Assignment Kit for Program 6

Personal Software Process (PSP) for Engineers: Part II

Personal Software Process for Engineers: Part II

Assignment Kit for Program 6

Overview

Overview This assignment kit covers the following topics.

Prerequisites Reading

- Chapters 10 and 11

Program 6 requirements

Program 6 requirements

Using PSP2.1, write a program to find the value of *x* for which integrating the t function from 0 to *x* gives a result of *p*.

Thoroughly test the program. At a minimum, calculate the values for the t-distribution integral for the values in Table 1. Expected values are also included in Table 1.

Test		**Expected Value**	**Actual Value**
p	***dof***	***x***	
0.20	6	0.55338	
0.45	15	1.75305	
0.495	4	4.60409	

Table 1

Search algorithm

Finding the value of *x*

Find the value of x for which integrating the t function from 0 to x gives a result of p.

- Start with a trial value for upper limit of 1 and calculate the value of the integration.
- Compare it to the desired value.
  - if the result of the integration is too low, pick a larger trial upper limit
  - if the result of the integration is too high, pick a smaller trial upper limit

Make successive trial integrations until the value of the integration is within an acceptable error, say 0.00001.

One way to make this calculation is as follows.

Step	*Action*
1	Start with a trial value of x (for example, 1.0).
2.	Make an initial integral and test to see if it gives the proper value; if not, continue.
3.	If it is too low, add $d = 0.5$ to trial x.
4.	If it is too high, subtract $d = 0.5$ from trial x.
5.	Integrate again and test if the result is within an acceptable error; if not, continue.
6.	If too low, adjust d; add d to trial x.
7.	If too high, adjust d; subtract d from trial x.
8.	Recycle at 5.

The rules for adjusting d are these.

1. As long as the tests for the error of the result give the same sign of the error, leave d unchanged.
2. Whenever the sign of the error changes, divide d by 2.

ote that this method of adjusting d could result in a trial value of $x = 0$.

o guard against a problem with Simpson's method, ensure that the program will handle a 0 value of the function being integrated.

Assignment instructions

Assignment instructions

Before starting program 6, review the top-level PSP2.1 process script below to ensure that you understand the “big picture” before you begin. Also, ensure that you have all of the required inputs before you begin the planning phase.

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 Counting 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

Continued on next page

Planning phase

Plan program 6 following the PSP2.1 planning phase and the PROBE estimating scripts.

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. ((Note: This step is completed by the SEI student workbook.)***
3	Resource Estimate	- Use the PROBE method to estimate the time required to develop this program. - ***Calculate the 70% size prediction interval. ((Note: This step is completed by the SEI student workbook.)*** - Using the To Date % from the most recently developed program as a guide, distribute the development time over the planned project phases. (Note: This step is completed by the SEI student workbook.)
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

Verify that you have met all of the exit criteria for the planning phase, **then have an instructor review your plan**. After your plan has been reviewed, proceed to the development phase.

Continued on next page

Use the PROBE method to create size and resource estimates.

ROBE 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 size 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 the β_0 and β_1 parameters from the estimated proxy size and actual added and modified size data. - If the absolute value of β_0 is not near 0 (less than about 25% of the expected size of the new program), or β_1 is 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 the β_0 and β_1 parameters from the plan added and modified size and actual added and modified size data. - If the absolute value of β_0 is not near 0 (less than about 25% of the expected size of the new program), or β_1 is 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 β_0 = 0 and β_1 = (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)

Continued on next page

Assignment instructions, Continued

PROBE Estimating 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 the β_0 and β_1 parameters from the estimated proxy size and actual total development time data. - If β_0 is not near 0 (substantially smaller than the expected development time for the new program), or β_1 is not within 50% of 1/(historical productivity), use procedure 5B.
5B	Time Estimating Procedure 5B	- Using the linear-regression method, calculate the β_0 and β_1 regression parameters from the plan added and modified size and actual total development time data. - If β_0 is not near 0 (substantially smaller than the expected development time for the new program), or β_1 is 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 β_0 = 0 and β_1 = (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 β_0 = 0 and β_1 = (actual total development time to date/plan total added and modified size to date). - If you only have actual time and size data, set β_0 = 0 and β_1 = (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

Continued on next page

Development phase

Develop the program following the PSP2.1 development phase script.

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

Verify that you have met all of the exit criteria for the development phase, then proceed to the postmortem phase.

Continued on next page

Design rreview

Review your designs following the PSP2.1 design review script.

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

Continued on next page

Code rreview

Review your code following the code review script.

Code Review Script

Purpose	To guide you in reviewing programs
Entry Criteria	- A completed and reviewed program design - Source program listing - Code Review checklist - Coding standard - Defect Type standard - Time and Defect Recording logs
General	Do the code review with a source-code listing; do not review on the screen!

Step	Activities	Description
1	Review	- Follow the Code 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 it is completed. - For multiple procedures or programs, complete a separate checklist for each.
2	Correct	- Correct all defects. - If the correction cannot be completed, abort the review and return to the prior process phase. - To facilitate defect analysis, record all of the data specified in the Defect Recording log instructions for every defect.
3	Check	- Check each defect fix for correctness. - Re-review all design changes. - Record any fix defects as new defects and, where you know the number of the defect with the incorrect fix, enter it in the fix defect space.

Exit Criteria	- A fully reviewed source program - One or more Code Review checklists for every program reviewed - All identified defects fixed - Completed Time and Defect Recording logs

Continued on next page

Postmortem phase

Conduct the postmortem following the PSP2.1 postmortem script.

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 counting 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, deleted, modified, base additions, reused, new reusable code, and added parts. - Enter these data in the Size Estimating template. - Determine the total program size - Enter this 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 counting 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

Verify that you have met all of the exit criteria for the PSP2.1 postmortem phase, then submit your assignment.

Continued on next page

Submitting your assignment

When you've completed the postmortem phase, submit your assignment package, source code, and test results to the instructor.

The order for the assignment package is
- PSP2.1 Project Plan Summary form
- Test Report template
- PSP2.1 Design Review checklist
- Code Review checklist
- PIP form
- Size Estimating template
- PROBE Calculation worksheet
- Operational Specification template
- Functional Specification template
- State Specification template
- Logic Specification template
- Time Recording log
- Defect Recording log
- source program listing
- test results

Guidelines and evaluation criteria for program 6

Evaluation criteria	Your process report must be • complete • legible • in the specified order	Your process data must be • accurate • precise • self-consistent

Suggestions

Remember, you should complete this assignment today.

Keep your programs simple. You will learn as much from developing small programs as from large ones.

If you are not sure about something, ask your instructor for clarification.

Software is not a solo business, so you do not have to work alone.

- You must, however, produce your own estimates, designs, code, and completed forms and reports.
- You may have others review your work, and you may change it as a result.
- You should note any help you receive from others in your process report. Log the review time that you and your associates spend, and log the defects found or any changes made.

Operational Specification Template

Student	____________________	**Date**	__________
Program	____________________	**Program #**	__________
Instructor	____________________	**Language**	__________

Scenario Number		**User Objective**	
Scenario Objective			
Source	**Step**	**Action**	**Comments**

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

Student		**Date**	
Program		**Program #**	
Instructor		**Language**	

Class Name	
Parent Class	

Attributes

Declaration	**Description**

Items

Declaration	**Description**

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

Student ______________________ Date ____________
Program ______________________ Program # ____________
Instructor ______________________ Language ____________

State Name	Description

Function/Parameter	Description

States/Next States	Transition Condition	Action

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

Student		Date	
Program		Program #	
Instructor		Language	

Design References

Parameters

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.

