An Analysis of Function Point Trends

Introduction

Function point analysis has played an important role in software measurement and analysis for 30 years. This study looks at the QSM software project database and examines a set of validated projects counted in function points that have completed since the year 2000 to see what they tell about productivity, schedule, and staffing. We are fortunate to have several thousand projects in this sample to work with as this allows us parse to the data many different ways and still have enough projects to be statistically significant. For this study only unadjusted function points were used.

Demographics

Our sample contains 2,231 projects completed since the year 2000.

The "Average" Function Point Project

What does a representative project sized in function points look like?

- **Domain:** 98% business IT projects
- **Size (median):** 160 function points
- **Schedule (median):** 7.03 months from the start of analysis through implementation
- **Effort (median):** 21.85 person months
- **Average Staff (median):** 2.3 full time equivalents
- **Labor Cost (median):** $262,200 At $75/hour based on a 160 hour work month
- **Requirements Effort (median):**13% of project effort spent in analysis and high level design
- **Development Type:** 75% are enhancements to existing systems

Figure 1 shows the size distribution of the function point projects. Only 7% are larger than 1000 FP.

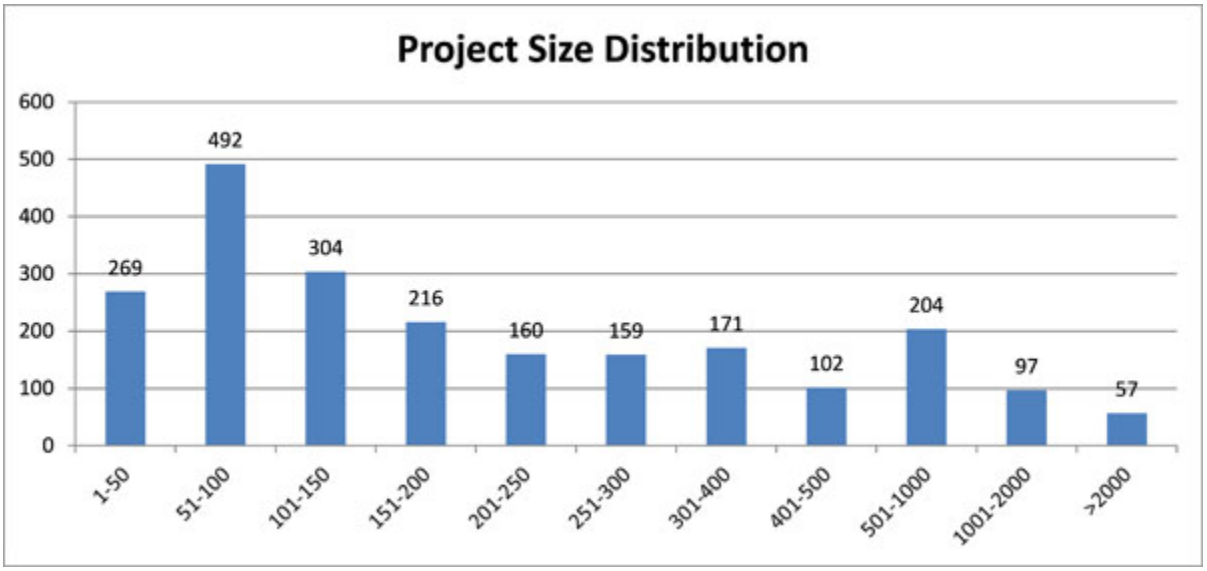

Figure 1

New Development, Enhancements, and Maintenance

QSM classifies software development projects by the ratio of new code to modified, deleted, or reused code:

- New development (> 75% new functionality)
- Major enhancement (25% - 75% new functionality)
- Minor enhancement (5% - 25% new functionality)
- Conversion (< 5% new functionality)
- Maintenance

The majority (75%) of function point projects in the QSM database are enhancements to existing systems. Conversion projects (those with less than 5% new functionality) are less productive than other project types. A more in-depth analysis of conversion projects is available here.

	New Development	Major Enhancement	Minor Enhancement	Conversion	Maintenance
Percent of Projects	16%	61%	14%	7%	2%
Median PI	14.0	10.1	11.2	9.9	10.4
Median size (FP)	291	119	153	109	68
Median effort mont	29.7	19.3	28.1	23.4	18.6
Median % Funct Eff	12%	11%	12%	10%	19%
Median FP/PM	9.16	5.79	5.19	5.06	2.70
Median Duration	7.57	7.23	6.42	6.43	4.73
Median Defects	37.00	16.00	38.50	35.00	16.00

Table 1

New development projects are the largest and most productive, but they account for only 16% of the sample. While extensive code reuse in enhancement projects provides additional functionality that the project does not have to develop, it also creates a more complex development environment for the new and modified code which will require additional analysis to ensure that it is compatible with the existing code and extensive regression testing to verify that there are not unforeseen impacts.

Productivity

Few topics in software measurement generate as much discussion and debate as productivity. While linear or ratio-based measures like hours per function point or function points per person month are widely used, QSM's productivity metric – the PI or Productivity Index, differs in two important ways from ratio-based productivity measures:

- It accounts for size, effort, **and** schedule performance.
- It accounts for the distinctly nonlinear relationship between these three metrics.

The scatter plot shown in Figure 2, allows the user to see how much a project or estimate is above or below average while providing a visual comparison to the organization's other completed projects. While past performance is no guarantee of the future (in the case of an estimate) it can serve to define the boundaries of what is possible.

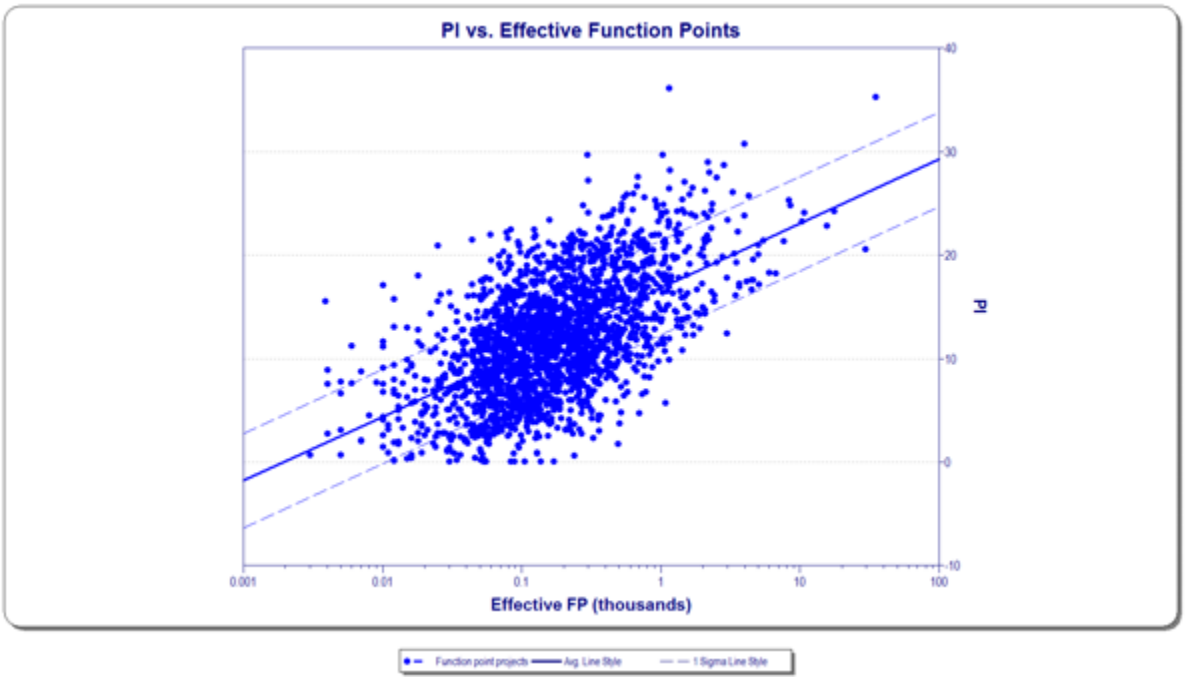

Figure 2

Our research shows that project size has an important impact on productivity, whether measured by our PI or in function points per person month. Simply put, larger projects are more productive. While we can only speculate, here are several possible causes.

- Larger projects are more important to organizations. They cost more and have higher visibility. As a result, they benefit from more experienced developers and better project management and tools.
- Another reason is less positive. Large projects, as Capers Jones tells us, are more likely to be cancelled.1 Since we calculate project productivity from completed projects, the impact of cancelled or failed projects is not reflected in our productivity measures. This is not just a preference of ours. Failed and cancelled projects frequently have not had effective metrics processes in place, in which case the data do not exist. If they do exist, they are by definition incomplete. And in any case, projects that fail are often more interested in covering their tracks than in publicizing what occurred. At the same time small projects that are experiencing problems may be allowed to limp along to completion. They aren't going to bankrupt the company! Regardless of our preference, interest remains in how many function points per person month (or hours per function point) constitutes normal productivity. Table 2 looks at the productivity of different size ranges. Since projects differ in scope, we have included only effort from the beginning of analysis up to implementation into production to normalize the comparison. (These activities correspond to the Requirements & Design and Code & Test phases in the SLIM model).

Productivity by Size Category		
Size (FP)	Count	FP/PM (Median)
<=50	269	3.49
51-100	492	5.13
101-150	304	6.54
151-200	216	6.67
201-250	160	7.65
251-300	159	8.49
301-400	171	9.55
401-500	102	9.72
501-1000	204	13.43
1001-2000	97	16.29
>2000	57	23.10

Table 2

Project Effort

The majority of projects are not multi-million dollar endeavors with dozens of programmers. 61% of the projects in our study expended 30 or fewer person months of effort. From a cost perspective, at a labor rate of $10,000/person month, the labor cost of these projects was $300,000 or less. Over 75% of the projects had 50 or fewer person months of effort. While Table 2 showed that productivity measured in function points per person month increases with project size, Table 3 shows a different trend: as effort grows, productivity decreases. Evidently, larger projects that complete are more efficient in their use of labor. They are more likely to have full time dedicated resources who do not have to divide their attention between multiple projects.

Effort	Project Count	% of Projects	FP/PM (Median)	Median Size (FP)
<=10 PM	487	21.83%	11.6	72
>10 <=20	551	24.70%	7.86	113
>20 <=30	323	14.48%	5.87	139
>30 <=40	194	8.70%	5.65	194
>40 <=50	127	5.69%	5.44	247
>50 <=60	109	4.89%	5.32	301
>60 <=70	67	3.00%	4.45	292
>70 <=80	60	2.69%	4.73	348
>80 <=90	38	1.70%	3.45	292
>90 <=100	30	1.34%	3.32	312
>100 <=150	102	4.57%	3.12	359
>150 <=200	52	2.33%	3.44	606
>200 <=300	45	2.02%	2.66	597
>300 <=400	12	0.54%	3.13	1,041
>400 <=500	13	0.58%	3.33	1,477
>500 <=1000	14	0.63%	3.11	1,989
>1000	7	0.31%	2.55	3500

Table 3

As Figure 3, below, demonstrates, project types differ in the average amount of effort they require. New development projects expend the most effort, followed closely by Minor Enhancements. As the median size of New Development projects is nearly twice that of Minor Enhancements, this is strong evidence that what is often considered “free” functionality from reuse actually has a productivity cost associated with it.

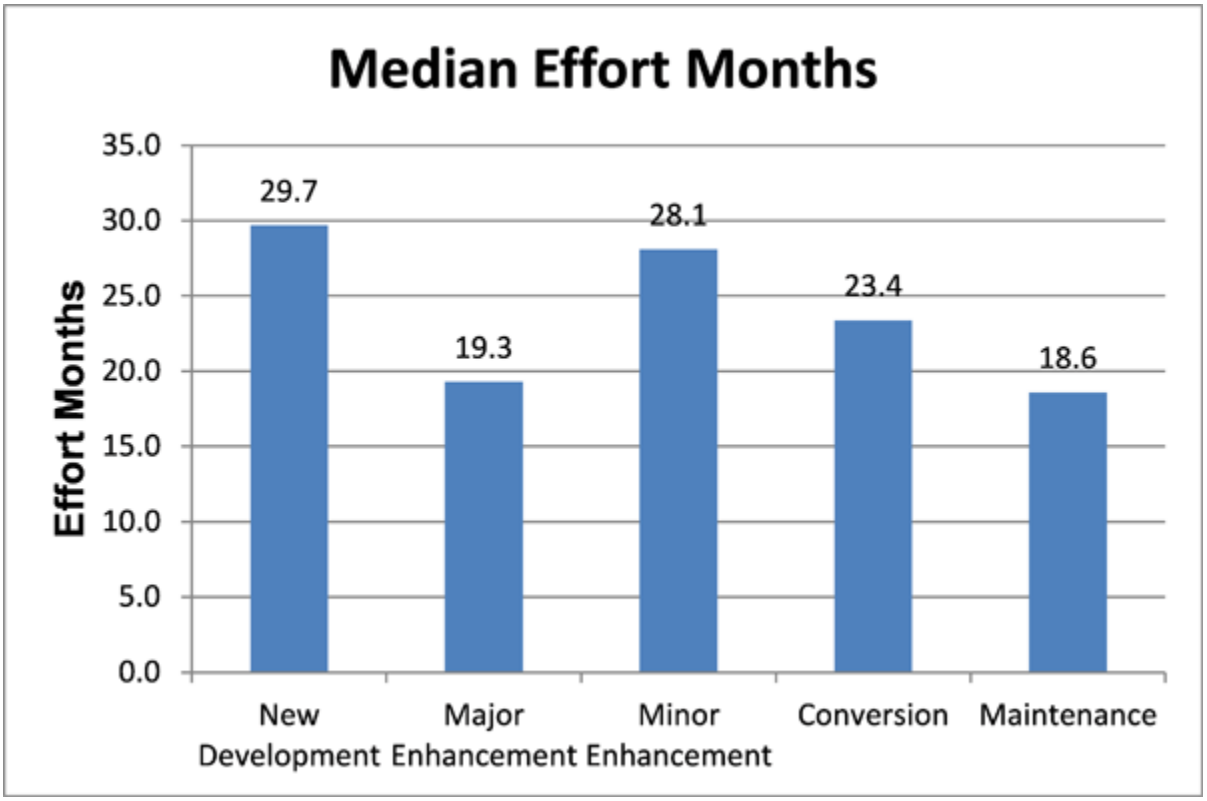

Figure 3

Schedule

In this section we look at schedule from two perspectives. First, we examine the schedule distribution of projects. Then, we look at how schedule compression and extension affect productivity.

Schedule Distribution:

As we have seen, the majority of the sample projects are enhancements to existing systems. The basic framework of the software they modify already exists. Their purpose is to add new features, modify how existing functionality works – which can include correcting defects -, improving performance, or any combination of these. In a word, their objectives are normally better defined than new development which has more aspects of a learning process for both developers and end users. As Table 1 illustrated, their median schedules are slightly shorter. Figure 4 illustrates project schedule distribution. Here are some takeaways:

- 50% of FP projects last 7 months or less
- 70% last 9 months or less
- 85% last a year or less

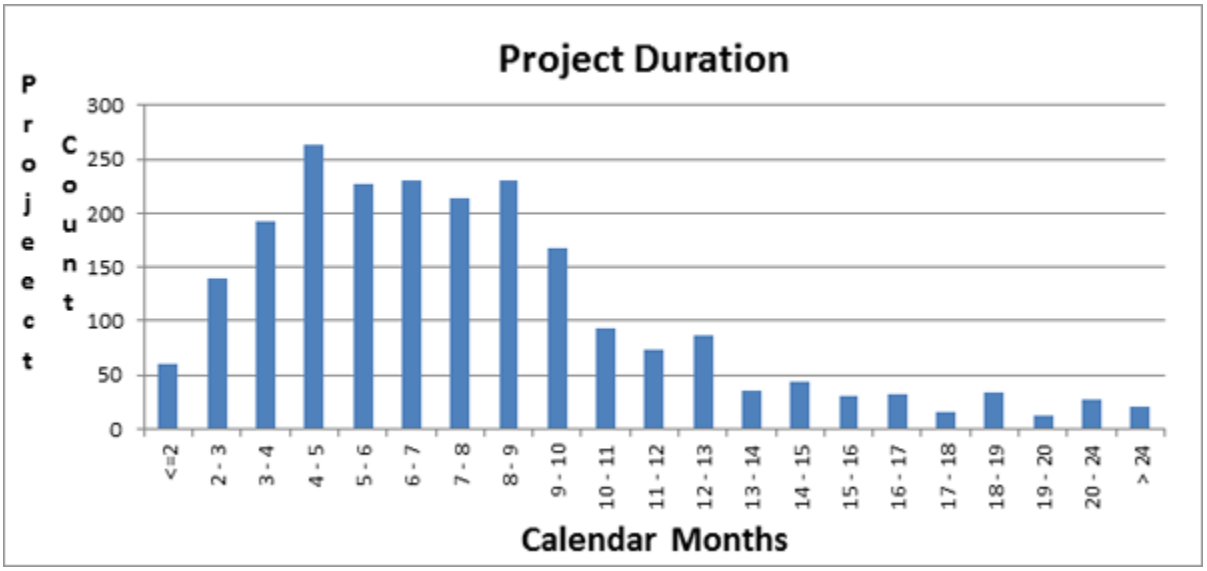

Figure 4

Perhaps an important reason why Agile has become so popular is that it works well with projects that last 7 months or less and produce about 160 function points using 30 person months of effort. Many lifecycle methodologies and process sets are too large and cumbersome to work effectively with projects where flexibility is a key requirement.

Schedule Compression and Extension

Figure 5 illustrates the relationship between schedule and size (in function points).

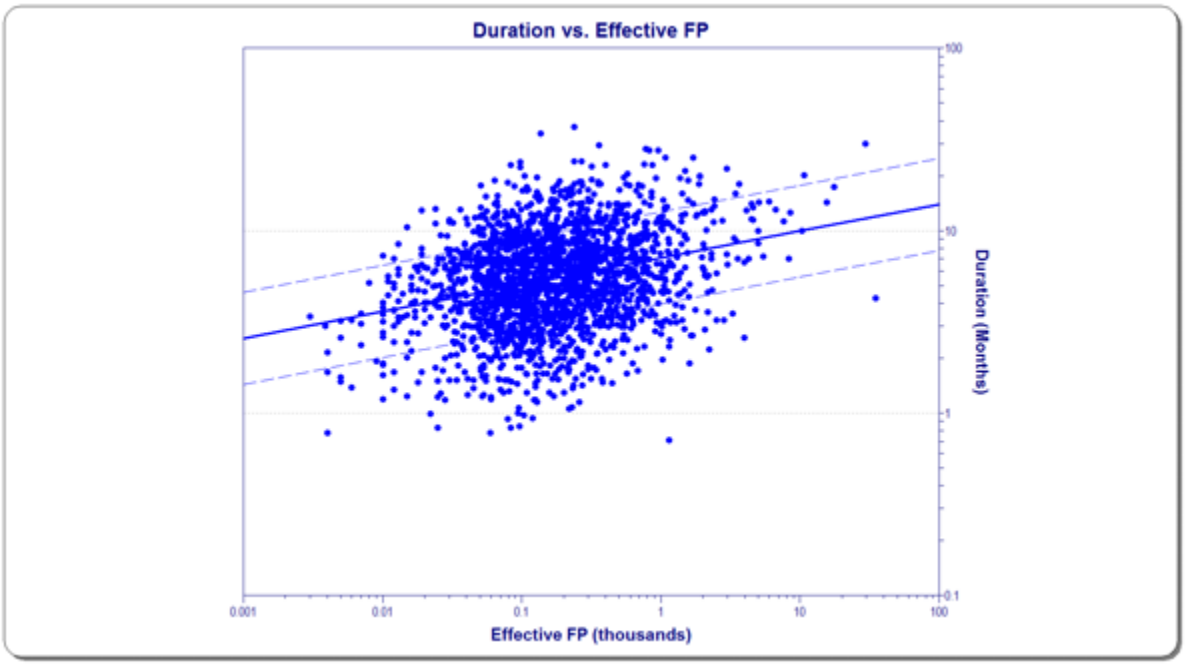

Figure 5

The solid line in the center represents average duration at various project sizes. The dashed lines above and below the average line represent plus and minus 1 standard deviation from the regression line. Roughly 2/3 of the projects are inside the dashed lines. As projects increase in size their schedules generally grow longer, but there is considerable variability: not all projects of the same size complete in the same time frame and project size is not the only factor that influences duration. Look more closely at the scale on the two axes: they are logarithmic, not linear. Simply stated, doubling the schedule does not double the output in function points. Similarly, reducing schedules by 50% does not cut output in half.

Since the relationship between size in function points and schedule is nonlinear, what happens to productivity when projects are planned with a schedule buffer? What happens when they must complete in less than average time? Figure 6 answers these questions conclusively: the more time projects are allowed, the better their productivity. While allowing a project more time than average to complete is not always an option, compressing a schedule should only be done in the full knowledge that it will lower productivity and quality while increasing cost.

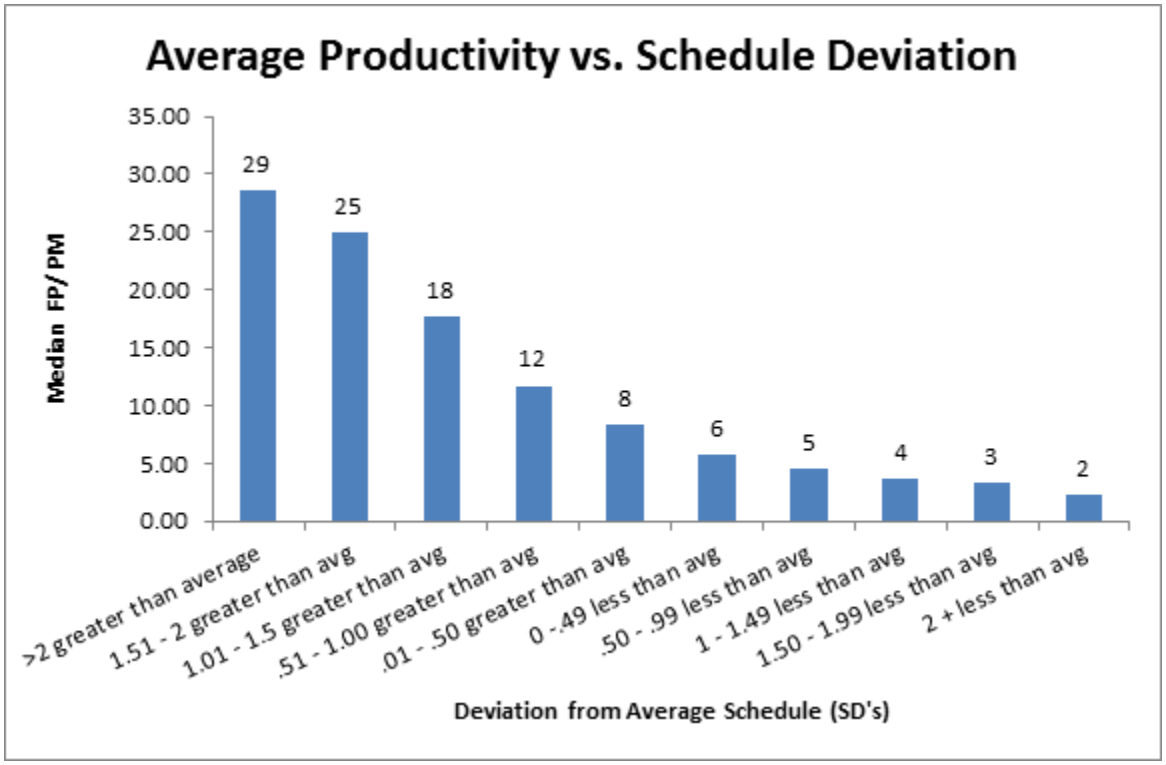

Figure 6

Impact of Analysis and Design

Several years ago QSM performed a study on how the amount of effort expended in analysis and design affects final productivity, quality, and time to market.[2] In that study, the median effort expended in analysis and design was 20% of total project effort. The projects were divided into two groups:

- Projects using more than 20% of total effort in analysis and design.
- Projects using less than 20% of total effort in analysis and design

The results were striking. Projects that spent more effort on analysis and design completed sooner, had fewer defects, and were more productive. Here we repeat that analysis using our function point sample. The results are summarized in Table 4.

Image not found or type unknown

Table 4

Once again, projects that allocated more than 20% of their effort to Analysis and Design completed sooner, expended less effort, and achieved higher productivity. They had fewer defects (and that is based on projects that averaged 9% larger). The data show conclusively that time and effort spent up front **defining** "**What** to produce" and **determining** "**How** to produce it" result in better productivity, reduced cost, higher quality, and shorter time to market. Projects that invest up front are better defined and run more smoothly: something both developers and management can appreciate.

Trends Over Time

While the focus of this paper has been the analysis of function point projects completed since the year 2000, we also looked at how FP projects have changed – and remained the same - over a longer time frame. The following tables and charts include function point projects put into production since 1990.

Software Languages

In the year 2013 very few students planning a career in information technology would pick COBOL as their first choice for a software language to learn; but the old war horse still has a significant presence. While few new development projects are coded in COBOL, enhancements to existing systems (many of them written in COBOL or PL/1) ensure that the market for COBOL programmers is still robust.

Top 10 Software Languages			
1990-1994	1995-1999	2000-2004	2005 +
COBOL	COBOL	COBOL	JAVA
PL/1	POWERBUILDER	JAVA	COBOL
NATURAL	C	PL/1	IEF/COOL:GEN
TELON	C++	C++	PL/1
SQL FORMS	VISUAL BASIC	VISUAL BASIC	Cognos Impromptu Scripts
C++	SQL FORMS	IEF/COOL:GEN	PACBASE
C	SQL	POWERBUILDER	.net
ASSEMBLER	PL/1	Oracle SQL Forms	LOTUS NOTE
CLIPPER	IEF/COOL:GEN	SQL	C++
IDEAL	ORACLE	Datastage Basic	J2EE

Table 5

The programming language rankings are based on the primary language of function points projects in the QSM database . COBOL, PL/1, and C++ have demonstrated staying power over time. Since 2000, Java has grown to be the leading software language. Powerbuilder was popular for a time; but is no more. What is not apparent from the table is the increasing number of projects that use multiple languages. Combinations such as Java for the primary and COBOL as the secondary language are found as enterprises put Web front ends on their legacy systems.

Productivity

During the 1990s, productivity, whether measured in function points per person month or by the QSM Productivity Index (PI), increased steadily. The upward productivity trend of the 90's was followed by a precipitous decrease from 2000 on. Viewed in isolation, this reversal may seem surprising. But, this trend must be viewed in the context of what changes were going on in industry trends for project size and schedule. Table 6 summarizes the productivity trend over time in the QSM function point database. To minimize the impact of outliers, median rather than average productivity has been used.

One factor contributing to the post-2000 decline in productivity has been the decrease in average project size. The median size of projects completed since 2005 is less than half of average project sizes from 1990 – 1994. As shown in Table 2, productivity increases with project size, so a significant decrease in size is accompanied by a decrease in productivity. Although it is outside of the scope of this analysis, investigating the reasons for this size decrease would make for an interesting study.

Median Productivity				
	1990-1994	1995-1999	2000-2004	2005+
FP/PM	11.10	17.00	9.21	5.84
FP/Mth	17.10	63.90	29.74	22.10
PI	15.3	16.4	13.9	10.95
Size (FP)	394.0	167.0	205	144

Table 6

Schedule and Effort

Table 7 charts project duration and effort over time. No overall trend is apparent for duration: projects from 1990 – 1994 had a significantly longer median duration than has been seen in later time periods. Median duration has varied up and down in a fairly tight range since then. The trend in effort has been continuously downward. Projects completed in the most recent time period expended 1/3 less effort than those completed from 1990 – 1994.

Median Schedule and Effort				
	1990-1994	1995-1999	2000-2004	2005 -
Duration (Mths)	10.06	6.67	6.57	7.13
Effort (Person Mths)	32.00	26.45	23.0	21.6

Table 7

Summary

Here are the most important observations we have drawn from the project data:

- **Function points have staying power.** While they are not the only sizing metric used in software, function points are widely used, especially in IT. They are used much less in telecommunications software and hardly at all in real time.
- **Function Point projects have gotten smaller.** Their median size is less than half of what it was 20 years ago.
- **Most function point projects modify existing systems.** 75% of them are enhancements.
- **Projects deliver faster and expend less effort than they did 20 years ago.** The average project from the beginning of analysis until implementation into production now lasts a little over 7 months.
- **Productivity has worsened.** Although the tools and methods available to developers are superior to those available 20 years ago, they have not improved average project productivity. The implication is that to improve productivity, the focus needs to be moved from the developers to other aspects of the software development process. One recommendation that stands out from our analysis would be to lessen schedule pressure (See Figure 7).
- **Time spent in Analysis and Design is a sound investment.** Projects that spend over 20% of their total effort in Analysis and Design complete sooner, cost less (use less effort), and have higher quality (fewer defects). Note that this item and the previous one are areas in which enlightened program and project management can directly affect productivity, cost, and quality.

[1] Crosstalk. July, 1998

[2] Using Metrics. 2007 CMMI Conference presentation