# Myth of the x10 Programmer

William Nichols

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

# Measurement of Programmer Differences Replicates!

*"One of the most replicated results in software engineering research is the 10-fold difference in productivity and quality between different programmers with the same levels of experience." – Steve McConnell [McConnell, 2002]*

| Source | Ratio | | |
|---|---|---|---|
| [Sackman, 1968] | 28-1 | [Card 1987] | |
| [Curtis 1981] | | [Boehm and Papaccio 1988] | |
| [Mills 1983] | | [Valett and McGarry 1989], | |
| [DeMarco and Lister 1985] | | [Boehm 1975, 2000] | 5-1 |
| [Curtis et al. 1986] | | [Schwartz, 1968] | Order of magnitude |

# Why does this matter

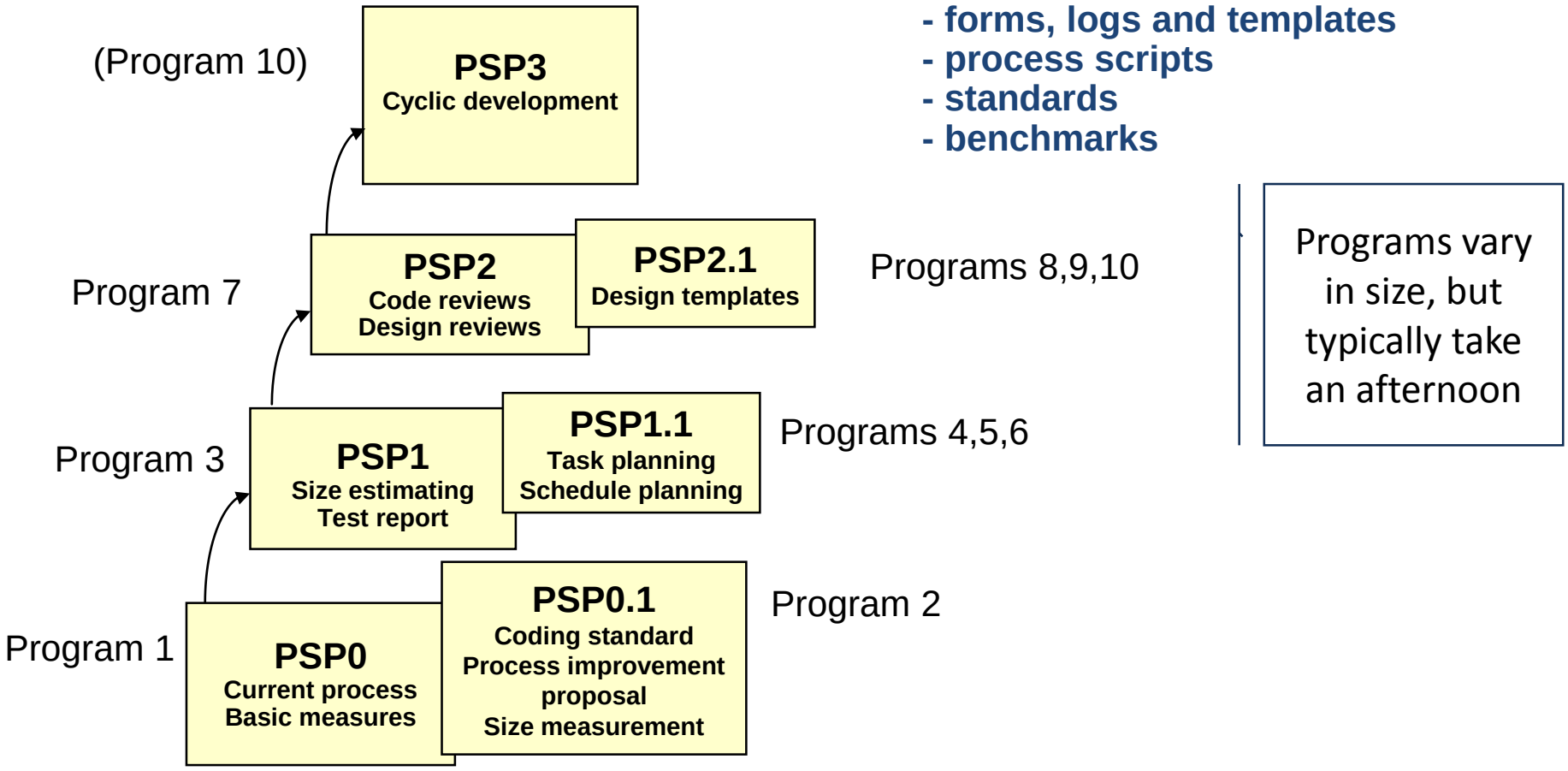
Huge productivity ranges between programmers should affect

- Training
- Hiring
- Evaluation
- Promotion
- Pay
- Planning

# Questions

1) Can we replicate the variation with our data?
2) What factors (experience, education) account for the variation?

# PSP course, 10 programs




**PSP components**
- **forms, logs and templates**
- **process scripts**
- **standards**
- **benchmarks**

Programs vary in size, but typically take an afternoon

# Approach

Focus only on Effort

Limit sample to C Programmers

Apply causal search techniques

Apply Linear Mixed Models for variation

# Causal search found nothing

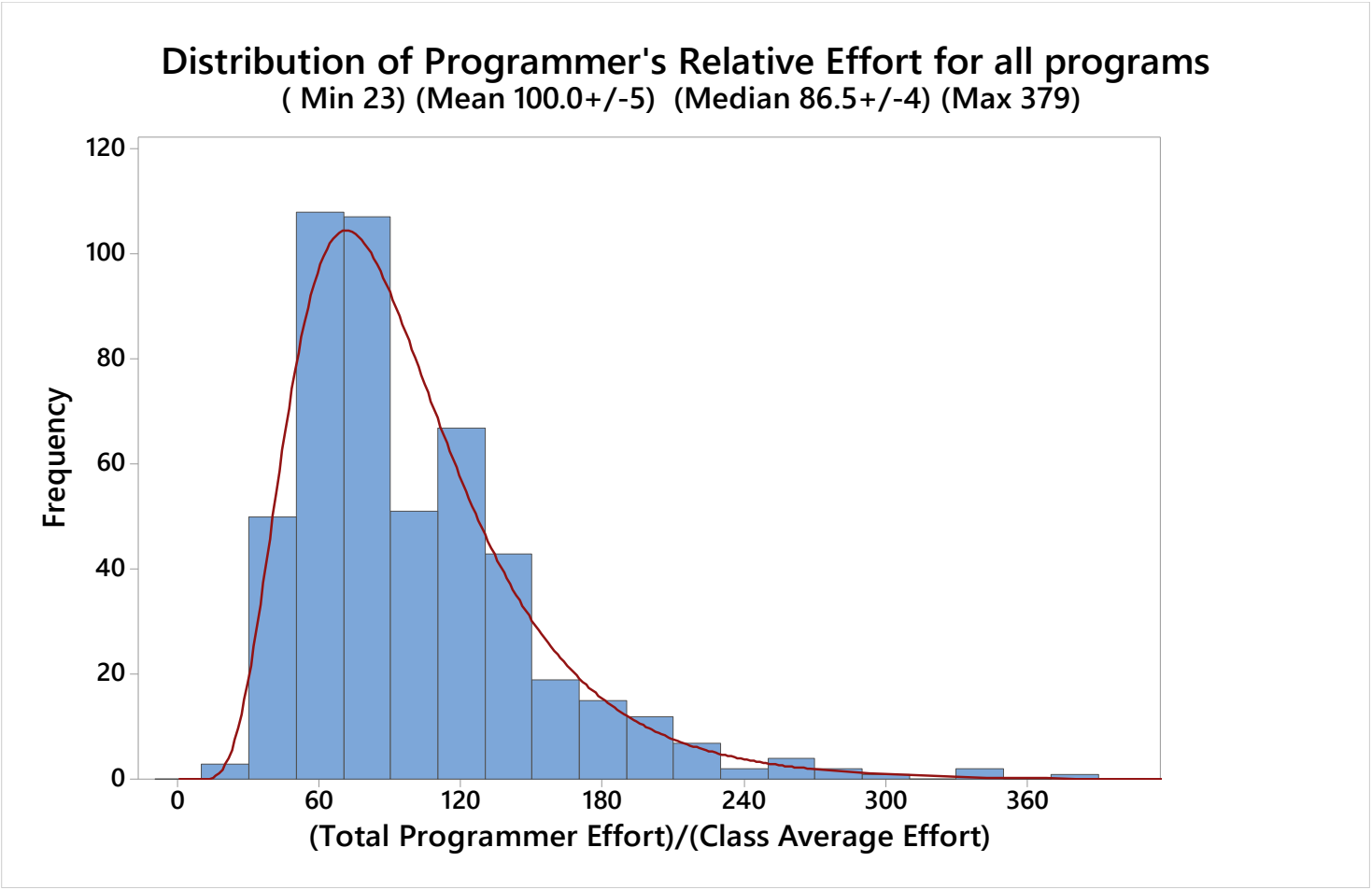
1) There were no correlations (let alone causation) between experience and productivity

2) The “within” variation was surprisingly strong.

3) Linear mixed model showed individual programmer results were only almost as variable as between programmers.

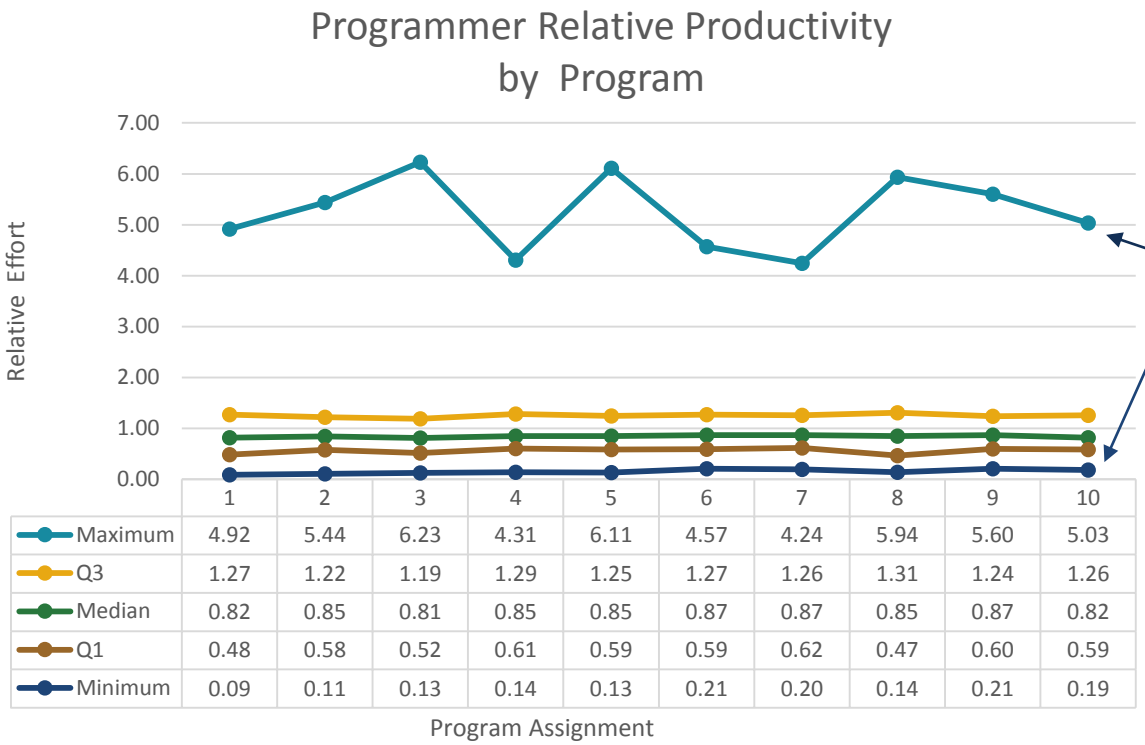
Followed up with closer look at the variation.

# Distribution of Relative Effort for All Programs




| | Mean | Standard deviation | 5% | 25% | Median | 75% | 95% |
|---|---|---|---|---|---|---|---|
| Relative effort | 1 | 0.51 | 0.44 | 0.63 | 0.86 | 1.24 | 1.93 |

# Min/Max Productivities by Program

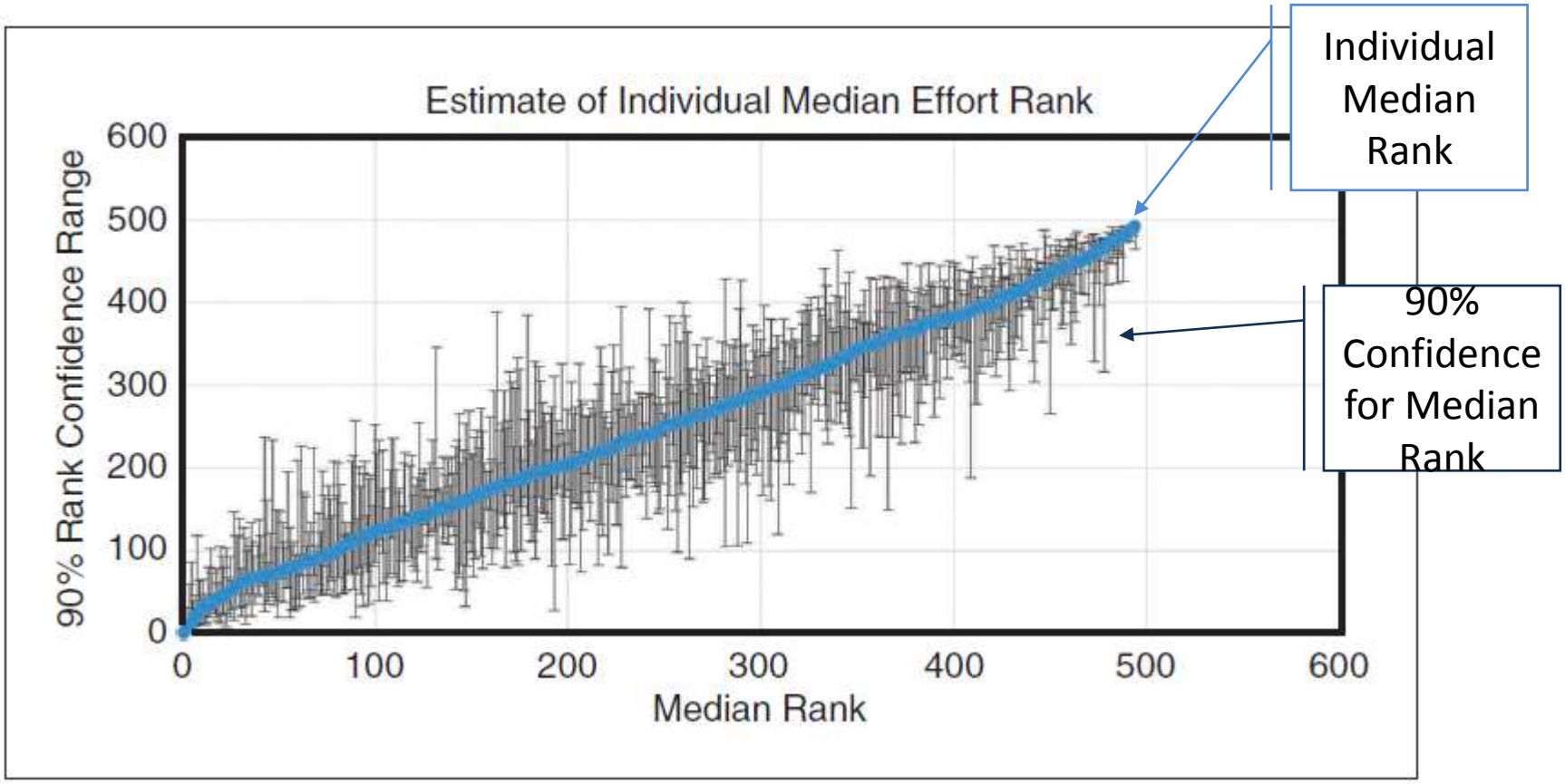



| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Maximum | 4.92 | 5.44 | 6.23 | 4.31 | 6.11 | 4.57 | 4.24 | 5.94 | 5.60 | 5.03 |
| Q3 | 1.27 | 1.22 | 1.19 | 1.29 | 1.25 | 1.27 | 1.26 | 1.31 | 1.24 | 1.26 |
| Median | 0.82 | 0.85 | 0.81 | 0.85 | 0.85 | 0.87 | 0.87 | 0.85 | 0.87 | 0.82 |
| Q1 | 0.48 | 0.58 | 0.52 | 0.61 | 0.59 | 0.59 | 0.62 | 0.47 | 0.60 | 0.59 |
| Minimum | 0.09 | 0.11 | 0.13 | 0.14 | 0.13 | 0.21 | 0.20 | 0.14 | 0.21 | 0.19 |

Program Max/Max more extreme

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Max/Min | 54.9 | 51.1 | 48.5 | 31.1 | 46.5 | 21.8 | 21.5 | 41.5 | 26.6 | 26.9 |
| Q3/Q1 | 2.6 | 2.1 | 2.3 | 2.1 | 2.1 | 2.1 | 2.0 | 2.8 | 2.1 | 2.1 |

# How did individual rank by effort vary?




90% confidence in individual rank spans half the sample.

# Similar within and between productivity variation

1) Within variation had not previously been reported
2) Single point rather than repeated measures
   - Are not reliable (high variation)
   - Exaggerate range
3) Can get what ever min/max ratio you want by varying sample size and other assumptions
4) Wide min/max range remains, but
5) Most developers are within a much narrower productivity range

# Limitations of This Study

This study only used C programmers (15% of the sample)

Programs are not broadly representative

- Mostly implement math and  numeric
- Small
- Simple and well defined

Drop out bias? (common problem among the studies)

Selection bias for entry?

Some “re-use” affects individual program results.

Significant differences remain.

# Implications for Researchers

Think carefully about how to analyze your data

What are you measuring?

Are the measurements reliable? Check your assumptions

What about validity?

- Content
- Construct
- Predicative

Research into environmental conditions affecting variation

If you want a learning set, consider the PSP data

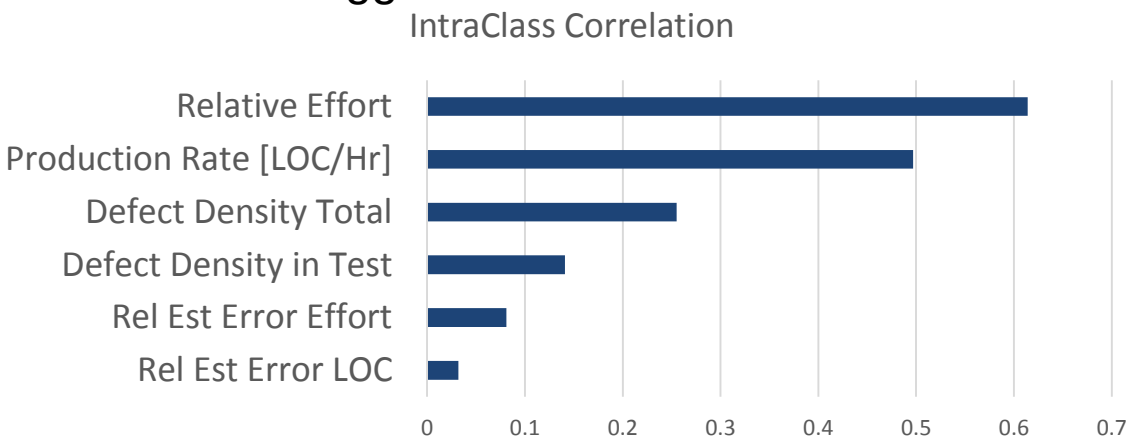
# Implications

This may have implications for Industry

- Min/max doesn't tell us much because it's mostly from the low end.
- Be cautious testing, evaluating, or ranking programmers
- Don't over react to variation
- Think about how to establish baselines or estimate (e.g. duration of rolling average for sprints)
- Look for controllable sources of variation (test, reviews)
- Recognize outlier events for intervention

# Future work

Linear mixed models are suggestive




| | Rel Est Error LOC | Rel Est Error Effort | Defect Density in Test | Defect Density Total | Production Rate [LOC/Hr] | Relative Effort |
|---|---|---|---|---|---|---|
| Series1 | 0.032 | 0.081 | 0.141 | 0.255 | 0.497 | 0.614 |

Relative Effort had the highest intra class correlation of these!

Look at effects across programming languages

# Conclusions

The scale of individual task variation has been under our noses for 50 years.

Basic questions of validity have not been addressed?

- No one thought to quantify the measurement reliability.
- What is the predictive validity? Across what units?
- What are we actually trying to measure? Construct validity?

What else have we missed?

# Acknowledgements

Mike Konrad and SCOPE project for supporting this work

Tim Menzies for his guidance and editing for the IEEE article

PSP Instructors and Students who contributed their data

The PSP team

# To use the PSP data

PSP STUDENT ASSIGNMENT DATA

**Permalink:**

http://ieee-dataport.org/open-access/psp-student-assignment-data

**DOI Link:** http://dx.doi.org/10.21227/a5vb-cf02

**Short Link:** http://ieee-dataport.org/1783

Citation: William Nichols, Watts Humphrey, Julia Mullaney, James McHale, Dan Burton, Alan Willett, "PSP Student Assignment Data", IEEE Dataport, 2019. [Online]. Available: http://dx.doi.org/10.21227/a5vb-cf02.

The PSP and TSP course materials are available on the SEI Digital Library at https://www.sei.cmu.edu/go/tsp for use under the terms of the Creative Commons license

# References


Glass, R. L. **Facts and Fallacies of Software Engineering**. (2002). Boston, MA: Addison Wesley.

McConnell, S. Chapter 30 . *What Does 10x Mean ? Measuring Variations in Programmer Productivity*. **Making software : what really works, and why we believe it** (2010**).**

Sackman, H., W. I. Erikson, and E. E. Grant. 1968. "Exploratory Experimental Studies Comparing Online and Offline Programming Performances." Communications of the ACM, Jan.

Schwartz, Jules. 1968. "Analyzing Large-Scale System Development." In Software Engineering Concepts and Techniques, Proceedings of the 1968 NATO Conference, edited by Thomas Smith andKaren Billings. New York: Petrocelli/Charter.