How to Use COSMIC Functional Size in Effort Estimation Models?

Cigdem Gencel

Blekinge Institute of Technology - Department of Systems and Software Engineering, Ronneby, Sweden
Cigdem.Gencel@bth.se

Abstract. Although Functional Size Measurement (FSM) methods have become widely used by the software organizations, the functional size based effort estimation still needs further investigation. Most of the studies on effort estimation consider total functional size of the software as the primary input to estimation models and they mostly focus on identifying the project parameters which might have a significant effect on the size-effort relationship. This study brings suggestions on how to use COSMIC functional size as an input for effort estimation models and explores whether the productivity values for developing different functionality types deviate significantly from a total average productivity value computed from total functional size and effort figures. The results obtained after conducting a multiple case study in which COSMIC method was used for size measurement are discussed as well.

Keywords: Functional Size Measurement, Effort Estimation, Functionality, COSMIC, Base Functional Component.

1 Introduction

Since the first introduction of Function Point Analysis (FPA) method by Albrecht in 1979 [5], Functional Size Measurement (FSM) methods have not only been improved, but also new variations and extensions have been developed to be able to measure the new types of applications [18][43].

Among these methods, the ones which conform to ISO/IEC 14143-1 standard [22][23] are accepted as international standards for FSM. Common Software Measurement International Consortium Full Function Points (COSMIC-FFP) [26][14], International Function Point Users Group (IFPUG) FPA [27][20], MarkII FPA [28][45], Netherlands Software Metrics Association (NESMA) FSM [29][41] and Finnish Software Metrics Association (FiSMA) FSM methods are the ones accepted as FSM standards up to now [30].

Besides the usage of software size for a number of reasons such as in project tracking or for normalization of other measures, one of the major uses of software size is that it is the primary input for most effort and cost estimation models.

However, effort estimation based on functional size still remains a challenge for software practitioners and researchers. As more empirical data are collected in benchmarking

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 196–207, 2008.

datasets, the studies to explore the nature of the relationship between functional size and effort has been arisen. Taking the functional size as the main input, most of the studies on effort estimation investigate some project related attributes which might have impact on functional size and effort relationship. Unfortunately, the common conclusion of the existing studies is that although different models are successfully used by different groups and for particular domains, they do not have unanimous acceptance by the software community they being not performing well enough.

Traditionally, the functional size of a software system is measured as a single total value obtained by a specific FSM method. All FSM methods have their own attribute definition models [37], which derive this single size value by expressing a relationship among the sub-attributes, called Base Functional Component (BFC) Types.

In [37], Kitchenham et al claimed that Function Points (FP) might be viewed as a means of measuring the "shape" of a software product in terms of a vector of significant elements. They added that if the elements of such a shape measure influence the product development effort, an attempt could be made to derive an effort estimation model based on these elements.

In other engineering disciplines, there are different representations of the size for the same product. For example, in civil engineering, different size measures are defined to size buildings [13][12]. A vector of measures such as the floor area -which is calculated by multiplying the length and width of the floor- and the height of the building, is one representation. Or, a derived measure such as the volume of a building which is calculated by the multiplication of length, width and height of the building is another. The selection depends on the needs of the engineers or managers. For example, if the volume measure is sufficient for effort and cost estimation purposes in a specific project, this measure is used. Similarly, we also need different representations of software size for different purposes in software engineering.

In our previous studies [17][11], we investigated whether effort estimation models based on Base Functional Component (BFC) types[1], rather than those based on a single total functional size value would improve estimation reliability. The assumptions of these studies were that the amount of effort to be utilized for developing a unit size of different BFC types would be different. For the empirical study, we used the projects data in the International Software Benchmarking Standards Group (ISBSG) dataset [21] and formed sub-sets based on different criteria. We made the statistical analysis on the projects which were measured by COSMIC method. We made multiple regression analysis for investigating the strength of the relationship between the functional sizes of BFC Types and development effort. The results of both studies showed significant improvement in the size-effort relationship.

In this study, we propose a new representation of COSMIC functional size to be used in effort estimation models. Instead of a single size figure, we define a vector of measures. We identify the elements (or sub-attributes) of functional size which provide different functionalities to the users considering the BFC Types and the effort collection mechanisms in software organizations. We also present the results of a multiple case study which we conducted to explore whether the productivity values

[1] BFC: "an elementary unit of FUR defined by and used by an FSM Method for measurement purposes". BFC Type: "a defined category of BFCs. A BFC is classified as one and only one BFC Type" [23].

for developing each element deviates significantly from a total average productivity value computed from total functional size and effort figures.

2 Background

2.1 Functional Size Measurement

Albrecht's 1979 proposal [5] for estimating the functional size became a significant contender for software size measurement and hence effort estimation. This method was aimed at overcoming some of the shortcomings of measures based on Source Lines of Code (SLOC) for estimation purposes and productivity analysis, such as their availability only fairly late in the development process and their technology dependence. The FPA method is based on the idea of measuring the amount of functionality delivered to users in terms of Function Points (FP) taking into account only those elements in the application layer that are logically 'visible' to the user and not the technology used. FPA was designed in a Management Information System (MIS) environment and has become a *de facto* standard in the MIS community.

During the following years, variations of the original method have been developed[2]. Some of them either provided unique viewpoints different from the dominant method of their time or extended the applicability of FSM methods to different functional domains in addition to business application software such as Real-time systems, Web applications, etc. Other methods designed to measure software which are developed using object oriented methodology.

In the '90s, work was initiated at the ISO level to lay the foundations for regulating standards in FSM, and the 14143 family [24][25][31][32][33] was developed with five instantiations matching with those requirements: COSMIC-FFP [26][14], IFPUG FPA [20][27], MkII FPA [45][28], NESMA FSM [41][29] and FiSMA FSM [30] methods.

Albrecht's original idea has become the basis for IFPUG FPA [3], one of the earliest ISO standardized FSM methods [20][27]. IFPUG FPA enjoys widespread popularity and large publicly available data sets for those who wish to train their own company-specific IFPUG model or to compare their measurements with others.

MkII FPA [44] was developed by Symons in 1988 in order to improve the original FPA method. This method brought some suggestions to reflect the internal complexity of a system. Currently, the Metrics Practices Committee (MPC) of the UK Software Metrics Association (UKSMA) is the design authority of the method [45]. It was also mainly designed to measure business information systems. Mk II FPA has been accepted as being conformant to ISO/IEC 14143 and become an international ISO standard in 2002 [28].

NESMA FPA [41] has the same rules as the IFPUG FPA method. The differences between these two methods is due to NESMA measurement manual provides different guidelines, hints and examples. It was accepted by ISO as an international standard in 2005 [29].

COSMIC-FFP [14], adopted in 2003 as ISO 19761 [26], has been defined as a 2nd generation FSM method as a result of a series of innovations, such as: a better fit with

[2] Please refer to [18] and [43] for a detailed discussion on and a history of FPA-like methods.

both real-time and MIS environments, identification and measurement of multiple software layers, different viewpoints from which the software can be observed and measured, and the absence of a weighting system.

Finally, FiSMA FSM, accepted as an international FSM standard in 2008 [30], was developed by a working group of FiSMA. It is a general parameterized size measurement method that is designed to be applied to all types of software. The difference of FiSMA FSM from other methods is that it service-oriented rather than process-oriented.

2.2 Software Size Based Effort Estimation

In parallel to these developments in FSM, significant research has been going on software size based effort estimation such as in [7][8][19][34][35][36]. In [6][16][40][42], significant variations in the impact of project cost drivers have been observed. Among the cost drivers investigated, Team Size, Programming Language Type, Organization Type, Business Area Type, Application Type, Development Type and Development Platform have been found to affect the size-effort relationship at different levels of significance.

In a number of studies such as [2][9][10][39], the related works on estimation models are assessed and compared. However, the common conclusion of these studies was that although different models are successfully used by different groups and for particular domains, none of them has gained general acceptance by the software community.

Other studies such as [1][3][17][11] focused merely on functional size and explored different ways to use it as an input to effort estimation models. In [1][3], the concept of software functional profile is defined as the relative distribution of its four BFC Types for any particular project. They investigated whether or not the size-effort relationship was stronger if a project was close to the average functional profile of the sample studied. It was observed that the identification of the functional profile of a project and its comparison with the profiles of their own samples can help in selecting the best estimation models relevant to its own functional profile.

In [17][11], it was explored whether effort estimation models based on the BFC types rather than those based on a single total value would improve estimation models. Both of these studies showed significant improvement in modeling the size-effort relationship.

The results of the literature survey show that functional size based effort estimation still require further investigation. This paper focuses on investigating how to use functional size in effort estimation models. A new representation of COSMIC functional size, which can be used for effort estimation purposes, will be defined.

3 Suggestions for a New Representation of COSMIC Functional Size

In COSMIC measurement process [15], the Functional User Requirements (FURs) are decomposed into their elementary components, called “Functional Processes”. A Functional Process is defined as “an elementary component of a set of FUR comprising a unique, cohesive and independently executable set of data movements”.

A Data Movement Type is defined as "a BFC which moves one or more data attribute types belonging to a single data group type". There are four kinds of Data Movement Types: Entry, Exit, Read, and Write. Each of these is defined as a BFC Type in [15] as;

- An *Entry* is a data movement type that moves a data group from a functional user across the boundary into the functional process where it is required.
- An *Exit* is a data movement type that moves a data group from a functional process across the boundary to the functional user that requires it.
- A *Read* is a data movement type that moves a data group from persistent storage within reach of the functional process which requires it.
- A *Write* is a data movement type that moves a data group lying inside a functional process to persistent storage.

After identifying the BFC Types in the Functional Processes, the second step involves calculating the functional size of each BFC by applying a measurement function to the BFC Types and the related attributes. Then the results are aggregated to compute the overall size of the software system.

Each of these BFC Types represents different types of functionalities to be provided to the users. That is the reason why FSM methods identify these elements to measure functional size. In our previous studies [17][11], we investigated whether effort estimation models based on the BFC types rather than those based on a single total value would improve estimation models. Both of these studies showed significant improvement in modeling the size-effort relationship. This approach has the potential to result in generic effort estimation models when significant amount of projects data can be collected in benchmarking datasets. However, until then, software organizations might collect their own data so that average productivity figures to develop each type of functionality can be found at least for that specific organization.

Since we are after finding a representation of functional size for effort estimation purposes, we also have to consider the effort collection mechanisms in software organizations in addition to the BFC Types which serve different functionalities. Based on these, we identify the significant elements for which we define a vector of measures.

In software development projects, if the components of a software product are developed by using different technologies, implemented on different processors or developed by utilizing different programming languages, the efforts utilized for each are can usually be identified. The components involve development of different types of functionalities. At a first glance, we identify two possible functionality types related to interface and data services components.

The Entry and Exit data movement types are the functionalities provided to the functional user for moving data groups across the boundary. We call them Interface functionalities.

The Read and Write data movement types are the functionalities provided to the functional user for moving data groups from persistent storage within reach of the functional process and to the persistent storage lying inside a functional process. We call them Data Services functionalities.

In the first version of COSMIC method, data type characteristics in business applications and in real-time systems were investigated [38][4]. The differences between the single-occurrence control data in real-time systems and the multiple-occurrence

group of data in business application software are discussed. Although the units of measure in COSMIC [15] are the same for measuring these different functionalities and the functional sizes of different BFC Types can be added to compute the total size, the amount of effort to be utilized per unit size might be different. Therefore, we also differentiate Business-Application Data Services from Control Data Services.

Thus, the new representation of COSMIC Functional size for effort estimation purposes involves a vector of measures for the following elements: Interface, Business-Application Data Services and Control Data Services. This new representation does not interfere with any of the principles of COSMIC. This is analogous to using the same unit of measure and measurement rules for measuring the length, width, and height of a building. The difference is that we do not use a compound measure as the volume measure, but a vector of measures for representing COSMIC functional size.

4 Case Study

We conducted a multiple-case study in order to evaluate the proposed representation for COSMIC functional size. Our research question for this case study was the following: "Are the productivity figures for developing each element, i.e. each functionality type, deviates significantly from the total average productivity figure for developing the whole software?"

We designed this case study as a multiple-case study which involves three new development case projects. We applied COSMIC to measure the functional size of the case projects. The case projects and the case study are described and discussed in the following sub-sections.

4.1 Description of the Case Projects and Organizations

Project-1 involves the development of a military inventory management system integrated with a document management system. The software development organization is an independent supplier, which is a CMMI/SW Maturity Level 3 company. The organization focuses mostly on web-based projects and has its own framework to develop web applications rapidly. The project was started in October 2004 and completed in August 2005. 7 persons worked for the project: 1 project manager, 1 senior software engineer, 2 software engineers (development team – full-time), 2 software engineers (development team - part-time), and 1 software engineer (test team – part-time).

Project-2 involves the development of a multimedia sponsored call system. The system enables advertising companies to relay their messages to their target market through call sponsorships and enables end-users to have sponsorships for their calls by receiving an interactive multimedia advertisement at the beginning or during the call. The software company develops telecommunications solutions and provides network infrastructure components for the telecommunications industry, having totally 80 personnel 50 of which are engineers. The company owns TSE-EN-ISO 9001:2000 quality certificate. 9 persons worked for the project: 1 project manager, 1 senior software engineer (development team leader), 6 software engineers (development team) and 1 software test engineer (test team leader).

Project-3 involves the development of an equipment identification registrar which detects and warns the operator against potential fraud risks such as Subscriber Identity

Module (SIM) card cloning and International Mobile Equipment Identity (IMEI) cloning. The same software company which developed Project-4 developed this project as well. 10 persons worked for the project: 1 project manager, 2 senior software engineer (development team leader), 6 software engineers (development team) and 1 software test engineer (test team leader).

The total efforts utilized for the software life cycle processes of the case projects are 6,308, 1,080 and 1,200 person-hours for Project-1, Project-2 and Project-3, respectively.

We used the CHAR Method [33] to determine Functional Domains of the case projects. The functional domains of Project-1, Project-2 and Project-3 are 'Information System', 'Complex Controlling Information System' and 'Information System, respectively.

Table 1. Functional Domains of the Case Projects determined by CHAR Method [33]

Project No	Functional Domain	Control- and communication-rich FUR	Data-rich FURs	Manipulation- and algorithm-rich FURs
1	Information System	Negligible	dominant	present
2	Complex Controlling Information System	Present	dominant	present
3	Information System	Negligible	dominant	present

4.2 Case Study Conduct and Data Collection

All the projects were measured by COSMIC FFP v.2.2 [15] utilizing the Software Requirements Specification (SRS) documents prepared according to the companies' SRS standards.

Two measurers together measured the functional size of Project-1 (see Table 2). One of the measurers works for the development organization and is involved in this project. The other is the author of this paper. Both are experienced in using COSMIC. The effort utilized to make measurement is 13 person-hours.

One measurer performed the functional size measurement for the other two projects (see Table 2). She is the author of this paper. The efforts utilized to make measurement are 15 person-hours for each of the other two projects.

The functional sizes of Project-1, Project-2 and Project-3 were measured as 1020, 321 and 275 COSMIC Function Points (CFP), respectively.

The research question in this case study was to explore whether the elements of functional size influence the product development effort, i.e. whether the Productivity Delivery Rates (PDR) to develop different elements might be different.

Table 2. Case Projects COSMIC FFP size measurement details

Project No	Number of Functional Processes	Number of Entries	Number of Exits	Number of Reads	Number of Writes	Functional Size (CFP)
Project-1	127	154	378	333	155	1,020
Project-2	50	80	79	99	63	321
Project-3	54	69	115	45	46	275

The PDR values for the projects, which are calculated as the ratio of effort to total COSMIC functional size, are given in Table 3. Since all of these projects involve algorithmic operations which cannot be measured by COSMIC method, we excluded these efforts from the total effort values.

For the case projects, the effort values were collected in detail so that the amount of effort utilized for each functionality type could also be identified. For Project-1, the software coding and unit testing efforts were collected based on the three types of functionalities. Therefore, for this project we calculated the PDR value based on Code and Unit Test Effort values. In this project, the Interface and the Permanent Storage/access functionalities were developed by using the Internal Development Framework (IDF) which was developed by the development organization. IDF is a tool to reuse CRUDL processes in standard web applications. By this tool, the interface and database components are generated in parallel with each other. For the processing component, Java is used as the primary programming language. These components were developed not only by different teams but also using different technologies.

For Project-2 and Project-3, the effort data are available for the whole development. Therefore, we calculated the PDR values for developing these different functionality types. In Project-2, Java was the primary programming language and in Project-3, Java and ANSI were the primary programming languages.

Table 3. PDR Values of the Projects for the Elements of COSMIC Functional Size

		Total Figures	The Elements of COSMIC Functional Size		
			Business-Application Data Services	Control Data Services	Interface
Project-1	Functional Size (CFP)	1,020	488	0	532
	Code &Unit Test Effort (person-hours)	1.333	742	-	591
	PDR (person-hrs/CFP)	*1.31*	*1.52*	-	*1.11*
Project-2	Functional Size (CFP)	321	146	16	159
	Development Effort (person-hours)	1,010	540	200	270
	PDR (person-hrs/CFP)	*3.15*	*3.70*	*12.50*	*1.70*
Project-3	Functional Size (CFP)	275	91	0	184
	Development Effort (person-hours)	1,130	450	-	680
	PDR (person-hrs/CFP)	*4.11*	*4.95*	-	*3.70*

4.3 Discussion of the Case Study Results

Since Project-2 and Project-3 were developed by the same organization, we investigated the deviation between PDR values of these two projects. The PDR of Project-3 deviates 30.5% from Project-2. The PDR values for developing each of the key size parameters are shown in Table 4.

Table 4. % Deviation in PDR for the elements from the Average PDR

Projects	***% Deviation in PDR for the COSMIC elements from the Average PDR***		
	Business-Application Data Services	Control Data Services	Interface
Prj-1	16.03	-	-15.26
Prj-2	17.46	296,82	-46.03
Prj-3	20.44	-	-9,97

The results show that, for all the projects the amount of effort required to develop business-application data services per unit size is greater than the average figures whereas it is less for developing the interface functionalities. For Project-2, PDR value for developing Control data services is so high. Therefore, its deviation from the average figure is also very high.

5 Conclusion

This study aimed to investigate whether a different representation of COSMIC functional size without changing any rules and principles of the method, would have the potential to improve effort estimation reliability.

Different functionality types are identified by grouping COSMIC BFC Types considering the effort collection mechanisms in the software organizations. Accordingly, Interface, Business Application Data Services and Control Business Data Services are identified as being different functionality types.

The case study results showed that there is a significant variation between the PDR values for developing different kinds of functionalities. Therefore, building estimation models using this new representation for COSMIC functional size rather than using a single total value is promising.

Moreover, by this representation of size, we get more information about the functional domain of the software and the types of functionalities to be provided to the users.

Acknowledgements

I would like to thank Pinar Efe and Figan Bilgin for their contributions in performing the case studies.

References

[1] Abran, A., Gil, B., Lefebvre, E.: Estimation Models Based on Functional Profiles. In: International Workshop on Software Measurement – IWSM/MetriKon, Kronisburg, Germany, pp. 195–211. Shaker Verlag (2004)

[2] Abran, A., Ndiaye, I., Bourque, P.: Contribution of Software Size in Effort Estimation, Research Lab. In: Software Engineering, École de Technologie Supérieure, Canada (2003)

[3] Abran, A., Panteliuc, A.: Estimation Models Based on Functional Profiles. III Taller Internacional de Calidad en Technologias de Information et de Communications, Cuba, (February 15-16, 2007)

[4] Abran, A., St-Pierre, D., Maya, M., Desharnais, J.M.: Full Function Points for Embedded and Real-Time Software. In: UKSMA Fall Conference, London, UK, October, pp. 30–31 (1998)

[5] Albrecht, A.J.: Measuring Application Development Productivity. In: Proc. Joint SHARE/GUIDE/IBM Application Development Symposium, pp. 83–92 (1979)

[6] Angelis, L., Stamelos, I., Morisio, M.: Building a Cost Estimation Model Based on Categorical Data. In: 7th IEEE Int. Software Metrics Symposium (METRICS 2001), London (April 2001)

[7] Boehm, B.W.: Software Engineering Economics, p. 487. Prentice-Hall, Englewood Cliffs (1981)

[8] Boehm, B.W., Horowitz, E., Madachy, R., Reifer, D., Bradford, K.C., Steece, B., Brown, A.W., Chulani, S., Abts, C.: Software Cost Estimation with COCOMO II. Prentice Hall, New Jersey (2000)

[9] Briand, L.C., El Emam, K., Maxwell, K., Surmann, D., Wieczorek, I.: An Assessment and Comparison of Common Software Cost Estimation Models. In: Proc. of the 21st Intern. Conference on Software Engineering, ICSE 1999, Los Angeles, USA, pp. 313–322 (1999)

[10] Briand, L.C., Langley, T., Wieczorek, I.: A Replicated Assessment and Comparison of Software Cost Modeling Techniques. In: Proc. of the 22nd Intern. Conf. on Software Engineering, ICSE 2000, Limerick, Ireland, pp. 377–386 (2000)

[11] Buglione, L., Gencel, C.: Impact of Base Functional Component Types on Software Functional Size based Effort Estimation. In: Jedlitschka, A., Salo, O. (eds.) PROFES 2008. LNCS, vol. 5089, pp. 75–89. Springer, Heidelberg (2008)

[12] CESMM - Civil Engineering Standard Method of Measurement, 3rd edn. Thomas Telford Ltd. (1991)

[13] Chen, W.F., Liew, J.Y.R.: The Civil Engineering Handbook, 2nd edn. CRC Press, Boca Raton (2003)

[14] COSMIC. COSMIC- v.3.0, Measurement Manual (September 2007)

[15] COSMIC: The Common Software Measurement International Consortium FFP, version 3.0, Measurement Manual (2007)

[16] Forselius, P.: Benchmarking Software-Development Productivity. IEEE Software 17(1), 80–88 (2000)

[17] Gencel, C., Buglione, L.: Do Different Functionality Types Affect the Relationship between Software Functional Size and Effort? In: Cuadrado-Gallego, J.J., et al. (eds.) IWSM-Mensura 2007. LNCS, vol. 4895, pp. 72–85. Springer, Heidelberg (2008)

[18] Gencel, C., Demirors, O.: Functional Size Measurement Revisited. ACM Transactions on Software Engineering and Methodology (TOSEM) 17(3), 71–106 (2008)

[19] Hastings, T.E., Sajeev, A.S.M.: A Vector-Based Approach to Software Size Measurement and Effort Estimation. IEEE Transactions on Software Engineering 27(4), 337–350 (2001)
[20] IFPUG. Function Points Counting Practices Manual (release 4.2), International Function Point Users Group, Westerville, Ohio (January 2004)
[21] ISBSG Dataset 10 (2007), http://www.isbsg.org
[22] ISO/IEC 14143-1: Information Technology – Software Measurement – Functional Size Measurement – Part 1: Definition of Concepts (1998)
[23] ISO/IEC 14143-1: Information Technology – Software Measurement – Functional Size Measurement – Part 1: Definition of Concepts (February 2007)
[24] ISO/IEC 14143-2: Information Technology – Software Measurement – Functional Size Measurement - Part 2: Conformity Evaluation of Software Size Measurement Methods to ISO/IEC 14143-1:1998 (2002)
[25] ISO/IEC 14143-6: Guide for Use of ISO/IEC 14143 and related International Standards (2006)
[26] ISO/IEC 19761:2003, Software Engineering – COSMIC-FFP: A Functional Size Measurement Method, International Organization for Standardization (2003)
[27] ISO/IEC 20926:2003, Software Engineering-IFPUG 4.1 Unadjusted Functional Size Measurement Method - Counting Practices Manual, International Organization for Standardization (2003)
[28] ISO/IEC 20968:2002, Software Engineering – MK II Function Point Analysis – Counting Practices Manual, International Organization for Standardization (2002)
[29] ISO/IEC 24570:2005, Software Engineering – NESMA functional size measurement method version 2.1 – Definitions and counting guidelines for the application of Function Point Analysis, International Organization for Standardization (2005)
[30] ISO/IEC 29881:2008, Software Engineering – FiSMA functional size measurement method version 1.1, International Organization for Standardization (2008)
[31] ISO/IEC TR 14143-3: Information Technology – Software Measurement – Functional Size Measurement – Part 3: Verification of Functional Size Measurement Methods (2003)
[32] ISO/IEC TR 14143-4: Information Technology – Software Measurement – Functional Size Measurement - Part 4: Reference Model (2002)
[33] ISO/IEC TR 14143-5: Information Technology – Software Measurement – Functional Size Measurement – Part 5: Determination of Functional Domains for Use with Functional Size Measurement (2004)
[34] Jeffery, R., Ruhe, M., Wieczorek, I.A.: Comparative Study of Two Software Development Cost Modeling Techniques using Multi-organizational and Company-specific Data. Information and Software Technology 42, 1009–1016 (2000)
[35] Jorgensen, M., Molokken-Ostvold, K.: Reasons for Software Effort Estimation Error: Impact of Respondent Role, Information Collection Approach, and Data Analysis Method. IEEE Transactions on Software Engineering 30(12), 993–1007 (2004)
[36] Kitchenham, B., Mendes, E.: Software Productivity Measurement Using Multiple Size Measures. IEEE Transactions on Software Engineering 30(12), 1023–1035 (2004)
[37] Kitchenham, B., Pfleeger, S.L., Fenton, N.: Toward a Framework for Software Measurement Validation. IEEE Transactions on Software Engineering 21(12) (December 1995)
[38] Maya, M., Abran, A., Oligny, S., St-Pierre, D., Desharnais, J.M.: Measuring the Functional Size of Real-Time Software. In: Proc. of 1998 European Software Control and Metrics Conference, Maastricht, The Netherlands, pp. 191–199 (1998)
[39] Menzies, T., Chen, Z., Hihn, J., Lum, K.: Selecting Best Practices for Effort Estimation. IEEE Transactions on Software Engineering 32(11), 883–895 (2006)

[40] Morasca, S., Russo, G.: An Empirical Study of Software Productivity. In: Proc. of the 25th Intern. Computer Software and Applications Conf. on Invigorating Software Development, pp. 317–322 (2001)
[41] NESMA. Definitions and Counting Guidelines for the Application of Function Point Analysis, v.2.0 (1997)
[42] Premraj, R., Shepperd, M.J., Kitchenham, B., Forselius, P.: An Empirical Analysis of Software Productivity over Time. In: 11th IEEE International Symposium on Software Metrics (Metrics 2005), p. 37. IEEE Computer Society, Los Alamitos (2005)
[43] Symons, C.: Come Back Function Point Analysis (Modernized) – All is Forgiven!). In: Proc. of the 4th European Conference on Software Measurement and ICT Control, FESMA-DASMA 2001, Germany, pp. 413–426 (2001)
[44] Symons, C.: Function Point Analysis: Difficulties and Improvements. IEEE Transactions on Software Engineering 14(1), 2–11 (1988)
[45] UKSMA. MkII Function Point Analysis Counting Practices Manual, v 1.3.1 (1998)