

NAME : NANDAWULA MARIA

REG NO : S20B23/207

ACCESS NO : A90651

COURSE : BSCS

COURSE UNIT : SOFTWARE PROJECT MANAGEMENT

LECTURER : MR. SIMON FRED LUBAMBO

Estimation techniques are of utmost importance in software development life cycle, where the time required to complete a particular task is estimated before a project begins. Estimation is the process of finding an estimate, or approximation, which is a value that can be used for some purpose even if input data may be incomplete, uncertain, or unstable

1. **Wideband Delphi Technique**

In the 1970s, Barry Boehm and John A. Farquhar originated the Wideband Variant of the Delphi Method. The term "wideband" 22 is used because, compared to the Delphi Method, the Wideband Delphi Technique involved greater interaction and more communication between the participants.

In Wideband Delphi Technique, the estimation team comprise the project manager, moderator, experts, and representatives from the development team, constituting a 3-7 member team. There are two meetings −

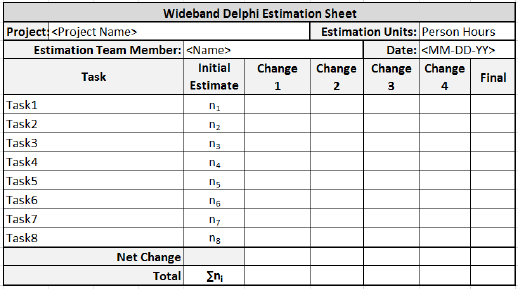
* Kickoff Meeting
* Estimation Meeting

Wideband Delphi Technique – Steps

**Step 1** − Choose the Estimation team and a moderator.

**Step 2** − The moderator conducts the kickoff meeting, in which the team is presented with the problem specification and a high level task list, any assumptions or project constraints. The team discusses on the problem and estimation issues, if any. They also decide on the units of estimation. The moderator guides the entire discussion, monitors time and after the kickoff meeting, prepares a structured document containing problem specification, high level task list, assumptions, and the units of estimation that are decided. He then forwards copies of this document for the next step.

**Step 3** − Each Estimation team member then individually generates a detailed WBS, estimates each task in the WBS, and documents the assumptions made.



**Step 4** − The moderator calls the Estimation team for the Estimation meeting. If any of the Estimation team members respond saying that the estimates are not ready, the moderator gives more time and resends the Meeting Invite.

**Step 5** − The entire Estimation team assembles for the estimation meeting.

**Step 5.1** − At the beginning of the Estimation meeting, the moderator collects the initial estimates from each of the team members.

**Step 5.2** − He then plots a chart on the whiteboard. He plots each member’s total project estimate as an X on the Round 1 line, without disclosing the corresponding names. The Estimation team gets an idea of the range of estimates, which initially may be large.



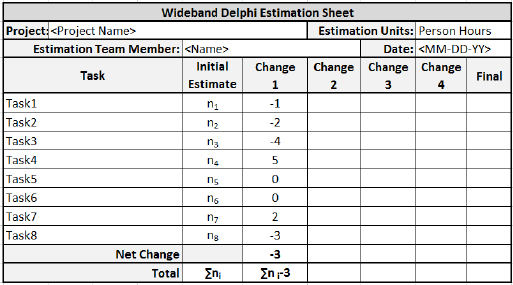
**Step 5.3** − Each team member reads aloud the detailed task list that he/she made, identifying any assumptions made and raising any questions or issues. The task estimates are not disclosed.

The individual detailed task lists contribute to a more complete task list when combined.

**Step 5.4** − The team then discusses any doubt/problem they have about the tasks they have arrived at, assumptions made, and estimation issues.

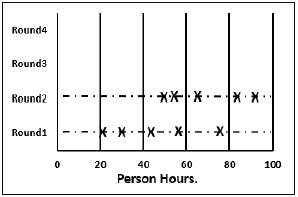
**Step 5.5** − Each team member then revisits his/her task list and assumptions, and makes changes if necessary. The task estimates also may require adjustments based on the discussion, which are noted as +N Hrs. for more effort and –N Hrs. for less effort.

The team members then combine the changes in the task estimates to arrive at the total project estimate.



**Step 5.6** − The moderator collects the changed estimates from all the team members and plots them on the Round 2 line.

In this round, the range will be narrower compared to the earlier one, as it is more consensus based.



**Step 5.7** − The team then discusses the task modifications they have made and the assumptions.

**Step 5.8** − Each team member then revisits his/her task list and assumptions, and makes changes if necessary. The task estimates may also require adjustments based on the discussion.

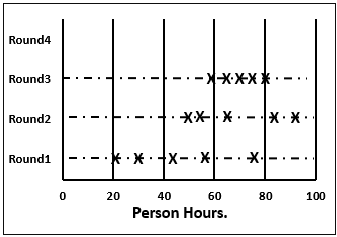
The team members then once again combine the changes in the task estimate to arrive at the total project estimate.

**Step 5.9** − The moderator collects the changed estimates from all the members again and plots them on the Round 3 line.

Again, in this round, the range will be narrower compared to the earlier one.

**Step 5.10** − Steps 5.7, 5.8, 5.9 are repeated till one of the following criteria is met −

* Results are converged to an acceptably narrow range.
* All team members are unwilling to change their latest estimates.
* The allotted Estimation meeting time is over.



**Step 6** − The Project Manager then assembles the results from the Estimation meeting.

**Step 6.1** − He compiles the individual task lists and the corresponding estimates into a single master task list.

**Step 6.2** − He also combines the individual lists of assumptions.

**Step 6.3** − He then reviews the final task list with the Estimation team.

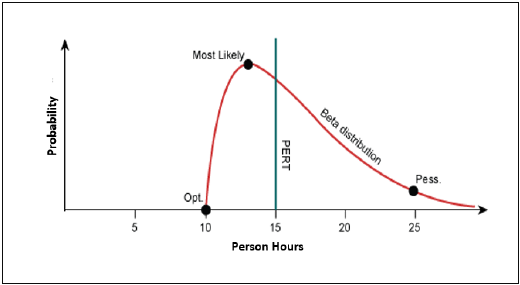
**Project Evaluation and Review Technique (PERT)**

Project Evaluation and Review Technique (PERT) estimation considers three values: the most optimistic estimate (O), a most likely estimate (M), and a pessimistic estimate (least likely estimate (L)). There has been some confusion regarding Three-point Estimation and PERT in the Industry. However, the techniques are different. You will see the differences as you learn the two techniques. Also, at the end of this chapter, the differences are collated and presented.

PERT is based on three values − most optimistic estimate (O), a most likely estimate (M), and a pessimistic estimate (least likely estimate (L)). The most-likely estimate is weighted 4 times more than the other two estimates (optimistic and pessimistic).

PERT Estimate (E) is based on the weighted average and follows beta distribution.

**E = (O + 4 × M + L)/6**



PERT is frequently used along with Critical Path Method (CPM). CPM tells about the tasks that are critical in the project. If there is a delay in these tasks, the project gets delayed.

Standard Deviation

Standard Deviation (SD) measures the variability or uncertainty in the estimate.

In beta distribution,

Mean = (O + 4 × M + L)/6

Standard Deviation (SD) = (L − O)/6

PERT Estimation Steps

**Step (1)** − Arrive at the WBS.

**Step (2)** − For each task, find three values most optimistic estimate (O), a most likely estimate (M), and a pessimistic estimate (L).

**Step (3)** − PERT Mean = (O + 4 × M + L)/6

PERT Mean = (O + 4 × M + L)/3

**Step (4)** − Calculate the Standard Deviation of the task.

Standard Deviation (SD) = (L − O)/6

**Step (6)** − Repeat steps 2, 3, 4 for all the tasks in the WBS.

**Step (7)** − Calculate the PERT estimate of the project.

E (Project) = ∑ E (Task)

**Step (8)** − Calculate the Standard Deviation of the project.

SD (Project) = √ (ΣSD (Task)2)

Convert the Project Estimates to Confidence Levels

PERT Estimate (E) and Standard Deviation (SD) thus calculated are used to convert the project estimates to confidence levels.

The conversion is based such that

* Confidence level in E +/– SD is approximately 68%.
* Confidence level in E value +/– 1.645 × SD is approximately 90%.
* Confidence level in E value +/– 2 × SD is approximately 95%.
* Confidence level in E value +/– 3 × SD is approximately 99.7%.

Commonly, the 95% confidence level, i.e., E Value + 2 × SD, is used for all project and task estimates.

Differences between Three-Point Estimation and PERT

Following are the differences between Three-Point Estimation and PERT −

|  |  |
| --- | --- |
| **Three-Point Estimation** | **PERT** |
| Simple average | Weighted average |
| Follows triangular Distribution | Follows beta Distribution |
| Used for small repetitive projects | Used for large non-repetitive projects, usually R&D projects. Used along with Critical Path Method (CPM) |
| E = Mean = (O + M + L)/3  This is simple average | E = Mean = (O + 4 × M + L)/6  This is weighted average |
| SD = √ [((O − E)2 + (M − E)2 + (L − E)2)/2] | SD = (L − O)/6 |

Use-Case Points

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 (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.

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 have more needs and increase the complexity of a Use-Case to a certain extent.

Other Actors, such as a user interacting through GUI 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 | .5 |
| T7 | Easy to use | .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 | .5 |  |  |
| T7 | Easy to use | .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 | .5 |
| F3 | Object-oriented experience | 1.0 |
| F4 | Lead analyst capability | .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 | .5 |  |  |
| F3 | Object-oriented experience | 1.0 |  |  |
| F4 | Lead analyst capability | .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**