IN2013: Object – Oriented Analysis and Design

Tutorial 2: Use Cases

Model Answer

This tutorial concerns use case modelling. The assignment is designed to take longer than an hour to complete. I would like you to try your best in class but be aware that it is likely that you may need to spend extra effort to complete the assignment at home, after the end of the Tutorial session.

You might start with a pencil and paper solution and proceed to entering the diagram into Visual Paradigm after obtaining a good sketch of what you consider a correct solution.

You can work alone or form a small team of 2-3 people with your colleagues from the same tutorial group. Group discussions of the possible solutions while you produce them, will bring additional benefits. Each member of the team must strive to develop their own solution.

Consider the BAPERS case study (provided in Attachment 1).

Exercise 1:

Create a use-case diagram for the full system described in the scenario.

1. Identify the actors. Does the case study offer scope for Time as an actor? Discuss if the 'automatic cues' should be modelled with an actor Time as a primary actor.

Model Answer:

a) Primary actors (the roles are listed):

- Office manager
- Shift manager
- Receptionist, and
- Technician

b) Secondary actors:

- *Payment processor*

Note that Customer is not listed among the primary actors although all jobs get delivered to the Lab by customers. Remember, actors interact ***directly*** *with the system, i.e., they are users of the system (primary actors), or the system uses them to complete a use case (secondary actors). The case study clearly states that all interactions with BAPERS are undertaken by staff members. Although Customers are indirectly involved in the process of recording jobs/tasks with BAPPERS – they provide the details to be entered into BAPERS – they themselves do not use the system, hence they are not actors.*

Time ***is not explicitly listed as an actor****, although one may develop an argument for Time to be an actor. The cues mentioned in the case study are meant to be raised when*

deadlines for jobs are approaching. A plausible interpretation of the case study would be for a cue to be triggered when the deadline for a job to be completed is about to expire (say 10 min in advance of the deadline). The model presented here is developed under the assumption that every time a Shift/Office manager logs in the system, the system will check if any of the jobs are about to expire and will raise the set cues then, i.e., at the time the Manager logs in. Clearly, this model is limited – since the checks are part of the Login use case, Managers are never going to see the cues if they either do not login or stay logged in all the time.

Develop an alternative model, in which Time is modelled as a primary actor for the use case ShowCue, instead of the use case being an extension for Login. Three aspects must be taken into account in this model refinement:

- *There will be a secondary actor – Shift/Office Manager which will be involved in the ShowCue use case.*
- *One should put in the preconditions of the refined ShowCue that the respective secondary actor must have logged in. If the secondary actor is not logged in, they will not be able to see the cue.*
- *You must decide what the policy is of raising cues: either raise it once or raise it periodically until the job has been completed (or even after the deadline if the job remains not completed after the deadline).*

2. Identify the use-cases.

Model Answer:

a) Login

b) Logout

c) ShowCue

d) ViewProgress

e) PlaceOrder – this is really a Job creation use case, which may require searching for an existing account and/or creating a new Customer Account.

f) SearchCustomer – an extension of PlaceOrder. This may if the Customer does not remember if they have an account with the Lab. If the customer, however, remember that they have an account and provide the account details, then search may not be needed.

g) CreateCustomerAccount – an extension of PlaceOrder.

h) TakePayment with two specialised use cases, TakeCashPayment and TakeCardPayment.

i) ProcessTask with two specialised use cases, StartTask and CompleteTask, the latter may be extended by CompleteJob when the last task of the job is completed.

j) CompleteJob - an extension of CompleteTask (see above).

k) SearchJob – used when working with a task or to take payment.

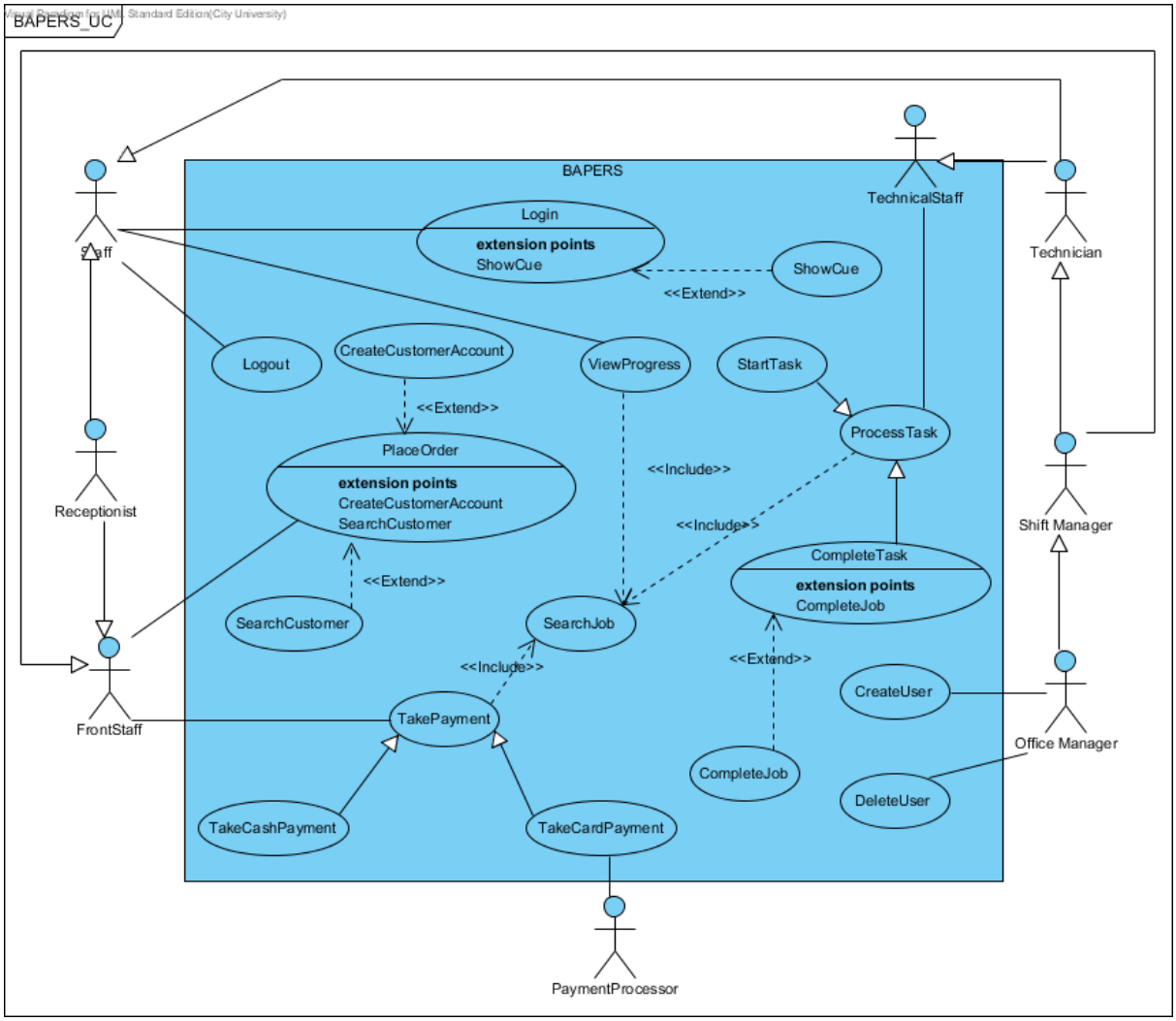
l) CreateUser

a. *DeleteUser. Note that we do not include any functionality for altering the details of an existing user, which might be useful, but is strictly not required by the case study.*

3. Construct a use-case diagram.

Model Answer:

A plausible model answer is provided in the diagram below. This diagram is included in a VP project, which is released on Moodle with this model answer.



Consider carefully the relationships shown in the diagram:

- *There is an extensive set of generalisation relationships between the actors, which capture the relationships between the roles of staff as provided in the case study description. I introduced two artificial roles: FrontStaff to deal with new jobs/user accounts and payments, and TechnicalStaff – dealing with the orders by staff working in the Lab. Finally, there is an actor Staff, who is a generalisation of all roles in "The Lab" and a primary actor for Login, Logout and ViewProgress use cases.*
- *<<include>> relationships between use cases. There are 3 instances of this relationship, all pointing to the use case SearchJob. Given the statement that there will be many (hundreds!) jobs under processing, one would consider searching for jobs as useful functionality that can be used while processing the tasks (one starts with locating the job first) and in payment collection – again one need first to search for the job(s) that are being paid for. Similarly, viewing the progress with jobs, (e.g., a phone enquiry by a customer) may require a search for a job, too.*

- *<<extend>>. Several extensions are included in the diagram. These model useful branching in the flows which may affect* ***postconditions.***

  Common mistake: *Often people who are new to use case modelling tend to us use extensions and alternative flows interchangeably. Although both extension and alternative flows capture branches in the flow and allow for different postconditions to be defined, there is clear difference in the goals (intent) with them. Extensions are used to capture useful optional branches of the use case flow and are shown in use case diagrams, while alternative flows are used to capture* ***exceptional (anomalous) branches in the flow*** *(e.g., errors of various types) and are not shown in use case diagrams. Alternative flows are only documented in use case specifications.*

- *Generalisation between use cases. There are two examples: TakePayment and Process Task.*
  - *In the first case there will be a commonality – find the job(s) being paid for, calculating the amount to pay, but clearly there will be significant differences in the interaction flows depending on the form of payment. The card payment requires interaction with the secondary actor PaymentProcessor (an external system), while cash payment requires a mere recording of the amount and the fact that the payment has been made in cash.*
  - *ProcessTask has two specialised use cases – StartTask and CompleteTask which have functionality in common – e.g., searching for the job to which the particular task belongs – but also significant differences. StartTask will require recording who is working on the task, clocking the time when the work starts, moving the material from a location (on shelf) to a Room, while completing the task will require clocking the completion time and the new location of the materials. Importantly, CompleteTask may be extended – if the completed task is the last task in the particular job: in this case completing the task would lead to an extension CompleteJob (i.e., the job is marked as completed).*

Exercise 2:

Develop a use case specification for at least two of the major use cases identified.

Model Answer:

From the use case diagram, we establish that PlaceOrder *is associated (has <<extend>> relationships) with two other use cases* CreateCustomerAccount *and* SearchCustomer. *Below all three use cases are documented with their alternative flows.*

ID: TM02	**Use case:** PlaceOrder
Brief description: The Customer brings in photographic material to be processed in "The Lab". A new Job sheet is made out with a unique ID, which lists the tasks to be performed, sets the completion time. The job is linked to a CustomerAccount. If necessary, a new Customer Account is created.	
Primary actors: FrontStaff (Receptionist, Shift or Office Manager)	
Secondary actors:	

None

Preconditions:

1. BAPERS is operational, the primary actor has logged in.

Flow of events:

1. Actor activated the functionality for creating a new job

   Extension point: SearchCustomer

   Extension point: CreateCustomerAccount
2. Actor selects the chosen CustomerAccount.
3. The system assigns a unique ID to the Job and links the job to the chosen Customer account.
4. While there are more tasks to add to the job:
   4.1. The actor selects a task from the list of available tasks and adds it to the job.
5. The actor selects the urgency of the job (either "urgent" or "normal").
6. The system records the job completion time.
7. The system computes the price of the job.
8. The actor confirms that the job is completed.
9. The system prints a label (to be attached to the photographic material) with the job ID and a collection note[1] (with the job ID, the Customer name and the job completion time).

Postconditions:

1. The system has recorded a new job with its ID, Customer who placed it, urgency, completion time, price and list of tasks to be completed.
2. The system printed a label and a collection note.

Alternative flow:

NoCustomerAccountProvided

PrintingFailure

Alternative flow: NoCustomerAccountProvided
ID: TM02.1
Brief description: No CustomerAccount is provided.
Primary actors: FrontStaff
Secondary actors: None
Preconditions: The session does not contain a valid CustomerAccount.

[1] This note is handed over to the customer as a proof that the photographic material has been collected for processing.

Alternative flow:

2. The system prompts the actor that no valid CustomerAccount has been chosen.
3. The actor acknowledges the notification.
4. The system removes the session.

Postconditions: None.

Alternative flow: PrintingFailure
ID: TM02.2
Brief description: Printer fails to print the required document: either is out of paper or some other problem occurs.
Primary actors: FrontStaff
Secondary actors: None
Preconditions: 1. The printer is out of paper, or a technical fault has occurred.
Alternative flow: 9. The alternative flow starts after step 9 of the normal flow. 10. BAPERS displays a message: "Printer fault". 11. If further details are known about the problem The system prints further details.
Postconditions: None

Use case ID: TM03	**Use case:** CreateCustomerAccount
Primary actors: FrontStaff	
Secondary actors: None	
Segment 1 Precondition (extension point: CreateCustomerAccount**)** No account exists in BAPERS for the customer who brought in photographic material.	
Segment 1 flow: 1) Actor activates the functionality for creating new customer account. 2) The system places an empty customer record in the session. 3) The system opens a form for entering the customer details such as Name, phone number, and email address. 4) The actor types in the required details and confirms their validity. 5) The system creates a new customer record in BAPERS.	

6) If the system succeeds

The system places the customer details in the session by adding a 'current customer' record in the session.

Else

Places an empty 'current customer' record in the session.

Postcondition:

1. CustomerAccount details are stored in the 'current customer' record in the session.

Use case ID: TM04	**Use case:** SearchCustomer
Primary actors: FrontStaff	
Secondary actors: None	
Segment 1 Precondition (extension point: SearchCustomer**)** Customer chooses the CustomerAccount search functionality by activated it.	
Segment 1 flow: 1) The system offers a search form for customer to specify the search criteria (e.g. name, email, phone number, or any combination of these) 2) While ***no more than one record*** is found 2.1 The actor specifies the search criteria (any combination of the three listed above will be acceptable) and confirms the validity of the search criteria. 2.2 The system undertakes a search based on the search criteria provided. 3) If a single customer record is found The customer record is placed in the 'current customer' record in the session. Else An empty record 'current customer' is placed in the session.	
Postcondition: CustomerAccount details are stored in the 'current customer' record in the session.	

*You will have noticed that I use **session** to pass data between use cases. This is not a standard practice – the sessions, supported by some programming languages explicitly, are typically added during implementation, but in my opinion are useful even in use case specifications because they capture intuitively the notion of **data dependence** between use cases.*

In the particular set of use cases I use a session to either pass the 'current customer' details, i.e., the details of the customer who will be linked with a new job (order). In case no customer is found – no job will be created.

Another subtle point with the specifications is that with extensions I do not use alternative flows. This is consistent with all the examples in Jim Arlow's book. As a result, it may be difficult to show some important anomalies, which may occur in the extension use case flows. For instance, you can see that in the postconditions for CreateCustomerAccount I only

included what happens in the session. Clearly, in case a new CustomerAccount is successfully created (step 5 of the flow) the data held by the system will change – a new customer record will be created. However, if the customer account creation fails, no such record will be created. Normally, I would have dealt with such an exception via an alternative flow, which I tried to avoid with the specification of this extension use case.

Being faithful to the line taken by Jim's book is clearly not necessary. The UML standard does not provide any standardised way of writing use case specifications. Thus, in your work you are free to use alternative flows in the extension use cases, especially if you find it too cumbersome to avoid them.

Created on 18th September 2020

Last updated on the 12th of September 2023

Peter Popov

Appendix

Bloomsbury's Automated Process Execution Recording System (BAPERS)

The case study describes the requirements for automating the processes at the photographic laboratory 'The Lab' operated by Bloomsbury's Image Processing Laboratory (**BIPL)**. **BIPL** handles the work of professional photographers and carries out *jobs* on behalf of customers, but the customers do not use the system. The access to the system is limited to the laboratory staff only.

Each job is given a *unique identifier* and assigned *urgency* (a job may be urgent or normal which affects the turnaround time) when the job is placed. Urgent jobs must be completed within 6 hours while the normal jobs – within 24 hours. A job may consist of any number of the *standard tasks* offered by BIPL.

Currently there are around 30 standard tasks. Each task has an identifier (e.g., a number), is carried out at a particular location within the laboratory, has a quoted price, and an (estimated) duration. Some of the tasks are shown in Table 1 below. The case study ***does not require*** that the list of tasks be maintained (i.e. BAPERS does NOT offer functionality for adding new tasks, removing, or updating existing tasks).

Table 1

Task ID	Task Description	Location	Price (£)	Duration (min)
1.	Use of large copy camera	Copy Room	19.00	120
2.	Black and white film processing	Development Area	49.50	60
3.	Bag up	Packing Department	6.00	30
4.	Colour film processing	Development Area	80.00	90
5.	Colour Transparency processing	Development Area	110.30	180
6.	Use of small copy camera	Copy Room	8.30	75
7.	Mount Transparencies	Finishing Room	55.50	45
…	Etc.	Etc.	Etc.	Etc.

At any given time, hundreds of jobs will be in progress or pending within the laboratory. Every accepted job must be chargeable to a valid *customer account*, either an existing account, or a newly created account (e.g. a photographer may walk in, leave a film to be developed, and pay cash on collecting the finished job).

BIPL want to enable the employee on the reception desk to enter the job on a computer terminal. The material will be labelled with the job number and taken down to the laboratory.

The laboratory staff will interrogate **BAPERS** to ascertain the tasks required. As the job is transferred from one location to another in the laboratory, the staff responsible for each task will record its completion on a computer terminal in their location before passing it on. A terminal will be required in each of the following locations: Copy Room, Dark Room, Development Area, Printing Room, Finishing Room, and Packing Department.

Many jobs will be going through the laboratory at any given time, and confusion between them must be avoided. At all costs, loss or mistreatment of the customer's material must not occur. Queues of work may build up at the processing stations, but flexible scheduling is required to allow priority to be given to *urgent* jobs over the *regular/normal* jobs. The system should provide functionality for inspecting the list of active/pending jobs as well as already completed

ones, including the inspection of the progress of individual tasks (active/pending tasks vs. completed ones).

BAPERS must therefore provide the following main facilities:

BAP-ACCT *Accept job at reception*: Identify existing (or create new) customer account (name, phone). Assign job number. Record the deadline for completion of the job. This functionality will be mainly used by the receptionists but will also be made accessible by either Office manager or Shift manager in case receptionist(s) are absent.

BAP-PROC *Process a job through laboratory*: Respond to enquiries from any computer terminal about status of any jobs in progress, or of all jobs (including the completed ones). Update status of any given job by recording completion of current task and commencement of next (possibly with transfer of material to a new location). This functionality is available to Technicians, Shift and Office Manager.

In addition, alert Shift and/or Office manager (by, for example, displaying a visual cue with appropriate text) if the expected time to complete outstanding tasks for any job is likely to exceed the set time period, i.e. if the deadline for the job is not likely to be met; the alerts should be performed only for these two user roles.

BAP-PAYM *Payment processing*. The customers are supposed to pay once the jobs they had placed have been completed. Customers can pay by cash or credit/debit card only. The system is connected to an external payment processor to move funds between the customers' credit/debit cards and the BIPL bank account. In case of a card payment BAPERS is expected to connect to the external payment processor and on successful completion of the card payment the payment is recorded with the following details: card payment, type of card, and the last 4 digits of the card used. Cash payments are recorded, too. Several jobs can be paid at once. Only payment in full is accepted (i.e. no partial payments are allowed). This functionality is available to Receptionist, Shift and Office Manager.

BAP-ADMN *Administering the system.* This includes creating a user account for BAPERS and setting up access privileges. The following user roles are essential to be implemented: Office manager, Shift manager, Receptionist and Technician. This functionality will be available to Office Manager only.

Created: 20th September 2019

Last modified: 1st September 2025

Peter Popov