FUNCTION POINT ANALYSIS IN COMPUTER SCIENCE

CAPSTONE PROJECTS

EVELYN TORRES

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

APPROVED:

Steve Roach , Ph.D., Chair

Rodrigo Romero, Ph.D.

Eric Smith, Ph.D.

Patricia Witherspoon, Ph.D.

Dean of the Graduate School

FUNCTION POINT ANALYSIS IN COMPUTER SCIENCE CAPSTONE PROJECTS

by

EVELYN TORRES, BSCS

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

December 2010

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Dedication

I dedicate this work to my mom, for always providing me with her unconditional support, to my son and daughter for their patience and understanding during this journey, to my husband for always being by my side, to my brothers who showed me that giving up is not an option; to my cousin David who I have heard countless of times saying “You are my example” and to my nieces and nephews who I love dearly.

Acknowledgements

I would like to express my appreciation to my thesis committee: Dr. Steve Roach, Dr. Rodrigo Romero, and Dr. Eric Smith. Thank you for your time and support. Special thanks to Dr. Roach for his time and patience as we conquered my endless list of questions. Dr. Roach, it has been truly been an honor to work with you. My appreciation also goes to a wonderful woman who has been my inspiration, thank you Dr. Ann Gates for all that you do.

My appreciation and love to all my friends who supported me, especially, Iris Leony and Maria Elena Ordoñez. Thank you both for being my side and reassuring me during those times of doubt.

Abstract

Capstone projects at the Computer Science Department at the University of Texas at El Paso allow students to demonstrate their knowledge of software engineering. During the capstone project, students work on a project that simulates industry; students have a client and a supervising team. Before a project is implemented, professors must assess the project to ensure consistency with previous projects and a significant experience for the students. Currently, a project is assessed primarily on past experiences. In industry, agreeing to completing and delivering a project without assessing the scope and estimating the project cost is neither acceptable nor profitable. Similarly in academia cost estimation beneficial, and one approach is to apply cost estimation techniques. Function Point Analysis (FPA) was applied to eight past capstone projects in computer science at UTEP. The capstone projects were conducted over the past 10 years. The research demonstrated that by applying a standard cost estimation approach, capstone projects can be assessed in an algorithmic way instead of relying solely on cost estimating experience. An accurate cost estimate of a capstone project leads to a valuable learning experience for the software engineering students.

Table of Contents

List of Tables

List of Figures

1. Introduction

The capstone project course for undergraduate Computer Science (CS) at the University of Texas at El Paso (UTEP) is a two-semester sequence. The two courses are tightly coupled, with the project starting in CS 4310 and continued through CS 4311. During the course sequence, students are expected to learn software engineering fundamentals and be able to apply them. The goals of the CS capstone project are to provide students with (a) a fundamental and functional understanding of the methods, tools, and techniques required of rigorous software engineering (b) the experience of working with an actual client; (c) the ability to apply software engineering principles to a software project; (d) the ability to prepare documentation (e) the experience of working effectively in teams.

In CS 4310, project teams are formed and the project is introduced. Individual students are assessed by examinations, and the team is assessed on their progress of the assigned project through inspection of team deliverables including prototypes, modes, and Software Requirement Specification (SRS). In order for students to succeed and move on to CS 4311, they must receive a passing grade both in the exams and in the project. In CS 4311 students work on designing a solution to the assigned project as well as implementing and testing the final product.

While many projects are suggested for the capstone project, we restrict our choices to those that meet the pedagogical constraints. First, the client must truly want the software product. Involvement of the client is essential. Second, the client must be willing to wait for two semesters or more to receive completed functional software.

One of the biggest challenges of selecting a capstone project is creating a boundary or limit on client requests. It is not unusual for the project to grow as the semester progresses, and professors and teaching assistants need support in appropriately limiting the scope of the project. One way to improve

negotiations with the client on the functionality of the system to be developed for the capstone project is to have a clear measurement of what can be expected from the teams.

Once the project has been selected, a form of preliminary cost analysis is conducted. When conducting cost analysis in a capstone project the main questions are:

- Do students have enough time to complete this project?
- Is the project challenging enough to require software engineering techniques?
- Is this a reasonable project compared to past projects?

There are many benefits to accurate cost estimation, in academia, including improved project planning, better risk management, higher quality end-product, improved consistency among projects, and a higher likelihood of producing a working product.

1.1 Approach

This thesis is concerned with the application of standard industrial cost estimation techniques to previous capstone projects in the CS Department at UTEP. Eight projects were selected from the CS project repository; these projects were selected for having the most appropriate available documentation. These eight capstone projects vary in complexity and domain; however, there are similarities across the projects. All CS capstone projects are allotted the same amount of time, two semesters, from requirements elicitation to implementation. All teams have similar numbers of members, and all teams have the same resources available to them.

1.2 Contributions

The contribution of this thesis are to gauge the effort required for capstone projects from year to year, asses the level of success of the course in terms of the complexity of the projects, and to provide an algorithmic approach based on Function Point Analysis (FPA) that can be used by non-expert function point counters, e.g., teaching assistants and professors. This approach has been demonstrated on several

analyzed previous CS capstone projects, and the results of these analyses will be useful in appropriately scoping capstone projects in the future.

1.3 Guide to this Thesis

Chapter 2 presents an overview of cost estimation, the negative impact of inadequate cost estimation, and some of the approaches currently used. Chapter 3 presents a detailed explanation of how to conduct FPA. Chapter 4 presents some preliminary work conducted using different cost estimation techniques. These experiments conducted were not capstone projects, but were instead other simple projects that help explain the use of cost estimation. Chapter 5 introduces and describes the eight computer science capstone projects analyzed and the total number of function points counted for each. Chapter 6 provides a summary and interpretation of the results, advice and suggestions to anyone conducting Function Point Analysis for future CS capstone projects.

2. Cost Estimation

Software cost estimation is the process of predicting the cost to develop a software system [**1**]. Software cost estimation is no easy task and gives management many challenges. Software is the most expensive components of many computer systems, mainly because of the software size and complexity. Unfortunately, many projects go over budget and are not completed on time [**3**]. Estimates must therefore be as accurate as possible and should be made early in the process. This is exactly where the difficulty lies: accuracy of estimates depends on the quality of information coming into the estimation process. This information quality is generally poor early in the project. By the end of a project, the quality of the information is much better; however, conducting an estimate after the project has been completed is not helpful for planning purposes. In industry, accuracy is important because of monetary reasons: when bidding on a project, the contract may be lost if one proposes too high an estimated cost. On the other hand, if the estimate is too low, the company will not profit from the project, and the project may be headed for failure before it even begins.

Runaway projects are defined as those projects that spiral out of control **[5]**, i.e. fail to produce the desired product or end up producing the product well over budget and schedule. There are many studies that assess the losses brought by software failures. These studies agree on the range is somewhere between $50 to $80 billion dollars annually **[6]**. Standish **[7]** defines successful projects as those projects that are on budget, on time and deliver the expected functionality; challenged projects are those that are completed and operational but over-budget, over- schedule and offers less features than originally requested; failed projects are those cancelled sometime during the development cycle. The Standish Chaos Reports **[7]** indicate that for the past ten years, the fraction of projects that are challenged or failed has hovered around two-thirds, as shown in Table 1.

Table 1- Standish Chaos Reports Results

	1994	1996	1998	2000	2002	2004	2009
Failed Projects	31%	40%	28%	23%	15%	18%	24%
Challenged Projects	53%	33%	46%	49%	51%	53%	44%
Successful Projects	16%	27%	26%	28%	34%	29%	32%

According to Glass **[5]** the most two common causes of runaway projects are poor estimation and unstable requirements. Even though a loss tends to be related to a dollar amount, the losses are not always monetary. In some cases the loss involves time or human lives. When looking at past software failures it may be difficult to pinpoint exactly what the error might have been; however, accurate cost estimation can result in better planning and more adequate testing, which leads to better overall software quality.

2.1. Examples of Failures due to Inadequate Cost Estimation

In the following discussion three infamous software failures are presented.

The Denver International Airport Baggage System

The Denver International Airport baggage system was intended to bring a major improvement to baggage handling in 1994. This system was to have a computer tracking system to automatically direct baggage to its appropriate locations by loading them on unmanned carts. The system was supposed to bring many benefits including: reducing flight delays, shorter wait time at luggage carousels and reduced labor costs **[15]**. The project went well over schedule and budget. The system originally was supposed to be completed in March 1994 **[14]**. Instead the project was never completed, and it delayed the opening of the airport until February 1995. The airport opened with separate baggage system for three concourses instead of the single system for all three as originally planned. In August 2005 it became known that the system would be abandoned mainly because modification and repairs were

costing an estimated $1 million per day. The total loss caused by this failure is estimated to be $200 million.

The IRS Tax Systems Modernization (TSM)

The IRS TSM project was launched in 1986 and its goal was to make the transition to electronic filing, to replace paper documents with digital ones, and to have a unified overall system. The project would be costly, but the benefits would be many. Things did not go as planned. According to Edward Cone in a 1996 article **[16]** some of the problems with the project included: the lack of an overall plan for TSM and the IRS' unwillingness to outsource tasks for which it lacks expertise. Based on research, the fault lies in a lack of management of IT projects skills. The IRS was inexperienced in managing large-scale and long-term IT projects and there was little to no organization or prioritization: As a result the project was headed for failure almost from the beginning. The project was in 1997 with a total loss of approximately $3.5 billion dollars.

FBI Virtual Case File

The Virtual Case File (VCF) was a software project started in 2000; the VCF's contractor was Science Applications International Corp. (SAIC). The purpose of this system was to automate the paperwork environment the FBI was operating under [**23**]. Instead of, it is now known as one of most highly publicized software failure in history. Originally the project was supposed to take three years and cost $380 million **[22],** and its main purpose was to update the bureau's IT infrastructure**.** After 9/11, increased pressure on the FBI to address the information sharing issue resulted in even higher expectations on the VCF system. The scope of the project was changed in December 2001 to a complete replacement of all previous systems and the migration of the entire existing database into an Oracle database. Signs of trouble started emerging in December 2002 when the FBI asked for additional funding from the US congress to attempt to salvage the project, In May 2004 $16 million was paid in,

and $2 million more were spent on external reviews. The findings of these reviews were that the project had failed. The project was officially ended in January 2005 and caused a loss of approximately $100 million dollars [**23**].

2.2. Software Sizing Approaches

Software sizing refers to predicting the volume of the software application to be developed **[21]**. Sizing alone is not considered a cost estimation approach, but rather it is a way of giving the development team an idea of what lies ahead. To explain the concept of software sizing, Jones compares it to knowing the square footage of a house to be built. In addition to the square footage materials, floor plans, and specifics to the building site must also be considered in order to achieve an accurate building schedule and budget.

Lines of Code (LOC) refer to the number of lines of source code in the software, excluding blank lines and lines used for comments. LOC was first introduced in 1960[**19**], and at that time it was used for three different purposes. First, the economics of applications were measured using "dollars per LOC". Second, it was a measure productivity, given in "LOC per development unit". Lastly, LOC was used to measure quality, given in terms of "defects per KLOC (thousand lines of code)" [**19**]. In the 1960s most of the software developed was built using basic assembly language. During these years when assembly language was dominant LOC served its purpose well enough.

LOC counting can be conducted at the completion of the implementation; however, it has been suggested that using LOC counting for measuring productivity yields incorrect results. Evidence of this was first seen in the 1970s. By this time the first generation of higher level languages became widely available [**19].** First signs of inaccuracy surfaced when IBM, who had been applying LOC counting, noticed a significant schedule and budget overrun. One of the possible reasons identified was that coding in assembly language meant 90% of the total effort would go directly into coding. However, when coding in higher level languages, the coding effort was reduced, but not necessarily the total effort.

Even though LOC counting may be considered problematic, it is still widely used **[18, 19]**. One of the ways LOC is still used now is as an input to other estimation tools such as COCOMO (described in Section 2.3).

2.3. Software Costing Approaches

In contrast to software sizing, *software cost estimation* refers to costing software by building a schedule, allocating resources, and creating a budget. Currently there are many approaches to cost estimation, from judging from experience to sophisticated software tools that take many cost factors into consideration. The estimate obtained as a result of the different cost estimation models can yield estimates in person-months, in calendar time or in a dollar amount. Judging costs from previous and similar project experiences could be quite helpful; however, it is not sufficient. Project managers are highly likely to work on a wide range of projects; therefore, finding an expert for every different project will be quite a challenge. Another problem with the expert judgment approach is that it is not an algorithmic approach: it may not be repeatable. More sophisticated approaches to cost estimation include the use of models such as the Constructive Cost Model (COCOMO) among many others.

There are two types of cost estimation models, algorithmic and non-algorithmic. Algorithmic models are based on mathematical computations such as statistics, standard deviations, regression models, and/or differential equations **[1].** Non-algorithmic models are not clearly outlined and require explicit knowledge and data of similar projects completed in the past. COCOMO is an algorithmic cost estimation model. While there are different approaches to conducting cost estimation, some may be more appropriate for certain projects than others; each approach has its own set of advantages and disadvantages.

2.3.1. Experience and Expertise

Experience plays an important role in any estimate; however, it is not sufficient as a method of conducting cost estimation. There are many reasons why experience alone is not enough. First it is unlikely that one person will have experience in many major projects, and there are many cost factors involved in similar projects that may in fact be very different. It is also unclear what makes an expert: how many projects constitute sufficient experience? If a project manager had been on a similar project before, then using the expert judgment approach would have some advantages. It would be almost effortless to conduct the estimate and the chances of it being accurate would rise. Expertise is probably the most widely used approach when conducting cost analysis in capstone projects.

2.3.2. COCOMO I

COCOMO [**8**] is a cost estimation model developed by Barry Boehm in 1981. The original model uses a basic regression formula. The formula uses parameters that are derived from historical project data and current project statistics. COCOMO 81, as the first COCOMO model is known, is meant to provide quick estimates for small to medium projects. This model calculates effort based on program size, or lines of code. However, COCOMO 81 does not account for other important cost factors and as a result may lead to a less than accurate estimate. COCOMO 81 operates on 3 different project types: Organic, Semi-Detached, and Embedded projects. Organic projects referred to those with small and experienced development teams. Semi-detached projects are those with medium, not-so-experienced teams. Embedded projects are those that are developed within a set of rigid hardware, software and regulations constraints [**9**].

2.3.3. COCOMO II

COCOMO II was released as an extension to the original COCOMO and was released in the 1990s [**10**]. This extension to the previous model was meant to accommodate changes to development

techniques over the years. COCOMO II computes the estimate as a function of program size and a set of cost drivers. In total COCOMO II uses 15 attributes which are ranked on a six point scale from very low importance to extra high importance. Each attributes is ranked according to the following table [**25**]:

Table 2- COCOMO II Attribute Multiplier Table

	Very Low	**Low**	**Nominal**	**High**	**Very High**	**Extra High**
Product Attributes						
Required software reliability	0.75	0.88	1.00	1.15	1.40	
Size of application database		0.94	1.00	1.08	1.16	
Complexity of the product	0.70	0.85	1.00	1.15	1.30	1.65
Hardware Attributes						
Run-time performance constraints			1.00	1.11	1.30	1.66
Memory constraints			1.00	1.06	1.21	1.56
Volatility of the virtual machine environment		0.87	1.00	1.15	1.30	
Required turnabout time		0.87	1.00	1.07	1.15	
Personnel attributes						
Analyst capability	1.45	1.19	1.00	0.86	0.71	
Software engineer capability	1.42	1.17	1.00	0.86	0.70	
Virtual machine experience	1.21	1.10	1.00	0.90		
Programming language experience	1.14	1.07	1.00	0.95		
Project Attributes						
Application of software engineering methods	1.24	1.10	1.00	0.91	0.82	
Use of software tools	1.24	1.10	1.00	0.91	0.83	
Required development schedule	1.23	1.08	1.00	1.04	1.10	

Each attribute is assigned a value then the product of all of them leads to a single value. Using this value effort is estimated using the following formula:

$$\mathbf{E(in\ person\text{-}months\ )} = a_i (\mathbf{KLoC})^{(b_i)} . \mathbf{EAF}$$

E is the effort (in person-months), KloC is the estimated number of lines of code (in thousands) and EAF is the value obtained in the step explained above, a_i & b_i are values given in a second table following table:

Table 3- Types of Projects in COCOMO

Software Project	a_i	b_i
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

Organic, Semi-detached and embedded refer to the modes described in COCOMO 81. Once one has the all the variables the calculation can be performed to obtain the effort required in the form of person-months.

3. Function Point Analysis (FPA)

FPA can be used as either a software sizing or costing estimate approach. FPA evaluates a system based on the capabilities provided to the user. As a sizing approach, a number of function points can lead to estimated lines of code. To use FPA as a costing approach, some historical data is needed to determine the average number of function points implemented for similar projects.

Function Point Analysis measures software by counting the functionality provided to the user, who is sophisticated, someone who would understand the system from a functional perspective. A software system can be defined as a set of processes or transactions. There are two basic types of elementary processes (data in motion and data at rest) in a software application [**1**]. A Function point or a transaction is a unit of measure to represent the size of an application [**11**].

3.1. Identifying Function Points

There are five types of transactions in FPA: External Inputs (EI), External Outputs (EO), External Inquiry (EQ), Internal Logical Files (ILF), and External Interface Files (EIF).

3.1.1. External Inputs

An external input (EI) is an elementary process in which data crosses the boundary from outside to inside, such as:

- Data entering the system
- Screens, forms, dialogs, controls
- User or other program adds, deletes, modifies data
- Any input that requires processing logic
- Application reading a table in a database

There are different ways to identify EI's, depending on what documentations or models may be available. When documents such as the requirements specifications are available, one way to identify EI's is by looking for keywords. The list of keywords often associated with EI's is provided below:

Table 4-Keywords Associated with EI

Add
Activate
Amend (change and delete)
Cancel
Change
Convert (change)
Create (add)
Delete
Deassign
Disable
Disconnect (change or delete)
Enable
Edit (change)
Insert (add and change)
Maintain (add, change, or delete)
Memorize (add)
Modify (change)
Override (change)
Post (add, change and delete)
Remove (delete)
Reactivate (change)
Remit
Replace (change)
Revise (change and delete)
Save (add, change or delete)
Store (add)
Suspend (change or delete)
Submit (add, change or delete)
Update (add, change or delete)
Voids (change and delete)

3.1.2. External Outputs

External Outputs (EO) are pieces of derived data leaving the system such as:

- Screens
- Reports

- Dialog boxes
- Control Signals

Just as EI's EO's can be identified from requirement specification documents. The list of keywords often associated with EO's is provided below:

Table 5-Keywords Associated with EO

Browse , Display , Get , On-lines , Output , Print , Query , Reports , Request , Retrieve , Seek , Select , View

3.1.3. External Inquiries

External Queries (EQ) are processes in which data is retrieved from either internal or external data sources. The output side of EQ does not contain any derived data, in other words the data is outputted just as retrieved. There are no logical functions performed on the data. The list of keywords often associated with EQ's is provided below:

Table 6-Keywords Associated with EQ

Browse Display Extract Fetch Find Gather Get Drop Down Lists

Look Ups On-lines Output Pick Lists Print Query Scan Seek Select Show View Reports

3.1.4. Internal Logical Files

Internal Logical Files (ILF) are groups of data maintained within a system [**1**]. These are easier to find looking at a database schema or a data flow diagram (DFD). Unlike, the three previous types of functions points described there are no keywords typically associated with ILF's. These can be visually identified from models or from database requirements in a software requirements specification document.

3.1.5. External Interface Files

External Interface Files (EIF) are groups of data maintained outside of the system being analyzed **[1]**. The data is maintained by another application and is used by the system being analyzed.

3.2. Assigning Complexity Function Points

After each of the function points have been identified and categorized into the five categories (EI, EO, EQ, ILF, and EIF), and each category is assigned a complexity. The complexity is assigned by counting the following:

- Data Element Type (DET)- A unique, user-recognizable, non- recursive, field;
- File Type Reference (FTR)- A file referenced by transaction, an FTR must also be an internal logical file or external interface file; and

- Record Element Type (RET)- A user recognizable subgroup of data elements within an ILF or EIF

Once each of the transactions has been identified and the corresponding elements have been counted, each transaction is assigned a complexity of low, average, or high. In the following tables, the number of function points is given in parentheses.

Table 7 - External Input Complexity Rating Table

Files Type Referenced (FTR)	**Data Elements**		
	1-4	5-15	Greater than 15
Less than 2	Low (3)	Low (3)	Average (4)
2	Low (3)	Average (4)	High (6)
Greater than 2	Average (4)	High (6)	High (6)

Table 8 - External Output Complexity Rating Table

File Types Referenced (FTR)	**Data Elements**		
	1-5	6-19	Greater than 19
less than 2	Low (4)	Low (4)	Average (5)
2 or 3	Low (4)	Average (5)	High (7)
Greater than 3	Average (5)	High (7)	High (7)

Table 9- External Inquiry Complexity Rating Table

File Types Referenced (FTR)	**Data Elements**		
	1-5	6-19	Greater than 19
less than 2	Low (3)	Low (3)	Average (4)
2 or 3	Low (3)	Average (4)	High (6)
Greater than 3	Average (4)	High (6)	High (6)

Table 10- Internal Logical Files (ILF) Complexity Rating Table

File Types Referenced (FTR)	**Data Elements**		
	1-5	6-19	Greater than 19
less than 2	Low (3)	Low (3)	Average (4)
2 or 3	Low (3)	Average (4)	High (6)
Greater than 3	Average (4)	High (6)	High (6)

Table 11- External Interface File (EIF) Complexity Rating Table

File Types Referenced (FTR)	Data Elements		
	1-5	6-19	Greater than 19
less than 2	Low (3)	Low (3)	Average (4)
2 or 3	Low (3)	Average (4)	High (6)
Greater than 3	Average (4)	High (6)	High (6)

After each function point is assigned a complexity and it is determined how many points should be assigned to each function point, the sum of these points is called the unadjusted function point count (UAF).

3.3. Calculating the Value Adjustment Factor (VAF)

Once the total unadjusted function point count is obtained, the Value Adjustment Factor (VAF) is calculated. The VAF is used to cover the system characteristics not measured in the five different types of transactions previously mentioned. All the characteristics are presented in the table below:

Table 12-General System Characteristic Brief Description

Characteristic	Description
Data communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
Distributed data processing	How are distributed data and processing functions handled?
Performance	Did the user require response time or throughput?
Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?
Transaction rate	How frequently are transactions executed daily, weekly, monthly, etc.?
On-Line data entry	What percentage of the information is entered On-Line?
End-user efficiency	Was the application designed for end-user efficiency?
On-Line update	How many ILF's are updated by On-Line transaction?
Complex processing	Does the application have extensive logical or mathematical processing?
Reusability	Was the application developed to meet one or many user's

	needs?
Installation ease	How difficult is conversion and installation?
Operational ease	How effective and/or automated are start-up, back up, and recovery procedures?
Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?

Each of the categories is assigned a degree on importance on a scale of 0 to 5, 0 meaning no influence and 5 being strong influence. The scale is broken down into more detail in the following table:

Table 13- General Characteristics Influence

0	Not present, or no influence
1	Incidental influence
2	Moderate Influence
3	Average Influence
4	Significant Influence
5	Strong Influence throughout

Once all the questions have been answered and a scale has been assigned to each category the following formula is used to obtain the VAF:

$$VAF= (65+TDI)/100$$

(Where TDI is the sum of the results from each question)

3.4. Calculating the final function point count

The final function point count is obtained by:

$$FP= UAF * VAF$$

where UAF is the unadjusted function point count & VAF is the value adjustment factor.

4. Preliminary Comparison of Cost Estimation

In an effort to calibrate our understanding of cost estimation, and to confirm our approach, we conducted a small comparison of cost estimation techniques based on results reported in the literature.

4.1. Experiment 1- COCOMO I vs. COCOMO II

The experiment is based on the following description on an online project **[24].**

> A bioinformatics company, providing advanced methods for data mining of genetic information, intends to construct a distributed application for analysis and navigation of biological networks. As part of this project, a database provider that exposes simple interfaces to UI programmer and hides complexities of the data layer should be build. As soon as the scope of this task is broadly defined as such, it is sliced into a separate project.
>
> The company has already made work on inception phase, and provided the document describing the project concept. The customer has the following preferences:
>
> 1. Transfer existing SQL Server database (~1 GB) hosted on Windows to PostgreSQL data end hosted on Linux.
> 2. Build components for this project using Java. The main component is database provider.
> 3. Since the project is small, the elaboration phase (or detailed design phase) is not necessary.
> 4. Project has time constraints.

The author makes the cost estimate using the COCOMO II, but in order to do that, LOC must be obtained. In this particular case the project has already been completed, and the LOC are counted using a code counter program. The results are as follows:

Table 14- LOC Count for Experiment 1

Folder	Total LOC
SQL Files	414
Java DB Provider Files	345
Java Servlet	156
Web Files	113

The COCOMO II software package, use Costar 7.0, was used to compute costs. Many of the cost factors were left at their nominal value of 1, but some of them where adjusted, such as the cost factor pertaining to database size. The results according to the author lead to estimated project duration of 4.6 months and project cost of $23,000(assuming a $5000 monthly developer salary). To compare COCOMO II to COCOMO 81, we used the project data in the COCOMO 81 formula:

$$\text{Effort Applied} = a_b(\text{KLOC})^{b}{}_{b} \text{ [man-months]}$$

We used the coefficients for a semi-detached project with average level of experience, based on the author's statement that the development team was familiar with the basic concepts of software engineering. For a semidetached project, the coefficients $a_b = 3.0$ and ${}^{b}{}_{b} = 1.12$, the LOC provided in the article were 1028. The obtained estimate using the COCOMO 81 formula was: estimated project duration of 9.45 months and project cost of $47, 250, twice the COCOMO II estimate. Depending on the size of the company, the difference between these two estimates may be significant. If this information were to be used for contract bidding purposes, then a $24,250 difference is significant and should not be taken lightly.

4.2. Experiment 2- Function Point Analysis vs. COCOMO I and COCOMO II

Next we replicated a completed function point counting exercise conducted by Shivprasad Koirala **[25]** and used his results to create two separate estimates. Koirala conducted a function point count on a system solely based on the user interface described for the system. After completing the entire function point counting exercise, he arrives at a total of 21 function points. Assuming the

implementation language will be Java, that is a total of 1155 lines of code. He estimates that three function points can be completed per day, but it is unclear whether this is based on historical data or on personal judgment. I then took this information used COCOMO 81 and COCOMO II online tools [**28, 29**]. As previously mentioned, COCOMO takes different cost factors into consideration. COCOMO II accounts for many more cost factors than COCOMO 81. With so many factors, the combinations are many; our approach was to conduct three runs in each model. In the first run all the factors were set to low significance, in the second run the factors were set to medium then finally set to high. We assume the tools provide correct results. The results were as follows:

Table 15- Cost Estimation for Experiment 2

Cost Estimation Model	**Effort (man-days)**	**Cost (@ $5,000 monthly salary)**
Function Point Counting	7 days	$1,166.00
COCOMO 81 (run 1)	3.89 months	$19,450.00
COCOMO 81 (run 2)	2.19 months	$10,950.00
COCOMO 81 (run 3)	3.28 months	$16,400.00
COCOMO II (run 1)	3 months	$15,000.00
COCOMO II (run 2)	2 months	$10,000.00
COCOMO II (run 3)	4 months	$ 20,000.00

The difference between the estimates is quite significant and confusing. The estimated cost based on the results above ranges from $1,100 to $20,000 that is quite a wide range.

4.3. Preliminary Comparison of Cost Estimation-Conclusions and Findings

We draw several conclusions from these exercises. Even when not using expert judgment as an approach, experience is beneficial. Each of the models described have their advantages and disadvantages. COCOMO takes many cost factors into consideration, some that are ignored by FPA, and as a result may lead to a more accurate estimate. Just as with cost estimation in industry, when conducting cost estimates in academia, sufficient and accurate historical data are critical. In conclusion,

as with many other things there is no single best approach to cost estimation, and any approach must be validated.

5. Function Point Analysis for UTEP CS Capstone Projects

In this section, we describe the application of FPA to CS capstone projects. Eight projects were analyzed by counting function points based on the documentation and prototypes for each project. Initially, we attempted to count function points from the Software Requirements Specification (SRS) documents. However, this proved difficult since there is no direct translation between function points and requirements. Identifying keywords associated with function points in the SRS was somewhat helpful. The structure of the IEEE standard SRS [**30**] sometimes encourages the disaggregation of specific features, leading to over-counting of the feature. This in turn leads to an inaccurate count of function points. For the projects that had prototype documentation, the process of counting function points from the prototypes proved much more efficient. We also identified function points from the test plan document. Since the test plan contains the possible inputs as well as desired and actual outputs, this has been the most successful approach.

Three of the projects are related to the Weather Station Project implemented in Fall 2009/Spring 2010: 1.Statistics, Correlation and Trend-Analysis; 2.Weather History; and 3.Current Weather Data. The Weather Station Project focused on gathering and analyzing climate change data. The motivation behind the project was recent drastic climate changes. The next three projects are related to the Automatic Weather Station Database (AWSD) implemented in Fall 2008/Spring 2009: 1.Search, View, Analyze; 2.Upload Data; and 3.Data Analysis. AWSD was similar to the Weather Station Project. The AWSD system was developed to provide support for the collection, assessment, distribution, reduction, and manipulation of weather and climate data. AWSD gave users access to search for data, perform simple data reduction (e.g. computing the average daily temperature or total monthly rainfall), and perform several data analysis operations such as plotting the average temperatures at a variety of sites. The next project is the Meteorological, Image, and Environmental Data Repository System (MInER) implemented

in Fall 2007/Spring 2008. MInER also involved the collection and analysis of weather data. One of the differences of MInER was that the data was collected by robotic trams set up in different areas of the world. The tram system was capable recording environmental data at regular intervals for a total of 300 meters. The data collected consisted of spectral readings, ground temperature, ground moisture, wind velocity, light spectrum, and temperature under the shade and photographs. The last project evaluated was the Environmental Observatory System (EOS) implemented in Fall 2006/Spring 2007. The EOS, just as many of the projects described previously, involved the collection of environmental data read from sensors placed in a remote field setting. EOS also allowed users to study the environmental data by performing analysis on them.

5.1. Statistics, Correlation and Trend-Analysis System (SCATS)

SCATS was a subsystem of the Weather Station Project. SCATS will allow users to conduct a variety of descriptive statistical analyses on gathered data, plot the results of the statistical analyses, conduct correlation and trend analysis on the data gathered by numerous weather stations, and generate a map extrapolation based on available data.

Main features of SCATS provided to users are:

- Create Data Table
- Modify Data Table
- Calculate Descriptive Statistics
- Generate Extrapolated Map
- Plot Data

Function Point Counting for SCATS

After conducting FPA on the project the results were the following:

Table 16- UAF for SCATS

Elementary Process	Type of Function Point	Complexity	UAF

Create Data Table	External Input (2 DET's) 1 FTR -> 1 ILF	Low Low	3 7
New Table Interface (Part of Create Data Table)	External Input (3 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Create by Station Id	External Input (8 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Select Instrument Types (Part of Create Data Table)	External Input (8 DET's)	Low	3
Select Time period (Part of Create Data Table)	External Input (11 DET's)	Low	3
Input Table Name (Part of Create Data Table)	External Input (2 DET's)	Low	3
Modify Table (Data Reduction)	External Input (5 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Merge Tables	External Input (4 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Display Table Results	External Output (2 DET's)	Low	4
Calculate Descriptive Statistics	External Input (4 DET's)	Low	3
Display Statistics Result	External Output (2 DET's)	Low	3
Plot Data	External Input (2 DET's)	Low	3
Display Plot Results	External Output (3 DET's)	Low	4
Generate Extrapolated Map	External Input (3 DET's)	Low	3
Display Extrapolated Map Results	External Output (1 DET's)	Low	4
Total Unadjusted Function Point Count			**83**
VAF			**.87**
Total Function Point Count			**72.21**

5.2. Weather History

Weather History (WH) was a subsystem of the Weather Station Project. The purpose of WH was to provide scientists with enough information to understand the changes in the climate. The WH system

provided access to historical data from data sources such as National Oceanic and Atmospheric Administration (NOAA). WH was designed to search for historical weather data specified by a list of weather stations, types of weather data, and a time range supplied by a user.

Main Features of Weather history provided to users:

- Acquire Historical Weather Data
- Acquire list of Weather Stations
- View Activity log
- Search Activity log

Function Point Counting for Weather History

After conducting Function Point analysis on the project the results were the following:

Table 17- FPA results for Weather History

Elementary Process	**Type of Function Point**	**Complexity**	**Unadjusted FP count**
Log-in	External Input (2 DET, 1 FTR) 1 FTR -> 1 ILF	Low Low	3 7
Acquire Historical Data	External Input (5 DET, 1 FTR) 1 FTR -> 1 ILF	Low Low	3 7
Acquire list of Weather Stations	External Inquiry (5 DET, 1 FTR) 1 FTR -> 1 ILF	Low Low	3 7
View Activity Log	External Output (6 DET, 1 FTR) 1 FTR -> 1 ILF	Low Low	4 7
Search Activity Log	External Input (4 DET, 1 FTR) 1 FTR -> 1 ILF	Low Low	3 7
Total Unadjusted Function Point Count			**51**
VAF			**.82**
Total Function Point Count			**41.82**

5.3. Current Weather Data (CWD)

CWD was a subsystem of the Weather Station Project; CWD provided access to current climate data from weather data sources such as Weather Underground.

Main Features of CWD provided to the users

- Install Client program
- Configure Client program
- Acquire Current Weather Data
- Get list from Server
- Get data from weather data source
- Configure server program
- Add Weather Station
- View Activity Log
- Acquire Current Weather Data

Function Point Counting for Current Weather Data

After conducting Function Point analysis on the project the results were the following:

Table 18- FPA results for Weather Data

Elementary Process	**Type of Function Point**	**Complexity**	**Unadjusted FP count**
Install Client program	External Input (1 DET)	Low	3
Configure Client program	External Input (5 DET, 1 FTR) 1 FTR -> 1 ILF	Low Low	3 7
Acquire Current Weather Data (Client)	External Input (1 DET, 1 FTR) 1 FTR -> 1 ILF	Low Low	7 3
Get list from Server	External Inquiry (3 DET, 1 FTR) 1 FTR -> 1 ILF	Low Low	3 7

Get data from weather data source	External Output (1 DET, 1 FTR) 1 FTR -> 1 ILF	Low Low	4 7
Configure server program	External Input (5 DET, 1 FTR) 1 FTR -> 1 ILF	Low Low	3 7
Add Weather Station	External Input (5 DET, 1 FTR) 1 FTR -> 1 ILF	Low Low	3 7
View Activity Log	External Input (1 DET)	Low	3
Acquire Current Weather Data (Server)	External Output (1 DET, 1 FTR) 1 FTR -> 1 ILF	Low Low	4 7
Total Unadjusted Function Point Count			**78**
VAF			**.82**
Total Function Point Count			**63.96**

5.4. Search, View, Analyze

Search, View, Analyze was a subsystem of the Automatic Weather Data Station (AWSD) project. This system allowed for the storing and retrieving of data collected from various weather stations. The need for this system was to have a web-based environment to easily access the weather data collected from various data loggers that are set up.

Main Features of Search, View, and Analyze provided to users:

- Set Account Settings
- Search Data
- Display Search Results
- Analyze Data (table Form)
- Display Table
- Analyze Data (Graph Form)
- Display Graph

Function Point Counting for Search, View, Analyze

After conducting Function Point analysis on the project the results were the following:

Table 19- FPA results for Search, View, Analyze

Elementary Process	**Type of Function Point**	**Complexity**	**Unadjusted FP count**
Set Account Settings	External Input (8 DET's, 1 FTR) 1 FTR -> 1 ILF	Low Low	3 7
Search Data	External Query (18 DET's, 3 RET's)	High	6
Display Station Picture and Info (part of Search Data Interface)	External Query (4 DET's)	Low	3
Display Search Results	External Output (6 DET's)	Low	4
Analyze Data (Table Form)	External Input (15 DET's)	Average	4
Display Table Results	External Output (6 DET's) 1 FTR -> 1 ILF	Low Low	4 7
Analyze Data (Graph Form)	External Input (11 DET's)	Low	3
Display Graph Results	External Output (4 DET's) 1 FTR -> 1 ILF	Low Low	4 7
Total Unadjusted Function Point Count			**52**
VAF			**.82**
Total Function Point Count			**42.64**

5.5. Upload Data

Upload Data was a subsystem of the Automatic Weather Data Station (AWSD) project This project is a subproject of the Weather Station Database project. This sub-project focused on the function to allow the users of the system to upload and store data in the database.

Main Features Upload Data provided to users:

- Provide user the interface to upload data
- Identify the list of weather stations associated with the user

- Display the list of weathers stations associated with the user
- Check quality of data(this feature did not end up being implemented)
- Store data in the database

Function Point Counting for Upload Data

After conducting Function Point analysis on the project the results were the following:

Table 20- FPA results for Upload Data

Elementary Process	**Type of Function Point**	**Complexity**	**Unadjusted FP count**
Upload Data Interface	External Input (5 DET's, 1 FTR) 2 FTR -> 2 ILF	Low Low	3 14
Query Uploaded Data	External Input (21 DET's, 1 FTR) 1 FTR -> 1 ILF	Average Low	4 7
Identify list of weather stations	External Query (15 DET's) (1 FTR -> 1ILF)	Low Low	3 7
Create Weather Station Profiler	External Input (10 DET's, 1 FTR) 1 FTR -> 1 ILF	Low Low	3 7
Display list of weather stations	External Output (3 DET's) 1 FTR -> 1 ILF	Low Low	4 7
Total Unadjusted Function Point Count			**59**
VAF			**.87**
Total Function Point Count			**51.33**

5.6. Data Analysis for AWSD

This project is a subproject of the Weather Station Database project. This sub-project focused on the function to allow the users to conduct statistical analysis on the weather data collected.

Function Point Counting for Data Analysis

After conducting Function Point analysis on the project the results were the following:

Table 21- FPA results for Data Analysis

Elementary Process	**Type of Function Point**	**Complexity**	**Unadjusted FP count**
Analysis Interface	External Input (30 DET's) (1 FTR -> 1ILF)	Average Low	4 7
Connect to R-to do analysis	External Inquiry (> 19 DET's)	Average	4
Display Error Messages (x3)	External Inquiry (> 3 DET's)	Low	9
Display Analysis Result	External Output (3 DET's)	Low	4
Data Reduction	External Input (11 DET's) (1 FTR -> 1ILF)	Average Low	4 7
Download Data	External Input (5 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Total Unadjusted Function Point Count			**49**
VAF			**.87**
Total Function Point Count			**42.63**

5.7. Meteorological, Image, 'n Environmental Data Repository System (MInER)

MInER was designed to collect weather data by using robotic trams equipped with sensors. The data was then stored for future retrieval and analysis. The data was then stored for future analysis.

Main Features of MInER provided to users:

- Register/Log In
- Retrieve Data
- Analyze Data
- Merge Data
- Submit Data
- Manage Users
- Manage Data

- Manage Sites

Function Point Counting for MInER

After conducting Function Point analysis on the project the results were the following:

Table 22- FPA results for MInER

Elementary Process	**Type of Function Point**	**Complexity**	**Unadjusted FP count**
Register	External Input (15 DET's, 1 FTR) 1 FTR -> 1 ILF	Low Low	3 7
Login	External Input (3 DET's, 1 FTR) 1 FTR -> 1 ILF	Low	3
Retrieve Data	External Input (21 DET's)	Average	4
Analyze Data	External Input (3 DET's)	Low	3
Merge Data	External Input (3 DET's)	Low	3
Submit Data	External Input (7 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Manage Users	External Input (5 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Manage Data	External Input (4 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Manage Sites	External Input (3 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Download Data	External Input (5 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Plot Data	External Input (4 DET's) 1 FTR -> 1 ILF	Low Low	3 7
View Data	External Input (5 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Total Unadjusted Function Point Count			**93**

The MInER project was not divided into subprojects; instead the entire functionality was assigned to all project teams. Different teams were able to implement different functionality of the project. The following table shows which functionality was completed by each of the four teams:

Table 23- FPA results for each MInER team

Elementary Process (UFP/Process)	**Team 1**	**Team 2**	**Team 3**	**Team 4**
Register /10	Y	Y	Y	Y
Login/3	Y	Y	Y	Y
Retrieve Data/4	Y	Y	Y	Y
Analyze Data/3				
Merge Data/3				
Submit Data/10	Y		Y	
Manage Users/10				Y
Manage Data/10				
Manage Sites/10				Y
Download Data/10	Y			
Plot/10				
View/10	Y		Y	
Total Unadjusted Function Point Count	**57**	**17**	**37**	**37**
VAF	**.87**	**.87**	**.87**	**.87**
Total Function Point Count	**49.59**	**14.79**	**32.19**	**32.19**

The data for team 2 was found to be insufficient and possibly inaccurate due to incomplete documentation. To avoid a negative impact on the final results the data for team 2 was discarded.

5.8. Environmental Observatory System

The Environmental Observatory System (EOS) system was designed to collect environmental data, and storing the data for future retrieval and analysis.

Main Features of EOS provided to users:

- Retrieve Data (by stations or from a map)
- Store Data
- Readout Data
- Deploy logger

- Manage Hardware

Function Point Counting for Environmental Observatory System

After conducting Function Point analysis on the project the results were the following:

Table 24- FPA results for EOS

Elementary Process	**Type of Function Point**	**Complexity**	**Unadjusted FP count**
Login	External Input (2 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Registration	External Input (12 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Retrieve Data	External Input (14 DET's) 2 FTR -> 2 ILF	Average Low	4 10
Retrieve Data from map	External Input (11 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Set display options	External Input (5 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Data Analysis	External Input (16 DET's) 1 FTR -> 1 ILF	Average Low	4 7
Displaying Data Analysis Results	External Output (2 DET's)	Low	4
Download Data	External Input (2 DET's) 1 FTR -> 1 ILF	Low Low	3 7
Store Data	External Input (16 DET's) 1 FTR -> 1 ILF	Average Low	4 7
Total Unadjusted Function Point Count			**90**

The EOS project was not divided into subprojects; instead the entire functionality was assigned to all project teams. Different teams were able to implement different functionality of the project.

The following table shows which functionality was completed by each of the four teams during the course of this project:

Table 25- FPA for EOS by Teams

Elementary Process (UFP/Process)	**Team 1**	**Team 2**	**Team 3**
Login (10)	Y	Y	
Registration (10)		Y	
Retrieve Data (14)	Y	Y	
Retrieve Data from map (10)			
Set display options(10)		Y	
Data Analysis (11)	Y	Y	Y
Displaying Data Analysis Results(4)		Y	Y
Download Data(10)		Y	
Store Data (11)	Y		
Total Unadjusted Function Point Count	**46**	**69**	**15**
VAF	**.82**	**.82**	**.82**
Total Function Point Count	**37.72**	**56.58**	**12.3**

The data for team 3 was found to be insufficient and possibly inaccurate due to incomplete documentation. To avoid a negative impact on the final results the data for team 3 was discarded.

6. Results, Findings and Interpretation

The total function point count for the eight projects analyzed ranged from 41 to 93 function points implemented. The average number of function points implemented during the second semester by each team member is 9.70. The table below provides the total number of function points per project, the number of function points per member per semester.

Table 26- Results Summary

Project	**Total Function Points**	**Function points per team member per semester**
Weather Station Project - Fall 2009/Spring 2010		
SCATS	72.21	12.04
Weather History	41.82	6.97
Current Weather Data	63.96	10.66
Automatic Weather Station Database - Fall 2008/Spring 2009		
Search, View , Analyze	52	8.67
Upload Data	51.33	8.56
Analysis	42.63	10.66
Meteorological, Image, and Environmental Data Repository System - Fall 2007/Spring 2008		
MInER	93	8.13
Environmental Observatory System - Fall 2006/Spring 2007		
EOS	90	11.94
Average	**63.37**	**9.70**
Standard Deviation	**20.12**	**1.88**

As expected the number of function points fluctuates from project to project, as it can be seen on the graph below:

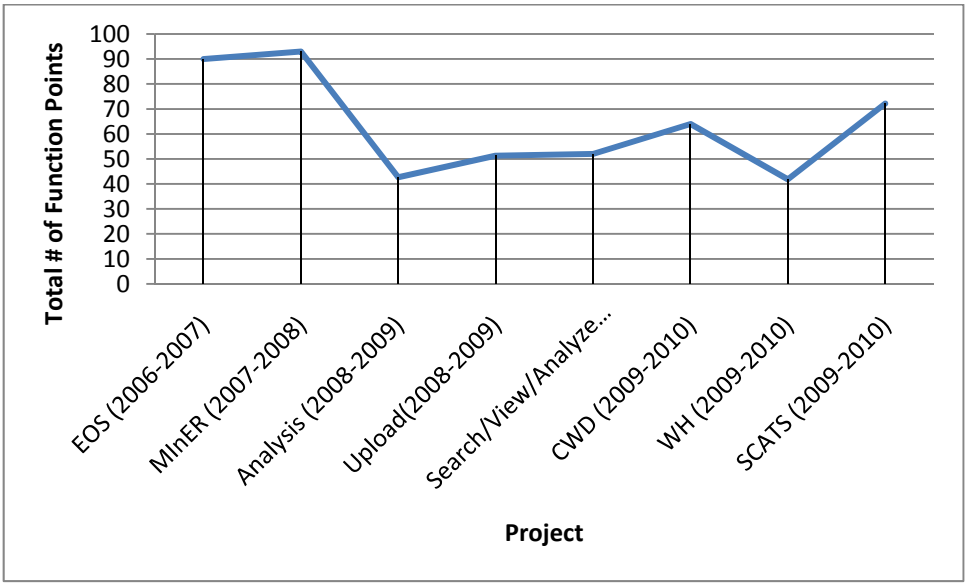

Figure 1-Graph of Total FP's

The two projects with the highest total function point count are EOS and MInER, these projects are the ones that were not divided into subprojects. The number of total function points for the other projects has remained consistent throughout the years. This indicates that the projects' level of complexity has been consistent throughout the years.

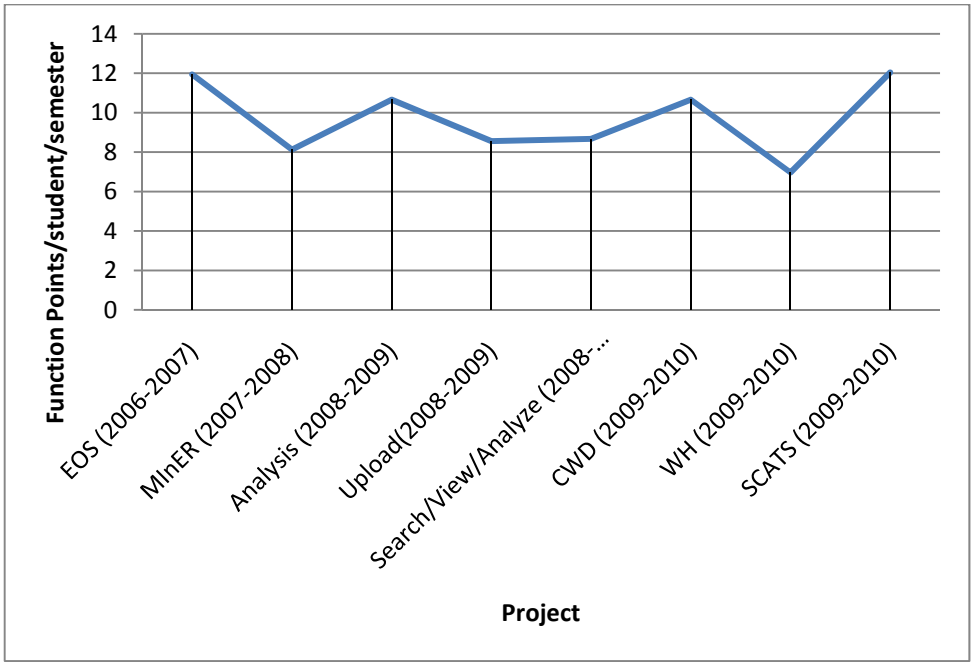

Figure 2- FP per Team Member

As it can be seen from the graph above, some projects Fall below the average function points per semester (9.70). However, overall it can be seen that there are no major decreases; this allows us to determine that the productivity of each team member throughout the years has remained consistent.

6.1. Conducting FPA for future CS capstone Projects

While conducting FPA for the CS capstone project one of the major difficulties was gathering the necessary data. I have found that the most helpful documents and deliverables when conducting FPA are:

- Complete SRSs- if projects are divided into subprojects, it would be preferable that each subproject has its own SRS.
- User interfaces- this is the most efficient way to count function points.
- Diagrams- Data Flow Diagrams and Database schema help to easily identify files accessed by a system.
- Test Plan- in addition to listing the test cases, the document should clearly indicate if the test was ran and what the result was.
- Project Status- a document clearly defining what functionality was expected, and what was completed.

References

1. D . Longstreet “Function Point Manual” [Online] http://www.softwaremetrics.com/freemanual.htm
2. H. Leung and F. Zhang. [Online]. ftp://cs.pitt.edu/chang/handbook/42b.pdf
3. The Captsone Experience at NJIT. [Online]. http://eljabiri1.tripod.com/sitebuildercontent/sitebuilderfiles/FP-COCOMO-1.pdf
4. Steve Roach, “Software Cost Estimation”, El Paso, University of Texas at El Paso, August 2009.
5. Glass, Robert “Facts and Fallacies of software engineering”. Pearson Education:2004
6. D. Galorath [Online] http://www.galorath.com
7. [Online]. http://www.projectsmart.co.uk/docs/chaos-report.pdf
8. Wikipedia. [Online]. http://en.wikipedia.org/wiki/COCOMO
9. National Aeronautics and Space Administration. [Online]. http://cost.jsc.nasa.gov/COCOMO.html
10. CSSE Website- Center for Systems and Software Engineering. [Online]. http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html
11. [Online]. devdaily.com. [Online]. http://devdaily.com/FunctionPoints/node2.shtml
12. B. Boehm and C. Abts. “Software Development Cost Estimation Approaches “ IBM Research 1998
13. Wikipedia. [Online], http://en.wikipedia.org/wiki/Therac-25
14. http://www.cds.caltech.edu/conferences/1997/vecs/tutorial/Examples/Cases/failures.htm
15. Wikipedia. [Online], http://en.wikipedia.org/wiki/Denver_International_Airport#Automated_baggage_system
16. http://www.informationweek.com/575/75mtirs.htm
17. Organizational data mining: leveraging enterprise data resources for optimal. By Hamid R. Nemati, Christopher D. Barko

18. B. M. y. Pareda and M. Pizka. Software and Systems Engineering. [Online]. http://www4.informatik.tu-muenchen.de/~pizka/mp07k.pdf
19. "Agile Inspection Keynote": [Online]. www.result-planning.com/tiki-download_file.php?fileId=187
20. Steve Roach, "Software Cost Estimation", El Paso, University of Texas at El Paso, August 2009.
21. Capers Jones "Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies"
22. http://en.wikipedia.org/wiki/Virtual_Case_File
23. IEEE, "Who killed the virtual case" [Online] http://spectrum.ieee.org/computing/software/who-killed-the-virtual-case-file
24. Software Project Cost Estimates Using COCOMO II Model. [Online]. http://www.codeproject.com/KB/architecture/cocomo2.aspx
25. S. Koirala. The Code project- Using function point to quote a software. [Online]. http://www.codeproject.com/KB/architecture/Softwarecosting.aspx
26. [Online]. www.leitner.org/courses/du/isys420-0103/metrics1.doc
27. [Online]. http://en.wikipedia.org/wiki/COCOMO
28. [Online]. http://sunset.usc.edu/research/COCOMOII/cocomo81_pgm/cocomo81.html
29. [Online]. http://csse.usc.edu/tools/COCOMOII.php
30. [Online]. http://www.cs.ucy.ac.cy/courses/EPL361/lectures/SRS.pdf

Vita

Evelyn Torres was born July 4th, 1982 in El Paso, TX. She received a Bachelor of Science in Computer Science in 2008 from the University of Texas at El Paso and a Master of Science in Information Technology in 2010 from the University of Texas at El Paso.

She served as a teaching assistant for the Computer Science Department at UTEP from 2008 to 2010.

Permanent address: 3452 Waterside

El Paso, TX 79936

This thesis was typed by Evelyn Torres.