BULETINUL Universităţii Petrol – Gaze din Ploieşti	Vol. LVIII No. 4/2006	41 - 56	Seria Ştiinţe Economice

Cycle: UML FOR MANAGERS (II) The Use Case Diagram

Liviu Dumitraşcu, Gabriel Irinel Marcu

Universitatea Petrol-Gaze din Ploieşti, Bd. Bucureşti 39, Ploieşti
e-mail: ldumitrascu@upg-ploiesti.ro, gimarcu@upg-ploiesti.ro

Abstract

UML allows the construction of more models of the same system. This paper aims to present one of these models, practically the first that we have to construct: the use case diagram. This model is based on the user's (client's) requirements presented in the application specifications or on the functions of the studied system.

Key words*: actor, system, use case, the use cases diagram, inclusion, extension, generalization, frames of interaction*

Basic Principles and Definitions

UML (**U**nified **M**odeling **L**anguage) allows the construction of more models of the same system: some describe the system from the user's (client's) point of view, others describe the inner structure, while others display a global or detailed view over the system.

The models are then completed and can be put together. They are elaborated during the living cycle necessary for the development of a system (ranging from the user's requirements up to the creation phase). We shall further on study one of these models, the first we have to construct: the diagram of the use cases.

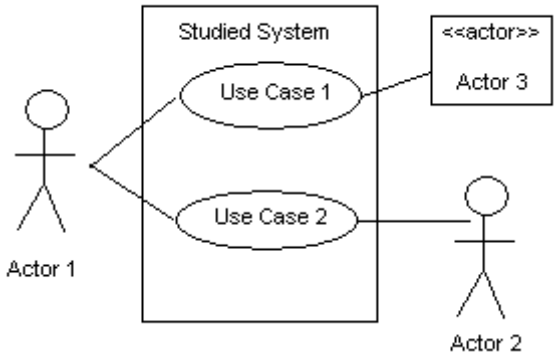
This *use case diagram* allows the study, analysis and organization of the demands. The diagram of the use cases represents the starting point for the analysis of a system.

The use case diagram represents an integrant part of the family of the UML language behavioral diagrams.

The use cases diagrams are used to describe the requirements of the system. They are exclusively based on the use cases.

The use case diagram describes the functional interactions between actors and the studied system (figure.1).

Figure 1



The use case diagram constitutes the first model of the studied system.

The diagram of the use cases represents one of the UML diagrams benefiting from the simplest syntax.

Elementary Concepts

The elementary concepts used when modeling the user's requirements the functions of the approached system are: the actors, the system, the use cases.

The Actors

An actor represents a role played by an external entity (human, peripheral user etc.) which interacts with the studied system.

An actor takes part in at least one use case.

An actor can interact with and/or directly change the state of the system by emitting and/or receiving meaningful messages.

The main actors are the users of the system who are generally easily identifiable.

The concept of actor allows for a classification into: main and secondary actor.

The main actor is that for whom the use case generates an observable result.

We name secondary actors other participants in the use case. They are generally requested for supplementary information.

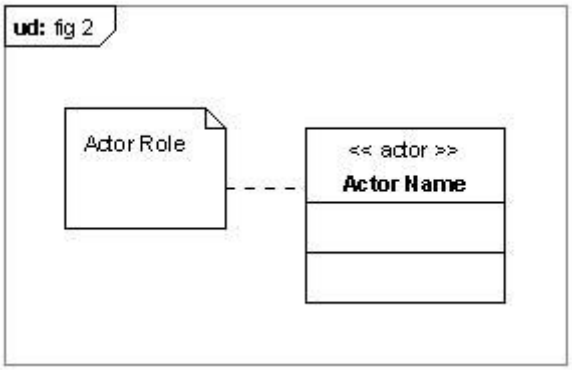
Requirements:

- If possible, arrange the main actors on the left side and the secondary actors on the right side of the use system.
- Associate one actor to each use case.
- If possible, eliminate the 'physical' actors and promoting the 'logical' ones.
- Identify as actors only the external entities and not the internal components of the studied system.
- Do not mistakenly associate the role with the concrete (external) entity. The same external concrete entity can successively play various roles as compared to the studied system and can be modelled by means of various actors. Reciprocally, the same role can be

simultaneously played by various external concrete entities, which will be thus modelled by means of the same actor.

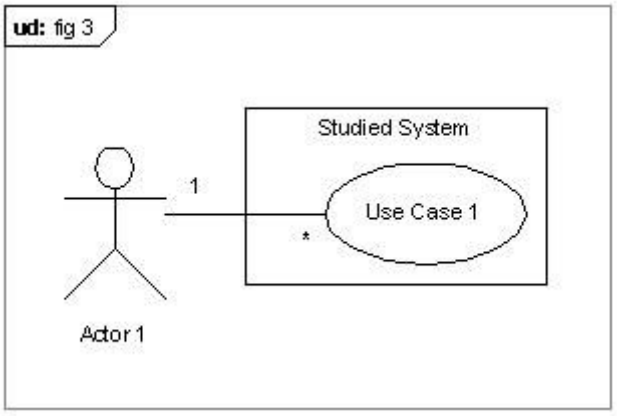
Add a comment to the identified actor in order to emphasize its role (figure 2).

Figure 2



An actor can use the same use case several times. (figure 3).

Figure 3

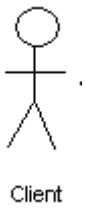


An actor is identified by name. Remember to think of its role!

A human actor is presented using stick man and a name (the name of the role).

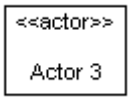
Figure 4 represents the actor (client).

Figure 4



Remark: For the non-human actors use a square containing the key word “actor” (figure 5).

Figure 5



The System

For an UML model, a system represents an application. It is identified by name and it contains all the appropriate use cases.

A system is represented by means of a rectangle containing the name of the system and the use cases of the application.

The actors, external to the system, are connected to the use cases through a relation of association meaning “takes part in”.

Figure 6 represents the actors within a web application for on-line (virtual) books shop.

Figure 6

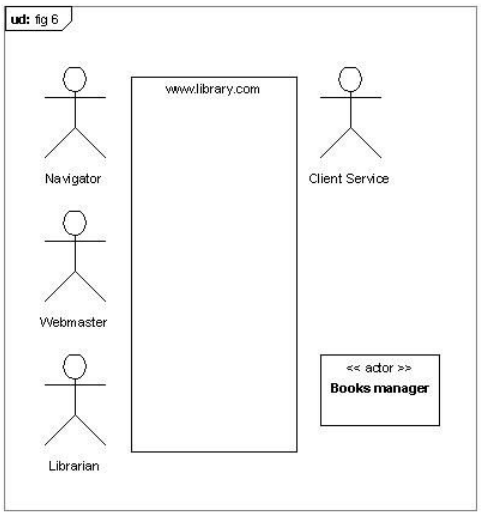
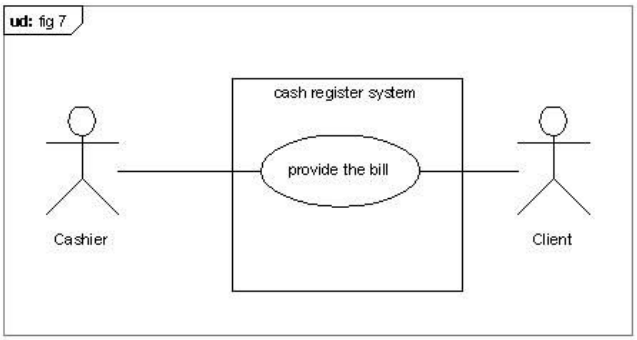


Figure 7 represents the use cases diagram of a simplified cash register system in a supermarket.

Figure 7



Use Case

A use case represents the specific way of a system actions modeling. The actors, placed outside the system, model everything interacting with it.

A use case represents a group of sequences of actions performed by the system and which generate an observable result, interesting to a private actor.

A use case performs a service (function) gradually, generating an incipit, a display and an ending for the actor taking place in the respective use case.

Each use case must follow a certain pattern of description (figure 8).

Figure 8

1. *This use case starts when…*
2. *The system answers by means of:*
   - *sequence of interactions;*
3. *This case ends when…*

The concept of use case allows a system approaching from a functional point of view. Each use case implies an expected behaviour from the studied system, taken as a whole, without imposing the way of realizing this behaviour. It allows for a description of what the future system is expected to while not specifying how to do it!

Remarks:

- The use case represents a group of actions and not a singular one.
- A use case may involve several actors.
- Name the use cases using a verb, followed by an object(e.g.: Performing a task)

C. Larman has introduced the notion of real use case and essential use case [3].

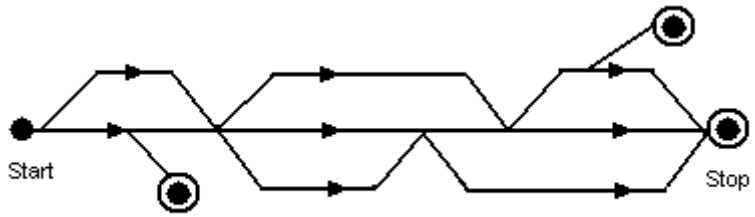
The essential use case describes a process from an analytical point of view, as independently as possible from the hard/soft environment.

The real use case describes a process from a conceptual point of view, in terms of: events, user interface etc.

We can consider use cases as a collection of scenarios dealing with success or failure describing the way in which a certain actor uses the system in order to reach an objective [4](figure 9).

The use case must have clearly defined its beginnings and ends.

Figure 9



There are two types of scenarios: the normal (real) scenario and its extensions.

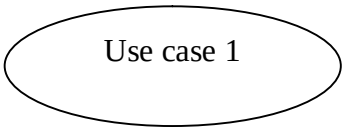
The real scenario allows for the fulfilment of the actors' objectives following the most appropriate way towards success.

Its extensions contain all the other success (normal end) or failure (exceptions) scenarios.

Each scenario is made of stages, sequentially numbered, in order to indicate the possible extensions. Thus, for a ´Y' stage, a first extension will be ´Ya' and will "first of all", identify the condition for generating the extension and then the answer is sent to the system. The answer coming from the system can comprise one or more stages, sequentially numbered. A second stage, referring to ´Y' will be ´Yb' etc.

A use case is presented by an ellipsis containing the name of the use case (figure 10).

Figure 10



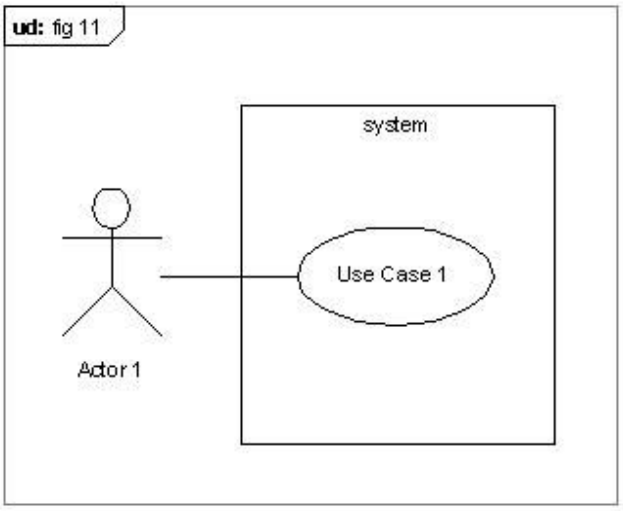
Remark: The name of the use case can also be placed below the ellipsis.

Association

Use cases are connected to the actors using lines, called associations and they emphasize the potential interactions between the system and the outer world. Each association practically represents: "takes part in".

Remark: A use case must be connected to at least one actor (figure 11).

Figure 11



If an actor does nothing else but receiving messages from the system (or sending messages) use an arrow at the end of line implying restriction from navigation (figure 12).

Figure 12

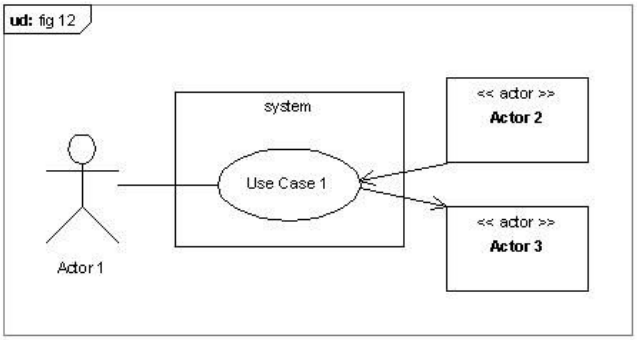
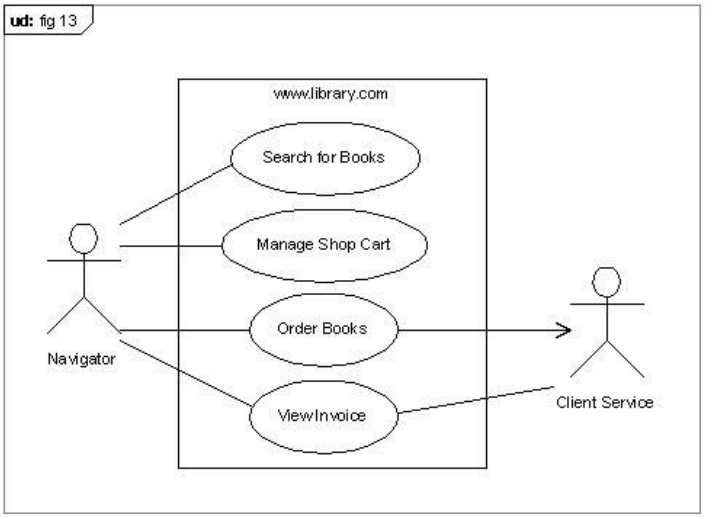


Figure 13 represents the use cases diagram of **Navigator** user within a virtual books shop web application.

Figure 13



Advanced Concepts

Advanced Concepts Regarding Use Cases

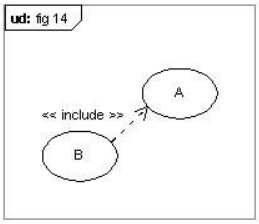
The advanced concepts applied to use cases are concerned with the relations types between use cases: inclusion, extension, generalization.

Inclusion

- In UML 2 one can specify the fact that a use case *includes* another use case.
- Use the inclusion relation for use cases when a behavioral sequence is identical in more use cases.
- The case A is included in case B if the behavior described by A is included in the one described by B. We can say that B depends on A. This dependency is symbolically represented by the key-word “includes”.

Figure 14 represents an inclusion relation between two use cases (A and B).

Figure 14

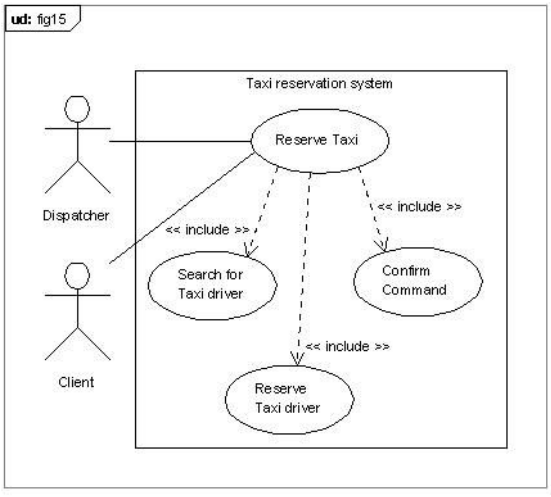


Remarks:

- The relation of inclusion can be graphically represented by means of a discontinuous line ended with arrow on which we place the key-word “includes”.
- Pay attention to the direction of the arrow! The arrow at the end of the discontinuous line shows “what” it includes and the use case placed on the opposite side (without an arrow) marks “who” includes.
- The main use case has explicitly to incorporate another one if the inclusion relation is used.
- At the same time, inclusions allow for the fragmentation of a complex case into simple sub-cases.

Figure 15 presents the preliminary use cases diagram of a taxi reservation system on which we use the inclusion relations.

Figure 15



Extension

In UML 2 one can observe that a use case extends another use case.

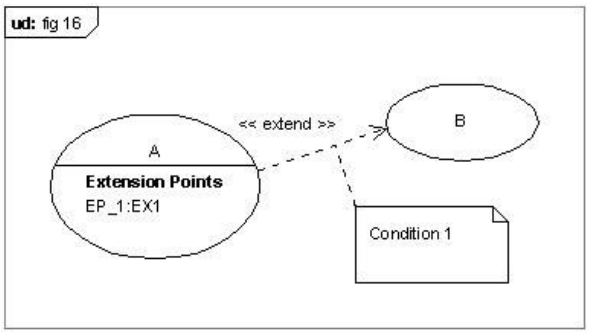
Use the extension relation between use cases in order to specify an optional complex behaviour from a compulsory behaviour.

If the behaviour of the use case B can be extended by means of the use case A then we could say that A extends B. An extension is more often than not submitted to a condition.

From a graphic point of view, the condition is expressed as a note. The extension relation becomes formal due to the key-word "extends" when the main use case implicitly incorporates another one, as an optional entity.

Figure 16 represents an extension relation between two use cases (A and B).

Figure 16



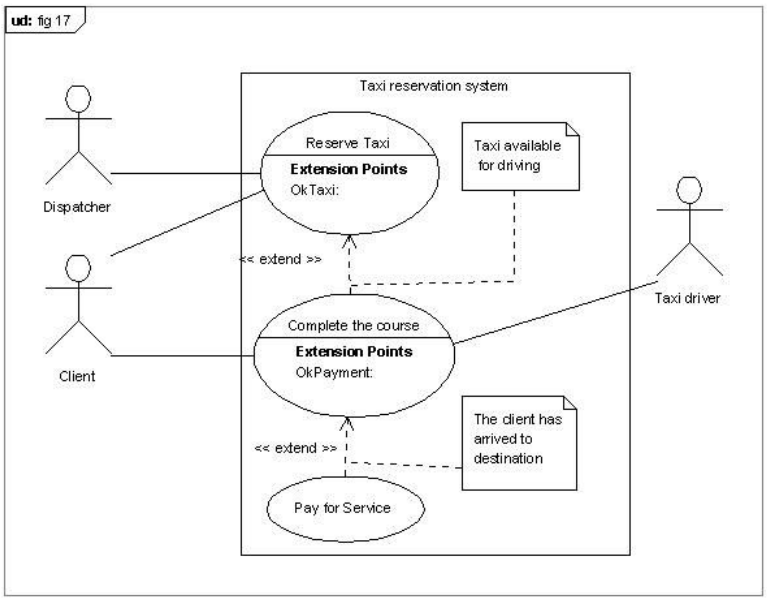
Remarks:

- The extension relation is graphically represented by means of a discontinuous arrowed line on which we place the key-word "extends".
- One should pay particular attention to the orientation of the arrow. The arrow at the end of the discontinuous line shows "who is extended", while the use case placed on the opposite side (without an arrow) marks "what it is extended with".

In an extension relation, the main use case (for example case B) implicitly incorporates another (for example case A) in an indirectly specified point, mainly the one preceding the extension.

Figure 17 presents the preliminary use case diagram for a taxi reservation system where we use the extension relations.

Figure 17



Generalization

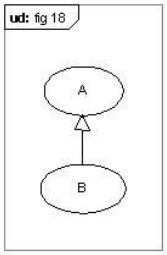
UML 2 makes it possible for one to mention a relation of generalization between use cases. This relation can be translated through the concept of inheritance in the object-oriented languages.

Use the relation of generalization between use cases in order to formalize the important variations of the same use case.

A use case A is a generalization of a use case B if B is a particular case of A.

Figure 18 represents a relation of generalization between two use cases.

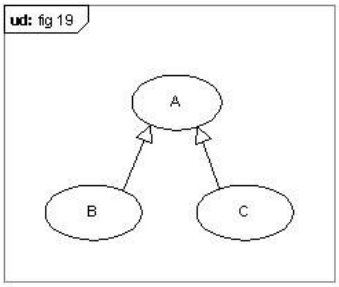
Figure 18



Remark: The relation of generalization can be graphically represented by means of an close headed line with the arrow pointing towards the more general modeling element.

Figure 19 represents a relation of generalization between three use cases (A, B and C).

Figure 19



Remarks:

- Use cases B and C represent specializations of the use case A. Generalization is the opposite of specialization.
- The arrow at the end of the line shows that A, typed in italics, is a generalized use case.
- The A use case displays the particularity of being abstract since it cannot be directly instantiated, but by means of the two specialized cases (B and C) placed at the end of the line without arrow.
- In a relation of generalization/specialization, the derived descendant use case inherits the behaviour of the parent use case. Each of the derived cases can incorporate specific behaviours added to the one inherited from the parent case.

Figure 20 presents the completed diagram of the use cases for a taxi reservation system, where we use the generalization/specialization relations.

Figure 20

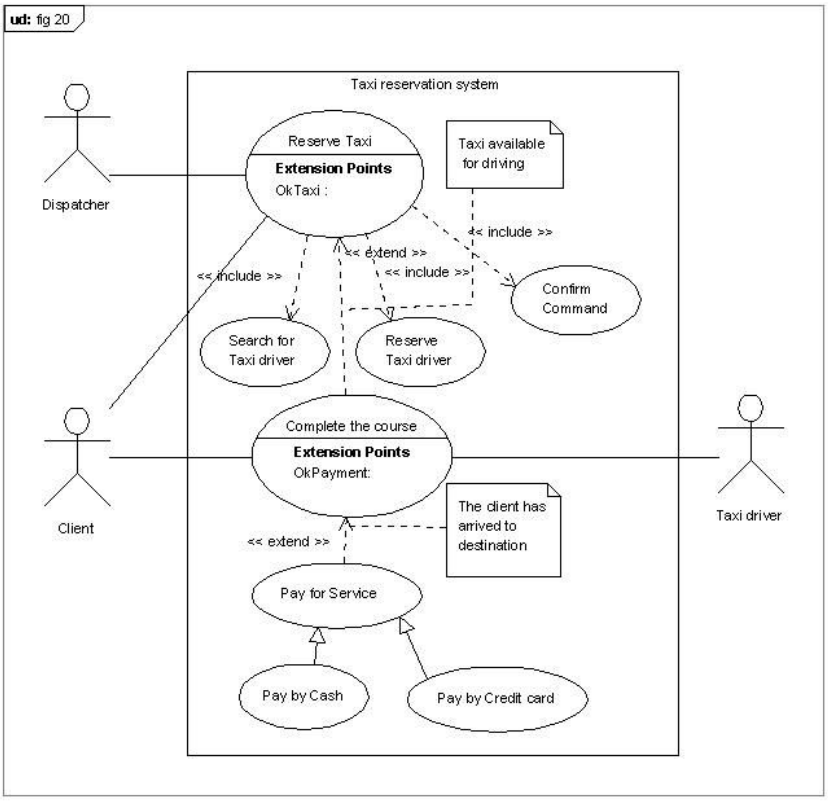
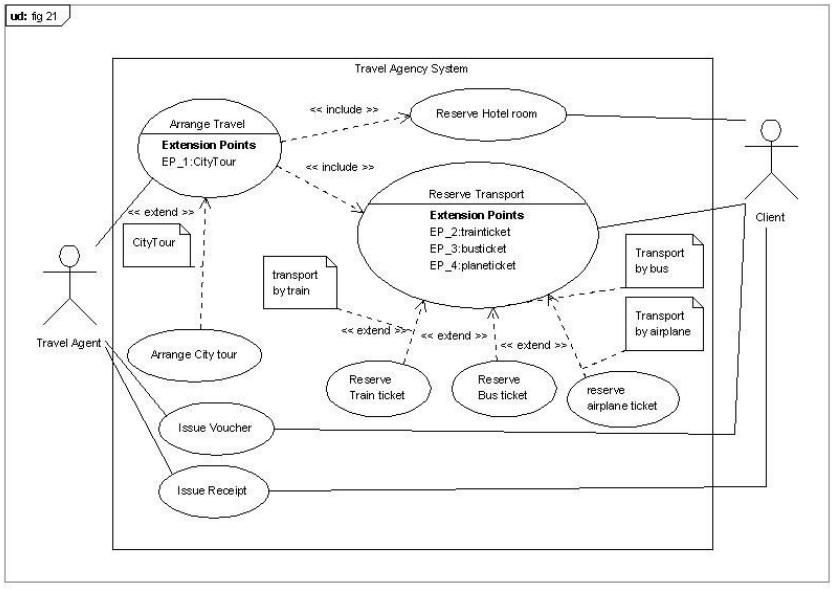


Figure 21 presents the use case diagram for the administrative activities modeling one trip abroad arranged by a travel agency.

Figure 21



Comments:

The use cases diagram (figure 21) models the administrative activity of a trip organized by a travel agency (Droopy Tour) through big European cities by train/plane/bus.

If ordered, the travel agency can provide a city tour (by bus).

The use cases "Reserve hotel rooms" and "Reserve transport" are internal cases because they are not directly connected to an actor (travel agent as main actor respectively the client as secondary actor).

The use case diagram also contains two cases: "Issue voucher" and "issue receipt" realized by the same two actors (travel agent and client).

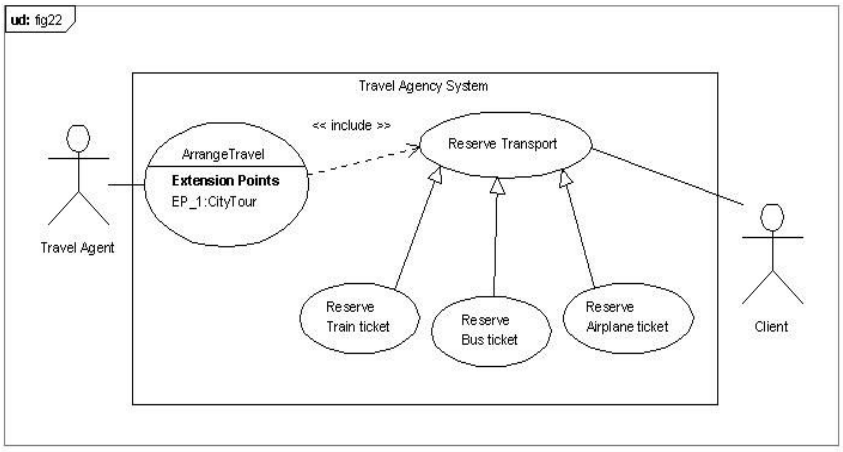
A first relation of extension has been applied to the case "Reserve transport" because choosing the means of transportation is exclusively the client's option.

A second relation of extension has been applied to the case "Arrange city tour" because setting up a city tour is again the client's option.

We could consider that "Reserve transportation" is a generalized use case (figure 22). This case has the particularity of being abstract (it requires for a format in italics!) because it cannot be instantiated directly, but by means of one of the specialized (concrete) use cases: "Reserve train tickets", "Reserve bus tickets", "Reserve plane tickets".

The two solutions presented here (extension, generalization) for the use case "reserve transport" are perfectly possible. It is hard to say which solution is the best fits for the situation. A model is correct if it fulfils the future users of the application (system). *Analyze and decide*!

Figure 22



Advanced Concepts Regarding the Actors

Generalization

UML 2 allows relations between actors. The only one possible is generalization.

An actor A is a generalization of an actor B if actor A can be replaced by B.

Remark: All use cases that are accessible to actor A are also accessible to actor B but not reciprocally.

Figure 23 presents a relation of generalization between two actors (actor 1 and actor2).

Figure 23

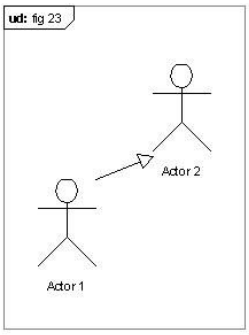
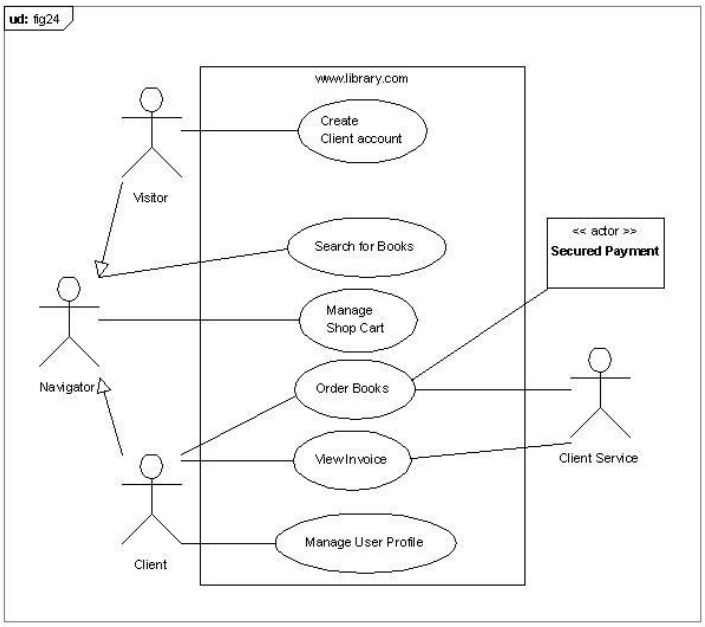


Figure 24 presents the advanced concept of generalization between the actors for a book shop web application.

Figure 24



Describing the Use Case

There are various ways of describing the use cases and UML does not impose any restriction. This freedom must be indeed recognized as confusing but UML can models any system; a sole method able to describe a use case is not eloquent.

We shall further on present a procedure meant to identify the use cases and a use case template for the description with narrative text use cases adapted from Pascal Roques[4].

How Do We Identify Use Cases?

The procedure meant to identify use cases is the following:

1. Delimiting the subject-the studied system frontiers (perimeter);
2. Identifying the main actors;
3. Identifying the objectives for each main actor;

4. Defining the autonomous use cases which correspond to the objectives of the main actors.

Most 20 Use Cases

- Limit the number of use cases at 20 (apart from the included or specialized cases and from extensions!)
- Do not overuse the relations between use cases. They can overload the diagram of the use cases thus reducing the clarity of the diagram.

How Do We Describe Use Cases?

The easiest method to describe the dynamics of the use cases is represented by the narrative (textual) description.

The narrative description of a use case contains the following elements:

- The identification summary which is compulsory;
- Describing the scenarios, again compulsory;
- The non-functional requests (optional).

The identification summary includes:

- The name of the use case;
- Objectives;
- The creation and updating information;
- The name of those in charge;
- The main and secondary actors;
- The number of the version.

The description of the scenarios contains:

- Preconditions;
- The description of the nominal scenario;
- The description of alternative scenarios;
- The description of error (exception) scenarios.

The non-functional requests generally contain the technical specifications as well as some potential restrictions regarding the human-machine interface (HMI).

Remarks:

- Do not mix the essential use cases with the real ones; the first ones are much more stabile and can be easily re-used.
- When editing narrative text use cases avoid redundant words.
- Edit the use cases in a “black box”- do not describe the internal functions of the system, its components or conception.
- Take into consideration everything which is visible to the actors.
- Complete the narrative description of the use cases with one or more behavioural (dynamic) diagrams UML 2: the activity diagram; the system sequence diagram with the interaction frames (*loop, ref, alt, opt* etc.); the interaction overview diagram.

The human-machine interface for the use case "Insert form in Dreamweaver 8" is displayed in figure 25.

Figure 25

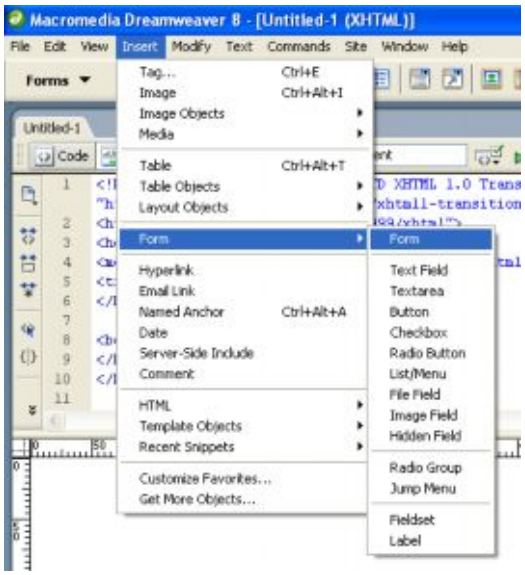


Figure 26 gives a real description of the use case “Insert form in Dreamweaver 8”.

Figure 26

Identification

Case name	Isert Form in Dreamweaver 8
Main Actor	Web designer
Objective	Describe the stages which the web designer actor has to go through in order to insert a form in a HTML page using Dreamweaver 8
Version	1.0
Person in charge	M. Ionescu in Ploiesti
Preconditions	o The site is created and organized in HTML pages o The option Form delimiter is activated; o The invisible elements are displayed.
Postcondition	-

The nominal scenario

1 The use case starts when the web designer clicks on the spot the form will be placed in

2 The web designer select from Insert menu option Form|Form

3 Dreamweaver 8 generates a red discontinuous rectangle which materializes the limits of the form

Alternative scenario

2. a The web designer wants to use the toolbar buttons

1. The web designer presses the Form button from Forms category. The nominal scenario go to the step 3

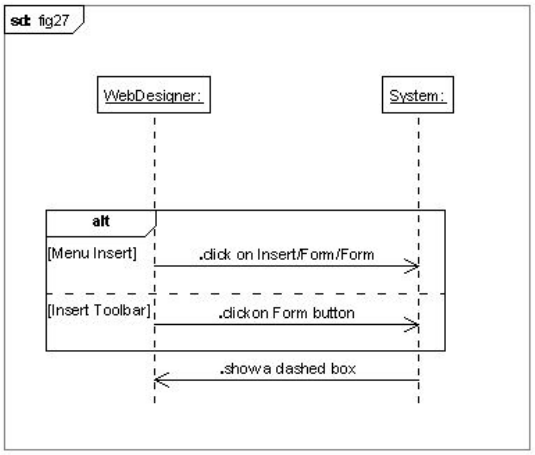
Remarks:

- Dreamweaver 8 is the strongest and the most complete soft able to create a web-site;
- By means of Dreamweaver 8 you can rapidly create web pages which contain text, images, charts, layers, frames, connections, hypertext, forms, multimedia objects;
- In order to insert a form in a web page by means of Dreamweaver 8, use one of the methods: the Insert Menu or the Insert toolbar;
- After inserting a form you will have to insert the objects of the form;

Figure 27 displays the System Sequence Diagram (SSD) which completes the real (narrative) description of the use case: "Insert form in Dreamweaver 8".

Remark: The System Sequence Diagram (SSD) uses the interaction frame **alt** which describes the alternative structure: the *Menu Insert* or the *Insert Toolbar Button*.

Figure 27



References

1. Balzert, H. - *UML 2 compact*, Eyrolles, 2006
2. Charroux, B., Osmani, A., Thierry-Mieg, Y. - *UML 2*, Pearson Education France, Collection Syntex, 2005
3. Larman, C. - *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design*, Prentice Hall, 1997
4. Roques, P. - *UML 2 par la pratique*, 5-e édition, Eyrolles, 2006

Ciclul: UML PENTRU MANAGERI (II)
Diagrama cazurilor de utilizare

Rezumat

UML permite construirea mai multor modele pentru acelaşi sistem. Acest articol are drept scop prezentarea unuia din aceste modele, practic primul care trebuie construit: diagrama cazurilor de utilizare. Acest model se bazează pe cerinţele clienţilor prezentate în caietul de sarcini.