|  |  |
| --- | --- |
| PSP2.1 Process Script | |
| Purpose | To guide the development of module-level programs |
| Entry Criteria | * Problem description * PSP2.1 Project Plan Summary form * Size Estimating template * Historical size and time data (estimated and actual) * Time and Defect Recording logs * Defect Type, Coding, and Size Measurement standards * Stopwatch (optional) |

|  |  |  |
| --- | --- | --- |
| Step | Activities | Description |
| 1 | Planning | * Produce or obtain a requirements statement. * Use the PROBE method to estimate the added and modified size ***and the size prediction* interval** of this program. * Complete the Size Estimating template. * Use the PROBE method to estimate the required development time ***and the time prediction interval***. * Complete a Task Planning template. * Complete a Schedule Planning template. * Enter the plan data in the Project Plan Summary form. * Complete the Time Recording log. |
| 2 | Development | * Design the program. * ***Document the design in the design templates.*** * Review the design, and fix and log all defects found. * Implement the design. * Review the code, and fix and log all defects found. * Compile the program, and fix and log all defects found. * Test the program, and fix and log all defects found. * Complete the Time Recording log. |
| 3 | Postmortem | Complete the Project Plan Summary form with actual time, defect, and size data. |

|  |  |
| --- | --- |
| Exit Criteria | * A thoroughly tested program * Completed Project Plan Summary form with estimated and actual data * Completed Size Estimating and Task and Schedule Planning templates * ***Completed Design templates*** * Completed Design Review and Code Review checklists * Completed Test Report template * Completed PIP forms * Completed Time and Defect Recording logs |

|  |  |
| --- | --- |
| PSP2.1 Planning Script | |
| Purpose | To guide the PSP planning process |
| Entry Criteria | * Problem description * PSP2.1 Project Plan Summary form * Size Estimating, Task Planning, and Schedule Planning templates * Historical size and time data (estimated and actual) * Time Recording log |

|  |  |  |
| --- | --- | --- |
| Step | Activities | Description |
| 1 | Program  Requirements | * Produce or obtain a requirements statement for the program. * Ensure that the requirements statement is clear and unambiguous. * Resolve any questions. |
| 2 | Size  Estimate | * Produce a program conceptual design. * Use the PROBE method to estimate the added and modified size of this program. * Complete the Size Estimating template and Project Plan Summary form. * ***Calculate the 70% size prediction interval. (You may use a spreadsheet.)*** |
| 3 | Resource  Estimate | * Use the PROBE method to estimate the time required to develop this program. * ***Calculate the 70% size prediction interval. (You may use a spreadsheet.)*** * Using the *To Date %* from the most recently developed program as a guide, distribute the development time over the planned project phases. |
| 4 | Task and  Schedule Planning | For projects lasting several days or more, complete the Task Planning and Schedule Planning templates. |
| 5 | Defect  Estimate | * Based on your to-date data on defects per added and modified size unit, estimate the total defects to be found in this program. * Based on your *To Date %* data, estimate the number of defects to be injected and removed by phase. |

|  |  |
| --- | --- |
| Exit Criteria | * Documented requirements statement * Program conceptual design * Completed Size Estimating template * For projects lasting several days or more, completed Task and Schedule Planning templates * Completed Project Plan Summary form with estimated program size, development time, and defect data, ***and the time and size prediction intervals*** * Completed Time Recording log |

|  |  |
| --- | --- |
| PSP2.1 Development Script | |
| Purpose | To guide the development of small programs |
| Entry Criteria | * Requirements statement * Project Plan Summary form with estimated program size and development time * For projects lasting several days or more, completed Task Planning and Schedule Planning templates * Time and Defect Recording logs * Defect Type standard and Coding standard |

|  |  |  |
| --- | --- | --- |
| Step | Activities | Description |
| 1 | Design | * Review the requirements and produce ***an external specification to meet them.*** * ***Complete Functional and Operational Specification templates to record this specification.*** * ***Produce a design to meet this specification.*** * ***Record the design in Functional, Operational, State, and Logic Specification templates.*** * Record in the Defect Recording log any requirements defects found. * Record time in the Time Recording log. |
| 2 | Design  Review | * Follow the Design Review script and checklist and review the design. * Fix all defects found. * Record defects in the Defect Recording log. * Record time in the Time Recording log. |
| 3 | Code | * Implement the design following the Coding standard. * Record in the Defect Recording log any requirements or design defects found. * Record time in the Time Recording log. |
| 4 | Code  Review | * Follow the Code Review script and checklist and review the code. * Fix all defects found. * Record defects in the Defect Recording log. * Record time in the Time Recording log. |
| 5 | Compile | * Compile the program until there are no compile errors. * Fix all defects found. * Record defects in the Defect Recording log. * Record time in the Time Recording log. |
| 6 | Test | * Test until all tests run without error. * Fix all defects found. * Record defects in the Defect Recording log. * Record time in the Time Recording log. * Complete a Test Report template on the tests conducted and the results obtained. |

|  |  |
| --- | --- |
| Exit Criteria | * A thoroughly tested program that conforms to the Coding standard * ***Completed Design templates*** * Completed Design Review and Code Review checklists * Completed Test Report template * Completed Time and Defect Recording logs |

|  |  |
| --- | --- |
| PSP2.1 Design Review Script | |
| Purpose | To guide you in reviewing detailed designs |
| Entry Criteria | * Completed program design ***documented with the PSP Design templates*** * Design Review checklist * Design standard * Defect Type standard * Time and Defect Recording logs |
| General | Where the design was previously verified, check that the analyses   * covered all of the design * were updated for all design changes * are correct * are clear and complete |

|  |  |  |
| --- | --- | --- |
| Step | Activities | Description |
| 1 | Preparation | * Examine the program and checklist and decide on a review strategy. * ***Examine the program to identify its state machines, internal loops, and variable and system limits.*** * ***Use a trace table or other analytical method to verify the correctness of the design.*** |
| 2 | Review | * Follow the Design Review checklist. * Review the entire program for each checklist category; do not try to review for more than one category at a time! * Check off each item as you complete it. * Complete a separate checklist for each product or product segment reviewed. |
| 3 | Fix Check | * Check each defect fix for correctness. * Re-review all changes. * Record any fix defects as new defects and, where you know the defective defect number, enter it in the fix defect space. |

|  |  |
| --- | --- |
| Exit Criteria | * A fully reviewed detailed design * One or more Design Review checklists for every design reviewed * ***Documented design analysis results*** * All identified defects fixed and all fixes checked * Completed Time and Defect Recording logs |

|  |  |
| --- | --- |
| PSP2.1 Postmortem Script | |
| Purpose | To guide the PSP postmortem process |
| Entry Criteria | * Problem description and requirements statement * Project Plan Summary form with program size, development time, and defect data * For projects lasting several days or more, completed Task Planning and Schedule Planning templates * Completed Test Report template * ***Completed Design templates*** * Completed Design Review and Code Review checklists * Completed Time and Defect Recording logs * A tested and running program that conforms to the coding and size measurement standards |

|  |  |  |
| --- | --- | --- |
| Step | Activities | Description |
| 1 | Defect Recording | * Review the Project Plan Summary to verify that all of the defects found in each phase were recorded. * Using your best recollection, record any omitted defects. |
| 2 | Defect Data Consistency | * Check that the data on every defect in the Defect Recording log are accurate and complete. * Verify that the numbers of defects injected and removed per phase are reasonable and correct. * Determine the process yield and verify that the value is reasonable and correct. * Using your best recollection, correct any missing or incorrect defect data. |
| 3 | Size | * Count the size of the completed program. * Determine the size of the base, reused, deleted, modified, added, total, added and modified, and new reusable code. * Enter these data in the Project Plan Summary form. |
| 4 | Time | * Review the completed Time Recording log for errors or omissions. * Using your best recollection, correct any missing or incomplete time data. |

|  |  |
| --- | --- |
| Exit Criteria | * A thoroughly tested program that conforms to the coding and size measurement standards * ***Completed Design templates*** * Completed Design Review and Code Review checklists * Completed Test Report template * Completed Project Plan Summary form * Completed PIP forms describing process problems, improvement suggestions, and lessons learned * Completed Time and Defect Recording logs |

|  |  |  |
| --- | --- | --- |
| PSP2.1 Plan Summary Instructions | |  |
| Purpose | To hold the plan and actual data for programs or program parts | |
| General | * Use the most appropriate size measure, either LOC or element count. * “To Date” is the total actual to-date values for all products developed. * A part could be a module, component, product, or system. | |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. | |
| Summary | * Enter the added and modified size per hour planned, actual, and to-date. * Enter the planned and actual times for this program and prior programs. * For planned time to date, use the sum of the current planned time and the to-date planned time for the most recent prior program. * CPI = (To Date Planned Time)/(To Date Actual Time). * Reuse % is reused size as a percentage of total program size. * New Reusable % is new reusable size as a percentage of added and modified size. * Enter the test and total defects/KLOC or other appropriate measure. * Enter the planned, actual, and to-date yield before compile. | |
| *Quality Indicators* | * ***Appraisal COQ: the percentage of development time in reviews.*** * ***Failure COQ: the percentage of development time in compile and test.*** * ***A/FR: the ratio of appraisal to failure COQ.*** * ***Enter the planned, actual, and to-date PQI (the process quality index)*** | |
| Program Size | * Enter plan base, deleted, modified, reused, new reusable, and total size from the Size Estimating template. * Enter the plan added and modified size value (A+M) from projected added and modified size (P) on the Size Estimating template. * Calculate plan added size as A+M – M. * Enter estimated proxy size (E) from the Size Estimating template. * Enter actual base, deleted, modified, reused, total, and new reusable size from the Size Estimating template. * Calculate actual added size as T-B+D-R and actual added and modified size as A+M. * Enter to-date reused, added and modified, total, and new reusable size. | |
| Time in Phase | * Enter plan total time in phase from the estimated total development time on the Size Estimating template. * Distribute the estimated total time across the development phases according to the To Date % for the most recently developed program. * Enter the actual time by phase and the total time. * To Date: Enter the sum of the actual times for this program plus the to-date times from the most recently developed program. * To Date %: Enter the percentage of to-date time in each phase. | |
| *Prediction Interval* | * ***Enter the 70% UPI and LPI total size and time ranges.*** | |

**(continued)**

|  |  |  |
| --- | --- | --- |
| PSP2.1 Plan Summary Instructions (continued) | |  |
| Defects Injected | * Enter the total estimated defects injected. * Distribute the estimated total defects across the development phases according to the To Date % for the most recently developed program. * Enter the actual defects by phase and the total actual defects. * To Date: Enter the sum of the actual defects injected by phase and the to-date values for the most recent previously developed program. * To Date %: Enter the percentage of the to-date defects injected by phase. | |
| Defects Removed | * Enter the estimated total defects removed. * Distribute the estimated total defects across the development phases according to the To Date % for the most recently developed program. * To Date: Enter the actual defects removed by phase plus the To Date values for the most recent previously developed program. * To Date %: Enter the percentage of the To Date defects removed by phase. * After development, record any defects subsequently found during program testing, use, reuse, or modification. | |
| Defect-Removal Efficiency | * Calculate and enter the defects removed per hour in design review, code review, compile, and test. * For DRL, take the ratio of the review and compile rates with test. * Where there were no test defects, use the to-date test defect/hour value. | |

|  |  |
| --- | --- |
| PROBE Estimating Script | |
| Purpose | To guide the size and time estimating process using the PROBE method |
| Entry Criteria | * Requirements statement * Size Estimating template and instructions * Size per item data for part types * Time Recording log * Historical size and time data |
| General | * This script assumes that you are using added and modified size data as the size-accounting types for making size and time estimates. * If you choose some other size-accounting types, replace every “added and modified” in this script with the size-accounting types of your choice. |

|  |  |  |
| --- | --- | --- |
| Step | Activities | Description |
| 1 | Conceptual Design | Review the requirements and produce a conceptual design. |
| 2 | Parts Additions | Follow the Size Estimating Template instructions to estimate the parts additions and the new reusable parts sizes. |
| 3 | Base Parts and Reused Parts | * For the base program, estimate the size of the base, deleted, modified, and added code. * Measure and/or estimate the side of the parts to be reused. |
| 4 | Size Estimating Procedure | * If you have sufficient estimated proxy size and actual added and modified size data (three or more points that correlate), use procedure 4A. * If you do not have sufficient estimated data but have sufficient plan added and modified and actual added and modified size data (three or more points that correlate), use procedure 4B. * If you have insufficient data or they do not correlate, use procedure 4C. * If you have no historical data, use procedure 4D. |
| 4A | Size Estimating Procedure 4A | * Using the linear-regression method, calculate theand  parameters from the estimated proxy size and actual added and modified size data. * If the absolute value ofis not near 0 (less than about 25% of the expected size of the new program), oris not near 1.0 (between about 0.5 and 2.0), use procedure 4B. |
| 4B | Size Estimating Procedure 4B | * Using the linear-regression method, calculate theandparameters from the plan added and modified size and actual added and modified size data. * If the absolute value of is not near 0 (less than about 25% of the expected size of the new program), oris not near 1.0 (between about 0.5 and 2.0), use procedure 4C. |
| 4C | Size Estimating Procedure 4C | If you have any data on plan added and modified size and actual added and modified size, set= 0 and  = (actual total added and modified size to date/plan total added and modified size to date). |
| 4D | Size Estimating Procedure 4D | If you have no historical data, use your judgment to estimate added and modified size. |

**(continued)**

|  |  |  |
| --- | --- | --- |
| PROBE Script (continued) | | |
| Step | Activities | Description |
| 5 | Time Estimating Procedure | * If you have sufficient estimated proxy size and actual development time data (three or more points that correlate), use procedure 5A. * If you do not have sufficient estimated size data but have sufficient plan added and modified size and actual development time data (three or more points that correlate), use procedure 5B. * If you have insufficient data or they do not correlate, use procedure 5C. * If you have no historical data, use procedure 5D. |
| 5A | Time Estimating Procedure 5A | * Using the linear-regression method, calculate theand parameters from the estimated proxy size and actual total development time data. * Ifis not near 0 (substantially smaller than the expected development time for the new program), oris not within 50% of 1/(historical productivity), use procedure 5B. |
| 5B | Time Estimating Procedure 5B | * Using the linear-regression method, calculate theand regression parameters from the plan added and modified size and actual total development time data. * Ifis not near 0 (substantially smaller than the expected development time for the new program), oris not within 50% of 1/(historical productivity), use procedure 5C. |
| 5C | Time Estimating Procedure 5C | * If you have data on estimated – added and modified size and actual development time, set= 0 and  = (actual total development time to date/estimated – total added and modified size to date). * If you have data on plan – added and modified size and actual development time, set= 0 and  = (actual total development time to date/plan total added and modified size to date). * If you only have actual time and size data, set= 0 and  = (actual total development time to date/actual total added and modified size to date). |
| 5D | Time Estimating Procedure 5D | If you have no historical data, use your judgment to estimate the development time from the estimated added and modified size. |
| 6 | Time and Size Prediction Intervals | * If you used regression method A or B, calculate the 70% prediction intervals for the time and size estimates. * If you did not use the regression method or do not know how to calculate the prediction interval, calculate the minimum and maximum development time estimate limits from your historical maximum and minimum productivity for the programs written to date. |

|  |  |
| --- | --- |
| Exit Criteria | * Completed estimated and actual entries for all pertinent size categories * Completed PROBE Calculation Worksheet with size and time entries * Plan and actual values entered on the Project Plan Summary |

Time Recording Log Instructions

|  |  |
| --- | --- |
| Purpose | * Use this form to record the time you spend on each project activity. * For the PSP, phases often have only one activity; larger projects usually have multiple activities in a single process phase. * These data are used to complete the Project Plan Summary. * Keep separate logs for each program. |
| General | * Record all of the time you spend on the project. * Record the time in minutes. * Be as accurate as possible. * If you need additional space, use another copy of the form. * If you forget to record the starting, stopping, or interruption time for an activity, promptly enter your best estimate. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Project | Enter the program name or number. |
| Phase | Enter the name of the phase for the activity you worked on, e.g. Planning, Design, Test. |
| Start Date and Time | Enter the date and time when you start working on a process activity. |
| Interruption Time | * Record any interruption time that was not spent on the process activity. * If you have several interruptions, enter their total time. * You may enter the reason for the interrupt in comments. |
| Stop Date and Time | Enter the date and time when you stop working on that process activity. |
| Delta Time | Enter the clock time you actually spent working on the process activity, less the interruption time. |
| Comments | Enter any other pertinent comments that might later remind you of any unusual circumstances regarding this activity. |

PSP Defect Recording Log Instructions

|  |  |
| --- | --- |
| Purpose | * Use this form to hold data on the defects that you find and correct. * These data are used to complete the Project Plan Summary form. |
| General | * Record each defect separately and completely. * If you need additional space, use another copy of the form. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Project | * Give each program a different name or number. * For example, record test program defects against the test program. |
| Date | Enter the date on which you found the defect. |
| Number | * Enter the defect number. * For each program or module, use a sequential number starting with 1 (or 001, etc.). |
| Type | * Enter the defect type from the defect type list summarized in the top left corner of the form. * Use your best judgment in selecting which type applies. |
| Inject | * Enter the phase when this defect was injected. * Use your best judgment. |
| Remove | Enter the phase during which you fixed the defect. (This will generally be the phase when you found the defect.) |
| Fix Time | * Enter the time that you took to find and fix the defect. * This time can be determined by stopwatch or by judgment. |
| Fix Ref. | * If you or someone else injected this defect while fixing another defect, record the number of the improperly fixed defect. * If you cannot identify the defect number, enter an X. |
| Description | Write a succinct description of the defect that is clear enough to later remind you about the error and help you to remember why you made it. |

PSP Defect Type Standard

|  |  |  |
| --- | --- | --- |
| **Type Number** | **Type Name** | **Description** |
| 10 | Documentation | Comments, messages |
| 20 | Syntax | Spelling, punctuation, typos, instruction formats |
| 30 | Build, Package | Change management, library, version control |
| 40 | Assignment | Declaration, duplicate names, scope, limits |
| 50 | Interface | Procedure calls and references, I/O, user formats |
| 60 | Checking | Error messages, inadequate checks |
| 70 | Data | Structure, content |
| 80 | Function | Logic, pointers, loops, recursion, computation, function defects |
| 90 | System | Configuration, timing, memory |
| 100 | Environment | Design, compile, test, or other support system problems |

PSP Process Improvement Proposal (PIP) Instructions

|  |  |
| --- | --- |
| Purpose | * To provide a way to record process problems and improvement ideas * To provide an orderly record of your process improvement ideas * To record any other noteworthy observations |
| General | Use the PIP form to   * record process improvement ideas as they occur to you * establish priorities for your improvement plans * describe lessons learned and unusual conditions   Keep PIP forms on hand while using the PSP.   * Record process problems even without proposed solutions. * Submit a PIP with each PSP assignment report. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Problem Description | Briefly describe any problems or experiences that led to this PIP. |
| Proposal Description | Describe the proposed improvement as explicitly as possible. |
| Other Notes and Comments | Briefly describe any other observations or facts that would later help you to   * remember what you did while writing this program * remember an idea for a future improvement * explain to your instructor something you did and why you did it |

|  |  |
| --- | --- |
| C++ Coding Standard | |
| Purpose | To guide implementation of C++ programs |
| Program Headers | Begin all programs with a descriptive header. |
| Header Format | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* Program Assignment: the program number \*/  /\* Name: your name \*/  /\* Date: the date you started developing the program \*/  /\* Description: a short description of the program and what it does \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| Listing Contents | Provide a summary of the listing contents |
| Contents Example | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* Listing Contents: \*/  /\* Reuse instructions \*/  /\* Modification instructions \*/  /\* Compilation instructions \*/  /\* Includes \*/  /\* Class declarations: \*/  /\* CData \*/  /\* ASet \*/  /\* Source code in c:/classes/CData.cpp: \*/  /\* CData \*/  /\* CData() \*/  /\* Empty() \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| Reuse Instructions | * Describe how the program is used: declaration format, parameter values, types, and formats. * Provide warnings of illegal values, overflow conditions, or other conditions that could potentially result in improper operation. |
| Reuse Instruction Example | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* Reuse instructions \*/  /\* int PrintLine(char \*line\_of\_character) \*/  /\* Purpose: to print string, ‘line\_of\_character’, on one print line \*/  /\* Limitations: the line length must not exceed LINE\_LENGTH \*/  /\* Return 0 if printer not ready to print, else 1 \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| Identifiers | Use descriptive names for all variable, function names, constants, and other identifiers. Avoid abbreviations or single-letter variables. |
| Identifier Example | Int number\_of\_students; /\* This is GOOD \*/  Float: x4, j, ftave; /\* This is BAD \*/ |
| Comments | * Document the code so the reader can understand its operation. * Comments should explain both the purpose and behavior of the code. * Comment variable declarations to indicate their purpose. |
| Good Comment | If(record\_count > limit) /\* have all records been processed? \*/ |
| Bad Comment | If(record\_count > limit) /\* check if record count exceeds limit \*/ |
| Major Sections | Precede major program sections by a block comment that describes the processing done in the next section. |
| Example | /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  /\* The program section examines the contents of the array ‘grades’ and calcu- \*/  /\* lates the average class grade. \*/  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/ |
| Blank Spaces | * Write programs with sufficient spacing so they do not appear crowded. * Separate every program construct with at least one space. |

**(continued)**

|  |  |
| --- | --- |
| C++ Coding Standard (continued) | |
| Indenting | * Indent each brace level from the preceding level. * Open and close braces should be on lines by themselves and aligned. |
| Indenting Example | while (miss\_distance > threshold)  {  success\_code = move\_robot (target \_location);  if (success\_code == MOVE\_FAILED)  {  printf(“The robot move has failed.\n”);  }  } |
| Capitalization | * Capitalize all defines. * Lowercase all other identifiers and reserved words. * To make them readable, user messages may use mixed case. |
| Capitalization Examples | #define DEFAULT-NUMBER-OF-STUDENTS 15  int class-size = DEFAULT-NUMBER-OF-STUDENTS; |

Test Report Template Instructions

|  |  |
| --- | --- |
| Purpose | * To maintain a record of the tests run and the results obtained * To be sufficiently complete so that you can later re-run the same tests and get the same results * To facilitate regression testing of modified or reused programs |
| General | * Expand this table or use multiple copies as needed. * Report all the tests that were successfully run. * Be as brief and concise as possible. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Test Name/Number | Uniquely identify each test for each program.   * the same tests with different data * the same data with different tests |
| Test Objective | Briefly describe the objective of the test. |
| Test Description | Describe each test’s data and procedures in sufficient detail to facilitate its later use as a regression test. |
| Test Conditions | * List any special configuration, timing, fix, or other conditions of the test. * When multiple tests are run with different parameters or under varying conditions, separately list each. |
| Expected Results | List the results that the test should produce if it runs properly. |
| Actual Results | List the results that were actually produced. |

Size Estimating Template Instructions

|  |  |
| --- | --- |
| Purpose | Use this form with the PROBE method to make size estimates. |
| General | * A part could be a module, component, product, or system. * Where parts have a substructure of methods, procedures, functions, or similar elements, these lowest-level elements are called items. * Size values are assumed to be in the unit specified in size measure. * Avoid confusing base size with reuse size. * Reuse parts must be used without modification. * Use base size if additions, modifications, or deletions are planned. * If a part is estimated but not produced, enter its actual values as zero. * If a part is produced that was not estimated, enter it using zero for its planned values. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. * Enter the size measure you are using. |
| Base Parts | If this is a modification or enhancement of an existing product   * measure and enter the base size (more than one product may be entered as base) * estimate and enter the size of the deleted, modified, and added size to the base program   After development, measure and enter the actual size of the base program and any deletions, modifications, or additions. |
| Parts Additions | If you plan to add newly developed parts   * enter the part name, type, number of items (or methods), and relative size * for each part, get the size per item from the appropriate relative size table, multiply this value by the number of items, and enter in estimated size * put an asterisk next to the estimated size of any new-reusable additions   After development, measure and enter   * the actual size of each new part or new part items * the number of items for each new part |
| Reused Parts | If you plan to include reused parts, enter the   * name of each unmodified reused part * size of each unmodified reused part   After development, enter the actual size of each unmodified reused part. |

Task Planning Template Instructions

|  |  |
| --- | --- |
| Purpose | * To estimate the development time for each project task * To compute the planned value for each project task * To estimate the planned completion date for each task * To provide a basis for tracking schedule progress even when the tasks are not completed in the planned order |
| General | * Complete the Schedule Planning and Task Planning templates together. * Select tasks that have explicit completion criteria, i.e., plan completed, program compiled and defects corrected, etc. * Expand this template or use multiple pages as needed. * Include every significant task. * Use task names and numbers that support the activity and are consistent with the project work breakdown structure. * Note that most support tools will do the earned-value calculations. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Program/Part | Enter the program or part to which the task relates. |
| Phase | Enter the phase for each task. |
| Task Name | Enter task names and/or numbers in the order in which you expect to complete them. |
| Plan: Task Hours | Enter the total planned hours for each task. |
| Plan: Cumulative Task Hours | Enter the cumulative sum of the total planned task hours. |
| Plan: Week Due | If the task has a specific due date, enter the week due here. |
| Plan: Week | * On the Schedule template, find the plan cumulative schedule hours entry that equals or just exceeds each cumulative task hours entry on this form. * The week number in that row of the Schedule template is the plan week number for the task on Task template. * If several weeks on the Schedule template have the same cumulative value, enter the earliest week number. |
| Plan: Week Predicted | * On the Schedule template, find the predicted cumulative earned value entry that equals or just exceeds each cumulative planned value entry on this form. * The week number in that row of the Schedule template is the predicted week number for the task on the Task template. * If several weeks on the Schedule template have the same cumulative value, enter the earliest week number. |
| Plan: Planned Value (PV) | * Total the planned hours for all tasks. * Find the percentage each task's planned hours is of total hours. * Enter this percentage as the planned value for each task. |
| Plan: Cumulative PV | Enter the cumulative sum of the planned values. |
| Actual: Task Hours | When a task is completed, enter the hours spent on the task. |
| Actual: Cumulative Earned Value (EV) | * Each week, total the EV for all completed tasks and enter that total beside the latest completed task. * Also enter the weekly and cumulative total EV on the Schedule template. |
| Actual: Week | As a task is completed, enter the week number it was completed. |

Schedule Planning Template Instructions

|  |  |
| --- | --- |
| Purpose | * To record the estimated and actual hours expended by calendar period * To relate the task planned value to the calendar schedule |
| General | * Expand this template or use multiple pages as needed. * Complete in conjunction with the Task Planning template. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Week No. | * From the project start, enter a week number, typically starting with 1. * For very small projects, it may be more convenient to use days instead of weeks. |
| Date | * Enter the calendar date for each week. * Pick a standard day in the week (for example, Monday). |
| Plan: Schedule Hours | * Enter the planned number of schedule hours that you expect to spend working on the project each week. * Consider non-work time such as vacations, holidays, etc. * Consider other committed activities such as e-mail, courses, meetings, and other projects. |
| Plan: Cumulative Schedule Hours | Enter the planned cumulative schedule hours through each week. |
| Plan: Cumulative Planned Value | For each week   * take the plan cumulative schedule hours from the Schedule template * on the Task template, find the task with nearest equal or lower plan cumulative task hours and note its plan cumulative value * enter this cumulative value in the Schedule template for that week * if the cumulative value for the prior week still applies, enter it again |
| Actual: Schedule Hours | At the end of each week, enter the actual schedule hours worked in that week. |
| Actual: Cumulative Schedule Hours | At the end of each week, calculate and enter the actual cumulative schedule hours for the week. |
| Actual: Week Earned Value | At the end of each week, calculate the total earned value for each task completed during the week and enter here. |
| Actual: Cumulative Earned Value | At the end of each week, calculate the cumulative earned value for the week. |
| Predicted: Cumulative Predicted Earned Value | At the end of each week, recalculate the cumulative predicted earned value for the current week through to the end of the schedule.   * Enter the current week’s actual cumulative earned value as the current week’s cumulative predicted earned value. * Calculate the average actual earned value per hour worked on the job to date (Actual Cumulative EV/Actual Cumulative Schedule Hours). * For each week n, starting with the next week, multiply the average earned value per planned hour by the planned hours for week n. Add the result to the cumulative predicted earned value for the preceding week and enter in the cumulative predicted earned value for that week. Repeat for each week until the cumulative predicted earned value reaches 100. |

|  |  |
| --- | --- |
| PSP2.1 Process Script | |
| Purpose | To guide the development of module-level programs |
| Entry Criteria | * Problem description * PSP2.1 Project Plan Summary form * Size Estimating template * Historical size and time data (estimated and actual) * Time and Defect Recording logs * Defect Type, Coding, and Size Measurement standards * Stopwatch (optional) |

|  |  |  |
| --- | --- | --- |
| Step | Activities | Description |
| 1 | Planning | * Produce or obtain a requirements statement. * Use the PROBE method to estimate the added and modified size ***and the size prediction* interval** of this program. * Complete the Size Estimating template. * Use the PROBE method to estimate the required development time ***and the time prediction interval***. * Complete a Task Planning template. * Complete a Schedule Planning template. * Enter the plan data in the Project Plan Summary form. * Complete the Time Recording log. |
| 2 | Development | * Design the program. * ***Document the design in the design templates.*** * Review the design, and fix and log all defects found. * Implement the design. * Review the code, and fix and log all defects found. * Compile the program, and fix and log all defects found. * Test the program, and fix and log all defects found. * Complete the Time Recording log. |
| 3 | Postmortem | Complete the Project Plan Summary form with actual time, defect, and size data. |

|  |  |
| --- | --- |
| Exit Criteria | * A thoroughly tested program * Completed Project Plan Summary form with estimated and actual data * Completed Size Estimating and Task and Schedule Planning templates * ***Completed Design templates*** * Completed Design Review and Code Review checklists * Completed Test Report template * Completed PIP forms * Completed Time and Defect Recording logs |

**Operational Specification Template Instructions**

|  |  |
| --- | --- |
| Purpose | * To hold descriptions of the likely operational scenarios followed during program use * To ensure that all significant usage issues are considered during program design * To specify test scenarios |
| General | * Use this template for complete programs, subsystems, or systems. * Group multiple small scenarios on a single template, as long as they are clearly distinguished and have related objectives. * List the major scenarios and reference other exception, error, or special cases under comments. * Use this template to document the operational specifications during planning, design, test development, implementation, and test. * After implementation and testing, update the template to reflect the actual implemented product. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Scenario Number | Where several scenarios are involved, reference numbers are needed. |
| User Objective | List the users’ likely purpose for the scenario, for example, to log onto the system or to handle an error condition. |
| Scenario Objective | List the designer’s purpose for the scenario, for example, to define common user errors or to detail a test scenario. |
| Source | * Enter the source of the scenario action. * Example sources could be user, program, and system. |
| Step | Provide sequence numbers for the scenario steps. These facilitate reviews and inspections. |
| Action | Describe the action taken, such as   * Enter incorrect mode selection. * Provide error message. |
| Comments | List significant information relating to the action, such as   * User enters an incorrect value. * An error is possible with this action. |

Functional Specification Template Instructions

|  |  |
| --- | --- |
| Purpose | * To hold a part’s functional specifications * To describe classes, program modules, or entire programs |
| General | * Use this template for complete programs, subsystems, or systems. * Use this template to document the functional specifications during planning, design, test development, implementation, and test. * After implementation and testing, update the template to reflect the actual implemented product. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Class Name | * Enter the part or class name and the classes from which it directly inherits. * List the class names starting with the most immediate. * Where practical, list the full inheritance hierarchy. |
| Attributes | * Provide the declaration and description for each global or externally visible variable or parameter with any constraints. * List pertinent relationships of this part with other parts together with the multiplicity and constraints. |
| Items | * Provide the declaration and description for each item. * Precisely describe the conditions that govern each item’s return values. * Describe any initialization or other key item responsibilities. |
| Example Items | An item could be a class method, procedure, function, or database query, for example. |

|  |  |  |
| --- | --- | --- |
| State Specification Template Instructions | | |
| Purpose | * To hold the state and state transition specifications for a system, class, or program * To support state-machine analysis during design, design reviews, and design inspections |
| General | * This form shows each system, program, or routine state, the attributes of that state, and the transition conditions among the states. * Use this template to document the state specifications during planning, design, test development, implementation, and test. * After implementation and testing, update the template to reflect the actual implemented product. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| State Name | * Name all of the program’s states. * Also enter each state name in the header space at the top of each “States/Next States” section of the template. |
| State Name Description | * Describe each state and any parameter values that characterize it. * For example, if a state is described by SetSize=10 and SetPosition=3, list SetSize=10 and SetPosition=3. |
| Function/Parameter | * List the principal functions and parameters. * Include all key variables or methods used to define state transitions or actions. |
| Function/Parameter Description | * For each function, provide its declaration, parameters, and returns. * For each parameter, define its type and significant values. |
| Next State | * For each state, list the names of all possible next states. * Include the state itself. |
| Transition Condition | List the conditions for transition to each next state.   * Use a mathematical or otherwise precise notation. * If the transition is impossible, list "impossible," with a note saying why. |
| Action | List the actions taken with each state transition. |

Logic Specification Template Instructions

|  |  |
| --- | --- |
| Purpose | * To contain the pseudocode for a program, component, or system * To enable precise and complete program implementation * To facilitate thorough design and implementation reviews and inspections |
| General | * Use this template to document the program’s detailed logic. * After implementation and testing, update the template to reflect the actual implemented product. * During detailed design, write the pseudocode needed to describe all of the program’s logic. * Use plain language and avoid using programming instructions wherever practical. |
| Header | * Enter your name and the date. * Enter the program name and number. * Enter the instructor’s name and the programming language you are using. |
| Design References | List the references used to produce the program’s logical design.   * the Operational, Functional, and State templates * the program’s requirements * any other pertinent source |
| Parameters | * Where needed, define any parameters or abbreviations used. * Avoid duplicating definitions on other templates and reference these other definitions where they are needed. |