ESTIMATION SYSTEM USING FUNCTION POINT

HIMALA DEWI A/P JAGANATHAN

BACHELOR OF COMPUTER SCIENCE
(SOFTWARE ENGINEERING)

UNIVERSITI MALAYSIA PAHANG

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS

JUDUL **ESTIMATION SYSTEM USING FUNCTION POINT**

SESI PENGAJIAN: **2012/2013**

Saya **HIMALA DEWI JAGANATHAN ____(HURUF BESAR)**

mengaku membenarkan tesis (PSM/Sarjana/Doktor Falsafah)* ini disimpan di Perpustakaan Universiti Malaysia Pahang dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Universiti Malaysia Pahang.
2. Perpustakaan Perpustakaan Universiti Malaysia Pahang dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara
   institusi pengajian tinggi.
4. **Sila tandakan (✓)

□ SULIT (Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972)

□ TERHAD (Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan)

☑ TIDAK TERHAD

Disahkan oleh

(TANDATANGAN PENULIS)

Alamat Tetap:**226, Taman Thivy Jaya**
Jalan Tok Ungku,
70300 Seremban,
Negeri Sembilan,
Malaysia

(TANDATANGAN PENYELIA)

PENYELIA
AZLINA BTE ZAINUDDIN

Tarikh: 29 MAY 2013

Tarikh:____________

CATATAN: * Potong yang tidak berkenaan.

** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali sebab dan tempoh tesis ini perlu dikelaskan sebagai SULIT atau TERHAD. Tesis dimaksudkan sebagai tesis bagi Ijazah Doktor Falsafah dan Sarjana secara penyelidikan, atau disertai bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

ESTIMATION SYSTEM USING FUNCTION POINT

HIMALA DEWI A/P JAGANATHAN

This thesis is submitted in partial fulfilment of the requirements for the award of degree of Bachelor of Computer Science (Software Engineering)

Faculty of Computer Systems & Software Engineering

Universiti Malaysia Pahang (UMP)

DECLARATION

I hereby declare that the work in this thesis “Estimation System using Function Point” is my own except as cited in reference. This thesis has not been accepted for any degree and is not concurrently submitted in any candidature of any other degree.

Signature : …………………………………………..

Name of Candidate : HIMALA DEWI A/P JAGANATHAN

Date : 5th June 2013

SUPERVISOR’S DECLARATION

“I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of the degree of Bachelor of Computer Science (Software Engineering)”

Signature : ....

Supervisor : Miss. Azlina Zainuddin

Date : ……………………………………………….

ACKNOWLEDGMENTS

First and foremost praise to God for all his blessings for giving me patience and good health throughout the duration of this research. I am very fortunate to have Miss Azlina Zainuddin as a research supervisor for all the encouragement and constant support in making this report successful. I feel I’m sincerely very lucky to get her as my supervisor. I honestly do appreciate all her consistent support from the beginning of this research.

I acknowledge my sincere thanks to my family members, especially my parent that brought me up until here. There are no words to describe their scarifications. I would like to thank my sister for all the moral support that she gave me. Not to forgotten my friends that helped me to complete this research paper successfully.

ABSTRACT

Software cost estimation is one of the most important parts in the project planning phase to ensure that the project will lead to success. Function point is one of the best methods used in estimating the software cost and size. Function point focuses more on measure the functionality thus make its estimation accurate and efficient. Function points features such as independent of programming language, product design and documented method makes it as an advantage. Besides measure the size and cost, function point also helps to measure the estimating the effort, schedule and defect in the project. The prototype of this research was developed using Microsoft Visual Studio 2008.

ABSTRAK

Penganggaran kos Perisian adalah salah satu bahagian yang paling penting dalam fasa perancangan projek untuk memastikan projek itu akan membawa kepada kejayaan. Fungsi titik adalah salah satu kaedah terbaik yang digunakan dalam menganggar kos perisian dan saiz. Titik fungsi lebih tertumpu kepada pengukur fungsi itu membuat anggaran yang tepat dan cekap. Titik fungsi mempunyai ciri-ciri seperti bebas daripada bahasa pengaturcaraan, reka bentuk produk dan kaedah didokumenkan menjadikan ia sebagai satu kelebihan. Selain mengukur saiz dan kos, titik fungsi juga membantu untuk mengukur menganggarkan usaha, jadual dan kecacatan dalam projek itu. Prototaip kajian ini telah dibina dalam Microsoft Visual Studio 2008

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

Figure Number **Page**

LIST OF APPENDICES

LIST OF ABBREVIATIONS

FP: Function Point

LOC: Lines of Codes

SLOC: Source Lines of Codes

COCOMO: Constructive Cost model

FPA: Function Point Analysis

IFPUG: International Function Point Users Group

ISBSG: International Software Benchmarking Standards Groups

ISO : International Organization for Standardization

CFP : Crude Function Point

RCAF: Relative Complexity Adjustment Factor

DFD : Data Flow Diagram

SRS: Software Requirement Specification

CHAPTER 1

1.1 Introduction

Software cost estimation is one of the most important parts in the software planning phase. A good planning and requirements from the beginning will lead a project to success. In software projects, size is not everything, but it does influence most of the things like cost and resource. So if we do not have an accurate prediction of size, it’s difficult to plan. A precise software to calculate the software cost estimation will be helpful to project managers to estimate their software size.

There are many factors, which lead to software projects fails. The major causes of software failure are poor planning and cost estimation. The initial cost and estimated schedule are not more frequently revised as more information available. Besides that, current practices in software cost estimation are being done manually and causes to project's failures. Such a major problem can be avoided if a software tool used to calculate the cost estimation of the projects to get an accurate result in estimating. An accurate and efficient cost estimation methodology for web-based application is very important for software development as it would assist the management team to estimate the cost. Furthermore, it will ensure the development of cost suits the planned budget and provides a fundamental motivation for the development of a web-based application project (Zulkefli, M., Zarinah, M.K., Habibah, A., Saadiah, Y, 2011). There have been various cost estimation models and methods that are being used in the software-development process. Function Points is one the example to calculate the estimation cost.

Function point is one of the most accepted and robust sizing techniques used in the software cost estimation process, function point, which formulate by Albrecht was established in the early of 1970 (M.A. Al-Hajri, A.A.A. Ghani, M.S. Sulaiman, M.H. Selamat. 2005). Function point has many advantages over the other cost estimation models like they are independent of programming language, product design or development style; it is a well-documented method and many more. In addition, function point estimation is achieved directly without the formalization of step by step analytical procedures. Besides that, research did by Graham C. Low and D. Ross Jeffery has proven that function point counts appear to be more consistent with measure software size.

1.2 Problem statement

Software cost estimation has a great impact on the software-development process. The success of the software project depends on factors such as time and cost. There have been researched conducted on this issue in Malaysia. However, the data used to have not been sourced locally. Therefore, the accuracy is questionable. Moreover, the factors can be dependent on local environment and needs specific to those in Malaysia. There is still much to be discovered and that is what this study is aimed at. According to a study made in Malaysia, they have come up with the statistic based on literatures that 52.7% projects were not able to be complete on time and over budget and 31.1% not fulfilled the scope (Iman.A, Ow.S.H.008.)

The causes of the project failure were mainly poor planning and estimation. Besides that, case studies on a group of50students were taken. 49undergraduates were from Faculty of Computer Science and Information Technology and an undergraduate from the department of Information science from University of Malaya, who took the course Project Management. The students were assigned a team project based on their preference. The students were divided into seven groups with seven to eight members comprising Malays, Chinese and Indians. All the projects were focused on the budget, schedule and quality. Based on the lecturer’s assessment on the students projects based on budget only one team managed to complete the project within the budget. The remaining six teams were over the budget. Besides that, analysis made on the cost estimation, two groups budgets were over the project cost, and four teams manage to make good estimation but failed to complete within budget. Another research was also carried out based on estimation using function point and source of line code. The research was carried out based on two programs using specification prepared by an experienced professional analyst (G.C. Low And D.R.Jeffery. 1990). The 1st program used was Fixed Asset Master File Update, and the 2nd program used was Fixed Asset Depreciation Calculation and Reporting. The data collected, and the relationship was divided into three major data sets. The first data set was comprised of twenty-two function point experienced analyst counting a priori from the program specification. The second data set was comprised of two groups of function point naïve analysts where one group very experienced in analysis and the other group do not. The third data set was consisting of twelve analyst estimating source lines of code from the program specification. The result of their research on the comparison of the consistency of the function point and source lines of code estimates suggests that on an organizational basis source bank line of codes, estimates are no better than function point estimates when estimating from a program specification. The research concluded that function point counts appear to be a more consistent a-priori measure of software size than source lines of codes. With the assistant of software estimation tool and a correct methodology used to calculate the estimation; such a problem can be avoided. By having software for the estimation calculation, the time taken to calculate for estimation can be reduced. Thus, the additional time left can be reallocated to focus on the difficult part on the software project phase like designing.

1.3 Objective

i. To analyze the consistency of the function point in measuring the software cost estimation

ii. To compare and find out the best way to practice the software cost estimation between function point and other method.

iii. To choose the best estimation method using the Analytical Hierarchy Process.

1.4 Scope

The study is carried out on the method of calculating the software cost estimation. The prototype focuses on the function point method. This prototype will be a standalone system. This project focuses on two modules of Ump- Automatic Sports Facilities Management System.

1.5 Thesis Organization.

Chapter one is mainly about the details of the case study. It contains introduction, problem statement, scope, and objective. Chapter two is about the literature review. A literature review is an account of what has been published on a topic accredited scholars and researchers. It is a part of the introduction to research report. The purpose is to convey the reader what knowledge and idea have been established on a topic and their strengths and weaknesses are. Chapter three is basically about the methodology used in this research paper. This chapter discusses about identifying the data collection instrument to be employed for the study, determining the sample or participants of the study, and analyses the data collected for the study. Chapter four discusses about the design of the system. Chapter five is about the implementation of the system followed by chapter six, which is about the result and discussion of the thesis. Finally, chapter seven is the conclusion of the thesis.

CHAPTER 2

LITERATURE REVIEW

2.1 Estimation System

Estimation is the process of finding an estimate, or approximation, which is a value that is usable for some purpose even if input data may be incomplete, uncertain, or unstable. Typically, estimation involves "using the value of a statistic derived from a sample to estimate the value of a corresponding population parameter.”

Estimation is one of the most important parts in the software planning phase. The estimation involves estimating the size of the software, budget, time and resource. Errors or mistakes in estimation can attribute to many factors. The meaning of an estimation system in this context is estimating the size of the software. An accurate estimate of software size is an essential element in the calculation of estimated project costs and schedules. The fact that these estimates are required very early in the project (often while a contract bid is being prepared) makes size estimation a formidable task. If the estimation is inaccurate there will be major problems. If the under estimation was done, it can result in under-staffing and may result in an over worked and burnt-out team. The estimation process has sub tasks. As the size increases, the interdependency among various elements of the software grows rapidly. There are different methodologies for arriving at and expressing the size/complexity of the Software Program. Some of the popular ones are Function Points, Lines of Codes, and Cocomo.

In conclusion, estimation is extremely a vital process in a software planning phase. Thus, having a good estimation system to estimate the size of the system would be an advantage. In addition, it will make sure that we would keep us on the track to complete the system on time and with the entire requirement are fulfilled.

2.2 Existing Application and its Problems

2.2.1 Lines of Code

Lines of code often referred as Source Lines of Code, SLOC or LOC. Lines of code are a formal method to measure the size by counting number of lines of Code. This metric was very popular primarily because of its use simple and easy. Lines of code measure the number of source instruction used to solve a problem. While counting the number of source instructions, lines used for commenting and blank lines are ignored. Although using lines of code is simple but there are many disadvantages in using it.

Usually estimation will be done at the beginning of the project but by using lines of code estimation at the beginning of the project will be very tricky. In order to make the estimation easy, the project manager will divide the project managers divide the problem into modules, and each module into sub modules, and so on until the sizes of the different leaf-level modules can be approximately predicted. The next disadvantage is some programmers may create a lengthy code structure to solve a problem as they do not make effective use of the available instruction set. At the same time, skilled programmers can code a simple and effective code for the same problem. Thus, a poorly written code cannot be a good metric for estimation purpose.

Lines of code only focus on the coding part and ignore other important parts such as designing and implementation. Coding is only a small part in a software development system. Besides that, LOC also have problem with its language dependence. LOC only allow usage of one language in the estimation process unlike function point that does not depend on what language used. It is not possible to directly compare projects developed by using different languages. For example, the time per line for a high-level language may be greater than for a lower-level language. There is no way to accommodate the fact that fewer lines of code may be required for a higher-level language to provide the same function. Moreover, LOC is lack of a universally accepted definition for exactly what a line of code really is. Jones (1986) identified 11 major variations of line counting methods. Since few authors state the line-counting rules they used, much of the literature has an "uncertainty of perhaps 500% attributable to line counting variations." The variations make it very difficult to compare studies using lines of code as a measure of software size (Matson.J.E, Barrett.E.B., Mellichamp.J.M., 1994). In conclusion, with so many disadvantages, using LOC for estimating a software system is not a good idea.

2.2.2 Constructive Cost model (COCOMO)

Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model. It was developed by Barry W. Boehm. The model is a combination of statistical figures, mathematical equations and expert judgments. Furthermore, COCOMO is an open model, so all the details such as the underlying cost estimation equations, Every assumption made in the model, Every definition, and The costs included in an estimate are explicitly stated. This model also have been widely use in the companies. COCOMO is composed into three levels of the models which is basic, intermediate and detailed. The basic COCOMO'81 model is a single-valued, static model that computes software development effort (and cost) as a function of program size expressed in estimated thousand delivered source instructions.

The intermediate COCOMO'81 model computes software development effort as a function of program size and a set of fifteen "cost drivers" that include subjective assessments of product, hardware, personnel, and project attributes. Some of the COCOMO's limitations are primarily COCOMO represents development from planning to implementation. It doesn't consider maintenance, rework, porting and integration, and reuse. COCOMO model ignores requirements and all documentation. Function point method measures the developed system by point counts that can determined relatively early in the development process. It measures software project size by studying external features of the projects (Gao.X, Lo.Bruce, .1995). So it avoids the difficulty of the COCOMO method which means in COCOMO does not support some newer programming environment like Authorware, make counting of LOC difficult as the definition of a line is less clear-cut. Besides that, it ignores customer skills, cooperation, knowledge and other parameters. It oversimplifies the impact of safety/security aspects. It ignores hardware issues. It ignores personnel turnover levels. It is dependent on the amount of time spent in each phase.

2.3 Function point

The Function Point methodology was developed by Allan Albrecht at IBM in 1979. This methodology is based on the belief that the size of a software project can be estimated during the requirements analysis. Function Point Analysis (FPA), or the method of sizing software in terms of its function and expressed in Function Points is now very widely used (Vicker.P.). It takes into account the inputs and outputs of the system. Function points can be determined from the requirement specifications, design specification, source listing or live system. Function point focuses on "functionality" or "utility" rather than counting LOC (Gao.X, Lo.Bruce, .1995). Since function point measure functionality, they should be independent of technology and language used for the software implementation (Low,G.C. And Jeffery,D.R. 1990)$^{.}$ It helps to estimate software effort more accurately without considering the languages or developing environment you choose (Zheng.Y, Wang.B, Zheng.Y, Shi.L. 2009).In addition, the ability of function point could help in effort, schedule, and defect estimation and aids in setting project scope. Function point does not counts the lines like how LOC does instead it counts the number of externals that make up the system. The function point approach has features that overcome the major problems with using lines of code as a measure of system size.

First, function points are independent of the language, tools, or methodologies used for implementation; i.e., they do not take into consideration programming languages, data base management systems, processing hardware, or any other data processing technology (Matson.J.E, Barrett.E.B., Mellichamp.J.M., 1994). Function points are based on the system user's external view of the system, nontechnical users of the software system have a better understanding of what function points are measuring (Matson.J.E, Barrett.E.B., Mellichamp.J.M., 1994). By using function point, its make us easy for comparing project productivity and organization besides help manager better understand and articulate to client the impact of change request and enhancements. Function points also allow the manager to make informed decision more objectively with the limited time and budget.

A case study done by few researches have proved that relationship between software effort and function point count can be assumed as a linear regression model y = a + bx where *a* is the slope of the line (gradient) and *b* is the y intercept. From the scatter diagram which was shown in the figure 1,the relationship between the variable and the dependent variable tends to be a straight line with highly positive correlation. The case study also concluded that though it is difficult to figure out the count of function point, the linear function will greatly simplify the process of software estimation, and then help the manager to have an accurate software effort measurement (Zheng.Y, Wang.B, Zheng.Y, Shi.L. 2009).

Another case study which was carried out in 1994 has also come out with a conclusion which is Function point counts appear to be a more consistent a priori measure of software size than source lines of code (Low,G.C. And Jeffery,D.R. 1990).

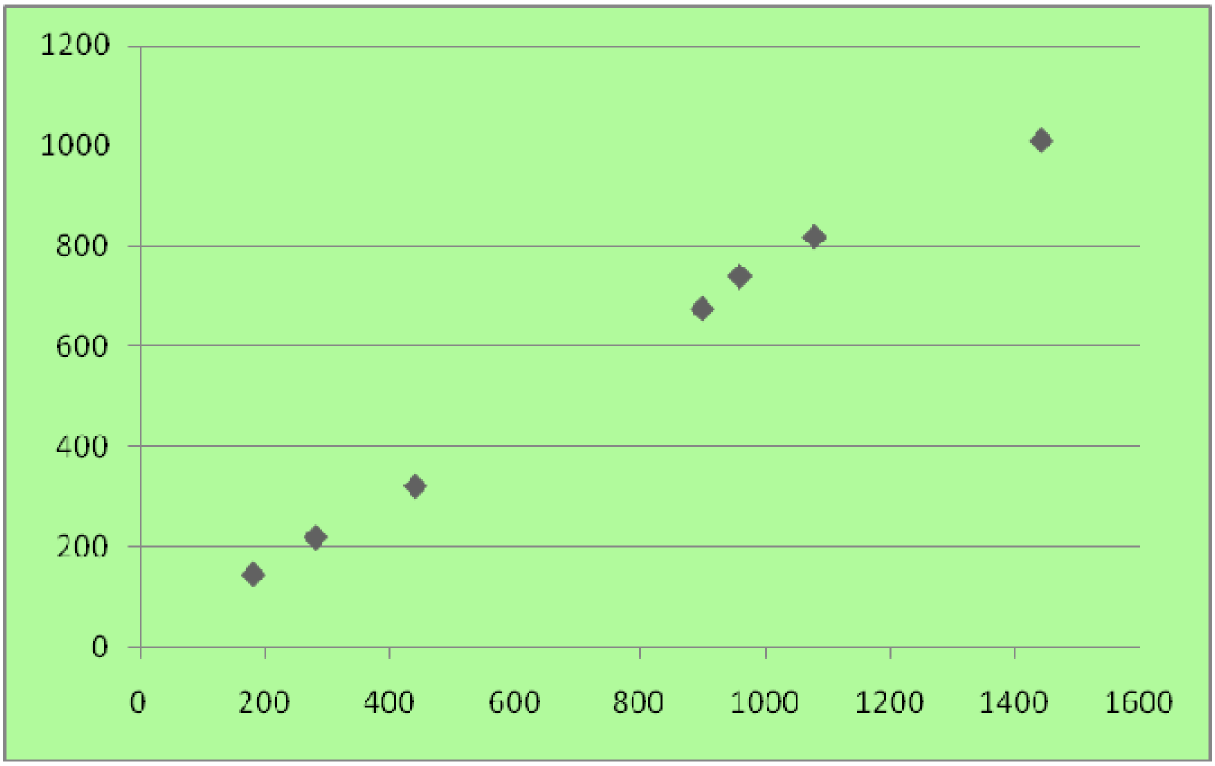


Figure 2.1 Relation between function point and software effort

Example of a linear regression model. The vertical axis is the x-axis that represents the software effort and the horizontal axis the y-axis that represents the Function point.

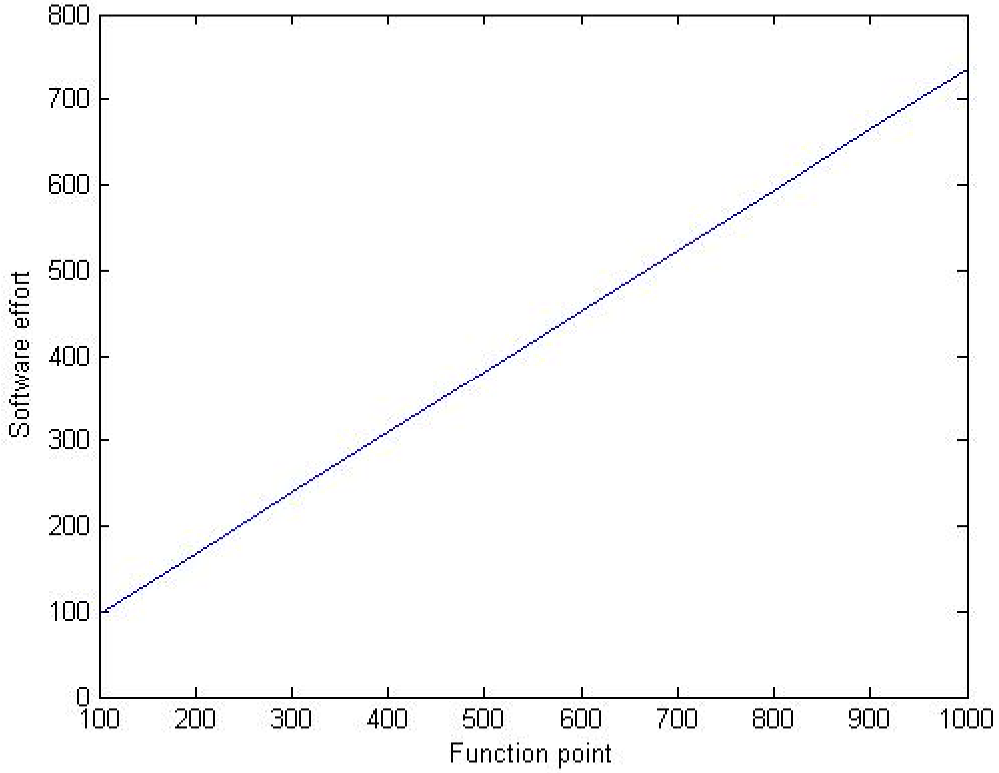


Figure 2.2 Linear relation between function point and software effort

2.4 Applied Function Point application in the Industry.

The function point methodology is being successfully applied by innumerable organizations world-wide to measure software size for existing applications, enhancements to those applications, and new development projects. Function Points Analysis (FPA) has become a world standard over the years. The mission of International Function Point Users Group (IFPUG) is to be a recognized leader in promoting and encouraging the effective management of application software development and maintenance activities through the use of Function Point Analysis and other software measurement techniques. IFPUG endorses FPA as its standard methodology for software sizing (Dekkers.T). Carol Dekkers from Quality Plus Technologies and Mauricio Aguiar from Caixa Economica Federal has stated that trough their experience shows that FPA is often more effective than peer or user walkthroughs in identifying the full set of functional user requirements and uncovering potential defects. In fact, benefits gained by applying FPA to functional user requirements can be more valuable than the mere function point size of the software (Carol A.D., Aguiar.M).

There are many reasons why most of the industries choose Function Point as a tool for estimating. The outcome of a Function Point count provides the metric ‘unit of software delivered and can be used to assist in the management and control of software development, customization or major enhancements from early project planning phases through to the ongoing support of the application. In addition, the software size facilitates the creation of more accurate estimates of project resources and delivery dates and facilitates project tracking to monitor any unforeseen increases in scope. Industry figures available from International Software Benchmarking Standards Groups (ISBSG) *Repository* for projects measured with IFPUG function points indicates that complete applications tend to have consistent and predictable ratios of each of the function types.

The research conducted by the Total Metrics Pty. Ltd shows that industry figures show that the risk of project failure rapidly increases with project size. Projects less than 3500 function points have a risk of failure of less than 20% in comparison with projects over 5000 function points which have a probability of cancellation close to 40%. This level of risk5 is unacceptable for most organizations. Data within the ISBSG Repository Release 6 supports the premise that smaller projects are successful. Over 65% of the projects in the repository are less than 500 function points and 93% of the projects are less than 2000 function points. Thus, they concluded that Industry experience suggests that the best managed projects which deliver quality software on time and within budget tend to less than 700 function points and up to 1500 function points.

While in Malaysia, many projects were developed but only very little percentage of projects that have succeed while other succeed with challenges such as over run budget, time overrun, and impaired functionality. There are many factors which can lead to such condition but the main factor would be choosing the wrong method to estimate the software size and budget. A research based on software cost estimation practices in Malaysia was conducted by Zulkefli, M., Zarinah, M.K., Habibah, A., and Saadiah, Y in the year 2012. The research has come out with a random survey in order to get an overview of current practice in cos testimation process among project managers and web developers in Klang Valley. The research have concluded that Both project managers and web developers agreed that Expert Judgment, Price-to-Win and Algorithmic Model methods produce most accurate result in cost estimation process and at the same time the result is incongruent to the method preferable used by the project manager and web developer. The method Parkinson-Ian shows the most accurate method to count the cost estimation. The paper conclude that in order to get accurate cost estimation result a good estimation process with the proper selection of cost estimation technique, correct size measure, person experiences, and familiarity of software developed.

We do have SIRIM and International Organization for Standardization (ISO) in Malaysia. SIRIM Berhad is a wholly-owned company of the Malaysian Government under the Ministry of Finance Incorporated. SIRIM is recognized the world over as a global research and standards development organization. SIRIM focus on discovering and developing new technologies to help businesses compete better through quality and innovation. Besides that, SIRIM also continuously reinventing the way we do things and ensuring that they remain market-driven, flexible, cost-effective, and responsive to our clients. ISO organization is responsible for developing standards for products and services that identify a need for standardization. The ISO is usually contacted by a sector of an industry or stakeholders in a product and asked to develop a standard, such as those created for manufacturing. ISO helps governments around the world create environmental, health and safety policies. The ISO helps test products during the standard-setting process. Many international trade agreements also incorporate ISO standards. ISO standards allow consumers to buy and use products safely. The use of ISO branding allows products to be produced safely and efficiently.

2.5 Comparision Study

	Function Point (FP)	Lines Of Code (LOC)
Definition	measure software size by quantifying its functionality provided to the user, based on the requirements and logical design.	measure the amount of code in a software program typically used to estimate the amount of effort that will be required to develop a program, as well as to estimate productivity once the software is produced.
Structure	Consist of 5 major components.	Consist of 2 parts. 1st part – provide a base estimate as a function of software size. 2nd part-modifies the base estimate to count for the influence of environment factors
Advantages	**Helps Monitor Scope Creep :** provide a mechanism to track and monitor scope creep. The FP count at the end of requirements and/or designs can be compared to FP actually delivered. If the project has grown, there has been scope creep. **Ease of Contract Negotiations:** From a customer view point, Function Points can be used to help specify to a vendor, the key deliverables, to ensure appropriate levels of functionality will be delivered, and to develop objective measures of cost-effectiveness and quality. They are most effectively used with fixed price contracts as a means of specifying exactly what will be delivered	**An intuitive metric** : Line of Code serves as an intuitive metric for measuring the size of software due to the fact that it can be seen and the effect of it can be visualized **Scope for Automation of Counting**: Since Line of Code is a physical entity; manual counting effort can be easily eliminated by automating the counting process. Small utilities may be developed for counting the SLOC in a program.

	Function Point (FP)	Lines Of Code (LOC)
	Use of Historic Data : Once project size has been determined in Function Points, estimates for Duration, effort, and other costs can be computed by using historic data. Since FP is independent of languages or tools, data from similar past projects can be used to produce consistent results	
	Enables Better Communication: FP can help improve communications with senior management since it talks in terms of functionality rather than any implementation details, technical aspects, or physical code.	
	Offers Better Benchmarking: Since FP is independent of language, development methodology, programming practices, and technology domain, projects using FP become better candidates for benchmarking across organizations and geographies.	
Disadvantages	**Necessitates Significant Level of Detail:** A great level of detail is required to estimate the software size in terms of Function Points	**Lack of Accountability:** It is completely inaccurate and unfortunate to have to measure the productivity of a development project with the outcome of one of the phases (coding phase) which usually accounts for only 30% to 35% of the overall effort.
	Requires Experience: Function Point Analysis requires good deal of experience if it were to be done precisely. FPA inherently requires sufficient knowledge of the counting rules, which are comparatively difficult to understand.	**Lack of Cohesion with Functionality :** skills developer may able to produce a program with less code compared to less skilled developer. so one program with less LOC may exhibit more functionality than another similar program

	Function Point (FP)	Lines Of Code (LOC)
	Size of a system in unadjusted function points (UFPs): The classification of all system component types as simple, average and complex is not sufficient for all needs.	**Developer's Experience :** Implementation of a specific logic differs based on the level of experience of the developer. Hence, number of lines of code differs from person to person.
	Interpreting on-line interactive transactions : Difficulty when each input data element is followed by an output response on the same screen. Does not sure if the screen was suppose to count as output, or input or both.	**Difference in Languages-** when applications are written in different language, the aspects of the application would be different, thus the amount of effort also will be different.
		Lack of Counting Standards- There is no standard definition of what a line of code is.

Table 2.1 Comparison between Function Point and Lines of Code

2.6 Function Point Technique

The first step that does to count the function point is identifying the system functional components to compute the crude functions points (CFP). There are five types of software system components that are considered in counting the CFP such as the user inputs, user outputs, user online queries, logical files and external interfaces. Each component is then further classified as being simple, average or complex depending on the number of data elements in each type and other factors. Each component is then assigned a points value on the basis of its type and complexity. The point's values of all the components are summed then to give a size for the system in crude functions points (CFP). Each software system component is multiplied according to its weight factor .

Software System Component	Complexity Level									Total CFP
	Simple			Average			Complex			
	Count	Weight factor	points	Count	Weight factor	points	Count	Weight factor	points	
	A	B	C = A×B	D	E	F = D×E	G	H	I = G×H	J=C+F+I
User inputs		3			4			6		
User Outputs		4			5			7		
User Online Queries		3			4			6		
Logical Files		7			10			15		
External Interfaces		5			7			10		
Total CFP										

Figure 2.2 Crude Function Point Calculation Table

The second step is calculating the relative complexity adjustment factor (RCAF). The RCAF summarizes the complexity characteristics of the software system by assigning grades (0 to 5) to the fourteen General Application Characteristics that affect the requirement development efforts.

No.	Subjects	Grades					
1	Requirement for reliable backup and recovery.	0	1	2	3	4	5
2	Requirement of data communication	0	1	2	3	4	5
3	Extend of distributed processing	0	1	2	3	4	5
4	Performance requirement	0	1	2	3	4	5
5	Expected Operational requirement	0	1	2	3	4	5
6	Extend of online data entries	0	1	2	3	4	5
7	Extend of multi screen or multi operation online data input	0	1	2	3	4	5
8	Extend of online updating of master files	0	1	2	3	4	5
9	Extend of complex inputs, outputs, online queries and files	0	1	2	3	4	5
10	Extend of complex data processing	0	1	2	3	4	5
11	Extend that currently developed code can be designed for reuse	0	1	2	3	4	5
12	Extend of conversion and installation included in the design	0	1	2	3	4	5
13	Extend of multiple installation in an organization and variety of customer organizations	0	1	2	3	4	5
14	Extend of change and focus on ease of use.	0	1	2	3	4	5
Total = RCAF							

Table 2.3 Relative Complexity Adjustment Factor Table

The last step is calculating number of Function Point. The number of function point is calculated by the following formula:

$$\mathbf{FP = CFP \times (0.65 + 0.01 \times RCAF)}$$

2.6.1 Example of Function Point Calculation.

The *Attend-master* software system

The attend-master is a basic employee attendance system that is planned to serve small to medium-sized businesses employing 10–100 employees. The system is planned to have interfaces to the company's other software packages: Human-Master, which serves human resources units, and Wage-Master, which serves the wages units. Attend-Master is planned to produce several reports and online queries. The scheme of the planned software system is shown in the data flow diagram (DFD) shown in Figure 2.4.

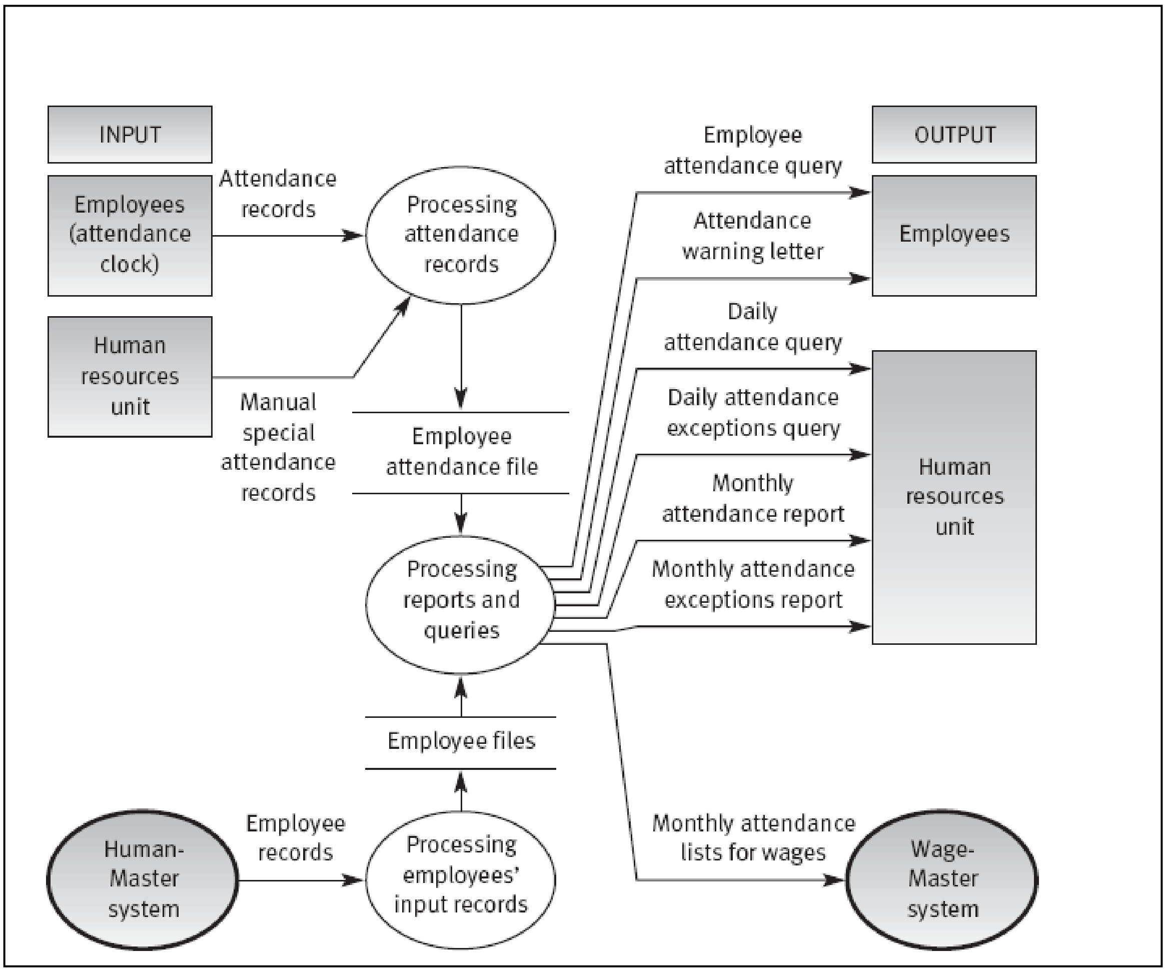

INPUT
Employees (attendance clock)
Attendance records
Processing attendance records
Human resources unit
Manual special attendance records
Employee attendance file
Processing reports and queries
Employee files
Human-Master system
Employee records
Processing employees' input records
OUTPUT
Employee attendance query
Attendance warning letter
Employees
Daily attendance query
Daily attendance exceptions query
Monthly attendance report
Monthly attendance exceptions report
Human resources unit
Monthly attendance lists for wages
Wage-Master system

Figure 2.3.The attend-master data flow diagram.

Step 1. Calculating the crude function point

Analysis of the software system as presented in the DFD summarizes the number of the various components:

Number of user inputs – 2

Number of user outputs – 3

Number of user online queries – 3

Number of logical files – 2

Number of external interfaces – 2.

The degree of complexity (simple, average or complex) was evaluated for each component (shown in table 2.5), after which CFP calculations were performed.

Software System Component	Complexity Level									Total CFP
	Simple			Average			Complex			
	Count	Weight factor	points	Count	Weight factor	points	Count	Weight factor	points	
	A	B	C = A×B	D	E	F = D×E	G	H	I = G×H	J=C+F +I
User inputs	1	3	3	-	4	-	1	6	6	**9**
User Outputs	-	4	-	2	5	10	1	7	7	**17**
User Online Queries	1	3	3	1	4	4	1	6	6	**13**
Logical Files	1	7	7	-	10	-	1	15	15	**22**
External Interfaces	--	5	-	-	7	-	2	10	20	**20**
Total CFP										**81**

Table 2.4 Calculate CFP

Step 2. Calculating the Relative Complexity Adjustment Factor (RCAF)

No.	Subjects	Grades					
1	Requirement for reliable backup and recovery.	0	1	2	3	4	[✓ 5]
2	Requirement of data communication	[✓ 0]	1	2	3	4	5
3	Extend of distributed processing	[✓ 0]	1	2	3	4	5
4	Performance requirement	0	1	2	3	4	[✓ 5]
5	Expected Operational requirement	[✓ 0]	1	2	3	4	5
6	Extend of online data entries	0	1	2	3	[✓ 4]	5
7	Extend of multi screen or multi operation online data input	0	1	[✓ 2]	3	4	5
8	Extend of online updating of master files	0	1	[✓ 2]	3	4	5
9	Extend of complex inputs, outputs, online queries and files	0	1	2	3	[✓ 4]	5
10	Extend of complex data processing	0	1	2	3	[✓ 4]	5
11	Extend that currently developed code can be designed for reuse	0	1	2	[✓ 3]	4	5
12	Extend of conversion and installation included in the design	0	1	[✓ 2]	3	4	5
13	Extend of multiple installation in an organization and variety of customer organizations	0	1	2	3	4	[✓ 5]
14	Extend of change and focus on ease of use.	0	1	2	3	4	[✓ 5]
Total = RCAF		**41**					

Table 2.5 Relative Complexity Adjustment Factor Example Table

Step 3. Counting the Function Point

FP = CFP × (0.65 + 0.01 × RCAF)

FP = 81 × (0.65 + 0.01 × 41)

= 85.86

2.7 Use of Analytic Hierarchy Process (AHP) as an indicator in Function Point.

Analytic Hierarchy process is a process where it takes consideration of all the choices not based on a particular choice. It can said to be as a group decision making where it involves, expert judgments and opinion asking. After consideration by all the choices then only the perfect decision is made. With the help of the AHP technique in estimating the system using function point would be a great advantage because it will let choices that were not chosen to improve their characteristics.

Using AHP in estimation system is a good choice because it allows us to do comparison with all the choice. By doing comparison with all the choices that have been stated, the percentage of confident for choosing the best answer is more.

CHAPTER 3

METHODOLOGY

3.1 Existing Process

The existing function point estimation process estimates the total function point value. The user need to identify the all the functionality that the project has before proceed with the next step. Then once the functionality has been identified, all the functionality is then multiplied with its respective weight factors before being add together to get the total to get the crude function point.

The next step is calculating the Relative complexity adjustment factor (RCAF). The user needs to identify the complexity characteristic of the software by assigning a grade between zero to five with zero meaning no influence and five meaning that characteristic has an extensive influence throughout the project. There is fourteen General Application Characteristics that affect the requirement development efforts. All the values have been assigned to the fourteen general application characteristics will be added together to get the total RCAF value before proceed with the last step.

The last step in the process is calculating the function point where the user needs to choose the RCAF value of the project that ranges from 0.65 to 1.35 where 0.65 is chosen if all complexity has no influence and the maximum 1.35 if all complexity has significant influence. Mostly the RCAF value will be lower than 1 because the majority the complexity factors would be small influence. Once the value has been finalized, the user needs find the total function point of the system by adding the RCAF value with

0.01 which is an empirically derived formula is multiplied with the total value of the RCAF. Lastly this value is then multiplied with the CFP to get the Function point.

$$\mathbf{FP = CFP \times (0.65 + 0.01 \times RCAF)}$$

3.2 Issues with Existing System.

The existing system for the function point calculation is seemed to be focusing on getting the function point, calculate the software cost, size, duration of the project and optimizing staffing size. It is clearly understood that by using function point, the ability to accurately estimate the project cost, duration and optimum project staffing size besides its also helps to determine other important metrics, such as project defect rate, cost per FP, and FP's per hour. But at the same time, according to the research conducted by Total Metrics Pty. Ltd shows that industry figures show that the risk of project failure rapidly increases with project size. The research also stated that if the function point is less than 3500 the possibility of failure is 20% and probability of cancelation of project is 40% if the function point is over 5000.

This function point method does not take account the risk factor in the calculation of finding the function point. If the risk factor is taken as a consideration, the chances for the project to fail or probability of cancelation can be reduced. Thus, many projects can be saved from failures. The International Function Point Users Group (IFPUG) should provide a guideline to all function point users where it contains the amount range per function point counts according to the developers skill so that, there will be a synchronization.

3.3 Technique

The technique we are going to use in this research is analytic hierarchy process (AHP). AHP technique is not making a decision by one particular person. It involves making decision in groups. It has particular application in group decision making, and is used around the world in a wide variety of decision situations, in fields such as government, business, industry, healthcare, and education. As we all know AHP is a structured technique for organizing and analyzing complex decisions. AHP is one of the best theories of measurement that consider both the judgments from the experts and the priority scales. The decision makers systematically evaluate its various elements by comparing decision situation such as choice, ranking, prioritization, and bench marking. The AHP technique can be done by using the following this steps which is state the objective then define the criteria and lastly pick the alternative.

In AHP, decision making involves many criteria and subcriteria used to rank the alternatives of a decision. AHP first decompose their decision problem into a hierarchy of more easily comprehended sub-problems, each of which can be analyzed independently. The elements of the hierarchy can relate to any aspect of the decision problem tangible or intangible, carefully measured or roughly estimated, well- or poorly-understood anything at all that applies to the decision at hand.

So, when we handling a software project, it would be easy for us to use this technique to solve our problems or making decisions when we are in a tight situation especially when we calculate how long it does take to finish the project. It also will assist the project manager based on risk. The applications of AHP to complex decision situations have numbered in the thousands, and have produced extensive results in problems involving planning, resource allocation, priority setting, and selection among alternatives.

As for AHP, we can that using the AHP involves the mathematical synthesis of numerous judgments about the decision problem at hand. The judgments are not based on 1 particular person, its made by few higher experienced authority to make the final judgment regarding which will be the best decision for that particular problem.

Decision Making with AHP Modeling

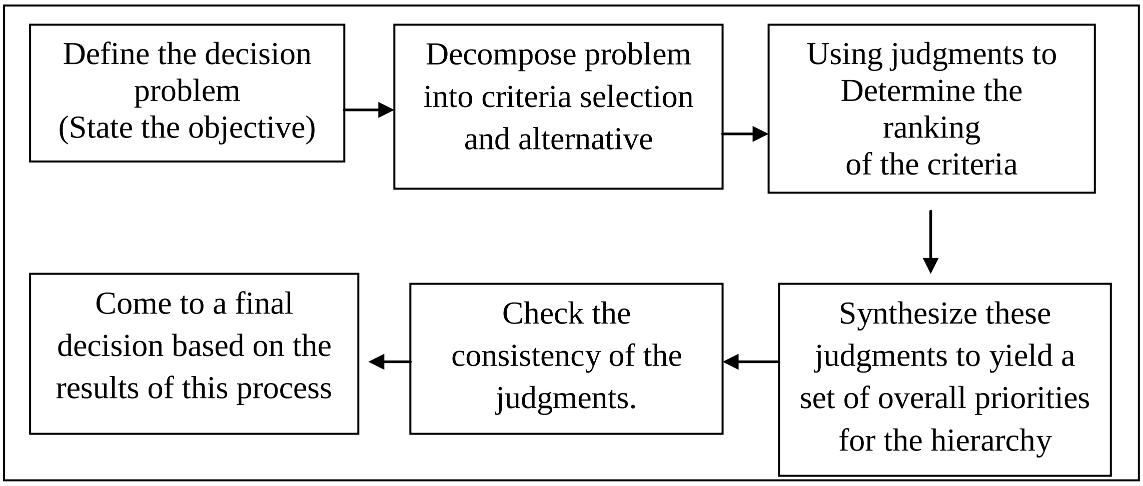

Define the decision problem (State the objective)
Decompose problem into criteria selection and alternative
Using judgments to Determine the ranking of the criteria
Synthesize these judgments to yield a set of overall priorities for the hierarchy
Check the consistency of the judgments.
Come to a final decision based on the results of this process

Figure 3.1 AHP Technique

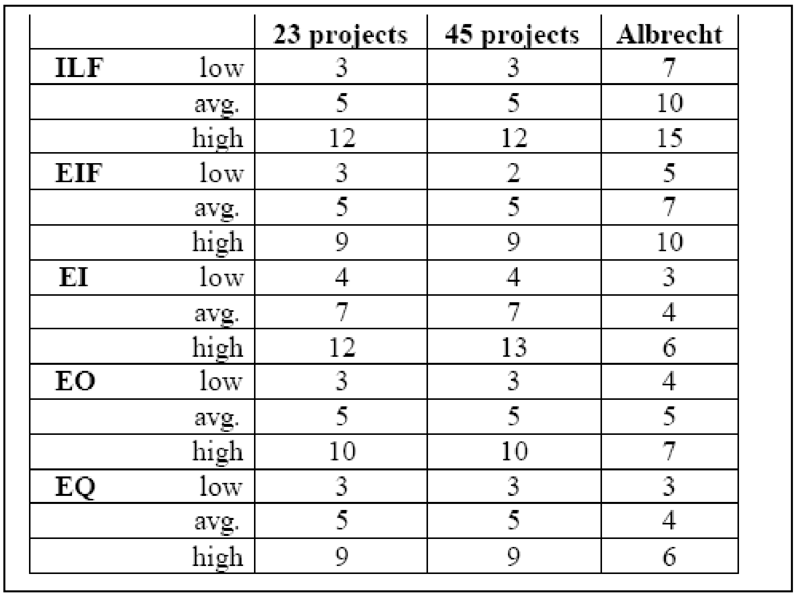
3.4 Validating Function Point Weighting Factor.

Albrecth (1979) and IFPUG (1994) use the weight table to assign he value to particular individual function type. To determine empirically the weight coefficient for the IFPUG function point method, a formal methodology has been proposed by Wittig et al. (1996). To establish the weight factor Wittig et al. (1996) use the Analytic Hierarchy Process(AHP). The AHP technique compares and relates pairs of individual. Users of an Information system will be asked assess, based on which of two function types are which the larger function types is and how large it is. When the samples needed is sufficient of all the combinations it will produce a value set for FPA function types.

Wittig et al. (1998) have comes out with a collection of questionnaire which consist of 25 component pairs. The 25 component pair is selected from the total of 105 where 15 different components of function types (5 types with 3 complexity levels each). Within each of the 5 function types all three combinations between the three complexity levels were included. It ensured, however, that the assessor was familiar with the function types. It further provided a variety of different instances of individual components.

Based on the research done by Wittig et al, during the first field test they found out that assistance was required to provide assessment criteria for functional size. the paper also stated that the IFPUG classification only covers individual function types and also stated is the danger that either the development effort or the benefit to the business may influence the judgment of functional size of a component.

The research comes out with the first results which is based on 23 projects. Later for a more guidance purpose further 22 projects were collected and combined with the previous 23 projects and analyzed. The data for the first result based on the 23 projects, the second result based on 45 projects and Albrecht weight factors are compared. The data are shown in the table below.



		23 projects	45 projects	Albrecht
ILF	low	3	3	7
	avg.	5	5	10
	high	12	12	15
EIF	low	3	2	5
	avg.	5	5	7
	high	9	9	10
EI	low	4	4	3
	avg.	7	7	4
	high	12	13	6
EO	low	3	3	4
	avg.	5	5	5
	high	10	10	7
EQ	low	3	3	3
	avg.	5	5	4
	high	9	9	6

Figure 3.2 Research Outcome

The additional observations are very close to the initial results of the AHP study. Overall the results appear to confirm Albrecht‘s weight factors. The transaction function types, however, seem to score higher at the expense of the data oriented function which appear to be overrated. The study also concluded that the function point calculation is not far apart from the AHP calculation.

3.4 Hardware

In order to develop the prototype of this thesis, hardware specifications are very important. Table 3.1 shows the hardware that has been used on to develop this system.

Hardware	No. of Quantity	Speculation	Function
Laptop	1	I. Intel® Core™ i5 CPU M640 @ 2.53GHz II. RAM 4.00GB III. Microsofts Windows 7	Prepare Document and prototype
Printer	1	Canon PIXMA E500	To print the documents
USB device	1	Pendrive 8GB	To store data as backup

Table 3.1 Hardware item that will be used for this thesis

3.5 Software

In order to develop this system, the software specification is important. Table 3.2 shows the software that has been used on to develop this system.

Software	Function
Microsoft Office Microsoft Word 2003 & 2007 Microsoft PowerPoint 2003 & 2007 Microsoft Project 2003 & 2007	 Prepare proposal and documentation Presentation Gantt Chart, schedule and planning
Web Browser Software: Mozilla Firefox Google Chrome	Find information in the internet
Visual Studio 2010	Do the prototype
Operating System	Microsoft Windows 7

Table 3.2 Software items that will be used for this thesis

3.6 Flow Chart

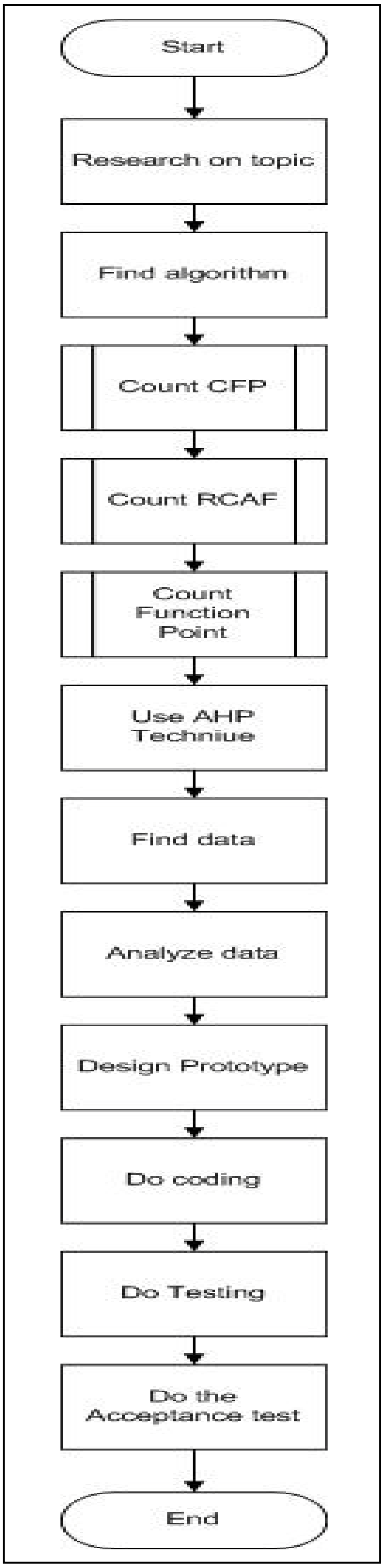


Figure 3.3 Flow chart

3.7 Gantt Chart

Please refer to Appendix A

CHAPTER 4

IMPLEMENTTATION

4.1 Introduction

This chapter discusses about the implementation of the system proposed thesis with the coding into the interface.

This phase is one of the important phases where the prototype is being developed based on the requirement that has been discussed in previous chapter. The data for this thesis was taken from UMP Automatic Sports Facilities Management System. Only two modules will be taken out of thirdteen modules in the project which is Booking module and Manage Facilities Module. The modules are taken based on the SRS (Software Requirement Specification) document. SRS is a document where it contains the description and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli.

4.1.1 Introduction about Booking Module

The booking module is one of the modules in the UMP Automatic Sports Facilities Management System where user needs to select the facilities and quantity first. Then proceed with the start booking date, start booking time, end booking date and end booking time. Then user needs to click save booking to validate the data and save the booking the database. If user does not want to continue the user can cancel the booking that the user wants to do by clicking cancel instead of save booking.

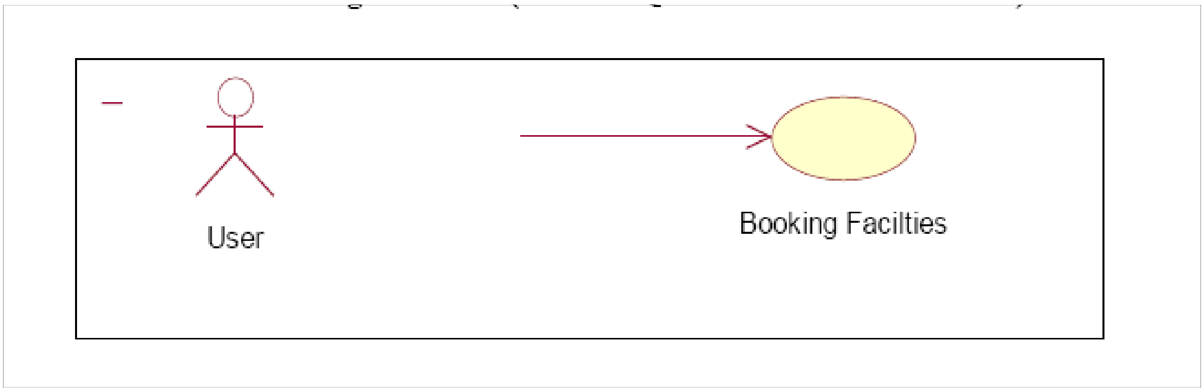


Figure 4.1 Booking Facilities Use Case Diagram

Basic Flow of the Module

i. User click the Booking Facilities Link
ii. Select Booking facilities(ex-Field/squash court)
iii. Then user enter the start booking date and time
iv. Enter the end booking date and time
v. Then save/ cancel booking to proceed
vi. If the Information entered is Invalid, an error message appears.
vii. If valid, information is saved in the database.

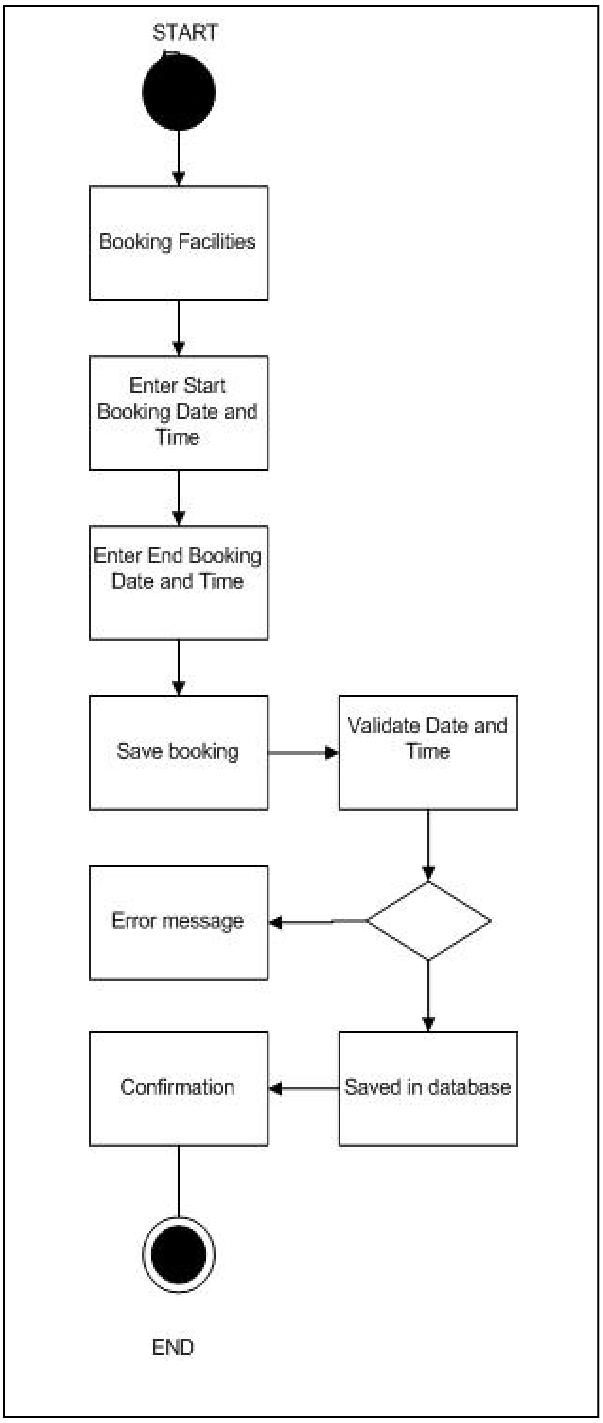


Figure 4.2 Booking Facilities flow

4.1.2 Manage Facilities Module

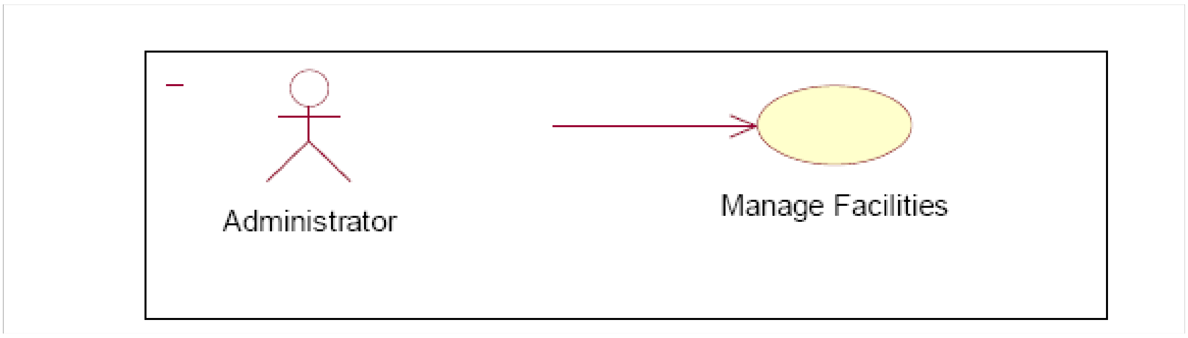


Figure 4.3 Manage Facilities Use case Diagram

Basic flow of the module

i. Admin click the manage facilities link
ii. List of facilities appear
iii. Choose the one function
    - Add facilities
        - Click add facilities
        - Fill in details such as facilities name, quantity, description and category
        - To save it click save / to cancel it click cancel and exit the application
        - If all the details valid, the data will be stored in the database
        - If there is an invalid detail the system will show error for that field fill in wrongly.
    - Edit facilities
        - Click on the edit image hyperlink
        - Admin can edit the field that he wanted to edit
        - Click on the update
    - Delete facilities
        - Click on the delete facilities link
        - A prompt message ask for confirmation

    - If admin proceed, the facilities that are not being used will be deleted.
- Update facilities
- Sort facilities
    - Click on the sort facilities link
    - Admin can sort which link he wants to be first, second and so on.

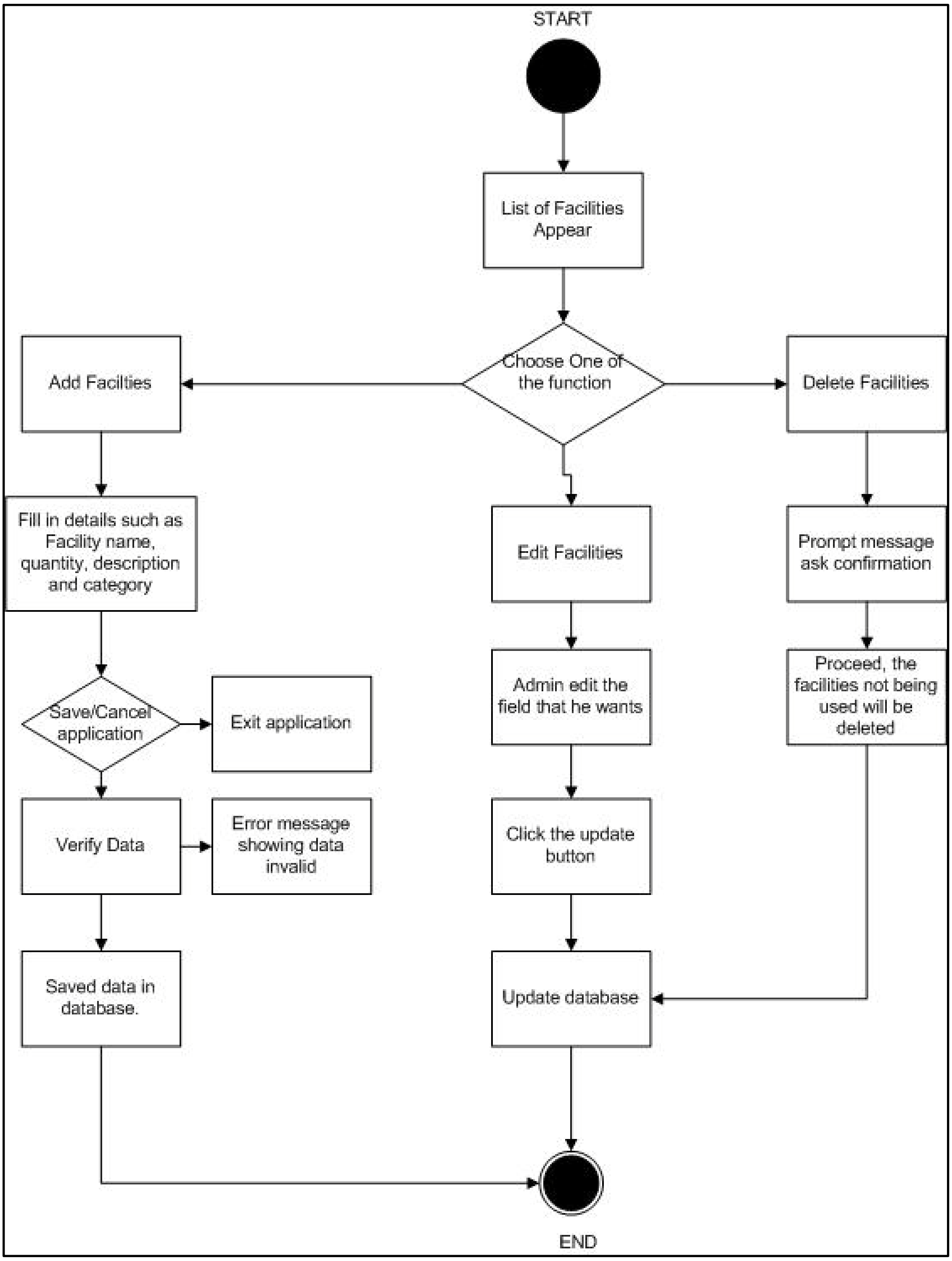

START
List of Facilities Appear
Choose One of the function
Add Facilties
Delete Facilities
Fill in details such as Facility name, quantity, description and category
Edit Facilities
Prompt message ask confirmation
Save/Cancel application
Exit application
Admin edit the field that he wants
Proceed, the facilities not being used will be deleted
Verify Data
Error message showing data invalid
Click the update button
Saved data in database.
Update database
END

Figure 4.4 Manage Facilities Module

4.2 Use Case Point Calculation

Use case point (UCP) modeling is a widely accepted estimation technique to capture the business processes and requirements of a software application. UCP is usually is used when the Unified Modeling Language (UML) and Rational Unified Process (RUP) methodologies are being used for the software design and development. The concept of UCP is based on the requirements for the system being written using use cases, which is part of the UML set of modeling techniques. The time to complete the application is affected by the number of steps to complete the use case, the number and complexity of the actors, the technical requirements of the use case such as concurrency, security and performance, and various environmental factors such as the development teams' experience and knowledge.

To determining the size estimate to develop a system using the UCP There are four steps to be followed which is

- Unadjusted Use Case Weight (UUCW) – the point size of the software that accounts for the number and complexity of use cases.
- Unadjusted Actor Weight (UAW) – the point size of the software that accounts for the number and complexity of actors.
- Environmental Complexity Factor (ECF) – factor that is used to adjust the size based on environmental considerations.
- Technical Complexity Factor (TCF) – factor that is used to adjust the size based on technical considerations.

4.2.1 Unadjusted Use Case Weight (UUCW)

It is calculated based on the number of use case and its complexity. Each classification of use case has a predefined weight assigned.

UUCW = (Total No. of Simple Use Cases x 5) + (Total No. Average Use Case x 10) +

(Total No. Complex Use Cases x 15)

= (1 x 5) + 0 + (1 x 15)

= 20

4.2.2 Unadjusted Actor Weight (UAW)

It is calculated based on the number and complexity of the actors for the system. Similar to finding the UUCW, each of the actors must be identified and classified as Simple, Average or Complex based the type of actor. Each classification also has a predefined weight assigned.

UAW = (Total No. of Simple actors x 1) + (Total No. Average actors x 2) +

(Total No. Complex actors x 3)

= (0 x 1) + (0 x 2) + (2 x 3)

= 6

4.2.3 Environmental Complexity Factor (ECF)

Environmental Complexity estimates the impact on productivity that various environmental factors have on an application. Each environmental factor is evaluated and weighted according to its perceived impact and assigned a value between 0 and 5. A rating of 0 means the environmental factor is irrelevant for this project; 3 is average; 5 mean it has strong influence.

Environmental Factor	**Description**	**Weight**	**Perceived Impact**	
E1	Familiarity with UML	1.5	4	6
E2	Application Experience	0.5	3	1.5
E3	Object Oriented Experience	1	5	5
E4	Lead analyst capability	0.5	2	1
E5	Motivation	1	1	1
E6	Stable Requirements	2	5	10
E7	Part-time workers	-1	0	0
E8	Difficult Programming language	2	1	2
Total				26.5

Table 4.0 Environmental factor table

ECF = 1.4 + (-0.03 x EF)

= 1.4 + (-0.03 x 26.5)

= 0.605

4.2.4 Technical Complexity Factor (TCF)

The TCF is one of the factors applied to the estimated size of the software in order to account for technical considerations of the system. It is determined by assigning a score between 0 (factor is irrelevant) and 5 (factor is essential) to each of the 13 technical factors listed

Factor	Description	Weight	Perceived Complexity	Calculated Factor
T1	Distributed system	0	2	0
T2	Response time/performance objectives	2	3	6
T3	End-user efficiency	2	3	6
T4	Internal processing complexity	1	2	2
T5	Code reusability	2	3	6
T6	Easy to install	0.5	5	2.5
T7	Easy to use	0.5	4	2
T8	Portability to other platforms	1.5	3	4.5
T9	System maintenance	1	3	3
T10	Concurrent/parallel processing	0.5	3	1.5
T11	Security features	2	3	6
T12	Access for third parties	1	5	5
T13	End user training	1	3	3
				47.5

Table 4.1 Technical Complexity Factor table

TCF = 0.6 + (TF/100)

= 0.6 + (47.5/100)

=1.075

4.2.5 Total UCP

There are few steps to be followed to calculate the total UCP. The first step is to add the UUCW value and UAW value together. Then the value is multiplied with the ECF and then the TCF.

UCP = (UUCW + UAW) x ECF x TCF

= (20 + 6) x 1.075 x 0.605

= 16.91

Estimated Effort = UCP x Hours/UCP

= 16.91 x 20

= 338.2 hours

Effort applied hour/per week

=338.2/30

=11.27 weeks

= 2.8 months

4.3 Basic COCOMO

The basic COCOMO'81 model is a single-valued, static model that computes software development effort (and cost) as a function of Program size which is expressed in estimated thousands of source lines of code (SLOC). COCOMO applies to three classes of software projects which are organic projects, semi-detached project and embedded projects. Organic projects means "small" teams with "good" experience working with "less than rigid" requirements meanwhile semi-detached projects means “medium" teams with mixed experience working with a mix of rigid and less than rigid requirements. An embedded project means projects developed within a set of "tight" constraints. It is also combination of organic and semi-detached projects including the hardware and software. The formula for basic COCOMO is stated below.

4.3.1 The Formula

The first step of basic COCOMO formula is we need to calculate the Effort Applied based on the given formula where we need to multiple the total KLOC with the coefficient of a_b before being powered by $^{b}{}_{b.}$

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

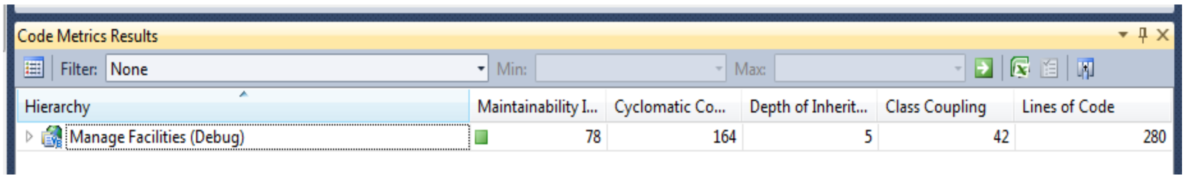
Development Time (D) = $c_b(Effort\ Applied)^{d}{}_{b}$ [months]

Since the KLOC is not very big and it’s just comprises 2 modules of a software project, it will be suitable to use organic class. The coefficients a_b, b_b, c_b and d_b are given in the following table

Software Project	a_b	b_b	c_b	d_b
Organic	2.40	1.05	2.50	0.38
Semi-Detached	3.00	1.12	2.50	0.35
Embedded	3.60	1.20	2.50	0.32

Table 4.2 Coefficient for basic COCOMO formula

Manage facilities



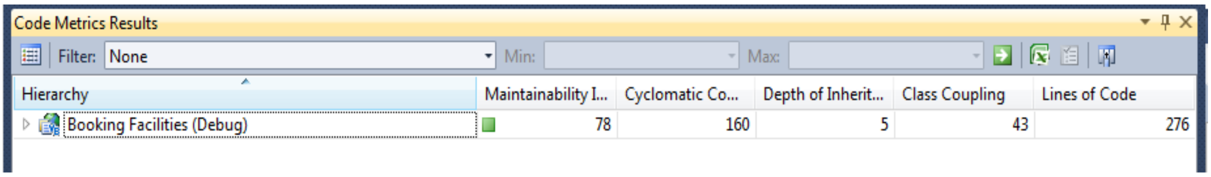
Code Metrics Results

Filter: None Min: Max:

Hierarchy	Maintainability I...	Cyclomatic Co...	Depth of Inherit...	Class Coupling	Lines of Code
Manage Facilities (Debug)	🟩 78	164	5	42	280

Figure 4.5 Total numbers of lines in Manage Facilities module

Booking Facilities



Code Metrics Results

Filter: None Min: Max:

Hierarchy	Maintainability I...	Cyclomatic Co...	Depth of Inherit...	Class Coupling	Lines of Code
Booking Facilities (Debug)	🟩 78	160	5	43	276

Figure 4.6 Total numbers of lines in Booking Facilities module

KLOC = (276 + 280) /1000

= 0.556 KLOC

Effort Applied (E) = $a_b(KLOC)^{b}{}_{b}$

=2.4 (0.556) ^ 1.05

=1.354

Development Time (D) = $c_b(\text{Effort Applied})^{d}{}_{b}$

= 2.5 (1.354) ^ 0.38

= 1.56 months

4.4 Function Point

Calculating the Crude Function Point (CFP)

Software System Component	Complexity Level									Total CFP
	Simple			Average			Complex			
	Count	Weight factor	points	Count	Weight factor	points	Count	Weight factor	points	
	A	B	C = A×B	D	E	F = D×E	G	H	I = G×H	J= C+F+I
User inputs	2	3	6	1	4	4	0	6	0	10
User Outputs	2	4	8	0	5	0	0	7	0	8
User Online Queries	0	3	0	0	4	0	0	6	0	0
Logical Files	1	7	7	1	10	10	0	15	0	17
External Interfaces	0	5	0	0	7	0	0	10	0	0
Total CFP	21			14			0			35

Table 4.3 Crude Function Point Calculation

RCAF

No.	Subjects	Grades					
1	Requirement for reliable backup and recovery.	0	1	[✓ 2]	3	4	5
2	Requirement of data communication	0	1	[✓ 2]	3	4	5
3	Extend of distributed processing	0	1	2	[✓ 3]	4	5
4	Performance requirement	0	1	2	[✓ 3]	4	5
5	Expected Operational requirement	0	[✓ 1]	2	3	4	5
6	Extend of online data entries	0	1	[✓ 2]	3	4	5
7	Extend of multi screen or multi operation online data input	0	1	2	[✓ 3]	4	5
8	Extend of online updating of master files	0	1	2	3	[✓ 4]	5
9	Extend of complex inputs, outputs, online queries and files	0	1	2	[✓ 3]	4	5
10	Extend of complex data processing	[✓ 0]	1	2	3	4	5
11	Extend that currently developed code can be designed for reuse	0	1	[✓ 2]	3	4	5
12	Extend of conversion and installation included in the design	0	[✓ 1]	2	3	4	5
13	Extend of multiple installation in an organization and variety of customer organizations	[✓ 0]	1	2	3	4	5
14	Extend of change and focus on ease of use.	0	[✓ 1]	2	3	4	5
Total = RCAF		27					

Table 4.4 RCAF calculation Point

FP = CFP × (0.65 + 0.01 × RCAF)

= 35 × (0.65 + 0.01 × 27)

= 32.2

Duration: 32.3 / 20 hours per week

: 1.61 weeks

~ Approximately 2 weeks

4.5 Estimation System Prototype

From the study that has been made, a prototype has been designed. This part will show the prototype interface with the explanation for each interface.

Main Page

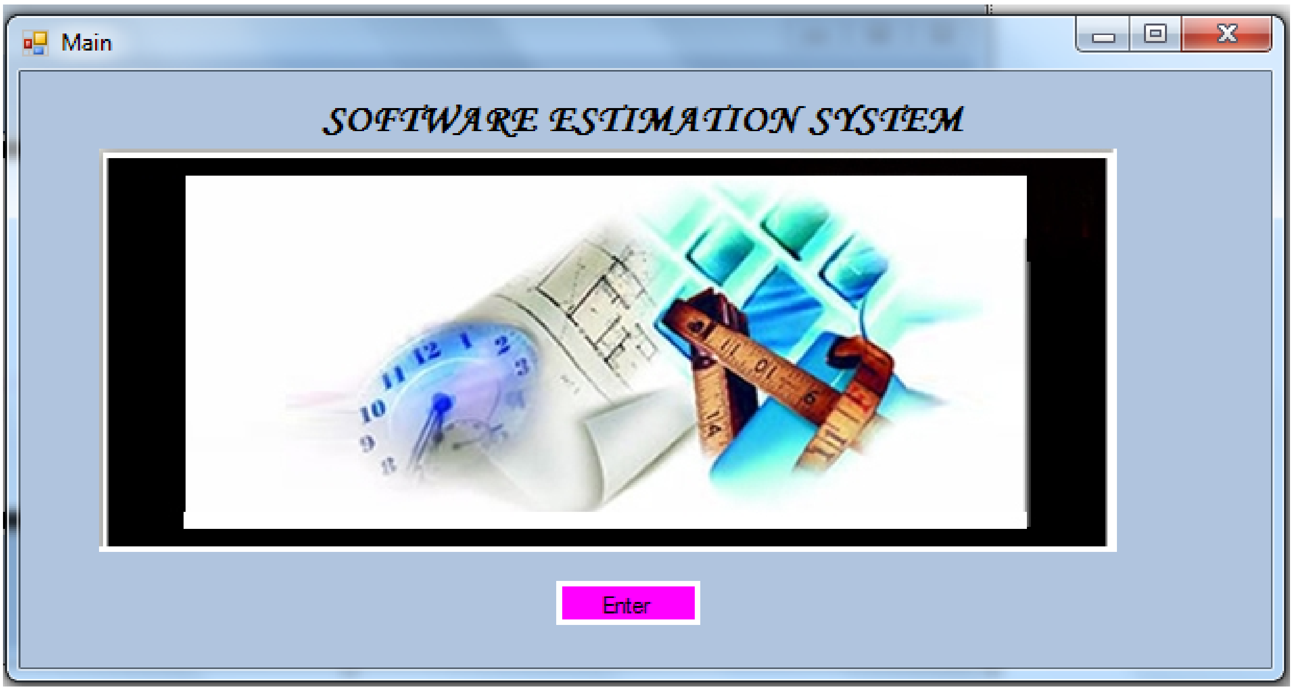
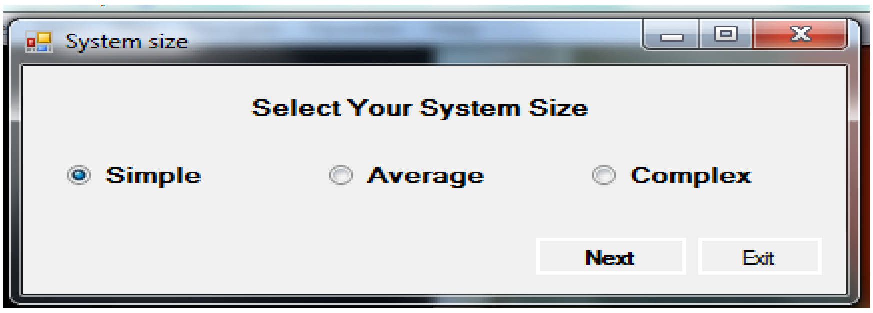


Figure 4.7 Prototype main page

This is the main interface of the prototype where there user just need to click the enter button to go to other interface.

Choose System Size Interface



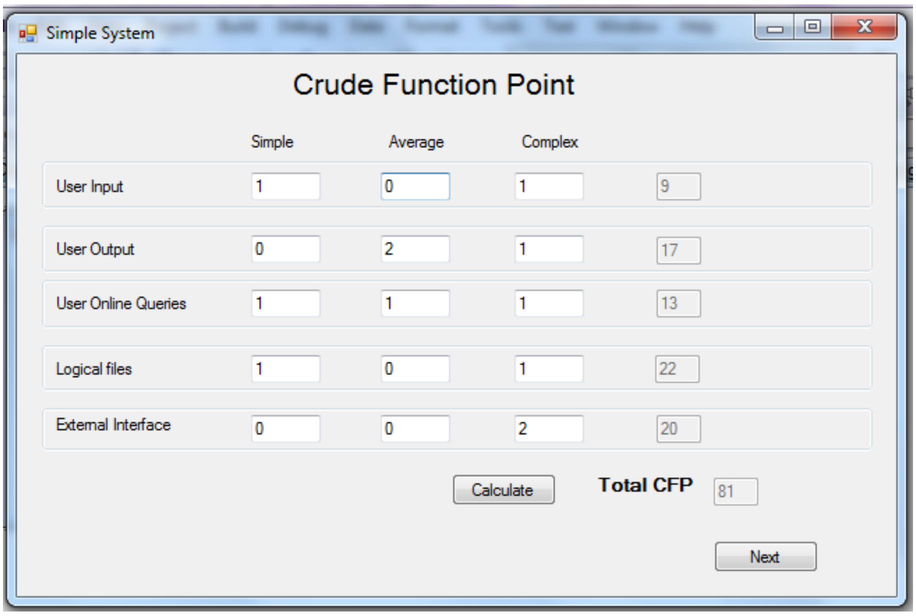
Select Your System Size

● Simple ○ Average ○ Complex

Figure 4.8 Choose system size interface

In this interface, the user needs to select the size of the user system. It is not going to be wrong if the user select the wrong system size because the user still able to do changes for the given criteria in the next interface and calculate the Crude Function Point.

Simple System Crude Function Point



Crude Function Point

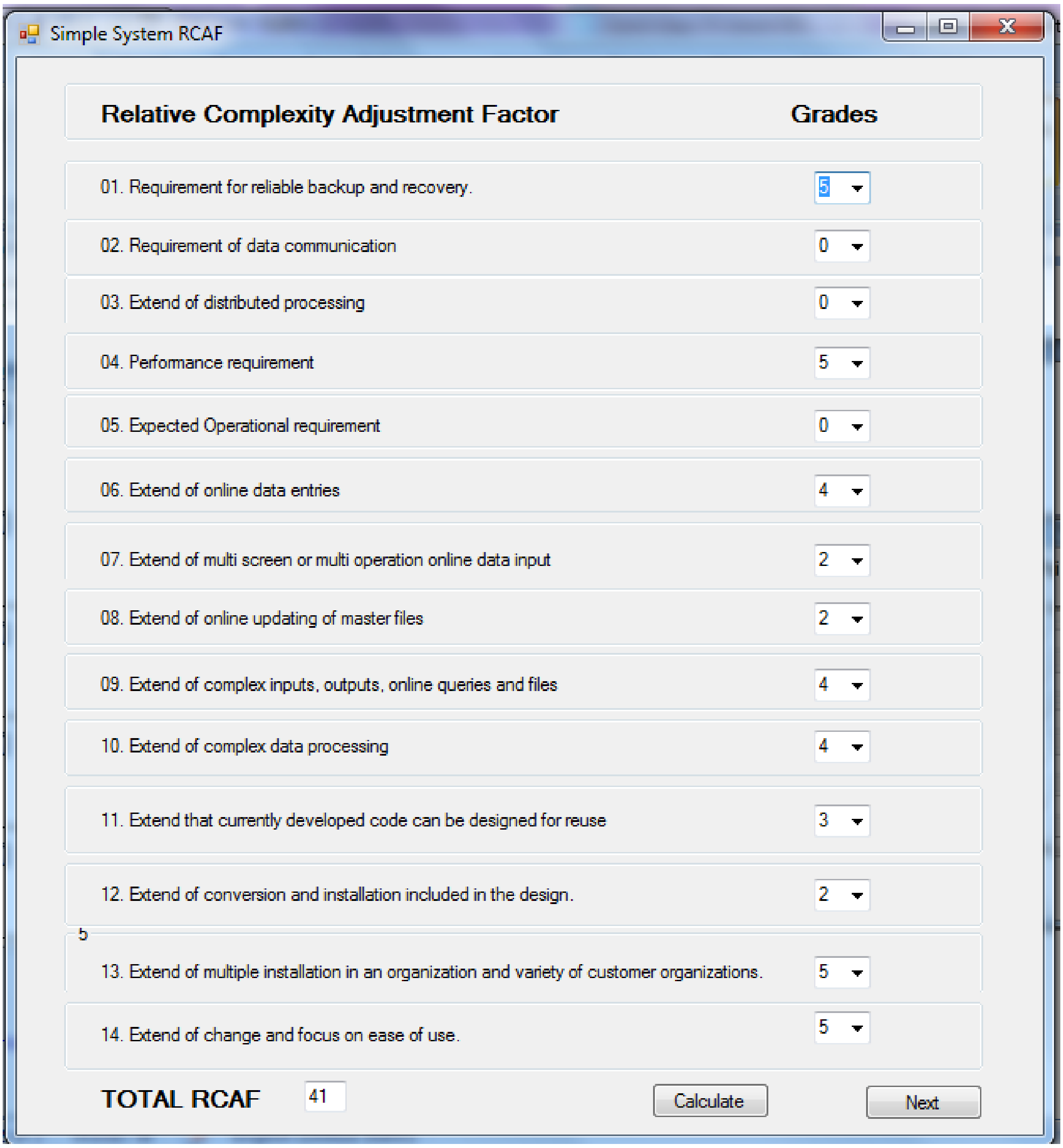
	Simple	Average	Complex	
User Input	1	0	1	9
User Output	0	2	1	17
User Online Queries	1	1	1	13
Logical files	1	0	1	22
External Interface	0	0	2	20

Total CFP 81

Figure 4.9 CFP calculation interface

This is the interface for the user to calculate the Crude Function Point. There has been value which is already inserted based on a simple project. If the user feels the value is not appropriate for the user system, the user can change it. Then click on the calculate button to calculate the total crude function point.

Simple system RCAF



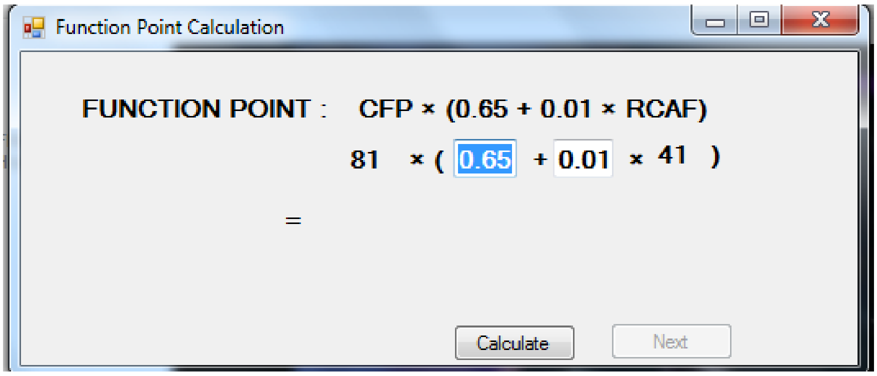
Relative Complexity Adjustment Factor	Grades
01. Requirement for reliable backup and recovery.	5
02. Requirement of data communication	0
03. Extend of distributed processing	0
04. Performance requirement	5
05. Expected Operational requirement	0
06. Extend of online data entries	4
07. Extend of multi screen or multi operation online data input	2
08. Extend of online updating of master files	2
09. Extend of complex inputs, outputs, online queries and files	4
10. Extend of complex data processing	4
11. Extend that currently developed code can be designed for reuse	3
12. Extend of conversion and installation included in the design.	2
13. Extend of multiple installation in an organization and variety of customer organizations.	5
14. Extend of change and focus on ease of use.	5

TOTAL RCAF 41

Figure 4.10 RCAF calculation interface

This interface will calculate the Relative Cumulative Adjustment Factor. Just as the previous interface the values are already inserted based on a simple project. If user feel the value is not appropriate to count the user's system the user can change the value based on the system.

Function Point Calculation

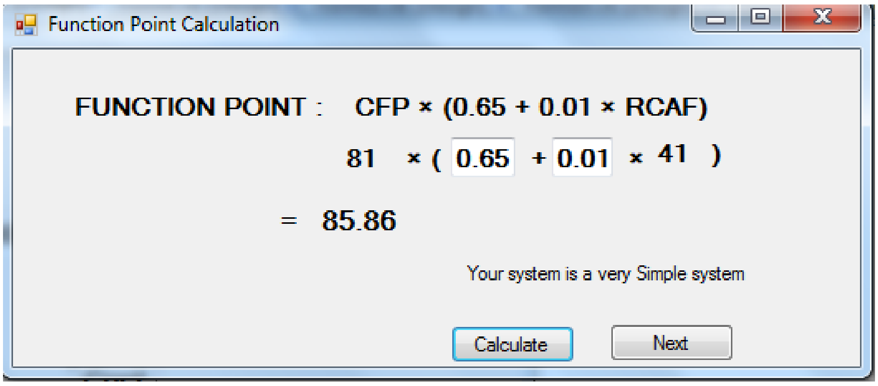


FUNCTION POINT : CFP × (0.65 + 0.01 × RCAF)

81 × (0.65 + 0.01 × 41)

=

Figure 4.10 Calculation Function Point



FUNCTION POINT : CFP × (0.65 + 0.01 × RCAF)

81 × (0.65 + 0.01 × 41)

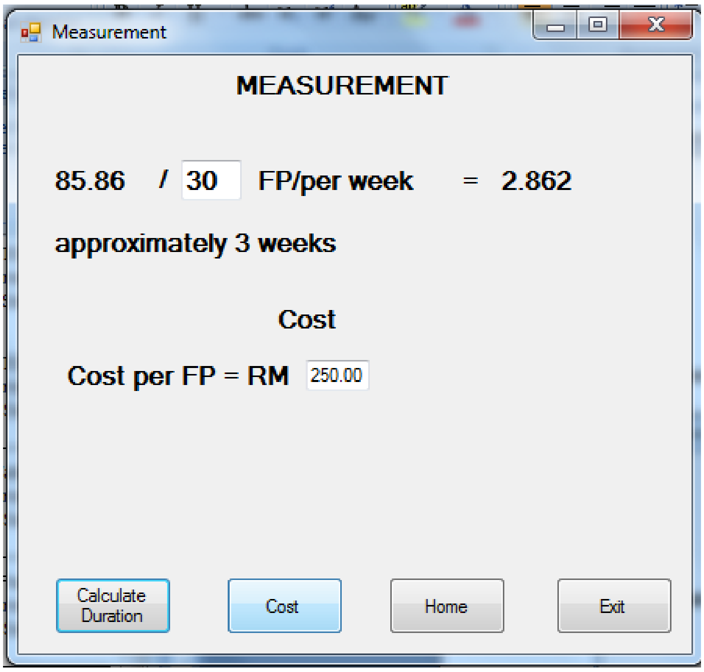
= 85.86

Your system is a very Simple system

Figure 4.11 The Next button is enabled after the calculation

The CFP value and RCAF that have been calculated earlier is brought forward to calculate the overall function point of the system. Once the Function Point is calculated then only the user is allowed to continue to the next page. Once calculated the system will show if the system is a simple, average or complex based on the Function point calculated.

Measurement



MEASUREMENT

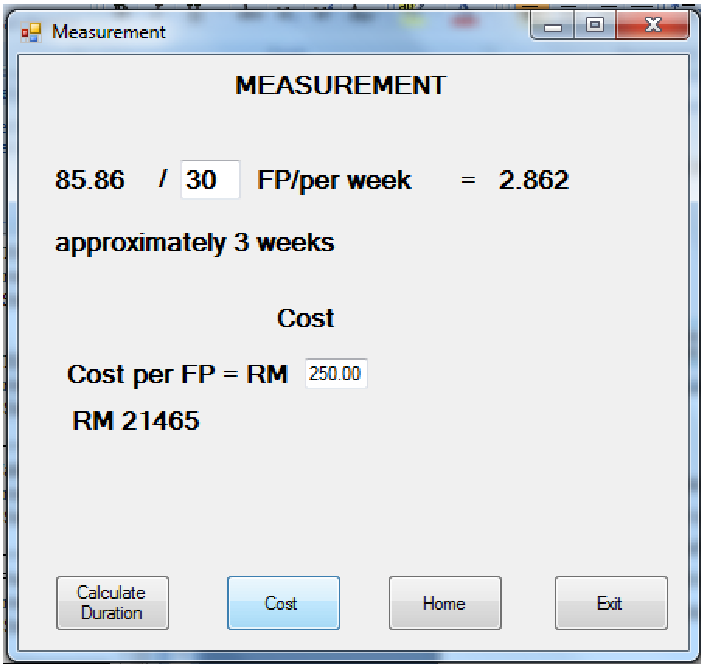
85.86 / 30 FP/per week = 2.862

approximately 3 weeks

Cost

Cost per FP = RM 250.00

Figure 4.12 Calculate Duration



MEASUREMENT

85.86 / 30 FP/per week = 2.862

approximately 3 weeks

Cost

Cost per FP = RM 250.00

RM 21465

Figure 4.13 Calculate the cost

This interface will count the duration to develop the system. The duration is calculated based on how many hour can the programmer can develop number of function points per week. Once the calculate duration button is clicked the calculated function point will be divided with the number of function point developed in hours per week. Then click on the cost button to calculate the cost to develop the system. If the developer feels the cost per function is too high or low the user can change cost.

4.6 Implementation Findings

Based on the study and comparison that has been done, it shows that among the chosen estimation method function points is the best method to be chosen as a software estimation technique. Two modules from a project have been decided to use in this study for finding out which of the three techniques among function point, use case point and basic COCOMO is the best technique. The result show when using function point technique, the software project can be done earlier compared with the other two techniques. The duration taken by function point to develop the two modules is approximately 2 weeks when compared with use case point where it takes 2.8 months and basic COCOMO 1.56 months. Thus, this proves that using function points for software estimation would be a good decision.

Chapter 5

DISCUSSION AND CONCLUSION.

5.1 Discussion

This chapter will be discussing on the result of the research on Estimation system Using Function point. Based on the prototype developed and after the testing phase, the prototype able to work without any error with the inputs from the users.

From the study that had been done on estimation system using function point, it is proven that the studies do support the title. The main purpose of this study was to prove that function point is the best estimation technique among the three techniques that had been proposed. The results do support the claims that function point is the best method to be chosen as the estimation technique in a software project.

Although function point do have few disadvantages it requires only those have enough experience in calculating the system size and needs great level of detail is required to estimate the software size but this can be overcome with the help from the professionals or senior manager in the company. A research paper shows that in Malaysia shows that based on their survey on 30 software companies in Kuala Lumpur and Selangor indicates that the "Expert Judgment" model achieve the highest average score in this research. It justifies that the project managers have the theoretical and practical knowledge of Expert Judgment in relation to estimation method. It also indicates that the Expert Judgment is the most popular method used. (Zulkefli, M., Zarinah, M.K., Habibah, A., Saadiah). There should be more exposure about function point and the significant advantages.

5.2 Constraints

One of the experienced constraints while doing this thesis is the need of Software Requirement Specification (SRS) document from the industries. The SRS document contains all the specific details of a project which makes it to be private and confidential. Sometimes before the company starts to develop the project, an agreement is signed between the two parties stating all the details regarding the projects should not be revealed out without owner's permission.

Besides, only limited people can use this technique. For function point technique, only people with more experience and qualified as professionals can use this technique. In order to be specific in estimating the size of software, function point needs all the specific details of a project accurately and only professionals know how to define the complexity of the project.

The prototype that was done is a standalone system. The user is tied to the specific platform for which the application is written. Upgrading the application to a new version involves installations on each workstation, which can become cumbersome for large organizations.

The source available for function point technique on the internet is not sufficient. There should be more examples on samples of project that have been estimated the size. Thus this will make one to understand about function point technique more.

The last constraint is those who are planning to use this function point technique should believe in this concept. Beliefs can be constructive and destructive. In order to count the size of software to be measured correctly, one should beliefs that function point will estimate the size accurately.

5.3 Conclusion

As a conclusion, Function Point (FP) is the best method that can be used for software cost and size estimation. Through the research and findings that was done, the objective have been achieved where the results of the findings shows function points is the best method to be chosen as the estimation technique. According to the results of several researches presented in this paper, the root cause for software project failures is inaccurate estimation in early stages of the project. So introducing and focusing on the estimation methods such as function points seems necessary for achieving to the accurate and reliable estimations.

Although FP have few disadvantage, but it does not give much impact to the estimation. By using FP as a method for software estimation, the project size and cost can be estimated accurately and efficiently. Besides that, FP value also allows user to estimating the effort, schedule and defect in the project. By having an accurate cost estimation and size in the project planning phase, the project can be delivered to the customer on time. Thus, this will maintain a good reputation of the company.

For future improvement, this technique should be applied in few projects in order to see the rate of success using this function point technique. In addition, the prototype should have more advance function where it can show the chart on successful project rate because this could be a factor to increase the confident of the particular person when estimating the size of the software. Furthermore, the prototype can be upgraded more where when there is too much different among the inputs in the crude function point, the system can remind the user about it.

Reference

Zulkefli, M., Zarinah, M.K., Habibah, A., Saadiah, Y. . 2011.Current Practices of Software Cost EstimationTechnique in Malaysia Context. ICIEIS Part I, CCIS 251, pp. 566–574.

M.A. Al-Hajri, A.A.A. Ghani, M.S. Sulaiman, M.H. Selamat,. 2005. Modification of standard function point complexity weights system," Journal of Systems and Software vol.74 ,195–206.

Iman.A, Ow.S.H . 2008. Project management Practices: The criteria for success and failure. Commuication of the IBIMA Volume 1.

Wikipedia contributors. 2012. Estimation. (Online) http://en.wikipedia.org/w/index.php?title=Estimation&oldid=517418923

C. Jones. *Proprammina Productivirv.* New York: McGraw-Hill, 1986.

Matson.J.E, Barrett.E.B.,Mellichamp.J.M., 1994. Software Development Cost EstimationUsing Function Points. *IEEE Transactions On Software Engineering*, 20. 275-285

Gao.X.,Lo.Bruce,.1995An Integrated Software Cost Model Based on COCOMO and Function Point Approaches. IEEE. 86-93.

Low,G.C. And Jeffery,D.R. 1990. Function Points in the Estimation and Evaluation of the Software Process. *IEEE Transactions On Software Engineering*. **16**. :64-71

Zheng.Y, Wang.B, Zheng.Y, Shi.L. 2009. Estimation of software projects effort based on function point. *4th International Conference on Computer Science & Education.* 941-943

Dekkers.T.: Function Point Analysis. *Software Estimation Series– sheet 2.*1-2

Carol A.D., Aguiar.M: Applying *Function Point Analysis to requirement completeness*. 8.

Greene.J. :Can Function Points Be Used To Real-Time, Scientific,Object-Oriented, Extreme Programmed, Or Web-Based Software. *IT metrics Strategies* vol. VII*, no. 7.* 1-16.

Bhushan, Navneet; Kanwal Rai (January 2004).*Strategic Decision Making: Applying the Analytic Hierarchy Process*. London: Springer-Verlag. ISBN 1-85233-756-7.

Linda M. Lair. M. Carol Brennan. 2006. ***Software Measurement and Estimation: A Practical Approach*.** Hoboken, New Jersey; John Wiley & Sons, Inc.

2013. Analytic Hierarchy Process (online). http://en.wikipedia.org/w/index.php?title=Analytic_hierarchy_process (21 March 2013)

2013. Source lines of codes (online). http://en.wikipedia.org/w/index.php?title=Analytic_hierarchy_process (11 September 2012)

Tichenor, Charley. "*Recommendations for Further Function Point Research."* SoftwareMetrics.Com (14 September 2012)

Yuri Marx Pereira Gomes (2006-2013). *Functional Size, Effort and Cost of the SOA Projects with Function Points*. http://www.servicetechmag.com/I68/1112-4 (21 February 2013)

IFPUG 1994. Function Point Counting Practices Manual, Release 4.0, International Function Point User Group, 5008-28 Pine Creek Drive, Westerville, OH 43081-4899, USA.

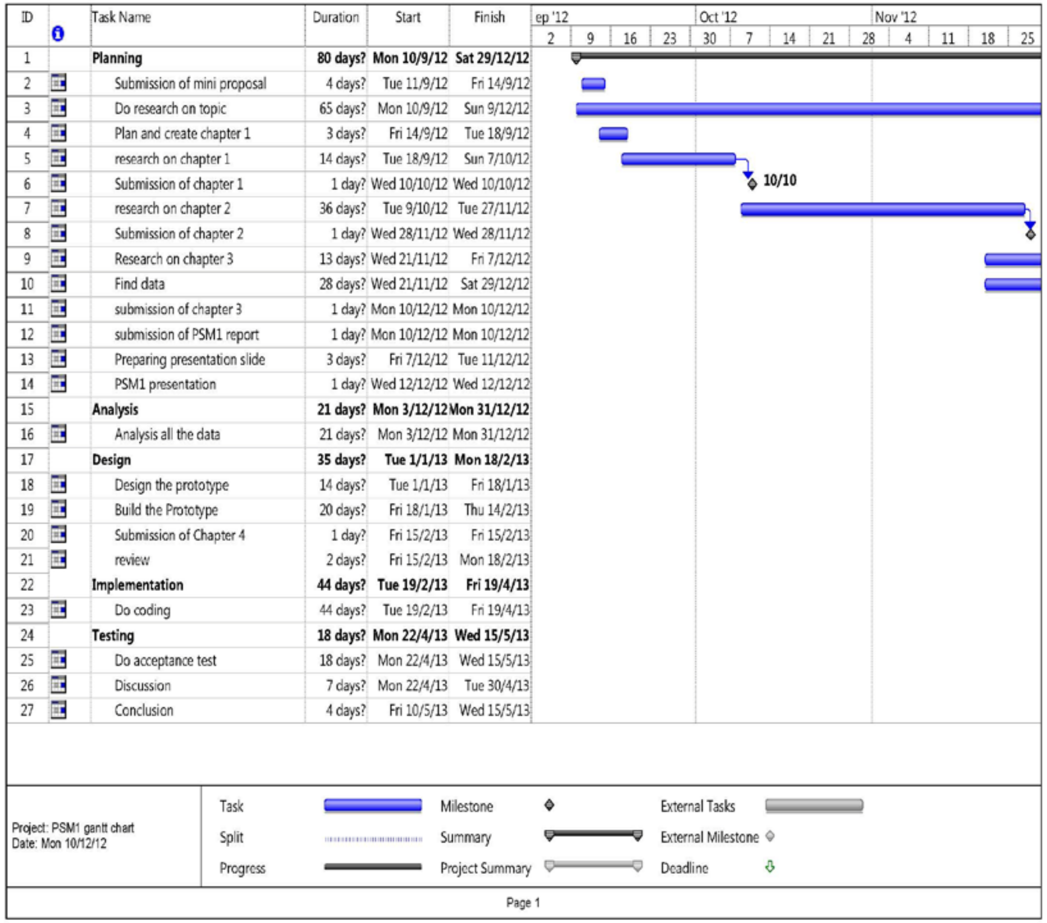
International Function Point User Group. What are function points. http://www.ifpug.org/?page_id=1143 . 20 November 2012

Rick Southard. 2000. Using Function Point Analysis. http://www.umsl.edu/~sauterv/analysis/function_point/UsingFPAS3.html. 22 March 2013. 22 Disember 2012

Alvin Alexander. 2013. How to Determine Your Application Size Using Function Points. http://conferences.embarcadero.com/article/32094. 15 Disember 2012

Abbas HeydarNoori. Function Point Analysis. https://cs.uwaterloo.ca/~apidduck/CS846/Seminars/abbas.pdf. 26 May 2013

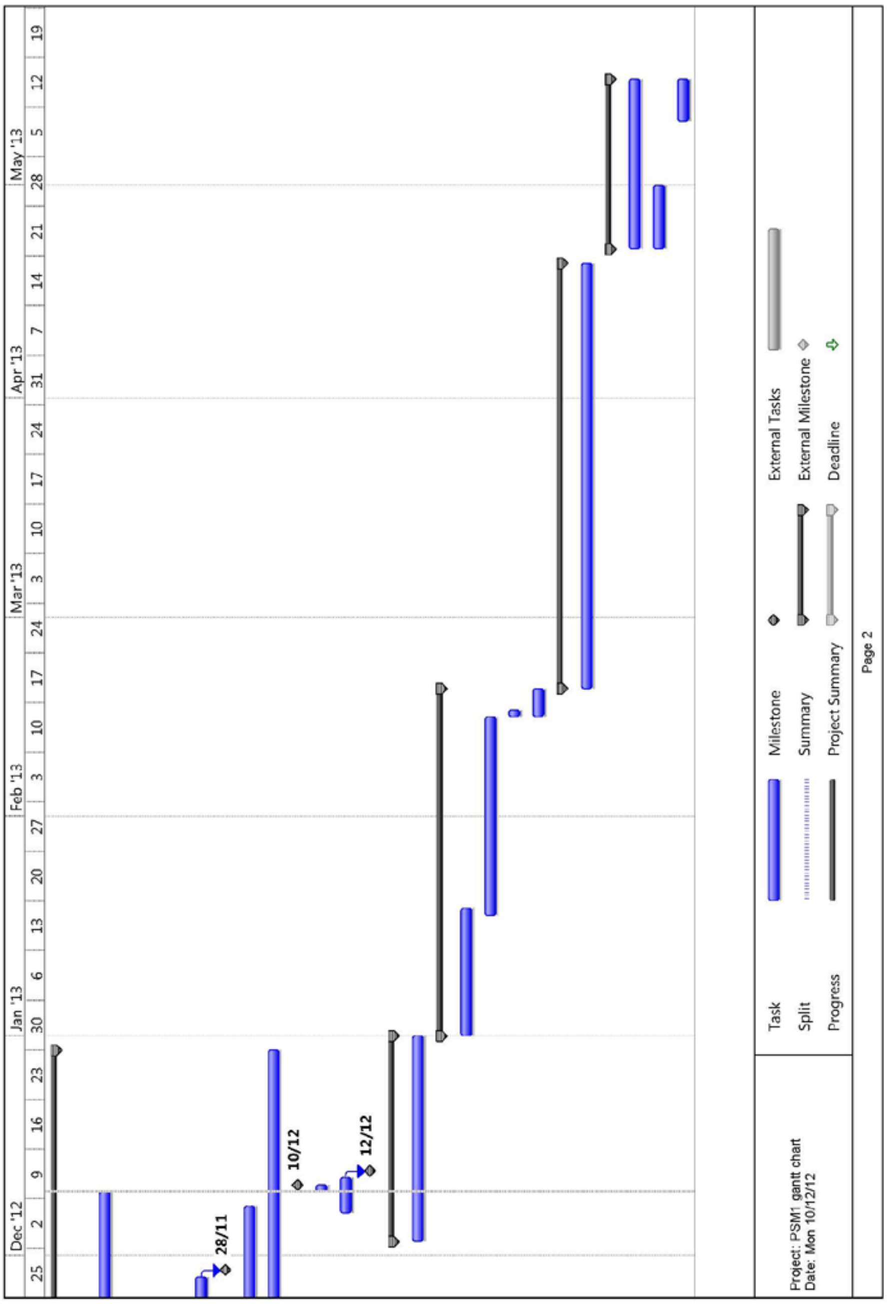
Appendix A



ID		Task Name	Duration	Start	Finish
1		**Planning**	**80 days?**	**Mon 10/9/12**	**Sat 29/12/12**
2	▦	Submission of mini proposal	4 days?	Tue 11/9/12	Fri 14/9/12
3	▦	Do research on topic	65 days?	Mon 10/9/12	Sun 9/12/12
4	▦	Plan and create chapter 1	3 days?	Fri 14/9/12	Tue 18/9/12
5	▦	research on chapter 1	14 days?	Tue 18/9/12	Sun 7/10/12
6	▦	Submission of chapter 1	1 day?	Wed 10/10/12	Wed 10/10/12
7	▦	research on chapter 2	36 days?	Tue 9/10/12	Tue 27/11/12
8	▦	Submission of chapter 2	1 day?	Wed 28/11/12	Wed 28/11/12
9	▦	Research on chapter 3	13 days?	Wed 21/11/12	Fri 7/12/12
10	▦	Find data	28 days?	Wed 21/11/12	Sat 29/12/12
11	▦	submission of chapter 3	1 day?	Mon 10/12/12	Mon 10/12/12
12	▦	submission of PSM1 report	1 day?	Mon 10/12/12	Mon 10/12/12
13	▦	Preparing presentation slide	3 days?	Fri 7/12/12	Tue 11/12/12
14	▦	PSM1 presentation	1 day?	Wed 12/12/12	Wed 12/12/12
15		**Analysis**	**21 days?**	**Mon 3/12/12**	**Mon 31/12/12**
16	▦	Analysis all the data	21 days?	Mon 3/12/12	Mon 31/12/12
17		**Design**	**35 days?**	**Tue 1/1/13**	**Mon 18/2/13**
18	▦	Design the prototype	14 days?	Tue 1/1/13	Fri 18/1/13
19	▦	Build the Prototype	20 days?	Fri 18/1/13	Thu 14/2/13
20	▦	Submission of Chapter 4	1 day?	Fri 15/2/13	Fri 15/2/13
21	▦	review	2 days?	Fri 15/2/13	Mon 18/2/13
22		**Implementation**	**44 days?**	**Tue 19/2/13**	**Fri 19/4/13**
23	▦	Do coding	44 days?	Tue 19/2/13	Fri 19/4/13
24		**Testing**	**18 days?**	**Mon 22/4/13**	**Wed 15/5/13**
25	▦	Do acceptance test	18 days?	Mon 22/4/13	Wed 15/5/13
26	▦	Discussion	7 days?	Mon 22/4/13	Tue 30/4/13
27	▦	Conclusion	4 days?	Fri 10/5/13	Wed 15/5/13

Project: PSM1 gantt chart
Date: Mon 10/12/12

Page 1


Dec '12
25
2
9
16
23
Jan '13
30
6
13
20
27
Feb '13
3
10
17
24
Mar '13
3
10
17
24
Apr '13
31
7
14
21
May '13
28
5
12
19
28/11
10/12
12/12
Project: PSM1 gantt chart
Date: Mon 10/12/12
Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline
Page 2