EMPIRICAL EVIDENCE FOR REINSTATING
THE FUNCTION POINT ANALYSIS SUPER FILE RULE

October 11, 1999
Charles B. Tichenor

Abstract

A “function point” is one standard unit of software size. Most software is decomposable into function points which can then be counted, giving its size. Standards for sizing software using function points are in the International Function Point Users Group (IFPUG) Function Point Counting Practices Manual. Some perceive a gap in one part of the current IFPUG sizing standards. According to IFPUG, high complexity software databases from 50 to 100 unique fields are considered having the same size as databases of more than 100 unique fields. These “super files” may contain up to 1000 unique fields or more. This gap renders it impossible for software developers to forecast their super file development costs using a repeatable, quantitative method. The situation is like an otherwise excellent office supply store trying to sell “very large” file cabinets without being able to define “very large.” This study uses simulations to show how reinstating a highly statistically significant super file sizing algorithm, the Super File Rule, closes the current gap.

Introduction

Forecasting the costs of operating a manufacturing production run begins with knowledge of the size of the customer’s order. Elementary operations and production management theory would state that, for example, a brewery bottleshop production run of 10,000 cases of beer requires about twice as many resources to bottle and package as a production run of 5,000 cases. Although there are other considerations in the resource forecasting process, the size of the order is by far the most important cost driver. A “case of product x,” the basic unit of production, is so easily quantifiable in the brewery business as 24 bottles of beer that there is no room for error in quantifying it.

It is important to be able to measure the cost of delivering “one case of product x” to the brewery’s customer. This brewery “cost/case” ratio can be critical in company pricing decisions. Since the “case” is readily quantifiable, this ratio can be relatively easy to measure. Imagine the inherent difficulties or skewness in pricing decisions if the “case” was believed to be a different size than it actually was.

In the software development industry, the same class of situation applies. It is important to be able to measure the cost of delivering “one standard unit of software” to the developer company’s customers. This “cost/unit” ratio should be critical in company pricing decisions.
Imagine the inherent difficulties or skewness in pricing decisions if the "unit of software" was believed to be a different size than it actually was, or was just not measured at all.

Statistical Significance of the Function Point Software Sizing Methodology

A function point is a way of classifying or measuring one unit of software size. Internationally recognized standards for counting function points are maintained by the International Function Point Users Group (IFPUG) in Westerville, Ohio. (It is assumed that the reader is familiar with the IFPUG methodology.) The methodology is explained in detail in the IFPUG Function Point Counting Practices Manual (CPM).

While it is important in this study to give the reason for needing to size software, it is also important to recognize the extremely high accuracy level of the current IFPUG methodology for sizing software by counting function points (except under conditions to be discussed).

The unadjusted function point count of the five software function types -- internal logical files (ILFs), external interface files (EIFs), external inputs (EIs), external outputs (EOs), and external inquiries (EQs) -- is found in the CPM in tables. For example, the CPM contains the table for ILFs much like the following.

<table>
<thead>
<tr>
<th>Table 1. Internal Logical Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Element Types (DET)</td>
</tr>
<tr>
<td>1-19  20-50  &gt;50</td>
</tr>
<tr>
<td>Record 1  L  L  A</td>
</tr>
<tr>
<td>Element 2-5  L  A  H</td>
</tr>
<tr>
<td>Types  &gt;5  A  H  H</td>
</tr>
<tr>
<td>(RET)</td>
</tr>
</tbody>
</table>

L = 7 unadjusted function points (ufp)
A = 10 ufp
H = 15 ufp

This table states that there are two component variables in an ILF. These are Data Element Types (DETS) and Record Element Types (RETs). The methodology states that if these
component variables are combined in ways meaningful to a user, then ILFs are generated. The unadjusted function point count is a third variable. It can take on values of either 7, 10, or 15 depending on whether the ILF is "low" (L), "average" (A), or "high" (H) complexity, as read from the table. Stated a little differently, DETs and RETs are independent variables. Given particular values for each of these independent variables, one can then determine the unadjusted function point count -- the dependent variable. For example, if the number of DETs is from 1 - 19, and the number of RETs is 2, then the ILF is of "low" complexity with an unadjusted function point count of 7.

It makes sense to perform a multiple regression to determine the statistical significance of the relationships between these three variables. The multiple regression in this study used DETs up to 100 DETs (the highest DET number before the Super File Rule DET domain begins and twice the size of the 50 DET factor), and 6 RETs. For the other function types, domains for multiple regression were twice the DET count of their tables, and RETs or File Types Referenced (FTRs) were 1 higher than their tables. Statistical significance was tested using the F statistic. Below are the results of these multiple regressions under these conditions. There is statistical significance when F equals about 5 or more.

Table 2.
Tested Statistical Significance of the Five Function Types

<table>
<thead>
<tr>
<th>Function Type</th>
<th>F Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILF</td>
<td>760</td>
</tr>
<tr>
<td>EIF</td>
<td>771</td>
</tr>
<tr>
<td>EI</td>
<td>197</td>
</tr>
<tr>
<td>EO</td>
<td>236</td>
</tr>
<tr>
<td>EQ (Output Side)</td>
<td>293</td>
</tr>
</tbody>
</table>

(The F value for EQs when the input side of the query is more complex than the output side is equal to the F value of an EI; therefore it is not listed separately in the table. Whenever the statistics of an EQ are given in this study, it is assumed that the output side is referenced, and the statistic for the input side is equal to that of an EI.

It is important to recognize how highly statistically significant these F values are. They show that the IFPUG sizing methodology for these five function types under these conditions is exceptionally strong -- an exceptionally sound testament to the IFPUG methodology under these
conditions. Perhaps these F values also set the standard for research on proposed supplements to the current IFPUG method.

- If the proposed methodology supplement is a good one, then its F values (ie, statistical significance) should be either comparable or better than those already in the methodology, as shown in Table 2. If the proposed supplement’s F values are near 0, then the proposed supplement should be rejected -- perhaps in favor of something else.

- Alternatively, if the IFPUG method’s F values for counting under conditions of the proposed supplement are either comparable or better than those already in the methodology, as shown in Table 2, then a case can be made for continuing to use it under those conditions. If the IFPUG method F values are near 0 under those conditions, then the IFPUG method should be rejected -- perhaps in favor of the proposed supplement.

This is the essence of this study’s hypothesis testing.

**Statement of the Problem**

In the manufacturing industry, customers are billed (in simple situations) based on the number of units of product they order. For example, if customers are billed at a straight rate, a customer receiving an order for 1000 cases of beer receives a bill from the brewery which is ten times higher than a customer who receives an order for 100 cases of beer. Software developers can bill their customers based in part on a given dollar per function point rate. As the number of function points delivered increase, the dollars charged also should increase.

Recall from Table 1 that a “high” ILF could have at least 2 RETs and more than 50 DETs. This is valued at 15 unadjusted function points. Suppose that the value adjustment factor (VAF) is calculated as 1.0. Then, if a customer orders an ILF with, say, 60 DETs and 3 RET this would be counted as having 15 function points.

Suppose, as a simple case, a software developer averages five days to develop a function point. (The term “develop” refers to all activities necessary in the software life cycle, such as meetings with customers to learn requirements, designing, coding, testing, implementation, writing the user manual, and perhaps writing training materials.) The developer would then schedule about 75 days to develop this ILF. If the developer charged its customer $1000 to develop each function point, then this ILF would be billed at $15,000.

Now suppose another customer wants a large master file developed as an ILF. For this example, suppose that the large master file contained 600 DETs and 30 RETs. Since the number of DETs and RETs are each ten times the amount in the first example, one might initially expect the developer to charge a much higher price to develop this large master file. However, according to the CPM, such a large master file must be counted as a “high” -- having more than 2 RETs and
more than 50 DETs. If the developer was contracted to charge following the rules in the CPM, the developer would have to charge about $15,000 for this large master file, also, and promise to schedule this development time at about 75 days. This is clearly an impossible situation for the developer.

This is the perceived gap studied in this research. What is needed here is a special rule to account for the additional functionality inherent in large ILFs -- such as large master files over 100 DETs.

The amount of functionality for ILFs and EIFs having more than 100 DETs is proposed to be measured by using the “Super File Rule.” This counting rule states that if an ILF and/or EIF has more than 100 DETs, then each RET should be counted as a separate ILF (or EIF), assuming it otherwise meets all of the criteria for an ILF (or EIF). For example, if an ILF has 300 DETs, and if it contains 10 RETs each having 30 DETs, then it is counted as 10 ILFs. Since each of those 10 has 30 DETs and 1 RET, then each of them is counted as “low” ILFs of 7 unadjusted function points each.

How sound is this Super File Rule? Its degree of soundness should be measurable according to the standards set earlier.

- If the Super File Rule is a good one, then its F values should be either comparable or better than those already in the methodology, as shown in Table 2. If the Super File Rule F values are near 0, then the Super File Rule should be rejected -- perhaps in favor of something else.

- Alternatively, if the IFPUG method’s F values for Super File-sized files are either comparable or better than those already in the methodology, as shown in Table 2, then a case can be made for continuing to use it under Super File conditions. If the IFPUG method F values are near 0 under Super File conditions, then the IFPUG method should be rejected -- perhaps in favor of something else.

This study analyzes the problem, stated formally but in layman’s terms, as follows.

H0: There is identical functionality delivered to users by large master files as by database files having less than 101 unique fields.

H1: There is more functionality delivered by large master files than by database files having less than 101 unique fields.

Limitations of the Study
This study uses the Super File Rule as originally developed by Mr. William Hufschmidt, Principal of Development Support Center, Inc., 1625 Lindhurst Dr., Ste. 100, Elm Grove, WI 53122. The rule was published in CPM 3.4.

This study is applicable to sizing software using the procedures from the CPM 4.0, which was the current version during this research.

The researcher is professionally certified by IFPUG as a Certified Function Point Specialist, and attests that the hypothesis test and resulting implications in this study is based on his best understanding of the IFPUG function point methodology. The intention of this study is to suggest a change to the IFPUG methodology under the quality assurance theory of continuous improvement.

Review of Related Literature

There is virtually no previously published research regarding the Super File Rule. Internet research on “Super File” conducted during the week ending August 15, 1998 using the seven popular search engines failed to generate any function point analysis references for this term. Other literature search attempts also found no such references. The only reference found was in the IFPUG Counting Practices Manual 3.4, Appendix A’s treatment of the Super File Rule.

Research Design and Methodology

Software sizing data collection was through computer simulation, generating a statistically large sample of simulated ILFS and EIFs of randomly sampled combinations of DETs and RETs within the domain of 101 to 1000 DETs and 1 to 10 RETs. Sensitivity analysis was also conducted.

The simulation has the following steps.

1. Develop a spreadsheet analysis showing the correlation between DETs and RETs as independent variables, and unadjusted function points as the dependent variable, for ILFs sized using the Super File Rule. The DETs will be randomly generated from 101 to 1000, and the RETs randomly generated from 1 to 10. Test the correlation for regression, and report the F value and significant F.

2. Develop a spreadsheet analysis showing the correlation between DETs and RETs as independent variables, and unadjusted function points as the dependent variable, for ILFs sized using the IFPUG CPM method. The DETs will be randomly generated from 101 to 1000, and the RETs randomly generated from 1 to 10. Test the correlation for regression, and report the F
value and significant F.

3. Develop a spreadsheet analysis showing the correlation between DETs and RETs as independent variables, and unadjusted function points as the dependent variable, for EIFs sized using the Super File Rule. The DETs will be randomly generated from 101 to 1000, and the RETs randomly generated from 1 to 10. Test the correlation for regression, and report the F value and significant F.

4. Develop a spreadsheet analysis showing the correlation between DETs and RETs as independent variables, and unadjusted function points as the dependent variable, for EIFs sized using the IFPUG CPM method. The DETs will be randomly generated from 101 to 1000, and the RETs randomly generated from 1 to 10. Test the correlation for regression, and report the F value and significant F.

5. Plot the 30 F value data points generated for each simulation using “Simulation Run #” as the x axis and the corresponding “F value” as the y axis for the ILF F test Super File and IFPUG CPM method data and the EIF F test Super File and IFPUG CPM method data.

6. Perform a sensitivity analysis, varying the RET domain from 1 to 50 and 1 to 100, keeping DET simulations from 101 to 1000. Vary the DET domain from 1 to 500 and 1 to 2000, keeping RET simulations from 1 to 10.

Presentation and Analysis of Data

Testing the null hypothesis means considering the correlation between the independent variables DETs and RETs, and the dependent variable unadjusted function point count, under the four base scenarios. These four scenarios are ILF unadjusted function points counted using the Super File Rule method, ILF unadjusted function points counted using the current IFPUG method, EIF unadjusted function points counted using the Super File Rule method, and EIF unadjusted function points counted using the current IFPUG method.

The below graph summarizes the results of the first simulation set. The simulation run number is indicated on the x axis, and the corresponding F values on the y axis. Similar graphs were developed for each simulation run.

Using the Super File Rule method, the F values ranged from a high of 846 in run 22, to a low of 448 in run 12. The computer spreadsheet’s “Significance F” (i.e., the p-value) calculated F = 846 was 3.97E-62, and the “Significance F” for F = 448 was 1.01E-49. These are statistically significant.
A similar simulation run was conducted for ILF unadjusted function point counts using the IFPUG method. The results of those simulations are plotted on the below graph. It is very apparent that the IFPUG method F values are very small compared to the Super File Rule F values.

Using the IFPUG method, the F values ranged from a high of 25 in run 22, to a low of 2 in run 21. The computer spreadsheet’s “Significance F” value calculated for F = 25 was 2.41E-09, and the “Significance F” for F = 2 was 1.32E-01.
A similar simulation run was conducted for EIF unadjusted function point counts using the IFPUG method. The results of those simulations are plotted on the below graph.

Using the Super File Rule method, the F values ranged from a high of 877 in run 1, to a low of 460 in run 30. The “Significance F” calculated by Excel for F = 877 was 7.85E-63, and the “Significance F” for F = 460 was 3.27E-50. These are statistically significant.
A similar simulation run was conducted for EIF unadjusted function point counts using the IFPUG method. The results of those simulations are plotted on the below graph. It is very apparent that the IFPUG method F values are very small compared to the Super File Rule F values.

Using the IFPUG method, the F values ranged from a high of 22 in run 26, to a low of 5 in run 28. The "Significance F" calculated by Excel for $F = 22$ was $3.60E-06$, and the "Significance F" for $F = 5$ was $7.94E-03$. 
Testing Hypotheses Against the Data

The null and alternate hypotheses are repeated here.

H0: There is identical functionality delivered to users by large master files as by database files having less than 101 unique fields.

H1: There is more functionality delivered by large master files than by database files having less than 101 unique fields.

The results of the simulations are summarized below in Table 3.
Table 3
Summary of Simulation Statistical Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Highest F</th>
<th>Lowest F</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILF Super File Rule</td>
<td>846</td>
<td>448</td>
</tr>
<tr>
<td>ILF IFPUG Method</td>
<td>22</td>
<td>2</td>
</tr>
<tr>
<td>EIF Super File Method</td>
<td>877</td>
<td>460</td>
</tr>
<tr>
<td>EIF IFPUG Method</td>
<td>22</td>
<td>5</td>
</tr>
</tbody>
</table>

These F values can be favorably compared to those for ILFs and EIFs in Table 2, being 760 and 771, respectively.

**Sensitivity Analysis**

Sensitivity analysis for the Super File Rule for both ILFs and EIFs was conducted. Two simulation sets were conducted allowing the number of RETs to randomly fluctuate up to 10, and allowing the number of DETs to vary -- at up to 500 DETs and 2000 DETs. This represents the sensitivity of the Super File Rule to changes in DETs. Two simulations sets were conducted allowing the number of DETs to randomly fluctuate up to 1000 DETs, and allowing the number of RETs to vary -- fluctuating from 1 to 50 RETs and fluctuating from 1 to 100. This represents the sensitivity of the Super File Rule to changes in RETs. The below table summarizes the average F values for both the base case and sensitivity analysis.
Table 4.
Average F Value Summary

<table>
<thead>
<tr>
<th></th>
<th>500 DET</th>
<th>1000 DET</th>
<th>2000 DET</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILF</td>
<td>463</td>
<td>616</td>
<td>3537</td>
</tr>
<tr>
<td>EIF</td>
<td>521</td>
<td>663</td>
<td>1086</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>10 RET</th>
<th>50 RET</th>
<th>100 RET</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILF</td>
<td>616</td>
<td>7270</td>
<td>43444</td>
</tr>
<tr>
<td>EIF</td>
<td>663</td>
<td>7292</td>
<td>49093</td>
</tr>
</tbody>
</table>

Research Conclusion

The basic analysis and the sensitivity analysis as summarized in Table 4 show that as Super Files increase in size and complexity, the correlation between the independent variables DETs and RETs, and the dependent variable unadjusted function point count, increases to very high levels. At the same time, the simulations and discussions showed how the corresponding correlations using IFPUG CPM diminish to virtually 0. It is concluded that this result is in agreement with H1. The Super File Rule is necessary to maintain the high statistical significance of the IFPUG function point analysis methodology.

Implications of Research Findings

There are several important implications of these research findings.

It is important to recognize the high statistical significance (represented by the F values in Table 2) of each of the five function types under the usual conditions where the Super File Rule does not apply. They show that the IFPUG sizing methodology for these five function types (when each ILF and/or EIF has 100 DETS or fewer) is exceptionally strong. There is an important implication here for future research. Researchers who offer competing software...
functional sizing methodologies, or "supplemental" methodologies, should package their ideas with corresponding F values (or other statistical methods). These F values should be closely compared with the IFPUG values. If the F values are lower, then one might question the contribution of the idea. If the F values are higher, then one might want to know the cost of converting to something else when the IFPUG methodology is already highly statistically significant.

The criteria for acceptance or rejection of use of the Super File Rule should rest with its statistical significance. Its statistical significance increases as the size and complexity of Super Files increase.

Organizations using function point analysis for forecasting the costs of developing software should experience better cost forecasts using the Super File Rule if the software contains large master files of Super File size. The forecasts for schedule should improve under these conditions, also. The accuracy of quality metrics for software containing super files based on some form of "defect/function point" should also improve.
References


