A Use-Case is a series of related interactions between a user and a system that enables the user to achieve a goal.

Use-Cases are a way to capture functional requirements of a system. The user of the system is referred to as an 'Actor'. Use-Cases are fundamentally in text form.

Use-Case Points – Definition

Use-Case Points (UCP) is a software estimation technique used to measure the software size with use cases. The concept of UCP is similar to FPs.

The number of UCPs in a project is based on the following −

- The number and complexity of the use cases in the system.
- The number and complexity of the actors on the system.
    - Various non-functional requirements (such as portability, performance, maintainability) that are not written as use cases.
    - The environment in which the project will be developed (such as the language, the team's motivation, etc.)

Estimation with UCPs requires all use cases to be written with a goal and at approximately the same level, giving the same amount of detail. Hence, before estimation, the project team should ensure they have written their use cases with defined goals and at detailed level. Use case is normally completed within a single session and after the goal is achieved, the user may go on to some other activity.

History of Use-Case Points

The Use-Case Point estimation method was introduced by Gustav Karner in 1993. The work was later licensed by Rational Software that merged into IBM.

Use-Case Points Counting Process

The Use-Case Points counting process has the following steps −

- **Calculate unadjusted UCPs**
- **Adjust for technical complexity**
- **Adjust for environmental complexity**
- **Calculate adjusted UCPs**

Step 1: Calculate Unadjusted Use-Case Points.

You calculate Unadjusted Use-Case Points first, by the following steps −

- Determine Unadjusted Use-Case Weight
- Determine Unadjusted Actor Weight
- Calculate Unadjusted Use-Case Points

Step 1.1 − Determine Unadjusted Use-Case Weight.

Step 1.1.1 − Find the number of transactions in each Use-Case.

If the Use-Cases are written with User Goal Levels, a transaction is equivalent to a step in the Use-Case. Find the number of transactions by counting the steps in the Use-Case.

Step 1.1.2 − Classify each Use-Case as Simple, Average or Complex based on the number of transactions in the Use-Case. Also, assign Use-Case Weight as shown in the following table −

Use-Case Complexity	Number of Transactions	Use-Case Weight
Simple	≤3	5
Average	4 to 7	10
Complex	>7	15

Step 1.1.3 − Repeat for each Use-Case and get all the Use-Case Weights. Unadjusted Use-Case Weight (UUCW) is the sum of all the Use-Case Weights.

Step 1.1.4 − Find Unadjusted Use-Case Weight (UUCW) using the following table −

Use-Case Complexity	Use-Case Weight	Number of Use-Cases	Product
Simple	5	NSUC	5 × NSUC
Average	10	NAUC	10 × NAUC
Complex	15	NCUC	15 × NCUC
Unadjusted Use-Case Weight (UUCW)			5 × NSUC + 10 × NAUC + 15 × NCUC

Where,

NSUC is the no. of Simple Use-Cases.

NAUC is the no. of Average Use-Cases.

NCUC is the no. of Complex Use-Cases.

Step 1.2 − Determine Unadjusted Actor Weight.

An Actor in a Use-Case might be a person, another program, etc.

Some actors, such as a system with defined API, have very simple needs and increase the complexity of a Use-Case only slightly.

Some actors, such as a system interacting through a protocol (TCP/IP) have more needs and increase the complexity of a Use-Case to a certain extent.

Other Actors, such as a user interacting through GUI (Web Interface) have a significant impact on the complexity of a Use-Case.

Based on these differences, you can classify actors as Simple, Average and Complex.

Step 1.2.1 − Classify Actors as Simple, Average and Complex and assign Actor Weights as shown in the following table −

Actor Complexity	**Example**	**Actor Weight**
Simple	A System with defined API	1
Average	A System interacting through a Protocol	2
Complex	A User interacting through GUI	3

Step 1.2.2 − Repeat for each Actor and get all the Actor Weights. Unadjusted Actor Weight (UAW) is the sum of all the Actor Weights.

Step 1.2.3 − Find Unadjusted Actor Weight (UAW) using the following table −

Actor Complexity	**Actor Weight**	**Number of Actors**	**Product**
Simple	1	NSA	1 × NSA
Average	2	NAA	2 × NAA
Complex	3	NCA	3 × NCA
Unadjusted Actor Weight (UAW)			1 × NSA + 2 × NAA + 3 × NCA

Where,

NSA is the no. of Simple Actors.

NAA is the no. of Average Actors.

NCA is the no. of Complex Actors.

Step 1.3 − Calculate Unadjusted Use-Case Points.

The Unadjusted Use-Case Weight (UUCW) and the Unadjusted Actor Weight (UAW) together give the unadjusted size of the system, referred to as Unadjusted Use-Case Points.

Unadjusted Use-Case Points (UUCP) = UUCW + UAW

The next steps are to adjust the Unadjusted Use-Case Points (UUCP) for Technical Complexity and Environmental Complexity.

Step 2: Adjust For Technical Complexity

Step 2.1 − Consider the 13 Factors that contribute to the impact of the Technical Complexity of a project on Use-Case Points and their corresponding Weights as given in the following table −

Factor	Description	Weight
T1	Distributed System	2.0
T2	Response time or throughput performance objectives	1.0
T3	End user efficiency	1.0
T4	Complex internal processing	1.0
T5	Code must be reusable	1.0
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portable	2.0
T9	Easy to change	1.0
T10	Concurrent	1.0
T11	Includes special security objectives	1.0
T12	Provides direct access for third parties	1.0
T13	Special user training facilities are required	1.0

Many of these factors represent the project's nonfunctional requirements.

Step 2.2 − for each of the 13 Factors, assess the project and rate from 0 (irrelevant) to 5 (very important).

Step 2.3 − Calculate the Impact of the Factor from Impact Weight of the Factor and the Rated Value for the project as

Impact of the Factor = Impact Weight × Rated Value

Step (2.4) − Calculate the sum of Impact of all the Factors. This gives the Total Technical Factor (TFactor) as given in table below −

Factor	Description	Weight (W)	Rated Value (0 to 5) (RV)	Impact (I = W × RV)
T1	Distributed System	2.0		
T2	Response time or throughput performance objectives	1.0		
T3	End user efficiency	1.0		
T4	Complex internal processing	1.0		
T5	Code must be reusable	1.0		
T6	Easy to install	0.5		
T7	Easy to use	0.5		
T8	Portable	2.0		
T9	Easy to change	1.0		
T10	Concurrent	1.0		
T11	Includes special security objectives	1.0		
T12	Provides direct access for third parties	1.0		
T13	Special user training facilities are required	1.0		
			Total Technical Factor (TFactor)	

Step 2.5 − Calculate the Technical Complexity Factor (TCF) as −

TCF = 0.6 + (0.01 × TFactor)

Step 3: Adjust For Environmental Complexity

Step 3.1 − Consider the 8 Environmental Factors that could affect the project execution and their corresponding Weights as given in the following table −

Factor	Description	Weight
F1	Familiar with the project model that is used	1.5
F2	Application experience	0.5
F3	Object-oriented experience	1.0
F4	Lead analyst capability	0.5
F5	Motivation	1.0
F6	Stable requirements	2.0
F7	Part-time staff	-1.0
F8	Difficult programming language	-1.0

Step 3.2 − for each of the 8 Factors, assess the project and rate from 0 (irrelevant) to 5 (very important).

Step 3.3 − Calculate the Impact of the Factor from Impact Weight of the Factor and the Rated Value for the project as

Impact of the Factor = Impact Weight × Rated Value

Step 3.4 − Calculate the sum of Impact of all the Factors. This gives the Total Environment Factor (EFactor) as given in the following table −

Factor	Description	Weight (W)	Rated Value (0 to 5) (RV)	Impact (I = W × RV)
F1	Familiar with the project model that is used	1.5		
F2	Application experience	0.5		
F3	Object-oriented experience	1.0		
F4	Lead analyst capability	0.5		
F5	Motivation	1.0		
F6	Stable requirements	2.0		

F7	Part-time staff	-1.0		
F8	Difficult programming language	-1.0		
Total Environment Factor (EFactor)				

Step 3.5 − Calculate the Environmental Factor (EF) as −

1.4 + (-0.03 × EFactor)

Step 4: Calculate Adjusted Use-Case Points (UCP)

Calculate Adjusted Use-Case Points (UCP) as −

UCP = UUCP × TCF × EF

Advantages and Disadvantages of Use-Case Points:

Advantages of Use-Case Points

- UCPs are based on use cases and can be measured very early in the project life cycle.
- UCP (size estimate) will be independent of the size, skill, and experience of the team that implements the project.
- UCP based estimates are found to be close to actuals when estimation is performed by experienced people.
- UCP is easy to use and does not call for additional analysis.
- Use cases are being used vastly as a method of choice to describe requirements. In such cases, UCP is the best suitable estimation technique.

Disadvantages of Use-Case Points

- UCP can be used only when requirements are written in the form of use cases.
- Dependent on goal-oriented, well-written use cases. If the use cases are not well or uniformly structured, the resulting UCP may not be accurate.
- Technical and environmental factors have a high impact on UCP. Care needs to be taken while assigning values to the technical and environmental factors.
- UCP is useful for initial estimate of overall project size but they are much less useful in driving the iteration-to-iteration work of a team.