Improving the Accuracy of Function Points Counts

Amit Javadekar

Abstract

The Function Point (FP) Analysis model was invented by Allan Albrecht in 1979 as an alternative to the "Lines of Code" model for sizing software. Since then it has arguably become the most popular method of software sizing and is still relevant almost 30 years after it was invented.

Practitioners of Function Points will agree that to gain expertise in Function Points counting one needs to take part in numerous Function Point counting exercises and learn to avoid some common mistakes that less experienced Function Point counters tend to make. This paper highlights some of the common mistakes that novice practitioners make as they learn the ropes of Function Point counting. It assumes that the reader is conversant with Function Points and has a fair understanding of the various terminologies involved in Function Point Analysis.

A list of abbreviations is included at the end of this article.

Common mistakes in Function Points counting

1. Counting code data (including static data) as logical files

The IFPUG (International Function Point User Group) Counting Practice Manual (CPM) groups application data into 3 types:

- Business Data
- Reference Data
- Code Data

Business data is mandatory for the functioning of the business while Reference data is generally used to support the functioning of the business. Code data is data that has been introduced into the application to meet technical requirements and does not have any business significance. Code data typically includes Code-Description pairs, static drop down lists used for error free data entry as well as other static data that does not change very often. As per IFPUG (International Function Point User Group) Function Points counting rules, code data is not to be counted as Data Functions. However inexperienced Function Point counters do not pay adequate attention to this rule and end up with wrong counts.

To understand code data better let us look at a Library Management System (LMS). The Book Information file has the following fields:

- ISBN#
- Title
- Publisher
- Author
- Price
- Binding

This forms the core of the LMS and is mandatory for the functioning of the application. Thus it is Business Data and counted as a Data Function. The LMS may also use Currency Exchange Rate Information to convert a book's Price from one currency to another. This Exchange Rate Information supports the business and is treated as Reference Data. It is also counted as a Data Function.

The LMS may have another data set called Review Rating Information that contains two fields:

- Star Code
- Star Description

This file is used to denote how the book was rated by various reviewers and contains data such as:

Star Code	Star Description
0	No Star
1	One Star
2	Two Stars
3	Three Stars
4	Four Stars
5	Five Stars

From the LMS user's perspective he/she can relate to and understand the meaning of Star Description but Star Code has no business significance. Wherever Star Code is used by the application it can easily be replaced with values from the corresponding Star Description without changing the meaning of the data. Such a file represents Code Data and should not be counted as a Data Function.

2. Not Counting Implicit External Queries

Any transaction that modifies or deletes existing application data is treated as an EI as per Function Points counting rules since these transactions maintain the application's ILFs (Internal Logical File). It is normal practice for a user to view the existing data before it is modified or deleted. This viewing of data involves reading the existing data from the applications Data Functions (ILFs or EIFs) and displaying it to the user. This operation satisfies the counting rules of an EQ and hence needs to be classified as an EQ. This requirement (of viewing existing data before it is modified or deleted) is never explicitly stated by the user. It is an implicit requirement that needs to be taken into consideration as functionality requested by and provided to the user by the application. Hence a "modify" or a "delete" transaction (almost) always gives rise to one additional implicit EQ.

As an example, suppose that information about a book in the Library Management System needs to be modified as the Price of the book has changed. In this case the "Modify Book Information" transaction would be treated as an EI. Prior to modifying the Price, the user would be shown the existing details about the book which the user can verify before updating the Price. This transaction which displays the existing book information would be treated as an implicit EQ and counted as additional functionality.

3. Not Counting Triggers and Responses as DETs

All transactions in an application are fired off in response to a trigger. This trigger may either come from a user interacting with the application through a GUI based screen (e.g. pressing an OK or Save button) or from the application itself (e.g. from the system clock) or from another application (e.g. a file received from another application triggers a program that processes the file). Similarly all (good) applications provide responses (messages) back to the triggering body (a human user or another application) to inform them of any errors that might occur during transaction processing. Confirmation messages are also sent back to the triggering body indicating successful/unsuccessful completion of the transaction.

Function Points counting rules require all triggers of a transaction (a transaction may have more than one trigger) to be counted as one DET for that transaction. Similarly all error, warning and confirmation messages are to be counted as one DET for that transaction. It is very common to find novice Function Point counters not taking this into consideration. They either do not count triggers as a DET and messages as a DET (thereby always losing 2 DETs) or end up counting each trigger and each message as a DET each (thereby inflating the number of DETs and assigning wrong complexities to their transactions).

In the "Modify Book Information" transaction described above, once the new Price is entered by the user, he/she will save the changes by clicking on the "Save" or "OK" button. The same can also be achieved through a menu option File → Save. In this case although there are 2 triggers to this transaction they would contribute only 1 DET.

4. Including Notification Messages as DETs (clubbing them with error and confirmation messages)

As discussed above, all error/warning and confirmation messages contribute one DET to a transaction's Function Point count. Another class of messages that is seen quite often in applications is the notification message. Examples of notification messages are ticker tapes running on the screen (e.g. on a financial portal there may be a ticker displaying share prices on the NYSE), automated e-mails generated in response to the occurrence of an event (e.g. an e-mail from the LMS application informing a library member that the lending period for a book is over and that the book needs to be returned to the library).

These types of messages are different from error/warning or confirmation messages discussed above. They are typically transactions by themselves and are not actually a part of any other transaction. As such notification messages usually satisfy the counting rules for EOs or EQs and should be classified as such. Very often Function Point counters believe that all messages contribute one DET to a transaction. This is not true. While all error/warning and confirmation messages contribute one DET to the transaction from where they originate, a notification message is an independent transaction with its own associated DETs and FTRs.

5. Counting DETs that do not cross the Application Boundary (in case of transaction functions)

Most transactions have two sides, an Input side and an Output side. The Input side is used by the user (human or another application or system) to feed data to the transaction. In case of an EI this data is usually used to update an ILF while in case of an EO/EQ this data is used as selection criteria to retrieve data that the user wants to

see. The Output side is used to present information to the user. In case of an EO/EQ it is the data that the user wishes to see, while in case of an EI it may contain error or confirmation messages, system generated values etc that are passed back to the user.

As part of their processing logic, transactions also use data that is not fed by the user on the Input side. Typically this data already exists in the system on ILFs or EIFs. Transactions read this data as part of their internal processing and use it to produce the Output. If this (internally used) data does not appear on either the Input or Output side then it means that this data has not crossed the application boundary. Such data that does not cross the boundary but is used internally by the transaction as part of its processing logic should not be counted as DETs of the transaction as per Function Points counting rules.

As an example let us assume that the Library Management System has a built-in query feature that allows the librarian to find out the total value of the books in the library in US$ on a given date. The processing logic of such a query would involve totalling the prices of all books (records) stored in the Book Information file and displaying the total. Although the Price field is used by the query, since it is not displayed to the user i.e. it does not cross the application boundary it is not counted as a DET of this transaction. Inexperienced Function Point counters count all data elements used by the transaction – Input, Output as well as those used internally - as DETs and thus violate an important rule of Function Point counting.

6. Having ILFs that do not have Associated EIs

In the Function Points parlance an ILF is an Internal Logical File that is maintained by the application. Maintainability refers to the capacity to add new data to the file or modify existing data or delete existing data from the file. Thus for a file to be classified as an ILF the application must have transactions that either add data or modify data or delete existing data from this file. Transactions that provide these functionalities are known as External Input(s). Hence for every ILF there must be some EIs associated with it that maintain the file.

As part of a Function Points review it is advisable to check that ILFs and EIs have been properly associated with one another.

7. Not counting Drop Down Lists as EQ/EO

Drop down lists (also called Combo Boxes, List of Values, etc.) are provided to make an application user friendly, easy to use and also to reduce errors during data entry. Drop down lists typically read data from a file and display it to the user whenever the user requests to see the list. Thus they meet the counting rules of an EQ or an EO and hence should be classified as EQs or EOs. Their DETs include the data elements that are displayed as part of the list, the trigger that the user provides to see the list and any error message that might be displayed e.g. if no data is available to be shown in the list.

It is important to bear in mind the rules related to Code Data discussed in Point 1 above while classifying drop down list boxes as EO/EQ.

It is very easy to miss out on functionality such as that provided by drop down list boxes while sizing an application. As these are minor transactions Function Point counters do not pay adequate attention to them. While missing a couple of drop downs may not impact the size to a large extent it is a bad habit to get into. The

cumulative effect of missing small functionalities may have a large impact on the Function Point count.

8. Not including Conversion Functionality

Conversion functionality is provided by applications as a onetime activity to convert (migrate) data present in existing data stores to new data stores. This is usually applicable in situations where a new software application replaces an existing application. Users would need to move their existing data from the old application to the new application. This onetime activity is counted as functionality provided by the application and the conversion routines are treated as EIs having their own DETs and FTRs. It is important to count the size of this conversion functionality (in Conversion Function Points) and add it to the size of the project. Very often this aspect of an application tends to get ignored resulting in a smaller size than is actually true.

9. Counting subsets of data as Independent Logical Data Files instead of RETs

Record Element Type (RET) represents logical subgroups of data within a data function. A common dilemma faced by Function Point counters (both experienced and new) is whether the subgroup of data should be treated as a separate logical file or not. One way of overcoming this dilemma is to test whether the subgroup of data has business relevance independent of the main data set. For this the 'Delete' transaction is found to be most useful. If the subgroup of data has to be deleted whenever the main data set is deleted then it (the subgroup) does not have any independent business relevance and in such cases the subgroup would be considered a RET of the data function (main data set). On the other hand whenever the subgroup of data continues to exist even if the main data set is deleted then the subgroup has independent business relevance and can be considered as an independent data function rather than a RET.

Conclusion

Maturity in Function Point counting is gained through practice. While it is inevitable that mistakes will be made as practitioners graduate from being novices to experts it is important that one learns from these mistakes and avoids them in future. This paper has been written with the intention of highlighting the common mistakes that the author has encountered while teaching and reviewing Function Point counting practices. True maturity is all about learning from others' mistakes. It is hoped that practitioners will learn from this paper and ensure that they do not repeat these mistakes in future.

About the Author

Amit Javadekar has 14 years of IT experience. He has held numerous technical and managerial positions and currently manages the Estimation Center of Excellence (ESTEEM) at Infosys Technologies. He holds a Masters degree in Computer Science and is also a PMP.

Project Perfect is a project management software consulting and training organisation based in Sydney Australia. Their focus is to provide organisations with the project infrastructure they need to successfully manage projects.

About Project Perfect

Project Perfect sell "Project Administrator" software, which is a tool to assist organisations better manage project risks, issues, budgets, scope, documentation planning and scheduling. They also created a technique for gathering requirements called "Method H"™, and sell software to support the technique. For more information on Project tools or Project Management visit www.projectperfect.com.au

References

Function Point Counting Practices Manual Release 4.2 by the International Function Point Users Group (IFPUG)

Abbreviations

IFPUG	International Function Point Users Group
ILF	Internal Logical File
EIF	External Interface File
EI	External Input
EO	External Output
EQ	External Query
DET	Data Element Type
RET	Record Element Type
FTR	File Type Referenced