Estimating Software Earlier and More Accurately

David Garmus and David Herron
The David Consulting Group

Software practitioners are frequently challenged to provide early and accurate software project estimates. A U.S. government study on software development projects revealed the extent of that challenge: 60 percent of projects were behind schedule, 50 percent of projects were over cost, and 45 percent of delivered projects were unusable. This article explores the use of a basic estimating model, which utilizes functional sizing as one of its key components. The primary value gained from utilizing a functional sizing technique such as function points is the capability to accurately estimate a project early in the development process.

Two issues are at the core of the estimation challenge. First, is the need to understand and define (as early as possible) the problem domain. Second, is the need to understand the capacity to deliver the required software solution within a specified environment. Then and only then can accurate predictions be made of the effort necessary to deliver the required product.

The problem domain can be defined as the definition of requirements. The problem domain must be accurately assessed in size and complexity. Furthermore, the ability to develop an initial estimate (early in the system's life cycle) must exist since we cannot presume to have all of the necessary information at our disposal. Therefore, a rigorous process must exist that permits further clarification of the problem domain as additional knowledge of the required solution is gained.

The capability to deliver is derived from an assessment of risk factors that are known to impact rate of delivery.

An effective estimation model considers three elements: functional size, complexity, and risk factors. When factored together, the opportunity to achieve an accurate estimate is significantly increased (see Figure 1).

Functional Size

By far, the project-sizing technique that delivers the greatest accuracy and flexibility is function point analysis. Based upon several recent analytical studies performed for client organizations, the function point sizing method was compared with other sizing techniques, including backfiring from source lines of code, approximation, ratios, and estimation. The results concluded that the function point method consistently produced more accurate sizing of the software product.

As to its flexibility, the function point methodology presents the opportunity to size a user requirement regardless of the level of detail available. An accurate

> *"An effective estimation model considers three elements: functional size, complexity, and risk factors."*

function point size can be determined from the detailed information included in a logical, user-defined requirements document or from the limited information provided in an early proposal.

The function point method is dependent upon the 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. (For exam-

ple, we know that we will generate an output report, although we may not know the detailed characteristics of that report.)

The first level of function point counting is to identify these five elements. As more information becomes available regarding the characteristics of these elements such as data fields, file types, and so on, the function point count will become more detailed. During the early phases of a count, it may be necessary to assume levels of complexity within the system (for example, whether the report will be simple or complex). The value of using function points is that it allows for this distinction and, in fact, requires it early in the process.

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 process using basic diagramming techniques with the requesting user can be executed to more fully define the requirements. Function points can be utilized early in the life cycle in conjunction with a context diagram or other diagramming devices such as mind maps or use case diagrams. The developed diagram is used to identify stated inputs, outputs, inquiries, internal stores of data, and external stores of data (see Figure 2). For an average-size project, hours (not days) are required to complete the diagramming and sizing task.

From the example in Figure 2, we can quickly identify, at a high level, the inputs, outputs, internal, and external files. At this level, we could easily assign average values of complexity and quickly determine a functional value that we would then use in our estimating model.

Project Complexity

The second element addressed in our estimation model is project complexity.
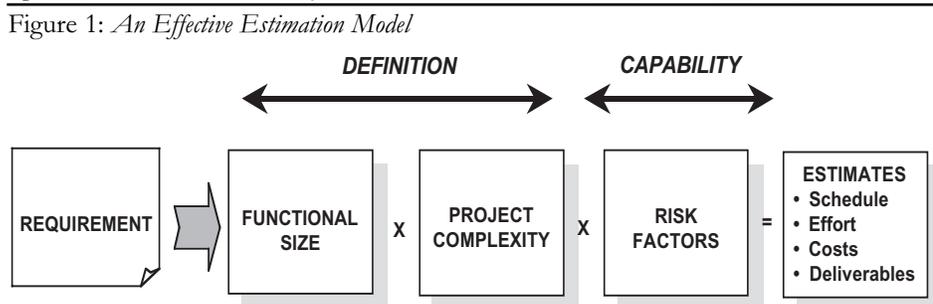
Figure 1: *An Effective Estimation Model*

Project complexity must be properly evaluated and should consider the impact of various contributing characteristics that may influence the ease or difficulty in developing the required solution. In part, complexity level is evaluated as part of the function point methodology.

A value adjustment factor (VAF) is used as a multiplier of the unadjusted function point count to calculate the adjusted function point count of an application. The VAF is determined by identifying 14 general system characteristics (GSCs). Each characteristic has an associated description that helps in determining the degree of influence (of that characteristic). The degree of influence for each characteristic ranges on a scale from zero (no influence) to five (strong influence). The 14 GSCs are totaled to calculate a total degree of influence (TDI).

A VAF is calculated from the following formula: VAF = (TDI * 0.01) + 0.65. When applied, the VAF adjusts the unadjusted function point count by ±35 percent to produce the adjusted function point count. Detailed guidance is contained within the International Function Point Users Group (IFPUG) Counting Practices Manual. Information on IFPUG can be found on their Web site at <www.ifpug.org>.

Each of the following 14 GSCs is evaluated and assigned a degree of influence between zero and five:

1. Data Communications: Describes the degree to which the application communicates directly with the processor.
2. Distributed Data Processing: Describes the degree to which the application transfers data among components of the application.
3. Performance: Describes the degree to which response time and throughput performance considerations influenced the application development.
4. Heavily Used Configuration: Describes the degree to which computer resource restrictions influenced the development of the application.
5. Transaction Rate: Describes the degree to which the rate of business transactions influenced the development of the application.
6. Online Data Entry: Describes the degree to which data are entered through interactive transactions.
7. End-User Efficiency: Describes the degree of consideration for human factors and ease of use for the user of the application measured.
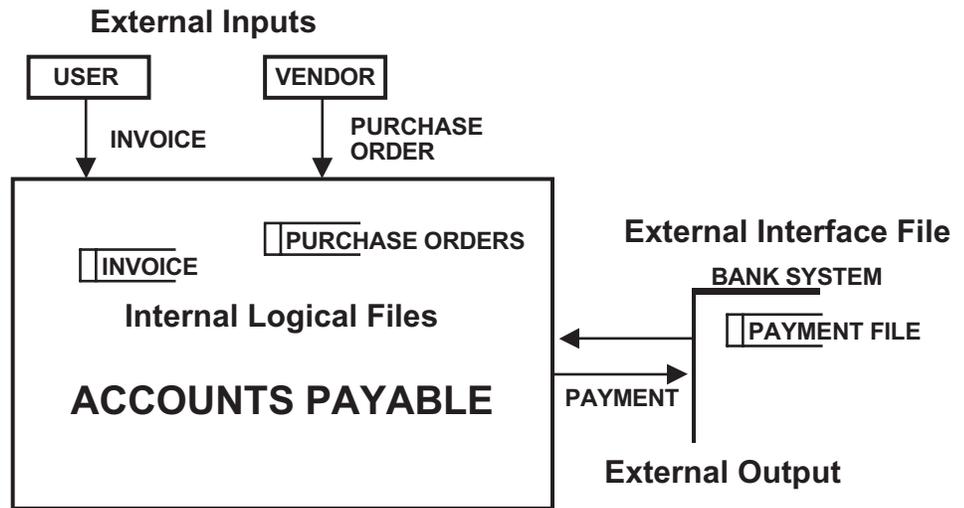
External Inputs

USER   VENDOR

INVOICE   PURCHASE ORDER

INVOICE   PURCHASE ORDERS

Internal Logical Files

ACCOUNTS PAYABLE

External Interface File

BANK SYSTEM

PAYMENT FILE

PAYMENT

External Output

Figure 2: *Example of a Function Point Diagramming Device*

8. Online Update: Describes the degree to which internal logical files are updated online.
9. Complex Processing: Describes the degree to which processing logic influenced the development of the application.
10. Reusability: Describes the degree to which the application and the code in the application have been specifically designed, developed, and supported to be usable in other applications.
11. Installation Ease: Describes the degree to which conversion from previous environments influenced the development of the application.
12. Operational Ease: Describes the degree to which the application attends to operational aspects such as start-up, back-up, and recovery processes.
13. Multiple Sites: Describes the degree to which the application has been developed for multiple locations and user organizations.
14. Facilitate Change: Describes the degree to which the application has been developed for easy modification of processing logic or data structure.

In addition to the 14 GSCs, a project's complexity assessment must take into consideration complex interfaces, database structures, and contained algorithms. These complexity factors impact the project delivery, and they do not serve as an adjustment to the function point count. The complexity assessment can be based upon five varying levels of complexity:

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/actuarial/rating/scheduling applications.
- Complex, nested algorithms.
- Multidimensional and relational data relationships with a significant number of attributive and associative relationships.

Level 4:
- Differential equations typical.
- Fuzzy logic.
- Extremely complex, logical, and mathematical algorithms typically seen in military/telecommunications/real-time/automated process control/navigation systems.
- Extremely complex data.

Level 5:
- Online, continuously available, critically timed.
- Event-driven outputs that occur simultaneously with inputs.
- Buffer area or queue determines processing priorities.
- Memory, timing, and communication constraints.
- The most advanced applications developed.

Risk Factors
The capability to effectively deliver software on time and within budget is based upon a variety of risk factors. The third element in our estimating model is an evaluation of risk factors, including soft-

MANGEMENT

- Team Dynamics
- Morale
- Project Tracking
- Project Planning
- Automation
- Management Skills

DEFINITION

- Clearly Stated Requirements
- Formal Process
- Customer Involvement
- Experience Levels
- Business Impact

DESIGN

- Formal Process
- Rigorous Reviews
- Design Reuse
- Customer Involvement
- Experienced Development Staff
- Automation

BUILD

- Code Reviews
- Source Code Tracking
- Code Reuse
- Data Administration
- Computer Availability
- Experienced Staff
- Automation

TEST

- Formal Testing Methods
- Test Plans
- Development Staff Experience
- Effective Test Tools
- Customer Involvement

ENVIRONMENT

- New Technology
- Automated Process
- Training
- Organizational Dynamics
- Certification

Figure 3: *Factors That Influence Risk*

ware processes utilized, staff skill levels (including user personnel), automation utilized, and the influences of the physical (development conditions) and business environment (competition and regulatory requirements). Categorized in Figure 3 are some examples of influencing factors that must be evaluated to produce an accurate estimate.

As each project commences, the size, complexity, and various risk factors are assessed, and an estimate is derived. Initially, the resulting estimate would typically be based upon industry data that reflect average occurrences of behavior given a project's size, complexity, and performance profile. Over time, as an organization develops a historical baseline of information regarding its own behaviors, performance profiles would reflect a more accurate representation of likely outcomes (see Figure 4). This information can be used to predict and explore what-if scenarios on future projects.

Bringing It All Together

As an example of how the elements of the estimating model fit together,

assume we are about to initiate a project that requires an enhancement to an existing system. The first step would be to fully understand the stated requirements and to evaluate those requirements from a functional perspective. Based upon our understanding of the functionality being added, changed, or deleted from the existing system, we would apply the function point method to determine the size of the requirements.

The function point method would consider all new or changed inputs, outputs, inquiries, interfaces, and internal stores of data. Based on a series of weights and algorithms, an unadjusted function point size would be derived. This unadjusted value would then be fine tuned by an examination of the 14 general system characteristics and value adjustment factors noted earlier. The result would equate to an adjusted function point value.

Let us assume in this example that the resulting function point size is 250 function points. This information alone tells us very little about what it will take to make the required changes to the

existing system. We need to consider any additional levels of complexity that may inherently be part of the change required, and we need to assess our capacity to design, develop, and deploy a project of this given size.

Industry data indicate that (on average) an enhancement project of a given size would equate to a range of delivery rates. Considering our example's size of 250 function points, we know from standard industry data that we could expect a delivery rate ranging from four to 25 function points per person-month. That is a wide range, which will of course be influenced by a variety of factors, including those previously mentioned.

To effectively complete the estimate (and our example), we must evaluate all of the risk factors that will influence the ability to deliver the required changes. If project data have been collected and analyzed for a statistically relevant period of time, performance profiles would be available that would pinpoint a likely outcome when we matched the current profile of risk factors to an existing profile.
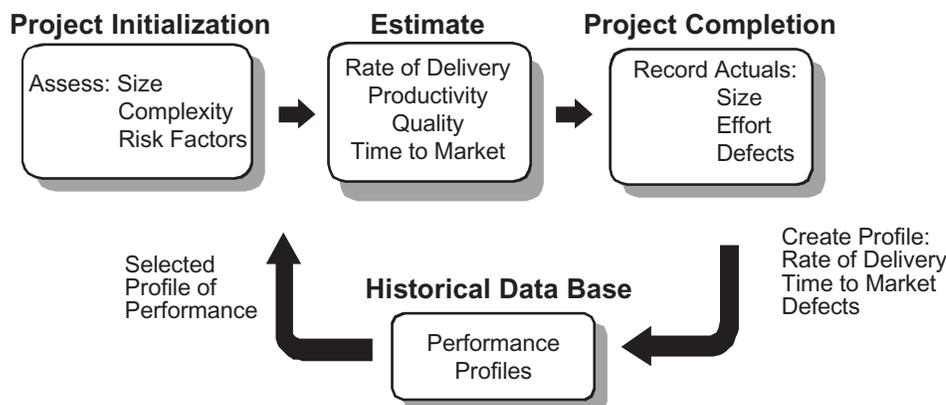
In place of an internal historical base of information, organizations are dependent upon industry data sources. These sources of industry data will produce varied results relative to an organization's actual experiences, but they are of significant value when there is little information available from within.

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 to ballpark their estimates. As they continue to develop an experience base of their own, they can transition from using industry data to using their own data.

The International Software Benchmarking Standard Group (ISBSG) is one of several opportunities that currently exist for gathering, retrieving, and sharing industry data. ISBSG operates on the principle of a well-defined collection process that feeds a central repository, making the data available for detailed access and comparison to industry best practices. The advantage of industry databases is the accessibility of detailed data. Information on ISBSG can be found at <www.isbsg.org.au>.

Figure 4: *Developing and Using a Historical Baseline*

Project Initialization

Assess: Size
Complexity
Risk Factors

Estimate

Rate of Delivery
Productivity
Quality
Time to Market

Project Completion

Record Actuals:
Size
Effort
Defects

Selected
Profile of
Performance

Historical Data Base

Performance
Profiles

Create Profile:
Rate of Delivery
Time to Market
Defects

Summary

Accurate and early estimating requires the following:

- 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 says in *Building Software Quality* [1], "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.

Reference

1. Glass, Robert. <u>Building Software Quality</u>. Prentice Hall PTR, 1997.

Additional Reading

1. Garmus, David, and David Herron. <u>Measuring the Software Process: A Practical Guide to Functional Measurement</u>. Prentice Hall, 1996.
2. Garmus, David, and David Herron. <u>Function Point Analysis: Measurement Practices for Successful Software Projects</u>. Addison-Wesley Information Technology Series, 2000.

About the Authors

David Garmus is a principal and founder of The David Consulting Group. He is an acknowledged authority in sizing, measuring, and estimating software application development and maintenance with more than 25 years of experience in managing, developing, and maintaining computer software systems. Concurrently, as a university instructor he taught courses in computer programming, system development, information systems management, data processing, accounting, finance, and banking. Garmus is the immediate past president of the International Function Point Users Group (IFPUG) and a member of the Counting Practices Committee from 1989 through 2001. He previously served IFPUG as chair of the Certification Committee, chair of the New Environments Committee, and on the Board of Directors as director of Applied Programs, vice president, and president. Garmus has a bachelor's of science degree from the University of California at Los Angeles, and a master's degree from Harvard University.

The David Consulting Group
1935 Salt Myrtle Lane
Orange Park, FL 32003
Phone: (904) 269-0211
Fax: (904) 215-0444
E-mail: dcg_dg@compuserve.com

David Herron is a principal and founder of The David Consulting Group. He is an acknowledged authority in using metrics to monitor information technology's (IT) impact on business, including advancement of IT organizations to higher levels on the Software Engineering Institute's Capability Maturity Model® and on the governance of outsourcing arrangements. He assists clients in establishing software measurement, process improvement, and quality programs and to enhance their project management techniques. Herron has more than 25 years experience in managing, developing, and maintaining computer software systems. He serves as a Cutter Consortium Expert Consultant. Herron attended Union College and Northeastern University. He is chair of the International Function Point Users Group (IFPUG) Management Reporting Committee, a member of the IFPUG IT Performance Committee, and a member of the American Society for Quality.

The David Consulting Group
19 Pointe View Drive
Medford, NJ 08055
Phone: (609) 654-6227
Fax: (609) 654-2338
E-mail: dcg_dh@compuserve.com