Tutorial: Using PSP0

Personal Software Process[SM] for Engineers: Part I

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Tutorial Objectives

After this tutorial, you will

- understand the PSP0 process
- know how to use PSP0 process scripts and forms
- be prepared to use PSP0 for program 1

Before You Begin

The PSP is not *the* process for writing software.

The PSP is a process for learning about process.

It's the process equivalent of a code sample.

It works so well for some students that they can use it on the job, but that's not its intended purpose.

After you've completed the course you should

- examine your PSP data
- review your experience and PIPs
- tailor the PSP to meet your needs.

PSP0 Process

PSP0 is a simple, defined, personal process.

- Make a plan.
- Use your current design and development methods to produce a small program.
- Gather time and defect data on your work.
- Prepare a summary report.

PSP0 Objective

The objective for PSP0 is to

- demonstrate the use of a defined process in writing small programs
- incorporate basic measurements in the software development process
- require minimal changes to your personal practices

PSP0 Process Phases -1

PSP0 has six phases.

Planning – produces a plan for developing the program defined by the requirements.

Design – produces a design specification for the program defined by the requirements.

Coding – transforms the design specification into programming language statements.

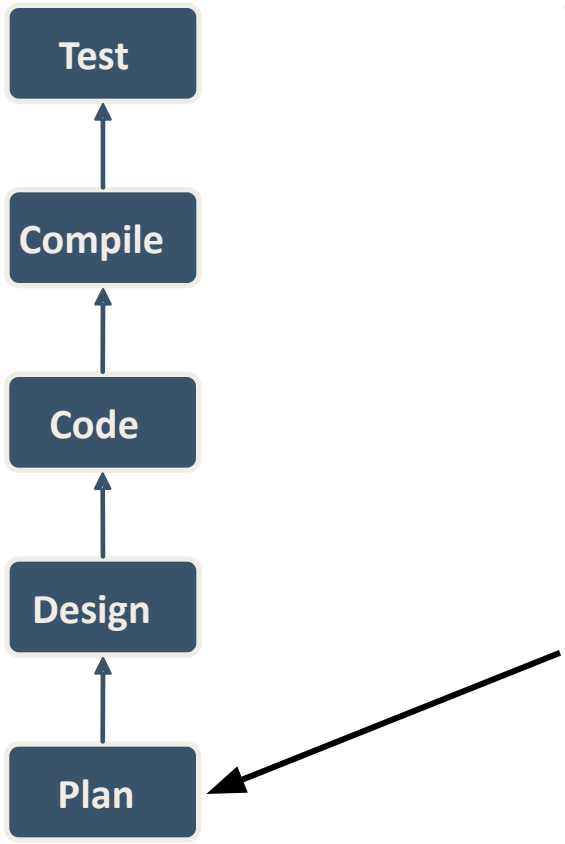
PSP0 Process Phases -2

Compile – translates the programming language statements into executable code.

Test – verifies that the executable code satisfies the requirements.

Postmortem – summarizes and analyzes the project data.

Phase Order



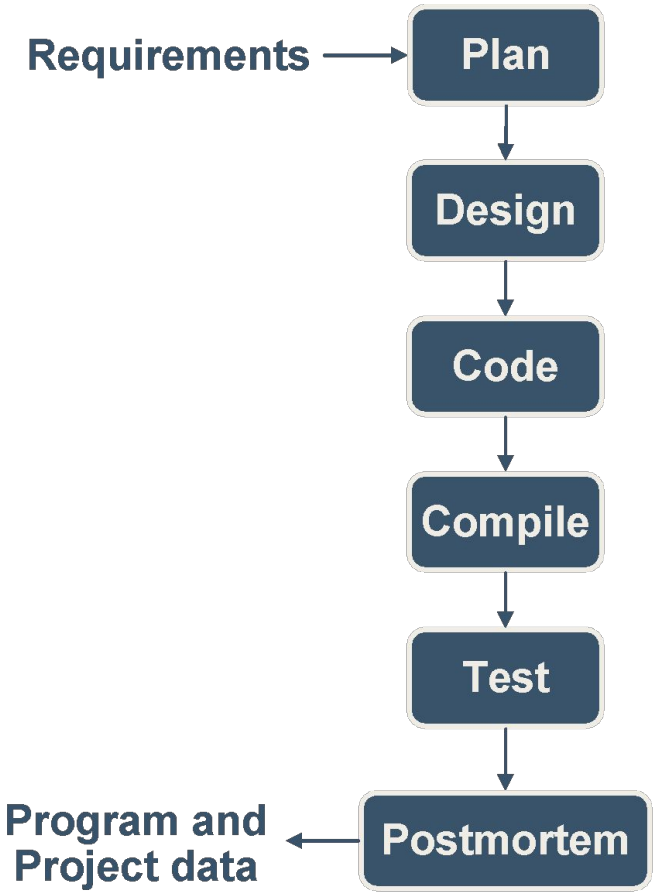
The PSP looks like a waterfall process but it's not.

The phase order is determined by the dependencies between phases.

- You can't test the code before it's compiled.
- You can't compile the code before it's written.
- You can't use the design if it's produced after the code is written.
- There's no reason to make a plan after you're done.

Conclusion…start here with a plan.

Process Flow



When programs are small or well understood, you can execute the phases in order.

Produce a plan.

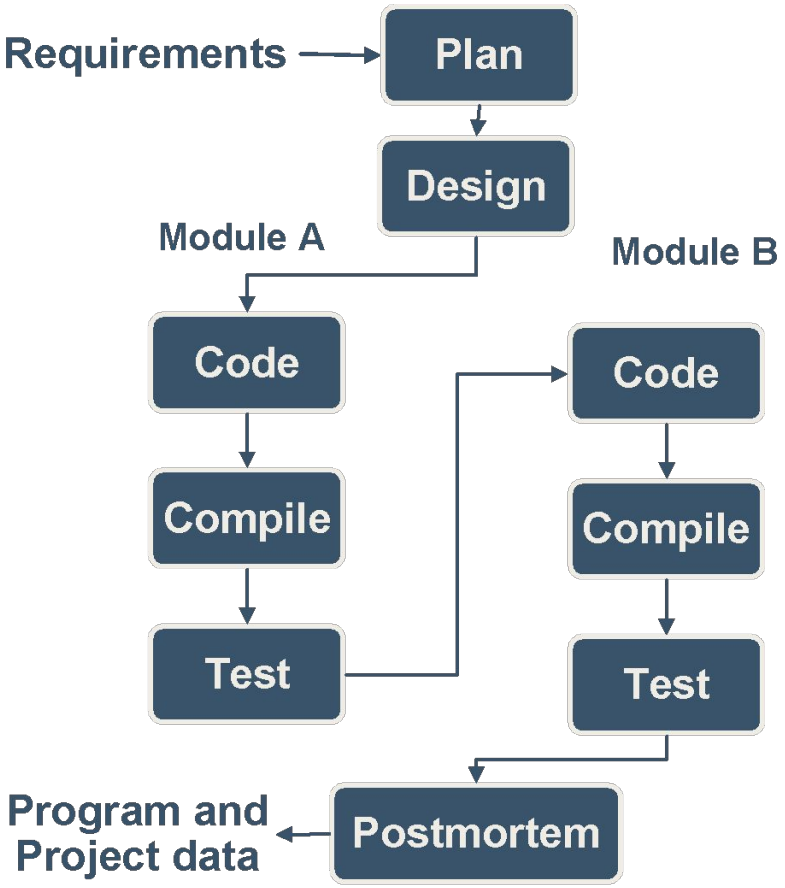
Design all modules.

Code all modules.

Compile the coded program.

Summarize the project data during the postmortem.

Cyclic Process Flow -1



Large programs or those that are not well understood may require an iterative approach.

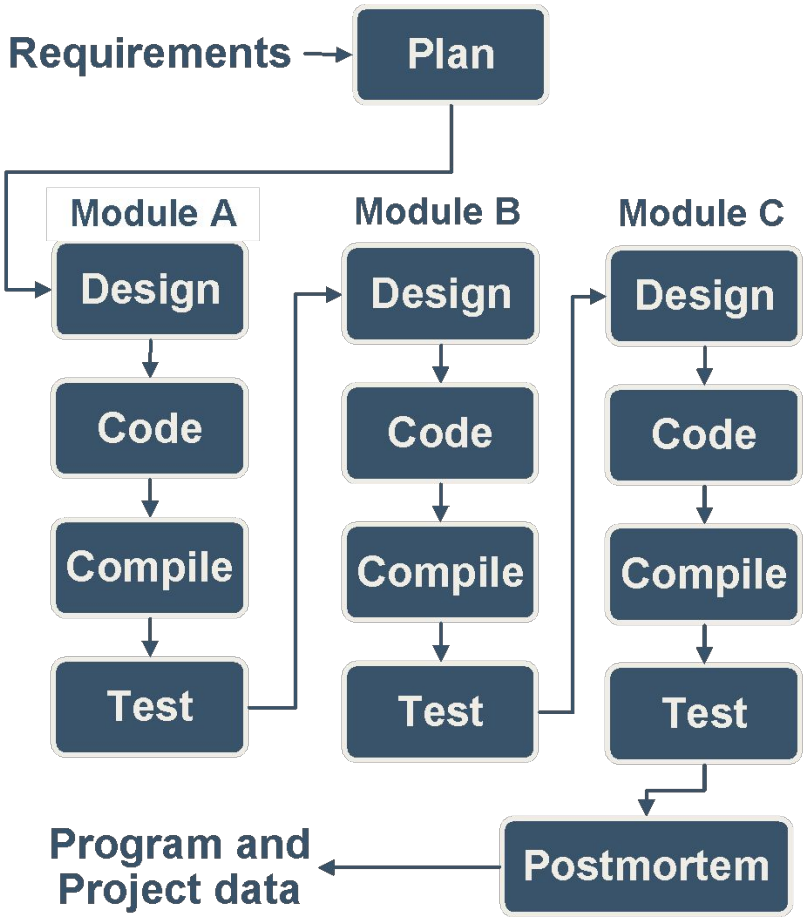
In this example the design is completed in one step.

Two modules are identified during the design, modules A and B.

Then each module is separately coded, compiled, and tested.

This example uses the PSP0 phases and two cycles of code-compile-test.

Cyclic Process Flow -2



There can be more than two cycles and cycles can also include the design phase as in this example.

Note that each cycle is focused on producing part of the program, e.g. Module A, Module B, Module C.

Part size is a key factor for determining cycles.

- a line of code is too small
- a program may be too large

One or more classes, methods, procedures, or functions are probably the right size.

You need to determine what works for you.

Process Scripts

Process scripts provide “expert-level” guidance on how to use the process.

They are one or two pages long.

They describe the

- Purpose
- Entry criteria
- General guidelines
- Steps
- Exit criteria

PSP0 Process Script	
Purpose	To guide the development of module-level programs
Entry Criteria	- Problem description - PSP0 Project Plan Summary form - Time and Defect Recording logs - Defect Type standard - Stopwatch (optional)

Step	Activities	Description
1	Planning	- Produce or obtain a requirements statement. - Estimate the required development time. - Enter the plan data in the Project Plan Summary form. - Complete the Time Recording log.
2	Development	- Design the program - Implement the design. - 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 Time and Defect Recording logs

The PSP0 Scripts -1

Planning: Estimate the development time.

Development: Develop the product using your current methods.

Postmortem: Complete the project plan summary with the time spent and defects found and injected in each phase.

The PSP0 Scripts -2

Design: Design the program using your current design methods.

Coding: Implement the program.

Compile: Compile until defect-free.

Test: Test the program and fix all defects.

Record defects in the defect log and time per phase in the time log.

Using Process Scripts

Process scripts guide you through the process.

You should

- check the entry criteria before starting a phase
- record the phase start time
- perform the phase steps and instructions
- record defects as they are found and corrected
- check the exit criteria before ending a phase
- record the phase end time
- go to the next phase

Force yourself to use this paradigm until it becomes a habit.

PSP0 Measures and Forms

PSP0 measures

- Time – track time in phase
- Defects – record defects as they are found and fixed

PSP0 has four forms

- PSP0 Project Plan Summary – summarizes planned and actual time and defects by phase
- PSP0 Time Recording Log – used to record time
- PSP0 Defect Recording Log – used to record defects
- PSP0 Defect Type Standard – used to define standard defect types

Some Common Errors

The following are common errors students make.

- Not tracking time in planning and postmortem
- Skipping phases or executing phases out of order
- Omitting some defects found in compile or test
- Forgetting to record the measured actual size of programs developed (PSP1.x and above)
- Confusing phases with activity, e.g. counting time spent re-designing or re-coding during test as design time or coding time.

The next few slides address some specific problems with time and defect tracking.

Defect Fix Time

Defect fix time is often misunderstood.

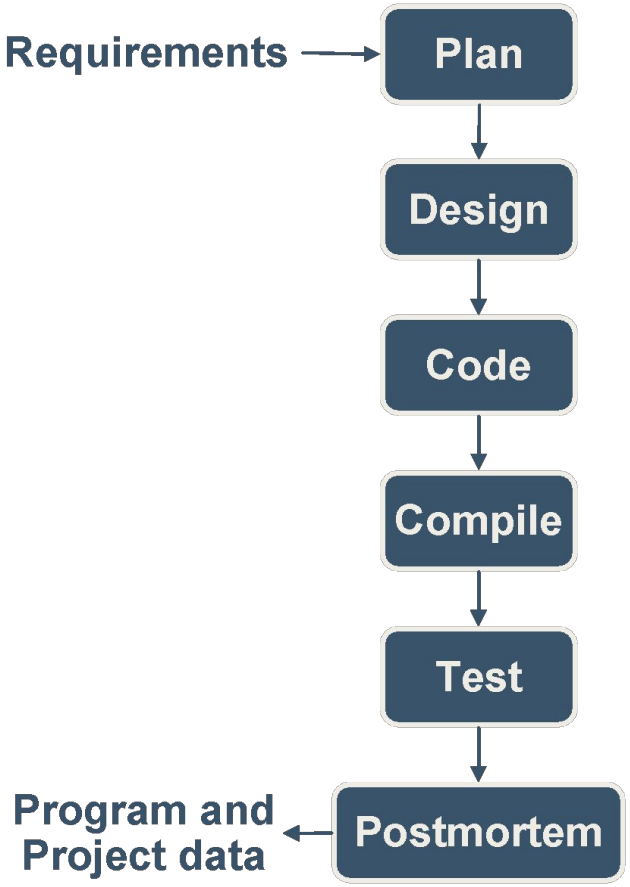
It is the time taken both to find and fix the defect.

Example

8:05 run compiler on p1a.c, “line 23 - type mismatch”
8:06 run editor on p1a.c
8:15 change declaration on line 6 from integer to real
8:16 run compiler on p1a.c, “no errors”

Q: What is the defect fix time?
A: 10 minutes

Defect Phase Injected



The injected phase for a defect depends on the phase the program is in.

Example:

Tom finds a major logic error in his program during test. He has to redesign and code part of his program.

Q: What phase is Tom's program in?

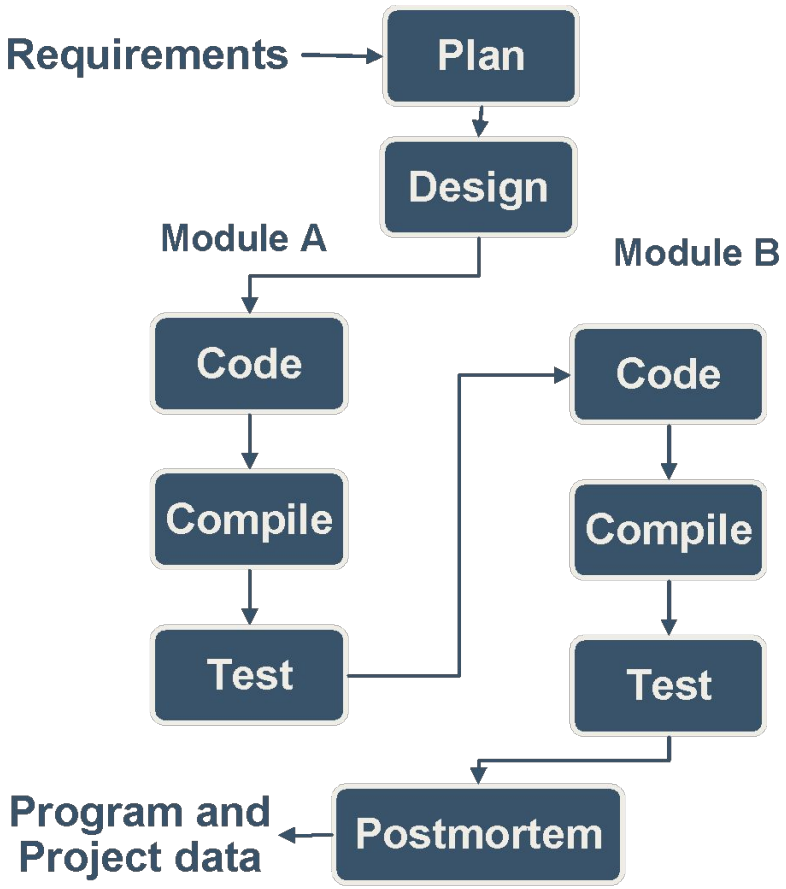
A: Test

Tom finds a defect in the new code he has written.

Q: In what PSP phase was the defect injected?

A: Test

Measurement in the Cyclic Process



Considerations

- Include a program part identifier in the notes field on time log entries.
- Add a similar annotation to defect log entries.
- Use “Test” as the phase removed when defects are found in a previously tested part.

Example

Tom finds and fixes an interface error in part A of his program while coding part B.

Q: In what PSP phase was this defect removed?

A: Test

Measurement Hints

Gather and record data on your process as you work, not afterwards. If you forget, promptly make your best guess.

Be precise and accurate.

- Track time in minutes.
- Count every defect.

You will be using your own data to manage your process; gather data that is worthy of your trust.

Messages to Remember

In using PSP0, your principal objective is to learn to gather and report accurate and complete data on your work.

Once you have completed this course, you will know how to adjust and extend the PSP to meet your future needs.

Until then, make your best effort to follow the PSP process scripts and instructions.