9-1999

The Adequacy of the Fourteen General Systems Characteristics as Function Point Adjustment Factors

Michael D. Prater

Joseph C. Willoughby

AFIT/GCA/LAS/99S-3

THE ADEQUACY OF THE FOURTEEN GENERAL SYSTEMS CHARACTERISTICS AS FUNCTION POINT ADJUSTMENT FACTORS

THESIS

Michael D. Prater
1ST Lieutenant, USAF

Joseph C. Willoughby
Captain, USAF

AFIT/GCA/LAS/99S-3

Approved for public release; distribution unlimited

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

AFIT/GCA/LAS/99S-3

THE ADEQUACY OF THE FOURTEEN GENERAL SYSTEMS CHARACTERISTICS AS FUNCTION POINT ADJUSTMENT FACTORS

THESIS

Presented to the Faculty of the School of Logistics

and Acquisition Management of the Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cost Analysis

Michael D. Prater, B.S.B.A.
1st Lieutenant, USAF

Joseph C. Willoughby, B.B.A., M.B.A.
Captain, USAF

September 1999

Acknowledgements

The successful completion of this thesis effort would not have been possible without the support of several special individuals. First and foremost, I would like to thank my family for their understanding and encouragement. My wife, April, was a constant source of motivation whenever I began to doubt myself. My boys, Josh and Kyle, had a unique way of helping me keep everything in perspective by making me laugh even on the toughest of days.

This entire project would not have been possible without the expertise of our thesis committee. We are indebted to Mr. Daniel Ferens and Major Bryan Turner for their insight and guidance. In addition, we would like to thank several members of the International Function Point Users Group (IFPUG) for sponsoring this study. First, we are grateful for the time and talents of the members of the Academic Affairs Committee. Specifically, Mr. John Van Orden was instrumental to the success of this project. Also, the staff at the IFPUG home office provided outstanding administrative assistance that greatly simplified our lives.

Finally, I would like to thank my classmates in the cost analysis program. In particular, I would like to thank my thesis partner, Captain Joe Willoughby. I learned as much from him as I did from the thesis process itself. His energy, positive attitude, and perseverance were the foundation from which this result blossomed. Joe, thanks for your patience and open-mindedness. Of all the benefits of this thesis, it is this friendship that I am most thankful for.

Michael D. Prater

This thesis could not have been completed without the support and encouragement of many people. I thank my advisor, Mr. Daniel Ferens, for his insight, guidance, and encouragement throughout the entire thesis process. I would also like to express my gratitude to Major Bryan Turner for taking the time out of his very busy schedule to serve as our reader. I am also grateful to Mr. John Van Orden, Chairman of the IFPUG Academic Affairs Committee, and the IFPUG home office staff for their first-class support and patience.

I would like to thank my wife, Mieko, and daughter, Jessica, for their never-ending support and understanding. Over the past 15 months, they both provided the encouragement and motivation to complete the thesis and AFIT program. Finally, I want to express my deepest thanks to my thesis partner, 2Lt Michael Prater, for his superior work ethic and steadfast commitment. The more I worked with him, the more impressed I became with his professionalism and tremendous intellectual capability.

Joseph C. Willoughby

Table of Contents

Page

Page

List of Tables

Table Page

List of Equations

Abstract

The purpose of this research is to assess the perceived adequacy of the 14 general systems characteristics (GSCs) in deriving a value adjustment factor (VAF) for calculating final function point counts. Two self–administered surveys were used to collect the necessary data to address the research goals. Based on the results of these surveys, it is clear that the 14 GSCs, in their current form (including definitions and examples), do not adequately represent the applications complexity required to adjust the function point counts of a software-sizing project.

The current GSCs remain a controversial aspect of function point analysis. While this research can not conclusively state that one factor is more accurate than any other, it is evident that certain GSCs are perceived to be more useful for their intended purpose. In addition, other potential factors may need to be considered as additions to the current GSCs. Although a reevaluation of the current GSCs individually is prudent, their controversial nature appears to be concentrated at more of a macro level. For the most part, the respondents recommendation is to leave the GSCs intact but provide better definitions and more relevant examples.

Notwithstanding these results, there appears to be a strong contingent within the function point community that would prefer that the GSCs be eliminated altogether. Furthermore, there is overwhelming concern with their relevance to today's technological environment.

THE ADEQUACY OF THE FOURTEEN GENERAL SYSTEMS CHARACTERISTICS AS FUNCTION POINT ADJUSTMENT FACTORS

I. Introduction

Overview

The need for software cost estimating has been well documented. Dr. Barry Boehm, in his classic 1981 text *Software Engineering Economics*, noted that the annual cost of software in the U.S. in 1980 was approximately 40 billion dollars, or about 2% of the Gross National Product (GNP) (Boehm, B., 1981: 17). It was expected to grow to nearly 13% of the GNP by 1990 (Boehm, B., 1981: 18). Today, in fact, the Department of Defense (DoD) alone spends nearly $30 billion a year on software (Ferens, 1999a). In addition, Capers Jones notes that software is "the driving force of modern business, government, and military operations" (Jones, 1998: 4). High costs, increasing demand, and an ever-increasing competitive environment for software have fueled this growth. The publicity and speculation surrounding the impending year 2000 problem should erase all doubts about the importance of software to our modern way of life.

The phenomenal growth in the software industry has come at a price, however. In a recent Government Accounting Office (GAO) report, approximately 60% of the software programs investigated experienced cost overruns, while 50% experienced schedule overruns (Ferens, 1999a). In addition, quality issues and complexity requirements have plagued performance objectives. Jones went so far as to say that "the software industry has achieved a notorious reputation as being out of control in terms of

schedule accuracy, cost accuracy, and quality control" (Jones, 1998: xiv). It is clear that cost, schedule, and performance issues are driving the need for improved software cost estimating capabilities. In general, the inherent nature of software, including its size and complexity, has created difficulties in managing the software development process. To say that cost is an important consideration in this process is a monumental understatement.

According to Garmus and Herron, "controlling costs is consistently reported as the number one issue in the software development environment" (Garmus and Herron, 1996: 8). In an era of dwindling budgets and constant downsizing, failure to accurately estimate these costs could have serious consequences. Dr. Boehm describes three potential problems associated with the inability to accurately project software costs. First, software project personnel have no firm basis for assessing how realistic budgets and schedules are. Second, software analysts can not make hardware-software tradeoff analyses for managerial decision making. Finally, managers have no firm basis for determining how much time and effort are needed to effectively manage the overall project (Boehm, B., 1981: 30). The net result could be the misallocation of funds or resources leading to the overruns mentioned above or worst case, the cancellation of the program.

Background

It is clearly evident that the need for accurate software cost estimating remains high. While there are numerous methods and models available to assist the analyst in estimating software costs, the key input in all of these approaches is software size.

There are several popular measures of size, including source lines of code (SLOC), function points, and object points. While SLOC techniques continue to be the most commonly used software sizing measure, function points have evolved as a popular and effective alternative. The need for an alternative resulted from difficulties associated with expressing software size in terms of source lines of code. The first problem is definitional. What exactly is a line of code? There are at least eleven major variations of a line of code (Low and Jeffery, 1990). A second problem with SLOC is its language dependence. As a result, Low and Jeffery concluded, "it is not possible to directly compare the productivity of projects developed in different languages using lines of code as the measure of systems size" (Low and Jeffery, 1990: 64). Furthermore, Jones states that the most common error in software estimation is the usage of SLOC metrics (Jones, 1998: 137). He goes on to say, "the main problem with LOC metrics is the fact that more than half of all software effort is not directly related to source code" (Jones, 1998: 137).

In response to the need for an effective, predictable, and reliable measure of systems size, Allan Albrecht introduced the concept of function points as a basis for sizing software deliverables in 1979 (Jones, 1998: 270). He hypothesized that five program attributes; inputs, outputs, inquiries, interfaces, and logical files; could be used to estimate size (Jones, 1998: 303). The growth in popularity of function points resulted from the extensive research performed by Albrecht and Gaffney for data processing programs. Their analysis on more than 30 data processing programs found function points to not only be a valid predictor of software size, but also to be superior to SLOC as a predictor of software development cost or effort (Ferens, 1999b).

With the function point community evolving, the International Function Point Users Group (IFPUG) was formed. Not only does IFPUG meet twice a year to discuss pertinent function point issues, publish and update standards, and modernize the basic counting rules when necessary, but they also support research endeavors, maintain a certification program, and encourage continuing education. As a result, IFPUG has become one of the largest measurement associations in the world (Jones, 1998).

In general, the traditional methodology supported by IFPUG requires the identification of external inputs, external outputs, external inquiries, internal files, and external interfaces in the program. These function types are then weighted based on several elements including the amount of data and the complexity of the data relationships (Garmus and Herron, 1996: 27). The sum of these weighted elements will give a measure of the "basic" or "unadjusted" function points. Finally, this measure can be adjusted by using a set of fourteen general systems characteristics (GSCs), found in Table 1, to derive a value adjustment factor (VAF) (Garmus and Herron, 1996). Detailed definitions and the IFPUG guidelines for assigning values to each of the general systems characteristics listed in Table 1 can be found in Appendix A.

The VAF is sometimes referred to as the applications complexity. It is intended to place a value on the additional functionality of systems, including such things as user friendliness, transaction rates, performance, and reusability (Garmus and Herron, 1996: 28). Per Garmus and Herron, each characteristic must be evaluated in terms of its degree of influence on a scale from zero to five, with zero being no influence and five representing strong influence (Garmus and Herron, 1996: 81). The resulting degrees of influence for all fourteen GSCs are summed, multiplied by .01, and added to .65 to arrive

at the VAF (Garmus and Herron, 1996: 90). The VAF has a range of .65 to 1.35. The final function point count is then computed by multiplying the unadjusted function point count by the VAF.

Table 1. General Systems Characteristics

1) Data Communication	8) On-line Update
2) Distributed Data Processing	9) Complex Processing
3) Performance	10) Reusability
4) Heavily Used Configuration	11) Installation Ease
5) Transaction Rate	12) Operational Ease
6) On-line Data Entry	13) Multiple Sites
7) End-user Efficiency	14) Facilitate Change

Specific Problem

The purpose of this research is to assess the adequacy of the general systems characteristics in deriving a value adjustment factor for calculating final function point counts. One of the criticisms of the function point methodology is that there are currently over 35 variants of the function point metric (Jones, 1998). Many of these alternatives to the IFPUG standard have been developed to address the needs of the real-time and embedded software communities (Jones, 1998). Not only do some of these variants approach calculating unadjusted function points differently, but there are also wide discrepancies in the measures used to assess the applications complexity. While IFPUG has the fourteen GSCs, Symon's Mark II function points utilize 19 adjustment factors, SPR features 3 adjustment factors, DeMarco uses 22 adjustment factors, and the list goes on (Jones, 1998: 311). In the same manner, the traditional IFPUG complexity adjustment can change the overall function point total by approximately plus or minus 35 percent, where as the SPR method ranges from plus or minus 40 percent, and the Mark II approach could result in a plus or minus 60 percent adjustment (Jones, 1998: 313).

With the validation of function point accuracy in business and data processing applications, the continuing attempts to apply the function point concept to real-time and scientific environments, and the ongoing efforts to educate the software estimating community, function point analysis continues to grow in popularity. However, the variations in complexity adjustment factors remain a controversial aspect of function point analysis. Jones summarizes this troublesome aspect of function point analysis by stating that "the ambiguity and partial subjectivity of the adjustments have led several researchers to assert that the counts might be more accurate if the complexity adjustments were dispensed with and not even used at all" (Jones, 1998:311).

Objectives

The first objective of this research is to determine which methods of adjusting function point counts for applications complexity (VAF) are being used in practice to arrive at a final function point count. Not only does this include the various alternatives to the IFPUG standard, but also, of the fourteen general systems characteristics, which attributes are actually being used. For those users who do not use the IFPUG standard, this research will also examine the reasons for employing alternative methods. A second goal of this study is to evaluate the accuracy and validity of each of the GSCs, as assessed by the users. Finally, this research should provide insight into the applicability of the GSCs across various platforms, languages, and environments.

Research Question

Do the fourteen general systems characteristics adequately represent the applications complexity required to adjust the function point counts of a software-sizing

project? Specifically, are the GSCs the best way to adjust for applications complexity or is there a better alternative to the current IFPUG standard?

Investigative Questions

Three specific questions must be addressed in order to properly assess the adequacy of the general systems characteristics:

1) Are the general systems characteristics commonly used to derive the value adjustment factor? In addition, which specific factors are used to derive the VAF? Also, are alternative methods employed to adjust for applications complexity?
2) What is the perceived accuracy and validity, in terms of representing applications complexity, of each general systems characteristic, as determined by the users? Also, which of the characteristics are considered most important?
3) How applicable are the general systems characteristics across various platforms, languages, and environments?

The answers to these investigative questions should allow conclusions to be made about the adequacy of IFPUG's fourteen general systems characteristics. Since a preponderance of evidence is needed one way or the other, a deficiency in any one investigative question may not be sufficient to reject the adequacy of the general systems characteristics. If the fourteen general systems characteristics are deemed adequate based on this research, the findings should encourage additional standardization within the industry. On the other hand, if the general systems characteristics are determined to be inadequate as currently defined, the findings should provide potential alternatives and areas for future research.

Organization of Research

This first chapter has highlighted the importance of the general systems characteristics in terms of developing an accurate size estimate using function points, and ultimately, in improving the capability to provide accurate software cost estimates. A brief introduction to the problem and the area of study, including a background summary, were provided. Also, the proposed research objectives and investigative questions were set forth. Chapter II will explore the pertinent literature to the subject area. Specifically, the literature review will focus on the alternative methods of adjusting function point counts for applications complexity (VAF) and examine previous research concerning the general systems characteristics. Chapter III will be a detailed discussion of the methodology employed in this investigation. Following the methodology, the analysis and findings will be presented in Chapter IV. Finally, Chapter V will summarize our conclusions and recommendations.

II. Literature Review

Overview

This chapter examines research efforts and literature in the area of software metrics currently used in the software industry as inputs to software cost estimating tools. Specifically, the software metrics that will be compared and contrasted include lines of code and function points. The different function points or direct derivatives of function points that will be examined include the traditional function point developed by Albrecht, Mark II function points, DeMarco function points, Boeing 3D function points, SPR Feature Points, and SEER Function Based Sizing. Finally, it is worth examining various function point metrics and how they benefit organizations involved in software development.

Source Lines of Code

Capers Jones has stated that the use of lines of code as a core metric for productivity and quality studies is a very hazardous practice and could be regarded as professional malpractice (Jones, 1997). Despite the fact that the software industry is aware of its numerous limitations, lines of code remain the most common measure of software size. An individual line of code is a very small piece of the software that delivers an instruction used in solving a solution to a problem. Codes are arranged in lines, and the size of the program is measured by counting the total number of lines of code.

Barry Boehm considers a line of code or "source instruction" to include only program instructions created by project personnel and processed into machine code by

some combination of preprocessors, compilers, and assemblers (Boehm, B.,1981: 59). This line of code definition includes job control language, format statements, and data declarations. A measured line of code does not include comment cards and unmodified utility software.

Electing to use lines of code as the measurement input for software cost estimating is unwise and can lead to disastrous results. Capers Jones identifies three major problems associated with using lines of code. The first and most glaring deficiency is that there has never been a national or international standard for a line of code that encompasses all procedural languages (Jones, 1991: 49). Programming languages can terminate a line of code physically or logically. A physical termination is caused by the ENTER key of a keyboard, which completes the current line and moves the cursor to the next line. A logical termination uses a formal delimiter, such as a semicolon, colon, or period to terminate a line of code. The various definitions of a measured line of code include one, some, or all of the following: executable statements, data definitions, comments, and blank lines. In measuring with LOC, no standard approach exists for measuring reusable code. Wide variances can exist if some individuals or firms count reused code at every occurrence, count it only once, or do not count the reused code at all, since it was not developed for the current project languages. A language such as BASIC allows many logical statements per physical line. Further standardization problems of the LOC metric result when different computer languages are used in a single application program.

Capers Jones points out that the widespread use of program generators, object-oriented languages, and reusable modules in software development make the lines-of –

code metric irrelevant (Jones, 1991). The utilization of the above tools in producing software results in a shrinkage of actual unique line coding, while at the same time it increases the functional content of the application.

The greatest potential problem that exists when using the LOC measurement is the paradox of reduced productivity with the use of higher order languages (Jones, 1991). When programmers utilize a higher order language versus a lower order language, a lower level of effort is required to accomplish the same programming task. This lower effort translates to increased economic productivity and reduced costs. However, utilizing LOC measurements will result in a higher cost per source line and a lower source code per time unit measurement.

Function Points

Function points were developed by Allan Albrecht, an IBM researcher, with the intent of offering an improved sizing metric over LOC measurement. Specifically, the creator of function points wanted a metric that would meet these four goals:

1. It dealt with the external features of the software.
2. It dealt with the features that were important to users.
3. It could be applied early in a product's life cycle.
4. It was independent of source code or language.

The basic premise behind function point analysis was to measure the software functionality from the perspective of the user (Jones, 1991: 44-46). As mentioned in Chapter I, Albrecht believed that the visible external aspects of software that could be counted consisted of five items: outputs, inquiries, inputs, files, and interfaces (Dreger, 1989: 4).

Outputs are items of business information processed by the computer for the end user.

Inquiries may be considered a simple output; more precisely, they are direct inquiries into a data base or master file that look for specific data, use simple keys, require immediate response, and perform no update functions.

Inputs are items of business data sent by the user to the computer for processing and to add, change, or delete something.

Files are data stored for an application, as logically viewed by the user.

Interfaces are data stored elsewhere by another application but used by the one under evaluation.

The original function point methodology and counting rules appeared in 1979. Albrecht made a major revision to the methodology and counting rules in 1984 and these procedures have become the standard adopted by IFPUG. The IBM 1984 revision of function point counting rules renamed the above functions to the following: external inputs, external outputs, internal file types, interface file types, and inquiry types.

Significant ambiguities exist in identifying function points as a result of Albrecht using functional value of the software *as determined by the user* as the identification guideline. Experts in the area of function points have published rules and definitions in an attempt to assist users in defining function points and reducing the ambiguity. In order to identify outputs, Dreger says "count each unique data or control output procedurally generated that leaves the application boundary" (Dreger, 1991:10). To identify inputs, "count each unique user data or control input that enters the application boundary and also updates (adds to, changes, or deletes from) a logical file, data set, table, or

independent data item" (Dreger, 1991:16). Inputs and outputs are considered unique in so far as they have different formats or require different processing logic. Since the function point analysis is independent of any type of programming language, the measurement avoids the distracting and paradoxical details of lines of code.

Calculating function points for a project involves three steps. First, it is necessary to classify and count the five user function types delivered by the development project. Each of the functions that are assigned one of the five categories is further classified as complex, average, or simple. The complexity weights in Table 2 are applied to the initial function point count to arrive at an unadjusted function point total (UFP).

Table 2. Function Point Complexity Weights

Nomenclature	Abbreviation	Simple	Average	Complex
External Input	IT	x3	x4	x6
External Output	OT	x4	x5	x7
Logical Internal File	FT	x7	x10	x15
External Interface File	EI	x5	x7	x10
External Inquiry	QT	x3	x4	x6

The second step is to determine the value adjustment factor (VAF). This consists of scoring the fourteen general systems characteristics that rate the general functionality of the application being counted (see Table 3) and using them in an equation to compute the VAF.

Table 3. General Systems Characteristics (Replicate of Table 1)

1) Data Communication	8) On-line Update
2) Distributed Data Processing	9) Complex Processing
3) Performance	10) Reusability
4) Heavily Used Configuration	11) Installation Ease
5) Transaction Rate	12) Operational Ease
6) On-line Data Entry	13) Multiple Sites
7) End-user Efficiency	14) Facilitate Change

Each characteristic has associated descriptions that help determine the degrees of influence of the characteristics, as found in Appendix A. The degrees of influence range from no influence to strong influence measured on a scale of zero to five. Once all fourteen GSCs have been rated, the values are summed and inserted into Equation 1 as A:

$$VAF = 0.65 + (.01 * A) \qquad (1)$$

where VAF is the value adjustment factor and A is the summation of all fourteen GSCs. The last step is to calculate the final function point count by multiplying the VAF times the unadjusted function point total :

$$FP = UFP * VAF \qquad (2)$$

where FP is the final function point count, UFP is the unadjusted function point total, and VAF is the value adjustment factor.

Concerns with the Value Adjustment Factor

As previously mentioned, fourteen general systems characteristics are used to compute the value adjustment factor. The value adjustment factor is then used to adjust the basic function point count. The general systems characteristics were created with the intent to measure the general aspects of size, as opposed to application specific size. The application of these characteristics optimally should adjust a function point count to better predict the effort. Among users of function points, there is some criticism of the use and practical value of these adjustment factors. Theoretical criticism of the VAF is that its construction involves operations that are inadmissible according to measurement theory. A second criticism is that complexity appears in computing the unadjusted function points and again in the general systems characteristics, therefore some double

counting is taking place. Lastly, there is the criticism that the GSCs are interrelated (Lokan, 1998:1).

A practical criticism of the VAF is that not all of the right things are counted as GSCs and the number of GSCs needs to be increased. Further criticism is that when computing the VAF, it is not appropriate to give all the GSCs the same weight. Still others believe that the VAF does not provide enough variation and that using the VAF does not improve the effort estimation (Lokan, 1998: 1). Dr. Cris Lokan, a noted researcher in the area of software engineering, analyzed the practical application of GSCs to 235 software development projects. His research concluded that some of the GSCs appear to be outdated and not applicable to today's on-line world (Lokan, 1998: 12). He states that "they [GSCs] have less discriminative value now, and should perhaps be redefined". Furthermore, his research concluded that the "VAF was found not to improve the relationship between function points and effort" and that it seems clear that the value adjustment factor should not be used" (Lokan, 1998: 12).

Mark II Function Points

In 1984 Charles Symons was hired by KPMG Management Consulting to advise clients on methods to improve their systems development performance. While performing this work, Symons developed an alternative to Albrecht's approach to function point analysis. Known as Mk II Function Point Analysis (FPA), this new approach to function points supposedly addressed the weaknesses of the traditional function point approach. Symons claims that the Albrecht approach suffers from the following weaknesses (Symons, 1991:18-22):

- It is often difficult to identify the 5 components as specified by Albrecht in an application.
- The weights and levels of complexity assigned to function point components are based on "trial and debate" and lack validity.
- The fourteen general systems characteristics are not comprehensive enough and each of them should not carry the same degrees of influence.
- There is no measure that accounts for internal processing complexity.
- The treatment of systems components as discrete rather than integrated.

Mk II function points are very similar to Albrecht's function points, except that Symons' method is more oriented towards development effort than user value. Symons reduces Albrecht's five categories down to three: inputs, entities, and outputs. For each transaction, unadjusted function points (UFPs) become a function of the number of input data element-types, entity-types referenced, and output data element types. The UFPs for the entire system are then summed. In addition to Albrecht's fourteen GSCs, Symons added five more environmental factors (Symons, 1989: 26):

- Need to interface with other software applications
- Special security considerations in software application
- Need to provide direct access for third parties
- Software with special documentation requirements
- Need for special user training facilities

Instead of general systems characteristics, Symons refers to them as the 19 application characteristics. Later, Symons would add an additional application characteristic that accounted for the need to acquire special hardware to run the software. Table 4 lists all 20 of the application characteristics developed by Symons. Unlike Albrecht's method, which utilizes supplied constants for the degrees of influence, Mk II FPA obtains a

weighted degree of influence value using a calibration process (Symons, 1989: 27). This is reflected in the equation for the technical complexity adjustment factor:

$$TCA = .65 + C * A \qquad (3)$$

where TCA is the technical complexity adjustment factor, C is the constant value obtained from calibrating similar projects, and A is the total degrees of influence. The value calculated for the TCA is then used to calculate the adjusted function point total:

$$FP = UFP * TCA \qquad (4)$$

where FP is the final function point count, UFP is the unadjusted function point total, and TCA is the technical complexity adjustment factor.

Table 4. Symons' Application Characteristics

1) Data Communication	11) Installation Ease
2) Distributed Data Processing	12) Operational Ease
3) Performance	13) Multiple Sites
4) Heavily Used Configuration	14) Facilitate Change
5) Transaction Rate	15) Interface Needs
6) On-line Data Entry	16) Security Considerations
7) End-user Efficiency	17) Third Party Direct Access
8) On-line Update	18) Documentation Requirements
9) Complex Processing	19) Training Facilities
10) Reusability	20) Hardware Requirements

DeMarco Function Points

While Albrecht was the first to introduce the function point concept, Tom DeMarco, a noted software consultant, was also considering an alternative software sizing metric at approximately the same time. In 1982, he published his findings in the book *Controlling Software Projects* (Jones, 1998). While similar conceptually in terms

of defining a functional metric, DeMarco's Bang Theory was a significantly different approach.

The basis for DeMarco's approach was counting primitive components of a model. He explained, "a component of the specification model is considered primitive if it is not partitioned into subordinate components" (DeMarco, 1982: 82). Through the partitioning process, six different types of primitive components emerge: 1) function primitives, 2) data elements, 3) objects, 4) relationships, 5) states, and 6) transitions (DeMarco, 1982).

DeMarco further describes the applicability of this metric to two types of systems: function-strong and data-strong. Function-strong systems are characterized by the operations they perform upon data. Data-strong systems, on the other hand, consider the data acted upon, the data groupings, and the interrelationships, rather than the operations (DeMarco, 1982).

For function-strong systems, the key component is the function primitive. These are the "lowest-level pieces into which the requirement is divided using the function network" (DeMarco, 1982: 82). The count of functional primitives provides a figure similar to Albrecht's unadjusted function points. While Albrecht developed the fourteen general systems characteristics to adjust for applications complexity, DeMarco adopted 16 categories for a similar purpose (see Appendix B for detailed definitions). In addition, he provided initial weighting factors for these attributes. However, since the complexity factors were environmentally dependent, DeMarco encouraged the user to develop appropriate weights for the particular environment being analyzed. His characteristics, along with their respective weighting factors, are listed in Table 5 (DeMarco, 1982). The

sum of the adjusted function primitives results in the *bang* metric for function-strong systems (functional size).

Table 5. DeMarco's Complexity Factors: Function-Strong Systems

Category	Weight	Category	Weight
1) Separation	0.6	9) Synchronization	1.5
2) Amalgamation	0.6	10) Output Generation	1.0
3) Data Direction	0.3	11) Display	1.8
4) Simple Update	0.5	12) Tabular Analysis	1.0
5) Storage Management	1.0	13) Arithmetic	0.7
6) Edit	0.8	14) Initiation	1.0
7) Verification	1.0	15) Computation	2.0
8) Text Manipulation	1.0	16) Device Management	2.5

While the function primitive is the principal component for function-strong systems, object counts are the preferred metric in data-strong systems (DeMarco, 1982). "An object is a single-minded grouping of stored data items, all of which characterize the same entity" (DeMarco, 1982: 82). Each object count is adjusted by a weighted complexity factor based on the relatedness of the object to other objects, as detailed in Table 6 (DeMarco, 1982). The summation of the adjusted object counts results in the *bang* metric for data-strong systems.

Table 6. DeMarco's Complexity Factors: Data-Strong Systems

Relatedness to Other Objects	Weight
1	1.0
2	2.3
3	4.0
4	5.8
5	7.8
6	9.8

3D Function Points

The Boeing Company has developed a derivative of the DeMarco functional metric as discussed above. Known as 3D function points, this new technique was developed to address two classic problems associated with the Albrecht approach. The first problem Boeing sought to address was the perceived difficulty in using function points. The second problem Boeing attempted to solve was the lack of applicability to scientific and real-time systems (Boehm, R., 1997: 8).

The basic premise of 3D function points is that the software application problem can be expressed in three dimensions: data, function, and control. Each dimension is responsible for creating complexity in the application. One dimension may dominate the application; however, all dimensions must be analyzed to obtain an accurate measurement (Garmus and Herron, 1996: 31).

The data dimension in 3D function points is similar to Albrecht's function points and include evaluation of inputs, outputs, inquiries, internal logical files, and external interface files. Data strong problems are typically associated with MIS software environments. The characteristics of the function dimension include the number and the complexity of functions that represent internal processing required to transform input data into output data and the sets of semantic data that govern the process (Garmus and Herron, 1996: 32). Function strong problems are systematic of scientific/engineering environments. The control dimension adds transitions, which enumerate changes in application state. Control strong problems are associated with real-time environments (Garmus and Herron, 1996).

The procedure for counting 3D function points is similar to other function point methodologies. The first step is to identify the measurable characteristics of each dimension that contribute to overall complexity. Next, rules are applied to the counting and assigning of complexity levels to the identified characteristics and a final function point count is computed (Garmus and Herron, 1996). At this time, it is debatable whether 3D function points are any easier to count than traditional function points. It is also debatable whether the technique addresses scientific and real-time applications any better than Jones' feature points, which are discussed below.

SPR Feature Points

In 1986 Capers Jones developed and tested an experimental technique for applying function point methodology known as feature points. Jones and his organization, Software Productivity Research, believed that the standard function point logic that was developed by Albrecht could be modified and adapted to systems software, embedded software, and real-time software. Since Albrecht's function point methodology was developed to solve measurement problems involving management information systems (MIS) applications, Jones believed that the method was not suitable for measuring the following: real-time software, systems software, embedded software, communication software, and scientific applications software (Jones, 1991:81).

Jones hypothesized that applying conventional function point methodology to these types of software applications results in misleading function point counts. These types of software applications are generally high in algorithmic complexity, but low in inputs and outputs. Jones defines an algorithm as "the set of rules which must be completely expressed to solve a significant computational problem" (Jones, 1991: 83).

To solve this problem, Jones created SPR Feature Points, which include a sixth parameter, algorithms, in addition to the five standard function point parameters. This new algorithm parameter can range in weight from one to ten and is assigned a default weight of three. Table 7 outlines the SPR Feature Point method. In addition, the feature point metric reduces the empirical weights for logical data files from an average value of ten down to an average value of seven (Jones,1991: 84). Jones demonstrates that when the number of logical data files and the number of algorithms are the same, both function points and feature points will calculate the identical number of points. However, when there are many more algorithms than files, the feature point methodology will calculate a higher number than the traditional function point methodology.

Table 7. Jones' SPR Feature Point Method

Significant Parameter	Empirical Weight	Total
Number of Algorithms	x 3 =	_____
Number of Inputs	x 4 =	_____
Number of Outputs	x 5 =	_____
Number of Inquiries	x 4 =	_____
Number of Data Files	x 7 =	_____
Number of Interfaces	x 7 =	_____
Unadjusted Total		_____

To calculate the adjusted feature point total using the SPR method, Jones uses a complexity adjustment factor that is the sum of a problem complexity and data complexity weighting scale. The calculated complexity adjustment factor corresponds to a complexity adjustment multiplier as listed in Table 8.

Table 8. SPR Feature Point Complexity Adjustment Values

Sum of Problem and Data Complexity	Adjustment Multiplier
1	0.5
2	0.6
3	0.7
4	0.8
5	0.9
6	1.0
7	1.1
8	1.2
9	1.3
10	1.4
11	1.5

This multiplier times the total unadjusted feature point count (UFP) results in a final adjusted feature point total:

Final Adjusted Feature Points = UFP * Adjustment Multiplier

Jones concedes limitations to the feature point model stating that there is no available taxonomy for classifying algorithms other than purely ad hoc methods based on what the algorithms might be used for (Jones, 1991). The creator of feature points believes that for applications where the number of algorithms is uncertain, or where algorithmic factors are not significant, Albrecht's function points are the appropriate sizing metric. But for applications that have a countable number of algorithms, and where the algorithmic factors are significant, feature points are more suitable.

SEER Function Based Sizing

In 1993, Dan Galorath of SEER Technologies presented yet another derivative of Albrecht's function points. Like Capers Jones, Galorath believed that the conventional function point methodology did not accurately count function points in complex software

that was not input/output oriented. To solve this problem he developed SEER function based sizing (SEER FBS). This methodology is fairly consistent with the standards set forth by IFPUG with one significant difference - the addition of an optional sixth categorical function known as internal functions (Galorath, 1998: 4-9). This enhancement to the SEER FBS model allows users to account for highly algorithmic processes that are common to software applications used in embedded systems. Basically, this function point type is intended to account for functions that manipulate data entirely within an application, or for other reasons never cross the application boundary and thus cannot be categorized as external inputs, external outputs, or external inquiries.

Galorath developed a complexity rating for internal functions that consists of low, average, or high. Table 9 provides internal function examples for each of the complexity ratings.

Table 9. SEER FBS Guidelines for Internal Function Classification

Classification	Examples
Low	Sorting routines.
Average	Reasonably complex functions such as commercial data compression algorithms
High	Signal processing or data reduction algorithms or other functions of high logical complexity

Galorath, 1998: 4-30

Using Function Points

Function point counts are valuable tools for management to assess the effectiveness of their organization. Managers in software organizations have discovered function points provide them with an excellent tool to measure software development

productivity and quality. Listed below are several variations of productivity, cost, and quality metrics utilizing function points that tell an organization how well they are performing (Longstreet, 1999).

Cost per Function Point is the average cost to deliver or maintain a function point. This data can be used to develop a historic database that can be used to estimate future projects.

Function Points per Calendar Month or Year is the average amount of time taken to deliver a function point to production with given staffing levels.

Function Points per Staff Month is the average number of function points delivered for applied month of effort.

Defects per Installed Number of Function Points correlates the quality of the software to the size of the application.

Maintenance Hours per Installed Function Points correlates support effort to application size for currently installed software and legacy systems. Applications with high ratios may be targets for re-engineering or replacement.

It is important to remember to use the above function point metrics as indicators of performance, not exact measures of performance (Longstreet, 1999). They should provide enough granularity to identify problem areas, show general trends, and demonstrate progress.

Another common use of function points is as an estimating technique for the cost benefit analysis that justifies application development. Even for strategic projects that need no quantitative justification, accurate estimation is needed to ensure proper staffing levels. Function points are also used to normalize other productivity and quality measurements. For example, 100 delivered defects on a 100-function point system is a significantly higher error rate than the same 100 delivered defects on a 10,000-function point system.

A recently developed use of function points is for the monitoring of outsourcing agreements. Firms that outsource significant parts of their information systems requirements are concerned that the outsourcing entity delivers the level of support and productivity gains that they promise. Many firms that are awarded the "outsourced" contract are using function points to demonstrate contract compliance.

Summary

The literature review has provided background information on the development and use of function points for software size estimating and the advantages function points have over traditional source lines of code (SLOC) measurements. More specifically, the literature review focused on the widely accepted IFPUG function point analysis and the alternative function point methodologies and how these methods account for system complexity. As discussed earlier, the alternative function point methodologies use different procedures for computing complexity adjustments that do not exclusively rely on IFPUG's fourteen GSCs. Jones states "the variations in complexity adjustments for function point totals have been a troublesome and contentious aspect of function point analysis" (Jones, 1998: 311). The development and use of several alternative methods for calculating systems complexity suggests that further refining of the current IFPUG GSCs maybe needed improve their applicability to specific platforms or applications.

III. Methodology

Introduction

This chapter outlines the procedures needed to collect and analyze the necessary data to answer the research and investigative questions presented in Chapter I. The first section will discuss the rationale for employing the selected methodology. The next section will document the procedures used to develop the instruments required to gather the pertinent data. Finally, the techniques utilized in analyzing the collected data will be presented.

Explanation of Method and Research Design

While a great deal of literature exists on function points, the applicability of function points, and the many variants found in applying function points (as demonstrated in Chapter II), few scientific studies have been conducted on the general systems characteristics. The literature is clear that the GSCs are controversial; however, the specific problems are less defined. As a result, this research first takes an exploratory approach to not only identifying which GSCs are actually used in practice, but also the underlying issues surrounding this controversial subject.

Two of the many reasons to employ an exploratory phase, as presented by Cooper and Emory, support this approach. Specifically, exploration is appropriate where "the area of investigation may be so new or so vague that a researcher needs to do an exploration just to learn something about the problem" (Cooper and Emory, 1995:118). In addition, when "important variables may not be known or thoroughly defined", exploration may be required (Cooper and Emory, 1995:118).

While an exploratory approach helps focus this study, this effort also consists of a descriptive element. As defined by Cooper and Emory, "descriptive studies are those used to describe phenomena associated with a subject population or to estimate proportions of the population that have certain characteristics" (Cooper and Emory, 1995:133). The descriptive nature of this study is the portion that attempts to answer the research questions presented in Chapter I.

Both the exploratory and descriptive phases of this research rely on qualitative techniques to pursue the stated objectives. "The first step in an exploratory study is a search of the secondary literature" (Cooper and Emory, 1995:119). Not only does the literature review presented in Chapter II provide the reader with vital background information in the subject area, but it also helps define the scope of this research and presents leads for potential avenues to explore.

In addition to the literature review, an experience survey was conducted. This approach requires a somewhat informal and flexible interview of individuals with experience in the subject matter. The goal is to seek ideas about important issues or aspects of the subject and discover what is important across the subject's range (Cooper and Emory, 1995:119). Specifically, this research utilized the skill and experience of two individuals. An instructor at the Air Force Institute of Technology who has published numerous papers in the field of software cost estimating and an owner of a software metrics consulting firm specializing in function points were identified as candidates for this experience survey. Both individuals provided insightful information that helped define the research objectives of this study.

The transition from exploration to description was not a purely sequential event. While the surveys employed in the next portion of this study primarily address the descriptive element of the research, the initial survey did contain a certain degree of fact finding which identified issues not previously considered.

In general, Cooper and Emory define a survey as questioning people and recording their responses for analysis (Cooper and Emory, 1995:269). The advantage of a survey as a primary data collection tool is its versatility. However, although surveys are efficient and economical, the quality of the information depends on the ability and willingness of respondents to cooperate (Cooper and Emory, 1995:269).

When is a survey appropriate? "The most appropriate applications are those where respondents are uniquely qualified to provide the desired information" (Cooper and Emory, 1995:270). The target respondents for this research are members or contacts of the International Function Point Users Group, which provide a convenient sampling unit. Although convenience sampling may lack objectivity, it facilitates quick and economical data collection. While the experience and demographics may vary widely, the members of this organization have a unique bond and perspective in their interest of function points. Thus, Cooper and Emory's criterion is satisfied, supporting the use of a survey. The next section will document the survey development process.

Survey Instrument

The first decision prior to actually developing the initial instrument was to determine the optimal method for conducting the survey. Common methods include personal interviews, telephone interviews, mail surveys, electronic mail surveys, or web based surveys. The goals of this data collection effort were to reach as many potential

respondents as possible, while holding down costs and easing the administrative burden. With the lead-time available for survey respondents to reply (approximately 30 days), a mail survey was deemed the most adequate means of accomplishing this task.

Several advantages of a mail survey include low cost, expanded geographic coverage, minimal staff requirements, anonymity, and ample time to respond. However, there are also disadvantages. Three concerns include a low response rate, no interviewer intervention, and often respondents represent the extremes of the population (Cooper and Emory, 1995: 287). While these negative elements are significant, steps were taken to mitigate risks in these areas. For example, the large population ensured a representative sample even with a low response rate. Also, the researchers' telephone numbers and e-mail addresses were included with the survey for respondents to use to clarify any questions with the instrument itself.

Survey Development Process

The exploratory phase described above identified the research questions to pursue and provided a foundation for the survey instrument development process. Once a mail survey was selected as the basic survey method, the instrument design began with the drafting of specific measurement questions.

Of the three types of information typically gathered by surveys, target data, information for classification and analysis, and administrative data, the first draft focused primarily on target data (Cooper and Emory, 1995). Target data consists of the facts, attitudes, preferences, and expectations about the central topic (Cooper and Emory, 1995: 302).

A major decision required in survey development is the amount of structure to be imposed on the responses. A combination of both open and closed responses provided the most flexible alternative in meeting the needs of this research. The closed-response questions primarily consisted of either dichotomous or multiple-choice questions. The dichotomous questions were in a "Yes" or "No" format. With respect to the multiple-choice questions, every effort was made to make the selection choices exhaustive. However, to account for unanticipated response categories, the category "other (please specify)" was included where applicable to provide for other options (Cooper and Emory, 1995).

In addition to the closed questions, several open-ended questions were also included. Not only were the respondents' opinions of interest, but their suggestions for improving the GSCs were also considerably pertinent. The flexibility of the open-ended response was ideal for accomplishing these objectives.

Once the initial phase of question construction was complete, the first draft of the survey instrument consisted of seven questions. The next stage consisted of pre-testing the survey instrument. In an effort to reduce the time required to field the survey, a modified pre-test approach was adopted. Specifically, the survey was sent to a panel of seven experts for review. Six of the panel members were representatives of the IFPUG Academic Affairs Committee. The seventh individual was a professor at the Air Force Institute of Technology. While all seven panel members were considered subject-matter experts, several also had vast research experience. This diverse combination of experts was essential to validating the survey design.

The purpose of this panel was to test the survey instrument for content, clarity, continuity and flow, question sequence, respondent interest, and overall appropriateness in answering the research questions (Cooper and Emory, 1995). The panel was given approximately one week to review the survey and provide feedback. Their comments were then incorporated into the second draft. Also, three additional multiple-choice questions were added to obtain information concerning respondent characteristics needed for classification and analysis. The second version of the survey was re-sent to the panel of experts for evaluation. With no further changes, the final ten-question survey instrument was approved for release. The administrative task of mailing and collecting the surveys was coordinated with the IFPUG home office. The survey was distributed via the U.S. Postal Service to approximately 508 IFPUG members worldwide. A copy of the survey and corresponding cover letter can be found in Appendix C.

In addition to the survey instrument documented above, a follow-up survey was also conducted based on the analysis of the initial results. The purpose of this second round of questions was to delve deeper into those controversial issues that surfaced in the initial survey. The development process for this second survey instrument was similar to that employed in the first data collection effort in terms of the construction of the instrument and the initial pre-testing. However, several key differences should be noted. First, due to the time limitations imposed on this research, the turnaround time for respondents to reply was significantly shorter for the follow-up survey (approximately 1 week). As a result, a regular mail survey was considered less appropriate in this instance. In an effort to provide the maximum response time possible, an electronic-mail survey

was determined to be the most appropriate method to achieve this goal, while also meeting the previously stated affordability objective.

In addition to impacting the survey method, the time constraint also was influential in the analysis of the proposed structure to impose on the responses. While the initial survey was a combination of open and closed questions, the follow-up survey consisted primarily of closed questions with only one open-ended question. Although this does not provide as much flexibility, it eases the administrative burden of consolidating the responses and shortens the time required to perform the necessary analysis. In addition to the dichotomous and multiple-choice questions, two questions required the respondents to rank the selections. Overall, the follow-up survey instrument consisted of a total of ten questions.

Finally, while the administrative task of sending the electronic mail survey was coordinated with IFPUG, the respondents replied directly to the researchers rather than back to IFPUG. Again, the time constraints imposed on this research were a driving factor in deciding the best survey collection approach. A copy of the follow-up survey and instructions can be found in Appendix D.

Analysis Technique

The data analysis performed on the survey results were consistent for both the initial and follow-up surveys. The first step was to edit the raw data. "Editing detects errors and omissions, corrects them when possible, and certifies that minimum data quality standards are achieved" (Cooper and Emory, 1995: 379).

Since the survey instruments contained both open and closed-ended questions, alternative analysis approaches were necessary to account for their inherent differences.

The closed–ended responses were initially coded and entered into a spreadsheet to ease the data manipulation and statistical analysis. The specific statistical techniques employed in the analysis fall under the broad heading of descriptive statistics. The primary means of examining the data consisted of frequency tables, where appropriate. In general, frequency counts were tabulated, the percentages of participants selecting a certain response were calculated, and the mean responses were displayed, where applicable.

In addition to the closed-ended questions, the open-ended responses were analyzed using content analysis. "Content analysis measures the semantic content or the 'what' aspect of a message" (Cooper and Emory, 1995: 385). The process begins by selecting mutually exclusive key words or phrases that reflect the objectives for which the data were collected (Cooper and Emory, 1995). With the applicable categories established, each response is then classified into a particular category. The result enables frequency counts and percentages to be used to summarize the data. Not only are the categories more meaningful and useful, but this type of analysis also allows the comments to be viewed from a somewhat more quantitative perspective.

Summary

This chapter documented the methodology employed in this research effort. Theoretical and practical justification was provided for the use of a survey. In addition, the advantages and limitations of a survey as a data collection technique were also presented. Not only was the overall format of the study discussed, but also the process used to construct the survey instrument was thoroughly defined. Finally, the specific analysis approach to be used in Chapter IV was outlined.

IV. Analysis and Findings

Overview

The purpose of this chapter is to provide the results of the methodology employed to answer each of the investigative questions. It presents the results of the two surveys, summarizes the analysis of the data collected, and provides an interpretation of the results. The survey results will be presented in a logical two-part fashion, with the first part of the chapter focusing on the specific results and analysis of the initial survey, and the second part examining the results and analysis of the follow-up survey. Finally, the analysis and interpretation of the results found in both surveys should answer each of the investigative questions listed in Chapter I, meeting the primary objectives of the research.

Initial Survey Results

As mentioned in Chapter III, the primary purpose of the initial survey was to collect information and measure attitudes regarding the use of GSCs in function point sizing. The initial survey, found in Appendix C, was designed to be self-administered and was sent directly from the IFPUG home office in Westerville, Ohio, to 508 IFPUG members worldwide via regular mail. The member's addresses were obtained from a mailing list routinely used by IFPUG to contact members. The cover letter instructed respondents to return the completed survey to IFPUG via regular mail or fax. The IFPUG home office address and phone number were printed on each page of the survey. IFPUG personnel received and filed all of the returned surveys in a central location. Approximately two weeks after the return date specified in the survey cover letter,

IFPUG forwarded all of the completed surveys to the Air Force Institute of Technology for coding and data analysis. Ninety-four surveys were returned out of a total potential sample size of 508, achieving a response rate of 18.5%. Self-administered mail surveys of this type with a response rate of 30 percent are often considered satisfactory (Cooper and Emory, 1995: 282). This 18.5 percent response rate should be considered low and may limit the generalizability of the results.

Survey Question 1. The first part of this question asked the participants if they use the IFPUG VAF to adjust the final function point count for applications complexity when sizing software applications and projects. As indicated in Table 10, an overwhelming majority of the respondents indicated that they utilize the VAF to adjust their counts.

Table 10. Respondents Using VAF to Adjust Final Function Point Count

	Yes	No	Subtotal	No Reply	Total
Number	77	16	93	1	94
Percentage	82.8%	17.2%	100%		

The second part was an open-ended question that asked respondents who answered "No" to the first part to briefly explain why they do not use the VAF to adjust function point counts. Even though the majority of the respondents indicated that they do use the IFPUG Value adjustment factor, several of these affirmative respondents also provided additional comments although not required. After editing and coding the information received from the open-ended question, the responses were grouped into the five primary categories listed in Table 11. A complete list of the edited responses to this survey question can be found in Appendix E.

Table 11. Reasons for not Using IFPUG Value Adjustment Factor

Reason	Responses
GSCs are arbitrary and outdated	6
Inaccurate-no improvement over unadjusted function points	6
GSCs don't relate to cost/effort	3
Confusing	2
Weighting Factors are arbitrary/inappropriate	2

Survey Question 2. This question asked the participants if they use the fourteen GSCs to compute a VAF as outlined by IFPUG when determining the functional size of software projects. The purpose behind this question was to confirm that those respondents who utilized the IFPUG VAF also incorporated the use of the GSCs. The results listed in Table 12 are consistent with the results measured in survey question 1.

Table 12. Respondents Using GSCs in accordance with IFPUG CPM

	Yes	No	Subtotal	No Reply	Total
Number	76	16	92	2	94
Percentage	82.6%	17.4%	100%		

Survey Question 3. This two-part question was designed to provide some preliminary insight into the perceived accuracy and validity of each of the general systems characteristics. The first half of the question solicited information from respondents by providing a numerical attitude scale to measure the validity and accuracy of the GSCs. Based on their own experience, respondents were asked to measure the accuracy and validity of each GSC based on the 5-point numerical scale in Table 13.

Table 13. Attitude Rating Scale

Accuracy and Validity	Value
None	1
Little	2
Average	3
High	4
Very High	5

Very high accuracy and validity is the most favorable attitude towards the GSC, and a value of 5 is assigned to this response. Conversely, a value of 1 is assigned to the response that indicates the individual GSC has no accuracy and validity. Since the value of 3 is assigned to a GSC that has average accuracy and validity, a mean response of less than 3.00 was used as the threshold criterion to identify GSCs that are perceived to be inaccurate and lacking validity. While the use of the mean provides limited information as a measure of central tendency with the ordinal scale developed in this survey question, it does provide an objective measure for discerning those GSCs that are perceived to be accurate and valid.

Since the mean was strictly used as a method to categorize the results as either accurate/valid or inaccurate/invalid, and not as a true measure of central tendency, it was determined that the limitations of this approach would not materially detract from the research. Using this criterion, six of the GSCs - reusability, online update, heavily used configuration, installation ease, operational ease, and online data -were perceived to be inaccurate or invalid. The rank order results of survey question 3 appear in Table 14.

The second part to question 3 was an open-ended inquiry that asked respondents to list additional systems characteristics besides the standard fourteen GSCs that affect complexity and that they consider important in adjusting final function point counts. After editing and coding the 23 responses to this open-ended question, the information was grouped into the eleven categories listed in Table 15. Most of the respondents indicated that additional factors are needed to capture the complexity of new technologies in today's environment. Areas such as real-time systems, web-based or client server

applications, and programming language issues (multiple languages, inherent functionality, etc.) seem to be the predominant issues for the users.

Table 14. Respondents Perceived Accuracy and Validity of GSCs

Factor	# Responding							% Responding				
	1	2	3	4	5	Total	Mean	1	2	3	4	5
Performance	4	10	22	18	17	71	3.48	5.6	14.1	31.0	25.4	23.9
Distributed Data Processing	5	12	20	25	10	72	3.32	6.9	16.7	27.8	34.7	13.9
Multiple Sites	4	11	24	24	9	72	3.32	5.6	15.3	33.3	33.3	12.5
Data Communications	6	9	25	23	9	72	3.28	8.3	12.5	34.7	31.9	12.5
Complex Processing	5	15	17	25	10	72	3.28	6.9	20.8	23.6	34.7	13.9
Facilitate Change	7	10	24	21	10	72	3.24	9.7	13.9	33.3	29.2	13.9
Transaction Rate	5	13	26	21	7	72	3.17	6.9	18.1	36.1	29.2	9.7
End-user Efficiency	10	12	24	15	11	72	3.07	13.9	16.7	33.3	20.8	15.3
Reusability	**11**	**15**	**22**	**14**	**10**	**72**	**2.96**	**15.3**	**20.8**	**30.6**	**19.4**	**13.9**
Online Update	**9**	**13**	**29**	**15**	**6**	**72**	**2.94**	**12.5**	**18.1**	**40.3**	**20.8**	**8.3**
Heavily Used Configuration	**8**	**22**	**21**	**18**	**3**	**72**	**2.81**	**11.1**	**30.6**	**29.2**	**25.0**	**4.2**
Installation Ease	**8**	**24**	**23**	**14**	**3**	**72**	**2.70**	**11.1**	**33.3**	**31.9**	**19.4**	**4.2**
Operational Ease	**14**	**19**	**22**	**12**	**5**	**72**	**2.65**	**19.4**	**26.4**	**30.6**	**16.7**	**6.9**
Online Data Entry	**20**	**18**	**19**	**7**	**8**	**72**	**2.51**	**27.8**	**25.0**	**26.4**	**9.7**	**11.1**

Table 15. Additional Complexity Factors Considered Important by Respondents

Additional Factor Suggested	Frequency
New technology: standard interfaces, protocols, and web applications	7
System reliability and security	3
Factor for real-time applications/Complex algorithms	2
Management/Development team's capability	2
Development platform/Primary programming language	2
Software designed for multiple user languages versus single language	2
Requirements volatility	1
Purchased application or custom built	1
Size of databases	1
Checkpoint parameters	1
Internal "Help" applications within software	1

Along with these comments, suggestions for improving the accuracy and the validity of the GSCs in terms of additional factors they consider important to the overall applications complexity can be found at the end of this report. Finally, future research should attempt to scientifically validate the accuracy of all the general systems

characteristics. A complete list of the edited responses to this survey question can be found in Appendix E.

Survey Question 4. This question asked the respondents if they felt the GSCs could be applied to all platforms and languages. Table 16 presents the distribution of responses to this question.

Table 16. Applicability of GSCs to Multiple Platforms and Languages

	Yes	No	Subtotal	No Reply	Total
Number	45	26	71	23	94
Percentage	63.4%	36.6%	100%		

As indicated in the table above, the majority of the respondents believe that the GSCs can be applied to all platforms and languages. Those respondents who answered "No" were again given the opportunity to briefly explain why they believe the GSCs are not applicable to all platforms and languages. After editing and coding the 27 responses to the open-ended question, the information could be grouped into the seven categories listed in Table 17. The primary argument against the applicability of the GSCs to all platforms and languages is that they are not relevant to today's environment. This is consistent with the comments mentioned in question 3. The users again addressed areas such as real-time systems, web-based or client server applications, and programming language issues (multiple languages, inherent functionality, etc). A complete list of the edited responses to this survey question can be found in Appendix E.

Survey Question 5. As discussed in Chapter II, there are several alternative methods to calculating function points in addition to the traditional procedure outlined by IFPUG. These alternative methods do not use the value adjustment factor that is used by IFPUG to adjust for complexity; therefore, the use of these alternative methods doesn't

strictly adhere to the use of the fourteen GSCs in accordance with the IFPUG's *Function Point Counting Practices Manual.*

Table 17. Reasons GSCs Cannot be Applied to all Platforms and Languages

Reason	Frequency
Not relevant to today's environment (networks/web applications)	11
Only applicable to mainframe environment	7
Fail to take into account the functionality provide by platforms/languages	3
Do not add any value to the metrics employed by user	2
Not applicable in real time environment	2
Not suitable for software reusability	1
Only applicable in MIS domain	1

Survey questions 5 and 6 only applied to those respondents that did not exclusively use the IFPUG general systems characteristics. All other respondents were to continue on to survey question 7. The purpose of survey question 5 was to provide some insight into the actual usage of alternative function point counting methods. Since the sample population consisted entirely of IFPUG members, the expectation that the respondents used alternatives to the methods outlined in the *Function Point Counting Practices Manual* was not very high. The survey question listed the most widely accepted alternative function point counting methods and asked the respondents to circle any of the methods they use for sizing software projects. The choices provided were British Mark II FPA, SPR Feature Points, DeMarco function points, SEER Function Based Sizing, and an open-ended choice of other, where respondents were given the option to write in a method not listed above. This survey question elicited only fifteen responses. The breakdown of responses is provided in Table 18.

Table 18. Use of Alternative Function Point Counting Methods

Method	Responses
British Mark II FPA	1
SPR Function Points	3
DeMarco Function Points	0
SEER Function Based Sizing	1
Other	10

The 10 "Other" responses are summarized in Table 19. It appears from this "Other" category, as well as from many of the respondents' comments at the end of the survey, that there is a perception by some users in the IFPUG community that adjusted function points are no more accurate than unadjusted function points. As such, they elect to simply forego the applications complexity adjustment and use the unadjusted function point count in their estimates.

Table 19. Other Function Point Counting Methods Used

Method	Responses
No complexity adjustment/use unadjusted function points	4
Customized/own methods	3
Backfiring	2
Smart Predictor Estimating Tool	1

Survey Question 6. Since the previous survey question identified users of alternative function point methodologies, the logical follow-on question should determine why the respondents elected to use a different methodology than the standard as outlined in the *Function Point Counting Practices Manual*. Survey question 6 asked those respondents that answered question 5 what the primary advantages are for using an alternative method of applications complexity adjustment versus the standard fourteen GSCs. The question provided the respondents with four fixed alternatives and a fifth response of "Other" which allowed the respondent to write in an advantage or reason not

listed as a response. Respondents were allowed to circle as many advantages or reasons as they felt applicable. Of the 15 respondents to question 5, 14 replied to question 6 (multiple responses were possible). The overwhelming response selected for using an alternative method was that it more accurately reflects the applications complexity. This is consistent with many of the criticisms found later in the open-ended written comments. Table 20 presents the results of this question.

Table 20. Reasons for Using Alternative Counting Methods

Reason	Responses
Easier to understand	5
Ease of use	3
Less complex	2
More accurately reflects complexity	11
Other	3

Survey Question 7. This survey question sought to identify the major uses of the various function point methodologies. Again the question allowed respondents to choose multiple answers along with an open-ended choice of "Other". Of the 94 total responses to the survey, 88 respondents answered this question, with six no replies. The results are provided in Table 21.

Table 21. Major Uses of Function Point Based Sizing

Use	Responses	Percentage
Size Analysis	79	90.8%
Productivity Analysis	75	86.2%
Cost Analysis	60	69.0%
Schedule Analysis	41	47.1%
Inventory	41	47.1%
Other Metrics	29	33.3%

Survey Question 8. In an effort to gain better insight into how experienced and familiar the respondents were with sizing software projects, respondents were asked how

many software-sizing estimates they had performed the previous two years. Respondents were provided four mutually exclusive intervals that ranged from zero to greater than ten estimates. The 91 responses received to this question are presented in Table 22.

Table 22. Number of Software Estimates Performed Previous Two Years

Number of Estimates	Number of Responses	Percentage
Less than 2	7	7.7%
Between 2 and 5	10	11.0%
Between 6 and 10	16	17.6%
Greater than 10	58	63.7%

While the majority of the respondents were fairly experienced in software-sizing estimates, a disproportionate number of respondents with fewer than 10 estimates in the past two years did not use the IFPUG value adjustment factor to adjust the final function point count. Out of the 93 respondents who replied to question 1, 33 (35.5%) completed less than 10 estimates in the past two years. Although a total of 16 respondents did not use the IFPUG VAF (17.2%), 8 out of the 16 (50%) were from the relatively inexperienced group. Additional research may be necessary to understand how a particular method was chosen, as well as the training received by new analysts.

Survey Question 9. Again, in an attempt to gain further insight into our respondents use and experience with function points, survey question 9 asked the respondents what the average final function point count is from their software-sizing estimates. A total of 91 responses were received to this question and the results are summarized in Table 23.

While most of the responses average between 100 and 5,000 function points, it does not appear that the overall size of the estimate is influential in the perceived accuracy and validity of the GSCs. Responses indicating an average function point count

greater than 1,000 were considered separately to see if their perceptions concerning the GSCs were materially different from the overall sample. Only the mean value for the "Transaction Rate" characteristic fell below the 3.00 threshold. The other factors all had mean values that were not significantly different from the overall results.

Table 23. Average Function Point Count in Sizing Estimates

Average Final Function Point Count	Number of Responses	Percentage
Less than 100	5	5.5%
Between 100 and 1,000	57	62.6%
Between 1,000 and 5,000	25	27.5%
Greater than 5,000	4	4.4%

Survey Question 10. Much of the discussion regarding the applicability of function points and the GSCs centers on the operational environment in which the software is utilized. To gain some insight into this area of discussion, survey question 10 asked respondents to identify the type of operational environment that the software they routinely size is used in. The question provided the respondents with five common operational environments along with a sixth response of "Other", which allowed the respondent to write in the applicable environment not listed as a response. As with previous questions, multiple answers were possible. A total of 92 responses were received to this question, with two no replies. As indicated in Table 24, most of the software sizing estimates using function points remains in the operational environment in which Albrecht first introduced it, for MIS purposes. However, it appears that attempts are being made to expand its applicability to other operational environments, as indicated below.

Table 24. Operational Environment Where Sized Software is Used

Operational Environment	Number of Responses	Percentage
Commercial MIS	69	75.0%
Military MIS	16	17.4%
Real-time Systems	25	27.2%
Scientific Applications	33	35.9%
Telecommunications	11	12.0%
Other	18	19.6%

In addition to the above findings, the survey results were reviewed specifically for the 25 responses to question number 10 that indicated they performed sizing estimates for real-time systems. The findings support many of the criticisms noted in the comments referring to the belief that the GSCs do not reflect the complexity of real-time systems. Specifically, of the fourteen GSCs, only the Performance, Multiple Sites, and Distributed Data Processing characteristics are perceived to be accurate and valid indicators of applications complexity for real-time systems using the mean response of 3.00 or above criterion, as described above in survey question 3.

Additional Comments. At the end of the survey respondents were provided space to make any additional comments regarding the focus area of the survey. As expected, a wide range of responses was received, with the majority of comments critical of the current VAF method and the GSCs. After editing and coding the open-ended responses, the collected information was categorized into eight specific subject areas. The results of this process are listed in Table 25. A complete list of the edited additional comments to this survey can be found in Appendix E.

Table 25. Categories of Additional Comments Provided by Respondents

Reason	Frequency
GSCs are outdated/of little value today	9
VAF needs to be revised to be applicable	8
Current FPA methodology works well for variety of metrics	5
Function points and complexity are separate issues	5
Too much subjectivity in VAF/GSC measurements	4
VAF needs to be removed from FPA methodology	4
GSCs not weighted properly	2
Other	2

Follow-up Survey Results

After analyzing the results of the initial survey, the decision was made to design and distribute a follow-up survey. This follow-up survey would focus on many of the issues and concerns identified in the initial survey by the respondents. In addition, certain questions used in the follow-up survey would attempt to create a profile of function point users. Like the initial survey, the follow-up survey located in Appendix D, was designed to be self-administered. In the interest of saving time, the follow-up survey was electronically forwarded from AFIT to the IFPUG home office in Westerville, Ohio, where it was then disseminated via electronic mail to 508 IFPUG members worldwide using an electronic mailing list.

A brief introductory paragraph preceded the survey and provided detailed instructions for completing the survey in its electronic format. Respondents were asked to complete and return the survey within five working days directly to the researchers at AFIT. Both an e-mail address and fax number were provided with the survey instructions to provide the respondents with alternative means of replying, while minimizing the turnaround and transit times. The research personnel logged and filed all of the returned surveys in a designated survey e-mail folder or a survey binder, depending on whether the survey was returned via e-mail or fax. Approximately two weeks after the

return date specified in the survey instructions, the researchers had received 70 surveys out of a total potential sample size of 508, achieving a response rate of 13.8%.

Survey Question 1. This survey question asked respondents if they strictly adhere to the procedures outlined in the *Function Point Counting Practices Manual* (CPM) when they perform function point analysis. This was a simple closed-ended question that required a "Yes" or "No" answer. A total of 69 responses were received to this question and the results are listed in Table 26.

Table 26. Respondents Adhering to Procedures in CPM

	Yes	No	Subtotal	No Reply	Total
Number	54	15	69	1	70
Percentage	78.3%	21.7%	100%		

IFPUG's CPM is considered the preeminent reference for function point analysis. The results from this question indicate that the majority of respondents believe that they strictly adhere to the manual when performing FPA. Using IFPUG's CPM requires the use of the VAF, along with the associated GSCs, to calculate a final function point total. The majority of respondents answering "Yes" implies there is some value in the use of the VAF and the GSCs.

Survey Question 2. A few of the comments from the initial survey indicated that the CPM did not provide adequate documentation for counting function points. As a follow-on to these comments, respondents were asked if the current CPM adequately documents the procedures necessary to perform function point analysis. Again, this was a simple closed-ended question that required a "Yes" or "No" answer and total of 69 responses were received, the results of which are listed in Table 27.

Table 27. FPA Procedures Adequately Documented in IFPUG CPM

	Yes	No	Subtotal	No Reply	Total
Number	46	23	69	1	0
Percentage	66.6%	33.4%	100%		

Although the majority of respondents believe that the CPM adequately documents function point counting procedures, the number of negative responses to this question may warrant further exploration into this issue.

Survey Question 3. The first two questions of the follow-up survey addressed the general applicability of the CPM to function point analysis. Survey questions 3 and 4 focus specifically on the usefulness of the CPM in calculating the VAF and using the fourteen GSCs. From the comments received in the initial survey, several respondents indicated that the CPM doesn't provide adequate definitions for each of the GSCs. As a follow-up to these comments, survey question 3 asks respondents if the CPM adequately defines the GSCs. The results from the 69 respondents are listed below in Table 28.

Table 28. The IFPUG CPM Adequately Defines the 14 GSCs

	Yes	No	Subtotal	No Reply	Total
Number	24	45	69	1	70
Percentage	34.8%	65.2%	100%		

Based on the above results, it appears that the CPM does not provide adequate definitions for each of the GSCs. Although the comments from the initial survey indicated that there was some degree of dissatisfaction with the definitions of the GSCs in the CPM, this level of dissatisfaction was somewhat of a surprise.

Survey Question 4. Survey question 4 was also drafted in response to several comments received from the initial survey that indicated that users were dissatisfied with the examples provided in the CPM in their applicability to today's information and

computing environment. The question asked respondents if the current CPM provides relevant examples of the GSCs that are applicable to today's technologies. The distribution of the 69 responses is listed below in Table 29.

Table 29. The IFPUG CPM Provides Relevant GSC Examples

	Yes	No	Subtotal	No Reply	Total
Number	4	65	69	1	70
Percentage	5.8%	94.2%	100%		

It is clear that the survey respondents do not perceive the examples provided in the current CPM to be relevant to today's technologies. Again, the basis for this question stemmed from the indication in the initial survey that the users were dissatisfied with the examples provided in the CPM in their applicability to today's software environment (real-time systems, web-based systems, etc.). The responses to this question clearly identify an area of concern among users of the current CPM. The results to these last two questions indicate IFPUG has some important decisions to make with respect to the definitions and examples provided in the CPM.

Survey Question 5. The initial survey results indicated that there was a lack of consensus among respondents on the usefulness and applicability of the current GSCs. However, among the responses to these open-ended questions, most of the comments could be categorized into five general areas of discussion. After analyzing the comments from the initial survey, the following five categories concerning the usefulness and applicability of GSCs were developed from the respondents' inputs:

- Keep current GSCs as is
- Eliminate GSCs altogether
- Revise the weighting scale of the GSCs

- Add and/or delete the GSCs to reflect today's technologies
- Improve definitions and provide more relevant examples of GSCs for today's technologies

Survey question 5 asked respondents the following question: If IFPUG was to re-evaluate the GSCs, what areas should be of primary focus? Using the five categories listed above, a simple ranking measurement technique was used. It required respondents to rank order the categories in order of importance with 1 being the most important and 5 being the least important. The technique for transforming the data into a summary rank order simply required summing all 69 respondents' rankings for each listing. Only one survey returned failed to indicate a response for this question. The category with the lowest total score indicates the highest preference rating. The summarized rank order is listed below in Table 30 along with the point sum for each category.

Table 30. Re-evaluating GSCs: Ranking Various Alternatives

Category Ranking	Responses	Point Sum
1. Better definitions and more relevant examples	69	146
2. Add and /or delete the GSCs to reflect today's technologies	69	160
3. Eliminate GSCs all together	69	241
4. Revise the weighting scale	69	246
5. Keep current GSCs as is	69	308

Consistent with survey questions 3 and 4, the number one recommendation survey respondents made was to provide better definitions and more relevant examples. Finishing a close second was the addition and/or deletion of GSCs. The issue of which items to add and/or delete is the subject of question 6. An interesting note on the above results concerns the idea of eliminating the GSCs altogether. Although it ranked a distant third overall, it had the second most number one rankings. The opinions on this element are very strong. Either respondent's want to eliminate them or they don't, there is very

little middle ground. For this option to be considered, IFPUG should explore a more scientific validation of the adequacy (accuracy) of FPA results with and without the GSCs.

Survey Question 6. Respondents' comments to the open-ended questions in the initial survey identified several factors they believe may improve the applicability of the GSCs to today's technologies. These factors were the most common items mentioned in the initial survey to improve the adequacy of the GSCs in assessing applications complexity to determine software size. After analyzing the comments from the initial survey, the following six factors were most frequently mentioned as factors that may affect applications complexity, but are currently not included:

- Application type (web-based, client-server, etc.)
- Existence of extensive algorithms
- Primary programming language
- System reliability
- Management and/or development team's expertise and capability
- Software written for multi-language use

The same approach used in question 5 was used here to determine the overall ranking for potential GSCs. The rankings along with the point sum for each factor were calculated from a total of 67 responses. Three surveys that were returned failed to indicate a response for this question. The results from this question are listed below in Table 31.

Table 31. Evaluating Potential GSCs: Ranking Various Suggestions

Factor Ranking	Responses	Point Sum
1. Application type (web-based client-server, etc.)	67	138
2. Existence of extensive algorithms	67	191
3. System reliability	67	267
4. Software written for multi-language use	67	277
5. Primary programming language	67	294
6. Management/development team's expertise and capability	67	301

It should be noted that these potential additions to the GSCs are the opinions of those practicing FPA in the field. Although Application Type and the Existence of Extensive Algorithms were consistently ranked the highest indicating where IFPUG may want to focus their attention, additional research is warranted to scientifically validate that the addition of these factors does indeed improve the accuracy of the GSCs in calculating the VAF. Also, it would be necessary to universally define these items, provide adequate examples, and determine the appropriate weighting.

Survey Question 7. Survey questions 7, 8, and 9 attempted to identify the levels of training and experience in function point counters. Wide variances among the function point counters in the levels of training and experience could lead to problems with standardization of the IFPUG FPA methodology. Furthermore, the lack of training or experience may result in a misunderstanding or misapplication of the GSCs and the VAF in performing function point analysis. According to Jones, "attempts to count function points by untrained, uncertified personnel can lead to variations of several hundred percent, which is about the range of variation noted with counts of source code volumes, where there are multiple standard and no certification programs" (Jones, 1998: 316).

Survey question 7 asked respondents what sort of initial training they received for applying the IFPUG FPA methodology. The question provided respondents with four

fixed alternatives. The results from the 68 respondents are listed below in Table 32. Two surveys that were returned failed to indicate a response for this question.

Table 32. Method of Initial Training in IFPUG FPA

Method	Responses	Percentage
1. Formal training through other organizations	42	61.8%
2. Formal training through IFPUG	17	25.0%
3. Self-taught using IFPUG counting Practice Manual	6	8.8%
4. Self-taught using other materials/references	3	4.4%

The above results indicate that the majority of the respondents receive their initial training from consultants, local users groups, etc., rather than directly from IFPUG. With this being the case, it is imperative that IFPUG ensure that those groups training others in function point analysis offer a standardized and IFPUG sanctioned product. Instructors should be certified and course materials should be approved by IFPUG, if feasible. Since there are numerous function point variants and even confusion among IFPUG members concerning the proper interpretation of the GSCs, the proper message must be sent at that initial training session. This will improve the consistent and uniform application of the IFPUG FPA methodology throughout the software sizing community.

Survey Question 8. A common concern for any professional is how to maintain proficiency in an acquired skill and also keep abreast of the latest trends and developments in the profession. Most often this is accomplished through recurring training or continuing education programs that are sponsored or recognized by a professional organization. Infrequent or the absence of individual training could possibly contribute to the misunderstanding or misapplication of the GSCs and the VAF in the IFPUG method. Survey question 8 asked respondents how often they receive recurring training or continuing education in the IFPUG FPA methodology. The question provided

the respondents with four mutually exclusive fixed alternatives. A total of 68 responses were received and the results are listed in Table 33. Two surveys that were returned failed to indicate a response for this question.

Table 33. Frequency of Recurring Training/Continuing Education

Frequency	Responses	Percentage
1. Other	31	45.6%
2. Annually	17	25.0%
3. Never	12	17.6%
4. Semi-Annually	8	11.8%

It is clear that the majority of the respondents receive recurring training or continuing education in the IFPUG FPA methodology at some interval other than annually or semi-annually. Respondents were not given the opportunity to elaborate on their responses; however, several commented that they receive ongoing training throughout the year by reading relevant publications and staying current with industry trends. While it is encouraging that most of the respondents receive some form of recurring training, 17.6% of the respondents indicated they never receive any sort of recurring training. IFPUG should strongly encourage its members to attend continuing education opportunities. Ideally, this training should be IFPUG sponsored to ensure a standard teaching curriculum.

Survey Question 9. In addition to a lack of training in a given area of expertise or skill, a lack of experience can lead to the misunderstanding or misapplication of a recognized procedure. Although experienced function point counters often calculate different numbers for identical software, it is probably safe to say that the methods used in calculating the function points are standard and accepted. The misunderstanding or

misapplication of the GSCs, along with the lack of confidence in the VAF may be directly related to the experience levels of the users of the IFPUG method.

This survey question sought to identify the level of experience of function point users responding to this survey. Respondents were provided five mutually exclusive fixed alternatives. This question prompted 68 responses with one no reply. The results to this question are listed below in Table 34.

Table 34. Years of Experience Using IFPUG FPA

Years of Experience	Responses	Percentage
Less than 1 year	9	13.2%
Between 1 and 3 years	13	19.1%
Between 3 and 5 years	14	20.6%
Between 5 and 10years	19	28.0%
Over 10 years	13	19.1%

The sample of respondents provides a fairly representative cross section of the entire population. No one-experience level is over represented in our sample, with 57% of the respondents having 5 years or less experience applying the IFPUG FPA, and the remaining 43% having greater than 5 years experience using the methodology.

Survey Question 10. The final question in the follow-up survey attempted to measure and identify alternative methods for calculating the VAF that are currently being practiced. The first part of this two-part question simply asked the participants if they use an alternative to the IFPUG method for calculating the VAF. This question prompted 68 responses. The results are listed below in Table 35.

Table 35. Use Other Than IFPUG Method for Calculating VAF

	Yes	No	Subtotal	No Reply	Total
Number	13	55	68	2	70
Percentage	19.2%	80.8%	100%		

Supporting the findings from question 1 in the initial survey, the overwhelming majority of the respondents adhere to the IFPUG method for calculating the VAF. As a result, IFPUG must carefully consider any changes to the GSCs used to determine the VAF. While the VAF issue is controversial, most practitioners continue to abide by the method set forth by IFPUG. Recognized as the benchmark organization for establishing standards within the function point user community, it appears respondents are looking for IFPUG to take the lead in determining the appropriate action.

The second part of survey question 10 asked respondents who answered "Yes" to the first part to briefly explain the alternative method they use for calculating the VAF. The ten alternative methods provided by respondents were grouped into four categories. The grouping of these alternative methods is listed in Table 36. Further research into these alternative methods may provide some insight on improving the current IFPUG VAF calculation.

Table 36. Alternative Methods for Computing VAF

Method	Responses
Use estimating experience to derive adjustment	2
Use factors related to productivity rates	3
Use COCOMO, SEER or related models	3
Other	2

Findings

Investigative Question 1. Are the fourteen general systems characteristics commonly used to derive the value adjustment factor? Additionally, are alternative methods employed to adjust for the applications complexity? The first part of this investigative question was answered by the responses to questions 1 and 2 from the initial survey and questions 1 and 10 from the follow-up survey. Results from all four of the

questions indicate that the majority of respondents routinely use the GSCs to derive the VAF.

The responses to survey question 5 from the initial survey along with the information gathered from question 10 of the follow-up survey indicate that there are some users of function points employing alternative methods to adjust applications complexity. However, the number of respondents utilizing an alternative method is relatively small and the majority of respondents continue to apply the IFPUG method in spite of its perceived limitations.

Investigative Question 2. What is the perceived accuracy and validity, in terms of representing applications complexity, of each general systems characteristic, as determined by the users? Also, which of the characteristics are considered most important? The responses to question 3 of the initial survey indicate that confidence in the perceived accuracy and validity of the GSCs in representing applications complexity is lacking. Based on the measurement criterion established by the researchers, six of the characteristics failed to achieve a rating of average, while the remaining eight characteristics rated average to slightly above average. These results may require further in-depth research to determine why the factors are perceived to be inaccurate or invalid.

The open-ended responses to question 3 identified the perceived inadequacy of the current GSCs to capture the applications complexity of new technologies in today's environment. Respondents repeatedly voiced the need for characteristics that address such areas as real-time systems, web-based or client server applications, and programming language issues. Regardless of the perceived accuracy or validity of a

GSC, further research is needed to scientifically validate the accuracy of all current or proposed characteristics before deleting or adding to the established GSCs.

Investigative Question 3. How applicable are the general systems characteristics across various platforms, languages, and environments? The responses to question 4 of the initial survey reveal that a majority of respondents believe that the GSCs can be applied to all platforms and languages; however, there was a sizeable number of responses that felt that the current GSCs were not applicable to all platforms and languages. The respondents who answered "No" provided several open-ended responses where they felt that the GSCs were not applicable. The major categories of responses mentioned included concern that the current GSCs were not relevant in today's network/web-based environment, that the current GSCs only apply to mainframe environments, and that the current GSCs fail to take into account the functionality provided by the platforms or languages.

Research Question. Do the fourteen general systems characteristics adequately represent the applications complexity required to adjust the function point counts of a software-sizing project? Specifically, are the GSCs the best way to adjust for applications complexity or is there a better alternative to the current IFPUG standard? The results of both surveys and the associated answers to the investigative questions above indicate that the GSCs do not adequately represent the applications complexity required to adjust the function point count. However, since a superior alternative method was not revealed through the research effort, continued reliance on the current GSCs along with further research to improve their applicability appears to be the safest course.

V. Conclusions and Recommendations

Overview

This chapter further discusses the findings presented in Chapter IV and documents the conclusions supported by this research. In addition, potential limitations of this study are addressed, as well as recommendations for future research and possible courses of action for IFPUG.

Conclusions

As stated in Chapter I, the purpose of this research was to assess the adequacy of the general systems characteristics in deriving a value adjustment factor for calculating final function point counts. Based on the results of the two survey instruments, the initial survey and subsequent follow-up, it is clear that the GSCs, in their current form (including definitions and examples), do not adequately represent the applications complexity required to adjust the function point counts of a software-sizing project. While the results of the surveys were conclusive, it should be noted that this finding is primarily based on the opinions of the users of the IFPUG function point methodology, and should not be construed as a scientific validation of the GSCs inadequacy.

Although the current GSCs remain controversial, they continue to be applied as outlined in the latest version of the IFPUG's *Function Point Counting Practices Manual.* As such, all fourteen GSCs are considered when calculating the value adjustment factor. However, it appears that this practice is adhered to for the lack of a superior alternative. While this research can not conclusively state that one factor is more accurate than any other in terms of assessing applications complexity, it is evident that 8 out of 14 GSCs are perceived to be more useful for their intended purpose. Specifically, reusability,

online update, heavily used configuration, installation ease, operational ease, and online data entry appear to be current characteristics that may need to be reconsidered in light of the emerging technology found in today's software environment. In addition, application type and the existence of extensive algorithms may be two potential factors that IFPUG may want to consider adding if further research supports their applicability in assessing applications complexity.

Although a re-evaluation of the current GSCs is prudent, the controversial nature of the GSCs appears to be concentrated at more of a macro level than with individual characteristics. Nearly 50 percent of the respondents ranked the alternative of leaving the GSCs intact, but providing better definitions and more relevant examples as their top choice for improving the applicability of the GSCs. While this appears to be the safest course of action, there appears to be a strong contingent within the function point community that would prefer that the GSCs be eliminated altogether. This is based on analysis that revealed 30 percent of respondents ranked this alternative as their first choice for re-evaluating and improving the IFPUG methodology. This supports Jones' argument, as stated in Chapter I, that "the ambiguity and partial subjectivity of the adjustments have led several researchers to assert that the counts might be more accurate if the complexity adjustment were dispensed with and not even used at all" (Jones, 1998: 311). As stated in Chapter IV, the opinions on this option are very strong. Either respondents want to eliminate them or they don't; there is very little middle ground. For this option to be considered, IFPUG should scientifically validate the accuracy of FPA results with and without the GSCs.

Finally, the survey results support the opinion that the current GSCs are applicable across various platforms and languages. However, the overwhelming concern is with their relevance to today's technological environment. The current GSCs may not adequately represent areas such as real-time systems, web-based applications, and client-server domains in assessing applications complexity.

Research Limitations

The main limitations of this research can be summarized within two distinct categories. First, the study design may limit the usefulness of the findings without first conducting further research. The survey instruments were adequate for addressing the basic research question set forth in Chapter I. However, the concept of the GSCs is a measurable, although somewhat subjective, metric. As such, their accuracy and adequacy should be quantifiable through a more scientific validation process. Although significant changes to the GSCs should not be considered until such a research effort has been performed, it does not diminish the overall importance of the findings presented here. IFPUG, as a standard setting organization, has a responsibility to its members to address those controversial issues that ultimately may improve standardization and increase acceptance within the function point community. This body of work has identified those issues and provided IFPUG with several concrete alternatives that should be considered.

Another potential limitation to the study design is the inherent disadvantage of using self-administered surveys. Specifically, a low response rate may limit the generalizability of these findings to the entire population. For the purpose of this study, the population consisted of IFPUG members and the sample size was simply the IFPUG

mailing list comprised of 508 members. As previously mentioned, self-administered mail surveys with a response rate of 30 percent are often considered satisfactory (Cooper and Emory, 1995: 282). The 18.5 percent rate for the initial survey and the 13.8 percent rate for the follow-up survey can be considered low by all accounts. In retrospect, the use of follow-up mailings or surveys should have been considered to increase the response rate

In addition to the previous limitations, a lack of control over the administration of the surveys to the IFPUG members limits the generalizability of the findings. Due to privacy concerns, the IFPUG home office was unable to release either the e-mail addresses or conventional addresses of the IFPUG members, who served as the population for this research effort. As a result, administering the survey indirectly through IFPUG prevented the use of an accurate survey tracking system. Therefore, there was no way to correlate the initial survey responses with the follow-up survey responses, which prevented the identification of respondents who did not reply at all, only replied to one survey, or replied to both surveys.

Finally, the second broad category threatening these findings is the concept of external validity. Cooper and Emory define external validity as the ability of the research findings to be generalized across persons, setting, and times (Cooper and Emory, 1995: 149). The introduction of generalizabilty above referred to the applicability of the findings to all IFPUG members due to the low response rate. The concern here is the ability to generalize across the entire function point using community, not just the IFPUG members identified as the population for this study. Obviously, by limiting the pool of respondents to IFPUG members, the generalizability to this broader category may be

suspect. A future study may want to consider expanding the pool of respondents to include non-IFPUG members to improve external validity.

Recommendations

The data presented in this research effort should provide IFPUG with valuable information in considering the future of the GSCs. Based on the results of this analysis, the logical recommendation is for IFPUG to maintain their current policy regarding the existing general systems characteristics, with immediate emphasis being placed on providing better definitions and relevant examples to reflect the applicability of the GSCs to today's technology. However, it is also apparent that additional research must be done to effect any substantive changes to the GSCs. Ultimately, it may be necessary to make drastic changes in the methodology employed to adjust functional size for applications complexity. Whether that involves adding and/or deleting factors or eliminating the GSCs altogether remains to be seen. Any such changes must be accompanied by scientific validation of the appropriateness of a new alternative. In addition, the results of any such study should conclusively show the superiority of a new alternative since any changes to the current approach will have a significant impact on the using community.

Finally, IFPUG must continue to emphasize training and continuing education as an integral part of the profession. Specifically, IFPUG must ensure that initial training, whether through IFPUG or other organizations, is standardized with an IFPUG sanctioned product. While there may always be dissent, it is imperative that a unified and consistent message be taught from the beginning.

Appendix A: General Systems Characteristics Definitions

The following definitions are from the IFPUG guidelines for assigning values to each of the fourteen general systems characteristics as provided by Garmus and Herron in their 1996 text *Measuring the Software Process*.

1. **Data Communication**. 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. Protocol is a set of conventions which permit the transfer or exchange of information between two systems or devices. All data communication links require some type of protocol. Score as follows:
   - 0 Application is pure batch processing or a stand alone PC.
   - 1 Application is batch but has remote data entry or remote printing.
   - 2 Application is batch but has remote data entry and remote printing.
   - 3 On-line data collection or TP (teleprocessing) front end to a batch process or query systems.
   - 4 More than a front-end, but the application supports only one type of TP communications protocol.
   - 5 More than a front-end, but the application 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 follows:
   - 2 Application does not aid the transfer of data or processing function between components of the systems.
   - 3 Application prepares data for end-user processing on another component of the systems such as PC spreadsheets and PC DBMS.
   - 4 Data is prepared for transfer, transferred, and processed on another component of the systems (not for end-user processing).
   - 5 Distributed processing and data transfer are on-line and in one direction only.

6 Distributed processing and data transfer are on-line and in both directions.

7 Processing functions are dynamically performed on the most appropriate component of the systems.

3. **Performance**. Application performance objectives, stated or approved by the user, in either response or throughput, influenced (or will influence) the design, development, installation, and support of the application. Score as follows:

   0 No special performance requirements were stated by the user.

   1 Performance and design requirements were stated and reviewed but no special actions were required.

   2 Response time or throughput is critical during peak hours. No special design for CPU utilization was required. Processing deadline is for the next business day.

   3 Response time or throughput is critical during all business hours. No special design for CPU utilization was required. Processing deadline requirements with interfacing systems are constraining.

   4 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 follows:

   0 There are no explicit or implicit operational restrictions.

   1 Operational restrictions do exist, but are less restrictive than a typical application. No special effect is needed to meet the restrictions.

   2 Some security or timing considerations exist.

   3 There are specific processor requirements for a specific piece of the application.

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 systems.

5. **Transaction Rate**. The transaction rate is high and it influenced the design, development, installation, and support of the application. Score as follows:

   0 No peak transaction period is anticipated.

   1 A peak transaction period (monthly, quarterly, seasonally, annually) is anticipated.

   2 A weekly peak transaction period is anticipated.

   3 A daily peak transaction period is anticipated.

   4 High transaction rates 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 rates 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. **On-line Data Entry**. On-line data entry and control functions are provided in the application. Score as follows:

   0 All transactions are processed in batch mode.

   1 1% to 7% of transactions are interactive data entry.

   2 8% to 15% of transactions are interactive data entry.

   3 16% to 23% of transactions are interactive data entry.

   4 24% to 30% of transactions are interactive data entry.

   5 Over 30% of transactions are interactive data entry.

7. **End-user Efficiency**. The on-line functions provided emphasize a design for end-user efficiency. They include the following:

- Navigational aids (for example, function keys, jumps, dynamically generated menus)
- Menus
- On-line help/documentation
- Automated cursor movement
- Scrolling
- Remote printing (via on-line transactions)
- Preassigned function keys
- Submission of batch jobs from on-line transactions
- Cursor selection of screen data
- Heavy use of reverse video, highlighting, colors, underlining, and other indicators
- Hard copy user documentation of on-line transactions
- Mouse interface
- Pop-up windows
- As few screens as possible to accomplish a business function
- Bilingual support (supports two languages; count as four items)
- Multilingual support (supports more than two languages; count as six items)

Score as follows:

0 None of the above.

1 One to three of the above.

2 Four to five of the above.

3 Six or more of the above but there are no specific user requirements related to efficiency.

4 Six or more of the above and stated requirements for end-user efficiency are strong enough to require design tasks for human factors to be included (for example, minimize key strokes, maximize defaults, use of templates, etc.).

5 Six or more of the above and stated requirements for end-user efficiency are strong enough to require use of special tools and processes in order to demonstrate that the objectives have been achieved.

8. **On-line Update**. The application provides on-line update for the Internal Logical Files. Score as follows:

   0 None.

   1 On-line update of one to three control files. Volume of updating is low, and recovery is easy.

   2 On-line update of four or more control files. Volume of updating is low, and recovery is easy.

   3 On-line update of major internal logical files.

   4 In addition, protection against data loss is essential and has been specially designed and programmed in the systems.

   5 In addition, high volumes bring cost considerations into the recovery process. Highly automated recovery procedures with minimum of operator intervention.

9. **Complex Processing**. Complex processing is a characteristic of the application. The categories include the following:

   - Sensitive control (for example, special audit processing) and/or application specific security processing
   - Extensive logical processing
   - Extensive mathematical processing
   - Much exception processing resulting in incomplete transactions that must be processed again; for example, incomplete ATM transactions caused by TP interruption, missing data values, or failed edits.
   - Complex processing to handle multiple input/output possibilities; for example, multi-media, device independence.

   Score this characteristic as follows:

   0 None of the above.

   1 Any one of the above.

   2 Any two of the above.

   3 Any three of the above.

   4 Any four of the above.

   5 All five 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 follows:
    - 0 There is no reusable code.
    - 1 Reusable code is used within the application.
    - 2 Less than 10% of the application considered more than one user's needs.
    - 3 Ten percent or more of the application considered more than one user's needs.
    - 4 The application was specifically packaged and/or documented to ease reuse, and application is customized to user at source code level.
    - 5 The application was specifically packaged and/or documented to ease reuse, and application is customized to use at source code level by means of user parameter maintenance.

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 systems test phase. Score as follows:
    - 0 No special considerations were stated by user, and no special set-up is required for installation.
    - 1 No special considerations were stated by user, but special set-up is required for installation.
    - 2 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.
    - 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 considered to be important.
    - 4 In addition to (2), automated conversion and installation tools were provided and tested.
    - 5 In addition to (3), 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 systems test phase. The application minimizes the need for manual activities, such as tape mounts, paper handling, and direct on-location manual intervention. Score as follows:

    0 No special operational consideration other than the normal back-up procedures were stated by the user.

    1-4 Select the following items that apply to the application. Each item has a point value of one, except as noted otherwise:

    - Effective start-up, back-up, and recovery processes were provided but operator intervention is required.
    - Effective start-up, back-up, and recovery processes were provided but no operator intervention is required (count as two items).
    - The application minimizes the need for tape mounts.
    - The application minimizes the need for paper handling.

    5 Application is designed for unattended operation. Unattended operation means no operator intervention is required to operate the systems other than to start up or shut down the application. Automatic error recovery is a feature of the application.

13. **Multiple Sites**. The application has been specifically designed, developed, and supported to be installed at multiple sites for multiple organizations. Score as follows:

    0 There is no user requirement to consider the needs of more than one user/installation site.

    1 Needs of multiple sites were considered in the design, and the application is designed to operate only under identical hardware and software environments.

    2 Needs of multiple sites were considered in the design, and the application is designed to operate only under similar hardware and/or software environments.

    3 Needs of multiple sites were considered in the design, and the application is designed to operate under different hardware and/or software environments.

4 Documentation and support plan are provided and tested to support the application at multiple sites, and application is as described by (1) or (2).

5 Documentation and support plan are provided and tested to support the application at multiple sites, and application is as described by (3).

14. **Facilitate Change**. The application has been specifically designed, developed, and supported to facilitate change. Examples include the following:

- Flexible query/report capability is provided.
- Business control data is grouped in tables maintainable by the user.

Score as follows:

0 There is no special user requirement to design the application to minimize or facilitate change.

1-5 Select which of the following items apply to the application:

- Flexible query/report facility is provided that can handle simple requests; for example, and/or logic applied to only one internal logical file (count as one item).

- Flexible query/report facility is provided that can handle requests of average complexity; for example and/or logic applied to more than one internal logical file (count as two items).

- Flexible query/report facility is provided that can handle complex requests; for example, and/or logic combinations on one or more internal logical files (count as three items).

- Control data is kept in tables that are maintained by the user with on-line interactive processes, but changes take effect only on the next business day.

- Control data is kept in tables that are maintained by the user with on-line interactive processes, and the changes take effect immediately (count as two items).

Appendix B: DeMarco's Complexity Factor Definitions

The following categories for correcting for variations in complexity were defined by DeMarco in his 1982 book *Controlling Software Projects*:

1. **Separation**: primitives that divide incoming data items
2. **Amalgamation**: primitives that combine incoming data items
3. **Data Direction**: primitives that steer data according to a control variable
4. **Simple Update**: primitives that update one or more items of stored data
5. **Storage Management**: primitives that analyze stored data, and act based on the state of that data
6. **Edit**: primitives that evaluate net input data at the man-machine boundary
7. **Verification**: primitives that check for and report on internal inconsistency
8. **Text Manipulation**: primitives that deal with text strings
9. **Synchronization**: primitives that decide when to act or prompt others to act
10. **Output Generation**: primitives that format net output data flows (other than tabular outputs)
11. **Display**: primitives that construct two-dimensional outputs (graphs, pictures, etc.)
12. **Tabular Analysis**: primitives that do formatting and simple tabular reporting
13. **Arithmetic**: primitives that do simple mathematics
14. **Initiation**: primitives that establish starting values for stored data
15. **Computation**: primitives that do complex mathematics
16. **Device Management**: primitives that control devices adjacent to the computer boundary

Appendix C: Initial Survey and Cover Letter

Survey: Complexity Adjustments for Function Point Analysis

Value adjustment factor (VAF) indicates the general functionality provided to the user. The VAF is sometimes referred to as the applications complexity. The VAF is a set of adjustment factors used to calculate a final adjusted function point total of a software project or application.

1. Do you use the IFPUG Value adjustment factor to adjust the final function point count when sizing software applications and projects?

   A) Yes B) No

   If you answered no to the above question, briefly explain why you do not use the IFPUG VAF to adjust the final function point count.

   _____
   _____
   _____
   _____

2. Do you use the 14 general systems characteristics to compute a value adjustment factor as outlined by the International Function Point Users Group (IFPUG) when determining the functional size of software projects in function points?

   A) Yes B) No

If you answered yes to Question 2 continue, otherwise skip to Question 5

3. Listed are the 14 general systems characteristics as outlined by IFPUG. Please indicate the accuracy and validity of each factor used in adjusting final function point counts.

   1-None 2- Little 3-Average 4-High 5- Very High

Factor	Value				
Data Communications	1	2	3	4	5
Distributed Data Processing	1	2	3	4	5
Performance	1	2	3	4	5
Heavily Used Configuration	1	2	3	4	5
Transaction Rate	1	2	3	4	5
Online Data Entry	1	2	3	4	5
End-user Efficiency	1	2	3	4	5
Online Update	1	2	3	4	5
Complex Processing	1	2	3	4	5
Reusability	1	2	3	4	5
Installation Ease	1	2	3	4	5
Operational Ease	1	2	3	4	5
Multiple Sites	1	2	3	4	5
Facilitate Change	1	2	3	4	5

Please list any additional complexity factors not listed above that you consider important in adjusting final function point counts in determining functional size of software.

_____
_____
_____

4. Do you feel the general systems characteristics can be applied to all platforms and languages?

A) Yes B) No

If you answered no, briefly explain why not.

_____
_____
_____
_____
_____

If you use the IFPUG general systems characteristics exclusively, continue with Question 7

5. Listed are several alternative methods of complexity adjustment for function point analysis. Please circle all the methods that you use for sizing software projects.

   a) British Mark II function point method (19 complexity adjustment factors)

   b) SPR function point method (3 complexity adjustment factors)

   c) DeMarco function point method (22 complexity adjustment factors)

   d) SEER Function Based Sizing

   e) Other (Please list) _____

6. What are the primary advantages or reasons for using alternative methods of complexity adjustment for function point analysis versus the 14 characteristics as outlined by IFPUG? Please circle all that apply.

   a) Easier to Understand

   b) Ease of Use

   c) Less Complex

   d) More accurately reflect complexity

   e) Other (Please list) _____

7. Do you use Function Point based sizing for (Please circle all that apply):

   a) Size Analysis

   b) Productivity Analysis

   c) Cost Analysis

   d) Schedule Analysis

   e) Inventory

   f) Other Metrics (Please List)_____

8. In the past two years, how many software-sizing estimates have you or your organization performed?

   a) Less than 2

   b) Between 2 and 5

   c) Between 6 and 10

   d) More than 10

9. What is the average final function point count from your software-sizing estimates?

   a) Less than 100

   b) Between 100 and 1,000

   c) Between 1,000 and 5,000

   d) Over 5,000

10. In what type of operational environment is the software that you size utilized?

    a) Commercial MIS

    b) Military MIS

    c) Real-time systems

    d) Scientific applications

    e) Telecommunications

    f) Other_____

If you perform size estimates for multiple application types, please indicate percentage breakdown. (Example: 50% Commercial MIS, 50% Telecommunications)

Additional Comments:

27 Jan 99

Captain Joseph C. Willoughby
2Lt Michael D. Prater
Graduate Students GCA-99S
Air Force Institute of Technology
WPAFB OH 45433-77765

IFPUG Member
C/O IFPUG Home Office
5008-28 Pine Creek Drive
Westerville OH 43081-4899

Dear IFPUG Member

Thank you for your interest in our survey. We appreciate your participation and the valuable insight you will lend to our research.

With the validation of function point accuracy in business and data processing applications, the continuing attempts to apply the function point concept to real-time and scientific environments, and the ongoing efforts to educate the software estimating community, function point analysis continues to grow in popularity. However, the variations in complexity adjustments for function point totals remain a controversial aspect of function point analysis. The enclosed survey is intended to collect information from users of function point analysis and their application of complexity factors. The information from the survey should provide us with some idea of the adequacy of the existing complexity factors.

While the attached 10-questions are intended to capture the essence of the issue, it is by no means comprehensive. We encourage you to include any additional comments you think will be valuable to our study. We are requesting your completed survey be returned to IFPUG by **March 19, 1999**.

We look forward to your input and if you have any questions, please feel free to contact us at jwilloug@afit.af.mil or mprater@afit.af.mil.

Sincerely,

//SIGNED//
JOSEPH C. WILLOUGHBY, Captain, USAF
Graduate Student, Air Force Institute of Technology

//SIGNED//
MICHAEL D. PRATER, 2Lt, USAF
Graduate Student, Air Force Institute of Technology

Attachment:
Survey

Appendix D: Follow-up Survey

Complexity Adjustments for Function Point Analysis

Dear IFPUG Member,

Below is a brief 10-question survey that will further assist us in our research of Complexity Adjustments for Function Point Analysis. This is a follow-on survey in the subject area, and if you are interested in receiving our results from the initial survey, please E-mail us at the address below with a request.

Please complete the survey by bolding or underlining the desired response, or typing in the desired number for ranking, when applicable. When completed, please E-mail to joseph.willoughby@afit.af.mil or fax to the attention of Captain Joe Willoughby at (937) 656-7988. If possible, please respond to the survey within five working days of receipt.

Your assistance is sincerely appreciated.

Joseph C. Willoughby, Captain, USAF
Michael D. Prater, 2LT, USAF
Air Force Institute of Technology

1. When performing function point analysis, do you strictly adhere to the procedures as outlined in the current IFPUG Counting Practices Manual (CPM)?

   ☐ Yes

   ☐ No

2. In general, does the current IFPUG CPM adequately document the procedures necessary to perform function point analysis?

   ☐ Yes

   ☐ No

3. In calculating the Value adjustment factor (VAF), does the current CPM ***adequately define*** the 14 general systems characteristics (GSCs)?

   ☐ Yes

   ☐ No

4. In calculating the VAF, does the current CPM provide ***relevant examples*** of the GSCs that are applicable to today's technologies?

☐ Yes

☐ No

5. As indicated in the survey results, there is lack of consensus on the usefulness/applicability of the current GSCs. If IFPUG was to re-evaluate the GSCs, what areas should be of primary focus? (Rank in order of importance: 1- most important 5-least important)

_____ Better definitions and more relevant examples of GSCs for today's technologies

_____ Revise the weighting scale

_____ Add and/or delete the GSCs to reflect today's technologies

_____ Eliminate GSCs all together

_____ Keep current GSCs as is

6. The initial survey results identified several factors that may improve the applicability of the GSCs to today's technologies. Listed below are the most frequently mentioned responses to factors, which may affect applications complexity, but are currently not included. Based on their impact to applications complexity, rank these factors in order of importance. (Rank in order of importance: 1-most important 6-least important)

_____ Application type (web-based, client-server, etc.)

_____ Existence of extensive algorithms

_____ Primary programming language

_____ System reliability

_____ Management /development team's expertise and capability

_____ Application written for multi-language use

7. What sort of initial training did you receive for applying the IFPUG FPA methodology?

   ☐ A. Formal training through IFPUG

   ☐ B. Formal training through other organizations (consultants, local user group, etc.)

   ☐ C. Self-taught using IFPUG Counting Practices Manual

   ☐ D. Self-taught using other materials/references

8. How often do you receive recurring training/continuing education in the IFPUG FPA methodology?

   ☐ A. Never

   ☐ B. Semi-Annually

   ☐ C. Annually

   ☐ D. Other

9. How many years of experience do you have using the IFPUG FPA methodology?

   ☐ A. Less than 1 year

   ☐ B. Between 1 and 3 years

   ☐ C. Between 3 and 5 years

   ☐ D. Between 5 and 10 years

   ☐ E. Over 10 years

10. Do you use an alternative to the IFPUG method for calculating the VAF?

    ☐ Yes

    ☐ No

    If you answered yes, briefly explain procedure:

Appendix E: Initial Survey Content Analysis and Comments

Content Analysis to Open-ended Questions:

- The number preceding the survey response is the survey index number assigned to completed and returned surveys.
- The following entries were transposed directly from returned surveys with a minimum of editing to maintain the integrity of the responses.

Question 1:

1. **Do you use the IFPUG Value Adjustment Factor to adjust the final function point count when sizing software applications and projects?**

A) Yes B) No

If you answered no to the above question, briefly explain why you do not use the IFPUG VAF to adjust the final function point count.

	# Mentioned
A. The GSCs are arbitrary & outdated	6
B. The GSCs don't relate to cost/effort	3
• Use cost models with unadjusted function points	
C. Inaccurate – no improvement over unadjusted FPs alone	6
D. Confusing	2
E. Weighting Factors are arbitrary/inappropriate	2
F. Other	1

Survey responses:

5. The 14 factors are arbitrary and outdated. The weighting of the factor is arbitrary. Research shows that correlation of size and effort is as strong for unadjusted size as for adjusted size.

7. The VAF is an unsuccessful attempt to make function points relate to cost/effort. The amount of impact each GSC can have makes them ineffective in this way. I use a cost model to get cost information and unadjusted function points for sizing estimates.

15. Respondent uses SEER-SEM for estimating effort and schedule. This tool requires unadjusted function points.

19. I only use it if there is a contractual obligation to do so. I don't recommend its use. The VAF is very dated, and there is no evidence that any of the factors can impact the size by +/- 5%. The factors are also very subjective. (E.g., How can installation and conversion requirements affect the application size for a legacy system, where the purpose is to control maintenance costs? Why is an on-line system worth 5% more than a batch system?)

28. The current GSCs do not represent the quality and technical software attributes exhibited by today's systems. Software product factors which have a major impact on productivity (E.g., language complexities are not represented). Current scenarios do not reflect industry norms.

38. We use backfire methods. Plus, Organization does not have a standard method by which to manage IT (application) projects. As such there is no agreed upon method to estimate size.

41. Not seen to be major enhancement to accuracy in current work.

48. Working with an outsourcing account that has elected not to use VAF-with my concurrence. Both of us feel that the factors are not reflective of application environment complexity. The factors are of little value and are out of date.

56. Only use because we are contractually obligated. However, we avoid them in estimates because they are confusing.

57. We calculate the VAF for submission of projects to ISBSG's repository. But we only use unadjusted function counts when estimating or comparing productivity rates.

64. The scaling issue....VAF transformation is appropriate only for an interval scale, while FP count is ordinal. VAF is almost always near 1 for our work anyway.

72. It does not provide additional, meaningful information. Our results have been within the bounds of acceptable variance without the VAF. (plus or minus 10% on FP based estimates). We compare the results to another FP count that did not use the VAF.

78. Current VAF is outmoded for today's environment.
- It doesn't correctly reflect the difference in complexity between a host vs. a C/S system.
- The correlation between UFPs and effort isn't worse than the correlation between adjusted FPs and effort (no use for the estimate).
- UFPs are easily interpreted as an indirect product measure of functionality.
- When adjusted....FPs are a mixture of memory attributes measuring what?

85. I have found the VAF to provide little value in our real-time environment. We use unadjusted FPs and use various productivity rates to arrive at very accurate estimates. We also successfully use unadjusted full function points.

90. We gave up on VAF due to lack of clarity in the definitions. Some factors seem out of date and are not useful

93. I use COCOMO II to estimate effort.

Question 3:

Please list any additional complexity factors not listed above that you consider important in adjusting final function point counts in determining functional size of software.

	# Mentioned
A. New technology: standard interfaces, protocols, and web applications	7
B. System reliability/security	3
C. Factor for real-time applications: reflect complexity of algorithms	2
D. Management/development teams capability	2
E. Development platform/primary programming language	2
F. Software Designed for Multiple language use versus single language use	2
G. Requirements Volatility	1
H. Purchased application or custom built	1
I. Size of databases	1
J. Checkpoint parameters	1
K. Help facilities in software applications	1

Survey Responses:

2. Respondent believes internationalism could be a factor: whether software is to be developed for single language support as opposed to multiple language support.

9. Conformance to standard interfaces and protocols.

10. Bilingualism is undervalued.

17. Whether the application is web based, client/server, etc.

18. New technology into an environment. Learning curve of a new language. None of the GSCs have an impact to functional size.

19. Even if you come up with a valid list today, it would probably change tomorrow.

26. All these factors should be considered when adjusting the final FP (the current GSCs). Unfortunately the associated definitions have not been updated to reflect today's technology. We have mainframe, client/server and web applications and continue to apply these GSCs and the VAF to all estimates. It's either "all or none" and we've chosen all.

27. System reliability/system high availability.

29. Development platform/primary programming language.

31. Size of databases.

40. People factor, client factor, political factor, and developer's ease with technology used.

44. Checkpoint parameters.

46. Operational ease...I think the items should be revised.

48. Help facilities are far more pervasive in today's applications than one or two low equations reflect-they should be an adjustment factor. Also, GSCs need to reflect web/networking requirements.

51. Complex security requirements, routing and approvals.

70. The existence of extensive algorithms should be better reflected.

71. We find these items useless for real-time. We look at architecture; math volume (represented by operands and operators in mode document); task interactions and system links.

80. Whether the application is a purchased package or custom built.

81. How volatile the requirements are.

86. Is it Web-based? Does it interface between multiple platforms?

92. Reliability, documentation, levels of automation, and mission criticality.

93. Organization maturity, team/people capabilities.

Question 4:

Do you feel the General Systems Characteristics can be applied to all platforms and languages?

A) Yes B) No

If you answered no, briefly explain why not.
Note: Some respondents provided responses even if they answered Yes to the above question

	# Mentioned
A. Not relevant to today's environment (networks/web applications)	11
B. Only applicable in mainframe environment	7
C Fail to take into account the functionality provided by platforms/languages	3
D. Do not add any practical value to their metrics	2
E. Not applicable in real-time environment	2
F. Not suitable for software reusability	1
G . Only applicable in MIS domain	1

Survey responses:

3. Respondent has not counted real time systems so he/she is unsure of applicability of GSCs to these systems.

14. The characteristics should ensure that they are taking into consideration the functionality provided by the languages and platforms being developed (E.g., Oracle Developer 2000 has a lot more inherent functionality than COBOL).

15. For reusability, difficult to quantify by percentage of code, for 4GL applications.

16. Yes.... but, they definitely lean towards mainframe.

21. Several do not fit well in today's computer environment. For example: #6 Online Data Entry (most new applications are 100% on-line, #8 Online Update (most have online update) and operational ease (clearly written for old host based environment). Makes it difficult to explain to new practitioners that function points are relevant in the new environment.

23. Yes, But they need to be updated to account for today's environments and technologies (i.e., the Internet).

25. They seem to be directed towards legacy applications.

29. Industry analyses (E.g., ISBSG database) suggests that the GSCs are not applicable.

31. They are too closely tied to the MIS domain and are often not applicable to other domains such as telecommunications.

36. Just don't seem to capture the complexity of today's client/server applications and the variety of technologies they use.

38. With the use of enterprise resource planning applications (E.g., Powerbuilder, PeopleSoft, Lotus Notes, Oracle forms); there are serious questions about the relevancy of function points to these tools.

48. As currently stated, they barely differentiate between batch and on-line. Batch is penalized when it is sometimes much more difficult to maintain.

56. Not applicable to C/S, real time, and multimedia.

57. The VAF is an arbitrary set of factors that probably applied to a typical software application built in 1985. However, today's software has varying profiles of quality and technical factors depending on their functional domain.

58. Some characteristics are only for batch or on-line; so they are not usable at PC or batch processing.

59. Usually self-evident when working with multiple client SMEs (small to medium enterprises).

65. Several apply to older platforms-the mainframe environment. They need updating to reflect current technology.

66. IFPUG needs to re-evaluate the 14 GSCs in light of the evolving increase of platforms, operating systems, languages, design techniques, and tools, etc.

67. They do not show the complexity behind web based systems.

86. It is definitely slanted towards mainframes. The GSCs need to be revised and updated. They are very outmoded.

90. The VAF is slanted toward a mainframe, large application environment.

91. The definitions need to expand to cover new technologies.

94. Requires poetic license in interpretation outside the mainframe world.

Responses to request for additional comments following Question 10:

At the end of the survey we asked respondents for any additional comments. We received a wide range of responses and here is our best attempt to categorize the data.

	# Mentioned
A. GSC characteristics are outdated/little value today	9
B. VAF needs to be removed from FPA methodology	4
C. VAF needs to be revised to be applicable	8
D. Respondent uses current FPA methodology for a variety of metrics	5
E. Respondents believe function points and complexity are two separate issues, requiring two separate metrics	5
F. Too much subjectivity in the VAF/GSC measurements	4
G. GSCs not weighted properly	2
H. Other	2

1. Individual accuracy is suspect. Definitions are old and mainframe oriented. There is doubt regarding the independence of each value and whether they should be of equal value. However, still get better correlation between FPs and effort/time scale if I use VAF then if I don't.
Respondent thinks that it is important to distinguish between technical complexity as measured by the VAF and functional complexity as determined by the assessing the function types.

7. Respondent believes that VAF should be eliminated-let the cost models estimate cost and let the FPs relate only to the functionality delivered.

8. The 14 GSCs should be better focused on functionality and streamlined.

9. a. Functional size has to be, by definition, independent of technology choices.

b. A GSC should only influence functional size if the Characteristic is independent of technology choices. In our opinion, the only GSC potentially in this category is Complex Processing.

c. The concept for VAF has value as a standard, separate metric for input in project estimation. Since the VAF reflects technology (software and hardware) choices, it can be used in evaluation of different delivery strategies. As a general principle, we should seek to choose a delivery strategy that minimizes the VAF.

d. The concept of a VAF has value as a standard, separate metric for input to resource allocation for support. It is probable that the VAF for project development is not the same as the VAF for product support, that is, a GSC may have one impact on the project but a different one upon support, e.g. installation ease.

e. The adjustment needs to be able to express the potential impact on the project. The current potential impact is too small for some characteristics, e.g. Distributed Data Processing, Performance, and Complex Processing.

f. The current factor can only be applied to the total product. It needs to be applicable:

- Project/product subsets-this is important where a project is "parceled" for development to different resource groups, both internal and external, or where different technologies are used (hybrid systems).
- Project phase-this is important where, say, the "build" is outsourced. Some characteristics make no sense at this level.
- Enhancements-this is similar to the first point. The current method means that the VAF for the application is applied but it often has no relevance whatsoever to the Enhancement Project in hand.

g. Some IFPUG Guidelines are out of date or irrelevant, in particular Operational Ease and Facilitate Change.

10. Online data entry is an outdated question.

12. Function point is the functional size and ***should not be mixed*** with complexity. Complexity often has a greater impact on the project than you can express through the 14 GSCs, so the complexity should be a metric of its own. Complexity (in process and/or product) is a factor which influences the process (effort, schedule) and perhaps the architecture of the product, but not the functional size.

13. Function points and complexity should be separated. Example: FPs x Complexity, where complexity could be expressed as type, productivity, etc.

19. Any factor which affects size should be applicable to the individual objects being sized (i.e. to transactions and files), not to the whole system (E.g., if 10% of a system involves distributed processing, where "processing functions are dynamically performed on the most appropriate component of the system" we should not allocate an extra 5% to the whole system). Consider what "extensive mathematical processing" means to developers of commercial MIS vs. what it means to developers of scientific applications-it is not the same!

21. One of the biggest criticisms for function points is that they do not appear to handle the newer types of platforms because some of the GSC descriptions do not fit well. However, some sort of adjustment factor is necessary because we also still are using a lot of the old host based legacy systems, some primarily batch. Some of the GSCs are difficult to evaluate, especially from the user's view, because they are based on

technology. For example, GSC #1 requires knowledge of the TP communications protocol. GSC # 2 is also difficult to evaluate because the user may not know where the processing occurs.

22. I think it is important to distinguish between technical complexity as measured by VAF and functional complexity as determined by assessing the function types. The individual accuracy is always suspect. Definitions are old and mainframe oriented. There has to be doubt about the independence of questions and whether they should all be of equal value. Having said all that I get better correlation for FPs against effort and time scale using the VAF than without.

24. Despite the apparent aging of the GSCs, we continue to use them because we are trying to follow the methodology, not change it!

27. GSCs seem to target computing as it was in the 1970's, not as it is now.

32. GSCs need to be brought in line with today's technology. They are definitely the weak part of FPs.

35. Respondent uses Mk II FPA method. VAF was dropped from this method after Mk II FPA Version 1.3 release. VAF may have been valid 10-20 years ago, but now it is clearly inappropriate and misleading. Furthermore the ISO Standard 14143 with which we wish to comply excludes technical and quality requirements from functional size measures.

39. It would be a great achievement for the FP method to either drop the use of the VAF or find another method. There should be a way of improving the counting of batch applications; the complexity in batch just disappears when using the current method.

40. Believe that the concept should be abolished. We use a risk analysis approach.

46. I think the items should be revised.

47. The current VAF is geared to old technology; needs to be updated to reflect current technology.

51. The measures do contain subjectivity that's hard to overcome, or the list of attributes just doesn't seem to be complete or weighted properly. There is a temptation to modify the score up or down a point. Also, each component is weighted the same which is rarely true!

56. Technical complexity should be removed. A better version of a Feature Point "user" algorithm would have value.

57. In Australia the VAF has not been used since 1994 by experienced FPA practitioners. The only time it is used is to keep consistency in databases. We usually

recommend clients measure functional size then collect all quality and technical factors which will affect their product and project factors, which will impact productivity and adjust productivity rate accordingly, not the size. None of the GSCs have validity to adjust size only-they have validity to adjust effort. They are a measure of quality and technical features, not size.

59. Our group performs all the FPA for IBM Netherlands. We use it for all kinds of metrics and proposals.

60. We use FPs for determining support ratios. Ex: Hrs of Support/FP and FP/Work-month.

70. We use FPA to estimate project size, and we use Cost/FP, time-to-market delivery rate to assess projects. We use support rate and reliability rate to assess production applications.

71. FPs have real problems in military systems. They need considerable adaptation, as such systems are heavily process oriented, not I/O bound. Feature points are a good first step, but real-time function points (by Reifer) are better.

73. An advanced set of value adding GSCs is critical to the ability of FPA to add value. I am all for their use, once some additional research has been done. The GSCs should (must) represent a leading edge rating of modern characteristics of projects.

81. I would try to eliminate judgmental language in the GSC descriptions. This would aid in a more uniform application of FPs.

83. GSCs need to be updated for today's technology. For example, GSC #6 "Online Data Entry".... almost all of today's systems have online data entry. The GSCs are the only part of FPA that is not very objective.

84. FPA is great for sizing, but we backfire to SLOC and use a COCOMO costing tool for schedule and cost estimation.

85. All of our telecommunications software operates in a real-time environment. In this regard, VAF has provided little value to us.

87. Everything I have read lately indicates that using/not using has no significant difference. I would like to see unbiased study.

91. I would like to see the GSCs remain the same as far as content and weighting, as this would preserve the validity of the data we have collected so far. I think all we need for a few of them is to expand the definitions to include new technologies.

92. Precision of IFPUG FPs is very poor. Many systems have functions with more than 50 DETs, which are not accounted for accurately.

More and more systems are developing complex internal processing, which IFPUG FPs totally ignore. For systems that are mostly internal, FPs are useless!

94. I question whether the VAF should have as great an impact as it does today.

Bibliography

Boehm, B. Software Engineering Economics. New York: Prentice-Hall, Inc., 1981.

Boehm, R. "Frequently Asked Questions (and Answers) Regarding Function Point Analysis." FAQ page, n. pag. http://compuserve.com/homepages/softcomp/fpfaq.htm. 25 June 1997.

Cooper, D.R., & Emory, C.W. Business Research Methods. Boston, MA: Irwin McGraw-Hill, 1995.

DeMarco, T. Controlling Software Projects. Englewood Cliffs NJ: Prentice Hall, 1982.

Dreger, J. B. Function Point Analysis. Englewood Cliffs NJ: Prentice Hall, 1989.

Ferens, D.V. Class handout: Lesson 1, COST 677, Quantitative Management of Software. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB OH, Winter Quarter 1999.

Ferens, D.V. Class handout: Lesson 5, COST 677, Quantitative Management of Software. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB OH, Winter Quarter 1999.

Galorath Incorporated. SEER-SEM™ User's Manual:Version 5. December 1998.

Garmus, D., & Herron, D. Measuring the Software Process. Upper Saddle River, NJ: Prentice Hall PTR, 1996.

Henderson, G.S. The Application of Function Points to Predict Source Lines of Code for Software Development. MS thesis, AFIT/GCA/ LSY/92S-4. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, September 1992 (AD-A258447).

Jones, T.C. Applied Software Measurement. New York: McGraw-Hill, 1991.

Jones, T.C. Estimating Software Costs. New York: McGraw-Hill, 1998.

Jones, T.C. "Should the 'Lines of Code' Metric Be Viewed as Professional Malpractice?" The Voice, 2: 10-15, (December 1997).

Lewis, M.D. A Comparative Study and Estimation of the Life-Cycle Cost Impact of Application of Real-Time Non-Intrusive (RTNI) Monitoring Technology to Real-Time Embedded Systems. MS thesis, AFIT/GSS/LAS/97D-2. School of Systems and Logistics, Air Force Institute of Technology (AU), Wright-Patterson

AFB OH, December 1997 (AD-A329955).

Lokan, C.J. "An Empirical Analysis of Function Point Adjustment Factors." ADFA Computer Science Technical Report CS03, Australian Defense Force Academy, Canberra, Australia, December 1998.

Longstreet, D. "Using Function Points." Unpublished article, n.pag. http://www.softwaremetrics.com. 30 July 1999.

Low, G.C., & Jeffery, D.R. "Function Points in the Estimation and Evaluation of the Software Process," IEEE Transactions on Software Engineering, 16(1): 64-71 (January 1990).

Symons, C. R. Software Sizing and Estimating Mk II FPA (Function Point Analysis). Chichester, England: John Wiley and Sons, Ltd., 1991

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1999	Master's Thesis

4. TITLE AND SUBTITLE
THE ADEQUACY OF THE FOURTEEN GENERAL SYSTEMS CHARACTERISTICS AS FUNCTION POINT ADJUSTMENT FACTORS

5. FUNDING NUMBERS

6. AUTHOR(S)
Michael D. Prater, 1Lt, USAF
Joseph C. Willoughby, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
2750 P Street
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCA/LAS/99S-3

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/IFSD
2241 Avionics Circle
WPAFB OH 45433-7334

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The purpose of this research is to assess the perceived adequacy of the 14 general systems characteristics (GSCs) in deriving a value adjustment factor (VAF) for calculating final function point counts. Two self-administered surveys were used to collect data. Based on the survey results, it is clear that the 14 GSCs currently do not adequately represent the applications complexity required to adjust function point counts.

The current GSCs remain controversial and while this research can not conclusively state that one factor is more accurate than any other, it is evident that certain GSCs are perceived to be more useful for their intended purpose. In addition, other factors may need to be considered as additions to the current GSCs. Although a reevaluation of the current GSCs individually is prudent, their controversial nature appears to be concentrated at more of a macro level. For the most part, the respondents recommendation is to leave the GSCs intact but provide better definitions and more relevant examples.

Notwithstanding these results, there appears to be a strong contingent within the function point community that would prefer that the GSCs be eliminated altogether. Furthermore, there is overwhelming concern with their relevance to today's technological environment.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Software Engineering, Computer Programs, Coding (Computers), Software(Computers)	110
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL