Estimating the size of web applications by using a simplified function point method

Edilson J. D. Cândido*
University of São Paulo
Inst. de Ciências Mat. e de Computação
São Carlos, São Paulo, Brazil
edilson@icmc.usp.br

Rosely Sanches
University of São Paulo
Inst. de Ciências Mat. e de Computação
São Carlos, São Paulo, Brazil
rsanches@icmc.usp.br

Abstract

Software size estimation is a key factor to determine the amount of time and effort needed to develop software systems, and the web applications are no exception. In this paper a simplified way of the IFPUG (International Function Point Users Group) function points based on the simplification ideas suggested by NESMA (Netherlands Software Metrics Association) to estimate size of management information systems is presented. In an empirical study, twenty web applications were analyzed. The estimates using the simplified method were close to the ones using the IFPUG detailed method. Based on the results, it was possible to establish a simplified method to estimate the size of web applications according to the development characteristics of the studied company.

1 Introduction

A software company needs an effective management of the software process to construct applications maintaining the schedule and the quality [28]. The software process includes quantifying and qualifying projects, products, processes and resources of software [12].

In terms of the perspective of the project, it is widely accepted that software size estimates are fundamental to determine the costs (regarding the amount of time and effort) of a software project [3], [6], [13], [26]. Software models like SW CMMI, SPICE and ISO 12207 discuss the importance of software size estimates as one of the project planning activities.

*Supported by CNPq-Brazil.

The web applications are not an exception of the premises cited above. However, some research carried out by the *Ministério da Ciência e Tecnologia* (Ministry of Science and Technology) shows that in Brazil, only approximately 29% of the software companies effectively do software size estimates [9]. Although there is no study exclusively related to companies which develop web applications, the fact that 45% of the companies analyzed develop web applications [10] indicates that these companies are in a similar situation.

A study of a simplified method to estimate the software size of web applications could help change this situation. Thus, the purpose of this paper is to describe a simpler, faster and more consistent method to help carry out effective software size estimates. Moreover, it encourages the use of software size estimates to obtain quality in the software process development.

The simplified method was created using a case study in a small company [11] whose main work is the development of web applications. It is based on counting function points from the IFPUG and on the ideas suggested by NESMA to determine simplified formulas of software sizes to measure management information systems.

Although the NESMA method, which is effective in estimating the size of management information systems, did not work satisfactorily concerning the web applications, it helped to obtain the simplified method, in which the media error was 4% when compared with the detailed method from IFPUG. This result enables us to use the simplified form, which is simpler and faster, instead of the detailed method to estimate the size of the applications developed in the studied company.

Other companies which have similar characteristics

of software development to the studied company can verify the utility of the method in terms of their development environment or use them as a role model to create their own methods.

This research was undertaken using data from twenty web applications from the company. The rest of the article is organized as follows: section 2 presents a short description of a software size estimate. Section 3 shows the simplified method and the results which were obtained. In section 4, the modeling of a tool to support the process is described. Finally, in section 5, the scope for future research and the conclusions are given.

2 Software Size Estimation

Software size estimation is considered as a fundamental activity regarding software management tasks. Work planning and subsequent estimates of the amount of time and effort are predicted based on the size of the software [25], [22]. The lines of code (LOC) and the functionalities of the software are two measures which are often used to determine the size of an application.

LOC are direct measures that can easily be counted and manipulated [27]. There are several ways to calculate the LOC. Jones [6] suggests eleven possible variations for counting the LOC of a program. Fenton et. al [12] presents some variations and the implications that they can cause. Still according to Fenton et. al, the most accepted definition of LOC is from Hewlett-Packard, where each program is considered as a simple list of archives and commentaries and blank lines are removed.

There is much discussion about the use of LOC in software size estimates. The critics are based on the arguments that lines of code are dependents of programming language and require details that can be difficult to overcome before the analysis and project have been finished [3], [5].

On the other hand, size estimates based on the functionalities of the software define elements that can be counted previously, in the beginning of the software development. The concepts of this type of counting were first made public by Albrecht in 1979 [21]. Since then, they have been refined, mainly after the creation of organizations exclusively directed towards the development of the technique [20], [24].

Last year, four methods (IFPUG, NESMA, MarkII, COSMIC-Full Function Points) of software size estimate were approved as an ISO standard for functional software size measurements, called ISO\IEC 20926:2003 (IFPUG), ISO\IEC 24570:2003 (NESMA), ISO\IEC 20968:2003 (MarkII), and ISO\IEC 19761:2003 (COSMIC-FFP). These four methods of functional software size estimation will now be presented.

2.1 IFPUG Function Points

The IFPUG [20] is a non-profit organization which was established in 1986 that promotes the use of function point analysis to measure the functionality provided by software. The IFPUG maintains the Function Point Counting Practices Manual (CPM), currently in version 4.1.1 [17], and the aim is to standardize the counting process.

The procedure of counting function points promoted by the IFPUG and described in the CPM has seven steps, as can be seen in figure 1.

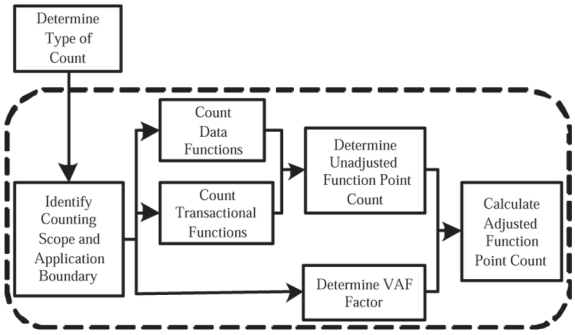

Figure 1. FP Counting Procedure

The counting procedure is described in the following summary:

1- Determine type of count: Development project function point count, enhancement project function point count or application function point count.

2- Identify counting scope and application boundary: The counting scope defines the functionality that will be included in a particular function point counting. The application boundary indicates the border between the software being measured and the user.

3- Count data functions: Data functions represent The functionality provided to the user to meet internal and external data requirements. Data functions are either internal logical files (ILF) or external logical files (EIF). The complexity of them is assigned based on the number of data element types

0-7695-2237-8/04 $20.00 © 2004 IEEE

(DETs)[1] and record element types (RETs). A DET is a unique user recognizable, non-repeated field. A RET is a user recognizable subgroup of data elements within an ILF or EIF.

4- Count Transactional functions: Transactional functions represent the functionality provided to the user to process data. Transactional functions are either external inputs (EI), external outputs (EO) or external inquiries (EQ). The complexity of them is based on the number of DETs and file type referenced (FTR). A FTR is an ILF read or maintained by a transactional function or an EIF read by a transactional function. Each transactional function has specific rules for the identification of FTRs.

5- Determine unadjusted function point count: The counts for each function type are classified according to complexity and then weighed using table 1. The total of all function types is the unadjusted function point count.

Table 1. Unadjusted FP Calculation

Function Type	Functional Complexity	Complexity Totals	Function Type Totals
ILFs	__ Low	x7 = ___	
	__ Average	x10 = ___	
	__ High	x15 = ___	_____
EIFs	__ Low	x5 = ___	
	__ Average	x7 = ___	
	__ High	x10 = ___	_____
EIs	__ Low	x3 = ___	
	__ Average	x4 = ___	
	__ High	x6 = ___	_____
EOs	__ Low	x4 = ___	
	__ Average	x5 = ___	
	__ High	x7 = ___	_____
EQs	__ Low	x3 = ___	
	__ Average	x4 = ___	
	__ High	x6 = ___	_____
Total Unadjusted FP Count			_____

6- Determine value adjustment factor (VAF): The value adjustment factor indicates the general functionality provided to the user of the application. The VAF consists of 14 general system characteristics (GSC) that assess the general functionality of the application. Each characteristic has associated descriptions that help determine the degree of influence of the characteristic. The degrees of influence range from zero to five, from no influence to strong influence.

7- Calculate adjusted function point count: The final adjusted function point count is calculated using a specific formula for a development project, enhancement project or application function point count.

In the last decade the method had been criticized. The way the counting is elaborated [2], the independence of technology [4], [16], [23], as well as the adjustments used were reasons for many controversies [2], [12].

In 1999, the IFPUG presented the concept of the elementary process:

- The Elementary Process: is the smallest unit of activity that is meaningful to the user(s). For example, a user requires the ability to add a new employee to the application. The user definition of employee includes salary and information about any dependents. From the user perspective, the smallest unit of activity is to add a new employee. Adding one of the pieces of information, such as salary or dependent, is not the kind of activity that would qualify as an elementary process.

Considering this, the method is better understood and the research is related more to ways of simplifying or comparing it to other techniques created to estimate the software size in a specific environment [14], [19], [29].

The publication of the method as an ISO standard (ISO\IEC 20926:2003) is another fact that shows the recognition of it. However, the ISO standard considers the unadjusted function points. The use of CGS became optional from 2002 as a condition for the method to be considered as an ISO standard. The CGS is still one of the most criticized points of the method because there are different interpretations of it and some are out of date. Due to this, this research does not consider the CGS. The analysis is based on the unadjusted function point count.

2.2 NESMA

The NESMA [24] was founded in 1989 and is the largest function point analysis user group in Europe.

[1] rules of identification and counting of the DETs, RETs and FTRs are not presented in this paper. For this, [17] must be consulted

0-7695-2237-8/04 $20.00 © 2004 IEEE

The organization maintains its own manual, and the current version is 2.0. The objectives of NESMA are to collect, maintain, exchange and develop the knowledge about function points analysis, to promote a standardization of the method, and to promote the increase of its use.

The NESMA recognizes three types of function point counts: detailed, estimate and indicative. The detailed count is similar to the used by IFPUG. According to it, they are similar in up to 95% [20]. This count is performed as follows:

- determine all functions of all function types (ILF, EIF, EI, EO, EQ).
- rate the complexity of every data function (low, average, high)
- calculate the total unadjusted function point count

The other two methods, the estimate and the indicative one (referred to as "the Dutch method") have been developed by NESMA to enable a simplified function point count early in the system life cycle.

The difference between detailed and estimate counts is that the complexity is not determined for each individual function, but by default. After identification of all data and transactional functions, ILF and EIF complexity are assigned as low, and EI, EO, and EQ complexity are assigned as average.

The Indicative function point count is based solely on the number of data function (ILF and EIF). The number of unadjusted function points is calculated as follows:

$$Indicative = (35 * NroILFs) + (15 * NroEIFs)$$

This formula is based on the assumption that there will be about three EIs (to add, change, and delete information in the ILF), two EOs, and one EQ on average for every ILF, and about one EO and one EQ for every EIF.

The numbers 35 and 15 are obtained assigning the complexity of transactional functions as average and the complexity of data functions as low. Furthermore, some extra functionality of two function points to ILF and one function point to EIF for some general supporting functionality are assumed.

2.3 Mark II

In the decade of 1980, the Mark II or Mk II method was defined by Charles Symons, inspired by Albrecht's proposal. After its development within KPMG between 1985 and 1986, with the protected status of a proprietary method, it is now in the public domain. Today, UKSMA (United Kingdom Software Metrics Association) is responsible for its continuing development.

The Mk II method is applied in the countries of the United Kingdom and presents some differences in relation to the one of IFPUG. The Mk II considers all the requirements as logical transactions, calculated from the numbers of input data elements, output data elements and entities. Moreover, the MK II gives higher function point counts than the IFPUG method for larger systems [7].

2.4 COSMIC-FFP

COSMIC (Common Software Measurement International Consortium) was created in 1998 with the purpose of developing a new method of functional software size measurement. The group analyzed existing methods of software size estimate (IFPUG, Mark II, NESMA) to create a method based on their best characteristics.

This new method was published in November of 1999 as the release 2.0 of the Cosmic-FFP (COSMIC Full Function Points) [6]. The manual has been available (in English, French, Japanese and Spanish) for public access since then. The Cosmic-FFP measurement method is designed to be applicable to software from the following domains [8]:

- Business application software which is especially needed in support of business administration, such as banking, insurance and accounting.
- Real-time software, whose task is to keep up with or control events happening in real-time. The method has not yet been designed to support software which is characterized by complex mathematical algorithms or by processing continuous variables such as audio sounds or video images.

The research about Cosmic-FFP is related to its applicability in the size estimate of real time software. Diab et. al [18] report a formalization of the Cosmic-FFP measure for the Real-Time Object Oriented Modeling (ROOM). The benefits of its formalization are eliminate variation which may lead to different counts for the same specification, depending on the interpretation made by each evaluator, and allows the automation of COSMIC-FFP measurement for ROOM specifications.

0-7695-2237-8/04 $20.00 © 2004 IEEE

Moreover, the formal definition of the method can provide a clear and unambiguous characterization of it concepts, which is helpful for measuring COSMIC-FFP for other object-oriented notations like UML.

Raman [1] develops a model of effort estimate for the Cosmic-FFP through logic fuzzy, based on the fuzzy set model and on the fuzzy regression linear model. Finally, Bootsma [15] describes how full function points had enabled more completely estimates of real-time software.

3 Simplified Method

3.1 The Company's Characteristics

The software development company investigated has 25 employees and is considered a small-sized company in Brazil, according to the *Ministério da Ciência e Tecnologia* [11]. The company's main business is to work like an Internet provider and to develop web applications. The technologies used for application development are PHP, HTML, Java, and MySQL.

3.2 Case Study

The simplified method was created through a case study where twenty web applications developed by the company were analyzed. These applications have the same characteristics of the most developed software in the company. The case study is divided into four steps:

1) Counting the function points of web applications using the detailed method promoted by the IFPUG: The first step is to count function points for each application using the IFPUG detailed method. The count was based on the requirements analysis, its applications, and was supported by the project manager of the company. The main purpose of the project manager was to clarify some doubts related to the requirements and to show the user's view when in doubt. In table 2, the result of the count is shown.

2) Counting the function points of web applications using the estimated and indicative methods suggested by NESMA: Firstly, the function points were calculated using the indicative method, which formula is ((ILF Number)*35 + (EIF Number)*15). The result can be seen in Table 3.

   Secondly, the function points were calculated using the estimate counting. Regarding this, data functions complexity are assigned as low and transactional functions complexity are assigned as average. The result can be seen in Table 4:

Table 2. IFPUF FP count

Application 1 - 350 fp	Application 11 - 168 fp
Application 2 - 157 fp	Application 12 - 265 fp
Application 3 - 188 fp	Application 13 - 111 fp
Application 4 - 283 fp	Application 14 - 248 fp
Application 5 - 282 fp	Application 15 - 131 fp
Application 6 - 69 fp	Application 16 - 274 fp
Application 7 - 192 fp	Application 17 - 240 fp
Application 8 - 101 fp	Application 18 - 251 fp
Application 9 - 89 fp	Application 19 - 206 fp
Application 10 - 238 fp	Application 20 - 163 fp

Table 3. Indicative FP count

Application 1 - 470 fp	Application 11 - 280 fp
Application 2 - 245 fp	Application 12 - 385 fp
Application 3 - 280 fp	Application 13 - 175 fp
Application 4 - 385 fp	Application 14 - 330 fp
Application 5 - 435 fp	Application 15 - 210 fp
Application 6 - 105 fp	Application 16 - 385 fp
Application 7 - 245 fp	Application 17 - 350 fp
Application 8 - 175 fp	Application 18 - 330 fp
Application 9 - 140 fp	Application 19 - 245 fp
Application 10 - 365 fp	Application 20 - 260 fp

Table 4. Estimate FP count

Application 1 - 396 fp	Application 11 - 202 fp
Application 2 - 188 fp	Application 12 - 316 fp
Application 3 - 224 fp	Application 13 - 130 fp
Application 4 - 350 fp	Application 14 - 277 fp
Application 5 - 313 fp	Application 15 - 160 fp
Application 6 - 84 fp	Application 16 - 330 fp
Application 7 - 228 fp	Application 17 - 289 fp
Application 8 - 116 fp	Application 18 - 283 fp
Application 9 - 103 fp	Application 19 - 229 fp
Application 10 - 281 fp	Application 20 - 199 fp

3) Comparison of steps 1 and 2: The results found in the counts using indicative and estimate methods were compared with the numbers of the IFPUG detailed method. The comparison showed that, in this case, the NESMA methods are not applicable to determine the function points of web applications correctly. However, these counts indicated

that most functions had low complexity. This fact motivated the creation of the simplified method.

4) Simplified method creation: The simplified method was based on maintaining all the functions with low complexity. The count result can be seen in table 5:

Table 5. Simplified Method

Application 1 - 324 fps	Application 11 - 166 fps
Application 2 - 154 fps	Application 12 - 257 fps
Application 3 - 183 fps	Application 13 - 107 fps
Application 4 - 283 fps	Application 14 - 227 fps
Application 5 - 258 fps	Application 15 - 131 fps
Application 6 - 69 fps	Application 16 - 267 fps
Application 7 - 185 fps	Application 17 - 235 fps
Application 8 - 97 fps	Application 18 - 232 fps
Application 9 - 85 fps	Application 19 - 186 fps
Application 10 - 230 fps	Application 20 - 163 fps

After performing the counts for the four methods, a comparison among the methods is shown in table 6. The function points determined using the IFPUG detailed method can be seen in the second column. The function points determined by the other methods and the errors found (in percentage), when compared with the IFPUG detailed method, are shown in columns 3, 4 and 5.

Regarding the table 7, it is possible to observe that the indicative method showed bad results, an error varying between 19% (application 19) and 73% (application 08). The counting with this method showed, on average, an error of 48%. The estimate count presented errors between 11% (applications 5 and 19) and 26% (application 4). The counting using the estimate method showed, on average, an error of 18%. In the indicative and estimate counting, the function points found were higher than the ones found in the IFPUG detailed method.

The simplified method, whose function complexities were assigned as low, showed satisfactory results. The error was between 1% (application 11) and 10% (application 19), four applications presented an identical number of function points to the IFPUG detailed method and the average error was 4%. Moreover, considering five applications that presented an error higher than 5%, four are similar software (e-commerce applications) and the other application was developed with another company (only some modules were developed by the studied company). This is an unusual fact in the development process of the company.

Table 6. Methods' Comparison

IFPUG	NESMA Indicative	NESMA Estimate	Simplified Method
350	34% (470)	13%(396)	-7%(324)
157	56% (245)	20% (188)	-2% (154)
188	49% (280)	19% (224)	-3% (183)
283	39% (385)	26% (350)	0 (277)
282	54% (435)	11% (313)	-9% (258)
69	52% (105)	22% (84)	0 (69)
192	28% (245)	19% (228)	-4% (185)
101	73% (175)	15% (116)	-4% (97)
89	57% (140)	16% (103)	-4% (85)
238	53% (365)	18% (281)	-3% (230)
168	67% (280)	20% (202)	-1% (166)
265	45% (385)	19% (316)	-3% (257)
111	58% (175)	17% (130)	-4% (107)
248	33% (330)	12% (277)	-8% (227)
131	60% (210)	22% (160)	0 (131)
274	41% (385)	20% (330)	-3% (267)
240	46% (350)	20% (289)	-2% (235)
251	31% (330)	13% (283)	-8% (232)
206	19% (245)	11% (229)	-10% (186)
163	60% (260)	22% (199)	0 (163)

The steps described above are shown in figure 2.

4 Tool Support

The function point analysis method allows measuring the software size. The amount of time and effort needed to develop the software are obtained from the software size estimated.

It motivated the modeling of a tool not only to be able to support the counting of function points but also to determine the employees' productivity and the time necessary (in hours) to develop each one of the applications. The tool model is shown in figure 3.

The tool consists of two modules. The first one is a client-server application responsible for obtaining and the storage information about the activities of software development. The module has the following functionalities:

- It presents reports to the project manager showing the number of hours needed to finish each activity and behavior of the project.
- It presents the single employee and team productivity for each project.

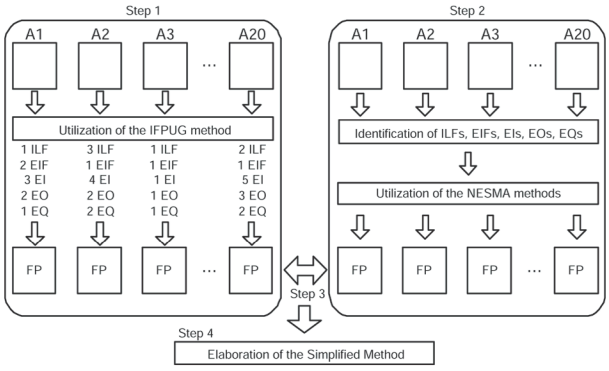

Figure 2. Creation of simplified method

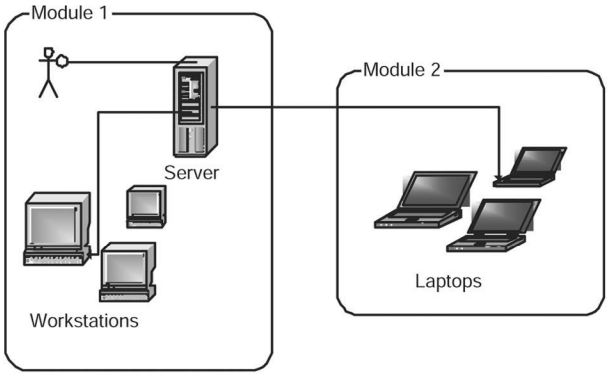

Figure 3. Tool model

- It sends data to the second module.

The second module is an application in a mobile device. Its function is to determine the function points through the simplified method. There is an information exchange between the two modules allowing the second module to receive productivity information from the first one. Once the function points are determined, it is possible to determine the amount of time needed to develop the application satisfactorily.

5 Conclusion and Future Work

The simplified method presented results closely to those found with the IPFUG detailed method. The simplified method is based on assigned low complexity to all data and transactional functions. Thus, when data and transactional functions are identified, their complexity is determined automatically.

However, it is important to remember that the results found are valid for the problem domain, programming language and systems used by the studied company. Generalizing this method for the application in other companies is not the main intention of this research.

Therefore, for future work, there is an intention of verify if the method can be used by other companies which develop web applications. Another aspect is to analyze how to create specific GSC for the company and calibrate the adjustment factor.

References

[1] Raman A. and Noore A. Software metrics for real-time systems using fuzzy sets. *System Theory. Proceedings of the 35th Southeastern Symposium on*, pages 74 – 78, March 2003.

[2] Kitchenham B. The problem with function points. *IEEE Software*, pages 29 – 31, 1997.

[3] Briand L. C. and Wieczorek I. Software resource estimation. *Encyclopedia of Software Engineering*, vol. 2(P-Z):pag. 1160 – 1196, 2002.

[4] Caldiera G. & Antoniol G. & Fiutem R. & Lokan C. Definition and experimental evaluation of functions points for object-oriented systems. *Proc. of the 5th International Symposium on Software Metrics*, pages 167 – 178, November 1998.

[5] Jones T. C. *Programming Productivity*. McGraw-Hill, 1986.

[6] Jones T. C. *Estimating Software Costs*. McGraw-Hill, 1998.

[7] Symons C. Conversion between ifpug 4.0 and mkii function points. *Software Measurement Services*, 1999.

[8] Cosmic-FFP. *Measurement Manual*. Cosmic, 2.1 edition, May 2001.

[9] MCT Ministério da Ciência e Tecnologia. Qualidade e produtividade no setor de software. *http://www.mct.gov.br/Temas/info/Dsi/Quali2001/Public2001.htm*, Tabela 40 - Práticas de Engenharia de Software Adotadas no Desenvolvimento e Manutenção de Software, 2001.

[10] MCT Ministério da Ciência e Tecnologia. Qualidade e produtividade no setor de software. *http://www.mct.gov.br/Temas/info/Dsi/Quali2001/Public2001.htm*, Tabela 01 - Atividades das Organizações no Tratamento de Software, 2001.

[11] MCT Ministério da Ciência e Tecnologia. Qualidade e produtividade no setor de software. *http://www.mct.gov.br/Temas/info/Dsi/Quali2001/Public2001.htm*, Tabela 06 - Porte das Organizações, Segundo a Força de Trabalho Total e Efetiva, 2001.

[12] Fenton N. E. and Pfleeger S. L. *Software Metrics: A Rigorous and Pratical Approach*. PWS, 2 edition, 1997.

[13] Hastings T. E. and Sajeev A. S. M. A vector-based approach to software size measurement and effort estimation. *IEEE Trans. Software Eng.*, vol. 27(nro. 4):pag. 337 – 350, April 2001.

[14] Mendes E., Mosley N., and Counsell S. Early web size measures and effort prediction for web costimation. *Software Metrics Symposium, 2003. Proceedings. Ninth International*, pages 18 – 39, September 2003.

[15] Bootsma F. How to obtain accurate estimates in a real-time environment using full function points. *Proceedings. 3rd IEEE Symposium on Application-Specific Systems and Software Engineering Technology*, pages 105 – 112, March 2000.

[16] Teologlou G. Measuring object oriented software with predictive object points. *10th Conference on European Software Control and Metrics*, May 1999.

[17] International Function Point Users Group. *Function Point Counting Practices Manual*. IFPUG, 4.1.1 edition, January 1999.

[18] Diab H., Frappier M., and Denis R. Formalizing cosmic-ffp using room. *ACS/IEEE International Conference on Computer Systems and Applications*, pages 312 – 318, June 2001.

[19] Tavares H., Carvalho A., and Castro J. Medição de pontos de função a partir da especificação de requisitos. *WER02 - Workshop em Engenharia de Requisitos*, pages 278 – 298, November 2002.

[20] IFPUG. International function point users group. *http://www.ifpug.org*, last access on April 2004.

[21] Albrecht A. J. Measuring application development productivity. *Proc. IBM Applications Development Symposium*, pages pag. 83 – 92, 1979.

[22] Dolado J. J. A validation of the component-based method for software size estimation. *IEEE Trans. Software Eng.*, vol. 26(nro. 10):pag. 1006 – 1021, October 2000.

[23] Schooneveldt M. Measuring the size of object oriented systems. *Proc. of the 2nd Australian Conference on Software Metrics*, 1995.

[24] NESMA. Netherlands software metrics association. *http://www.nesma.org/english/index.htm*, last access on April 2004.

[25] Agarval R., Kumar M., Mallick S., Bharadwaj R., and Anantwar D. Estimating software projects. *Software Engineering Notes*, vol. 26(nro. 4):pag. 60 – 67, July 2001.

[26] Lai R. and Huang S. A model for estimating the size of a formal communication protocol specification and its implementation. *IEEE Trans. Software Eng.*, vol. 29(nro. 1):pag. 46 – 62, January 2003.

[27] Pressman R. *Engenharia de Software*. McGraw-Hill, 5 edition, 2001.

[28] Humphrey W. S. *Managing the Software Process*. Addison-Wesley, 1989.

[29] Kusumoto S., Imagawa M., Inoue K., Morimoto S., Matsusita K., and Tsuda M. Function point measurement from java programs. *ICSE 2002. Proceedings of the 24rd International Conference on*, pages 576 – 582, May 2002.