Function Point: how to transform them in effort? This is the problem!

Gianfranco Lanza

Abstract

The need to estimate the effort and, consequently, the cost of a software project is one of the most important issues in ICT world. The use of function point to measure the functional dimension of a software project is the base in CSI Piemonte for deriving the effort and consequently the cost.

If the process to measure the functional dimension in function point can be considered as a scientific method, based on rules defined in IFPUG Function Point Counting Practice Manual, the process to determine the effort is not so deterministic and scientific but it is depending on many non functional or technical factors.

In CSI Piemonte it has been developed a model to determine the effort by Function Point through a process that, step by step, takes into account many possible technical factors that can influence the productivity.

- *The first step is to determine the "medium productivity" about the project through two parameters: the programming language (according to Capers Jones Programming Language Table [1] or ISBSG Repository [2]) and the functional dimension in Function Point (the productivity decrease if the dimension of the software project arises).*
- *The second step is to analyse the Technical Factors (algorithmic complexity, relations with other applications, with other projects, time binding, etc.) that influence the productivity: in accord to a nominal value they can arise, reduce or not modify the "medium productivity".*
- *The Third step is to take into account the risks of the project, expressed as a contingency: an additional percentage of the total effort calculated as a multiplication of the probability that the risk happens and its relative impact on the project.*
- *The Fourth step is to consider other activities, not depending by the dimension of the project, but worthwhile in calculating the effort (for example to develop a prototype to evaluate a new technology).*

At the end the total effort is divided into RUP Activities for a better scheduling of the project, in according to top down estimate technique.

The effort estimate derived by Function Point has to be compared with the effort estimate derived by experience, thus it's not the only method used to estimate effort and cost.

1. Introduction

CSI Piemonte is a non lucrative consortium working for the Public Administration.

Its objective is to improve the effectiveness and the efficiency of the P.A. through an integrated services system for the citizens.

The increasing economic narrowness, the need to realise services more efficient and less expensive, the need to do benchmarking, have been the factors for the development of a metric program that, in according to the paradigm Goal Questions Metrics, should determine the metrics for a better knowledge of its productive process and so to improve it.

The use of function point to measure the functional dimension of a software product started in 2004. During these years it was developed a predicting work effort model to determine the amount of effort for the software development and, thus, the relative cost.

The estimate of effort and cost through the functional dimension is only one of the estimate methods used, but it is not unique, on every project it's recommended to do also an estimate through an analytical way, using experience (Bottom up method or/and Delphi method). If you have more estimates than only one, you should improve the final confidence in the value of estimate in terms of effort and costs.

CSI Piemonte has begun to create a repository for collecting the baseline of the projects, with their functional dimension and their relative effort.

2. Scope

CSI Piemonte uses function point on projects belonging to the classes furniture of "Developing Projects" and "Enhancement Functional Maintenance", in according with "LGC-FP Linee Guida per l'uso contrattuale dei Function Point - GUFPI" [3]. The majority of the income of CSI Piemonte is concerning these two classes of furniture.

Thus, all the developing projects are estimated in Function Point, except those projects in which the Function Point Metric is not suitable (i.e. Projects with a high algorithm component or firmware projects or the editing part of an internet portal project).

Regarding the Enhancement Maintenance Projects (in the following as EMP) they are estimated in Nesma Function Point [4].

The application of a model for effort estimate is restricted to developing projects with a function point dimension greater than 50 FP and EMP projects with a functional dimension greater than 20 Nesma Function Point.

3. The "Medium productivity"

The model used to determine the work effort related to the software development cycle consists of several steps, first of all is to determine a base productivity to apply for the current project.

The best thing would be to obtain this value from the CSI Piemonte projects repository, selecting the value according to the historical data of similar projects (regarding functional dimension, technical environment and application domain). Until now, the number of projects in the repository is not great enough to obtain significant values, so the base productivity is chosen regarding literature values in according to the Programming Languages Tables of Capers Jones or in some cases, the ISBSG repository. Both Capers Jones Programming Languages Table [1] and ISBSG Data [2] give you a range of productivity (expressed as Function Point/PersonMonth).

In CSI Piemonte the majority (over 90%) of software projects are using Java Language, so in this paper the example is concerning developing projects in Java. It's evident that today most of the software projects uses not only one language, (as it was several years ago, with projects developed in COBOL or similar), but they use in many cases several languages (i.e. Java, PL/SQL, HTML…), we consider the main language on the project.

It is evident that the "base productivity" can be often a value not very significant to determine the final effort, but in any case you have to start by an initial value. The first effort driver in order to calibrate the "base productivity" is the functional dimension of the projects. Table 1 shows the "base productivity" for Java related to the functional dimension in FP: with the growth of dimension, the productivity decreases by two points; there are five classes of projects. The productivity is expressed in fp/person month.

Table 1

Function Point Dimension		Base Java Productivity(FP/mu)
Small	50-350	18
Middle-Small	350-650	16
Middle	650-1100	14
Middle-Large	1100-2000	12
Large	>2000	10

Applying these values to the dimension in FP of a project we obtain a base work effort of the software development project in Month/person or in Days/person (usually we consider a month of 21 days).

This effort should cover all the activities concerning the software project according to the RUP (Rational Unified Process) disciplines.

For example if there is a project valued 574 FP, from the table above we obtain a "base productivity" of 16 fp/month and thus a "base work effort" of 35.87 Month/person(574/16) or 753 days/person.

In the case of EMP it is to consider not only the functional dimension of EMP (in Nesma FP) but also the functional dimension of the application on which the maintenance will be applied to for selecting the base productivity, (for example if we had an EMP for our projects of 574 FP, we will choose a productivity of 16 fp/month, independently of the EMP Nesma functional dimension in FP).

4. The "Productivity Factors"

Every project has its own characteristics, so it is clear that the range of productivity can be very large! The most critical step in evaluating the effort of the project is to consider all the factors that can influence the productivity. Through the Functional Requirements it is possible to measure the functional dimension in Function Points.

The Technical and Quality requirements should give us information about the factors that influence productivity. But how can we measure them? And, above all, which are these factors?

In 2003 in CSI Piemonte was done an investigation to find them and to evaluate the impact that they have on productivity (growing or decreasing it).

To determine the impact of productivity factors it was chosen an approach like COCOMO [6] (the productivity factor is the multiplication of several cost drivers).

The objective was to identify a set of factors and for each of them to determine the possible impact on productivity on a scale of five values (very low, low, nominal, high and very high). For every impact it would have been assigned a percentage value (positive or negative, the nominal value was 1) representing the percentage of work effort relating to the factor.

To a certain number of professionals on different roles (Analysts, Programmers, Vendors, and Project Managers) was asked which, in their opinion, were these factors; to make simpler the choice it was prepared a list of possible factors similar to the list of COCOMO cost drivers.

- The first step was to reduce the list, identifying only those more significant for CSI Piemonte.
- The second step was to describe each of them, in the simpler and more objective way.
- The third step was to identify the meaning of nominal value. This was a delicate and critical issue. In some cases this has been implied a redefinition of the descriptions done at step two.
- At the end people indicated a percentage of work effort for every impact level on each factor; it was made a weighted media of this values and it was obtained the table with the relative value.

In figure 1 is reported the description of one of these factors "COCA", while in Figure 2 there is the list of all the factors with their relative values.

COCA: Application Complexity

It indicates the impact level that the application context complexity has upon the project. The following factors determine the level impact:

- The application is influenced by a legislative context: the application has to be developed considering possible legislative changes during its life software cycle.
- The subject managed in the project is not simple but it is articulated and complex.
- The project has to implement a service for public bodies and private bodies.

Very low	Low	Nominal	High	Very High
	The project is influenced by none of the elements or by only one	The project is influenced by two of the elements	The project is influenced by all the elements	

Figure 1

INPR: Input Projects		Very Low	Low	Nom	High	Very High
COCA	Application Complexity	-	0,89	1,00	1,26	-
QREQ	Requirements Quality	-	1,13	1,00	0,97	0,89
OUPR: Output Projects						
INTG	Products Integration	0,86	0,91	-	1,19	1,31
DOCU	Documentation Level	0,88	0,91	1,00	1,15	1,23
PROC: Productive process						
SCED	Schedule compression	0,93	0,95	1,00	1,17	1,33
INTP	Projects integration	0,82	0,92	1,00	1,22	1,55
INFR:Technical and software platform factors						
ARCH	Target distinct architectures	-	-	1,00	1,13	1,25
TEAM: Team group factors						
APEX	Domain context know how	1,18	1,07	1,00	0,91	0,88

Figure 2

It's worthwhile to note that in some cases there are not all the impact values (in some cases there are not the extreme values, in one case there is not the nominal factor). The final productivity factor is obtained by the multiplication of the single factors, the final value is multiplied for the "base work effort" decreasing or growing it.

The practical application of these factors needs great attention as the impact on productivity can be very significant. The determination of impact level can not be only a simple deterministic method but, first of all, it is important to see if the factor could be relevant to the project or not. It is not mandatory for the current project to define an impact on every factor. The process to determine the productivity factors must be done by people with a great experience both in the context domain and in the technical domain; the process is managed by a metric professional.

Suppose that for the project of 574 fp, with a "base work effort" of 35.8 Month/person or 753 days/person, we determine a very high impact for productivity factor COCA (Application complexity: 1.26) and a Low impact for INTG (Products integration, 0.91); the resulting total productivity factor is 1.26*0.91 = 1.1443.

The resultant effort is 753*1.1443 = 862 days.

5. The "Risks"

The process for the effort estimate has to take into account to perform a risks analysis. But what's the difference between risk and productivity factor? Both the risk of a project and a productivity factor may have a strong impact upon the final effort of the project.

The risk is the possibility that an event occur, but it's not sure that the event will really occur! A risks analysis has to consider all the risks of the project. But how can I do to determine the impact of the risk upon the project?

A possible answer is to consider the contingency. What is contingency? The contingency is the multiplication of the probability that an event occur with the percentage of its impact upon productivity. The result is a percentage value that is applied to the "work effort" to obtain a number of effort days. The number of the effort days deriving from contingency may be considered in the whole effort or not; this can be a marketing decision.

Thus, after the risks analysis, the work effort estimate of a development software project is composed by two amounts of effort days:

1. A number of effort days derived by multiplying the "base work effort" with the "productivity factor".
2. A number of effort days of contingency.

In CSI Piemonte it has been defined a table of possible values to determine the contingency. This table helps to identify the probability of a risk occurrence and its relative impact upon productivity. In the table the value "weight" is presented as a percentage).

Table 2

Value	**Meaning**	**Weight**
0	no probability/ no impact	0
1	minimum probability/ minimum impact	3
2	low probability/ low impact	5
3	medium probability/ medium impact	17
4	high probability/ high impact	25
5	very high probability/ very high impact	30
6	maximum probability/ maximum impact	45

In Figure 3 there is an example of contingency relating at three risks:

1. A medium probability with a very high impact for requirements instability.
2. A low probability with a medium impact for the resources know how
3. A medium probability with a medium impact for sharing resources.

Risks	*Probability*	*Impact*	*Contingency*	*effort days*	*Note*
Requirements Quality	0	0	0,0%	0	
Requirements instability	3	5	5,1%	44	
Turn over	0	0	0,0%	0	
Resources know how	2	3	0,9%	7	
Technological binding	0	0	0,0%	0	
Sharing resources	3	3	2,9%	25	
....	0	0	0,0%	0	
....	0	0	0,0%	0	
....	0	0	0,0%	0	
....	0	0	0,0%	0	
....	0	0	0,0%	0	
....	0	0	0,0%	0	
....	0	0	0,0%	0	
....	0	0	0,0%	0	
....	0	0	0,0%	0	
Contingency			**8,8%**	**76**	Total contingency effort

Figure 3

The final contingency represents the 8.8% of the "total work effort", corresponding in a precise number of days. Thus our project of 574 function point has a contingency of 76 days.

6. The "Extra Function Point Effort"

In the development of software cycle there are some activities that are not depending on the functional dimension. Nevertheless these activities are necessary and thus they have to be considered in the total effort. How can they be estimated?

As they are not depending on functional dimension, the function point can't help us. These activities have to be estimated in according to the traditional method, based on experience. The effort days estimated in this way have to be added to the effort estimate in the previous steps. One of these activities could be the realisation of decoding tables. Some others activities could be necessary for removing a risk.

For example, the presence of a risk can bring to a contingency of a certain percentage. We can ask if there is any action for reducing the risk (and, consequently, also the relative contingency) or, in a better way, eliminating the contingency completely. The actions that have to be done in removing the risk could be estimated with a traditional manner, not depending on functional dimension.

For example, if there is a technological risk about a new software platform, it would be useful to develop a prototype to test it. The effort to develop the prototype probably is not dependent on the functional dimension of the project, so it has to be estimated in a traditional manner. This effort is a "non dimension functional effort" and it's necessary to reduce or to eliminate a risk, thus the relative contingency.

It's very important to consider a critical issue as a productivity factor, or as a risk, or as an effort to reduce the risk, only once! If there is a critical issue relating to the project that can impact productivity you have to choice if to consider it as a productivity factor (the probability that it will occur is more than 50%), or if to consider it as a risk or if to consider it as a non functional effort for reducing or eliminating the relative risk.

Regarding our project of 574 Function Point we decided to eliminate the contingency of seven days (see figure 5) relating to the resources know how with 5 days of specific training. In this way the contingency becomes 69 days and there is an "extra function point effort" of five days.

The total effort is 862 effort days plus five days for training and 69 days of contingency.

7. Effort split in RUP activities

The "base work effort estimate" was obtained through a "base productivity" in according to the functional dimension of the project. It was adjusted through the productivity factors, then it was valued the contingency and last, but not least, it was considered all the activities not depending from the functional dimension.

Now the total effort represents the effort in days, or in month, necessary for the software development cycle through the RUP disciplines:

- Requirements.
- Analysis and Design.
- Implementation.
- Test.
- Deployment.
- Configuration and change management.
- Project management.
- Environment.

It is worthwhile to split the total work effort of the software development software cycle upon these disciplines; we need a "weight" for each discipline on the entire software development cycle.

In CSI Piemonte the split has done till now according to literature values [5], adjusted for the requirements and Analysis and Design disciplines (A bit more value in percentage for the requirements and a bit less value for the Analysis and Design, as in CSI Piemonte the requirements very often require great attention). The values are shown in Figure 4.

Disciplines Effort	
Disciplines	**%**
Requirements	16%
Analysis and Design	14%
Implementation	27%
Test	20%
Deployment	6%
Discipline a supporto	
Configuration Mgmt	6%
Project Mgmt & Environment	11%
Total	100%

Figure 4

The disciplines Project Management and Environment have been put together. These values are indicative, they can change project by project according to the functional dimension (i.e. big projects have a lower implementation percentage of effort than small projects), to the scope (enhancement project or developing project), to the risks, to the productivity factors and so on.

The essential thing is that every change of these values must be shared with all the people interested in the RUP disciplines. This activity must be always managed by a metric expert.

If we maintain the split of figure 4, the disciplines effort of our project is illustrated in figure 5. If necessary we have to split the contingency days trough the disciplines.

Disciplines Effort		Effort
Disciplines	**%**	**Days/person**
Requirements	16%	138
Analysis and Design	14%	121
Implementation	27%	233
Test	20%	171
Deployment	6%	52
Discipline a supporto		
Configuration Mgmt	6%	52
Project Mgmt & Environment	11%	95
Total	100%	862

Figure 5

8. Effort Repository

A firm repository from which to obtain indicators is a must that every firm should achieve. The creation of a metric system implies, first of all, a metric program. A metric program has to:

- Define the metrics.
- Build the indicators.
- Define the process to measure.
- Identify the tools.
- Define an organisation for collecting and analysing the measures.
- Create a metric culture.

In CSI Piemonte about three years ago the management started a program metric. The most important thing in a program metric and, thus in a firm system metric, is the culture diffusion upon metrics. If people don't understand why to measure, the failure of a system metric is sure! In CSI Piemonte many training courses were made to make aware the importance to measure. The responsibility of system metric is of the PMO office, all the Function Point measures are made by PMO professionals, very often CFPS certified.

When a project finishes and the relative software service becomes effective, the final product is counted and the baseline with the functional dimension in function point is put into the firm repository. The effort needed to realise the product is also registered with the baseline.

Into the repository it is registered a "normalised effort": all the productivity factors or problems happened during the software development cycle are take into account and they can determine a growth or decrease of the real productivity; in this manner the productivity registered in the repository is a "normalised productivity", so it is a more suitable value to use for the prevision model.

Till now there are not so many projects to afford us using the repository productivity, but we are confident that within the end of the year there will be a sufficient number of projects to do so.

9. Conclusion

The use of Function Point to help people in the software development effort estimation is a delicate process. CSI Piemonte decided to estimate the costs of the projects starting by the effort estimate in days. The cost is obtained from the days effort, so it is the relative price and the function point price. This mechanism guarantees a more right price and, above all, a better quality product.

This is a little different from what is defined in the "LGC-FP Linee Guida per l'uso contrattuale dei Function Point - GUFPI" [3], in which the project price derives from a "base function point price"; the problem is that doesn't exist a "base function point price"; if you have to consider the actual market function point price, the risk is that it is very underpriced, with, as a consequence, the risk to have a poor quality of the final product.

This estimation model is used by CSI Piemonte to regulate the contracts with its suppliers.

One of the most important advantages seen in using function point has been the possibility to make clear the requirements from the beginning, something that very often didn't happen before.

Last but not least it's fundamental to know that the use of function point to determine an estimate effort must be a help for the project manager in the estimation process but it never can substitute the estimation process. We cannot think of function point as the solutions of all the problems, they have to be considered in the right way, without unreal expectations. It's necessary to apply other estimate methods (bottom up, Delphi) for a better estimate.

For the future we are thinking to introduce in our estimation prevision model the possibility to apply different productivity factors and risks on parts of the software to estimate. In many cases a project has in fact several different modules (a management module, a statistical module, a reports module….), each of them with their characteristics and problems, and so with their own productivity.

The collecting data and the experience will undoubtedly improve our estimation prevision model, at least we hope!

The success of a system metric is, above all, a matter of culture and, in any case, we should never switch off our brain!

10. References

[1] Capers Jones - Programming Languages Table – 1996
[2] ISBSG – Repository Data - Release 9 - 2004
[3] LGC-FP Linee Guida per l'uso contrattuale dei Function Point – GUFPI - 2006
[4] Function Point Analysis for Software Enhancement – Guidelines - version 1.0 – NESMA - 2001
[5] Barry Boehm - COCOMO II - 2001
[6] COCOMO II – 2001