# Writing Effective Use Case

Tài liệu dành cho Sinh viên học môn Phân tích và Thiết kế HTTT D15CNTT.PTIT

Use case analysis is a major technique used to find out the functional requirements of a software system. Use case, an important concept in use case analysis, represents an objective user wants to achieve with a system. It can be in text form, or be visualized in a use case diagram, like this:
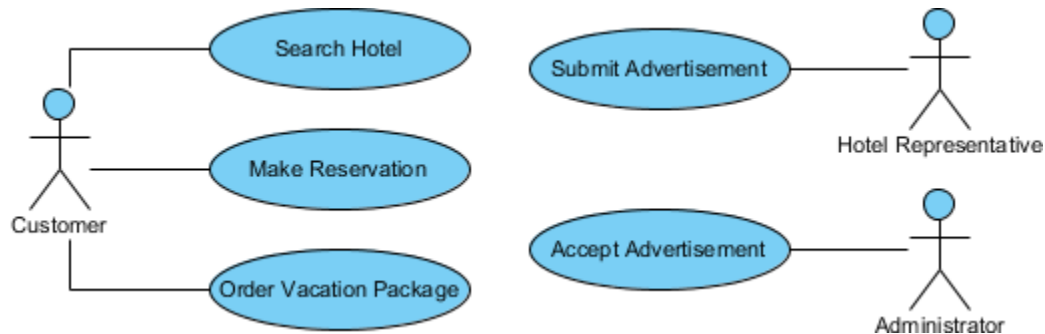
The beauty of use case is that it aims at describing a system from external usage viewpoint, rather than from developer's perspective. Therefore, writing use case can be the deciding factor for building a system that meets users' needs. In this tutorial, you will learn how to make use of various functions to write effective use case.

What is a Use Case?

A use case is an objective user(s) wants to achieve with a system. Use cases are named with verb or verb + noun phrase. It is usually short yet descriptive enough to describe a user objective. You are encouraged to use concrete and specific verbs and nouns to avoid ambiguity. Verbs like 'do' and 'perform' and nouns like 'data' and 'information' should be avoided whenever possible.

User performs use case to yield observable goal. Take online hotel reservation system as an example. "Make reservation" is undoubtedly a use case as this is what user wants to achieve with the system. The function of looking up a hotel on an online map can also be what a user needs. However, it is not a use case because it is only a part of the reservation process instead of an objective.

Some analysts try to make use of use case to describe user interface requirements (e.g. support multiple look & feel), performance requirements (e.g. load in background), deployment arrangement (e.g. ready server) and even implementation level (internal) requirements (e.g. construct database). All these are wrong and will not help you in identifying the objectives user want to achieve, thus the functions the system should deliver.

What Makes a Use Case "Effective"?

A use case can be a simple title that describe a user goal, such as "Make Reservation" in hotel reservation system. It is intended to provide an overview of what the user want without knowing how to achieve the goal. In order to identify how to achieve a goal, you can also document its scenario and steps (i.e. interaction) involved between user and system, with main flow, exceptional flow, conditional flow, etc.
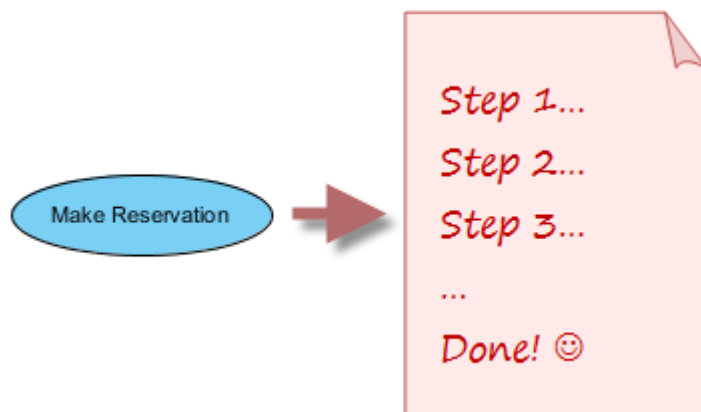

In VP-UML, there are some other features that helps you leverage the power of use case. You can use the use case grid to tidily list the use cases in model, and to model the other aspects of use case by elaborating use case with sub-diagram of another diagram type (e.g. use a sub-requirement diagram to model the additional functional and performance requirements).
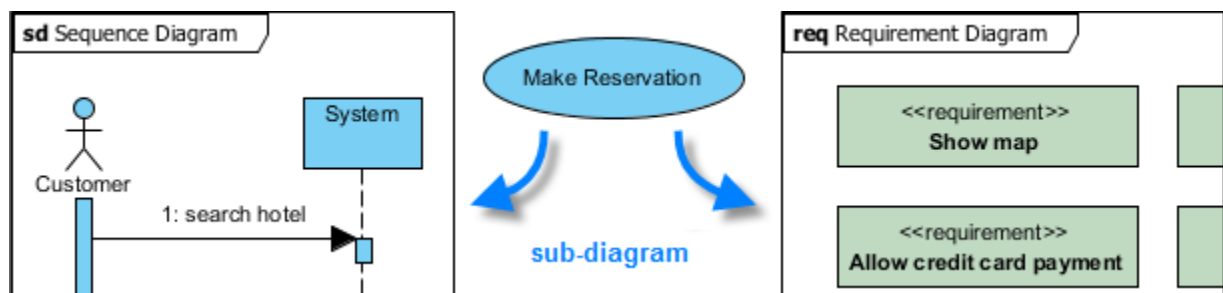
Use Case Diagram

A use case diagram is a kind of Unified Modeling Language (UML) diagram defined by Object Management Group (OMG), created for use case analysis. Use case diagram provides a graphical overview of goals (represented by use cases) users (represented by actors) want to achieve by using the system (represented by system boundary but is often opt out in diagram). Use cases in a use case diagram can be organized and arranged according to their relevance, level of abstraction and impacts to users. They can be connected to show their dependency, inclusion and extension relationships. The main purpose of modeling use case with use case diagram is to establish a solid foundation of the system by identifying what the users want. Base on the result of analysis you can move forward to study how to fulfill those user needs.
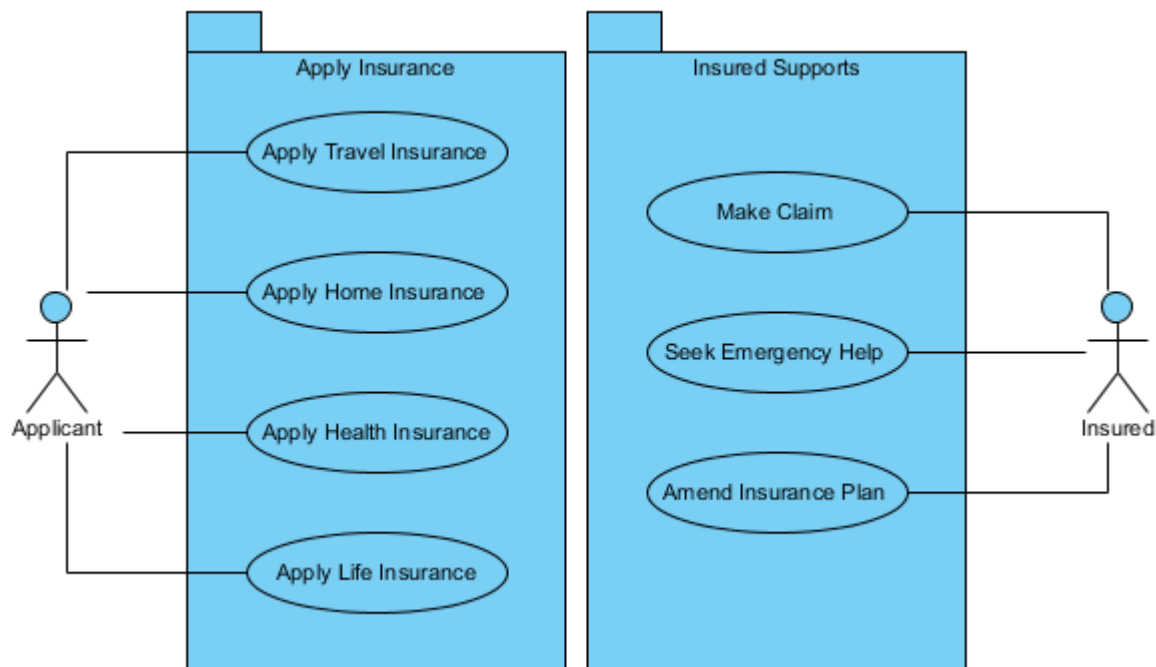

A use case diagram is mainly formed by actors, use cases and associations (connectors).

An actor is any person or external system that interacts with the system in achieving a user goal. There are two kinds of actor - primary and secondary. Primary actor is anyone or thing that interacts with the system to gain direct benefit. Secondary actor is anyone or thing that involve in achieving a use case yet, they do not gain direct benefit from the system. Very often, secondary actor is someone who assist the primary actor to achieve a use case.

Drawing Use Case Diagram in VP-UML

In this tutorial, we will make use of an online hotel reservation system as an example to demonstrate how to write effective use case with VP-UML. Let's begin by drawing a use case diagram. We will carry on with writing effective use case with the resulting design.

1. Create a blank use case diagram by clicking **UML** in toolbar and selecting **Use Case Diagram** from the drop down menu.
2. Enter the name of diagram: *Hotel Reservation*.
3. Select **Actor** in the diagram toolbar. Click on the diagram to create an actor and name it **Customer**.
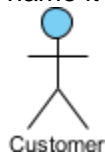
4. A customer can make a hotel reservation, which is a use case of the system. Let's create a use case from the *Customer* actor. Move the mouse pointer over the Customer actor. Press on the **Association -> Use Case** resource icon and drag it out.
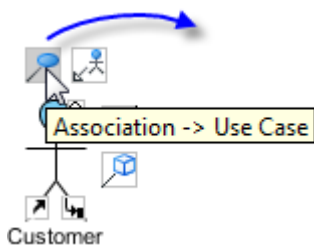
5. Release the mouse button to create the use case. Name it Make Reservation. The association between actor and use case indicates that the actor will interact with the system to achieve the use case associated.
6. Complete the design to make it look like this:
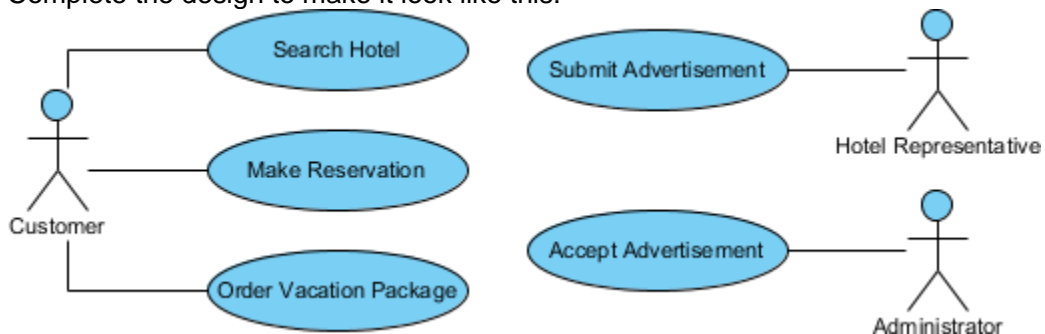

Capture Use Case Scenario with Flow of Events

A named use case depicts the 'what' aspect of a use case by telling you what the users need. The 'how' aspect of use case that explains how a user goal can be accomplished can be further analyzed by using flow of events, which is a technique for analyzing the interaction between actor and system in accomplishing a use case.

Flow of events constitutes a high level user-and-system conversation, which aims to find out the intents or actions of actor, known as inputs, and how system react to those actor inputs. You should be concise when deciding what to include in the events flow. Do not include how system process user's input internally, or even implementation detail like an insertion of database record. This is wrong as flow of events, in fact use case analysis, is aimed to view things from actor's perspective. Implementation details is not of flow of events' interest. Implementation detail can, however, be modeled with sequence diagram in form of sub-diagram of use case.

Let's write the flow of events of use case. We will describe some of the important skills in writing flow of events shortly. Now, we go back to our use case diagram first.

1. Right click on Make Reservation and select **Open Use Case Details...** from the popup menu.
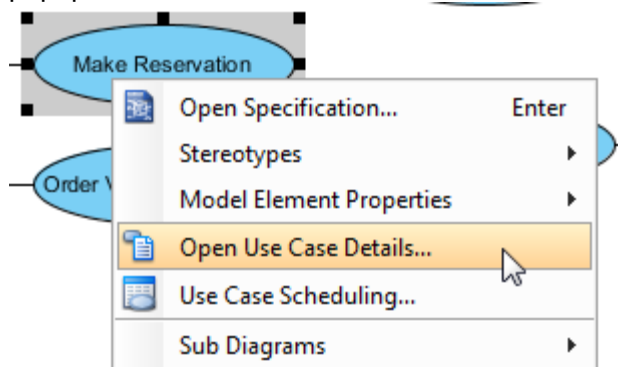

2. Open the **Flow of Events** tab. The flow of events editor is formed by rows, known as steps. Each step represent an actor input or system response.

3. Click on the first step and enter the first user input: *Enter city, arrival, departure, room type and click Search*.
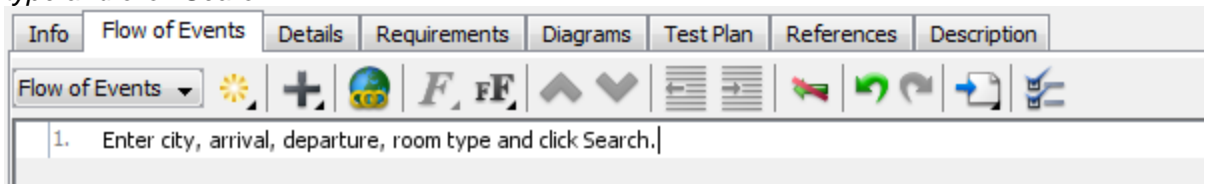
4. Use of the format functions to set the word *Search* in blue and bold, for emphasis.
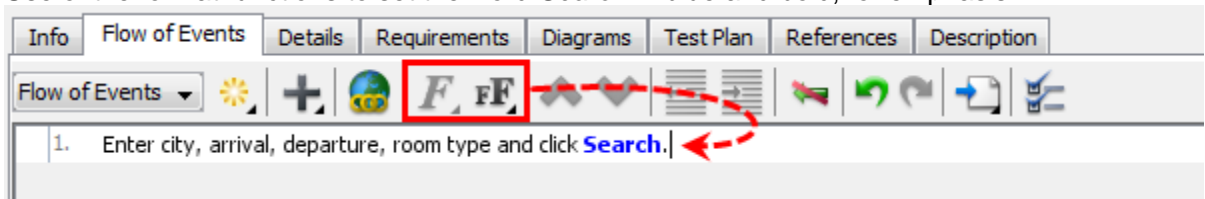
5. Press **Enter** to complete this step. Step 2 will be created for you.

6. Step 2 is about how the system react to user's input. You may start by writing "System...", but there is a better way to represent system response. Click on the first button in toolbar. Select **System Response** from the drop down menu.
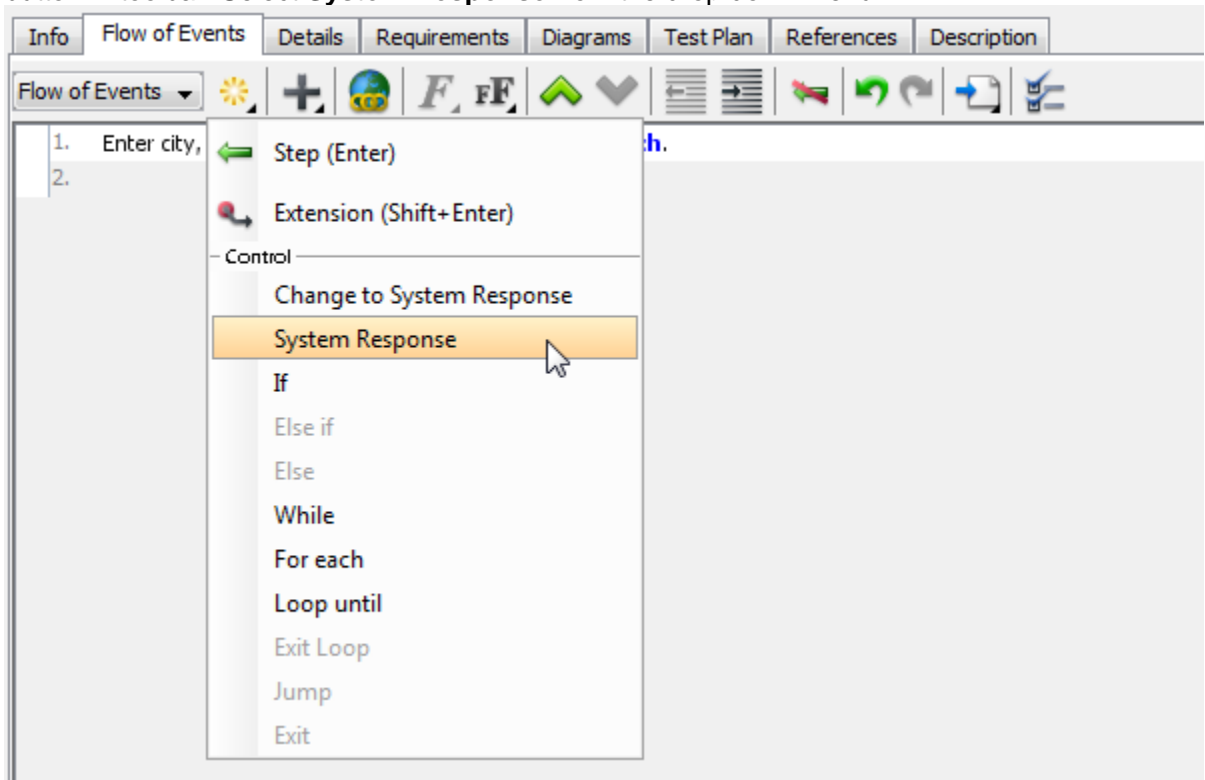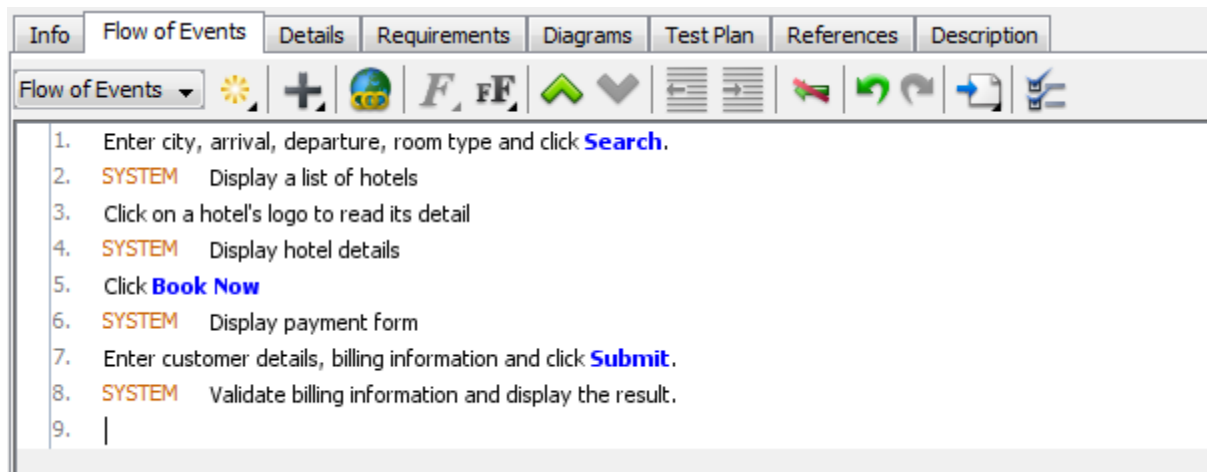

7. You can now enter the content of step 2: *Display a list of hotels*.
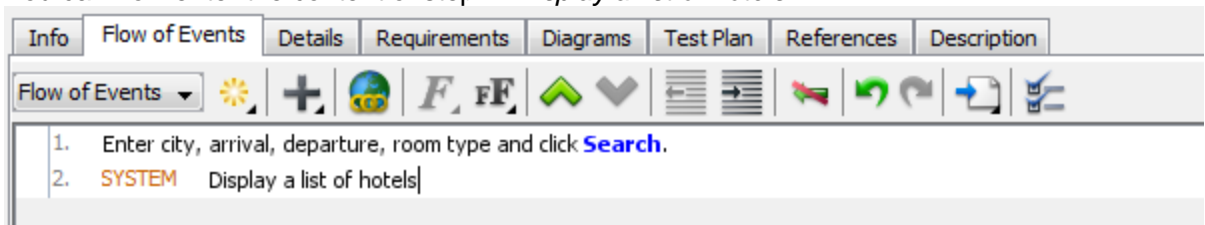

8. Add the following steps:

User input	System response

|	
Click on a hotel's logo to read its detail	
	Display hotel details
Click Book Now	
	Display payment form
Enter customer details, billing information and click Submit.	
	Validate billing information and display the result.

9.

10. There will be two possible results of validation here, either fail or success. If success, user will click Confirm to complete the reservation. If fail, user will need to enter the customer details and billing information again. To describe this kind of conditional flow, we can make use of the **If, Else if, Else, While, For each, Loop until** controls. Let's add an **If** condition now. Click on the first button in toolbar. Select **If** from the
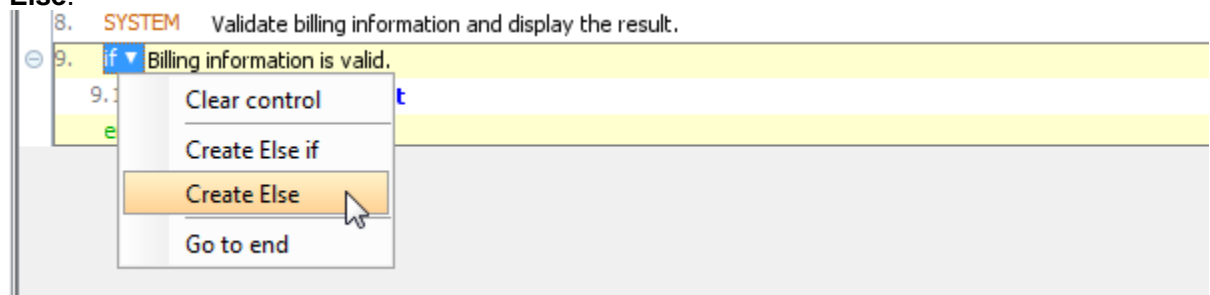
drop down menu.
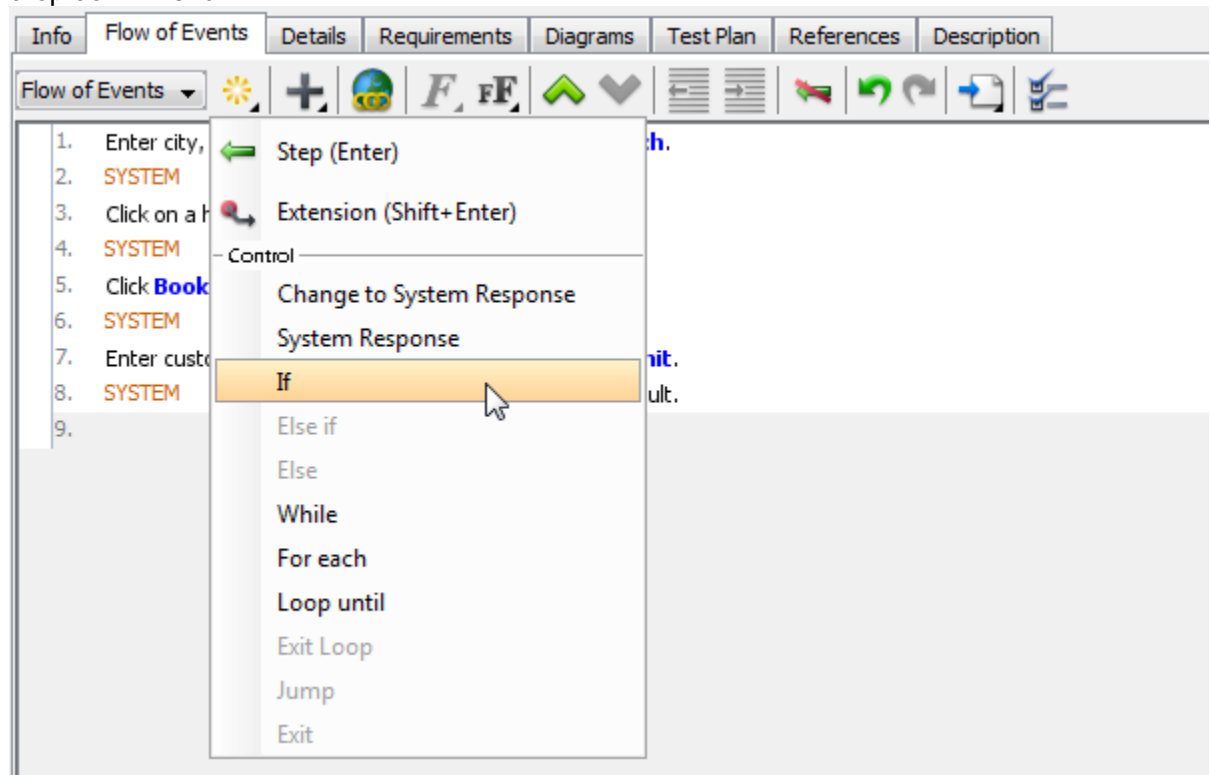
11. Enter after **if**: *Billing information is valid.* Press **Enter** and enter what to do: *Click Confirm Payment*.
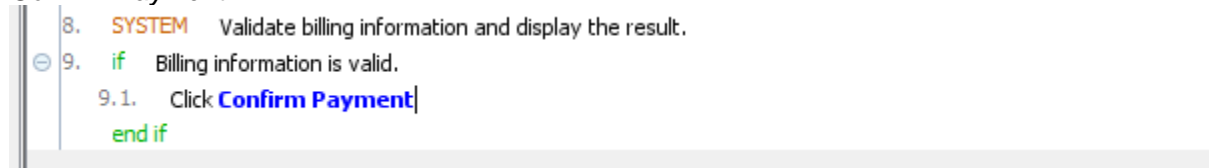

12. Move the mouse pointer to **if** and click on the down arrow button. Select **Create Else**.


13. Press **Enter** to move to the body of else clause.

14. As mentioned, if the billing information is invalid, user is required to enter the customer details and billing information again. In other words, user need to re-preform step 7. Instead of re-writing the step, we can add a **Jump** control here to direct the flow back to step 7. Click on the first button in toolbar. Select **Jump** from

the drop down menu.
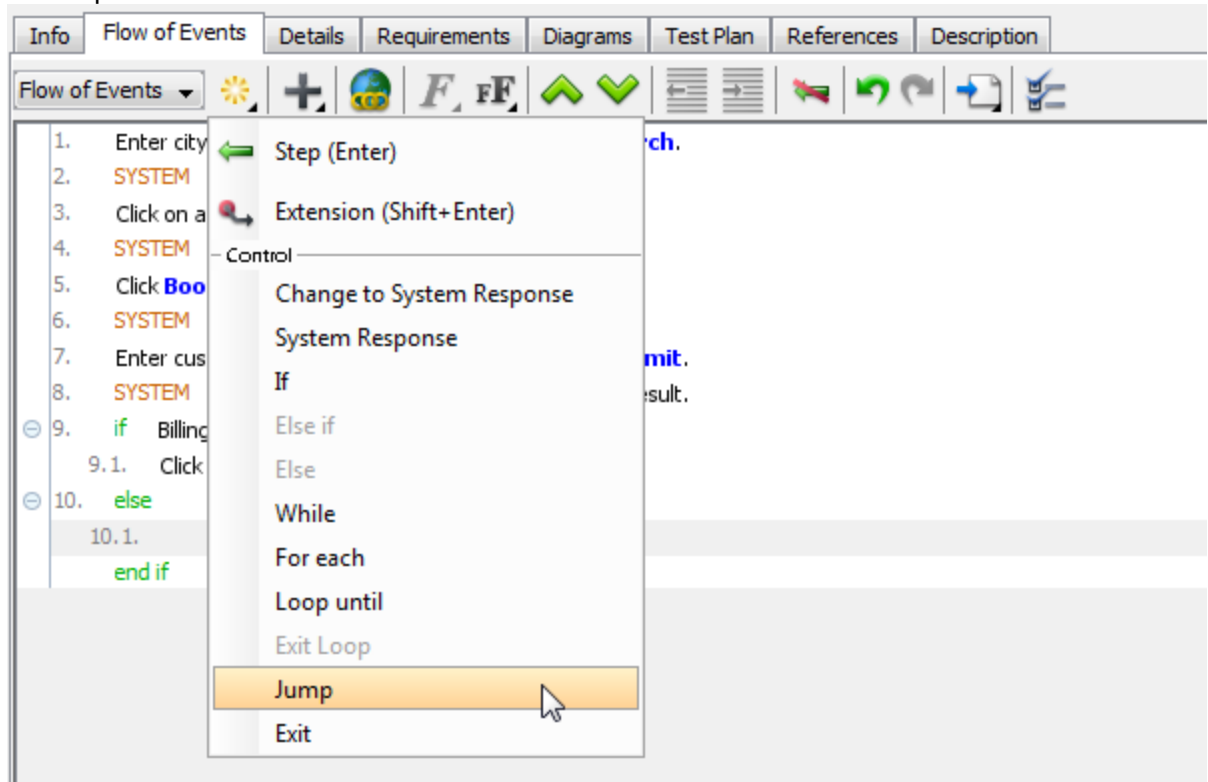
15. Click on the arrow in front of step 7 to select it.
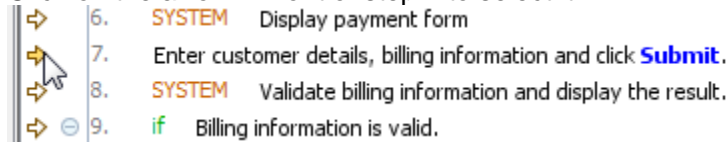
16. At the bottom of the flow of events editor you can find the **Extension** section. An extension represents a variation of use case being extended. The variation may be triggered when walking through the main flow, under certain condition. Let's take this use case as example. Let's say it normally takes one working day to process a reservation. However, there may be users who have urgent reservation needs and want to complete the process immediately after the submission of request. If the system is going to support this case, we can include the additional steps required as an extension to this use case. Click on step 9.1, where the extended flow will take place.

17. Click on the first button in toolbar. Select **Extension (Shift-Enter)** from the drop down menu. This brings you to the extension section with step 9.1.a created.

18. Enter in 9.1.a: *Process rush reservation.* Press **Enter**.

```
Extension:
    9.1.a.   Process rush reservation.
             <Type here to add an extension without any parent>
```

19. Enter the steps involved:

User input	System response
Select Rush Reservation option	
	Display a customer code and telephone number. Ask user to call for a special arrangement.

```
Extension:
⊖ 9.1.a.   Process rush reservation.
    1.   Select Rush Reservation option
    2.   SYSTEM   Display a customer code and telephone number. Ask user to call for a special arrangement.|
             <Type here to add an extension without any parent>
```

20.