**Software Size Estimation**

Table of Contents

[Lines of Code 3](#_Toc83392905)

[Token Count 4](#_Toc83392906)

[Function Point Analysis 5](#_Toc83392907)

[Counting Boundary 6](#_Toc83392908)

[Data Functions and Transactions Functions 6](#_Toc83392909)

[Unadjusted Function Point Count 7](#_Toc83392910)

[Adjusted Function Point 8](#_Toc83392911)

Estimating the size of the software is important for various purposes during the software development process, such as determining how many engineers will be required or what the cost of the project might be. However, it can be difficult to measure.

There are several approaches to determining software size:

* Lines of Code (LOC)
* Token Count (Halstead Product Metrics)
* Function Point Analysis (FPA)

## Lines of Code

The LOC approach quite literally counts the **number of lines of code** in the project. When doing this, **comments and blank lines are ignored**. Often, the value is presented as **thousands of lines of code** (KLOC).

The LOC method allows us to measure the performance and contribution of individual engineers in a project.

## Token Count

The LOC approach is inconsistent, because it assumes that all code is the same. However, a hundred lines of complex code is not the same as a hundred lines of simple code. Thus, LOC can give us misleading results.

To get around this problem, a person named Halstead stated that any software program can be measured by counting the number of **operators** and **operands**.

Using these two numbers, he proposed three values:

* Program Vocabulary
* Program Length
* Program Volume

The **program vocabulary** () is the number of **unique operators** () plus the number of **unique operands** ().

The **program length** () is the **total** number of **operators** () plus the **total** number of **operands** () appearing in the program.

The **program volume** () is the size of the program and is given by

## Function Point Analysis

FPA is based on the idea that software size should be measured according to the **functionalities specified by the user**. FPA is a standardized methodology for measuring various functions of a software from the user’s point of view. The size of the software is measured in **function points**.

The process of calculating function points goes as follows:

1. **Determine the type of project** for which the function point count is to be calculated.
2. Identify the **counting scope** and the **application boundary**.
3. Identify **data functions** and their **complexity**.
4. Identify **transactional functions** and their **complexity**.
5. Determine the **unadjusted function point count (UFP)**.
6. Determine the **value adjustment factor (VAF)**, which is based on 14 **general system characteristics (GSCs)**.
7. Calculate the **adjusted function point count (AFP)**.

There are lots of unknown words here, so let’s go over the steps in a little more detail.

### Counting Boundary

Once the project type is identified, the first thing we need to do is identify the **counting boundary**.

The counting boundary defines the scope of the project, thus telling us which functions are part of the project and which functions are not. For example, if we create an application that allows users to buy products and we use a function from an external payment gateway to actually allow users to pay, that function is outside the counting boundary. We will not be counting those functions.

### Data Functions and Transactions Functions

Data functions relate to data that is stored somewhere. They are of two types, **Internal Logical Files (ILF)** and **External Interface Files (EIF)**.

An ILF is a set of **data present within the system**. The majority of the data will be interrelated and captured via the inputs received from external sources.

An EIF is a set of data that the application will use or reference, but the data itself is **not maintained by the application**.

Transactional functions relate to data that is being transferred from one place to another. They are of three types, **External Inputs (EI)**, **External Outputs (EO)** and **External Inquiries (EQ)**.

EI are **end-user actions**, such as the act of entering login details or clicking a button.

EO are **outputs provided by the system** to the end-users, such as information being printed to the display.

EQ are **queries from the end-user**, such as a search query. Essentially, the user is able to ‘search’ for an answer.

### Unadjusted Function Point Count

Based on the complexity of each of the function types above, a **weight is assigned**.

|  |  |  |  |
| --- | --- | --- | --- |
| **Measurement Parameter** | **Low** | **Average** | **High** |
| External Inputs (EI) | 7 | 10 | 15 |
| External Outputs (EO) | 5 | 7 | 10 |
| External Inquiries (EQ) | 3 | 4 | 6 |
| Internal Files (ILF) | 4 | 5 | 7 |
| External Interfaces (EIF) | 3 | 4 | 6 |

We **count** the number of functions of each type, and **multiply** the number by the appropriate **weight** from the table above. The total is the **UFP**.

The question is, how do we know what complexity to assign to a particular function. There are details to this, but for now, we will assume that we already know the complexity.

### Adjusted Function Point

Next, we need to find the VAF. Every system has 14 GSCs, listed below.

|  |  |
| --- | --- |
| **Factor** | **Meaning** |
|  | Data Communications |
|  | Performance |
|  | Transaction Rate |
|  | End-User Efficiency |
|  | Complex Processing |
|  | Installation Ease |
|  | Multiple Sites |
|  | Distributed Data Processing |
|  | Heavily Used Configuration |
|  | Online Data Entry |
|  | Online Update |
|  | Reusability |
|  | Operational Ease |
|  | Facilitate Change |

For each characteristic, we assign a **Degree of Influence (DI)**, which is a value between and , depending on how much influence that characteristic has on the project. The degrees are mapped as follows:

: No Influence : Incidental : Moderate : Average

: Significant : Essential

Thus, we can calculate the **Total Degree of Influence (TDI)** by summing up all the DI values for the 14 characteristics. From here, we can calculate the **VAF**.

And finally, we can calculate the **AFP**.

Example:

Consider a project where , , , , . Consider that all weighing factors are average, the system has significant performance, average end-user efficiency, moderate distributed data processing and significant data communication. All other GSCs are incidental.