This paper is an extract from *Project Control for Software Quality*, Editors, Rob Kusters, Adrian Cowderoy, Fred Heemstra and Erik van Veenendaal. Shaker Publishing, 1999. ISBN 90-423-0075-2.

ON THE COMPATIBILITY BETWEEN FULL FUNCTION POINTS AND IFPUG FUNCTION POINTS

Serge Oligny, Alain Abran

Abstract

Release 1.0 of the Full Function Points measurement method was proposed in 1997 to measure the functional size of real-time or embedded software. Since then, field tests have shown the applicability and usefulness of this measurement method not only for real-time or embedded software, but also for other types of software like system software and MIS software.

This paper investigates the issue of measurement compatibility between the designs of both the FFP and IFPUG measurement methods. Such compatibility is required to perform mathematical operations involving results from both methods and mixing them into a single functional size measure. The compatibility of both measurement objects and measurement processes is analyzed and the accuracy of their aggregation function is identified as being dependent on the level of granularity at which the measurement functions are applied. Comparing the two approaches, we find that the precision of this aggregation function corresponds to the lowest common denominator of the two approaches.

1. Context

Release 1.0 of Full Function Points (FFP) was proposed in 1997 as a functional size measurement method derived from the International Function Point Users Group (IFPUG) method.

Strictly speaking, the Full Function Points measurement method is a conceptual superset of the IFPUG method, as documented in [1, p. 15]. The extensions proposed by the FFP measurement method were originally intended to extend the applicability of the IFPUG method to the field of real-time and embedded software, as presented in [3, 4]. The FFP approach suggests adding the points obtained by the IFPUG method to the points obtained through the FFP extensions for the measurement of control data and control transactions.

The following issue can then be raised: *Is it conceptually sound to add the points obtained through the FFP extension to the points obtained through the IFPUG approach?*

This paper explores this issue and shows that the two approaches offer a degree of compatibility which allows FFP and IFPUG functional size figures to be combined into a single size figure.

The comparison between the two approaches is based on version 4.0 of the IFPUG method [2], and the extensions proposed in version 1.0 of the FFP method [1]. These are the two most recent versions currently available.

2. A common framework for comparison

A functional size measurement method consists of applying a set of rules and procedures to a given software; the result of the application of these rules and procedures is a numerical figure representing the functional size of the software. In both the FFP and IFPUG Function Points approaches, the rules and procedures are described in a document referred to as the “Counting Practices Manual”. The essence of each approach is described in these measurement standards documents.

A closer look at the contents of these manuals shows that both approaches:

- On the one hand, are intended to be independent of the implementation decisions embedded in the operational artifacts of the software to be measured, and,
- On the other hand, involve an obligation to apply measurement rules and procedures to some visible artifacts related to the software to be measured.

A more detailed view of the measurement approach proposed by FFP and by IFPUG Function Points is presented in Figure 1 below. This figure illustrates the common framework for comparing FFP and IFPUG Function Points.

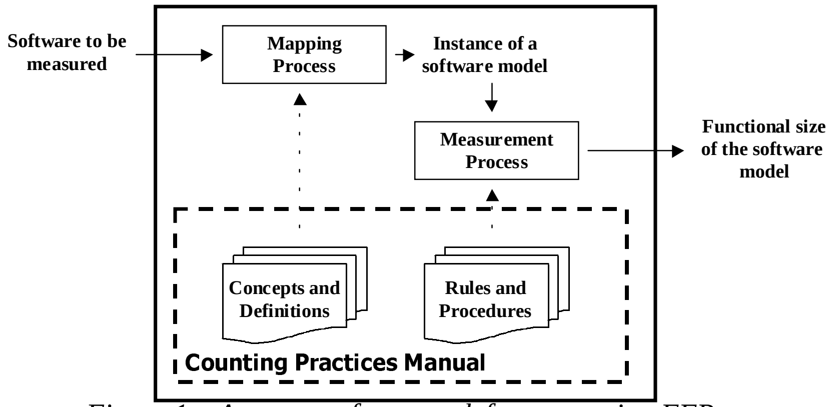

Figure 1 – A common framework for comparing FFP and IFPUG Function Points

This framework illustrates that, prior to applying the measurement rules and procedures, the software to be measured must be mapped onto a specific software model that captures the concepts and definitions required for a functional size measurement exercise. Although this model is not explicitly defined as such in the Counting Practices Manual of each approach, both rely on an implicit model of the software to be measured. It is on the objects of this implicit software model that a) the software to be measured is mapped, and b) the rules and procedures of each approach are applied in order to produce a numerical figure representing the functional size of the software. Therefore, two distinct and related processes are necessary to measure the functional size of software: **mapping** of the artifacts of the software to be measured onto an implicit software model and then **measuring** specific features of this software model.

The mapping process takes as input the artifacts of the software to be measured (as they are found/documented within the organization) and produces as output an instance of a software model. This instantiated software model is defined by the concepts and definitions found in each approach.

Next, the measurement process takes as input the instantiated software model and produces as output the numerical figure representing the functional size of the software model. By convention, this numerical figure is then extended to represent the functional size of the software itself. The functional size figure is therefore derived by applying the documented set of rules and procedures described in each approach.

The framework presented in Figure 1 will be used as a common basis for comparing FFP and IFPUG Function Points. For the two approaches to be compatible, their software models must be compatible and their measurement processes must be compatible.

3. Comparing the software models

In order to compare the software models generated by the FFP and the IFPUG Function Point approaches, it is necessary to extract from each approach the concepts that characterize them. Five concepts characterize the software model of the FFP approach, while four concepts characterize the software model of the IFPUG approach. These are presented in Table 1 below. The software model of each approach will be deemed to be compatible if all concepts are compatible.

Table 1 – Concepts characterizing the software models measured by FFP and IFPUG Function Points

CONCEPTS	FFP	IFPUG
Boundary	Boundary	Boundary
Users	Users	Users
Data objects	Group of Control Data	Logical Files
Process objects	Control Process	Elementary Process
Sub-process objects	Sub-process	N/A

The definitions of each concept according the FFP and IFPUG Counting Practices Manuals are presented in Table 2. Key embedded definitions have also been included in Table 2, where relevant, in order that each concept can be grasped with a minimum of ambiguity. A comparative analysis of the concepts of each approach is presented next.

3.1. Boundary

According to the FFP Counting Practices Manual [1, p. 15] , the FFP boundary concept is identical to the one proposed by IFPUG. The boundary concepts of the two approaches are entirely compatible therefore.

3.2. Users

The FFP definition of users includes and extends the IFPUG definition to encompass other software and mechanical devices interacting with the measured software. The FFP definition is therefore a superset of the IFPUG definition. The definitions of users in the two approaches are thus deemed to be compatible.

3.3. Data objects

The IFPUG definition of data objects and associated rules and hints [2, section 5, p. 6 and 13] relates them to what is perceived by the users of the software. The FFP definition [1, p. 17] also relates data objects to the users' perspective, although, given the FFP definition of the concept of users, the vocabulary is adapted to widen this perspective. Both approaches identify data objects from a logical perspective, as distinct from the implementation instances. The FFP approach explicitly excludes technical and implementation considerations in the identification of data objects. Both approaches distinguish between data objects which are read-only and data objects which are updated. At the core of both approaches lies the idea that the concept of data objects enables the grouping of logically related data from a perspective which is similar in nature but which differs in scope due to the differences in the definitions of users. The FFP concept of data objects is a superset of IFPUG's data objects. The concepts are therefore deemed to be compatible.

CONCEPTS	FFP definition	IFPUG definition
Boundary	Identical to IFPUG [1, p. 15]	*"The border between the application or project being measured and the external applications or the user domain. A boundary establishes what functions are included in the function point count."* [2, Glossary p. 2]
Users	*"Human beings, applications or mechanical devices which interact with the measured application."* [1, p. 45]	*"(1) The person or organization that uses the measured application. Included would be the requirement author, end users, management users, auditors, and operations. (2) The human being who uses the application"* [2, Glossary p. 5]
Data object	"***Group of data****: data identified and grouped together based on the functional perspective."* "***Control data****: data used by the application to control, directly or indirectly, the behavior of an application or a mechanical device."* "***Functional perspective:*** *point of view of the functionality delivered by the application; it excludes technical and implementation considerations.*" [1, p.16]	"***Data function types****: the functionality provided to the user to meet internal and external data requirements. Data function types are either internal logical files (ILFs) or external interface files (EIFs)."* "***ILF****: an ILF is a user identifiable group of logically related data or control information maintained within the boundary of the application being counted".* "***EIF****: a user identifiable group of logically related data or control information maintained outside the boundary of the application being counted."* [2, Glossary p. 2, 3 and 4]
Process object	"***Control process****: process that controls, directly or indirectly, the behavior of an application or a mechanical device."* "***Process****: A set of operations or activities which acts on inputs to produce a result.*" [1, pp. 18, 44]	"***Elementary process****: the smallest unit of activity that is meaningful to the end user in the business. It must be self-contained and leave the business of the application being counted in a consistent state."* [2, Glossary p. 3]
Sub-process object	"***Sub-process****: […] the smallest processing step identifiable from a functional perspective as either an entry, exit, read or write."* "***Functional perspective:*** *point of view of the functionality delivered by the application; it excludes technical and implementation considerations.*" [1, p. 44]	Not applicable

Table 2 – Definitions of FFP and IFPUG software model concepts

3.4. Process objects

The FFP definition of process object is based on the traditional definition of process ("*…input and… results…*") and the link between the object and the concept of users. The IFPUG definition of process object is based implicitly on its size ("*…smallest unit…*") under two conditions (*"…self-contained… leave the application… in a consistent state"*) and on the concept of users (*"…meaningful to the end user…"*). The IFPUG approach does not provide rules to evaluate what is bigger and what is smaller, although it does provide a few hints [2, section 6, p. 14, 25 and 39] to guide the practitioners.

The IFPUG definition implicitly define the process object to be measured as a grouping of sub-processes leaving the software (application) in a consistent state from the perspective of the end users. This grouping is a subset of the processes found in the software. The FFP definition explicitly define the process object to be measured as the set of all processes found in the software. The two approaches are deemed to be compatible since the FFP definition of the process objects is a superset of the IFPUG definition.

3.5. Sub-Process

The sub-process concept is not formally identified in the IFPUG software model. In FFP, the sub-process concept is dependent on the control process concept [1, p. 18], for which there is an equivalent in the IFPUG approach (section 3.4 above). The sub-process concept is strictly a logical sub-division of the control process proposed by the FFP approach. It lies at a level of granularity below the lowest level proposed by the IFPUG approach. Therefore, under the restriction of aggregating the FFP results at the process object level (section 4.3 below), the sub-process concept is a subset of the process objects proposed by the IFPUG approach.

3.6. Summary

The software model generated by each approach have been characterized through five concepts (FFP) and four concepts (IFPUG) respectively. It has been shown that four of these concepts are compatible, being either identical or defined by the FFP Counting Practices Manual as a superset of the corresponding concept in the IFPUG approach. A fifth concept is proposed by the FFP approach (sub-process) which is a subset of the process objects in the IFPUG approach. This concept does not invalidate the compatibility of the other four concepts in the two approaches, therefore the software models proposed in each approach are deemed to be compatible.

4. Comparing the measurement processes

The compatibility of the measurement processes will be evaluated on the basis of three distinct criteria: a) the compatibility of the objects receiving “points”, b) the compatibility of the “measurement” functions, and c) the compatibility of the “aggregation” functions.

In order for the two approaches to be compatible, they must use compatible objects to generate “point” figures. As presented in section 3, the objects used to generate points are drawn from similar and compatible concepts embedded in the software model implicit in each approach.

Furthermore, for the two approaches to be compatible, they must generate points using compatible functions. These functions accept characteristics of the measured object as input and produce a numerical figure representing the functional size of these objects as output.

Finally, for the two approaches to be compatible, they must generate the total size of the measured software using a compatible function for the final aggregation of the individual numerical figures assigned to each measured object.

4.1. Compatibility of the measured objects

On the one hand, both the FFP and the IFPUG measurement approaches use types of data as a measured object type of the software model (“data objects”). As presented in section 3.3, these objects are compatible.

On the other hand, the FFP measurement method uses the “sub-process objects” as the measured object type of its implicit software model. Through its aggregation function (section 4.3), the FFP approach offers the ability to derive a functional size for a control process. A control process is a “process object” compatible with the IFPUG measurement approach (section 3.4) but at a different level of granularity. Therefore, the measured objects of the two approaches are compatible, with the restriction that the measurement results be kept at the “process object” level when using the FFP approach.

4.2. Compatibility of the measurement functions

Both the FFP and the IFPUG measurement methods propose five distinct measurement functions, depending on the nature of the measured objects. These are presented in Table 3, below. In this Table, each measurement function is identified (ID column) for easier reference. The approach, the “concept object” and the object instances (Acronym) measured by each function are mentioned and the argument(s) are listed. Functions arguments are equivalent across the two approaches since they are based on identical definitions and their values are obtained through a simple count of the number of their instances in a measured object.

Table 3 – Inventory of the measurement functions proposed by the FFP and the IFPUG measurement methods

ID	Approach	Concept object	Acronym	Argument
1	FFP	Data object – read	RCG (mult.)	DET and RET
2	FFP	Data object – updated	UCG (mult.)	DET and RET
3	FFP	Data object – single read	RCG (single)	DET only
4	FFP	Data object – single updated	UCG (single)	DET only
5	FFP	Sub-process objects	ECE,ECX,ICR,ICW	DET only
6	IFPUG	Data object – read	EIF	DET and RET
7	IFPUG	Data object – updated	ILF	DET and RET
8	IFPUG	Process object – input	EI	DET and RET
9	IFPUG	Process object – output	EO	DET and RET
10	IFPUG	Process object – inquiry	EQ	DET and RET

4.2.1. Compatibility of "data object" measurement functions

Measurement functions 1 and 2 on the one hand (FFP) and measurement functions 6 and 7 on the other hand (IFPUG) are exactly the same since, from [1, p. 17], "*…multiple occurrence groups of data have the same structure as ILFs and EIFs in FPA; they are counted in exactly the same way as these two FPA function types.*"

Measurement functions 3 and 4 (FFP) are unique in their mathematical forms and use only a DET count as the argument. They have no direct equivalents in the IFPUG measurement approach. They are designed to yield a numerical value encompassing the progression in magnitude proposed by the IFPUG approach, as stated in [1, p. 13]. Unlike measurement functions 1, 2, 6 and 7 however, the range of values of these two measurement functions is not upper-bounded. Consequently, these two FFP measurement functions offer a superset of the range of values offered by the other "data object" measurement functions, implicitly assuming a constant RET count of 1.

FFP measurement functions 3 and 4 are thus deemed to be entirely compatible with IFPUG measurement functions 6 and 7 respectively, in the range of values offered by IFPUG. Compatibility is preserved outside this range of values on condition that the extrapolation trends shown by measurement functions 6 and 7 are accepted.

4.2.2. *FFP "process object" measurement functions*

First consider FFP measurement function 5. This measurement function assigns "points" to "sub-process" objects based on the DET count referred to by each measured sub-process. As stated in section 4.1 above, compatibility is to be considered at the process level, as in the IFPUG approach. By virtue of its aggregation function, the FFP approach allows the assignment of points at the process level. The number of points assigned to a process is simply the arithmetic sum of the points assigned to each of its constituent sub-processes through measurement function 5.

Furthermore, it is to be noted that the definition of the sub-process objects entails an indirect consideration for RET; from [1, p. 18]: "*…if a process enters two groups of data, there are at least 2 external control entries (ECEs)*" and each data object has a different RET by definition. Such statements are part of the definition of all four types of sub-process [1, pp. 18-19], meaning that a sub-process implicitly refers to only one RET.

It is thus implicit that a measurement function exists in the FFP approach that allows the measurement of "process objects". This implicit function is a composite of measurement function 5 and the FFP aggregation function; it uses, implicitly or explicitly, both DETs and RETs as arguments. It is to be noted that there is no upper bound to the range of values offered by this implicit FFP process object measurement function, by virtue of the properties of its embedded aggregation function.

4.2.3. *IFPUG "process object" measurement functions*

IFPUG process object measurement functions 8, 9 and 10 are similar in form. They differ slightly in the range and distribution of their output values based on the mandatory functional nature of the process object onto which they are applied. It is to be noted that measurement functions 8, 9 and 10 have their highest and lowest values bounded by constants. Furthermore, consider the following specific measurement rules proposed by IFPUG as part of the definition of one type of process object:

"*An external input (EI) processes data or control information that comes from outside the application's boundary... The processed data maintains one or more ILFs*" [2, section 6, p.4].

This measurement rule implicitly refers to the inner processing of the measured process object by alluding to processing steps that would be identified as at least one "entry" and at least one "write" sub-process respectively in the FFP measurement approach. Figure 5 below illustrates this comparison.

It can thus be seen that, although not explicitly defined in the IFPUG measurement functions, the IFPUG approach implicitly refers, by construction, to concepts compatible with the FFP "sub-process objects".

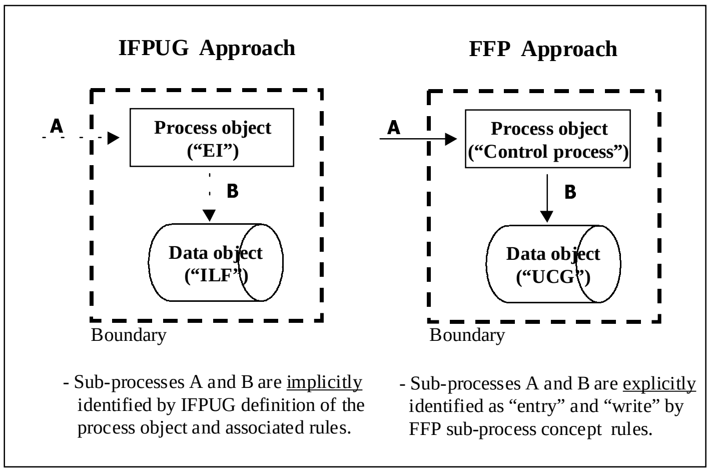
IFPUG Approach
FFP Approach
A
Process object ("EI")
B
Data object ("ILF")
Boundary
A
Process object ("Control process")
B
Data object ("UCG")
Boundary
- Sub-processes A and B are implicitly identified by IFPUG definition of the process object and associated rules.
- Sub-processes A and B are explicitly identified as "entry" and "write" by FFP sub-process concept rules.

Figure 5 – Comparison of the "process object" measurement functions of the two approaches

4.2.4. Compatibility of "process object" measurement functions

Although different in their form, it can be seen that, at the "process object" level, FFP measurement function 5 and IFPUG measurement functions 8, 9 and 10 all refer to common compatible objects. Furthermore, they generate values of compatible magnitude within the bounded range offered by IFPUG measurement functions 8, 9 and 10 at the same level of granularity.

As a corollary to this analysis, it can be seen that the relation between the points assigned to a process object by IFPUG measurement functions 8, 9 and 10 and the count of this process object's implicit sub-processes (entailed by specific IFPUG definitions and rules) is coarser than the corresponding relation offered by the FFP process object measurement function. This is a direct consequence of the finer level of granularity of the objects onto which the FFP approach assigns points: the sub-processes.

The set of measurement functions offered by both approaches is therefore deemed to be compatible within the bounded range of values offered by the IFPUG approach. Outside this range, the FFP approach is deemed to be compatible with the IFPUG measurement functions under the condition of accepting that the FFP approach offers a logical extension that is based on the same concepts as the ones used inside the range.

4.3. Compatibility of the aggregate function

Both the FFP and the IFPUG measurement approaches propose a mechanism to construct the functional size of the software model from the functional size of the individual measured objects in this model. This mechanism use the same type of objects in both approaches: data object and process object. This aggregation function is simply the arithmetic sum of the functional size of the measured objects. It is exactly the same function in both measurement approaches and, therefore, the two approaches are compatible under this criterion.

It is to be noted, however, that the accuracy of this function is dependent on the level of granularity at which the measurement functions are applied. With both approaches, when adding points using the aggregation function, the precision of this function will correspond to the lowest common denominator of the two approaches.

5. Summary and conclusion

The analysis of the compatibility of the measurement approaches proposed by IFPUG (4.0) and FFP (1.0) was conducted within the common framework documented in section 2. It was established that, in order to be deemed compatible, the two approaches would have to offer a compatible software model and a compatible measurement process.

By breaking up the software models of each approach into their constituent concepts and comparing the corresponding concepts, it was shown in section 3 that these software models are compatible, either because the concepts are identical or because the concepts proposed by the FFP approach are supersets or subsets of the corresponding concepts proposed in the IFPUG approach.

By breaking up the measurement process into its three constituents and examining the compatibility of each constituent, it was shown in section 4 that the IFPUG measurement process offers a bounded range of functional size values, while the FFP measurement process does not. Therefore, the compatibility of the two measurement processes is established according to two ranges of values:

- Within the range of values offered by the IFPUG measurement process, both measurement processes are entirely compatible at the data and process object level.
- Outside the range of values offered by the IFPUG measurement process, the two measurement processes are compatible under the following conditions:

a) results are considered at the process object level,
b) FFP measurement functions 3 and 4 (Figure 4) are an extrapolation of the range of values provided by IFPUG measurement functions 6 and 7 (Figure 4),
c) FFP measurement function 5 (Figure 4) combined with FFP aggregation function provide an extrapolation of the range of values provided by IFPUG measurement functions 8, 9 and 10 at the process object level.

The conditions for compatibility outside the range of values provided by the IFPUG approach are deemed reasonable for many purposes, including functional measurement of sizeable chunks of a software, measurement of entire software products and measurement of software portfolios.

The impact of these constraints is limited when compared to the benefits of using only one functional measurement approach throughout the organization or the benefits of adequately including real-time, embedded and technical software functional size in the economics of an organization's software process management.

Practice tends to show [5, 6] that the FFP approach, while offering results very similar to those of the IFPUG approach when applied to MIS software, offers more adequate results when applied to real-time, embedded or technical software by virtue of the fact that a) its measurement functions are not bounded by constants and, b) the level of granularity is more relevant to these type of software.

Furthermore, in situations requiring the measurement of smaller pieces of software, the FFP approach offers a finer degree of granularity than the one offered by the IFPUG approach by virtue of the identification and measurement of sub-processes.

6. References

[1] "*Full Function Points: Counting Practices Manual*", Software Engineering Management Research Laboratory, Université du Québec à Montréal, Technical Report no. 1997-04, September 1997. See www.lrgl.uqam.ca/ffp.html

[2] "*Function Points Counting Practices Manual – Release 4.0*", International Function Point Users Group (IFPUG), Westerville, Ohio, USA, January 1994.

[3] "*Adapting Function Points to Real-Time Software*", Abran, A., Maya, M., Desharnais, J.-M. and St-Pierre, D., American Programmer, vol. 10 no. 11 (November), pp. 32-43, 1997.

[4] "*Measuring the Functional Size of Real-Time Software*", Maya M., Abran A., Oligny S., St-Pierre D., Desharnais J. M., Proceedings of the 9th European Software Control and Metric Conference (ESCOM-ENCRESS 98), Rome, Italy, May 1998.

[5] "*Measuring ALL the Software not just what the Business Uses*", Morris P.; Desharnais J.-M., Proceedings of the International Function Point Users Group (IFPUG) Fall Conference held in Orlando, Florida. Westerville, Ohio, USA, September 1998.

[6] "*Functional Size of Real-Time Software: Overview of Field Tests*", Oligny S.; Abran, A.; Desharnais, J.-M.; Morris, P. , Proceedings of the 13th International Forum on COCOMO and Software Cost Modeling, Los Angeles, CA, October 1998.