Functional Size Measurement

close the connection to SQL Server...

firstName = (string)reader["FirstName"];
lastName = (string)reader["LastName"];  

// Is the current Employee Jim Dandy?
if ((firstName == "Jim") && (lastName == "Dandy"))

the New Employee (Jim Dandy) found. If not, insert him

// New employee not added...
// Hire date and Birth date...

// Shows use of the FieldCleanup() and ConvertToSqlData
sqlServer.ExecuteNonQuery("INSERT INTO Employees (LastName,
How long can you wait for CMMI Compliance?

Manage your software development in guaranteed compliance with the CMMI® NOW!

processMax includes all the necessary policies, procedures, guidelines, criteria, templates, and forms in role-based, step-by-step instructions, ready for use—everything you need for CMMI compliance. Integrated with robust document management and workflow, processMax is the intranet web-based solution for effective and efficient management of your software development projects. processMax also includes a complete measurement and reporting capability. Measurement collection and report generation are fully automated—giving managers real-time, fact-based insight into all aspects of project progress. processMax is the proven solution—as verified by more than 50 successful independent appraisals.

processMax
THE PROCESS SOLUTION

www.pragmasystems.com

pragma Systems Corporation, 1810 Samuel Morse Reston, VA 20190
877.838.PMAX E-mail: info@pragmasystems.com
processMax is a registered trademark of pragma SYSTEMS Corporation.
CMMI is a registered mark of Carnegie Mellon University. Copyright © 2006 pragma SYSTEMS Corporation.
A Functional Size Unit (FSU) is to software acquisition and development what a square foot is to the building construction industry. An FSU is a standard unit of software size for measuring the functionality of software. It is analogous to a “gallon” of gasoline, a “cord” of wood, or a “meter” of length [1]. “Cord of wood” provides the best analogy because we can only measure by taking a bunch of irregular pieces of wood and filling a space that corresponds to our definition of ‘cord’. In functional sizing, we take different software functional entities (logical data groups and varying types of transactions) and assign weights to them in order to translate the functionality they represent into a quantity of FSUs. The fact that we follow a standard set of rules in the process makes the results consistent across projects and domains.

Use of a standard unit enables us to build a frame of reference. We know instantly that a 40,000 sq. ft building is not likely to be a residential dwelling, because our frame of reference contains knowledge of the size unit as well as the environment/domain in which the size unit is being applied. We also know that 2500 sq. ft. of residential floor space in New York City is likely to cost 10 times more than 2500 sq. ft in a small town in Nebraska. The size measurement alone tells us very little, but when we combine it with other project attributes, we can derive meaningful metrics for decision-making. This is the real significance of functional sizing.

In building construction, as well as other domains, the builder can accurately predict the cost of constructing a house, simply by knowing the desired floor space and the degree of customization (quality) desired. The builder, has created several metrics relating to his productivity (delivery rate) that allow predicting the schedule, the staff needed, and the cost, early and with minor expense and time. Such estimates serve both the builder and the acquirer, as is the case with software. Sizing can be done very early during the requirements phase, at minimal time and cost, and it is consistent and reasonably accurate for purposes of early estimating. We size at various points in the project whenever scope changes or requirements changes occur, and again at the end of the project. We save that information to our project repository along with the actual effort and cost data. Thus, over time, we build a rich database of project history and use it to make future estimates and predictions, and to manage risk on future projects.

While we do have the ‘standard unit’, we currently do not have such a universal frame of reference for software as our friends in the building trades have. We all know from visual experience how much a square foot is, but we cannot visualize what a functional size unit is, or to what exactly it maps. We know that it makes no sense to talk about a residential dwelling that is ten sq. ft., unless it is a dollhouse, but do we know if ten FSUs is a sensible measure of some piece of software? Most of us (project teams) do not know what our delivery rate is because it has never been considered in quantitative terms before. We like to think we (our organization or our project team) are efficient and productive but we typically cannot support our claims with any quantitative evidence.

continues on page 2
Also, when the term ‘productivity’ is used, we, (software practitioners) tend to think it applies to the individual rather than the project or organization, so we become apprehensive when someone tells us that a benefit of functional sizing is that it enables us to track our productivity (delivery rate).

Functional sizing helps us, at the project or organizational level, to build such a frame of reference for software. Sizing, by itself, is nothing but a number. It is not a panacea; it does not increase productivity; it does not increase quality; it does not reduce cost [2]. However, it does provide consistent objective quantitative data, which we can then use to make better decisions that will increase productivity, increase quality and reduce costs.

Functional Size Measurement (FSM) [1] has the following characteristics:

- Works for all types of software (scientific, business apps, web portals, embedded systems, etc.)
- Works for all types of projects (new development, enhancements, maintenance, etc.)
- Is language independent
- Is technology independent
- Is repeatable (consistent) – two analysts independently measuring the same software will arrive at the same size
- Produces statistically significant results
- Can be applied early in the development life cycle

The articles in this issue of the Software Tech News were selected to help you understand what functional sizing is really all about, to acquaint you with the various methods of functional sizing that are in use today, and to provide some insight into the real world application of FSM for development, maintenance and acquisition.

In the first article, author, David Garmus discusses the high-level principles of sizing and estimating using Function Points (FP), the functional sizing methodology supported and standardized by the International Function Point Users Group (IFPUG). He discusses how the sizing methodology addresses software complexity and asserts the need to have a project repository with more than size data in it.

In the second article, Sheila Dennis describes the actual function point counting process used to derive the seemingly magic functional size. This summary primarily demonstrates how the standard IFPUG counting rules come into play. Visit the IFPUG website (http://www.ifpug.org/) to learn more about Function Point Counting practices.

If your organization is just getting started with functional sizing, you might want to consider using the project repository maintained by the International Standards Benchmarking Group (ISBSG), which currently contains data on more than 4000 projects. In the third article, Peter Hill, CEO of ISBSG, talks about applying functional sizing early in the software life cycle. He asserts that different techniques are appropriate depending on when sizing is performed and for what purpose. He defines the methodologies on six levels ranging from broad approximations to detailed measures. Then he describes some practical sizing methods and, in so doing, explains how the ISBSG data can be used.

The information about FSM is vast. It is of global significance and it touches many software domains. This issue only scratches the surface of FSM knowledge, but it does demonstrate that FSM is not something you do occasionally. Nor is it very useful in a short-term scenario. Its value increases as it is applied to more and more projects, and as the organization builds its project repository.

This issue focuses on describing FSM primarily from the perspective of the International Function Point Users Group (IFPUG), which has maintained the function point counting practices since the late 1970s, but there are other functional sizing methodologies as well. The next issue of STN will focus on methods for sizing real time and embedded systems. COSMIC FFP, created by the Common Software Measurement International Consortium (COSMIC) in the late 1990s was specifically designed for this purpose. Yet, IFPUG asserts that it has also evolved to address the issues of sizing real time systems. “Which method is best?” is not necessarily the right question. Instead, you might ask, “Which method is right for you?” Read both issues before you decide.


About the Author

Ellen Walker, a DACS Analyst, is currently developing a series of publications on software “best practices” as part of the DACS Gold Practice Initiative. She has spent the past 20 years as a software developer in various roles spanning the entire software life cycle including project management of multiple business process re-engineering efforts within the DoD community. She is also experienced with assessment initiatives such as the Capability Maturity Model for Software (CMM-SW) and the quality management practices of the New York State Quality Award program. Ellen has an MS in Management Science (State University of New York (SUNY) at Binghamton), and bachelor degrees in both Computer Science (SUNY – Utica/Rome) and Mathematics (LeMoyne College).

Author Contact Information
Email: ellen.walker@itt.com
The DACS Gold Practice Initiative:

- Promotes effective selection/use of software acquisition & development practices
- Defines essential activities/benefits of each practice
- Considers the environment in which each practice is used
- Addresses the timeliness of practice benefits
- Recognizes interrelationships between practices that influence success or failure
- Contains quantitative and qualitative information
- Includes discussion forums, case studies and survey opportunities
- A continually evolving resource for the DoD, Government, Industry and Academia
- Free to use/free to join

Current Gold Practices:

- Architecture-First Approach
- Binary Quality Gates at the Inch-Pebble Level
- Commercial Specifications and Standards/Open Systems
- Ensure Interoperability
- Formal Inspections
- Integrated Product and Process Development
- Model-Based Testing
- Plan for Technology Insertion
- Statistical Process Control
- Track Earned Value

Learn More About the DACS Gold Practice Initiative:
http://www.goldpractices.com

Become a Registered Member!
http://www.goldpractices.com/registration/register_step1.php

Data & Analysis Center for Software
P.O. Box 1400
Rome, NY 13442-1400
http://iac.dtic.mil/dacs

By David Garmus, David Consulting Group

Introduction

Software practitioners are frequently challenged to provide early and accurate software project estimates. It speaks poorly of the software community that the issue of accurate estimating, early in the lifecycle, has not been adequately addressed and standardized. The CHAOS Report by The Standish Group’s study on software development projects revealed:

- 60% of projects were behind schedule
- 50% were over cost, and
- 45% of delivered projects were unusable.

At the heart of the estimating challenge are two issues: 1) the need to understand and express (as early as possible) the software problem domain and 2) the need to understand our capability to deliver the required software solution within a specified environment. Then, and only then, can we accurately predict the effort required to deliver the product.

The software problem domain can be defined simply as the scope of the required software. The problem domain must be accurately assessed for its size and complexity. To complicate the situation, experience tells us that at the point in time that we need an initial estimate (early in the systems lifecycle), we cannot presume to have all the necessary information at our disposal. Therefore, we must have a rigorous process that permits a further clarification of the problem domain.

Various risk factors relating to environment, technology, tools, methodologies, and people skills and motivation influence our capability to deliver.

An effective estimating model, as contained in Figure 1, considers three elements: size, complexity and risk factors, to determine an estimate.

Project Size

By far, the project sizing technique that delivers the greatest accuracy and flexibility is the International Function Point Users Group (IFPUG) Function Point methodology. Based on logical, user defined requirements, IFPUG Function Points permit the early sizing of the software problem domain. In addition, the IFPUG Function Point methodology presents the opportunity to size a user requirement regardless of the level of detail available.

An accurate Function Point size can be determined from the detailed information included in a thorough user requirements document or a functional specification. An adequate Function Point size can even be derived from the limited information available in an early proposal.

The IFPUG Function Point methodology is dependent upon identification of five elements: inputs, outputs, inquiries, internal stores of data and external references to data. During the early stages of development, these elements are exposed at a functional level (e.g., we know we will generate an output report, although we may not know the detailed characteristics of that report). The first ‘level’ of Function Point counting continues on page 5.
identifies these five elements. As more information becomes available regarding the characteristics of these elements; i.e., data fields, file types, etc., the more detailed the Function Point count. During the early phases of a count, it may be necessary to assume levels of complexity within the system (e.g., is our report going to be simple or complex). The value of using IFPUG Function Points is that it allows for this distinction, in fact requires it early in the process.

Alternative sizing methods, such as counting lines of code, are dependent upon information that is not available until later in the development life cycle. Other functional measurement methods require detailed knowledge about system processing that is not available early enough for accurate counting, e.g., states and transitions.

IFPUG Function Points accurately size the stated requirement. If the problem domain is not clearly or fully defined, the project will not be properly sized. When there are missing, brief or vague requirements, a simple interview process with the requesting user can be conducted to more fully define the requirements. Function Points can be utilized to identify stated inputs, outputs, inquiries, internal stores of data and external stores of data.

For an average size project, hours, not days, are required to complete the diagramming and sizing task.

Project Complexity
In addition to the project size, project complexity must be properly evaluated. To some extent, complexity levels are evaluated by the IFPUG Function Point fourteen General System Characteristics (GSCs):

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

Additional Project Complexity Affecting Delivery
Over and above the GSCs, the assessment of a project’s complexity must evaluate complex interfaces, data base structures and contained algorithms. These additional factors significantly impact the effort necessary to deliver a software project. The assessment of complexity might consider as an example the five varying levels of complexity shown in Table 1, which are addressed in detail in the book titled “Function Point Analysis; Measurement Practices for Successful Software Projects”, Addison-Wesley by David Garmus and David Herron, contains significantly more detail:

<table>
<thead>
<tr>
<th>Level 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple addition/subtraction</td>
</tr>
<tr>
<td>Simple logical algorithms</td>
</tr>
<tr>
<td>Simple data relationships</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Many calculations including multiplication/division in series</td>
</tr>
<tr>
<td>More complex nested algorithms</td>
</tr>
<tr>
<td>Multidimensional data relationships</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Significant number of calculations typically contained in payroll/rating/scheduling applications</td>
</tr>
<tr>
<td>Complex nested algorithms</td>
</tr>
<tr>
<td>Multidimensional and relational data relationships</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Differential equations typical; Fuzzy logic; Extremely complex data</td>
</tr>
<tr>
<td>Extremely complex logical and mathematical algorithms typically seen in military/telecommunications/real-time/automated process control/navigation systems</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Level 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-line, continuously available, critically timed event driven outputs occur simultaneously with inputs</td>
</tr>
<tr>
<td>Memory, timing and communication constraints</td>
</tr>
</tbody>
</table>

Table 1. Levels of Complexity

Capability to Deliver
The capability to deliver software is also based upon a variety of negative risk factors and positive influencing factors that influence a development organization’s ability to deliver software in a timely and economical fashion. These factors include such things as the software processes that will be used, the skill levels of the staff (including user personnel) who will be involved, the automation that will be utilized and the influences of the physical (e.g., development conditions) and business environment (e.g., competition and regulatory requirements). In fact, there are numerous factors that influence our ability to deliver software in a timely fashion with high quality. Some continues on page 6
Sizing and Estimating Projects  Continued from page 5.

Figure 2. Influencing Factors

Figure 3. Project Estimation Process
examples of influencing factors that must be evaluated in order to produce an accurate estimate are categorized in Figure 2.

The key to effectively utilizing these factors is centered on the development of an historical baseline of performance. An organization should develop profiles, which reflect rate of delivery for a project of a given size, complexity and influencing factors. In turn, this information can be used to predict and explore ‘what-if’ scenarios on future projects. Virtually all of the project estimation tools in the market place today follow the process indicated in Figure 3.

Industry Data
Companies have not typically invested the resources to develop internal rate of delivery performance baselines that can be used to derive estimating templates. Therefore, industry data baselines of performance delivery rates are of significant value. The industry data points allow organizations to use these generic delivery rates as a means to ‘ball park’ their estimates. As they continue to develop an experience base of their own, they can transition from the use of industry data to use of their own data.

The desire for industry data is so great that many companies are willing to accept publicly available industry data at face value. Of growing concern is the fact that many providers and publishers of industry data have collected information that has not been validated, is not current or is incomplete. To avoid any such pitfalls, the following criteria should be applied when obtaining industry data:

- For what industry and business area is the data representative?
- What is the mix of data; e.g., platform, language, application type?
- What is the time period represented by the data?
- How valid is the data?

In Summary
Accurate and early estimating require:

- Proper identification of the problem domain, including functional size and complexity.
- An assessment of the organization’s capacity to deliver based upon known risk factors.
- Use of industry data points as necessary to provide delivery rates or as a point of comparison.

As Robert Glass indicated in his book “Building Software Quality,” “...if there is one management danger zone to mark above all others, it is software estimation.”

Furthermore, an investment in skills training and risk profile development is critical. Project managers must be equipped with the necessary tools and techniques to estimate projects accurately. The return on that investment (ROI) is obvious to any organization that has misspent dollars because of inaccurate estimating.

About the Author
David Garmus is a Principal of The David Consulting Group, a CMMI® Approved Transition Partner and PSM Transition Organization that supports software development organizations in achieving software excellence with a metric-centered approach. David is past president of the International Function Point Users Group (IFPUG) and a member of the IFPUG Counting Practices Committee. He received a BS from the University of California at Los Angeles and an MBA from the Harvard University Graduate School of Business Administration.

Please refer to the website of The David Consulting Group at www.davidconsultinggroup.com to obtain other articles written by David Garmus on software sizing and project management and measurement.

Author Contact Information
David Garmus
PH: 904-269-0211
Email: dg@davidconsultinggroup.com
International Function Point Users’ Group
IFPUG is a non-profit, member-governed organization whose mission is to be a recognized leader in promoting and encouraging the effective management of application software development and maintenance activities through the use of Function Point Analysis as its standard methodology for software sizing. In support of this, IFPUG maintains the Function Point Counting Practices Manual and also provides a forum for networking and information exchange that promotes and encourages the use of software product and process metrics. Visit the IFPUG website at:
http://www.ifpug.org

Related ISO/IEC Standards
The link below will take you directly to the ISO search screen. Enter “functional size measurement” in the keyword field to get a list of relevant ISO standards.
Introduction to Function Points

By Sheila P. Dennis and David Garmus, David Consulting Group

IBM first introduced the Function Point (FP) metric in 1978.[1] Function Point counting has evolved into the most flexible standard of software sizing in the information technology (IT) domain today.

There are several characteristics that account for the flexibility which drives the popularity and usage of the function point methodology, but the foremost appeal is the ability to measure the size of any software deliverable in logical, user-oriented terms. All sizing is based upon identifying, evaluating and weighting functional entities (inputs, outputs, inquiries and data usage) in an application or project from a user perspective. The term user is defined to be any person and/or thing that interacts with the software. Clearly, the user could even be another system or even a machine. By applying this approach, function points can be universally applied regardless of platform, environment, language, or other technical considerations.

Using Function Points

The IT industry, including the Department of Defense, has successfully used function point analysis to size a wide spectrum of applications and projects, including general business, complex financial and accounting, logistics and communications systems. It has been proven on a variety of development platforms and environments (e.g. mainframe, client-server, web, stand-alone PC, data warehouse); for a variety of development types (real-time, batch, interactive or control systems); for internal or external development efforts (e.g. on-shore, off-shore, contractor-based); and for vendor and/or COTS package integrations.

As projects are completed and software deliverables are produced, function point sizing, together with a collection of other meaningful measures, can be used in a variety of initiatives.

• Balance Scorecard. Using function points as the basis for size normalizes metrics across platforms and projects. Measures based upon function point sizing (e.g. delivery rate (hours/function point), defect rate (defects/1000 function points)) are currently being used as the cornerstone at select DOD installations supporting the enterprise IT Balance Scorecard.

• Benchmarking. Measuring DOD’s IT performance against competing outsourcing environments, as well as other governmental organizations, has become increasingly important in identifying opportunities for improvement in time to delivery, cost reduction, and customer satisfaction. Cost per function point delivered and function points supported/developed per Full Time Equivalent are only two of many metrics used in current benchmark initiatives.

• Outsourcing Service Level Agreements. Commercial and governmental uses of delivery rates by platform and defect density are effective in providing a contractual basis for performance standards for outsourcing requirements when function points are used as the common size measure.

Function Point Counting Process

The function point counting practices are governed by the International Function Point Users Group (IFPUG), a not-for-profit organization, consisting of IT measurement industry leaders and practitioners from over 30 countries. For more than 25 years, IFPUG has maintained the guidelines and rules for counting practices through committees whose members have extensive software development and measurement expertise. Each new edition, or version, of the rules (currently version 4.2) is verified by the IFPUG body through practical application and statistical methods to ensure consistency, usability and reliability. In this section, we will provide an overview of the counting process using simple counting examples. The counting process included in this article is not all-inclusive and must be complemented with the rules as defined in the IFPUG Counting Practices Manual[2].

The function point method evaluates the software deliverable of a project and measures its size based on well-defined functional characteristics of a software system[3]. Therefore, one of the first steps in counting is to identify the functional processes of a project and categorize them into function point entities. After the identification, a complexity level (Low, Average, High) is evaluated for each of the entities, and then assigned a weight (3 - 15 points) based upon complexity. For simplicity, we will use examples based upon a Human Resources (HR) system.

Identifying and Classifying. The two major functional characteristics that are considered in function points are data types (files, tables, records) and activity-based transaction types (inputs, outputs, queries).

Data types are user defined and recognized logical data groups of data, usually physically stored as files or tables. We classify the data into two separate categories, internal and external.

• Data manufactured and stored within the system are internal logical files (ILFs). For HR, employee data would be an ILF.
Function Points  Continued from page 8.

• Data maintained within a different system but necessary to satisfy a particular process requirement are called external interface files (EIFs). For HR, IRS tax tables could be a potential EIF.

Transactional types are elementary processes that control, maintain or display data. We classify transactions according to whether they relate to data entering the system, or leaving the system.

• Data entering a system are called external inputs (EIs). For HR, examples of EIs could be (1) an incoming feed from another system or (2) adding an employee through screen entry.

• Data leaving the system are classified as external outputs (EOs) or external inquiries (EQs). For HR, an online display of employee data would be a typical EQ. Reports or feeds to other systems are also EOs or EQs.

Evaluating Complexity. After the logical entities for a project are classified into the function point entities (ILFs, EIFs, EIs, EOs, and EQs), a complexity level of low, average or high is assigned using IFPUG derived complexity matrices (Figure 1). The matrices are dependent upon the components of data types and transactions.

• Record Element Types (RETs) are mandatory or optional sub-groups of data.

• Data Element Types (DETs) are non-repeated fields or attributes.

• File Types Referenced (FTRs) are the internal or external data types (ILFs or EIFs) that are used and/or maintained by the transaction.

The number of different Record Element Types, and unique Data Element Types used and/or maintained, determine the complexity of data types. However, the complexity of transaction types are determined by the number of data types referenced (ILFs and EIFs) and unique Data Element Types. In general, the higher use of components results in a higher complexity.

• For the HR system, assume that “Employee Data” has two groups of data, employees and dependents, with a total of 35 unique, non-repeated, user recognizable fields. Then “Employee Data” would be rated Average.

• If the transaction to add an employee (EI) had at least 16 unique fields to enter on the screen in order to update “Employee Data”, and “Employee Data” was the only type of internal or external data to be used in this process, then the complexity would be based on 1 FTR with 16+ DETs, or Average.

• An HR report of employee tax data (EO) having 16 or more unique data fields, and using both the “Employee Data” and the external data from the IRS Tax tables, would be of High complexity.

### Table 1. Function Point Counting Weights [2]

<table>
<thead>
<tr>
<th>FP Type</th>
<th>Low</th>
<th>Avg</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>EI</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>EO</td>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>EQ</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>ILF</td>
<td>7</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>EIF</td>
<td>5</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>

Assigning Weight. Once the complexity is identified, values for each entity are assigned using the IFPUG standard weights shown in Table 1.

In the HR application, the previously identified entities would be weighted as follows:

• Employee Data (ILF) - Avg - 10 function points
• Add an Employee (EI) - Avg - 4 function points
• Employee Report (EO) - High - 7 function points

Calculating a Project Count. Assuming that the three functional processes listed above represented the requirements for a project, then the total unadjusted count for the project would be the sum of all the entities, or 21 function points.
At this milestone of the counting process, there is a Value Adjustment Factor (VAF) factor that is used to adjust the project count. The VAF, with a range of .65 to 1.35, is derived from evaluation of fourteen (14) General System Characteristics (GSCs) based upon the technical characteristics of the application. Complex processing, distributed processing, online data entry, security influence and transaction rates are a few of the aspects considered in the GSCs. In the HR project, if the GSC for HR was 1.1, then the project count would be adjusted by multiplying the unadjusted count (21 function points) by the GSC (1.1) for an adjusted total project count of 23 function points.

The final function point calculation yields a single number that represents the total amount of functionality being delivered. Once completed, the function point size of an application or a new development project can be communicated in a variety of ways. As a stand-alone value, the function point size of a system tells us how large the overall software deliverable will be. When the function point value is segmented into a more detailed display, it can communicate to end users the functional value of specific components of the system. Finally, more mature software measurement organizations can use function points to predict outcomes and monitor program progress.

Summary

Function points is the most effective and flexible way of normalizing critical measures in the successful management and monitoring of both internal measurement initiatives and outsourcing arrangements. Use of function points as the standard size component of cost and quality measures can satisfy both the IT organization's need to monitor the outsourcing contract and the user's need to ensure the value of the deliverable. In addition, the use of function points provides the opportunity to make comparisons to industry performance levels.

References

Software Early Lifecycle – Functional Sizing

By Peter Hill, CEO, International Software Standards Benchmarking Group (ISBSG)

Introduction

Sizing a software project early in its lifecycle is not an easy task, but we often have to find an answer to the question: “How big is it?” Using functional units, there are a number of ways of establishing a rough size early in the life cycle of a project.

Functional size measurement is now a widely accepted form of software sizing in industry, particularly in those companies that have reached the higher levels on the CMMI scale or certification to international standards. There are a number of functional size measurement (FSM) options available. The two FSM approaches that are used internationally are IFPUG and COSMIC FFP. Other approaches that conform to the ISO standard are primarily country based: NEMSA (Netherlands), FiSMA (Finland) and MARK II (UK). The most commonly used FSM method to date, which is also the longest established, is the IFPUG (International Function Point User Group) method.

Although simple in concept, Functional Size Measurement is not a trivial task.

However, there are a large number of techniques available to provide estimates of the functional size of projects. In this article, we will explain the different levels of functional sizing and then look at some simple but effective ways of roughly determining the Functional Size of a project without doing a detailed function point count. As well as being used for software size estimation early in the life of a project, these techniques can be used as an introduction for those practitioners who do not currently use a functional size measure.

A NOTE OF CAUTION: It is important to understand that the software size value alone, no matter how exact, does not help in estimating and controlling changes to a development project. It is useful in estimating effort, duration etc, but only together with other project characteristics and information. For change management and all other communication between the customer and supplier of the software, the establishment of, and agreement on, the baseline of the software scope is essential to the successful management of the project.

Sizing Levels: From Estimation to Detailed Functional Size Measurement

To get started on functional size measurement it helps to understand that there are a number of levels (See Table 1) of sizing which can be chosen depending upon the amount of information that is available about the project.

The advantages of estimating software size are countered to some extent by the possible lack of precision of the results. It is important to distinguish each measure as either “exact measure” (i.e. count) or “approximation” (i.e. estimate).

Table 1. Accuracy Levels for Software Sizing

<table>
<thead>
<tr>
<th>Level</th>
<th>Measure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1</td>
<td>Detailed Linked and Labelled Measure</td>
<td>This is the highest level of accuracy for the determination of functional size; it is often obtained either from solid, complete requirements or at the end of a completed project where every functional requirement implemented in the project is fully known.</td>
</tr>
<tr>
<td>Level 2</td>
<td>Detailed Linked Measure</td>
<td>This level of accuracy is normally achieved once formal requirements have been documented and reviewed by users and development teams. There are no ‘to-be-determined’ requirements and the requirements phase is complete. At this level of accuracy, the functional size is no longer an approximation, but rather a formal project “count”.</td>
</tr>
<tr>
<td>Level 3</td>
<td>Detailed Measure</td>
<td>Specific details are known about the functional requirements of the project.</td>
</tr>
<tr>
<td>Level 4</td>
<td>Default Complexity Measure</td>
<td>The rough functional size estimation techniques for this level take each identified transactional/file function and assign the average values for the component type specific to the functional size measurement method on which it is based.</td>
</tr>
<tr>
<td>Level 5</td>
<td>Rough Measure</td>
<td>At this level, more project information is known beyond the project characteristics (e.g. number of reports, number of 3rd normal form tables, etc.). The rough functional size estimation techniques for this level of accuracy rely on using average values and approximations for functions, (e.g. project documents are scanned to determine how many functions exist (not by transactional types) and an average value assigned to each.</td>
</tr>
<tr>
<td>Level 6</td>
<td>Size Approximation</td>
<td>Functional size is approximated without identifying exact functions. Such rough approximations are based exclusively on project characteristics, which have some correlation to size (e.g. number of reports, number of 3rd normal form tables, etc.). The size can be very roughly estimated based on the answers to a few questions about the project’s known characteristics.</td>
</tr>
</tbody>
</table>
A measurement can be conducted to a number of “accuracy levels”, based on:

- the purpose of the measurement and desired accuracy of the result
- the quality of project or application documentation available
- the time in which the measurement must be completed

Each size estimation technique can be classified based on the following characteristics (levels are listed from 1 to 6, reflecting decreasing levels of accuracy) listed in Table 1. During a project, it is likely that you will start with a Level 6 technique and move towards Level 1 as the project characteristics become better defined.

It is important to choose an estimation technique based on the documentation and time available plus the measurement purpose.

Table 2 lists the basic attributes of each of the sizing levels, to help you to choose the one most suited to your need.

As the project progresses the size estimate should be validated and refined (eventually moving from low-accuracy to high-accuracy techniques).

### Some Practical Software Size Estimation Methods

There are a large number of software functional size estimation techniques available for levels 4 to 6. Three practical and well-documented techniques are described here. These derived methods use some sort of algorithms.

Table 2. Basic Attributes of Sizing Levels

<table>
<thead>
<tr>
<th>Lev.</th>
<th>FSM result</th>
<th>Best suited for</th>
<th>Issues</th>
<th>Pre-requisites</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Most detailed&lt;br&gt; Easily auditable&lt;br&gt; Accurate (+/-5%)&lt;br&gt; Very well documented&lt;br&gt; Easily maintained</td>
<td>Benchmarking projects&lt;br&gt; Detailed estimates&lt;br&gt; Project tracking&lt;br&gt; Detailed baseline model&lt;br&gt; Metrics reporting for strategic level</td>
<td>Very time intensive&lt;br&gt; Requires very skilled counters&lt;br&gt; Expensive for large systems</td>
<td>High quality documentation&lt;br&gt; Data model&lt;br&gt; Full access to system experts</td>
</tr>
<tr>
<td>2</td>
<td>More detailed&lt;br&gt; Easily auditable&lt;br&gt; Accurate (+/-5%)&lt;br&gt; Very well documented&lt;br&gt; Easily maintained</td>
<td>Benchmarking projects&lt;br&gt; Detailed estimates&lt;br&gt; Project tracking&lt;br&gt; Detailed baseline model</td>
<td>Time intensive&lt;br&gt; Expensive for large systems</td>
<td>Good/high quality documentation&lt;br&gt; Data model&lt;br&gt; Full access to system experts</td>
</tr>
<tr>
<td>3</td>
<td>Detailed&lt;br&gt; Auditable&lt;br&gt; Accurate (+/-10%)&lt;br&gt; Well documented&lt;br&gt; Very maintainable</td>
<td>Benchmarking projects&lt;br&gt; Detailed estimates&lt;br&gt; Baseline application measurement for portfolio sizing&lt;br&gt; Detailed baseline model</td>
<td>Time intensive&lt;br&gt; Reasonably cost effective for large systems</td>
<td>Good quality documentation&lt;br&gt; Data model (if available)&lt;br&gt; Access to system experts</td>
</tr>
<tr>
<td>4</td>
<td>Less detailed&lt;br&gt; Auditable&lt;br&gt; Reasonably accurate (+/- 15%)&lt;br&gt; Documented&lt;br&gt; Maintainable</td>
<td>Portfolio baseline assessment&lt;br&gt; Benchmarking development or support ratios&lt;br&gt; Quality metrics&lt;br&gt; High level estimates&lt;br&gt; Baseline model</td>
<td>Efficient&lt;br&gt; Cost effective for large systems</td>
<td>Average quality documentation&lt;br&gt; Data model (if possible)&lt;br&gt; Access to system experts</td>
</tr>
<tr>
<td>5</td>
<td>Low detail&lt;br&gt; Less accurate (+/-20%)&lt;br&gt; Documented (issues and assumptions)&lt;br&gt; ‘Skeleton’ (base for more refined measurement)</td>
<td>Portfolio baseline assessment&lt;br&gt; Benchmarking support ratios&lt;br&gt; Baseline model</td>
<td>Very efficient&lt;br&gt; Cost effective for large systems with little enhancement</td>
<td>Summarised system documentation&lt;br&gt; Access to system experts (for the duration of measurement)</td>
</tr>
<tr>
<td>6</td>
<td>Very little detail – size results only&lt;br&gt; Accuracy historically has been demonstrated to be within (+/- 30%)&lt;br&gt; Not documented&lt;br&gt; Not maintainable</td>
<td>Portfolio baseline assessment&lt;br&gt; Software asset valuation&lt;br&gt; Project scoping&lt;br&gt; Estimating count durations&lt;br&gt; Benchmarking support ratios</td>
<td>Very efficient&lt;br&gt; Very cost effective for large systems with very little enhancement</td>
<td>Accurate completion of a questionnaire&lt;br&gt; Access to system experts (short interview)</td>
</tr>
</tbody>
</table>
1. Early Estimation of Functional Size
Using the ISBSG Data

In the very early phases of a software development project it is not practical or even possible to know in detail all of the items that make up all of the function point components. However, it is often possible to detail one of the components with a fair degree of certainty - such as the Internal Logical Files or External Inputs.

It is likely that the functional component that you will have the most knowledge of is the Internal Logical Files. These closely resemble a count of the entities in a logical data model, modelled to second normal form. Even in the early phase of a project, (e.g. the Feasibility Study), it is often possible and very useful, to work with the client to develop a logical data model of the proposed application, as a means of assisting the client to define the requirements. When completed, this model can then be used to provide the starting point to establish the number of Internal Logical Files as part of the functional sizing.

An IFPUG function point count identifies all occurrences of the following five base functional component types, (BFCTypes):

• Internal Logical Files (ILF) – data maintained by processes within the software
• External Interface Files (EIF) – data referenced by processes within the software
• External Inputs (EI) – processes which enter data to be stored internally within the software
• External Outputs (EO) - processes which extract derived data to be provided to the user
• External Queries (EQ) - processes which retrieve stored data to be provided to the user

Since the early days of the ISBSG project collection and analysis, it has been observed that the relationships between these five component types have remained relatively constant; i.e. each component type contributes a consistent percentage of the total function points in the overall total count for the application.

Investigation into the rationale for the relationships shows that there are good reasons why this consistency exists. It would be expected that for any complete ‘application’ which operates as a software ‘system’ the data that is entered, will be processed and stored for later retrieval. It therefore follows that we would expect a strong relationship between Input functions (data entered) and the Logical Files (internal data storage); and the Output and Query functions which retrieve data stored from the internal stores and the external stores, Interface Files.

The knowledge of these relationships has offered a number of valuable uses for the practitioner, one of which is the ability to predict the functional size of a new development project, or an implemented application, when the number of Internal Logical Files or External Inputs is known with any degree of certainty.

However, the relationships have only been found to be relevant to software that operates as a ‘self-contained’ system; i.e. a cohesive set of functionality that is loosely coupled with other applications.

For these reasons, it is not advisable to use the following early prediction sizing techniques to predict the size of any enhancement project that has a mix of added, changed and deleted functionality scattered over several functional areas within an application. This size estimation method is rated as a sizing Level 5.

Figure 1 shows the relationships between the five components of the IFPUG functional size method from project data in the ISBSG repository. These relationships can be used to estimate the functional size of a project.

The percentage values depict the relative contribution of each type for projects sized by IFPUG Version 4 functional sizing method and rated “A” or “B”.

EXAMPLE: If the customer has identified a need and, on developing a logical data model to reflect that need, there are found to be 40 logical tables, it may be reasonably assumed that these relate to approximately 40 Internal Logical Files.

Analysis of the ISBSG Repository also shows that most Internal Logical Files in applications are rated as being ‘low’ to continues on page 14
‘medium’ in complexity. The mean score attributed to them across all projects is 8.6 function points.

Based upon the above, one can assume that the total score for the Internal Logical Files component of the function point count will be:

$$40 \times 8.6 \text{ (mean score for Internal Logical Files)} = 344 \text{ FPs}$$

Figure 1 shows that the Internal Logical Files component of the function point count is typically around 21.7%. On this basis, the total functional size of the required application is predicted to be around:

$$344 \text{ FPs} \times 100/21.7 = 1,585 \text{ FPs}$$

This would be best relayed to the customer as 1,590 FPs plus or minus 400 FPs*.

If the development project is to replace an existing application or delivers similar user functionality to another application then you may use some of the measures of components from these other applications as a guide.

**EXAMPLE:** The number of unique reports and extract files output from the existing application, which the project is to replace, can be assumed equivalent to the External Outputs component in the new project. Analysis of the ISBSG Repository shows that most External Outputs are rated as being ‘medium’ in complexity. The mean score attributed to them across all projects in the repository is 5.4 function points. If the existing application has 47 different reports and 3 different extract files then the total number of External Outputs can be assumed to be 50. (Note: ensure that you exclude any obsolete, unused reports from your calculations).

Based upon the above, one can assume that the total score for the External Outputs component of the functional size measure will be:

$$50 \times 5.4 \text{ (mean score for External Outputs)} = 270 \text{ FPs}$$

Figure 1 shows that the External Outputs component of the functional size measure is typically around 25%. On this basis, the total functional size for the required application is predicted to be around:

$$270 \text{ FPs} \times 100/25 = 1,080 \text{ FPs}$$

This would be best relayed to the customer as 1,100 FPs plus or minus 270 FPs*.

Note: The techniques discussed above are only valid if your application or development project is loosely coupled from other applications and fits the profile of projects currently in the ISBSG Repository. Early research indicates that the above relationships may not hold for the domains of real-time, control, scientific or embedded software.

### 2. KISS Quick Software Size Estimation Technique

KISS Quick is the functional size estimation approach related to the FiSMA functional size measurement method. KISS, (“Keep It Simple”), Quick is linked to IFPUG FP sizing by means of using average multipliers for the functional components. The first step in approximating functional size using KISS Quick is to complete a standard questionnaire (see Table 3) with 28 questions concerning the numbers of occurrences of functional components. The multipliers, which are then used to convert the number of functional components to actual FiSMA sizing units, are based on the specific measurement method (IFPUG only shown; multipliers equal zero where the method does not count that functionality type ). For typical MIS applications, KISS Quick provides typically an accuracy level 4 result for functional size.

### 3. EARLY & QUICK Software Sizing Technique

The Early & Quick Technique combines different approaches in order to provide better size estimates. It makes use of both analogical and analytical classification of functions and different levels of detail for different branches of the system, (aggregations and multilevel approach). The overall uncertainty level in the resulting functional size estimate, expressed as a range of minimum, likely, and maximum values, is the weighted sum of the individual components’ uncertainty levels. This technique provides a table of statistically validated values, derived from ISBSG and other sources. Due to its multi-level/mixed approach, the sizing level for E&Q depends on how many details the measurer has/can explore:

- Level 5 for higher hierarchical components (Macro processes, General Processes, Multiple & generic/unconstrained processes, over 200 levels).

*WARNING:* Whether the above quick predictive technique is used or a detailed function point count is performed to establish size to be used for an early cost indicator for the project, a contingency of 20% to 30% should be added to allow for functionality not apparent early in the life cycle. Historical data indicates that this scope creep typically occurs because of additional functionality being identified as user requirements evolve in subsequent development phases.
Functional Sizing  Continued from page 14.

Table 3. KISS Quick Questionaire

<table>
<thead>
<tr>
<th>Navigation &amp; query functions (no update)</th>
<th>IFPUG Multipliers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 # starting icons?</td>
<td>0</td>
</tr>
<tr>
<td>2 # log-in and log-out windows?</td>
<td>3</td>
</tr>
<tr>
<td>3 # different menus?</td>
<td>0</td>
</tr>
<tr>
<td>4 # parameter selection lists (drop-down lists)?</td>
<td>3</td>
</tr>
<tr>
<td>5 # inquiry windows (db retrieving, on screen)?</td>
<td>4</td>
</tr>
<tr>
<td>6 # browsing list windows (occurrences of same type of data)?</td>
<td>4</td>
</tr>
<tr>
<td>7 # screens for starting report generation?</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User input functions (update)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8 # 3-functional (create, update and delete) user input windows?</td>
<td>18</td>
</tr>
<tr>
<td>9 # 2-functional (create and/or update and/or delete) windows?</td>
<td>12</td>
</tr>
<tr>
<td>10 # 1-functional (create or update or delete) windows?</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-interactive user output functions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11 # output forms (fixed layout)?</td>
<td>5</td>
</tr>
<tr>
<td>12 # reports?</td>
<td>7</td>
</tr>
<tr>
<td>13 # text messages or e-mails written and sent?</td>
<td>4</td>
</tr>
<tr>
<td>14 # monitor screen outputs?</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interface services from other applications</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>15 # messages received from other applications?</td>
<td>5</td>
</tr>
<tr>
<td>16 # batch records received from other applications?</td>
<td>5</td>
</tr>
<tr>
<td>17 # signals received from a device?</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Interface services to other applications</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>18 # messages to other applications?</td>
<td>5</td>
</tr>
<tr>
<td>19 # batch records sent to another application?</td>
<td>5</td>
</tr>
<tr>
<td>20 # signals sent to a device?</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Persistent data storage functions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>21 # entities or classes (OO)?</td>
<td>7</td>
</tr>
<tr>
<td>22 # other logical record types?</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Independent algorithmic functions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>23 # independent calculation routines?</td>
<td>0</td>
</tr>
<tr>
<td>24 # independent simulation routines?</td>
<td>0</td>
</tr>
<tr>
<td>25 # independent formatting routines?</td>
<td>0</td>
</tr>
<tr>
<td>26 # independent database cleaning routines?</td>
<td>0</td>
</tr>
<tr>
<td>27 # independent security routines?</td>
<td>0</td>
</tr>
<tr>
<td>28 # other independent algorithmic routines?</td>
<td>0</td>
</tr>
</tbody>
</table>

- Logical Data Groups
- Level 4 for lower hierarchical components (Typical Processes, Base and Implicit Functional Processes; Internal and External Logical Data Groups with generic complexity)
- Level 3 (or Level 2, if the measurer even documents the link between data and processes) for portions where the specified Logical Data Groups

Low/Average/ High complexity is determined.

The starting point of this technique is the product breakdown structure of the system being studied, whose basic elements are the following software objects:

- logical data groups (files)

continues on page 16
- elementary (functional) processes

Further aggregations are provided:
- logical data groups (files) can be grouped in multiple data groups
- elementary (functional) processes can be grouped in small, medium or large “typical” and “general” software processes
- general processes can be grouped in small, medium or large “macro” software processes

Table 4 shows the descriptions for all the software objects and their aggregates.

Each “software object” is assigned a size range based on statistical/analytical tables (depending on the referring measurement method). To estimate the functional size of a software system or project using this technique, a list of processes and data groups is all that is required, even comprising non-homogeneous levels of detail. Early & Quick functional size estimates may be denoted as “detailed”, “intermediate” or “summary”, depending on the detail level used for the early classification of functionalities.

The following provides Early & Quick hints, levels and ranges for IFPUG measures:

**Early & Quick for IFPUG Function Point Size (E&QFP 2.0)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Brief definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDG</td>
<td>Logical Data Group</td>
<td>A group of logical attributes – a conceptual entity that is functionally significant as a whole for the user. An Internal Logical File or External Interface File (ILF, EIF - IFPUG), or permanent Objects of Interest (OOI - COSMIC).</td>
</tr>
<tr>
<td>MDG</td>
<td>Multiple Data Group</td>
<td>A set of two or more Logical Data Groups. Its size is evaluated based on the (estimated) amount of included Logical Data Groups.</td>
</tr>
<tr>
<td>BFP</td>
<td>Base Functional Process</td>
<td>The smallest software process with autonomy and significant characteristics, allowing the user to achieve a unitary business objective. It corresponds to an External Input, External Output, or External Query (IFPUG), i.e. a functional process (MkII &amp; COSMIC).</td>
</tr>
<tr>
<td>IFP</td>
<td>Implicit Functional Process</td>
<td>A functional process that is typically not documented (e.g. implied inquiries as list-boxes). It can be estimated either directly, or by means of other software functions (e.g. associated to referred logical files).</td>
</tr>
<tr>
<td>TP</td>
<td>Typical Process</td>
<td>A particular case of a General Process: a set of most frequent transactions on a Logical Data Group (or a small set of LDGs). Usually denoted as “Management of [LDG/OOI]”. It can be of 3 “flavours”: CRUD (Create, Retrieve, Update and Delete), CRUDL (CRUD + elementary List), CRUDL + Report (totalizations or other derived data).</td>
</tr>
<tr>
<td>GP</td>
<td>General Process</td>
<td>A set of two or more average FPs. It can be likened to an operational sub-system, which provides an organised whole response to a specific application goal. Its size is evaluated based on the (estimated) quantity of included FPs.</td>
</tr>
<tr>
<td>MP</td>
<td>Macro Process</td>
<td>A set of two or more average General Processes. It can be likened to a relevant sub-system, or even a bounded application, of an overall Information System. Its size is evaluated based on the (estimated) quantity of included General Processes.</td>
</tr>
</tbody>
</table>

IFPUG Data Functions and their E&Q Equivalent Logical Data Groups are identified for Internal Logical Files and External Interface Files. These groups are classified on a multiple scale of complexity:
- Low, Average, High, or generic (unspecified type),
- Low, Average, High, or generic (specified type: Internal or External),
- Low Multiplicity or High Multiplicity

The first levels correspond exactly to those used by IFPUG, while multiplicity is used for particularly complex macro-files, grouping several distinct logical files.

IFPUG Transactional Functions and Their E&Q Equivalent Base Functional Processes (BFPs) can be identified for External Inputs, External Outputs and External Queries, while Typical Processes, General Processes and Macro Processes are higher aggregations of Base Functional Processes. Accordingly:
- BFPs can be classified as Input (BFPI), Output (BFPO) or Query (BFPO),
- Typically, General and Macro Processes are classified as small, average or large, depending on the estimated amount of their components.

Due to the assigned minimum-maximum ranges, there is no need to evaluate the functional complexity of such processes, thus reducing the measurement effort.

continues on page 17
Implicit functional processes (e.g. list boxes) can be treated in two ways:

- Direct estimation (one simple query per estimated instance)
- Derived estimation via an average correlation from the quantity of data groups (each query is typically populated by a logical data group).

Ranges and Numerical Assignments

Each Early & Quick FP element is assigned three estimated values (Unadjusted FP or UFP), i.e. minimum, likely and maximum UFP. Aggregated elements such as multiple data groups and general and macro processes are classified according to the ranges of their (estimated) subordinate components. Table 5 shows components ranges and numerical assignments for E&QFP 2.0.

### Table 5. E&QFP 2.0 Components Ranges and Numerical Assignments

<table>
<thead>
<tr>
<th>Type</th>
<th>Level</th>
<th>Ranges / IFPUG Equivalent</th>
<th>Min. UFP</th>
<th>Most likely UFP</th>
<th>Max. UFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical Data Group</td>
<td>Low</td>
<td>Low complexity ILF/EIF</td>
<td>5.0</td>
<td>6.5</td>
<td>7.0</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>Medium complexity ILF/EIF</td>
<td>7.0</td>
<td>9.2</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>High complexity ILF/EIF</td>
<td>10.0</td>
<td>13.7</td>
<td>15.0</td>
</tr>
<tr>
<td></td>
<td>Generic</td>
<td>Generic complexity ILF/EIF</td>
<td>5.0</td>
<td>6.9</td>
<td>15.0</td>
</tr>
<tr>
<td>Internal Logical Data Group</td>
<td>Low</td>
<td>Low complexity ILF</td>
<td>7.0</td>
<td>7.0</td>
<td>7.0</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>Medium complexity ILF</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>High complexity ILF</td>
<td>15.0</td>
<td>15.0</td>
<td>15.0</td>
</tr>
<tr>
<td></td>
<td>Generic</td>
<td>Generic complexity ILF</td>
<td>7.0</td>
<td>7.4</td>
<td>15.0</td>
</tr>
<tr>
<td>External Logical Data Group</td>
<td>Simple</td>
<td>Low complexity EIF</td>
<td>5.0</td>
<td>5.0</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td>Average</td>
<td>Medium complexity EIF</td>
<td>7.0</td>
<td>7.0</td>
<td>7.0</td>
</tr>
<tr>
<td></td>
<td>Complex</td>
<td>High complexity EIF</td>
<td>10.0</td>
<td>10.0</td>
<td>10.0</td>
</tr>
<tr>
<td></td>
<td>Generic</td>
<td>Generic complexity EIF</td>
<td>5.0</td>
<td>5.5</td>
<td>10.0</td>
</tr>
<tr>
<td>Multiple Logical Data Groups</td>
<td>Low</td>
<td>2–4 generic LDG</td>
<td>10.0</td>
<td>21.0</td>
<td>30.0</td>
</tr>
<tr>
<td></td>
<td>High</td>
<td>5–8 generic LDG</td>
<td>27.0</td>
<td>45.0</td>
<td>60.0</td>
</tr>
<tr>
<td>Base Functional Process</td>
<td>Input</td>
<td>EI</td>
<td>3.0</td>
<td>4.3</td>
<td>6.0</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>EO</td>
<td>4.0</td>
<td>5.4</td>
<td>7.0</td>
</tr>
<tr>
<td></td>
<td>Query</td>
<td>EQ</td>
<td>3.0</td>
<td>3.8</td>
<td>6.0</td>
</tr>
<tr>
<td>Implicit Process</td>
<td>Direct</td>
<td>Low/Medium EQ</td>
<td>3.0</td>
<td>3.3</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td>Derived</td>
<td>One avg. per each LDG (*)</td>
<td>2.7</td>
<td>3.0</td>
<td>3.3</td>
</tr>
<tr>
<td>Typical Process</td>
<td>Small</td>
<td>CRUD (Create, Retrieve, Update and Delete) (Low/Medium complexities); CRUD + List (Low complexities)</td>
<td>12.0</td>
<td>14.0</td>
<td>16.7</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>CRUD (Medium/High complexities); CRUD + List (Medium complexities); CRUD + List + Report (Low cplx's)</td>
<td>15.8</td>
<td>17.8</td>
<td>20.5</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>CRUD (High complexities); CRUD + List (Medium/High cplx's); CRUD + List + Report (Medium cplx's)</td>
<td>21.2</td>
<td>23.2</td>
<td>25.9</td>
</tr>
<tr>
<td>General Process</td>
<td>Small</td>
<td>6–10 generic FP’s</td>
<td>22.0</td>
<td>37.0</td>
<td>57.0</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>10–15 generic FP’s</td>
<td>37.0</td>
<td>57.0</td>
<td>81.0</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>15-20 generic FP’s</td>
<td>57.0</td>
<td>81.0</td>
<td>110.0</td>
</tr>
<tr>
<td>Macro Process</td>
<td>Small</td>
<td>2–4 generic GP’s</td>
<td>75.0</td>
<td>170.0</td>
<td>325.0</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>4–6 generic GP’s</td>
<td>150.0</td>
<td>285.0</td>
<td>485.0</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>6–10 generic GP’s</td>
<td>220.0</td>
<td>455.0</td>
<td>810.0</td>
</tr>
</tbody>
</table>

(*) Multiply by factor 3 or 6.5, respectively, in case of Low or High Multiplicity LDG’s.
**Early & Quick Summary**

The general E&Q technique fully complies with the concepts, definitions and the structure of any functional size measurement method, as defined by ISO/IEC. Thus, this technique can be extended to any Functional Size Measurement method that is found to be compliant with the ISO/IEC standard. The E&Q technique has proved to be very effective, providing a result within ±10% of the actual size in most cases, while the savings in measurement effort can be between 50% and 90% (depending on the aggregation level used, up to Macro Processes).

**Using Functional Size to Estimate Project Effort and Duration**

Once you have established your project’s approximate functional size expressed as a number of function points (or FSM units), you can use the ISBSG data to estimate the likely project effort and duration.

**Summary & Tips**

There are a large number of techniques available to provide estimates of the functional size of projects. In this article, we have explained the different levels of functional sizing and then looked at just three simple but effective ways of roughly determining the Functional Size of a project without doing a detailed function point count. It is important to remember that estimating functional size only results in an approximate size expressed in functional size units, it is not an estimate of effort, duration or cost. Always estimate using two different methods, (perhaps macro and micro), to reduce risk and provide cross checking. Express your estimates as a range, (worst case, likely, best case), or plus and minus a percentage.

The ISBSG software project repository currently contains data on 3,850 completed software projects. These projects have come from more than 20 countries and cover a large range of industries, applications, platforms and languages. Demographics of the ISBSG repository can be downloaded from the ISBSG website.

The ISBSG makes its data available and publishes books and analysis reports to help IT practitioners and IT customers better manage IT resources.

**About the Author**

Peter Hill is the CEO of the International Software Benchmarking Standards Group.

He has been in the Information Services industry for forty years with broad experience covering a number of industries including manufacturing, distribution, freight and aviation. He has worked in both Australia and New Zealand.

Mr Hill has been a speaker at conferences in Australia, China, Denmark, Finland, Malaysia, Netherlands, New Zealand, Spain and UK, with numerous articles published, covering key aspects of the Information Services industry. He is a past Chairman, Secretary and Fellow of the Victorian branch of the Australian Computer Society.

Mr Hill has compiled and edited five books for the ISBSG: “Software Project Estimation”, “The Benchmark Release 6”, “The Benchmark Release 8”, “Practical Project Estimation” and “The Software Metrics Compendium”. He runs courses on Project Management, with an emphasis on software acquisition projects and on the practical use of software metrics. He holds an MBA (Technology Management) from LaTrobe University.


**Author Contact Information**

Peter Hill, CEO, ISBSG  
Ph: +61 3 96450369  
Email: peter.r.hill@isbsg.org

GUFPI - ISMA (Gruppo Utenti Function Point Italia - Italian Software Metrics Association)  
Contact: Dr Domenico Natale  
Phone: +39-6-5025 2396  
FAX: +39-6-50254190  
E-mail: dnatale@sogei.it  
Web Site: www.gufpi.org

FiSMA (Finnish Software Measurement Association)  
Mr Pekka Forselius  
Phone: +35 8505 160416  
FAX: +35 8934 42771  
E-mail: pekkaf@sttf.fi
The first 50 people to send in a completed survey will receive a FREE DoD/IT Acronym CD from the DACS.

This valuable CD-ROM contains over 9,000 Department of Defense and Information Technology acronyms. There are hundreds of acronym lists available but none are as well done as this CD AND specifically targeted towards DoD and Information Technology. This unique-shaped CD-ROM plays in your computer’s regular, hub-mounted, CD drive. You’ll use this great resource over and over again. It’s FREE, just for filling out our brief survey on the next page!

Data & Analysis Center for Software (DACS)

http://iac.dtic.mil/dacs/
Software Tech News Subscriber Survey


2. When did you receive the newsletter? (month/year)

3. How satisfied were you with the CONTENT of the newsletter? (Article Quality)
   - Very Satisfied
   - Satisfied
   - Neither Satisfied nor Dissatisfied
   - Dissatisfied
   - Very Dissatisfied

4. How satisfied were you with the APPEARANCE of the newsletter?
   - Very Satisfied
   - Satisfied
   - Neither Satisfied nor Dissatisfied
   - Dissatisfied
   - Very Dissatisfied

5. How satisfied were you with the OVERALL QUALITY of the newsletter?
   - Very Satisfied
   - Satisfied
   - Neither Satisfied nor Dissatisfied
   - Dissatisfied
   - Very Dissatisfied

6. How satisfied were you with the ACCURACY of the address on the newsletter?
   - Very Satisfied
   - Satisfied
   - Neither Satisfied nor Dissatisfied
   - Dissatisfied
   - Very Dissatisfied

7. Approximately how much of the newsletter do you read?
   - The entire issue
   - Most of the content
   - About half the content
   - Briefly Skimmed
   - Didn't Read

8. Would you read this newsletter in an E-mail newsletter format?
   - Definitely
   - Probably
   - Not Sure
   - Probably Not
   - Definitely Not

9. How did you request the product or service?
   - Phone Call
   - E-mail
   - DACS Website
   - Subscription Form
   - Other

10. Would you recommend the Software Tech News to a colleague?
    - Definitely
    - Probably
    - Not Sure
    - Probably Not
    - Definitely Not

11. What topics would you like to see this newsletter devoted to?

    Comments (Optional)

    Contact Information (Optional*)
    
    Name: Position/Title:
    Organization: Office Symbol:
    Address:
    City: State: Zip Code:
    Country: E-mail:
    Telephone: Fax:
    Functional Role:
    
    Organization Type: Air Force Army Navy Other DoD
    Commercial Non-Profit Non-US US Government FFR&D Other

*Note: You must give us your address to receive the CD.
About the Software Tech News

STN Editorial Board

Philip King
Editor
ITT Industries, DACS

Paul Engelhart
DACS COTR
Air Force Research Lab (IFED)

Morton A. Hirschberg
Editorial Board Chairman
Army Research Lab (retired)

Ellen Walker
ITT Industries, DACS

Thomas McGibbon
DACS Director
ITT Industries, DACS

Article Reproduction

Images and information presented in these articles may be reproduced as long as the following message is noted:

“This article was originally printed in the Software Tech News, Vol. 9, No. 2. Requests for copies of the referenced newsletter may be submitted to the following address:

Philip King, Editor
Data & Analysis Center for Software
P.O. Box 1400
Rome, NY 13442-1400

Phone: 800-214-7921
Fax: 315-334-4964
E-mail: news-editor@dacs.dtic.mil


In addition to this print message, we ask that you send us three copies of any document that references any article appearing in the Software Tech News.

Cover Design by Joseph Barbaccia,
ITT Industries

About This Publication:

The Software Tech News is published quarterly by the Data & Analysis Center for Software (DACS). The DACS is a DoD sponsored Information Analysis Center (IAC), administratively managed by the Defense Technical Information Center (DTIC). The DACS is technically managed by Air Force Research Laboratory, Rome, NY and operated by ITT Industries, Advanced Engineering and Sciences Division.

To Subscribe to this Publication Contact:

Phone: 800-214-7921
Fax: 315-334-4964
E-mail: news-editor@dacs.dtic.mil
Web: www.dacs.dtic.mil

Distribution Statement:
Unclassified and Unlimited
STN Vol. 9, No. 2

In This Issue

Tech Views .................................................................1


Introduction to Function Points .................................8

Software Early Lifecycle – Functional Sizing ..................11

Advertisement

The DoD Software Tech News is now accepting advertisements for future newsletters. In addition to being seen by the thousands of people who subscribe to the DoD Software Tech News in paper copy, the Tech News will also be placed on the Data & Analysis Center for Software’s website (http://iac.dtic.mil/dacs/), exposing your product, organization, or service to hundreds of thousands of additional eyes.

Interested in learning more? For rates, layout information, and requirements contact:

Philip King, STN Editor
Data & Analysis Center for Software
P.O. Box 1400
Rome, NY 13442-1400

Phone: (800) 214-7921
Fax: (315) 334-4964
E-mail: news-editor@dacs.dtic.mil

Data & Analysis Center for Software
P.O. Box 1400
Rome, NY 13442-1400

Return Service Requested