Incorporating some PSP practices into introductory programming courses: A case study in Universidad del Quindío.

Sergio Cardona, Universidad del Quindío, Colombia
Rafael Rincón, Universidad Eafit, Colombia
Diego Vallespir, Universidad de la República, Uruguay

1.1 INTRODUCTION

The sustainability of the software industry depends largely on the formation of professionals with high skills and abilities to develop quality software.The incorporation of the appropriate practices for software development improves the capacity and productivity in the Information Technology organizations.Unfortunately, this could mean high investments of time and training for organizations.

We understand that the academy has the compromise and commitment of forming professionals with the self-management and administration skills of their software process that can be defined, measured, and controlled. The academic curricula must consider the skill development and technical capacity for the construction of quality software. In this regard, some universities have used quality oriented process models, and students apply the best software development processes for: management, cost, time,removal, estimation ofsize , management of standards, and prevention of flaws (Cardona & Bermúdez, 2012).

The Personal Software Process (PSP) is a software development process for an individual(Humphrey, 1995).This process supports the Software Engineer for the construction of quality products. The PSP is also a formation complement that aims for quality culture in software development, and in the curricula from some universities, it is offered as an elective course. In classroom experience reports(Bermón, Fernandez, Sanchez, Javier, & Seco, 2009), when the PSP is used in the first programming course, the complexity of its implementation is identified because the students not only learn to program, but also learn the good practices of software development that the PSP proposes.

This article presents the results of a research which applied a learning strategy on an experimental group implementing some PSP practices on a first programming course in the second half of 2012. The introduction of some PSP practices pretended that the students will apply individual techniques for the development of skills in aspects like, planning, time estimation, and management of software flaws.The results showed that the students meaningfully adopted practices associated with time and flaw management.

Initially, the related works along with the conceptual support for the development of this research are presented. Then, the methodology defined for its development is also presented. Following that the learning strategy design; and finally, the results and conclusions.

1.2 RELATED WORKS

Since Watts Humphrey presented the PSP in his book "*A Discipline for Software Engineering*", diverse investigations about the impact that the use of The PSP generates in undergraduate and graduate courses in universities have been carried over(Abrahamsson & Kautz, 2002), (Prechelt & Unger, 2001), (Towhidnejad & Hilburn, 1997), (Hayes W, 1998), (Wesslén, 2000), (Börstler, Carrington,& Hislop,2002), (Runeson, 2003). The PSP has been also used for experimenting in Software Engineering courses(Honig, 2008), (Venkatasubramanian, Roy, & Dasari, 2001). Likewise, there isthe report of learned lessons from the PSP implementation on the academic sector with the software industry support (El Eman, Shostak, & Madhavji, 1996),(Rincón, 2010).Also, there are academic experiences related with TSP(Bayona, Calvo, Gonzalo, & San Feliu, 2008), (Honig, 2008).

The analysis of the related works resumes what is proposed by (Börstler et al., 2002),three primary factors,which influence the teaching of PSP, outstand: the work environment, the coverage level, and the support tools. The work environment refers to the target audience, the course level, and the subject content. The coverage level is associated with the applied PSP practices. The support tools are related to the support means for the registration of every single activity proposed by the PSP. This paper contributes with a new analysis factor associated with or without the application of a learning strategy. Below are the obtained results of the PSP implementation in different universities worldwide.

Table : Academic experiences of the PSP

University	Target students	Level of coverage	PSP supporttools	Learning strategy
Lund(Runeson, 2003)	Undergraduate and graduate	Full PSP[1]	Spreadsheets	N/A
Zagreb (Car, 2003)	Undergraduate	PSP-Lite[2]	local development	N/A
Purdue(Lisack, 2000)	Undergraduate	PSP-Lite	Spreadsheets	N/A
Carlos III(Bermón et al., 2009)	Undergraduate	PSP-Lite	Student Workbook	N/A
Umea	Undergraduate	PSP-Lite	Local development	N/A
Utah	Undergraduate and graduate	Full PSP	Local development	N/A

Based on Table 1, it can be established that every single reported experience has a feature given the context and the formation interests for its students.Having in mind the space limitation for the article, a detailed analysis of every single academic experience in the implementation of the PSP is not done.

1 It refers to the implementation of the entire body of knowledge of PSP.

2 It refers to a simplified or adapted version of PSP.

1.3 METHODOLOGY

The following objectives for the development of this experimental research are defined.

- Analyze the state of the art and the most significant experience results worldwide of the use of the PSP in the academia to identify their impact in the student formation process in Software Engineering and the like, and that these can help as a reference for a theoretical support of the research.
- Design the scenarios, activities, and learning resources that allow, by means of a formative strategy, the appropriation and application of individual practices of the PSP.
- Conduct a pilot test with the Programming course students, in order to verify and assess that the strategy contributes to the development of individual practices of software development of students.

The research was piloted in the classroom. The populations under study were two groups of a first course in Computer Programming of the Systems Engineering undergraduate program at the Universidad Del Quindío. The methodology is shown in Figure 1.

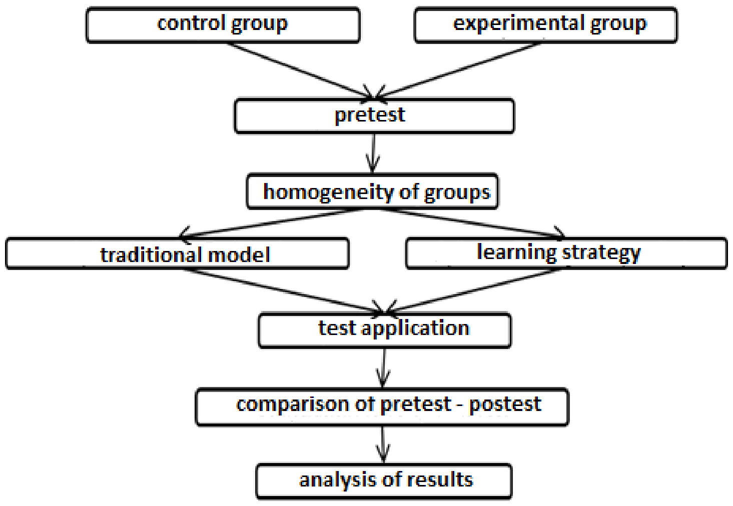

Figure : Methodology for the research

1.3.1 Pre-test

The initial diagnosis applies an instrument with nine questions (Table 2) with options (Never - Sometimes – Always). The questions sought to know the level of adoption of some individual practices for software development in students. The number of students in the control and experimental groups that answered the survey was 31 and 35 respectively.

Table 2: Questions and categories

Number	Question	Category
1	Register the time spent during the programming activity.	Time managemen t
2	Register the interruption time during the programming activity.	
3	Register the flaws that emerge in the making of a programming activity.	Handling

4	Understandthe encoding flaws generated during programming.	and managemen t of flaws
5	Apply some methodology to solve flaws in the codification process.	
6	Take into account encoding standards when programming.	Size product
7	Estimate the number of code lines needed to build a program.	
8	Plan activities to perform a programming job.	Product planning
9	Applies stages of a development process to build a program	

With the pre-test exam applied to both groups, the level of homogeneity in each one of the questions is analyzed, both for the experimental and control groups. To check the homogeneity of the groups, an Analysis of Variance ANOVA (ANalysis Of VAriance) was run, whose response variable is the qualification of the question, and the factor is the group with the control and experimental levels. For each question the assumptions of randomness, the homogeneity of variance, and the normal distribution of residualswere applied. When these assumptions were not met, then the Kruskay and Wallisnon-parametric test was applied. Table 3 shows the results of thestatistical analysis of each one of the questions.

Table 3: Homogeneity analysis per question

Question	p_valor	Variance homogeneity	Shapiro-Wilknormal distribution of residuals	Kruskay& Wallis non-parametric test
1	0,1409	0,140908	4,35409E-15	0,1848
2	0,5739	0,573945	3,54809E-14	0,569942
3	0,2237	0,413037	7,81931E-10	0,162059
4	0,2356	0,0750583	4,44089E-16	0,204958
5	0,6374	0,808796	1,11022E-16	0,712758
6	0,1495	0,149459	1,26715E-10	0,1848
7	0,4727	0,0511165	1,0578E-9	0,451819
8	0,1023	0,617618	3,42777E-7	0,132956
9	0,4686	0,151367	1,7582E-10	0,48251

Based on the results above, it can be stated that both the control and the experimental groups are homogeneous for the nine questions defined in the instrument, and it was decided to continue the research methodology.

1.3.2 Learning strategy

The learning strategy was conducted with 23 students from the experimental group during 10 weeks of the academic semester. Parallel to the development of the subject content, the fundamental concepts of PSP0,PSP0.1, and PSP1 levels were being incorporated. Six programming exercises were proposed, which were solved directly in the laboratory course, under the teachermonitoring. For the PSP 0.1 PSP0 levels,the first fourexercises were solved and the remaining two were for the PSP1 level. The registration of each one of the activities was performed on templates designed for that purpose, and the feedback of the results was done in the following class highlighting the importance of the proposed activities.

1.3.3 Thematic structure of the course

The course structure is defined by thematic units required in a first programming course. The fundamental concepts of the PSP0, PSP0.1 and PSP1 levels were incorporated progressively. Table 4 shows the thematic content and the PSP themes that were given in the course transversally.

Table 4: Thematic content VS PSP themes

Unit	Thematic content	PSP topics
Java Programming Language	Variables, Operators and Expressions Primitive Data Types Objects concepts	● Software Quality Concepts ● Software Development Process ● Current process development
Conditional Programming	Simple decisions (if, if-else) Nested Decisions Multiple decisions (switch)	Personal process reference ● Introduction to PSP ● Introduction to PSP0 ● Time planning
Methods	Methods concepts Methods that return value Methods that do not return value Parameter Passing	Reference personal process ● Time and control management PSP0 ● Time and flaws registration ● Types of flaws standards
Iterative Programming	Counters and Accumulators Cycle conditioned at the end (do-while) and conditioned at the beginning (while, for)	Reference personal process – PSP0.1 ● Size planning and measuring ● Encoding standards
Arrangements	Operations with arrangements dimensional arrangements Management methods	Reference personal process PSP0.1 ● Encoding standards ● Process Improvement Proposal (PIP) Personal Project Management – PSP1

The PSP Themes were oriented only in the experimental group; while in the control group, the traditional thematic contentcourse was developed. To the PSP0 level practices, ateaching guide with the theoretical foundations necessary for learning and implementing the following practices was designed:

- Time registration for the completion of the project.
- Flaw registration and its types.
- Summary of the project plan.
- Standards to document and report the types of flaws.

In the PSP0.1 a guide for students to learn to perform the count of code lines (LOC) of their programs was built, as well as documenting the activities of its development process in order to identify opportunities for improvement in their work. Theelementstakenintoaccount for thislevelwere:

- Definition of a standard for code line counting, and anencoding standard during product construction process.
- Documentation of the Process Improvement Proposal (PIP).

For the PSP1, it was also designed a guide that explained using examples; how the template must be filled out for the test report and the estimate for the size of the product.

1.3.4 Design of the learning strategy

For each one of the thematic units of the course, the learning scenarios which define the necessary theoretical elements, the work methodology, and the activities undertaken by the students were designed. Table 4 shows the description of the Iterative Programming thematic unit, and the same was done for the rest of the course units.

Table 4: Thematic structure of the course

Unit	Methodology	Activities
Iterative Programmin g	The teacher presents the fundamental concepts of PSP, the process script, time control and registration in each phase of the process. He will explain the time log template which details the actual working time and the interruptions. He will explain to students how to perform the estimation of time for their work, and a series of suggestions to manage time when making a programming job.	The student will read articles about the fundamental concepts of PSP0 and PSP 0.1. In each of the programming tasks, the student must use the process script, and the teacher will assignthe exercises 1A, 2A, 3A and 4th, so students develop the proposed programs. For each of the programming tasks, it is required the delivery of the time template. Based on the results delivered by the students, the teacher will conduct a performance analysis of the group works.

For each one of the activities, an evaluation plan based on criteria was defined which takes into account the following aspects:

- Observation of attitudes and skills that students are developing.
- Students' response facing the questions related to the individual development.
- Monitoring the development of practices that the students do in the lab.
- Monitoring to the tasks that students do during their independent work.
- Conducting of individual assessments.

These elements of formative character will have a summative evaluation in a range from 0 to 5.

1.4 RESULTS

In order to determine whether the intervention with the PSP practices in the experimental group was successful;it was verified in the post-test if,in each one of the questions,the ownership of homogeneity with the control group was retained.

1.4.1 Post-test

The results obtained in the post-test show that the property of homogeneity of the groups is preserved for questions 3, 6, 7, 8 and 9, so it can be said that the learning strategy for the categories of product size and product planning did not have a significant impact within the individual practices of software development.

For questions 1, 2, 4 and 5, the obtained results show that the homogeneity property of the groups is not preserved; and therefore, for the categories of time management and flaw management, the learning strategy was successful. For example, Figure 2 shows that for question 1, the experimental group applies more this PSP practice than the control group.

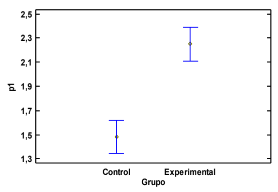

Figure 2: Question 1,Time control forpost-test

1.4.2 Analysis of the results

We compared the answers of the pre-test and post-test of the students intervened to construct a "result" variable.Thus, if the post-test grade is higher than the pre-test one, the variable takes the value of 1. If the grade is less or equal in the post-test, the variable takes the value of 0. If the pretest and post-test graded the answer always with (3), the variable takes the value of 1. Thus, the "result" variable has only two possible values 1 and 0; therefore, it is a discrete variable with Bernoulli distribution and p = 0.5 because it is using the criterion that at least 50% of students will improve from the pre-test to the post-test. The answers with value of 1 were addedand the variable "number of students who improved with the intervention" was obtained, which due to be the sum of variables with Bernoulli distribution corresponds to a variable with binomial distribution with n = 23 (number of students from the experimental group), and the probability p = 0.5 indicates that at least half of the students improved with the intervention strategy. Then, a system of hypotheses that allowed selecting those questions where students improve their practicesarose. For the experiment it is established a probability for error of 4.7% to say that the question in the intervention was successful, which is equivalent to say that the results have a confidence level of 95.3%.

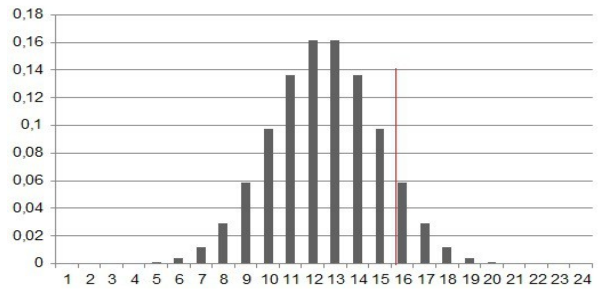

Figure 3: Binomial distribution for post-test

Figure3shows the binomial distribution for the 23 students from the experimental group with p=0,5. Those questions where 16 students or more improve with the intervention are the ones that allow saying that it was successful.

1.4.3 Analysis of the results from the experimental group

The quantitative results of the experimental group in the pre-test and the post-test show a significant improvement in the nine questions applied to students. For example, in question 2 for the pre-test related to the interruptions registrationpractice, 91% of students never apply it, and 9% sometimes. While the same question for the post-test shows that only 13% of students never apply it, 74% sometimes, and 13% always do. Figure 4 shows the frequency of answersfor questions 2 and 3.

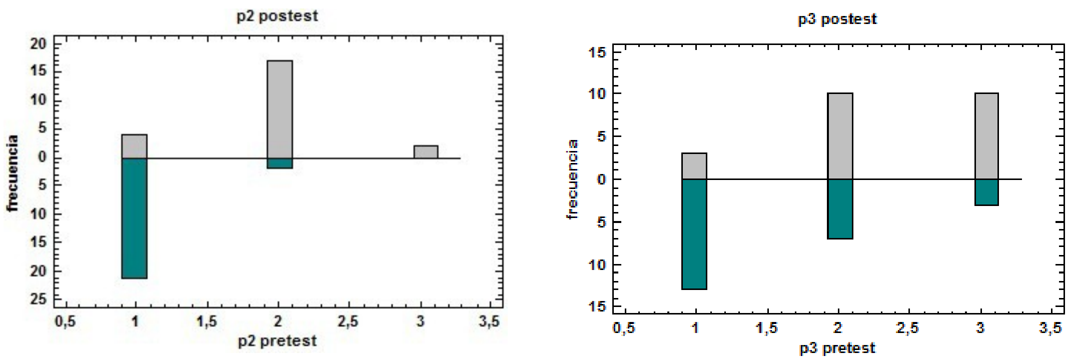

Figure 4: Pre-test and post-test results of the experimental group

For question 4 in the post-test, 52% of students in the pre-test answer that they always manage the flaws introduced during their individual work of software development. In question 5 on the posttest, 74% of students always apply a methodology for the solution of flaws. In both questions, it is evidencedan improvement in the outcomes.

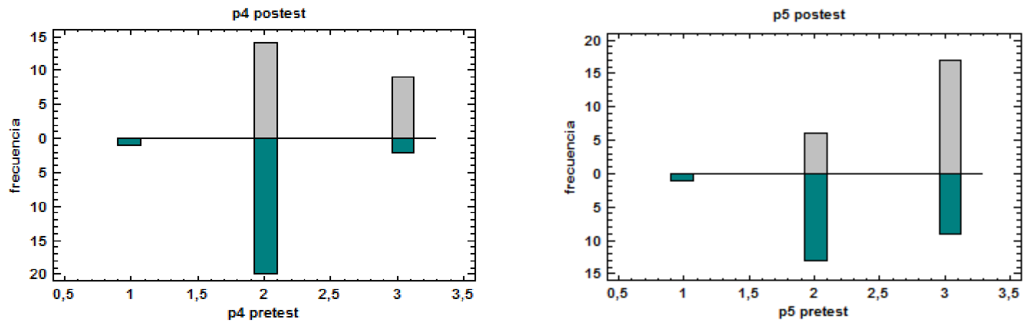

Figure 4: Pre-test and post-test results of the experimental group

Our results show that the experimental group improved on the post-test compared to the pre-test in every question.

1.5 CONCLUSIONS

The development of this work showed a number of challenges associated with the learning of the software process which is associated with the maturity of the students to recognize the value of a discipline applied to the software process (issuethat they have not experienced in early stages yet), and a forced introspection to know how the software is developed individually understanding their development habits and practices to improve them. It was also necessary to consider some theories about teaching strategies, which put in our particular context, involved the incorporation of ideas about how to intervene current practices for students to learn. The most frequent difficulties and mistakes of students were identified, and they were encouraged to reflect high quality in their work.

The academic environment also requires political will and commitment from the academic directors since the teachers, who teach the courses related to PSP practices, require spending a great deal of time to give immediate feedback on the work and exercises of the students, conducting permanent support, and also teaching the topics and concepts related to PSP. This academic strategy becomes complex because teaching courses related to PSP practices requires a greater dedication than in a regular course by the teacher and the student.

Based on the obtained results, we found that the incorporation of some PSP practices in students of the experimentalcoursehave been successful regarding the adoption of the practices associated with time management and registration, and the management and registration of flaws.

1.6 REFERENCES

Abrahamsson, P., & Kautz, K. (2002). Personal Software Process : Classroom Experiences from Finland. *Lecture notes in computer science*, *2349*, 175–185.

Bayona, S., Calvo, J. a., Gonzalo, C., & San Feliu, T. (2008). Teaching Team Software Process in Graduate Courses to Increase Productivity and Improve Software Quality. *32nd Annual IEEE International Computer Software and Applications Conference* (pp. 440–446). Turku: Ieee. doi:10.1109/COMPSAC.2008.135

Bermón, L., Fernandez, A., Sanchez, M., Javier, G.-, & Seco, A. (2009). Experiencias Docentes en la Aplicación del Proceso Software Personal en Primero de Grado de Ingeniería Informática.

Fomento e Innovación con Nuevas Tecnologías en la Docencia de la Ingeniería (pp. 107–114). Vigo.

Börstler, J., Carrington, D., Hislop, G. W., Lisack, S., Olson, K., & Williams, L. (2002). Teaching PSP : Challenges and Lessons Learned. *IEEE Software*, *19*(5), 42–48.

Car, Z. (2003). A method for teaching a software process based on the personal software process. *21 International Association of Science and Technology for Development* (pp. 1–6). Innsbruck.

Cardona, S., & Bermúdez, R. (2012). Ambiente virtual de aprendizaje para la implementación de prácticas de PSP y TPS en un curso de programación de computadores. *IV Congreso Iberoamericano "Soporte del Conocimiento con la Tecnología"* (pp. 409–416). Bucaramanga.

El Eman, K., Shostak, B., & Madhavji, N. (1996). Implementing Concepts from the Personal Software Process in an Industrial Setting Implementing Concepts from the Personal Software Process in an Industrial Setting. *Proceedings of the Fourth International Conference on the Software Process* (pp. 117–131). Brighton.

Honig, W. L. (2008). Teaching Successful "Real-World" Software Engineering to the "Net" Generation: Process and Quality Win! *21st Conference on Software Engineering Education and Training* (pp. 25–32). Charleston: Ieee. doi:10.1109/CSEET.2008.38

Humphrey, W. (1995). *A discipline for software engineering* (p. 789). Addison-Wesley.

Lisack, S. K. (2000). The Personal Software Process in the Classroom : Student Reactions (An Experience Report). *Software Engineering Education and Training* (pp. 169 – 175). Austin.

Prechelt, L., & Unger, B. (2001). An Experiment Measuring the Efects of Personal Software Process (PSP) Training. *IEEE Transactions on Software Engineering* (pp. 465 – 472).

Rincón, R. (2010). *Análisis y capitalización de las experiencias y lecciones aprendidas de la implementación de PSP (Personal Software Process) y TSP (Team Software Process) desde el sector académico a las empresas de software mexicanas. Informe Final Sabático.* (pp. 12–15). Medellín.

Runeson, P. (2003). Using Students as Experiment Subjects – An Analysis on Graduate and Freshmen Student Data. *Proceedings of the 7th International Conference on Empirical Assessment in Software Engineering* (pp. 95–102). Keele.

Towhidnejad, M., & Hilburn, T. (1997). Integrating the Personal Software Process (PSP) across the Undergraduate Curriculum. *Frontiers in Education Conference, 1997. 27th Annual Conference. Teaching and Learning in an Era of Change* (pp. 162–168). Pittsburgh.

Venkatasubramanian, K., Roy, S. B. T., & Dasari, M. V. (2001). Teaching and Using PSP in a Software Engineering course : An Experience Report. *Software Engineering Education and Training Annual Conference*. Chennai.