Thesis Report

Master's Program in Administrative Computer Science

University of Quebec in Montreal

Statistical Analysis on the Productivity of Data Processing with Development Projects using the Function Point Technique

by

Jean-Marc Desharnais

December, 1988

Table of Contents

Summary

The objective of this thesis was to determine whether it is possible to use the Function Point technique developed by Albrecht as the basis for a cost-estimating model for development projects in the field of management information services (MIS).

To accomplish this objective, we conducted a survey of 10 organizations based on 81 MIS development projects completed between 1983 and 1988. The size and complexity of these projects were evaluated using the standard parameters of the Function Point technique.

However, to render the measurement more consistent and less dependent on technology, we rebuilt a logical model based on the structured methods from recent methodologies for measuring the data instead of using the logical models from the databases in each organization. These models were rebuilt with the project managers.

The elapsed time of the 81 measured projects varied between 1 and 38 months (with an average of 11.8 months), their size and complexity between 73 and 1250 function points (with an average of 316 points) and the recorded effort level between 546 and 23,940 person-hours).

An initial series of statistical analyses enabled confirmation of the existence of a positive relationship between the size and complexity of the projects on the one hand, and the effort level required for their development on the other hand. Nevertheless, for the 81 projects in total, it was not possible to find statistically valid cost models that could explain over 54% of the variation in the required effort level.

Further analyses enabled isolation of the information services environment as the true second most significant productivity factor. To statistically validate this hypothesis, we separated the 81 projects into three groups according to their technical environment as follows:

- Traditional environment (46 projects)
- "Improved" traditional environment (25 projects)
- 4 GL or micro environment (10 projects)

The following model allows for estimating the cost of software development projects in a traditional environment:

Person-hours = 20.9 (function points) - 439

This model, statistically validated on a sampling of 46 projects, explains 60% of the variation observed in the required effort level.

The following model allows for estimating the cost of the software development projects in an "improved" traditional environment:

Person-hours = 20.0 (function points) - 989

This model, statistically validated on a sampling of 25 projects, explains 81% of the variation observed in the required effort level.

Due to the small size of the sampling, it was not possible to statistically validate a cost estimating model for 4 GL or micro environment.

As well, it was not possible to statistically show that one of the other measured factors had a significant influence on the other projects.

Introduction

This project began almost two years ago.

An initial proposal in the framework of the Master's program in management information services at UQAM was submitted nearly a year ago. At the time, 10 organizations volunteered for the research and agreed to the data collection.

All of these businesses have their main software activity Center in Quebec; these Centers are from small to large (50 to 500 employees in system development and system maintenance). These organizations are in the public (4), semi-private (2) and private (4) sectors.

The objective was to obtain measurements for 40 to 50 projects. This objective was exceeded; 82 projects were included in the sampling.

For each of the projects, we had to calculate the number of function points, compile the number of hours of work charged and obtain information on the work environment, the staff's experience and the users' special requirements for their project. These variables are explained tin greater detail in Chapter 2.

This document first introduces the theory of the Function Point technique. After a brief review it explains Function Point technique, the inherent limits of the technique and the criticisms expressed by some authors. The end of the first section presents the conditions that would enable the use of this metric for estimating.

The second section deals with the methodological approach using the Function Point technique for measuring projects, analyzing them according to various statistical techniques and finally validating the estimating models obtained. The variables of size, complexity and cost are also explained in this section.

The third section provides the results of the statistical analyses. The results backing the conclusions are included in this section with the ensuing interpretations. In particular, this section contains research results on a simplified cost model and the environmental influences on the costs.

Section One: Theory

This section explains the Function Point technique introduced by a brief review as well as the inherent limits of this metric. It concludes with the presentations of the conditions that would enable extending its use, not only for productivity but also for estimating.

1.1 History

Improving productivity has been a major concern of software managers for over 15 years.

To evaluate productivity, a relationship has to be made between the cost of the software project on the one hand and, on the other hand, the size and complexity of the software. Due to management methods practiced in software, the subsequent evaluation of the project cost is generally simple. The same cannot be said about evaluating the size and complexity of the developed software.

Many studies have been conducted to develop a simple and widely accepted method for measuring the size and complexity of projects (**1, 3, 6 to 11, 14**)[1]

During the 1970's and at the beginning of the 80's, the number of lines of code was the most frequently-used measurement. This measurement was satisfactory as long as the main software language used was very similar in terms of productivity.

As programming languages and techniques were being perfected and as they diversified, the use of the number of lines of code as the main metric became extremely limiting and paradoxical. Capers Jones provides the following example in this regard (**11**)

Lines of Code Paradox

Languages	Assembler	PL/1	APL
Lines of code	100,000	25,000	10,000
Effort (months)	200	100	80
Total cost	$1,000,000	$250,000	$125,000
LOC-months	500	250	125
Cost per LOC	$10	$20	$40

Table 1.1

This example shows that the number of lines of code taken in absolute value is not a very transportable productivity measurement. To develop the same work, the total cost with APL language is 8 times less than with Assembler language; yet, the unit cost of Assembler is four times less.

Towards the end of the 1970's, researchers McCabe and Halstead, knowing the limitations of the number of lines of code as a metric, proposed various approaches based on measuring the intellectual effort of the software staff (**7, 9, 14**).

[1] The numbers in bold refer to the bibliography.

In trying to avoid the limitations and paradoxes of the lines of code calculation and through indirect backing from the idea of measuring intellectual effort, Allan Albrecht developed the Function Point Analysis (**1**). There is also an original perspective to his approach, that is, to measure the project's size based on the services delivered to the user and not on the technical effort required by the software staff.

Two factors promoted the popularity of this technique:

- The increased power of computers: this factor contributed to freeing software of purely technical concerns (limited programming due to insufficient memory) and thus enabled them to focus on concerns that were oriented more towards the users' request
- The arrival on the market of new programming languages that were easier to manipulate. This factor reinforced the tendency to consider "the users' viewpoint", but it also heavily complicated measuring the size of the software with lines of code (see example on the paradox in the number lines of code).

These two factors contribute towards emphasizing two rules in composing a metric for measuring the size of a project:

- Measure the product delivered in terms of services provided to the user (output) instead of in terms of programming (input)
- Measure the size and complexity of the product independently of the technology (software, hardware) used to develop it[2].

As regards the use of Function Point technique in North America, we should note the creation of a users' group in the fall of 1984. This group was formed in Toronto and it included, among others, Bell Canada, the Royal Bank, the Bank of Nova Scotia, AT&T… It chose the name IFPUG (International Function Point Users Group). Its members also come from the U.K., Europe, Australia and New Zealand. Canadian participation is very large in the group and this is reflected in the steering committee as well. The chairperson at the last two meetings (Dallas in May, 1988, and Montreal in October, 1988) was Ms. Darlene Brown of the Royal Bank of Canada. The group has also made its mark on other recognized groups like QAI (Quality Assurance Institute) and Guide (IBM mainframe users group).

1.2 Definitions and Criticisms

This section gives a brief definition of Function Point Analysis (FPA).

1.2.1 Definition of the Function Point Technique

The Function Point Analysis technique aims to measure in a standard way the size and complexity of software applications; this technique is mainly applied to business software development projects.

[2] This metric aims to measure development projects. It has also been extended to maintenance projects in certain areas.

The principle of the technique is based on 5 components of information processing: input, output, inquiries, internal logical files and external logical files.

This technique consists in taking inventory of the number of components[3] of each type and determining its complexity based on a predetermined scale (low, medium and high).

Weighting of Components by Complexity

Components	Low Complexity	Medium Complexity	High Complexity
Internal Entities	7	10	15
External Entities	5	7	10
Input	3	4	6
Output	4	5	7
Inquiries	3	4	6

Table 1.2.1

Table 1.2.1 shows the weighting (in points) applied systematically to the results of the inventory of the components. The size of the application is the sum of the points accumulated by each of the components. The measuring rules in the second section of this document explain this technique's operation in greater detail.

1.2.2 History of the Function Point Technique

The Function Point technique did not appear in its current form in 1979 (**1**). The first version of the measuring model included the 5 components as described in 1.2.1 but the calculation mode was simplified. Here is a diagram of this calculation mode:

Albrecht's Model 1979

[3] The expression "Function Types" is also used.

The 1979 model attributes 4 points for each input (e.g. screen), 5 points for each output (e.g. reports) etc. Three global complexity factors are added to this basic definition. These factors are compiled globally in the project and enable adjusting it to more or less 25%.

To be specific, there are two types of logical files (or entities) according to this technique:

- Internal logical files that correspond to the entities[4] maintained by the application
- External logical files that correspond to the entities consulted by the application.

There are three types of transactions:

- **Input** that allows the addition, modification and destruction of data in the internal files
- **Output** that reflects the modifications made to data in the internal files
- **Inquiries** that allows the user to obtain results from the data.

It was only in 1984 that the calculation algorithm shown on 1.2.1 (**2**) was introduced. This new model contained a complexity calculation for each of the components. The complexity calculation was based on the number of elements (or attributes) included in the component and the «relationship» between these elements and the other components in the application, mainly in the logical files.

Also the technique included the use of a questionnaire containing 14 general questions tied to the features required by the user for the project measured. This questionnaire appears in Appendix B. The rules for using this questionnaire are explained in Section Two of this paper.

Albrecht's Model 1984

The algorithm in Albrecht's model was used to measure the 82 projects. We should note that the points in each of the components might differ according to whether this component is simple,

[4] The term "entity" is preferred to the term «logical files». The reasons for this preference are explained in Section Two of this paper.

medium or very complex. The number of attributes associated with the component and the number of relationships of this component with the entities determine its complexity.

At this time, large corporations showed a greater interest in the technique and formed a users group called the International Function Point Users Group (IFPUG).

One of the Group's missions was to enable new users to correctly use the technique. Despite four years of constant work on the part of the steering committee members and several other volunteers, there are still no official rules for the Function Point calculation, only guidelines.

The consequence of this situation is that the numerous organizations still do not have the same calculation basis and that this basis may even vary within a company. However, there are two main tendencies which are not very compatible in applying the basic rules:

- The source and definition of the components come from the organization's type of database; for example, IBM uses the IMS database as its reference point
- The source of the components comes from the organization's logical (or conceptual)[5] models. This was proposed by Ken Zwangiz (**19**) and Charles Symons (**17**).

At the IFPUG conference in October, 1988, the steering committee, with the support of Allan Albrecht, ruled on the need to establish a Function Point measurement standard based on the logical models. In April, 1989, this orientation should be definitive and standards should be adopted.

A third model was introduced by Charles Symons at IEEE in January, 1988 (**17**). This model again used the algorithm shown previously adding environmental factors. The environmental factors are not related to the users' requests; instead they are related to the technical conditions of the project's development. The inclusion of environmental factors enables the use of the Function Point technique for estimating projects while taking into account their development conditions. This table will be taken up again in Section 2.2.2 of this paper (the measurement parameters).

[5] The distinction between the conceptual model and the logical model is not always clear in the organizations' practices. Two of these papers attempt to clarify these concepts.

Symons' Model 1988

In the articles cited above, Symons also introduced a calculation technique based on Function Point called Mark II. The technique proposed differed largely from Albrecht's technique, both in the approach and calculation modes. That is why IFPUG's steering committee rejected Symons' proposal in the fall of 1988.

1.2.3 Intrinsic Limitations of this Metric

This section describes the metric's main intrinsic limitations as perceived during this survey and through a reading of certain basic documents (**1, 2, 3**).

- The algorithms used to determine the size of the projects were set up based on the observation's made by Albrecht on 23 business projects (**1**), which means that there may be a bias in the project measurement that was not anticipated in the sampling (e.g., scientific project measurement)
- The number of points attributed to the components, according to their complexity, does not reflect the project's technical complexity; instead, it reflects the complexity as seen by the user
- The approach quantifies what is delivered to the user, but does not qualify it. The Function Point technique measures the size of projects, not their quality in this sense
- There is room for interpretation in the counting of function points whatever the measurements' starting point[6]

[6] There is room for interpretation in the application of the basic rules, whatever the approach, for the following reasons:

- The questions concerning the scope factors are multiple choice, thus subject to interpretation by the respondent
- The choice is translated by a degree of influence between 0 and 5. This choice expresses the user's thoughts on the project's complexity as regards to the questions. It is of a qualitative nature. However, we use it in the calculation algorithm as a quantitative value
- The breakdown of the projects may influence the points calculation as regards the external entities[7].

1.2.4 The Criticisms

We should examine the criticisms of Boehm and Symons in light of these limitations. The two researchers base their criticisms on two very different approaches:

- Boehm is an ardent defender of measuring project size with the number of lines of code and emphasizes that the Function Point technique does not solve the weaknesses in using the number of lines of code
- Symons used the Function Point technique in the beginning of the 1980s when he was at Xerox. He identified a certain number of weaknesses in Albrecht's technique and proposed a new solution: Mark II.

A) Boehm's criticisms

Barry Boehm (**4**) mentions six weak points related to the number of lines of code and indicates that the Function Point calculation technique only answers 2 of these points. It is inferior to using the lines of code for two others. According to Boehm, this metric is only worthy for the small and medium-sized projects.

The limitations usually given for project size measurement techniques based on the number of lines of code according to Boehm are:

1. They are at too low a level for certain objectives, like estimating
2. They are too high a level for other objectives, the more complex calculations having the same weight as other less complex ones
3. This metric is not uniform; it varies from one language to another

---

- When the source and definition of the components come from the type of database, the interpretation comes from the wide variety of databases on the market. The technological changes may cause different results in the calculations. For example this is what IBM recently experienced in trying to measure systems built with DB2.

In the approach with entities, the measurement is also subject to differences in viewpoint as to structured methods, but to a much lesser degree. The analyst's viewpoint nevertheless plays a part which should be studied in more detail when a larger number of projects measured in this way is available.

[7] The project's breakdown may influence the counting regarding the external logical files. If we decide to break down an application into 3 smaller applications, for one of the small applications, the use of a file belonging to the other two small applications is calculated as an external logical file. The calculation of the global application does not count this file since it is part of the same application. Thus, the function points total for the 3 small applications is greater than the total of points for only one global application, meanwhile the deliverable is the same product. This criticism comes mainly from Charles Symons (**17**); IFPUG refers to this matter as being an «associativity» problem.

4. It is hard to suitably define what should and should not be counted
5. It is not well correlated with what it delivers (added value); the programmers may try to multiply the lines of code uselessly
6. The quality of the software is not taken into account.

According to Boehm, the Function Point technique properly addresses problems 1 (estimation) and 3 (uniformity). This metric has no advantage over the lines of code for problems 5 (added value) and 6 (quality). As for problems 2 (complexity)[8] and 4 (definition), this metric has more difficulties than lines of code.

B) Symons' criticisms

Charles R. Symons highlights some limitations of the Function Point technique (**17**) and proposes a different approach called «Mark II».

The main criticisms are:

1. The classification of Function Types into simple, medium and complex only simplifies the adjustments. This classification does not reflect the entire complexity of the work necessary to develop the user's systems
2. Although discussed and justified, the choice of weights (points per type of component) has not been validated in environments other than those of IBM
3. All of the measurements are tied to the complexity of the internal processes if we take into account that the logical files are tied to the physical structure of the files (IMS type)
4. Complexity is also taken into account in the 14 questions regarding the scope factors, adding confusion to the notion of complexity
5. Project breakdown plays an important role in the total of points (see preceding note on project breakdown)
6. The number of factors is limiting. As well, some overlap (e.g., on-line update input and data communications) and others require more precision (e.g., end-user efficiency, installation ease)
7. The number of degrees of influence (0-5) is sometimes clearly insufficient.

1.2.5 Advantages of the Function Point Technique

Despite its weaknesses the Function Point technique is still the most suitable metric for measuring Function Point Analysis projects. Capers Jones stated recently that the Function Point technique is «the most precise technique for measuring the productivity of information systems in all of computer history» (**12, 20**). The main benefits of this technique are that it allows measuring software projects:

- Without reference to the technology used (independently of the technology)

[8] This is why the Function Point technique should only be applied to business software. Capers Jones proposes a parallel technique that would more suitably measure scientific systems. He calls this Feature Points.

- In the software manager's language (measuring the system's output)
- In user's language (screens, reports, etc.)
- In an easy way (a number of points)
- Quickly (calculation is made in a few hours)
- And relatively early in the project's development process (at the beginning of functional analysis).

As well, several techniques can be used to minimize the weaknesses highlighted by Boehm and Symons. Here is what was done in the framework of this research:

- Use of an approach that is more independent of technologies than that used by IBM among others. This means abandoning measurements based on IMS-type files and adopting a measurement based on logical models[9]
- Consideration of small and medium-size business projects. In this way we compare similar projects
- Determining an interpretation mechanism for the scope factors
- Using same level entities in the logical models
- Considering only what the metric attempts to measure.

1.2.6 The value of the Estimating Technique

The Function Point technique was first used for measuring productivity. The models of Zwanzig and Symons allow a more direct estimating approach by introducing environmental factors. However, neither of the researchers compared the method's validity with other approaches, or measured a sufficient number of projects to statistically validate their approach. Ken Zwanzig had only measured twenty or so projects before his death in January, 1986. Charles Symons had not measured more than twenty projects with the Mark II technique in January, 1988, at the time of the publication of his research at IEEE.

A) Chris Kemerer's Study

Research at the university level was conducted to validate the Function Point technique for estimating in comparison with 3 other techniques (**13**). To that end, Chris Kemerer measured approximately 15 software development projects and compared the results of the estimates according to the algorithms proposed by 4 estimating techniques used extensively by North American organizations: SLIM, COCOMO, Function Point (Albrecht's model 1984) and Estimacs.

Kemerer's study concluded that the Function Point technique, despite its weaknesses, is still the most valid of the main metrics proposed on the market for measuring the relationship between effort and size attributed by the metric.

Three other conclusions should be noted:

[9] The term "conceptual model" is also used. See the measurement rules.

- Albrecht's estimating model is valid on data other than that which he used in his study
- The number of "adjusted" points, resulting from answers to the fourteen questions (scope factors) does not provide a better explanation than the number of "non-adjusted" points. In other words, the result of the regression between the efforts and the non-adjusted points and the efforts and the adjusted points is similar
- Software other than business software presents additional difficulties in estimating.

B) Comments on Kemerer's Study
As mentioned, Kemerer was not the first to have thought of using the Function Point technique for estimating. Nevertheless, he was the first to conduct a comparative study of the estimating methods including the Function Point technique in his analysis.

His conclusions are a tremendous learning experience in the framework of this synthesis:

- The Function Point technique is well adapted to measuring the size of business software
- It is possible to compare measurements coming from different organizations
- The scope factors, as considered in Albrecht's algorithm, do not provide convincing results[10]
- The model's validity has yet to be determined the entity-based measurements[11] .

1.3 Research on a Cost Model

This research is intended to establish a cost model based on the Function Point technique. However, validating this type of model requires certain basic conditions summarized as follows:

- Measuring what Albrecht wanted to measure, that is the components requested and seen by the user
- Ensuring oneself of the consistency of the measurements throughout the projects and the organizations
- Ensuring oneself of the consistency in recording the efforts (correct records of time by the individuals, definition of the efforts considered in calculating time by the organization).

[10] It seems that these factors are not used by IBM for measuring since past results did now show appreciable differences in the calculation of points with or without the factors.

[11] One of Symons' criticisms is that this cost model has never been validated outside of IBM with projects measured on a basis other than the IMS-type files. On the groundwork of 15 projects, Kemerer suggests that this basis is valid with projects other than those used by Albrecht. It is even more valid than three other models proposed by other researchers. Since, in all likelihood, Kemerer measured the projects according to the same basis as Albrecht (IMS-type files), Symons' criticism regarding technological dependency of Function Point still holds true.

Section Two: Methodology

The purpose of this section is to give a brief description of the basic measurement rules. The rules described in this section are based on Albrecht's 1984 document (**2**) and Ken Zwanzig's document, also published in 1984, and which is presented in the Group's Guide. They are also inspired from the rules that are part of a knowledge system developed in the fall of 1988 by DMR Group Inc. These rules are based on the measurement of a hundred or so projects comprising various database types (relational, hierarchical and network) in several organizations and on the use of a structured approach.

2.1 Measurement Rule

The purpose of this section is to introduce a summary of basic rules for measures rendered within the framework of the thesis report.

2.1.1 Measurement Rule Objectives

The basis of the measurement rules is to be found in the Albrecht and Zwanzig documentation.

The rules must make possible:

- The measurement of what the user asks and receives
- An assessment of independent technology
- The reduction of measurement time to a minimum for all participants
- The consistency of measurement independently of time, project and individual
- The obtaining of a sufficiently accurate measurement (correlated to effort).

2.1.2 Summary of Important Rules

Information processing can be partitioned into two distinct parts: **data** and **transactions**. For Allan Albrecht, data is divided into internal and external files. Charles Symons and Ken Zwanzig prefer the use of the term **entity** rather than the expression **logical record** because it corresponds more closely to user views and to current approaches in information processing. The term entity will be used here. Transactions consist of input, output and inquiries. The application's technical complexity corresponds to the 14 factors relating to scope, also called «General System Characteristics».

Information Processing

Table 2.2

Information processing highlights, through transactions, the relationship between users and the application. Entities are part of the relationship with the user, through the applications.

A) The Concept of Internal and External Entities

The concept of entity refers to the construction of a conceptual data model within the framework of structured approaches. It differs somewhat from Albrecht's logical model concept. For the latter, the logical model is constructed according to the database used to develop an application. In an IMS-type file, for example, the logical file corresponds to among other things the root segment. However, the root segment is identical to a conceptual model entity.

Since data measurement is based on entities, it is important to clearly define this concept according to the context.

Data Model «Level»

First of all, what model is concerned by the measurement? The conceptual model or the logical model ? Practice has demonstrated that the most accurate counting is carried out using the data model at its lowest, most detailed level prior to the application of any access or performance changes on specific software (DBMS). Some refer to this level as «conceptual» and others as «logical».

Let us take note that the data model in question here is not that of the entire organization. It consists only of the part required for the measurement of a specific application. The information produced by the data model is that which is of interest for its business activities.

Definition of Entity

An entity contains a certain number of data related to the interpretation of the user working within an organization. The entity is thus not linked to the manner in which the information is processed, which also differentiates it from the logical record used in databases. This entity is called «primary entity» by Charles Symons. Charles Symons distinguishes three types of entities:

Primary entities:	They make up the primary concern level of an application. They are typically the subjects of the database (root segment, primary tables).
Secondary entities:	They consist of data grouped into tables (parameters) and they usually contain an ID code and a description.
Other entities:	They consist of data grouped in a summary file for consultation.

In entity computation with the Function Point technique, according to Symons, only primary entities should normally be considered. A review of current practice, according to Albrecht's rules as well as to Zwanzig's rules, shows that this definition is not adequate and does not correspond to the continuity of measurement needs of organizations having already adopted this technique.

Entities and Relationships

In an entity relationship model, such as the one presented below, the primary entities according to Symons' definition would be: project, values and phases. In this example, we can note that between projects and values there is a relationship of many to many (more precisely 0,n and 1,n) known as «**computation**». This relationship is required by the user to better «manage» the entities, which are directly linked to the organization's business practices.

Entity Relationship Model

These relationships will often contain data not represented elsewhere in the model. This relationship brings about the creation of an additional entity and appropriate relationships in the logical model at the functional analysis level.

There are other relationships, such as divide, which are not many to many and, consequently, do not create new entities at the logical model level.

In practice, it is also from many to many relationships that one can infer which tables are required by the user to better manage the entities that are directly linked to business practices. The term entity thus appears on two levels:

- The business level, where entities concerning the organization's business practices come into account
- The management level, which represents the entities perceived by the user as being an entity management aid relating to business practices.

How can one distinguish an entity at the technical file management level? The entities concerned here appear at the beginning of functional analysis, i.e., prior to the start of any optimization relating to the use of the database selected to support the application. However, the following question arises: Does this entity exists to facilitate the management of entities linked to business practices or because it is necessary to optimize database use? For example, in a relational file, the analyst may decide it is necessary to index the names of those responsible for a project to optimize disk space use. However, the user does not require this index because his purpose is not to oversee project managers.

Why is it necessary to compute two levels of entities? There are two main reasons for this:

- Management entities are also part of user views. Even though they do not reflect his business practices, they are "logically" required by the user to manage his application. In this respect, entity computation is in line with the basic philosophy of the Function Point computation technique, which is to reflect user views
- Current practice, thought imperfect, considers these entities as part of the basic logical files. The abandonment of entity computation would make any future comparison difficult.

What is an item (attribute) according to Function Point Analysis? Item found in the conceptual model are utilized by the user to manage his business practices. These items are not redundant, i.e., they respect first normal form rules. When one constructs a logical model, it is necessary to eliminate redundant items and technical items (automatic dating, filler, foreign keys, etc…).

Internal Entities

An entity is **internal** if the application resulting from the evaluated project allows users to create, delete, modify and (or) read entity items. Users must have requested, and have thorough knowledge of, this function for it to be considered.

An entity's complexity is based on the number of items and number of relationships between the entities themselves. All of an entity's items, which are not foreign keys, are computed. The entity relationship model determines the number of relationships within the entity.

The rule of determining complexity is as follows:

	1-19 Items	20-50 Items	51+ Items
1 REL	L	L	M
2-5 REL	L	M	H
6+ REL	M	H	H

For example, this table means that for an entity having 20 items and relationships with two other entities, the entity will have a medium level of complexity.

Here is a sample entity computation based on the conceptual model described above:

Reference	Description	Items	Entities	Complexity
No 1	Phases	1	1	Low
No 2	Projects	6	2	Low
No 3	Values	4	1	Low
No 4	Computes	4	2	Low

External Entities

Entities used but not maintained by the application are external entities. If the application does not maintain the data found in the entity, one speaks of an external entity.

They can have their origin in a shared database, an ordinary file or a reference table copied into the programs.

Item and relationship computation is identical to that for internal items, except that the number of points allotted for complexity is different.

B) The Concept of Transactions

Contrary to «data», which is preserved, transactions represent the dynamic aspect of information processing in an application.

The expression «data flow» is equivalent to transaction in the sense that it describes situations or events to which the application reacts. The data flow consists of the «objects» computed in the data flow diagram. Those which represent data crossing the application's boundary must be computed. This is what is delivered to users.

Transactions are classified as input, output or inquiries. There are two approaches available to evaluate the number of transactions:

- Starting from the process models
- Starting from the screens and reports as they appear to the user.

Both approaches lead to similar results. The first approach (models) is used when the project manager wishes to estimate his project and the development team has not designed the screens and reports. The second approach is preferable to the first if the screens and reports are available because it is more concrete and bears closer relationship to what is actually accomplished.

Input

Input introduces certain changes in the data found in the application. The user generates input. Changes can be input through a direct entry using a screen, or indirectly, using a transaction update file from another application. Data entry can also be accomplished through a batch process. In all cases there is change in the data, whether it be an addition, a modification or a deletion.

The rule for determining complexity is as follows:

	1-4 Items	5-15 Items	16+ Items
1 ENT	L	L	M
2 ENT	L	M	H
3+ ENT	M	H	H

For example, this table means that, for an entry having ten items included in two distinct entities, the entry is of medium-level complexity.

Here is an example of an entry computation using a screen allowing the addition and modification of items in one entity, or the deletion of one item occurrence.

Recording of Projects		
Add:	Modify:	Delete:
Project name:		Person in charge:
Points total:		Factors total:
Start date:		End date:
Overall effort:		

In this example, we must compute three entry screens (logical) since we can add, modify and delete projects within the files. It is by talking with the project manager that one can learn, for example, that the number of items which can be added is seven, that by removing the project name all other information will disappear (delete). With the list of items in each entity, it is possible to determine that the number of entities affected by the seven items and requiring addition of modification is two. As far as the `delete` function goes, it seems initially to concern only one entity, but a more detailed examination shows that another entity's information is also affected if the project is removed. In conclusion, the function also affects two entities.

Here is the entry table for the project entry screen:

Reference	Description	Items	Entities	Complexity
No 1	Add a project	7	2	Medium
No 2	Modify a project	7	2	Medium
No 3	Delete a project	1	2	Low

Output

Output is a result produced by the application. It is produced as a regular part of processing independently of conditions established in the data. A monthly report is produced regardless of what has happened. The user initiates inquiries directly or indirectly.

Different sets of attributes presented with a different technology are not output (e.g.: if a report is produced on paper and on microfilm, it is not counted as two output operations).

All data items that are produced are computed, including any derived data items such as totals and each level of recapitulation. All foreign keys, identifiers and regular attributes are computed.

The rule for determining complexity is as follows:

	1-5 Items	6-19 Items	20+ Items
1 Items	L	L	M
2-3 Items	L	M	H
4+ Items	M	H	H

For example, this table means that, for an entry having ten items included in two distinct entities, the output is of medium-level complexity.

Here is an example of output computation using a periodical report:

Project Value				
Project	Person in charge	Unadjusted points	Factors	Adjusted points
No 012	Guy	250	15	200
No 025	Fred	180	35	180
No 052	Linda	300	40	315
No 098	Nancy	100	10	75
Points total:		830		770

The project manager indicates that this report is produced for managers on a monthly basis. In this example, there are six items to consider: project, person in charge of the project, unadjusted points, factors, adjusted points, points total. The information processed by the application is also included in the evaluation of output. Two entities contribute to this report.

Reference	Description	Items	Entities	Complexity
No 1	Monthly report	7	2	Medium

Inquiries

An inquiry is an output produced in response to a request (input). This transaction is characterized by the fact that it does not modify entities and it is not the result of a modification. It can presented using a screen or a report, whatever the technique. The inquiry involves a direct response to a request, but not necessarily an immediate response.

The rule for determining complexity is dual. For the data entry part, the rule is as follows:

	1-4 Items	5-15 Items	16+ Items
1 ENT	L	L	M
2 ENT	L	M	H
3+ ENT	M	H	H

For the data output part, the rule is as follows:

	1-5 Items	6-19 Items	20+ Items
1 ENT	L	L	M
2-3 ENT	L	M	H
4+ ENT	M	H	H

The final computation is based on the selection of the most complex part of the inquiry (input or output). In practice, output side is most always more complex than input side.

Here is an example of an inquiry screen based on the input presented above:

Recording of Projects		
Add:	Modify:	Delete:
Project name:		Person in charge:
Points total:		Factors total:
Start date:		End date:
Overall effort:		

The project manager indicates that the user has requested the possibility of being able to consult the entries made in respect of the project prior to their modification. This user requirement implies that he be allowed to add an entry which does not modify the files (project name) and to obtain, in return, information from various files (the content of the information on items appearing on the screen for a specific project).

Here is how this requests appears:

Reference	Description	Items	Entities	Items	Entities	Complexity
No 1	Project inquiry	1	1	7	2	Medium

There is, in this request, one entry item linked to an entity (project) and seven output items linked to two entities. The determination of complexity is based on the most complex. Since, the most complex aspect here is output, we have a medium level of complexity.

2.2 Selections and Measures for this Project

This section deals with the characteristics of the sample, the selection criteria of choice reserved for the composition of this sample, the parameters of the measure, the approach of the measure and the resulting data.

2.2.1 Characteristics and Criteria for the Selection of Projects

The projects selected have the following points in common:

- They are business projects (e.g., client file, accounting file, inventory file)
- These were developed in the 80's
- The size of most of the projects is small, or average, i.e., these are projects the duration of which varies between 3 months and 2 years in 90% of the cases.

The selection criteria refer to conditions of acceptance of projects in the sample. The selection criteria retained consist of availability of information and the nature of projects. The criteria of availability of the information regarding projects were selected in order to allow greater reliability of the data and also the consistency in regards to information received for each project. These criteria are:

- The project has been terminated recently (2 years must not have elapsed from the moment of measure and the end of the project). A project that has been completed for quite a while would render any comparison doubtful. The organizational context could have changed in 2 years. Also, the information would probably be difficult to find, and the people in charge no longer available
- It must be possible to reconstruct the original (logical) model of the data, if this is non existent. The basic measure being this model, we obviously have to reconstruct
- The hours attributed to the project must be exhaustive and reliable. It is a fundamental variable of the study
- We must be able to answer most of the additional questions related to the project. There are questions corresponding to the scope and also a few questions regarding the experience of the personnel, the duration of the project and the setting for programming. These factors are explained in the measure parameters section (2.2.3).

The criteria on the nature of the projects were selected in order to ensure the consistency of the sample. We have to make sure that we study comparable projects. These criteria are:

- These projects must be administrative in nature. One of the Kemerer's comments in regards to Putnam and Boehm is that they measure projects different in kind. Scientific, military and administrative types of projects do not have the same demands. Scientific-type projects generally have a larger number of internal treatments and less transactions (input, output, inquiries). The military applications demand more rigorous control that are not always calculated with the Function Point technique. To meet these specifics requirements, Capers Johns has developed a technique called Feature Points (**20**)
- The projects should enter in the category of developments or important improvements (more than three persons-month) to an existing system. Since the objective of this work is to

demonstrate that it is possible to use the Function Point for projects evolving, obviously we have to be selective about these projects. Improved projects for a period of more than three months are considered development projects in most of the companies that were researched

- The projects were achieved in only one delivery (normally this excludes projects scoring higher than 2000 points). The delivery concept generally refers to a more restricted team of workers, which allows for a selection of projects with comparable scope.

2.2.2 Parameters of the Measurement

Here is the list of the variables measured for all of the projects:

Function Point (non-adjusted measurement)

The number of points obtained by adding the points from the transactions and from the entities according to the algorithm proposed by Albrecht (section 2.1.2).

Person-hours of work

The total number of persons-hours for the project. The term "effort" is used as well. This does not usually include the user's time. It spans from preliminary analysis to the implementation. It does not include the support group's time, unless support is directly provided to the project.

Programming environment

Three categories are taken into account in this study. These categories essentially correspond to:

- The use of more traditional languages in a mainframe environment (e.g.: COBOL language with an IMS-type database or the equivalent) (category 1)
- The use of higher level languages like ADSO, CAPEX, etc. (but not for the first time) (category 2)
- The use of fourth-generation languages (category 3). Two projects developed on a PC with a dBase III-type database were also included in this category.

Project staff's experience

This pertains to the team's average experience in years. For example, if the team was made up of 3 people who had 3, 4 and 5 years' experience respectively, the average used was 4. The objective was to obtain an approximation of the team's global experience in the techniques it worked with. The maximum experience retained was 5.

Project manager's experience

This is the number of years of experience as a project manager. The maximum experience retained was 5.

Scope factors

The factors used were Allan Albrecht's from 1984. They are multiple choice questions (6 choices per question varying from 0 to 5) with a weight equal to 0% if the answer is 0 and 5% of the total if the answer is 5 (see Appendix and 2.1.2).

It should be noted that the programming environment, the team's experience and the project manager's experience are environmental factors as shown in Charles Symons' calculation technique (section 1.2.2). In fact, these variables are related to the technical environment and not to the user's request.

2.2.3 Measurement Approach

In our approach we have to take into account the qualifications of the people who achieved the measurement and how the measurement session was developed.

Staff

The components (input, output, inquiries, internal entities and external entities) of all of these projects were all measured according to the same standards by a restricted number of people. Nearly half of these projects were measured or supervised by only one person, the author of this document. This same person carried out some verification on the other projects (who measured the project, is the result coherent considering the standards used, the quality of the time recording, etc.).

These verifications led to lengthier investigations on some projects and the rejection of 2 projects on the basis of the criteria mentioned previously. The other people who carried out the measurements all received the same training.

Development

At the time of the project measurement, the project manager (or, at the very least, someone who was very familiar with the project) was present. A measurement session was developed in approximately the following way:

- Counting points based on the logical model of the data (or rebuilding of this model based on the physical databases.)
- Registration of the entities (internal and external) (section 2.1.2)
- Inventory of the transactions (section 2.1.2)
- Registration and counting of the input, output and inquiries (section 2.1.2)
- Calculation of adjusted points according to the algorithm proposed by Albrecht (section 2.1.2)
- Answer to 14 questions on the scope factors (Appendix)
- Use of the multiplier for calculating the adjusted points (section 2.1.2)
- Listing of person-hours calculated in the project (section 2.2.2)
- Answers regarding other factors (section 2.2.2).

2.2.4 The Resulting Database

This section presents the database's main features according to the parameters in section 2.2.2. The main features are described and then presented in a table.

Year

The resulting database comprises 82 projects, which were completed between 1983 and 1988. Most of the projects (66) were completed in 1985, 1986 and 1987. Therefore 80% of the projects measured ended in the last 3 years. The year 1988 is excluded since, at the time of the measurements, only 3 months had gone by. We can conclude that this database is characteristic of currently developed projects.

Year

No:	From:	To:	Count:	Percent:
1	83	84	3	3.659
2	84	85	6	7.317
3	85	86	22	26.829
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
5	87	88	16	19.512
6	88	89	5	6.098

- Mode

Length

The average length of the projects was 11.8 months. The shortest was 1 month and the longest is 38 months Two-thirds of the projects are less than one year long and 92% of the projects are less than two years long. This is an indication of the average scope of the projects.

Length

No:	From:	To:	Count:	Percent:
1	0	6	15	18.293
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
3	12	18	25	30.488
4	18	24	5	6.098
5	24	30	4	4.878
6	30	36	1	1.22
7	36	42	3	3.659

- Mode

Efforts

The average effort is 5,527 person-hours. However it should be noted that one project of 44,520 person-hours increases this average (the following project has 23,940 person-hours). The median is 3,738 person-hours. It provides a more accurate idea of the efforts for most of the projects. In fact, 54 projects (66%) have less than 5,000 person-hours. The minimum effort recorded is 546 person-hours.

Hours

No:	From:	To:	Count:	Percent:
1	500	1500	11	13.415
2	1500	2500	12	14.634
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
4	3500	4500	14	17.073
5	4500	5500	4	4.878
6	5500	6500	6	7.317
7	6500	7500	3	3.659
8	7500	8500	3	3.659
9	8500	9500	3	3.659
10	9500	10500	1	1.22

- Mode

Unadjusted function points

The unadjusted function points average is 316. The size of the smallest projects is 73 points[12]. The largest is 1,250. The median, which has 274 points, is not too far from the average. Nearly three-quarters of the projects (60) have between 100 and 400 unadjusted function points. The sampling is thus relatively homogenous in size.

Unadjusted points

No:	From:	To:	Count:	Percent:
1	0	100	3	4
2	100	200	22	27
3	200	300	22	27
4	300	400	16	20
5	400	500	8	10
6	500	600	5	6
7	600	700	2	2
8	700	800	2	2

[12] This project was accepted, although it was below 100 points, since it constituted a whole with its entities and its transactions.

Environment Type

The sampling comprises of 47 projects from a more traditional environment (COBOL programming with IMS or IDMS type databases), 25 projects that used a combination of traditional techniques with screen and report generators, 8 projects that used fourth generation languages and 2 projects that used a relational database on computer. With 57% of the projects in one type of environment, 30% in another and 13% in still another, the sampling cannot be called homogenous in terms of environment. This must be taken into account in the analyses.

Environment Type

No:	From:	To:	Count:	Percent:	
					- Mode
2	1	2	25	30	
3	2	3	10	12	

Team Experience

The average experience of the team is 2.55 years in 60 projects. This information was not available for 22 projects. Since team experience is related to techniques and tools used, we cannot conclude that experience is low. This information was the hardest to obtain, probably due to the constant technological changes.

Team's Experience

No:	From:	To:	Count:	Percent:	
1	0	1	7	8.75	
2	1	2	20	25	
3	2	3	19	23.75	
4	3	4	13	16.25	
					- Mode

Project Manager's Experience

The average experience of the project managers in project management is 2.7 years for 79 projects. This information was not available for 3 projects. The standard deviation was 1.28, which indicates that the variation is not too large.

Project Manager's Experience

No:	From:	To:	Count:	Percent:	
1	0	1	5	6.329	
2	1	2	18	22.785	
3	2	3	9	11.392	
4	3	4	19	24.051	
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	- Mode

Scope Factors

The scope factors are the total of the answers between 0 and 5 attributed to each of the 14 questions. The possible minimum is 0 and the maximum is 70, meaning that the theoretical average is 35. These questions reflect the complexity of the software projects with regard to the entire industry. 64 projects (78%) have a total of 35 or less, showing that the complexity of most is lower than the theoretical average. These answers confirm that the sampling is made up of small and medium-sized projects. Note that the actual average is close to 28, as is the median. The mode is 34 (16 projects have between 30 and 35).

Scope Factors

No:	From:	To:	Count:	Percent:	
1	0	5	1	1.2	
2	5	10	4	4.9	
3	10	15	8	9.8	
4	15	20	8	9.8	
5	20	25	12	14.6	
6	25	30	15	18.3	
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	- Mode
8	35	40	11	13.4	
9	40	45	4	4.9	
10	45	50	1	1.2	

2.3 Statistical Analyses

The section explains the underpinnings and what they consisted of.

2.3.1 Why the Statistical Analyses?

The main subject of this thesis is to show that it is possible to use the Function Point technique to make estimations of the effort required to complete development projects.

Section one of this paper shows that it is possible to measure the number of function points relatively accurately at the beginning of the functional requirements phase using data and process models. This meets an initial condition for estimating, that is, the possibility of measuring early in the project.

The second condition for estimating consists in showing the value of the estimating technique. Can we convert the points calculated into effort (person-hours)? Is the value relatively stable, allowing for a reliable estimate? These are questions the statistical analyses must answer.

2.3.2 Statistical Analyses

Statistical analyses were carried out to find cost models relating the efforts and size of the projects. The other variables measured served to explain the variations in the costs of the projects.

The statistical analyses deal with the variables described previously or with calculations derived from these variables. For example, the adjusted points are the multiplication of the unadjusted points by the scope factors[13].

The main statistical analyses dealt with the relationships between the size (unadjusted function points and adjusted function points) of the projects and the efforts (same y variable) while progressively adding other variables such as: environment, experience, length, the project's final year and the relationship between the percentage of the transactions versus the percentage of data. To that end, a good number of single regressions (linear, quadratic and cubic) and multiple regression were affected. The value of the statistical parameters was analyzed for each (R^2, R^2 adjusted, F, t, etc.) and the model used or rejected in the conclusions in Section Three.

So as not to make the reading too heavy, only the main diagrams used were included in the text. Nevertheless, the results for all of the analyses are shown on a table in Appendix B.

[13] The exact formula is Adjusted points = Unadjusted function points X (.65 +E factors/100).

Section Three: Results of the Statistical Analysis

3.1 Search for a General Simplified Cost Model

First, we tried to determine a very simple relationship between the size of the project, as measured by the number of function points and its cost, measured in persons-hours of efforts required for developing the software. The first regression analysis provided the following results.

Person-Hours (PH) = Function points (FP) - 220 ($R^2 = .498$, n=81)

Figure 3.1

This model with satisfactory statistical parameters ("scattergram" given on figure 3.1) only enables the explanation of less than 50% of the variations observed in the 81 projects studied.

Next, we looked at whether there was existed non-linear relationship instead between these two parameters. These were the results:

$PH = 15.4 + .002(FP)^2 + 99.8$ ($r^2 = .498$, n =81)
$PH = -9.3PF + 774.4\ (FP)^2 +.000008\ (FP)^3 = 5676$ ($R^2 = .506$, n =81)
$PH = 641.7\ (FP)^{5} - 5731$ ($R^2 = .478$,n =81)

In conclusion to this first step, we observe that a simple model, with only one variable, explains less than 50% of the variation observed, whatever the form of the relationship. Therefore, this type of model is not very useful for estimating. Other research is required.

For the study to be exhaustive we carried out the same series of analyses using the measurement in "adjusted points "instead of " "non-adjusted points". The models obtained are only slightly higher: they supply an explanation percentage that is 4% higher than the R^2 level. This slight difference is not enough to justify the use a more subjective measurement for which we have already discussed the limitations in section 2.1.2.

3.2 Influence of Some Experience Factors

Two factors were capable of being measured and analyzed: the project manager's experience and the team's experience. This influence of the project manager was measured in years.

The model does not have any significant advantage over the preceding models. On the one hand, its multiple correlation coefficient is not higher and, on the other hand, the coefficient value of the variable "Project Manager's Experience" is not statistically significant. Note that if this variable were significant, it would show that the more experience the project manager had, the more the project should cost less. The non-significance of this does not mean that the project manager's experience does not affect the productivity. It means that other variables have the opposite effect. For example, we know that businesses have the tendency to assign the most difficult projects to the project managers with the most experience.

The team's experience is harder to measure once the team is broken up. The value was obtainable for 58 others. .e. 2.5 years. The model obtained is:

PH = 17.1 FP + 61.1 Team's Experience - 339 (R2 = .498,n =81)

Again, the coefficient of the second explanatory variable (team's experience) is not statistically significant. The first model obtained is still preferable.

3.3 Influence of the Technical Environment

There were good reasons to believe that the technical environment of the software development projects, as defined in section 2.2.4, was an important factor of productivity. Comments from the software specialists in this survey were the basis for distinguishing these environments. This variable should be handled carefully however since it is of the scalar type. Nevertheless, figure 3.2 shows a clear influence of the phenomenon compared to the other variables analyzed.

The best way to measure this influence was to separate the 81 projects according to the technical environments in which they were developed. There were three groups:

- Standard traditional environment: programming with traditional languages (e.g.: COBOL) with hierarchical-type databases (46 projects)
- Traditional environment with screen and report generators (25 projects)
- Environment with fourth generation languages (Ex: Oracle) or use of micro-computers with "pseudo-relational" type databases.

By separating these groups, it was observed that the cost-estimating model for each environment was improved in two instances.

3.3.1 Traditional Environment

The following cost model was obtained for this environment:

PH = 20.9 FP - .439 (R^2 = .600,n =46)

This model provided a better explanation than the general model (.60 compared to .49) but it was still insufficient for estimating. Other variables had to be explored. The information gathered for this thesis (project manager's experience and team's experience) only afforded a slight increase of the significant influence.

3.3.2 Improved Traditional Environment

The following cost model was obtained for this environment:

PH = 20.2 FP - 989 (R2 = .810,n = 25)

Figure 3.3

This model (scattergram shown in Figure 3.3) explains more than 80% of the variation observed. Its statistical parameters are significant. The curvilinear models analyzed did not prove to be more satisfactory than the simple linear model above.

3.3.3 4GL or Microcomputer Environment

The following cost model was obtained for this environment:

PH = 5.2 PF -132 (R^2 = .483, n=10)

This model does not provide a high level of explanation. Also, taking into account the small size of this sampling, the model is not statistically significant.

Conclusion

This study allows us to draw conclusions on the historical basis of the Function Point technique, the measurement rules and the use of the technique as an estimating instrument and a measurement of productivity.

The Function Point technique, although relatively new, has a historical basis that rests on the needs of the software managers who must meet the needs of the information system users.

It also is coherent with the evolution of software by enabling a measurement that is relatively independent of the technology and thus not greatly variable with the software field. The rules for using the Function Point technique may be sufficiently precise if the technique is based on the structured approach instead of on the databases.

In the framework of this study and with this technique, we measured 82 projects developed at approximately 10 different organizations between 1983 and 1988. The task of measuring the 82 projects studied consisted, among others, in rebuilding the logical model of the data from the physical databases. This approach allows for comparing the projects regardless of the type of database used. The level of precision is required by the manager and not the analyst.

This technique enables the project manager to make estimations after preliminary analysis if he or she takes into account the technical environment. The statistical validation of the estimating models developed in the framework of this study has highlighted the need to separate the needs of the user (Function Point technique) from the responsibilities of the software division (environmental factors).

In short, it is possible to measure the productivity of the software development projects with this technique. The measurement's independence of the software techniques enables us to establish what promotes better productivity if the sampling is sufficiently large and the environmental variables are well-identified. The research work pursued by the author, once the academic work was completed (more than 200 projects measured), indicates that it is also possible to identify other variables with a measurable influence on the productivity of the software.

Appendix A

General System Characteristics

The fourteen questions in relation to Albrecht's scope factors are destined to modify calculated (up or down) according to additional essential requirements and constraints imposed by the user. These factors are also related to requests of the user.

The following is a list and a brief explanation of the fourteen scope questions.

1. Data communications

The data and control information used in the application are sent or received over communication facilities. Terminals connected locally to the control unit are considered to use communication facilities.

Score As

0	Application is pure batch processing or a standalone PC.
1-2	Application is batch but has remote data entry or remote printing.
3-5	Online data collection or TP (teleprocessing) front end
3	Application is batch but has remote data entry and remote printing.
5	Application is more than a front-end, and supports more than one type of TP communications protocol.

2. Distributed Data Processing

Distributed data or processing functions are a characteristic of the application within the application boundary.

Score As

0	Application does not aid the transfer of data or processing function between components of the system.
1	Application prepares data for end user processing on another component of the system such as PC spreadsheets and PC DBMS.
2-4	Data is prepared for transfer, then is transferred and processed on another component of the system (not for end-user processing).
5	Processing functions are dynamically performed on the most appropriate component of the system.

3. Performance

Application performance objectives, stated or approved by the user, in either response or throughput, influence (or will influence) the design, development, installation, and support of the application.

Score As

0-3	No special performance requirements were stated by the user. Performance and design requirements were standards
4	In addition, stated user performance requirements are stringent enough to require performance analysis tasks in the design phase.
5	In addition, performance analysis tools were used in the design, development, and/or implementation phases to meet the stated user performance requirements.

4. Heavily Used Configuration

A heavily used operational configuration, requiring special design considerations, is a characteristic of the application. (For example, the user wants to run the application on existing or committed equipment that will be heavily used).

Score As

0-3	Operational restrictions do exist, but are less restrictive than a typical application. No special effort is needed to meet the restrictions.
4	Stated operation restrictions require special constraints on the application in the central processor or a dedicated processor.
5	In addition, there are special constraints on the application in the distributed components of the system.

5. Transaction Rate

The transaction rate is high and it influenced the design, development, installation, and support of the application.

Score As

0-3	No peak transaction period is anticipated.
4	High transaction rate(s) stated by the user in the application requirements or service level agreements are high enough to require performance analysis tasks in the design phase.
5	High transaction rate(s) stated by the user in the application requirements or service level agreements are high enough to require performance analysis tasks and, in addition, require the use of performance analysis tools in the design, development, and/or installation phases.

6. Online Data Entry

Online data entry and control functions are provided in the application.

Score As

0-2	0%-15 %	of transactions are interactive data entry.
3-4	15%-30%	of transactions are interactive data entry.
5	30%-50%	of transactions are interactive data entry.

7. End-User Efficiency

The online functions provided emphasize a design for end-user efficiency.

Score As

0-3	None of the above.
4	Stated requirements for end-user efficiency are strong enough to require design tasks for human factors to be included
5	Stated requirements for end-user efficiency are strong enough to require use of special tools and processes to demonstrate that the objectives have been achieved.

8. Online Update

The application provides online update for the internal logical files.

Score As

0	None.
1-2	Online update of one to three control files is included. Volume of updating is low and recovery is easy.
3	Online update of major internal logical files is included.
4	In addition, protection against data lost is essential and has been specially designed and programmed in the system.
5	In addition, high volumes bring cost considerations into the recovery process.

9. Complex Processing

Complex processing is a characteristic of the application. The following components are present:

- Sensitive control and/or application specific security processing
- Extensive logical and mathematical processing
- Much exception processing resulting in incomplete transactions that must be processed again.

Whish of the following characteristics concern the application:

- Extensive logical and mathematical processing
- Much exception processing resulting in incomplete transactions that must be processed again
- Sensitive control and/or application specific security processing.

Score As

0	None of the above.
1-3	Any one of the above.
4	Any two of the above.
5	All of the above.

10. Reusability

The application and the code in the application have been specifically designed, developed, and supported to be usable in other applications.

Score As

0-1	Reusable code is used within the application.
2-3	The application considered more than one user's needs.
4-5	The application was specifically packaged and/or documented to ease reuse, and the application is customized by the user at source code level.

11. Installation Ease

Conversion and installation ease are characteristics of the application. A conversion and installation plan and/or conversion tools were provided and tested during the system test phase.

Score As

0-1	No special considerations were stated by the user, and no special setup is required for installation.
2-3	Conversion and installation requirements were stated by the user, and conversion and installation guides were provided and tested. The impact of conversion on the project is not considered to be important.
4-5	In addition to 3 above, automated conversion and installation tools were provided and tested.

12. Operational Ease

Operational ease is characteristic of the application. Effective start-up, back-up, and recovery procedures were provided and tested during the system test phase. The application minimizes the need for manual activities, such as tape mounts, paper handling, and direct on-location manual intervention.

Score As

0	No special operational considerations other than the normal back-up procedures were stated by the user.
1-2	Effective start-up, back-up, and recovery processes were provided, but operator intervention is required.
3-4	The application minimizes the need for tape mounts and for paper handling.
5	The application is designed for unattended operation

13. Multiple Sites

The application has been specifically designed, developed, and supported to be installed at multiple sites for multiple organizations.

Score As

0	User requirements do not require considering the needs of more than one user/installation site.
1-3	Needs of multiple sites were considered in the design, and the application is designed to operate only under identical hardware and software environments.
4-5	Documentation and support plan are provided and tested to support the application at multiple sites.

14. Facilitate Change

The application has been specifically designed, developed, and supported to facilitate change. Here are some example:

- Flexible query and report facility
- Business control data is kept in tables that are maintained by the user with online interactive processes.

Score As

0	None of the above.
1-3	Any one of the above.
4-5	In addition business control data is kept in tables that are maintained by the user with online interactive processes.

Appendix B

List of Analysis

Section 2.2.4 for explain about colomns

Regressions Results	Nb.	R^2	F-test	Interc.	T- value	Var 1	T- value	Var 2	T- value	Var 3	T- value
Hours vs Adj. Points	81	0.54	95	-31	-0.05	17.6	9.7				
H. vs AP square	81	0.55	47	522	0.54	14	2.7	0.004	0.745		
H. vs AP cubic	81	0.56	32	-1444	-0.92	33.4	2.5	-0.04	-1.4	0.000029	1.57
H. vs AP, Manager Exp.	78	0.54	44	48	0.05	17.7	9.4	-51	-0.2		
H. vs AP, Staf Exp.	60	0.54	33	653	0.49	17.3	7.7	-290	-0.8		
H. vs AP, Envir.	81	0.63	67	2760	3.2	17.9	11	-1855	-4.3		
H. vs AP, Duration	81	0.6	59	-707	-1.14	207	3.3	11.6	4.7		
H. vs AP, Transact. & Data	81	0.55	48	1219	1.01	17.8	9.8	-2274	-1.2		
H. vs AP (envir. 1)	47	0.78	163	-1919	-2.3	27.4	12.8				
H. vs AP (envir. 2)	25	0.85	128	-371	-0.64	19.6	11.32				
H. vs AP (envir. 3)	10	0.64	14	-282	-0.46	6	2.5				
H. vs AP (envir. 2 & 3)	35	0.51	34	44	0.05	14	5.9				
Hours vs Unadj. Points	81	0.5	78	-220	-0.32	17.3	8.85				
H. vs UP, square	81	0.45	65	100	0.09	15.4	7.3	0.002	0.36		
H. vs UP square, UP cubic	81	0.52	27	-2434	-1.27	38.5	2.5	-0.05	-1.54	0.00003	1.62
H.vs UP, Manager Exp.	78	0.49	37	-200	-0.2	14.4	8.5	-27.5	-0.1		
H. vs UP, Staf Exp.	60	0.49	28	729	0.51	17	7	-371	-1		
H. vs UP, Envir. (type)	81	0.6	58	2616	3.02	18	10.2	-2015	-4.49		
H. vs UP, duration	81	0.57	51	-797	-1.2	235	3.6	10.3	3.87		
H.vs PB, Transact.& Data	81	0.5	40	814	0.63	17.5	8.89	-18.75	-0.95		
H.vs PB (envir.1)	47	0.75	135	-2465	-2.6	28.6	11.6				
H.vs PB (envir.2)	25	0.81	98	-989	-1.39	20.1	9.9				
H. vs PB,(envir.3)	10	0.48	7	-132	-0.17	5.24	2.73				
H.vs PB,(envir.2 & 3)	35	0.47	29	-223	-0.24	13.8	5.4				

Appendix C

Study Data

	Team Exp.	Manager Exp.	Year end	Length	Effort	Transact.	Entities	Points ajust.	Envergure	Points non ajust.	Langage
1	1	4	85	12	5152	253	52	305	34	302	1
2	0	0	86	4	5635	197	124	321	33	315	1
3	4	4	85	1	805	40	60	100	18	83	1
4	0	0	86	5	3829	200	119	319	30	303	1
5	0	0	86	4	2149	140	94	234	24	208	1
6	0	0	86	4	2821	97	89	186	38	192	1
7	2	1	85	9	2569	119	42	161	25	145	2
8	1	2	83	13	3913	186	52	238	25	214	1
9	3	1	85	12	7854	172	88	260	30	247	1
10	3	4	83	4	2422	78	38	116	24	103	1
11	4	1	84	21	4067	167	99	266	24	237	1
12	2	1	84	17	9051	146	112	258	40	271	1
13	1	1	84	3	2282	33	72	105	19	88	1
14	3	4	85	8	4172	162	61	223	32	216	1
15	4	4	85	9	4977	223	121	344	28	320	1
16	3	2	85	8	1617	119	48	167	26	152	2
17	4	3	85	8	3192	57	43	100	43	108	1
18	4	4	86	14	3437	68	316	384	20	326	2
19	3	4	87	14	4494	9	386	395	21	340	2
20	4	2	86	5	840	58	34	92	29	86	1
21	4	4	86	12	14973	318	269	587	34	581	2
22	2	4	85	18	5180	88	170	258	34	255	1
23	2	4	86	5	5775	306	132	438	37	447	1
24	4	1	87	20	10577	304	78	382	39	397	1
25	1	4	86	8	3983	89	200	289	33	283	1
26	4	1	85	14	3164	86	230	316	33	310	1
27	2	0	86	6	3542	71	235	306	37	312	1
28	3	1	85	14	4277	148	324	472	39	491	1
29	4	4	85	16	7252	116	170	286	27	263	1
30	4	1	85	14	3948	175	277	452	37	461	1
31	4	3	86	6	3927	79	128	207	27	190	1
32	1	1	86	9	710	145	38	183	27	168	3
33	4	4	87	9	2429	174	78	252	41	267	3
34	1	1	85	5	6405	194	91	285	35	285	1
35	2	2	88	3	651	126	49	175	38	180	3
36	1	3	86	17	9135	137	119	256	34	253	2
37	2	4	87	11	1435	289	88	377	28	351	3
38	*	*	87	8	5922	260	144	404	24	360	1
39	1	4	88	4	847	158	59	217	18	180	3
40	3	3	88	16	8050	302	145	447	52	523	2
41	1	1	87	9	4620	451	48	499	28	464	1
42	2	4	87	34	2352	661	132	793	23	698	3

43	1	1	88	10	2174	64	54	118	25	106	1
44	*	4	86	39	19894	284	230	514	50	591	1
45	2	1	86	18	6699	182	126	308	35	308	1
46	2	3	87	27	14987	173	332	505	19	424	1
47	2	2	88	9	4004	252	7	259	28	241	1
48	4	3	85	11	12824	131	180	311	51	361	1
49	2	3	85	8	2331	106	39	145	6	103	1
50	3	3	85	9	5817	96	108	204	29	192	1
51	2	3	85	7	2989	116	72	188	18	156	1
52	3	3	85	6	3136	86	49	135	32	131	1
53	2	3	85	17	14434	221	121	342	35	342	1
54	1	1	87	12	2583	61	96	157	18	130	1
55	1	3	86	12	3647	132	89	221	5	155	2
56	3	7	86	13	8232	45	387	432	16	350	2
57	1	1	86	12	3276	55	112	167	12	129	2
58	1	4	87	8	2723	124	52	176	14	139	2
59	3	3	87	5	3472	120	126	246	15	197	2
60	1	2	87	6	1575	47	32	79	14	62	2
61	1	1	86	12	2926	126	107	233	23	205	2
62	3	2	86	6	1876	101	45	146	15	117	2
63	1	1	86	5	2520	78	99	177	14	140	1
64	4	7	86	13	1603	69	74	143	14	113	1
65	1	3	86	8	3626	194	97	291	35	291	2
66	2	*	87	10	6783	224	110	334	28	311	2
67	2	4	87	15	11361	323	184	507	35	507	2
68	1	3	86	10	1267	42	31	73	27	67	2
69	1	2	87	5	2548	74	43	117	25	105	2
70	3	4	87	10	1155	101	57	158	9	117	2
71	0	4	86	6	546	97	42	139	6	99	3
72	2	3	84	13	2275	134	77	211	13	165	2
73	4	5	86	26	9100	482	227	709	26	645	2
74	0	2	84	6	595	213	73	286	6	203	3
75	0	*	84	22	3941	139	143	282	22	245	2
76	2	3	86	24	13860	473	182	655	40	688	2
77	4	4	85	12	1400	229	169	398	39	414	3
78	4	3	83	12	2800	227	73	300	34	297	1
79	4	4	86	24	9520	395	193	588	40	617	1
80	4	3	86	12	5880	469	176	645	43	697	3
81	4	4	85	36	23940	886	241	1127	34	1116	1

Bibliography

1- Albrecht Allan J., "Application Development and Maintenance Measurement and Analysis Guideline" IBM Corporate Information System and Administration, White Plains, NY, 1981.

2- Albrecht Allan J., "Application Development and Maintenance Measurement and Estimate Validation" IBM Corporate Information System and Administration, White Plains, NY, 1984.

3- Boehm W. Barry, "Software Engineering Economics", Prentice Hall, Englewoods Clifts, N.J., 1981.

4- Boehm W. Barry, "Improving Software Productivity", Computer, September 1988, pp. 43-57.

5- Boehm W. Barry, "Understanding and Controlling Software Costs", IEEE Transactions on Software Engineering, Vol. 14 no. 10, October 1988, pp. 1462-1477.

6- Brooks P. Frederick, "No Silver Bullet, Essence and Accidents of Software Engineering", Computer, April 1977, pp. 10-18.

7- Conte S.D., Dunsmore H.E. et Shen V.Y., "Software Engineering Metrics and Models", The Benjamin/Cummings Publishing Company Inc., Menlo Park, California, 1986.

8- De Marco T., "Controlling Software Projects: Management, Measurement & Estimation", New York, Yourdon Press, 1982.

9- Halstead M.H., "Elements of Software Science", Elsevier, North Holland, New York, 1977.

10- Jeffrey D. Ross, "Time-Sensitive Cost Models in the Commercial MIS Environment", IEEE Transactions on Software Engineering, Vol. SE-13, no 7, July 1987, pp. 852-859.

11- Jones Capers, " Programming Productivity", McGraw-Hill, New York, NY, 1986.

12- Jones Capers, "Building a better metric", Computerworld Extra, 20 June 1988, pp. 38-39.

13- Kemerer Chris F., "An Empirical Validation of Software Cost Estimation Models", Communications of the ACM, October 1986.

14- McCabe T.J., "A Complexity Measure", IEEE Transactions on Software Engineering, Vol. SE-2, December 1986, pp. 308-320.

15- Putnam L.H., "A general empirical solution to the macro software sizing and estimating problem", IEEE Transactions on Software Engineering, July 1978, pp. 345-361.

16- Putnam L.H., "The real economics of software development", in The Economics of Information Processing, R. Goldberg and H. Lorin, New York:Wiley, 1982.

17- Symons Charles R., "Function Point Analysis: Difficulties and Improvements", IEEE Transactions on Software Engineering, Vol. 14, no 1, January 1988.

18- Walston C.E. & Felix C.P., "A Method of Programming Measurement & Estimating", IBM Systems Journal, Vol. 16 no. 1, 1977, pp. 54-73.

19- Zwanzig Kenneth C., "Estimating Measurement", Guide 56 Proceedings, 1985.

20- International Function Point Users Group, "A Short History of Function Point and Feature Points", (Capers Jones), Fall Conference Proceedings, October 11-14 1988, Montreal, Canada.