CSSE3012 The Software Process

Week 13
Personal Software Process

School of Information Technology
and Electrical Engineering

The University Of Queensland

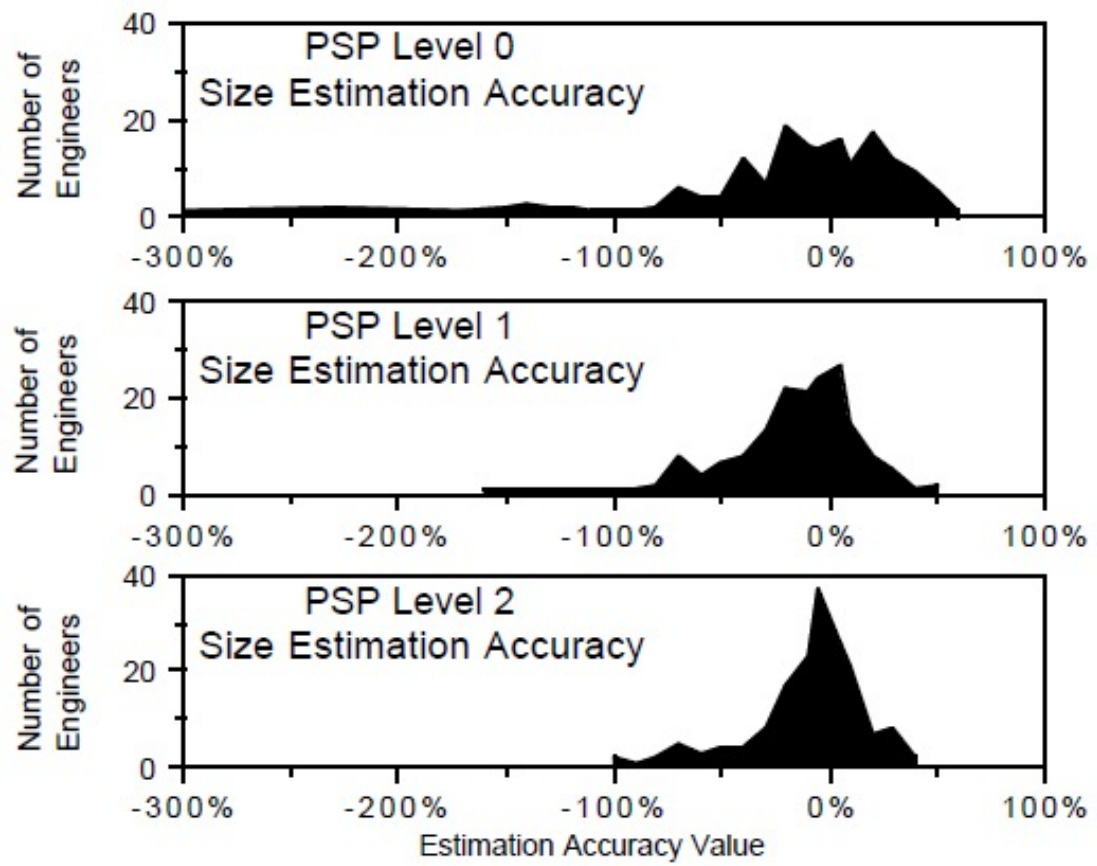
Personal Software Process™

- PSP is based on process improvement principles
- Practitioners establish personal process goals
  - define the methods they will use
  - measure their work
  - analyse the results
- Based on these analyses, they adjust their methods to better meet their personal goals
  - improve their personal capability

PSP Properties

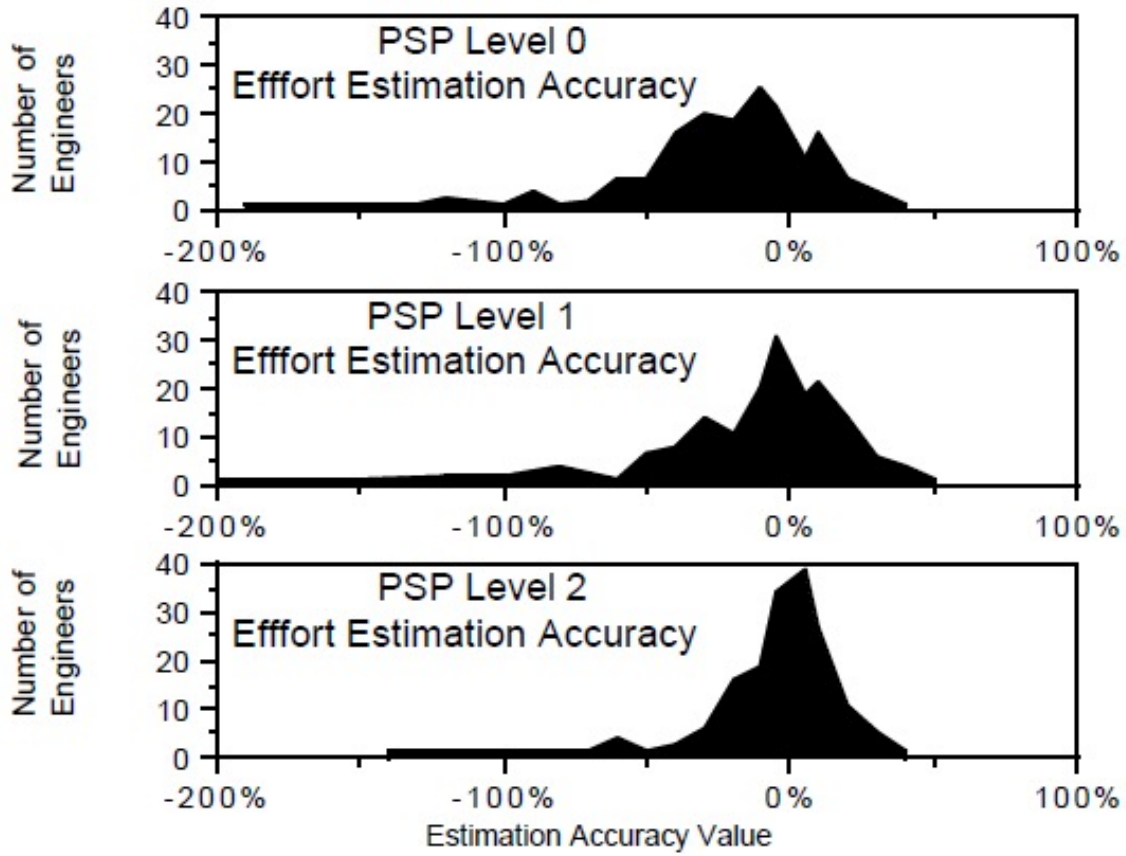
- Process for individual engineers
- CMM level 5 compliant
- Can be applied to most structured personal tasks
  - writing small programs or documents
  - defining requirements or processes
  - conducting reviews or tests

Size Estimation Results



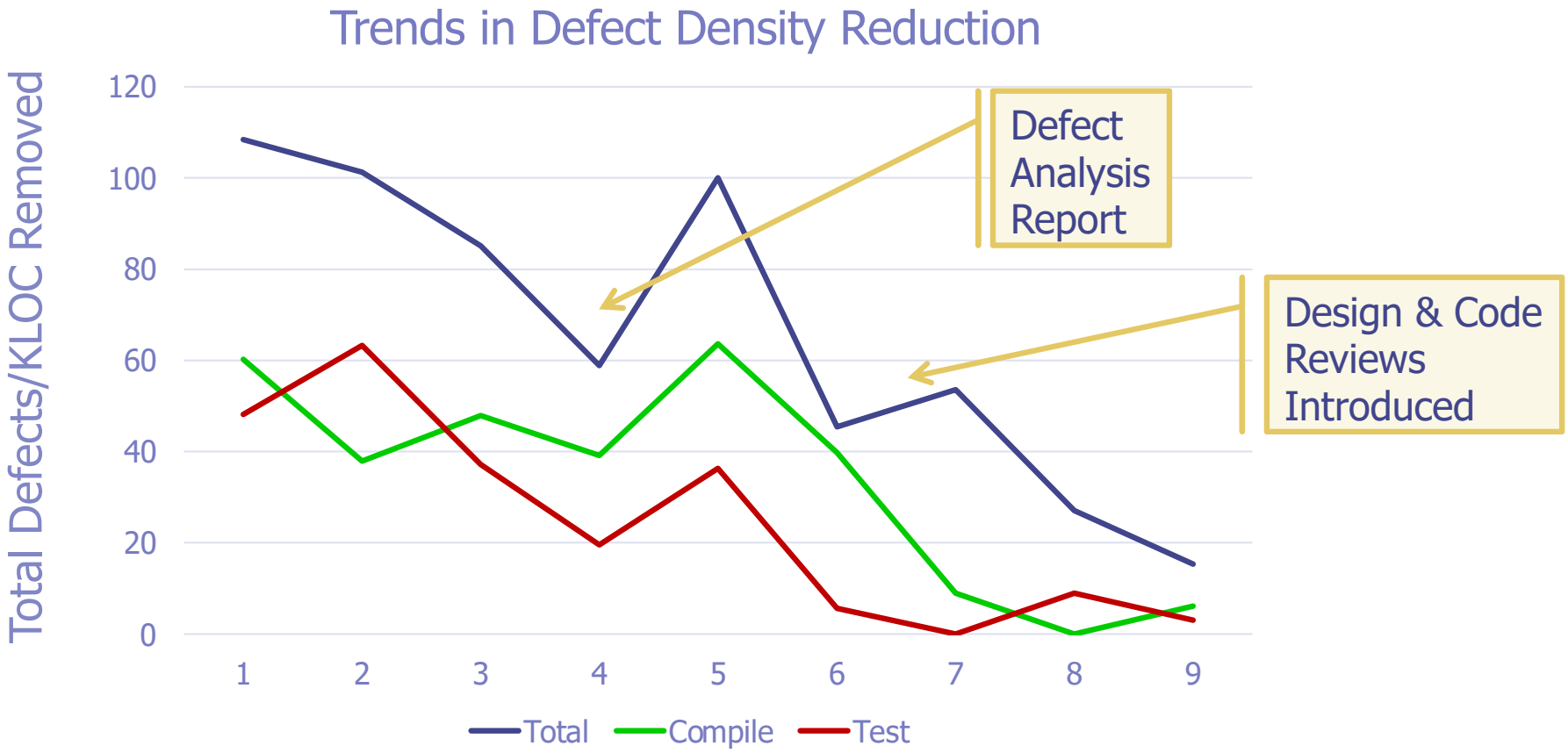
1997 SEI Study

Effort Estimation Results



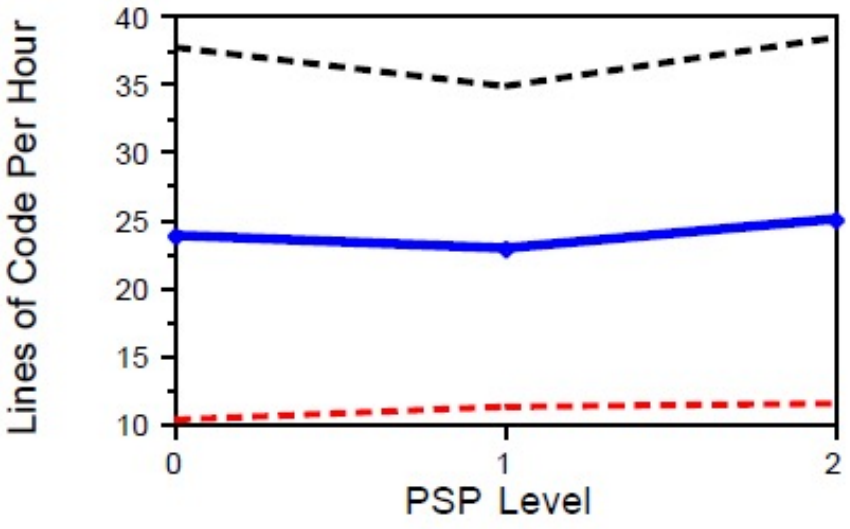
1997 SEI Study

Product Quality Results



1997 SEI Study

Productivity Results

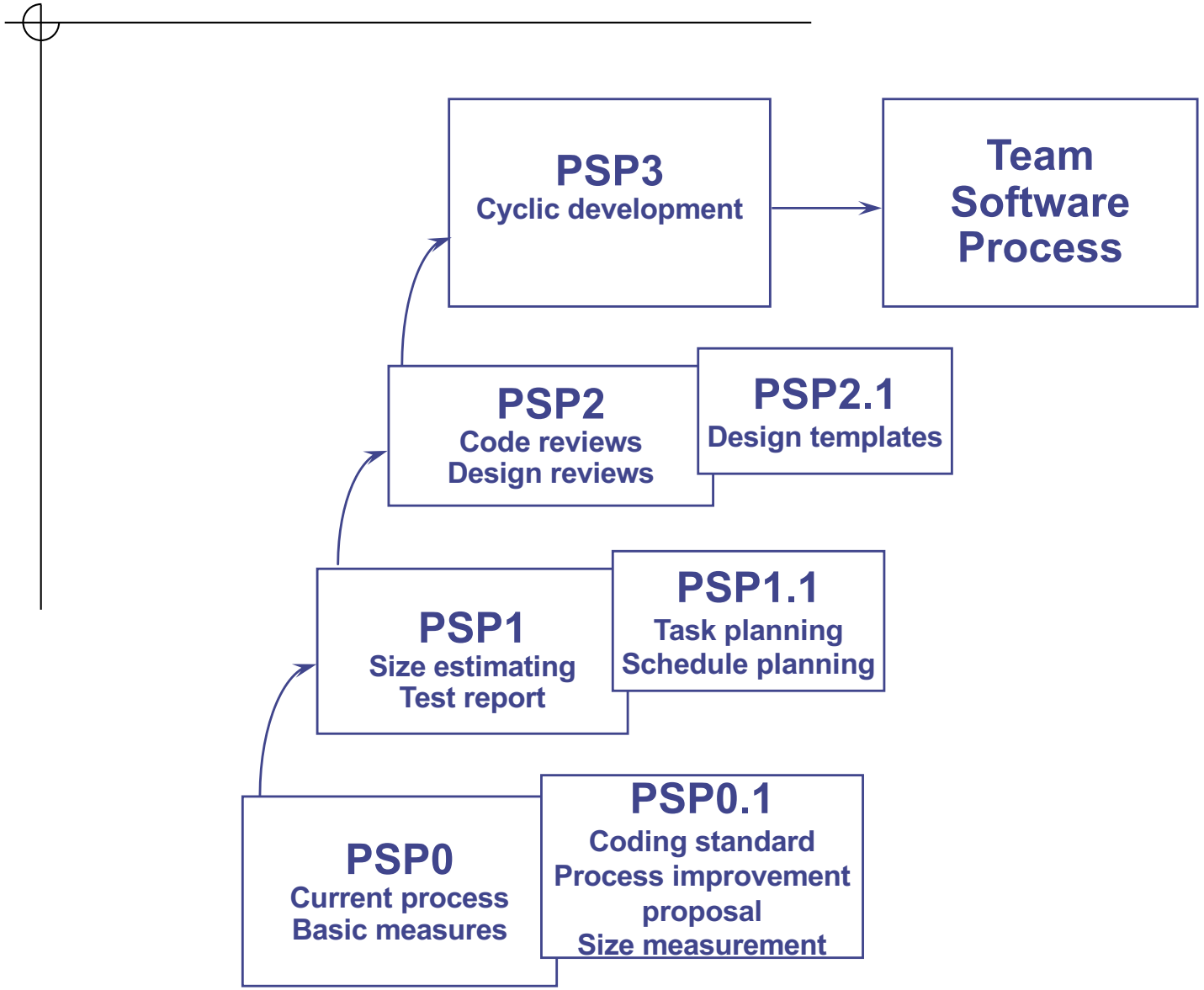


1997 SEI Study

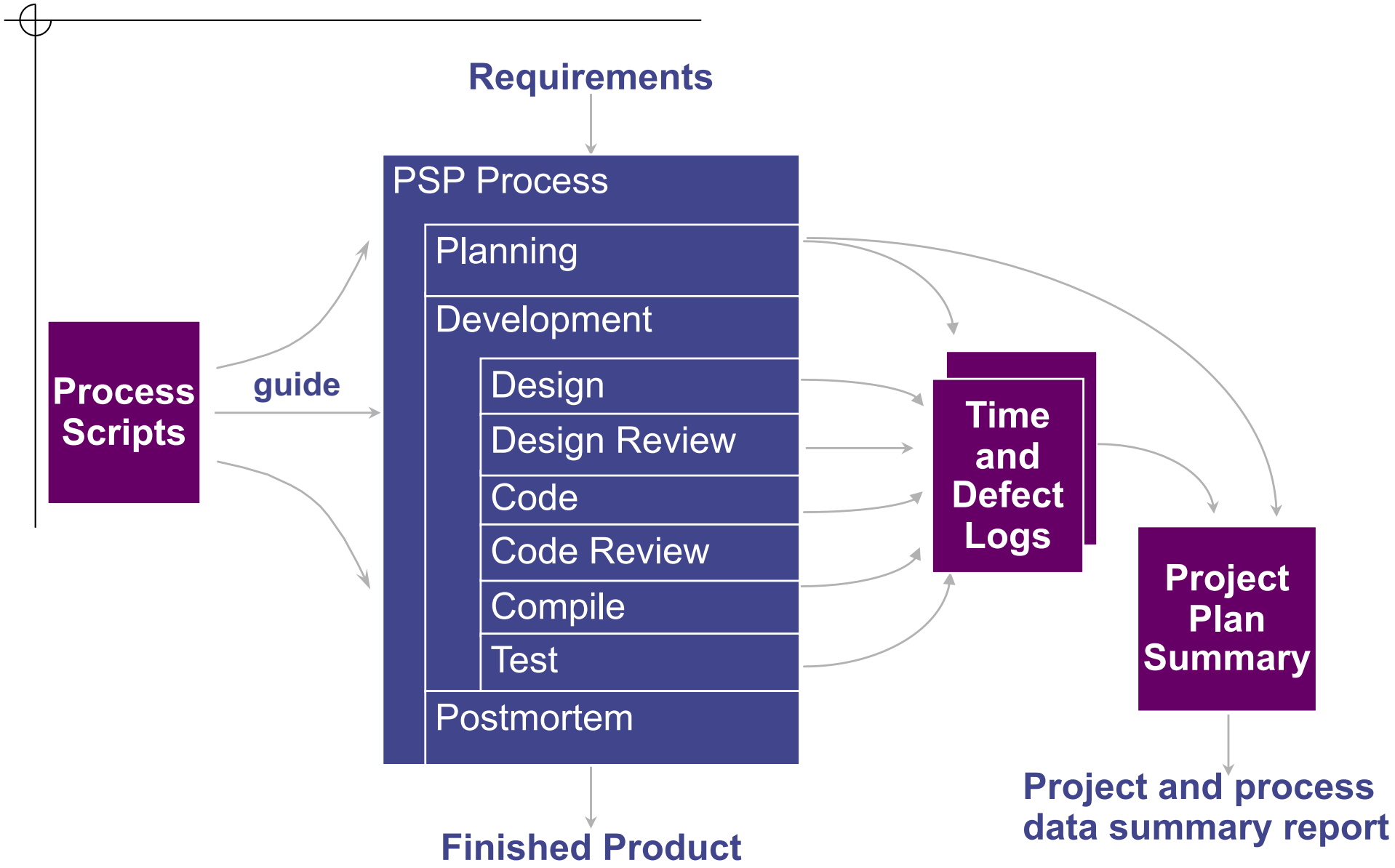
PSP: Learning by Experience

- Start with engineer's current process
- Gradually introduce new methods
- Practice these methods on module-sized programs
- Engineers see for themselves how these methods help

PSP Process Stages



PSP Process Flow



PSP0: Personal Measurement

- 3 phases
  - planning
  - development (design, code, compile, test)
  - post-mortem
- Gather data by phase
  - time spent
  - defects found
- Generates **real** personal data
- Provides base benchmark for measurement
- **PSP0.1**: coding standard, size measurement, process improvement proposal (PIP)

Basic Measures

- **Development time** (minutes, time log)
  - designed to account for *interruptions*
- **Defects** (count, defect log)
  - any change to the design or code to get the program to compile or test correctly
- **Size** (lines of code, project summary)
  - for estimating development time
  - new, modified and reused code is distinguished

Basic PSP Elements

- Process scripts
- Project plan summary form
- Time recording log
- Defect reporting log
- Defect type standard

Phase Number	Purpose:	To guide you in developing module-level programs.
	Inputs Required	• Problem description • PSP Project Plan Summary form • Historical estimated and actual size and time data • Time and Defect Recording Logs • Defect Type Standard • Stop watch (optional)
1	Planning	• Produce or obtain a requirements statement. • Estimate the total new and changed LOC required and the prediction interval. • Estimate the required development time and the prediction interval. • Produce a schedule plan (if more than a couple days). • Enter the plan data in the Project Plan Summary form. • Complete the Time Recording Log.
2	Development	• Design the program, using design templates where appropriate. • 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 with estimated and actual data • Completed design templates • Completed Design Review Checklist and Code Review Checklist • Completed Test Report Template • Complete PIP forms • Completed Defect and Time Recording Logs

Project Plan Summary

- Project plan data
- Actual project results
  - size
  - time
  - defect data
- Cumulative data on all PSP projects to date

Time Recording Log

Student JD Veloper Date 7/1

Instructor Humphrey Program # 1A

Date	Start	Stop	Interruption Time	Delta Time	Phase	Comments
7/1	8:00	8:16	5	11	Plan	Estimated time

C18 Defect Recording Log

Defect Types	
10 Documentation	60 Checking
20 Syntax	70 Data
30 Build, Package	80 Function
40 Assignment	90 System
50 Interface	100 Environment

Instructor: JD Veloper
Humphrey

Date: 7/1

Program #: 1A

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
7/1	1	40	code	compile	2	

Description: Missing declaration

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
	2	20	code	compile	1	

Description: Typo in function name

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Why Measure Time Usage?

- Time is a non-renewable resource!
- Realistic planning requires knowing how you spend your time
- Tracking provides a more accurate record than relying on your memory
- To manage your time, plan your time and then follow the plan
  - *easier said than done!*

Why Measure Time Usage?

- Working to a plan helps guide your behaviour
  - less time procrastinating
  - more focus on the actual task
  - less likely to be distracted
  - more likely to be efficient
- Learn from your mistakes by planning better next time

Why Record Defects?

- Identify types of defects you introduce
- Improve your skill as a programmer
- Reduce number of defects
- Each change you make counts as one defect

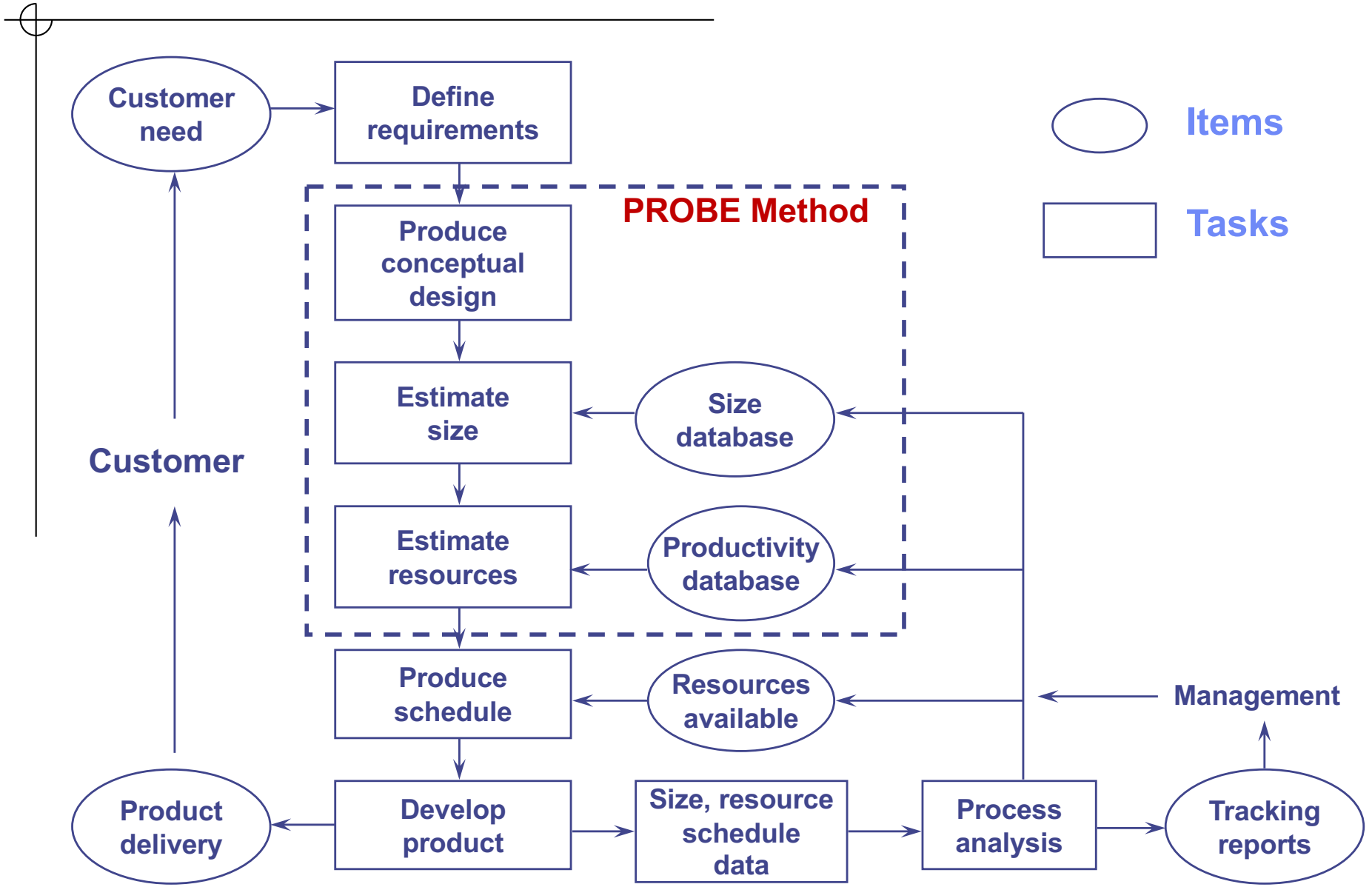
Defects

- Anything that detracts from a program's ability to completely and effectively meet the user's needs
- Caused by a programmer's mistake
- Even experienced programmers make a mistake about every 7-10 lines of code
- Defect prevention and removal are essential
  - typically account for 50% of project effort!

PSP1: Personal *Planning*

- Adds PROBE (PROxy Based Estimating) method
  - estimate sizes and development times for new programs based on personal data
  - based on linear regression with prediction intervals to indicate size and time estimate quality
- Adds schedule and task planning

Project Planning Framework



Why Estimate Size?

- Make better plans
  - more accurately size the job
  - divide the job into separable elements
- Assist in tracking progress
  - judge when job scope changes
  - more accurately measure work
- Value of PSP
  - learn estimating methods
  - build estimating skills

Resource Planning Process

- Start with a size estimate
- Identify available data
- Use regression when you have more than three sets of data that correlate
- Use data for estimated LOC to actual hours
  - where available
- Calculate the prediction interval

Schedule Estimating

- Scheduling requires three things
  - estimated direct project hours
  - calendar of available direct hours
  - order in which tasks will be done
- Then, you need to
  - estimate hours needed for each task
  - spread these hours over calendar of available time

Schedule Example

- Jo decides to plan and track the next PSP assignment
- Based on her historical data, planned time for each phase is:

Task	Planned hrs	Cumulative hrs
Plan	1.0	1.0
Design	4.5	5.5
Code	5.0	10.5
Compile	0.5	11.0
Test	1.5	12.5
TOTAL	12.5	

Schedule Example

- Jo decides to plan and track the next PSP assignment
- Based on her historical data, planned time for each phase is:

Task	Planned hrs	Planned Value	Cum. hrs	Cum. Planned Value
Plan	1.0	8%	1.0	8%
Design	4.5	36%	5.5	44%
Code	5.0	40%	10.5	84%
Compile	0.5	4%	11.0	88%
Test	1.5	12%	12.5	100%
TOTAL	12.5	100%		

Schedule Example

- Jo knows she will be able to spend 3.5 hours per day on this assignment
- Producing the following schedule:

Day No.	Direct hrs	Cum. hrs
1	3.5	3.5
2	3.5	7.0
3	3.5	10.5
4	3.5	14

Schedule Example

- Now Jo can determine the day on which each task should complete

Task	Planned hrs	Cum. hrs	Completed (day)
Plan	1.0	1.0	1
Design	4.5	5.5	2
Code	5.0	10.5	3
Compile	0.5	11.0	4
Test	1.5	12.5	4
TOTAL	12.5		

Schedule Example

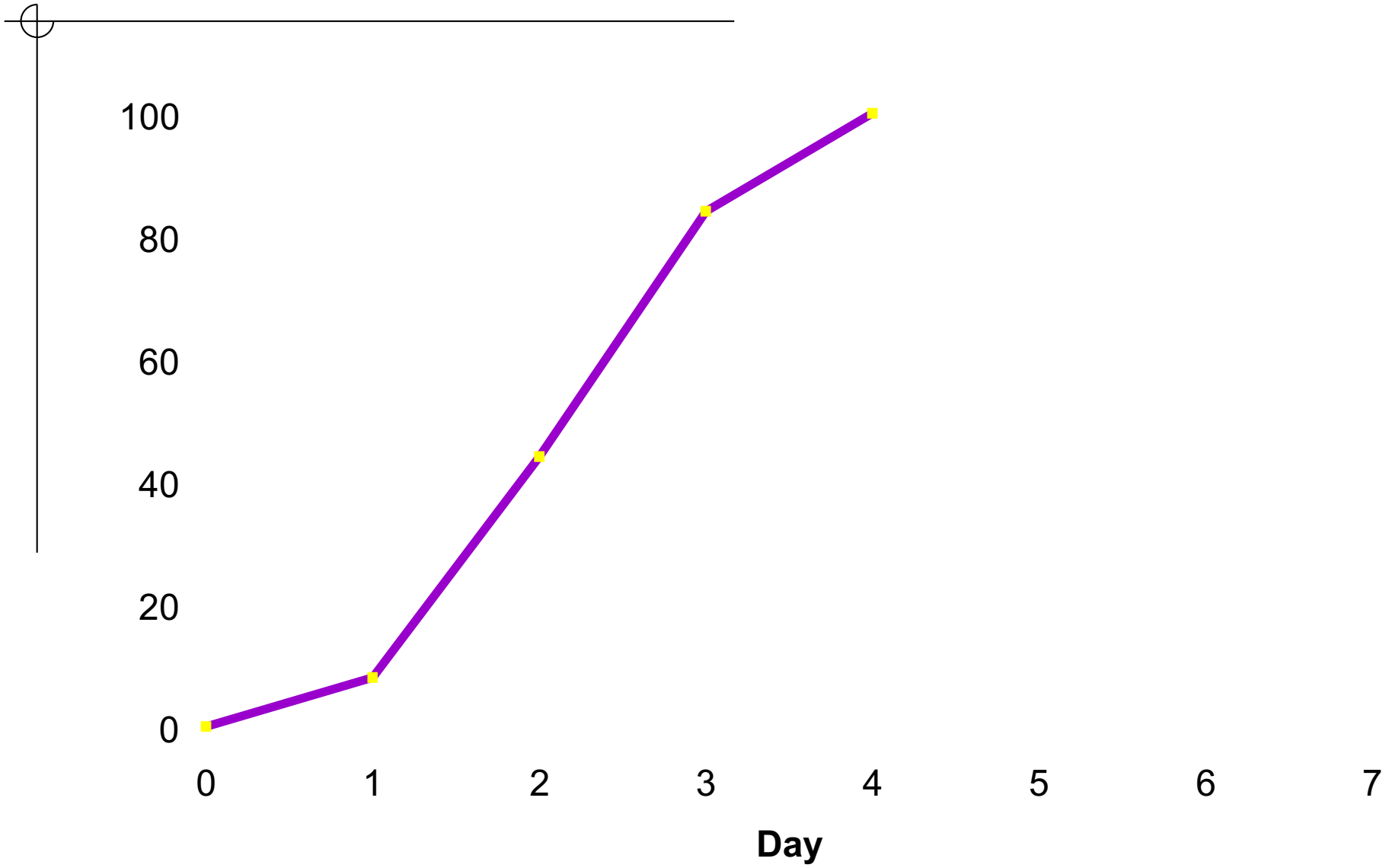
- Final step is to calculate the (cumulative) planned value for each day (based on completed tasks)

Day No.	Daily hrs	Cum. hrs	Cum. Planned Value
1	3.5	3.5	8%
2	3.5	7.0	44%
3	3.5	10.5	84%
4	3.5	14	100%

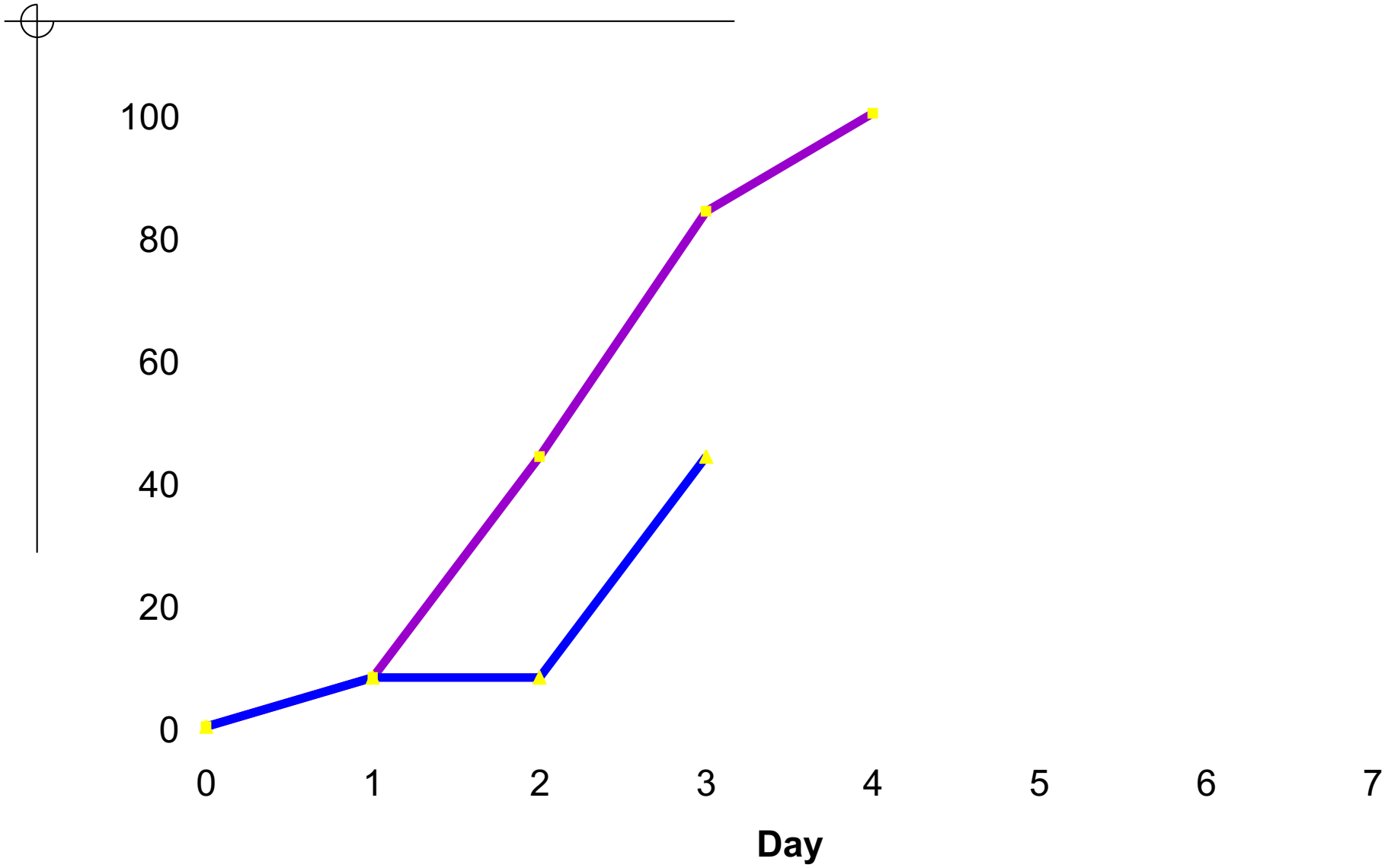
Schedule Example

- Schedule can be represented as a chart
  - showing cumulative planned value per day
- As tasks get completed, we track earned value
- Earned value of a task is its original planned value
  - **independent** of actual time taken to complete it
- Use extrapolation to predict completion of project

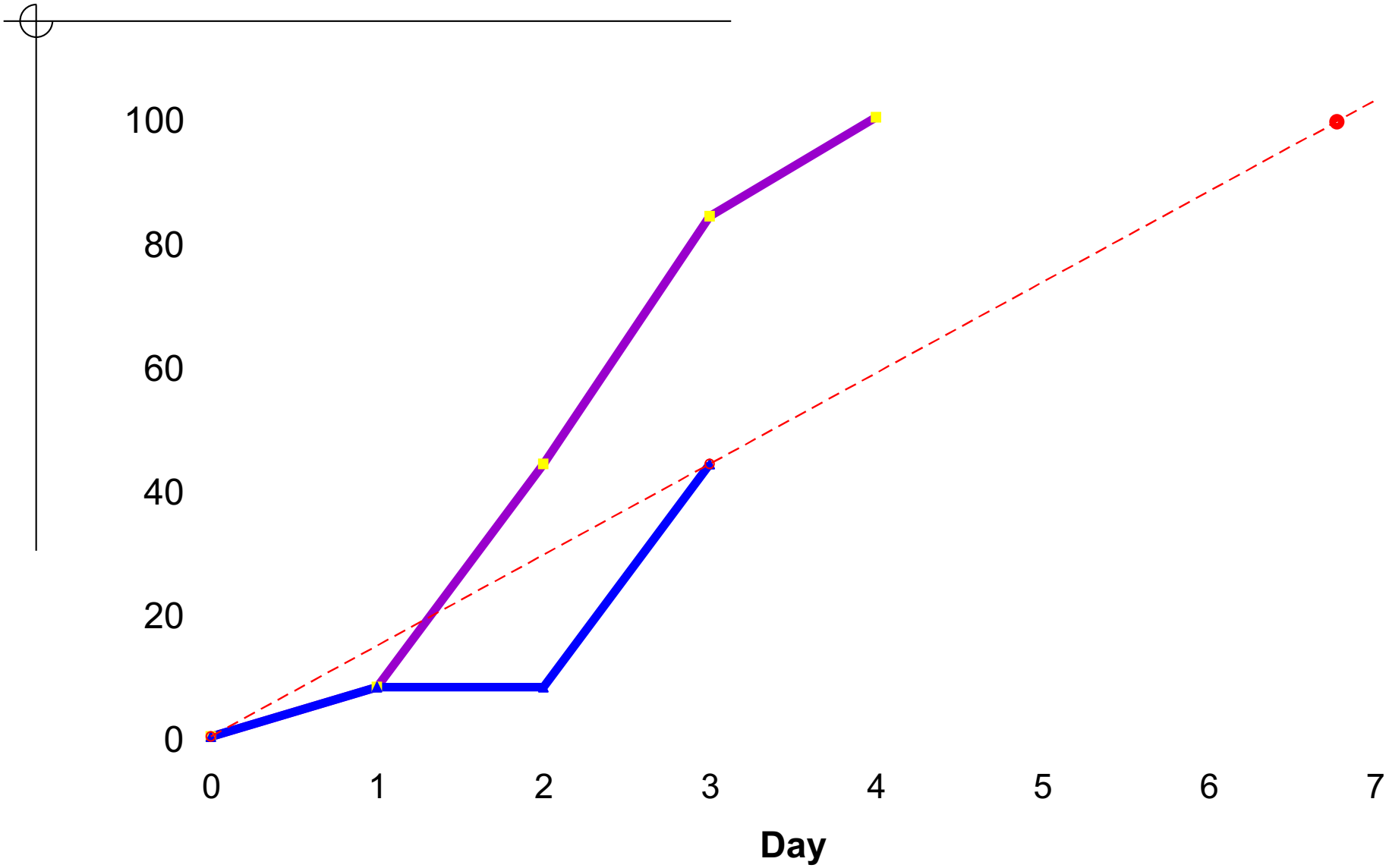
Planned Value



Earned Value after 3 days



Earned Value Prediction



PSP2: Personal Quality

- Introduces defect management
  - analyse data from PSP exercises
  - construct checklists for design and code reviews
  - use checklists
  - see if/how checklists help

- **PSP2.1**: design specification and analysis techniques

Quality is *Personal*

- Defects are basic quality measure
  - typical rate: 100 defects per KLOC
- Quality software process should yield lower defect count

- Engineers (not organisations) introduce defects
  - Engineers (not organisations) should
    - remove them
    - determine their causes
    - learn to prevent them

Quality is *Personal*

- Testing removes only a fraction of the defects
  - more defects enter testing, more will slip through
- If you want to get a quality product out of test
  - you must put a quality product in

Reviews vs. Testing

- More efficient to find defects in reviews than in testing
  - unit test – about 2 to 4 defects per hour
  - code reviews – about 10 defects/hour
  - experienced reviewers can find 70+% of defects
  - unit test rarely exceed a 50% yield
- PSP data shows that reviews find 2 to 5 times as many defects per hour compared to unit test

Expecting Defects

- Use past defect injection signature
- Use program size estimate
- Allocate to phases using the to-date %

  ➔ Number of Defects to Expect!
- Plan for their removal

Defect Estimation Example

- Defect rate = 53 defects/KLOC
- New program: 195 LOC (estimated)

- Defect estimate = 53/1000 * 195
  ≈ 10 (rounded)

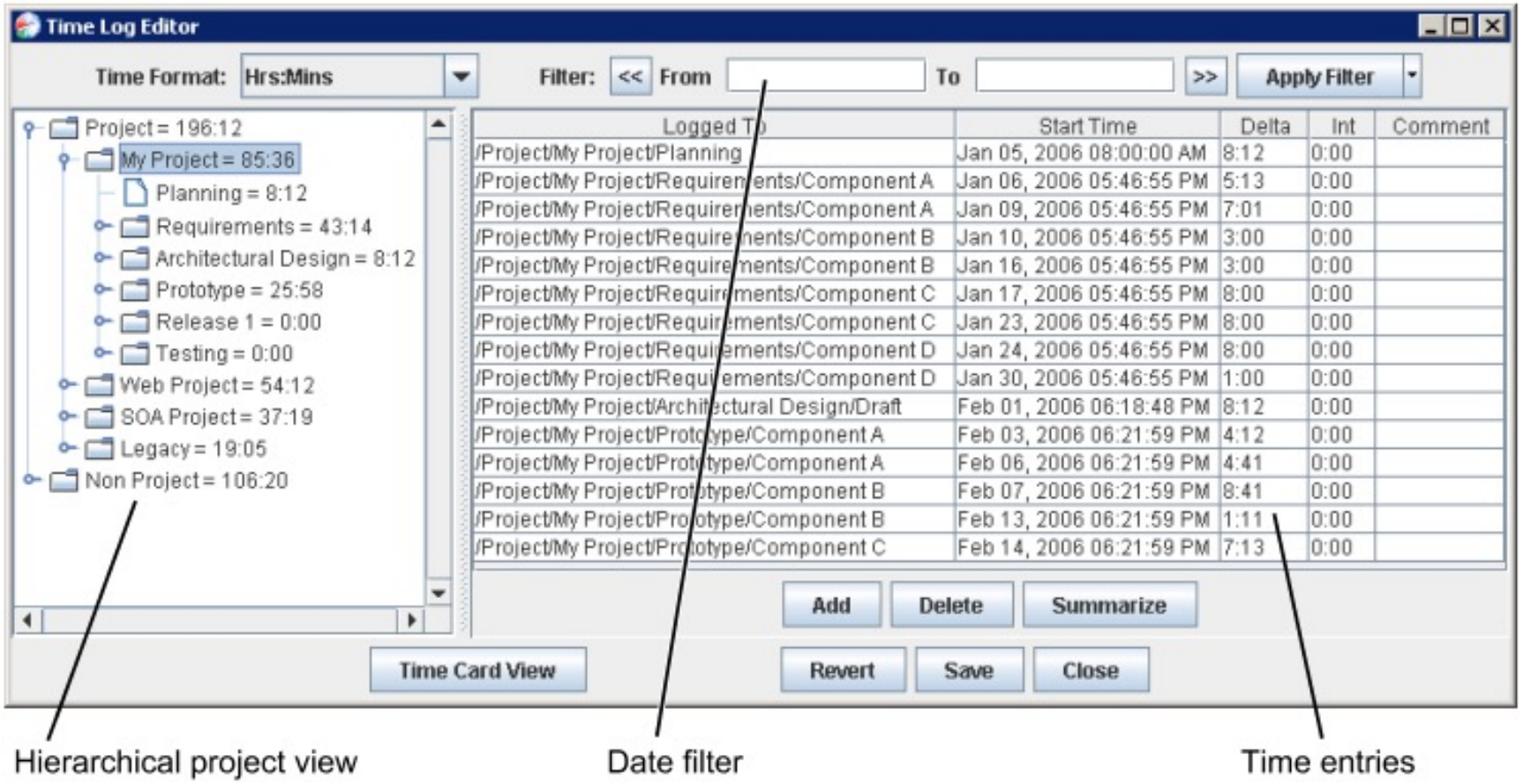
Summary: PSP =

- "CMM level 5 for individuals"
  - process for individual use
  - based on scaled-down industrial software practice
  - objective: help software engineers to work better

- Demonstrates value of
  - defined process
  - personal performance metrics

PSP Tool

- Software Process Dashboard
  - https://www.processdash.com/



Follow-up Reading

- Watts Humphrey (Addison Wesley)
  - *A Discipline for Software Engineering*, 1995
  - *Introduction to the Personal Software Process*, 1997
  - *PSP: A Self-Improvement Process for Software Engineers*, 2005
- Why Should you Use a PSP?
  - https://dl.acm.org/doi/10.1145/219308.219313
- SEI PSP Body of Knowledge
  - https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=8907

Next Steps

- Thursday Lecture
  - Final exam hints
- Tutorial
  - Final exam prep