FUNCTION POINT SUMMARY


Based on counting number of data structures being used. It assumes data structures number is a good size indicator. Function points are especially useful in business applications; less so in scientific, real-time, and similar algorithm based applications.
Five most important factors are:

1. Number of input types (I) - User inputs that lead to data structure changes (not execution control). Count each input type that has a different format or is treated differently.

2. Number of output types (O) - Same as 1., but for outputs.

3. Number of inquiry types (E) - Inputs that control execution but do not change internal data structures. Examples are menu selection and responses to queries.
4. Number of logical internal files (L) - Internal data generated by the system and used and maintained by the system, such as index files.

5. Number of interfaces (F) - Data that are input to another application or shared with another application.

The number of unadjusted function points (UFP) is:

\[ UFP = 4I + 5O + 4E + 10L + 7F \]
These coefficients are for average complexity. They can be adjusted as:

<table>
<thead>
<tr>
<th>Type</th>
<th>Simple</th>
<th>Average</th>
<th>Complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>O</td>
<td>4</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>E</td>
<td>3</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>L</td>
<td>7</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>F</td>
<td>5</td>
<td>7</td>
<td>10</td>
</tr>
</tbody>
</table>

Now include application characteristics that influence development by assigning a factor from 0 to 5 for each of the following attributes (0 is no influence; 5 is strong influence):
Data communications
Distributed functions
Performance
Heavily used configuration
Transaction rate
On-line data entry
End-user efficiency
On-line update
Complex processing
Reusability
Installation ease
Operational ease
Multiple sites
Facilitate change
Then, the degree of influence (DI) is the sum of the scores for the 14 characteristics. This number is converted to a technical complexity factor (TCF) using the formula

\[ TCF = 0.65 + 0.01DI \]

Finally, the (adjusted) function point measure FP is computed as

\[ FP = UFP \times TCF \]

FP can be converted to lines of code.
Software Estimating Rules of Thumb

from Computer, March 1996, C, Jones (116-118)

Developers should use project estimating tools. People estimates are always too optimistic; automated tool estimates tend to be more conservative--they often predict slightly higher costs and longer schedules than are realized.

Here are some sizing rules of thumb. They are mainly for sanity checks.
Rule 1: One function point = 100 logical source code statements (based on procedural languages like Cobol, C, Fortran; can be much higher for assembly language, much higher for program generators.).

Rule 2: Raising the number of function points to the power 1.15 predicts the approximate page counts for paper documents associated with software projects.

Rule 3: Creeping user requirements will grow at an average rate of 1 percent per month over the entire project schedule.
Rule 4: Raising the function point count to the 1.2 power estimates the approximate number of test cases created. Assume that each test case will be executed about four times during software development.

Rule 5: Raising the number of function points to the 1.25 power predicts the defect potential for new software projects. This is the life-cycle total of errors that must be eliminated. Usually, 85% to 99% of defects will be removed.
Rule 6: Each software inspection, review, or test step will remove 30% of the bugs that are present. This rule implies six to 12 consecutive defect-removal operations to achieve high-quality levels.

Rule 7: raising the number of function points to the 0.4 power predicts the development schedule in calendar months.
Rule 8: Dividing the number of function points by 150 predicts the approximate number of personnel needed to develop an application. This includes programmers, quality assurance personnel, testers, technical writers, database administrators, and project managers.

Rule 9: Dividing the number of function points by 500 predicts the number of maintenance personnel to keep an application updated (i.e., need one maintenance programmer per 500 function points of software operations). A corollary is that raising the function point count to the power 0.25 predicts the number of years the application will stay in use.
Rule 10: Multiply software development schedules by number of personnel to predict the approximate number of staff months of effort. For example, a project of 1000 function points has a schedule of about 16 months (1000^{0.4}, from Rule 7). Using Rule 8, $1000/150 = 6.6$ people. 16 times 6.6 = 106 staff months.