Introduction to the Personal Software Process

A Simple Guide
To Managed
Software Development

B.J. Johnson
2009-11-13

Agenda

- What is PSP?
- Highlight Reel
- Baseline Process
- Some Basic Concepts
- PROBE
- Example steps
- Summary & Questions

So, To Business . . .

- Personal Software Process is:
  - A disciplined, data-driven approach
  - A way to apply CMMI principles
  - Focused on single developer
  - Method for improving planning skills
  - Method for improving estimation skills
  - Method for reducing defects
- Goal is to produce quality, defect-free software, delivered on schedule

PSP Isn’t Really New

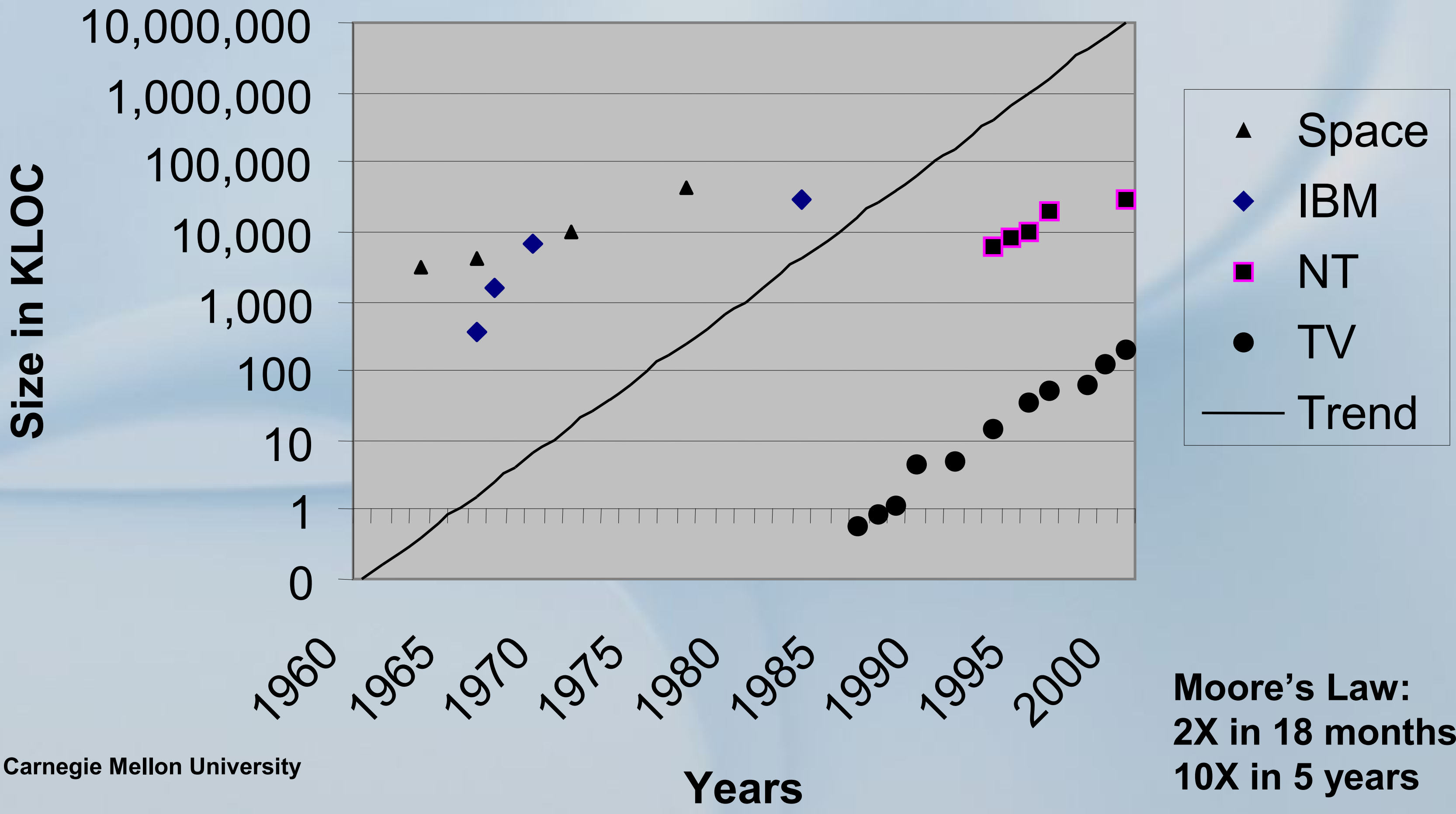
- Invented by Watts Humphrey
  - Joined SEI in 1986
  - Led S/W CMM development at SEI
  - Published PSP book in 1995 (hold up book here)
- PSP/TSP subsequently taken over by SEI
  - Training course available through SEI
  - Also taught at USC by Jim Alstad of BSS
- Lesson materials are available
  - From SEI/Carnegie-Mellon
  - Free download (for students only)
  - ZIP file contains slides and materials

Why Do We Need It?

- By some accounts:
  - over half of all software projects are significantly late and over budget
  - nearly a quarter of them are cancelled without ever being completed
- Even the best of us can make errors
  - Bugs may be hard to find
  - Bugs in production are expensive to fix
  - Large programs decrease productivity

Carnegie Mellon Slide Says:

Software Products are Bigger



Moore's Law:
2X in 18 months
10X in 5 years

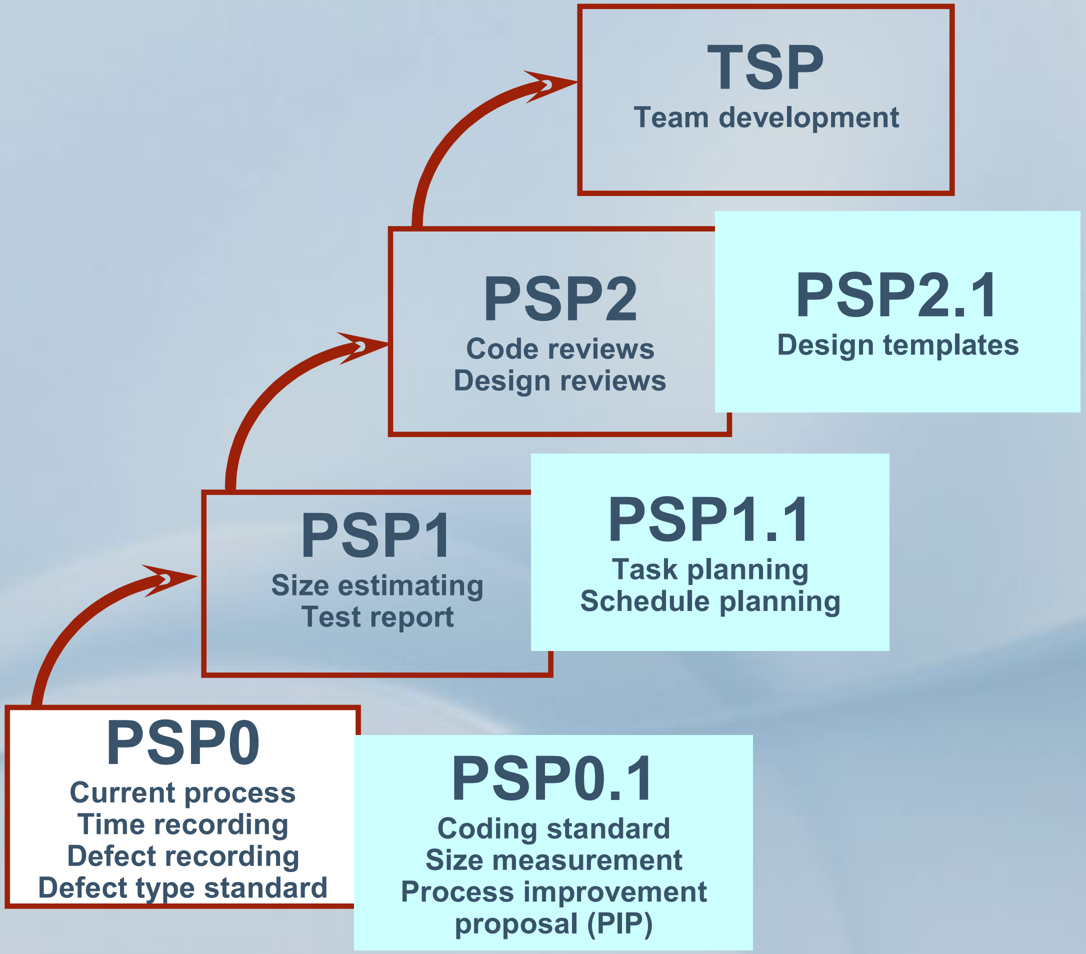
What Can PSP Buy You?

- Insight into *your* process
- Ideas for process improvements
- Framework to implement improvement
- Control over *your* process
- Sense of accomplishment
- Sense of improved teamwork

Highlights of PSP

- Matches up with CMMI Levels
- Has levels of implementation
- Based on real-world metrics
- Covers all parts of software process
  - Planning, estimation and design
  - Coding and reviews
  - Testing and “post-mortem” metrics
- Scripts tell you what to do
- Templates tell you how to do it
- Forms help you track progress

Level Up For Success …



CMMI Level 5

CMMI Level 4

CMMI Level 3

PSP Levels Zero &One

- PSP 0 (the baseline)
  - Your current process
  - Adding measurements
  - Coding/defect standards (in 0.1)
- PSP 1 (personal planning)
  - Test reporting
  - Size/resource estimation
  - Task/schedule planning (in 1.1)

PSP Levels Two & Three

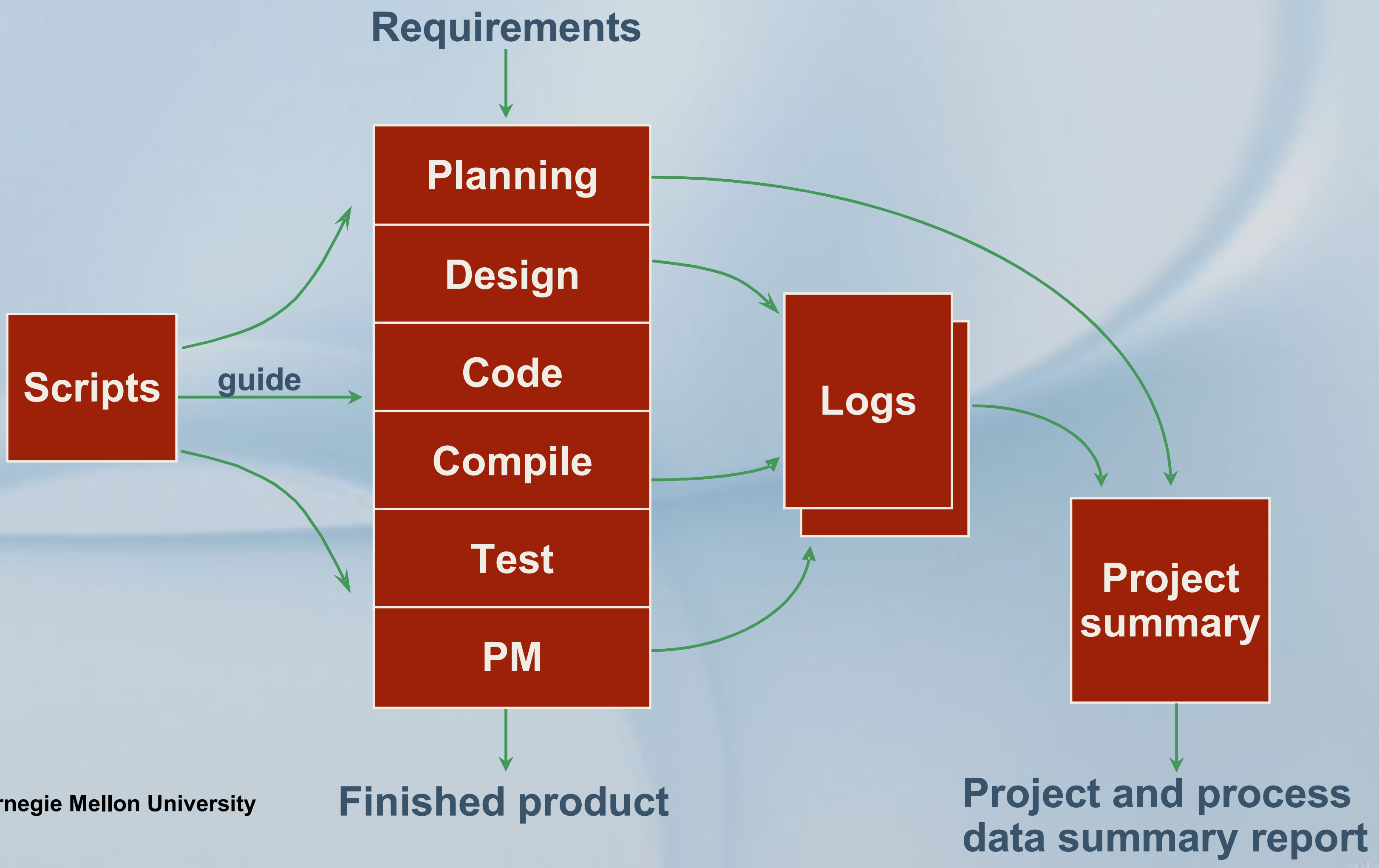
- PSP 2 (quality management)
  - Design reviews
  - Code reviews
  - Design completeness (in 2.1)
- PSP 3 (cyclic development)
  - Divide and conquer approach
  - Functional decomposition
  - Tracking of all cycles
  - A.K.A. Team Software Process

Cover the Baseline; PSP 0

- What we'll cover today
- Build around your existing process
- May be something like:
  - Get some requirements/think about them
  - Maybe make some working notes
  - Start coding/debugging
  - Integrate the parts
  - Test the product
  - Release the product/write documentation
  - Fix bugs forever

Carnegie Mellon Slide:

The PSP Process Flow



Time Recording

- Record numbers of minutes
- Log *everything* you do
- Even track non-productive time
  - Preparing/filling in forms
  - Looking up code on the web
  - Break times
  - Interruptions

Time Recording Log

- TRL example goes here

Defect Recording

- Start with basic defects
- Defects assigned numeric values
- Much like what is on Boeing Code Review sheets
- 10 = documentation error
- 20 = syntax error
- 30 = packaging error
- Etc.

Defect Recording Log

- drl example goes here

Project Plan Summary

- This is where you enter your numbers
- Tracks each project phase
- Tracks both estimated and actual
- Contains calculated totals
- Higher PSP levels have more stuff to calculate and track

Project Plan Summary

- pps example goes here

Formalize *Your* Baseline

- Planning
- Design
- Code
- Compile
- Test / Debug
- Add a “post-mortem” step

So, How Do We Estimate?

- Welcome to PROBE!
- <u>**PRO**</u>xy <u>**B**</u>ased <u>**E**</u>stimation
- Uses a table of values based on the different types of functions/objects
- All estimations based on SLOC
- Logical vs. Physical SLOC counting
- Different languages have different SLOC for the same functions
- See SEI definitions

The Book C++ PROBE Table

C++ Object Size in LOC per Method					
Category	**Very Small**	**Small**	**Medium**	**Large**	**Very Large**
Calculation	2.34	5.13	11.25	24.66	54.04
Data	2.60	4.79	8.84	16.31	30.09
I/O	9.01	12.06	16.15	21.62	28.93
Logic	7.55	10.98	15.98	23.25	33.83
Set-up	3.88	5.04	6.56	8.53	11.09
Text	3.75	8.00	17.07	36.41	77.66

So, Step One … Pre-plan

- Read the requirements
- UNDERSTAND the requirements
- Check the scripts
- Set up your templates
- Create a “conceptual design”
  - Like an architectural level design
  - Functions/objects are itemized
  - Shouldn’t take more than ½ hour for most programs

Step Two … Planning

- Check the PROBE table and figure out the lines of code needed
- If starting from an existing program, separate out and count deleted, changed, added, modified LOC
- Calculate the total number of LOC
- This is your estimated LOC count
- Enter info into Project Plan Summary form
- Enter time into Time Recording Log

Step Three … Design Phase

- Now, create an actual design for the project
- Use functional decomposition
- Modify the Project Plan Summary form entries if necessary
- Enter time in the Time Recording Log

Step Four … Code (at last!)

- Write the code
- Don’t compile it yet!!!
  - This is counter-intuitive to usual practice
  - We almost always compile as we go, but NOT for PSP!!!
  - Critical to later steps and to higher PSP levels
- Fill out Time Recording Log

Step Five … Compile It

- Compile all your code modules
- Log all your defects to the Defect Recording Log
- Log as much detail as you can:
  - Line numbers
  - Compile error codes and messages
  - This is where syntax errors will show up
  - Use cut-and-paste if you want
- Critical to later PSP levels and to your development
- Fill out Time Recording Log

Step Six … Test Your Code

- When things break, these are defects
- Log them to the Defect Recording Log
- Fix the defects and re-test
- Keep doing this until they are all gone
- This is where logic errors will show up
- Fill out Time Recording Log

Step Seven … Post-mortem

- You now have a completed program
- This is where your metrics come in
- Record the actual time required for each phase from the Time Recording Log on the Project Plan Summary form
- Figure out (and record!) percent of total time each phase took
- Enter the actuals for all LOC categories
- Figure out LOC/hour

For The Next Project

- You now have one project’s data
- Next project, use the percents needed for each phase to help calculate times
- When you have three projects recorded, you can use linear regression to estimate time
- $y = mx + b$, just like in algebra!
  - The x values are the actual values
  - The y values will be the predicted values

How Do I Get “m” and “b”?

- PSP calls them “Betas”
  - β_0 is the “b” or intercept
  - β_1 is the “m” or slope
- You MUST have at least three programs’ worth of historical data to use regression!!!
- If you don’t, use total LOC divided by LOC/hour to estimate time required

More On Regression

- There’s lots of statistics involved
  - Standard deviation
  - Variance
  - Prediction intervals
  - Etc.
- There are spreadsheets available to do the calculations
- Check the SEI website for more details

…And Finally …

- Process Improvement
  - This is what happens when moving from level to level in PSP
  - Adding additional parts to the process
  - Adding additional tracking metrics
- PSP allows for PPDP which is Personal Process Development Process!!

More Information/Training

- You can download a complete set of student training materials from the SEI website
- It's in an 8 Meg zip file so you may need to do it from home
- You give them your address info in an online form and you can download it
- Go to: http://www.sei.cmu.edu/tsp/tools/student/index.cfm

Thank You So Much!