1-2002

Ventures into Capturing Effort in Programming

Barbara Bernal-Thomas
Southern Polytechnic State University

Briana B. Morrison
University of Nebraska at Omaha, bbmorrison@unomaha.edu

Ventures into Capturing Effort in Programming

Barbara Bernal Thomas [1] and Briana B. Morrison[2]

Abstract

The quest for teaching a method of data collection in programming experiences was marked with successes and failures. We believe that software development curricula must provide students with knowledge and experience related to the practice of data collection, which will measure the effort put into a software project. By recording their past effort in software projects, students can more accurately estimate the amount of effort and time required to complete a future software project. Students can also learn the amount of effort required to develop "correct" software and begin to estimate the amount of time required, per software phase, to fix errors. This paper recounts the educational challenges that were found in the quest to teach a method of gathering data called Personal Software Process (PSP) in the Computer Science curriculum at Southern Polytechnic State University (SPSU). After faculty training, a plan was devised for incorporation of PSP into the curriculum in stages. These stages began with a Pilot experience in the fall of 2000, a total incorporation in the first programming course in the spring of 2001, and incorporation into the second programming course in the fall of 2001. A method for assessment of this new ingredient was outlined.

This paper shows the progressive venture into this quest and explains the findings and conclusion that were made regarding the data collecting method called PSP in our curriculum. Finally, future plans of incorporating alternative methods and the future of PSP elements into the curriculum with the assessment plans are discussed.

Introduction

The undergraduate program in Computer Science at SPSU has a series of required courses called Computer Science I (CS I), Computer Science II (CS II), Data Structures, and Software Engineering (see figure 1). The approach taken at SPSU has been to introduce students to gathering data during programming efforts using Personal Software Process (PSP) in these required programming courses. Which course is best to initiate this record keeping? In the introductory programming courses, students could collect data during a "closed" lab. For the students that did not have the introduction to PSP in their programming courses, the Software Engineering course could introduce the complete PSP in conjunction with the development of a team software engineering project.

[1] Southern Polytechnic State University, School of Computing and Software Engineering, 1100 South Marietta Pkwy, Marietta, Georgia 30060, USA. Email: bthomas@spsu.edu Voice: (770) 528-4283 Fax: (770) 528-5511.

[2] Southern Polytechnic State University, School of Computing and Software Engineering, 1100 South Marietta Pkwy, Marietta, Georgia 30060, USA. Email: bmorriso@spsu.edu Voice: (770) 528-4295 Fax: (770) 528-5511.

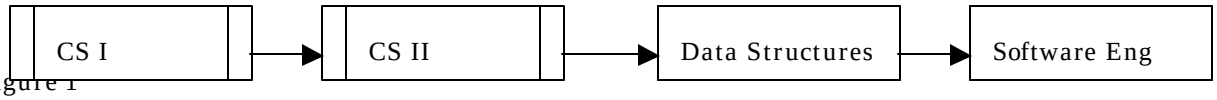

Figure 1

The faculty training started during the summer 1999 where three faculty participated in the Personal Software Process (PSP) and Team Software Process (TSPi) Faculty Workshop held in the University of South Carolina, Columbia, SC. The following two summers (2000 & 2001) these workshops were held at SPSU with some more of SPSU faculty participating.

The initial approach taken was to introduce PSP to students in the first required programming course (CS I) and in the Software Engineering course. In the introductory programming courses, students collected basic data during a "closed" lab while in the Software Engineering course students were exposed and expected to practice the complete PSP metrics gathering in conjunction with the development of a team software engineering project. The first iteration in the Spring of 2001 repeated the process with modifications indicated by the results of the assessment in the pilot. The second iteration was to be repeated in the Fall of 2001, with expansion into the second programming course.

Pilot Experience – Fall 2000

Implementation

Prof. Morrison taught four sections of Computer Science I with the inclusion of an introduction into the PSP forms in the laboratory programming exercises. In order to introduce students to the practice of collecting data about their programming efforts, PSP was introduced in the beginning introductory programming course (CS I) in a new closed lab facility. The updated CS I course started in the Fall of 2001 consisted of two lectures each week and one two hour closed lab, 11 labs in total for the entire semester. For each of the closed labs, the students work in pairs and were usually asked to design the solution to a program before they came to lab. During lab, we discussed the proposed solution and suggest improvements in the design. Then the students implemented and tested the program. Beginning with the fourth closed lab of the term, the students were introduced to the concepts of PSP and asked to record their efforts using an abbreviated PSP data collection sheet. [Grove, 1998] Since the students worked in pairs, one student would record the time data while the second student "drove" and typed in the code. Students were also asked to record their defect data beginning with the fifth lab. All the data was then collected into spreadsheets to allow for analysis. Students were also required to count the size of their programs to determine their productivity rates. [Morrison & Thomas, 2001]

In addition, the students were given a question on their final test to test their ability to record data correctly. They were given a programmer's "scenario" which described (along with times) the development of a program and were asked to complete the PSP data collection sheet.

Prof. Thomas taught two sections of Software Engineering in which each individual student participated in four programming experiences practicing PSP. The objectives to the added programming exercises using PSP is the experience of a process-based approach to developing software that enables students to measure and analyze their personal productivity in developing software. The students used the results and findings from each experience to estimate and predict the following exercise. They learned from their performance variations. Since the students are implementing such a small sample in their programming experience with PSP, the class also contains some familiarization exercises that exemplify and reinforce the PSP metrics, tables, forms, checklists and templates. [Morrison & Thomas 2001]

Results

In the introductory programming course, we had originally asked the students to collect data for both the first program they designed before lab and another program created during lab. This turned out to be too much for the students, as they had trouble keeping the two programs separate and sometimes would combine their data. A mid-semester adjustment was made and they were only required to track the data for the program created during the lab. Because the entire program was developed during lab, the time keeping effort became minimal, but the error tracking became more difficult. The students were more concerned with making the program work than tracking their errors.

As with most manual data collection, accuracy is always a concern. It was observed that many times the students would "guesstimate" their actual time or number of errors, especially when collecting data outside of the lab. By limiting the collection of data to only during lab, their time recording became much more accurate.

After beginning to analyze the data collected in the introductory programming course, we noticed some discrepancies in what we had expected. The students reported they were finding many errors during design and coding and few errors during testing phase. After grading the question on the final exam where the students were required to complete a PSP data sheet from a given scenario, the reasons behind this discrepancy became clear. The students were recording where they fixed the error, not when they found the error.

The Software Engineering students reported an overall understanding of the PSP methodology. They were not entirely convinced of that methodology, but did agree to the benefit of recording metrics while in the phases of software engineering. The need for an accurate prediction and estimation method for future software projects was thought to be a key ingredient for success.

First Iteration – Spring 2001

Implementation

Changes made to the CS I course in Spring of 2001 were as a result of the problems found during Pilot implementation. Specifically, the number of labs that PSP was required was reduced. The students were asked to record PSP data for 5 out of 11 labs. Each lab that they were required to record data for only involved the design and implementation of a program during lab. The recording sheet was changed to only record defects during the coding, review, compile, and test phases (planning, design, review, and wrap-up were eliminated). [Grove, 1998] Lectures included additional emphasis on program development phases and how to accurately record data. The students were tested earlier in the semester on their recording techniques in an effort to correct discrepancies. Lastly, questions were added to the course survey to ask information specifically related to PSP in an effort to see if the students are actually using their PSP data to estimate workload.

The objectives for the Software Engineering course were inline with the pilot experience, adding individual programming exercises using PSP to the curriculum of the course. Specifically, the students developed and implemented four programming exercises using the PSP forms. The last experience included separate hand-in of the individual design, formal design review done by another student in the other class, and return of reviewed design. The student revise their design as needed by the reviewer, coded the revised design, and handed the code without compiling. Thomas exchanged the code between the classes and students then performed the formal code review with a checklist. The students then received their code back with defect reports, did the corrections and then finished the programming experience. The experience of PSP data gathering in the programming exercises enabled students to measure and analyze their personal productivity in developing software. The students used the results

and findings from each of the PSP metrics to estimate and predict the following programming exercise. They learn from their personal performance variations. Since the students were implementing such a small sample to speed their education of PSP, some familiarization exercises that exemplify and reinforce the PSP tables, forms, checklists and templates were done. [Humphrey, 1995]

Results

For the Spring 2001, the CS I students greatly appreciated the reduction in the number of labs that required PSP (found through survey results). However, we found that the students were still "guestimating" their actual time and defect numbers. During lab the students could either solve the programming problem or record errors, but they were unable to do both. After further questioning of students, they indicated that could not "split their time" or their concentration. When they became so engrossed in solving the programming problem, they totally forgot about capturing their numbers. Or if one partner was diligently trying to record the time and errors, they were unable to help in solving the problem. When the students did diligently record the data, it was much more accurate than the first group. We feel this was due to the increased lecture time on the phases of software and the test questions earlier in the semester.

Another result that came from the second iteration was that the CS I students did not feel that they learned anything from PSP, and that it was in no way useful or relevant to the class. The labs that required PSP were consistently rated lower in terms of enjoyment then those without PSP. One student noted, "…don't try to pack too much into one lab mainly if it is the first time doing something (PSP)." The students regularly said that the PSP took too much time for no results. Although we discussed in class (with class results shown) what the numbers represented, it had no impact on the students.

In the Software Engineering course the students felt that the PSP needed to be covered in a programming course and not in this team developing software course. The additional time to practice the metric gathering was thought to be a big investment. The two sections of this class were a day section and a night section. Each section had different views on the "venture into capturing effort in programming". The day section did not see the benefit in the investment of time and effort to understand the PSP, while the night section's opinion was the opposite, they knew the benefit of accurately estimating and prediction for software projects. But both classes reported that they would rather not have this component in this class because it took so much time and effort away from the team software engineering project that was the purpose of the course.

Second Iteration – Fall 2001

Implementation

After reviewing the problems associated with implementing PSP in the CS I introductory programming course during the pilot and first implementation, it has been removed from the curriculum of this course, with one exception. Because it appeared that the students "only had so much brain power" available, we felt it was more important that the student understand the basics of programming before trying to capture their own personal process. It is much like teaching someone to ride a bike. You don't time them until they learn the mechanics. Our programming students have not "mastered the mechanics" until the Data Structures, or third, programming course. Therefore it was also decided to remove PSP from the second programming course (CS II). We have decided to wait until after the students become comfortable with phases of programming and have completed learning the language elements to introduce the PSP process. We are hopeful that by then the students will use and appreciate the data they will gather.

The one exception to this is a special program where we are retraining existing workers to become software engineers. These student workers attend class Monday through Friday from 8 a.m. until noon and work on assignments and have open lab time from 1 p.m. until 5 p.m. In this compressed environment, PSP was a qualified success. We believe these students adapted to process more readily due to the fact that students were already professionals within the industry and understood the reasons behind tracking work effort. Because the course was taught in a compressed "captured audience" mode and there were no tardies, no skipping students, the students felt more comfortable with the process. This special program has four separate iterations for four separate groups staggered for two years. The CS I and CS II was taught be the same professor. During the first implementation (2000) of this special program, she had not participated in the PSP training, so the PSP was very light. By the second iteration the summer training was done and the inclusion of the PSP was stronger and successful. The PSP is included presently in both the CS I and CS II classes for this special program.

What We Learned

Benefits and Disadvantages

One of the reasons to introduce PSP in the introductory programming course is to help the students develop a better feel for the different stages in the life-cycle of a program. At the end of the course, students were able to identify correctly the life-cycle phases and the tasks involved in each phase.

One of the benefits of introducing PSP during the introductory programming course was to encourage students to realize the amount of time required to develop programs. Because the students are just beginning to learn how to program, they are not aware of how many problems they might encounter when learning and using a programming constructs. Most students became too concerned with "making the program work", rather than tracking data. The other problem was motivation. Because the students saw no relevance to what they were doing, they were not motivated to track their data. Another disadvantage of collecting PSP data is the voluminous data that is collected. It is very labor intensive to both record the data on the students' part, and to enter the data into a spreadsheet for analysis on the instructor's part. Students commented that the system should be able to record automatically how long they were in the compile mode.

Future Plans

Our plans are now to move the introduction of PSP to the third programming course, Data Structures (in the fall of 2002), along with a closed lab element. The Data Structures course was chosen to allow students to plan time to track errors and develop their own personal checklist of errors. Students would use this personal data in the follow-on Software Engineering course for team project planning. Also, by the time the students reach Data Structures, they have already mastered the fundamentals of programming and completely understand the software life-cycle. Students have spent enough long nights trying to finish programs that some amount of data to allow them to estimate the amount of time necessary to finish a program would actually provide motivation for them to capture the data as well.

We are also working on an on-line tool that would allow the students to record their PSP on-line and have the data deposited into spreadsheets automatically to remove a great deal of the time-consuming nature of entering data (both for the students and the professors!).

References

Grove, Ralph F. (1998) "Using the Personal Software Process to Motivate Good Programming Practices," *SIGCSE Bulletin Conference Proceedings of the 3rd Annual Conference on Integrating Technology into Computer Science Education ITiCSE'98,* ACM Press, New York, NY.

Humphrey, Watts S. (1995) *A Discipline for Software Engineering,* Addison Wesley, Reading, Massachusetts.

Humphrey, Watts S. (2000) *Introduction to the Team Software Process*, Addison Wesley, Reading, Massachusetts.

Morrison, B. and Thomas, B. B. (April 2001) "The Educational Quest of Capturing Effort in Programming", 2001 ASEE Southeast Section Conference.

Barbara Bernal Thomas

Barbara Bernal Thomas is a full professor in the School of Computing and Software Engineering at Southern Polytechnic State University (SPSU) for the last sixteen years. The areas of Software Engineering, User-Centered Design and Computer Graphics & Multimedia are the focus endeavors. She is a co-founder of the SPSU Usability Research Lab and is directly involved in corporate-sponsor ULAB projects. She has given numerous papers, tutorials and presentations locally and internationally on User-Centered Design, Usability and Software Engineering topics. Barbara is involved with computer educational support for local businesses in the Atlanta area. She does specialized software development as a consultant.

Briana B. Morrison

Briana B. Morrison is an assistant professor in the School of Computing and Software Engineering at Southern Polytechnic State University. She holds an M. S. in Computer Science from SPSU and a B. S. in Engineering in the program of Computer Engineering from Tulane University. Prof. Morrison has seven years of industrial experience at IBM in the role of Staff Programmer, all of which contribute to her expertise in software development. Her research areas include introductory programming, effectiveness of closed labs, and object oriented methodology in programming.