The Estimating Challenge **2**
Reasons for Poor Estimates 2
Results of Poor Estimating 3

Phase-Limited Commitment **4**

System Sizing Estimates **6**

Function Point Analysis **7**
Overview of Function Point Counting 7
Developing a Function Point Estimate 8
Function Point Index or Function Point Productivity 9
Measuring the Work Product Output 9
Comparison of Function Points and Lines of Code 9
Function Point Definitions and Guidelines **9**
External Inputs 10
External Outputs 10
Function Complexity Weights 11
Logical Internal Files 11
External Interface File 12
External Inquiry Type 12
Processing Complexity 13
Processing Complexity Factors 13
A Sample Function Point Computation **16**
A Final Note on Function Points **18**

Phase-Distribution Estimating **19**
Effort Distribution Models 19
Sizing A Project Using Effort Distribution Models 19

The Project Task Estimates Guidelines **20**
The Steps Involved in the Estimating Process **20**
1. Assessing the Planning Factors 21
2. Assessing Personal and Organizational Experience 22
3. Assessing Historical Data and Available Methods 23
4. Review Estimating Methods 24
5. Document Estimates and Assumptions on Which Estimates are Based 24
Factors to Consider When Estimating **24**
General Factors That Will Affect All Project Phases 24
Factors That Will Affect Analysis 25
Factors That Will Affect Design 25
Factors That Will Affect Development 25
Factors That Will Affect Installation 25
Task-Based Estimating Methods **26**
Intuitive Estimating 26
Participative Estimating 27
Statistical Estimating 28
Formula Estimating 29

Final Thoughts on the Estimating Process **30**
Avoid SWAG estimates 30
Use estimates based on past experience 30
Involve the whole Project Team 30

There's no point in being exact about something if you don't even know what you're talking about.

--John von Neumann

The Estimating Challenge

Estimating work effort is one of the most difficult activities in project management. It's also one of the most important. It is unfortunate that projects are justified at the worst time -- at the beginning. At this stage in the project cycle, the estimating process is complicated by a high degree of uncertainty about the project requirements, the Project Team, and the ultimate project plan.

Tom DeMarco cites the chief causes of poor software estimation as:

1. **We don't develop estimating expertise.** The initial estimate is usually arrived at by trying to deduce the completion date that the boss already has in his head when he assigns us to a project. Subsequent estimates are refinements of this little game. As a result, we never do any real estimating.
2. **We don't make adequate provision to offset the effect of our biases.** This sounds like the type of problem that we should be able to fix. Our built-in biases will tend to produce errors of similar magnitude and in the same direction each time. However, human nature being what it is, our biases are usually invisible to us; although biases in others will be quite obvious. Another aspect of human nature is that we usually tend to be optimistic. We will almost always estimate on the low side. Our ego gets in the way of our being truly objective about our abilities.
3. **We don't have an adequate understanding of what an estimate ought to be.** Kidder has proposed that an estimate usually is "the earliest date by which you can't prove you won't be finished." A better definition is that an estimate is "a prediction that is equally likely to be above or below the actual result."
4. **We don't cope well with political problems that hamper the estimating process.** Most estimates have political undertones. An estimate to complete a project becomes a goal rather than an honest assessment of completion. Since the goal has been arrived at without any real understanding of the amount of work involved, the completion date becomes the overriding concern. The most important casualty in this process is product quality, for which we will continually pay over the product's useful life.
5. **We don't base our estimates on past performance.** We need to have quantifiable indications as to project scope, quality, complexity and performance if we are to provide meaningful estimates.

This uncertainty increases the difficulty of the estimating process and decreases the likelihood that estimates will match future project activity. The estimating process is further complicated by the political pressures in the project organization -- Project Managers are often pressured into producing unrealistically low estimates to satisfy anxious users, to preserve inadequate organizational budgets, or to get "pet" projects over the organization's funding hurdle. These low estimates may, in turn, be transformed into "official" project goals, despite the best intentions of the manager to revise them at some future date -- this leads to ***real credibility issues*** between users and IS.

Reasons for Poor Estimates

- Undefined objectives
- Dictated deadlines

- Incomplete plans
- Unrealistic resource assumptions
- Naive estimates
- Customer/management arrogance
- Lack of training in developing estimates
- Adverse relationship between estimators and customers
- Absence of accountability
- Lack of candor

Results of Poor Estimating

- Unsatisfactory project outcomes
- Missed deadlines
- Wasted talent
- High costs
- Loss of credibility
- Customer dissatisfaction

In light of increasing business and financial risk from ever more complex projects, inaccurate project estimating has become a major concern for Project Sponsors.

A documented software development life cycle combined with careful analysis of business requirements and risks with clearly defined task deliverables will help ensure realistic estimates despite project uncertainty or political pressures.

Estimating requires a variety of information:

- Information about tasks and deliverables (contained in the task and deliverable descriptions)
- Information about project planning factors (documented in the list of assumptions)
- Information about past projects (contained in historical project files)
- information about organization estimating methods or guidelines (documented in organization standards)
- Information about the Project Manager's own level of experience

"Estimating should not have accuracy as a goal: the true goal is to improve the quality of decisions today." -- Rich's Axiom

The truth about estimating. . .

- We can only estimate based upon some preliminary design;
- Estimates are only good for things we have done before, using people we can depend on;
- Murphy's law is always at work;
- There is ALWAYS a learning curve;
- The things we forget to include in the estimate are the things that cause it to be inaccurate; and,
- Getting better at producing estimates comes from documenting them and tracking them over time.

Phase-Limited Commitment

Development projects inevitably entail modification and revision as the effort progresses. Because it's impossible to anticipate all of these modifications at the time when initial estimates are developed, using a "***Phase-Limited Commitment***" approach to work effort estimating, as well as to other planning activities.

In this approach, work effort estimates for imminent subphases are developed in detail, while only general estimates are developed for subsequent subphases and phases. As each subphase becomes imminent, the general estimates are redone in detail. This scheme allows a manager to adapt easily to project modifications, and eliminates the reworking that would take place if detailed estimates were created for the entire project too early in the development cycle. This is the most realistic way to estimate a project, although it may be more difficult to sell to upper management.

Software delivery proceeds using a "phase-limited commitment" approach. The basics of phase-limited commitment" is as follows:

1. Customer commitment for expenditure of funds only through completion of the current phase of development;
2. Project Manager must obtain customer approval to proceed to the next phase at the end of each phase; and,
3. Customer must review and approve deliverables associated with the current phase as well as the proposal for the next phase of development prior to the start of the next phase.

The project phases are: 1) Feasibility Analysis, 2) Requirements Definition, 3) Design, 4) Code and Unit Test, 5) System Testing (integration, interface, cohabitation and stress tests), 6) Customer Testing and Training, and 7) Installation. After installation three phases remain, Operate, Use, and Evaluate the new application.

With the completion of each phase, downstream estimates are produced for the future phase and a Within each phase there may be a number of formal Executive Steering Committee review and approval points, and a final project phase review at the end of each phase. At any one of those interim review points, the Executive Steering Committee is able to affect, redirect, or approve the work being done. At the formal phase review, the Executive Steering Committee has the option to approve the work being done and continue with development of the next phase, require rework to be done based on the outcome of the review, or cancel the project. In preparing for the formal Executive Steering Committee review, the Project Manager must:

- Create a proposal an initial schedule for the next development phase;
- Present the deliverables produced from current phase;
- Request acceptance of current phase deliverables; and,
- Request for approval to proceed to next development phase.

Another problem which complicates the estimation of work effort is uncertainty about task performers. Ideally, estimates take into account the personal abilities and characteristics of the person who will perform the task. Often the task performer has not been selected at the time estimates are being developed. To compensate for this, a Project Manager must make assumptions about the skill level required to satisfactory task performance, and estimate effort on this basis. When personnel are later selected for each task, it is essential for team buy-in that the individual assigned to the task be given the opportunity to review the task and create their own estimate for the work. The Project Manager must review the validity of earlier estimates -- by assessing actual skill level of the performer to that skill level estimated. If there is a difference, either the estimate (and associated schedules) must be adjusted, or a performer with the assumed skill level must be found.

There are two main products which result from the work effort estimation:

- a ***list of work effort estimates*** for each task in the phase at hand
- a ***list of assumptions*** on which the estimates are based

Using "Phase-Limited Commitment", this list may include detailed estimates for imminent subphases *only*.

System Sizing Estimates

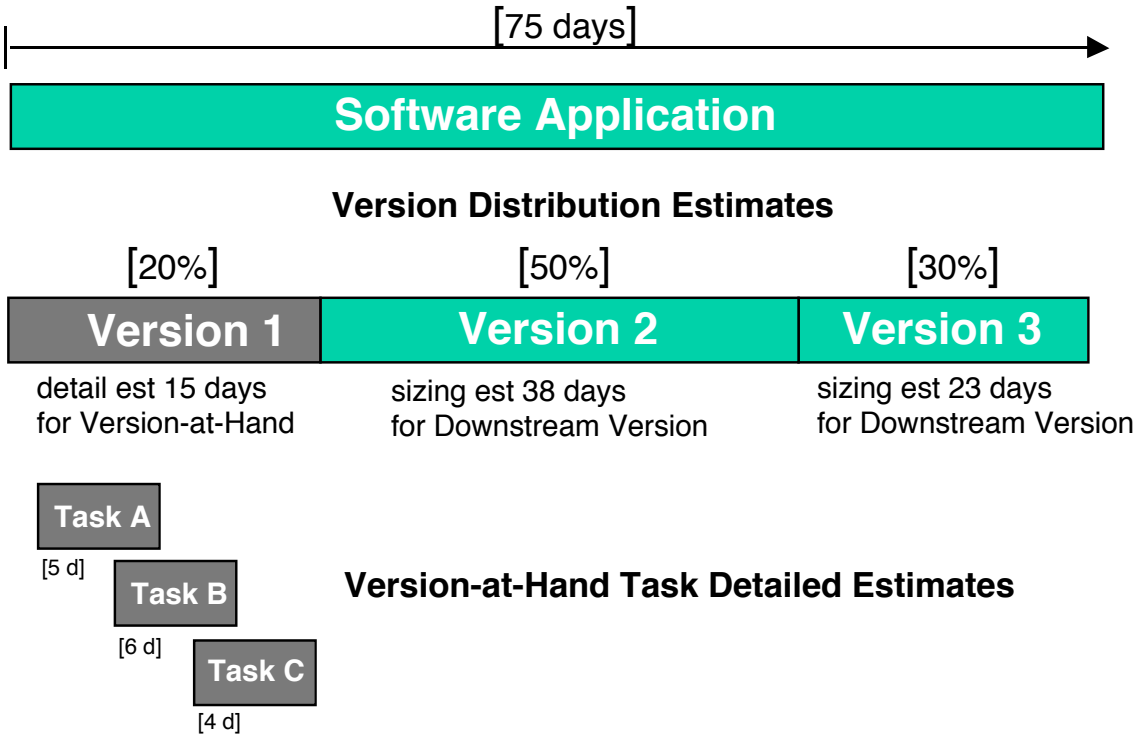
Size estimates are usually developed using Effort Distribution, Deliverables-Based, Parametric, and Function Point models. The objective is to develop a high-level estimate of the proposed project in terms of overall effort, cost, and duration (work days, weeks, or months). A size estimate is usually needed during the Project Definition stage for such strategic concerns as project approval, portfolio assessment, and assessing resource needs. Size estimates serve two key purposes:

1. Used by management to decide if the project is feasible.
2. Assess the effects of the proposed project on the portfolio of current and proposed projects.

Note: Size estimates are not suitable for developing project schedules.

Such estimates need to be presented using the following elements:

1. Define the estimate in terms of deliverables to be completed.
2. Specify resource assumptions (e.g., two full-time senior analysts, one half-time database analyst, and one senior user representative available approximately ten hours per week).
3. Use a range value for estimates (e.g., 41-47 days). A unitary number implies a level of accuracy that is difficult to attain and results in unrealistic expectations by management and customers.



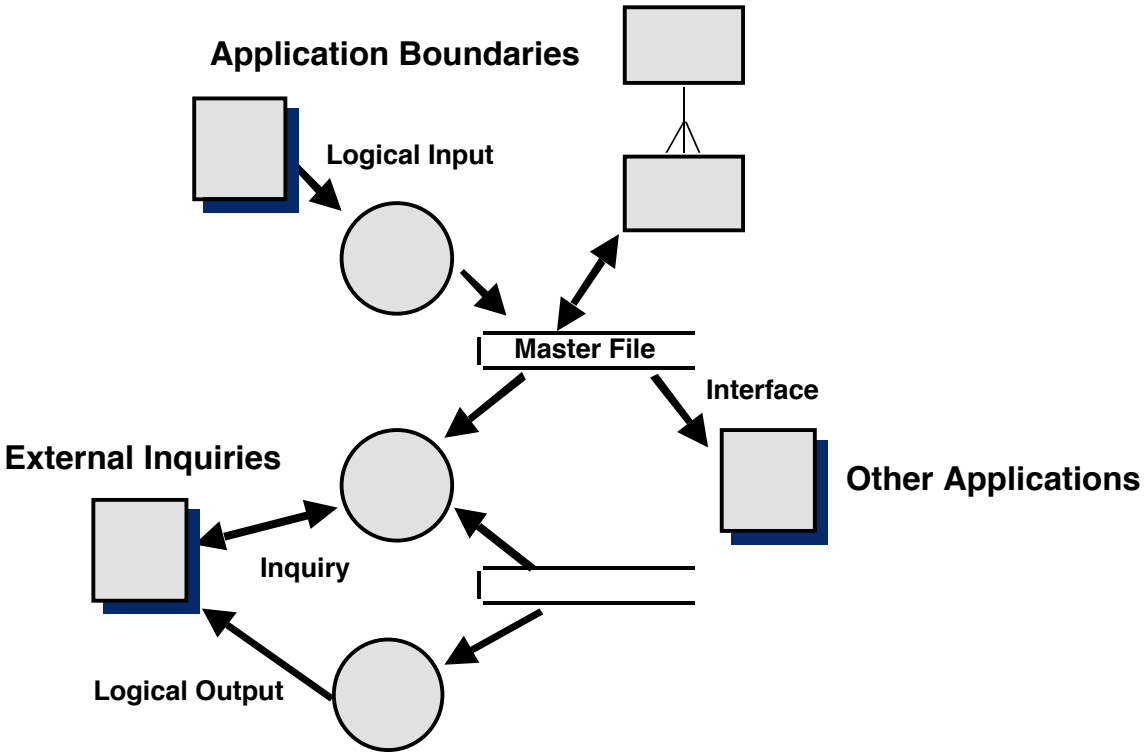
Function Point Analysis

Overview of Function Point Counting

Function Point is a software sizing, productivity measurement, and estimating technique developed by Alan Albrecht [1979] and others in the late 1970s. It is based on two assumptions:

- The complexity and size of a software system are major determinants of the length of the development process.
- The complexity and size of a software system can be derived by examining and counting the data complexity and volume.

Function points attempt to count what customers pay for -- reports, screens, maintained data -- rather than more technical measures such as lines of code. Over the past 10 years, function points have become the most widely used measure of software development productivity. The information in this section provides the basics of function point counting. For a more comprehensive description, the International Function Point Users Group (IFPUG) maintains and publishes a complete counting guide.



The above figure shows the basic components that are counted for function points. The table following shows the basics of the calculation algorithm. The remainder of the section explains the details of the calculations.

However, it should be emphasized the Function Point techniques ignore the differences between individual programmer r analyst productivity and quality requirements, and are not suitable for systems with complex algorithmic processing or hardware or telecommunications intensive projects. As a result,

Function Point counts and associated estimates are used to confirm the estimates developed by other techniques such as Participative Estimating.

With function points, initial application requirements statements are examined to determine the number and complexity of various inputs, outputs, calculations, and databases required. By using values that A. J. Albrecht has established, points are assigned to each of these counts. These points are then summed to produce an overall function point rating for the product. Based on prior experience, this final function point figure can be converted into a reasonably good estimate of the information technology development resources required. Function Point Analysis estimates alone should not be converted into project schedules. Further refinement is required as is discussed below in "Phase-Distribution Estimating" and "The Project Task Estimates Guidelines."

Function Point estimation is based on counting five Application Elements or Functions for the new system or enhancement. It is important to understand that the counting is based on logical, not implementation-based or physical, functions. For example, the same data may be input to the system via an on-line screen and a batch update. Provided the processing of both inputs is the same, the data would be counted as one logical input although it exists in two different physical formats.

A simplified function point count can be used to estimate the size of a project and consequently determine the recommended team size and time frame. The function point count is based on the following parameters and multipliers:

Inputs: Screens, forms, dialog boxes, and other messages with which an end-user or another application adds, deletes, or changes the applications' data.

Outputs: Screens, reports, graphs, or messages the application generates for use by an end-user or another application. Outputs can process, combine, and/or summarize complex data and can also be highly formatted.

Inquiries: Input/output combinations through which input values result in a immediate, simple output. Inquiries result in a direct search of the database for specific data, which is then used as a key to create the simple output. Inquiries retrieve data directly form the database and provide only the most rudimentary formatting.

Logical Internal Files: Major logical groups of end-user data or control information that are completely controlled by the application. A logical file can be a single flat file or a single table in a relational database.

External Interface Files: Files controlled by other applications with which this application must interact. External interface files include each major logical group of data or control information that enters or leaves the application

Data flow diagrams and data models are ideal vehicles for deriving Function point counts as they depict graphically all data components. This overview is designed to introduce Function Point at a level that can be used to calculate a "first cut" Function Point count for developments and enhancements. For more detailed descriptions of the technique, Brian Dreger's *Function Point Analysis* provides a complete and standard approach to this technique.

Developing a Function Point Estimate

Function Point estimation is based on counting five Application Elements or Functions for the new system or enhancement. It is important to understand that the counting is based on logical, not implementation-based or physical, functions. For example, the same data may be input to the system via an on-line screen and a batch update. Provided the processing of both inputs is the same, the data would be counted as one logical input although it exists in two different physical formats.

The general approach is:

Function Points are based on the user's external view or logical model of the application. Function Points are established for an application by listing, classifying, and counting different elements of an application's data. It should be noted that the use of the word "function" is somewhat misleading as

the counting is based on data, not function. By counting data, Function Points de facto count processes, since processes exist to process data.

While function points help in making early size estimates of the application programs, they have drawbacks. The most important are that the complexity factors are judgmental and it is not possible to automatically measure the function point content of programs.

Albrecht thus advised that function points be used to make initial estimates and then, based on experience, these function point values are converted to LOC. Planning then proceeds using the LOC measure. The function point metric is most helpful when:

- A new program development is being estimated.
- The applications involve considerable input-output or file activity.
- An experienced function point expert is available.
- Sufficient data is on hand to permit reasonably accurate conversion from function points to LOC.

Function Point Index or Function Point Productivity

The index is a value expressed in hours-per-function point. the Productivity Index, derived historically for the organization, is then applied to the Adjusted Function Point score to arrive at an estimate for a particular project. This estimate is an actual hours (work effort -- WE) estimate for the proposed system. This, of course, must be followed separately by an *elapsed* time (ED) calculation based on resources available and other external factors.

Measuring the Work Product Output

Because the end result of Function Point estimating is a total of person-hours of work, it may be regarded as a measurement of effort and cost. It is valid to associate Function points with the value of the application to the user in the sense that they represent the amount of function delivered, that is, user facilities. There is no relationship to any regard of return on investment expected from the application.

Comparison of Function Points and Lines of Code

Size in Lines of Code

Size in FPs	COBOL	C	C++	Visual Basic	Smalltalk
1	90	125	50	30	21
100	9,000	12,500	5,000	3,000	2,100
500	45,000	62,500	25,000	5,000	10,500
1,000	90,000	125,000	50,000	30,000	21,000
5,000	450,000	625,000	250,000	150,000	105,000

Source: Adapted from “Patterns of Software Failure and Success” (Jones 1996)

Function Point Definitions and Guidelines

The following section provides basic definitions of the functions involved in function point sizing and the complexity weights for each function type. These will enable the production of an initial sizing of the system, which can then be used to develop and initial estimate using the organization’s Productivity Index.

External Inputs

Count each unique user data or user control type that enters the external boundary of the application being measured. An external input is considered unique if it has a different format, or if the external design requires a processing logic different from other external input types of the same format. Include external inputs that enter directly as transactions from the user, and those that enter as transactions from other applications.

Each external input type should be classified to one of three levels of information processing function as follows:

- Low - Few data element types are entered, or are made available from logical internal files, by the external input type. User human factors considerations are not significant in the design of the external input type.
- Average - The level of information processing function of the external input is not clearly low or high.
- High - Many data element types are entered, or are made available from logical internal files, by the external input type. User human factors considerations significantly affect the design of the external input type.

Do not include input files of records as external input types because these are counted as external interface file types.

Do not include the input part of the external inquiry types as external input types because these are counted as external inquiry types.

External Outputs

Count each unique user data or control output type that leaves the external boundary of the application being measured. An external output type should be considered unique if it has a different format or if the external design requires a processing logic different from other external output types of the same format. Include external output types that leave directly as reports and messages to the user, and those that leave as reports and messages to other applications, such as output files of reports and messages.

Each external output type should be classified to one of three levels of information processing function, using definitions similar to those of external input types:

- Low - Few data element types are included in the external output type, and few logical internal file types are referenced by the external output type. User human factors considerations are not significant in the design of the external output.
- Average - The level of information processing function of the external output type is not clearly either low or high.
- High - Many data element types are included in the external output type, and many logical internal file types are referenced by the external output type. User human factor considerations significantly affect the design of the external output type.

For reports, the following additional definitions should be used:

- Low - One or two columns. Simple data element transformations.
- Average - Multiple columns with sub-totals. Multiple data element transformations.
- High - Multiple and intricate data element transformations. Multiple and complex file references to be correlated. Significant performance considerations.

Do not include output files of records as external output types because they are counted as external interface file types.

Do not include the output response of the external inquiry types as external output types because these are counted as external inquiry types.

Function Complexity Weights

Complexity ratings are used to distinguish the relative difference of processing or programming complexity in like system functions. Simple reports, for example require less complex programming that a report with many fields and subtotals. Complexity is divided into three basic categories:

- *Simple:* The minimum level.
- *Average:* The normal level.
- *Complex:* The high level.

The following tables provide the standard weights for the five function types. For inquiries, the more complex of the input or output component is the one used to develop the Function Point score. For example, if the input component is simple (complexity weight of 3) and the output component is complex (complexity weight of 7), then the output component complexity weight (7) is used for the inquiry. The scores are transferred to the Summary Sheet at the end of this overview.

Input Complexity	1-4 attributes	5-15 attributes	16+ attributes
0 or 1 files accessed	3	3	3
2 files accessed	3	4	6
3+ files accessed	4	6	6

Complexity Weight: Simple-3 Average-4 Complex-6

Output Complexity	1-5 attributes	6-19 attributes	20+ attributes
0 or 1 files accessed	4	4	5
2 or 3 files accessed	4	5	7
4+ files accessed	5	7	7

Complexity Weight: Simple-4 Average-5 Complex-7

File Complexity	1-19 attributes	20-50 attributes	51+ attributes
1 logical record/entity	7	7	10
2-5 logical records/ entities	7	10	15
6+ logical records/ entities	10	15	15

Complexity Weight: Simple-7 Average-10 Complex-15

Logical Internal Files

Count each major logical group of user data or control information in the application as a logical internal file type. Include each logical file or (within a data base, each logical group of data from the viewpoint of the user) that is generated, used, and maintained by the application. Count logical files as described in the external design, not physical files.

The logical internal file types should be classified to one of three levels of information processing function, as follows:

- Low - Few record types. Few data element types. No significant performance or recovery considerations.
- Average - The level of information processing function of the logical internal file type is not clearly either low or high

- High - Many record types. Many data element types Performance and recovery are significant considerations.

Do not include logical internal files that are not accessible to the user through external input, output, or inquiry types.

External Interface File

Files passed or shared between applications should be counted as external interface file types within each application. Count each major logical group of user data or control information that enters or leaves the application as an external interface file type.

External interface file types should be classified to one of three levels of information processing function, using definitions similar to those for logical internal file types:

- Low - Few record types. Few data element types. No significant performance or recovery considerations.
- Average - The level of information processing of the logical external interface type is not clearly either low or high.
- High - Many record types. Many data element types. Performance and recovery are significant considerations.

Inquiry Input Complexity	1-4 attributes	5-15 attributes	16+ attributes
0 or 1 files accessed	3	3	4
2 files accessed	3	4	6
3+ files accessed	4	6	6

Complexity Weight: Simple-3 Average-4 Complex-6

Inquiry Output Complexity	1-5 attributes	6-19 attributes	20+ attributes
0 or 1 files accessed	4	4	5
2 or 3 files accessed	4	5	7
4+ files accessed	5	7	7

Complexity Weight: Simple-4 Average-5 Complex-7

Interface File Complexity	1-19 attributes	20-50 attributes	51+ attributes
1 logical rec/entity	5	5	7
2-5 logical recs/ents	5	7	10
6+ logical recs/ ents	7	10	10

Complexity Weight: Simple-7 Average-10 Complex-15

External Inquiry Type

Count each unique input/output combination (where an input causes and generates an immediate output) as an external inquiry type. An external inquiry type should be considered unique if it has a format different from other external inquiry types in either its input or output types, or if the external design requires a processing logic different from other external inquiry types of the same format. Include external inquiry types that either enter directly from the user or that enter from other applications.

The external inquiry types should be classified to one of three levels of information processing function:

- Classify the input part of the external inquiry type using definitions similar to the external input type.
- Classify the output part of the external inquiry type using definitions similar to the external output type.
- The level of information processing function of the external inquiry type is the greater of the two classifications.

To help distinguish external inquiry types from external input types, consider that the input data of an external inquiry type is entered only to direct the search, and no update of logical internal file types should occur.

Do not confuse a query facility as an external inquiry type. An external inquiry type is a direct search for specific data, usually using only a single key. A query facility provides an organized structure of external input, output, and inquiry types should all be counted to measure a query facility.

Processing Complexity

In Function Point sizing, some 14 characteristics have been identified as influencing the processing complexity of a project. These are termed Processing Complexity or Application Characteristics.

There are six degrees of influence (risk) in each processing complexity factor. After reviewing each characteristic, assign the value for the degree of influence that the characteristic has on the development, or enhancement, of the application.

0 = No present, or no influence.
1 = Minor (insignificant) influence.
2 = Moderate influence.
3 = Average influence.
4 = Significant influence.
5 = Strong influence throughout.

Dreger provides a more quantified model of processing complexity, but for initial estimates of in a project, a subjective assessment of the level of influence of each processing complexity factor on the project is usually more appropriate since the details of the technical issues have not usually been finalized.

Processing Complexity Factors

1. **Data Communications**. The data and control information used in the application are sent or received over communication facilities. Terminals connected locally to the control units are considered to use communication facilities.

   0 Application is pure batch processing.
   1 Remote printing or remote data entry.
   2 Remote printing and remote data entry.
   3 TP front end to a batch process.
   4 More than front-end, but only one TP protocol.
   5 Application is dominantly interactive TP.

2. **Distributed Function**. Distributed data or processing functions are a characteristic of the application.

   0 Application does not aid the transfer of data or processing function between components of the system.

   1 Application prepares data for end user processing on another component of the system such as a PC.

2-4 Data is prepared for transfer, transferred, and processed on another component of the system.

5 Processing functions are dynamically performed on the most appropriate component of the system.

3. **Performance**: Application performance objectives, stated or approved by the user in either response or throughput, influenced the design, development installation, and support of the application.

   0-3 No special performance requirements are stated by the user. Performance analysis and design considerations are standard.

   4 Stated user performance requirements are stringent enough to require performance analysis tasks in the design phase.

   5 Stated user performance requirements are stringent enough to, in addition, require performance analysis tools to be used in the design, development, and/or installation phases.

4. **Heavily Used Configuration**: Requires special design considerations and is a characteristic of the application.

   0-3 Typical application run on standard production machine. No stated operation restrictions.

   4 Stated operation restrictions requires special constraints on the application in the central processor.

   5 In addition, there are special constraints on the application in distributed components of the system.

5. **Transaction Rates**: The transaction rate is high and if it influenced the design, development, installation, and support of the application.

   0-3 Transaction rates are such that performance analysis considerations are standard.

   4 High transaction rates stated by the user in the application requirements or service level agreements are high enough to require performance analysis tasks in the design phase.

   5 High transaction rates stated by the user in the application requirements or service level agreements are high enough to, in addition, require the use of performance analysis tools in the design, development, and/or installation phases.

6. **On-line Data Entry**: On-line data entry and control functions are included in the application.

   0-2 None to 15% of the transactions are interactive data entry.

   3-4 15% to 30% of the transactions are interactive data entry.

   5 30% or more of the transaction are interactive data entry.

7. **Design for End User Efficiency**: The on-line functions provided emphasize a design for end user efficiency.

   0-3 No stated special user requirements concerning end user efficiency.

   4 Stated requirements for end user efficiency are strong enough to require tasks for human factors to be included.

   5 Stated requirements for end user efficiency are strong enough to require use of special tools such as prototyping.

8. **On-line Update**: The application provides on-line update for the logical internal files.

   0 None.

   1-2 On-line update of control files. Volume of updating is low and recovery is easy.

   3 On-line update of major logical internal files.

   4 In addition, protection against data loss is essential.

5 In addition, high volumes bring cost considerations into the recovery considerations.

9. **Complex Processing**: Is a characteristic of the application. Which of the following applies to the application?

   * Extensive logical and/or mathematical processing.
   * Much exception processing, many incomplete transactions, and much reprocessing of transactions.
   * Sensitive control and/or security processing.

   0 None of the above apply.

   1-3 Any one of the above applies.

   4 Any two of the above applies.

   5 All of the above applies.

10. **Usable in Other Applications**: The application, and the code in the application, has been specifically designed, developed, and supported to be usable in other applications.

    0-1 A local application addressing the needs of one user organization.

    2-3 Application used or produced common modules that considered more than one user's needs.

    4-5 In addition, the application was specifically packaged and/or documented to ease re-use.

		%		%
0	=	0	to	10
1	=	10	to	20
2	=	20	to	30
3	=	30	to	40
4	=	40	to	50
5	=	50	to	100

11. **Installation Ease**: Conversion and installation ease are characteristics of the application. A conversion and installation plan and/or conversion tools provided were tested during the system test phase.

    0-1 No special conversion and installations considerations were stated by the user.

    2-3 Conversion and installation requirements were stated by the user and conversion and installation guides were provided and tested.

    4-5 In addition, conversion and installation tools were provided and tested.

12. **Operational Ease**: Operational ease is a characteristic of the application. Effective startup, backup, and recovery procedures are provided and were tested during the systems test phase. The application minimizes the need for manual activities, such as tape mounts, paper handling, and direct on-location manual intervention.

    0 No special operational considerations were stated by the user.

    1-2 Effective startup, backup, and recovery processes were required, provided, and tested.

    3-4 In addition, the application minimizes the need for manual activities, such as tape mounts and paper handling.

    5 Application is designed for unattended operation.

13. **Multiple Site**: The application has been specifically designed, developed, and maintained to be installed at multiple sites.

0 No user requirement to consider the needs of more than one user site.

1-3 Needs of multiple sites were considered in the design.

4-5 Documentation and support plan are provided and tested to support the application at multiple sites.

14. **Facilitate Change**. The application has been specifically designed, developed, and maintained to allow the end user to modify the application.

0 No special user requirement to design the application to minimize or facilitate change.

1-3 Flexible query capability is provided.

4-5 In addition, control data is kept in tables that are maintained by the user with on-line interactive processes.

A Sample Function Point Computation

Function Points are scored for each function identified in the proposed system, and those scores are then adjusted for the intrinsic complexity of each function. It is important to avoid duplicated counting. Each business function is counted once and once only. Details of the various functions and the complexity factors, along with guidelines on their use, are found in later sections of this overview.

After analysis of the system components and functions, a count is made of the number of each type of the preceding elements under a subclassification of complexity, that is, *simple*, *average*, or *complex*. These elements are then weighted by a relative complexity weight for each of the three subclassifications. These weights have been established from experience by IBM and many other organizations over a large number of projects and have been accepted unaltered by other users of the Function point technique. The *Unadjusted Function Point* score is then calculated by multiplying each function by its complexity and adding the results together.

Further, analysis is then made of the processing complexity of the system, a scored value estimated for each and these factors applied as a further weighting to the point score to arrive at a total *Adjusted Function Point* sizing.

Processing Complexity (currently 14 specific characteristics have been identified) are factors in a project that, if present, can influence the overall complexity of the project. An example is the extensive use of communications facilities in the system. These processing complexity factors should be a subset of factors for review in any risk analysis approach undertaken by the project.

Count the number and complexity of each item in the application. Use the following Function Point Multiplier table to calculate an estimate of the function point count for the application.

Business Function	Number	Complexity	Weight	Line Total	Type Total
Inputs	74	Simple	x 3	222	
	17	Average	x 4	68	
	0	Complex	x 6	0	91
Input Total					290
Outputs	23	Simple	x 4	92	
	3	Average	x 5	15	
	3	Complex	x 7	21	29
Output Total					128
Files	14	Simple	x 7	98	
	9	Average	x 10	90	

	0	Complex	x 15	0	23
File Total					188
Inquiries	18	Simple	x 4	72	
	6	Average	x 5	30	
	0	Complex	x 6	0	
	3	Complex	x 7	21	27
Inquiry Total					123
Interface Files	4	Simple	x 5	20	
	4	Average	x 7	28	
	0	Complex	x 10	0	8
Interface Total					48
Total Unadjusted Function point					777

Processing Complexity			
Factor	Value	Factor	Value
1. Data communications	5	8. On-line update	5
2. Distributed function	0	9. Complex processing	2
3. Performance	2	10. Code re-usability	0
4. Heavily-used configuration	2	11. Conversion/installation ease	5
5. Transaction rates	0	12. Operational ease	5
6. On-line data entry	5	13. Multiple site installation	0
7. End user efficiency	4	14. Facilitate change	5
Total Degree of Influence			40

Adjusted Function Point = Unadjusted Function Point x ((0.65) + (0.01 x Total Degree of Influence))	816

Table - Function Point Calculation for ACME Ordering System

Raw Function Point =

	Inputs	x	(3/4/6)
+	Outputs	x	(4/5/7)
+	Inquiries	x	(3/4/6)
+	Files	x	(7/10/15)
+	Interfaces	x	(5/7/10)

Adjusted Function Point =

RFP x ((.65) + (.01 x DI)) = AFP
RFP are raw function points
DI is the degree of influence
AFP are the adjusted function points
777 x ((.65) + (.01 x 40)) =816

Effort Conversion =

1. Assume this is a COBOL project using 20 hours per function point (check with IFPUG for the latest tables for development hours per function point by language) to arrive at a conservative effort-hour estimate: **20 x 816 = 16,320 hours**
2. Convert the effort hours to work months: divide the total effort hours by the average productive hours per month per person (114
3. Hours per month used here, represents 6.5 work hours per day and 17.5 work days per month): **Work Months = 16,320 / 114 = 143.2**
4. Compute the project duration value as a range, based on the "best case," total number of qualified staff recommended to work on the project and "worst case," total number of qualified staff *available* for the project:

Project duration = Work Months / FT
143.2 / 13 = 12 months

Project Sizing/Scheduling Guidelines

Project Size in FPs	Team Size	Achievable Productivity Rates (FP/staff month)	Schedule (Mos)
100	2	90	1
300	3	40	2.5
600	4	30	5
>1200	8	25	6-8

A Final Note on Function Points

Function Points have been adopted by most major computer organizations because they easily provide a standard and technology-independent method of sizing software products. However, Function Points are a software sizing technique and, as such, do not measure the non-system effort involved in developing most information systems. As a result, Function Points should be treated as a ball-park sizing technique and used to confirm that the development effort estimates (software and non-software) created through work breakdown structures and Participative Estimates are in an acceptable range.

Phase-Distribution Estimating

Effort Distribution Models

In this method, an effort distribution ratio, expressed as a percent of total effort, is created for each phase of the project lifecycle. The underlying data can be collected from in-house projects -- if comprehensive and accurate time reporting systems exist -- of from published industry data. Your organization may need to adjust the percentages of different phases to fit the specific needs and environment of your project.

Customer Requirements	15%
Architecture/External Design	25%
Internal Design	20%
Code/Debug	10%
Unit Test	5%
Integration Test	15%
System/Acceptance Test	10%

Sizing A Project Using Effort Distribution Models

1. Develop a comprehensive project description. The description document must be reviewed and approved by the Project Sponsor. Review and approval by "your manager" is not sufficient.
2. Select an appropriate Effort Distribution model for the project. You may need to adjust the phase percentages t suit the specifics of your project
3. Develop a detailed task schedule for the first twenty percent of the project. If detailed tasks are not possible for the entire twenty percent, develop the detailed task schedule for as much of the project as possible and continue with summary tasks (focusing on deliverables) until you reach the twenty percent point.
4. Estimate the effort for each task planned in step 3.
5. Compute the effort and duration estimates for the remaining phases using the Effort Distribution Model selected in step 2. Make sure to document any assumptions regarding resource skill level and the productivity levels expected.
6. Review the estimated effort and duration values for the various phases and see if you need to make any changes in the estimated values to reflect the specifics of your project.
7. Finalize the estimated duration, full-time resources and skill sets, and project management effort for each phase.
8. Develop a phase-based Gantt chart for the various phases of the project. Be careful to accurately show the phase overlaps and sequences (individual phases may need to be broken into sub-phases).
9. Use range values to present the estimates to your manager/customer.

The Project Task Estimates Guidelines

Estimates are produced for each task according to the task definitions and should express the level of effort and skill involved in each task. Task estimates will be used as the basis for all scheduling and budgeting over the life of the project.

The quality of the estimate depends on the planning horizon. Knowledge is about 100% for current events, but reduces for more distant events. Perfect planning and estimating cannot be expected to be done at the start of the project. Estimates are produced several times during the project according to "Phase-Limited Commitment." Task level estimates are produced in detail for "phase-at-hand" work and more generally for work farther out. Estimating effort will be on-going throughout the life of the project, but at a decreasing rate.

The development of reliable work effort estimates begins with an intensive analysis of the above elements affecting the estimating process. Take time to plan and document your estimates -- you'll have to live with them. The resulting estimates may be expressed in a variety of ways ***-- hours, days, weeks,*** or ***staff-months.*** They may be documented at the subtask level, as well as at the task, subphase, phase, and project level, to support future planning and project control.

The development of reliable work effort estimates is a difficult process, not only because it requires intensive analysis and preparation, but also because it's conducted with a high degree of uncertainty about the future course of the project and about the people who will perform each project activity.

1. Consider the nature of the work (complexity, stability, dependencies and technologies involved)
2. Consider the people factors (skills, experience, and environment)
3. Review estimating methods and determine which *combination* of methods will be used
4. Estimate each task using information from task and product descriptions
5. Update the initial project schedule and document all estimates and assumptions made

The Steps Involved in the Estimating Process

1. Assess planning factors
2. Assess personal experience
3. Investigate historical data and available methodologies
4. Review and select an estimating method or combination of methods
5. Apply the estimating method to each task in the work breakdown structure using information from task and product descriptions
6. Document estimates and assumptions on which estimates are based

Documenting these assumptions allows a Project Manager to review each estimate later in the project, and to modify the estimates if initial assumptions prove to be inaccurate or totally invalid.

The list of work effort estimates is primarily used in the development of schedules and budgets. Raw estimates will be translated into dates and dollars for upper management and allow the Project Manager to use them as a project tracking tool.

Let's examine the work effort estimation process in detail. Beginning with the first step in the process -- analyzing the elements involved in estimating, including project planning factors, levels of personal knowledge and experience, historical data, and corporate guidelines and methodologies.

1. Assessing the Planning Factors

The factors which will impact all project planning may be grouped into four main categories:

- Organizational factors
- Nature of the Work factors
- Environment factors
- Personnel factors

Organizational Factors

The organization in which a project takes place may impose standards and restrictions on the work process. These standards and restrictions are considered organizational factors and include such things as:

- The number of hours in the corporate work week
- The number of days in a work month (most organizations use 18-20 days/month)
- Policies regarding overtime
- Mandatory holidays

If the organization operates within the context of a government organization, additional standards or restrictions may be imposed (EEO/AA on resource selection, may not be able to get the necessary skill set). Always verify that there are no hiring freezes in place when you begin to asses the impact of organizational factors on your estimates.

If organizational factors impose tasks performers on the project who lack the skill level assumed in the work estimates, the estimates will have to be modified, along with any schedules based on the estimates -- it's important to identify an organizational factors which may potentially alter work effort estimates, and create your plans accordingly.

Nature of the Work factors

Nature of the work factors or project specific factors include such things as:

- The complexity and stability of user specifications
- Development standards required
- The availability of required developmental technologies
- The complexity of the technologies
- Task complicity
- Tasks requiring excessive functionality, code reuse, performance, or reliability
- Availability of development platforms (e.g., hardware and software)
- Integration of components delivered by different development teams or third party developers

These factors are unique to the project and may directly impact the type and amount of work required. If the developmental technologies required in a project are not available, it's possible that work activity will be delayed awaiting their delivery -- if a project requires a new type of support software, and that software isn't available when the Project Team is ready to begin system testing, all dependent task activity will halt (ZAP-9 acquisition, OS/2e and Presentation Manager). If user specifications require extensive validation of all system transactions, then the tasks relating to the development of validation logic will consume more time than specifications requiring little validation. Likewise if development standards require extensive documentation when a prototype is used, the documentation task will require greater effort than if the standard requirements were minimal.

The complexity of developmental technologies may also affect task performance -- some software development life cycle disciplines require that a higher percentage of time be spent on requirements definition than did the traditional methods (but, conversely, the structured methods may also result in a substantial reduction of time spent on maintenance tasks, as compared to the traditional methods). It's safe to assume that the more complex the set of nature of the work factors, the greater the amount of effort required to complete development tasks.

Environment Factors

Elements of the project environment may also have special impact on task performance, and should therefore be considered prior to estimating the work effort. These elements include such things as:

- The physical surroundings in which tasks are performed
- The location of team members in these surroundings
- The level of support services available to the project

In considering the physical surroundings in which tasks are performed, a Project Manager should determine whether the working conditions will be noisy and crowded, or quiet and private -- one set of conditions can diminish work efficiency, another enhance it. Tom DeMarco and Tim Lister provides an excellent analysis of this in the book, *Peopleware*.

If team members are decentralized, team communication and interaction will require more effort to ensure that tasks requiring a high degree of cooperative effort are accomplished properly.

In reviewing the level of support services available to a project, a Project Manager should determine whether ancillary services such as preparing data and reproducing documents are easily obtained, or whether vital time will be lost in turnaround delays.

Personnel factors

The final set of planning factors which may impact work effort estimation are personnel factors. By these we mean such things as:

- The availability of team personnel
- Internal personnel considerations
- Number of customers to be supported
- Number of project dependencies

Availability of team personnel naturally has a direct bearing on task performance. Work effort estimates assume a particular skill level for adequate completion. If people with the assumed skill level are not available for a project, each task may require proportionately more effort from the less skilled team members.

Related to the availability of team members is a final personnel consideration -- the internal requirement for other activities that may impair team productivity -- if team members are frequently required to attend non-project meetings, training, or other activities, proportionately less time will be devoted to project work, and an additional amount of time will have to be allowed in each task for reorienting to the work after each interruption.

2. Assessing Personal and Organizational Experience

It is often the case that the Project Manager or lead is making the initial estimates. It is important to remember than when tasks are assigned to team members, those team members need to be given an opportunity to review and understand both the task definitions and the estimates. If there is any disagreement, the Project Manager and team member must discuss and reconcile the differences.

After analyzing the factors which affect task performance, a Project Manager must next assess his or her past project experience and determine its effect on the estimating process. In doing so, a

manager must consider personal experience with project job performance, with the project organization and with the estimating process itself.

Personal Experience

In assessing experience with project job performance, a manager must determine his or her level of familiarity with each type of project work activity. A ***low*** level of familiarity will of course, reduce a manager's ability to accurately estimate the required work effort. It's important to identify unfamiliar tasks to that additional measures can be taken to ensure valid work effort estimates in these areas. A ***high*** level of task familiarity may also reduce a manager's ability to accurately estimate the required work effort -- many people unconsciously minimize the difficulty of task which ***they*** perform well. The Project Manager needs to ask of their own estimating experience:

- Did you consistently underestimate or overestimate?
- Did you underestimate on those tasks with which you were most familiar, or in which you are most skilled and overestimate in other areas?
- Did you take into account the other factors we've discussed that affect task performance?

Related to experience with project activities is experience with the ***business functions*** of the proposed system. It's not necessary to be a CPA to manage the development of an automated accounting system -- a high degree of familiarity with the business function involved leads to better understanding and will help you estimate the amount of work effort required to automate it. More effort is required to automate a complex function than to automate a simple one, even though each project may entail the same types of development tasks.

Another feature of the business function that may affect the required amount of work effort is its ***stability***. A business function that is known to be unstable -- that is, one that's subject to constant modification or improvement -- will require more effort than a stable function (order entry).

Organizational Experience

In assessing experience with the organization, a Project Manager should determine whether there is a tendency in the organization to transfer personnel without warning to other projects, whether team members are likely to have additional non-project work assignments, and whether team members are frequently required to participate in meetings, training, and other types of non-project activities.

Another organizational factor to consider when developing work effort estimates is the number of ***productive*** hours in the work day -- do team members typically take extended lunch hours, take extended breaks, or routinely leave the office early?

Yet another organizational factor is the quality of user and IS relations. And still another is the quality and availability of support services such as data preparation and document reproduction. Each of these factors must be taken into account when developing estimates for work effort, since each may extend the actual time required to complete a task.

If your past performance as an estimator was less than optimum, or if you have no experience in this area, you'll see that there are estimating techniques that can improve your chances of success. Your use of these techniques will be significantly better if you first assess your experience to identify and compensate for areas where your experience is weak.

3. Assessing Historical Data and Available Methods

The last area a Project Manager should investigate prior to estimating work effort concerns the availability of historical project data and corporate estimating methods.

By ***historical project data,*** we mean documentation about the development of other information systems. If this documentation is available and you can establish its relevance to your project, you may be able to use historical data about past performance to estimate the work effort for the current

project. Historical data may also be used selectively to devise estimates for standard tasks that occur in all development projects.

4. Review Estimating Methods

By ***corporate estimating methods*** we mean formulas or guidelines for predicting work effort. In the absence of a standard method, your company should at least provide guidelines for preparing and presenting estimates -- they may stipulate that estimates should include the work users will perform on the Project Team, or they may require that estimates reflect ***only*** the efforts of the development staff involved in the project. They may require that estimates be expressed in a standard unit of measure such as days, weeks, or staff-months. They may ask that you estimate professional and clerical effort separately. Each of these requirements should be investigated to prevent needless reworking of estimate formats later in the project.

5. Document Estimates and Assumptions on Which Estimates are Based

Having evaluated all of the elements that impact the preparation of work effort estimates, it's critical to document these elements in ***a list of assumptions.*** This list of assumptions should be used along with task/deliverable information to derive work effort estimate ***that reflect task performance under actual conditions.*** Having a documented list of estimating assumptions will also allow you to review the estimates and readjust them should any of the assumptions prove to be false later in the project.

Next we're ready to look at the next step in work effort estimation -- reviewing and selecting estimating techniques. We'll discuss four basic techniques of preparing work effort estimates. based on your list of assumptions, we'll also discuss methods for selecting an appropriate estimating technique.

Factors to Consider When Estimating

General Factors That Will Affect All Project Phases

- Consulting with or coaching other team members
- Non-project administration
- Non-project education
- Non-project meetings
- Interruptions including phone calls
- Non-project paperwork
- Wait time for meetings, results, etc.
- Switch time, e.g., the time required to “switch” between tasks
- Experience of project personnel with the tools and techniques to be used
- General experience level of project personnel
- Attitude of project personnel (indifferent, enthused, etc.)
- Availability of automated tools
- Availability of clerical support
- Work environment
- Project staffing levels
- Schedule constraints
- Experience of the Project Team with this kind of application
- Length of project

Factors That Will Affect Analysis

- Number of problems/opportunities
- Number of users
- Number of distinct groups within the organization that are concerned with these issues
- New system vs. replacement vs. enhancement
- Familiarity of users with the current system
- Presence of respected users in positions of authority
- Willingness of respected users to make decisions
- Orientation or organization to short term progress vs. long term results
- Thoroughness and accuracy of corporate data model
- User availability
- User commitment
- Perceived value of the system to the organization
- Geographical dispersion of the users
- Familiarity of analysts with the business
- Ability of users to verify abstract models
- Perceived value of a systematic approach

Factors That Will Affect Design

- Thoroughness and accuracy of requirements documents
- Complexity of user interface
- Interfaces with existing applications
- File conversions
- Database architecture

Factors That Will Affect Development

- Logic complexity (mathematical complexity)
- Critical performance requirements
- Interactions and decision points
- Security requirements
- Volatility of processing
- Degree of test coverage required
- Availability and willingness of users to test

Factors That Will Affect Installation

- Geographical dispersion of users
- File conversions

Project Team members can spend their time in several ways. In addition to productive time that will be expended on the project, there will also likely be idle time.

Productive time will be time spend advancing the project, producing units of work and, in general making progress. The amount of productive time needed is a function of:

a) number of "units of work" to be produced and the
b) rate at which units can be produced by team members.

The amount of time that can be spent productively will depend on several factors: i.e., user knowledge and decisiveness, application complexity, systems personnel knowledge, availability of tools, administrative support, working environment, etc.

Idle time will be time during which project personnel are assigned to the project and recording their time against the project but during which no progress is being made, no project output is being produced and the overall effort is standing still. The amount of idle time will be directly affected by the availability of inputs/workproducts needed, the availability of tools and the project direction.

Task-Based Estimating Methods

There are a number of different methods for detailed task estimation. Each has its own strengths and weaknesses. When possible, use more than one method for each task and scrutinize tasks with a large variance between estimates:

- Why are the estimates different?
- Do both methods consider all personnel factors?
- Are the completion criteria clearly defined?
- Can this task be further decomposed?

Later in the Planning phase, when team members are assigned to tasks, the Project Manager or Project Leads will collect refined task and estimate information from the person responsible for doing the work. The refined information will be fed back into the process. This may result in updates to the WBS and the task descriptions. These changes in turn will flow down into a revision of the project schedule.

Assumptions: Throughout the planning process the Project Team is making assumptions about task complexity and scope, availability of resources with appropriate skill levels, and a variety of other project related factors. As assumptions arise during the estimation process they should be carefully documented so that they can be reviewed along with the estimates. If assumptions are later discovered to be inaccurate then the estimates based on them can be reviewed. This can be a critical negotiating point later in the project, but it ***will only work if assumptions are documented.***

Next, we'll see how this information is used to develop work effort estimates using four basic estimating techniques:

- Intuitive
- Participative
- Statistical
- Formula

Each is appropriate with specific sets of information and each has strengths and weaknesses. These techniques can be used individually or in combination. Intuitive technique can be used with the historical technique; the formula method with the Participative technique, etc.

Intuitive Estimating

The intuitive method is perhaps the most widely used estimating technique. As the term suggests, intuitive estimates are derived from personal knowledge and experience. Because it's the least reliable of the techniques, it's most appropriately used as a default technique in the absence of strong historical data or internal estimating guidelines.

- Depends on the estimator's knowledge and experience
- Uses no other aids or tools
- Is quick and cheap
- Is subject to the estimator's experience and ability

Since intuitive estimating is based on the subjective judgment of a single individual -- the estimator -- it's most effective when he or she has long experience with a variety of project work activities. Unfortunately, few people who estimate intuitively have the full range of experience required to estimate well, so some of their estimates are often based on pure assumption. Intuitive estimating does have one advantage -- it's easier and faster than the other techniques we'll discuss. It doesn't require extensive research and analysis, as does the historical method. It doesn't require the involvement and coordination of team effort, as does the Participative method.

If you have to estimate intuitively -- that is, in the absence of historical project data or corporate guidelines -- there are some steps that will improve the quality and consistency of your estimates:

1) Document a set of personal assumptions and standards for estimates
2) Recognize those task areas where you have limited familiarity
3) When you distribute your estimates, acknowledge their intuitive basis and solicit comments and modifications

Participative Estimating

A second technique that may be used in the absence of historical project data or corporate estimating guidelines is Participative estimating. Like intuitive estimating, the Participative technique relies on subjective judgment to estimate work effort, but here the estimates are developed by a group of people with a variety of skills and experience. Because each team member estimates using his or her special competence, or has the estimates validated by other estimators in the group, this technique produces more reliable results than intuitive estimating.

- Uses others to review or generate estimates
- May use sources on the Project Team, other Project Manager, or sources outside the team
- May be done in committee
- Has the advantage of increased accuracy and completeness

It also has disadvantages -- it's liable to be inconsistent because it does involve a variety of estimators with differing methods and perspectives. To minimize this problem a Project Manager can outline a set of standard estimating assumptions and guidelines for the team to follows.

If you choose to use the Participative method, how do you locate competent participants? Your own Project Team may be an ideal source of estimators, since teams are generally composed of people with a variety of complementary skills and work experience. As a result, at least one team members should be familiar with each project task.

Using your Project Team as estimators has certain advantages:

1) It allows team members to become familiar with the requirements of tasks they'll perform later in the project
2) It enhances the validity of each estimate, since the estimates are devised by the same people who will perform the tasks
3) It produces higher commitment to the project, since the team members were consulted and had the opportunity to influence the estimates.

If Project Team members are unfamiliar with certain tasks, you can enlist other coworkers in the estimating process (**subject matter experts**) -- you might ask another Project Manager to devise estimates for tasks with which he or she has had previous experience. Or you might ask non-IS personnel to estimate work effort for tasks with which they are familiar, such as developing documentation or manual procedures

There are several techniques to develop these estimates:

1) ***Delegate*** to each participant those tasks with which they are most familiar

2) Use of a ***committee*** where a groups of experience estimators meets and reviews the project objectives and requirements and collectively develops work effort estimates for each task in the WBS

This technique works best when all group members are well qualified, have a common objective, and are committed to the estimating process. A strong, articulate Project Manager can optimize the committee estimating process by setting meeting agendas, mediating disputes, and soliciting consensus.

Another type of committee estimating, a Project Manager first devises a complete set of intuitive estimates, then assembles a qualified group to review, discuss, and modify these figures.

Statistical Estimating

Both Intuitive and Participative estimating are based on subjective knowledge and experience, and as a result, are subject to the distortions of personal interpretation. By comparison, a third estimating method called the Statistical method derives work effort estimates from objective data about past project and task performance. The Statistical method may be used only where a full set of historical project documentation is available. In many organizations, historical documentation isn't retained. If it does exist, it may not exist in a form to support the Statistical method.

- Depends on documented data from past projects
- Requires data which establishes *comparability* and supports estimates at the *task* level

If this data is missing from the historical files, or if the data doesn't fully match features of the current project, the historical technique may not be applied indiscriminately. Data about the completion of task groups or subphases, even if they represent comparable undertakings, will be of little use in the derivation of current task estimates. Key requirements for statistical estimating are:

1. The historical data must provide sufficient information to establish *comparability* with the current project. To ensure comparability, the historical data must describe:
   a. The technical and organizational environment in which the project took place
   b. The business function being automated
   c. The type and complexity of the system that was developed
   d. The assumptions behind the estimates
2. The historical data must provide information about the completion of *individual tasks*

If historical data is available about comparable tasks, this data may be assumed to represent a reasonable estimate of the amount of work effort required to complete the task. If reasonable comparability can't be established between the historical data and the current project, the historical data may still contribute to the development of your estimates. You may be able to identify tasks common to both the historical and current project which are unaffected by project differences. Administrative tasks might not vary greatly from project to project. If ***limited*** comparability can be established between the historical data and the current project, then the historical data can be used as a ***base estimate*** in the derivation of current work effort.

Let's assume, for instance, that you're replacing an obsolete computer system with one that employs current technologies and incorporates recent business developments. Let's further assume that there is extensive historical data describing the development of the obsolete system. Most of the tasks to be performed in the new system were performed during development of the obsolete system. Based on your assessment of the differences in technology and business functions in the two new systems, you might judge that the current system will require twice the development effort of the obsolete one. You might then double your estimates of the work effort required to perform those tasks for the current project. While this example is oversimplified, it illustrates that historical data may be relevant -- at least in part -- to the current estimating process, even where there are known project difference.

Formula Estimating

The final estimating method is the formula technique. This technique uses a precise model or formula to predict the work effort required to complete each task and task group within the systems development cycle. The formula attempts to isolate and quantify the key factors affecting development of a task deliverable. The formula technique:

- Uses guidelines and formulas to quantify as many factors involved in the tasks as possible
- Depends highly on accurate weights and proper calculation
- May take into account such factors as team experience or project complexity

An example of a formula which transforms data about a particular work activity into a work effort estimate might be to compute the estimated time in hours to complete a specific clerical task -- the transcribing of dictation:

estimated work effort (in hours) = (D/5) * (A) * (6-C)/6)

Where

"D" = the minutes of dictation

"A" = the transcriber's ability in range from 1.2 (below average) to .8 (above average)

"C" = complexity of the transcription (1 is complex, 2 is average, 3 is easy).

The result is expressed as a particular estimating unit -- in this case, hours. According to this formula, 60 minutes of complex dictation by a transcriber of below average ability would require 12 hours to complete.

Each of these factors would be assigned a weight in the formula. Other factors such as program size, computer type and size, portability requirements, programmer's experience and programmer's knowledge would also be weighted and incorporated into the formula. The combined weights of all key factors would yield an estimate of the amount of effort required to develop the program. Additional weighting factors for programming would include:

- The individual program
- The numbers and types of inputs and outputs
- The extent and complexity of necessary calculations
- The number of other program interfaces
- The type of human interface involved

The reliability of estimates derived through the formula technique depends upon the use of realistic weighting factors and upon the proper application of the formula. In order to determine whether accurate weighting factors have been used in the formula, it's important to know whether the weights were derived on the basis of extensive project experience or whether they represent the intuitive judgment of the estimator. It's important to know, too, whether the weights in the formula are based upon current technology. If, for example, a formula makes frequent reference to second generation equipment and technology, it may be obsolete for use in best current practice project environments.

As with historical estimating, a degree of ***comparability*** must be established between the formula and the current project environment. People often believe that because formula estimates are derived mathematically, they're highly accurate. It's important to recognize that this isn't also so -- a number of other factors must be assessed to determine the efficacy of any formula for estimating work effort in your own project environment. Weighting adjustment factors must be developed for all areas of the systems development life cycle and the quality control group should be responsible for developing these.

Final Thoughts on the Estimating Process

Software project estimation is a difficult process. The best way to estimate is to have done the project before, and based on lessons learned, estimate how long it would take to do it again. This is not usually possible -- or particularly useful. The next best way is to follow a repeatable approach that allows you to identify key development areas and based on past experience with similar work, estimate about how long this project should take.

However, any software project is a process of gradual refinement. It begins with a fuzzy picture of what you want to build, and as the project progresses, the end result comes into focus. As the end result becomes clearer, so does the estimate of how much time and resources are required to complete the project This characteristic of software projects makes any estimation method tricky. We must be willing to revise, our original estimates as the project progresses. This process builds on the experience in the project and helps to remove schedule surprises.

Avoid SWAG estimates

Any size estimate should be based on at a minimum, a basic understanding of the business functions that are to be supported.

Use estimates based on past experience

Use past experience (your own and measurements from other projects) to compute the function point count and to estimate resources. It is often a mistake to assume that this project will go much faster than the last.

Involve the whole Project Team

The team as a whole will be involved in creating the product, and needs an opportunity to contribute and buy-in to the original estimates. Ask each team member to estimate parts of the project and then compare estimates and resolve differences. Reach team consensus on the high and low range of the estimates.