Model-based Simplified Functional Size Measurement – an Experimental Evaluation with COSMIC Function Points

Vieri del Bianco, Luigi Lavazza, Geng Liu, Sandro Morasca

Dipartimento di Scienze Teoriche e Applicate
Università degli Studi dell'Insubria – Varese, Italy

{luigi.lavazza|sandro.morasca}@uninsubria.it
{vieri.delbianco|giulio.liu}@gmail.com

Abedallah Zaid Abualkishik

IT department, Al-Khawarizmi International College
Abu Dhabi, United Arab Emirates

azasoft@yahoo.com

Abstract. Functional Size Measurement methods –like the COSMIC method– are widely used but have two major shortcomings: they require a complete and detailed knowledge of user requirements and they are carried out via relatively expensive and lengthy processes. To tackle these issues, simplified measurement processes have been proposed that can be applied to requirements specifications even if they are incomplete or not very detailed. Since software requirements can be effectively modeled using languages like UML and the models increase their level of detail and completeness through the development lifecycle, our goal is to define the characteristics of progressively refined requirements models that support progressively more sophisticated and accurate measurement processes for functional software size. We consider the COSMIC method and three simplified measurement processes, and we show how they can be carried out, based on UML diagrams. Then, the accuracy of the measurement supported by each type of UML model is empirically tested, by analyzing the results obtained on a set of projects. Our analysis shows that it is possible to write progressively more detailed and complete user requirements UML models that provide the data required by simplified methods, which provide progressively more accurate values for functional size measures of the modeled software. Conclusions. Developers that use UML for requirements model can obtain an estimation of the application's functional size early on in the development process, when only a very simple UML model has been built for the application, and can get increasingly more accurate size estimates while the knowledge of the product increases and UML models are refined accordingly.

Keywords: Functional Size Measurement; Function Points; COSMIC Function Points; Simplified measurement processes; model-based measurement; UML.

1 Introduction

Functional Size Measurement (FSM) aims at providing a measure of functional user requirements. User requirements can be expressed by using various notations, including UML. It has been shown that the most popular FSM methods –namely, IFPUG Function Points (FP) and COSMIC Function Points (CFP)– can be applied to requirements written in UML, especially if the UML models have been written with FSM in mind, so that all (and only) the information required by FSM methods is suitably represented in the models [1,2].

UML models are collections of diagrams. While progressing in the development, UML models become more and more complete and detailed and in general include an increasing number of diagrams. This means that UML models convey an increasing amount of information, which can be used for FSM [3]. This is interesting for the application of simplified FSM methods, which require only a subset of the information needed to carry out the complete official measurement processes described in manuals, such as the COSMIC counting manual [5]. Different UML models (i.e., models involving different subsets of diagram types) can support different simplified FSM methods [4].

The majority of simplified FSM methods address the simplification of Function Point Analysis, since the IFPUG process of measuring FP involves activities –such as the classification of transactions and data and the evaluation of the complexity of every transaction and logic data file– that require a relevant measurement effort, and can be carried out only when the specification of user requirements is fairly complete and detailed.

However, the process of measuring CFP (which is generally faster and less expensive than FP measuring) may also need to be simplified so it can be carried out faster and at a smaller cost than the official process required by the official counting manual [5] and on incomplete requirements specifications. This may happen because size estimates are usually needed by a given deadline (e.g., for cost estimation and bidding) or because detailed requirements specifications are not available (and will not be available for a while). Simplified measurement processes for CFP have been proposed (see for instance the section on “early or rapid approximate sizing” in [6]).

The availability of “simplified” measurement processes for CFP, which require descriptions of requirements at different levels of detail, and the fact that UML models evolve through the requirements elicitation phase by growing in completeness and details suggest the following research questions:

RQ1. During the requirements elicitation and specification phase, is it possible to write progressively more complete and detailed UML models that support progressively more accurate simplified CFP measurement methods?

RQ2. What is the accuracy of different simplified CFP measurement methods, i.e., how close are the estimated sizes they provide to the actual ones?

RQ3. Do simplified CFP measurement methods provide an accuracy level that increases with the amount of information required?

Note that we do not intend to address question RQ3 quantitatively. Rather, we look for a trade-off between the information elicitation effort required by a given size estimation method and the resulting estimate accuracy that can –subjectively– be considered reasonable.

To answer questions RQ1-RQ3, we measured a set of software applications via different simplified CFP measurement methods, using progressively more detailed and complete UML models; we obtained the values of the parameters on which the estimation methods are based and computed the estimated sizes and compared them with the sizes measured according to the COSMIC counting manual.

The remainder of the paper is organized as follows. Section 2 describes simplified measurement processes for COSMIC function points. Section 3 illustrates the UML models that support the simplified measurement processes. Section 4 illustrates the experimental analysis. Section 5 accounts for related work. Section 6 discusses the threats to the validity of the study, while Section 7 draws some conclusions and outlines future work.

2 Simplified Processes for Measuring COSMIC Function Points

The COSMIC FSM method requires that:

1. The functional processes of the application being measured be identified.
2. The data groups mentioned in the user requirements be identified.
3. For each functional process, the unique data movements involving each identified data group be counted. Data movements are classified into Entries and Exits (i.e., I/O movements) and Reads and Writes (to persistent storage). A data group is considered persistent if its value is stable between two consecutive functional process executions.

2.1 Size Estimation Based on the Mean Number of Data Movements per Functional Process

A first very rough simplification of the measurement process was proposed by COSMIC [6] and requires that only the first of the activities required for CFP measurement (the identification of functional processes) is performed. The only requirement for applying this simplified process according to [6] is the availability of historical data that allow the computation of the mean number of data movements per functional process (M_{DM}). If the software application to be measured is similar to those previously measured, it is reasonable to assume that the mean number of data movements per functional process of the new application will be close to M_{DM}. Thus,

$$CFP = M_{DM} \times \#FPr \qquad (1)$$

where #FPr is the number of Functional processes.

2.2 Size Estimation Based on the Number of Functional Processes and the Number of Data Groups.

It is reasonable to assume that the size in CFP increases with the number of data groups (#DG): the more data groups, the more opportunities for data movements. A simplified computation of CFP can thus be obtained via a model like the following:

$$\text{CFP} = f(\#\text{FPr}, \#\text{DG}) \tag{2}$$

that is, a model that computes the estimated size by means of some formula (to be defined) applied to #FPr and #DG. This procedure is simpler than the "full" COSMIC counting process, as data movements do not need to be identified and classified. Besides, a conceptual model of the data involved in the application is usually built very early in the requirements modeling process; thus, its availability is generally an easily satisfied assumption.

Equation (2) could be derived via regression analysis, provided that historical data reporting both #FPr and #DG are available.

2.3 Size Estimation Based on the Data Groups Involved in Each Functional Process

The two methods described above are based on measures (#FPr and #DG) that characterize the whole application. It is reasonable to expect that a more accurate estimate can be derived if information that characterizes each functional process individually – like the number of data groups involved in each functional process –is used. If the historical dataset includes such data, statistical analysis can yield models of type

$$\text{CFP} = f(\#\text{FPr}, \text{AvDGperFPr}) \tag{3}$$

where AvDGperFPr is the average number of data groups involved in each of the functional processes in the application to be measured.

3 UML Models Supporting Simplified CFP Measurement

In this section, we describe the UML models that are needed to support the simplified approaches to CFP measurement described in Section 2. We also present the model supporting the measure of CFP performed as described in the manual [5]. We use the Software Warehouse Portfolio application described by Fetcke [7] as an example.

The UML models used for measurement are models of the functional user specifications. They do not contain any design element; on the contrary, only information that is relevant to the user and is directly related to user's needs and requirements is allowed in models.

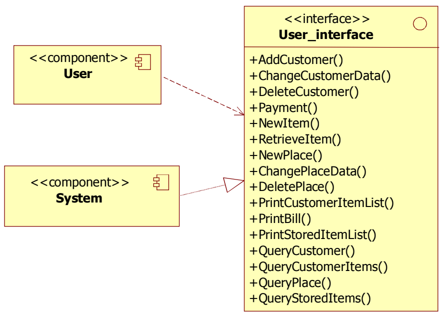


Fig. 1. UML component diagram showing the functional processes.

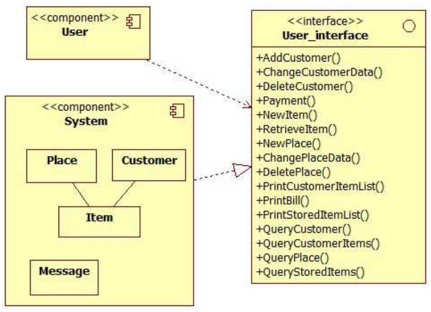
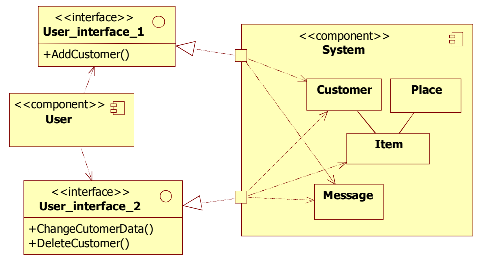


Fig. 2. UML component diagram showing the functional processes and the data groups.

Fig. 1 illustrates a UML diagram that can effectively support the first simplified measurement method, described in Section 2.1. It is a component diagram, where the interface realized by the system lists the functional processes that can be triggered by the user. So, #FPr can be obtained by counting the operations listed in the User_interface. Recall that M_{DM} can be obtained based on historical data.

Fig. 2 illustrates the same diagram as Fig. 1, in which the system component has been refined and detailed with the description of the classes that represent the data managed by the system. These classes correspond to the data groups of the COSMIC software model. It is easy to see that the diagram in Fig. 2 provides all the data needed to use equation (2), i.e. #FPr and #DG.



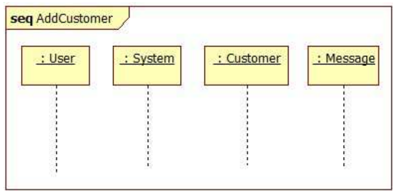


Fig. 3. UML component diagram showing the dependence of each functional process on data groups.

Fig. 4. UML sequence diagram showing the class (data group) instances participating in the AddCustomer functional process.

Fig. 3 illustrates a diagram providing the information needed to use equation (3). In the diagram, UML ports are used to precisely indicate which classes (i.e., data groups) are used in each functional process. To this end, sets of functional processes that use the same set of classes are grouped into a single interface. In Fig. 3, only the interfaces needed to add, change, and delete clients are shown. It can be noticed that grouping functional processes according to the used classes may lead to a rather large number of interfaces, which could decrease the readability of the diagram. This is true, but interfaces that are homogeneous with respect to the used classes not only allow for a quite precise estimation of size (see Section 4), but explicitly represent the logical relationship between interface elements and system data: this poses the basis for the identification of important traceability information when the design model is built.

An alternative to the model shown in Fig. 3 is a sequence diagram that shows only the classes involved in the functional process (Fig. 4). In fact, the diagram represents a specific functional process (AddCustomer) and the involved class instances. We can see that AddCustomer uses two data groups: Customer and Message. This type of diagram is convenient because it can be refined into the diagram described in Fig. 5, which supports full-fledged COSMIC measurement.

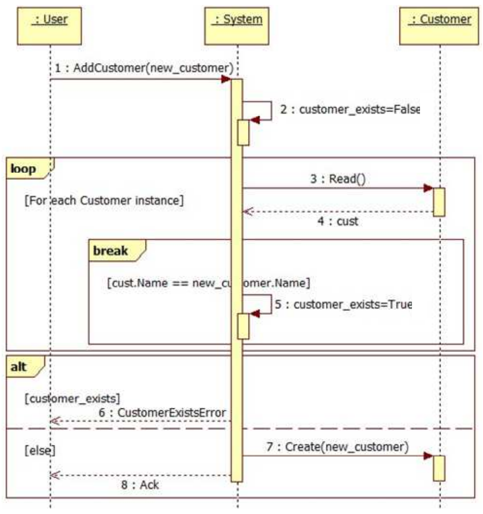


Fig. 5. UML sequence diagram showing the data movements of a given functional process

Fig. 5 illustrates a sequence diagram that contains all the information needed to measure the size of the functional process according to the COSMIC official manual [5]. Messages that cross the application boundary (in our case, messages from or to the user) are entries and exits, while messages directed to class instances representing data groups are reads or writes. Details about COSMIC measurement based on UML diagrams can be found in [1] and [2].

4 Empirical Analysis

To answer the questions defined in the Introduction, we modeled a set of software applications and measured them. Then, we applied the simplified measurement methods, obtaining size estimates that were finally compared with the measures obtained via the standard COSMIC method [5].

The projects considered were sample projects provided by COSMIC to illustrate the counting process (5 projects), academic examples used in teaching (5 projects) and project management tools (one project).

The UML models were built by a PhD student following the methodology described in [1]. The quality of the model was then checked by two of the authors. Part of the dataset containing the measures of the models of the applications described above is given in Table 1.

Table 1. The dataset

Pid	CFP	#FPr	#DG	Avg #DG per FPr	Avg DM per DG per FPr	Avg FPr size (DMperFPr) others	Avg #DG per FPr others	Avg CFP/#DG others
1	86	16	6	2.88	1.90	5.12	3.01	8.7
2	56	11	11	3.55	1.60	5.14	2.98	9.6
3	91	15	10	4.00	1.57	5.08	2.93	9.1
4	69	19	12	2.32	1.72	5.26	3.06	9.6
5	103	19	16	3.06	1.93	5.09	3.00	9.5
6	64	14	7	2.64	1.71	5.17	3.03	9.2
7	116	20	14	3.60	1.65	5.07	2.95	9.3
8	124	20	10	2.80	2.38	5.04	3.02	8.9
9	117	19	9	3.47	1.78	5.05	2.97	8.8
10	90	13	14	3.92	1.99	5.04	2.95	9.5
11	10	3	10	2.07	2.40	5.43	3.30	9.1

In Table 1, “Avg#DGperFPr” is the average number of data groups involved in the project’s functional processes; “AvgDMperDGperFPr” is the average number of data movements per data group per functional process; “AvgFPrSize (DMperFPr) others” is the mean number of data movements per functional process, computed on all the other applications; “Avg#DGperFPr others” is the mean number of data groups per functional process, computed on all the other applications; “AvgCFP/#DG_others” is the mean number of data movements (i.e., size) per data group, computed on all other applications.

When estimating the size of an application using equation (1), we used the M_{DM} of the other projects. The M_{DM} is equivalent to the mean CFP/NumFPr, i.e., to the mean size of Functional processes. Using this model we got the estimates reported in Table 2. The obtained estimates are characterized by MMRE = 17.8%, Pred(25) = 72.7%, percentage error range = [-27.8%, 44.9%].

While analyzing the dataset, we discovered that the mean number of data movements per data group involved in a functional process, computed for each application, was fairly constant throughout the applications of our dataset: the mean is 1.88 and the standard deviation 0.29 (i.e., 15% of the mean). We exploit this fact to define the following model:

$$\text{CFP} = \text{NumFPr} \times \text{AvDGperFPr} \times \text{AvDMperDGperFPr} \qquad (4)$$

where (AvDGperFPr × AvDMperDGperFPr) is an estimate of the number of data movements per functional process, i.e., an estimate of the mean size of functional processes: multiplied by the number of functional processes it yields an estimate of the number of data movements, i.e., the size of the application.

By using this model, we obtained the estimates reported in Table 2 and characterized by MMRE = 15.3%, Pred(25) = 81.8%, percentage error range [-15.3%, 33.9%].

Table 2. Estimates obtained via equations (1) and (4)

		Estimates obtained via eq. (1)			**Estimates obtained via eq. (4)**		
P.Id	**Act. Size [CFP]**	**Est. Size [CFP]**	**Err. [CFP]**	**% Err.**	**Est. Size [CFP]**	**Err. [CFP]**	**% Err.**
1	86	82	-4	-4.7%	88	2	2.3%
2	56	57	1	1.8%	75	19	33.9%
3	101	86	-15	-14.9%	132	31	30.7%
4	69	100	31	44.9%	85	16	23.2%
5	103	92	-11	-10.7%	105	2	1.9%
6	64	72	8	12.5%	71	7	10.9%
7	116	101	-15	-12.9%	139	23	19.8%
8	124	101	-23	-18.5%	105	-19	-15.3%
9	117	96	-21	-17.9%	127	10	8.5%
10	90	65	-25	-27.8%	97	7	7.8%
11	252	326	74	29.4%	218	-34	-13.5%

5 Related Work

Many techniques for early size estimation have been proposed for Function Points (e.g., the Early and Quick Function Point by Conte et al. [8]). The empirical evaluation of these techniques indicates that some actually yield reasonable estimates [11]. On the contrary, hardly any work has been devoted to defining simplified measurement processes for the COSMIC method.

In [9], the dataset published in [10] was used to derive a linear OLS regression model that can be used to estimate the size in CFP, given the number of transactions identified via Function Point Analysis. This can be considered a sort of simplified CFP measurement method, since the identification transaction functions is an activity mush simpler and shorter than both the full fledged CFP or FP counting processes

Several authors studied the possibility of basing standard CFP measurement [5] on UML models of user requirements; i.e., they consider the models that are available after the completion of the requirements elicitation and specification phase.

Hericko and Zivkovic address size estimation in iterative development [3]. Their approach enables early size estimation using UML. However, they do not consider simplified measurement processes. In fact, their method deals with the evolution of the functionality through iterations, rather than the level of detail that can be achieved in the requirements elicitation and specification phase, as we do.

6 Threats to Validity

A possible threat to internal validity is the limited number of projects in our sample.

The main threat to the external validity of the study may come from the projects chosen, which are a limited sample of a much larger population. However, this kind of threat is typical in most empirical software engineering studies. Also, the sample of projects is a “convenience” sample, i.e., it is made of projects that were selected because the data that we needed for our study were available. Note that, however, we are not interested here in specific models (e.g., we are not interested in the coefficients of the models), but, rather, in the performance of the techniques we propose. At any rate, it is not easy to assess the extent to which our results may apply in general.

There may be a threat to construct validity due to the use of MMRE, which has been criticized in the past as an accuracy indicator [13].

One might also observe that only one of the projects used within this empirical study is a real implemented project, and that this fact could possibly decrease the reliability of the results or their generality. However, this is not actually a problem, for two reasons. One is that the requirements of all our projects were realistic: any of our projects could be implemented, thus requiring for size measurement, effort estimation, etc. The second is that we are interested in the *comparison* of measures obtained via simplified and full-fledged processes. Therefore, some characteristic requirements that affect the standard size measure are bound to affect in the same way the simplified measure, so that the results are hardly affected.

7 Conclusions

Simplified FSM methods are often used when a project manager needs an estimate of the functional size of the software application to be built before the requirement specification phase is completed, or when the cost or time allowed for measurement are limited. When UML is used in the early phases of development, it is convenient to apply simplified FSM methods to UML models. In this paper we showed that it is possible to build UML models that adequately support the application of simplified measurement methods and the standard COSMIC method. In particular, during the requirements specification phase, UML models grow in detail, thus providing the information required by progressively more accurate size estimation methods. In fact, it was possible to define quantitative size estimation models based on only the number of functional processes, or the number of functional processes and the number of data movements in each functional process.

To practitioners, our results provide an interesting hint: the information contained in the UML models illustrated in Section 3 is just the information required to document applications’ requirements properly; accordingly, in the requirements specification phase the analyst *must* indicate what data are involved in each functional process, and how they are used. Therefore, size estimates can be seen as ‘by products’ of the progressive refinement of UML requirements models.

Acknowledgments

The research presented in this paper has been partially funded by the project “Metodi, tecniche e strumenti per l’analisi, l’implementazione e la valutazione di sistemi software” funded by the Università degli Studi dell’Insubria and by “Progetto dote 2 - programma UNIRE - accordo per lo sviluppo capitale umano nel sistema universitario lombardo”, co-funded by Regione Lombardia and Università degli Studi dell’Insubria.

References

1. Lavazza, L., del Bianco, V., Garavaglia, C.: Model-based Functional Size Measurement, ESEM 2008, 2nd Int. Symp. on Empirical Software Engineering and Measurement, Kaiserslautern, Germany. October 9-10, 2008.
2. Lavazza, L., del Bianco, V.: A Case Study in COSMIC Functional Size Measurement: the Rice Cooker Revisited, Amsterdam, IWSM/Mensura 2009, November 4-6, 2009.
3. Hericko, M., Zivkovic, A.: The size and effort estimates in iterative development, Information and Software Technology vol. 50 n.7, 2008, pp.772-781.
4. del Bianco, V., Lavazza, L., Morasca, S.: A Proposal for Simplified Model-Based Cost Estimation Models, 13th Int. Conf. on Product-Focused Software Development and Process Improvement – PROFES 2012, Madrid, June 13-15, 2012.
5. COSMIC – Common Software Measurement International Consortium: The COSMIC Functional Size Measurement Method - version 3.0.1 Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2003), May 2009.
6. COSMIC: The COSMIC Functional Size Measurement Method - Version 3.0 - Advanced and Related Topics, December 2007.
7. Fetcke, T.: The warehouse software portfolio, a case study in functional size measurement, Technical report no.1999-20, Département d’informatique, Université du Quebec à Montréal, Canada, 1999.
8. Conte, M., Iorio, T., Meli, R., Santillo, L.: E&Q: An early and quick approach to functional size measurement methods, 1st Software Metrics European Forum (SMEF 2004), Roma, January 2004.
9. Lavazza, L.: Convertibility of functional size measurements: new insights and methodological issues, 5th Int. Conf. on Predictor Models in Software Engineering, 2009.
10. Van Heeringen, H.: Changing from FPA to COSMIC - A transition framework. in Proceedings Software Measurement European Forum (SMFE), Rome, Italy, 2007.
11. Lavazza, L., Liu, G.: A Report on Using Simplified Function Point Measurement Processes. The Seventh Int. Conf. on Software Engineering Advances. Lisbon, 2012.
12. Barkallah, S., Gherbi, A., Abran, A.:COSMIC Functional Size Measurement Using UML Models. In proceeding of: Software Engineering, Business Continuity, and Education - International Conferences ASEA, DRBC and EL, pp.137-146. 2011.
13. Kitchenham, B.A., Pickard, L.M., MacDonell, S.G., Shepperd, M.J.: What accuracy statistics really measure. IEE Proceedings - Software, 148(3), June 2001, pp. 81-85.