Lab.2 Functional modelling. USE CASES

Bibliography:
Tom Pender, UML Bible
http://www.agilemodeling.com
http://www.andrew.cmu.edu/course/90-754/umlucdfaq.html
http://www.gatherspace.com/static/use_case_example.html

Def. (Wikipedia): A **use case** in software engineering and systems engineering is a description of a system's behaviour as it responds to a request that originates from outside of that system.

A use case defines the interactions between external actors and the system under consideration to accomplish a goal.

Def. **Actor** specifies a role[1] played by a person or a thing (another system, a device) when interacting with the system.

Actors exist *outside the system* under study, and they take part in a sequence of activities in a dialogue with the system to achieve some goal.

A **use case analysis** is the primary form for gathering usage requirements for a new software program or task to be completed.

The primary goals of a use case analysis are:
- designing a system from the user's perspective,
- communicating system behavior in the user's terms,
- specifying all externally visible behaviors.

A more specific set of goals for a use case analysis is to clearly communicate:
- system requirements,
- how the system is to be used,
- the roles the user plays in the system,
- what the system does in response to the user stimulus,
- what the user receives from the system,
- what value the customer or user will receive from the system.

A use case **SHOULD:**

- Describe what the **system shall do** for the actor to achieve a particular goal.
- Include no implementation-specific language.
- Be at the appropriate level of detail.
- Possible include an initial UI (user interface) prototype. (Further details regarding user interfaces and screens are represented later in the software development life cycle, during the activity of UI design – lab.4)

[1] **Obs**. The same person can use the system as different actors because at different time moments the person can play different roles. For example, "Joe" could be playing the role of a Customer when using an Automated Teller Machine to withdraw cash, or playing the role of a Bank Teller when using the system to restock the cash drawer.

Use cases may be described at the abstract level (business use case, sometimes called essential use case), or at the system level (system use case). The differences between these is the scope.

- The **business use case** is described in technology free terminology which treats the business process as a black box and describes the business process that is used by its business actors (people or systems external to the business) to achieve their goals (e.g., manual payment processing, expense report approval, manage corporate real estate). The business use case will describe a process that provides value to the business actor, and it describes *what* the process does. Business Process Modeling is another method for this level of business description.
- The **system use cases** are normally described at the system functionality level (for example, create voucher) and specify the function or the service system provides for the user. A system use case will describe *what* the actor achieves interacting with the system. For this reason it is recommended that a system use case specification begin with a verb (**e.g., *create* voucher, *select* payments, *exclude* payment, *cancel* voucher**). Generally, the actor could be a human user or another system interacting with the system being defined.

A. USE CASE DIAGRAM (UCD)

1. Introduction

Goal : (WHAT MUST THE SYSTEM DO!!!)
Describes the **functionality** of the system from the **user point of view**.
A top-down, horizontal perspective.
A high level of abstraction.

UCD - Levels of granularity:

- entire enterprise or line of business
- individual system
- single sub-system or component.

Goal-focused modelling:

- the purpose (features, functionality) of the system (WHAT).
- NOT solutions to achieve the goal (not HOW).

2. MODELING ELEMENTS

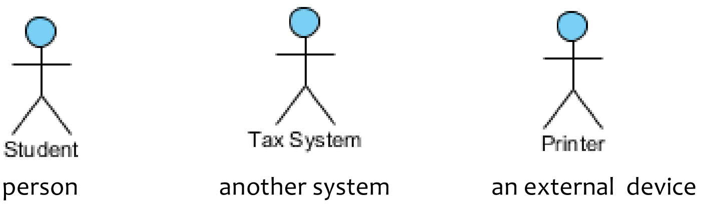
Actor – entity that interacts with the system; role played by a person, another system or a hardware device.
System – the sistem that is modeled.
Use case – a service the system knows to perform; key function of the system.
Relation types:

- *association* – interaction between actor and use-case.

- *<<include>>* – between two use-cases; the included UC is a reusable use-case that is unconditionally incorporated into the execution of the including UC; the including (calling) UC decides when and why to use the included UC.
- *<<extend>>* – between two use-cases; the extending UC is a reusable use-case that conditionally interrupts the execution of the extended UC to augment its functionality; the extending UC decides when it should be used.
- generalization – inheritance relationship between actors or between use-cases.

3. NOTATION

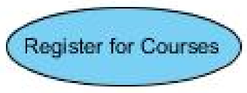
3.1 Actor:



a person another system an external device

Actors are always involved with at least one use case and are always drawn on the outside edges of a use case diagram.

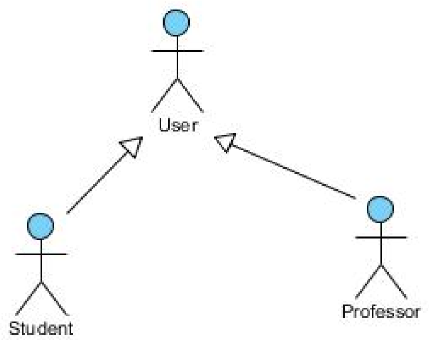
3.2 Use-case:



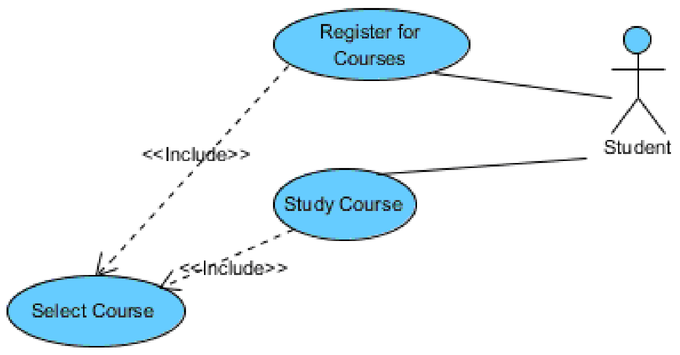
Begin the name of the use-case with a verb !.

3.3 Relationships:

generalization

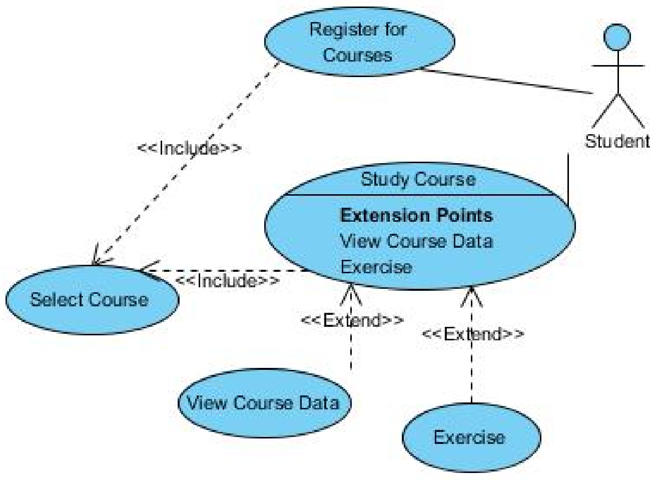


<<include>>



"Select Course" will be executed each time "Register for Courses" or "Study Course" will be started.

<<extend>>



"View Course Data" UC will be executed only in the extension point "View Course Data" of the "Study Course" UC. The extension point can be activated by pressing a button or by fulfilling another condition.
Similar for "Exercise" UC.

DISCUSSION:

An extending use case continues the behavior of a base use case. The extending use case accomplishes this by conceptually inserting additional action sequences into the base use-case sequence. This allows an extending use case to continue the activity sequence of a base use case when the appropriate extension point is reached in the base use case and the extension condition is fulfilled. When the extending use case activity sequence is completed, the base use case continues.

An extending use case is, effectively, an alternate course of the base use case. In fact, a good rule of thumb is you should introduce an extending use case whenever the logic for an alternate course of action is at a complexity level similar to that of your basic course of action. An extending use case can also be introduced whenever an alternate course for an alternate course is needed; in this case, the extending use case would encapsulate both alternate courses.

Applying an «extend» relationship requires four elements:

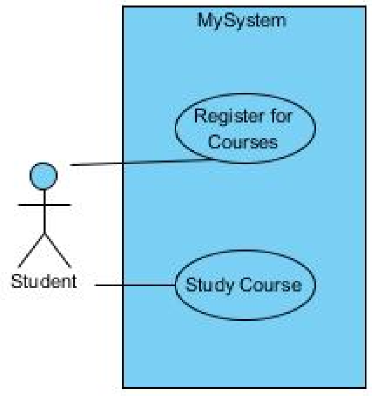
- The base use case: The use case that will be augmented by the extension use case (the "Study Course" use case, for example).
- The extension use case: The use case that provides the added behavior ("View Course Data" and "Exercise" are extension uses cases in this example).
- The «extend» relationship: A dashed arrow with the base attached to the extension use case and the arrow attached to the base use case.
- Extension points: One or more locations in the base use case where a condition is evaluated to determine whether the extension should interrupt the base use case to execute. The extension points may be listed within the use case icon or simply identified within the use case narrative.

The extension point is a condition that determines whether the extension should be used. There is no such condition in an «include» relationship. The extension point defines what the extension use case is watching for in order to know when it needs to insert itself into the executing use case.

<<Include>>	**<<Extend>>**
Augments the behavior of the base use case.	Augments the behavior of the base use case.
The included use case *is always used* to augment the executing use case.	The extension use case *might be used* (*sometimes*) to augment the executing use case.
The *executing use case* decides when to call the included use case. The included use case is unaware of the base use case.	The *extension use case* decides when it will insert itself into the execution of the base use case. The base use case is unaware of the extension.
The relationship arrow is drawn from the executing use case to the included use case. The base of the arrow indicates that the base use case directs the included use case to execute.	The relationship arrow is drawn from the extension use case to the executing use case. The base of the arrow indicates that the extension use case is making the decision whether to interrupt the executing use case.

3.4 System boundary boxes (optional).

You can draw a rectangle around the use cases, called the system boundary box, to indicate the scope of your system. Anything within the box represents functionality that is in scope and

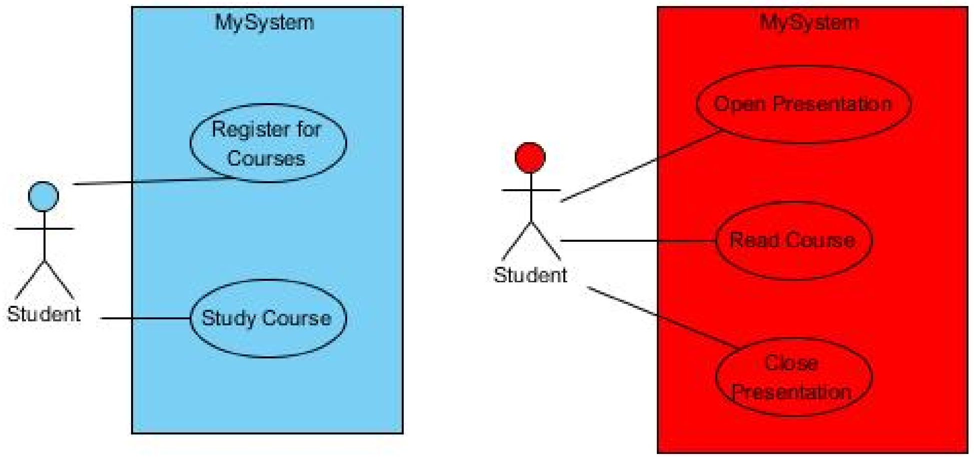


anything outside the box is not in the scope.

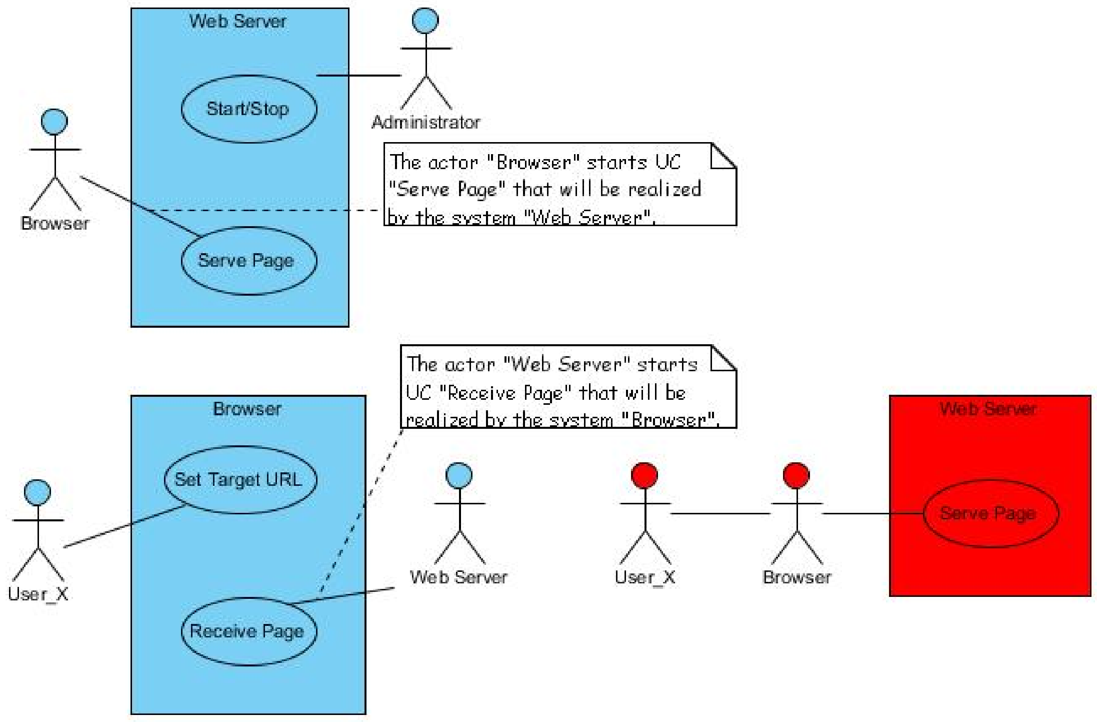
System boundary can also be used to identify which use cases will be delivered in each major release of a system. Next figure shows how this could be done.

4. RULES

1. A UC defines a service (in a single session/request) not a sequence of operations.



2. Identify (only one) system with which all actors communicate. Actors do not inter-communicate directly.


Web Server
Start/Stop
Serve Page
Administrator
Browser
The actor "Browser" starts UC "Serve Page" that will be realized by the system "Web Server".
Browser
Set Target URL
Receive Page
User_X
The actor "Web Server" starts UC "Receive Page" that will be realized by the system "Browser".
Web Server
User_X
Browser
Web Server
Serve Page

3. Identify all of the top-level behavior – the things the system knows to do WHAT !!! (not how).

4. Analyze <<include>> vs. <<extend>>

X include(uses) Y ⇒ X ***allways*** involves doing Y at least once.

Y extends X ⇒ Y is a special case behavior of the same type as the more general Y; will be executed ***sometimes***.

B. USE CASE NARATIVE

UC diagram – illustrates actors and UC-s as pieces of behavior, and their relationships, with no details on behavior.

UC narrative : document that explains that UC is a behavior of the system, having a beginning (trigger), a middle (dialog) and an end (termination).

UC narative represents a complete series of events, described from the point of view of the Actor.

The narative[2] describes how the Actor will interact with the system to achieve a specific goal.

One or more scenarios may be generated from a use case, corresponding to the detail of each possible way of achieving that goal.

The elements of a Use Case narrative often include:

1. Name of the use case
2. Version
3. Goal
4. Summary
5. Actors involved
6. Preconditions
7. Triggers
8. Basic flow of events
9. Alternate flows of events
10. Exceptions that can occur
11. Extension points
12. Postconditions
13. Business rules
14. Notes
15. Author and Date

[2] Recommendation on language: avoid technical jargon; prefer the language of the end user or domain expert.

There is no standard template for documenting detailed use cases. There are a number of competing schemes, and individuals are encouraged to use templates that work for them or the project they are on. Standardization within each project is more important than the detail of a specific template. There is, however, considerable agreement about the core sections; beneath differing terminologies and orderings there is an underlying similarity between most use cases. Different templates often have additional sections, e.g., assumptions, exceptions, recommendations, technical requirements. There may also be industry specific sections.

Use case name

A use case name provides a unique identifier for the use case. It should be written in verb-noun format (e.g., *Borrow Books*, *Withdraw Cash*), should describe an ***achievable goal*** (e.g., *Register User* is better than *Registering User*) and should be sufficient for the end user to understand what the use case is about.

Goal-driven use case analysis will name use cases according to the actor's goals, thus ensuring use cases are strongly user centric. Two to three words is the optimum. If more than four words are proposed for a name, there is usually a shorter and more specific name that could be used.

Version

Often a version section is needed to inform the reader of the stage a use case has reached. The initial use case developed for business analysis and scoping may be very different from the evolved version of that use case when the software is being developed. Older versions of the use case may still be current documents, because they may be valuable to different user groups.

Goal

Without a goal a use case is useless. There is no need for a use case when there is no need for any actor to achieve a goal. A goal briefly describes what the user intends to achieve with this use case.

Summary

A summary section is used to capture the essence of a use case before the main body is complete. It provides a quick overview, which is intended to save the reader from having to read the full contents of a use case to understand what the use case is about. Ideally, a summary is just a few sentences or a paragraph in length and includes the goal and principal actor.

Actors involved

An actor is someone or something outside the system that either acts on the system – a *primary actor* – or is acted on by the system – a *secondary actor*. An actor may be a person, a device, another system or sub-system, or time. Actors represent the different roles that something outside has in its relationship with the system whose functional requirements are being specified. An individual in the real world can be represented by several actors if he has several different roles and goals in regards to a system. These interact with system and do some action on that.

Preconditions

A *preconditions* section defines all the conditions that must be true (i.e., describes the state of the system) for the *trigger* (see below) to meaningfully cause the initiation of the use case. That is, if the system is not in the state described in the preconditions, the behavior of the use case is indeterminate. Note that the preconditions are *not* the same thing as the "trigger" (see below): the mere fact that the preconditions are met does NOT initiate the use case.

However, it is theoretically possible *both* that a use case should be initiated whenever condition X is met *and* that condition X is the only aspect of the system that defines whether the use case can meaningfully start. If this is really true, then condition X is *both* the precondition and the trigger, and would appear in both sections. But this is *rare*, and the analyst should check carefully that they have not overlooked some preconditions which are part of the trigger. If the analyst has erred, the module based on this use case will be triggered when the system is in a state the developer has not planned for, and the module may fail or behave unpredictably.

Triggers

A 'triggers' section describes the event that causes the use case to be initiated. This event can be external, temporal or internal. If the trigger is not a simple true "event" (e.g., the customer presses a button), but instead "when a set of conditions are met", there will need to be a triggering process that continually (or periodically) runs to test whether the "trigger conditions" are met: the "triggering event" is a signal from the trigger process that the conditions are now met.

There is varying practice over how to describe what to do when the trigger occurs but the preconditions are not met.

- One way is to handle the "error" within the use case (as an exception). Strictly, this is illogical, because the "preconditions" are now not true preconditions at all (because the behavior of the use case is determined even when the preconditions are not met).
- Another way is to put all the preconditions in the trigger (so that the use case does not run if the preconditions are not met) and create a different use case to handle the problem. Note that if this is the local standard, then the use case template theoretically does not need a preconditions section!

Basic flow of events

At a minimum, each use case should convey a *primary scenario*, or typical course of events, also called "basic flow", "happy flow" and "happy path". The main basic course of events is often conveyed as a set of usually numbered steps.
For example: The system prompts the user to log on, The user enters his name and password, The system verifies the logon information, and The system logs user on to system.

Alternative flows of events

Use cases may contain secondary paths or alternative scenarios, which are variations on the main theme. Each tested rule may lead to an alternative path and when there are many rules the permutation of paths increases rapidly, which can lead to very complex documents. Sometimes it is better to use conditional logic or activity diagrams to describe use case with many rules and conditions.
Exceptions, or what happens when things go wrong at the system level, may also be described, not using the alternative paths section but in a **section of their own**.
Alternative paths make use of the numbering of the basic course of events to show at which point they differ from the basic scenario, and, if appropriate, where they rejoin. The intention is to avoid repeating information unnecessarily.
An **activity diagram** can give an overview of the basic path and alternative path.

Postconditions

The *post-conditions* section describes what the change in state of the system will be after the use case completes. Post-conditions are guaranteed to be true when the use case ends.

Business rules

Business rules are written (or unwritten) rules or policies that determine how an organization conducts its business with regard to a use case. Business rules are a special kind of requirement. Business rules may be specific to a use case or apply across all the use cases, or across the entire business. Use cases should clearly reference business rules that are applicable and where they are implemented.
Business Rules should be encoded in-line with the Use Case logic and execution may lead to different post conditions. E.g. Rule2: “a cash withdraw will lead to an update of the account and a transaction log” leads to a post condition on successful withdrawal - but only if Rule1: “there must be sufficient funds” tests as true.

Notes

Experience has shown that however well-designed a use case template is, the analyst will have some important information that does not fit under a specific heading. Therefore all good templates include a section that allows less-structured information to be recorded.

Author and date

This section should list when a version of the use case was created and who documented it. It should also list and date any versions of the use case from an earlier stage in the development which are still current documents. The author is traditionally listed at the bottom, because it is not considered to be essential information; use cases are intended to be collaborative endeavors and they should be jointly owned.

C. METHODOLOGY

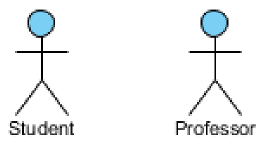
To write effective use cases, be productive without perfection

When it comes to writing effective use cases, you don't need to be a perfectionist and concern yourself with getting it right the first time. Developing use cases should be looked at as an iterative process where you work and refine. You can always refine it later.

1. STEPS TO DEVELOP A USE-CASE DIAGRAM (UCD):

1. Define the context of the system: identify the actors and their responsibilities.
2. Identify the use-cases: the functions of the system in terms of specific goals and/or results that must be produced.
3. Evaluate the actors and use-cases to find opportunities for refinement (such as splitting or merging definitions).
   a. Evaluate use-cases to find <<include>> relationships.
   b. Evaluate use-cases to find <<extend>> relationships.
   c. Evaluate the actors and use-cases for generalization opportunities (shared properties).

1.1 Identifying actors.



Identify the representative ROLES (not individual people !).

1.2. Identifying use-cases.

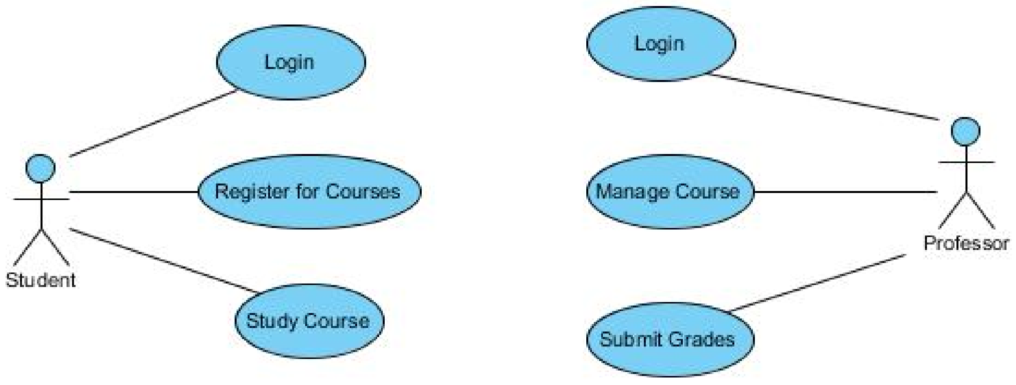
Procedure:

Identify potential services by asking your stakeholders the following questions from the point of view of each actor:

- What are users in this role trying to accomplish?
- To fulfill this role, what do users need to be able to do?
- What are the main tasks of users in this role?

- What information do users in this role need to examine, create, or change?
- What do users in this role need to be informed of by the system?
- What do users in this role need to inform the system about?

Outline the goals of each actor. (Once you have established your actors and the goals of each actor, you have now created your initial list of high level use cases).

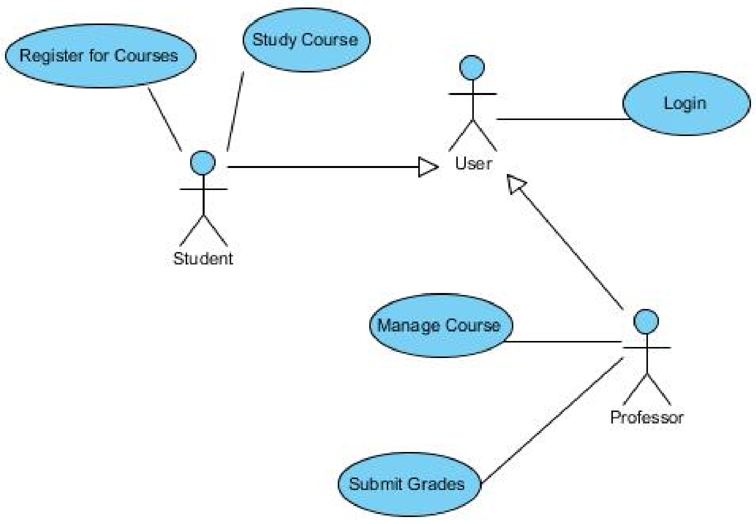


1.3. Evaluate the actors and use-cases to find opportunities for refinement

1.3.c. Identify reuse opportunity for use cases

In the first version of the use case diagram in our example there is duplicate behavior in both the Student and Professor which includes "Login". We can introduce a more general user that has this behavior and then the actors will "inherit" this behavior from the new user.

The next use case diagram illustrates that a generic user logs in and that a student and a professor have their own behavior but also have the behavior of the generic user.



The benefits of generalization are that duplicate behavior and attributes are eliminated, that will ultimately make the system more understandable and flexible.

The inheritance can be applied both to use cases and to the actrs.

1.3.a. Evaluate use-cases to find <<include>> relationships

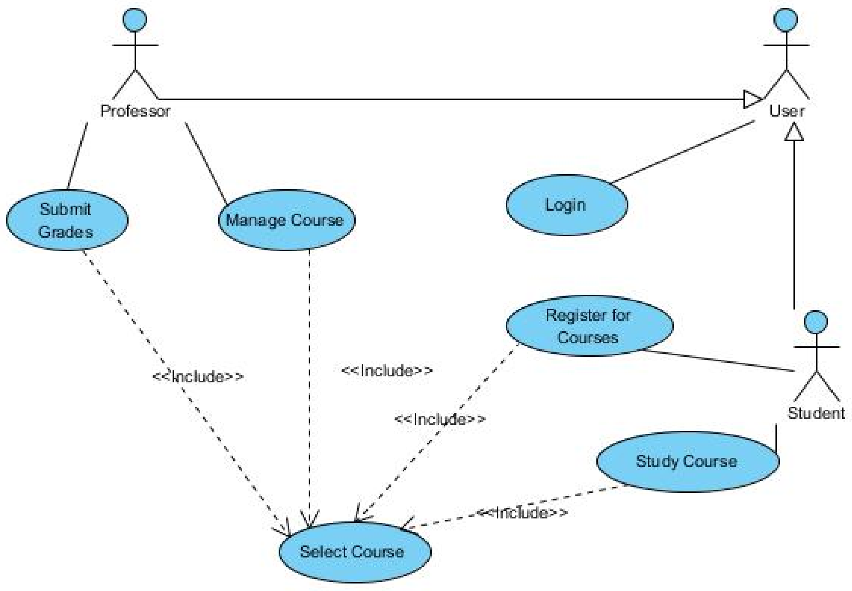
In order to work with a course, both professor and student must first select one from all the courses they teach or assist respectively.

We can consider that selecting a course is complex enough to become a use case.

Selecting a course must be done for either submitting grades, managing a course, registering for courses or studying a course.

Consequently, "Select Course" use case will be always executed to support "Submit Grades", "Manage Course", "Register for Courses" or "Study Course" use cases.

This is represented by <<include>> relationships on the UC diagram as follows:

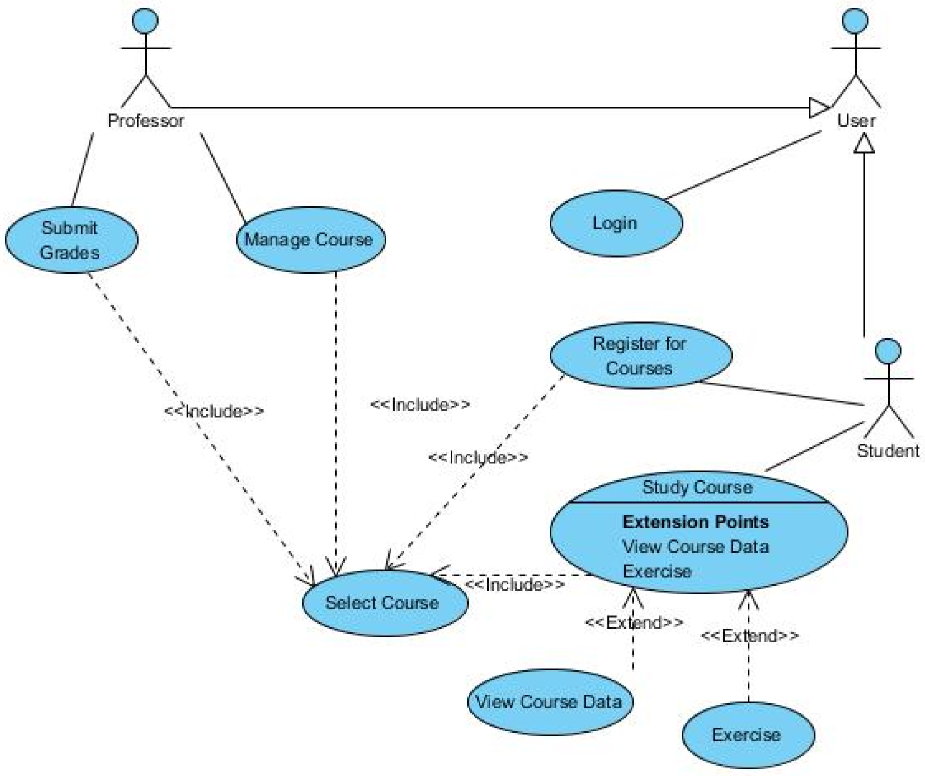


1.3.b. Evaluate use-cases to find <<extend>> relationships

When studying for a course the student can either read course documentation or exercise using homework and tests provided for the course.

Consequently, either "View Course Data" or "Exercise" use case will be executed, according to the selection of the student.

This is represented by <<extend>> relationships on the UC diagram as follows:



2. STEPS TO DEVELOP A USE-CASE NARATIVE:

2.1. Identify the key components of the use case

The actual use case is a textual representation illustrating a sequence of events. The table below gives a basic understanding of what is in the use case.

Use Case Element	Description
Use Case Number	ID to represent your use case
Application	What system or application does this pertain to
Use Case Name	The name of your use case, keep it short
Use Case Description	Elaborate more on the name, in paragraph form.
Primary Actor	Who is the main actor that this use case represents
Precondition	What preconditions must be met before this use case can start
Trigger	What event triggers this use case
Basic Flow	The basic flow should be the events of the use case when everything is perfect; there are no errors, no exceptions. This is the "happy day scenario". The exceptions will be handled in the "Alternate Flows" section.
Alternate Flows	The most significant alternatives and exceptions

2.2. Name and briefly describe your use case

Use Case Number:	1
Use Case Name:	Buyer Places a Bid
Description:	A buyer has identified an item he wishes to buy, so he will place a bid for an item with the intent of winning the auction and paying for the item.

2.3. Create the use case basic flow

The basic flow of a use case represents the most important course of events or what happens most of the time, sometimes referred to as the "Happy Day Scenario" because it is what occurs when everything goes well -- no errors or exceptions. Another reason why the basic flow is so critical is because it's much easier to fully comprehend the exceptions once the norm is understood and if the basic flow represents 70% of the system, the development staff is much more prone to implementing the correct code in the first pass.

2.4. Create the use case alternate flows

The basic flow is the key ingredient to your use case and some can argue that you can stop once you're done with the basic flow. It really depends on the level of detail you wish to achieve. However, providing more detail to the consumers of your use case is always a good thing.

The alternate flows provide the following:

- An exception or error flow to any line item in your basic flow
- An additional flow, not necessarily error based, but a flow that COULD happen

2.5. Product your effective use case document

A few reasons why it's that much easier to learn a system through use cases than a traditional requirements document is probably because with use cases you are introduced to concepts at a high level, walk through a living scenario, and then presented with specifications last.

The purpose of the use cases is for effective knowledge transfer from the domain expert to the software developer -- these use cases will serve as software requirements. If they don't make sense to the person building the software, they are not effective.

D. USING VISUAL PARADIGM FOR UML

Elements:

- Textual analysis, a variant for identifying actors and use cases (steps 2 and 3 in section 1.3).
- Use case diagrams
- Use case scheduling: To schedule the use cases by assigning priorities.
- Use case description (use case narrative): A use case description describes the use case, including the preconditions, post-conditions, flow of events, etc.
- Use case detail: A use case detail holds one or more use case description.

E. INDIVIDUAL WORK

1. a). Study at

http://www.visual-paradigm.com/product/vpuml/provides/umlmodeling.jsp

the following DEMOS:

Use case diagram
Flow of events editor
Specify testing procedures
Generate sequence diagram from flow of events

b). Study the following presentations from the laboratory folder:

- Part_1_Using_Use_Case_Diagram.pdf
- Part_2_Use_Case_Diagram_Tutorial.pdf, exercises 1-5; in order to practice, use data from the folder "resurse_ Use_Case_Diagram_Tutorial".
- Document_Use_Case_using_Use_Case_Detail.pdf

or
Subscribe to „Study FREE courses" at http://www.visual-paradigm.com/training/
Study „Using use case diagram" course Part1 and Part2:exercises 1-5.
Study „Document Use Case using Use Case Detail" course.

c). Study the UC narrative in file from laboratory folder:
- Process Order Use Case Document.pdf
(extracted from http://www.cragsystems.co.uk/development_process/develop_use_case_document.htm)

Exercise 1:
Using VP-UML, represent the use case diagram in **1.3.b** , section **C**.

Exercise 2:
Draw a use case diagram of the following simple Answering Machine.

- Use Case: Leave a Message
    - Actor: Caller
    - Pre-Condition: Room on Tape
    - Post-Condition: New Message
- Use Case: Review Messages
    - Actor: Owner
    - Primary Path: Review New Messages
    - Alternate Path: No New Messages

- Use Case: Review Messages Locally
    - Primary Path: Review New Messages
    - Alternate Path: No New Messages
    - Extends: Review Messages
- Use Case: Review Messages Remotely
    - Primary Path: Review New Messages
    - Alternate Path: No New Messages
    - Extends: Review Messages
    - Uses: Authorize Access
- Use Case: Authorize Access
    - Primary Path: User Authorized
    - Alternate Path: User Not Authorized

Exercise 3:

Create a UML Use Case diagram from the following (bulletized) requirements. **In addition, answer the questions provided (before drawing the diagram).**

We are developing a system with which teachers can record and update grades of students. Teachers should also be able to distribute report cards. Here is a complete list of the requirements for the system:

- A teacher can record grades. Whenever grades are recorded, they are also saved to disk.
- A teacher can update grades. Whenever grades are updated, the existing grade is loaded. Then the updated grade is saved to disk.
- A teacher, a registrar, and/or a student can view grades.
- Whenever any of these people view grades, they must always log on to the system. If their log on fails, they must re-authenticate their user name and password.
- A part-time student is a kind of student.
- A registrar can generate report cards.
- A teacher can distribute report cards.

(1) Identify the actors (i.e. name them):
…
(2) Are any of these actors a specialized type of another more general actor? If so, identify which one(s) is the generalized actor and which is the specialized actor. What type of relationship should there be between the generalized and the specialized actor?
…
(3) Identify the Use Cases (i.e. name them):
…
(4) Are any Use Cases always used by (an)other Use Case(s)? If so, what type of relationship is there? If so, identify which Use Case(s) is (are) always using (an)other Use Case(s) and which Use Case(s) is (are) always used.
…
(5) Are any of the Use Cases “sometimes” used by (an)other Use Case? If so, what type of relationship is there? If so, identify which Use Case(s) is (are) “sometimes using” and which Use Case(s) is (are) “sometimes used.”
…
(6) Draw the described Use Case diagram, including all *actors*, *Use Cases*, and *relationships*. Be sure to use the correct notation for all actors, Use Cases, and relationships. Also be sure to label each and every actor, Use Case, and relationship.

Exercise 4: Draw a use case diagram of the following application.

A library contains books and journals. The task is to develop a computer system for borrowing books. In order to borrow a book the borrower must be a member of the library. There is a limit on the number of books that can be borrowed by each member of the library.
The library may have several copies of a given book.
It is possible to reserve a book.
Some books are for short term loans only. Other books may be borrowed for 3 weeks. Users can extend the loans.

Give a use case description for two use cases:
• Borrow copy of book
• Extend loan

Exercise 5:

Consider the next description of a CAR RESERVATION SYSTEM business process.

A software application must be developed in order to assist the company people in working with clients' data in a car reservation company.
The application data will be stored in structured text files.

Identify actors and use the following descriptions as the basis for the use cases. Draw a use case model showing any relationships between the use cases. Represent then in VP for UML the use case diagram, edit the details (use case narrative – adapted from the following business process description) and generate the sequence diagram from the flow of events.

CAR RESERVATION SYSTEM business process:

MAKE RESERVATIONS
A customer contacts a reservation officer about a car rental.
The customer quotes the start and end dates needed, the preferred vehicle, and the pickup office.
The reservation officer looks up a prices file and quotes a price.
The customer agrees to the price.
The vehicle availability is checked to see if an appropriate vehicle is available for the required time at the required office.
If the requested vehicle is available at the nominated pickup office, then it is reserved for the customer. An entry is made in the vehicle availability registering the reservation.
The reservation officer issues a rental number to the customer.
A rental agreement is then created in a rental file, including the rental number, the rental period, the vehicle type and the pickup office.

Exceptions
1. An appropriate vehicle is not available at the pickup office. The customer is offered an alternative vehicle.
2. The customer does not agree to a price and asks for an alternate vehicle and/or period.

MAINTAIN AVAILABILITY SYSTEM
A vehicle availability file is checked to see if a vehicle of a given type is available at the requested pickup office for a requested rental period. There is a record for each vehicle which includes the times when a vehicle is available and when it is rented.
If it is available, then the vehicle is reserved for the requested period.

Exception
1. If a reservation cannot be made because of lack of vehicles a problem report is issued to be used in planning vehicle levels.

INITIATE RENTAL
A customer arrives at a pickup office and quotes a rental number to the rental officer.
The rental file is checked to the customer's rental number.
If a correspondent record is found, then the rental agreement is retrieved and discussed with the customer.
If the customer accepts, then a rental agreement is printed.
The customer signs the agreement and lodges a credit card number.
The rental officer requests the customer to select one of a number of insurance options. Following the selection an insurance policy is filled out and attached to the signed agreement..

Exceptions
1. A customer does not have a prior reservation. In that case a vehicle availability check is made. If a vehicle is available, the customer is offered the vehicle and a price is quoted. If the customer accepts then a rental is initiated.
2. If the kind of reserved vehicle is not available to a customer with a prior reservation (because of a late return) then an alternate proposal is made to the customer.

PROCESS VEHICLE RETURNS
The customer records the mileage and fuel level and quotes them to the rental officer.
The amount of fuel to be added to the vehicle is added to the rental account on the rental agreement file.
The rental account is checked by the customer.
The customer pays the rental amount.

Exceptions
1. The returned vehicle is damaged requiring a forfeit of the deposit and filling in of an insurance claim.
2. The customer disputes the account.

PROVIDE MANAGEMENT REPORTS
The reservation statistics file is updated at the time that reservations are made. The file is also updated when vehicles are rented to customers without a prior reservation.
The statistics are updated at the time of return to show the length of reservation and the amount of moneys received.
A rental summary is generated on request from management.

A FURTHER EXTENSION
Extend the design by adding a further feature to support regular rentals to company customers. This will make an agreed number of vehicles available to employees of the company at nominated offices on a daily basis. The company can nominate and provide a list of authorized employees, who can make the pickups. The company will be presented with an account on a monthly basis. To do this you may need to first write a new use case and/or amend existing use cases.

FURTHER INDIVIDUAL STUDY:
At : http://www.cragsystems.co.uk/SFRWUC/

Specifying Functional Requirements With Use Cases
    Part 1. Actors and Use Cases
    Part 2. Use Case Diagrams and Text