Chapter 1

Case Study

To investigate the issue of data quality in PSP, Dr. Johnson taught a modified version of the PSP curriculum over the course of a semester in 1996. Ten students participated in this course, completing nine projects each. (A final project for one student had so many incomplete values that I did not include it in the study.) At the end of the course I collected these projects and then examined them to uncover any errors the students may have made in filling out the PSP forms.

1.1 The Modified PSP Curriculum

In order to produce data of the highest possible quality, Dr. Johnson made some changes to the standard PSP curriculum before teaching the course:

Increased process repetition. In the standard PSP curriculum, students are assigned ten programs during the course (in addition to several midterm and final reports). Over the course of these ten programs, students practice seven different PSP processes, which means that the development process used by the students changes for seven out of ten programs. After his initial experience teaching the PSP, Dr. Johnson came to believe that the high overhead of this almost constant "process acquisition" led to data errors and had a significant impact upon the overall data values. To ameliorate this problem, the modified PSP curriculum included only five PSP processes, enabling students to practice most processes at least twice before moving on. The modified curriculum also included only nine programs instead of ten, providing additional time in each assignment for data

collection and analysis.

Increased process description. In his initial experience teaching the PSP, Dr. Johnson also found that students had a great deal of trouble learning to do size and time estimation correctly. For example, PSP time estimation requires choosing between three alternative methods for estimation depending upon the types of correlations that exist in the historical data from prior programs. To help resolve this and other similar problems, he added four additional worksheets to the standard PSP form set: (1) a Time Estimating Worksheet to provide a guide through the various methods of time estimation; (2) a Conceptual Design Worksheet to help in developing class names, method names, method parameters, and method return values; (3) an Object Size Category Worksheet to help in size estimation; and (4) a Size Estimating Template Appendix to provide a place to record planned and actual size for prior projects (this proved invaluable to me later on - it allowed me to determine whether errors in size and time estimation occurred during data transfer between projects or in the actual calculations).

Technical reviews. At the completion of each project, students divided into pairs and carried out an in-class technical review of each other's work. A two page checklist facilitated this process. It included such questions as, "Did the author follow the PSP Development Phases correctly?" and, "Is the Projected LOC calculated correctly?" A second "Technical Review Defect Recording Log" form included columns for number, document, severity, location, and description. The review took about 60 minutes to complete. The students then submitted the technical review forms with the completed projects. Dr. Johnson reviewed the projects a second time for grading purposes, using the Technical Review Defect Recording Log to record any additional errors.

Tool support. Finally, Dr. Johnson provided four spreadsheets to support records of planned and actual data values. In addition, students had access to tools to count lines of code for Java programs and to compare two versions of a Java program and report lines of code added and deleted. In the textbook PSP curriculum, a LOC counting tool

is assigned as one of the programs. The modified curriculum included completely new program assignments more suited to the Java language.

1.2 Data Entry

At the end of the course, Dr. Johnson gave me the PSP forms for all 90 projects, as well as all the technical review checklists and the instructor grading sheets for each project. At that time I designed and implemented a database system to store all of the PSP data from this course and to subject it to further data quality analysis (See Chapter 5). Then I entered each project into the PSP tool; recording primary data such as time, defects, and program size. I then compared the values calculated by the tool with the corresponding values calculated by the students. Each time an error was found, I reviewed the record sheets from the technical review process and the grading sheets used by the instructor to determine which reviewer first identified the error.

I did not check the Task Planning Template, the Schedule Planning Template, or the Object Size Category Worksheet for errors.

After entering the first few projects, it became harder and harder to find errors by simply comparing a value calculated by the student with one calculated by the PSP tool. This was because it was possible for students to do calculations correctly, but with incorrect data from prior projects. This produced values that differed from the ones generated by the PSP tool, but which could not be counted as errors (see section 1.2.1). This meant that I often had to hand-calculate certain values such as *Total LOC (T), to date* using the paper PSP forms for the current and previous project. For the more complicated calculations, such as linear regression for time estimation, I modified the PSP tool to allow me to update the inputs. The values defaulted to the database values stored in the tool, and then I overrode ones that varied from the ones actually used by the student.

After getting about halfway through project 4, I also realized that some of the pri-

mary data was showing inconsistencies. For example, there were overlapping time log entries and defect logs showing more fix time per phase than the time log showed total for the phase. Since it would be interesting to see any possible clues about a collection phase problem, I went back to the beginning and examined all the primary data for various problems. I recorded errors found under the general type of “Impossible Values”, and continued to do inconsistency checks as I entered the rest of the projects.

Whenever I found an error, I recorded the following information in another database:

- A code identifying the student responsible for the error.
- A code identifying the person who first identified the error; the instructor, another student, or myself.
- The number of the assignment in which the error occurred.
- The programming language used for the assignment. (As it turned out, all projects were done using Java.)
- The PSP process that the student was using when the error was injected.
- The PSP phase in which the student was working when the error was injected.
- A code identifying the general error type. For example:

  BF Blank Field
  CI Calculation done incorrectly

- A code identifying the specific error. For example:

  BLC Base LOC, actual: incorrect
  BC Base LOC, plan: different from Size Estimating Template

- A code identifying the error severity. For example:

0 Error has no impact on PSP data
1 Results in a single bad value, single form

- The age of the error. This represents the number of assignments since the introduction of the data field or PSP operation in which the error occurred.
- The incorrect value (where applicable).
- The value that should have been used (where applicable).

1.2.1 Error Counting Method

Upon finding an incorrect value, I recorded an error for the field, but from that point on I treated the incorrect data value as correct. For example, consider the sample Time Recording Log shown in Table 1.1.

Table 1.1: Example of Time Recording Log Error

Time Recording Log						
Date	Start	Stop	Interruption Time	Delta Time	Phase	Comments
12/01/97	10:40	11:20	0	30	plan	total minutes should be 40
12/01/97	11:20	11:30	0	10	design	
12/01/97	12:30	12:50	0	20	code	
12/01/97	12:50	12:55	0	5	compile	
12/02/97	09:00	09:15	0	15	test	
12/02/97	09:15	09:25	0	10	postmortem	

Since it appears the user incorrectly subtracted *Stop* from *Start* when calculating the number of minutes spent in planning, an error would be recorded for the *Delta Time* field. But when looking at *Total Actual Minutes* (the sum of all the *Delta Time* fields in the Time Recording Log) on the Project Plan Summary form, 90 minutes would be treated as the

correct value, even though the time spent in planning was actually 40 minutes, making the true value of *Total Actual Minutes* 100 minutes.

I do feel confident that every error that I recorded exists in the student data - I checked and rechecked, sometimes five or six times, to be absolutely sure. On the other hand, I don't believe that I uncovered all or even most of the errors present. While the PSP tool did enable me to determine the correctness or incorrectness of values generated during the analysis stage, it provided only limited insight into collection stage errors. For example, in a Time Recording Log, it was possible to check the *Delta Time* computation, but not the accuracy of *Date, Start, Stop*, or *Interruption Time*. Of course, the tool could not, in general, detect the absence of entries for work that was done but not recorded. Two other areas that created similar problems were the Defect Recording Log and the measured and counted *Program Size* fields for the Project Plan Summary.

1.3 Data Correction

It could be the case that although users of PSP make many errors, these errors are only "noise" and do not make a significant impact upon the trends and conclusions reached from the method. I attempted to look into this more closely by computing the measures of yield, defect density, and so forth for individual data and the aggregate results twice: once using the original data supplied by the students, and once using correct versions of the data produced by the PSP tool.

To do this, I copied the original database (which now contained all PSP values for the 89 complete projects) and copied it. Using the second database, I then went again through each phase of each project, attempting to correct the primary data and then re-running the calculation steps.

As I attempted to correct the students' data, it soon became clear that their errors fell into two classes. Some errors I could easily correct with reasonable confidence, such as

a mismatch between the number of defects entered in the Defect Recording Log and the total calculated for the Project Plan Summary. But in other cases, such as a blank *Phase Injected* for a defect, it was impossible to determine a correct value.

This can be illustrated with the Time Recording Log example used in Table 1.1 to demonstrate the error counting method. It is clear that there is an error in the first entry, and it seems obvious that it occurred when calculating *Delta Time*. However, we really can't tell for sure which field was entered incorrectly. Should *Start* be 10:50, or should *Stop* be 11:10, or was there a ten-minute interruption that was not recorded, or was *Delta Time* actually 40 minutes but calculated incorrectly as 30 minutes?

1.3.1 Correction Rules

Despite the obvious impossibility of doing a perfect job in correcting the student data, I formulated a set of rules that could be consistently applied in attempting to make corrections. Underlying all of these rules is basic the assumption that even though primary data may be faulty, it is probably more trustworthy than calculations performed upon it. So, for the Time Recording Log example above, the *Start*, *Stop*, and *Interruption Time* values would be considered correct, and the *Delta Time* value would be changed to 40 minutes.

These are the rules I used in data correction (some were only used once, or a few times):

- **Errors in Time Recording Log entries:** Assume that the start/stop/interruption times are correct and that the delta time is wrong, unless two Time Recording Log entries overlap. In that case, use the preceding and following entries and the delta time for the current entry to formulate plausible start/stop times. Generally this will mean starting the second entry where the first one stops.
- **Missing Time Recording Log entries:** If a Time Recording Log is missing an entry for an entire phase, but the Project Plan Summary form contains a value for the phase

under *Time in Phase (min.), Actual*, formulate an appropriate Time Recording Log entry with fabricated date and time values.

- **Conflicts between Defect Recording Log and Project Plan Summary:** Assume that the number of defects and the phases recorded in the Defect Recording Log are correct and that the discrepancy occurred as a result of incorrectly adding up the numbers of defects injected/fixed per phase and/or incorrectly transferring these totals to the Project Plan Summary form.

- **Conflicts between Defect Recording Log and Time Recording Log:** If, for the Defect Recording Log, the total of all fix times for defects removed in a certain phase is more than the time recorded for that phase in the Time Recording Log, insert a Time Recording Log entry with start and stop times that, combined with existing Time Recording Log entries for the phase, will produce a delta time of the total fix times plus one minute for each defect. This will represent the minimum amount of time required to find and remove defects.

- **Post Mortem phase used in Defect Recording Log:** If the post mortem phase is used on a Defect Recording Log entry for *Phase Removed*, increment the count in the *Defects Removed, After Development, Actual* field on the Project Plan Summary form once for each such defect.

- **Blank Injection Phase for Defect Recording Log:** If the *Inject* field is blank for a Defect Recording Log entry, but the *Fix Defect* field contains a phase name instead of a defect number, use the phase name to fill in the *Inject* field.

- **Blank** *Time in Phase (min.), Plan* **field on Project Plan Summary form:** Use the value for *Time in Phase (min.), Actual* for the same phase.

- **Conflicts in** *Program Size (LOC)* **fields on Project Plan Summary form:** Assume that *Base, Deleted, Modified, Added, and Reused* are correct and that errors are the

result of incorrect calculations. Note: this is not a truly satisfactory assumption because *Total LOC, Actual* should be the result of a measurement rather than a calculation and should therefore be relied upon. However, given correct values for *Base, Deleted, Modified, Added, and Reused*; *Total LOC* can be calculated, whereas it is impossible to even guess at the correct values for the other fields. Unfortunately errors in the *Program Size (LOC)* fields were some of the most common errors. Combined with the importance of these fields in both size and time estimation and my inability to provide adequate corrections, estimates made with the ”corrected” data were undoubtedly severely affected.

1.4 Data Comparison

After I partially corrected the project data according to the rule set, I investigated which values to compare to best reveal the effects of errors. Projects 8 and 9 had the most fields to compare since they were completed using PSP2, and provided the best opportunities for observing the cumulative effect of errors made in earlier projects. Project 9 was the best project for comparison because students had had the most practice in PSP by the time this project was completed and because it provided more time for cumulative effects to exhibit their true characteristics. Unfortunately, one student did not fully complete this assignment, resulting in fewer data points for the final project.

One of the more interesting areas for comparison would have been size and time estimation. This was not possible due to the difficulties in adequately correcting the *Program Size (LOC)* fields. Instead, I selected a few fields from each of the other major sections of the Project Plan Summary, including some fields that represented fairly simple calculations but included to date values from all nine projects, and other fields that were more local to the current project but were the result of more difficult operations.