**1. Best Guess –** This technique is purely guesswork and based on the some sort of experience.   
The method is very common, but since it is based on your gut feeling, its uncertainty contingency is probably around 200% or even higher.  
  
**2. Ad-hoc method -** The test efforts are based on tentative timeframe. The timeline set by managerial or marketing personnel or by client without any guess/experience. Alternatively, it is done until the budgeted finances run out.   
This is very common practise in extremely immature organizations and has error margins of over 100% at times.  
  
**3. Experience Based -** Analogies and experts:

* Metrics collected from previous tests.
* You already tested similar application in previous project.
* Inputs are taken from Subject Matter experts who know the application (as well as testing) very well.

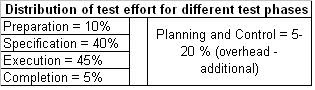
**4. Work Breakdown Structure -** It is created by breaking down the test project into small pieces. Modules are divided into sub-modules. Sub modules are further divided into functionalities and functionalities are divided in sub-functionalities.  
Review all the requirements from Requirement Document to make sure they are added in WBS. Now you figure out the number of tasks your team needs to complete. Estimate the duration of each task.  
  
**5. Delphi technique –** Same as above WBS. Here functionalities and each task is allocated to each team member. Then team member gives estimate that they will take this much hours to complete the task.  
Averagely, this technique gives good confidence in the estimation. This technique can be combined with other techniques.  
  
**6. Three-point estimation –** This technique is based on statistical methods In this technique, task is broken down into subtasks (similar to WBS) and then three types on estimation are done on each chunk –

* Optimistic Estimate (Best case scenario in which nothing goes wrong and all conditions are optimal.) = a
* Most Likely Estimate (most likely duration and there may be some problem but most of the things will go right.) = m
* Pessimistic Estimate (worst case scenario which everything goes wrong.) = b

Formula to find Value for Estimate (E) = a + (4\*m) + b / 6   
Standard Deviation (SD) = = (b - a)/6   
More information about this technique can be find at:   
http://www.projects.ed.ac.uk/methodologies/Full\_Software\_Project\_Template/EstimationGuidelines.shtml  
  
  
**7. Function Point/Testing Point Analysis:** The FP technique is a direct indicator of the functionality of software application from the user's perspective. This is the most accepted technique used to estimate the size of a software project.   
**This technique is a part of TMap. Base of this technique is function point technique. Here we convert function points into test points. In Test Point analysis, we usually carry out the following:**

* Dynamic Test Points
* Static Test Points
* Environmental Factor
* Productivity Factor
* Primary Test Hours
* Control Factor
* Total Test Hours

You can download the detailed self explained presentation here –   
http://www.eurostarconferences.com/community/member/eurostar-presentations-archive/test-effort-estimation-with-test-point-analysis.aspx  
  
In Testing, **This estimation is based on requirement specification document, or a previously created prototype of the application**. To calculate FP for a project, some major components are required.   
The major components are:   
**Unadjusted Data Function Points: i. Internal Files, ii. External Interfaces**   
**Unadjusted Transaction Function Points: User Inputs, ii. User Outputs & iii. User Inquiries**   
Capers Jones basic formula:   
Number of Test cases = [Number of Function Points] x 1.2   
Total Actual Effort, TAE = (Number of Test cases) \* (**Percentage of development effort**/100)   
**This method is done in a case when**a detailed low level design document or requirement document is available (i.e measure of function point is available) & Previous data for development and testing is available. But now days, when we are using agile and iterative methodologies to deliver projects, so most of the times all this documentation is not available.  
  
**8. Percentage of development effort method**   
Here the assumption is that a more complex business application may require more testing effort. The test effort required is a direct proportionate or percentage of the development effort.  
  
  
**Note:** The development effort can be estimated using line of code (LOC) or function point (FP) which is not in the our scope.   
Example:   
If a previous project with 500 FPs required 50 man hours for testing, the percentage of testing effort is calculated as:   
P = (50 / 500) \* 100 =10%   
For the current project with a development effort, say 1500 FPs, the testing effort is:   
Total Actual Effort, TAE = 1500 \* (P/100) = 1500 \* (10/100) = 150 man hours.  
  
**9. Percentage distribution** – Here all the phases of SDLC are divided in parts and assigned effort in %. Like –   
Project management 7%   
Requirements 9%   
Design 16%   
Coding 26%   
Test (all test phases) 27%   
Documentation 9%   
Installation and training 6%   
Now testing % is further distributed into all testing phases:

[](http://4.bp.blogspot.com/_vdqOsYKAf0Y/S86aszRR4zI/AAAAAAAAA6w/dimN0MuQaDE/s1600/Distribution+of+test+efforts.JPG)

OR

|  |  |
| --- | --- |
| **All phases** | **%** |
| Component testing | 16 |
| Independent testing | 84 |
| **Total** | **100** |
|  |  |
| **Independent testing** | **%** |
| Integration testing | 24 |
| System testing | 52 |
| Acceptance testing | 24 |
| **Total** | 100 |
|  |  |
| **System testing** | **%** |
| Functional system testing | 65 |
| Non-functional system testing | 35 |
| **Total** | 100 |
|  |  |
| Test Planning and Design Architecture | 50% |
| Review phase | 50% |

**10.** **Use case point estimation method:**Use case point (UCP) method is gaining popularity because now-a-days application development is modelled around use case specification. The test case development is normally kicked off after baseline use case. So the various factors in use case give a direct proportion to the testing effort.    
Use case is a document which well specifies different users, systems or other stakeholders interacting with the concerned application. They are named as ‘Actors’. The interactions accomplish some defined goals protecting the interest of all stakeholders through different behaviour or flow termed as scenarios.  
  
**UCP Estimation Method in brief**   
**1.** Obtain unadjusted actor weight (UAW)  
**2.** Determine unadjusted use case weight (UUCW)   
**3.** Calculate unadjusted use case points (UUCP) UUCP = UAW + UUCW  
**4.** Determine the technical/environmental factor (TEF)  
**5.** Compute the adjusted use case point (AUCP) AUCP = UUCP \* [ 0.65 + (0.01 \* TEF)]   
**6.** Arrive at final effort using a conversion factor Total Actual Effort = AUCP \* CF  
  
**Conclusion**   
There can’t be a sole hard and fast rule for estimating the testing effort for a project. There may be different other methods also which can be effectively used for the purpose. It is recommended to add on to the possible knowledge base of test estimation methods and estimation templates constantly revised based upon new findings.