
The Principles of Sizing and Estimating Projects Using International Function Point User Group (IFPUG) Function Points

2006

David Garmus

This article will consider the use of a basic estimating model utilizing functional sizing as one of the key components. The value to be gained from utilizing a functional sizing technique, such as Function Points, is primarily in the capability to accurately size and estimate a project early in the development process.

19 Pointe View Drive
Medford, NJ 08055
T: 609.654.6227

1935 Salt Myrtle Lane
Orange Park, FL 32003
T: 904.269.0211

The David Consulting Group
Achieving Software Excellence

..... www.davidconsultinggroup.com

INTRO

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 can not 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.

ESTIMATING PRINCIPLE

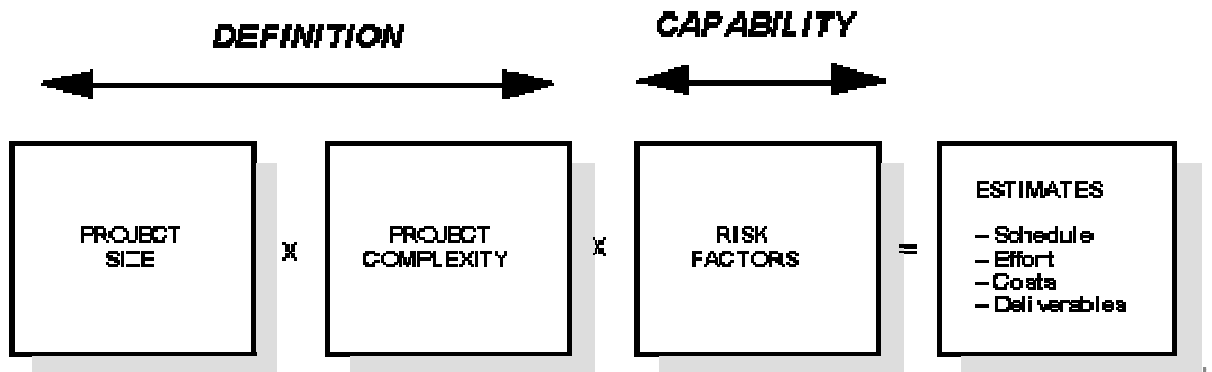


Figure 1

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

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 following five varying levels of complexity (Function Point Analysis; Measurement Practices for Successful Software Projects, Addison-Wesley by David Garmus and David Herron contains significantly more detail):

Level 1

*Simple addition/subtraction
Simple logical algorithms
Simple data relationships*

Level 2

*Many calculations including multiplication/division in series
More complex nested algorithms
Multidimensional data relationships*

Level 3

*Significant number of calculations typically contained in payroll/rating/
scheduling applications
Complex nested algorithms
Multidimensional and relational data relationships*

Level 4

*Differential equations typical; Fuzzy logic; Extremely complex data
Extremely complex logical and mathematical algorithms typically seen in
military/telecommunications/real-time/automated process
control/navigation systems*

Level 5

On-line, continuously available, critically timed event driven outputs

The David Consulting Group
Achieving Software Excellence

*occur simultaneously with inputs
Memory, timing and communication constraints*

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 examples of influencing factors that must be evaluated in order to produce an accurate estimate are categorized in Figure 2:

MANAGEMENT		DEFINITION		DESIGN	
Y	Team Dynamics	Y	Clearly Stated Requirements	Y	Formal Processes
Y	High Morale	Y	Formal Processes	Y	Rigorous Reviews
Y	Project Tracking	Y	Customer Involvement	Y	Design Reuse
Y	Project Planning	Y	Experience Levels	Y	Customer Involvement
Y	Automation	Y	Business Impact	Y	Experienced Development Staff
Y	Management Skills			Y	Automation
BUILD		TEST		ENVIRONMENT	
Y	Code Reviews	Y	Formal Testing Methods	Y	New Technology
Y	Source Code Tracking	Y	Test Plans	Y	Automated Processes
Y	Code Reuse	Y	Development Staff Experience	Y	Adequate Training
Y	Data Administration	Y	Effective Test Tools	Y	Organizational Dynamics
Y	Computer Availability	Y	Customer Involvement	Y	Certification
Y	Experienced Staff				
Y	Automation				

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 below in Exhibit 3.

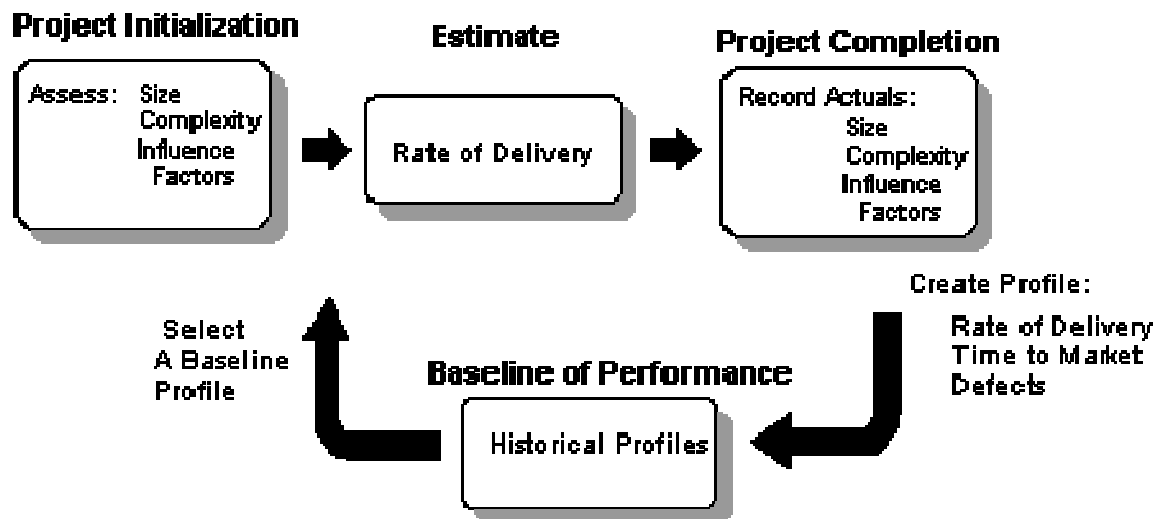


Figure 3

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 *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 proper tools and techniques necessary to accurately estimate projects. The return on that investment 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 & measurement.