Using Logic Concepts on Software Measurement

Luiz A. de Lima[1], Jair M. Abe[1,2], Caique Z. Kirilo[1,2] , Jonas P. da Silva[1],
Alvaro C. de Frederico[1], Kazumi Nakamatsu[3]

[1]Graduate Program in Production Engineering - Paulista University, UNIP, São Paulo, Brazil
[2]Software Engineering Research Group, Paulista University, UNIP, São Paulo, Brazil
[3] School of Human Science and Environment/H.S.E, University of Hyogo, Gakuennishi-machi, Nishi-ku,Kobe-shi, 8-2-1, JAPAN
luiz@wcisp.com.br, jairabe@uol.com.br, caique.kirilo@hotmail.com, jonassunip@gmail.com,
alvaro.correa.frederico@gmail.com, nakamatu@shse.u-hyogo.ac.jp

Abstract: The decision making on the Brazilian software and service market for the acquisition or transformation of software since the end of the 1990s has as a criterion for the use of the metric technique of function point Software Size. In this paper, support for technology, distortions between expert opinions in function point counting of new software, the inclusion of benefits in decision making by managers or managers of companies. For this, the objective is to make use of concepts supported in paraconsistent logic.

Key-Words: Paraconsistent Decision Method, Paraconsistent logic, Paraconsistent Annotated Evidential Logic Eτ, Decision-Making, Software Measurement, Quality.

1. Introduction

The word measure in the systems area is a way of quantifying the characteristics of the processes involved in the whole lifecycle of a project in the creation of software. This measurement is hugely relevant for companies in public and private sector because it supports pricing, help in defining resources, scheduling, cost control, product quality, and especially in decision making when contracting software or services.

In the constant search for Engineering in measuring software size, the SLOC (Source Lines of Code) metric technique, standardized by [1]. Moreover, measures the size of the software by the number of lines of the original code in line with the programming language used in the construction of the software. In the mid-1970s, a new form of measurement, known as the function point analysis technique, proposed by [2] Employee of the International Business Machines (IBM). This method was introduced in New York around October 1979 in a symposium entitled "Measuring the Productivity of Application Development." In 1986, the IFPUG (International Function Point Users Group) Of the function point analysis technique and in the construction of the manual of the practice of counting, known as for as [3].

In the 90's, the Brazilian Function Point Users Group (BFPUG) was created, a Brazilian group dedicated to the technique of point function analysis[10]. This decade also saw the emergence of the first [4][10], Based on fundamental concepts of non-functional requirements of users and functional users, where the requirements must be raised by those involved who know the characteristics of functional size measurement method.

This paper is a qualitative study. Qualitative research is a type of the investigation that “produce findings not arrived at by statistical procedures or other means of quantification” [5]. Regarding this research, we aim to analyze qualitatively the data about software metrics process collected through multiple case study involving public and private sector companies which focus is on software development.

Regarding the software engineering research area, the analysis of threats is crucial to reduce or mitigate the risk of the study [6]. According to these authors, four types of threats must be verified: Descriptive Validity (Factual Accuracy), Theoretical Validity, Generalizability and Interpretive Validity (Objective Researcher). Figure 1 presents the questions for each threat that must be answered to ensure the research completeness.

Table 1. Categorization of Validity Threats adapted [6]

Threats	Questions
1. Descriptive Validity (Factual Accuracy)	Could we describe the objective/subjective truth accurately?
2. Theoretical Validity	What is a confounding factor

ISSN: 2367-8895

		(uncontrollability)? Do we capture what we intend to achieve?
3.	Generalizability	To what degree can we generalize results? INTERNAL (Within groups/communities/a company) EXTERNAL (Across groups/communities/companies)
4.	Interpretive Validity (Objective Researcher)	Are the conclusions/inferences drawn reasonable given the data representing an objective/subjective truth?

The logic for Aristotle is a tool for right thinking. The propositions raised as an argument and inferred, in conclusion, are based on observations[14]. Therefore, the conclusion and the propositions cannot be treated as only truth or falsehood, but always observe, seeking to feed the reasoning knowledge. The propositions raised from reality must follow three Fundamental Principles of Logic: Principle of identity (X is equal to itself and unlike all the rest); Principle of non-contradiction (no statement can be both true and false at the same time); Law of excluded middle (there is no third possibility, besides true and false).

Paraconsistent Logic is among the non-classical logical calla [9] since it contains provisions contrary to some of the fundamental principles of Aristotelian Logic, such as the principle of contradiction. From the Aristotelian point of view, the three principles of logic prevail. The predecessors of the Paraconsistent Logic were the Polish logician J. ŁukasiewiczLvov in 1878 and the Russian philosopher N.A. Vasilév. Vasilév baptized a logic that became known as imaginary. Łukasiewiczanounced the trivalent Logic: True, False, Possible. The first logical to structure a paraconsistent propositional calculation was the Polish S. Jaśkowski, the disciple of Łukasiewicz.

The term "Paraconsistent" literally means 'next to consistency.' However, 1976 the philosopher Francisco Miró Quesada, called the logic of "Paraconsistent." According to the Paraconsistent Logic, a sentence and its negation may both be true [15]. In the mid-1950s, the Polish S. Jaskowski and the logical mathematician Newton C. A. da Costa proposed the contradiction in the logical structure and became known as the founders of ParaconsistentLogic [8].

2. Problem Formulation

2.1 Types of Function Point Counts

The technique of counting function points has a differential of the SLOC technique, the unlinking of the technology used in the programming language, that is, it is possible to measure the size independently of the technology utilized in the construction of the software. Initially, identify the type of project to be counted, since each one has its characteristics and formulas.

Projects can be identified in three categories, such as project application, project development, improvement project. In the application design class, we have a counting feature, where software must exist and be installed in computational environments. Already in the type of development project, it is characterized by not having built software, installed in any computational environment, since it is in the design of new software.

This type has another strong characteristic since we can only estimate the function point, as well as the improvement project. This type is only used for changes made to application projects, since something is needed to be changed, unlike the development plan, there is nothing to change, because there is no software yet.

Note that in figure 1, we can start an estimate by choosing the type of development project or type of improvement project. Only after this counting estimated using the function point counting technique, we were able to update or start a count of the application project type, thus finalizing the project count.

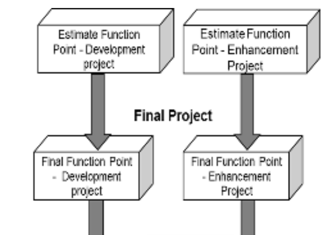

Fig.1 Relationship between of Types of Function Point Counts (Source: Author)

2.2 Function Point Counting Process

2.2.1 Gather documents, Determine the type of function point count, the application boundary, scope, requirements functional

The counting process is started as shown in figure 2, we have, as the first stage, the elicitation, consolidation, validation of information and functional requirements on the system. Thus, it becomes possible to identify the type of counting that can be used and delimit the boundary of the application and scope.

2.2.2 Measure data function, Measure transactional function

At this point in the counting process, we identify the two data functions: Internal Logical File (ILF), External Interface File (EIF) and the three transaction functions: External Inquiries, External Inputs, and External Outputs.

The Internal Logical File (ILF) has the characteristics of being a file maintained by the counted application.

The External Interface File (EIF) has characteristics of being a file kept by another application outside the border.

Calculate the function point count: The counting process goes from the first stage of elicitation, consolidation, validation of information and requirements until finalization of the count with adjustment factors, characteristics that can influence the counting and formalization with those involved.

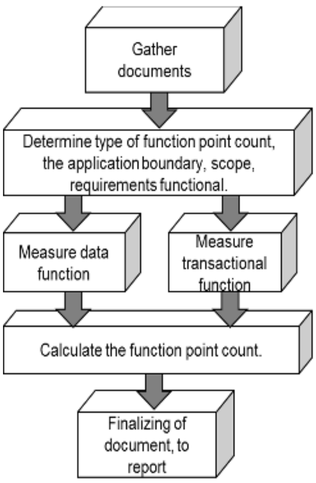

Fig.2 Function Point Counting Process (Source: author)

The counting process is started as shown in figure 2, where we have as elicitation stage information and functional requirements on the system, making it possible to identify the type of count that will be used to delimit the application boundary and scope of the count.

2.2.3 Functional Complexity (data function and transactional function)

At this point in the counting process, we identify the two data functions: Internal Logical File (ILF), External Interface File (EIF) and the three transaction functions: External Inquiries, External Inputs, and External Outputs.

Functional complexity: It is the degree of particular complexity assigned a function, using the rules defined in this international standard.

File Types Referenced (FTRs) is a data function read and/or maintained by a transaction function.

At this point in the counting process, we cross over the information to obtain the function point quantity according to the functional complexity according to the data functions: Internal Logical File (ILF), External Transaction File (EIF), and transaction functions: External Inquiries, External Inputs, External Outputs.

Data Element Types (DET) is a unique attribute, recognized by the user and not repeated.

		DETs		
		1 - 4	5 - 15	> 15
FTRs	0 - 1	Low	Low	Average
	2	Low	Average	High
	>2	Average	High	High

Fig.3 Function Point Complexity, External[3].

		Type-Size
		EI
Functional Complexity	Low	3
	Average	4
	High	6

Fig.4 Function Point Complexity, size, External Inputs [3].

		DETs		
		1 - 5	6 - 19	> 19
FTRs	0 - 1	Low	Low	Average
	2 - 3	Low	Average	High
	> 3	Average	High	High

Fig.5 Function Point Complexity, EQ, EO. [3].

		Type-Size	
		EQ	EO
Functional Complexity	Low	3	4
	Average	4	5
	High	6	7

Fig.6 Function Point Complexity, size, EQ, EO. [3].

Record Element Types (RETs) is a subset of elementary data recognized by the user within a data function. At this point of the count, we cross-check the information to obtain the function point quantity according to the functional complexity according to the data functions: Internal Logical File (ILF) and External Interface File (EIF).

		DETs		
		1 - 19	20 - 50	> 50
FTRs	1	Low	Low	Average
	2 - 5	Low	Average	High
	> 5	Average	High	High

Fig.7 Function Point Complexity, ILF, EIF. [3].

		Type-Size	
		ILF	EIF
Functional Complexity	Low	7	5
	Average	10	7
	High	15	10

Fig.8 Function Point Complexity, size, ILF, EIF. [3].

2.2.4 Calculate the function point count

The calculation is done by standardized formulas located in the manual of accounting practices [3], such as:

- Formula Development Function Point: DFP = (UFP + CFP) * VAF
- Formula Enhancement Project Function Point: EPP = [(ADD + CHGA + CFP) * VAFA] + (DEL * VAFB)
- Formula Initial Adjusted Application Functional Size: AFP = ADD * VAF
- Formula Adjusted Application Functional Size After Enhancement Projects: AFPA = [(AFPB + ADD + CHGA) - (CHGB + DEL)] * VAFA

In the process of finalization, the report of the count must be formalized totalizing the number of counted function points

3. Problem Solution

3.1. An Expert system based on Paraconsistent logic

It is common to be accustomed to using in a limited way the classical logic that allows us to obtain only two binary form results (0 or 1) or even qualitatively true or false. There are other logics known as non-classical which makes the result more likely, unlike classical logic. This logic can be seen in [7] which is synthesized by a paraconsistent method of decision. The use of non-classical logic (paraconsistent annotated evidential logic) for decision making, it is proposed by eight steps below: [8][15]:

1. Setting the level of demand: depends on the responsibility that implies in the decision making.
2. Choice of influence factors: need to elicit factors that led to success/failure in the enterprise.
3. Establishment of the sections for each factor: the possible answers (parameterized, by bands) to support decision making.
4. Construction of databases: used as a repository of information, such as weight, matrix, data to support decision making.
5. Field research: Critical to assessing the real condition of each of the factors of influence.
6. Calculation of the resulting annotations: before the result obtained from the research, calculate the MAX (maximum degree of favorable evidence and the minimum degree of contrary evidence) and MIN (minimum degree of favorable evidence and the minimum degree of contrary evidence).
7. Determination of the global analysis: all the calculations necessary to compose support for decision making.
8. Decision making: Used the analyzer algorithm that will provide such answers for support in decision making.

3.2. Unifying concepts

In the face of scenarios involving a specialist, such as CFPS (function point specialists), we see the possibility of applying non-classical logic [9] as support in decision-making to determine if counting in new projects, requiring a defense for recounting from the project.

When defining paraconsistent logic, the following proposition is proposed: "The need to defend the project. "We are opening a new opportunity to propose a solution to the software market with benefits for both companies and suppliers (software factory) in the acquisition of new software[11][12]. Moreover, for this we will use a repository of counts made in the banking niche that resulted in many defenses, which could be avoided, thus reducing project costs.

With the observation of the information in the bases, we will qualitatively analyze the behavior in the answers of the experts in counts using the technical point of function analysis and decisions[13] will be suggested to answer the proposition mentioned above.

4. Conclusion

ISSN: 2367-8895

In this paper, we had the perception of the feasibility and importance of research in the national scope of companies that need to contract new software, besides attending software factories that act as suppliers.

Significant results are expected to culminate in a technological tool to support decision making by managers and mitigate rework and combat wasteful efforts.

Software Engineering Research Group by Brazil has supported this work.

References:

[1] IEEE Standard for Software Productivity Metrics (IEEE Std 1045 1992). S .l. : IEEE, 1993.

[2] Albrecht, Allan J. Measuring Application Development Productivity. In: Proceeding of the Joint SHARE/GUIDE/IBM Application Development Symposium, vol 10, pp.83-92. October 1979,http://www.bfpug.com.br/Artigos/Albrecht/MeasuringApplicationDevelopmentProductivity.pdf. S .l. : Symposium, 1979.

[3] CPM. Function Point Counting Practices Manual, Release 4.3.1. *Release 4.3.1.* S .l. : IFPUG, 2010.

[4] ISO/IEC14143-1:. Information technology Software measurement Functional size measurement Definition of concepts.", Standardization, International Organization, ISO. Geneva, 1996: s.n., 1996

[5] Strauss, Anselm, and Corbin, Juliet. Basics of qualitative research. Newbury Park, CA: Sage, 1990. Vol. Vol. 15.

[6] K. Petersen and Gencel, C. Worldviews, research methods, and their relationship to validity in empirical software engineering research, in Proceedings of the 2013 Joint Conference of the 23rd International Workshop on Software Measurement and the 2013 Eighth International Confe. S .l. : pp.81–89., 2013.

[7] Carvalho, F. R., & Abe, J. M. Tomadas de Decisão com Ferramentas da Lógica Paraconstistente Anotada. São Paulo : Edgard Blucher Ltda., 2011.

[8] Abe, J. M., S., Akama and K., Nakamatsu. *Introduction to Annotated Logics - Foundations for Paracomplete and Paraconsistent Reasoning, Series Title Intelligent Systems Reference Library,* 2015. p. 190. Vols. 88, Publisher Springer International Publishing, DOI 10.1007/978-3-319-17912-4, Hardcover ISBN 978-3-319-17911-7, Series ISSN 1868-4394, Edition Number 1, 190 pages, 2015

[9] Abe, Jair Minoro. Paraconsistent logics and applications. In: 4th International Workshop on Soft Computing Applications. S .l. : IEEE, 2010. pp. 11–18.

[10] ISO/IEC20926: Software Engineering - IFPUG functional size measurement method, International Organization for Standardization, ISO. Genève, 2009: s.n., 2009.

[11] Abe, J. M. *Paraconsistent Intelligent Based-Systems: New Trends in the Applications of Paraconsistency, editor, Book Series: “Intelligent Systems Reference Library,” Springer-Verlag.* Germany, 2015: s.n., 2015. Vols. 94, ISBN:978-3-319-19721-0, 306 pages.

[12] Akama, S. *Towards Paraconsistent Engineering, Intelligent Systems Reference Library,.* 2016. Vols. 110, ISBN: 978-3-319-40417-2 (Print) 978-3-319-40418-9 (Online), Series ISSN 1868-4394, Publisher Springer International Publish-ing, DOI: 10.1007/978-3-319-40418-9, 234 pages, 2016.

[13] Carvalho, F. R. Aplicação de Lógica Paraconsistente Anotada em Tomadas de Decisão na Engenharia de Produção. *Tese (Doutorado) – Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia de Produção.* 2006.

[14] Jones, Capers., 2010. Software Engineering Best Practices.: McGraw Hill

[15] Costa, N. C. A.,Abe, J.M., Silva Filho, João Inácio da, Leite, C. F. S. and Murolo, A. 1999. LógicaParaconsistenteAplicada.: 1.ed. São Paulo: Atlas, v. 1. p.214.