# Demonstrating the Impact of the PSP on Software Quality and Effort: Eliminating the Programming Learning Effect

**Diego Vallespir** **Fernanda Grazioli**

**Leticia Pérez** **Silvana Moreno**

Grupo de Ingeniería de Software
Universidad de la República
Uruguay

# Agenda

Introduction

Research question

Experiment setup

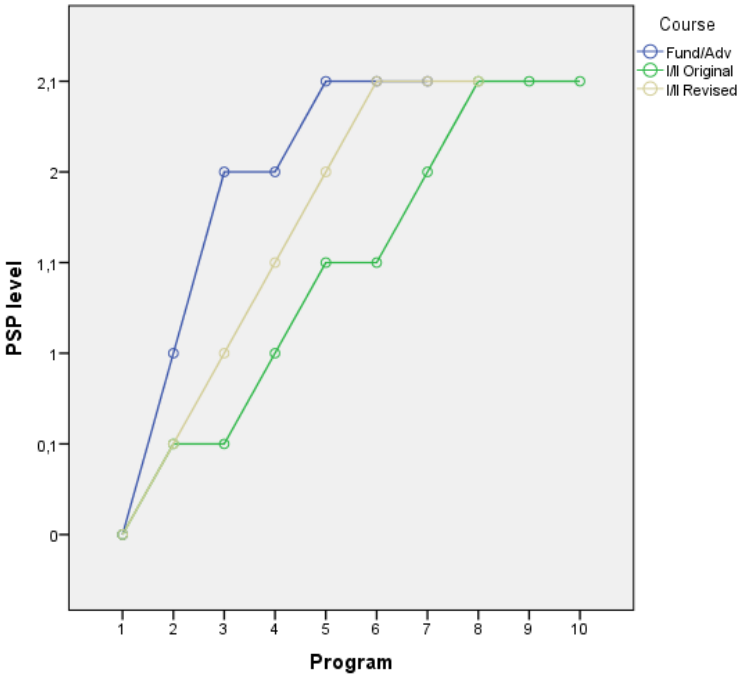
Results

Conclusions and Future Work

# What do we know about PSP results?

- Statistical analysis of the evolution of the results showing that during the course the engineer improves his/her performance
  - Then it can be statistically inferred that the PSP is responsible for the quality improvement
  - But, in fact, this is only one possible reason
- Possible reasons for the improvement
  - Learning the domain of the PSP course exercises
  - Repeat the programming recording data
  - Etc.

# One prior work




- A Cross Course Analysis of Product Quality Improvement with PSP
  Grazioli and Nichols
  TSP Symp. 2012
- An Analysis of Student Performance during the Introduction of the PSP: An Empirical Cross Course Comparison
  Grazioli, Nichols and Vallespir
  TSP Symp. 2013

But repetition still has a chance of be an important factor in the improvements

# Our Research

Research question:

Are the performance improvements observed in the PSP courses due to the introduction of the phases and techniques of the PSP or to programming repetition?

We want to observe

- Product quality
- Testing effort

# Controlled Experiment

We designed and performed a controlled experiment with twelve software engineering undergraduate students at the Universidad de la República.

The students performed the exercises from the PSP for Engineers I/II course without applying the PSP techniques.

# Experiment Setup - Measures

Product quality measures

- Defect density in unit test
- Total defect density of the program

Effort used in unit testing measures

- Time in unit testing per KLOC
- Average time in unit testing per defect found.

# Experiment Setup - Hypotheses

- A statistical hypothesis is an assumption about a population parameter. Hypothesis testing refers to the formal procedures used in experimentation to accept or reject statistical hypotheses
- The hypotheses aim at knowing if comparing a developed program to another one developed previously, the software engineer improves his performance in any of the aspects mentioned
- So, we compare programs by pairs to find if the changes in each performance dependent variable are statistically significant

# Experiment Setup - Subjects

- Computer science students in the final years (4th or 5th year of the career)
- With at least four courses on programming
- The students participate in the experiment in order to obtain credits. So, they are motivated
- They don’t know they are taking part in an experiment
- They know that the data they collect will be used in research work

# Experiment Setup – Experiment Design

- 12 students
- Each one developed 8 programs (same programs for each student and developed in the same order)
- Repeated measure design
  - Several measures are taken for the same subjects
- PSP0 used in the first program
- PSP0.1 for the other programs
  - We are only collecting data and not introducing PSP phases or techniques (reviews, design, PROBE, etc.)

# Results – Median and Interquartile Range

*Median and interquartile range for the four variables under study*

| ***Defect Density in Unit Testing (#defects found in UT / )*** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **Pr 1** | **Pr 2** | **Pr 3** | **Pr 4** | **Pr 5** | **Pr 6** | **Pr 7** | **Pr 8** |
| **Median** | 24.55 | 56.98 | 18.13 | 18.48 | 36.38 | 18.40 | 13.78 | 8.59 |
| **IQR** | 13.65 | 21.20 | 31.84 | 18.14 | 30.19 | 17.11 | 25.11 | 12.20 |
| ***Total Defect Density per (#defects found / )*** | | | | | | | | |
| **Median** | 111.11 | 136.59 | 72.51 | 74.04 | 137.00 | 61.33 | 63.80 | 40.06 |
| **IQR** | 49.19 | 151.87 | 89.24 | 51.52 | 124.61 | 51.31 | 83.18 | 63.14 |
| ***Time Spent in Unit Testing per (minutes in UT / )*** | | | | | | | | |
| **Median** | 331.28 | 1297.97 | 301.52 | 241.94 | 638.80 | 652.71 | 540.85 | 338.76 |
| **IQR** | 335.59 | 1044.97 | 345.24 | 301.34 | 1136.47 | 1297.96 | 523.87 | 490.12 |
| ***Average Time Spent in Unit Testing per Defect (minutes in UT / #defects found in UT)*** | | | | | | | | |
| **Median** | 11.33 | 16.61 | 15.00 | 11.75 | 20.50 | 37.00 | 29.00 | 39.00 |
| **IQR** | 7.75 | 17.46 | 10.00 | 15.75 | 12.17 | 40.75 | 37.00 | 28.25 |

# Results – Hypotheses test

- In a context of few samples and repeated measures the most suitable statistical hypotheses test is the Wilcoxon signed-ranks test
- We used the 2-tailed Wilcoxon test because we do not know a priori if the dependent variables will increase or reduce their values
- In the results
  - Each cell contains the p-value (2-tailed) of the Wilcoxon test.
  - The cells in green or red indicate that the null hypothesis has been rejected (p<=0.05).
    - The green ones indicates improvement
    - The red ones indicate the opposite
  - The grey cells indicate that it has not been possible to reject the null hypothesis.

# Results – DDUT

***Hypotheses test for defect density in unit testing***

| Prog. | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 1 | p=0.028 | p=0.722 | p=0.158 | p=0.347 | p=0.136 | p=0.388 | p=0.006 |
| 2 | | p=0.006 | p=0.003 | p=0.019 | p=0.002 | p=0.010 | p=0.002 |
| 3 | | | p=0.754 | p=0.084 | p=0.937 | p=0.754 | p=0.272 |
| 4 | | | | p=0.117 | p=0.929 | P=1.000 | p=0.136 |
| 5 | | | | | p=0.015 | p=0.084 | p=0.006 |
| 6 | | | | | | p=0.929 | p=0.084 |
| 7 | | | | | | | p=0.209 |

**DDUT for program 2 is higher than in the rest of the programs**

**Possible reason: Program 2 is different from the rest**

# Results – DDUT

***Hypotheses test for defect density in unit testing***

| Prog. | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 1 | p=0.028 | p=0.722 | p=0.158 | p=0.347 | p=0.136 | p=0.388 | p=0.006 |
| 2 | | p=0.006 | p=0.003 | p=0.019 | p=0.002 | p=0.010 | p=0.002 |
| 3 | | | p=0.754 | p=0.084 | p=0.937 | p=0.754 | p=0.272 |
| 4 | | | | p=0.117 | p=0.929 | P=1.000 | p=0.136 |
| 5 | | | | | p=0.015 | p=0.084 | p=0.006 |
| 6 | | | | | | p=0.929 | p=0.084 |
| 7 | | | | | | | p=0.209 |

**No continuous improvement**

**No improvement from program 3, 4, 6 or 7 to following ones**

**Differences found between exercises (5,6) and (5,8) could be due to program 5 and not to the process**

# Results – TDD

***Hypotheses test for total defect density***

| Prog. | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 1 | p=0.239 | p=0.239 | p=0.010 | p=1.000 | p=0.004 | p=0.041 | p=0.008 |
| 2 | | p=0.034 | p=0.010 | p=0.158 | p=0.003 | p=0.006 | p=0.005 |
| 3 | | | p=0.695 | p=0.182 | p=0.041 | p=0.530 | p=0.034 |
| 4 | | | | p=0.050 | p=0.108 | p=0.480 | p=0.050 |
| 5 | | | | | p=0.004 | p=0.084 | p=0.012 |
| 6 | | | | | | p=0.754 | p=0.347 |
| 7 | | | | | | | p=0.158 |

**No continuous improvement**

**Program 6 and 8 show an improvement in the total defects density**

# Results – TSUT

***Hypotheses test for time spent in unit testing***

| Prog. | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 1 | p=0.005 | p=0.937 | p=0.388 | p=0.023 | p=0.019 | p=0.308 | p=0.754 |
| 2 | | p=0.023 | p=0.003 | p=0.209 | p=0.433 | p=0.034 | p=0.003 |
| 3 | | | p=0.530 | p=0.117 | p=0.136 | p=0.480 | p=0.638 |
| 4 | | | | p=0.012 | p=0.015 | p=0.209 | p=0.480 |
| 5 | | | | | p=0.209 | p=0.308 | p=0.041 |
| 6 | | | | | | p=0.117 | p=0.028 |
| 7 | | | | | | | p=0.530 |

**No continuous improvement**

**Programming repetition does not result in an improvement in the time spent in UT**

# Results – TSUT

***Hypotheses test for average time spent in unit test per defect***

| Prog. | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 1 | p=0.050 | p=0.155 | p=0.575 | p=0.059 | p=0.021 | p=0.047 | p=0.010 |
| 2 | | p=0.859 | p=0.389 | p=0.929 | p=0.038 | p=0.093 | p=0.010 |
| 3 | | | p=0.214 | p=0.386 | p=0.051 | p=0.386 | p=0.041 |
| 4 | | | | p=0.594 | p=0.051 | p=0.093 | p=0.009 |
| 5 | | | | | p=0.008 | p=0.047 | p=0.004 |
| 6 | | | | | | p=0.575 | p=0.878 |
| 7 | | | | | | | p=0.790 |

**No continuous improvement**

**In fact in several cases more effort is needed in UT per defect found**

**More experiments are needed in order to investigate the reasons**

# Results – Sum Up

Since the experiment does not change the level of PSP used (PSP0.1 from program 2 to 8) the results of this experiment indicate that the programming repetition:

- Do not continuously improve defect density in UT
- Seems to improve in the last 3 programs the total defect injection (This can be due more to the data collection about the defects injected than to the learning effect of the application domain).
- Do not continuously improve the time spent in UT per KLOC
- It seems to deteriorate the efficiency of UT

# Conclusions and Future Work

- The presented results contribute to the elimination of an important threat to the validity of different experiments performed with the PSP
- Besides, this experiment shows that without the adequate practices the quality of software and the performance of the process cannot be improved by the simple reason of the programming learning effect

- As future work we intend to replicate this experiment, analyze other data and design a more complex experiment that will enable us to isolate and study the different practices of the PSP and the synergy produced between them

# Questions

**Diego Vallespir**

dvallesp@fing.edu.uy

Grupo de Ingeniería de Software
Universidad de la República
Uruguay

**“Si he de morir, que me muera de tanto vivir
con la furia de la tempestad, incendiándome el alma al partir
Si he de partir, que me parta la vida un amor
y transforme mis huesos en flor” – La Catalina**

**To my mother, at one month of not being with us**

TSP Symposium 2013: When software really matters
September 16-19, 2013 in Dallas, Texas, USA