40 Years of Function Points: Past, Present, Future

SEPTEMBER 18, 2019

By Luigi Buglione

Just 40 years ago in October 1979, Dr. Allan Albrecht proposed for the first time a technique for sizing the functionality of a software system. His technique was adopted, became an international standard, and inspired several other techniques and more. This paper shows the past, present and (possible) future contributions of Function Point Analysis.

The PAST: Origins, Motivation and Rationale

In the 1970s, Lines of Code (LOCs) were improperly used for sizing software systems (and still are today in some functional domains such as the military and real-time and automotive systems, just to name a few) and it led to what Capers Jones labelled the "productivity paradox" [1]: writing more LOCs does not mean to necessarily be more productive…the programming style, the expressivity of a certain programming language, the rule for counting LOCs (physical or logical, with/without commented lines…) created a huge variability in project (yes, project!) estimation. Because at that time, we only had mainframes, not PCs or mini-computers. Thus, the (software) product effort for deploying was most of the overall project effort and software functionalities were intended as the main deliverable of a software project. As a consequence, for many years (and still in many contracts), Function Points (FPs) were (and are) wrongly intended as the "project size" while they only express the product functional size. Anyway, Albrecht's goal was to overcome the "productivity paradox" and find a way to normalize productivity calculations from a business perspective [2]. The great idea was to base his technique on something technology-independent: processes and data. Thus, an FP calculation (and related product functional size) derived from the analysis of a set of

Functional User Requirements (FURs) is the same despite the technology and organizational style adopted, while effort, duration and cost/price are, of course, variable according to such elements. The typical example proposed over the years was to look at FP as square meters for measuring the "size" of a flat: the number of square meters can be the same for two flats in two different places, but the time for building them can differ (e.g. a flat can be built with bricks or be prefabricated), as well as their production costs and commercial value (a flat in Manhattan, New York, shall cost more per square meter than another one in another place); cost is not value.

The first paper, dated 1979, posed foundations for such a goal, as stated in its title ("Measuring application development productivity"). It represented a big-bang for estimations, trying to excerpt inputs/outputs/queries and data from a functional analysis in particular because such a new number could have been derived before programming and not later, as did with a LOC count. Since you cannot foresee with a certain level of precision the number of LOCs your team will produce tomorrow…too much variability!

There are many advantages but also some open issues: the correlation between the project effort (man/days) and the product functional size (in FP) was not so high, and a Value Adjustment Factor (VAF) was introduced in order to improve such a relationship and the R2 value in a linear regression analysis. VAF was initially calculated on 10 General System Characteristics (GSCs) with a variability of ±25% in the first 1979 paper, enlarged to 14 GSCs in the second and final paper, dated 1983 [3], with a ±35% variability on the initial "unadjusted" FP value. VAF was, therefore, the first way to indirectly "size" the contribution of Non-Functional Requirements (NFRs), both at the product as well as the project level. This second and final paper from Allan Albrecht stated the final FPA structure, splitting the initial MASTER DATA function type into ILF and EIF, and stated the final weighting system (as still valid in the current IFPUG CPM v4.3.1 version, dated 2010 [4]) from that time. Thus, it's possible to compare -from a functional sizing viewpoint- a software product realized today with another one released years ago for benchmarking purposes.

In 1987, IFPUG was born and held in its hands the management and evolution of Albrecht's technique. In the following years, several techniques moved away from Albrecht's ideas, and just a few became ISO standards, as shown in Fig. 1:

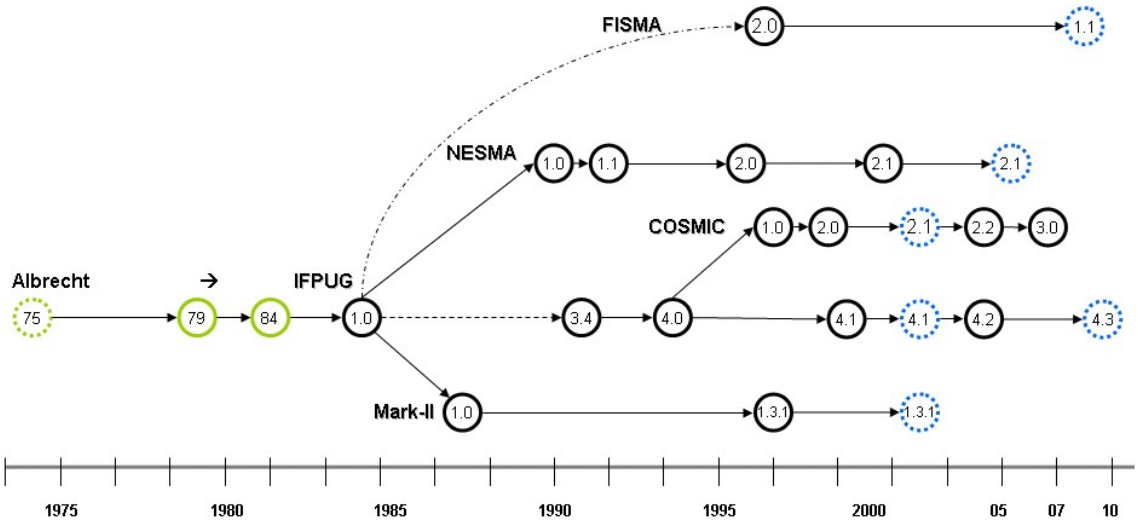

Fig. 1: ISO FSM Standards (with dotted blue lines)

They are (in order of appearance): Mark-II (1988), NESMA FPA (1990), FISMA (199x) and COSMIC (1998), with a lot of minor variants (here [5] a list of some of them).

First usages of FPA were to determine the functional size of software products for estimation and benchmarking purposes and -to simplify- as a contractual unit for payment in an ICT contract.

In 1998, ISO started the creation of a family of standards under the "14143" label with common principles for the so-called Functional Sizing Measurement (FSM) methods, stating in a clear way that such methods sized a product (not a project) and only when moving from product FURs, which excluded related ISO versions that initially included adjustment factors such as VAF [6]. The rationale? Such "adjustment/calibration" factors came from NFRs, thus, are out of scope for an FSM method. As evidence: ISO 20926:2003 was the ISO standard for IFPUG CPM v4.1 "unadjusted." Versions 4.x -from 1999 on- refined version definitions and basic concepts, in particular "user" and "boundary." Version v4.3 (2010) definitively dropped VAF from the normative text (also in ISO/IEC 20926:2009) and maintained it for historical purposes in Appendix C.

In the meantime, since FURs and NFRs need to be treated in a parallel way, the Software Non-Functional Assessment Process (SNAP) project started in 2007, releasing v1.0 in 2011 [7], for a new, first Non-Functional Sizing Measurement (NFSM) method, which moved from the ISO standard on product software quality (9126 before [8], and evolved into the current 25010:2011 standard [9]) while trying to complement the functional sizing as much as possible. Fig. 2 helps to determine the right project scope, with three kinds of requirements:

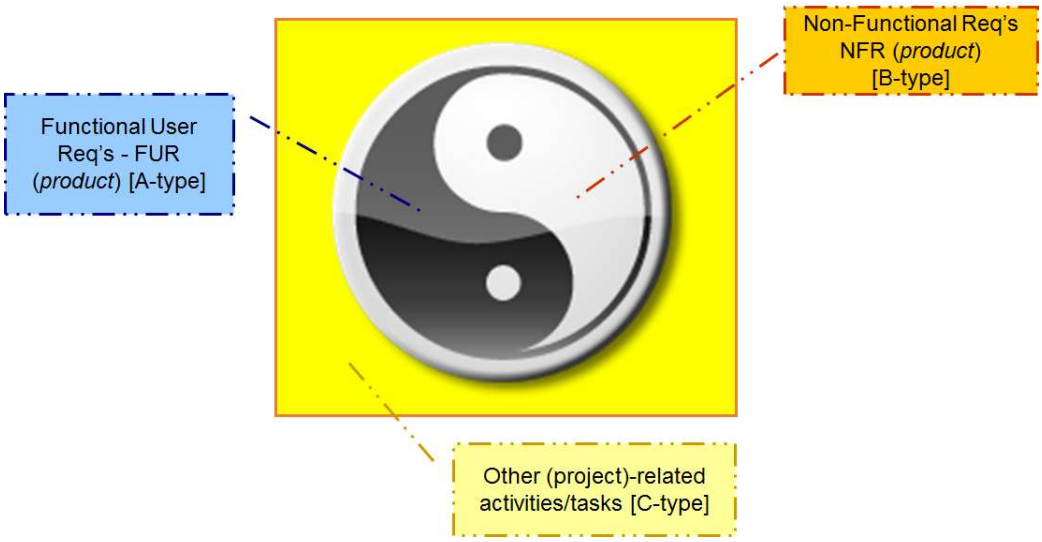

Fig. 2: Three kinds of requirements (from the 'ABC' schema)

The information was written in different manner than CPM v4.1. I clearly stated in a 2012 paper I wrote for *MetricViews* [10], *The 'ABC' Schema*; such taxonomy was also used in a 2015 paper co-written by IFPUG/COSMIC for better expressing a taxonomy for NFRs [11]. This classification is critical from the beginning of any project for properly analysing and comparing them with a good benchmarking analysis. Fig. 3 summarizes how a user requirement could be deployed and split, in the wider case, into three chunks: product FURs (A), product NFRs (B) and project constraints/requirements (C).

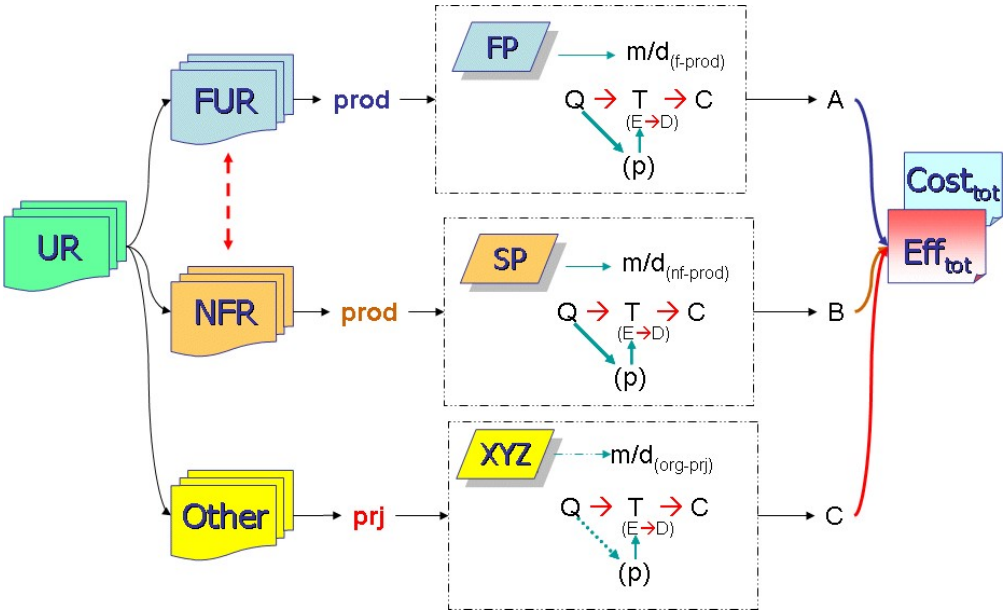

Fig. 3: The ABC Schema [10]

From user requirements to the final overall project effort and costs – The "ABC" schema [10] in 1997 ISBSG (isbsg.org) was born and all of the most active Software Measurement Associations (SMAs) adhered to this benchmarking initiative. The 2019 release [12] includes more than 9,000 projects, mostly sized using the IFPUG and COSMIC FPA methods. Another complementary norm was the ISO/IEC 14143-5:2004 [13], which proposes criteria for defining "functional domains" and allows a reasonable comparison among software systems with similar characteristics and effort distribution by requirement types (ABC). It does not make sense to compare apples with oranges…

The PRESENT: What's going on?

FSM methods are diffusely used in Information & Communication Technology (ICT) contracts, with a higher concentration in some countries (e.g. Italy, Brazil, Poland, India), and represent a quantitative basis for sizing the product functional size and can help in benchmarking analysis to determine which could be (approximately) the non-functional effort in a project, as shown in Fig.4:

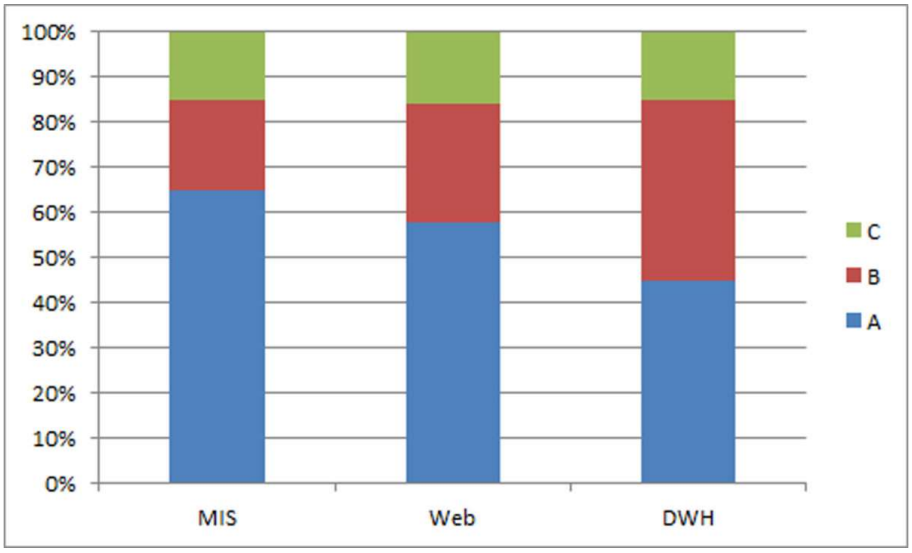

Fig. 4: Effort distribution by requirement type (ABC) per functional domain: an example

An example of effort distribution by requirement type (ABC) per functional domain is splitting what is non-functional according to the ABC schema. A B-type requirement can be realized and deployed by an IT specialist (e.g. a database administrator, usability expert…) that typically costs less than another professional (e.g. a project/service manager, a measurement specialist, a quality assurance person…) running a C-type requirement but more than a professional (e.g. an analyst/programmer) running an A-type requirement. Fig. 5 shows the two opposite pyramids for a typical distribution of project effort by requirement type and cost per man/day for each kind of professional [14].

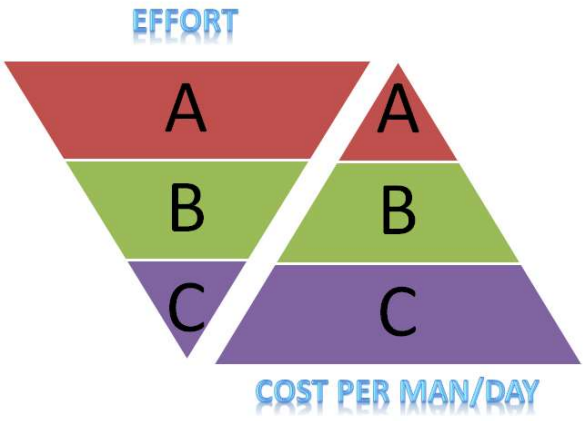

Fig. 5: Effort distribution by requirement type and cost/man-day (according to the ABC schema)

Effort distribution by requirement type and cost/man-day (according to the ABC schema). Again, lessons learned from 40 years of experience helped to better define (and refine) principles and rules about FPA's scope of application. The '123' schema is another classification [15] for stating which kind of requirements can be present in a certain phase of a project (1: Dev, 2: Ops, 3: Svc, Maintenance).

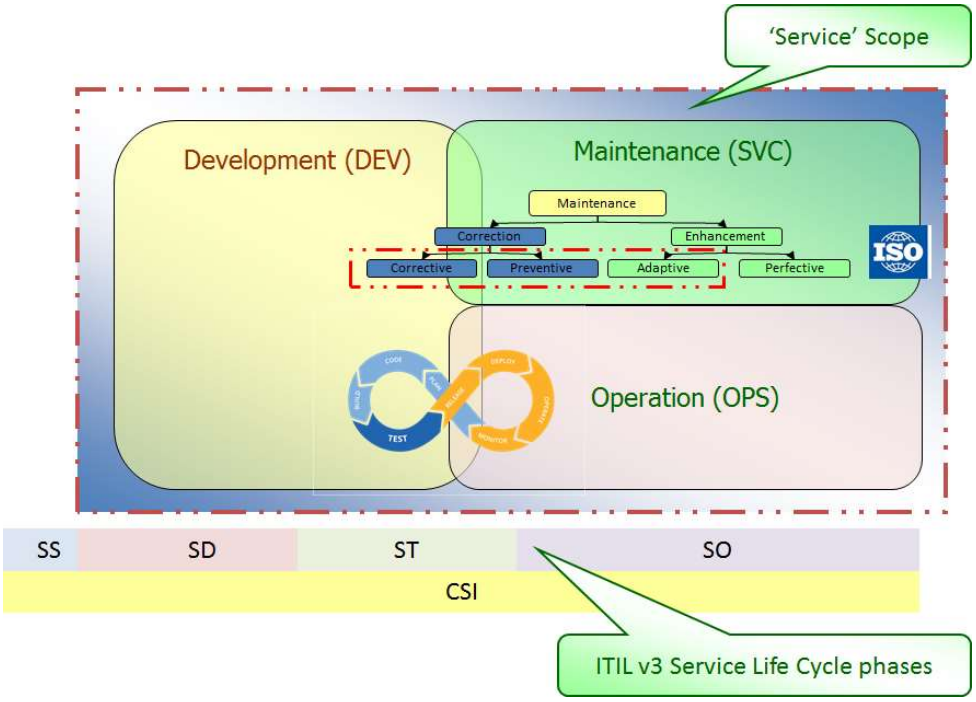

Fig. 6: The '123' schema jointly with the 'ABC' schema

Thus, in the OPS phase a software is used, not produced/changed, and generates a "zero FP" count, as well as when a change request will include only B-type requirements (e.g. for a corrective/perfective maintenance, as stated in ISO/IEC 14764:2006 standard [16], also cited in CPM v4.3.1 – part 3, chapter 4, pages 20-21). Even if definitions and criteria about what FUR or NFR has been created, explained and diffused over time, there is still a cultural debt in contractual practices and business to use at least one Unit of Measure (UoM) for sizing A-type requirements (FPs, whatever kind) jointly with the UoM for sizing B-type requirements (e.g. with IFPUG SNAP points or measures from ISO/IEC 25023 [17], as well as the C-type activities that complete the scope for estimating the whole effort needed for scoping a project. Only when sizing all of the requirements for the three (ABC) types, it's possible to reduce the "scope creep," while there is still an historical misconception about what an FP sizes and what it does not. But it would be sufficient to know in an FSM method, which are its Base Functional Components (BFCs) for including (or not) such an activity or activities.

Last but not least, FP and automation. Dr. Albrecht created a "design measure" for allowing an estimate in the early phases of a lifecycle. Some tools today, after 40 years, would derive FPs (whatever kind) parsing software code or working on some forms of FUR notations. Some (common-sense) observations and thoughts: automation is useful if it's respectful of four criteria: faster, more accurate, more timely and lower in cost. If we need to size a new FUR, a tool parsing code (as stated in the new ISO 19515 standard on Automated FP [18]) would be useless and costly. Or, using a tool assuming some UML notations as inputs would imply more man-hours (and related costs) for translating a human-based written requirement into a meta-language format. Also, an organization needs to carefully analyse the return on investment for such choice(s) according to the four above mentioned criteria. Quickly creating a draft baseline in which time and effort are critical assets could be ok under the preconditions that it is verified by a human CFPS and that the UoM under scope are the same. Otherwise, automation could become risky or difficult to manage.

The FUTURE: What can we expect?

As many people would say, the future is now…but what can we expect from FPA in the near future? FPA has strong foundations and it is technology independent; what we have learned from the past is that the next steps should be:

- A better and more affordable User Requirements (UR) management, scoping and measurement during the early phases of a project: this is the main and primary goal that should be achieved.

- Adoption of FPA to new technologies, through the proper interpretation of FPA basic rules from 1979/84. We cannot yet measure and size through FPA new technologies such as cloud computing [19], Internet of Things (IoT) [20], artificial intelligence and any new tech the upcoming years will bring us.

Our best bet is not to invent something new, but to deeply analyse our current processes and data to determine new and different ways to engineer a software system and still size a FUR with FPA!

"*We intend to continue using and improving the function **value** measurement*." (Allan Albrecht, October 1979)

References

[1] Jones, C., What Are Function Points? SPR website, URL: http://tiny.cc/tgur7y

[2] Albrecht A. J., "Measuring application development productivity" in Proc. Joint SHARE, GUIDE, and IBM Application Development , 1979, pp. 83-92. http://tiny.cc/2ywacz

[3] Albrecht A. J. & Gaffney J. E., "Software function, source lines of code, and development effort prediction: A software science validation," IEEE Trans. Software Eng., vol. 9, no. 6, November 1983, pp. 639-647. http://tiny.cc/1zwacz

[4] IFPUG, *Function Point* Counting *Practice Manual (CPM)*, release 4.3.1, January 2010, URL: ifpug.org

[5] Lother M., Dumke R., Points-Metrics: Comparisons and Analysis", in: Current Trends in Software Measurement, Shaker Publishing, 2001, pp.228-267

[6] ISO/IEC, *International Standard 14143-1 – Information Technology – Software Measurement – Functional Size Measurement – Part 1: definition of concepts*, February 2007

[7] IFPUG, *Software Non-functional Assessment Process (SNAP) Assessment Practice Manual (APM)*, version 1.0, September 2011, URL: ifpug.org

[8] ISO/IEC, IS 9126-1:2001 – Software Engineering – Product Quality – Part 1: Quality Model, International Organization for Standardization, 2001

[9] ISO/IEC, IS 25010:2011 -Systems and software engineering-Systems and software Quality Requirements and Evaluation (SQuaRE)-System and software quality models, International Organization for Standardization, March 2011

[10] Buglione L., The Next Frontier: Measuring and Evaluating the NonFunctional Productivity, MetricViews, August 2012, URL:https://www.ifpug.org/Metric%20Views/MVBuglione.pdf

[11] COSMIC/IFPUG, Glossary of terms for Non-Functional Requirements and Project Requirements used in software project performance measurement, benchmarking and estimating, v1.0, September 2015

[12] ISBSG, D&E (Development & Enhancement) repository, R2019, URL:isbsg.org

[13] ISO/IEC, *Technical Report 14143-5 – Information Technology – Software Measurement – Functional Size Measurement – Part 5: determination of functional domains for use with functional size measurement*, 2004 (R2019)

[14] Buglione L., Come gestire progetti uniformi e misurabili: focus su tipologie e requisiti, ZeroUnoWeb, May 3 2019, URL: http://tiny.cc/y1tr7y

[15] Buglione L., Interpretare DevOps per misurare bene (e meglio) i progetti, PMExpo2017, Presentation, October 2017, URL:https://www.pmexpo.it/2017/programma/009tk

[16] ISO/IEC, *International Standard 14764:2006 – Software Engineering – Software Life Cycle Processes – Maintenance*, 2006

[17] ISO/IEC, *International Standard 25023:2016 – Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality, June 2016*

[18] ISO/IEC, *International Standard 19515:2019 – Information technology – Object Management Group Automated Function Points (AFP), 1.0, May 2019*

[19] Woodward S., Function Point Analysis Clarifies in a Cloudy world, Metricas 2012, Sao Paulo (Brazil), Nov 28-29 2012, URL:http://www.bfpug.com.br/metricas2012/woodward.pdf

[20] Cagley T., Function Points and IOT, or How My Kitchen Is Spying On Me!, IFPUG ISMA17, Bangalore (India), March 8, 2019

About the Author

Luigi Buglione is the IFPUG Director for Conference/Education and President of the Gruppo Utenti Function Point Italia – Italian Software Metrics Association (GUFPI-ISMA) (www.gufpi-isma.org). He works as a Measurement and Process Improvement Specialist at Engineering Ing. Inf. SpA in Rome, Italy and Associate Professor at the École de Technologie Supérieure (ETS) – Université du Québec, Canada. He achieved several certifications, including IFPUG CFPS, CSP, CSMS, and COSMIC CCFL about Software Measurement. He is a regular speaker at international conferences on Sw/Service Measurement, Process Improvement and Quality and is actively part of International and National Technical Associations on such issues. He received a Ph.D. in MIS and a degree cum laude in Economics. Luigi can be reached at luigi.buglione@eng.it.