How to Avoid Traps in Contracts for Software Factory Based on Function Point Metric

Claudia Hazan[1] claudinhah@yahoo.com

Eduardo A. Oliveira[2] eduaopec@yahoo.com.br

José Roberto Blaschek[3] blaschek@attglobal.net

[1] Serviço Federal de Processamento de Dados (SERPRO) – CETEC
Rua Teixeira de Freitas,31 – 4º andar – Lapa
Rio de Janeiro –RJ – Brasil

[2] Serviço Federal de Processamento de Dados (SERPRO) – SUPDE
Rua Teixeira de Freitas,31 – 9º andar – Lapa
Rio de Janeiro –RJ – Brasil

[3] Fundação COPPETEC
Centro de Tecnologia – UFRJ – Bloco H sala 203
Rio de Janeiro – RJ - Brasil

Abstract

There is a growing demand for development and maintenance projects of software. The organizations have contracted software factories in order to support this demand. However, the contract management is not a trivial task. Several organizations have used the Function Point (FP) metric as a basis of the contracted payment. In fact, the use of FP assists on the contracts management. The contractor can manage the software product required and received instead of managing the contracted teams. But, the establishment of a contract based on FP is not a silver bullet that solves all conflicts between contractors and contracted. This paper presents the main errors observed during the validation of FP counting in contracted projects; discusses some problems related to contracts based on FP metrics and gives suggestions to solve it. The main objective of this work is to promote the correct use of FP counting rules and to present the Function Point Analysis (FPA) limitations, such as: maintenance projects counting, defects treating, schedule estimation, changing requirements issues related to software contracts.

Key Words:

Function Point, Function Point Counting, Function Point Analysis, Software Contracts

1. Introduction

Currently, the organizations depend on useful information in a structured way to meet specific objectives, to improve their performance in their business. Information technology (IT) plays a strategic role in business process reengineering and in creating information systems. It is important to highlight that an information system increases the performance of an organization's human resources and improves the quality of its products.

In this context, there is a growing demand for development of new systems and maintenance of existing applications. This scenario has contributed to an increase in the IT backlog of organizations. Thus, contracting external services at software factories is the solution for the organizations to supply the demands of software project.

In many Brazilian's organizations, especially in the government area, the Function Point (FP) metric is used as a basis for software contracts pricing decisions. As a matter of fact, it is very difficult to manage contracts per hour with a model of software factory. Some companies have begun to use others functional metrics sizes, such as: the number of use cases and use case points (UCP). These metrics are easier to calculate than FP, but they are very subjective, depending on the system requirement modeling, i.e., the way that the software engineer documents the requirements. The use of FP metric offers a number of benefits, such as: the availability of the Counting Practice Manual (CPM) [IFPUG, 2005] which has well-defined rules for the FP counting, requirement modeling independence and user view[1] oriented.

The problem of contracting out the software factory seems to be solved in a very simple way: include a contract clause specifying that software project price will be based on the FP metric, and beside this, another clause that defines the CPM as basis for FP counting. In order to make the contract clearer, the current version of CPM must be included. Many companies have done it, but are facing problems and conflicts with software factories contracted. Why does it happen?

The answer is that there are many issues to be covered in a software factory contracts that are not included in the CPM, such as: How to measure the frequent requirements changes that plague software projects? How to measure the different kinds of maintenance projects besides enhancement projects? How to address issues of different productivity rates according to programming languages and other non-functional requirements specific to each project in software contracts based on fixed price by FP?

The purpose of this work is to reduce the conflict between contractors and contracted. This conflict is one of the main risks of software projects. This paper answers questions about the use of the FP metric in software contracts, shows some common mistakes about FP counting, and discusses some concepts inherent in FP counting, such as: logical data vs. physical data, elementary process, external output vs. external inquiry and data conversion functions. This paper discusses also some common problems of software contracts based on the FP metric and suggests solutions.

This paper is organized in the following way: Section 1 presents the motivation and the purpose of this paper; Section 2 shows an overview about Function Point Analysis; Section 3 presents common FP counting mistakes observed during FP counting validation; Section 4 shows the main problems found in the software contracts and give some suggestions for solving them. These problems were identified in analyses of several software contracts of Brazilian organizations, especially government organizations; Finally, Section 5 concludes this paper and presents recommendations to future works.

2. A Function Point Analysis Overview

The FP metric was created by Allan Albrecht [Albrecht, 1979] in order to reduce the difficulties associated with the use of Line of Code (LOC) as a software functional size metric and to support the effort prediction of the software development process [Albrecht, 1983]. In 1986, a research of Quality Assurance Institute showed that the FP is the best metric to the quality and productivity measurements of information systems [Perry, 1986]. In 1993, FP has become the most used and studied metric in the history of Software Engineering [Jones, 1995]. While other metrics have emerged in nineties, such as: Full Function Points [Abran, 1997] and Use Case Points [Karner, 1993], the FP metric continued being the most used in the software industry. In 2003, the ISO / IEC 20926 recognized the Unadjusted Function Point metric as an international standard [Dekkers, 2003]. Nowadays, the endeavor for the adherence to the software quality models has also collaborated for the extensive use of FP in the software industry. In fact, the metric supports the deployment of several of CMMI practices, as described by Lipton [Lipton, 2000], Dekkers [Dekkers, 2002] and Hazan [Hazan, 2003]. This section has the purpose of presenting an overview of FP counting, based on CPM [IFPUG, 2005].

---

[1] A user view represents a formal description of the user's business needs in the user's language [IFPUG, 2005].

Function Point is a functional size metric of software projects. The functional size is defined as “a size of the software derived by quantifying the Functional User Requirements” [Dekkers, 2003]. The user function requirements are a subset of software project requirements. These requirements are descriptions of the several functions that clients and users need the software to provide, typically focus on “what the system must do” i.e. in the user business process tasks required [Sommerville, 2007] [IFPUG, 2006]. The main purposes of the Function Point Analysis (FPA) are the following [IFPUG, 2005]:

- Measure functionality that the user requests and receives;
- Measure software development and enhancement independently of technology used for implementation.

The unadjusted function point count (UFPC) reflects the specific functionality provided to the user by the software project. The UFPC consist on mapping the user functional requirements to FPA function types: Internal Logical File (ILF), External Interface File (EIF), External Input (EI), External Output (EO), and External Inquiry (EQ), as described below (Figure 1).

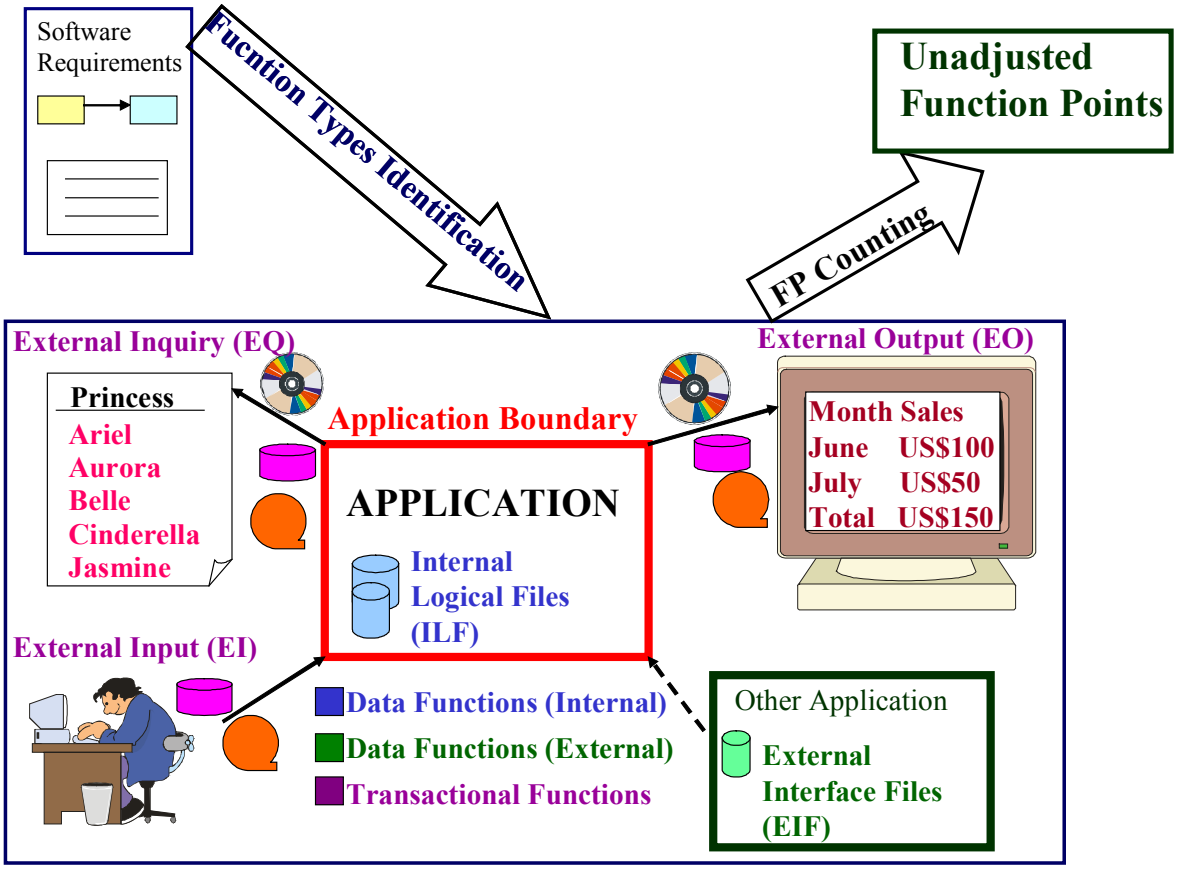

Figure 1: The Five Function Types of the Function Point Analysis

- The Internal Logical File is a data logical group, maintained within the boundary of the application, through one or more elementary processes.
- The External Interface File is a data logical group, referenced by one or more elementary processes of the application. However, they are kept within the boundary of another application.
- An External Input is an elementary process that processes data or control information that comes from outside the application boundary. The primary intent of an EI is to maintain one or more ILFs and/or alter the behavior of the application.
- An External Output (EO) is an elementary process that sends data or control information outside the application boundary. The primary intent of an EO is to present information to a user[2] through a processing logic which must: contain at least one mathematical formula or calculation, or create derived data, or maintain ILFs, or alter the behavior of the application.
- An External Inquiry (EQ) is an elementary process that sends data or control information outside the application boundary. The primary intent of an EQ is to present information to a user through only a retrieval of data or control information from an ILF or EIF.

Each identified function has an associated complexity: Low, Average or High and a contribution to unadjusted function point counting, based on its function complexity [IFPUG, 2005].

The adjusted function point counting considers the value of adjustment factor (VAF). This factor is based on the degree of influence of the 14 general systems characteristics that rate the general functionality of the application being counted. Each characteristic has associated descriptions that help determine the degree of influence of that characteristic [IFPUG, 2005]. The VAF can vary from 0.65 to 1.35.

It is important to highlight that the use of the procedure for FP counting, described on CPM [IFPUG, 2005], imply in the existence of the application logical design. Therefore, in the earlier phases of the software life cycle, the Function Points can't be measured but estimated. The following section presents the main mistakes found in the reviews of FP estimating and counting performed by the author.

---

[2] A user can include a hardware device, another software application or the user himself.

3. Ten Mistakes Found in Function Points Counting

Several mistakes have been found during reviews of FP estimating and counting, performed by the authors. The main cause of these mistakes is the lack of knowledge about FPA concepts and rules described in CPM. In Brazil, there are a lot of FP counters that don't have a full understanding about CPM counting rules. These people read some papers, get the overview of FPA technique, and start to count FP without having enough skills. Sometimes, these people publish papers, especially in academic congresses, spreading wrong concepts about FPA. The author of this work is worried about disseminating FPA in the correct way. It is important to count FP correctly, following CPM rules. Wrong FP counting generates useless information and costs. These mistakes are often found in academic papers and usually in the industry are described in the next section.

❶ Error in the definition of Functional Size X Development effort

Let's start with the most observed error in the industry and academy. Observe a dialogue between a software metrics consultant and a project manager: The consultant says: "I am researching about functional size estimations methods based on FP." The manager replies: "Why FP? Why don't you research about COCOMO?" The COCOMO doesn't estimate project size. Many people think that FPA is a method for estimating schedule, costs and effort. It is wrong. The FP is a functional size metric which is used to estimate only the size of the software projects. It is important to emphasize that the size estimation is an important input for derivation of the cost, effort and schedule estimations [SEI, 2006]. Thus, FP constitutes an input to cost and effort estimate models, for example COCOMO II.

❷ Error in the usage of Function Point Counting formulas described in CPM

This error may occur as a consequence of the previous error, described above. In order to maximize returns in contracts of Fixed Price by FP, some contracted organizations want to "increase" artificially the FP counted, due to a great development effort of some complex software projects. So they create components for the CPM formulas, as other adjustment factors to reach these objectives. For instance, if the database of the project is complex, there is an adjustment factor of 1.2 to multiply by adjusted function points. This is a wrong practice. The FP counting must be based on CPM counting rules and formulas. This kind of error can also occur due to the lack of knowledge of the CPM, as in the example published in the paper of WER (Workshop of Requirement Engineering) [Carvalho, 2005]. The paper shows the FP counting of a Web System, where initially the calculation of unadjusted function points presented was 130 UFP. Afterwards, in the calculation of the adjusted function points of the development project, is mentioned that: "a function of data model

conversion was developed, with an effort estimated around 10% of the total. Therefore, the Conversion Function Point (CFP) was calculated as: CFP = 10% x UFP = 0.1 x 130 = 13. And the Adjusted Function Point of the Development Project (DFP) is calculated by the formula: DFP = (UFP + CFP) x VAF = (130 + 13) x 1.14."

There is a mistake in the calculation of the CFP component of the DFP formula, presented above. Although the published paper by Carvalho [Carvalho, 2005] references the use of CPM to count the adjusted function points, there isn't a rule in CPM that considers effort within the FP calculation. Of course, FP is a functional size metric, independent of the effort to develop the application. The functions of data conversion must be identified and counted according to CPM counting rules. The functions of data conversion which are associated with the initial data loading are run only once, during the application installation task. Generally, they are classified as EIs. Sometimes, these functions are also identified as EOs, associated with the load control reports containing summarized and totalized data. These reports must be required by the users to be counted.

❸ Error: External Inquiries X External Outputs

This mistake is caused by the lack of knowledge of the identification rules of External Outputs and External Inquiries. It is very common too. Some people count FP based on the following "rule": "all reports must be classified as External Outputs and all inquiries are External Inquiries". Unfortunately, this simple "rule", easy to apply, is wrong. Notice that this "rule" is not compliant to the CPM counting rules. For instance, consider a functionality of "Students Inquiry", which presents a student list containing the name of the student and the age of the student. This function retrieves the student birth date in the ILF of Students to compute the student age. The inquiry function: "Students Inquiry" must be classified as EO, due to the student age calculation. Another example, consider a functionality "Report of Students", which retrieves the student name and identification code in ILF of Students and presents the information in alphabetical order. This report must be classified as an External Inquiry. It is not an External Output because it doesn't contain calculation, derived data, ILF updating or system behavior changing. The processing logic "data is resorted" may be performed by EQ.

❹ Error in the identification of Logical Files

This error is frequently observed in the reviewing of FP counting. This mistake was found in the majority of the FP counting, during a recent auditing of FP counting performed by the author. It is common that many physical entities are counted incorrectly as Internal Logical Files or External Interface Files. This is another wrong "rule":"all physical table or file

must be counted as ILF or EIF". This wrong "rule" is not compliant to CPM counting rules. It is important to emphasize that files in FPA are group of logically related data or control information and not physical tables or files. Thus, some physical tables or files are not counted as ILF or EIF, some may be a part of ILF or EIF, and others are not counted. This error frequently results in a FP counting bigger than the real size of the projects.

❺ Error in the Elementary Processes Identification

The difficulty in elementary process identification has been frequently observed in the industry. The most common error is the identification of sequential activities as independent elementary processes. The wrong "rule" is: "each application screen is an elementary process". Notice that "an elementary process must be self-contained and leave the business of the application being counted in a consistent state" [IFPUG, 2005]. It means that an elementary process must be an independent activity. For example, suppose a functionality of inquiry which contains one input parameter screen and another output screen to present the retrieval results. Some people count this functionality as two elementary processes: one for the parameter screen and another for the screen of results. This practice is wrong. The two screens take part of a unique inquiry activity, i.e. a unique elementary process of inquiry. Only one of these screens doesn't leave the application business in a consistent state. An automatic e-mail sending, that makes part of the application functionalities, is frequently counted incorrectly such as an independent elementary process. The right way to count it is as part of the elementary process associated with the functionality. Let's present a hint to help on the identification of elementary processes: Sequential functionalities make part of the same elementary process and independent functionalities make part of different elementary processes. Another wrong "rule" related to this error is the following: "all requirements identified in the software requirement document, for instance, as relation of use case, is an elementary process". This wrong "rule" has been observed in some FP counting. For example, suppose the use case "Update Students Information". There are functions such as: "Cancel", "Exit" and "Clear", which are not elementary processes, because they are related to non-functional requirements such as usability. These functions provide support to: the application navigation and the user efficiency improvement. In this example, the functional requirement "Update Students" Information is the unique elementary process.

❻ Error in the Identification of Implicit Inquiries

This error is also related to the elementary processes identification. Some FP counters don't identify the functions of implicit inquiry associated to data updating. This mistake happens when the FP counter perform the counting, based on application screens instead of application functional requirements. Consequently the implicit inquiry (data edition associated

with data updating) is not counted. Notice that the data inquiry which precedes the updating is an elementary process, independently of the data updating process. The user may only inquiries the data without updating anything. Therefore, the implicit inquiries must be counted, since they haven't been counted already, according to the rules of uniqueness of EQ or EO, described on CPM.

❼ Error in the Determination of the Value of Adjustment Factor

Some Brazilian organizations don't use the adjustment factor in the FP counting. The unadjusted function point counting is considered in few software contracts and some other use a fixed adjustment factor. It's a mistake to use a fixed adjustment factor for all software projects, because the value of adjustment factor must be calculated for each development or enhancement project according to CPM counting rules. In addition, many mistakes have been observed in the determination of the degree of influence of the fourteen general systems characteristics (GCS) in the value of adjustment factor calculation. In spite of improving definition of GCS observed in the CPM release 4.2.1, there is still much difficulty in identifying the degree of influence (DI) of the characteristics. For instance, the characteristic of reusability is one that has presented evaluation mistakes. It is very common the FP counters considers the DI = 4 or 5 due to reuse of components from other applications and having internal reuse only. In this example, the correct DI = 1.

❽ Error in the Calculation Formula Implemented in Spreadsheet of FP Counting

Many organizations develop spreadsheets to automate the FP calculations. It is important to observe that FPA function types don't have the same functional contributions for FP counting. The functional contributions of the data functions: ILF and EIF are different; the functional contributions of the transactional functions: EIs and EQs are the same, but the functional contribution of the EO is different. In an actual case, the FP counting spreadsheet of an organization has only three functional contributions for all function types: Low - 3 FP, Average - 4 FP and High - 6 FP. In another actual case, the functional contribution of the ILF and EIF was the same: low - 7 FP, Average - 10 FP and High - 15 FP. It is wrong. The correct functional contribution of the function types are the following [IFPUG, 2005]:

Function Type	Complexity Low	Complexity Average	Complexity High
Internal Logical File (ILF)	7 FPs	10 FPs	15 FPs
External Interface File (EIF)	5 FPs	7 FPs	10 FPs
External Input (EI)	3 FPs	4 FPs	6 FPs
External Output (EO)	4 FPs	5 FPs	7 FPs
External Inquiry (EQ)	3 FPs	4 FPs	6 FPs

❾ Error in the Determination of Complexity of Changed Functions in Enhancement Projects

Another very common error in the FP counting of enhancement projects is the difficulty in the identification of complexity of changed functions. For example, consider the change in an EI of high complexity. The change is: add one more DET (data element type) to the function. Notice that this function continues identified as high complexity. In practice, this function is counted incorrectly as simple several times, because the change was the addition of only one DET in an implemented function. The CPM 4.2.1 mentions that the CHGA (FP of functions that were modified by the enhancement project) considers the new functionality available to the user by the application. Even tough, the modification in a high complexity function is small; it must be classified as high complexity since it remains with a complexity that is considered high.

❿ Error in the CPM Use: Function Point Counting of Maintenance Projects

The majority of software factory contracts based on FP mention that the size of the development and maintenance software projects are measured using the method of FP counting, described in the CPM. However, the CPM considers the FP counting for development and enhancement projects only. The other kinds of maintenance project, such as corrective maintenance, cosmetic maintenance, adaptive maintenance without changing business requirements, are measured incorrectly following the CPM enhancement formula. According to CPM 4.2.1, these modifications requests can't be measure in FP. In fact, the FP metric is a functional size metric. These maintenance projects don't have functional size, because there aren't changes in the functionalities (add, change or delete functionalities) based on the user viewpoint, considering business requirements. The next section presents the recommendations to deal with maintenance project in software contracts based on FP.

4. Software Contracts Problems and Solutions

The FP metric has been used by most of Brazilian government organizations and by several other enterprises too, as a monetary unit ($ / FP) in their contracts of software developing and maintenance. In this kind of software contract, named contract of fixed price by FP, the FP metric represents an asset to the client. This contract type permits a best risk balance between contractor and contracted [Aguiar, 2000]. This section has the purpose of presenting some hints to avoid traps in software factory contracts base on fixed price by FP.

❶ Getting a Software Requirement Document with Quality

As mentioned, the FPA measures functionality that the user requests and receives [IFPUG, 2005]. The software requirements document constitutes the common agreement between contractors and contracted. It is essential that Term of Agreement associated with the requirements document is signed by the contractor. In addition, the contractor should ensure the quality of the software requirement document sent to the software factory. Notice that if the contractor organization provides a software requirement document with defects, for instance missing information, then the organization will receive a software product without the expected functionality and will have to pay for it. The Requirement Engineering presents several techniques to support the requirements verification and validation. However, most of these techniques are very expensive. This work suggests the use of a Function Point estimation method, called CEPF, described by Hazan [Hazan, 2005a]. This method supports to the estimator finding defects in requirement document, while estimating the project size in FP, without additional cost or effort, as demonstrated in [Hazan, 2005b]. Considering the auditing on FP counting, it is important that the requirement document, the FP counting document and the term of agreement remain together in the same place.

❷ Establishing Rules to the Treatment of the Scope Creep

The Requirement Engineering and the industry recognize that the software projects requirements don't remain "frozen" until the project conclusion. The software requirements evolve since their conception until after the system is in operation, due to several factors described by Kotonya [Kotonya, 1998]. Thus, it is important that a contract based on FP establishes how to deal with changing requirements. This work suggests the following: to establish in the contract a percentage to each software process task, for example: requirements: 20%, design: 10%, implementation: 50%, test: 15%, installation: 5%[3]; and tracking the progress of each functional requirement identified in the software project requirement document. Thus, when a requirement is changed, it should be identified in which software process task the requirement was modified. The next step is to retrieve the FP counting of the original requirement in order to apply the percentage of the process tasks finished. For example, suppose a changing in a business rule of a "report of clients", identified as EO – average – 5 FP, at the end of the requirements phase. Following to the suggestion above, it should consider the new changed requirement, EO –average – 5 FP, plus 20% of the original one (1 FP) to the contracted payment (total: 6 unadjusted FPs).

---

[3] These percents were achieved based on a hypothetical software process. It is important to emphasize that each organization should define these values, based on its own software process tasks and the effort consumed in each one.

❸ Establishing Clauses of Quality Assurance

As mentioned, the FP counting considers the functionality requested and received by the user, and then the FP of the delivered functionality must be considered to the contracted payment, only if the functionality received doesn't present defects. However, the following problem may happen: the contracted delivers functionality containing bugs, then the contractor complains, making the contracted correct these bugs, but the contracted gives back the functionality to the contractor with bugs diverse from the former, thus setting a long enduring cycle. Notice that this situation may generate an enormous delay in receiving of the end software product, in addition to the contractor client's loss of trust in the contractor. Thus, it should be established contractual clauses to ensure the deliver of a software project with quality. For example, to include in the contract a clause of penalty, if the software project contracted has more than 0.3 defects per FP. It is important to define in the contract what are considered defects, such as: documentation defects, non-structured source code, inconsistencies among documents, etc. It also may be established severity degrees to the defects.

❹ Establishing Clauses of Schedule and Assurance of a Delivery Rate

Many contractor organizations include contractual clauses about productivity control. For example, the software factory contracted must have a productivity rate of 12 HH/FP. Sometimes, the contractor asks for the contracted to report its productivity rate. However, it is not correct. The productivity of a software factory is a strategic data of the contracted organization. Maybe, the contracted doesn't want to disseminate its productivity rate. The problem which the contractor wants to solve is related to the demands not received within the schedule. This problem can be solved in another way, without the control of the contracted productivity. In fact, the contract is based on FP; the management of the contracted productivity rate doesn't make sense. It should establish in the contract the estimation model used to get the schedule based on estimated FP. To ensure that the established schedule will be met by the software factory contracted then it should contain a penalty clause, considering the project schedule delays. For the organizations which don't have a schedule estimation model, it suggests the use of the Capers Jones formula described on [Jones, 2007]. This formula is simple to apply and consists on the exponential of the project size measured in FP to an exponent (t). Hazan [Hazan, 2005a] presents hints to identify the exponent t to several kinds of software project. It is important to underline that the formula is appropriated only to projects greater than 100 FP. To smaller projects, the contract schedule may be fixed in the contract. Another important point to consider is: the contracted organization doesn't develop any project to the contractor in order to force the increase of the

FP contracted price. Thus, it is important to define as a contractual clause the establishment of a minimal delivery rate of FP/month, for instance 500 FP/month. It should consider a penalty clause also related to this issue. The establishment of a minimal and maximal delivery rate is important to the contracted software factory too. Probably, the software factory uses this information to dimension their teams in order to support the contractor's demands.

❺ Establishing the CPM as a basis to count FP instead conversions

In Brazil, considering software contracts based on FP, it is common that some contracted organizations use FP as monetary unit to the payment of several activities related to software, but without FP counting. For example, course of FPA, mentoring of programming languages, etc. Another actual situation which has happened is the following: although the contract is based on FP, there isn't FP counting for any contracted project. The payment to the contracted software factory occurs in the following way: the software factory, through its hours appropriation process (time sheet), sums the total of allocated hours to support the contractor's demands; the contractor pays the software factory, observing the total of hours presented in the time sheet, the total of hours is divided into the FP price according to the contract. For example, consider the following situation: a project which consumed 10,000 hours of effort based on the project timesheet, the value of US$50 per hour. The total price that the contractor wants do receive is US$500,000 (50 x 10,000). In addition, suppose that the FP price according to the contract is US$ 500. Thus, the contractor reports that the project size measurement is calculated by the following way: US$500,000 / U$500 = 1000 FP. It is a wrong practice; the FP must be counted based on CPM counting rules. There isn't a formula to convert other metrics to FP. Sometimes, the contracted software factory converts hour to FP. For example, 12 work hours equal 1 FP. It is incorrect too, because the effort hour/FP isn't fixed. The effort depends on others non-functional requirement in addition of the FP size. The consequences of wrong FP counting are the following: not adequate payment, due to the payment based on FP counting; and the generating of wrong indicators, such as quality indicators – defects /FP and productivity indicators hours /FP. This scenario is very serious. Although the contract is based on FP, it is put in operation as a contract based on hour allocation, without contractor control of the allocated hours. Thus, it results on several management risks only to the contractor. It is important to emphasize a real case analyzed, a contractor organization used FP as a monetary unit to pay the outsource work team allocated on its client. The contract should have been established based on work force allocation, however incorrectly the organization used FP to do the contracted payment. At the end, the contracted presented an FP counting document with incorrect information to reach to the number of FP to justify the team with persons allocated full time in the client. Several

serious mistakes were found in the counting document, such as: getting the total FP contribution of EIF, without specifying any EIF in the counting spreadsheet and the software requirement document too. Every month the software factory contracted received, for instance 500 FPs to pay the allocated work force, independently of the number of FP delivered by them. This is a real case about conversion hour to FP. Another situation, less common, is the conversion Use Case Points (UCP) to FP, i.e. the people count UCP and "define" formulas to convert UCP to FP. It is important to emphasize that FP counting must be performed based on CPM instead of other conversion formulas. There isn't formula to convert hour to FP or UCP to FP. Contractor and contracted should implement a Software Metrics Office with FP counters who have the CFPS (Certified Function Point Specialist) to perform auditing in the FP counting. If the organizations don't have employees CFPS, then they can outsource this service to a software metrics consulting organization.

❻ Establishing rules to addressing maintenance projects

Several Brazilian organizations have established their software projects contracts based on FP, reporting that: "the FP counting is performed according to the current release of CPM". However, a problem emerges; the FP counting is applied only to development and enhancement projects according to CPM. Then, one question emerges: how to address the other maintenance projects in contracts based on FP counting? The contractor could send to contracted software factory only enhancement projects. It is simple, but impracticable. For example, suppose a project to change the caption of application screens. This maintenance project has zero FP. This project isn't an enhancement, it is a cosmetic maintenance. However, there is an effort to implement this project by the contracted software factory. It should establish an hybrid contract based on FP to development and enhancement projects and based on hours allocated to others projects not considered by CPM. But, in practice it is a little bit complicated; the contractor organizations prefer the contracts based on FP, because they don't want to have effort to manage the work hours of contracted organization team. Thus, the recommended solution is to count the FP of these projects, setting some rules and formulas in the contract. The first step is to identify all types of maintenance project which the organization intends to contract, for instance:

- Corrective: when the contractor requests to the contracted to correct faults in legacy systems, which aren't developed by it. The IEEE Standard for Software Maintenance [IEEE, 1998] defines this type of maintenance as "reactive modification of a software product performed after delivery to correct discovered faults".

- Cosmetic: when the contractor requests to the contracted to change static text in screens, such as screen captions or organization logotype.

- Adaptive: when the contractor requests to the contracted to change technical requirements, for instance a release update or the web system needs to run in another browser. The IEEE Standard for Software Maintenance [IEEE, 1998] defines this type of maintenance as “modification of a software product performed after delivery to keep a computer program usable in a changed or changing environment”. CPM considers that adaptive maintenance is initiated by business requests to add, change and/or delete business functionality. Thus, CPM considers this kind of maintenance as enhancement. In this work, the concept of adaptive maintenance is different of CPM definition. This work considers that an adaptive maintenance project includes modifications required to meet changing technical requirements only.

- Special Data Service: when the contractor requests to the contracted to develop a functionality to create a report or to do a data loading. This functionality won’t be incorporated to the application.

It is important to define the concept of these maintenance projects in the contract too. The next step is to establish a percentage of FP to changed functionality. For example, consider the percentage of 10% for changed functionality in a cosmetic maintenance project. Thus, if the contractor requests a change in the caption of a screen, which is contained within inquiry functionality, identified as: EQ – Low – 3 FP. Then, the size of this project, supposing the VAF = 1, is: (3 x 1) x 0.10 = 0.3 Adjusted FPs. This percentage must be established, based on experiments performed in the contractor organization environment and included in the contract with the agreement between contractors and contracted.

5. Conclusion

It is very concerning certain organizations are using incorrectly the FPA concepts, maybe due to a lack of knowledge on CPM counting rules. It is important to count FP correctly following the CPM, especially when the organization is using FP as monetary unit in software contracts. This work has presented some results of researches performed on common errors in FP counting observed during validations of FP counting of contracted projects and problems in software contracts based on FP.

This paper recommends the use of FP as a basis to the software factory contracted payment. However, contracts based on fixed price by FP are not a good solution, because the development effort depends on FP and a number of non-functional requirements. FP is a functional size metric, thus it doesn't consider non-functional requirements. Therefore, it is difficult for a software factory to work with a fixed price by FP.

The best solution is to do contracts based on price by work hour. However, the work hour must be defined based on FP in order to avoid the contractor management of the contracted work hours. The proposal is the following: counting application FP; converting FP to hours, based on defined model, considering all types, programming languages and others non-functional requirements significant to the organization; paying by hours. Notice that FP continues being the basis to the contract software, however there is a variable price to FP. It is correct because the cost estimation depends on the effort estimation.

As a future work, it suggests the definition of a model to convert FP to hours, based on non-functional requirements such as: usability, security, programming language, development process, type of project etc. This model is similar to an effort estimation model, but it must be specific for each organization. It is important to use historical data of concluded projects and experiments to define the organization model.

References

[Abran, 1997] ABRAN et al. *Full Function Points: Counting Practices Manual Procedure and Counting Rules.* Technical Report, Universite du Quebec, Montreal, November 1997.

[Aguiar, 2000] AGUIAR, M. *Contratando o Desenvolvimento com Base em Métricas*. Newsletter - O Melhor do BFPUG, Setembro 2000. Available at: http://www.bfpug.org/fpug_rio/Newsletter/0009/Contratando_Desenvolvimento_Base_Metricas.html
Accessed: 01/06/2008

[Albrecht, 1979] ALBRECHT, A. *Measuring Application Development Productivity*. Proceedings of the IBM Application Development Symposium Monterey, California, October, 1979, pp 83-92.

[Albrecht, 1983] ALBRECHT, A.; GAFFNEY, J. *Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation*. IEEE Transactions on Software Engineering, Vol. SE-9, no. 6, Nov, 1983 pp. 639-648.

[Carvalho, 2006] Carvalho, A.; Chiossi, T.; Drach, M. Aplicabilidade de Métricas por Pontos de Função a Sistemas Baseados em Web. 9th International Workshop on Requirements Engineering (WER2006), Rio de Janeiro, Brazil, July 2006, pp.109-115.

[Dekkers, 2002] DEKKERS, C. *How Function Points Support the Capability Maturity Model Integration*. Crosstalk– The Journal of Defense Software Engineering, February 2002 pp.21-24.

[Dekkers, 2003] DEKKERS, C. *Measuring the "logical" or "functional" Size of Software Projects and Software Application*. Spotlight Software, ISO Bulletin May 2003 pp10-13.

[Hazan, 2005a] HAZAN C.; STAA, A. v. *Análise e Melhoria de um Processo de Estimativas de Tamanho de Projetos de Software*. Monografias em Ciências da Computação nº 04/05, Departamento de Informática PUC-Rio, ISSN 0103-9741, Fevereiro 2005.

[Hazan, 2005b] HAZAN, C.; BERRY, D.M.; LEITE, J.S.P. *É possível substituir processos de Engenharia de Requisitos por Contagem de Pontos de Função?* 8th International Workshop on Requirements Engineering (WER2005), Porto, Portugal, June 2005, pp. 197-208.

[IEEE, 1998] IEEE Computer Society. *IEEE Standard for Software Maintenance*. IEEE Std 1219, 1998.

[IFPUG, 2005] IFPUG. *Counting Practices Manual*. Version 4.2.1, January, 2005.

[IFPUG, 2006] IFPUG Counting Practices Committee (CPC). *Practical Guidelines for Identifying Unique Elementary Processes*. 2006.

[Jones, 1995] JONES, C. *Software Challenges: Function Point: A New Way of Looking at Tools*. Computer, August, 1995. pp. 66-67. Available at: http://dlib2.computer.org/co/book/co1995/pdf/rx102.pdf

Accessed: 01/10/2006

[Jones, 2007] JONES,C. *Estimating Software Costs – Bringing Realism to Estimating*. 2nd Edition, Mc Graw Hill, New York, 2007. New York.

[Karner, 1993] KARNER, G. *Use Case Points - Resource Estimation for Objectory Projects, Objective Systems*. University of Linköping, Sweden, 1993.

[Kotonya, 1998] KOTONYA, G.; SOMMERVILLE, I. *Requirements Engineering: Processes and Techniques*. John Willey & Sons Ltd, 1998.

[Lipton, 2000] LIPTON, D. *Function Points and the SEI Capability Maturity Model*. Q/P Management Group, 2000. Disponível em: http://www.qpmg.com/seicmm2.htm Acesso em: 01/10/2006.

[Perry, 1986] PERRY, W.E. *The Best Measures for Measuring Data Processing Quality and Productivity*. Quality Assurance Institute Technical Report, 1986.

[SEI, 2006] Software Engineering Institute. *CMMI for Development*. Version 1.2, Pittsburgh, PA 15213-3890, August 2006.

[Sommerville, 2007] SOMMERVILLE, I. *Software Engineering*. Pearson Education Limited, 8th Edition, 2007.