Methods of functional measurement of software

Katalin Nátz*, Tamás Orosz**, Zsigmond Gábor Szalay*
* Szent István University, Budapest, Hungary
** Óbuda University/AMK, Székesfehérvár, Hungary
* natz.kata@gmail.com
**orosz.tamas@amk.uni-obuda.hu
szalay.zsigmond.gabor@szie.hu

***Abstract* — Nowadays software development is the most expensive element of a projects in the information technology sector. Many software projects fail due to budget overruns, late delivery or incorrectly planned forecasts.**
Software products could be manufactured if they have such frameworks that the budget can be constrained. Before developing a software product, it is important to estimate the cost of software development.
The estimation of software costs - the so-called software cost estimation model should predict the effort that is indispensable for building a software system. In the last 40 years, many estimation models have been composed for the following aspects: budgeting, risk analysis, project planning and control, and investment analysis for software improvements. Software provides the IT landscape for business processes and enables the reusability of the software.
There are numerous ways of measuring the functional size of the software: COCOMO (Constructive Cost Model) and ESTIMACS based on the parameters of implementation ability, flexibility and traceability as well as techniques for software cost estimation. Index terms - Function points (FP), Cosmic, SLOC (Lines of code), Use Case Points (UCP) etc.

***Keywords*: Function point, Story point, Use Case point, Software measurement, Business processes, Financial planning**

I. INTRODUCTION

Applications and large integrated systems such as enterprise information and cloud-based systems are evolving very rapidly, and the number of users associated with them is also growing. These applications work with so much data and are so complex that it is difficult for the average user to understand how they work. New coding tools are already enabling software developers to meet ever-increasing and increasingly complex customer needs more efficiently and quickly. An understandable, comparable and clear method is needed to interpret, estimate and monitor projects [12].

The purpose of FSM (Functional Size Measurement) methods is to determine the size of software by quantifying the totality of functions provided to users. Functional point analysis (FPA) was the first FSM method [1][3][9]; it was originally introduced in the mid-1970s. Today, businesses around the world use this methodology. Allan Albrecht, of IBM, was the first to develop a method to assess the limit and extent of measurement. Its name is forever intertwined with the issue of software functionality. However, one of the main results is that it made the productivity of software projects measurable using the method [4]. Since its founding in 1986, the International Function Point Users Group (IFPUG) has continuously developed and updated the original Albrecht methodology for software (IFPUG Counting Practices Manual) [3][4]. In 2009, a new version of the IFPUG Function Point (FP) standard, CPM 4.3.1, was released and has been in use since early 2010 [4].

The appearance of the FPA technique has enabled the ICT community to facilitate the practice of measuring software functionality in relation to the "line of code" (LOC) approach [5]. However, all this requires clear, complete and detailed descriptions, application and design documents [2].

In one of their research, Meli and Santillo mention two cases where it would be very important to set up an alternative estimation method that is compatible with FP standard rules. One case came to the fore when a development project was at such an early stage when it was simply not yet possible to calculate FP according to IFPUG standards [18].

The second case was encountered when the detailed documentation for the measurement of the existing software, which should have been prepared during the development of the project, was not available [2].

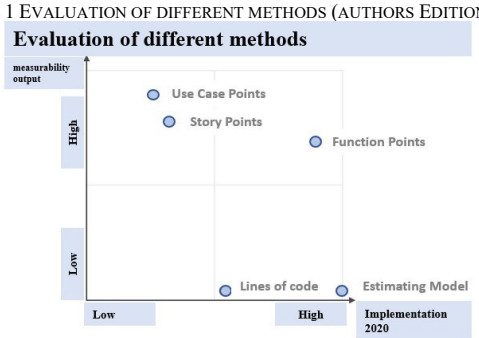
The demand for methods to estimate function points has been increased for organizations involved in the software business. This need is becoming increasingly indispensable in agile software development, which need usable and cost-effective software [15].

II. METHODS FOR MEASUREMENT

During software development, it is not easy to decide which method would be applicable to both traditional and agile and mixed-development software.

There are several methods for measuring emissions (Table 1), which must meet the following conditions: the method of measurement used in the future must be independent of the type of software or the specific technology [16].

TABLE 1 EVALUATION OF DIFFERENT METHODS (AUTHORS EDITION)



The functional size of the developed software can be determined, for example, by the amount of source code. Today, however, thanks to modern programming languages and the development environment, more and more functions can be developed with less and less source code, so the Lines of Code method did not meet the conditions. Within the company, several methods have been used for years, which are as follows:

- Story Point
- Use Case Points
- Lines of code
- COCOMO
- Function Point

III. STORY POINT

In agile software development, the extent of performance is usually determined by user stories, so-called it is described with a User Story and the magnitude of the related development is estimated in the "Story Point" unit.

The Story Point, which also determines the complexity of the User Story, is estimated within a given development team (and not always by a score, but by, for example, T-shirt size - S, M, L, XL, XXL). Of course, comparability of different teams is not possible, but it is not a goal. Within the company group, the projects are approx. 40-55% do not use the agile method.

IV. USE CASE POINTS

The Use Case Points method is an estimation method that starts from the function point method and forms a bridge to the object-oriented world; however, Use Case Points and function points cannot be mapped directly from each other. In the case of the UCP method, the Use Cases and the so-called actors are measured, and the associated transactions are classified. For each Use Case, the complexity of the associated transactions is also determined.

To be able to use the UCP method in general in all areas of the company, it would require UML modeling and comprehensive implementation of Use Cases, however, this is impossible due to the current corporate structures and their operation. However, if it did, it would be the most cost-effective method in the long run - UML can be used to automate the process.

V. LINE OF CODE

Lines of codes (SLOC) are software metrics that can be used to measure the functional size of an application by counting the number of lines that can be found in the text of the program's source code. SLOC is one of the most widely used sizing metrics in telecommunication sector and otherwise used to forecast the effort required to develop an application and to estimate productivity or maintainability after the software has been designed. There are some disadvantages to Line of Codes metrics that effectively operates the quality of the software because the output of SLOC metrics is used as input to other software estimation methods such as the COCOMO model [14][15].

VI. COCOMO

The COCOMO model (Constructive Cost Model) offers direct estimates of the effort. This is a simple model based on inputs related to the size of the system and several cost drivers that affect productivity.

COCOMO is a summary of three models: a base model that is applied at the beginning of the project, an intermediate model that is applied after the requirements are specified, and an advanced model that is applied after the design is completed [16]

VII. FUNCTION POINT ANALYSIS (FPA)

The function point, as a unit of measure, determines the functional size of the applications or software that are created or modified or deleted during the prepared development, including the functions supported by the IT system beyond the measurement boundaries. However, of the functions developed, only those that are ordered by the client can be measured.

The FPA method is suitable for measuring entire applications, projects or releases. The method is independent of the type of application or development. It can also be called a structured problem-solving algorithm, which breaks down systems into smaller components in order to make the technical description and implementation of a given project more understandable and analyzable.

A. Basics of FPA

The user view is decisive for the determination of the functional size, if one wants to determine the function points. There are special rules in the standard for assigning the point value to transactions and databases. Regarding to the number of elementary functions and the standardized values, the assigned FPs is accumulated.

In the analysis, the elementary processes are the smallest unit that has a meaning for user and constitutes a complete transaction. It is counted as the same compared with another one if it requires the same set of Data Element Types, requires the same set of File Type Referenced and requires the same set of processing logic to complete the elementary process [4][8][13].

B. Why or why not should we use the FPA method?

FPA offers many advantages over other software sizing methods [4][8][9]:

• It is independent of the programming language, development technology, platform or the knowledge of the project members.

• It connects directly with user requirements and features.

• Those who engage in FPA analysis play a consultative role in this case. The function points can help to create effective communication between the developers and users, and to give a better understanding to the non-technical user.

• It can be used on both "Waterfall" and "Agile" projects.

• It is applicable throughout the software development lifecycle.

• It provides a way to determine productivity and estimate the cost.

• FPA provides consistent, documented and repeatable measurement methods.

• FPA can highlight gaps in functional requirements, thus avoiding the early introduction of errors in the application.

• A goal is also to minimize the effort and severity of the measurement process [11][13].

C. *Transactional functions*

Transactions are the function that the user provides to process application data. There are three transactions - external input (EI), external output (EO) and external inquire (EQ) (Table 2)

External Inputs: The main purpose of an input is to maintain one of several internal databases or to change system behavior. To process technical data or control information, an input contains processing logic that goes into the application beyond the limit value [4][9].

External Outputs: The output provides information to the user. It contains at least one of the following forms of processing logic: mathematic computations, maintenance of datasets, generation of derived data, change of system behavior [4][9].

External Inquires: An inquiry is the presentation of information to the user, and it references a dataset to read technical data or control information, and it does not meet the requirements for an EO. It can be also a list box. It is important to mention that sorting and arrangement of data cannot be evaluated at all. [4][9].

In practice, the difference between EO and EQ is often difficult to impossible. A query should only be evaluated if it is clearly evident that the requirements for an output are not met, e.g. in a "simple" list box or multiple search.

D. *Data Functions*

Data functions show logical data groupings and databases that the user needs for his work. Data functions have two types - Internal Logical Files (internal data) and External Interface Files (external data).

Internal Logical Files (ILF): A dataset that is maintained within the considered application is classified as internal dataset: user-recognizable, logical groups of functional data, maintained by elementary processes of the application [4][7].

External Interface Files EIF): A dataset that is read-only but not maintained within the considered application and that is classified as ILF in at least one other application is classified as an External Interface File: user-identifiable, logical group of functional data that is not changed in the application / to be cared for. It is maintained by elementary processes of another application [4][7].

E. *Complexity of the transactions and data functions*

Important for the determinations of the complexity are: for a transaction the number of used fields and datasets and for a dataset the number of included fields and field groups.

TABLE 2 COMPLEXITY OF EI, EO, EQ (AUTHORS EDITION)

EI		DET		
		1-4	5-15	>=16
FTR	0-1	3	3	4
	2	3	4	6
	>=3	4	6	6

EO		DET		
		1-5	6-19	>=20
FTR	0-1	4	4	5
	2-3	4	5	7
	>=4	5	7	7

EQ		DET		
		1-5	6-19	>=20
FTR	0-1	3	3	4
	2-3	3	4	6
	>=4	4	6	6

Both files include Data Element Type and / or Record Element Type. Both files contain the data element type and / or the data record element type. A data element type (DET) is a unique, user-recognizable, non-repeated attribute or field of an ILF or EIF (Table 3). A record element type (RET) is a user-definable subset of data elements in ILF or EIF. When ILF / EIF are referred to as Transactional Functions for Processing Information, they are termed as File Type Referenced (FTR) [9].

TABLE 3 COMPLEXITY OF ILF, EIF (AUTHORS EDITION)

ILF		DET		
		1-19	20-50	>=51
RET	1	7	7	10
	2-5	7	10	15
	>=6	10	15	15

EIF		DET		
		1-19	20-50	>=51
RET	1	5	5	7
	2-5	5	7	10
	>=6	7	10	10

VIII. CONCLUSION

There are several methods for determining the functional size of applications. It is always the situation that decides which method the company chooses. If UML models are available, it is almost clear that the Use Case Point method is worthwhile, as it can automate the process. If the complete documentation is available, the function point analysis can be used to obtain results.

Nonetheless, all studies advise to use COSMIC FPA for size estimation. But as the documentation is often incomplete in an agile development, it should conduct many interviews, select the information and identify the data movements. In this case, there are such assumptions that no automated estimates can be made.

Regarding the multinational companies, FPA method is perhaps the easiest way to measure applications and there is the possibility to figure out automation. Unlike other methods, is a more tangible methodology for many companies [16].

To conduct a Function Point Analysis, it will be necessary to use the functional requirements provided by the project team. We differentiate between transactional functions and data functions. After identifying the transactional functions and data functions we must determine the complexity and the functional size for each. Regardless of the implementation technique used, it is possible to determine the size of the application. With the help of the results the average productivity can be estimated. If you know the experience of productivity, you can roughly calculate the project hours and the associated budget [4].

According to the measurement results, it can be summarized that the FPA method is not 100% accurate, the subjective factors cannot be disregarded. The smaller a project is, the greater the deviation and variability of the FPA calculation is due to the characteristics of the method. The reason can be the lack of weighting and granularity. Therefore, a modified model should be built where the KPI -s are refined after weighting. According to the new trends, the agile method is to be enriched, since the agile methodology was introduced in the near past.

REFERENCES

[1] A.J. Albrecht: Measuring application development productivity, in: Proceedings of the Joint SHARE/GUIDE/IBM Application Development Symposium, 1979, pp. 83–91.

[2] Mahir Kaya, Onur Demirörs: E-Cosmic: A Business Process Model Based Functional Size Estimation Approach, 2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications, DOI:10.1109/SEAA.2011.60

[3] The Object Management Group: Automated Function Points, Publisher OMG, 2014

[4] B. Pönsgen: Function-Point Analyse. Ein Praxishandbuch. 2nd edn. Dpunkt.verlag, Heidelberg (2012)

[5] N. Balaji, N. Shivakumar & V. Vignaraj Ananth: Software Cost Estimation using Function Point with Non-Algorithmic Approach, Global Journal of Computer Science and Technology Software & Data Engineering Volume 13 Issue 8 Version 1.0 Year 2013

[6] IFPUG: Function Point Counting Practices Manual, Release 4.0. Couting Practices Committee, The international Function Points Users Group (2000)

[7] D. Longstreet: Fundamentals of Function Point Analysis, Total Metrics, Software Development Magazine, 2005

[8] S. Palmquist, M.A. Lapham, S. Miller, T. Chick, I. Ozkaya: Parallel Worlds: Agile and Waterfall Differences and Similarities. Software Engineering Institute (2013)

[9] P. Vickers: An Introduction to Function Point Analysis. School of Computing and Mathematical Sciences Liverpool John Moores University, UK, 1998.

[10] C. Dekkers: Counting Function Points for Agile / Iterative Software Development; http://www.ifpug.org/Articles/Dekkers-CountingAgileProjects.pdf; last accessed 2018/05/06

[11] B. Molnár: Funkciópont elemzés a gyakorlatban; MTA Információtechnológiai Alapítvány (2003), DOI: 10.13140/RG.2.2.12580.68484

[12] Ömüral, Demirörs: Effort estimation methods for ERP projects based on function points; DOI: 10.1145/3143434.3143464

[13] K.Nátz, T. Orosz: Function point analysis by an SAP application, in AIS 2018: 13th International Symposium on Applied Informatics and Related Areas

[14] K. Bhatt1, V.Tarey, P. Patel: Analysis Of Source Lines Of Code(SLOC) Metric, International Journal of Emerging Technology and Advanced Engineering (ISSN 2250-2459, Volume 2, Issue 5, May 2012)

[15] S. Bhatia, J. Malhotra: A survey on impact of lines of code on software complexity, published in 2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014), DOI: 10.1109/ICAETR.2014.7012875

[16] M.Alkoffash, A. Alrabea: Which Software Cost Estimation Model to Choose in a Particular Project, Journal of Computer Science 4 (7): 606-612, 2008, DOI: 10.3844/jcssp.2008.606.612

[17] D. Hallman: Die COCOMO-Modelle im Licht der agilen Softwareentwicklung,in BAMBERGER BEITRÄGEZUR WIRTSCHAFTSINFORMATIK UND ANGEWANDTEN INFORMATIK, 2018, https://doi.org/10.20378/irbo-53211

[18] R. Meli and L. Santillo, "Function Point Estimation Methods: Comparative Overview", FESMA '99 Conference proceedings, Amsterdam, October 1999.