *(see Receptionist activity)*
  - Decides whether to books picnic
  - Provides customer information (*see Receptionist activity)*
  - Provides picnic information (*see Receptionist activity)*
  - Provides special product information (*see Receptionist activity)*
  - Receives fax of contract
  - Signs contract
  - Faxes contract to PRU
  - Pays picnic deposit
    - Faxes credit card info
    - Sends check

- Receptionist
  - Describes menus (10 standard ones)
  - Assists customer decision making (or waits)
  - Records customer information (on contract)
  - Records picnic information (on contract)
  - Process special product *(may go through iterations)*
    - Gives special product information to owner *(see owner activity*)
      - Receives special product cost information from owner
      - Calls the customer back
      - Provides special product cost information to client
      - Client provides feedback regarding special product cost information
      - Faxes contract to customer

- Owner
  - Receives special product information
  - Determines cost of special product
  - Sends special product cost information to receptionist

Buying Supplies Use Cases

- Owner
    - Reads weekly orders
    - Purchases materials
        - Purchases supplies
        - Purchases food

Marketing Use Cases

- Owner
    - Track customers learning about PRU
    - Identify repeat customers
    - Mail repeat customers special offers
    - Identify customers who received but did not sign and return contracts

T. Perform a verification and validation walkthrough of the functional models of the catering system described in exercises R and S.

In order to verify and validate the functional model, a set of rules have been defined to ensure the consistency among the activity diagram (described in R), the use-case description (described in S), and the use-case diagram (defined in R).

First, when comparing an activity diagram to a use-case description, there should be at least one event recorded in the normal flow of events, subflows, or alternate/exceptional flows of the use-case description for each activity or action that is included on an activity diagram, and each event should be associated with an activity or action.

Second, all objects portrayed as an object node in an activity diagram must be mentioned in an event in the normal flow of events, subflows, or alternate/exceptional flows of the use-case description.

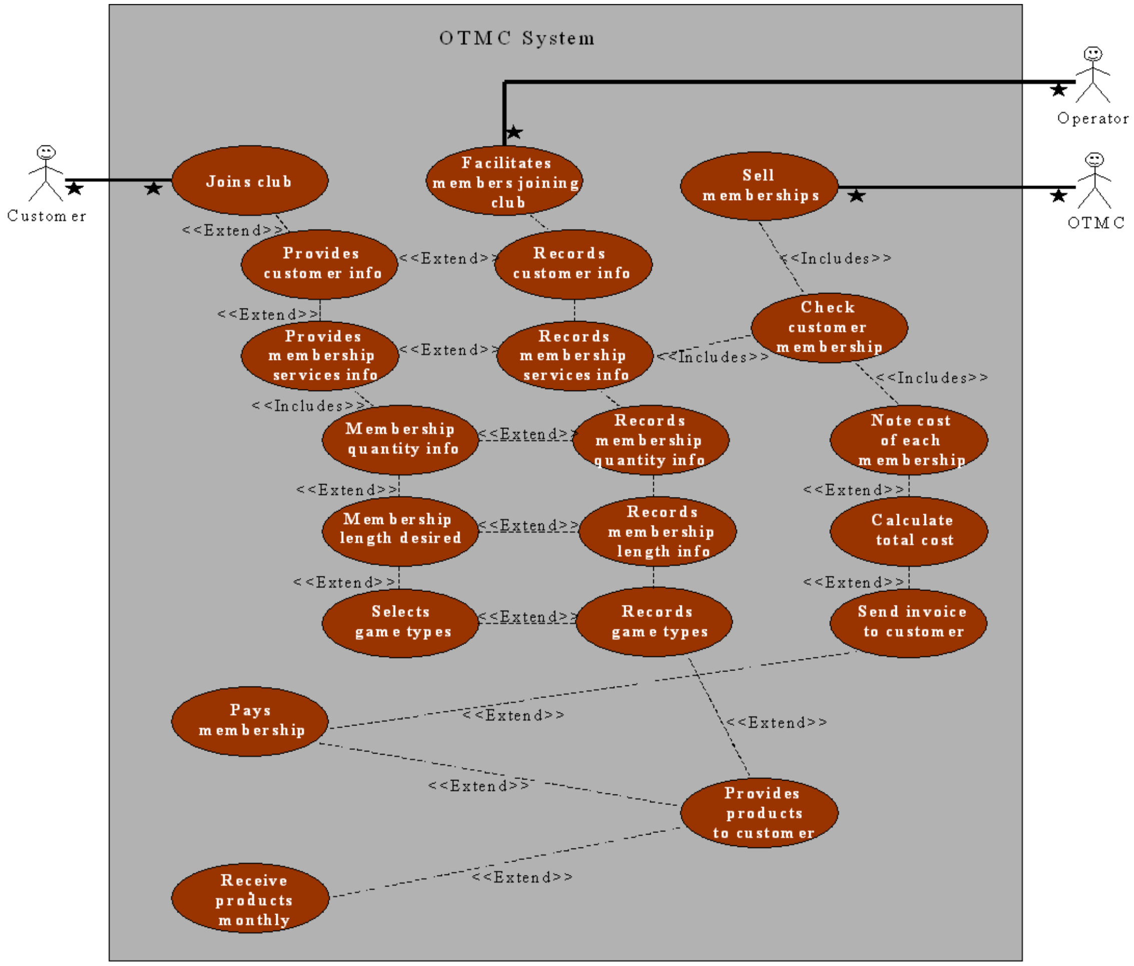
Third, sequential order of the events in a use-case description should occur in the same sequential order of the activities contained in an activity diagram.

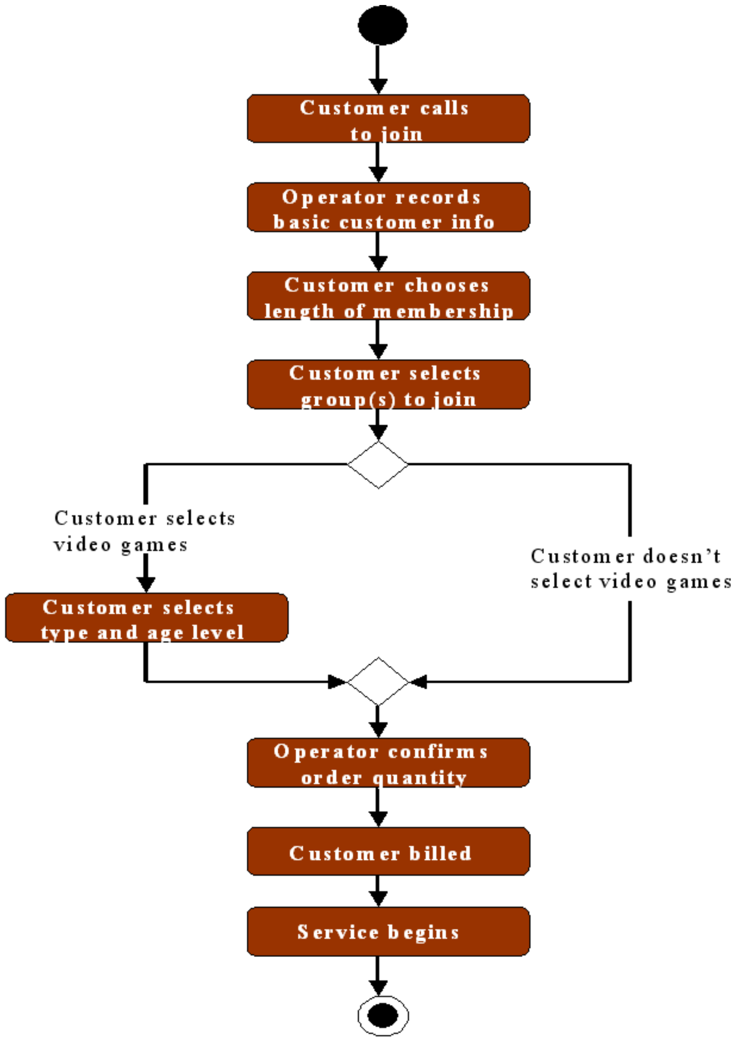
Fourth, when comparing a use-case description to a use-case diagram, there must be one and only one use-case description for each use case, and vice versa.

Fifth, all actors listed in a use case description must be portrayed on the use-case diagram. Furthermore, each one must have an association link that connects it to the use case and must be listed with the association relationships in the use-case description. In some organizations, we should also include the stakeholders listed in the use-case description as actors in the use-case diagram.

Sixth, all other relationships listed in a use-case description (include, extend, and generalization) must be portrayed on a use-case diagram.

U. Draw a use-case diagram and a set of activity diagrams for the following system. Of-the-Month Club (OTMC) is an innovative young firm that sells memberships to people who have an interest in certain products. People pay membership fees for one year and each month receive a product by mail. For example, OTMC has a coffee-of-the-month club that sends members one pound of special coffee each month. OTMC currently has six memberships (coffee, wine, beer, cigars, flowers, and computer games) each of which costs a different amount. Customers usually belong to just one, but some belong to two or more. When people join OTMC, the telephone operator records the name, mailing address, phone number, e-mail address, credit card information, start date, and membership service(s) (e.g., coffee). Some customers request a double or triple membership (e.g., two pounds of coffee, three cases of beer). The computer game membership operates a bit differently from the others. In this case, the member must also select the type of game (action, arcade, fantasy/ science-fiction, educational, etc.) and age level. OTMC is planning to greatly expand the number of memberships it offers (e.g., video games, movies, toys, cheese, fruit, and vegetables) so the system needs to accommodate this future expansion. OTMC is also planning to offer 3-month and 6-month memberships.


OTMC System
Operator
Customer
OTMC
Joins club
Facilitates members joining club
Sell memberships
<<Extend>>
Provides customer info
<<Extend>>
Records customer info
<<Includes>>
<<Extend>>
Check customer membership
Provides membership services info
<<Extend>>
Records membership services info
<<Includes>>
<<Includes>>
<<Includes>>
Membership quantity info
<<Extend>>
Records membership quantity info
Note cost of each membership
<<Extend>>
<<Extend>>
Membership length desired
<<Extend>>
Records membership length info
Calculate total cost
<<Extend>>
<<Extend>>
Selects game types
<<Extend>>
Records game types
Send invoice to customer
Pays membership
<<Extend>>
<<Extend>>
<<Extend>>
Provides products to customer
Receive products monthly
<<Extend>>


Customer calls to join
Operator records basic customer info
Customer chooses length of membership
Customer selects group(s) to join
Customer selects video games
Customer doesn't select video games
Customer selects type and age level
Operator confirms order quantity
Customer billed
Service begins

V. Create a set of detail use-case descriptions for the system in Exercise R.

Major use cases are listed by actor:

Note that many details of the case are values (such as the specific clubs) which should be captured in database or tables as these may change over time. An interesting question to pose to students will be what to do with the parts of the system pertaining to computer games if these change from selling them as products on CD to providing access for a period of time to a web-site.

OTMC

- Sell memberships
- Provide products to members (likely outside scope of the present system)

Customer

- Joins club
  - Provides customer information
  - Provides membership services information (e.g. which clubs joined)
    - (For each service) Provides multiple membership information
    - (For each service) Provides membership length desired
    - (For computer game membership) Selects game type(s)
- Pay membership
  - Check customer memberships
  - Note cost of each membership
  - Calculate total costs
  - Send invoice to customer
- Receive products (monthly)

- Telephone Operator
  - Facilitates customer joining club
    - Receives customer information
    - Receives membership services information
      - (For each service) Receives multiple membership information
      - (For each service) Receives membership length desired information
      - (For each computer game membership) Receives game type information

Note: Items in red are part of the "to-be" system or may be stubs programmed for future expansion. It is a business decision whether to prepare for the business policy changes in the new system or to wait for later updates. This presents an interesting discussion to have with students regarding how they want to handle future anticipated features which may or may not be realized by the organization. My own inclination in this case would be to include them in the design, but this is most likely something that business sponsors may have to decide.

W. Perform a verification and validation walkthrough of the functional models of the Of-the-Month Club system described in exercises U and V.

In order to verify and validate the functional model, a set of rules have been defined to ensure the consistency among the activity diagram (described in U), the use-case description (described in V), and the use-case diagram (defined in U).

First, when comparing an activity diagram to a use-case description, there should be at least one event recorded in the normal flow of events, subflows, or alternate/exceptional flows of the use-case description for each activity or action that is included on an activity diagram, and each event should be associated with an activity or action.

Second, all objects portrayed as an object node in an activity diagram must be mentioned in an event in the normal flow of events, subflows, or alternate/exceptional flows of the use-case description.

Third, sequential order of the events in a use-case description should occur in the same sequential order of the activities contained in an activity diagram.

Fourth, when comparing a use-case description to a use-case diagram, there must be one and only one use-case description for each use case, and vice versa.

Fifth, all actors listed in a use case description must be portrayed on the use-case diagram. Furthermore, each one must have an association link that connects it to the use case and must be listed with the association relationships in the use-case description. In some organizations, we should also include the stakeholders listed in the use-case description as actors in the use-case diagram.

Sixth, all other relationships listed in a use-case description (include, extend, and generalization) must be portrayed on a use-case diagram.