

# Counting Function Points for Websites

Massimiliano Conte, Roberto Meli, Luca Santillo

## Abstract

*Measuring the functional size of a Web System is a hard task, because of extreme flexibility, autonomy, and hyper-textual nature of data and functions navigation. This unstructured and extempore interaction makes it difficult to identify a unique user view, a consistent application boundary, and the exact logical data groups and elementary processes that make up a website.*

*This article introduces and discusses a self-consistent set of counting guidelines, to help practitioners to interpret the IFPUG counting rules in the web environment and to ensure comparable measurement results among counters.*

## 1. Introduction

Functional Size Measurement methods are designed in order to provide an objective measure of the functionality that a software system provides to its users [1]. For instance, the IFPUG Function Point method accomplishes this task by measuring what an application allow the user to do and to get, in terms of data functions (information structures) and transactional functions (information processing) [2].

Websites, on the other hand, are implemented in order to provide the users with information that they are interested to, in a specific domain, and eventually to interact with that information. Therefore, in implementing web systems, the focus is set on what information is to be published, how this information can be rendered to be attractive to thee readers, and how to link information pieces (pages) to make it easier and quicker to find them. The real goal is not (only) to provide information to the users, but rather to provide *usable* information to them. In this perspective, the sole functional size of a web system easily turns to be a not-so-interesting aspect in assessing the overall value (and /or in estimating the corresponding implementation effort). On the other hand, the market and the community do need to apply function point to web systems, too, in order to have a single measure to handle and manage large generalized software environments (as in the context of a large agreements, for instance). Such need is even stronger where function points are used as a input to effort estimation or cost estimation models.

Applying Function Point Analysis to a website presents several difficulties. Although Function Point can be measured, in principle, for any given software, when adopting the IFPUG counting rules for a web system the measurer has to address significant conceptual issues. These issues mainly concern a clear and non-ambiguous identification of the “user view”, from whom all other counting elements are affected: the application boundary, data functions and transactional functions. Such issues require the development of a set of guidelines on how to apply the counting rules in practice for counting web systems. Those guidelines would make counts provided by different measurers more consistent and homogeneous.

## 2. Web Systems

Web Systems can be defined as a logically related set of interactive software functions, which fulfil business requirements in form of internet, intranet or extranet application.

Examples include: showcases, newsletters, bulletin board, collections of downloadable publications, surveys, discussion forums, ordering systems, search engines, etc.

Web Systems (regardless of the type – internet, intranet, extranet) fall into two distinct categories, based on their base characteristics, although the two classes are more and more interrelated nowadays: “websites” and “web applications”.

### **Websites**

Most websites can be considered as electronic publishing, involving text, audio and video data, put online for browsing and based mainly on the hypertext paradigm, or the hypermedia. For this kind of software, the user is fully autonomous when choosing browsing paths, and (s)he is not driven in any procedural way. Hence, it is not possible to identify specific transactions, in a classical sense. Moreover, data flow mainly in one direction, from server to client, except obviously for the browsing commands. Web pages, in this domain, are mainly static, although sometimes they are dynamically built by the server, based on some user choices or information specification (logs, cookies, and the like).

### **Web Applications**

Over time, true structured applications are spreading and gaining relevance in the web domain. The paradigm for such web applications is the classical structured information system, based on a central unit communicating with one or more peripheral terminals for data entry or data retrieval. Examples are: e-commerce systems (internet), task/resources data collection (intranet), warehouse data retrieval (extranet), etc.

Websites and web applications, in real-world practice, are usually integrated, so that both of them present a browsable (navigation) component, and a operational (transaction) component. Examples are companies’ sites who allow the user to acquire some products or services online: such sites have a browsable component (company’s presentation) and an operational component (product catalogue and purchasing system).

In this work, we could generally speak of “web systems”, although the proposed guidelines concern mainly the navigation component (the transactional component is measured according to the standard counting rules, and does not require specific guidelines for application; thus, websites are mainly concerned by the proposed work).

## **3. Function Point Analysis (IFPUG)**

Function Point Analysis is a software functional size measurement method, i.e. a way to quantify software size by means of functional user requirements measurement.

Function Points quantify software functionality in terms of data functions and transactional functions. Both types of functions are identified and classified based on the end user perception of the software application being measured. Comprehension of the user perspective is thus essential to perform a function point count correctly. Deviations between counts provided by different practitioners are typically due to different interpretations of the same requirements, or of what the user view is, by the practitioners.

Once the user view is declared, FPA requires to accomplish the following counting steps:

- Determine Type of Count
- Identify Counting Scope and Application Boundary
- Count Data Functions
  - Identify Logical Files
  - Classify logical files as ILF (Internal Logical Files) or EIF (External Interface Files)
  - Count internally-defined items (Data Elements and Record Elements – DET and RET) per each logical file

- Assign an Unadjusted value to each logical file (Tab. 1)
- Count Transactional Functions
  - Identify Elementary Processes
  - Classify elementary processes as EI (External Inputs), EO (External Outputs), or EQ (External inQuiries)
  - Count internally-defined items (Data Elements and File References – DET and FTR) per each elementary process
  - Assign an Unadjusted value to each logical file (Tab. 1)
- Determine Value Adjustment Factor
- Calculate Adjusted Function Point Count

Table 1. IFPUG Functional Complexity and Unadjusted Function Points [2].

	<b>Low Complexity</b>	<b>Average Complexity</b>	<b>High Complexity</b>
<b>ILF</b>	7	10	15
<b>EIF</b>	5	7	10
<b>EI</b>	3	4	6
<b>EO</b>	4	5	7
<b>EQ</b>	3	4	6

#### 4. Preliminary aspects for measuring Web Systems

The IFPUG documentation does not provide specific guidelines or examples regarding web systems counting. Therefore, it is adequate to define some guidelines on how to interpret the standard counting definitions and rules in a coherent way for web systems. Such guidelines should be expressed in general, but should not be taken as a replacement for the standard rules. Every counting item should therefore be validated against the standard rules.

In order to fully understand the guidelines, it is required to consider some preliminary considerations for their derivation.

##### 4.1. Outline of Web Systems building approaches

As web systems are becoming more complex, large or relevant to institutional representatives, their development process evolved from simple, direct publishing activity to separate design tasks for contents, layout structures and interfaces. Such modern process typically requires different roles and analysis and building phases: while the structures' design and building, along with interfaces, can be considered as analogous to traditional software development processes, contents analysis and issuing tasks are more of a publishing kind.

The mix of structure and contents also demands for distinguishing, across all the web system life cycle steps following its first implementation, contents maintenance (publishing) from essential structure updates, underlying the contents (enhancement). Depending on the purpose of the count, applying function point analysis to a web system (website) may require to separately consider structure and contents, and to differentiate the corresponding implementation or update tasks. In order to accomplish this differentiation, we need to determine the correct abstraction level to adopt for identifying the software functions to be measured.

## 4.2. Abstraction levels and functions identification

A fundamental principle to be applied to model, describe and analyse a software system in an effective way, is the abstraction principle, that is the principle of neglecting aspects which are not relevant for the purposes of the analysis, while focusing on the most relevant ones. Nowadays, systems complexity easily overcomes the capability of the human mind. When applying the abstraction principle, we recognize that the subject of the analysis is too complex – rather than trying and modelling that system in all its aspects and details, we do select a subset of it, or of its aspects. We do not forget that other aspects exist, but we choose not to look at those, at a given stage. Such technique is indeed significant to manage complexity and to exchange effective results with the user and the other stakeholders, across any system lifecycle step, at an appropriate detail level.

For function point analysis, it is essential to determine the right abstraction level to analyse functional user requirements, in order to identify files and transactions consistently with respect to the counting purposes. Therefore, we need to provide an explicit definition of abstraction level; assuming as a starting point the following definitions:

- *model*: a description or analogy used to help visualize a concept that cannot be directly observed [3];
- *abstraction*: the process of suppressing irrelevant detail to establish a simplified model, or the result of that process [4];
- *point of view (generic), or viewpoint*: a form of abstraction achieved using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within a system [3];

we further define (in order to apply the guidelines subsequently reported):

- *abstraction level*: a specific set of criteria or principia for coherent identification (aggregation or decomposition) of elements being analysed (e.g., for our scope, blocks of a website, made of webpages, or blocks within each webpage).


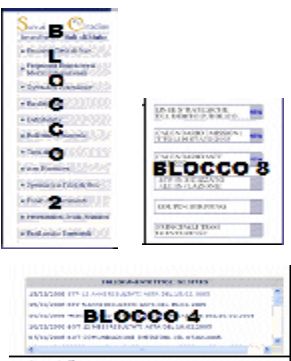

Based on the purpose of the count, that is the explicit reason why the size of a web system is required for (e.g., effort, cost or duration estimation), a specific abstraction level must be established before assessing and measuring the system's functionality, for instance whether a piece of functionality is to be measured "as one", or "only once", or many time as it is used with differences in structure or semantically separate contents..

We introduce the two most significant abstraction levels that should be adopted for alternative measures of a web system (website). Such abstraction levels will be further mentioned in subsequent sections of this work, to explain certain practical choices suggested for appropriate identification of data functions and transactional functions in function point counting. The proposed abstraction levels are denoted as *content-driven* and *structure-driven*, based on the fundamental criterion for their classification (Table 2):

- actual semantically-identified contents,
- recursive structure, regardless of (final) contents.

Typically, a web project that is inclusive from the very beginning of providing specific contents (along with the preliminary design and implementation of the structure of the site and of all its pages) will be measured at the *content-driven* abstraction level, while web (site) projects for which the client is requiring only a set of reusable structure blocks, to be filled with unknown contents at a later stage, after the project completion and possibly by resources other than the project team, may be measured only at the *structure-driven* abstraction level (contents unknown at the measurement stage, to distinguish blocks that are under any other aspect identical).

Table 2. Content-driven and structure-driven abstraction levels.

		Webpage “X”	
<p>At a first abstraction level, we do not distinguish between contents and structure: functions are identified based on the analysis of each webpage per se and on the semantic of contents carried by the blocks within the page (“semantic blocks”, or <i>content-driven</i>).</p>			
		<b>Structure</b>	<b>Contents</b>
<p>At a higher abstraction level, we separate blocks within a page; such blocks can be reused by the designer on different pages regardless of the contents (which could be even not established at design time) (<i>structure-driven</i>); in this case, contents are provided only at runtime, as publishing task (such task is not part of the development project, and cit cannot help in identifying semantically the blocks, which are seen only once per block type, or layout).</p>		 <p>...</p> <p>...</p>	 <p>...</p> <p>...</p>

By IFPUG definition, the purpose of a function point count is to provide an answer to a business problem. That means that, based on the specific business problem one has to face and solve, the measurement approach requires to be tailored in terms – for instance – of the abstraction level to be adopted for counting websites. Although it is trivial, it is worth noting that measures (counts) obtained by different abstraction levels must not be added or mixed in any other way. In any case, we do need to specify clearly which abstraction level is adopted, when performing each count.

## 5. Function Point counting guidelines for Web Systems

### 5.1. User View

Based on the definition of “user view” provided by the IFPUG glossary, we state that a web system user view is related to all the business requirements that affect what is perceived by the end user of the web system (website); the end user is who browse on the website to look at, search, possibly update information on that site. Therefore, several roles are involved in building up the user view, by means of the requirements subsets they help to specify:

- User:
  - the “browser” (user), who looks for the information;
  - the customer (when expressing requirements regarding the web system).

- Developer:
  - resources who implement the system (provider, analysts, development team, etc.);
  - resources who manage the system (webmaster);
- Publisher:
  - any role devoted to define and publish the system contents (content providers);

It is to be noted that for IFPUG FPA; differently from other measurement methods, the user view must be identified as unique and fixed. For web systems, though, it is required to determine the appropriate abstraction level before commencing the measurement, based on the purpose of the measurement. The user view and the abstraction level are different concepts and should not be confused as the same concept.

## 5.2. Determine the Type of Count

No specific guideline is needed about establishing development or baseline (application) count types. For enhancement projects of web systems, it is not always easy to distinguish enhancement interventions from content management (publishing) activities, that is from tasks aimed to update the information carried on by the systems. This occurs because many of the modern languages used for web systems do not separate the data component (published information) from the transactional component (presentation/publishing means).

For instance, with HTML one has to operate on the same physical object (HTML “page”) in order to both changing contents or the structure.

From this perspective, it is worth recalling that function point analysis recognizes a project as an enhancement one only if functionality (functions) is added, changed, or deleted, with respect to a previous release for the system being measured, where “changed” basically denotes adding/removing data element types to files or transactions, or modifying the processing logic of transactions.

In no way function point analysis can support the measurement of publishing tasks, even when organized as a project; such kind of activity must then be evaluated by means of other metrics (e.g. directly linked to resource effort).

## 5.3. Identify the Counting Scope

The counting scope can only be identified when the purpose of the count has been clearly stated. As stated above, indeed, the purpose determines the abstraction level to be adopted, and the latter points to which functions are to be considered in the count.

Nowadays, typically web pages are created by means of filling one or more templates with appropriate contents. Therefore, once the template(s) are established, the remainder of a web project is basically publishing in nature.

Nonetheless, if the purpose of the count is to provide a function point size of exactly what the end user is provided with, after the publishing completion, a content-driven abstraction level is appropriate, where functions used to retrieve (and eventually update) existing information are taken into account semantically per content, regardless of the underlying structures that implement the templates.

On the other hand, *publishers* are provided with the templates by a true development team. Thus, if the purpose of the count is to estimate or to assess the project factors as time, effort and costs, it is more appropriate to adopt a *structure-driven* abstraction level, where templates (and possibly default data types) are taken into account, regardless of whatever content will be put in practice in the structures, in the future.

The two levels of abstraction, introduced in Table 2, require specific guidelines for the identification of data functions and transactional functions. The guidelines are therefore reported as two subgroups: section 5.5 introduces the counting guidelines at the first

abstraction level (*content-driven*), while section **Errore. L'origine riferimento non è stata trovata.** introduces the counting guidelines at the second abstraction level (*structure-driven*).

#### **5.4. Identify the Application Boundary**

In a “web-like” environment as the internet is, determine boundaries between web systems and sites is truly a hard task. Given the facility to jump from one page to another, by means of hyperlinks, the user feels as if all the web is a mare magnum of information, with changing structure. On the other hand, the point of view of those designing and publishing web sites is linked to specific communication goals and boundaries, e.g. to analyse the best paths to access information on a specific system, or a specific, delimited subset of sites. From this perspective, information semantics and business goals should help to locate boundaries and to determine self-consistent “applications”.

Generally speaking, we should avoid a “technical approach”, where a set of web pages is seen as a collection of text portions, pictures, and other physical components. We recall here that function point analysis is aimed to assign a functional size to software based on the amount of functionality that the software is providing to the user. Even a multimedia-rich publication could be seen as “functionally structured”, based on the unique “information services” provided to its users. Subjective judgements in this field should not be considered unavoidable, at least at the same degree they are present even for traditional software analysis. What really matters here is decisions about the boundaries are documented e possibly kept stable as the systems evolve.

The base criterion to consider is that the boundary identification must be always linked to the user view, therefore:

- a set of web pages included in a project/intervention does not necessarily belong to a unique web system boundary;
- it doesn't matter where information is physically stored (client, server X, server Y), but rather whether it belongs to a an homogeneous and autonomous functional set;
- functions should be grouped by boundary according to their logical and operational affinity.

Physical features that can help in identifying the boundary, since they descend from implementation decisions made according to the external user view, are:

- *home page*: a set of web pages pointing back to the same home page typically belongs to the same system;
- *domain (main URL)*: a set of web pages with different URLs (addresses) belong to separate web systems.

Finally, search functionality or other functionality which are used by the web system being measured, but provided by other web systems or application, should not be included in the given web system count.

#### **5.5. Counting at the *content-driven* abstraction level**

When analysing a website from the *content-driven* abstraction level, a page is made of specific information blocks, semantically and clearly differentiated by their contents. At this abstraction level, the count must be based on the differences between the information shown to the end user.

##### *5.5.1. Count Data Functions*

Most web systems can be compared to electronic publishing solutions made available online for information sharing and browsing. When browsing the site, the user is autonomous in

his/her path choices. Such autonomy makes it harder to identify user-recognizable data logical files. In practice, the same data could be grouped or mixed differently, in a functional perspective, based on different usage and retrieving paths by the users. On the other hand, also, the data structures do not help us in this identification task, since the links between data portions are implemented as hyperlinks, whose design is best driven by usability or other quality-related aspects, rather than by functional aspects.

Identifying logical files from the published information, in the browsable (navigation ) component, must be driven by its semantics and user-perceived structure. Information provided to the user may be regarded as being of two kinds: texts (textual items) and resources. The difference between the two is not so sharp, since the distinction itself is based on semantics, regardless of any physical format chosen to implement them (HTML code, PDF, Flash, dynamic DB, etc.). We report some descriptions to help in distinguishing them.

**Texts** (textual items) are characterized by the fact that the same text should not be identified with more than one instance: each text item carries a different subject, even with multiple “pages”, and no other item shows the same subject. Each “text” is perceived by the user as *per se* and unique. The text information cannot be decomposed into separate portions without losing significance and integrity. The user can “read”, search, print texts by means of standard browser features – rarely (except in the case of blogs) he/she can update the contents. Examples are: company’s description, “who we are”, mission statement, etc.

**Resources**, on the other hand, usually present more than one occurrence; multiple instances of the same resource do share the same structure and usually have user-significant separate attributes (data element types). The user can search by criteria, filter, and eventually update (some of) the attributes of a given resource., etc. A clear example is a “catalogue of products”.

For instance, many public administration websites are created with the primary intent to provide the users (e.g. citizens) with specific information items and services, in terms of laws, forms, file workflow checking, and so on. Besides this (structured) information, a PA websites usually reports also some generic presentation data – “mission”, institutional tasks, organization structure, and the like. While the first sample is clearly provided as a set of resources to the users, the latter are more generic informative texts.

The main guideline for data functions suggests to identify and count:

- 1 unique logical file for *all* the textual data in the web system being measured (no structure, no multiple instances),
- a logical file per each resource autonomously required and browsable.

### 5.5.2. Count Transactional Functions

Web pages, as stated before, are typically created based on one or more templates, designed and provided as a step of the web project. A template is a graphical/layout model of a page, including a set of blocks where objects as logos, pictures, menus, text items, link, buttons, and the like may be placed at a later stage. Figure 1 shows a template example.

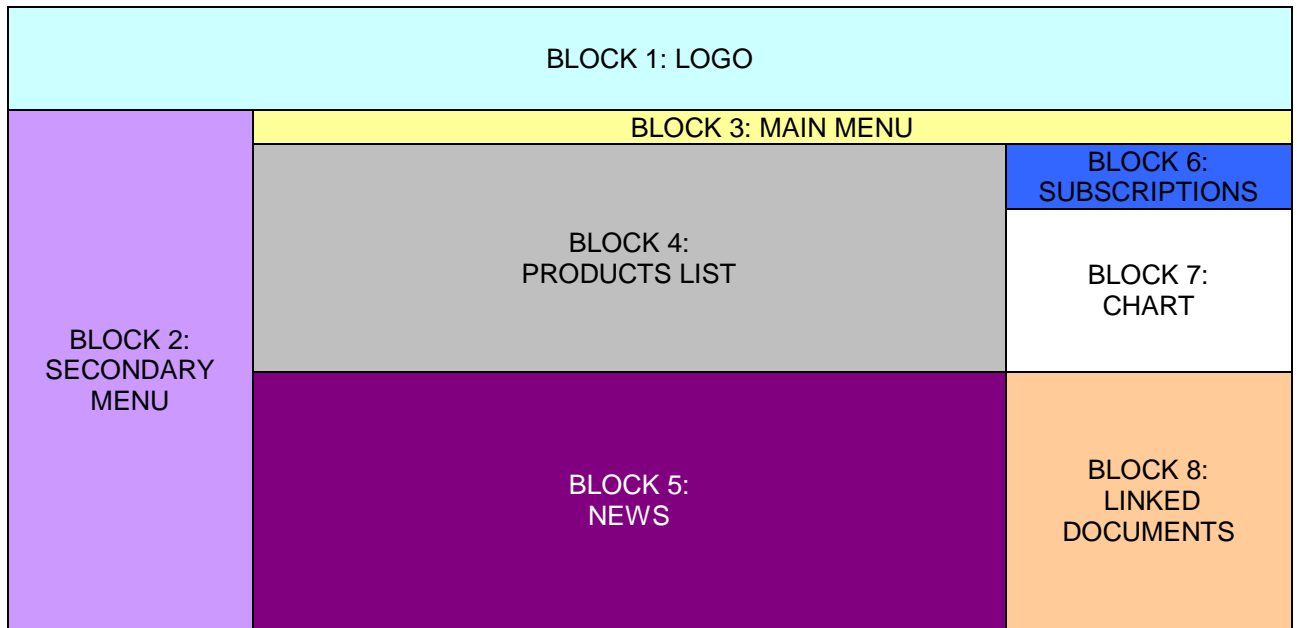


Figure 1. A generic web-page template.

Starting from a given template, each page is then created, filling each block with appropriate contents, as shown in Figure 2, below.

The screenshot shows the website 'Dipartimento del Tesoro' with various content areas highlighted by large black text labels:

- BLOCK 1:** The top header area containing the logo and navigation links.
- BLOCK 2:** The left sidebar menu with categories like 'Servizi per il cittadino' and 'Investire in Titoli di Stato'.
- BLOCK 3:** The main navigation bar with links for 'ENGLISH VERSION', 'E-MAIL', 'Ricerca', 'MAPPA DEL SITO', and 'FAQ'.
- BLOCK 4:** The main content area displaying a table of state bond placements ('COLLOCAMENTO TITOLI DI STATO') with columns for date and details.
- BLOCK 5:** A section titled 'Cosa c'è di nuovo' (What's new) listing recent news items.
- BLOCK 6:** A chart titled 'EVOLUZIONE DELLA STRUTTURA DEL DEBITO E SUA MISURA'.
- BLOCK 7:** A list of strategic lines for public debt ('LINEE STRATEGICHE DEL DEBITO PUBBLICO').
- BLOCK 8:** A list of interest rates ('PRINCIPALI TASSI DI INTERESSE').

Figure 2. Block-decomposition of a web-page.

Identifying blocks within a page is an essential tool to correctly determine the amount of separate elementary process in a function point count for web systems. In practice, all the blocks are activated (loaded) as the pages is loaded, but that doesn't mean that the primary intent of the user is to have all of them shown at the same time for functional reasons (rather, for usability, integration, or multiple choices reasons). It is not so rare to find blocks on the

same page that are totally independent one of each other, and that should therefore be treated and counted as separate processes. The proposed guideline for transaction is therefore to look at the structure blocks in the templates to identify elementary processes. Care must be taken however, in avoiding separating blocks on a pure layout basis, rather than looking at the functionality they're supposed to accomplish. Some blocks should be aggregated into one process, as the input/output side of queries in a page, for instance. Once the correct block analysis/decomposition/aggregations has been made, the standard counting practices can be applied to further refine the transactions' count.

## 5.6. Counting at the *structure-driven* abstraction level

When analysing a website from the *structure-driven* abstraction level, a page is an instance of a template, where the publisher decided what information to fill in each block at a later stage. At this abstraction level, the count scope must include only the pre-defined templates and the generic data groups that the templates are based upon.

### 5.6.1. Count Data Functions

At the *structure-driven* abstraction level, loading a "page" corresponds to executing several data loading processes at a time. Each "block" loads its own data, regardless of what that data will be when specified by the publisher. Semantics cannot be taken into account at this level, so we have to look at the usage of the blocks to identify logical files.

Therefore, the guidelines suggest to identify a logical file per each unique, identified block at the *structure-driven* abstraction level.

### 5.6.2. Count Transactional Functions

As stated in section **Errore. L'origine riferimento non è stata trovata.**, a template made of several blocks leaves us with the task of identifying independent blocks as separate, unique processes. Furthermore, we must consider that what is not counted as a functional item (a function) at the content-driven abstraction level, can still be the like at this other structure-driven abstraction level. For instance, a menu section is not considered, at the *content-driven* abstraction level, being it a technical solution to provide navigation between "true" data to the user; whereas, it could be a significant block to be filled by "menu items" at the structure-driven abstraction level, if the user expressed requirements about controlling and/or customising the items of that menu by means of a structured source data group – at this abstraction level, the "menu" truly is an elementary process to be considered in the count.

The guidelines for transactional functions at the *structure-driven* abstraction level are:

- identify each content-provider-recognizable block, across all the templates, designed to carry non static data;
- review blocks aggregating those blocks that are not independent/autonomous;
- count an elementary process per unique, identified block, and classify it based on its primary intent.

## 5.7. Determine Value Adjustment Factor (VAF)

Since VAF does mix functional and non-functional aspects of the software being measured, it is suggested to keep its value equal to unity, to avoid adjustments on the size. Other factors could then be considered depending on the purpose of the measurement (e.g. cost drivers based on the environment, team, experience, and so on, for effort and cost estimation).

## 5.8. Calculate Adjusted Function Point Count

No specific guideline is needed for this step. Since VAF is suggested to be fixed and equal to unity, unadjusted and adjusted function points coincide.

## **5.9. Conclusions**

Counting function point for web systems is a hard task, it requires high experience and guidelines to ensure that the counting practices are applied coherently across different practitioners. Moreover, interpreting the measurement results for such systems is not easy, since web systems are often implemented with different purposes and approaches than traditional systems and applications; the concept of functional size for web systems could be sometimes not significant or not clear as it is in other domains.

On the other hand, we do need to measure such systems, as we already do for other software domains; therefore, experimental projects should be started in order to test and improve a method for measuring web systems, or to provide alternative approaches, and to avoid the misuse of functional metrics in this domain.

## **5.10. References**

- [1] ISO/IEC 14143-1:1998, “Information technology – Software measurement – Functional size measurement – Part 1: Definition of concepts”, International Organization for Standardization, Geneva, 1998.
- [2] IFPUG, “Function Point: Counting Practices Manual”, Release 4.2.1, International Function point Users Group, 2005.
- [3] Merriam Webster's Collegiate Dictionary, 10a ed.
- [4] ISO/IEC 10746-2:1996, “Information technology - Open Distributed Processing - Reference Model: Foundations”, International Organization for Standardization, Geneva, 1996.