Chapter 1

Results

1.1 Summary Results

Despite modifications to the PSP curriculum to increase data quality, my analyses yielded 1539 data errors. In this chapter I will report on the types of errors found, their severity, their age, the manner in which they were detected, collection stage errors, and error impact.

1.1.1 Error Types

I found that the errors naturally fell into one of seven general types. They are listed below in descending order of frequency, and include the number of errors found of that type and the percentage of all errors represented by this type. See also Figure 1.1.

Calculation Error. (705 errors, 46%). This error type applied to data fields whose values were derived using any sort of calculation from addition to linear regression. If the calculation was not done correctly, an error was counted. This type was not used for values that were incorrect because other fields used in the calculation contained bad numbers.

Blank Field. (275 errors, 18%). This error type was used when a data field required to contain a value, such as the *Start* field in a Time Recording Log entry, was left blank. This type was not used in fields where a value was optional, such as comment fields.

Inter-Project Transfer Error. (212 errors, 14%). This error type was used for incorrect values in fields that involved data from a prior project. Typically these fields were "to date" fields that involved adding a to date value from a prior project with a similar value

in the current project. Unfortunately, it was often impossible to determine in these cases if the error arose from bringing forward a bad number, or incorrectly adding two good numbers, or bringing forward the correct number and correctly adding it to the wrong number from the current form. However, in two important areas, time and size estimation, the forms were modified so that students were required to fill in the prior values to be used in the estimation calculations. In these cases it was obvious when incorrect values originated in the transfer.

Entry Error. (142 errors, 9%). This error type applied when the student clearly did not understand the purpose of a field or used an incorrect method in selecting data. Examples include filling in the *Fix Defect* field in the Defect Recording Log with a phase name, or having the *Defects Injected, To Date* values in the Project Plan Summary originate from a different project than the *Program Size (LOC), To Date* values.

Intra-Project Transfer Error. (99 errors, 6%). This error type is similar to the error type involving incorrect transfer of data between projects, except that it applied to values being transferred from one form to another within the current project. For example, filling in 172 for *Estimated New and Changed LOC* on the Size Estimating Template, but using 290 for *Total New and Changed, Plan* on the Project Plan Summary.

Impossible Values. (90 errors, 6%). This error type indicates that two values were mutually exclusive. Examples of this error type include overlapping time log entries, defect fix times for a phase adding up to more time than the time log entries for the phase, or phases occurring in the Defect Recording Log in a different order than those in the Time Recording Log.

Sequence Error. (16 errors, 1%). This error type was used when the Time Recording Log showed a student moving back and forth between phases such as Compile and Test instead of sequentially moving through the phases appropriate for the process.

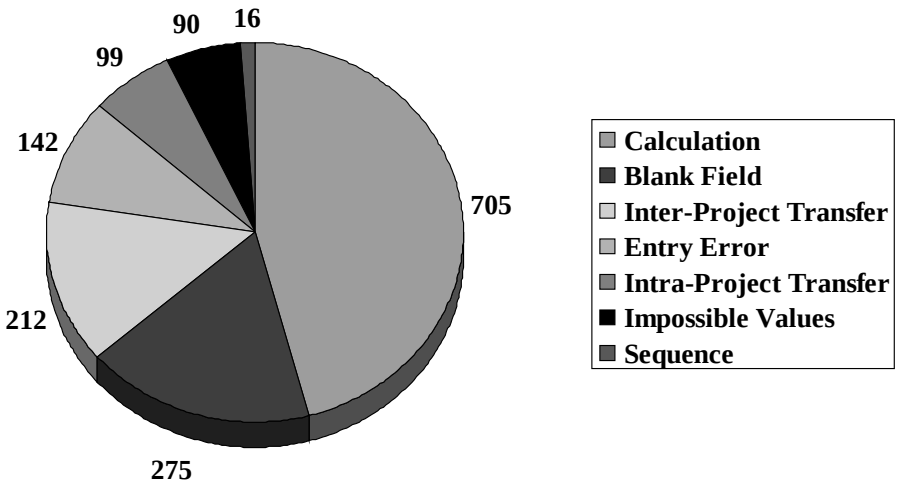

Figure 1.1: *Errors by Type*

1.1.2 Error Severity

Some PSP data errors have relatively little "ripple effect" upon other data values, while others can have an enormous impact. To gain insight into the distribution of the ripple effect, I classified the errors into one of five "severity" levels. These levels are presented below in increasing order of ripple effect. As before, I am including the total number of errors found for a given severity level and its percentage of the total. See also Figure 1.2.

Error has no impact on PSP data. (104 errors, 7%). This level included errors such as missing header data, incorrect dates in the time recording log, and filling in fields for a more advanced process.

Results in a single bad value, single form. (674 errors, 44%). This level was used if a significant field which affected no other fields, such as *LOC/Hour, Actual*, was blank or incorrect.

Results in multiple bad values, single form. (197 errors, 13%). This level indicates when an incorrect or blank value was used in the calculation of values for one or more other fields on the same form, but when none of these other values were used beyond the current

form. For example, in PSP1 on the Size Estimating Template, incorrectly calculating a prediction interval. This results in a bad prediction interval and a bad prediction range, but these values are not used anywhere else in the process.

Results in multiple bad values, multiple forms, single project. (41 errors, 3%). This level indicates when an incorrect or blank value was used to determine the values for one or more other fields on one or more different forms in the same project, but when none of these other values were used beyond the current project. For example, in PSP1, on the Size Estimating Template, calculating an incorrect value for *Estimated Total New Reused (T)*. This results in an incorrect value for *Total New Reused, Plan* on the Project Plan Summary form, but this value is not referenced by future projects.

Results in multiple bad values, multiple forms, multiple projects. (523 errors, 34%). This level was used if an incorrect or blank value affected future projects. For example, when *Defects Injected, Planning, Actual* on the Project Plan Summary does not match the number of defects entered for the planning phase in the Defect Recording Log.

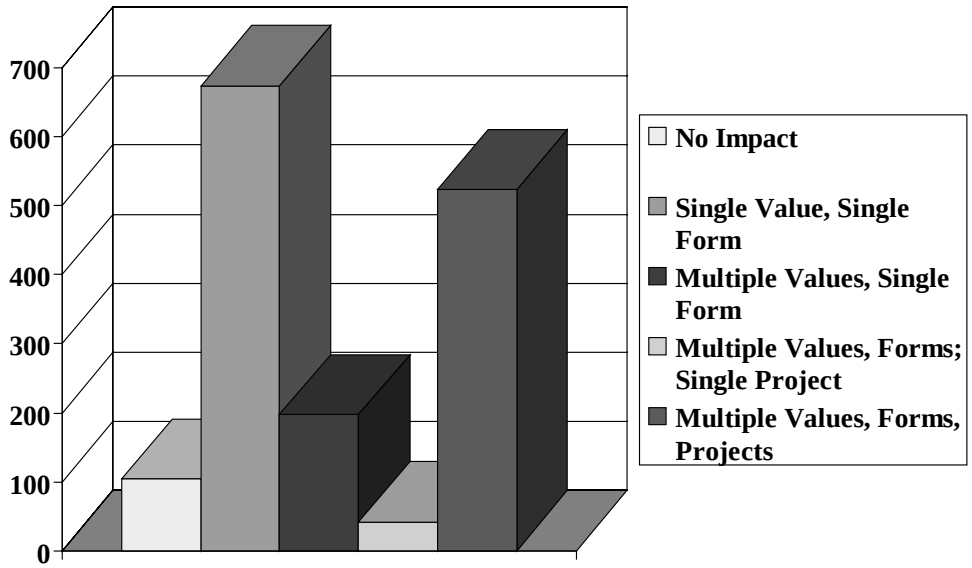

Figure 1.2: *Errors by Severity Level*

1.1.3 Age of Errors

In any learning situation, a certain number of errors are to be expected. I hypothesized that perhaps the errors I discovered were simply a natural by-product of the learning process, and would "go away" as students gained experience with the various techniques in the PSP.

To evaluate this hypothesis, I calculated the "age" of errors – in other words, the number of projects since the introduction of the data field in which the error could be observed. If the errors were simply a by-product of the learning process, then one could expect a low average "age" for errors: people might make an error in a field initially, but then stop making the error after gaining more experience with the data field in question.

For example, the calculation of delta times for the Time Recording Log entries was introduced in the first project. If a student made an error in this field during the first project I recorded a error with an age of zero. If a similar error was made during the second project I recorded an error with the age of one. By the ninth project this type of error would have an age of eight.

Figure 1.3: Average Error Age by Project - All Errors

Project #	PSP Process	# of Errors	Average Age
1	PSP0a	51	0.00
2	PSP0.1a	59	0.73
3	PSP0.1a	63	1.76
4	PSP1a	150	1.27
5	PSP1a	165	2.27
6	PSP1a	186	3.30
7	PSP1.1a	160	3.26
8	PSP2a	351	3.04
9	PSP2a	354	3.84

I first analyzed the errors to determine the average error age in each project. Figure 1.3

shows the average age for all errors in each project.

I then filtered out the 309 errors with an age of zero. This eliminated errors that could have been the result of students being introduced to new fields and/or PSP operations for the first time. Figure 1.4 shows the resulting data.

Figure 1.4: Average Error Age by Project Where Error Age is Greater Than Zero

Project #	PSP Process	# of Errors	Average Age
1	PSP0a	0	NA
2	PSP0.1a	43	1.00
3	PSP0.1a	63	1.76
4	PSP1a	70	2.71
5	PSP1a	165	2.27
6	PSP1a	186	3.30
7	PSP1.1a	135	3.86
8	PSP2a	214	4.99
9	PSP2a	354	3.84

When combining the 1539 errors from all projects, the average error age was 2.78 projects. When only the 1230 errors with an age greater than zero were included, the average error age rose to 3.48 projects.

1.1.4 Error Detection Methods

In this case study, there were three ways an error could be detected: by another student during technical review, by the instructor during the grading/evaluation process, or through the use of the PSP data entry tool. As shown in Figure 1.5, students were made aware of about 5% of the mistakes in their completed projects during the course of the class.

Figure 1.5: Errors by Detection Method

Detection Method	#	%
Grading/Evaluation (instructor)	32	2.08
Technical Review (students)	40	2.60
PSP Tool	1467	95.32

1.2 Analysis Stage Errors

Our two stage model of PSP data quality indicates that errors can be introduced during either collection or analysis. Most of the errors that I found occured during PSP analysis activities, with 700 errors occurring in the Plan phase and 561 errors in the Postmortem phase. Some of the errors occurring in other phases, such as errors in *Delta Time* calculations, were also analysis errors.

1.2.1 The Most Severe Errors

34% of errors found were of the most serious type - persistent errors. These were the errors resulting in multiple bad values on multiple forms for multiple projects. An error of this type not only causes incorrect values in the current project, but may still be causing flawed results ten projects later, even if all subsequent calculations are done correctly. Figure 1.6 shows the four most common errors of this type.

There were two main ways that the error in transferring time estimation data appeared to occur: incorrectly transferring the value from the correct field, or accidentally transferring the correct value from an incorrect field. Specifically, many times instead of transferring *Total New and Changed (N)* (Plan or Actual), students transferred *Total LOC (T)*. This could easily occur because the Project Plan Summary form has over 90 fields even at the level of PSP1, and on the form the two values are vertically adjacent. It would be

Figure 1.6: Most Frequently Occurring Persistent Errors

Description	#
Time Estimation: historical data not transferred correctly	61
Size Estimation: historical data not transferred correctly	56
Time Log: delta time incorrect	48
Project Plan Summary: Total LOC, actual, not equal to B-D+A+R	45

particularly easy to make this mistake with the Actual values because the fields are separated by one column of data from the labels. Additionally, it appeared that students made spreadsheets or ”cheatsheets” of these fields to avoid thumbing through the entire stack of completed projects every time a time or size estimation was needed for a new project. I infer this because the same incorrect value for a particular project would be transferred for time and/or size estimation in every subsequent project.

Similar factors surrounded the error in transferring data for size estimation. These transfer errors were not insignificant. Over the 56 errors resulting from incorrect transfer of data used for size estimation, the sum of the errors was 7753 LOC, with an average error of 138.4 LOC. The sum of the LOC as they should have been transferred was 10,255, with an average of 183 LOC per field. Thus, the average incorrectly transferred number was in error by an amount equaling 75.6% of the number that should have been transferred.

The 48 errors in calculating *Delta Time* in the Time Recording Log were also notable in several respects. First, the errors were not insignificant. The average mistake was 37.8 minutes, which was an average of 39.9 percent of the correct value. Second, 34 (71%) were in error by amounts that indicated small errors in simple arithmetic, as seen in Table 1.7. Third, the distribution of this error across projects is as shown in Table 1.8. Despite nine projects worth of experience, this error never ”went away”. However it did appear to occur less frequently after Project 6. Interestingly, the assignment for this project was a Time Recording Log applet, which at least some students seem to have used for subsequent

projects.

Figure 1.7: Top Four Delta Time Errors

Size of Error in Minutes	Number of Occurrences
60	16
10	9
5	5
120	4

Figure 1.8: Delta Time Errors by Project

Description	Errors	Time Log Entries	% in Error
Project 1	7	84	8.33
Project 2	2	88	2.27
Project 3	8	92	8.70
Project 4	8	108	7.41
Project 5	2	102	1.96
Project 6	9	121	7.44
Project 7	2	77	2.60
Project 8	5	122	4.10
Project 9	5	105	4.76

1.3 Collection Stage Errors

Analysis stage errors were relatively easy to find and correct. However, the accuracy of recorded process measures from the collection stage was much more difficult to examine because the time of collection had already passed and, unlike the analysis operations, was impossible to reproduce. However, I found both direct and indirect evidence for collection errors during the case study.

1.3.1 Direct Collection Error Evidence

Direct evidence of collection problems appeared in the 90 errors classified as “Impossible Values”. I grouped these errors into three major subtypes.

Internal Time Log Conflicts. There were five time logs with overlapping entries, indicating some sort of problem with accurately collecting time-related data.

Internal Defect Log Conflicts. 51 errors showed problems with correctly collecting defect data. 48 of these errors were Defect Recording Log entries showing defects injected during the Compile and Test phases, but not as a result of correcting other defects found during Compile or Test.

Discrepancies Between Time and Defect Logs. In 22 cases, Defect Recording Log entries were entered with dates that did not match any Time Recording Log entries for the given date. For example, a defect would be recorded as injected during the Code phase on a Wednesday, but the time log would show that all coding had been completed by Monday and that the project was in the Test phase on Wednesday. For 10 projects, the total *Fix Time* for defects removed during a particular phase added up to more time than was recorded for that phase in the Time Recording Log. Finally, in two cases, the Defect Recording Log showed a different phase order than the Time Recording Log.

1.3.2 Indirect Collection Error Evidence

Besides the recorded errors, there were other indicators that collection problems had occurred. Some Time Recording Logs showed a suspicious number of even-hour (e.g. 6:00 to 7:00, 10:00 to 12:00) entries. Others showed long stretches of consecutive entries with no breaks or interruptions. Often, the total *Fix Time* for the defects in a phase was far less than the time spent in the phase. For example, the Time Recording Log might show three hours spent in the Test phase, but the Defect Recording Log would show two defects that took eight minutes to fix. Obviously, it is not impossible that this would occur, but it is

much more likely that not all defects found in test were recorded.

In a similar vein, some projects had suspiciously few defects overall, such as seven defects for a project with 284 new lines of code and almost 11 hours of development time, (including 40 minutes in compile for two defects requiring 6 minutes of fix time). My analysis of the PSP data for that same project yielded 27 errors.

Finally, Dr. Johnson has anecdotally observed the following trend in every PSP course he has taught so far: the students turning in the highest quality projects also tend to record far higher numbers of defects than the students who turn in average or lower quality projects. If this trend is real, then we can provide two possible explanations. It may be the case that the students turning in lower quality projects tend to make far fewer errors than those turning in the higher quality projects, although this seems *extremely* unlikely. What appears more likely is that the students turning in the highest quality projects also exhibit the lowest level of collection error, which indicates that substantial but non-enumerable collection error exists in the PSP data I examined.

1.4 Effects of Data Correction

When I compared the original and corrected data, I found substantial differences in such measures as the Cost-Performance Index (planned time-to-date/actual time-to-date) and Yield (percentage of defects injected before first compile that were also removed before first compile), as shown in Figure 1.9 and Figure 1.10.

A CPI value of 1 indicates that planned effort equals actual effort. CPI values greater than 1 indicate overestimation of resource requirements, while CPI values less than 1 indicate underestimation of resource requirements. In half of the subjects, correction of the CPI value reversed its interpretation (from underplanning to overplanning, or vice-versa). In the remaining cases, several corrected CPI values differed dramatically from original values. For example Subject A's original CPI was 0.32, indicating dramatic underplanning,

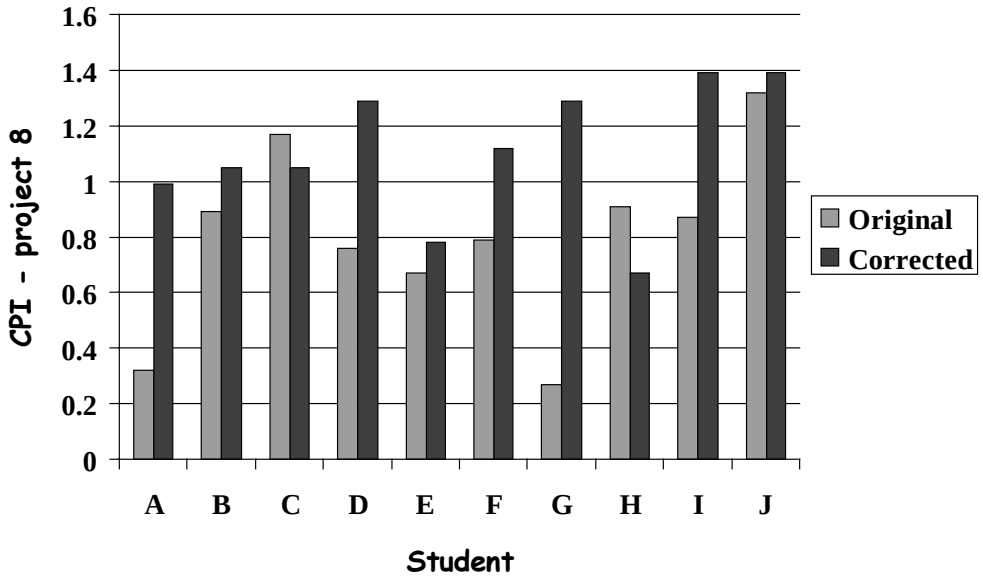

Figure 1.9: Effect of Correction on Cost-Performance Index (CPI)

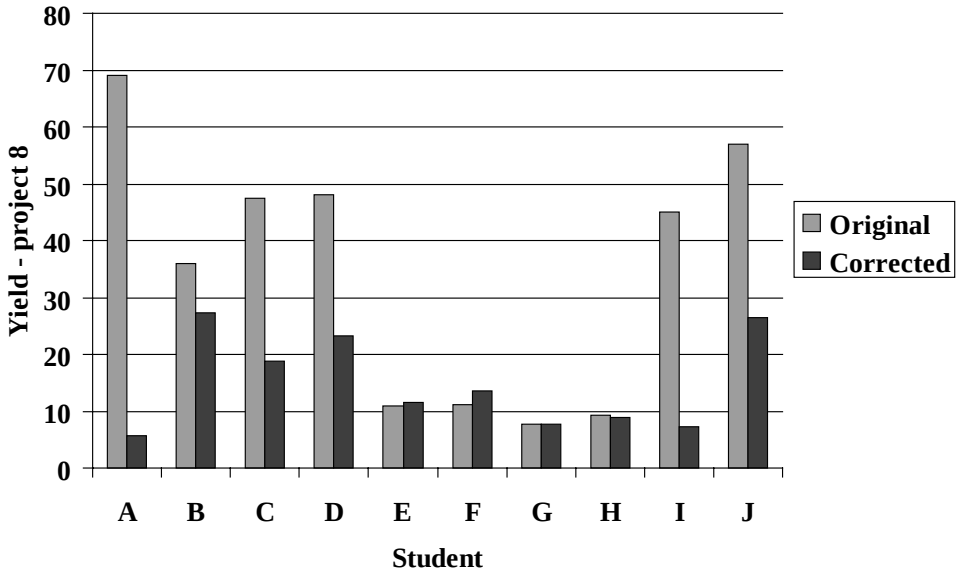

Figure 1.10: Effect of Correction on Yield

while the corrected CPI was 0.99, indicating an average planned resource requirements virtually equal to the average actual resource requirements.

Correction of yield values tended to move their values downward, sometimes dramatically. In half of the subjects, the corrected yield was less than half of the original yield values, indicating that subjects were removing a far smaller proportion of defects from their programs prior to compiling than indicated by the Yield measurement.

Bibliography

[1] Progress software. Information is available at: www.progress.com/core/develop.htm.