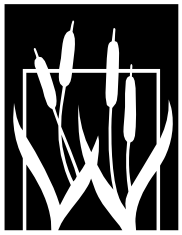

Whitemarsh
Information Systems Corporation

Function Points
A Strategy to Determine the Size and Cost
of
Database-centric Business Information Systems

Whitemarsh Information Systems Corporation
2008 Althea Lane
Bowie, Maryland 20716
Tele: 301-249-1142
Email: Whitemarsh@wiscorp.com
Web: www.wiscorp.com

Table of Contents

1. Objective

The objective of this paper is to present the Whitemarsh approach to the definition of Function Points within database-centric Business Information Systems, the process of counting Function Points, and the use of two metrics derived from Function Point counts to accomplish Business Information System estimating.

The following individuals have been helpful in reviewing an early draft: Hank (Henry) Lavender, Russ Eggen, Chris Cordes, Michael Gould, and Rocky Phelps.

2. Topics Covered

The topics is this paper are:

- Function Points defined
- What is Counted
- What Function Points Measure
- Computing Function Points
- Applying Function Point Counts

3. Function Points Defined

The most obvious first question is: What is a "Function Point"?

From the Wikipedia:

A Function Point is a unit of measurement to express the amount of business functionality an information system provides to a user.

From the Developer Daily (www.devdaily.com):

Simply stated, Function Points are a standard unit of measure that represent the functional size of a software application. In the same way that a house is measured by the square feet it provides, the size of an application can be measured by the number of Function Points it delivers to the users of the application.

The Developer Daily then provides an example:

A good example is when I had my house built two years ago. I worked with a very straightforward home builder and he basically said "Al, you have two choices here. First, how many square feet do you want to build? Second, what quality of materials do you want to use?"

He continued "Let's say that you want to build a house that is 2,000 square feet. If you want to use cheap materials we can build it for $80 per square feet. That's $160,000. If you want to go top of the line then you're looking at more like $110 per square foot, and that's $220,000. What would you like?"

The IEEE provides a definition for Function Point as follows:

<provide $19 for a download of the article that contains the definition> Since these Short Papers are Free, the request was declined.

Finally, from David Longstreet (http://www.softwaremetrics.com/fpc.html)

Human beings solve problems by breaking them into smaller, understandable pieces. Problems that may initially appear to be difficult are found to be simple when dissected into their components, or classes. When the objects to be classified are the contents of software systems, a set of definitions and rules, or a scheme of classification, must be used to place these objects into their appropriate categories. Function Point analysis is one such technique: FPA is a method to break systems into smaller components, so they can be better understood and analyzed. It also provides a structured technique for problem solving.

- *Breaks Systems into logical business components*
- *Ordinal Measure*
- *Sizes business functionality*
- *Easily learned and applied*

Function Points are a unit measure for software much like an hour is to measuring time, miles are to measuring distance or Celsius is to measuring temperature. Function Points are an ordinal measure much like other measures such as kilometers, Fahrenheit, hours, so on and so forth.

Function Points measure software by quantifying its functionality provided to the user based primarily on the logical design. Frequently the term end user or user is used without specifying what is meant. In this case, the user is a sophisticated user. Someone that would understand the system from a functional perspective --- more than likely someone that would provide requirements or does acceptance testing.

Whitemarsh was very impressed with the Function Point materials from Longstreet Consulting and highly recommends them for training and consulting on this topic. Their particulars are:

David Longstreet
Longstreet Consulting
2207 SW Walnut
Blue Springs, MO 64015
Web: http://www.softwaremetrics.com/
Email: David Longstreet <David@SoftwareMetrics.Com>
Phone: 1-816-739-4058

In the approach advocated by David Longstreet, he suggests breaking down Business Information Systems into the object types that have to be built, can be readily observed and can be easily counted. For each object class, create at least two sizes: small and large. Since all Whitemarsh database applications are Business Information Systems, the components that are both readily observable and countable are:

- Third Normal Form Data models
- Screens
- Browses within screens
- Menus
- Third Normal Form processes that were hand coded

The Whitemarsh development environment is Clarion from SoftVelocity (www.softvelocity.com). Whitemarsh chose this environment back in the late 1980s because Clarion's product development direction was towards ELIMINATING the computer programmer from most if not all the mundane, boring design and coding of Business Information System components that are the *same* through code generation.

By *same*, it is not meant, *same* in content, but *same* in form and style. Content is, of course, individualized to the functional intent of the database and Business Information System. Thus, while a sales system is inherently different from an HR system, or from an inventory System, or from a customer management system, or from an order entry system, or from an invoicing system, all these systems are can be seen as being built through a standard set of "content independent" Information Technology building blocks of:

- Third Normal Form Data models
- Screens
- Browses within screens
- Menus
- Third Normal Form processes that are hand coded

4. What is Counted

In the prior section, what must be counted are those things that must be built by the designer, analyst, or programmer. These include therefore, requirements, design, and the components of the Business Information System itself, that is, menus, screens, and the like.

For the purposes of this paper, Requirements and Design activities are excluded from Function Point counts. That's because these activities are completely different from the actual activities of Business Information System building. Mixing the two sets of activities almost always leads to disaster.

Recently, the "Agile folks" have championed having the steps of requirements, analysis, design, build, and test completely integrated and iterative. That seems to work for only those IT systems that are inherently simple, small, and intuitive. The alternative to Agile is the BDUF approach. That is, Big Design Up Front. That approach, however also seems to commonly lead to disaster because by the time the BDUF is complete, the problem has either gone away or has substantively changed.

Whitemarsh's Business Information System development approach, which was first developed in the early 1980s, and is fully described in the Whitemarsh book, *Strategy for Successful Development of Business Information Systems*, is a modified BUDF approach. That is, one that is mission and data centric, is combined with prototyping, and ultimately accomplished Business Information System construction through the use of code generaton.

Because of the Whitemarsh development approach, and because of the development environment (e.g., Clarion) there is a profound impact on the types and quantity of components that have to be built by the person constructing the Business Information System.

If, as in the case of Clarion, there are many and very sophisticated templates that can be employed, the very existence of the components and quantities of those components which exist are affected. For example, suppose a browse window that displays a collection of data records, similar to that depicted in Figure 3 has to be built. There exists, at a minimum, the following Information Technology components within that browse window:

- Menu item to invoke the process that displays the browse.
- Window frame
- Data browse that lists the columns
- Vertical and horizontal navigation arrows and implied processes
- Close button for the window
- Add, Delete, and Modify buttons and implied data processes
- Data record sorting by column selection capability
- Help button and supporting text and display facilities
- Minimize, maximize, and close window controls

All these components exist independent of whether its construction environment is Clarion or some other Windows based tool. The question therefore, is not whether these components exist, but whether they have to be counted to arrive at an accurate Function Point count. The answer is simple. If the programmer has to construct them, then yes. Else, no.

Similarly, there would have be to be component parts breakdowns for every other Business Information System object type. That is, for data models, menus, screens, and the like. Now, in Clarion, and in other Business Information System development environments to a greater or lesser degree, many of these Business Information System components are either greatly facilitated, or generated altogether. Thus, the statement, "the development environment dramatically affects the types of components that either or have to be counted" is a key determination prior to any Function Counting process. In Clarion, for example, it's not that these components don't exist, its that because they are automatically generated that they don't have to be individually identified nor have to have a Function Point count assigned to them.

Another issue in Function Point counting is the state of the Business Information System process model and its supporting database data model prior to the counting effort. If both the data model and the process model are in Third Normal Form, what is counted and the value assigned to what is counted are greatly reduced.

Most everybody in database understands what Third Normal Form means with respect to data modeling. Bill Kent in 1982 (http://www.bkent.net/Doc/simple5.htm) has summarized it simply that a database table's design must successfully address the following three issues to ensure that a table is in Third Normal Form.

- Single-valued vs. multi-valued facts.
- Dependency on the entire key.
- Independent vs. dependent facts.

Every column in the table must represent only single valued facts. Every column in the table must be dependent on the entire primary key of the table, not a partial value of the primary key. Finally, the non-key columns in the table must not rely on another non-key column in the table for a complete understanding of its value-based meaning. When data models are in Third Normal Form, the Business Information Systems that are built to collect, store, retrieve, and delete data are in their simplest form. That's because there are fewer Business Information System components to count, which, in turn, caues the overall Business Information System Function Point count to be lower.

In a Whitemarsh conducted study in the middle 1980s, the count of Business Information System components for Third Normal Form data-driven Business Information Systems was 4.6 times lower than for a process driven, but functionally equivalent Business Information Systems.

There is also a Third Normal Form process that is analogous to Third Normal Form data. These processes are characterize by having:

- Only one purpose which is reflected in its name
- No nested subprocesses
- No logic branching based on data values

Traditionally, these three characteristics are also described by minimizing (e.g., low) coupling and maximizing (e.g., high) cohesion. See for a greater explanation, http://en.wikipedia.org/wiki/Coupling_(computer_science).

When both data and process are in Third Normal Form, there are fewer components to count, and each component is simpler to construct, faster to design, easier to program, and has far fewer programming logic errors. Hence, each such component can have a lower Function Point count assigned to it. This paper therefore presumes that there has been an effort prior to the actual construction of the Business Information System to have both data and process in Third Normal Form.

5. What Function Points Measure

Function Points measure the size of a unit of accomplishment within a domain. In this situation, the domain is a database-centric Business Information System. The classes of accomplishment are those enumerated above (e.g., Third Normal Form Data model, ..., Third Normal Form process). The product of Function Point analysis is the quantity of Function Points. Function Points do not attempt to represent the cost and/or time of accomplishing the unit of accomplishment. That's determined when the Function Point count is accomplished through a chosen methodology and development environment.

Suppose, for example, the quantity of work for a Business Information System was equivalent to a "mile-sized" Business Information System. While there certainly are no mile-sized Business Information Systems, this serves to illustrate what a Function Point is and is not. A common question would be how long does it take to traverse the mile, that is, to build the Business Information System.

What would an answer of 20 minutes mean? Is the traveler walking fast or slow? To assess that you'd first have know if the mile is across flat terrain, up and down steep hills, over reasonable turf or through loosely packed sand. Is it a sunny cool day, or driving rain, or 100 degrees Fahrenheit with high humidity? It would also depend on whether the person is a child or a normal size adult. If the walker is a four year old child, they are walking very fast, but with short steps. Or if a normal sized person, the stride would be normal. If the adult is a "fast walker," the 20 minute-mile would be considered slow.

These measures are not relevant to Function Points because they relate to the length of time require to traverse the mile. Rather, these measures related to the person, terrain, and the environment. The only valid Function Point measure for this example would be "feet," that is, 5280 in a mile. That's because the measure is independent of the mode, person, or method of

accomplishment. For a child, that 5280 feet might be accomplished in 5280 paces. For a normal size adult it might be 2112 paces. The final determiner would be how fast the person (child or adult) is accomplishing each pace to accomplish the 20 minute mile. For the adult it's 1.76 paces per second. For the child it's 4.4 paces per second. In this case the child is clearly walking faster.

Function Points are not about the length of your stride, how many strides per minute, the weather, or how tall or short you are. Those are all dependent metrics associated with the duration of traversing the mile, the terrain of the mile, or a way of indicating that you walking slow or fast. Function Points are independent metrics about the mile itself. That is, that there are 5280 feet in the mile.

A common complaint about Function Points is that its an accurate count only after building the Business Information System. True enough. Because of this problem, history-based metrics have to be built through an approach such as:

- Determine what classes of objects are to be counted.
- Determine what defines a simple and a complex object.
- Determine the quantity of functionality represented by an object class that represents a single Function Point.
- Standardize the Business Information System development methodology
- Standardize the Business Information System development environment.
- Keep accurate metrics about the accomplishment of the several Business Information System efforts.
- Classify the types of Business Information Systems that are to be analyzed.
- Perform a Function Point count of representative Business Information Systems within a Class
- Determine the key metrics: Quantity of Function Points per database table, and cost per Function Point.

The first four steps are essential to "regularize" the Business Information System environment. Once regularized, and with several already created Business Information Systems, a thorough Function Point count can occur. Once counted and with the count verified, the quantity of Function Points per database table can be determined.

Thereafter, the cost per Function Point can be determined. Armed with those two metrics, estimates of subsequent Business Information Systems can be developed. It is critical at this point that the estimated dollars (and time derived from money) be a guideline NOT a stricture. If the estimate becomes a stricture while there is just a small "statistical sample" the behavior of the system builders will be affected from what should be normal. Once the Key Metrics have been stabilized they can be used with confidence to engineer Business Information System estimates and schedules.

When this type of effort is performed over a number of different Business Information Systems within each class, the ability to employ Function Points to make early and accurate

estimates of Business Information System development is possible. The following example shows how Function Points can be "boot-strapped" into existence within your enterprise.

In the late Spring of 2000, Whitemarsh had an opportunity to develop a membership management system for an international organization. Long before this time, Whitemarsh had standardized its Business Information System development methodology and its Business Information System development environment. Whitemarsh received the contract to build the system based on best-effort rather than a cost or time estimate. By the middle of the Summer of 2001, the effort was finished and the cost to the client was $350K. The client was receiving operational versions of the membership management system with incrementally increasing functionality starting in November 2000.

The effort was functionally object-oriented and except for specialized processes, was 95% code-generated from highly engineered database and functional designs.

In mid July the IT group indicated to Whitemarsh that the governing board of the organization was concerned about the cost of the effort. Whitemarsh had known of Function Points, but had never applied it because of the common complaint (see above). Whitemarsh proceeded to perform a thorough Function Point count. The Function Point quantity was 6605. Based on billing records the cost per Function Point came to about $50. Capers Jones research in 2000 indicated that the cost of a Business Information System Function Point was about $455. It was because of the methodology and software development environment employed by Whitemarsh that the cost was reduced by about 90%.

A key metric produced from this effort was the quantity of Function Points per table. It came to just over 80. That metric is very useful because in a data-driven effort, the quantity of database tables can be determined very early in the Business Information System development cycle. That metric, coupled with the $50 per Function Point provides a quick method of estimating the Business Information System.

The rest of this paper sets out the strategy employed to count Function Point for the membership management system. This strategy is illustrative of the approach that has to be taken to arrive at a correct Function Point count for your class of Business Information System, your development methodology, and your development environment for your organization.

You should carefully review the Function Point materials at the Longstreet website and tailor these materials to your classes of Business Information System, your development methodology, and your development environment so that you can develop both accurate counts and also the two key metrics.

One key feature about the Clarion development environment is that it has a DCT metadata file for the database dictionary (the DCT file) and a metadata file for the Business Information System itself (the APP file). The DCT metadata file contains the database data model that is equivalent to a robust SQL data definition language (DDL) file. The APP metadata file is a pre-code-generation version of the complete Business Information System. The DCT metadata file can be enhanced with custom coded data integrity procedures. The APP metadata file can contain linked-in custom coded processes.

Clarion takes the DCT and the APP files, along with the linked-in custom coded processes, and automatically generates the actual Business Information System programming language statements. This set of program code is then compiled, linked and turned into an executable. It is because of the existence of the DCT and the APP files that Clarion can always support changes to the database data model, menus, screens, browses, and hand-coded processes independently one from the other. This is a true use of object-oriented engineering principles of encapsulation, polymorphism, and of metadata-based code-generation-reuse. Clarion is the realization of the promise of object-oriented strategies.

6. Computing Function Points

As stated above, the class of applications developed by Whitemarsh is database-centric Business Information Systems. These are mainly on-line systems with several hundred to thousands of database tables. Business Information Systems typically have an overall menu structure on the opening frame of the application. The predominant processes center on Add, Delete, and Modify screens. Each of these screens may, in turn, have multiple browses. Finally, there commonly exists embedded routines for data integrity checking beyond that provided by the database's data model, and for accomplishing and other processes. These custom-coded processes, if they are in Third Normal Form, range from 5 to 50 lines of source code.

Table 1 presents the quantity of Function Points per object class for particular object classes. In this example, a Table Reference means a Select, Get, Add, Delete, or Modify to a database table.

Object Class		Function Point Count	Description or Notes
3NF Database table	Simple	4	Database table consisting of 5 or fewer database columns.
	Complex	8	Database table consisting of more than 5 database columns
Simple Screen	Simple Browse	4	Less than 5 data element Less than 2 table references
	Complex Browse	8	More than 5 screen elements More than 2 table references
Complex Screen	Simple	4, 4, 4, 4 Thus 16	Example: Four simple browses. Since each is has fewer than 9 screen elements and less than 2 Table References, each is 4

Object Class		Function Point Count	Description or Notes
	Complex	8, 8 Thus 16	Example: Two distinct browses. Since each has more than 10 screen elements and more than 2 table references
Update	Simple	4	Fewer than 3 Table References and fewer than 15 database columns
	Complex	8	More than 3 Table References, and more than 15 database columns
Menu	Functional Hierarchy in Menu	2	Each top level menu item with nested hierarchies of submenu items within each menu item
3NF Embed Process	Simple	4	Custom coded procedure routine which involves 5 or fewer database columns and fewer than three database table references.
	Complex	8	Custom coded procedure routine which involves more than 5 database columns and more than three database table references.

Table 1. Object Classes and assigned Function Point count.

The figures that follow depict that items within a Clarion generated Business Information System that are readily available to be counted. Counting other objects in Clarion is a complete waste of time because Clarion auto-generates them. The figures correspond to the object classes in Table 1.

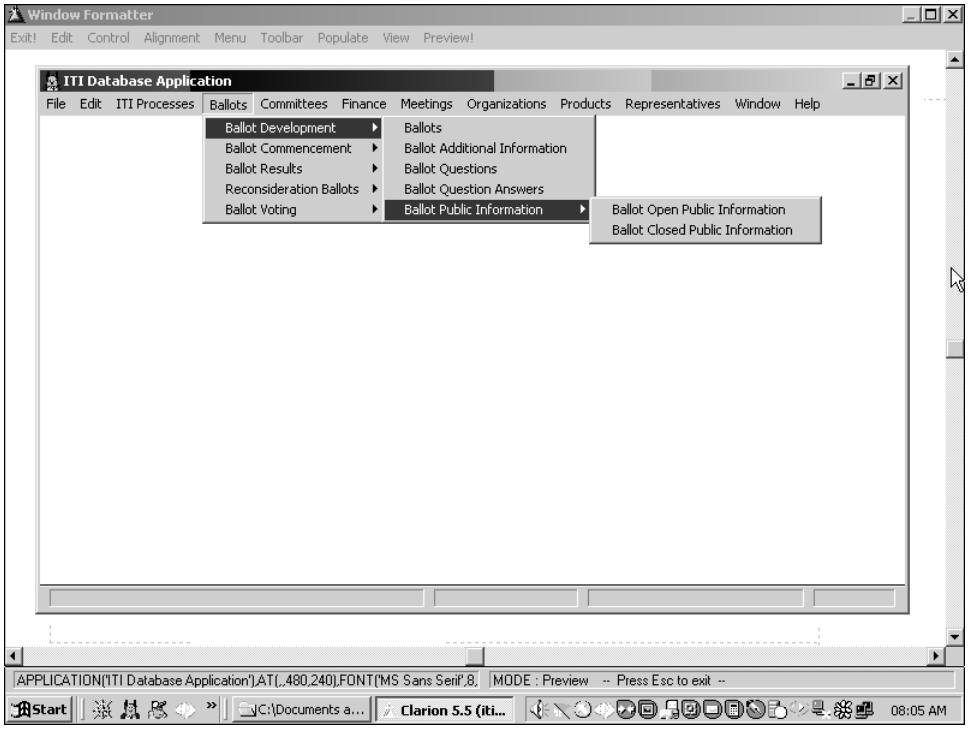

Figure 1. Menu.

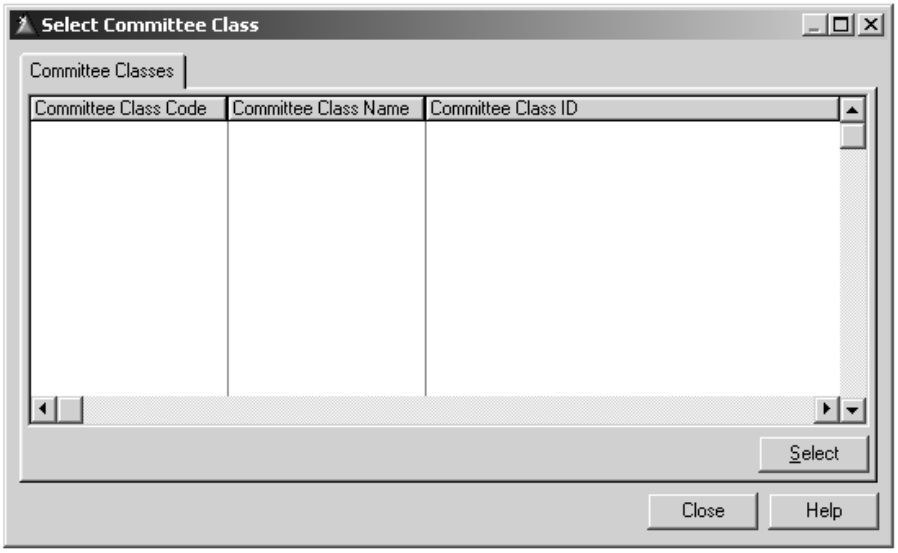

Figure 3. Simple Browse.

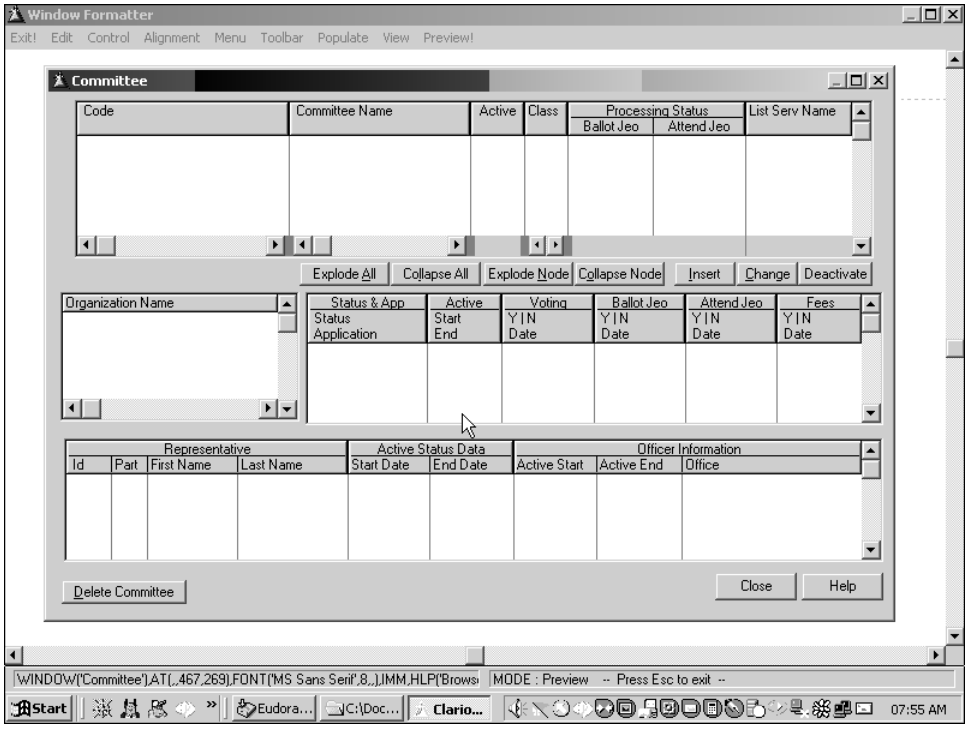

Figure 2. Complex Browse

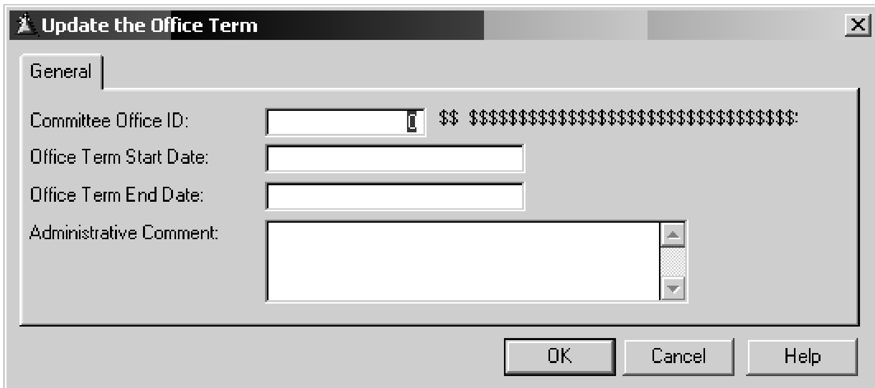

Figure 4. Simple Update

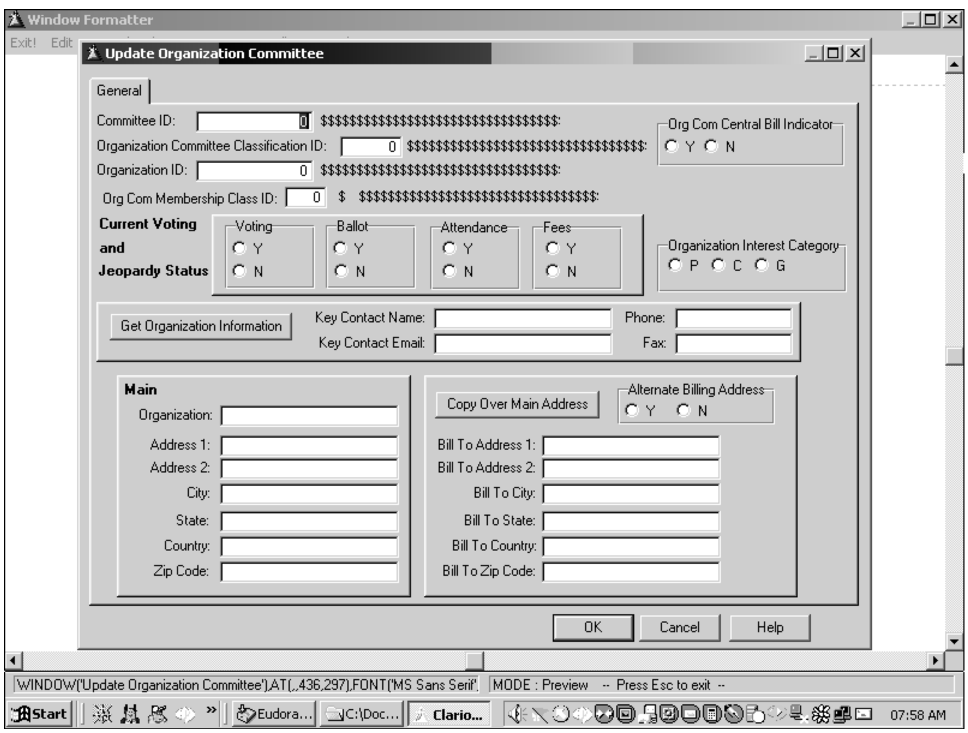

Figure 5. Complex Update.

Custom Coded Procedure Routine

```
BuildCommitteesOfInterest   Routine
        NewRepLV = FstTag:PtrM(Tag:NewRepresentativeTag)
        QCount = Records(CommitteeTagQueue)
        Display
        Loop QIndex = 1 to Qcount
          Get(CommitteeTagQueue, QIndex)
          CommitteeLV = CTQ:CommitteeID
          Access:NewRepComm.PrimeRecord
          NEWRPCOM:NewRepresentativeID = NewRepLV
          NEWRPCOM:CommitteeID = CommitteeLV
          NEWRPCOM:RepresentativeParticipantClassificationID = 1
          IF Access:NewRepComm.INSERT()<>Level:Benign then
             ACCESS:NewRepComm.CancelAutoInc.
        End
        NewTag:PtrM(Tag:NewRepresentativeTag)
        ThisWindow.reset(1)
```

Figure 6. Simple Custom Code Embed

Custom Coded Procedure Routine

```
ProcessNewRepresentative    Routine
   Do ExistingRepresentativeCheck
   If ExistingRepresentativeCheck = 'N' Then
      Message('Failed Existing Representatives Check....exiting')
      Exit
   Else
      Do ExistingOranizationCheck
      If ExistingRepresentativeCheck = 'N' Then
         Message('Failed Existing Representatives Check....exiting')
         Exit
      Else
         If ExistingRepresentativeId = 0 then
            Do RepresentativeCreate
            If RepresentativeCreateCheck = 'N' then
               Message('Representative insert failure....exiting')
               Exit
            Else
               TargetRepId = REP:RepresentativeID
               If ExistingOrganizationId = 0 then
               Do OrganizationCreate.
               If OrganizationCreateCheck = 'N' then
                  Message('Organization insert failure....exiting')
                  Exit
               Else
                  Do OrgComInsert
                  If OrgComInsertCheck = 'Y' Then
                     Do ComRepPartInsert
                     If ComRepPartInsertCheck = 'N' Then
                        Message('ComRepPart insert failure....exiting')
                        Exit
                     End
                  Else
                     Message('Organization Committee insert failure....exiting')
                  End
               End
            End
         End
      End
   End
```

Figure 7. Complex Custom Code Embed.

SQL Database Table

```
CREATE TABLE [dbo].[BallotQuestionAnswer] (
	[BallotQuestionAnswerID] [int] IDENTITY (1, 1) NOT NULL ,
	[BallotQuestionID] [int] NULL ,
	[BallotQuestionAnswerNumber] [int] NOT NULL ,
	[BallotQuestionAnswerText] [varchar] (255) NOT NULL ,
	[CommentRequired] [varchar] (1) NULL ,
	[LastModRepID] [int] NULL ,
	[LastModDate] [datetime] NULL
) ON [PRIMARY]
GO
```

Figure 8. Simple Database Table.

SQL Database Table

```
CREATE TABLE [dbo].[Representative] (
        [RepresentativeID] [int] IDENTITY (1, 1) NOT NULL ,
        [Salutation] [varchar] (4) NULL ,
        [ActiveStartDate] [datetime] NULL ,
        [ActiveEndDate] [datetime] NULL ,
        [PasswordExpiryDate] [datetime] NULL ,
        [FirstName] [varchar] (30) NOT NULL ,
        [MiddleName] [varchar] (30) NULL ,
        [LastName] [varchar] (30) NOT NULL ,
        [Password] [varchar] (32) NOT NULL ,
        [Address1] [varchar] (80) NOT NULL ,
        [Address2] [varchar] (80) NULL ,
        [City] [varchar] (30) NOT NULL ,
        [State] [varchar] (30) NOT NULL ,
        [ZipCode] [varchar] (10) NOT NULL ,
        [Country] [varchar] (30) NULL ,
        [EmailAddress] [varchar] (100) NULL ,
        [Phone] [varchar] (30) NOT NULL ,
        [Fax] [varchar] (30) NULL ,
        [AlternateBillingAddress] [varchar] (1) NULL ,
        [AccessibilityNeed] [char] (1) NULL ,
        [BillingAddress1] [varchar] (80) NULL ,
        [BillingAddress2] [varchar] (80) NULL ,
        [BillingCity] [varchar] (30) NULL ,
        [BillingState] [varchar] (30) NULL ,
        [BillingCountry] [varchar] (30) NULL ,
        [BillingZipCode] [varchar] (10) NULL ,
        [BillingEmail] [varchar] (120) NULL ,
        [BillingPhone] [varchar] (30) NULL ,
        [BillingFax] [varchar] (30) NULL ,
        [BillingContact] [varchar] (80) NULL ,
        [LastModRepID] [int] NULL ,
        [LastModDate] [datetime] NULL
) ON [PRIMARY]
GO
```

Figure 8. Complex database table.

7. Applying Function Point Counts

The membership management system had already been built when the Function Points were counted. That is the reason why the counts are so precise. Once these counts were created, a key metric, Function Points Per Table was computed. At that point, other already built Clarion Business Information Systems were evaluated to assess whether the key metric could have been used to predict their costs. Records had been kept on a daily basis as to the quantity of hours for each Business Information System. The predictions were all within 10%. The business functions and the counts for the membership management system were:

- Ballot -- 0845
- Committee -- 1701
- Invoice -- 514
- ITI Processes -- 2199
- Meetings -- 545
- Organization -- 429
- Product -- 180
- Representative -- 162

The Function Point total is 6605. The key metric, Function Points Per Table was computed by dividing the total function count by the quantity of tables. That is: 6605/ 80 = 82.5 Function Points per table.

The total quantity of hours charged was then determined. The duration was 14 staff months. The total quantity of staff hours, at 160 hours/month, 2240 hours. That meant that each Function Point took 2260 hours / 6605 Function Points = 20 minutes. The rate charged the client was $125 per hour. Consequently, each Function Point cost $42.

To accomplish a rough order of magnitude (ROM) estimate, each table's worth of "Business Information System" costs 82.5 * $42 or $3,465.

The immediate value of this statistic is that if there is an addition to the membership management system that requires and additional 14 tables, the estimate for that extension should be 14 * $3,465 = $48,510

It must be stated that the $42 per Function Point was for the actual building and testing of the membership management system. The estimate did **NOT** include Business Information System documentation or end-user documentation.

The experienced-based estimate for System and User Documentation is 60 hours per main function (e.g., Ballot, Committee, and Invoice). Hence 480 more hours. When this 480 additional hours are added to 2260, the result, 2740 elevates the time to create a Function Point to 2760 staff hours /6605 Function Points or 25 minutes. At the rate of $125 per hour, that’s $52.23 per Function Point.

8. Conclusions

The practical application of the points made in this paper include:

- Reliable and repeatable Function Point counts can be determined for specific classes of Business Information Systems.

- It is critical to standardize the Business Information System development methodology and Business Information System development environment to standardize the objects that exist and are able to be counted.

- It is bottom-line extraordinarily valuable to employ development environments such as Clarion (www.softvelocity.com) that generate the vast majority of any Business Information System, reduce Function Point cost, and eliminate the vast majority of coding bugs.

Without development environments such as Clarion, Function Points cost upwards to about $450 versus about $50.

9. References

The following references to Whitemarsh materials provide a more detailed exposition practical application of the significant content of this paper.

The following documents are available free from the Whitemarsh website:

Paper	URL
Information Systems Planning: Book, Course, and Presentation (short and long) – samples Knowledge Worker Framework: Book, Course, and Presentation (short and long) – samples	http://www.wiscorp.com/EnterpriseDatabase.htm
Database Architecture Classes: sample	http://www.wiscorp.com/DatabaseDesign.htm
Resource Life Cycle Analysis: Paper	http://www.wiscorp.com/MetabaseProducts.htm
Database Project Work Breakdown Structure – sample	http://www.wiscorp.com/DatabaseProjects.htm

The following documents are available for Whitemarsh Website Members. The URLs that follow provide descriptions of the pages. Members should log in and proceed to the appropriate page, e.g., Enterprise Database, find the book, paper, or course and perform the download.

Paper	URL
Data Management Program - Work Breakdown Structures	http://www.wiscorp.com/wwmembr/mbr_products_edb.html
Enterprise Database Overview Enterprise Database Principles Information Systems Planning (book, course, and papers)	http://www.wiscorp.com/wwmembr/mbr_products_edb.html

The following Whitemarsh books, which can be ordered from the Whitemarsh website contain material related to this short paper.

- Strategy for Successful Development of Business Information Systems
- Data Interoperability Community of Interest Handbook
- Enterprise Architectures