

Empirical Research into PSP

Mark Paulk
ISR Faculty Meeting
20 October 2008

<http://www.cs.cmu.edu/~mcp/>

Erdogmus, 2008 – “Must Software Research Stand Divided?”

Empirical vs constructionist software research

- Empirical research is boring.
- Empirical research takes too long.
- Empirical research is too soft. It's more like social or management science, and it doesn't produce any technology.
- We don't need empirical evidence to know whether something works. If it's widely used, it works. If nobody uses it, it doesn't work.
- It's impossible to prove a technique effective when there are so many factors, so why bother?
- Software practice and technology are changing too fast. There's no point.
- If it worked on a few students with a few toy examples, that doesn't mean it will work for me. Why should I care?
- Empirical research is dangerous. People misunderstand, misrepresent, or over-interpret the findings.
- Empirical researchers are biased. Why should they be trusted?
- Empirical research uses evidence models from other disciplines. These models don't apply to software development.

Validity

Construct validity means that the independent and dependent variables accurately model the abstract hypotheses.

Internal validity means that changes in the dependent variables can be safely attributed to changes in the independent variables.

External validity means that the study's results generalize to settings outside the study.

Perry, Porter, & Votta, "Empirical Studies of Software Engineering: A Roadmap"

3

Status of Empirical Research in the Software Industry

"a truly 'empirical' basis for software engineering remains a distant dream"

- **Fenton, "Viewpoint Article: Conducting and Presenting Empirical Software Engineering"**

"What we believe about which approaches are best is based on anecdotes, gut feelings, expert opinions, and flawed research, not on careful, rigorous software engineering experimentation."

- **Fenton, Pfleeger, & Glass, "Science and Substance"**

"A survey of 400 recent research articles suggests that computer scientists publish relatively few papers with experimentally validated results."

- **Tichy, Lukowicz, Prechelt, & Heinz, "Experimental Evaluation in Computer Science: A Quantitative Study"**

4

Personal Software Process (PSP)

Applies the CMM concepts of process discipline and quantitative management to the work of the individual software professional in a classroom setting.

Focuses on planning, quality, and productivity.

Typically involves the development of ten programs, using increasingly sophisticated processes.

Watts Humphrey, A Discipline for Software Engineering, 1995.

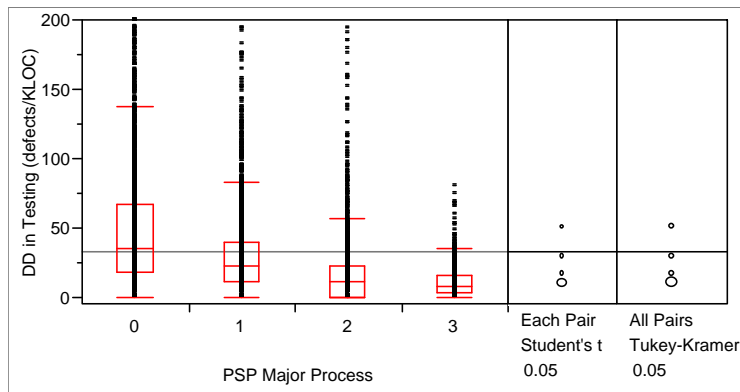
5

PSP Assignments, Processes, and Major Processes

<u>Major PSP</u>	<u>Process</u>	<u>Asgn</u>	<u>Process</u>	<u>Practices</u>
0		1A	PSP0	current process, basic measures coding standard, process improvement proposal, size measurement
		2A	PSP0.1	
		3A	PSP0.1	
1		4A	PSP1	size estimating, test report task planning, schedule planning
		5A	PSP1.1	
		6A	PSP1.1	
2		7A	PSP2	code reviews, design reviews design templates
		8A	PSP2	
		9A	PSP2.1	
3		10A	PSP3	cyclic development

6

Improvements in PSP Quality (PSP2001)



The seminal study on PSP is
Hayes, W. and J.W. Over. "The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers." Software Engineering Institute, Carnegie Mellon University, 1997.

7

Issues to Consider

Designed experiment vs retrospective analysis

Validity of data

Independent and dependent variables

Outliers

Appropriate statistical analysis techniques

Data transformations

Generalizability

8

Empirical – Retrospective – Not Experimental

Analyzing PSP data may be empirical but it is not experimentation.

- ... Herbsleb, “Hard Problems and Hard Science: On the Practical Limits of Experimentation”

9

PSP Data Sets

	2001	2001	2001	1997	1997	1997
		C	C++		C	C++
9A	830	152	82	163	61	13
10A	737	133	66	145	57	11
Totals	9934	1758	920	1970	677	140

10

Removing Data Points

“It is often difficult to tell from published papers whether any data points have been removed before analysis, and if they have, the reasons why.”

- **Fenton & Neil, “A Critique of Software Defect Prediction Models”**

Removed

- **data points from classes that did not use the standard ten assignments**
- **observations with internal inconsistencies**
 - 2.8 - 4.4% of data

11

Independent and Dependent Variables

Operationally defining “quality”

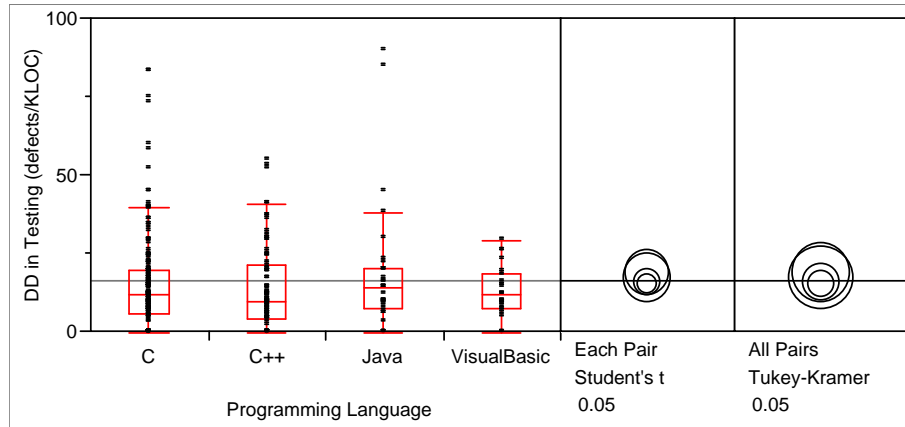
- **defect density in testing**
- **defect removal effectiveness in design/code reviews**

Independent variables: programmer ability, program size, design/code time, design/code review rate, defect density in design/code review, defect density in compile

Other factors: Assignment, Finishing All Ten Assignments, PSP Classes, Highest Degree Attained, Years of Programming Experience, Number of Languages Known, Percent of Time Programming, Programming Language

12

Programming Language Used



13

Programmer Ability

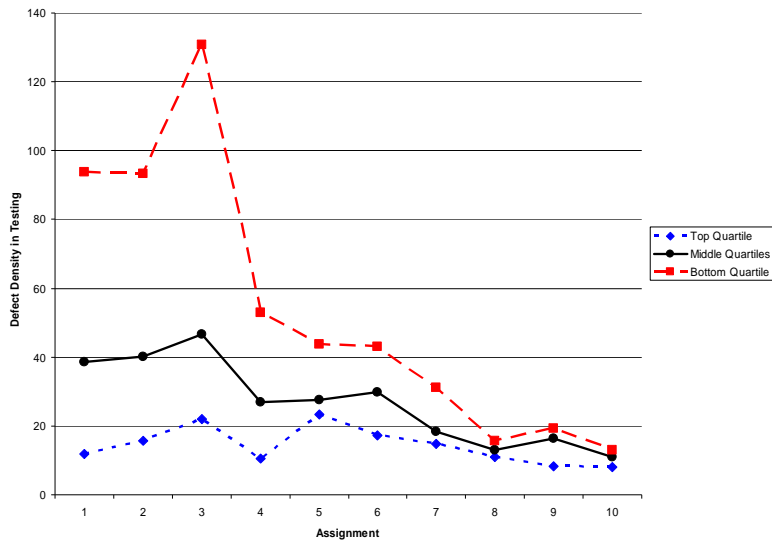
Fundamental quality principle – competent programmers

Used average quality for assignments 1A to 3A to categorize students

- top quartile
- middle two quartiles
- bottom quartile

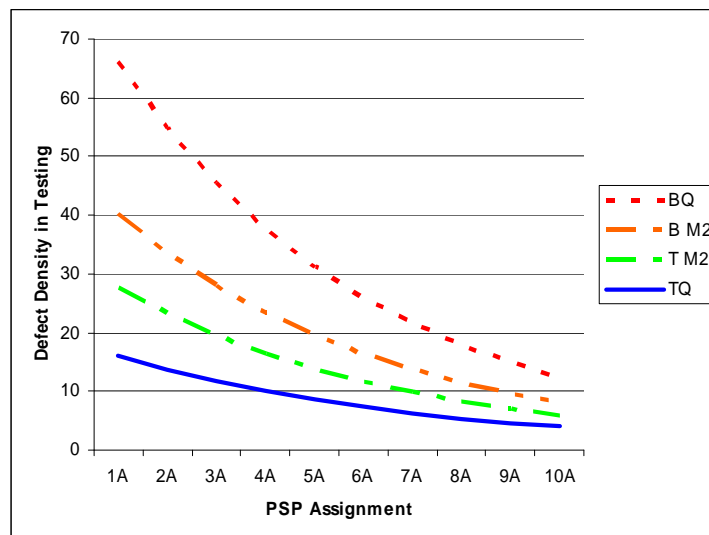
14

Trends by Programmer Ability



15

Trends in Quality by Quartile Using Averages of Student Regressions



16

Issues in the Exploratory Analysis

Number of data points for each *PSP major process* is different since some students drop out

Each data set contains multiple data points per student, therefore the usual assumption of independence is violated.

Software data is usually skewed.

17

Non-Normality

The number of defects found in a PSP review cannot be accurately characterized by the Poisson distribution.

- **u-charts assuming the Poisson distribution may be misleading**

The negative binomial distribution should be considered for counts of defects.

Robust techniques (in the presence of non-normal data) should be preferred for software data.

Distributional assumptions should be tested against the data being analyzed in any specific case.

18

Mixed Models

Natural growth curves preserve the concept that individual differences are both meaningful and important, even when everyone develops the same way.

Hayes and Over used repeated measures ANOVA – a special case of growth curve models.

Mixed models incorporate

- repeated measures
- both fixed and random effects

19

Some Concerns in Generalizing PSP Results

Classroom measures – lack of team/project measures

Small programs – industry programs are much larger and more complex

Missing data – non-process data was not consistently collected (for confounding variables)

No requirements phase – many real-world problems are because of requirements problems

20

Effective Teams

Group thinking is usually better, less variable, and more precise than individual thinking.

- *Hare, Hoerl, Hromi, and Snee, "The Role of Statistical Thinking in Management"*

Range of team performance, rather than being an order of magnitude, tends to be between 85% and 115% of the norm.

- *DeMarco and Lister, **Waltzing with Bears: Managing Risks on Software Projects***

21

Useful Empiricism

"Our studies should strive to establish principles that are causal, actionable and general."

"What's important is not whether the study is textbook perfect, but whether the study and its conclusions taken as a whole are credible."

- *Perry, Porter, & Votta, "Empirical Studies of Software Engineering: A Roadmap"*

22

Conclusions 1 of 2

Program size, programmer ability, and disciplined processes significantly affect software quality.

Factors frequently used as surrogates for ability and technology do not significantly impact software quality.

Recommended practices are not necessarily followed even when processes are consistently performed.

Programmer performance improves 2-5X or more when disciplined processes are followed, and variability decreases by 2-5X or more.

23

Conclusions 2 of 2

Top-quartile performers improve by about 2X; bottom-quartile performers improve about 5X.

Defect count data for the classroom data sets do not fit the Poisson distribution but frequently fit the negative binomial distribution.

Process-based regression models can address 60-80% of the variation in predicting software quality when disciplined processes are followed.

Mixed models rigorously confirm these results.

24