Table of Contents

[**Chapter 10** 2](#_Toc128832653)

[**The Construction Phase** 2](#_Toc128832654)

[**\*Review Elaboration Phase.** 2](#_Toc128832655)

[**Construction** 2](#_Toc128832656)

[**Chapter 11** 4](#_Toc128832657)

[**The Construction Phase : Analysis** 4](#_Toc128832658)

[**Preview:** 4](#_Toc128832659)

[**Back to the Use Cases** 4](#_Toc128832660)

[1. Pre-Conditions 4](#_Toc128832661)

[2. Post Conditions 4](#_Toc128832662)

[3. Main Flow 5](#_Toc128832663)

[4.Alternate Flows 5](#_Toc128832664)

[5.Exception Flows 5](#_Toc128832665)

[**The UML Sequence Diagram** 6](#_Toc128832666)

[1.Define: 7](#_Toc128832667)

[2.Message arrows: 8](#_Toc128832668)

[3.Focus of Control 8](#_Toc128832669)

[4. Fragment 8](#_Toc128832670)

[5. How to create sequence diagram. 9](#_Toc128832671)

# **Chapter 10**

# **The Construction Phase**

## **\*Review Elaboration Phase.**

* Need to understand the problem as fully as possible, without going into too much detail.
* Built a Use Case Model, and created as many Use Cases as possible, without filling complete details of the Use Cases, instead we supplied a very brief description of each one.
* Build a conceptual model, where we captured the concepts driving our development🡺 foundations of the design.
* Ranked each of our Use Cases, and in doing so, we have planned the order of the Use Case development.

Summary:

* A complete review of the phase would be held, and a Go/No Go decision needs to be made.
* have discovered during Elaboration that we really cannot provide a solution for our customer ñ better to find out now than at the end of coding!

## **Construction**

Aim: need to build the product, and take the system to the state where it can be delivered to the user community

Note:

Approaching project by following a series of short waterfalls, with a small number of Use Cases developed in each iteration.

* At the end of each iteration, we will review progress, and preferably timebox the iteration.
* At the end of each iteration, we will achieve a running system by testing then we reach the review.

Each stage of the waterfall will produce a set of documents or UML models. • In Analysis, we will produce some Expanded (or Full) Use Cases

• In Design, we will produce Class Diagrams, Interaction Models and State Diagrams .

• In Code, we will produce running and unit tested code.

# **Chapter 11**

# **The Construction Phase : Analysis**

## **Preview:**

* We need to revisit the Use Cases we are building in this iteration, and enhance and expand those Use Cases.
* By concentrating on the full detail of only a few Use Cases per iteration, we are reducing the amount of complexity we have to manage at any one time.
* Only concerned with the problem, and not the solution.

## **Back to the Use Cases**

**Use Case:**  Place Bet

**Short Description:**

The user places a bet on a particular horse after choosing a race

**Actors**: Gambler

**Requirements**  R2.3; R7.1

**Pre-Conditions:**

**Post-Conditions:**

**Main Flow:**

**Alternate Flow(s):**

**Exception Flow(s):**

*Figure 42 - The Short Use Case, Place Bet*

1. Pre-Conditions

This section describes the system conditions which must be satisfied before the Use Case can actually take place.

1. Post Conditions

The post conditions describe the state the system will be in at the end of the Use Case.

\*Note:

* The post-condition is conventionally written in past tense language.
* There can be more than one post condition, depending on the outcome of the Use Case.
* These different post conditions are described using “if then” language.

1. Main Flow

The main flow section describes the most likely, or the most conventional, flow of events through the Use Case.

\*Note:

* In the main flow, we need to detail the interactions between the actor and the system.
* Notice that every actor/system interaction is broken down into steps. In this case, there are seven steps in the main flow of the Use Case.

### 4.Alternate Flows

Alternate flows are simply less common (but legitimate) flows through the Use Case.

\*Note:

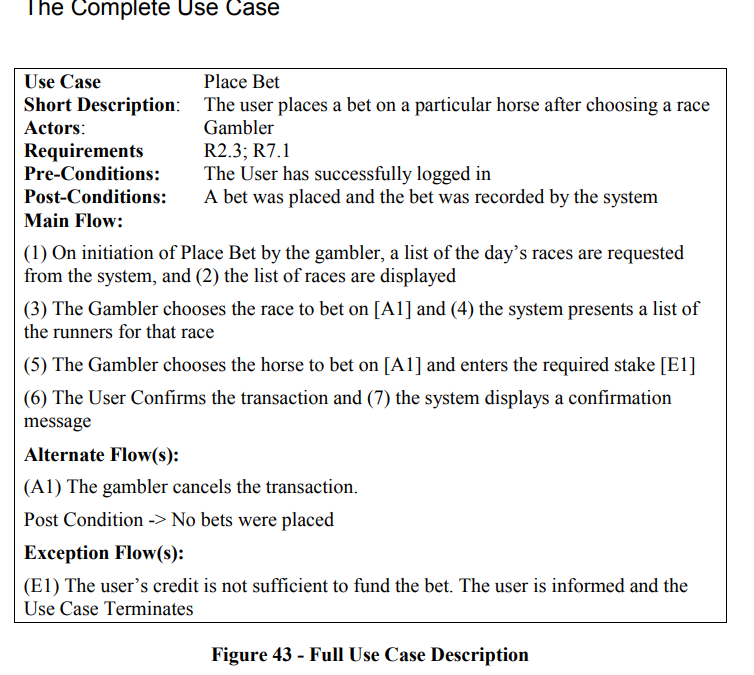
* The alternate flow will typically share many steps with the main flow, so we can notate the point in the main flow where the alternate flow takes over.
* A notation:[ A(number)]

### 5.Exception Flows

The exception flow describes exceptional situations. In other words, a flow where an error has occurred, or an event that couldn’t have otherwise been predicted.

\*Note:

* A notation of exception flow with may exist in main flow: [E(number)]
* When we move to program code. the items under Exception Flow should map to exceptions in the program - if your target language supports exceptions



## **The UML Sequence Diagram**

Why using UML Sequence Diagram?

It may be too difficult to distinct between analysis and design often the Use Case descriptions become littered with design decisions.

Since developers maybe type themselves down to specific design decision in Use Case description.

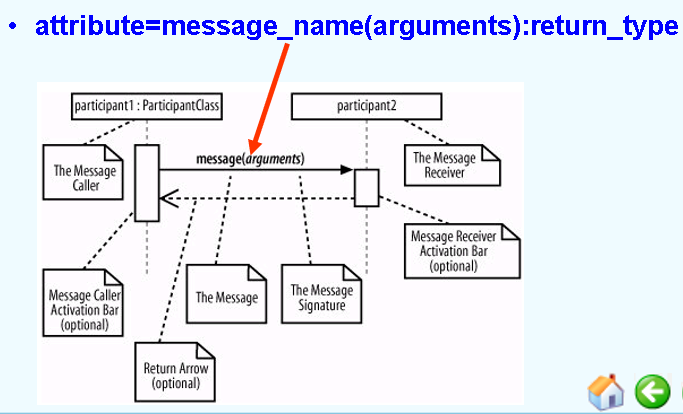
When building the Use Cases, we need to treat the system as a “black box”, which can accept requests from actors and return results to the actor.

We are not concerned (yet) with how the black box fulfils that request.

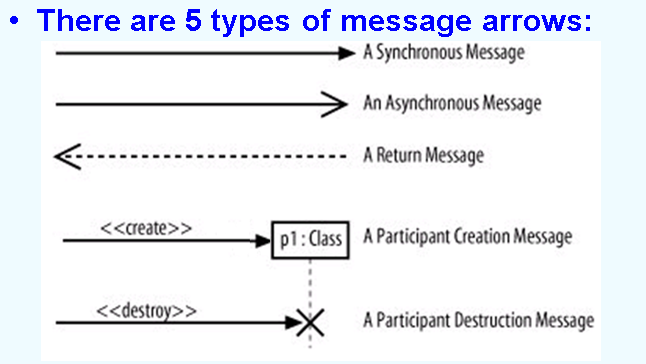
* Therefore, We recommend the use of a UML Sequence Diagram which m is useful in a variety of different situations, and can be used in analysis to help us with this “black box” analysis of the system.

### 1.Define:

* + A sequence diagram shows object interactions arranged in time sequence
  + The diagram shows
    - The objects participating in the interaction
    - The sequence of messages exchanged
  + A sequence diagram contains:
    - Objects with their “lifelines”
    - Messages exchanged between objects in ordered sequence
    - Focus of control (optional)



### 2.Message arrows:

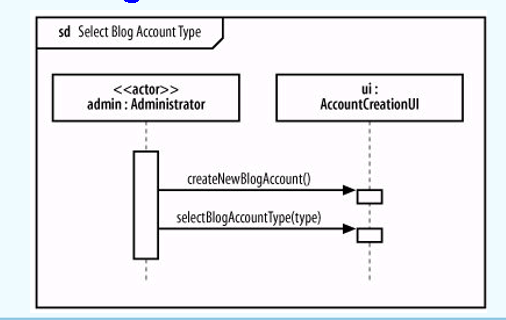


### 3.Focus of Control

* + Focus of Control represents the relative time that the flow of control is focused in an object
    - It represents the time an object is directing messages
  + Focus of Control may be shown on a sequence diagram

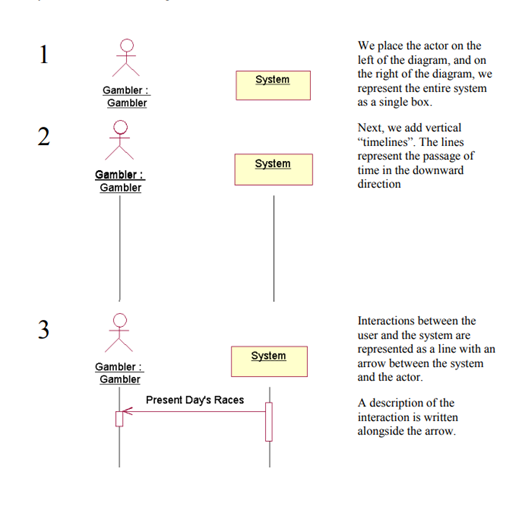
1. Fragment

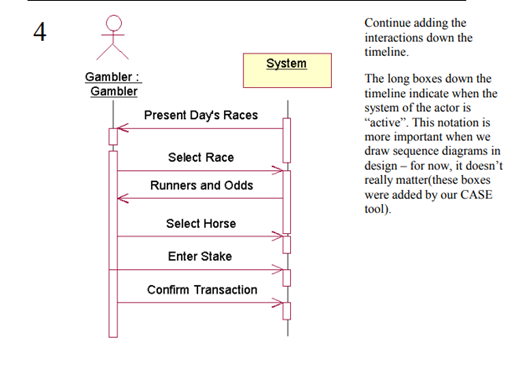
A sequence diagram may be broken into fragments



Some fragment types:

* + ref: an interaction defined elsewhere
  + loop: repeat the interactions in the fragment many times
  + break: exit the enclosing loop fragment
  + alt [guard conditions]: execute the corresponding set of interactions, base on which guard condition is true
  + opt [check]: optional fragment, executed only if the check value is true

1. How to create sequence diagram.



Once the System Sequence Diagram is complete, it is a fairly simple and mechanical task to write the description of the main flow for the use case.

There is no need to laboriously draw these diagrams for every single alternate and exception flow, although it would be worthwhile for very complicated or interesting alternatives.