Requirements Analysis Process

A requirements analysis process involves the following steps:

Step 1: Identify Key Stakeholders and End-Users

The first step of the requirements analysis process is to identify key stakeholders who are the main sponsors of the project. They will have the final say on what should be included in the scope of the project.

Next, identify the end-users of the product. Since the product is intended to satisfy their needs, their inputs are equally important.

Step 2: Capture Requirements

Ask each of the stakeholders and end-users their requirements for the new product. Here are some requirements analysis techniques that you can use to capture requirements:

1. Hold One-on-One Interviews

Interview each stakeholder and end-user individually. This technique will help you gather specific requirements.

2. Use Focus Groups

Conduct group interviews or group workshops to understand the flow of information between different stakeholders and end-users. This technique will ensure that there will be no conflict of interest later on during the project.

3. Utilize Use Cases

Use cases provide a walkthrough of the entire product through the eyes of the end-user. This technique will help visualize how the product will actually work.

4. Build Prototypes

A prototype provides users a sample look and feel of the final product. This technique will help address feasibility issues and identify problems ahead of time.

Step 3: Categorize Requirements

Since requirements can be of various types, they should be grouped to avoid confusion. Requirements are usually divided into four categories:

* Functional Requirements - Functions the product is required to perform.
* Technical Requirements - Technical issues to be considered for the successful implementation of the product.
* Transitional Requirements - Steps required to implement a new product smoothly.
* Operational Requirements - Operations to be carried out in the backend for proper functioning of the product.

Step 4: Interpret and Record Requirements

Once the requirements are categorized, determine which requirements are actually achievable and document each one of them. Here are some techniques to analyze and interpret requirements:

Define Requirements Precisely

Ensure that the requirements are clearly worded, sufficiently detailed, and related to business needs.

Prioritize Requirements

Prioritize requirements and list them out based on which ones are the “most critical” and which ones are just “nice-to-have”.

Carry Out an Impact Analysis

Carry out an impact analysis to make sure that you fully understand the consequences of the requirements.

Resolve Conflicts

Arrange a meeting with key stakeholders and resolve conflicting requirements. You can also perform a scenario analysis to explore how the requirements would work for different possible scenarios.

Analyze Feasibility

Perform a detailed analysis of the product based on the requirements gathered to determine its reliability and to identify any major problems.

Once all the requirements are analyzed, create a detailed written document and circulate it among the key stakeholders, end-users and development teams.

Step 5: Sign off

Once a final decision is made on the requirements, ensure that you get a signed agreement from the key stakeholders. This is done to ensure that there are no changes or uncontrolled growth in the scope of the project.

Now, we will learn the requirement analysis techniques.

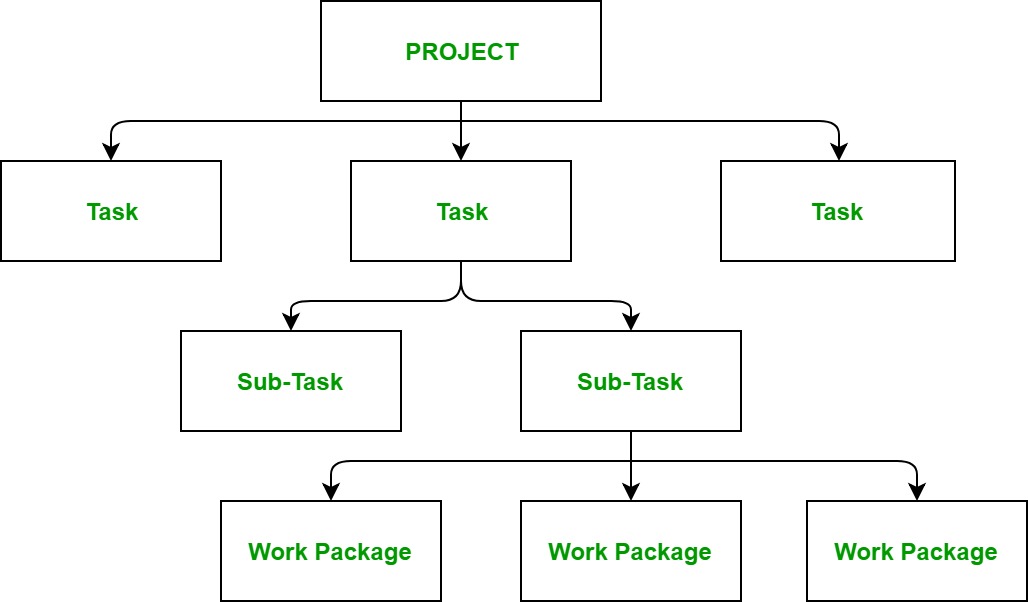
# Software Engineering | Work Breakdown Structure

* **Last Updated :** 15 Mar, 2019

A **Work Breakdown Structure** includes dividing a large and complex project into simpler, manageable and independent tasks. The root of this tree (structure) is labelled by the Project name itself. For constructing a work breakdown structure, each node is recursively decomposed into smaller sub-activities, until at the leaf level, the activities becomes undividable and independent. It follows a Top-Down approach.

**Steps:**

* **Step-1:** Identify the major activities of the project.
* **Step-2:** Identify the sub-activities of the major activities.
* **Step-3:** Repeat till undividable, simple and independent activities are created.



**Construction of Work Breakdown Structure:**  
Firstly, the project managers and top level management identifies the main deliverables of the project. After this important step, these main deliverables are broke down into smaller higher-level tasks and this complete process is done recursively to produce much smaller independent tasks. It depends on the project manager and team that upto which level of detail they want to break down their project.

Generally the lowest level tasks are the most simplest and independent tasks and takes less than two weeks worth of work. Hence, there is no rule for upto which level we may build the work breakdown structure of the project as it totally depends upon the type of project we are working on and the management of the company. The efficiency and success of the whole project majorly depends on the quality of the Work Breakdown Structure of the project and hence, it implies its importance.

**Uses:**

* It allows to do a precise cost estimation of each activity.
* It allows to estimate the time that each activity will take more precisely.
* It allows easy management of the project.
* It helps in proper organisation of the project by the top management.

Estimate Effort

**Function Points Analysis**

Function Point Analysis (FPA) is a sizing measure of clear business significance. First made public by Allan Albrecht of IBM in 1979. It depends mainly on estimation the lines of code for the software which is also considered as a critic for this technique.

FPA can be helpful to estimate the effort for a software project at the early stage when the requirements are known, but the details of implementation have not yet been specified or evaluated. Which is actually the most case of the software projects

To use the FPA, these are the steps to follow after defining the scope and decompose the system functionality and components:

1. Identify inputs, outputs, file accesses and interfaces to external systems
2. Determine the functional complexity of each function
3. Calculate unadjusted FPs by summing weightings
4. Calculate Value Adjustment Factor for the software
5. Apply VAF to UFP to calculate adjusted FPs

**Constructive Cost Model** (**COCOMO**)

The **Constructive Cost Model** (**COCOMO**) is a procedural software cost estimation model developed by Barry W. Boehm

Program size is expressed in estimated thousands of source lines of code (KLOC). COCOMO applies to three classes of software projects:

* **Organic projects** – “small” teams with “good” experience working with “less than rigid” requirements.
* **Semi-detached projects** – “medium” teams with mixed experience working with a mix of rigid and less than rigid requirements.
* **Embedded projects** – developed within a set of “tight” constraints. It is also a combination of organic and semi-detached projects.(hardware, software, operational, …)

.

COCOMO is used for estimating the development effort and time.

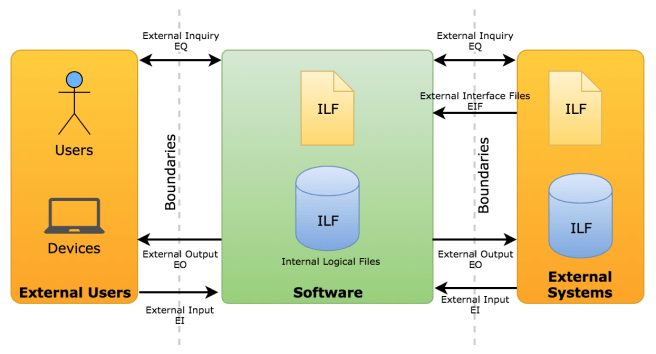
**Let us start**

Step 1

We will start with the FPA after we scoped the requirements and decompose the functions, we are ready to identify the inputs, outputs, file accesses and interfaces to external systems. FPA is measured based on these below elements:

1. **Internal Logical Files (ILF):** It is a group of logically related data that is stored and maintained within the application, for example, databases and files
2. **External Interface Files (EIF):** is a group of logically related data that will be used by the application. The difference that these data will not be maintained in the application, for example, external databases.
3. **External Input (EI):** It is mainly the data transactions which will be inserted into the application from outside the application boundary, for example, Data entry process.
4. **External Output (EO):** It is mainly the output of the system functions, for example, a transactional data into the database, messages or a report
5. **External Inquiry(EQ)**: It used to present information to a user through the retrieval of data from ILF or EIF, for example, search queries, or exporting a report

The image below, illustrate the software context based on FPA, and how other users or systems interact with our software. Now, we will need to list the 5 elements for each subsystem, component, or function to do the next step.



Software Context

FPA classifies the complexity of each function type as below

|  |  |  |  |
| --- | --- | --- | --- |
|  | Complexity | | |
| Function type | Simple (S) | Average (A) | Complex (C) |
| Internal Logical File | 7 | 10 | 15 |
| External Interface File | 5 | 7 | 10 |
| External Input | 3 | 4 | 6 |
| External Output | 4 | 5 | 7 |
| External Inquiry | 3 | 4 | 6 |

Step 2

The next step is to relate our functions to these complexity levels and apply the weightings for each one, for example, let us assume that we have the following outcome from our functional points

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Components List** | **Inputs (EI)** | **Outputs (EO)** | **Files (ILF)** | **Inquiries (EQ)** | **Interfaces (EIF)** |
| Component 1 | 1 S\*3 = 3 | 1 S\*4 = 4  2 C\*7 = 14 | 2 A\*10 = 20 | 2 S\*3 = 6 | 1 C\*10 = 10 |
| Component 2 | 2 A\*4 = 8  1 C\*6 = 6 | 3 A\*5 = 15 | 1 C\*15 = 15 | 2 A\*4 = 8 | 2 S\*5 = 10 |
| Component 3 | 3 A\*4 = 12  2 C\*6 = 12 | 3 S\*4 = 12 | 1 S\*7 = 7 | – | 2 A\*7 = 14 |

As we can see in the table, that we have 3 components and after we applied the weights for each one, we can see that each one can have more than one input for example, and we can estimate each input weight according to our judgment of this input complexity. In component 3 we have 3 average inputs and 2 complex input but we do not have any inquiries.

Step 3

You can do the same for all the software components or functions and this will lead us to the next step of calculating the unadjusted function points by summation of all weights

**Unadjusted Function Points (UFP) = (n × EI) + (n × EO) + (n × EQ) + (n × ILF) + (n × EIF)**

In the example above the **UFP = 176**

Step 4

The next step, we will need to calculate Value Adjustment Factor, the VAF consists of **14 General System Characteristics (GSCs)** which are listed below, These GSCs represent characteristics of the application under consideration how the degree of influence for each factor on the system.

|  |  |  |
| --- | --- | --- |
| # | Technical factors | Brief Description |
| F1 | Data communications | How many communication facilities are there to aid in the transfer or exchange of information with the application or system? |
| F2. | Distributed data processing | How are distributed data and processing functions handled? |
| F3. | Performance | Did the user require response time or throughput? |
| F4. | Heavily used configuration | How heavily used is the current hardware platform where the application will be executed? |
| F5. | Transaction rate | How frequently are transactions executed daily, weekly, monthly, etc.? |
| F6. | On-Line data entry | What percentage of the information is entered On-Line? |
| F7. | End-user efficiency | Was the application designed for end-user efficiency? |
| F8. | On-Line update | How many ILF’s are updated by the On-Line transaction? |
| F9. | Complex processing | Does the application have extensive logical or mathematical processing? |
| F10. | Reusability | Was the application developed to meet one or many users needs? |
| F11. | Installation Ease | How difficult are conversion and installation? |
| F12. | Operational ease | How effective and/or automated are a start-up, back up, and recovery procedures? |
| F13. | Multiple sites | How the application was specifically designed, developed, and supported to be installed at multiple sites for multiple organizations? |
| F14. | Facilitate change | Was the application specifically designed, developed, and supported to facilitate change? |

Each factor may have a value within 0 (no influence) to 5 (strong influence) to calculate the Total Degree of Influence (TDI). The VAF can vary in range from 0.65 (when all GSCs have no influence) to 1.35 (when all GSCs have strong influence) according to the equation below

**VAF = 0.65 + (∑\_(i=1)^14 Fi \* 0.01)**

**Let us assume that we considered the influence of the GSCs and we calculated the VAF as follow:**

**VAF** = 0.65 + ((1+4+5+4+1+0+2+4+3+4+4+1+1+1) \* 0.01) **= 0.65 + 0.35 = 1**

Step 5

So, this will lead us to the final step of FPA which to calculate The Adjusted Function Points:

**AFP = UFP \* VAF = 176\*1 = 176**

Step 6

Now, we are ready to use the COCOMO estimation technique, this model is based on KLOC and to obtain our software KLOC, we will use the output from the functions points analysis.

According to [Quantitative Software Management](http://www.qsm.com/resources/function-point-languages-table), they created a table contains updated function point language gearing factors for 37 distinct programming languages/technologies. We will use this table to calculate the KLOC by using this equation

**KLines of code (KLOC) = AFP \* QSM Index (programming language) / 1000**

We assume that we will use .NET programming language, according to QSM table the average of .NET is 57

**KLOC = 176 \* 57 / 1000 ≈ 10 KLOC**

Step 7

According to COCOMO Complexity, the software effort is calculated based on predetermined coefficients based on complexity and, lines of code, for example, if we considered that we are using organic project type our calculation will be as follow:

**Effort Applied** **(E)**= a\*(KLOC)^b = 3.2 \* (10) ^ 1.05 ≈ **35 Person Months**

**Development Time (T)** = c\*(Effort Applied)^d = 2.5 \* ( 35) ^ 0.38 ≈ **9.7 Months**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Software Project** | **a** | **b** | **c** | **d** |
| **Organic** | 3.2 | 1.05 | 2.5 | 0.38 |
| **Semi-detached** | 3 | 1.12 | 2.5 | 0.35 |
| **Embedded** | 2.8 | 1.20 | 2.5 | 0.32 |

**People required (P)** = Effort Applied / Development Time = 35/9.7 ≈ **+/- 3.6 Persons**

**Development Productivity =**LOC/Effort Applied = 10,000/35 ≈ **286 LOC/Person Month**

We have calculated this without calculating the Effort Adjustment Factor (EAF), Intermediate COCOMO provides 15 attributes rated on a six-point scale that ranges from “very low” to “extra high”, these 15 attributes called the cost drivers. For each one of them, you can describe how the project is related to this attribute, for example, Required development schedule which is 10 months we can select a nominal value for this attribute.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Cost Drivers | Ratings | | | | | |
|  | Very Low | Low | Nominal | High | Very High | Extra High |
| Product attributes |  |  |  |  |  |  |
| Required software reliability | 0.75 | 0.88 | 1.00 | 1.15 | 1.40 |  |
| Size of the application database |  | 0.94 | 1.00 | 1.08 | 1.16 |  |
| The complexity of the product | 0.70 | 0.85 | 1.00 | 1.15 | 1.30 | 1.65 |
| Hardware attributes |  |  |  |  |  |  |
| Run-time performance constraints |  |  | 1.00 | 1.11 | 1.30 | 1.66 |
| Memory constraints |  |  | 1.00 | 1.06 | 1.21 | 1.56 |
| The volatility of the virtual machine environment |  | 0.87 | 1.00 | 1.15 | 1.30 |  |
| Required turnabout time |  | 0.87 | 1.00 | 1.07 | 1.15 |  |
| Personal attributes |  |  |  |  |  |  |
| Analyst capability | 1.46 | 1.19 | 1.00 | 0.86 | 0.71 |  |
| Applications experience | 1.29 | 1.13 | 1.00 | 0.91 | 0.82 |  |
| Software engineer capability | 1.42 | 1.17 | 1.00 | 0.86 | 0.70 |  |
| Virtual machine experience | 1.21 | 1.10 | 1.00 | 0.90 |  |  |
| Programming language experience | 1.14 | 1.07 | 1.00 | 0.95 |  |  |
| Project attributes |  |  |  |  |  |  |
| Application of software engineering methods | 1.24 | 1.10 | 1.00 | 0.91 | 0.82 |  |
| Use of software tools | 1.24 | 1.10 | 1.00 | 0.91 | 0.83 |  |
| Required development schedule | 1.23 | 1.08 | 1.00 | 1.04 | 1.10 |  |

After identifying the weights for all cost drivers, you can multiply all of them to get the EAF. Then, we can now calculate the adjusted effort according to the below equation:

**The adjusted effort applied = a\*(KLOC)^b \* EAF**

If we assume that our EAF is 1.17 then the adjusted effort will equal to 35 \* 1.17 ≈ 41 Person Months. After that, you can recalculate all the other values again. The first calculation is called the Basic COCOMO while the second is considered the Intermediate COCOMO.

Step 8

Now, we can apply the cost estimate by calculating the cost of every staffed person \* Effort Applied, for example, if all staff have fixed 2K $ Person Month the cost will be

**2,000 \* 35 = 70K $**

**Gantt chart represents following things :** 

* All the tasks are listed at leftmost column.
* The horizontal bars indicate or represent required time by corresponding particular task.
* When occurring of multiple horizontal bars takes place at same time on calendar, then that means concurrency can be applied for performing particular tasks.
* The diamonds indicate milestones.

**Advantages :** 

* **Simplyfy Project –**   
  Gantt charts are generally used for simplifying complex projects.
* **Establish Schedule –**   
  It simply establishes initial project schedule in which it mentions who is going to do what, when, and how much time it will take to complete it.
* **Provide Efficiency –**   
  It brings efficiency in planning and allows team to better coordinate project activities.
* **Emphasize on scope –**   
  It helps in emphasizing i.e., gives importance to scope of individual tasks.
* **Ease at understanding –**   
  It makes it for easy stakeholders to understand timeline and brings clarity of dates.
* **Visualize project –**   
  It helps in clearly visualizing project management, project tasks involved.
* **Organize thoughts and Highly visible –**   
  It organizes your thoughts and can be highly visible so that everyone in enterprises can have basic level of understanding and have knowledge about what’s happening in project even if they are not involved in working.
* **Make Practical and Realistic planning –**   
  It makes the project planning practical and realistic as realistic planning generally helps to avoid any kind of delays and losses of many that can arise.

**Disadvantages :** 

* Sometimes, using Gantt chart makes project more complex.
* The size of bar chart dost not necessarily indicate amount of work done in project.
* Gantt charts and projects are needed to be updated on regular basis.
* It is not possible or difficult to view this chart on one sheet of paper. The software products that produce Gantt chart needed to be viewed on computer screen so that whole project can be seen easily.