Copyright 2018 Carnegie Mellon University. All rights reserved.  
  
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.  
  
Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.  
  
NO WARRANTY. THIS MATERIAL IS FURNISHED ON AN “AS-IS” BASIS WITH NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, ANY WARRANTY WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT, OR THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.  
  
[Distribution Statement A] This material has been approved for public release and unlimited distribution. The United States Government has Unlimited Rights in this material as defined by DFARS 252.227-7013.

The text and illustrations in this material are licensed by Carnegie Mellon University under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

The Creative Commons license does not extend to logos, trade marks, or service marks of Carnegie Mellon University.



TSP Planning Guidelines

|  |  |
| --- | --- |
| **General** | If your team does not have historical data, use these guidelines to support your planning. As you accumulate historical TSP data, develop guidelines that reflect your team’s performance.  In all cases, use your judgment. If a guideline does not fit your situation, develop your own estimate instead. |
| **TSP Plans** | * The overall plan is initially produced by the entire team. It is later adjusted as a result of load balancing. * The bottom-up plan is produced by each team member for the next phase. * The balanced plan is the next-phase plan that results after load balancing. |
| **Requirements** | * The requirements process depends heavily on the nature of the project, so general guidelines are not provided in this document. * In general, expect that all input materials will have to be inspected and substantially reworked. |
| **Requirements Inspections** | * The QUAL guideline provides a general factor to use in determining the time needed for the requirements inspection. * If this number is unrealistic for your project and situation, use your best estimate. |
| **High-Level Design** | * General guidelines are not provided for high-level design because the process is highly variable. However, the time required for high-level design is principally a function of the size, complexity, and general nature of the system to be developed. * Without prior data, many teams underestimate the time required for high-level design. In many cases, the underestimate is substantial. |
| **Implementation  Phase** | Implementation covers detailed design through unit test.   * Calculate total implementation time from LOC/hour rates. * New or large modifications: about 10 LOC per hour * Small changes to large systems: about five LOC per hour * Select a value that seems appropriate for your project. * Maintenance fixes: five to 20 hours per fix depending on complexity and degree of testing. |
| **Implementation Allocation** | To estimate the implementation phases, make an overall estimate of implementation based on a LOC/hour rate, then use the following percentages to calculate time allocated to each phase.   |  |  | | --- | --- | | Detailed design | 22.1% | | Detailed design review | 11.1% | | Detailed design inspection | 8.8% | | Coding | 20.0% | | Code review | 10.0% | | Compiling | 3.4% | | Code inspection | 8.8% | | Unit test | 15.8% | |

|  |  |
| --- | --- |
| **Integration and  System Test** | For integration and test phases, estimate the defect-free test time. Then, add five hours per defect for integration test, and 10 hours per defect for system test. |