A Comparison of Function Point Analysis versus Engineer Experience in Predicting Semiconductor FAB Integration Software Project Durations.

Jonathan L. Maddix

Abstract

Function point analysis is a method of determining a software project’s size based upon the user requirements. The research done in this paper takes this and compares how well it fairs versus software engineering experience or guesstimates. Why should one care? This is further development of software metrics and standards that are the glue that holds together an engineering field. The research done here is an analysis of how well project duration can be estimated specifically for enhancement projects. The impact will allow better tools and techniques to be developed in the Semiconductor software field. The problem of determining software project duration is one that has plagued the industry and Factory Automation projects. Customers become frustrated at the constant miscalculations and changing of completion dates. To address this issue, I have done Function point analysis on enhancement projects and along with historical data compared them against experienced engineer guesses on how long a project would be completed. Then, statistical analysis was done upon the results. Function point analysis alone cannot accurately predict project durations; it should be coupled with historical analog based data.

Introduction

The research done here is to further develop statistics and metrics so that factory automation software can be more cost efficient. The development of standards in measurements can only improve the bottom line of manufacturing computer chips.

Background

Even with all the spotlights on the Software industry in recent years, Software metrics such as exactly defining a methodology of estimating project duration is very much a weak process. In practice as anyone knows it is usually done ad hoc with reliance on engineer experience and savvy. In other words, many shops simply rely on what an experienced engineer says in determining how long a project will take.

In the 70’s, IBM decided to produce a system of estimating software development effort that was language independent. This task was given to Allan Albrecht who along with others came up with Function Point analysis.

In the late eighties The International Function Point Users Group (IFPUG) was founded. They developed their own Counting Practices Manual and in 1994 released version 4.0.

During the eighties and nineties, many companies have adopted this as the metric to estimating software costs, size, and aid project management.

Function Point analysis (FPA) is a technique that is a measure of the size of computer applications and the projects that build them. The size is measured from a functional, or user, point of view. Function points are independent of the computer language, development methodology, technology or capability of the project team that develops the application in question. Simply put, Function points measure size. Readers who would like to view how to perform Function point analysis can refer to the newest release manual [1].

ISBSG is the International Software Benchmarking Standards Group. They have collected project estimations from hundreds of organizations and compiled them into a database. This database can be accessed and used to compare against similar software projects in industry.

There has been much use of this technique in determining metrics of medium to large-scale applications but little has been done to projects that are of a legacy nature. In our department we primarily do enhancements to legacy code based upon user requests from the Semiconductor manufacturing process engineers. They want further enhancements primarily to software that is already functioning and running on the manufacturing line. One of the things that causes much concern is that completion dates that are given at the beginning of the enhancement project are not usually kept and dates being moved is almost a given occurrence. They are now sometimes referred to as “floating” dates. Our industry does a lot of maintenance and software evolution. Software evolution is the process of changing and updating legacy software.

Case studies

There has been interest of course in testing the reliability of FPA counts and it has its critics as well as promoters. Kemerer [2] did a comprehensive study on the reliability of doing this user functionality count and found that Function points offered a reliable method of software metrics if done properly. In particular Agarwal [3] provided a good background for this study in the performing of software metrics and how they can be used.

The IRS [4] has used function point counts in their software efforts reducing costs. Obviously counts can be of some benefit, just further study is needed.

Purpose

The significance of the Function point analysis research is fundamental to the progression of Software Engineering.
becoming a true Engineering field. Measurement and metrics are needed to quantify and qualify the development of software so that it may become a more predictable and efficient endeavor. The discovery and refinement of metrics for Software development means better productivity and predictability.

This type of research has not been adequately done in the semiconductor industry and would greatly benefit the start of metrics for 200 mm and 300 mm factory software solutions. The ultimate goal would be to develop tools and processes to better enable project duration estimates and metrics for Factory Integration. Some of these would be automate the process of estimation and develop an internal repository of duration; etc data so that internal targeted regression equations could be developed.

The other benefit to the Software metrics cause on a whole is to help the drive toward a greater understanding of what software metrics are, how they can be done, and applied. The more research done on this subject can only make the standardization of metrics and the methodology to achieve metrics easier to achieve.

**Development /Research Approach and Methodology**

This section lists the methods and approach that I used in order to collect the data and to analyze it statistically. Projects were gathered from the IBM Equipment Integration project database. All the project data is for enhancement projects for factory automation in the IBM Burlington VT site.

**Measures and Operational Definitions**

This is the listing of the measures and operational definitions that were used in this study. Function points were determined using the International Function point users group method of counting function points and then these were converted into function points per hour of work. In other words the function points were counted for each project and then divided by the average function points. The average function points were determined by taking an average of the function points found for all the projects and dividing this by the average of actual project duration. This gave me the average number of function points found for a project in Equipment Integration. In other words a historical reference to go by.

The resulting hours that the function point estimates found were then taken and compared against the actual project duration, the time the project actually took to complete. The result was either a negative number for when the project was underestimated e.g. the estimation said the project would take a shorter time period than it actually did, or a positive number for an estimate that said the project would take a longer time than it actually did. The same procedure was done with the Engineer estimates.

One thing to remember is that the hours were based on a 40-hour workweek for both the FPA method and the engineer method.

**Function Point measures and operational definitions**

**Measure:**
The number of function points per the IFPUG version 4.1 counting manual.

**Characteristic of interest:**
Namely the number of function points per hours per software project.

**Measuring instrument:**
The measurements are the IFPUG version 4.1 counting procedures for finding function points in a software project.

**Engineer measures and operational definitions**

**Measure:**
The number of hours per software project estimated originally by the engineer.

**Characteristic of interest:**
The hours of engineer estimated project duration.

**Measuring instrument:**
The project database for Equipment Integration showing the estimates for software project duration and actual project durations.

**Hypothesis to be tested**

There is no statistical difference in the software project duration estimates between engineer estimates and function point analysis.

**Alternative Hypothesis**

There is a statistical difference in the software project duration estimates between engineer estimates and function point analysis.

**Operational Procedures for the Research process**

**Steps to perform Function Point Analysis**

In order to perform the Function Point analysis, these are the basic steps [5]:

1. **Plan the Count**

This involves determining what type of count is to be done, in this case an enhancement count. And then one needs to identify the counting boundary. This can be the most complicated of the steps especially trying to discern between where the enhancement begins or ends. In this case we do have some help, we have a comparison program built into PVCS and all enhancements are versioned which can be used to compare the different source code files. The counter is also the system expert so this cuts out some of the needed communications between counter and domain expert.

2. **Identify all data functions** (internal logical files and external interface files) and their complexity. Expressed as an integer value.
3. Identify all transactional functions (external inputs, external outputs, and external inquiries) and their complexity. Again expressed as an integer value.

4. Determine the unadjusted function point count
This is calculated from the totals of the functions, inputs, outputs, etc. These are then total up to arrive at the unadjusted function point count.

5. Calculate the Value Adjustment Factor (VAF).
The IFPUG manual describes this step closer to the end of the counting process. Here we total the scores for all 14 characteristics and multiply by the unadjusted function point count to calculate the VAF. The VAF is calculated on the basis of the identification of 14 general system characteristics.

6. Calculate the adjusted function point count
This is the unadjusted function point value multiplied by the VAF. Because a float is multiplying the unadjusted function point value, we will round up the results to get an integer for the adjusted function point count.

7. Review and count again
This ensures that the count was accurate and no steps were missed.

Function points are listed as integer values and of course called points. The VAF is listed as a floating-point number. Enhancement projects have a formula specifically for them; it is used at the end to determine EFP or enhancement project function point count.

Then we find a sum of the ranks from the both samples such that:

\[ T = \sum_{i=1}^{n} R_i \]

Now, because we desire a significance level of .05 of each Hypothesis, then if the value of T is t then we shall reject our Hypothesis if either:

\[ P\{T \leq t\} \leq .05 / 2 \quad \text{or} \quad P\{T \geq t\} \leq .05 / 2 \]

In our case because our samples are equal in length, we shall take the minimum value of the sum of the ranks of either sample. We then compute \( T_2 = n_1(n_1 + n_2 + 1) - T_1 \).

\( T \) is the minimum of \( T_1 \) and \( T_2 \). Sufficiently small values of \( T \) cause rejection of the null hypothesis that the sample means are equal.

The P value can also be computed as:

\[ \text{mean is: } \frac{T - n(n + m + 1)}{2} \]
\[ \text{standard deviation is: } \sqrt{\frac{nm(n + m + 1)}{12}} \]

This approximates a normal distribution if \( n \) and \( m \) are of a moderate size, greater than 7 should suffice.

Results

Data obtained and Statistical analysis

Independent variables

The independent variables for this study were the IFPUG function point analysis counting procedure estimates and the estimates listed in the Equipment Integration database. The actual duration of the projects also are independent variables used in this study.

Dependent variables

The dependent variables for this study were Function point project estimates, the engineer project estimates; both measured as hours, based upon an 8 hour work day or 40 hours per week.

Statistical Analysis

The data collected is ordinal in nature so a Mean (average) for each Dependent variable will be collected. In order to compare apples to apples, I compared the average number of estimated project hours. Function points found and engineer estimates were each converted to hours of estimated project duration. To maintain a normal sample, 30 separate software projects were tested. Those tested were only taken from projects that were enhancements and those done by engineers with at least 10 years of experience [4].

The data collected is also inferential in nature. For each Hypothesis the groups (Engineer experience and Function point analysis) are independent so for each case I used the Mann-Whitney test or the Rank Sum test. For the Hypothesis I collected the estimation data for all 30 software projects and then performed the test.

Here is how the test was done:

We shall let one set of values be ranked in order from smallest value to largest as \( X_1, ..., X_n \) and let the second set of values be ranked the same manner as \( Y_1, ..., Y_m \). We shall then take from both sets and rank them from smallest to largest value giving them an ordinal rank:

\( R_i = \text{rank of the data value } X_i \)

In our case because our samples are equal in length, we shall take the minimum value of the sum of the ranks of either sample. We then compute \( T_2 = n_1(n_1 + n_2 + 1) - T_1 \).

\( T \) is the minimum of \( T_1 \) and \( T_2 \). Sufficiently small values of \( T \) cause rejection of the null hypothesis that the sample means are equal.

The P value can also be computed as:

When the Hypothesis is true,

\[ \text{Where } n \text{ is the number of samples from the first set and } m \text{ is the number from the second set.} \]

\[ \text{mean is: } \frac{T - n(n + m + 1)}{2} \]
\[ \text{standard deviation is: } \sqrt{\frac{nm(n + m + 1)}{12}} \]

This approximates a normal distribution if \( n \) and \( m \) are of a moderate size, greater than 7 should suffice.

Results

To the Hypothesis question:

There is no statistical difference in the software project duration estimates between engineer estimates and function point analysis.
Here are the results of the tests:

The value of the ranks of our test data for Function points converted to hours of effort for the testing sample is 831. Using a p-value software-calculating program we obtain a p-value of 0.21876. We want a significance level of .05. Again, we shall reject the above hypothesis if:

\[ P(T \leq t) \leq 0.05 / 2 \quad \text{or} \quad P(T \geq t) \leq 0.05 / 2 \]

So our p value is 0.21876 and .05 / 2 is 0.025. So, because 0.21876 is not less than 0.025, we can not reject our null hypothesis as not being true. Our conclusion for the function points analysis and experienced engineering estimates suggests that there is not really a statistical significant difference between the two.

But with the testing I did find that the function point analysis method of estimation was much closer to predicting a project’s duration than the estimates of experienced engineers. Function point analysis over estimated on average 0.411 hours a project’s duration, while the engineers’ estimates were over by 84.267 hours or about 2 workweeks of effort.

Figure 1 shows a graph of the estimates for FPA method and the engineers matched with the actual project durations.

Figure 2 shows the Function point analysis estimates and engineer estimates differences. These are the differences from the actual project durations. Wide variances could be attributed to non-measured factors such as getting tool time or test time for a factory automation project.

**Figure 1. Estimates versus actual**

**Figure 2. Duration Differences from actual**

**ISBSG Historical data comparison**

I also did research using the International Software Benchmarking Standards Group’s historical database on function points. This data was collected using the ISBSG’s Regression Equation tool. Inputs were the function points found per project, the team size of 1, and the platform, which was PC. These were inputted and the tool applied ISBSG developed regression equations and outputted estimates of duration. As Figure 3 shows, there were wide variances in the durations predicted for the projects versus the actual durations. The average duration in days for the ISBSG data was 56,900 against the average actual duration of 39.37. Compare this to the average engineer estimation of 49.90 and the Function point analysis estimation of 39.42. The FPA only fared well because it is historical average of the function point counts per hour.
involved in doing enhancement projects that are not reflected in Function point analysis. For example it has been the author’s experience that the process of waiting for semiconductor tool testing time can delay and add on to the time period a project takes. This would not be in the FPA counts. Further studies would need to take into account these other attributes to make metrics even more precise.

The warning here is that one needs to find the FPA count average per hour based on the historical project data. In other words, one needs to take into account the past FPA average counts to predict how long a project will take in the future. This is very application specific and industry specific.

Conclusions

In conclusion, it is obvious that doing function points alone without historical data is not the way to go. A historical database, which has average counts for previous projects would be more accurate in predicting how long a project, will take. Good requirements are also the key to doing counts. Without a well documented requirements process, FPAs can not be done. As for the accuracy, if the software department is filled with good experienced engineers, accurate project estimates can be made; but they are not as accurate as historical data in conjunction with function point counts. The extra two weeks added on to the engineer’s estimates could be a factor that costs a company more money, time and incorrect calculation of what projects can be scheduled. Scheduling of high priority enhancement projects is very important in semiconductor factory automation where time is of the essence. Better estimates can cut a further notch off the bottom line for management.

Future Work and Further research

In an expansion on this research, I would like to design a project repository database with a Java front end. This would allow users to input completed projects as well as do estimates on new project based upon the requirements. It would be specifically targeted toward Semiconductor software projects but the results of my comparison of this tool and other estimation methods could be used in general estimation tool development by others.

Further data gathered and more counts on each new project will help to build a database of accurate FPA’s. This data could be distributed to our sister plant in Fishkill, NY further cutting the costs of software integration in a very cyclic industry.

Again, very important to all this is the development of a Function point count database. This would keep track of historical counts and allow queries to find the average count per type of software project. This research focused on enhancements but there are other types of projects to be looked at. This research looked at all enhancements and treated them as being the same but a database would include further hierarchical categories within enhancement projects. This would lead to further defined metrics.

This database would include the option of allowing users the ability to select attributes based upon what kind of project they are planning. For example is this a data

Discussions

So what does this all mean? Collecting this information will better help an industry that needs software metrics further defined and utilized. Consistently in the semiconductor industry factory automation departments are the hit hard with layoffs and resource actions. The better estimation of how long a project will take can show the importance of keeping these departments intact and show that they contribute to the bottom line.

For the software industry as a whole the more data that is collected and gathered for determining if function points are the way to go toward estimating projects, the better defined the field of software engineering will become.

Before starting this study I would have thought the engineer’s experience would fair very well against the FPA and I was correct in this assumption. I was surprised at how well the Function point analysis compared against the engineer estimates. Niessink’s study [6] also found that engineers who had experience fared very well versus function counts. The projects that I used were from Engineers who had at least 10 years experience in the field, doing this estimation over and over. But the next study should take projects from all engineers; this would hypothetically show a further gap in estimates versus how long the project really took considering I would be looking at data from engineers with less experience. Even though statistically the two methods were insignificant in difference, FPA was only about .4 hours off on average versus about 2 weeks for the engineers. This could save money and time in the long run. In the semiconductor industry every project penny saved could mean a huge difference. It could also free up engineers to plan other projects to be worked on during the time that they said they would still be working on the previous project.

As the data clearly shows, if one were to take just function points alone without historical data, the counts would not be as accurate as they should. Many other factors are
Applying the research to application development

This research more specifically targets the IBM Factory Integration development process but can be applied to enhancement projects in any organization. Much of the code out there is still legacy code that needs to be enhanced and built upon. The outcome of this research can be used as the basis for other projects involving duration estimates and legacy code and can be built upon. An interesting concept would be to use this research to develop an expert system that does legacy enhancement estimations automatically. This CASE tool could be beneficial to IBM and any other organization that enhances legacy code. The subject of Function Point analysis and enhancement projects is a fairly under-developed topic that needs further research and practical application. Such an application would need a rule based forward chaining expert system, probably written in prolog.

References


